# Bilgisayar Ağları PROJE RAPORU

# A-YAZILIM TANIMI

**Yazılımcıların Adı-Soyadı:** Aziz Eren Sağanda – 170420048, Emirhan Ese – 170420016, Melih Afşar – 170420034, Kenan Baylan-171420005

* ***Geliştirme Ortamları***

***Programlama Dili:***Program Python dili ve Dockerfile kullanılarak yazılmıştır.

***Programın Adı:*** SDN Demo

***İşletim Sistemi****:* Program yazılımcıların kendi bilgisayarlarındaki Windows 11 (64 bit) işletim sisteminin içinde VMware Workstation Pro Hypervisor’u kullanılarak Ubuntu Server 20.04 LTS sürümünde yazılmıştır.

# B- YAZILIM TASARIMI

## Problemin Tanımlanması / Programın Amacı

Büyük bir ağ tasarlanırken önceden olası durumları deneyimleyebilmek önemli bir avantajdır. Biz de projemizde Docker ve Containernet teknolojilerinden yararlanarak bir ağın sanallaştırılması ve yönetilmesi senaryolarını göstermeyi amaçladık.

## Problemin Çözümü / Çözüm Tasarımı

**Hostların Konteynırize Edilmesi:**

* ***Dockerfile.client***

Text

Description automatically generated

Client konteynırımızın oluşturulacağı Dockerfile’ı yazarken base image olarak Ubuntu focal versiyonunu seçtim. Docker hub’dan çekilen bu base imaj saf Ubuntu imajı olup ekstradan herhangi bir program veya desktop environment’i içermemektedir ve takriben 190 MB yer kaplamaktadır. Gerekli programların yüklenmesi için RUN tagini kullanarak apt paket yöneticisiyle ihtiyaç duyduğumuz programların paket yöneticisindeki ismini kullanabiliriz. Dockerfile’ımızı build ettikten sonra Image’ımız hazır olur. Artık oluşturduğumuz imajı ana programımızda client rolünde kullanabiliriz.

* ***Dockerfile.server***

Text

Description automatically generated

Server konteynırımızın oluşturulacağı Dockerfile’ı yazarken base image olarak Python 3.10-slim versiyonunu seçtim. Bu imajın Python’u çalıştırmak için base paketleri içeren bir sürüm olması ve küçük boyutlu olması tercih sebebim olmuştur. Gerekli programların yüklenmesi için RUN tagini kullanarak apt paket yöneticisini kullandık. Python Pip’in Docker’deki varsayılan root kullanıcısının dosya yolunu kullanmaması için Virtual Environment oluşturduk. Ardından server konteynırımızda web server çalışması için Flask’ın yüklenme adımına geçtik. “requirements.txt” dosyasındaki gereksinimlerin pip install komutuyla yüklenmesini sağladık. Son olarak konteynırıın 80 portunun açılması için Expose tagini ekleyerek dosyayı sonlandırdık. Ardından docker build komutuyla Dockerfile’ı build edip yeni imajımızı oluşturmuş olduk. Bu işlemlerin ardından docker run komutuyla konteynırı başlatabiliriz.

* ***app.py***

Text

Description automatically generated

Flask server’in çalışması için gerekli parametreler ve opsiyonlar app.py dosyasında bulunmaktadır. Server wildcard mask olan 0.0.0.0’da bütün gelen bağlantıları 80 portunda dinlemektedir. İstek atılması durumunda “hello world” mesajı döndürülmektedir.

* **Main programımız**

Text

Description automatically generated

ContainerTreeTopo fonksiyonunda ağaç veri yapısı kullanılarak ağaca node’lar halinde host server ve client’lar eklenmiştir. Fanout parametresiyle ağacın kaç tane yaprak node’u olacağını seçebiliriz. Yeni node’ların rollerini elle belirlemek modüler aritmetikten faydalanarak nodeların sıra sayısının 3’ün bölümünden kalanlarına göre dağıttık. Örnek vermek gerekirse ilk node client, sonraki host, en son olarak kalan 2 olduğundan son node’umuz server rolü üstlenecektir. En başta varsayılan olarak 1 adet Open vSwitch (owswitch) eklenir. Owswitch Linux kernelinden güç alarak bize sanal olarak trunk ve access mode destekli 802.1Q Vlan’lar oluşturmamızı sağlar. Bu sayede SDN’imiz içinde switching yapmamız mümkün olur.

* **Ağa NAT özelliğinin eklenmesi**

Text

Description automatically generated

Network Address Translation (NAT) temel olarak genel ağdaki IPv4 adreslerinin tükenmesine karşı geliştirişmiş bir çözüm olup genel IP adreslerini lokal ağdaki non-routable bloklarda bulunan IP adreslerine çevirir. Bu durum aynı zamanda yerel ağlarımızın WAN’dan (İnternet) izole edilerek bağlantının filtrelenmesini sağlar. Bizim konteynırlarımızın internete bağlanmasını sağlamak amacıyla NAT controller’ı ekledik ve 10.0.0.0/8 bloğundan IP alan konteynırlarımızın genel IP’miz üzerinden internet sitelerine ulaşabilmesini sağladık.

* **Programın Çalıştırılması**

Text

Description automatically generated

Programı sudo python3 komutuyla başlatabililriz.

Text

Description automatically generated

*pingall* komutuyla ağdaki tüm node’ların birbirini pinglemesini sağlayarak bağlantıyı test edebiliriz.

Text

Description automatically generated

*dump* komutuyla ağdaki elemanların bilgilerini görüntüleyebiliriz.

Text

Description automatically generated

Xterm komutuyla sanal ağımızda bir terminal oluşturabiliriz.

Text

Description automatically generated

Konteynırlarımızdan dış IP adreslerini pingleyebiliyoruz.

Text

Description automatically generated

Aynı zamanda WAN adreslerinin DNS çözümlemesi de yapılabilmekte.

Text

Description automatically generated

*docker ps* komutuyla Docker konteynırlarımızın çalıştığını görebiliriz.

Text

Description automatically generated

*Ubuntu* server sanal makinemizde oluşturulan sanal ağ adaptörlerini ifconfig komutuyla görüntüleyebiliriz.

Graphical user interface, text, application

Description automatically generated

Sanal ağ adaptörümüz üzerinden geçen trafiği wireshark’tan dinleyebiliriz.

Text

Description automatically generated

*curl* ile webserver’imize bağlanalım.

Text

Description automatically generated

Bağlantının kurulması için geçen süreyi time komutuyla öğrenebiliriz.

Graphical user interface, application, table

Description automatically generated

Paketimizi Wireshark’ta da görüntüleyebiliriz.

Text

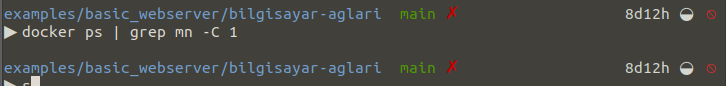
Description automatically generated

*quit* komutuyla containernet CLI’dan çıkabiliriz.

*Text

Description automatically generated*

Tekrar *ifconfig* komutunu girdiğimizde sanal ağ adaptörlerinin gittiğini görebiliriz.



*docker ps* komutunu girdiğimizde konteynırların kapanmış olduğunu da teyit edebiliriz.

**GitHub Linki:**

<https://github.com/aesaganda/Bilgisayar-Aglari>

**Docker Hub Linki (*Konteynır imajları için):***

<https://hub.docker.com/repository/docker/aesaganda/aesaganda>