# SOFTWARE-DEFİNED NETWORKİNG (SDN)
# Yazılım Tanımlı Ağlar

Projenin amacı: Bu projede, 1 SDN denetleyicisi, çoklu sunucu ve istemcileri ve sunucu ile istemciler arasında ara düğümleri olan Yazılım Tanımlı Ağ tabanlı bir ağ
(SDN) sistemi modellemesi.

Projenin Görevleri:
1. SDN tabanlı ağ sistemi kavramını anlayın
2. Her modülün planını tanımlayın
3. Ağ sistemini TCP / IP iletişim protokollerine dayalı olarak uygulayın
4. Proje uygulamasını gösterin 5. Performans değerlendirmesi ile birlikte proje raporunu gönderin

Projede kullanılan yapılar şunlardır: Mininet, Floodlight, Sanal bilgisayar uygulamaları veya Linux işletim sistemi

Yapmış olduğumuz sistem modellemesinin ne olduğunu kısaca açıklayalım.
SDN nedir sorusunun cevabı, kısaca ağ kontrol düzleminin (control plane) yönlendirme-veri düzleminden (data plane) fiziksel olarak ayrılması ve bir kontrol düzleminin birkaç cihazı kontrol ettiği yer olarak tanımlayabiliriz.

Mininet'in Kurulumu
git clone https://github.com/mininet/mininet
cd mininet girişi
install.sh -a diyerek kurulum işlemi başlar

Floodlight'ın Kurulumu
sudo apt-get install build-essential openjdk-7-jdk ant maven python-dev eclipse yazılır.
Daha sonra Floodlight'ı GitHub üzerinden
komut satırına;
git clone git://github.com/floodlight/floodlight.git
yazılarak Floodlight kontrolcüsü indirilir.

Özel Topolojinin Oluşturulması
Mininet'in sağladığı sanallaştırma yapısıyla, basit veya karmaşık ağ mimarileri tasarlanabilmektedir.

```python
from mininet.topo import Topo
from mininet.node import Controller, RemoteController
from mininet.cli import CLI

class MyTopo( Topo ):

    def build( self ):
        "Create custom topo."

        # Add hosts and switches
        h1 = self.addHost( 'h1' )
        h2 = self.addHost( 'h2' )
        s1 = self.addSwitch( 's1' )
        s2 = self.addSwitch( 's2' )
        s3 = self.addSwitch( 's3' )
        s4 = self.addSwitch( 's4' )
```

```
        # Add links
        self.addLink( h1, s1 )
        self.addLink( s1, s2 )
        self.addLink( s1, s3 )
        self.addLink( s2, s4 )
        self.addLink( s3, s4 )
        self.addLink( s4, h2 )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Soket Programlama

```
print("waiting client to send message, Bye or bye to End")
s = socket.socket()
print("socket.socke()==",s)
s.bind(('', port))
s.listen(5)
c, addr = s.accept()
print ("Socket Up and running with a connection from",addr)
while True:
    rcvdData = c.recv(1024).decode()
    print ("S:",rcvdData)
    if(rcvdData=='t'):
        from datetime import datetime
        now = datetime.now()
        current_time = now.strftime("%H:%M:%S")
        c.send(current_time.encode())
    elif(rcvdData=='d'):
        from datetime import date
        today = date.today()
        #dd/mm/YY
        d1 = today.strftime("%d/%m/%Y")
        c.send(d1.encode())
    else:
        sendData = input("N: ")
        c.send(sendData.encode())
        if(sendData == "Bye" or sendData == "bye"):
            break
c.close()
```

Socket programlama kullanılarak iki host arasında haberleşme sağlanıyor. Yukarıda verilen kodlar server hostunda açılması gerekmekte.

```python
import socket

print("send message, Bye or bye to End")
s = socket.socket()
s.connect(('10.0.0.1',12345))
while True:
    str = input("t for time, d for date , IP for ip address: ")
    #str = raw_input("S: ")      #python 2
    s.send(str.encode());
    if(str == "Bye" or str == "bye"):
        break
    print ("N:",s.recv(1024).decode())
s.close()
```

Bu kodlar ise 2. Host yani client olarak seçilen hostta açılıp kolayca mesajlaşma sağlanabilmekte.

Özel Flow (Akış) Oluşturulması:

Özel akışların olması sebebi ise projenin bir isteri olan hostlar arasında paket ya da mesaj gönderiminin istenen farklı yollarla yapılması sağlanmasıdır.
Oluşturulan özel akışlar sayesinde 1. Hosttan çıkan paketin 2. Hosta ulaşana kadar hangi yollardan geçmesi gerektiğini karar verebiliriz.

```python
import json
import requests

# Replace with the IP address of your Floodlight controller
controller_ip = "127.0.0.1"

# Define the flows to be created
flows = [
    {
        "switch": "00:00:00:00:00:00:00:01",
        "name": "flow1",
        "cookie": "0",
        "priority": "32768",
        "ingress-port": "1",
        "active": "true",
        "actions": "output=2"
    },
```

```
    },
    {
        "switch": "00:00:00:00:00:00:00:04",
        "name": "flow10",
        "cookie": "0",
        "priority": "32768",
        "ingress-port": "3",
        "active": "true",
        "actions": "drop"
    }

]

# Send a POST request to the Floodlight controller to create the flows
url = f"http://{controller_ip}:8080/wm/staticflowentrypusher/json"
headers = {"Content-Type": "application/json"}
for flow in flows:
    response = requests.post(url, data=json.dumps(flow), headers=headers)
    print(response.text)
```

NOT: Projenin kaynak kodları ve dosyaları Onedive klasörüne eklenmiş olup link kısmına eklenmiştir!

Ahmad Masood SAHAK
Mert KEZER
Emirhan Çam
Selçuk ARSLAN
Serhat ERGÜL