# YAZILIM TABANLI AĞ

Bilgisayar Ağları Dönem projesi


| | |
|---|---|
| Mert Fazla | 20010310002 |
| Seyfullah Kurt | 20010310026 |
| Beytullah Aziz Yapar | 20010310028 |
| Alparslan Nedimoğlu | 18670310078 |

# İçindekiler

GİRİŞ ..... 1
Projenin Amacı ..... 1
Yazılım Tabanlı Ağ (SDN) Nedir? ..... 1
SDN Controller Nedir? ..... 1
Switch Nedir? ..... 2
Topoloji nedir? ..... 2
Client ve Server Nedir? ..... 2
MAC ve İP Adresi Nedir? ..... 3
TCP ve UDP Nedir? ..... 4
Request ve Response Nedir? ..... 5
Gerekli Programlar ..... 6
KURULUMLAR ..... 7
Neden Virtual Box? ..... 7
VirtualBox Kurulumu ..... 7
Neden Ubuntu? ..... 8
Ubuntu Kurulumu ..... 8
Neden Pyton? ..... 8
Pyton Kurulumu ..... 9
Neden mininet? ..... 9
Neden pox? ..... 9
Mininet ve Pox Kurulumu ..... 9
DÜZENLEME ve ANALİZ ..... 10
POX Controller için yapılacak component planı ..... 10
Component kodunun analizi ..... 11
ÇALIŞTIRMA ve SONUÇ ..... 15
KAYNAKÇA ..... 17

# ŞEKİL LİSTESİ

Şekil 1.1 ..... 1
Şekil 1.2 ..... 2
Şekil 1.3 ..... 3
Şekil 1.4 ..... 3
Şekil 1.5 ..... 4
Şekil 1.6 ..... 5
Şekil 1.7 ..... 5
Şekil 1.8 ..... 6
Şekil 1.9 ..... 6
Şekil 1.10 ..... 6
Şekil 2.1 ..... 7
Şekil 2.2 ..... 8
Şekil 3.1 ..... 10
Şekil 3.2 ..... 11
Şekil 3.3 ..... 11
Şekil 3.4 ..... 12
Şekil 3.5 ..... 12
Şekil 3.6 ..... 13
Şekil 4.1 ..... 15
Şekil 4.2 ..... 15
Şekil 4.3 ..... 15
Şekil 4.4 ..... 16

# GİRİŞ

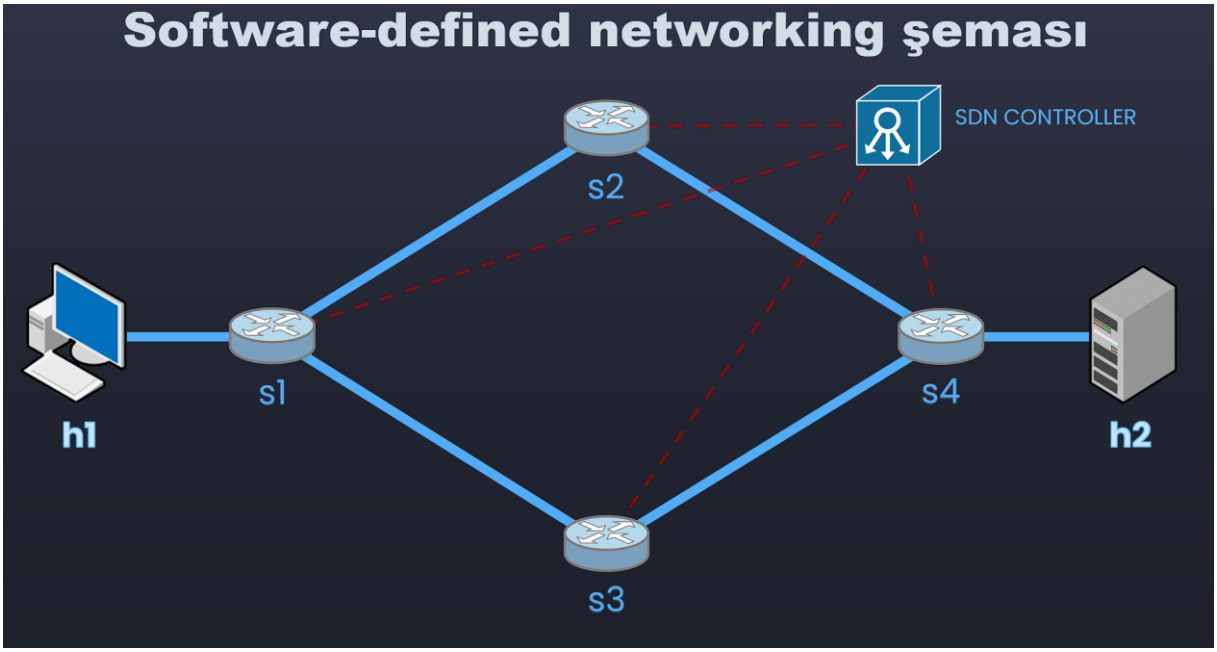
## Projenin Amacı

Projenin amacı en temel seviyede, yazılım tabanlı ağların anlaşılmasını sağlamaktır.

## Yazılım Tabanlı Ağ (SDN) Nedir?

Ağ`ın yazılımlar sayesinde merkezi olarak yönetilmesini veya programlanmasını sağlayan bir ağ mimarisidir. Tüm ağın tutarlı ve bütün olarak ele alınmasına ve yönetilmesine yardımcı olur.

SDN controllerin fiziksel bir topoloji oluşturabilir. Düğümler bağlanır ve bazı algoritma eşleştirmelerine dayanarak ağ üzerinden yollar oluşturur. Son olarak, yollar cihazların yönlendirme motorlarına programlanır. Bu, SDN controllerin tüm ağdaki trafik akışlarını daha iyi yönetmesine ve değişiklikleri daha hızlı yanıtlamasına olanak tanır.




*Şekil 1.1*

## SDN Controller Nedir?

Şekil 1.1`de Switchlere kırmızı çizgilerle bağlanmış objedir. SDN Controller, gelişmiş ağ yönetimi ve uygulama performansı için akış denetimini yöneten, SDN mimarisindeki bir uygulamadır. SDN Controller tipik olarak bir Server üzerinde çalışır ve Switchlere paketlerin nereye gönderileceğini söylemek için protokoller kullanır.

SDN denetleyicileri, trafiği bir ağ operatörünün uygulamaya koyduğu yönlendirme politikalarına göre yönlendirir ve böylece tek tek ağ cihazları için manuel yapılandırmaları en aza indirir. Kontrol düzlemini ağ donanımından çıkarıp bunun yerine yazılım olarak çalıştıran merkezi denetleyici, otomatikleştirilmiş

ağ yönetimini kolaylaştırır ve iş uygulamalarını entegre etmeyi ve yönetmeyi kolaylaştırır. Gerçekte, SDN denetleyicisi ağ için bir tür işletim sistemi görevi görür.

Denetleyici, yazılım tanımlı bir ağın çekirdeğidir. Ağın bir ucundaki ağ cihazları ile diğer ucundaki uygulamalar arasında bulunur. Uygulamalar ve ağ cihazları arasındaki herhangi bir iletişim, denetleyici üzerinden geçmelidir.

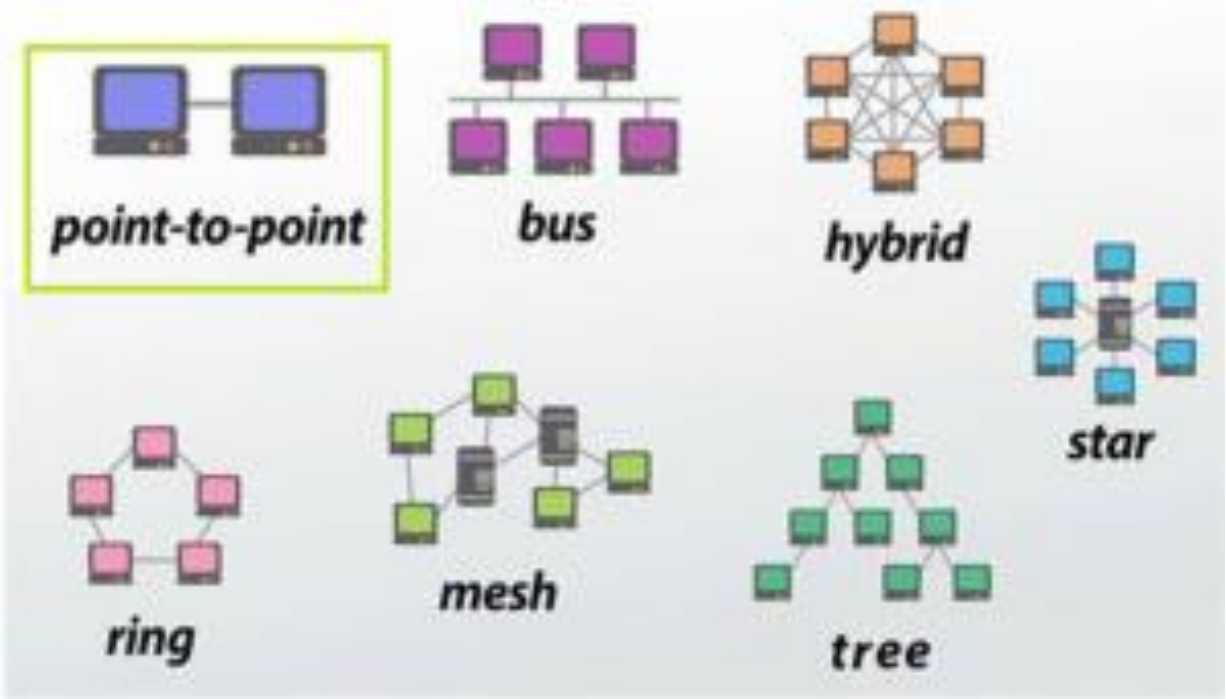
## Switch Nedir?

Switch bilgisayarların ve diğer ağ öğelerinin birbirlerine bağlanmasına olanak veren ağ donanımlarından biridir. Şekil 1.1`de s1, s2, s3 ve s4 ile gösterilen öğeler birer Switchtir.

Dağıtıcı yıldız topolojisine sahip yerel ağlar oluşturmanın yanında, paket anahtarlamalı farklı ağları birbirine bağlamakta da kullanılabilirler.

## Topoloji nedir?

Ağ topolojisi, bir bilgisayar ağının çeşitli öğelerinin düzenlenmesidir. Temelde bir ağın topolojik yapısı fiziksel veya mantıksal olarak Tasvir edilebilir. Client ve Server arasındaki yolların haritası gibi düşünebilirsiniz.

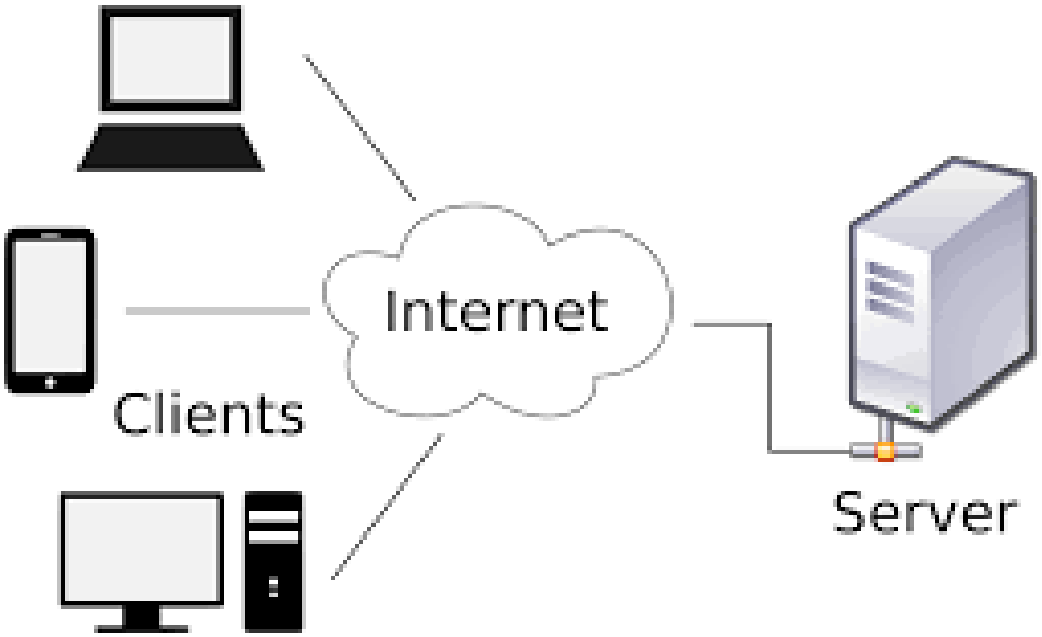



*Şekil 1.2*

## Client ve Server Nedir?

Client-Server, Clienti Serverdan ayıran bir ağ mimarisidir. Her bir Client yazılımı, Servera ya da uygulama Serverına isteklerini (request) gönderir.

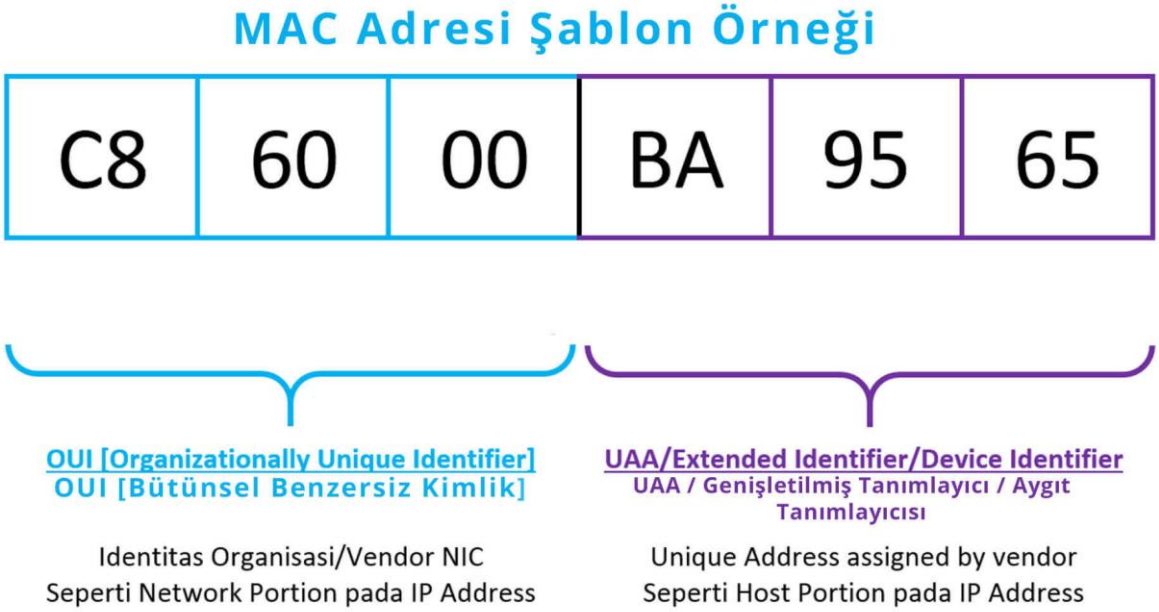
Bu fikrin pek çok çeşitli uygulaması olmasına karşın, en güzel örneği İnternet üzerindeki Web sayfalarıdır. Bir web sayfası incelenirken, bilgisayar ve web tarayıcısı Client olarak adlandırılır. Web sayfasını oluşturan gelişmiş bilgisayarlar, veritabanları ve uygulamalar da Server olarak adlandırılır. Web tarayıcısı, web sitesinden bir istekte bulunur ve Server istenen bilgileri toplar ve onu bir web sitesi şekline getirerek web tarayıcısına geri yollar, kullanıcılar da ekranda web sitesini görmüş olur.




*Şekil 1.3*

## MAC ve İP Adresi Nedir?

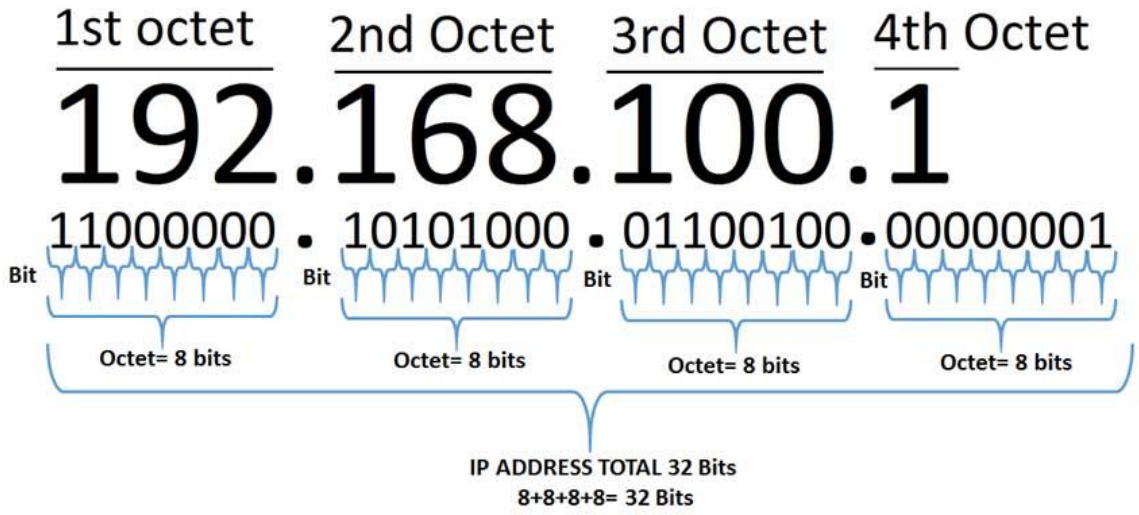
Mac ve IP adresi, ağa bağlı cihazı tanımlar. Mac'iniz tıpkı ev adresiniz gibi bilgisayarınızın fiziksel adresidir. Bu şekilde, ağdaki farklı cihazları tanımlamak için Mac adresleri kullanılabilir. Bir IP adresi, bir numara vererek bir ağ bağlantısını tanımlar. Mac ve IP adresi makalesinde, her iki adres de cihazları ağlara bağlamak için önemlidir. Bir cihaza bir MAC adresi atanır ve cihazın ağ kartına özeldir. IP adresleri, bilgisayarlar ve ağlar arasındaki bağlantıları tanımlar.




*Şekil 1.4*

Üreticiler, bilgisayar donanım bileşenlerinde NIC'lere (ağ arayüzü denetleyicileri) numaralar atar. Bu numaraya Mac adresi denir.

MAC adresi, cihazın hangi verileri alacağını kontrol eder. Bu veriler ağ üzerinden alınır. Cihazlar İnternete Ethernet veya Wi-Fi üzerinden bağlanır ve ağdaki MAC adreslerini kullanır. MAC adresinin 12 hanesi onaltılıktır. Bu 12 basamak, iki nokta üst üste ile ayrılmış 6 çift olarak görüntülenir. MAC adresinin ilk yarısı satıcı kimliğidir ve ikinci yarısı cihazı tanımlamak için kullanılır. Birinci kısım aynı zamanda Enterprise Unique Identifier olarak da adlandırılır.



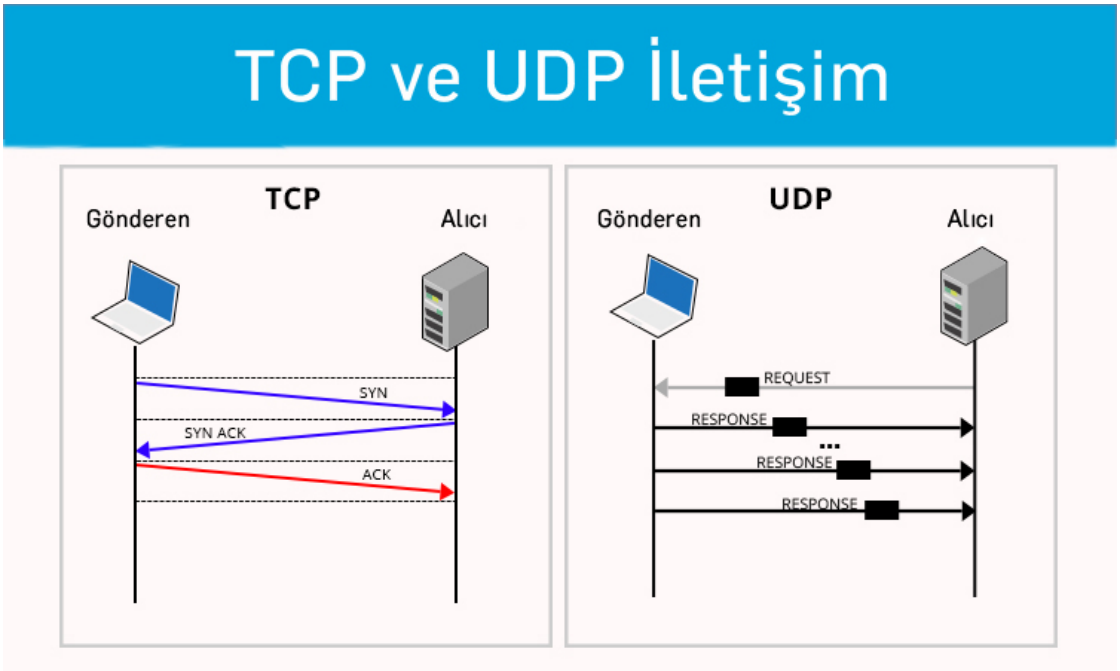

*Şekil 1.5*

Bir IP adresi, cihazınızı küresel pazarda tanımlamanıza yardımcı olur. Bir IP adresi, noktalarla ayrılmış dört ondalık sayı olarak yazılır. Bazı numaralar özel amaçlar için ayrılmıştır. IPv4 32 bit ve IPv6 128 bittir. Ağ ve cihaz tanımlaması, IPv4 adresleri kullanılarak yapılır. Bir IPv6 adresi, iki nokta üst üste ile ayrılmış dört onaltılık basamaktan oluşan sekiz set olarak yazılır. IPv6 ayrıca bir ağ kimliğine ve bir cihaz kimliğine sahiptir. Yazılım yapılandırması, IP adreslerinin küresel ağdaki cihazlara bağlanmasına yardımcı olur.

## TCP ve UDP Nedir?

TCP, TCP/IP protokol takımının aktarım katmanı protokollerinden birisidir. Gelişmiş bilgisayar ağlarında paket anahtarlamalı bilgisayar iletişiminde kayıpsız veri gönderimi sağlayabilmek için TCP protokolü yazılmıştır.

UDP, TCP/IP protokol ailesindeki iki taşıma katmanı protokolünden biridir. Bağlanmadan veri gönderin. UDP protokolü, gelişmiş bilgisayar ağlarında paket anahtarlamalı bilgisayar iletişiminin bir datagram modunu sağlamak için oluşturulmuştur.




*Şekil 1.6*

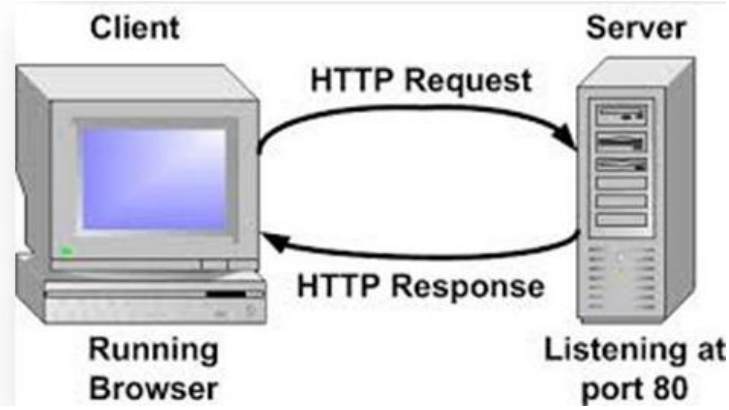
TCP: Verilerin gelip gelmediğini kontrol ettiği için UDP'den daha yavaştır.

UDP: Ses ve video göndermek için kullanılır. TCP'den daha hızlı, ancak daha az güvenli. Bir veri adına datagram denir. Datagramlar ve segmentler arasındaki fark, sıra numaraları içermemeleridir.

Sonuç olarak, TCP hızı yavaş olabilir, ancak TCP protokolü hızlıdır. UDP protokolü üzerinden doğruluk ve güvenlik için daha fazla seçenek sunar. UDP, ses ve video gibi verileri aktarmak için kullanışlıdır. UDP oyunlar için çok kullanışlıdır.

## Request ve Response Nedir?

Bilgisayar biliminde, Request-Response veya istek-yanıt, bilgisayarların bir ağda birbirleriyle iletişim kurmak için kullandıkları temel yöntemlerden biridir; burada ilk bilgisayar bazı veriler için istek gönderir ve ikinci bilgisayar isteğe yanıt verir. Daha spesifik olarak, bir talep sahibinin, talebi alan ve işleyen ve sonuçta yanıt olarak bir mesaj döndüren bir yanıtlayıcı sisteme bir istek mesajı gönderdiği bir mesaj alışveriş modelidir. Bu, herhangi bir şey tartışılmadan önce arayanın alıcının telefonu açmasını beklemesi gereken bir telefon görüşmesine benzer. Bu, iki uygulamanın bir kanal üzerinden birbiriyle iki yönlü görüşme yapmasına izin veren basit ama güçlü bir mesajlaşma modelidir; özellikle istemci-sunucu mimarilerinde yaygındır.




*Şekil 1.7*

## Gerekli Programlar

Projemiz için  sanal makine emulatörü ve topoloji görselleçtirici ve yazılım tabanlı ağ denetleyici (SDN Controller) gerekiyor bu yüzden VirtualBox Mininet ve Pox a ihtiyacımız var.


*Şekil 1.8*


*Şekil 1.9*


*Şekil 1.10*

# KURULUMLAR

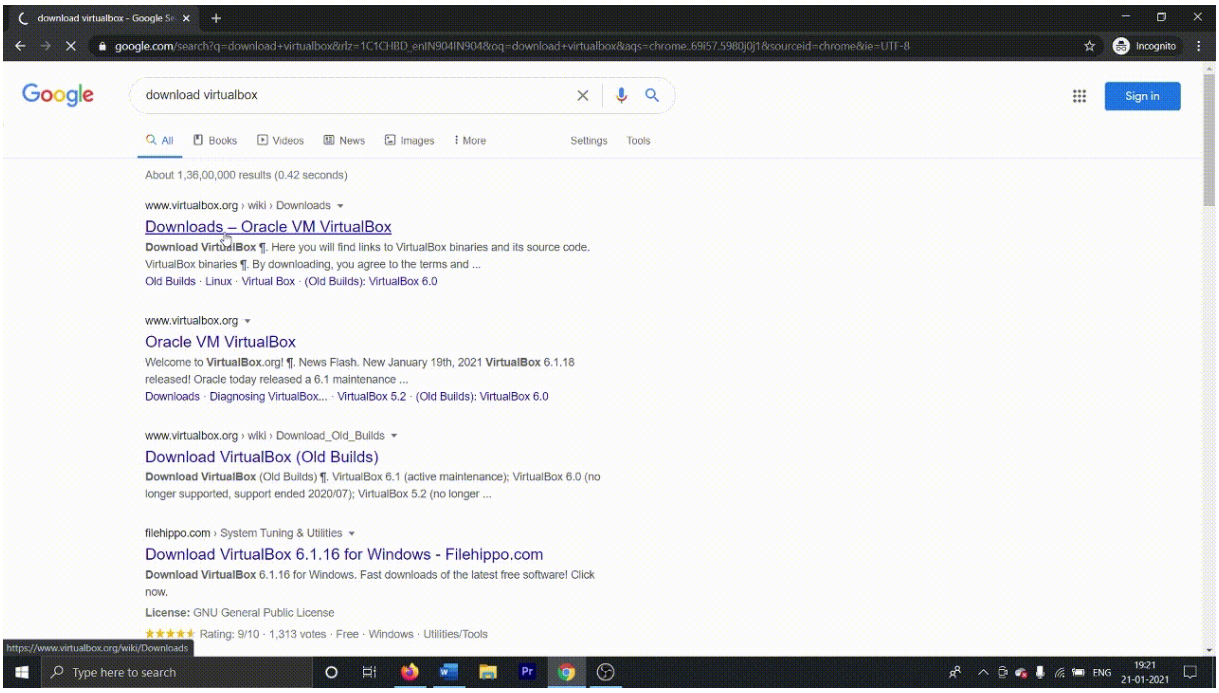
Programı çalıştırmak için sanal makine kullandık. VirtualBox kurduk içinde Ubuntuyu çalıştırdık sonra Mininet ve Pox kurduk.

## Neden Virtual Box?

Projemizi sanal makine üzerinden çalıştırmak istedik ve güncel ve performanslı olduğu için sanal makine uygulaması olarak Virtual Box'u seçtik.

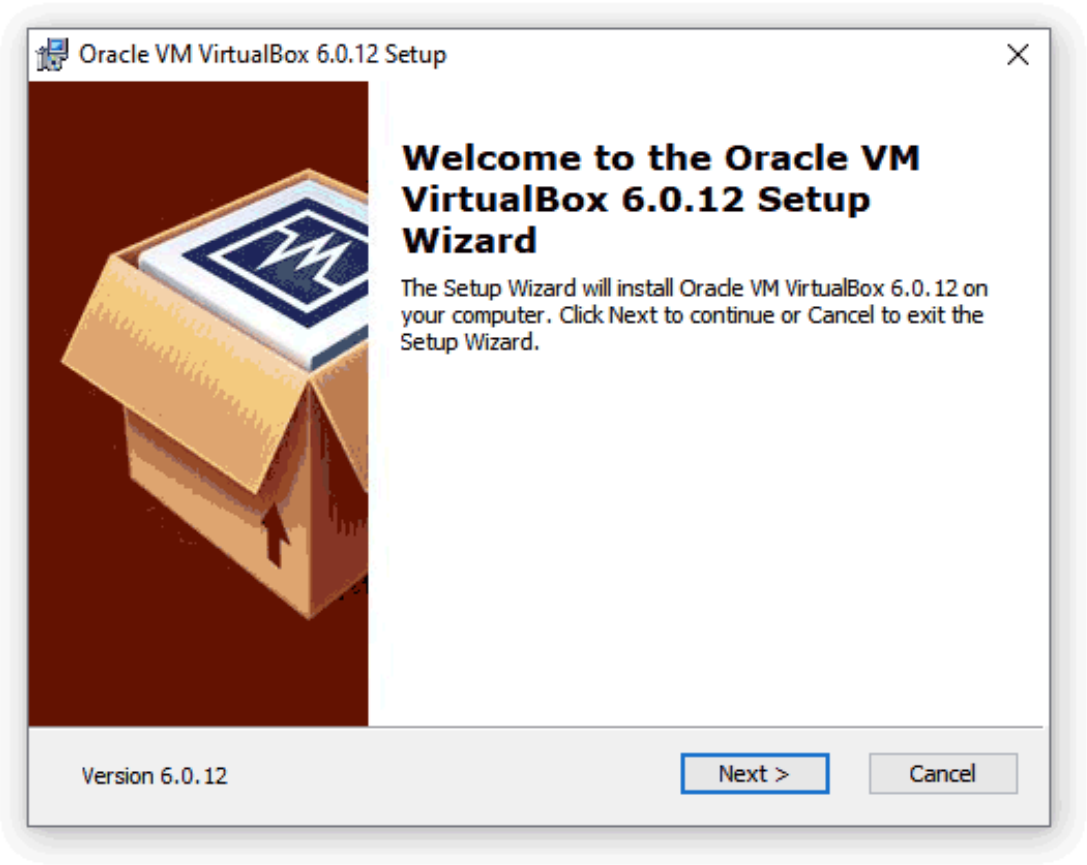
## VirtualBox Kurulumu

Her işletim sistemi ve sürümü için indirme sayfasından (https://www.virtualbox.org/wiki/Downloads) indirebileceğimiz Oracle VM VirtualBox. Windows ana için VirtualBox'ı seçiyoruz.




*Şekil 2.1*

Çift tıklama ile bir program çalıştırın (yönetici ayrıcalıklarına sahip bir kullanıcı hesabı altında olduğunuzdan emin olun) ve İleri – Evet'i seçin.




*Şekil 2.2*

## Neden Ubuntu?

Minineti çalıştırmak için linux tabanlı bir işletim sistemine ihtiyaç duyuyorduk bu yüzden basit arayüze sahip ve verimliği yüksek olan işletim sistemi Ubuntuyu seçtik

## Ubuntu Kurulumu

İlk olarak Ubuntu İSO'sunu indirin (http://www.ubuntu.com/download/desktop).

VirtualBox'ı çalıştırın.

Ubuntu VM yeni makinesine tıklayın ve Başlata tıklayın.

İndirdiğiniz Ubuntu İSO dosyasını seçin.

Ubuntu'yu kurmaya başlamak için Devama tıklayın.

İşletim sistemini kurma işlemi, gerçek bir makineye kurulumdan farklı değildir. Kurulu sistem, saat dilimi, klavye vb. İçin dili seçebilirsiniz. Yükleme sırasında bilgisayar adını, kullanıcı adını, şifreyi ve oturum açma modunu belirtmeniz gerekir.

Tamamladıktan sonra bilgisayarı yeniden başlatmalısınız.

## Neden Pyton?

Kullanacağımız programlar Python ile çalışacağı içi Python a ihtiyacımız var.

## Pyton Kurulumu

Python'u kurmadan önce console'u açarak şu kodu girin.

```
python
```

Bu koddan sonra hata alıyorsanız Python'u kurabilirsiniz. Eğer hata almıyorsanız Python zaten kuruludur.

Ubuntu için Python'u depodan kurma komutunu girdikten sonra kurulumun bitmesini bekleyin.

```
Sudo apt-get install python
```

## Neden mininet?

Projemiz için vazılım tabanlı ağ sanallaştırıcıya ihtiyaç duyuyorduk. Hızlı olmasından, özelleştirilebilir topolojiler sunmasından ve internet üzerindeki kaynak sayısı fazla olmasından dolayı Minineti seçtik

## Neden pox?

Mininette oluşturduğumuz topolojiye istediğimiz kuralları atamak için bir yazılım tabanlı ağ kontrolcüsüne ihtiyaç duyuyorduk. Mininetin içinde hazır olarak kurulu gemesinden dolayı ve aynı şekilde kaynak sayısı fazla olmasından dolayı Poxu seçtik.

## Mininet ve Pox Kurulumu

Kaynaktan yerel olarak yüklemek için önce kaynak kodunu almanız gerekiyor.

```
git clone https://github.com/mininet/mininet
```

Yukarıdaki **git** komutu en son ve en optimize Minineti kontrol edecek.

```
cd mininet
git tag  # yüklenebilir versiyonların listesi
git checkout -b mininet-2.3.0 2.3.0 # indirmek istediğiniz versiyonu yazın
cd ..
```

bundan sonra minineti indirmeye hazırız aşağıdaki komutu kullanarak indirebiliriz.

```
mininet/util/install.sh [options]
```

# DÜZENLEME ve ANALİZ

## POX Controller için yapılacak component planı

Yapılacak olan paket aktarım işleminde Pox controller'da kullanılmak üzere component kodlanacaktır. Bu kodlaması yapılacak olan componentte clientten gönderilecek olan paketin servera ulaşması beklenmektedir. Daha sonra paketin servera ulaştıktan sonra serverdan clientte ACK paketi gönderilmesi beklenmektedir. Bu gönderim sırasında daha önce Mininet üzerinden belirlediğimiz protokolde 4 adet switch bulunmaktaydı. Paketin hangi yolu (switchler üzerinden) izleyeceği SDN controller tarafından yapılmaktadır. SDN controller, clientten servera gönderilecek olan paketin ve serverdan clientte gönderilecek olan paketin hangi switch yolunu izleyeceği de belirlenmelidir.

Yapılacak olan bu paket iletişimini sağlayacak işlemlerden sonra, programı kullanan kullanıcının gönderilen paketin hangi switchler üzerinden gönderildiğini de görmesi amacıyla paket izleme fonksiyonu da kodlanmalıdır. Bu sayede hem SDN controller hangi swtichler ile bağlantı kurduğu ve clientten ya da serverdan gönderilen paketin hangi switchler üzerinden gönderildiği, paket hangi switchleri izlediği görülecektir.

Bahsedilen işlemleri yapacak olan bu kod dosyası python programlama dili tabanlı Pox component dosyası olacaktır. Kullanıcı, terminal üzerinden kodlaması yapılacak olan bu component dosyasını çalıştırıp dolayısıyla programı çalıştıracaktır.

```
asus@asus-VirtualBox:~/pox$ sudo python pox.py component
```

*Şekil 3.1*

Şekil 3.1'de yazılacak olan component dosyamızı POX controller vasıtasıyla nasıl çalıştıralacağı gösterilmektedir.

Şekil 3.1'deki terminalde belirtilen işlemleri analizleyecek olursak:

Birincil parametre olarak "sudo", superuser do anlamına gelmektedir. Bu komut sayesinde kısıtlı olan ve bu kısıtlama ile gerçekleştiremeyeceğimiz olan işlemleri gerçekleştirmeyi sağlarız.

İkincil parametre olarak belirtilen "python" pox.py ve yazıcak olduğumuz python dosyasını çalıştırmak için pythonu etkinleştirmeyi sağlar. Bu sayede bundan sonra çalıştırmak istediğimiz python dosyalarımızı çalıştırırız.

Üçüncül paramtre olarak "pox.py" python dosyası ve ismi pox.py olan dosyayı çalıştırma işlemini gerçekleştirir. Pox.py dosyası kısaca POX Controlleri çalıştırmaya ve terminalde belirticeğimiz component için hazır olmayı sağlar. Bu sayede POX Controllerin kullanması üzere yazılan component belirtilebilir. Pox.py dosyası pox ana klasörü altından boot dosyasının boot fonksiyonunu import etmektedir. POX Controller, bu fonksiyon ile başlatılır. Bu işlemden sonra yapılacak olan işlem terminalde controllerin çalıştırması üzere component veya componentler belirtilmelidir.

Dördüncü parametre olarak controllerin hangi işlemleri yapacağını belirtmesi amacıyla component dosyası belirtilir. Bu component dosyası python dosyası olsa bile ".py" olarak uzantı eklenmez. Ayrıca bu componentlerin varsayılan bir dosya konumu bulunmaktadır. Bu dosya konumu "/ext" dir. "ext" dosyası POX Controller ana klasörünün içinde bulunmaktadır. Controller için yazılacak olan her component bu dosya içerisinde yer almalıdır. Bu sayede yazılan componentler çok daha kolay ulaşılacak ve çağrılabilinecektir. Şekil 3.1'de belirtilen component dosyamız "component" isimli bir python dosyasıdır. POX Controllerin burdan sonraki yapacağı işlemler bu component isimli kod dosyasında olacaktır.

## Component kodunun analizi

POX Controllerin gerçekleştireceği işlemleri barındıran component dosyasında 2 adet import işlemi yapılacaktır.

```
from pox.core import core
import pox.openflow.libopenflow_01 as of
```

*Şekil 3.2*

Şekil 3.2'de component için 2 adet import işlemi.

"pox.core import core", pox klasöründen core dosyasından core modülünü aktarmak için kullanılmıştır. Core modülü, POX Controlleri kullanarak bir SDN denetleyicisi oluşturmak amacıyla bir dizi core sınıf ve işlev sağlar. Core modülü ile akış tabloları oluşturulabilir, yönetilebilir ve SDN ağındaki switchler ve diğer ağ cihazlarıyla etkileşim kurulabilir.

"pox.openflow.libopenflow_01 as of", pox altından openflow klasöründen libopenflow_01 dosyasını of adı olarak import eder. Bu import dosyası aracılığı ile switchler arası paket aktarım olduğunu anlayabilecek ve paketleri istediğimiz switchin portuna yönlendirebileceğiz. Kısacası import edilen "of" dosyası aracılığı ile kişisel paket akış şemamızı oluşturabiliriz. Ayrıca paketin her aktarım işlemini algılayabiliceğimizden paket hangi swtichde olduğunu da anlayabileceğiz.

Bu dosyaları import ettikten sonra component için kişisel paket akış şeması tasarlanabilir.

```
def launch():
    core.addListenerByName("UpEvent", _handle_UpEvent)
    core.openflow.addListenerByName("ConnectionUp", start_switch)
    core.openflow.addListenerByName("PacketIn", _handle_PacketIn)
```

*Şekil 3.3*

Şekil 3.3'de componentin çalışma sırasındaki işlemleri yer almaktadır.

Şekil 3.1'de belirtilen şekilde POX Controlleri yapılan component ile çalıştırıldığında ilk işlem olarak componentin launch() methodu çalıştırılacaktır. Şekil 3'te görülen "launch()" methoduna bakıldığında "addListener" fonksiyonel işlemleri yer almaktadır. "addListenerByName" methodu 2 adet parametre almaktadır. İlk parametre belirlenecek olan fonksiyonun (2. paramtere) ne zaman çalıştıralacağıdır. İkinci parametre çalışacak fonksiyonu belirtir. Bu methodlar bir kere oluşturulduğunda program çalıştığı sürece (iptal methodu kullanılmadıkça) çalışmaya devam eder.

Componentte 3 adet addListenerByName methodu kullanılmıştır. Bu methodlar program çalıştırıldıktan itibaren çalışmaya devam edecektir. Bu methodlar dinleyici method olarak belirtilebilir. İlk dinleyici olan "UpEvent" programın ilk çalıştığı sırada çalışacaktır. Bu dinleyici POX Controllerin çalışmasına bakar ve çalıştığı ilk durumda "_handle_UpEvent" methodunu çalıştırır. İkinci dinleyici olan "ConnectionUp" POX Controllerin switchlere bağlanmasını takip eder. Bu dinleyici, POX Controllerin switchlere bağlanmasını takip eder. POX Controller switchlere bağlandığı an belirtilen "start_switch" methodu çalışır. Üçüncü dinleyici "PacketIn" programın en önemli dinleyicilerinden biridir. Bu dinleyici, SDN ağındaki switchlere paket mesajı gönderildiğinde _handle_PacketIn methodunu başlatır. Bu dinleyici sayesinde aktarılan bir paketin hangi switchlerdeyse ona göre bir aktarım tablosu oluşturabiliriz. Bu sayede paketin yolunu kişisel olarak belirlenebilecektir.

```
def _handle_PacketIn(event):
    packet = event.parsed
    packet_in = event.ofp
    switch_id = event.dpid
    do_switch_router(event, packet, packet_in, switch_id)
```

*Şekil 3.4*

Şekil 3.4'te belirtilen görüntüde, "packetIn" dinleyicisi ile tetiklenen _handle_PacketIn methodu yer almaktadır.

_handle_PacketIn methodunda standart parametresi olan event parametresi kullanılmaktadır. Paketi belirten bu parametre ile paket ile ilgili işlemler yapılabilmektedir. Şekil 4'de görülen _handle_PacketIn methodunu analiz edildiğinde, event.parsed fonksiyonu switchler ile alınan ve POX Controllere bir paket mesajı olarak gönderilen paketi belirtir ve bunu packet değişkenine atar "event.ofp" switchde bulunan paket mesajının birçok özelliğini belirtir ve bunu packet_in değişkenine atar. "event.dpid" paketin olduğu switchin özel olarak atanmış olan numarasını belirtir ve bunu switch_id değişkenine atar. Daha sonra tüm belirleme işlemleri tamamlandıktan sonra do_switch_router isimli methodu çalıştırır. Bu methodun paramterlerine ise daha önce belirlenen packeti packet_in, switch_id ve ek olarak event eklenir.

```
def do_switch_router(event, packet, packet_in, switch_id):
    if((str(packet.payload)).find("ARP")==-1):
        print("------------------------------------------------------------------------")
        print(" Kullanilan switch numarasi: ", switch_id, "\n Paketi gonderen kaynagin MAC adresi > Hedefin MAC
        adresi: \n", packet.src, ">", packet.dst,  "\n Paketi gonderen kaynagin IP adresi > Hedefin IP adresi:
        \n",  packet.payload.srcip , ">" , packet.payload.dstip)
        print("------------------------------------------------------------------------")
```

*Şekil 3.5*

Şekil 3.5'te görülen do_switch_router isimli methodun ilk kısmı.

Belirlenen "do_switch_router" isimli fonksiyonda ilk olarak şuanki paketin olduğu switch başta olmak üzere izlemesi yapılmak üzere bir komut tasarlanmıştır. Yapılan paket izlemesinin sonucunu ekrana (terminal ekranında) yazdırılır. Belirlenen paket izlemesi ile kullanıcının görebilecekleri şunlardır.

- Paketin olduğu switchin id (dpid) numarası
- Paketi gönderen kaynağın (source) MAC adresi
- Paketin ulaşması beklenen hedefin (destination) MAC adresi
- Paketi gönderen kaynağın (source) IP adresi
- Paketin ulaşması beklenen hedefin (destination) IP adresi

Bu belirtilen özellikler her paket aktarım işleminde görülebilecektir. Ancak bu işlem her paket iletişiminde gerçekleştiği için kullanıcının görmek istemediği paket iletişimini de görecektir. Bunu engellemek amacıyla bir filtreleme sistemi kodlanmalıdır. Bunun için "packet.paylod" fonksiyonu kullanılır. Bu fonksiyon ile packet bilgilerine ulaşılır (ekrana yazdırmak istenilen bilgiler de dahildir). "str(packet.paylod).find("ARP")" methodu ile "packet.paylod" sonucu string elementine çevrilip bu string içinde "ARP" kelimesini içerdiği aranır. Eğer str(packet.paylod) içerisinde aranan kelime varsa o paketin izlenmesinde kullanılan bilgiler ekrana yazılmaz. "ARP" kelimesini engelleme sebebimiz Adres Çözümleme Protokolü olan ARP, kullanıcının kendi isteği dışında paket gönderimi yapmaktadır. Her paket gönderiminde "packetIn" dinleyicisi çalışacağı dolayısıyla paket izlenmesi için bilgiler de ekranda

gözükücektir. Bunu engellemek için “ARP” kelimesi packet.paylod içerisinden filtrelenir. Bu sayede sadece kullanıcının göndermek istediği paketin izlenmesindeki bilgiler görüntülenecektir.

```
if switch_id == 1:
    if packet.src == "00:00:00:00:11:00":
        msg = of.ofp_flow_mod()
        msg.data = event.ofp
        msg.match = of.ofp_match.from_packet(packet)
        msg.idle_timeout = 30
        msg.hard_timeout = 60
        msg.actions.append(of.ofp_action_output(port=  2))
        event.connection.send(msg)
```

*Şekil 3.6*

Şekil 3.6’da paket yönlendirmesi yapan kod kümesi görülmektedir. (do_switch_router isimli methodun sonraki kısmı).

Gelen paketi yönlendirmek için ilk önce paket hangi switchde olduğu belirlenmelidir. Paket hangi switchdeyse o switche özel olarak bir yönlendirme yapılır. Her switchin kendisine özel portları vardır. Bu portlar paket iletişimini sağlayan yollardır. Paket bu portlara gönderilir. Bazı portlar uç sistemlere bazı portlar başka bir switchlere bağlıdır. Paketin hangi yolu izleyeceğini yani gelen paket hangi porttan gönderileceğini SDN Controller belirler.

Paketin portlardan akışını sağlayan kod örneği Şekil 6’da belirtilmiştir. Bu koda bakıldığında ilk olarak paketin hangi swtichde olduğu belirlenmektedir. Bu koda göre paketin olduğu switchin switch_id numarası (dpid) “1” ise paket yönlendirmesine giriş yapılmaktadır. Eğer paketin olduğu switch numarası 1 olduğu varsayıldığında ve if şartı sağlandığında, paket hangi kaynaktan geldiği tespit edilmelidir. Paketin hangi uç sistemden geldiği MAC adreslerine göre tespit edilmektir. MAC adresi bir sistemin kimliğini tanıtan ve o sisteme özel olarak belirtilmiş benzersiz ID’sidir. Yapılan ağ protokolünde 2 adet uç sistem vardır. Bunlardan birisi client birisi server olarak varsayılmıştır. Client MAC adresi “00:00:00:00:11:00” ve server MAC adresi “00:00:00:00:22:00” olarak belirlenmiştir. Paketin switchlerdeki portlara yönlendirmesinde bu MAC adresleri kullanılmaktadır. Eğer paketin kaynak MAC adresi (source) “00:00:00:00:11:00” ise paketin clientten servera doğru bir aktarımı söz konusudur. Eğer paketin kaynak MAX adresi “00:00:00:00:22:00” ise paketin serverdan cliente doğru bir aktarımı söz konusudur. Paketin Şekil 3.6’da belirtildiği gibi clientten servera, yani MAC adresi 00:00:00:00:11:00 olan kaynaktan gönderim yapıldığını varsayalım. O halde şekil 3.6’da belirtilen if şartı sağlanacaktır.

Bu işlemlerden sonra paketin port yönlendirme işlemleri başlatılır. Yapılan işlemler analiz edildiğinde; paketin portlardan akışını yolunu kişiselleştirmeyi başlatan temel fonksiyon “ofp_flow_mod()” yazılmıştır. Fonksiyonun dönderdiği değer “msg” değişkenine yazdırılmıştır ve bundan sonraki her işlem oluşturulan bu “msg” üzerinden yapılacaktır. Bunun sebebi yazılan “ofp_flow_mod()” fonksiyonu paket akış yönü değiştimek için ayarlanacak olan değerleri verdiğinden akış yolundaki tüm değiştirmeler bu değişken üzerinden yapılacaktır. Daha sonra oluşturulan bu “msg” değişkenine switchde bulunan paketin özelliklerini eklemek için “msg.data = event.ofp” kodu kullanılır. Buradaki event, içerisinde switchde bulunan paketin birçok özelliğini içermektedir. “event.ofp” ile bu paketin türünü, mesajın uzunluğunu, içerdiği veriler gibi birçok özellikler alınır ve msg.data içerinde tutlur. Daha sora “ofp_match_from_packet(packet)” kodu ile kurulan akışda hangi paketin eşleşeceğini belirtir. Paketin zaman aşım süreleri belirtilir. Bu zaman aşımlarını belirtmek için “idle_timeout” ve “hard_timeout” kullanılır. “idle_timeout” switchin akış şemasında kaldırılmadan önce devre dışı kalabiliceği maksimum süreyi belirtmektedir. Burdaki süre, eğer bir akış aktif değlse ve belirli bir süre boyunca harhangi bir paketle eşleşmiyorsa bu akış, akış tablosundan kaldırılacağı süreyi belirtir. “hard_timeout” akışın

herhangi bir paketle eşleşip eşleşmediğine bakmaz ve belirlenen süre sonra akışı kaldırır. Bu her iki zaman aşımını belirten fonksiyonlar saniye türünden belirtilir. Daha sonra paket, switchin hangi portundan gönderileceği belirtilir. Şekil 3.6'da görülen örnek yönlendirmede "msg.actions.append(of.ofp_action_output(port=2))" ile paketin, switchin 2 numaralı ID'sine sahip portuna iletimi sağlanır. Burda belirtilen "2" numaralı port her switch için, istenilen akış yolları için değişiklik gösterebilir. Bütün özellikleri ve paketin hangi porttan gönderileceği belirtilen paketi içeren "msg" değişkeni, "event.connection.send(msg)" ile işleme sokulur. Bu fonksyion ile oluşturulan mesaj SDN Controller ile iletişimini kurup paketi harekete geçirir ve paketin switch arasındaki yönlendirmesini sağlar. Sonuç olarak paket, istenilen akış yolu ile gönderilmiş olur.

# ÇALIŞTIRMA ve SONUÇ

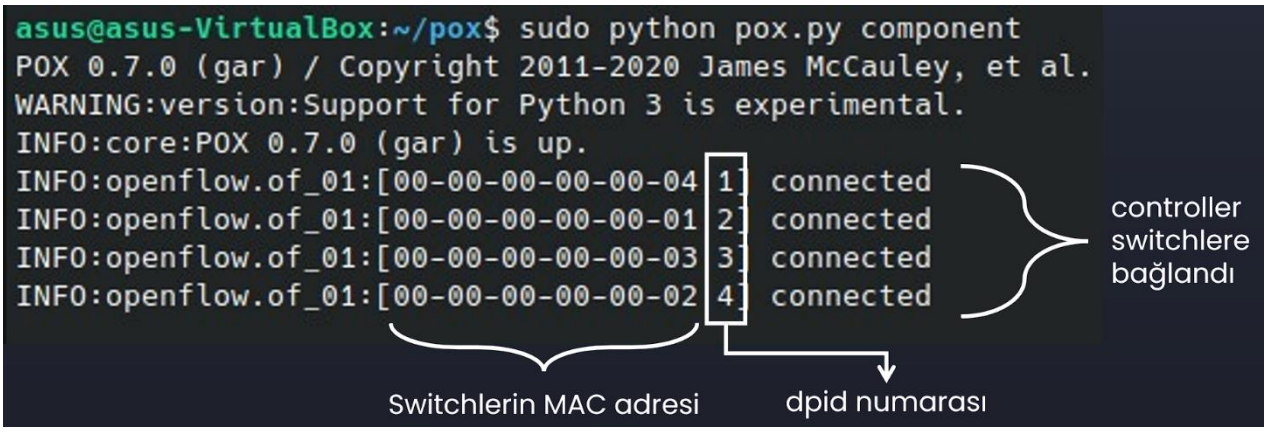
İlk önce iki konsol uygulaması açıp birinde minineti diğerinde poxu çalıştırıyoruz.

Sonra minineti çalıştırığımız konsolda şekil 4.1 deki komutu yazıyoruz ve network.py dosyamızı mininet üzerinde açmış oluyoruz .

```
asus@asus-VirtualBox:~/mininet/custom$ sudo python network.py
Unable to contact the remote controller at 127.0.0.1:6633
***Creating hosts
***Pairing links
*** Configuring hosts
h1 h2
*** Starting CLI:
```

*Şekil 4.1*

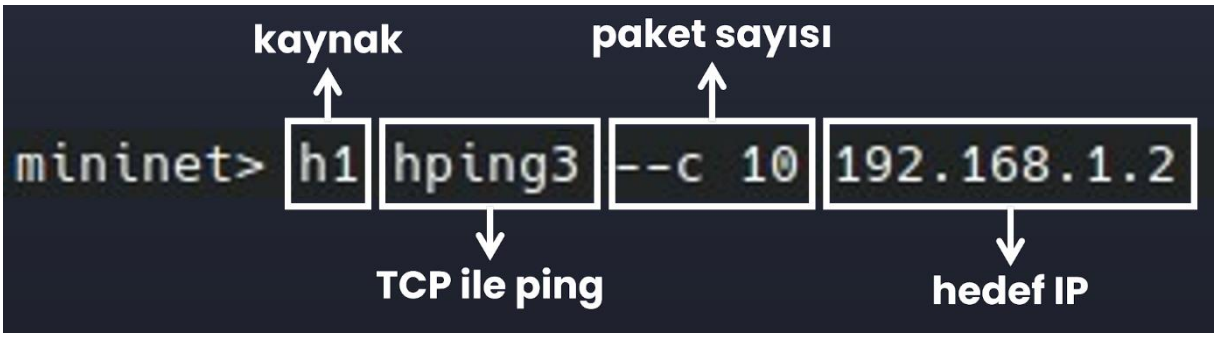
Çalıştıktan sonra hostlarımız h1 ve h2 oluşturuluyot linkleri bağlıyor ve hostları configüre ediyor.




*Şekil 4.2*

Bu işlemden sonra poxun olduğu konsola şekil 4.2 deki komutu giriyoruz ve pox üzerinde component dosyamızı çalıştırıyoruz ve switchlerimizzin bağlandığını görüyoruz.

Bundan sonra ise mininetin açlılı olduğu konsola gidip 4.3 deki komutu giriyoruz.




*Şekil 4.3*

Bu komut h1 hostundan 192.168.1.2 ip adresine sahip hosta TCP ile 10 adet ping göndermemizi sağlıyor.


*Şekil 4.4*

sonuç olark elimizde şekil 4.4 deki gibi bir çıktı oluyor ve burda paketin izlediği yolu görmüş oluyoruz.

# KAYNAKÇA


http://www.volkantay.com/2020/11/21/ag-topolojileri/. (tarih yok).

https://dogus.com.tr/yazilim-tabanli-ag-sdn-nedir/#:~:text=Yaz%C4%B1l%C4%B1m%20tabanl%C4%B1%20A%C4%9F%3A%20Network'%C3%BCn, k. (tarih yok).

(tarih yok). *https://en.wikipedia.org/wiki/Request%E2%80%93response.*

https://mertmekatronik.com/mac-%C4%B1p-nedir. (tarih yok).

https://mertmekatronik.com/mac-%C4%B1p-nedir#:~:text=Mac%20ve%20IP%20adresleri%2C%20a%C4%9Fa, b. (tarih yok).

https://tr.wikipedia.org/wiki/%C4%B0stemci-sunucu. (tarih yok).

https://tr.wikipedia.org/wiki/A%C4%9F_anahtar%C4%B1. (tarih yok).

https://tr.wikipedia.org/wiki/TCP. (tarih yok).

https://www.beyaz.net/tr/network/makaleler/sdn_yazilim_tanimli_aglar.html. (tarih yok).

https://www.bilisimhocasi.com/mac-adresi-nedir-mac-adresi-nasil-ogrenilir. (tarih yok).

https://www.hosting.com.tr/bilgi-bankasi/sdn-nedir/. (tarih yok).

https://www.karel.com.tr/bilgi/tcp-ve-udp-arasindaki-farklar-nedir. (tarih yok).

https://www.techtarget.com/searchnetworking/definition/SDN-controller-software-defined-networking-controller. (tarih yok).