BARTIN ÜNİVERSİTESİ MÜHENDİSLİK, MİMARLIK VE TASARIM FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

BİLGİSAYAR AĞLARI DÖNEM PROJESİ

SOFTWARE DEFINED NETWORK RAPORU

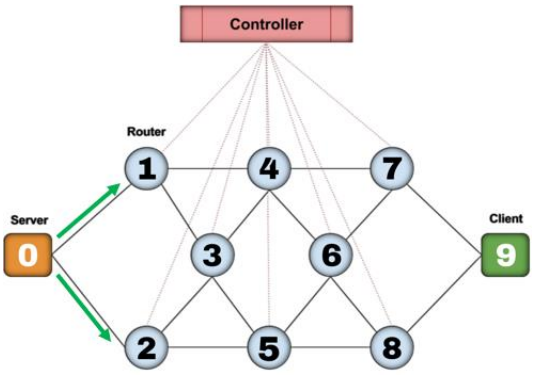
- Abdül Samed Doğan – 19010310017
- Allanur Muhammetnazarov – 18670310065
- Samet Yıldırım – 18010310010
- Kerem Can Korkmaz – 18010310016
- Ali Murat Kalaycı – 19010310038
- Gürkan Yavuz – 19010310056
- Mahmud Kabbisho - 18670310081

Dr. Öğr. Üyesi Evrim GÜLER

ÖZET

Projenin amacı, Client ile Server arasındaki haberleşmeyi sağlamaktır. Sender görevi gören Server, paketi gönderen birimdir. Receiver birimi olan Client ise paketi alıp geri bildirim yapmaktadır. Routlerlar aracılığıyla Sender ile Receiver arasındaki haberleşme gerçekleşir. Bu haberleşme yapılırken 8 adet port kullanılır. Paketler rastgele portlar üzerinden Receivera ulaşır ve ulaştığı yoldan Sendera geri gelir.

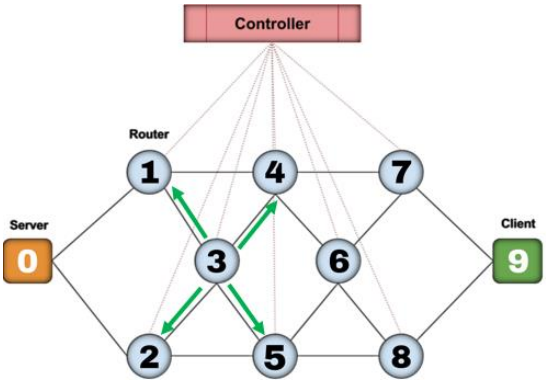
- **Gidebileceği Yollar**




```
int[] paths = {1, 2};

Random randomGenerator = new Random();
int randomInt = randomGenerator.nextInt(2);

int router_ad = paths[randomInt];

int NewPort = PORT + router_ad;

link = new Socket(host, NewPort);
```

Projeye başlarken ilk olarak yol haritası oluşturduk. Bu haritada routerın gidebileceği yolları tespit edip dizi içerisinde tanımladık. Daha sonra dizi içerisindeki sayılardan rastgele bir sayı oluşturduk. Oluşan sayıyı 1300'e ekledik ve çıkan sonucu routerın portu olarak tanımladık. Örneğin çıkan sonuç 1 ise routerımızın portu 1301 olacak. Son olarak yeni portumuz ile bir socket oluşturduk ve haberleşmeyi sağladık.

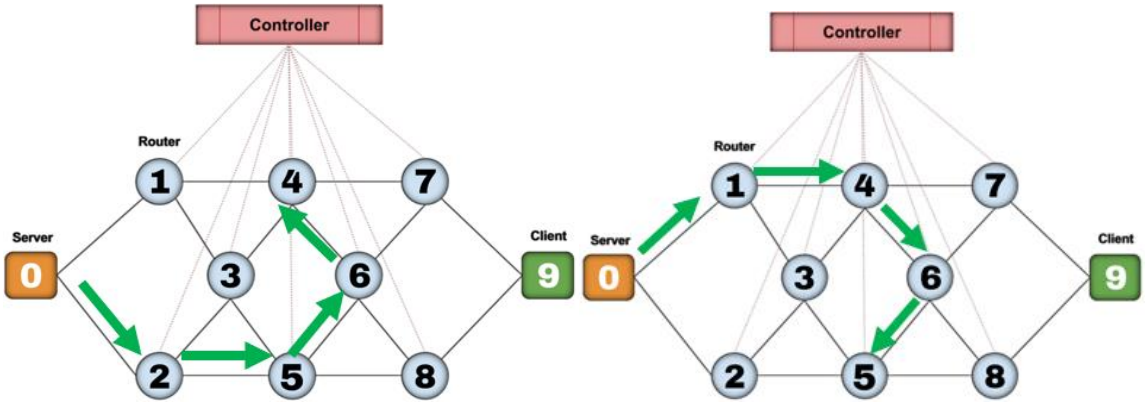
- **Geçtiği Yolu Silme Durumu**




```
int[] paths = {1,2,4,5};

int[] paths2 = Functions.pastpath(message, paths);

Random randomGenerator = new Random();
int randomInt = randomGenerator.nextInt(paths2.length);

int router_ad = paths2[randomInt];

int NewPort = PORT2 + router_ad;
link2 = new Socket(host, NewPort);
```

Paketin 3'ten sonra gidebileceği 4 adet yol vardır fakat paket geldiği yola geri dönmemelidir. Örneğin 1 numaralı porttan 3 numaralı porta geldiyse 3'ten sonra gidebileceği 4 adet değil 3 adet yol olması gerekir, 1 numaralı porta gidemez. Bu durumu ele alarak paketin gidebileceği yolu bulan bir fonksiyon oluşturduk ve bu fonksiyonu routerlara ekledik.

- **İstisnai Durumlar**



Paketimiz 4 numaralı routerdayken 2 ve 5, 5 numaralı routerdayken 1 ve 4 yollarından geçmişse yolun tıkanmaması için 3 numaralı routerın devre dışı bırakılması gerekmektedir. Yolun çıkmaza girmemesi ve programın hata vermemesi için bir fonksiyon oluşturduk.

- **Fonksiyonun Çalışma Prensibi**

```
public static int[] pastpath(String paket, int[] paths) {
    ArrayList<Integer> pathArray = new ArrayList<Integer>();
    ArrayList<Integer> ortak = new ArrayList<Integer>();

    for (int eleman : paths) {
        pathArray.add(eleman);
    }

    String[] strings = paket.split("_");

    String str = strings[1];

    char[] c_arr = str.toCharArray();

    if (paths[0] == 3 && paths[1] == 6 && paths[2] == 7) {

        int toplam = 0;
        for (int i = 0; i < c_arr.length; i++) {
            int eleman = Character.getNumericValue(c_arr[i]);
            System.out.print(eleman);
            if (eleman == 2 || eleman == 5) {
                toplam++;
            }
        }

        if (toplam == 2) {
            ortak.add(3);
        }
    }
```

```
if (paths[0] == 3 && paths[1] == 6 && paths[2] == 8) {

    int toplam2 = 0;
    for (int i = 0; i < c_arr.length; i++) {
        int eleman = Character.getNumericValue(c_arr[i]);

        System.out.print(eleman);

        if (eleman == 1 || eleman == 4) {
            toplam2++;
        }
    }

    if (toplam2 == 2) {
        ortak.add(3);
    }
}

int[] pathes = new int[str.length()];

for (int i = 0; i < c_arr.length; i++) {
    pathes[i] = Integer.parseInt(String.valueOf(c_arr[i]));
    for (int j = 0; j < paths.length; j++) {

        if (pathes[i] == paths[j]) {
            ortak.add(paths[j]);
        }

    }
}
```

```
for (int a = 0; a < paths.length; a++) {

    if (ortak.contains(paths[a])) {
        Object c = paths[a];
        pathArray.remove(c);

    }

}

int[] result = new int[pathArray.size()];

for (int s = 0; s < pathArray.size(); s++) {
    result[s] = pathArray.get(s);

}

return result;
```

Paketin kendisini ve o anki routerın gidebileceği yollarını parametre olarak alan, sonuç olarak da routerın gidebileceği yolları döndüren bir fonksiyonumuz var. Fonksiyonumuz 2 temel durumu kontrol etmektedir. 1. durum; 1 paket 1 routerdan sadece 1 kere geçebilir ve router 4 ve 5'te iken istisnai bir durumu kontrol eder.

İlk olarak if sayesinde o anki routerın gidebileceği yollara bakıyoruz ve bu yollar 3, 6, 7 ise o anki routerın 4 olduğunu anlıyoruz. Daha sonra paketin gittiği yolları eleman değişkenine atıyoruz ve eleman değişkeninin içerisinde hem 2 hem de 5 var ise paketin gidebileceği yollardan 3 numaralı routerı çıkartmak için 3 numaralı routerı ortak dizisine atıyoruz. Aynı işlemi 5. router için tekrarlıyoruz ve iç içe for döngüsü kullanarak routerın gidebileceği yollar ile paketin daha önce gittiği yolları kontrol ediyoruz. Eğer ortak router var ise bunu ortak dizisine atıyoruz.

Bir sonraki for döngüsünde ortak dizisiyle routerın gidebileceği yolları kontrol ediyoruz ve ortak yolları pathArray dizisinden çıkartıyoruz. Son olarak pathArray dizisini result dizisine atarak result dizisini geri döndürüyoruz.

- **SENDER**

```java
import java.io.*;
import java.net.*;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Sender extends Thread {

    private static InetAddress host;
    private static final int PORT = 1300;

    public static void main(String[] args) {
        try {
            host = InetAddress.getLocalHost();
            System.out.println("Enter TcpRouter IP Address:");
        } catch (Exception uhEx) {
            System.out.println("Host ID not found!");
            System.exit(1);
        }
        accessServer();
    }
```

```java
do {
    int[] paths = {1, 2};

    Random randomGenerator = new Random();
    int randomInt = randomGenerator.nextInt(2);

    int router_ad = paths[randomInt];

    int NewPort = PORT + router_ad;

    link = new Socket(host, NewPort);
    link.setSoTimeout(4000);
    Scanner input = new Scanner(link.getInputStream());
    PrintWriter output = new PrintWriter(link.getOutputStream(), true);

    message = "PCK";

    message = "PCK" + counter + "_" + String.valueOf(router_ad);
    output.println(message);
    attempt++;

    String str = input.nextLine();

    str2 = str.substring(0, 3);
```

```java
private static void accessServer() {
    Socket link = null;

    try {

        for (int i = 0; i < 6; i++) {

            System.out.println("How many packets? ");
            Scanner userEntry = new Scanner(System.in);
            String message, str2, response;
            int number;
            response = userEntry.nextLine();
            number = Integer.parseInt(response);
            int counter = 0, attempt = 0;
            long startTime = System.nanoTime();
```

```java
    while (!str2.equals("ACK")) {

        link.close();
        link = new Socket(host, NewPort);
        input = new Scanner(link.getInputStream());
        output = new PrintWriter(link.getOutputStream(), true);

        System.out.println(message + counter + " Resending...");
        output.println(message + counter);
        attempt++;
        str = input.nextLine();
        str2 = str.substring(0, 3);
    }
    System.out.println(str + " received from receiver successfully");

    counter++;

    if (counter == number) {

        output.println("***CLOSE***");
    }
    try {
        Thread.sleep(50);
    } catch (InterruptedException ex) {
        Logger.getLogger(Sender.class.getName()).log(Level.SEVERE, null, ex);
    }
} while (counter < number);

long endTime = System.nanoTime();
System.out.println("Total number of try: " + attempt);
System.out.println("Time taken to send all packets: " + (endTime - startTime) + " nano seconds.");
}
```

Senderda kullanıcıya kaç paket gönderilecek sorusu sorulur. Alınan paket sayısı kadar paket yollanır. Paketler rastgele bir şekilde 1. veya 2. routerı seçer ve bu router adresinden bir socket oluşturulur. Daha sonra bu socketten yazma ve okuma nesneleri oluşturulur. Router bağlantıyı kabul ettiğinde dinlemeye başlar ve sender yazma nesnesi ile paketleri gönderir. Her paketin sonuna kaçıncı paket olduğu ve hangi routerı seçtiği yazılır. Paketler bu şekilde gönderilir.

Paketler receivera ulaşıp ACK mesajı döndüğünde router yazma nesnesinden yazma yapar ve sender okuma nesnesi ile gelen mesajları okur. Daha sonra gelen ACK mesajları ekrana yazdırılır. Paketlerin gitme ve geri dönme süresi hesaplanıp ekrana yazdırılır. İstenilen kadar paket gittiğinde yazma paketi ile CLOSE mesajı yollanır ve işlemler durdurulur.

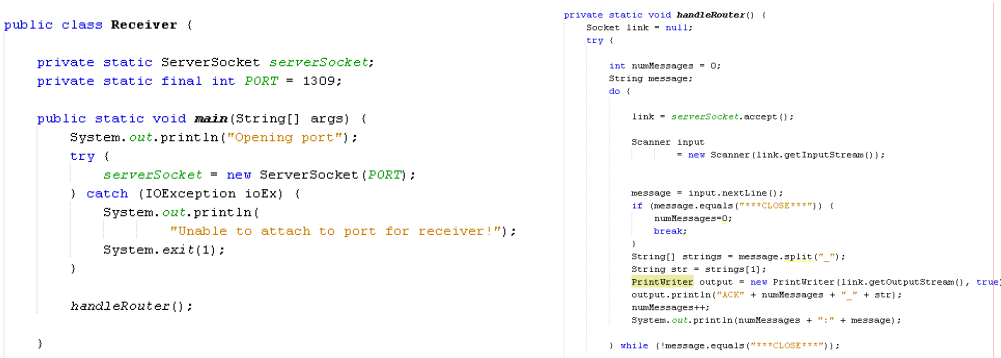
- **ROUTER**

```
private static String handleClient() {
    Socket link = null;
    String str2 = null;
    try {
        System.out.println("Router1");
        String message;
        do {

            link = serverSocket.accept();
            Scanner input = new Scanner(link.getInputStream());
            message = input.nextLine();
            int[] paths = {3, 4};

            int[] paths2 = Functions.pastpath(message, paths);

            Random randomGenerator = new Random();
            int randomInt = randomGenerator.nextInt(paths2.length);

            int router_ad = paths2[randomInt];

            int NewPort = PORT2 + router_ad;
            link2 = new Socket(host, NewPort);
            if (message.equals("***CLOSE***")) {
                break;
            }

            PrintWriter output = new PrintWriter(link.getOutputStream(), true);

            Scanner input2 = new Scanner(link2.getInputStream());

            PrintWriter output2 = new PrintWriter(link2.getOutputStream(), true);
            System.out.println("message from sender " + message);
            message = message + String.valueOf(router_ad);
            int random = randomGenerator.nextInt(100);
            System.out.println("Generated random number for the packet is: " + random);
            if (random > -1) {

                output2.println(message);

                String str = input2.nextLine();
                System.out.println("message from receiver: " + str);
                output.println(str);
                link2.close();
            } else {
                output.println(str2);

                link2.close();
            }

            try {
                Thread.sleep(50);
            } catch (InterruptedException ex) {
                Logger.getLogger(Router1.class.getName()).log(Level.SEVERE, null, ex);
            }

        } while (!message.equals("***CLOSE***"));
```

Her routerın kendine ait bir port numarası vardır. Bu kod bloğu Router 1 için geçerlidir. Diğer routerlar için aynı kalıp bulunmaktadır fakat bazı kısımları farklılık göstermektedir.

Router kendi port numarası ile port açar ve kendisiyle haberleşen birim için bağlantıyı kabul eder. Kabul edilen bağlantı için yazma ve okuma nesneleri oluşturur. Okuma nesnesi ile bu routera yollanan paket okunur. Eğer gelen paket CLOSE mesajı ise işlemler durdurulur. Daha sonra kendi haberleşebileceği router için fonksiyon kullanılarak rastgele bir router seçilir ve socket oluşturulur. Bu socket ile okuma ve yazma nesnesi oluşturulup mesajın sonuna seçilen router numarası yazılır. Yazma nesnesi aracılığıyla paket bir sonraki routera yollanır. Okuma nesnesi ile cevap gelen ACK mesajı okunur. Gelen ACK mesajı yazma nesnesi aracılığıyla bağlantıyı kabul ettiğimiz birime gönderilir. Bu işlemler gerçekleşirken gelen mesaj ve giden mesaj ekrana yazdırılır.

- **RECEIVER**



```
public class Receiver {

    private static ServerSocket serverSocket;
    private static final int PORT = 1309;

    public static void main(String[] args) {
        System.out.println("Opening port");
        try {
            serverSocket = new ServerSocket(PORT);
        } catch (IOException ioEx) {
            System.out.println(
                    "Unable to attach to port for receiver!");
            System.exit(1);
        }

        handleRouter();

    }
```

```
private static void handleRouter() {
    Socket link = null;
    try {

        int numMessages = 0;
        String message;
        do {

            link = serverSocket.accept();

            Scanner input
                    = new Scanner(link.getInputStream());


            message = input.nextLine();
            if (message.equals("***CLOSE***")) {
                numMessages=0;
                break;
            }
            String[] strings = message.split("_");
            String str = strings[1];
            PrintWriter output = new PrintWriter(link.getOutputStream(), true);
            output.println("ACK" + numMessages + "_" + str);
            numMessages++;
            System.out.println(numMessages + ":" + message);

        } while (!message.equals("***CLOSE***"));
```

Receiver, gelen bağlantı isteğini kabul eder. Bu bağlantı için okuma ve yazma nesneleri oluşturur. Okuma nesnesi ile gelen mesaj okunur ve ekrana yazdırılır. Gelen mesaj CLOSE ise işlemler durdurulur. Okuma bittikten sonra Yazma nesnesi ile ACK mesajının sonuna geldiği yol yazılır ve geri gönderilir.

Projenin GitHub Bağlantısı:

AbdulSametYildirim/BilgisayarAglari_Grup14 (github.com)

KAYNAKÇA

- evrimguler/SocketProgrammingJAVA: JAVA based socket programming (github.com)
- Bilgisayar Ağları (A Bilgisayar Mühendisliği Bölümü (Mühendislik Fakültesi)) (bartin.edu.tr)