

**KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ**

**MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**BİTİRME PROJESİ TEZİ**

**YAZILIM TANIMLI AĞLAR MİMARİSİ**

**18110131032 – 18110131035**

**KÜBRA KABALCI – BURCU GENÇ**

**KSÜ Rektör Yard. Prof. Dr.**

**İBRAHİM TANER OKUMUŞ**

**OCAK 2022**

**ÖZET**

Yazılım tanımlı ağlar bir bilgisayar mimarisi olup, ağ kontrol düzleminin çeşitli aygıtlarını denetlemesini sağlar. Yazılım tanımlı ağların amacı, ağ erişiminin, halihazırda hızlı olan depolama ve sanal sunucular gibi hızlı olmasını sağlamak. Sanal makine üzerinden Floodlight VM kuruldu ve çalıştırıldı.[2] OpenFlow, bir iletişim protokolü ve ağ kontrol ve yönlendirme işlevlerini ayıran yazılım tanımlı ağ (SDN-Software Defined Networking) mimarisinin temel öğesidir. Openflow ağına bağlanabilmek için simülasyon aracı olarak minineti kurmuş olduk. Mininet üzerinden Controller fonksiyonuna uzaktan ip ile erişim sağlanır. WireShark kurulumu gerçekleştirildi. OpenFlow protokolün üzerinden paket yakalayıcılarına, akış tablosuna, Openflow table üzerinde hangi akış nereye yönlendiriliceğine, işlem hattı boyunca paket akışına, port çeşitlerine, aksiyon setlerine, sayaç listelerine ve meter table üzerinde bir sayaç girişindeki bir sayaç bandının sayaç bileşenlerine bakılmıştır. İleriki aşamalarda Network’te farklı uygulamaları tanıyarak farklı hizmetler vereceğiz. Hizmet kalitesi sınıflarına yazılım tanımlı ağlarda controller üzerinde Meter ve Queues ile çalışmalar gerçekleştirilecektir.

|  |  |  |
| --- | --- | --- |
| Anahtar Kelimeler | : | Floodlight VM, Openflow, Mininet, Controller, Wireshark, port, Meter table, Flow, Flow Table, Sayaçlar, akış tablosu, Queues, Network |
| Sayfa Adedi | : | 46 |
| Danışman | : | İbrahim Taner Okumuş |

**TEŞEKKÜRLER**

Tez çalışmamızda bu konuya yönlendiren, karşılaştığımız zorluklara karşı bilgi ve tecrübesi ile yardımcı olan değerli danışman hocam KSÜ Rektör Yardımcısı Prof. Dr. İbrahim Taner Okumuş’a teşekkürlerimizi sunarız.

**İÇİNDEKİLER**

**Sayfa**

ÖZET i

TEŞEKKÜR ii

İÇİNDEKİLER iii

1. GİRİŞ 1

2. LİTERATÜR TARAMASI 2

3. MATERYAL VE METOT 3

3.1. Ağ(Network) Nedir? 3

3.1.1. Local Area Network (LAN) 3

3.1.2.Wide Area Network (WAN)……………………………………………

3.1.3 Metropolian Area Network (MAN)……………………………………. 3

3.2. Ağ Protokolleri 3

4. OSI REFERANS MODELİ ……..……………………………………………….. 4

4.1- Fiziksel Katman .4

4.2. Veri Bağlantı Katmanı .… ……………………………………………………….. .5

4.3. Ağ Katmanı………………………………………………………………………….5

4.4. Taşıma Katmanı……………………………………………………………………5

4.5. Oturum Katmanı……………………………………………………………………6

4.6. Sunum Katmanı………………………………………………………………………6.

4.7. Uygulama Katmanı…………………………………………………………………6.

5. TCP/IP REFERANS MODELİ 7

5.1. Uygulama katmanı………………………………………………………………….7

5.2 Taşıma katmanı………………………………………………………………………..7

5.3. Ağ katmanı………………………………………………………………………….7

5.4. Fiziksel katman……………………………………………………………………..8

6- YAZILIM TABANLI AĞLAR (SDN) 8

7- OPENFLOW.................................................................................................................9

7.1. Openflow Nedir?........................................................................................................9..

7.2. Openflow Mimarisi...................................................................................................9..

7.3. Openflow Anahtarı.....................................................................................................9

7.3.1. Kontrol Katmanı...............................................................................................10

7.3.2. Veri Katmanı................................................................................................. .10

7.3.3.Yönetim Katmanı........................................................................................... .10

8- FLOW VE FLOW TABLE

8.1. Flow.......................................................................................................................11

8.2. Flow Table.............................................................................................................11

9- OPENFLOW SWİTCHES............................................................................................12

10- OPENFLOW TABLES................................................................................................12

10.1. Openflow Anahtarı Aracılığıyla Paket Süreci……………………………………14

11- FLOODLİGHT VM…………………………………………………………… ……15

12- MİNİNET ....................................................................................................................15

13- AĞ SİMÜLASYONU…………………………………………………………………16

13.1 Mininet Ortamının Temel Komutları ve İşlevleri ………………………………17

14- ÖZEL TOPOLOJİ OLUŞTURMA ……………………………………………………18

14.1. Python Scripti Parametreleri ……………………………………………………19

15- İPERF ARACI………………………………………………………………………..21

15.1. İperf Aracı Parametreleri…………………………………………………………21

16- HİZMET KALİTESİ QOS…………………………………………………………….23

17-OVS İLE QUEUE YAPILANDIRILMASI………………………………………..….24

17.1. Hangi Anahtarlar Kuyrukları Destekler?.............................................................24

17.2. OVS’ de Kuyrukları Yapılandırma…………………………………………..….24

18-OPEN VSWİTCH NEDİR?........................................................................................25

18.1. OVS ‘de Temel Komutlar………………………………………………………….26

18.1.1 Ovs-Vstcl…………………………………………………………………….26

18.1.2 Ovs-Ofctl…………………………………………………………………….27

* + 1. Ovs-Dpctl……………………………………………………………………27

19- KUYRUKLAR ARASI TRAFİK AKIŞI……………………………………………..28

19.1. Switchte Kuyruk Yapılandırma………………………………………………….28

**20. Meter ………………………………………………………………………………….29**

21.DPCTL komutları………………………………………………………………………30

21.1.Meter komutları………………………………………………………………………30

SONUÇ VE BULGULAR 32

KAYNAKLAR 37

1. **GİRİŞ**

Yazılım tanımlı ağlar, bir bilgisayar ağ mimarisi olup, ağ düzleminin çeşitli aygıtlarının denetlenmesini sağlar. Yüksek bant genişliği, uygun maliyetli, dinamik, uyarlanabilir yönetilebilir yapıdadır. Doğrudan programlanabilir kontrol ve yönlendirme işlevlerini birbirinden ayırır. Çözümlemek için ise Openflow protokolü bir gereklilik söz konusudur. Amaç ağ yapısının hızlı erişilebilir, maliyeti en aza indirgenebilir, ağ protoklünü doğrudan programlanabilir, ağ yapısını yönetebilir ve güvenli hale getirmeyi amaçlar. Openflow protokolü, yazılım tanımlı ağlarda, ağ yöneticisine ağın tapolojisi ve ağda yer alan paketleri filtreleme değişimlerinde yönlendirme işlemlerini sağlar. Bu protokolün içerisinde bulunan en önemli bileşenler, controller ile OpenFlow ağ anahtarlarıdır. OpenFlow protokolü, sistemdeki eksikliklere karşı yöntemler bulmayı hedefler.

Yazılım tanımlı ağların getirisi, cihazların davranışlarını birbiri ile olan bağlantılarının sağlanmasının hızlı ve güvenilir bir şekilde kontrol edilebilmesini, gelecekte oluşabilecek olan problemlere hızlı alternatif çözümler bulabilmek, işlem gücüne ve bant genişliğine ihtiyaç duyulması dahilinde hizmet verilebileceği böylelikle yönetim süreçleri hızlanacak maliyetlerde azalmaları sağlayacaktır.

Günümüzdeki şirketler ağ yapısının maliyet, performans gücüne, hız ve güvenliğine dikkate alırlar. Piyasada birçok ürün hizmet olması ağ yönetimi konusunda şirketler birbirleriyle uyum içerisinde ilerlemelidirler. Bu ilerlemeyi sağlamak için performansı yüksek tutarak, güvenlik sorunu yaşamadan sağlamayı yazılım tanımlı ağlarda sağlamak mümkündür.[3]

1. **LİTERATÜR TARAMASI**

OpenFlow Switch Specification[1] yayını üzerinden ilgili bölümler incelenmiştir. Yapılan inceleme sonuçları edinilen bilgi ilerleyen sayfalarda paylaşılacaktır.

Ali Pak’ın hazırlamış olduğu hizmet kalitesi tezinden yaralandık.

**3- MATERYAL VE METOT**

* 1. **- Ağ (Network) Kullanımı**

Bir veya birden fazla bilgisayarın; dosya ve veri alış verişi yapabilmesi veya haberleşmeleri için birbirine bağlanarak oluşturduğu yapılara network (ağ) denir. Ağ topolojisi üç başlıkta incelenir: LAN(Local Area Network), WAN(Wide Area Network) ve MAN(Metropolian Area Network) olmak üzere yapılarına göre standartları vardır:

**3.1.1. Local Area Network (LAN):** Birden fazla bilgisayarın oluşturmuş olduğu en küçük bilgisayar ağlarına LAN adı verilir. Günümüzdeki en hızlı ağ yapısıdır ve yaygın olarak Ethernet Protokolü kullanılmaktadır. İçerisinde bir veya birden fazla Ethernet anahtarlama (switch) cihazı bulundururlar. LAN üzerindeki bilgisayarlar birbirlerine coğrafi olarak yakındır ve 10, 100, 1000 ve 10000 Mbit hız ile haberleşmektedirler.

**3.1.2.Wide Area Network (WAN):** Hız olarak LAN ağ yapısına göre yavaş olmasına rağmen, en az iki LAN’ın yönlendirici (router) ile birleşmesi sonucu kurulan ağlara WAN denir. Şehirlerarası ve uzak mesafeler için tasarlanmış bir ağ yapısıdır.

**3.1.3 Metropolian Area Network (MAN):** Aynışehir içinde bakır veya Fiber Optik altyapılar ile oluşturulan ağlara verilen isimdir. Günümüzde yaygın olarak MetroEthernet, DSL ve Kablonet teknolojileri bu amaçla kullanılmaktadır

* 1. **Ağ Protokolleri**

Ağ protokolleri, ağ üzerindeki iki veya daha fazla cihaz arasındaki iletişimi tanımlayan kurallar, prosedürler ve formatlardan oluşan resmi standartlardır. Ağ protokolleri, ağ iletişiminin uçtan uca olan işleyişini ve verilerin zamanında ve güvenli bir şekilde iletilmesi süreçlerini yönetmektedir. Ağ protokolleri, bilgisayarlar,sunucular,yönlendiriciler ve diğer ağ özellikli cihazlar arasındaki iletişimi başlatmak ve gerçekleştirmek için gereken tüm süreçleri, gereksinimleri ve kısıtlamaları içermektedir. Verilerin ağ üzerinde güvenilir bir şekilde kullanılması için protokollere ihtiyaç vardır. Protokoller OSI Referans modeli ve TCP/IP referans modeli olmak üzere iki farklı şekilde dağılım göstermişlerdir.

**4- OSI REFERANS MODELİ**

OSI (Open Systems Interconnection) modelini ISO (International Organization for Standardization) geliştirmiştir. Amaç iki bilgisayar arasındaki iletişimin nasıl olacağını tanımlamaktır. 1978 yılında ilk defa ortaya çıkarılan bu standard 1984 yılında yeni bir düzenleme yardımıyla OSI (Open System Interconnect) referans modeli olarak yayınlanmıştır. OSI öncesindeki dönemde, yalnızca bilgisayar donanımı üreten kuruluşlara özgü ağlar vardı. Bu ağların özellikleri, çoğunlukla yalnızca o üreticinin donanımının bağlanmasına izin verecek biçimde tanımlanmıştı. Onlardan ayrı olarak OSI, çeşitli üreticilerin ürünlerinin bağlanabileceği bir ağ için, bir sektör etkinliği olarak ortaya çıkmıştır. OSI Modeli herhangi bir donanım ya da bilgisayar ağı tipine göre değişiklik göstermemektedir. OSI'nin amacı ağ mimarilerinin ve protokollerinin bir ağ ürünü bileşeni gibi kullanılmasını sağlamaktır. OSI modeli 7 katmana ayrılmıştır.

((<https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/07/osi-katmanlar%C4%B1)>)

OSİ MODEL

|  |  |
| --- | --- |
| Layer 7 | Application (Uygulama katmanı) |
| Layer 6 | Presentation (Sunum Katmanı) |
| Layer 5 | Session (Oturum Katmanı) |
| Layer 4 | Transport (Taşıma Katmanı) |
| Layer 3 | Network (Ağ Katmanı) |
| Layer 2 | Data Link (Veri Bağlantı Katmanı) |
| Layer 1 | Physical (Fziksel Katman) |

**4.1. Fiziksel Katman**

Fiziksel katman verinin kablo üzerinde alacağı yapıyı tanımlar. Veriler bit olarak iletilir. Bu katman bir ve sıfırların nasıl elektrik, ışık veya radyo sinyallerine çevrileceğini ve aktarılacağını tanımlar. Gönderen tarafta fiziksel katman bir ve sıfırları elektrik sinyallerine çevirip kabloya yerleştirirken, alıcı tarafta fiziksel katman kablodan okuduğu bu sinyalleri tekrar bir ve sıfır haline getirir.

Fiziksel katman veri bitlerinin karşı tarafa, kullanılan medya(kablo, fiber optik, radyo sinyalleri) üzerinden nasıl gönderileceğini tanımlar. Veri iletiminin mümkün olabilmesi için iki tarafın aynı kurallar üzerinde tanımlanmış olması gerekir.

Hub (Göbek) 1.katmanda çalışan bir cihazdır. Bu cihazlar gelen veriyi bir takım elektrik sinyalleri olarak gören ve bu sinyalleri çoğaltıp, diğer portlarına gönderen bir cihazdır.

**4.2. Veri Bağlantı Katmanı**

Veri bağlantı katmanı fiziksel katmana erişmek ve kullanmak ile ilgili kuralları belirler. Bu katmanda Ethernet ya da Token Ring olarak bilinen erişim yöntemleri çalışır. Bu erişim yöntemleri verileri kendi protokollerine uygun olarak işleyerek iletirler. Veri bağlantı katmanında veriler ağ katmanından fiziksel katmana gönderilirler. Bu aşamada veriler belli parçalara bölünür. Bu parçalara paket ya da çerçeve (frame) denir. Çerçeveler verileri belli bir kontrol içinde göndermeyi sağlayan paketlerdir.[2]

Veri bağlantı katmanının büyük bir bölümü ağ kartı içinde gerçekleşir. Veri bağlantı katmanı ağ üzerindeki diğer bilgisayarları tanımlama, kablonun o anda kimin tarafından kullanıldığının tespiti ve fiziksel katmandan gelen verinin hatalara karşı kontrolü görevini yerine getirir.

**4.3. Ağ Katmanı**

Ağ katmanı veri paketine farklı bir ağa gönderilmesi gerektiğinde yönlendiricilerin kullanacağı bilginin eklendiği katmandır. Bu katmanda veriler paket olarak taşınır. Ağ katmanında iki istasyon arasında en ekonomik yoldan verinin iletimi kontrol edilir. Bu katman sayesinde verinin yönlendiriciler (router) aracılığıyla yönlendirilmesi sağlanır. Ağ aşamasında mesajlar adreslenir ayrıca mantıksal adresler fiziksel adreslere çevirilir. Bu aşamada ağ trafiği, yönlendirme gibi işlemler de yapılır. IP protokolü bu katmanda çalışır.

**4.4. Taşıma Katmanı**

Taşıma katmanı üst katmanlardan gelen veriyi ağ paketi boyutunda parçalara böler. TCP, UDP, SPX protokolleri bu katmanda çalışır. Bu protokoller hata kontrolü gibi görevleri de yerine getirir.

Bu katmanda veriler kesim (segment) halinde taşınır.

Taşıma katmanı üst katmanlara taşıma servisi sağlar ayrıca ağın servis kalitesini artırır (QoS – Quality of Service).

Taşıma katmanı verinin uçtan uca iletimini sağlar. Verinin hata kontrolü ve zamanında ulaşıp ulaşmadığı kontrol edilir. Taşıma katmanı ayrıca veriyi üst katmanlara taşıma görevi yapar.

**4.5. Oturum Katmanı**

Oturum katmanında iki bilgisayardaki uygulama arasındaki bağlantının yapılması, kullanılması ve bitilmesi işlemleri yapılır. Bir bilgisayar birden fazla bilgisayarlarla aynı anda iletişim içinde olduğunda, gerektiğinde doğru bilgisayarla konuşabilmesini sağlar. Bu, sunum katmanına yollanacak veriler farklı oturumlarla birbirinden ayrılarak yapılır.

NetBIOS, RPC, Named Pipes ve Sockets gibi protokoller bu katmanda çalışır.

**4.6. Sunum Katmanı**

Sunuş katmanının en önemli görevi yollanan verinin karşı bilgisayar tarafından anlaşılacak şekilde çevrilmesidir. Bu sayede farklı programların birbirlerinin verisini kullanabilmesi mümkün olur.

Sunum katmanı uygulama katmanına verileri yollar daha sonra bu katmanda verinin yapısı, biçimi ile ilgili düzenlemeler yapılır, verinin formatı belirlenir. Ayrıca verinin şifrelenmesi, açılması, sıkıştırılması da bu katmanda yapılır.

GIF, JPEG, TIFF, EBCDIC, ASCII vb. bu katmanda çalışır.

**4.7. Uygulama Katmanı**

Uygulama katmanı bilgisayar uygulaması ile ağ arasında bir arabirim sağlar. OSI katmanları arasında sadece bu katman diğer katmanlara servis sağlamaz. Uygulamaların ağ üzerinde çalışması sağlanır.

Uygulama katmanı ağ servisini kullanacak olan programdır. Bu katman kullanıcıların gereksinimini karşılar. SSH, telnet, FTP, TFTP, SMTP, SNMP, HTTP, DNS protokolleri ve tarayıcılar bu katmanda çalışır. E-posta ve veritabanı gibi uygulamalar bu katman aracılığıyla yapılır.

**5- TCP/IP REFERANS MODELİ**

TCP/IP (Transmission Control Protocol / Internet Protocol), bilgisayarlar ile veri iletme ve alma birimleri arasındaki organizasyonu sağlayan, bu şekilde bir noktadan diğerine veri iletişimini olanaklı kılan birçok veri iletişim protokolüne verilen genel isimdir. Yani TCP/IP protokolleri, bilgisayarlar arasındaki veri iletişim kurallarını koyar.

TCP/IP Referans Modelinin Katmanları, OSI Referans Modelinin Katmanlarına kıyasla 4 tanedir. Örnek tablo aşağıdaki gibidir.

Osı Referans Modeli TCP/IP Referans Modeli

|  |
| --- |
| Application (Uygulama katmanı) |
| Presentation (Sunum Katmanı) |
| Session (Oturum Katmanı) |
| Transport (Taşıma Katmanı) |
| Network (Ağ Katmanı) |
| Data Link (Veri Bağlantı Katmanı) |
| Physical (Fziksel Katman) |

|  |
| --- |
| Uygulama Katmanı |
| Taşıma Katmanı |
| Ağ Katmanı |
| Fiziksel Katman |

**5.1. Uygulama katmanı:** Bu katmanda gönderilecek veri tipi ve veriyi işleyen uygulamalar bulunur. OSI modelindeki sunum ve oturum katmanları TCP/IP modelinde uygulama katmanı içerisinde yer alır.

**5.2 Taşıma katmanı:** Bu katmanda verinin nasıl gönderileceği belirlenir. Veri güvenliği, hata kontrolü gibi işlemler yapılır. TCP bu katmandadır. TCP ‘de gönderilen her veri paketinin ardından verinin yerine doğru bir şekilde ulaşıp ulaşmadığı kontrol edilir.

**5.3. Ağ katmanı:** IP katmanı olarak da adlandırılan bu katman da verilerin gideceği adres veriye eklenir yani veri bu katmandan gönderilir ve yönlendirilir. IPv4 ün gelecekte yetersiz kalma durumuna karşı IPv6 sistemine geçmek için çalışmalar başlatılmıştır.IPv4 32 bit iken IPv6 ile 128 bitlik adresler kullanılacak. Bu sayede daha fazla cihaza IP adresi atanabilecek.

**5.4. Fiziksel katman:** Bu katman verinin hangi yolla gönderileceği belirlenir. İletişim ortamının özelliklerini, haberleşme hızını ve kodlama şemasını belirler. Ethernet, Wi-Fi, Token Ring, ATM gibi protokoller bu katmanda çalışır.

**6- YAZILIM TABANLI AĞLAR (SDN)**

Bilgisayar ağlarında router, switch, firewall gibi pek çok ağ cihazı bulunmaktadır. Bu cihazların çalışmasından farklı farklı protokoller sorumludur. Gelişen teknoloji ile beraber veri merkezleri de gittikçe büyümekte ve bilgisayar ağ cihazlarının sayısı gittikçe artmaktadır. Bu sebeple ağ yönetimi daha karmaşık ve zor bir hal almaktadır. Geleneksel ağ yönetimi yaklaşımı bu durumda yetersiz kalmaya başlamıştır. Bu sebepten dolayı daha iyi bir ağ yaklaşımına ve yeni metotlara ihtiyaç duyulmaktadır. SDN bunlara çözüm olarak ortaya çıkmıştır. SDN yönetim kolaylığı, donanım bağımsızlığı, dinamik, esnek ve ölçeklenebilir ağ mimarisini temin eder. Dolayısıyla bu geniş ve karmaşık ağ yönetimine etkili bir çözüm sunar.

Teknoloji çağını yaşadığımız bu zamanlarda internete bağlanma oranı da hızlı bir artış göstermektedir. İnternete bağlanma oranı arttıkça da sürekli veri üretilir ve bu veriler gittikçe büyür. Sonuç olarak büyük veri konsepti ortaya çıkar ki bu da büyük hacimli, karışık ve düzensiz bilgi demektir. Büyük veri geleneksel yöntemlerle işlenemez yönetilemez ve depolanamaz. Büyük veri anlamlı ve değerli bilgiler içerdiği için işlenmesi gerekir. Büyük veriyi işlemek için daha geniş bant aralığına sahip bir işlem gerekir ve bu esnada SDN devreye girer.

SDN mimarisinde üç adet katmanımız (uygulama, kontrol ve veri katmanları) ve iki arayüzümüz vardır (uygulama-kontrol ve kontrol-veri). Kontrol katmanı paketlerin gönderildiği kısımdır. Veri katmanı paketlerin iletimindeki trafiği düzenleyen kısımdır. Ağ trafiğinde gidilecek yeri günümüzde router ve switchler belirler. Bunlar da kontrol ve veri tabakalarında bulunurlar ki bu tabakalarda birbirine entegre edilmiş şekilde aynı donanım üzerinde bulunurlar. SDN konsepti temelde bu iki tabakayı ayırmaya odaklanmıştır. Kontrol düzlemi yüksek performanslı bir sunucuya taşınır ve ağ yönetimi merkezi denetleyici yazılımı ile gerçekleştirilir. Veri tabakası OpenFlow’un sol tarafında bulunur ve routerların, switchlerin yalnızca paketlerin yönlendirilmesinden sorumlu olmasını sağlar. Bu mimari ağın doğrudan programlanabilmesini sağlamakla beraber ağ servisleri ve uygulamaları için gerekli alt yapıyı da oluşturmaktadır.

Kontrol tabakası ağ işletim sistemi olarak da bilinmektedir. Bu tabaka ağ uygulamaları ve veri tabakası arasındaki iletişimi sağlar. Kontrol tabakası ve veri tabakası arasındaki iletişim açık kaynak ağ protokolü (OpenFlow) ile sağlanır [18].

**7- OPENFLOW**

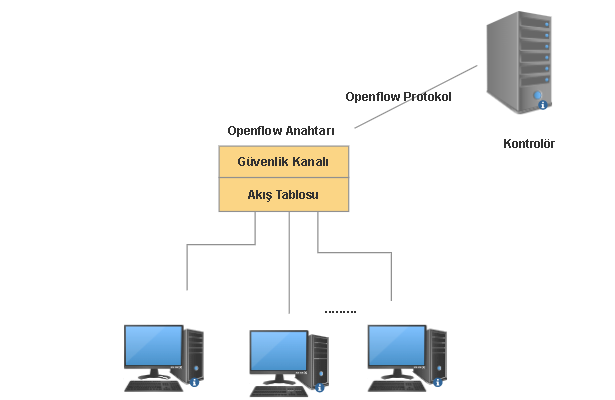
**7.1. OPENFLOW NEDİR?**

OpenFlow, bir iletişim protokolü ve ağ kontrol ve yönlendirme işlevlerini ayıran yazılım tanımlı ağ (SDN-Software Defined Networking) mimarisinin temel öğesidir. SDN ağın küresel görünümünü koruyan yazılım tabanlı denetleyicilerde ağ zekasını ve kontrolünü merkezileştirir. Şu anda OpenFlow, bir SDN denetleyicisi ve ağ donanımı arasında iletişim kurmak için tek açık standart tabanlı protokolüdür. Ağları daha dinamik, yönetilebilir, düşük maliyetli ve uyarlanabilir hale getirerek hem fiziksel hem de sanal anahtarlar ve yönlendiriciler gibi ağ donanımlarının doğrudan programlanmasını sağlar.

OpenFlow'un önemli bir özelliği, SDN kontrol yazılımı kullanılarak statik veya dinamik olarak programlanabilen önceden tanımlanmış eşleşme kurallarına dayanarak, ağ trafiğini tanımlamak için akışları kullanmasıdır. Daha basit bir deyişle, OpenFlow bir denetleyiciden anahtarlara (switches) bilgileri iletir ve anahtarların ne yapacağını söyler. Buna karşılık, anahtarlar denetleyiciye sayaçları ve diğer verileri sağlar.

**7.2. Openflow Mimarisi**

Openflow mimarisi temelde üç adet bileşenden oluşmaktadır. Bunlar denetleyici, anahtar ve denetleyici ile anahtar arasındaki iletişimi sağlayan protokoldür.



Openflow Temel Bileşenleri

**7.3. OPENFLOW ANAHTARI**

Geleneksel ağ anahtarları, birden fazla fiziksel hattı uçtan uca ekleyerek giriş portlarına gelen paketleri çıkış portlarına iletmekte görevlidir. Bu iletim mevcut ağ cihazlarında üç katmanda gerçekleşmektedir. Bunlar kontrol katmanı, veri katmanı ve yönetim katmanıdır.

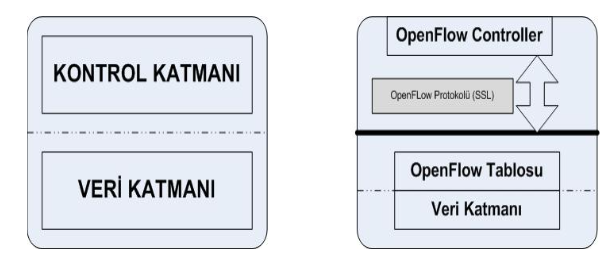
**7.3.1. Kontrol Katmanı:** Anahtar üzerinde çalışan protokollerin karar aldığı kısımdır. Anahtar giriş portlarına gelen paketlerin nereye gideceği bu katmanda belirlenir. Flowların yani paketlerin işlendiği katmandır.

**7.3.2. Veri Katmanı:** Anahtar giriş portlarına gelen paketlerin yani flowların tutulduğu katmandır. Kontrol katmanından aldığı bilgilere göre flowların iletilmesinden sorumludur.

**7.3.3.Yönetim Katmanı:** Anahtar üzerindeki yapılandırma ve yönetim işlemleri bu katmanda

gerçekleşmektedir [3]

Mevcut ağ anahtarlarında bu üç katman tek bir cihaz üzerinde bulunarak paketlerin iletilmesinden sorumludur. Bu da anahtarlar üzerinde bir sorunla karşılaşıldığında gerekli çözümün uygulanmasını zorlaştırmaktadır. Anahtarın tamamında karmaşık ayarlar yapılmasına hatta bazı durumlarda problemin giderilememesine sebep olmaktadır.



OpenFlow Ağ Anahtarı Genel Yapısı

OpenFlow anahtarı bu soruna çözüm olarak ortaya çıkmıştır. Openflow anahtarlarında sadece veri katmanı yani flow iletiminden sorumlu katman bulunmaktadır. Kontrol ve yönetim katmanı anahtarın dışına çıkarılarak, flow iletiminde karar alırken anahtara esneklik kazandırılmıştır. Normalde tek bir cihaz üzerinde gerçekleşen flow iletimi openflow anahtarı, openflow denetleyicisi ve anahtardenetleyici arası iletişimi sağlayan openflow protokolü olmak üzere üç kısımda gerçekleşmektedir.

OpenFlow anahtarının en temel bileşenleri ise flowlar ve flow tablolarıdır.

**8- FLOW VE FLOW TABLE**

**8.1. Flow**

Flow tanım olarak; aynı kaynak IP’si, hedef IP’si, kaynak portu, hedef portu, protokol demet öğelerini paylaşan ve ayarlanabilen bir D zaman aşımı süresiyle ayrılmış tek yönlü paketler serisidir [4].

**8.2. Flow Table**

Flow tabloları, ağ anahtarları, yönlendiriciler, ateş duvarları gibi, günümüzdeki ağ bileşenlerinin çoğunda flow bileşenlerinin saklandığı bir yapıdır. Ağ anahtarlarında ve

yönlendirici cihazların içinde veri –satıhı (data-plane ) içinde tutulmaktadır.

|  |
| --- |
| **FLOW TABLE** |

|  |  |  |
| --- | --- | --- |
| Başlık | Eylemler | Sayaçlar |
| Başlık | Eylemler | Sayaçlar |
| . | . | . |
| . | . | . |
| . | . | . |
| Başlık | Eylemler | Sayaçlar |

Flow table

OpenFlow ağ anahtarlarında bulunan flow tabloları 3 adet temel bileşenden oluşmaktadır.

•Flow’u tanımlayan paketin başlığı

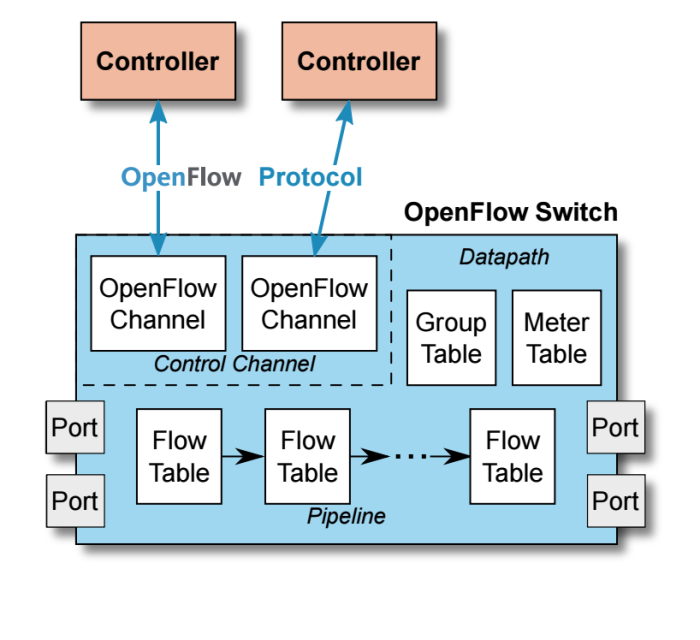
•Paket in nasıl işleneceğini belirten bölüm (action)

•Flow paketlerinin, boyutu, izleme numarası ve zaman aşımı bilgilerinin bulunduğu istatistik veri bölümü.

**9- OPENFLOW SWİTCHES**

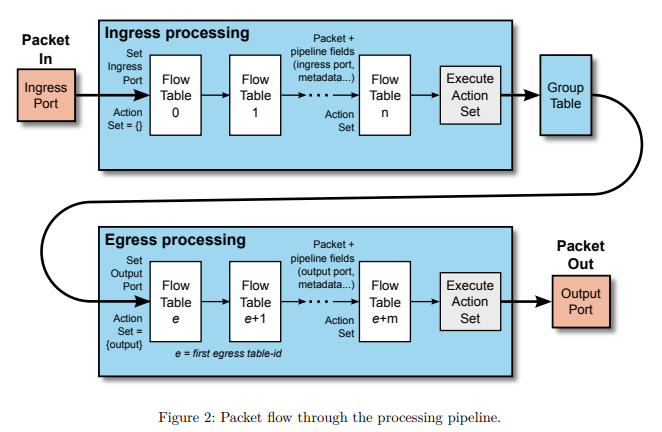
OpenFlow anahtarları, bir veya daha fazla akış tablosundan, bir grup tablosundan oluşur

paket arama ve iletme işlemlerini gerçekleştiren bir sayaç tablosu, sayaç girişleri, akış ölçer başına tanımlama, bir veya daha fazla OpenFlow kanalı harici bir denetleyici ve akış girişlerini iletmek için bağlantı noktası. Dışardan controller aracılığıyla OpenFlow kanallarına gelir. İlgili anahtarlarla flow’u nasıl işleyeceğini belirten flow table girdileri üzerinde kontrol imkanı sağlanır. Akış tablosu ve grup tablosu temelde paket arama ve iletmelidir.



**10- OpenFlow Tables**

Gelen paket Flow table 0’a gelir. Bu paket için uygun olan fonksiyonu bulana kadar tablo bu eyleme göre çıkış portuna gönderir. OpenFlow paket için ayarlanan eylemi günceller ve paketi sonraki akış tablosuna iletir. Son akış tablosunda OpenFlow anahtarı paket içeriğini değiştirmek ve paket iletme için çıkış portunu belirlemek için tüm eylemleri yürütür.[5]

****

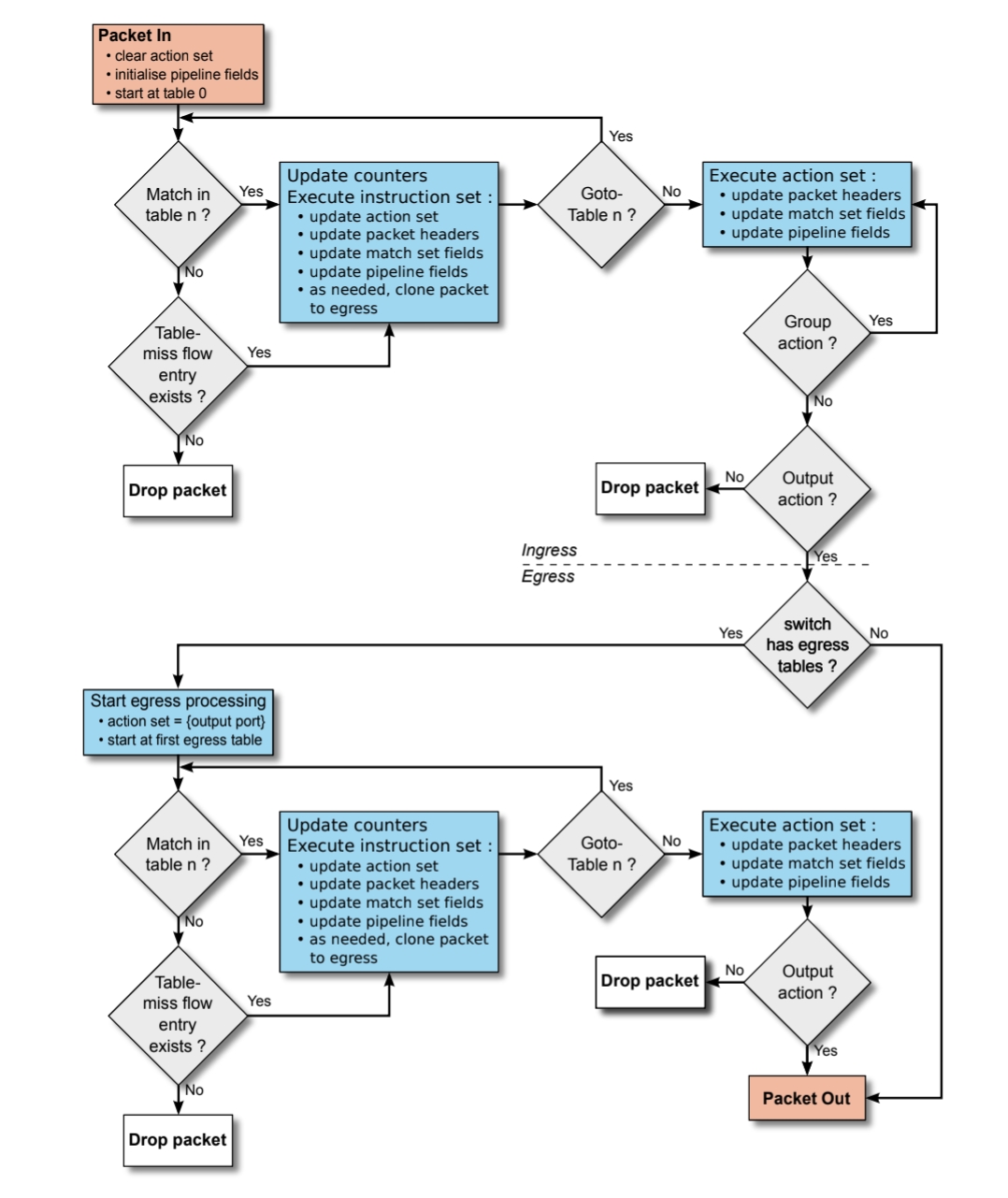
Bir OpenFlow anahtarı aracılığıyla paket sürecini özetlemektedir. Paket tarafından takip edilen süreç:

(a) Anahtar, birinci akış tablosunda bir tablo araması yaparak başlar, ve boru hattı işlemeye dayalı olarak, içinde tablo aramaları yapabilir .

(b) Paket başlık alanları çıkarılır ve paket boru hattı alanları alınır.

(c) Akış girişinin tüm eşleşme alanları karşılık gelen başlık alanları ve ardışık düzen alanlarıyla eşleşiyorsa, paket bir akış girişiyle eşleşir.

**10.1. Openflow Anahtarı Aracılığıyla Paket Süreci:**



1. **FLOODLİGHT VM**

YTA (Yazılım Tanımlı Ağlar) için kullanılan bir araçtır. Floodlight, açık kaynak kodludur ve Java programlama dilini destekleyen bir uygulamadır (Floodlight, 2020). Floodlight sitesinde yazılımcılar için gerekli tüm adımlar ve kurulumlar ayrıntılı şekilde sunulmakta olup, önerilen metodun program aşamaları için yine açık kaynak kodlu ve özellikle Java tabanlı yazılımların tasarlandığı Eclipse ortamı kullanılmaktadır. Eclipse, kurulum sırasında Floodlight VM ile entegreli olarak gelmektedir. Uygulamada oluşturulan topolojiler Mininet üzerinde Linux tabanlı proğramlama dili ile oluşturulmaktadır. Uygulamada trafiği gözetlemek için gerekli olan veriler anahtar, port numarası, alınan ve iletilen paketlerin miktarı, bant genişliği, düşmeler, çarpışmalar ve bunların gerçekleştiği saniye (sn) cinsinden süre bilgileridir. Bu verilere Floodlight VM içerisinde gerçekleştirilmektedir.[6]

Floodlight, mininet, Open vSwitch ve Floodlight v1.1 ile önceden yapılandırılmış bir sanal makineye sahiptir. Henüz yapmadıysanız, buradan indirebilirsiniz:

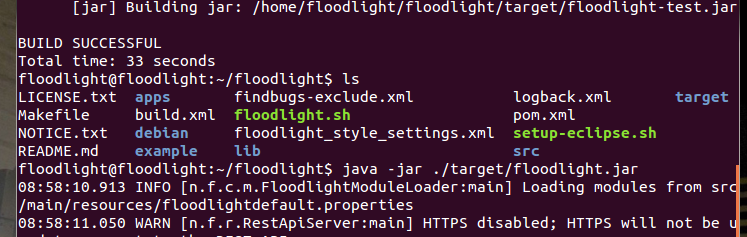
* **opennetlinux.org/binaries/floodlight-vm.zip**

Floodlight VM kuruldu, derledik ve çalıştırdık.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Şekil1. Floodlight kurulumu**



**Şekil2. Floodlight kurulumu**

1. **MİNİNET**

Mininet YTA’ da OpenFlow ağına bağlanabilmek için kullanılan bir simülasyon aracıdır. Mininet üzerinde ping atamaları gerçekleştirilebilir. Mininet basit veya karmaşık ağ yapılarını tasarlayabilmektedir. Bunlar; host, switch, router ve serverden oluşmaktadır.

Mininet'in amacı:

— OpenFlow için ağları test etmek için basit ve ucuz bir yol sağlar.

— Birden çok araştırmacının aynı ağ topolojisi üzerinde bağımsız olarak çalışmasına izin verir.

— Büyük ve karmaşık bir topolojinin test edilmesine izin verir.

— Ağ genelinde hata ayıklamak ve testleri çalıştırmak için araçlar içerir;

— Çok sayıda topolojiyi destekleyin ve temel bir topoloji kümesi içerir;

— Ağ oluşturmak ve test etmek için basit Python API kodları yazarak çalıştırılabilir.

Mininet:

* OpenFlow uygulamaları geliştirmek için basit ve ucuz bir ağ test ortamı sağlar
* Birden çok eşzamanlı geliştiricinin aynı topoloji üzerinde bağımsız olarak çalışmasına olanak tanır
* Tekrarlanabilir ve kolayca paketlenebilen sistem düzeyinde regresyon testlerini destekler
* Fiziksel bir ağ bağlantısı kurmaya gerek kalmadan karmaşık topoloji testini mümkün kılar.[7]

**NOT:** Floodlight’ı çalıştırmadan mininet’in içerisine giremeyiz. Öncelikle floodlight’ın içerisine gireriz ($cd floodlight) Ardından $java jar ./target/floodlight.jar deyip çalıştırırız.

Oluşturduğumuz topolojiyi böylelikle terminale yazıp mininetin içerisine girmiş oluruz.

1. **AĞ SİMÜLASYONU**

Artık Floodlight çalıştığına göre, onu bir OpenFlow ağına bağlamanız gerekir. Bunun için en iyi araçlardan biri bir ağ simülasyon aracı olan Mininet'tir .

* Floodlight sanal makinesi Floodlight ve Mininet içerir.
* VMware Fusion veya VirtualBox'ta başlatılır.
* Giriş (kullanıcı adı Floodlight ve şifre Floodlight)

Mininet'i yerel olarak çalışan Floodlight'a karşı çalıştırmak mümkündür ("sudo mn" yazmanız yeterlidir), ancak onu kendi oluşturduğunuz bir remote controller karşı da çalıştırabilirsiniz. Bunu yapmak için aşağıdaki ilgili komutu yazarız:[8]

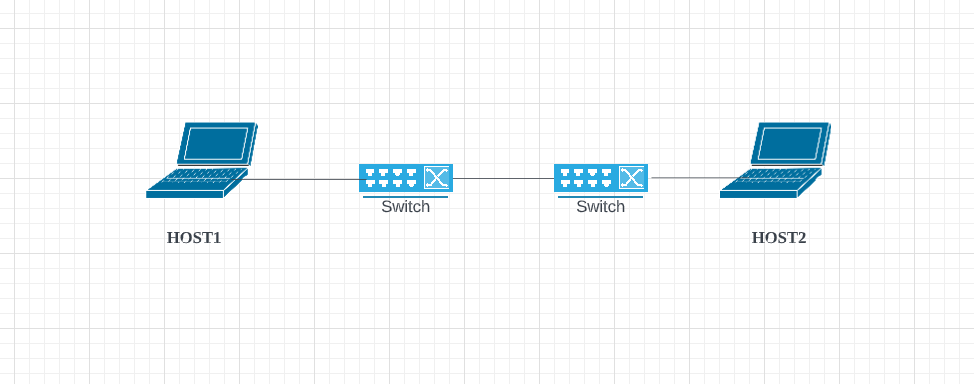
sudo mn --controller=remote,ip=127.0.0.1,port=6653 –switch ovsk,protocols=OpenFlow13

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Şekil 3. Mininet**

Şekil 3’te iki host ve iki switch olduğunu görmüş olduk.



**13.1 Mininet Ortamının Temel Komutları ve İşlevleri**

* **Nodes:** Şu anda etkin olan mininet topolojisinin tüm düğümlerini listeler. Bu düğümler, denetleyici, anahtarlar ve ana bilgisayarları içerir.
* **net :** O anda çağrılan düğümler arasındaki tüm bağlantıları görüntüler.
* **dump :** Bu komut, aktif topolojide yer alan tüm düğümler hakkında bilgi dökümü yapar. Bu, kullanıcıya bilgi sağlar. Düğümlerin IP adresi ve her düğüm için işlem tanımlayıcısı gibi.
* **sh :** Bu komut, programcının mininet ortamındaki kabuk komutlarını kullanma konusundaki yetersizliklerinin üstesinden gelmek için kullanılır.

* **xterm :** Bu komut, ayrı bir terminal için bağımsız uçbirim sağlar. Topolojideki düğüm. Bu ile çeşitli testler yapılabilir.
* **ping :** Bu komut, düğümlerin düğümler arasında ping atmasına izin verir. Ping komutu temel olarak düğümlerin erişilebilirliğini test etmek için kullanılır. Birinden diğerine. Yalnızca düğümler arasında ping yapmak istiyorsak, aşağıdaki komut yardımcı olur: host1 ping host2. Numaradan bağımsız olarak ping komutu paketi birer birer gönderir.
* **pingall :** İki komut arasında ping atan önceki komutun aksine düğümler Bu komut, içindeki tüm düğümler arasında ping yapmak için kullanılır. Her düğüm, topolojideki diğer tüm düğümlere tek tek ping gönderir.
* **iperf :** İperf aslında ağı ölçmek için kullanılan bir araçtır. Bant genişliğini karşılaştıran hem TCP hem de UDP bant genişliği performansını ölçebilir. Düğümler, hub gibi veya tek bir topolojide anahtar gibi kullanılır.

1. **ÖZEL TOPOLOJİ OLUŞTURMA**

Özel topolojiler de basit bir Python API kullanılarak kolayca tanımlanabilir ve proje.py adında dosyayı adlandırdık ilgili python kodunu yazmış olduk.

Öncellikle oluşturacağımız topolojinin görseli aşağıdadır. Görselde iki host üç switch yer almaktadır.

metin, metre, cihaz, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Özel topoloji örneği oluşturarak, basit python apı kullanarak kendi hostlarımızı ve switchlerimiizi oluşturmuş olduk. Aşağıdaki görselde de görüldüğü üzere switch ve hostlar ekleniyor ve ardından bağlantılar ekleniyor.[9] Doğrudan bağlı üç switch artı iki uç noktasında bir switch için bir ana bilgisayar:

HOST1----SWİTCH1----SWİTCH2----SWİTCH3----HOST2

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**14.1. Python Scripti Parametreleri**

• Topo: Mininet topolojileri için kullanılan temel sınıfı belirtmektedir.

• **addHost**: Topolojiye host eklemek için kullanılır.

• a**ddSwitch**: Topolojiye switch eklemek için kullanılır.

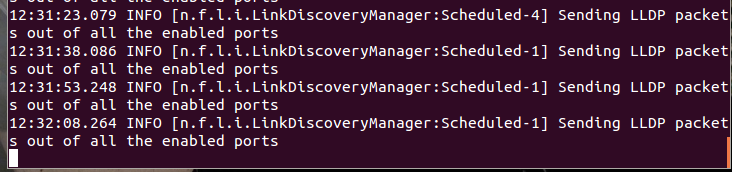
• **addLink:** Topolojide yer alan nesneler arasında çift yönlü bir bağlantı oluşturmak için kullanılır. Aksi belirtilmediği sürece Mininet’teki bağlantılar çift yönlüdür.

Daha sonra terminale geldiğimizde oluşturduğumuz topolojinin proje.py olarak çalıştırılarak topoloji gerçekleştirilmiştir. İçerisinde de oluşturduğumuz topolojinin switchleri ve hostları görselde görüldüğü gibidir. Mininet içerisinde pingall dedik ve hostların birbirleri arasında paket gönderdiğini görmüş olduk.[10]

Tüm etkin bağlantı noktalarından LLDP paketleri gönderme.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



1. **İPERF ARACI**

İnternet protokolü ile yönetilen ağlarda, erişilebilen max bant genişliğinin ölçümü için kullanılan araçtır.

İperf aracının kullanılabilmesi için mininet ortamında xterm h1 ve xterm h2 olmak üzere iki ayrı host açılır. Hostların biri istemci diğeri sunucudur.

Sunucu ekranına iperf -s, istemci ekranına ise iperf -c sunucu\_ip\_address yazılarak iki uç arasında trafik oluşturulur.

Trafik oluşturmak için mininet simülasyon ortamında iperf aracı kullanılır.[11]

**15.1. İperf Aracı Parametreleri**

|  |  |  |
| --- | --- | --- |
| Kullanım Yerleri | Parametre | Açıklaması |
| Genel | **-p**, --port | İletimin yapılacağı port değerinin belirlenmesi için kullanılır. Varsayılan değer 5001’dir. |
|  | **-u** , --udp | UDP paketinin iletimi yapılacağı belirtir. |
|  | **-i ,** --interval | Periyodik olarak gönderilen bilgilendire raporunun saniye cinsinden gönderilme sıklığını gösterir. |
|  | **-h,** --help | Komut listesini görüntüler. |
| Server/Sunucu | **-s,** --server | Host’un sunucu modunda çalıştırılacağını belirtir. |
| Client/İstemci | **-b**, --bandwith | UDP paketlerinin gönderimi sırasında kullanılacak bant genişliğini ifade eder. |
|  | **-c**, --client | Host’un istemci modunda çalıştırılacağını belirtir. |
|  | **-t,** --time | Paket iletiminin ne kadar süreceğinin saniye cinsinden belirtir. |
|  | **-S**, --tos | Paketlerin hangi hizmet sınıfından olduğunun belirtilmesi için kullanılır. Ör. AF11=0x28 |

**Mininet>**pingall deyip ping atayacağımız hostlar önümüze çıkar (h1veh2).

Aşağıdaki görselde;

**Mininet>**xterm h1 dedik ve birinci host’ a giriş yaptık.

HOST1 için;

**Floodlıhght>**iperf [-s],-c

**Floodlıhght>**ifconfıg dedik ip adresine bakıldı.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Mininet>**xterm h2 dedik ve ikinci host’ a giriş yaptık. Ardından sunucu kısmına ,

**Floodlıght>**iperf -s 10.0.0.2 (h2 ipsi) yazarak ıp ataması gerçekleştirildi.

Host1’in terminal ekranına Iperf –s –u komutu yazılarak Host1’in sunucu olduğu ve UDP paketi göndereceği belirtilmiştir. Host2’nin terminal ekranına ise Iperf -c 10.0.0.1 -u UDP paket belirlenir .[12]

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. **HİZMET KALİTESİ QOS**

Ağ İletişimi Hizmet Kalitesi (İngilizce Quality of Service, kısaca QoS), Ağ üzerindeki uygulamaları önceliklendirerek zaman kaybını azaltmayı hedefleyen bir ağ servisi. Bir ağ bağlantısı üzerinden çalışan bir trafik veya program türüne öncelik veren çeşitli tekniklere karşılık gelir. Aynı ağ altyapısı ile çalışan çeşitli uygulamalarda network cihazlarının hangi uygulamaya öncelik vereceğini belirtmeye yarar. Bu sayede kesinti ve paket gecikmelerine daha az tahammül olan uygulamalara öncelik verilir.

Örneklemek gerekirse bir otoyol üzerindeki araç trafiğinde, ambulans, itfaiye, polis gibi araçların geçiş üstünlüğü bulunmaktadır. Bilgisayar ağlarında da benzer şekilde bazı uygulamalar kritik önem arz ederken bazı uygulamalar ise önemsizdir.

Çok fazla kişinin çalıştığı bir iş yerinde, kullanıcıların yoğun internet trafiği, kişisel zevkleri için müzik ve video erişimleri önemli yük oluşturabilir. Bu yoğunluğun şirketin voip telefon hatlarını bozması, banka, muhasebe ve müşteri hizmetleri sunan uygulamaları yavaşlatmasını engellemek gereklidir.

Ağ üzerindeki uygulamaları önceliklendirerek zaman kaybını azaltmayı hedefleyen bir ağ servisidir. Birden çok İnternet kullanıcısının aynı modem üzerinde eş zamanlı İnternette girmesi sonucu bant genişliği düşer.[13]

1. **OVS İLE QUEUE YAPILANDIRILMASI**

**17.1. Hangi Anahtarlar Kuyrukları Destekler?**

Kuyruklar, çok kullanışlı olmasına rağmen, OpenFlow protokolünün dışında tanımlanır. OpenFlow, söz konusu kuyruklar hakkında denetleyiciyi bilgilendirmek için yalnızca mevcut anahtar kuyruk mekanizmalarının etrafında bir sarmalayıcı sağlar. Kuyruklar, bant dışı geçiş ile tanımlanmalı/başlatılmalıdır. Projemizde OVS switch komutlarıyla oluşturulmuş olup paket gönderimi trafik akışını oluşturmuş olduk.

**17.2. OVS’ de Kuyrukları Yapılandırma**

İlk olarak, bir QoS politikası uygulamak istediğimiz anahtar bağlantı noktasını belirliyoruz - bu bağlantı noktası eth1 olarak adlandırılır ve anahtarımıza s1 yeni eklediğimiz bağlantı noktasıdır. Ardından, linux-htb türünde yeni bir QoS örneği oluşturuyoruz (çoğu Linux dağıtımında ortaktır). Bu QoS politikasının maksimum hızını 5Mbps olarak belirledik. Daha sonra bu port üzerinde belirli kuyrukları oluşturuyoruz. Kuyruk 0, minimum 3 Mbps ve maksimum 3 Mbps hıza sahip kuyruğumuzdur – başka bir deyişle, işlenecek paketleri olduğunda mümkün olduğunca 3 Mbps'ye yakın kalmaya çalışacak bir kuyruk. Minimum oranınız elbette maksimum oranınızı geçmediği sürece maksimum ve minimum oranları beğeninize göre değiştirebilirsiniz. Ayrıca, kapsamlı QoS politikasının hızı içinde kalmalısınız.

$ ovs-vsctl set port eth0 qos=@newqos -- --id=@newqos create qos \

type=linux-htb other-config:max-rate=5000000 queues:0=@newqueue -- \

--id=@newqueue create queue other-config:min-rate=3000000 \

other-config:max-rate=3000000

İlk olarak, bir QoS politikası uygulamak istediğimiz anahtar bağlantı noktasını belirliyoruz - bu bağlantı noktası eth0 olarak adlandırılır ve anahtarımıza s0 yeni eklediğimiz bağlantı noktasıdır. Ardından, linux-htb türünde yeni bir QoS örneği oluşturuyoruz (çoğu Linux dağıtımında ortaktır). Bu QoS politikasının maksimum hızını 5Mbps olarak belirledik. Daha sonra bu port üzerinde belirli kuyrukları oluşturuyoruz. Kuyruk 0, minimum 3Mbps ve maksimum 3Mbps hıza sahip kuyruğumuzdur – başka bir deyişle, işlenecek paketleri olduğunda mümkün olduğunca 3Mbps'ye yakın kalmaya çalışacak bir kuyruk. Minimum oranınız elbette maksimum oranınızı geçmediği sürece maksimum ve minimum oranları beğeninize göre değiştirebilirsiniz. Ayrıca, kapsamlı QoS politikasının hızı içinde kalmalısınız (bu örnekte 5 Mbps).

1. **OPEN VSWİTCH NEDİR?**

Bazen OVS olarak kısaltılan açık vSwitch, dağıtılmış bir sanal çok katmanlı anahtarın açık kaynaklı bir uygulamasıdır. Open vSwitch'in temel amacı, bilgisayar ağlarında kullanılan çoklu protokolleri ve standartları desteklerken donanım sanallaştırma ortamları için bir anahtar yığını sağlamaktır.

**18.1. OVS ‘de Temel Komutlar**

OVS, farklı yapılandırma komutlarıyla zengin özelliklere sahiptir, ancak yapılandırmanızın ve sorun gidermenizin çoğu aşağıdaki 4 komutla gerçekleştirilebilir:

**ovs-vsctl :** ovs-vswitchd yapılandırma veritabanını yapılandırmak için kullanılır (ovs-db olarak bilinir)

**ovs-ofctl :** OpenFlow anahtarlarını izlemek ve yönetmek için bir komut satırı aracı

**ovs-dpctl :** Open vSwitch veri yollarını yönetmek için kullanılır

**ovs−appctl :** Open vSwitch arka plan programlarını sorgulamak ve kontrol etmek için kullanılır.

**18.1.1 Ovs-Vstcl**

Bu araç, OVS anahtar işlemlerini yapılandırmak ve görüntülemek için kullanılır. Bağlantı noktası yapılandırması, köprü eklemeleri/silmeleri, bağlama ve VLAN etiketleme, bu komutla kullanılabilen seçeneklerden yalnızca birkaçıdır.

Aşağıda en kullanışlı 'göster' komutları verilmiştir:

**ovs-vsctl –V:** openvswitch'in geçerli sürümünü yazdırır.

**ovs-vsctl show:** Anahtar veritabanı yapılandırmasına kısa bir genel bakış yazdırır.

**ovs-vsctl list-br:** Yapılandırılmış köprülerin listesini ovs-vsctl list-ports <bridge>yazdırır : Belirli bir köprüdeki bağlantı noktalarının listesini yazdırır. ovs-vsctl list interface: Arayüzlerin bir listesini yazdırır.

**ovs-vsctl add-br <bridge>:** Switch veritabanında bir köprü oluşturur.

**ovs-vsctl add-port <bridge> <interface>:** Bir arabirimi (fiziksel veya sanal) bir köprüye bağlar

**18.1.2 Ovs-Ofctl**

Bu araç, OpenFlow anahtarlarını yönetmek ve izlemek için kullanılır. OVS, merkezi yönetim için yapılandırılmamış olsa bile, özellikler, yapılandırma ve tablo girişleri dahil olmak üzere OVS'nin mevcut durumunu göstermek için ovs-ofctl kullanılabilir.

Aşağıda yaygın gösteri komutları verilmiştir:

**ovs-ofctl show <bridge>:** OpenFlow özelliklerini ve bağlantı noktası açıklamalarını gösterir.

**ovs-ofctl dump-flows <bridge> <flow>:** Belirtilen köprünün akış girişlerini yazdırır. Belirtilen akışla, konsola yalnızca eşleşen akış yazdırılacaktır. Akış atlanırsa, köprünün tüm akış girişleri yazdırılacaktır.

**ovs-ofctl add-flow <bridge> <flow>:** Belirtilen köprüye statik bir akış ekleyin. Bir akış için koşulların tanımlanmasında faydalıdır (örn. önceliklendirme, düşürme, vb.).

**ovs-ofctl del-flows <bridge> <flow>:** Belirtilen köprünün akış tablosundan akış girişlerini silin. Akış atlanırsa, belirtilen köprüdeki tüm akışlar silinecektir.

**18.1.3 Ovs-Dpctl**

ovs-dpctl, her ikisinin de akış tablosu girişlerini göstermesi bakımından ovs-ofctl'ye çok benzer. ovs-dpctl'nin yazdırdığı akışlar her zaman tam bir eşleşmedir ve son birkaç saniye içinde sistemden fiilen geçen paketleri yansıtır. ovs-dpctl, bir OpenFlow anahtarını değil, bir çekirdek veri yolunu sorgular. Akış verilerinde hata ayıklamak için kullanışlı olmasının nedeni budur.

Ardından akış tablosu verilerini görüntülemek için aşağıdakileri kullanın:

ovs-dpctl dump-flows

1. **KUYRUKLAR ARASI TRAFİK AKIŞI**

**19.1. Switchte Kuyruk Yapılandırma**

Switchlere kuyruk bağlayarak kuyruklar arası trafik akışını izleyebiliriz.

Switch1’e kuyruk bağlamak için;

**$ sudo ovs-vsctl list qos** // tanımlanan kuyrukları listeler

**$ sudo ovs-ofctl dump-flows s1**  //s1 anahtarında tanımlanan kuyrukların istatistiklerini verir

**$ sudo ovs-vsctl set port s1-eth1 qos=@qos1 -- --id=@qos1 create qos type=Linux-htb queues:0=@queue0 queues:1=queue1 -- --id=@queue0 create queue other-config:max-rate=30000 -- --id=@queue1 create queue other-cnfig:max-rate=50000** //q0 ve q1 iki tane kuyruk oluşturuyor s1-eth1 portuna

**$ sudo ovs-ofctl add-flow s1 priority=65535,tcp,tcp=dst=9090,actions=set\_queue:0, normal** //9090 portuna giden tcp trafiği q0 a bağlanıyor.

**$ sudo ovs-ofctl add-flow s1 priority=65535,tcp,tcp=dst=9091,actions=set\_queue:1, normal** //9091 portuna giden tcp trafiği q1 e bağlanıyor.

Switch1 üzerinde kuyruk yapılandırmada ;

Q0 ve q2 iki tane kuyruk oluşturuluyor s1-eth1 portuna ardından 9090 portuna TCP trafiğiyle kuyruklara bağlanır. S1 anahtarında tanımlanan kuyrukların istatistiklerine bakıldığında iki kuyruğunda oluşmuş olup, birbirlerine trafik göndermediği görülür. Trafik göndermek için ise oluşturmuş olduğumuz özel topolojinin mininet simülatörü içerisinde xterm h1 h2 yani hostları çağırılır ve h2 içerisine *iperf -s -p 9090*  yazılarak sunucu ekranına 9090 portunda TCP türünde bir trafik gideceğini belirtmiş olup h1 istemci ekranına ise *iperf -c 10.0.0.2 -p 9090* TCP portuna ip ataması gerçekleştirilir ve 0.0-10.0 sec aralığında transfer ve bandwith genişliğini görmüş oluruz. Aynı şekilde 9091 portuna TCP trafiğiyle kuyruklara bağlanır. S1 anahtarında tanımlanan kuyrukların istatistiklerine bakıldığında iki kuyruğunda oluşmuş olup, birbirlerine trafik göndermediği görülür. Trafik göndermek için ise oluşturmuş olduğumuz özel topolojinin mininet simülatörü içerisinde xterm h1 h2 yani hostları çağırılır ve h2 içerisine *iperf -s -p 9091*  yazılarak sunucu ekranına 9091 portunda TCP türünde bir trafik gideceğini belirtmiş olup h1 istemci ekranına ise *iperf -c 10.0.0.2 -p 9091* TCP portuna ip ataması gerçekleştirilir ve 0.0-10.0 sec aralığında transfer ve bandwith genişliğini görmüş oluruz.

Aşağıdaki görselde de görüldüğü üzere kuyruklar arası trafik akışını göstermektedir:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Istemci ve sunucu ekranlarına ilgili TCP portunu tanımladıktan sonra s1 anahtarında tanımlanan kuyruk istatistiklerine bakıldığında q0 ve q1’ in birbirlerine trafik gönderdiğini aşağıdaki görselde görülmektedir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**20. Meter**

Topoloji oluştururken meter configürasyonunun Dpctl ile yapılabilmesi için switch=user olarak belirtilmelidir. Open Flow switch için kullanılan protokolümüzü (OpenFlow13) dahil ediyoruz.

OpenFlow protokolü, kontrolör ile OpenFlow switch’ler arasındaki iletişimin standartlaştırılması için oluşturulmuştur. Bu protokol kontrolörden switch’e, asenkron ve simetrik mesajlar olmak üzere 3 farklı mesaj türünü desteklemektedir [2]

Kontrolörden switch’e giden mesajlar sayesinde kontrolör direkt olarak switch’leri yönetebilmektedir. Asenkron mesajlar ağda meydana gelen olaylar veya switch’lerdeki değişiklikler ile ilgili olarak kontrolörün bilgilendirilmesi amacıyla switch’ler tarafından gönderilmektedir. Simetrik mesajlar ise switch ya da kontrolör tarafından herhangi bir talebe ihtiyaç duyulmadan gönderilebilmektedirler [16].

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

21.DPCTL komutları

Komut yapısı : dpctl [OPTIONS] SWİTCH COMMAND [ARG...] DPCTL komutları; özellikler, konfigürasyon, durum tanımlamaları, tablo durumları, grup, Meter, port, akış modları olarak kategorilere ayrılmaktadır [19] . Çalışmamızda kullanılacak olan Meter komutlarının bilinmesi önem taşımaktadır.

21.1.Meter komutları

Meter-mod: Meter-mod komutu aracılığıyla Meter tablolarına kayıt ekleme, kayıtları silme veya düzenleme gibi modifikasyon işlemleri yapılabilmektedir.

Kullanılışı: metin, ekran görüntüsü, ekran içeren bir resim

Açıklama otomatik olarak oluşturuldu

Yukarıdaki komut satırı ile birlikte Meter tablosuna yeni bir Meter eklenmesi istenmektedir. Komut satırında yer alan parametre ve değerlerden cmd=add yeni bir Meter ekleneceğini, flags=1 kullanılacak ölçüm biriminin Kbps olduğunu, Meter=1 Meter numarasını, drop Meter’ın band türünü, rate=400 ise Kbps cinsinden limit değerini göstermektedir. Bu komuta cevaben switch aşağıdaki mesajı göndererek oluşturulan Meter’a ait özellikleri bildirmektedir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Eklenen akış-kaydına göre switch’e gelen paketlerden IPv4 36 protokolüne göre çalışan ve AF11 (DSCP=40) DSCP değerine sahip paketlerin önce Meter-1 den geçmesi daha sonrada 2 numaralı porttan çıkış yapması sağlanmıştır.

**EtherType , bir**[Ethernet çerçevesindeki](https://en.wikipedia.org/wiki/Ethernet_frame) iki [sekizli](https://en.wikipedia.org/wiki/Octet_(computing)) bir alandır . Çerçevenin yükünde hangi [protokolün](https://en.wikipedia.org/wiki/Communications_protocol)[kapsüllendiğini](https://en.wikipedia.org/wiki/Encapsulation_(networking)) belirtmek için kullanılır ve yükün nasıl işlendiğini belirlemek için [veri bağlantı katmanı tarafından alıcı uçta kullanılır.](https://en.wikipedia.org/wiki/Data_link_layer) Aynı alan, bazı Ethernet çerçevelerinin boyutunu belirtmek için de kullanılır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu , [IPv4 başlığının](https://en.wikipedia.org/wiki/IPv4_header)*Protokol* alanında ve [IPv6 başlığının](https://en.wikipedia.org/wiki/IPv6_header)*Sonraki Başlık* alanında bulunan **IP protokolü numaralarının** bir listesidir . Kapsüllenmiş protokol için bir tanımlayıcıdır ve başlığın hemen ardından gelen verilerin düzenini belirler. Her iki alan da sekiz [bit](https://en.wikipedia.org/wiki/Bit) genişliğindedir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

İp\_dscp=40 Gelen paketin sınıfını belirtmek için kullanılan bir eşleşme parametresidir. “40” değeri AF sınıfından düşürülme önceliği düşük bir paketi temsil etmektedir.

Apply:output=2 Eşleşme alanlarındaki kayıtlara uygun olarak gelen paketler için çalıştırılacak komutları belirtir. “output=2” değeri paketin 2 numaralı çıkış portundan iletiminin sağlanacağını belirtmektedir.

meter: 1 Switch’e gelen paketlerin Meter’lardan geçirilmesi için kullanılan komuttur. “1” değeri switch’te kurulu olan ilk Meter’ı (Meter\_id=1) belirtir.

Host 1 – istemci olarak işaretlendi

Host 2 – sunucu olarak işaretlendi

SONUÇ VE BULGULAR

Kuyruk eklenmeden önce istemci ve sunucu arasında ping ataması gerçekleştiğinde elde ettiğimiz sunucu ekranında elde e(10.0.0.2 ıple) bant genişliği: 26.9 Gbits/sec , transfer: 31.3 GBytes’tır. Istemci (10.0.0.1 ıp ataması ) ile elde ettiğimiz bant genişliği: 31.0 Gbits/sec, transfer:36.1 GBytes ‘tır

Kuyruk eklemeden önceki hali:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Sw1 ‘ e kuyruk eklemede önce max-rate=50000 belirleyip, TCP=9090 ve TCP=9091 portuna trafik yolladığımızda 9090 portu için;

İstemci ekranındaki bant genişliği:2.39 Mbits/sec , transfer:3.12 Mbytes,

Sunucu ekranındaki bant genişliği:1.66 Mbits/sec , transfer:3.12 Mbytes’tır.

9091 portu için;

İstemci ekranındaki bant genişliği:2.39 Mbits/sec, transfer: 3.12 Mbytes,

Sunucu ekranındaki bant genişliği:1.65 Mbits/sec , transfer:3.12 Mbytes’tır.

Böylelikle kuyruk eklediğimizde trafik akışında daha az paket yollarken, kuyruk olmadığı durumda daha fazla paket yollamaktadır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

UDP portuna aynı şekilde yapılandırdığımızda;

9090 ve 9091 portu için:

İstemci ve sunucu ekranındaki bant genişliği:1.05 Mbits/sec , transfer:1.25 Mbytes,

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Sw1 kuyruk yapılandırmada max-rate=800K belirlediğimizde bizden 800K altı bant genişliğini vermesini istiyoruz . Sonuca baktığımızda bant genişliğini 544-569 MBits/sec olarak vermekte yani belirlemiş olduğumuz bant genişliğinden daha az paket yolladığını aşağıdaki görselde görülmektedir:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Kuyruk olmadığı zamanda 559 datagram(veri paketi) düzensiz alındı. Kuyruk yapılandırılması olduğunda 1 datagram(veri paketi) düzensiz alındı.

metin içeren bir resim

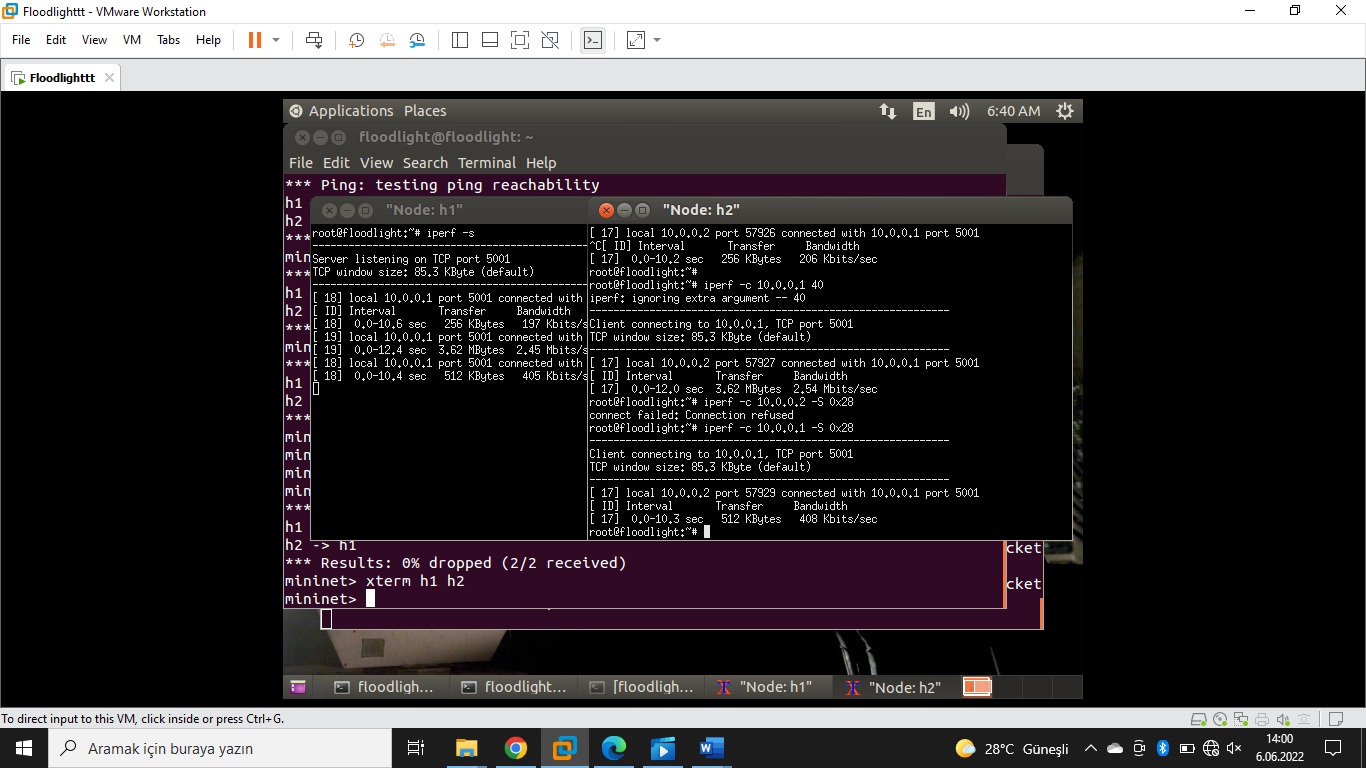
Açıklama otomatik olarak oluşturuldu

Meter eklenmemiş haliyle paket gönderildi ve drop olarak belirlenen 400 kbitsin üstündeki paketleri alabildiği görüldü.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Meter eklenerek denendiğinde ise istenildiği gibi 400 Kbits üzerine çıkılamamıştır.



Drop modu ile Meter’a ait önceden belirlenen bant genişliği miktarını aşan paketlerin düşürüldüğü görülmüştür. Bu durum belirli bir bant genişliğinin önceden tanımlanması halinde tanımlanan bu bant genişliği içerisinde gelen trafiğe ait paketlerin düşürülmeyeceğini göstermektedir. Böylece belirli bir oranda bant genişliği garantisi sağlanmış olmaktadır. Ayrıca Drop modu sayesinde ağ trafiğinde oluşabilecek aşırı yüklenmelerin engellendiği de ortaya konmuştur.

**KAYNAKLAR**

[1]OpenFlow Switch Specification

[2]<https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343544/Installation+Guide>

[3]<https://www.mediaclick.com.tr/tr/blog/sdn-nedir>

[4]<https://dergipark.org.tr/tr/download/article-file/934088>

[5]<http://mininet.org/overview/>

[6]<https://silo.tips/download/a-anahtarlarinda-openflow-protokol-ve-pox-denetleycs-kullanimi-seluk-yazar-yksek>

[7]ONF, Open Networking Foundation official web page, URL(erişim tarihi:12.12.2017) <https://www.opennetworking.org>.

[8] P. Göransson, C. Black ve T. Culver, Software Defined Networks A Comprehensive Approach 2. Edition, Elsevier Inc. Morgan Kaufmann, 2017.

[9] F. Hu, Network Innovation through Openflow and SDN Principles and Design, Boca Raton, FL, USA: CRC Press, Taylor & Francis Group, 2014..

[10] CPqD, OpenFlow Software Switch, URL(erişim tarihi: 5.02.2019) <https://cpqd.github.io/ofsoftSwitch13>

[11] E. L. Fernandes, E. Rojas, J. Alvarez-Horcajo, Z.L.Kis, D.Sanvito, N.Bonelli, C.Cascone ve C.E.Rothenberg, The Road to BOFUSS: The Basic OpenFlow User-space Software Switch, 21 January 2019. URL(erişim tarihi: 15.03.2019) <https://www.researchgate.net/publication/330553425>

[12] E. L. Fernandes, DPCTL OfSoftSwitch Yönetim Aracı, 10 September 2018. URL(erişim tarihi: 10.02.2019) <https://www.github.com/CPQD/ofsoftSwitch13/wiki/Dpctl-Documentation>.

[13]((Jun Xu, Member, IEEE, and Mukesh Singhal, Fellow, Cost-Effective Flow Table Designs for High-Speed Routers: Architecture and Performance Evaluation, IEEE TRANSACTIONS ON COMPUTERS, VOL. 51, NO. 9, s. 1090, 2002. ))

[14]((<https://docplayer.biz.tr/995001-Ag-anahtarlarinda-openflow-protokolu-ve-pox-denetleyicisi-kullanimi-selcuk-yazar-yuksek-lisans-tezi-bilgisayar-muhednisligi-ana-bilim-dali.html)>)

[15]<https://tr.wikipedia.org/wiki/QoS_(port_%C3%B6nceli%C4%9Fi)>

