Retelistica in Java

Java Sockets, TCP, UDP, etc.

Dinu Florin - Silviu
Tindeche Alexandru
Tudose Stefan

- grupa 231 -

Outline

Implementarea time, daytime si a unui server DNS in Java folosind sockets de UDP sau TCP in interiorul unei infrastructuri Docker

1. Descrierea temei de laborator

Descrierea temei de laborator

- Se cere implementarea in Java a unui server si a unui client de **time** prin UDP si de **daytime** prin TCP
- Am adaugat si un server de **DNS**, evident ca prin UDP, care sa ajute la adresarea serverelor din retea
- Implementarea clientilor si serverelor a fost facuta cu **Java 17** si **Maven**, iar compilarea are loc in containere Docker bazate pe maven:3-eclipse-temurin-17
- Implementarea retelei a fost facuta cu **Docker**
- Compilarea si impachetarea intr-un JAR a programelor de Java s-a facut in interiorul **containerelor** de Docker

2. Concepte teoretice

Unde vrem sa ajungem?

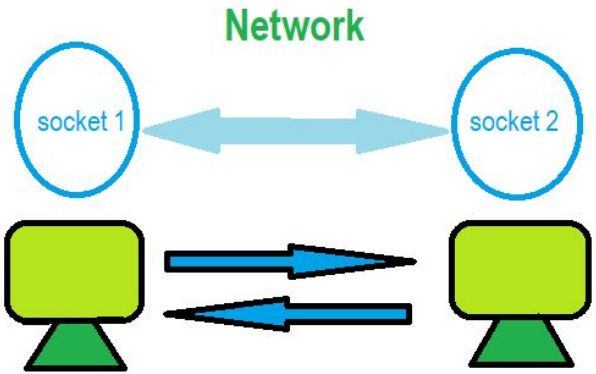
- Dorim comunicarea intre 2 procese care ruleaza pe doua calculatoare diferite

Cum reusim sa facem asta?

- Ne folosim de socket-uri: deschidem cate un socket in fiecare dintre programe, fiecarui socket fiindu-i atribuit un port

Ce este un socket?

- Un socket este un mod de comunicare 2-way intre doua porturi diferite (pot fi intre doua calculatoare diferite sau pe acelasi device)
- Faciliteaza comunicarea interproces
- Ca sa putem trimite date intre porturi, fiecare trebuie sa aiba deschis cate un astfel de socket



Structura server-client

- De obicei, in comunicarea dintre socket-uri, unul dintre ele “asculta” iar celalt creeaza o conexiune si transmite date
- Cel care “asculta” se numeste server, iar celalt client

TCP (Transmission Control Protocol)

- Protocol de comunicare care asigura transmiterea in intregime a mesajelor intre doua porturi
- TCP inainte de a transmite datele creeaza o conexiune intre cele 2 porturi prin 3-way handshake

UDP (User Datagram Protocol)

- Acest protocol nu asigura o conexiune directa intre cele 2 procese inainte de a transmite date (without prior arrangement) si fara a se asigura ca celalt proces este gata sa primeasca date
- Util pentru ca este mai simplu si mai rapid, nu avem atat de mult “overhead”

Socket-uri in C

1. Creearea socket-ului:

```
int sock = socket(domain, type, protocol);
```

2. Setarea unor parametrii specifici (optional)

```
int sockport (socket_descriptor, level, option_name,
*option_value, option_lenght);
```

3. Bind

- la creare nu avem o adresa alocata socketului,
iar functia bind fix asta face
- asociaza un socket descriptor cu o adresa de endpoint (sockaddr_inpt TCP/IP, contine adresa IP si port)

```
int bind(socket_descriptor, struct sockaddr *address,
address_lenght);
```

Socket-uri C - server (TCP)

4. Listen

   - La crearea unui socket, acesta nu corespunde niciunui tip, adica nu este nici activ (transmisie de date), nici activ (“asculta”)
   - Serverele TCP apeleaza listen pentru a face un socket pasiv
   - In general, noi avem un loop infinit in care tratam requesturile, iar daca in timp ce se proceseaza un request vine altul, acesta din urma este pus intr-o coada, denumita backlog, a carui dimensiune o setam in functie

```
int listen(socket_descriptor, back_log_size);
```

5. Accept

   - Dupa ce am setat socket-ul pentru a “asculta”, trebuie sa il facem sa si accepte requesturi
   - Creeaza un socket temporar pentru fiecare cerere

```
int accept(socket_descriptor,
struct sockaddr *restrict address,
address_length);
```

Serverul C - server

- Pentru a trimite un raspuns catre client, folosim functiile `read()` si `send()`

```
read(socket, buffer, 65536); // citire din buffer

send(socket, string, strlen(string), 0); // trimiterea
pachetului
```

Socket-uri C - client (TCP)

4. Connect

   - Trimite un request catre socket-ul care asculta si se conecteaza

```
int connect(socket_descriptor, struct sockaddr *address,
address_length);
```

Socket-uri in Java (UDP) - server

1. Crearea socket-ului

```
socket = new DatagramSocket(int port);
```

2. Crearea packet-ului pentru primirea datelor de la client

```
packet = new DatagramPacket(byte[ ] buffer, int buffer_length);
```

3. Primirea pachetului

```
socket.receive(packet);
```

4. Trimiterea unui raspuns

- Trebuie sa facem rost de portul si adresa clientului

```
address = packet.getAddress();
port = packet.getPort();
```

- Creem un packet de trimitere si il trimitem folosind `send()`

```
packet = new DatagramPacket(byte[ ] buffer,
    int buffer_length, InetAdress address, int port);
socket.send(packet);
```

Socket-uri in Java (TCP) - server

1. Crearea socket-ului

```
socket = new ServerSocket(int port);
```

2. Crearea socket-ului pentru primirea datelor de la client

```
clientSocket = socket.accept()
```

3. Creeare de stream-uri de input si output

```
PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
BufferedReader in = new BufferedReader(new

InputStreamReader(clientSocket.getInputStream()));
```

Nota: stream-urile folosesc aceleasi metode ca cele uzuale de java (ex. println)

Socket-uri in Java (UDP) - client

1. Crearea socket-ului

```
socket = new DatagramSocket();
```

2. Crearea packet-ului pentru trimiterea datelor la server

```
packet = new DatagramPacket(byte[ ] buffer, int buffer_length,
 InetAddress serverAdress, int serverPort);  (unde buffer e mesajul trimis)
```

3. Trimiterea datelor la server

```
socket.send(packet);
```

4. Primirea unui raspuns

- Trebuie sa cream un nou packet udp pentru primire

```
packet = new DatagramPacket(buf, buf.length);
```

(in buf va fi stocat raspunsul)

- Primim pachetul de la server

```
socket.receive(packet);
```

Socket-uri in Java (TCP) - client

1. Crearea socket-ului ce va reprezenta conexiunea cu serverul

```
socket = new Socket(String hostName, int port);
```

2. Creeare de stream-uri de input si output

```
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
BufferedReader in = new BufferedReader(new

                    InputStreamReader(socket.getInputStream()));
```

Severe multiprotocol in Java

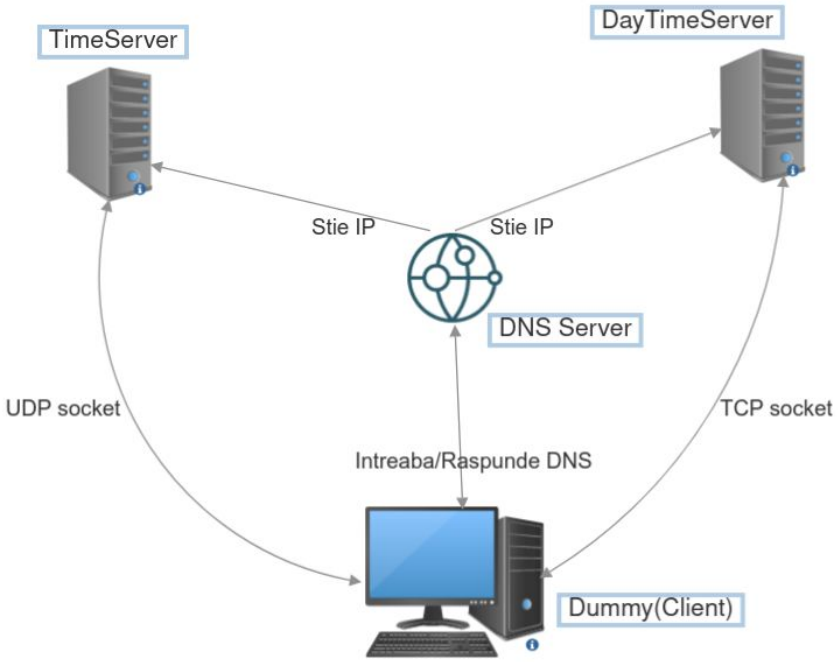
- Un tip de server care poate primi mai multe tipuri de request-uri (HTTP, FTP, SHH, DNS) si da un raspuns corespunzator in functie de tipul primit
- In Java lucreaza cu thread-uri (de exemplu cate unul pentru fiecare protocol, respectiv cate o clasa pentru fiecare protocol) si ofera un raspuns corespunzator, folosind implementarea specifica protocolului respectiv

Daemonizarea in Java

- Putem daemoniza un proces folosing clasa `ProcessBuilder()`

3. Structura Docker

Reteaua finala Docker



Containere Docker (1)

```yaml
dummy:
  container_name: dummy
  depends_on:
    dns-server:
      condition: service_started
    time-server:
      condition: service_started
    time-client:
      condition: service_completed_successfully
    daytime-server:
      condition: service_started
    daytime-client:
      condition: service_completed_successfully
  build:
    context: .
    dockerfile: ./dummy/Dockerfile
  networks:
    net1:
      ipv4_address: 172.16.0.2
  command: sh -c "echo 'nameserver 172.16.0.1' > /etc/resolv.conf && tail -f /dev/null"
  volumes:
    - ./compiled_jars:/app
```

1. **dummy** (client)
   a. Este bazat pe *ubuntu:22.04*
   b. Contine utilitarele de retea si Oracle Java JDK 17
   c. Are *nameserver 172.16.0.1* adaugat in */etc/resolv.conf*
   d. Are IP-ul: *172.16.0.2*
   e. Depinde de pornirea dns-server, time-server si daytime-server si de completarea cu succes a time-client si daytime-client
2. **dns-server**
   a. Este bazat pe *maven:3-eclipse-temurin-17*
   b. Compileaza, impacheteaza si ruleaza aplicatia Java de DNS
   c. Are IP-ul: *172.16.0.1*
   d. Mapeaza portul *53* catre host pe *5300*

```yaml
dns-server:
  container_name: dns-server
  build:
    context: .
    dockerfile: ./dns/Dockerfile
  ports:
    - 5300:53/udp
  networks:
    net1:
      ipv4_address: 172.16.0.1
```

Containere Docker (2)

3. **time-server**
    a. Este bazat pe *maven:3-eclipse-temurin-17*
    b. Compileaza, impacheteaza si ruleaza serverul de time pe UDP
    c. Are IP-ul: *172.16.0.3*
    d. Mapeaza portul local *7070* pe host *7070*
4. **time-client**
    a. Este bazat pe *maven:3-eclipse-temurin-17*
    b. Compileaza si impacheteaza clientul de time
    c. Copiaza pachetul JAR in volumul /app/compiled_jars mapat pe host in ./compiled_jars (relativ fata de docker-compose.yml)
    d. Nu are entrypoint, pur si simplu compileaza si atat

```yaml
time-server:
  container_name: time-server
  build:
    context: .
    dockerfile: ./TimeServer/Dockerfile
  ports:
    - 7070:7070
  networks:
    net1:
      ipv4_address: 172.16.0.3
time-client:
  container_name: time-client
  build:
    context: .
    dockerfile: ./TimeClient/Dockerfile
  volumes:
    - type: bind
      source: ./compiled_jars
      target: /app/compiled_jars
```

Containere Docker (3)

5. **daytime-server**
    a. Este bazat pe *maven:3-eclipse-temurin-17*
    b. Compileaza, impacheteaza si ruleaza serverul de daytime pe TCP
    c. Are IP-ul: *172.16.0.4*
    d. Mapeaza portul local *4040* pe host *4040*
6. **daytime-client**
    a. Este bazat pe *maven:3-eclipse-temurin-17*
    b. Compileaza si impacheteaza clientul de time
    c. Copiaza pachetul JAR in volumul /app/compiled_jars mapat pe host in ./compiled_jars (relativ fata de docker-compose.yml)
    d. Nu are entrypoint, pur si simplu compileaza si atat

```yaml
daytime-server:
  container_name: daytime-server
  build:
    context: .
    dockerfile: ./DayTimeServer/Dockerfile
  ports:
    - 4040:4040
  networks:
    net1:
      ipv4_address: 172.16.0.4
daytime-client:
  container_name: daytime-client
  build:
    context: .
    dockerfile: ./DayTimeClient/Dockerfile
  volumes:
    - type: bind
      source: ./compiled_jars
      target: /app/compiled_jars
```

4. UDP - time

5. TCP - daytime

6. Server DNS

Server DNS - configurare

- Citeste zone fileul si il mapeaza pe un obiect de tip ArrayList<Object> ale carui elemente pot fi mai departe downcastate ca obiecte de tip Zone
- Deschide un socket de UDP pe *0.0.0.0:53* cu un buffer de *65536 bytes*
- Are un loop infinit in care primeste un request, apoi creeaza un thread nou in care il proceseaza

```java
public DNSServer(){
    // Initialize variables
    this.mapper = new ObjectMapper();
    this.typeFactory = mapper.getTypeFactory();
    this.zones = new ArrayList<>();

    // Read zones from resource zones.json
    try {
        // Zone file source of inspiration: https://github.com/hpwCodeORG/hpwDNS
        this.zones = this.mapper.readValue(getClass().getResourceAsStream( name: "/zones.json"), typeFactory.constructCollectionType(ArrayList.class, Object.class));
    } catch(Exception e){
        System.out.println("Can't read zones: " + e);
        System.exit( status: 1);
    }

    // DNS Server
    try {
        // Start socket on port 53 accept 10 connections from 0.0.0.0
        this.dnsSocket = new DatagramSocket( port: 53, InetAddress.getByName( host: "0.0.0.0"));
        byte[] buffer = new byte[65536];
        DatagramPacket request = new DatagramPacket(buffer, buffer.length);
        System.out.println("DNS Server started on port 53");

        while(true){
            // Receive DNS requests
            this.dnsSocket.receive(request);

            // Start new thread to process request
            new Thread(() -> {
                this.processRequest(request);
            }).start();
        }
    } catch(Exception e){
        System.out.println("Can't open server socket: " + e);
        System.exit( status: 1);
    }
}
```

Server DNS - Zone

- Contine informatii despre origine, ttl, start of authority, nameserver si A record domain
- SOA are ca parametri principali *mname* care este serverul DNS primar, timpul de refresh, expirare si ttl
- Nameserver are o lista de hosturi
- A record are numele (@ reprezinta domeniul root), ttl si value cu adresa efectiva de IP
- Contine 3 domenii hardcodate:
  - example.org - folosit ca exemplu
  - timeserver.org - folosit pentru time server
  - daytimeserver.org - folosit pentru daytime server

```
public class Zone {
    1 usage
    public String $origin;
    no usages
    public int $ttl;
    no usages
    public SOA soa;
    no usages
    public List<NS> ns;
    2 usages
    public List<A> a;

    1 usage   fredtux
    public static class SOA {
        no usages
        public String mname;
        no usages
        public String rname;
        no usages
        public String serial;
        no usages
        public int refresh;
        no usages
        public int retry;
        no usages
        public int expire;
        no usages
        public int minimum;
    }

    1 usage   fredtux
    public static class NS {
        no usages
        public String host;
    }

    1 usage   fredtux
    public static class A {
        public String name;
        no usages
        public int ttl;
        public String value;
    }
}
```

Server DNS - procesarea requestului

- Creaza un obiect de tip DNSRequest
- Daca nu e marcat ca fiind negasit, atunci il cauta in zone si returneaza un pachet bazat pe zona respectiva
- Daca e marcat ca fiind negasit, atunci face un pachet cu zona nula
- Crearea pachetului se bazeaza pe popularea campurilor necesare din zona si convertirea lor intr-un *byte[]*
- *DNSRequest* doar gaseste numele domeniului din pachetul de request pentru a-l trimite ca sa fie folosit mai tarziu

```java
private void processRequest(DatagramPacket request){
    // Handle DNS requests
    byte[] requestData = request.getData();
    int requestDataLength = request.getLength();

    // DNS Response
    try{
        byte[] responseData = this.handleDNSData(requestData, requestDataLength, request, unfound: false);
        if(responseData == null)
            responseData = this.handleDNSData(requestDataLength, request, unfound: true);

        DatagramPacket response = new DatagramPacket(responseData, responseData.length, request.getAddress(), request.getPort());

        // Send DNS response
        try{
            lock.lock();
            this.dnsSocket.send(response);
            lock.unlock();
        } catch(Exception e){
            System.out.println("Can't send response: " + e);
        }
    } catch(Exception e){
        System.out.println("Can't handle DNS request: " + e);
    } finally {
        System.out.println("Exiting child...");
    }
}
```

```java
private byte[] handleDNSData(byte[] requestData, int requestDataLength, DatagramPacket request, boolean unfound) throws Exception{
    DNSRequest dnsRequest = new DNSRequest(requestData, requestDataLength);
    String domain = dnsRequest.getDomain();
    if(unfound){
        return this.craftDNSPacket(requestData, requestDataLength, request, z: null, domain);
    }
    System.out.println("Searching for domain: " + domain);

    // Check if domain is in zones
    for(Object zone : this.zones){
        Zone z = this.mapper.convertValue(zone, Zone.class);
        if(domain.equals(z.$origin)){
            if(z.a != null){
                return this.craftDNSPacket(requestData, requestDataLength, request, z, domain);
            }
        }
    }

    return null;
}
```

```java
private byte[] craftDNSPacket(byte[] requestData, int requestDataLength, DatagramPacket request, Zone z, String domain) throws Exception{
    // Create a new DNS message
    Message dnsMessage = new Message();

    // Set the ID field of the DNS header
    Header header = dnsMessage.getHeader();

    // Create the question section
    String domainName = domain;
    int recordType = Type.A;
    int recordClass = DClass.IN;
    Record question = Record.newRecord(Name.fromString(domainName), recordType, recordClass);
    dnsMessage.addRecord(question, Section.QUESTION);

    // Create the answer section
    String ipAddress;
    if(z == null) {
        ipAddress = "0.0.0.0";
        header.setRcode(Rcode.NXDOMAIN);
    }
    else {
        ipAddress = z.a.get(0).value;
    }
    int ttl = 3600;
    InetAddress addr = InetAddress.getByName(ipAddress);
    Record answer = Record.newRecord(Name.fromString(domainName), Type.A, recordClass, ttl, addr.getAddress());
    dnsMessage.addRecord(answer, Section.ANSWER);

    // Make dnsMessage a response
    header.setFlag(Flags.QR);

    // Add the request id to dnsMessage
    header.setID(new Header(requestData).getID());

    // Convert the DNS message to a byte array
    byte[] dnsPacket = dnsMessage.toWire();

    return dnsPacket;
}
```

7. Exemple practice

Arhitectura (completari la slideurile anterioare)

- Pentru aplicatiile de java am folosit *maven:3-eclipse-temurin-17* bazat pe Ubuntu 20.04 cu OpenJDK [3]
- Pentru dummy (client) am folosit *ubuntu:22.04*
- Am expus toate porturile necesare ca sa poata fi accesate si de pe host
- Am creat o retea de tip *bridge* pe *172.16.0.0/16* si gateway *172.16.1.1*
- A fost testat pe Ubuntu 22.04 ca host

Dig - host

```
┌──(tux@tuxmachine)-[~/…/Facultate/ReteleDeCalcul/Lab/Proiect]
└─$ docker compose up
[+] Building 0.0s (0/0)
[+] Running 6/0
 ✔ Container time-client     Created                                    0.0s
 ✔ Container dns-server      Created                                    0.0s
 ✔ Container daytime-client  Created                                    0.0s
 ✔ Container time-server     Created                                    0.0s
 ✔ Container daytime-server  Created                                    0.0s
 ✔ Container dummy           Created                                    0.0s
Attaching to daytime-client, daytime-server, dns-server, dummy, time-client, time-server
daytime-client exited with code 0
time-client exited with code 0
dns-server      | DNS Server started on port 53
dns-server      | Searching for domain: example.org.
dns-server      | Exiting child...
```

```
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26860
;; flags: qr; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.org.                   IN      A

;; ANSWER SECTION:
example.org.            3600    IN      A       192.255.255.255

;; Query time: 128 msec
;; SERVER: 172.16.1.1#5300(172.16.1.1) (UDP)
;; WHEN: Wed Jun 28 14:47:01 EEST 2023
;; MSG SIZE  rcvd: 45
```

```
┌──(tux@tuxmachine)-[~]
└─$
0  0 zsh *  1 zsh -                    2023-06-28 14:47  tuxmachine
```

Dig - dummy

```
┌──(tux@tuxmachine)-[~/…/Facultate/ReteleDeCalcul/Lab/Proiect]
└─$ docker compose up
[+] Building 0.0s (0/0)
[+] Running 6/0
 ✔ Container time-client     Created                                    0.0s
 ✔ Container dns-server      Created                                    0.0s
 ✔ Container daytime-client  Created                                    0.0s
 ✔ Container time-server     Created                                    0.0s
 ✔ Container daytime-server  Created                                    0.0s
 ✔ Container dummy           Created                                    0.0s
Attaching to daytime-client, daytime-server, dns-server, dummy, time-client, time-server
daytime-client exited with code 0
time-client exited with code 0
dns-server      | DNS Server started on port 53
dns-server      | Searching for domain: example.org.
dns-server      | Exiting child...
dns-server      | Searching for domain: example.org.
dns-server      | Exiting child...
```

```
; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> example.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61867
;; flags: qr; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.org.                   IN      A

;; ANSWER SECTION:
example.org.            3600    IN      A       192.255.255.255

;; Query time: 4 msec
;; SERVER: 172.16.0.1#53(172.16.0.1) (UDP)
;; WHEN: Wed Jun 28 11:47:56 UTC 2023
;; MSG SIZE  rcvd: 45

root@e75734172671:/app#
0  0  docker *  1  zsh -                    2023-06-28  14:48  tuxmachine
```

Daytime

```
└─$ docker compose up
[+] Building 0.0s (0/0)
[+] Running 6/0
 ✔ Container time-client     Created                                                    0.0s
 ✔ Container dns-server      Created                                                    0.0s
 ✔ Container daytime-client  Created                                                    0.0s
 ✔ Container time-server     Created                                                    0.0s
 ✔ Container daytime-server  Created                                                    0.0s
 ✔ Container dummy           Created                                                    0.0s
Attaching to daytime-client, daytime-server, dns-server, dummy, time-client, time-server
daytime-client exited with code 0
time-client exited with code 0
dns-server      | DNS Server started on port 53
dns-server      | Searching for domain: example.org.
dns-server      | Exiting child...
dns-server      | Searching for domain: example.org.
dns-server      | Exiting child...
dns-server      | Searching for domain: daytimeserver.org.
dns-server      | Exiting child...
```

```
;; ANSWER SECTION:
example.org.            3600    IN      A       192.255.255.255

;; Query time: 4 msec
;; SERVER: 172.16.0.1#53(172.16.0.1) (UDP)
;; WHEN: Wed Jun 28 11:47:56 UTC 2023
;; MSG SIZE  rcvd: 45

root@e75734172671:/app# ll
total 20
drwxrwxr-x 2 1000 1000 4096 Jun 28 07:58 ./
drwxr-xr-x 1 root root 4096 Jun 28 11:45 ../
-rw-r--r-- 1 root root 2139 Jun 28 11:46 DayTimeClient.jar
-rw-rw-r-- 1 1000 1000   46 Jun 28 07:59 README.md
-rw-r--r-- 1 root root 1836 Jun 28 11:46 TimeClient.jar
root@e75734172671:/app# java -jar DayTimeClient.jar daytimeserver.org
Current day time: 2023/06/28 11:49:04
root@e75734172671:/app#
0  0 docker *  1 zsh -                                  2023-06-28  14:49  tuxmachine
```

Time

```
[+] Running 6/0
 ✔ Container time-client     Created                                    0.0s
 ✔ Container dns-server      Created                                    0.0s
 ✔ Container daytime-client  Created                                    0.0s
 ✔ Container time-server     Created                                    0.0s
 ✔ Container daytime-server  Created                                    0.0s
 ✔ Container dummy           Created                                    0.0s
Attaching to daytime-client, daytime-server, dns-server, dummy, time-client, time-server
daytime-client exited with code 0
time-client exited with code 0
dns-server      | DNS Server started on port 53
dns-server      | Searching for domain: example.org.
dns-server      | Exiting child...
dns-server      | Searching for domain: example.org.
dns-server      | Exiting child...
dns-server      | Searching for domain: daytimeserver.org.
dns-server      | Exiting child...
dns-server      | Searching for domain: timeserver.org.
dns-server      | Exiting child...

example.org.            3600    IN      A       192.255.255.255

;; Query time: 4 msec
;; SERVER: 172.16.0.1#53(172.16.0.1) (UDP)
;; WHEN: Wed Jun 28 11:47:56 UTC 2023
;; MSG SIZE  rcvd: 45

root@e75734172671:/app# ll
total 20
drwxrwxr-x 2 1000 1000 4096 Jun 28 07:58 ./
drwxr-xr-x 1 root root 4096 Jun 28 11:45 ../
-rw-r--r-- 1 root root 2139 Jun 28 11:46 DayTimeClient.jar
-rw-rw-r-- 1 1000 1000   46 Jun 28 07:59 README.md
-rw-r--r-- 1 root root 1836 Jun 28 11:46 TimeClient.jar
root@e75734172671:/app# java -jar DayTimeClient.jar daytimeserver.org
Current day time: 2023/06/28 11:49:04
root@e75734172671:/app# java -jar TimeClient.jar timeserver.org
Time right now: 11:50:06
root@e75734172671:/app#
0 > 0 > docker * > 1 > zsh -                    2023-06-28 < 14:50 < tuxmachine
```

8. Concluzii

Concluzii

- Java contine foarte multe **librarii** care usureaza destul de mult dezvoltarea de aplicatii
- Docker ajuta la **portabilitatea** proiectului indiferent de sistemul de operare de pe host sau orice alte programe instalate
- Java nu are suport nativ pentru fork() decat prin SNI (Simple Native Interface), dar poate lucra foarte usor cu **threaduri**

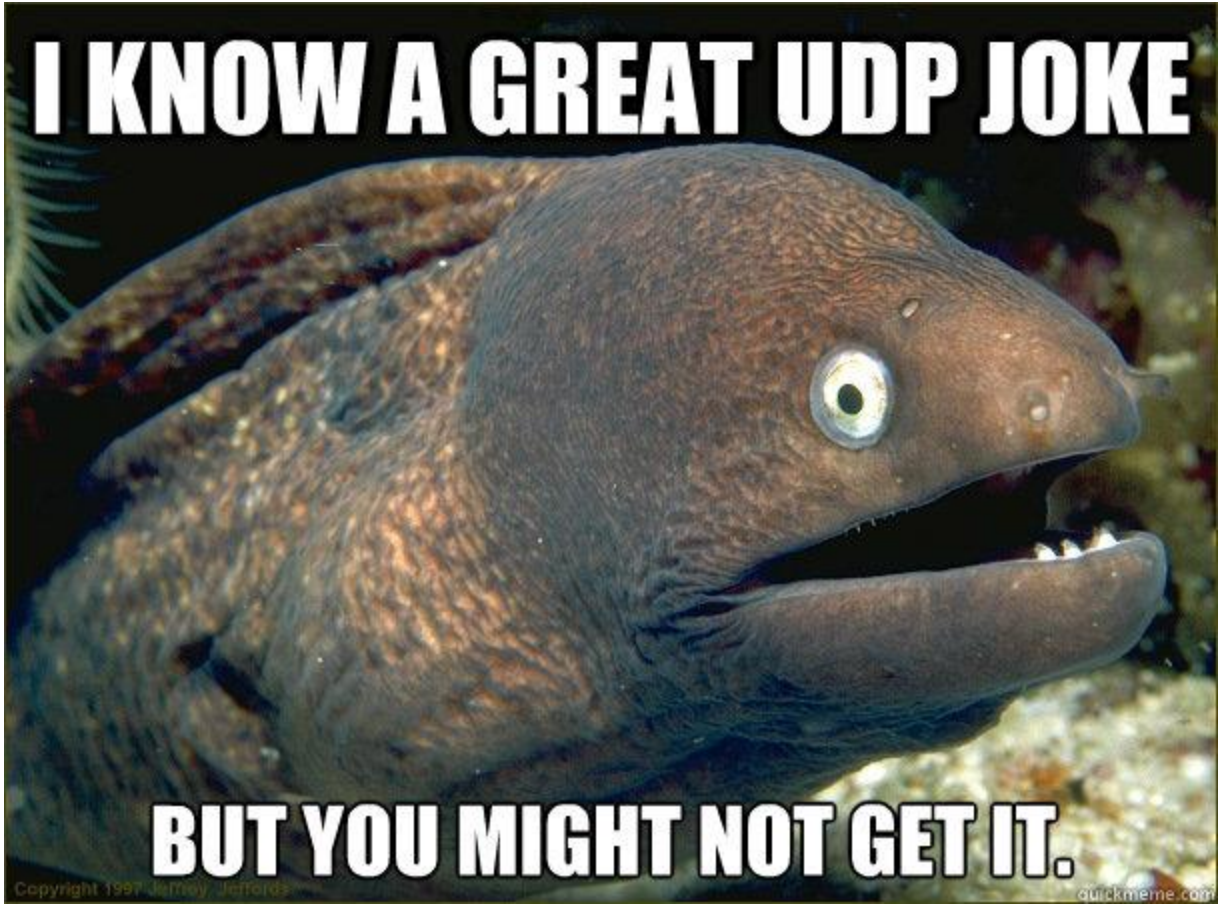
9. Bibliografie

Bibliografie slideuri

1. Curs “Retele de Calculatoare”, 2023, FMI, UNIBUC - Sergiu Nisioi
2. Laborator “Retele de Calculatoare”, 2023, FMI, UNIBUC - Vlad Olaru
3. “Docker,” n.d. https://hub.docker.com/layers/library/maven/3.8.3-eclipse-temurin-17/images/sha256-af8068725ab22efc4f46dee1740684a7108dca39824aeaf8dd97b2176016d175.
4. GeeksforGeeks. “Socket Programming in C C.” GeeksforGeeks, February 20, 2023. https://www.geeksforgeeks.org/socket-programming-cc/.
5. Stevewhims. “Sockaddr - Win32 Apps.” Microsoft Learn, January 7, 2021. https://learn.microsoft.com/en-us/windows/win32/winsock/sockaddr-2.
6. “ProcessBuilder (Java SE 11 & JDK 11),” April 11, 2023. https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/ProcessBuilder.html.

Bibliografie cod

Toata bibliografia folosita in interiorul programelor de Java sau a containerelor de Docker este inclusa in fisierele respective



Sursa: https://www.mimuw.edu.pl/~mrp/sen.html