T.C.
KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM TANIMLI AĞLARDA HİZMET KALİTESİ İÇİN OPENFLOW METER ÖZELLİKLERİNİN İNCELENMESİ VE UYGULAMASI

ALİ PAK

YÜKSEK LİSANS TEZİ

ENFORMATİK ANABİLİM DALI

KAHRAMANMARAŞ 2019

T.C.
KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM TANIMLI AĞLARDA HİZMET KALİTESİ İÇİN OPENFLOW METER ÖZELLİKLERİNİN İNCELENMESİ VE UYGULAMASI

ALİ PAK

Bu tez,
Enformatik Anabilim Dalında
YÜKSEK LİSANS
derecesi için hazırlanmıştır.

KAHRAMANMARAŞ 2019

Kahramanmaraş Sütçü İmam Üniversitesi Fen Bilimleri Enstitüsü öğrencisi Ali PAK tarafından hazırlanan "YAZILIM TANIMLI AĞLARDA HİZMET KALİTESİ İÇİN OPENFLOW METER ÖZELLİKLERİNİN İNCELENMESİ VE UYGULAMASI" adlı bu tez, jürimiz tarafından 27/06/2019 tarihinde oy birliği / oy çokluğu ile Enformatik Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Prof. Dr. İbrahim Taner OKUMUŞ (DANIŞMAN) ....…….....
Bilgisayar Mühendisliği
Kahramanmaraş Sütçü İmam Üniversitesi

Doç. Dr. Muhammet Fatih HASOĞLU (ÜYE) …….
Bilgisayar Mühendisliği
Hasan Kalyoncu Üniversitesi

Dr. Öğr. Üyesi Mücahid GÜNAY (ÜYE) ....…….....
Bilgisayar Mühendisliği
Kahramanmaraş Sütçü İmam Üniversitesi

Yukarıdaki imzaların adı geçen öğretim üyelerine ait olduğunu onaylarım.

Prof. Dr. Mustafa YAZICI …….
Fen Bilimleri Enstitüsü Müdürü

TEZ BİLDİRİMİ

Tez içerisindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada, alıntı yapılan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Ali PAK

Not: Bu tezde kullanılan özgün ve başka kaynaktan yapılan bildirişlerin, çizelge, şekil ve fotoğrafların kaynak gösterilmeden kullanımı, 5846 sayılı Fikir ve Sanat Eserleri Kanunundaki hükümlere tabidir.

YAZILIM TANIMLI AĞLARDA HİZMET KALİTESİ İÇİN OPENFLOW METER ÖZELLİKLERİNİN İNCELENMESİ VE UYGULAMASI (YÜKSEK LİSANS TEZİ)

ALİ PAK

ÖZET

Hizmet kalitesi, bilgisayar ağlarında farklı hizmet ihtiyaçlarına sahip uygulamalara kaynak tahsisi veya önceliklendirme yapılarak, kaynakların verimli bir şekilde kullanımını sağlamak için gereklidir. Geleneksel ağ mimarisine aykırı olarak Yazılım Tanımlı Ağlar programlanabilirlik konusunda sunduğu esneklik ve merkezi yapıdaki kontrolör sayesinde bütünsel bir hizmet kalitesi politikasının uygulanmasını kolaylaştırmaktadır.

Bu çalışmada Yazılım Tanımlı Ağlarda kontrolör ve ağ cihazları arasındaki iletişim kurallarını belirten OpenFlow protokolünün Meter özellikleri incelenmiş ve hizmet kalitesi yapılarının nasıl sağlanabileceği gösterilmiştir. Meter özelliklerinin uygulanabilmesi için kontrolör ve ağ cihazlarının OpenFlow1.3 protokolüyle uyumlu olması gerekmektedir. Bu yüzden çalışmamızda Floodlight kontrolör ve ofSoftSwitch kullanılmıştır. Floodlight kontrolöre Meter ve akış-kaydı modülleri eklenmiş ayrıca kontrolörde bulunan istatistik modülüne de güncellemeler yapılmıştır.

Mininet emulatöründe oluşturulan test ortamında iki uç arasında bir trafik oluşturularak bu trafiğin Floodlight kontrolör tarafından yönetilmesi sağlanmıştır. İstatistik modülü aracılığıyla elde edilen sonuçlara bakıldığında Meter modlarından dscp_remark ile paketlerin düşürülme önceliklerinin değiştirilerek trafik önceliklendirmesi yapıldığı, Drop modu ile paketlere belirli bir oranda bant genişliği garantisinin sağlandığı ayrıca DSCP bitleri kullanılarak trafik bölmelendirmesi işleminin yapılabildiği gözlenmiştir. Sonuç olarak OpenFlow Meter özellikleri kullanılarak hizmet kalitesi yapılarının oluşturulabildiği ortaya konmuştur. Bu özelliklerin hizmet kalitesi destekli yönlendirme algoritmaları ile daha verimli çalışacağı öngörülmektedir.

Anahtar Kelimeler: Hizmet Kalitesi, Yazılım Tanımlı Ağlar, OpenFlow1.3, Meter, Floodlight Kontrolör

Kahramanmaraş Sütçü İmam Üniversitesi
Fen Bilimleri Enstitüsü
Enformatik Anabilim Dalı, Haziran / 2019

Danışman : Prof. Dr. İbrahim Taner OKUMUŞ
Sayfa sayısı: 49

AN ANALYSIS AND IMPLEMENTATION OF OPENFLOW METER SPECIFICATIONS FOR QUALITY OF SERVICE PURPOSES IN THE SOFTWARE DEFINED NETWORKING
(M.Sc. THESIS)

ALİ PAK

ABSTRACT

Quality of Service is required to ensure efficient use of resources, by providing bandwidth reservation or prioritizing network traffics, for the applications with different service needs in the computer networks. In contrast to the traditional network architecture, Software-Defined-Networks facilitate the implementation of a holistic Quality-of-Service policy with its features such as flexibility in terms of programmability and the centralized controller

In this study, Meter features of OpenFlow protocol, specifying the communication rules between the controller and the network devices in SDN, has been examined and how to provide the structure of the Quality of Service has been demonstrated. In order to use the Meter features, controllers and network devices must be compatible with the OpenFlow1.3 protocol. Therefore, the Floodlight controller and the ofSoftSwitch have been used in this study. Meter and flowSetter modules have been added to the Floodlight controller, also statistics module has been updated.

In the test environment, created by Mininet, it is provided that the traffic has been established between two Hosts and this traffic has been managed by the Floodlight controller. According to the results, obtained from the statistics module, it has been observed that traffic prioritization made by changing the drop precedence levels of the packages with the Dscp_remark mode, a certain amount of bandwidth guarantee provided with the Drop mode. Moreover, traffic splitting can be performed by using DSCP bits. As a result, this study has demonstrated that the QoS structures can be created by using OpenFlow Meter features in the SDN environment. It is estimated that these features will work more efficiently with the QoS supported routing algorithms.

Keywords: Quality of Service, Software Defined Networking, OpenFlow Protocol v.1.3, Meter, Floodlight Controller

Kahramanmaraş Sütçü İmam University
Institute for Graduate Studies in Science and Technology
Department of Informatics, June / 2019

Supervisor : Prof. Dr. İbrahim Taner OKUMUŞ
Page number: 49

TEŞEKKÜR

Öncelikle aldığımız her nefesin sahibine ilim yolunda bizlere nasip ettiği istek, çalışma gücü ve başarılar için sonsuz hamdımı belirtmek isterim.

Yüksek lisans çalışma konumun belirlenmesinden bugüne kadar geçen sürecin her aşamasında ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübeleriyle beni yönlendiren ve geleceğe dair umutlarıma ilham olan sayın hocam Prof. Dr. İbrahim Taner OKUMUŞ'a sonsuz teşekkürlerimi sunarım.

ALİ PAK

HAZİRAN 2019

KAHRAMANMARAŞ / TÜRKİYE

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

Sayfa No.

TABLOLAR DİZİNİ

Sayfa No.

SİMGELER ve KISALTMALAR DİZİNİ

DiffServ : Differentiated Services – Farklılaştırılmış Hizmetler

DSCP : Differentiated Service Code Points- Farklılaştırılmış Hizmet Kodu Noktaları

IntServ : Integrated Services – Birleştirilmiş Hizmetler

NOS : Network Operating System – Ağ İşletim Sistemi

ONF : Open Network Foundation

QoS : Quality of Service – Hizmet Kalitesi

SDN : Software Defined Network- Yazılım Tanımlı Ağ

TOS : Type of Services – Hizmet Türleri

1. GİRİŞ

İnternet erişilebilirliğinin ve kullanım alanlarının artmasıyla birlikte internet kullanıcı sayısı 4.43 milyar seviyelerine ulaşmıştır [1]. Endüstri 4.0 devrimiyle hayatımıza girmeye başlayan Nesnelerin İnterneti (IoT - Internet of Things) kavramı ile birlikte, bilgisayar ve telefon gibi internete bağlanmamızı sağlayan cihazlardan bağımsız olarak, sensör vb. cihazların da internet ağına dahil olmaya başladıkları görülmektedir. Ayrıca firmaların büyük çoğunluğunun mobiliteye çok fazla önem verdiği ve uygulamalarını bulut veri tabanına göre geliştirdiği gerçeği göz önüne alındığında geleneksel ağ mimarisinin esnek olmayan yapısından dolayı internet üzerinde oluşacak bu yoğunluğu yönetebilmesi ve ortaya çıkacak yeni ihtiyaçları giderebilmesi pek mümkün görünmemektedir.

Günümüz internet ağlarının merkezinde internet protokolü bulunmaktadır. İnternet protokolü gönderilen paketlerin hedef adreslerine başarılı bir şekilde ulaştırılacağına dair bir garanti sunmamaktadır. Data uygulamaları ise gönderilen paketlerin alıcıya güvenli bir şekilde ulaştığından emin oldukları bir iletim ortamına ihtiyaç duymaktadır. Bu yüzden TCP (Transmission Control Protocol) protokolü internet protokolü ile birlikte çalışan en önemli protokoldür. Bu protokolde gönderilen paketin alıcıya ulaşmadığı durumlarda paketin tekrar gönderilmesi sağlanmaktadır. Böylelikle data paketlerinin alıcıya ulaştığından emin olunmaktadır.

Gerçek zamanlı uygulamalarda devam eden bir olaya ait görüntü ve sesin doğru sırayla alıcıya ulaşması gerekmektedir. Bu yüzden paketlerin iletimi sırasında paket kaybı, gecikme, gecikme farklılığı gibi durumlarda TCP protokolüne göre alıcının paketin ulaşmadığını bildirmesi ve göndericinin ise paketleri tekrar göndermesi gerekmektedir. Video konferans, broadcasting, ses iletişimi gibi gerçek zamanlı uygulamalarda ise paketlerin tekrar gönderilmesi kullanışlı bir yaklaşım değildir. Bu yüzden gerçek zamanlı uygulamalarda UDP (User Datagram Protocol) protokolü kullanılmaktadır. UDP'de paketlerin tekrar gönderilmesi yoktur ve paketin iletimi için herhangi bir garanti verilmemektedir.

Geleneksel ağ mimarisinde gerçek zamanlı uygulamalara ve data uygulamalarına aynı öncelikte hizmet sağlayan Best Effort yaklaşımı vardır. Best Effort modeli herhangi bir hizmet kalitesi garanti etmemektedir. Best Effort yaklaşımının aksine gerçek zamanlı çalışan uygulamalar yüksek seviyeli bir hizmet kalitesi ile garanti edilmeye ihtiyaç duymaktadırlar. Bu noktada geleneksel ağ mimarisinin yerine kullanılamaya başlanan Yazılım Tanımlı Ağlar (YTA)

(SDN - Software Defined Networking) gerçek zamanlı uygulamalara ait trafiklere hizmet önceliği konusunda iyi bir çözüm olarak karşımıza çıkmaktadır.

YTA'da kontrol düzleminde yer alan kontrolör ile iletim düzleminde yer alan network cihazları arasındaki iletişim *Southbound Communication* olarak adlandırılır. Bu iletişime ait kurallar ise yaygın olarak OpenFlow protokolü ile belirtilir. ONF (Open Networking Foundation) tarafından yayımlanan OpenFlow1.3 protokolü ile kullanılmaya başlanılan Meter özelliği, YTA'da basit ya da karmaşık seviyede hizmet kalitesi yapılarının oluşturulmasına olanak sağlamaktadır [2].

Hizmet kalitesi (QoS - Quality of Service) ise ağ üzerindeki uygulamalara ya da veri türlerine kaynak tahsisi veya önceliklendirme yapılarak ihtiyaç duyulan bant genişliğinin paket kaybı, gecikme ve gecikme farkı gibi kısıtlar altında kullanılabilmesini hedefleyen bir yaklaşım olarak ifade edilebilir.

Hizmet kalitesi için IntServ (Integrated Service) ve DiffServ (Differentiated Service) olmak üzere iki temel yaklaşım bulunmaktadır. IntServ modelinde kaynak rezervasyonu, DiffServ modelinde ise hem kaynak rezervasyonu hem de paketlerin farklı sınıflara ayrılarak önceliklendirilmesi ile hizmet kalitesi yapıları oluşturulmaktadır.

DiffServ modelinin kullanıldığı bu çalışmada, OpenFlow1.3 protokolüyle kullanılmaya başlanılan Meter özellikleri hizmet kalitesi yapıları oluşturmak amacıyla kullanılmıştır. Ayrıca IPv4 (Internet Protocol Version 4) başlık kısmında yer alan DSCP (Differentiated Service Code Point) bitleri kullanılarak trafik sınıflandırma işlemi de yapılmıştır.

1.1 İlgili Çalışmalar

Geleneksel yöntemlerle hizmet kalitesi IntServ modeli kullanılarak sunulmaktaydı [3]. IntServ yaklaşımında ilk önce uygulamalar ağdan kaynak talebinde bulunurlar. Paketin alıcıya giderken izleyeceği yol boyunca kullanılacak olan bant genişliğinin garanti edilmesinden sonra uçlar arasındaki veri transferi gerçekleştirilir. Büyük ölçekli ağlarda her bir akış için bu işlemlerin ayrı ayrı yapılması gerektiğinden dolayı bu yaklaşımda ölçeklenebilirlik problemiyle karşılaşılmaktadır. IntServ, ölçeklenebilirlik ve karmaşık problemlerin çözümü noktasında ihtiyaçları karşılayamadığı için DiffServ modeli ortaya konmuştur [4]. DiffServ modeli, switch'e gelen paketlere kuyruk önceliği tanıyarak ya da paketlerin farklı kuyruklara yönlendirilmesini sağlayarak bant genişliği garantisi ya da trafik önceliklendirmesi yapılmasına olanak sağlamaktadır.

Trafik yönlendirme işlemi sırasında akışa ait önceden belirlenmiş bant genişliği değerini aşmayan paketler bir kuyruğa, aşan paketler farklı bir kuyruğa yönlendirilerek trafiğin önceden belirlenen akış kurallarına uygun olan kısmının belirlenen hizmet kalitesini alması sağlanırken uymayan kısmının daha düşük bir seviyeden hizmet kalitesi alması sağlanabilmektedir [5].

Geleneksel ağ yapılarında bant genişliği kontrolü kuyruklar aracılığı ile sağlanırken 2012 yılında ONF tarafından yayınlanan OpenFlow1.3 protokolü ile bant genişliği kontrolü Meter tablolarıyla da ilişkilendirilmiştir [2]. Meter'lar, Rate-Limiter (Miktar sınırlandırıcı) ya da DiffServ şeklinde hizmet kalitesi yapıları oluşturmak için kullanılabilmektedirler.

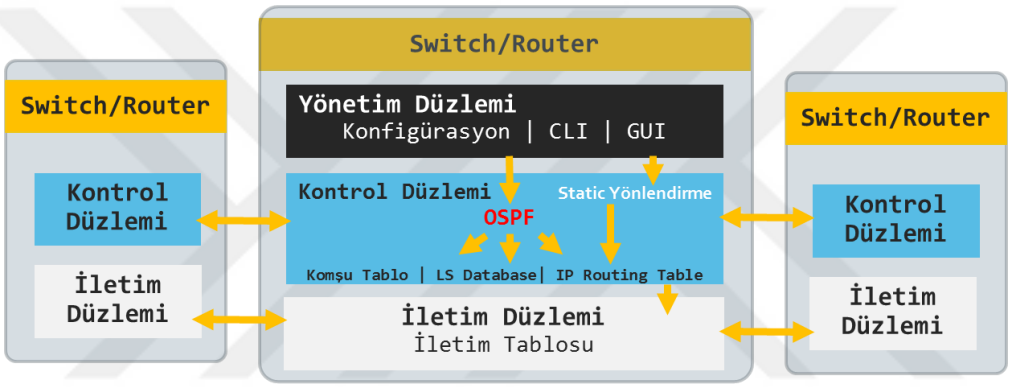
Eş zamanlı (Real-Time) akış gerektiren VoIP (Voice over Internet Protocol), Video konferans, VoD (Video on Demand) gibi uygulamalarda hizmet kalitesi sunulabilmesi için bant genişliği rezervasyonu gerekmektedir. Ağda bulunan her bir bağlantı için eşit bant genişliği rezervasyonu yapmak yerine o bağlantıdaki kullanılabilir bant genişliği miktarınca bant genişliği rezervasyonu yapıldığında ağdaki yük dengesi sağlanmış olacaktır [6].

OFMAQ algoritmasının kullanıldığı bir çalışmada multimedya trafikler için hizmet kalitesi oluşturulurken, gereken tahmini bant genişliği, p(t), değeri kullanılarak Meter bant genişliği ayarlanmaktadır. Daha sonra OpenFlow protokolünü destekleyen switch'ler tarafından ölçülen multimedya paketlerin kullandığı bant genişliği miktarı, r(t), ile tahmini bant genişliği karşılaştırılmaktadır. Karşılaştırma neticesinde eğer p(t) <= r(t) ise multimedya paketleri için ayrılan bant genişliği miktarı arttırılmaktadır. Aksi durumda bant genişliği azaltılmaktadır. Tıkanıklık durumlarında ise geleneksel yeniden yönlendirme algoritmaları yerine NDSR algoritması kullanılarak düşük öncelikli paketlerin başka rotalardan gitmesi sağlanırken multimedya paketlerinin kendi rotasında devam etmesi sağlanmıştır [7].

DiffServ hizmet seviyesinde anlaşmalar (SLA - Service Level Agreement) ile akışların trafik profilleri ve alacakları hizmet kalitesi sınıfı belirlenir. Akış esnasında trafik profiline uymayan paketler düşürülür ya da daha düşük bir seviyeden hizmet alabilmesi için yeniden işaretlenir. Meter'lar, hizmet kalitesi amacının dışında, Dscp_remark modu sayesinde trafik bölmelendirmesi amacıyla da kullanılabilmektedir. IPv4 başlık kısmında yer alan DSCP bitlerinin yeniden işaretlenmesi ile her bir paketin daha önceden belirlenen akış kurallarına göre farklı portlardan çıkış yapması sağlanmıştır [8] .

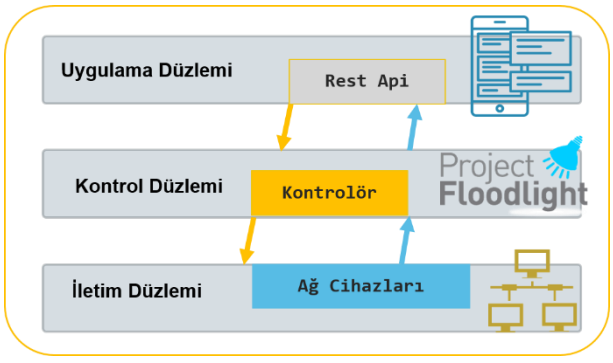
2. YAZILIM TANIMLI AĞLAR

Ağ cihazlarında genel olarak kontrol ve iletim düzlemleri bulunmaktadır. Kontrol düzlemi ağdaki cihazların yönetiminden, iletim düzlemi ise dataların ağda bulunan cihazlar arasındaki iletiminden sorumludur. Geleneksel ağ mimarisinde kontrol düzlemi ve iletim düzlemi bütünleşik halde çalışmaktadır (Şekil 2.1). Kapalı sistem olarak tanımlayabileceğimiz bu yapıda, kontrol düzleminde yer alan yazılımlara müdahale edilmesi gereken durumlarda ağda yer alan bütün cihazlarda aynı değişikliğin yapılması gerekmektedir. Sistemin tekrar çalışır hale gelmesi sürecinde zaman kaybı, maliyet artışı, veri kaybı, prestij kaybı gibi durumlar söz konusu olabilmektedir. Dolayısıyla ağ yapısının işleyişi ile ilgili herhangi bir değişiklik yapılması oldukça güç bir durumdur.



Şekil 2.1 Geleneksel ağ mimarisi

Yazılım Tanımlı Ağlar kontrol düzlemi ile iletim düzleminin fiziksel olarak birbirinden ayrıldığı ve kontrol düzleminin birden fazla cihazı yönetebildiği yapılar olarak tanımlanmaktadır [9]. YTA mimarisindeki bu yapı sayesinde düzlemler arasında herkesin erişebileceği arayüzler geliştirilebilecek ve kontrol düzleminin programlanabilirlik anlamındaki esnekliği arttırılmış olacaktır. Şekil 2.2'de YTA mimarisi görülmektedir.



Şekil 2.2 Yazılım-Tanımlı-Ağ mimarisi

YTA mimarisinde uygulama, kontrol ve iletim (data) düzlemleri yer almaktadır. Uygulama düzleminde bulunan uygulamaların kullanımıyla birlikte yapılandırma, yönetim, güvenlik, ağ kaynaklarının hızlı ve dinamik olarak optimize edilmesi gibi konularda kolaylık sağlanması hedeflenmektedir. Uygulamalar, kontrolörden ağın durumu hakkında raporlar alabileceği gibi göndereceği komutlar ile de ağın yönetimine müdahale edebilmektedir.

Kontrol düzleminde bulunan network işletim sistemi (NOS - Network Operating System) ya da kontrolör olarak ifade edilen yapılar ağın yönetiminden ve kontrolünden sorumludurlar. Ayrıca kontrolörler, uygulamalardan aldıkları bilgileri iletim düzlemindeki ağ cihazlarında bulunan akış-tablolarına (Flow Table) bir akış-kaydı (Flow Entry) olarak ekleyerek paketlerin bu kayıtlara göre portlar arasında yönlendirilmesini sağlarlar.

İletim düzlemi ise kontrolörlerden aldığı bilgilere göre ağ içerisindeki yönlendirme işlemlerinin gerçekleştirilmesinden sorumludur.

Ağ işlemleri sırasında kullanılan standartların, farklı üreticilerin farklı cihazları ya da üreticilerin kullandıkları farklı protokoller yerine, ONF tarafından belirlenmesi ile ağ tasarımı ve işleyişinin daha basit hale getirilmesi hedeflenmektedir. YTA mimarisi, bir çok geliştiricinin yanı sıra ağ cihazları üreten firmalar tarafından da desteklenmeye ve geliştirilmeye devam etmektedir. Bu çalışmalar ilerlediğinde ve yaygınlaştığında geleneksel ağ mimarisi, yerini esnek, programlanabilen ve merkezi bir kontrolöre sahip olan YTA mimarisine bırakmış olacaktır.

2.1 Kontrol Düzlemi

YTA'da kontrolörler, uygulama düzlemi ile kontrol düzlemi ve kontrol düzlemi ile iletim düzlemi arasındaki iletişimin sağlanmasının yanı sıra ağdaki yönetim işlemlerinin yapılmasından da sorumludurlar. YTA'da kontrolör ile ağ cihazları arasındaki iletişim *Southbound iletişim* olarak adlandırılır. Bu iletişime ait kurallar yaygın olarak OpenFlow protokolü ile belirtilir. Bu yüzden *Southbound iletişiminin* gerçekleştirilebilmesi için OpenFlow protokolünü destekleyen kontrolör ve ağ cihazlarının kullanılması gerekmektedir.

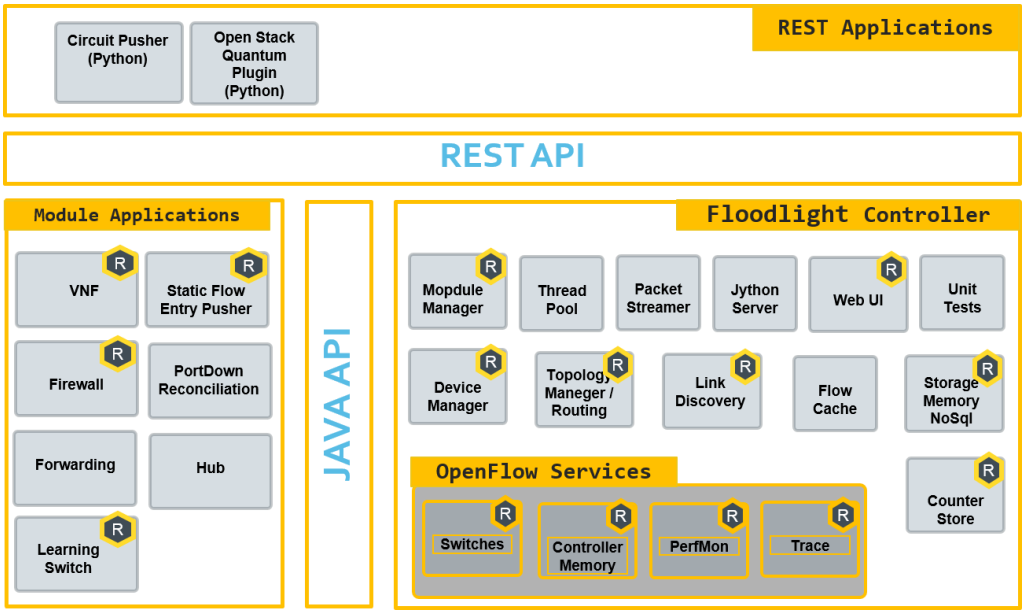
Bu çalışmada OpenFlow protokolünün Meter özellikleri kullanılarak hizmet kalitesi yapılarının oluşturulması hedeflenmektedir. Meter özellikleri ise OpenFlow1.3 versiyonu ile desteklenmeye başlanmıştır [2].

Kontrol düzleminde Nox, Beacon, Ryu, Floodlight, OpenDaylight, Onos gibi kontrolörler kullanılarak ağın yönetimi sağlanmaktadır. Bu kontrolörlerin birçoğu açık kaynak kod ile geliştirilmekte ve OpenFlow protokollerini desteklemektedirler. Bu çalışmada OpenFlow1.3 versiyonunu destekleyen, modüler yapısı sayesinde modül yükleme, modül düzenleme gibi işlemlere olanak tanıyan ve hem gerçek switch'lerde hem de sanal switch'lerde çalışabilen, Floodlight kontrolör kullanılmıştır [10].

2.1.1 Floodlight kontrolör

Floodlight, Java diliyle yazılmış ve otonom olarak çalışabilen modüllerden oluşan bir yapıya sahiptir. Bu modüller sayesinde Floodlight, OpenFlow protokolüne uygun olarak, YTA mimarisinde yer alan yapılar arasındaki iletişim ve yönetim işlemlerini başarıyla gerçekleştirebilmektedir.

Floodlight, diğer OpenFlow kontrolörlerden farklı olarak, kontrolör ile birlikte Java modülleri ve Rest Api uygulamalarını da içerisinde barındıran bir yapıya sahiptir. Floodlight kontrolör bir ağı izlemek ve kontrol etmek için ortak işlevler kullanırken Rest Api uygulamaları ise farklı kullanıcı ihtiyaçlarını karşılayabilmek için farklı özelliklerin çalıştırılmasını sağlamaktadırlar. Şekil 2.3'te Floodlight kontrolör, Java modülleri ve Rest Api uygulamaları arasındaki ilişki gösterilmiştir.



Şekil 2.3 Floodlight kontrolör

Floodlight kontrolör başlatıldığında, kontrolör ve Java modülleri birlikte çalışmaya başlamaktadırlar. Rest Api'ler ise kendilerine tahsis edilen 8080 portu üzerinden kontrolöre *http REST* mesajı göndererek modüllerden bilgi alabilir veya herhangi bir hizmetin yönetimine müdahale edebilirler [11].

2.1.1.1 Floodlight kurulumu

Floodlight kurulumu için ön koşul olarak Linux işletim sistemi, JDK (Java Development Kit), Ant/Maven programları, python geliştirme paketi ve Eclipse IDE'nin (Java geliştirme ortamı) bilgisayarınızda kurulu olması gerekmektedir [12]. Daha sonra Floodlight kontrolör aşağıdaki kod dizini yardımı ile GitHub platformundan indirilerek kurulum işlemleri yapılabilir.

```
$ git clone git://github.com/floodlight/floodlight.git
$ cd floodlight
$ git submodule init
$ git submodule update
$ ant
$ sudo mkdir /var/lib/floodlight
$ sudo chmod 777 /var/lib/floodlight
```

İndirilen ve kurulumu gerçekleştirilen Floodlight kontrolör aşağıdaki komut satırı aracılığıyla çalıştırılabilir.

```
$ java -jar target/floodlight.jar
```

2.1.1.2 Floodlight modül ekleme

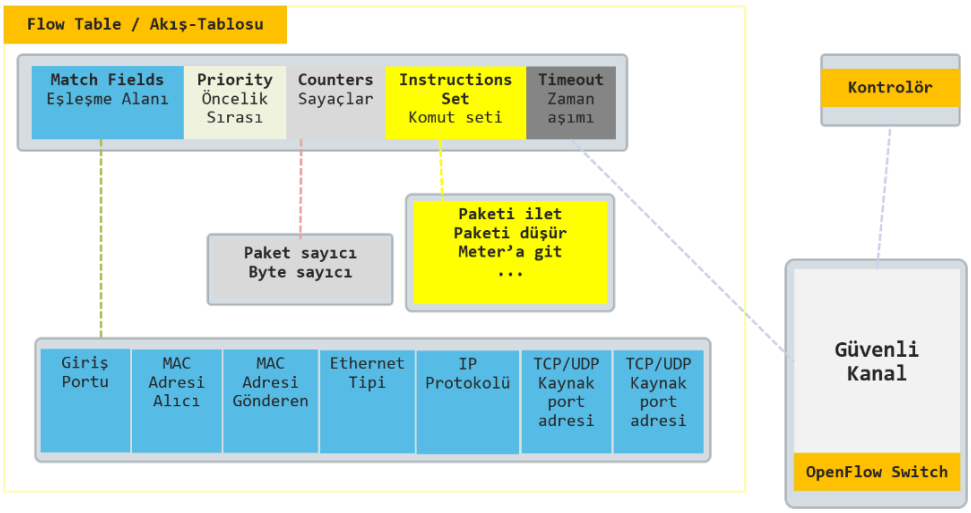
Aşağıdaki işlem basamakları kullanılarak Eclipse IDE platformunda Floodlight kontrolöre modül ekleme işlemleri gerçekleştirilebilir [13] .

- Paket gösterici kısmından "*floodlight*" seçeneğini genişleterek "*src/main/java*" klasörünü bulunuz.
- "*src/main/java*" klasörüne sağ tıklayarak "*New/Class*" seçeneğine tıklayınız.
- Paket kısmına "*net.floodlightcontroller.paketAdiniz*" ifadesini giriniz.
- İsim alanına *modulAdınızı* giriniz.
- Arayüzler(Interfaces) sekmesinin yanındaki ekle (add) butonuna tıklayınız.
- Modülünüzün etkileşime geçeceği arayüzleri seçerek ok butonuna basınız.
  (ör. "*IOFMessageListener*" , "*IFloodlightModule*")
- Diyalog penceresinden tamamı seçiniz.

- Modül kayıt işlemi için *src/main/resources/META-INF/services/net.Floodlight controller.core.module.IfloodlightModule* dosyasını açın ve *net.floodlightcontroller. paketAdiniz.modulAdiniz* satırını ekleyiniz.
- Modülün kontrolör başlatıldığında yüklenmesi için *src/main/resources/floodlightdefault.properties* dosyasını açın ve listede yer alan diğer modül isimlerine dokunmadan *net.floodlightcontroller.paketAdiniz.modulAdiniz* ifadesini en alt kısma ekleyiniz.

2.1.1.3 Floodlight - Akış tablosu ilişkisi

Switch'ler, içerilerinde bulunan akış-tabloları sayesinde kendilerine gelen paketlere ne yapılacağına dair karar verebilmektedirler (Şekil 2.4). Paket geldiğinde öncelik (priority) sırasına dikkat edilerek eşleşme (match) kriterlerine bakılır. Paketteki eşleşme değerleri ile akış-tablosundaki eşleşme değerleri uyuşuyorsa eylemler (actions/instructions) kısmında yer alan işlemler gerçekleştirilir. Bu işlemler paketi ilet, sil, güncelle vb. olabilir. Switch'e akış-tablolarında kaydı olmayan bir paket geldiğinde ise paket direkt olarak kontrolöre gönderilir. Kontrolör ağda bulunan switch'lerle güvenli kanal (Secure Channel) aracılığıyla bir TCP bağlantısı oluşturur. Eğer ki ağa sonradan eklenen bir switch için yeni bir TCP bağlantısı oluşturulmuşsa kontrolör bu bağlantı ile ilgili yeni akış-kaydını oluşturarak akış-tablosuna ekler.



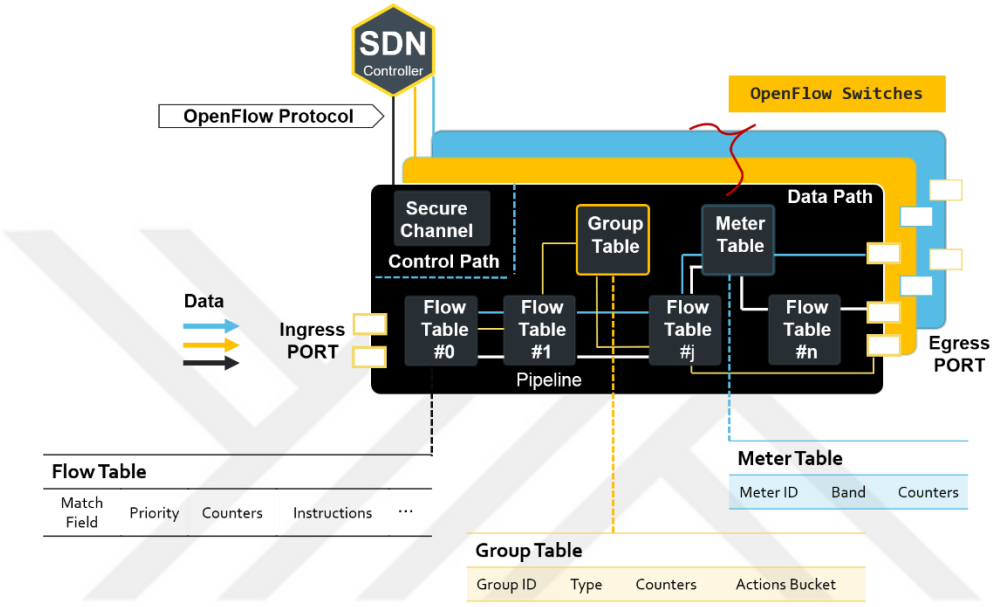
Şekil 2.4 Akış-Tablosu

2.1.2 OpenFlow protokolü

OpenFlow, YTA'da kontrolör ile iletim düzlemi arasındaki iletişime ait standartları belirten bir protokoldür. Bu protokol herhangi bir üreticiden veya üreticiye ait cihazlardan bağımsız olarak ONF tarafından belirlenmiştir. Bu protokolün kullanılmasındaki temel amaç kontrolör

ve ağ cihazları arasındaki iletişimi standartlaştırmak ve ileride OpenFlow standartlarına uygun cihazların geliştirilmesidir [14].

Southbound iletişimin gerçekleşebilmesi için kontrolör ve ağ cihazlarının OpenFlow protokolünü tanıması gerekmektedir. Dolayısıyla bu iletişim sadece OpenFlow kurallarına göre tasarlanmış kontrolör ve OpenFlow destekleyen ağ cihazları ile yapılabilmektedir. Şekil 2.5 OpenFlow Switch'lerin iç yapısını göstermektedir.



Şekil 2.5 OpenFlow switch

OpenFlow destekleyen switch'lerde, gelen paketlerin nasıl yönlendirileceğine ait bilgileri tutan *Akış-Tabloları*, kontrolör ile iletişim kurmak için kullanılan *Güvenli Kanal*, hizmet kalitesi sağlamak için kullanılan *Meter Tablosu* (Meter Table) ve gerçek zamanlı uygulamalar (Multicasting / Broadcasting) için kullanılan *Grup Tablosu* bulunmaktadır.

Kontrolör ile switch, bir IP adresi ve önceden belirlenmiş bir port (varsayılan: 6653) aracılığıyla güvenli kanal üzerinden bağlantı kurmak zorundadırlar. Bu iletişim TCP bağlantısıyla yapılabileceği gibi TLS (Transport Layer Security) tabanlı asimetrik şifreleme ile de yapılabilmektedir [15] .

Akış-tabloları, her bir akışa ait eşleşme alanları, sayaçlar, eylemler, komutlar, çıkış portları gibi kayıtları tutmaktadır. OpenFlow'u destekleyen switch'ler, kendilerine gelen paketleri nasıl yönlendireceklerine dair bilgileri akış-tablolarından alırlar. Eğer ki akış tablolarında switch'e gelen pakete ait bir kayıt yoksa (Table Miss) güvenli kanal aracılığıyla kontrolör ile iletişime

geçilir. Kontrolör paketin nasıl yönlendirileceğine dair kararı verir ve akış-tablosuna yeni bir kayıt eklenmesini sağlar.

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].

2.1.2.2 Kontrolörden switch'e mesajlar

Kontrolörün direkt olarak switch'leri yönetmek için kullandığı mesajlardır. Bu kategoride; Features, Configuration, Modify-State, Read-State, Packet-Out, gibi mesajlar yer almaktadır.

- Features: Kontrolörün switch'in yeterlilikleri hakkında bilgi almasını sağlayan mesajlardır. Kontrolör kendisine bağlanmak isteyen her cihazdan bu bilgiyi ister.
- Configuration: Kontrolörün switch'e yapılandırma bildirimi yaptığı mesajlardır.
- Modify-State: Kontrolörün switch'in akış-tablosuna ve portlarına müdahale etmesini sağlayan mesajlardır.
- Read-State: Kontrolörün switch'ten istatistiksel verileri veya port bilgilerini almasını sağlayan mesajlardır.
- Packet-Out: Kontrolörün kendisine gelen packet-in mesajlarına cevap olarak belirli bir port üzerinden switch'lere gönderdiği mesajdır.

2.1.2.2 Asenkron mesajlar

Kontrolörden herhangi bir talep gelmesine bakılmaksızın switch tarafından gönderilen mesajlardır. Bu mesajlar paketlerin switch'e ulaşıp ulaşmadığı, switch'in durumundaki değişiklikler ya da meydana gelen bir hata nedeniyle kontrolörü bilgilendirmek amacıyla gönderilebilirler. Asenkron mesajlar; Packet-in, Flow-remowed, Port-status, Error olmak üzere 4 alt kategoride incelenebilirler.

- Packet-in: Akış-tablosunda kaydı bulunmayan yeni bir paket geldiğinde switch'in bu paketle ilgili olarak kontrolöre gönderdiği mesajdır.

- Flow-removed: Akış-tablosu içerisinde artık kullanılmayan veya süresi geçmiş akışların durumlarının kontrolöre bildirilmesi için kullanılan mesajlardır.
- Port status: Portların durumlarındaki değişikliklerin kontrolöre bildirilmesinde kullanılır.
- Error: Switch üzerinde meydana gelen problemlerin kontrolöre iletilmesi için kullanılan mesajlardır.

2.1.2.3 Simetrik mesajlar

Simetrik mesajlar kontrolör ya da switch tarafından herhangi bir talep gelmesine gerek duyulmadan gönderilebilirler. Bu kategorideki mesajlar; Hello, Echo, Vendor/Experimenter olmak üzere 3 grupta incelenebilirler.

- Hello**:** Switch ile kontrolör arasında bağlantı kurulduğunda ilk gönderilen mesajdır.
- Echo**:** Kontrolör ile switch arasındaki bağlantının aktif olup olmadığının anlaşılması için kullanılır. Ayrıca bant genişliği ya da gecikmenin tespiti amacıyla da kullanılır.
- Vendor**:** Gelecekteki OpenFlow revizyonları için kullanılacak olan mesaj türüdür.

2.1.3 Meter tablosu

Meter'lar kendilerine yönlendirilen paketlerin miktarını ölçebilen ve bu paketleri kontrol edebilen switch elemanlarıdır. Meter'lar belirlenen bant genişliği oranının aşılması durumunda bant türüne göre paketlerin düşürülmesini veya yeniden işaretlenmesini sağlarlar. Meter'larda yer alan bant türlerinden Drop, bant genişliği oranını aşan paketlerin düşürülmesini sağlarken, Dscp_remark ise IPv4 paketlerinin başlık kısımlarında yer alan DSCP bitlerinin yeniden işaretlenmesini sağlayarak paketlerin düşürülme önceliklerini arttırmaktadır.

Meter'lar aynı zamanda kendilerine gelen paketlere ait akış sayısı, paket sayısı, byte miktarı, Meter'ın çalışma süresi gibi istatistiki bilgileri de kaydetmektedirler. Ayrıca Meter'da bulunan her bir bant için, gelen paket sayısı ve byte miktarı istatistikleri de elde edilebilmektedir.

Meter'ların kullanılabilmesi için öncelikle bu özelliği destekleyen switch'lere kurulmaları daha sonra ise switch'e gelen paketlerle ilişkilendirilmeleri gerekmektedir. Akış-tablolarında her bir akış-kaydına ait bir komut kümesi (instruction set) bulunmaktadır. Gelen paketlerin, akış-kayıtlarından herhangi biri ile eşleşmesi durumunda, komut setlerinde yer alan goto_*Meter(Meter_id)* komutuyla istenen Meter'dan geçmesi sağlanır. Böylelikle Meter'lar ile paketler ilişkilendirilirler.

Meter tablosu switch'lere kurulan ve akış-kayıtları aracılığıyla paketlerle ilişkilendirilen Meter'ların listesinin tutulduğu yerdir.

2.1.3.1 Meter tablosu temel yapısı

Tablo 2.1 Meter tablosu temel yapısı

Meter Identifier	Meter Bands	Counters

Meter Identifier

Meter'ı benzersiz bir şekilde tanımlamak için kullanılan 32 bitlik işaretsiz sayısal değerdir.

Meter Bands

Meter'ın icra edeceği fonksiyona dair özelliklerin belirtildiği alandır. Her Meter'da bir ya da birden fazla Meter Band bulunabilir. Meter Band'lar kendilerine ait belirtilen sınır değer (band rate) aşıldığında işlem yapmaya başlamaktadırlar. Paketler, Meter Band'lardan sadece biri tarafından işleme alınırlar. Meterlar kendilerine gelen o anki paket miktarından düşük olan en yüksek sınır değere sahip Meter Band'ın uygulanmasını sağlarlar. Eğer o anki paket miktarı sınır değerlerinden daha düşükse, hiç bir Meter Band uygulanmaz. Tablo 2.2'de Meter Band temel yapısı gösterilmektedir.

Tablo 2.2 Meter-Band temel yapısı

Band Type	Rate	Counters	Type Specific Arguments

- Band Type: Paketlere Drop veya Dscp_remark işlemlerinden hangisinin uygulanacağını belirtir.
- Rate: Meter tarafından Meter Band seçimi için kullanılmaktadır. Meter Band'ın işlem yapmaya başlayacağı en düşük değeri belirtir.
- Counters: Meter Band'a paketler geldikçe, gelen paket sayısı ve byte miktarı bilgilerinin tutulduğu sayaçlardır.

Tablo 2.3'te Meter Band 'ta bulunan sayaçlar gösterilmektedir.

Tablo 2.3 Meter-Band ‘ta bulunan sayaçlar

Sayaç Adı	Bit	Seçmeli / Zorunlu
In band packet count	64	s
In band byte count	64	s

- Type Specific Arguments**:** Bazı band türlerine ait opsiyonel özelliklerin belirtilmesi için kullanılır. Örneğin, Dscp_remark band türü tanımlanırken prec_level değeri bu şekilde tanımlanmalıdır.

Counters

Meter’a gelen bütün paketlere ait akış sayısı, paket sayısı, paket miktarı ve Meter’ın aktif olma süresi gibi bilgiler istatistik amaçlı olarak bu sayaçlarda tutulur. Tablo 2.4’te Meter sayaç tablosu verilmiştir.

Tablo 2.4 Meter'da bulunan sayaçlar

Sayaç adı	Bit	Seçmeli / Zorunlu
Flow Count	32	S
Input Packet	64	S
Input Byte	64	S
Duration Second	32	Z
Duration NanoSecond	32	S

2.1.3.2 Meter modifikasyon işlemleri

Meter’ların modifikasyon işlemleri kontrolör tarafından OFPT_METER_MOD mesajı ile yapılmaktadır [2]. Tablo 2.5 Meter modifikasyon mesaj yapısını göstermektedir.

Tablo 2.5 Meter modifikasyon mesaj paketi

HEADER	
COMMAND (16 Bit)	FLAGS (16 Bit)
METER_ID (32 Bit)	
METER_BAND_HEADER	

Meter modifikasyon (Meter-mod) mesaj paketinde kullanılan parametreler ve aldıkları değerler Tablo 2.6’da gösterilmiştir.

Tablo 2.6 Meter modifikasyon mesaj paketi parametreleri

ALAN	PARAMETRE	DEĞER
COMMAND	ADD	0
	MODIFY	1
	DELETE	2
FLAGS	KBPS	1
	PKPS	2
	BURST	4
	STATS	8
METER_ID	GENERAL METER ID	0X0000...0XFFFF0000
	SLOW DATAPATH	0XFFFFFFFd
	CONTROLLER	0XFFFFFFFE
	AU	0XFFFFFFFF

Command

Meter ekleme, düzenleme ve silme işlemlerinin yapılması için kullanılan alandır. Bu alana OFPMC_ADD, OFPMC_MODIFY, OFPMC_DELETE komutlarından bir tanesi getirilmelidir.

- OFPMC_ADD: Yeni bir Meter eklemek için kullanılır. Eklenmek istenen Meter'a ait Meter_id, switch'te daha önceden var olan bir Meter_id ile çakışıyorsa switch OFPMMFC_METER_EXIST hata mesajını gönderir.
- OFPMC_MODIFY: Belirtilen Meter'ın modifiye edilmesi için kullanılır. Switch'te bulunan Meter'lardan biri silinmiş ve bu Meter'la ilgili bir değişiklik yapılmak isteniyorsa switch, OFPMMFC_UNKNOWN_METER hata mesajını gönderir.
- OFPMC_DELETE: Belirtilen Meter'ın silinmesi için kullanılır. Silme işlemi için sadece Meter_id bilgisine ihtiyaç duyulmaktadır. OFPM_ALL komutuyla sanal Meter'lar hariç tüm Meter'lar silinebilir.

Flags

Meter konfigürasyon bayrakları OFPMF_KBPS, OFPMF_PKTPS, OFPMF_BURST, OFPMF_STATS değerlerinden birini almak zorundadır.

- OFPMF_KBPS: Meter'a gelen paketlerin Kbps (kilo-bit-per-second) cinsinden ölçüleceğini ifade eder.
- OFPMF_PKTPS: Ölçü birimi olarak saniyede gelen paket sayısının kullanılacağını gösterir.

- OFPMF_BURST: Meter'a anlık olarak yüksek değerlerde paket gelmesinin istendiği durumlarda kullanılır. Eğer ki böyle bir durum istenmiyorsa bu bayrak aktif edilmez.
- OFPMF_STATS: Meter istatistiklerinin toplanması için kullanır.

Meter_id

Meter_id alanı switch'te yer alan her bir Meter'a ait benzersiz bir değer tanımlamak için kullanılır. Meter_id değeri 1 ile başlar ve switch'in desteklediği maksimum değere kadar artar.

Meter Band Header

Meter Band başlık kısmında type, len, rate ve burst size alanları yer almaktadır. Tablo2.7'de Meter Band başlık kısmında yer alan değerler ve kapladıkları alanlar bit cinsinden gösterilmiştir.

Tablo 2.7 Meter-Band başlık yapısı

METER BAND HEADER	
TYPE (16Bit)	LEN (16Bit)
RATE (32Bit)	
BURST SIZE (32Bit)	
BAND	

Type

Meter Band türü, Meter'ın işlevinin belirtilmesi amacıyla kullanılan alandır. OFPBT_DROP, OFPBT_DSCP_REMARK, OFPBT_EXPERIMENTER değerlerinden biri kullanılabilir. Tablo 2.8 Meter band başlık alanına ait parametre ve değerleri göstermektedir.

Tablo 2.8 Meter-Band başlık parametreleri

ALAN	PARAMETRE	DEĞER
TYPE	DROP	1
	DSCP_REMARK	2
	EXPERIMENTER	0XFFFF

- OFPBT_DROP: Basit bir miktar sınırlayıcı tanımlamak için kullanılır. Drop türünde, belirtilen bant genişliği sınırını aşan paketlerin düşürülmesi sağlanır. Dolayısıyla belirtilen sınıra kadarki kullanılan bant genişliği garanti edilmiş olur.
- OFPBT_DSCP_REMARK: Basit düzeyde DiffServ politikası tanımlamak için kullanılır. Band genişliği miktarını aşan paketlerin başlık kısımlarında yer alan DSCP

bitlerinin yeniden işaretlenerek paketlerin düşürülme önceliklerinin değiştirilmesi sağlanır. Drop türünden farklı olarak Prec_Level değerinin de belirtilmesi gerekmektedir. Bu değer paketlerin önceliklerinin hangi seviyede azaltılacağını belirtmek için kullanılmaktadır.

- OFPBT_EXPERIMENTER: Deneysel çalışmalar için kullanılmaktadır.

Len

Meter Band'ın byte cinsinden kullanacağı alanı ifade eder.

Rate

Meter Band'ın işlem yapmaya başlayacağı sınırı temsil eder. Meter'a gelen paketler bu sınır değerini aştıklarında Meter türüne göre işlem yapmaya başlarlar. Bu değer varsayılan olarak saniyedeki kilo-bit sayısını (Kbps) ölçü birimi olarak kullanır. Bazı durumlarda flags alanı Meter'a gelen saniyedeki paket sayısı (Pktps) olarak belirtilmiş olabilir. Bu durumlarda rate alanının birimi Pktps olmaktadır.

Burst_size

Burst_size değerinin kullanılabilmesi için flags alanında OFPMC_BURST bayrağının tanımlanmış olması gerekmektedir. Burst_size değeri Meter Bandın işlem yapması istenen maksimum değeri ifade eder. Genel olarak Kbps cinsinden bir değer alır.

2.1.3.3 Meter istatistikleri

Kontrolörler switch'lerden Meter istatistiklerini almak için OFPST_METER istatistik talep mesajını gönderirler. Bu mesaj paketinin içerisinde Meter_id alanı bulunmaktadır. Bu alana istatistik bilgileri istenilen Meter'ın Meter_id değeri yazılır. Bütün Meter'lardan istatistik alınmak isteniyorsa OFPM_ALL ifadesi yazılarak istatistik talebi gönderilir [2].

Bu talebe cevaben switch'ler OFP_METER_STATS mesajı ile kaydettikleri veri setlerini kontrolöre gönderirler. Bu paket içerisinde Meter_id, len, flow_count, packet_in_count, byte_in_count, duration_second, duration_nsec değerleri ve ofp_Meter_band_stats paketi bulunmaktadır.

- Meter_id: İstatistik bilgisi gönderilen Meter'a ait Meter_id değeridir.
- Len: Gelen istatistiğin byte cinsinden kapladığı alanı belirtir.

- Flow_count: Meter'la ilişkilendirilen akış sayısını belirtir.
- Packet_in_count: Meter'a gelen paket sayısı belirtir.
- Byte_in_count: Meter'a gelen paketlerin byte cinsinden kapladığı alanı belirtir.
- Duration_second: Saniye cinsinden Meter'ın çalıştığı süreyi gösterir.
- Duration_nsec: Nanosaniye cinsinden Meter'ın çalıştığı süreyi gösterir.

ofp_Meter_band_stats paketi içerisinde ise packet_band_count ve byte_band_count değerleri bir dizi içerisinde gelmektedir.

- Packet_band_count: Meter Band'a gelen paket sayısını gösterir.
- Byte_band_count: Meter Band'a gelen paketlerin byte cinsinden kapladığı alanı belirtir.

2.1.3.4 Meter yapılandırma bilgileri

OFPST_METER_CONFIG yapılandırma talep mesajı ile bir ya da daha fazla Meter'a ait yapılandırma bilgileri elde edilebilir. Meter_id alanına switch'teki bir Meter'ın Meter_id değeri yazılarak sadece o Meter'a ait yapılandırma bilgileri talep edilebileceği gibi OFPM_ALL ifadesi yazılarak bütün Meter'lara ait yapılandırma bilgileri de talep edilebilir. Bu talebe karşılık olarak switch'ler OFPMP_METER_CONFIG mesajını gönderir. Bu mesaj paketinde length, flags, Meter_id alanları bulunmaktadır [2].

- Length: Yapılandırma mesajının byte cinsinden kapladığı alanı belirtir.
- Flags : Meter'a tanımlanan bayrak bilgilerini ifade eder.
- Meter_id: Meter numarası.

2.1.3.5 Meter özellik bilgileri

OFPST_METER_FEATURES mesajı ile switch'te yer alan bütün Meter'lara ait özellik bilgileri elde edilmektedir. Talep mesajının içerisi boştur. Cevap mesajının içerisinde ise max_Meter, band_types, capabilities, max_bands, max_color değerleri bulunmaktadır [2].

- max_Meter: Switch'in desteklediği maksimum Meter sayısını belirtir.
- band_types: Desteklenen Meter Band türlerini belirtir.
- capabilities: Desteklenen flags değerlerini belirtir.
- max_band: Her bir Meter için desteklenen maksimum band sayısını belirtir.
- max_color: Desteklenen maksimum renk değerini belirtir.

2.1.3.6 Meter hata kodları ve mesajları

Meter işlemleri sırasında oluşabilecek hatalara ait kodlar ve açıklamaları Tablo 2.9'da yer almaktadır.

Tablo 2.9 Meter hata mesajları ve kodları

Hata Mesajı	Hata Kodu	Açıklama
OFPMMFC_UNKNOWN	0	Bilinmeyen hata.
OFPMMFC_METER_EXIST	1	Eklenmek istenen Meter zaten var.
OFPMMFC_INVALID_METER	2	Belirtilen Meter özellikleri geçersizdir.
OFPMMFC_UNKNOWN_METER	3	Meter düzenlenemedi. Var olmayan Meter.
OFPMMFC_BAD_COMMAND	4	Desteklenmeyen ya da bilinmeyen komut.
OFPMMFC_BAD_FLAGS	5	Flag konfigürasyonu desteklenmiyor.
OFPMMFC_BAD_RATE	6	Belirtilen band rate deüeri desteklenmiyor.
OFPMMFC_BAD_BURST	7	Burst_size değeri desteklenmiyor.
OFPMMFC_BAD_BAND	8	Meter Band desteklenmiyor.
OFPMMFC_BAD_BAND_VALUE	9	Meter Band değeri desteklenmiyor.
OFPMMFC_OUT_OF_METERS	10	Meter bulunmamaktadır.
OFPMMFC_OUT_OF_BANDS	11	Meter için maksimum Band sayısı aşıldı.

2.2 İletim Düzlemi

2.2.1 Yazılımsal switch'ler (OpenFlow Software Switch)

Yazılımsal switch'ler, yazılım tanımlı ağlarla ilgili yapılan çalışmaların test edilebilmesi için geliştirilmekte ve kullanılmaktadır. Mininet simülatör içerisinde bulunan OVS ve OVSK'lar (OpenFlow Virtual Switch) OpenFlow1.3 protokolü ile birlikte kullanımına başlanılan Grup ve Meter özelliklerini desteklememektedir. Bu özelliklerin desteklenebilmesi için Ericsson Research TrafficLab tarafından Brezilyada kurulan Telekomünikasyon Araştırma ve Geliştirme Merkezinde (CPqD - Centro de Pesquisa e Desen-volvimento em Telecomunicações) bir yazılımsal switch geliştirilmiştir [17]. Geliştirme merkezinin adından dolayı genel olarak bu switch CPqD ismiyle anılmıştır. Ayrıca birçok kullanıcı tarafından OpenFlow1.3 SoftSwitch, OF13SS, ofSoftSwitch13 gibi isimlerle de kullanılmıştır. Switch'e resmi bir isim verilmemiş olmasından dolayı ortaya çıkan bu isim kargaşasını aşabilmek adına son olarak BOFUSS (Basic OpenFlow User-Space Software Switch) isminin yaygınlaştırılmasına çalışılmaktadır [18].

2.2.2 DPCTL

DPCTL, OpenFlow switch'leri üzerinde kontrol sağlayabilen bir yönetim aracıdır. OpenFlow1.3 versiyonu ile desteklenmeye başlanan Grup ve Meter özellikleri için geliştirilen Yazılımsal switch'in (ofSoftSwitch / CPqD / BOFUSS) yönetilmesinde de DPCTL kullanılabilmektedir. Bu araç ile birlikte akış-tablosuna kayıt ekleme, switch'lere Meter ekleme, switch'lerin durumlarını ve özelliklerini öğrenme, switch konfigürasyonu ve istatistiklerin elde edilmesi gibi yönetimsel işlemler yapılabilmektedir [19].

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.

Meter komutları

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

Kullanılışı:

dpctl unix:/var/run/s1.sock Meter-mod cmd=add, flags=1, Meter=1 drop:rate=10000

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=10000* ise Kbps cinsinden limit değerini göstermektedir. Bu komuta cevaben switch aşağıdaki mesajı göndererek oluşturulan Meter'a ait özellikleri bildirmektedir.

Meter_mod{cmd="add", flags="0x1"", Meter_id="1"", bands=[{type = drop, rate="10000", burst_size="0"}]}

- Meter-config: Meter tablosunda yer alan Meter'lara ait yapılandırma bilgilerinin neler olduğunu öğrenmek için Meter-config komutu kullanılır.

Kullanılışı: *dpctl unix:/var/run/s1.sock Meter-config*

Bu komuta cevaben switch, Meter tablolarında olan kayıtlara ait özellikleri bildirmektedir.

stat_repl{type="mconf", flags="0x0", stats=[{Meter= 1"", flags="1", bands=[{type = drop, rate="10000", burst_size="0"}]}]}

- Stats-Meter: Meter'larda kaydedilen istatistiklerin elde edilmesi için kullanılır.

Kullanılışı: *dpctl unix:/var/run/s1.sock stats-Meter*

Switch, istatistik isteğine cevap olarak barındırdığı bütün Meter'ların içerisindeki sayaçlarda tutulan değerleri gönderir. Bu cevap paketi içerisinde Meter_id, Meter'dan geçen akış sayısı, Meter'a gelen paket sayısı, paketlerin byte cinsinden kapladığı alan, Meter'ın çalışma süresi ve Meter'da bulunan bantlara gelen paket sayısı ve paketlerin byte cinsinden kapladığı alan bilgileri bulunmaktadır.

stat_repl{type="mstats", flags="0x0", stats=[{Meter= 1"", flow_cnt="0", pkt_in_cnt="0", byte_in_cnt="0"duration_sec="0", duration_nsec="0", bands=[{pkt_band_cnt="0", byte_band_cnt="0"}]}]}

- Meter-features: Meter özelliklerinin öğrenilmesi için kullanılan komuttur.

Kullanılışı: *dpctl unix:/var/run/s1.sock Meter-features*

Switch özellik talep mesajına karşılık olarak, maksimum Meter sayısı ve maksimum band sayısı gibi değerleri göndermektedir.

stat_repl{type="mfeat", flags="0x0"{max_Meter="256", band_types="1",capabilities ="d", max_bands = 16, max_color = 8}}

2.3 Mininet

Mininet BSD açık kaynak kod lisansı altında yayınlanmakta olan ve hala geliştirme ve destekleme çalışmaları aktif olarak devam eden bir ağ emulatör programıdır. Mininet, yazılım tanımlı ağların hızlı bir şekilde prototipinin oluşturulması, fiziksel bir ağa gereksinim duyulmadan komplex ağ yapılarının test edilmesi ve aynı anda birden fazla topolojinin bağımsız olarak çalışabilmesine sunduğu destekten dolayı tercih edilmektedir.

Mininet tek bir komutla (*sudo mn*) saniyeler içerisinde gerçek bir çekirdekte de çalışabilecek switch ve uygulama kodlarını çalıştırarak realist bir sanal ağ kurabilmektedir [20]. Mininet'te oluşturulan ağlarla, Komut Satırı Arayüzü (CLI- Comment Line Interface) ve Uygulama Programı Arayüzü (API-Application Programming Interface) kullanılarak kolay bir şekilde etkileşim sağlanabilmektedir. Bu yüzden Mininet geliştiriciler, eğitimciler ve araştırmacılar için kullanışlı bir araçtır.

2.3.1 Mininet'in avantajları

- Hız: Basit bir ağ topolojisi birkaç saniye içerisinde oluşturulabilmektedir. Bu da çalıştır-düzenle-hata ayıkla döngüsünün çok hızlı bir şekilde yapılabileceğini göstermektedir.
- Gerçek programlarla birlikte çalıştırma: Linux ortamında çalıştırılabilen gerçek programlar (WireShark, Web Server, TCP Window Monitoring Tool…) Mininet ile birlikte çalıştırılabilmektedir.
- Paket gönderimini özelleştirme: Mininet'te yer alan switch'ler OpenFlow protokollerine göre programlanabilir. Dolayısıyla paket gönderim işlemleri özelleştirilebilmektedir.
- Birden çok platform desteği: Mininet bilgisayarlarda, bir sunucuda, sanal makinede, Linux dağıtımlarında ya da bulut platformlarında çalıştırılabilir.
- Paylaşım: Başkaları da kendi bilgisayarlarında diğer kullanıcıların geliştirdiği kodları direkt olarak çalıştırabilir.
- Kolay kullanım: CLI ya da API'ler ile Mininet ortamında istenilen testler yapılabilir.
- Açık kaynak kod: Mininet bir açık kaynak kod projesidir. Dolayısıyla programa ve kaynaklarına ücretsiz olarak erişebilir ve geliştirme sürecine katkı verilebilir [21].

2.3.2 Mininet sınırlılıkları

- Kaynakların sanal ortamda oluşturulan host ve switch'ler arasında dengeli bir şekilde paylaştırılması gerekmektedir.
- Mininet tüm sanal host'lar için tek bir Linux çekirdeği kullanır, dolayısıyla başka işletim sistemi çekirdeklerine bağlı yazılımların çalıştırılması mümkün değildir.
- Varsayılan olarak Mininet'te oluşturulan ağ, yerel ağdan ve internet ağından izole bir şekilde çalışmaktadır.
- Mininet'in sanal zaman kavramı güçlü değildir. Bu yüzden zamanlama ile ilgili ölçümler gerçek zamana göre yapılmaktadır. Bu durumda 100 Gbps gibi gerçek zamanlı çalışan uygulamalardan daha hızlı işlemler için sonuçların simule edilmesi zor olacaktır [22].

2.3.3 Mininet temel komutları

- Help: CLI komutlarının listesine erişmek için kullanılır.
- Dump: Mininet'te yer alan aygıtlara bağlı olarak ağ bilgilerini verir.
- Net: Cihazların birbirleri ile olan bağlantılarını gösterir.
- Node: Mininet'te oluşturulan cihazların (Switch-Host) listesini verir.

- Xterm: Mininet'teki cihazlara ait terminal ekranına erişmek için kullanılır.
- Ping: Cihazlar arasındaki bağlantı durumlarını kontrol etmek için kullanılır.
- Iperf: Cihazlar arasındaki bant genişliğini ölçmek için kullanılır.
- Custom : Python dosyalardaki özel sınıf ve parametrelerin okunması için kullanılır.

Terminal ekranında *$sudo mn –h* komutuyla Mininet'te kullanılan bütün komutlara ve parametrelerine erişilebilmektedir.

2.3.4 Mininet'te topoloji kurulumu

Mininet'te CLI veya python scriptlerinden oluşan bir API aracılığıyla bir topoloji oluşturulabilmektedir [22].

2.3.4.1 Komut satırı (CLI) ile topoloji kurulumu

Kod yapısı:

$ sudo mn --controller=[remote], ip=[Kontrolör IP], port=[Kontrolörün dinlediği port] --topo=[topoloji adı],[topoloji parametreleri] --Switch=[Switch türü]…

Sudo mn komutuyla Mininet'te 1 switch ve 2 host'tan oluşan bir topoloji oluşturulabilmektedir. Daha sonrasında yer alan parametre ve parametrelerin aldığı değerler ise ihtiyaç duyduğumuz özelliklere göre esnetilebilmektedir [22].

Parametreler

- Controller : Bilgisayarımızda çalışan kontrolöre erişmek için kullanılır. Default, none, nox, ovsc, ref, ryu, remote[param=value] değerlerinden biri yazılabilir.
- Ip: Kontrolöre ait IP adresini belirtir.
- Port: Kontrolörün dinlediği port adresini (default: 6653) ifade eder.
- Topo: Mininet içerisinde minimal, reversed, torus, Single, Linear ve Tree olmak üzere bazı topolojiler kurulu olarak gelmektedir. Mininet içerisinde kurulu olarak gelen topolojiler dışında bir topolojiye ihtiyaç duyulması durumunda python scriptleri kullanılarak özel bir topoloji oluşturulmalıdır.

  Topoloji örnekleri

  - ✓ $ sudo mn –topo single,3

    Bir switch'e bağlı 3 host'tan oluşan bir topoloji kurulmasını sağlar.
  - ✓ $ sudo mn –topo tree, depth=3,fanout=2

    3 tane switch ve her bir switch'e bağlı iki adet host'tan oluşan bir topoloji kurulmasını sağlar.

- ✓ $ sudo mn –topo linear,3

  3 tane ardışık switch ve her bir switch'e bağlı birer tane host'tan oluşan bir topoloji kurulmasını sağlar.

- Switch : Topolojide kullanılacak olan switch türünü belirtmek için kullanılır. Mininet'te kurulu olarak gelen OpenFlow vSwitch (OVS - OVSK) ya da daha sonradan kurulan kullanıcı (user) switch'leri belirtilebilir.

2.3.4.2 Python scriptleri (API) ile topoloji kurulumu

Python scriptleri ile oluşturulan .py uzantılı dosyalar ile de topoloji oluşturulabilmektedir. Aşağıdaki komut satırı aracılığıyla Şekil 2.6'da gösterilen Topoloji.py dosyası çalıştırılarak 2 switch ve 2 host'tan oluşan bir topoloji oluşturulmuştur.

$ sudo mn –-custom ~/mininet/custom/Topoloji.py –topo=mytopo –Switch=user --controller=remote

```
""" Topoloji.py

    h1--------s1--------s2--------h2
"""

from mininet.topo import Topo

class MyTopo( Topo ):

    def __init__( self ):

        # Topoloji başlatılıyor
        Topo.__init__( self )
        # Switch ve Hostlar ekleniyor
        host1 = self.addHost( 'h1' )
        host2 = self.addHost( 'h2' )
        switch1 = self.addSwitch( 's1')
        switch2 = self.addSwitch( 's2')

        # Bağlantılar oluşturuluyor
        self.addLink( host1, switch1)
        self.addLink( switch1, switch2 )
        self.addLink( switch2, host2 )

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

Şekil 2.6 Python scriptleriyle topoloji oluşturma

Python scripti parametreleri

- Topo: Mininet topolojileri için kullanılan temel sınıfı belirtmektedir.
- addHost: Topolojiye host eklemek için kullanılır.
- addSwitch: 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.

2.3.5 Iperf

Iperf, internet protokolü ile yönetilen ağlarda, erişilebilen maksimum bant genişliğinin ölçümü için kullanılan bir araçtır [23]. Zamanlama ve protokollerle ilgili çeşitli parametrelerin kullanılmasını destekleyen Iperf aracı, yapılan her test sonrasında bant genişliği, paket kaybı, gecikme, gecikme farklılığı gibi parametrelerin raporlanmasını da sağlamaktadır. Iperf aracının kullanılması için Mininet ortamında *xterm h1 h2* komutuyla iki ayrı host'a ait terminal ekranı açılır. Host'lardan biri sunucu diğeri ise istemci olmalıdır. Sunucu ekranına *Iperf –s*, istemci ekranına ise *Iperf –c sunucu_ip_adresi* yazılarak iki uç arasında bir TCP trafiği oluşturulabilir [24]. Tablo 2.10'da Iperf aracı ile kullanılan bazı parametre ve açıklamaları gösterilmektedir.

Tablo 2.10 Iperf aracı kullanım parametreleri

Kullanım Yeri	Parametre	Açıklaması
Genel	**-p** , --port	İletimin yapılacağı port değerinin belirtilmesi için kullanılır. Varsayılan değer 5001'dir.
	-u, --udp	UDP paketinin iletimi yapılacağını belirtir.
	-i, --interval	Periyodik olarak gönderilen bilgilendirme raporunun saniye cinsinden gönderilme sıklığını gösterir.
	-h, --help	Komut listesini görüntüler.
Server/Sunucu	**-s**, --server	Host'un sunucu modunda çalıştırılacağını belirtir.
Client/İstemci	**-b**, --bandwidth	UDP paketlerinin gönderimi sırasında kullanılacak bant genişliğini ifade eder. Varsayılan değer 1 Mbit/s'dir.
	-c, --client	Host'un istemci modunda çalıştırılacağını belirtir.
	-t, --time	Paket iletiminin ne kadar süreceğinin saniye cinsinden belirtir. Varsayılan değer 10 saniyedir.
	-S, --tos	Paketlerin hangi hizmet sınıfından olduğunun belirtilmesi için kullanılır. Ör. AF11 =0x28

3. HİZMET KALİTESİ

Hizmet kalitesi, bir ağ içerisindeki kaynakların tahsisi veya hizmet farklılıklarının sağlanması şeklinde tanımlanabilir.

Hizmet farklılıkları, paketlere atanan öncelik değerlerine göre oluşturulmaktadır. Örneğin, bir video dosyasına yüksek öncelik verilmişse bu dosyaya ait paketler switch'e geldiği zaman ilgili kuyruğun en önüne getirilecek ve hızlı bir şekilde iletimi sağlanacaktır. Öte yandan düşük öncelikli olarak işaretlenmiş bir web tarayıcı programına ait paketler geldiğinde ise bant genişliği uygun duruma gelene kadar bu paketler kuyrukta bekletilecek ve daha sonra iletilecektir. Bekleme sırasında ağda bir tıkanıklık oluşması durumunda düşük önceliğe sahip paketler ilk önce düşürülür ve yüksek öncelikli paketler için ağ trafiğinde bant genişliği olarak yer açılmış olur. Bu durumda web tarayıcı programının gönderdiği paketler yerine ulaşamayacak ve dolayısıyla uygulamalar olması gerektiği gibi çalışamayacaktır. Hizmet kalitesi sunulmasına duyulan ihtiyaç bu noktada ortaya çıkmaktadır.

Ağda tıkanıklık olması durumunda sunucuların diğer hizmetler için verecekleri cevap süreleri (response time) önem kazanmaktadır. Hizmet kalitesi yapılarının cevap süresine olan etkisi üzerine yapılan bir çalışmada [25] iki sunucu, bir switch üzerinden bir host'a data göndermektedir. Birinci sunucu 1 Mbit'lik veri gönderirken ikinci sunucu 3,5 Mbit'lik data göndermektedir. Toplamda 4,5 Mbit'lik bir trafik switch'e gelmektedir. Switch ile host arasındaki bant genişliği 4 Mbit/s'dır. Trafik gönderilmeye başlandığında tıkanıklık olacaktır. Bu çalışmada tıkanıklık anında sunucuların erişilebilirlik durumları karşılaştırılmak istenmektedir. Trafik önceliklendirme olmadan yapılan test sonuçlarına göre her iki sunucunun da cevap süreleri 3000 milisaniyedir. Hizmet kalitesi oluşturularak yapılan testte ise iki adet kuyruk kullanılmıştır. Sunucu 1 için 1,5 Mbps'lik bir bant genişliği, sunucu 2 için ise 2,5 Mbps'lik bant genişliği rezerve edilmiştir. Test sonuçlarına göre sunucu 1 deki rezerve edilen bant genişliği ve gönderilen trafik aynı miktarda olduğu için cevap süresi 0 (sıfır) olmuştur. Sunucu 2 de ise rezerve edilen bant genişliği 2,5 Mbps, gönderilen trafik 3,5 Mbps olduğu için tıkanıklık oluşmuş ve rezerve bant genişliğini aşan paketler düşürülmüştür. Bu yüzden Sunucu 2'nin cevap süresi 5000 ms olarak elde edilmiştir. Bu veriler hizmet kalitesi yapılarının ağ performansını nasıl etkilediğini göstermektedir.

Hizmet kalitesine, yaygın olarak ses ve video iletimi uygulamalarında ihtiyaç duyulmasına rağmen, Nesnelerin İnterneti paradigmasındaki bazı uygulamalarda ihtiyaç duymaktadır. Örneğin, bir fabrikada makinaların herhangi birinde oluşabilecek bir arızanın hızlı bir şekilde

tespit edilmesi çok önemlidir. Çünkü üretim yerlerinde arızalara geç müdahale edilmesi üretim hatalarına ya da büyük maddi kayıplara neden olmaktadır. Üretim alanlarındaki sensörlerden gelecek bilgilere öncelik verilmesi durumunda gerçek zamanlı bir durum kontrolü yapılabilecek ve üretim işlemlerinin hizmet kalitesi arttırılmış olacaktır.

Hizmet kalitesi yapıları oluşturabilmek için IETF (Internet Engineering Task Force) Integrated Services (IntServ) ve Differentiated Services (DiffServ) olmak üzere iki farklı yapı sunmaktadır [26].

3.1 Integrated Services (IntServ)

IntServ, hizmet kalitesi oluşturabilmek için kaynak tahsisi yöntemini kullanmaktadır. Ağ'da bulunan tüm aygıtlar (Router, Switch) aynı kaynak tahsisi politikasını uygulamak zorundadır. Hizmet kalitesi yapısına sahip olmak isteyen uygulamalar, uygulama ile hedef arasında kalan bütün switch'lerde aynı bant genişliğini rezerve etmelidir. Rezervasyon işlemleri sırasında her bir switch'e bir rezervasyon talebi gelir. Hizmet kalitesinin garanti edilebilmesi için gönderici ve alıcı arasında ki bütün switch'lerin rezervasyon talebini kabul etmiş olması gerekmektedir.

IntServ, Flow Spec (Flow Specification- Akış Özellikleri) ve RSVP (Resource Reservation Protocol) olmak üzere iki fonksiyona bölünmüştür. Flow Spec, uygulamaların rezervasyon ihtiyacının belirlenmesi için kullanılırken, RSVP ise switch ve uygulamalar arasındaki rezervasyon talebi ve talebin cevaplanması işlemleri sırasında kullanılan protokoldür [26].

3.1.1 Flow spec

Flow spec, TSPEC (Traffic Specification) ve RSPEC *(Request Specification)* olmak üzere iki alt bölüme ayrılmıştır. *TSPEC* uygulamalardan gelen trafik detayları için kullanılır. Böylelikle switch'in ne kadar rezervasyon yapacağı tespit edilmektedir. *RSPEC ise* trafik akışının farklı ihtiyaçlarının belirtilmesi için kullanılır. Best Effort, Controlled Load ve Guaranteed olmak üzere 3 hizmet seviyesi vardır [26].

3.1.2 RSVP

RSVP, rezervasyon işlemleri için uygulama ile switch arasındaki iletişimden sorumlu olan protokoldür [27]. RSVP bu iletişim için *path* ve *resv* olmak üzere 2 farklı mesaj kullanmaktadır.

- Path: Sunucu tarafından gönderilen *path* mesajı, datanın aktarılacağı yol bilgisi ile birlikte TSPEC bilgilerini alıcıya iletilir.

- Resv: Alıcı tarafından trafiğin başlatılması için gönderilen *resv* mesajı, flowspec bilgilerini taşır ve *path* mesajına cevap olarak sunucuya gönderilir. Resv mesajındaki bilgilere göre alıcı ve verici arasındaki bütün uçlarda eşit miktarda bant genişliği tahsisi yapılır. IntServ'de kullanılan *soft state* metodu sayesinde belirlenen süre içerisinde sunucudan herhangi bir mesaj gelmez ise yapılan rezervasyonlar iptal edilmektedir.

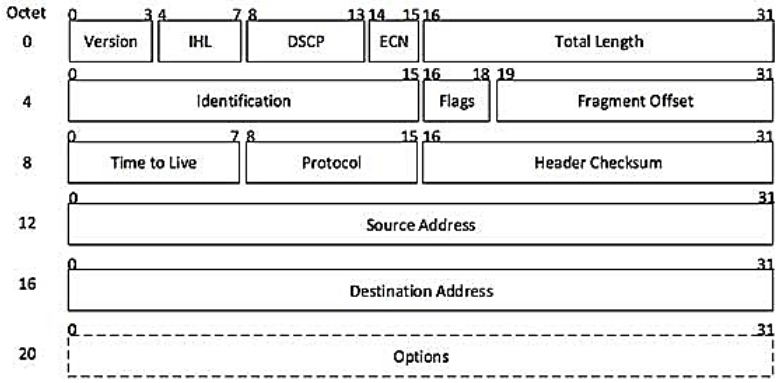
3.2 Differentiated Services (DiffServ)

DiffServ, kaynakların farklı sınıflara ait trafikler tarafından belirli kurallara göre kullanılması ilkesine göre çalışmaktadır. Bu yaklaşımda her bir paket IPv4 başlık kısmında yer alan DSCP bitleri aracılığıyla farklı önceliklere sahip trafik sınıflarına yerleştirilirler. Böylece ağdaki cihazlar kendilerine gelen paketlerin başlık kısmında yer alan DSCP değerini okur ve trafik önceliklendirme sınıfı bilgisine göre yönlendirme işlemini gerçekleştirir.

Her bir sınıfa ait paketlerin iletimi ile ilgili özellikler PHB (Per Hop Behaviour) terimi ile ifade edilir. PHB sınıflara ait kuralların ve özelliklerin paketlere aktarılması için kullanılır. Bu terim paketlerin ağdaki her bir düğüm noktasında nasıl bir davranışla karşılaşacağını temsil etmektedir. DiffServ kullanılan bir ağda yer alan bütün cihazlar aynı DiffServ kurallarına göre hareket etmek zorundadır.

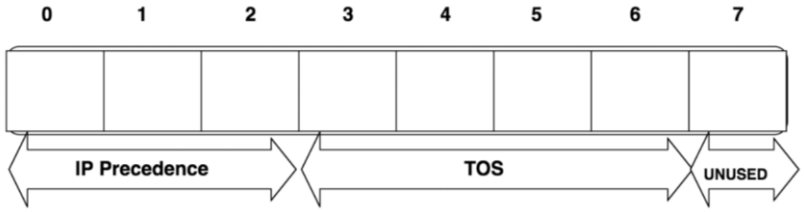
3.2.1 İnternet protokolü (IP)

İnternet protokolü, internette kullanılan temel protokoldür. Ağ üzerinden, hangi türden paketlerin gönderildiğine bakılmaksızın, datalar internet protokolü aracılığıyla iletilmektedir. IP, paketlerin iletimi konusunda herhangi bir garanti vermemektedir. Tamamen bağlantısız çalışmaktadır bu yüzden doğru ve hatasız iletim yapılabilmesi için kendi üzerinde yer alan TCP gibi protokollerin denetimi altında çalışmaktadır [28]. Diğer protokollerde olduğu gibi her bir IP paketine taşıdığı dataların yanı sıra bazı detaylar ve kontrol bilgileri eklenir. Paketlerin başlık kısımlarında yer ala bu bilgiler, paketin ağdaki cihazlar tarafından nasıl yorumlanacağını ve nasıl işletileceğini belirtmektedir. Şekil 3.1'de IPv4 başlık kısmı gösterilmektedir.



Şekil 3.1 IPv4 başlık kısmı

IPv4 başlık kısmı temel olarak 20 Byte'lık bir alandır. Başlık kısmında yer alan her bir bölüm, hedef ve aradaki cihazlar için bir kontrol ayarını belirtmektedir. IETF tarafından yayınlanan RFC791 belgesinde yer alan TOS(Type of Service) alanı aracılığıyla hizmet kalitesi işlemleri yapılmaktaydı [29]. Şekil 3.2'de TOS yapısı görülmektedir.



Şekil 3.2 TOS yapısı

8 bitten oluşan TOS alanındaki bitlerin 4 tanesi verim, güvenilirlik, gecikme ve maliyet anlamına geliyordu fakat bu bitler genel olarak boş bırakıldı. Bitlerin 3 tanesi paketlerin önceliklendirilmesi (IP Precedence) için kullanılırken sonuncu bit ise pek kullanılmadı. Sonuç olarak 8 bit içerisinden sadece önceliklendirme bitleri kullanıldı ve 2^3= 8 farklı değer aldı. Tablo 3.1'de TOS yapısında yer alan paket önceliklendirme değerleri görülmektedir.

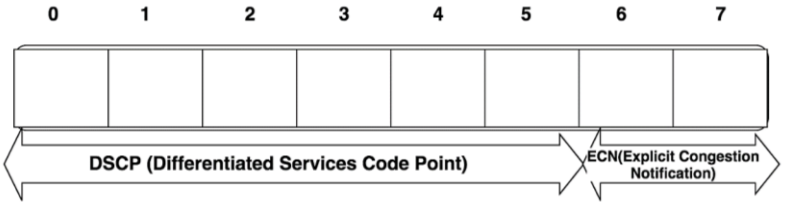
Tablo 3.1 Paket önceliklendirme değerleri (TOS Byte)

Paket Önceliklendirme Değerleri (TOS)	
000	Rutin
001	Priority
010	Immediate
011	Flash
100	Override Flash
101	Critical
110	Internetwork
111	Internet

3.2.2 Farklılaştırılmış Hizmetler Alanı (DS Field) ve DSCP bitleri

RFC2474 [30] ve RFC3168 [31] ile birlikte TOS alanı, 6 bitlik Farklılaştırılmış Hizmetler Alanı Kod Noktası (DSCP) ve 2 bitlik Açık Tıkanıklık Bildirimi (ECN- Explicit Congestion Notification) alanı olarak, Farklılaştırılmış Hizmetler Alanı (DS Field) adı ile yeniden yapılandırılmıştır.

DSCP kısmında 64 farklı değer (2^6 = 64) oluşturulabilmektedir. Altı bitlik alanının en sonundaki bit 0 (sıfır) olarak sabitlenmiştir. Diğerleri ise 0 ve 1 değerlerini almaktadır. ECN bitleri ise henüz kullanılmamaktadır. Şekil 3.3'de DSCP ve ECN bitleri gösterilmiştir.



Şekil 3.3 DSCP ve ECN bitleri

TOS alanının ve DSCP bitlerinin varsayılan değeri 000000'dır. TOS alanında yer alan önceliklendirme bitleri (ilk 3 bit), DSCP'de trafiğin sınıflandırılması için kullanılmaktadır. Önceliklendirme bitlerinin artık Sınıf Seçici bitler olarak kullanılması sayesinden DSCP ve TOS alanları arasındaki uyumluluk sağlanmıştır. Şekil 3.4'te Farklılaştırılmış Hizmetler Alanı (DS Field) gösterilmektedir.

Farklılaştırılmış Hizmetler Alanı (DS FIELD)							
DSCP (Differentiated Services Code Point)						**ECN**	
0	1	2	3	4	5	6	7
CLASS (Sınıf Seçici)			**DROP PRECEDENCE** (Düşürülme Önceliği)			**ECN** (Akış Kontrol Bitleri)	

Şekil 3.4 Differentiated-Service alanı

Sınıf seçici kısmında Best Effort (BE), Assured Forwarding (AF) ve Expedited Forwarding (EF) olmak üzere üç farklı hizmet kalitesi sınıfı seçimi yapılabilmektedir. Düşürülme önceliği kısmında AF sınıfı paketler için 3 farklı düşürülme önceliği değeri atanabilmektedir. Akış kontrol bitleri ise rezerve edilmiş olup henüz kullanılmamaktadırlar. Tablo 3.2'de 8 farklı Sınıf seçici değeri gösterilmektedir.

Tablo 3.2 DSCP sınıf değerleri

Sınıf	Sınıf Seçici Bitleri
CS0	**0 0 0**
CS1	**0 0 1**
CS2	**0 1 0**
CS3	**0 1 1**
CS4	**1 0 0**
CS5	**1 0 1**
CS6	**1 1 0**
CS7	**1 1 1**

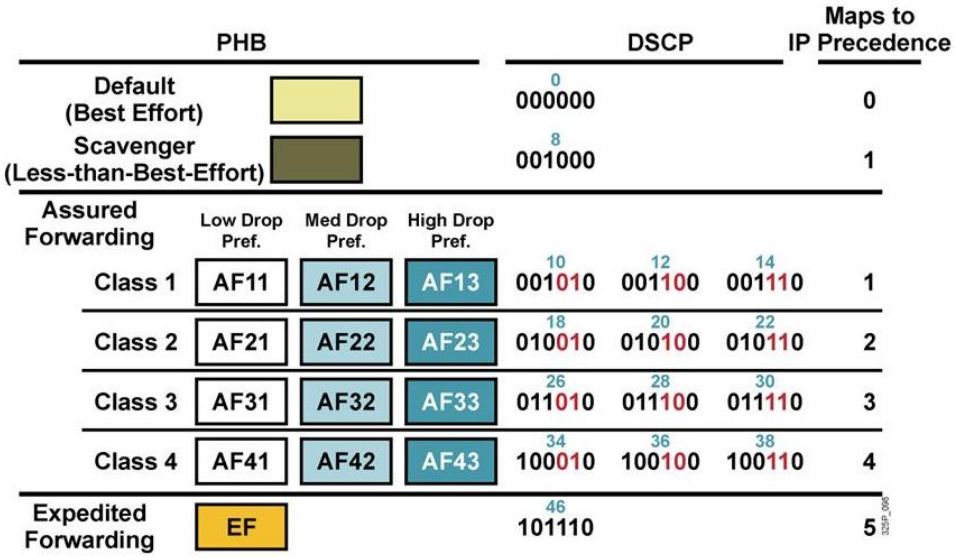
3 bitlik sınıf seçici alanında (2^3=8) 8 farklı (CS0-CS7) sınıf değeri seçilebilmektedir. Bu değerlerden CS0 Best Effort sınıfını, CS1-CS4 AF sınıfını, CS5'te EF sınıfını temsil etmektedir. CS6 ve CS7 sınıfları ağ protokolü ve kontrol işlemleri için rezerve edilmiştir. Bu yüzden CS5 sınıfı en yüksek önceliğe sahip sınıftır.

Sınıf değeri yüksek olan paketler diğer paketlere göre daha yüksek önceliğe sahiptir. Örneğin, CS5 sınıfından bir paket CS4 sınıfından bir pakete göre daha önceliklidir. Bu sınıflara düşük(1), orta(2) ve yüksek(3) olmak üzere 3 farklı seviyede düşürülme önceliği değerleri eklenerek daha detaylı bir sınıflandırılma yapılmıştır. CS0 varsayılan sınıf, CS5 en yüksek önceliğe sahip sınıf, CS6 ve CS7 rezerve sınıflar olduğu için düşürülme önceliği atamaları sadece CS1-CS4 (AF) sınıflarına yapılmıştır. Tablo 3.3'te AF sınıfına ait düşürülme öncelik değerleri görülmektedir.

Tablo 3.3 AF sınıfı (CS1-CS4) düşürülme öncelik değerleri

Sınıf	Sınıf Seçici	Düşürülme Önceliği		Açıklama
CS1	0 0 1	**01**	0	CS1 sınıfı düşürülme önceliği düşük paket
CS1	0 0 1	**10**	0	CS1 sınıfı düşürülme önceliği orta paket
CS1	0 0 1	**11**	0	CS1 sınıfı düşürülme önceliği yüksek paket
CS2	0 1 0	**01**	0	CS2 sınıfı düşürülme önceliği düşük paket
CS2	0 1 0	**10**	0	CS2 sınıfı düşürülme önceliği orta paket
CS2	0 1 0	**11**	0	CS2 sınıfı düşürülme önceliği yüksek paket
CS3	0 1 1	**01**	0	CS3 sınıfı düşürülme önceliği düşük paket
CS3	0 1 1	**10**	0	CS3 sınıfı düşürülme önceliği orta paket
CS3	0 1 1	**11**	0	CS3 sınıfı düşürülme önceliği yüksek paket
CS4	1 0 0	**0 1**	0	CS4 sınıfı düşürülme önceliği düşük paket
CS4	1 0 0	**10**	0	CS4 sınıfı düşürülme önceliği orta paket
CS4	1 0 0	**1 1**	0	CS4 sınıfı düşürülme önceliği yüksek paket

CS1-CS4 sınıfları içerisinde düşürülme önceliği yüksek olan paketlerin ağda oluşabilecek tıkanıklıklar sırasında ilk önce düşeceği bilinmelidir. Örneğin, CS2 sınıfından düşürülme önceliği yüksek olan bir paket (010110), düşürülme önceliği orta ve düşük olan paketlerden daha önce düşürülecektir. Şekil 3.5'te standart PHB grupları gösterilmektedir.



Şekil 3.5 Standart PHB grupları

3.2.2.1 Best Effort PHB

Best Effort (BE) sınıfından paketlerin iletimi ile ilgili herhangi bir taahhüt yoktur, paketler ağdaki trafik durumuna göre alıcıya en iyi şekilde iletilmeye çalışılır. DSCP alanının varsayılan değeri (000000) BE sınıfını ifade etmektedir.

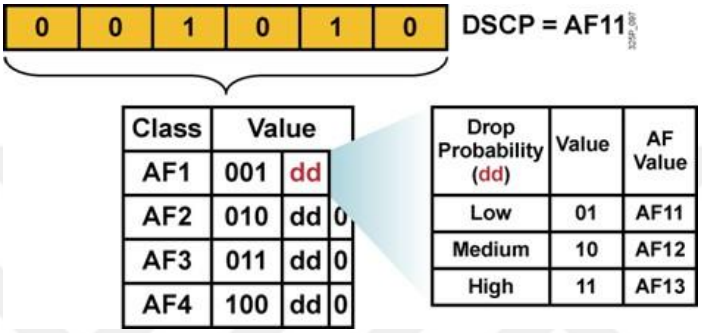
3.2.2.2 Assured Forwarding PHB

Assured Forwarding (AF) kuyruklama (Queueing) ve tıkanıklıktan kaçınma (Congestion Avoidance) olmak üzere iki farklı hizmet kalitesi fonksiyonuna sahiptir. Kuyruklama fonksiyonunda her bir switch, paketleri AF1x - AF4x olmak üzere 4 ayrı sınıfa ayırır ve her bir sınıfa ait paketler farklı kuyruklara yerleştirilir. Her bir kuyrukta düşük, orta ve yüksek olmak üzere (AFx1-AFx2-AFx3) 3 farklı tıkanıklık sınırı (threshold) vardır. Dolayısıyla paketlerin işaretlenmesi için 12 farklı DSCP değerine ihtiyaç duyulmaktadır [32] .

AF PBH bir AF sınıfına belirli bir bant genişliği miktarını garanti eder ve eğer uygun bant genişliği varsa ilave bant genişliğine erişim izni sağlayabilir.

Düşürülme öncelikleri düşük (1) , orta (2) ve yüksek (3) olmak üzere üç seviyedir. Örneğin, AF12 ifadesi CS1 (AF1) sınıfından düşürülme önceliği orta(2) seviye olan bir paketi ifade

etmektedir. Düşürülme seviyesinin arttırılabilmesi için gelen paketin düşük ya da orta seviyeli bir önceliğe sahip olması gerekmektedir. Yüksek düşürülme önceliğine sahip bir paketin düşürülme önceliği daha fazla arttırılamayacaktır. Bu seviyelerden *düşük seviye*, gelen paketler için taahhüt edilen bant genişliği oranını (CRI- Committet Rate Information) temsil etmektedir. *Orta seviye,* taahhüt edilen oranı aşan fakat aşım oranını (ERI - Excees Rate Information) aşmayan paketleri temsil ederken, *yüksek seviye* ise aşım oranını aşan paketleri temsil etmektedir. Aşım oranını aşan paketler en yüksek düşürülme önceliğine sahiptirler. Şekil 3.6'da AF PHB yapısı görülmektedir.

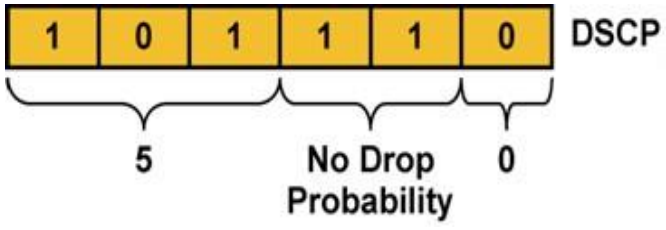


Şekil 3.6 Assured Forwarding PHB

3.2.2.3 Expedited Forwarding PHB

Expedited Forwarding (EF) PHB diğer tüm sınıf kategorilerine göre en yüksek önceliğe sahiptir. Bu nedenle daha çok kritik trafik durumlarında kullanılır. Bu sınıfın herhangi bir düşürülme önceliği yoktur. Bu yüzden hızlandırılmış yönlendirme olarak adlandırılmıştır. Şekil 3.7'de EF PHB yapısı görülmektedir.

Expedited Forwarding, kuyruklama ve yönetim (policing) olmak üzere iki farklı hizmet kalitesi fonksiyonu sunmaktadır. Kuyruklama fonksiyonu EF paketlerinin kuyrukta geçireceği zamanın minimize edilmesini sağlamaktadır. Oluşturulan öncelik kuyruğu sayesinde gecikme, gecikme farklılığı ve paket kaybı gibi sorunlar ortadan kaldırılmaktadır. Bu yüzden gerçek zamanlı iletişim gerektiren ses ve video iletimi hizmetleri için EF PHB kullanılmaktadır [33].



Şekil 3.7 Expedited Forwarding PHB

3.3 Karşılaştırma ve Seçim

Her iki yöntemle de hizmet kalitesi yapıları oluşturulabilmektedir. DiffServ akışlara farklı öncelikler verirken, IntServ ağda yer alan bütün bağlantılar için eşit bant genişliği rezervasyonu ile hizmet kalitesi sunmaktadır. Geniş ölçekli bir ağ yapısında her bir akış için rezervasyon yapmak ve bu rezervasyonların takip edilmesini sağlamak zor olacaktır. Dolayısıyla kaynakların verimli kullanılamaması sorunu ortaya çıkacaktır.

DiffServ ise sınıf tabanlı bir önceliklendirme yaptığı için switch'lerin sadece gelen paketin sınıfına ait bilgileri alması ve yönlendirme işlemini yapması gerekmektedir. Bu yüzden tüm ağ cihazlarında aynı sınıf bilgileri kullanılmalıdır. Bu durum paketlerin takibini kolaylaştırmaktadır ayrıca ağın daha da genişletilebilmesine olanak sağlamaktadır.

IntServ'de başka bir olumsuz durum ise bir süre sonra rezervasyonların iptal edilmesidir. İptal işleminden sonra uygulamalardan herhangi biri tekrar paket gönderimi yapmak istediğinde uçlar arasındaki rezervasyon işlemlerinin yeniden yapılması gerekmektedir. DiffServ'de ise iletim için gerekli sınıf konfigürasyonları ağ cihazlarında daha önceden yapıldığı için uygulamaların hangi aralıklarla paket gönderdiklerine bakılmaksızın sorunsuz bir şekilde iletim yapılabilecektir.

Hizmet kalitesi yapısı oluştururken DiffServ mimarisinin kullanılmasının daha uygun olacağı görülmektedir.

4. OPENFLOW METER ÖZELLİKLERİYLE HİZMET KALİTESİ OLUŞTURMA ÇALIŞMALARI ve TEST SONUÇLARI

Çalışmamız için tercih ettiğimiz Floodlight kontrolörün kullanılabilmesi için ilk önce bilgisayarımıza Linux dağıtımlarından Ubuntu v16.04'ün kurulumu yapılmıştır. Floodlight geliştirmelerinde Java dili kullanıldığı için JDK 8 ve Eclipse IDE programlarının kurulumundan sonra Floodlight kontrolörün, GitHub platformundan indirilerek kurulumu gerçekleştirilmiştir (Bkz. 2.1.1/Floodlight kurulumu).

4.1 Meter Kurulumu

Meter'ların Floodlight kontrollerde kullanılabilmesi için ilk önce bu özelliği destekleyen switch'lere Meter-mod mesajı (ADD) ile kurulmaları gerekir. Daha sonra Meter'lar MODIFY ve DELETE mesajları ile düzenlenebilir ya da silinebilirler [34]. Meter kurulum işlemleri Floodlight kontrolöre eklenen Meter modülüyle ve DPCTL yazılımsal switch yönetim aracı ile yapılmıştır.

4.1.1 Floodlight Meter modülü

Floodlight kontrolöre eklenen Meter modülünde bulunan addMeter sınıfı aracılığıyla ağda yer alan switch'lere Meter kurulumu yapılmaktadır. Drop ve Dscp_remark modlarında oluşturulan Meter'lar, paketlerin 400 Kbps değerini aşması durumunda devreye girerek işlevlerini yerine getirmektedirler. Şekil 4.1'de addMeter sınıfına ait kodlar görülmektedir.

```
public void addMeter() {
// Drop modunda meter ekleme
OFFactory meterFactory = OFFactories.getFactory(OFVersion.OF_13);
OFMeterMod.Builder meterModBuilder = meterFactory.buildMeterMod()
.setMeterId(1)
.setCommand(OFMeterModCommand.DROP)
.setBurstSize(0)
.setRate(400);
OFMeterBandDrop.Builder bandBuilder = meterFactory.meterBands().buildDrop();
OFMeterBand band = bandBuilder.build();
List<OFMeterBand> bands = new ArrayList<OFMeterBand>();
bands.add(band);
//Dscp_remark modunda bir meter ekleme
OFMeterMod.Builder meterModBuilder2 = meterFactory.buildMeterMod()
.setMeterId(2)
.setCommand(OFMeterModCommand.REMARK)
.setBurstSize(0)
.setRate(400);
OFMeterBandDscpRemark.Builder remarkBuilder = meterFactory.meterBands().buildDscpRemark();
remarkBuilder=remarkBuilder.setPrecLevel(1);
OFMeterBand band2= remarkBuilder.build();
bands.add(band2);
meterModBuilder.setMeters(bands).setFlags(OFMeterFlags.KBPS).build();
}
```

Şekil 4.1 Floodlight ile Meter oluşturma kodları

4.1.2 DPCTL ile Meter kurulumu

DPCTL yazılımsal switch yönetim aracı kullanılarak switch'lere Meter eklenebilmektedir [19]. Aşağıda yer alan komutlar kullanılarak switch'lere Drop veya Dscp_remark modlarında Meter'lar eklenmiştir.

$sudo dpctl unix:/temp/s1 Meter-mod cmd=add, flags=1, Meter=1, drop: rate=400

$sudo dpctl unix:/temp/s1 Meter-mod cmd=add, flags=1, Meter=2, dscp_remark: rate=400, prec_level=1

Komut satırında yer alan parametrelerden *s1* Meter'ın kurulacağı switch'i, *Meter-mod cmd=add* Meter modifikasyon mesajlarından add komutunun çalıştırılacağını, *flags=1* ölçü birimi olarak Kbps'ın kullanılacağını , *Meter=1/2* Meter_id değerini, *drop/dscp_remark* Meter band türünü, *rate* Meter Bandın aktif hale geleceği minimum değeri , *prec_level=1* paketlerin düşürülme öncelik değerini temsil etmektedir.

4.2 Akış-Kaydı Ekleme

Meter'lar kurulduktan sonra paketlerin bu Meter'larla ilişkilendirilmesi gerekmektedir. Bu yüzden akış-tablolarına, paketlerin Meter'lardan geçebilmesi için akış-kayıtlarının eklenmesi gerekir [34]. OpenFlowJ-Loxigen kütüphanesinde yer alan yapıcılar (Constructors) aracılığıyla akış-tablolarına yeni kayıtlar eklenebilmektedir [35].

Switch'e gelen bir paket'e ait akış-tablosunda bir eşleşme olmadığı zaman, switch kontrolör ile güvenli kanal üzerinden iletişime geçerek bu paketle ilgili olarak ne yapılacağını *Table-miss* mesajı aracılığıyla sorar. Kontrolör de yeni bir akış-kaydı oluşturarak switch'e gönderir. Bu yüzden kontrolör kendisine gelen *Table-miss* mesajına cevap verebilmek için yeni bir akış-kaydı oluşturmaktan ve bu kuralı switch'e bildirmekten sorumludur. Paketlerin Meter'lardan geçebilmesi için akış-tablosunda yer alan instructions(komut) kısmına goto_Meter (Meter_id) ifadesinin eklenmesi gerekir.

Switch'lere akış-kaydı ekleme işlemleri, Floodlight kontrolöre eklenen flowSetter modülüyle ve oluşturulan flowRules.py dosyası aracılığıyla StaticEntryPusher.java modülüne erişim sağlanarak yapılmıştır.

4.2.1 Floodlight flowSetter modülü

Paketlerin Meter'lardan geçmesi için oluşturduğumuz flowSetter.java dosyasına ait kodlar Şekil 4.2'de gösterilmektedir. Eklenen akış-kaydına göre switch'e gelen paketlerden IPv4

protokolüne göre çalışan ve AF11 (DSCP=28 Hex) DSCP değerine sahip paketlerin önce Meter-1 den geçmesi daha sonrada 2 numaralı porttan çıkış yapması sağlanmıştır.

```
public void addFlow() {
List<OFInstruction> instructions = new ArrayList<OFInstruction>();
OFInstructionMeter meter = myOF13Factory.instructions().buildMeter().setMeterId(1).build();
OFActionOutput output = myOF13Factory.actions().buildOutput().setPort(2).build();
ArrayList<OFAction> actionList = new ArrayList<OFAction>();
actionList.add(output);
OFInstructionApplyActions
applyActions=myOF13Factory.instructions().buildApplyActions().setActions(actionList).build();
instructions.add(meter);
instructions.add(applyActions);
Match myMatch = myOF13Factory.buildMatch()
.setExact(MatchField.ETH_TYPE, EthType.IPv4)
.setExact(MatchField.IP_DSCP, IpDscp.DSCP_28).build();  // 28 = AF11
OFFlowAdd flowAdd=myOF13Factory.buildFlowAdd()
.setMatch(myMatch) .setInstructions(instructions).setOutPort(OFPort.of(outPort)).build();
// Akış kuralının Switch'e Gönderilmesi
IOFSwitch mySwitch =switchService.getSwitch(dpid);
mySwitch.write(flowAdd);}
```

Şekil 4.2 Floodlight akış-kaydı ekleme kodları

4.2.2 StaticEntryPusher Api ile akış-kaydı ekleme

Floodlight içerisindeki StaticEntry modülünde yer alan staticEntryPusher.java dosyasına Rest Api'ler aracılığıyla komut gönderilerek akış-tabloları ile ilgili kayıt ekleme, kayıt silme, kayıt listeleme vb. işlemler yapılabilmektedir [36]. StaticEntryPusher'a komut satırı veya oluşturulan bir python dosyası aracılığıyla erişilebilmektedir. Oluşturduğumuz flowRules.py python dosyası aracılığıyla şekil 4.3'te yer alan akış-kayıtları switch'lere eklenmiştir. Switch-1'e eklenen s1-flw-1 adlı akış-kaydına göre switch'e gelen paketlerden DSCP değeri AF11(0x28 Hex) olan ve UDP türündeki paketlerin Meter-1 den geçmesi ve daha sonra 2 numaralı porttan çıkış yapması sağlanmıştır.

```
import httplib
import json
class StaticEntryPusher(object):
...
pusher = StaticEntryPusher('127.0.0.1')
flow1 = {
    'switch':"00:00:00:00:00:00:00:01",
    "name":"s1-flw-1",
    "eth_type":"0x0800",
    "ip_tos":"0x28",
    "ip_proto":"0x11",
    "active":"true",
    "instruction_apply_actions":"output=2",
    "instruction_goto_meter":"1"
    }
pusher.set(flow1)
...
```

Şekil 4.3 StaticEntryPusher akış-kaydı ekleme kodları

4.3 Meter İstatistikleri Elde Etme

Meter'lardan istatistik elde etme işlemleri Floodlight kontrolörde bulunan istatistik modülü aracılığıyla veya DPCTL switch yönetim aracı ile yapılabilmektedir.

4.3.1 Floodlight istatistik modülü güncelleme

Floodlight kontrolörün içerisinde yer alan istatistik modülü; flow, port, queue, grup, Meter, table vb. birçok istatistiki bilgiyi sunabilmek için tasarlanmıştır. Fakat şu anda istatistik toplama aracı sadece port ve flow istatistiklerini sunabilmektedir [37]. Meter istatistikleri henüz verilmemektedir. Bu yüzden Floodlight içerisindeki istatistik modülünde yer alan statisticsCollector.java dosyasına Şekil 4.4'te kodları verilen MeterStatsCollector sınıfı eklenerek Meter istatistiklerinin elde edilmesi sağlanmıştır [38]; [39].

MeterStatsCollector sınıfı, ağda bulunan tüm switch'lerden barındırdıkları Meter'lara ait istatistikleri toplamak için oluşturulmuştur. OFMeterStatsRequest mesajı tüm switch'lere gönderilmektedir. Switch'ler de sahip oldukları tüm Meter'lara ait istatistikleri OFMeterStatsReply mesajı ile kontrolöre göndermektedir. Bu mesajın içerisinde Meter_id, ByteInCount, FlowCount, DurationSec, DurationNSec, BandStats değerlerini tutan bir dizi yer almaktadır. Gelen Meter istatistikleri içerisinde yer alan byte_count ve byte_band_count değerleri getValue methodu ile 10'luk sayı sistemine çevrilerek bit cinsinden değerler elde edilmiştir. Daha sonra verilerin anlaşılır hale getirilmesi için Bit-MByte dönüşümü yapılmıştır.

```
protected class MeterStatsCollector implements Runnable{

    public void run() {
        meterStats.clear();
        Map<DatapathId, List<OFStatsReply>> replies = getSwitchStatistics(switchService.getAllSwitchDpids(), OFStatsType.METER);
        for (Entry<DatapathId, List<OFStatsReply>> e : replies.entrySet()) {
            IOFSwitch sw = switchService.getSwitch(e.getKey());
            for (OFStatsReply r : e.getValue()) {
                OFMeterStatsReply psr = (OFMeterStatsReply) r;
                for (OFMeterStats pse : psr.getEntries()) {
                    if(sw.getOFFactory().getVersion().compareTo(OFVersion.OF_13) >= 0){
                        Pair<Long,DatapathId> pair = new Pair<Long,DatapathId>(pse.getMeterId(),e.getKey());
                        meterStats.put(pair, MeterStats.of(
                                e.getKey(),
                                pse.getMeterId(),
                                pse.getByteInCount(),
                                pse.getPacketInCount(),
                                pse.getFlowCount(),
                                pse.getDurationSec(),
                                pse.getDurationNsec(),
                                pse.getBandStats().get(0).getPacketBandCount().getValue(),
                                pse.getBandStats().get(0).getByteBandCount().getValue() ));
                    double byteCount =pse.getByteInCount().getValue();
                    double byteCountMByte=byteCount/1048576;
                    double byteBandCount=(pse.getBandStats().get(0).getByteBandCount().getValue());
                    double byteBandCountMByte=byteBandCount/1048576;

                    }
```

Şekil 4.4 Floodlight Meter istatistik kodları

4.3.2 DPCTL ile Meter istatistiklerini elde etme

DPCTL yazılımsal switch yönetim aracı kullanılarak Meter istatistikleri elde edilebilir [19]. Aşağıdaki komut satırı aracılığıyla s1 switch'inde yer alan bütün Meter'lara ait istatistikler elde edilmiştir.

$sudo dpctl unix:/temp/s1 stats-Meter

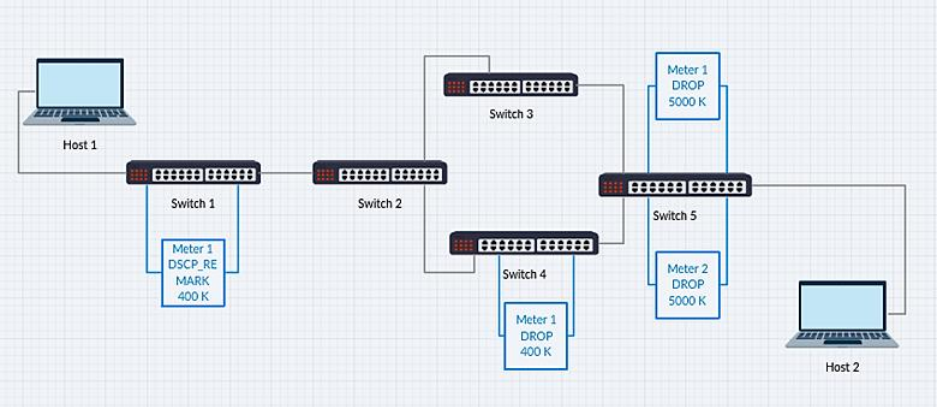
4.4 Test Ortamı ve Elde Edilen Sonuçlar

Bu çalışmada ağ yönetim aracı olarak Java dili ile geliştirilen Floodlight Kontrolör kullanılmıştır. Floodlight geniş bir geliştirici ve kullanıcı ağına sahip olmasının yanı sıra modüler yapısı ile birlikte hem gerçek switch'lerle hem de sanal switch'lerle çalışmamıza olanak sağladığı için tercih edilmiştir.

Test ortamını oluşturabilmek için yazılım tanımlı ağ modelleme simülatörü Mininet kullanılmıştır [20]. Mininet simülatöründe varsayılan olarak bulunan OVS ve OVSK switch'ler Meter özelliğini desteklememektedir. Bu yüzden çalışmamızda OpenFlow1.3 protokolü ile kullanılmaya başlanılan, grup ve Meter özelliklerinin desteklenebilmesi için geliştirilen, yazılımsal switch (OfSoftSwitch/CPqD/Bofuss) kullanılmıştır [17]. Bu switch'lere ait özellikler yazılımsal switch yönetim aracı DPCTL ile kullanılabilmektedir. Bu yönetim aracı sayesinde akış tablolarına akış-kaydı eklenenebildiği gibi switch'lerede Meter eklenebilmektedir. Ayrıca Meter yapılandırma ayarları ve istatistikleri de bu araç sayesinde elde edilebilmektedir [19].

4.4.1 Topoloji oluşturma

Mininet simülatörü içerisinde yer alan standart topolojilerin yerine python scriptleri aracılığıyla 2 host ve 5 switch'in yer aldığı bir topoloji tasarlanmıştır (Şekil 4.5).



Şekil 4.5 Topoloji

Aşağıdaki komut satırı aracılığıyla Şekil 4.6'da gösterilen seneryo.py dosyası çalıştırılarak topoloji gerçekleştirilmiştir.

$ sudo mn –-custom ~/mininet/custom/seneryo.py –topo=mytopo –Switch=user --controller=remote

```
from mininet.topo import Topo

class MyTopo( Topo ):

    def __init__( self ):

        Topo.__init__( self )

        # Switch ve Host'lar ekleniyor
        host1 = self.addHost( 'h1' )
        host2 = self.addHost( 'h2' )
        switch1 = self.addSwitch( 's1')
        switch2 = self.addSwitch( 's2')
        switch3 = self.addSwitch( 's3')
        switch4 = self.addSwitch( 's4')
        switch5 = self.addSwitch( 's5')
        # Bağlantılar ekleniyor
        self.addLink( host1, switch1)
        self.addLink( switch1, switch2 )
        self.addLink( switch2, switch3 )
        self.addLink( switch2, switch4 )
        self.addLink( switch3, switch5 )
        self.addLink( switch4, switch5 )
        self.addLink( switch5, host2 )
topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Şekil 4.6 Seneryo.py

4.4.2 Switch'lere Meter kurulumu

Çalışmamızda DPCTL yönetim aracı kullanılarak topoloji içerisinde yer alan switch'lere Tablo 4.1'de özellikleri belirtilen Meter'lar eklenmiştir.

Tablo 4.1 Switch'lere eklenen Meter'lar

Meter Tablosu			
Switch	**Meter Id**	**Meter Mod**	**Band Rate**
Switch 1	1	DSCP_REMARK	400 Kbps
Switch 4	1	DROP	400 Kbps
Switch 5	1	DROP	5 Mbps
Switch 5	2	DROP	5 Mbps

Topoloji içerisinde yer alan switch'lerden (Bkz. Şekil 4.5) Switch-1 ile trafik önceliklendirme ve trafik yönlendirmesi yapılacağı için aşağıdaki komut satırı aracılığıyla bu switch'e Dscp_remark modunda bir Meter eklenmiştir.

$sudo dpctl unix:/temp/s1 Meter-mod cmd=add, flags=1, Meter=1, dscp_remark: rate=400, prec_level=1

Switch-4 bant genişliği garantisi sağlamak ve trafik akışını rahatlatmak amacıyla kullanılacağı için bu switch'e aşağıdaki komut satırı aracılığıyla Drop modunda bir Meter eklenmiştir.

$sudo dpctl unix:/temp/s4 Meter-mod cmd=add, flags=1, Meter=1, drop: rate=400

Switch-5'e gelen paketlerin, düşürülme veya yeniden işaretleme işlemlerine tabi tutulmadan, miktarlarının ölçülmesi hedeflenmektedir. Bu yüzden Switch-3 ve Switch-4'ten gelecek paketler için Meter'ların işlem yapmaya başlayacakları sınır değerini 5 Mbps'e çıkartan aşağıdaki komut satırları aracılığıyla iki adet Drop modunda Meter eklenmiştir.

$sudo dpctl unix:/temp/s5 Meter-mod cmd=add, flags=1, Meter=1, drop: rate=5000

$sudo dpctl unix:/temp/s5 Meter-mod cmd=add, flags=1, Meter=2, drop: rate=5000

4.4.3 Akış-tablolarına kayıt ekleme

Floodlight kontroller içerisinde yer alan ve Rest Api'ler aracılığıyla kullanılan StaticEntry modülü, kullanıcılara bir OpenFlow ağındaki switch'lere akış-kaydı ekleme imkânı sunmaktadır[36]. Akış-kaydı ekleme işlemi terminal ekranında komutlar yazılarak yapılabileceği gibi Python dilinde yazılmış komut dizileri ile de yapılabilmektedir. Bu çalışmada Python komut dizileri kullanılarak switch'lerde yer alan akış-tablolarına kayıtlar eklenmiştir. Şekil 4.7'de python scriptleri ile oluşturulan flowAdd.py dosyasından bir kısım görülmektedir.

```
1  import httplib
2  import json
3  class StaticEntryPusher(object):
4  ...
5  pusher = StaticEntryPusher('127.0.0.1')
6  flow1 = {
7      'switch':"00:00:00:00:00:00:00:01",
8      "name":"s1-flw-1",
9      "eth_type":"0x0800",
10     "ip_tos":"0x28",
11     "ip_proto":"0x11",
12     "active":"true",
13     "instruction_apply_actions":"output=2",
14     "instruction_goto_meter":"1"
15      }
16 pusher.set(flow1)
17 ...
```

Şekil 4.7 Python kodları ile akış kaydı ekleme

Akış-kayıtları için oluşturulan python dosyası içerisinde kullanılan parametreler ve aldıkları değerler aşağıda açıklanmıştır. Topoloji içerisinde yer alan bütün switch'lere eklenen akış-kayıtları Tablo 4.2'de yer almaktadır.

- switch: Akış kaydının yapılacağı switch'i belirtmektedir. "00:00: 00:00: 00:00: 00:01" değeri ile 1 numaralı kimlik bilgisine (DatapathId) sahip olan switch için bir akış-kaydı yapılacağı belirtilmiştir.
- name: Akış-kaydı için verilen benzersiz bir isimdir. "S1-flw-1" değeri bir numaralı switch'e ait birinci akış-kaydı olduğunu belirtmektedir.
- eth_type: Bir eşleşme alanı (match) parametresidir. Kullanılan protokolü ifade eder. "0x800" değeri IPv4 protokolünü ifade etmektedir.
- ip_tos: Gelen paketin sınıfını belirtmek için kullanılan bir eşleşme parametresidir. "0x28" değeri AF sınıfından düşürülme önceliği düşük bir paketi temsil etmektedir.
- ip_proto: internet protokolü ile birlikte kullanılan diğer protokolleri (TCP, UDP, ICMP) ifade eder. "0x11" değeri UDP protokolünü temsil etmektedir.
- active : Boolean türünde değer alan bu parametre akış-kaydının aktif ya da pasif hale geçmesini sağlar. "true" değeri akış-kaydının aktif olduğunu gösterir.
- instruction_apply_actions: Eşleşme alanlarındaki kayıtlara uygun olarak gelen paketler için çalıştırılacak komutları belirtir. "output=2" değeri paketin 2 numaralı çıkış portundan iletiminin sağlanacağını belirtmektedir.
- instruction_goto_Meter: Switch'e gelen paketlerin Meter'lardan geçirilmesi için kullanılan komuttur. "1" değeri switch'te kurulu olan ilk Meter'ı (Meter_id=1) belirtir.

Tablo 4.2 Switch'lere eklenen akış-kayıtları

Akış-Kayıtları				
Switch	**Flow Id** (Akış No)	**Match Field** (Eşleşme Alanı)	**Action** (Eylem)	**Instruction** (Komut)
Switch 1	1	ip_tos: 0x28 (AF11)	Out: 2	goto_Meter_1
Switch 2	1	ip_tos: 0x28 (AF11)	Out: 2	-
Switch 2	2	ip_tos: 0x30 (AF12)	Out: 3	-
Switch 2	3	ip_tos: 0x32 (AF13)	Out: 3	-
Switch 3	1	ip_tos: 0x28 (AF11)	Out: 2	-
Switch 4	1	ip_tos: 0x30 (AF12)	Out: 2	goto_Meter_1
Switch 4	2	ip_tos: 0x32 (AF13)	Out: 2	goto_Meter_1
Switch 5	1	ip_tos: 0x28 (AF11)	Out: 3	goto_Meter_1
Switch 5	2	ip_tos: 0x30 (AF12)	Out: 3	goto_Meter_2
Switch 5	3	ip_tos: 0x32 (AF13)	Out: 3	goto_Meter_2

4.4.4 Trafik oluşturma

Mininet simülasyon ortamında Iperf aracı [23] kullanılarak Host1 --> Host2 yönünde 1200 Kbps bant genişliğinde, AF11 sınıfından (DSCP = 0x28 Hex) UDP paketleri gönderilmiştir.

Mininet'te *xterm h1 h2* komutuyla Host1 ve Host2'ye ait terminal ekranları açılmıştır. Daha sonra 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 -b 1200 –S 0x28* komutu yazılarak Host2'nin Host1'den 1200 Kbps bant genişliğinde UDP paketleri istediğini belirtiyoruz. Ayrıca bu paketlerin AF11 sınıfından olması istenmektedir. Bu trafik herhangi bir süre belirtilmediği için varsayılan değer olan 10 saniye boyunca sağlanmıştır.

4.4.5 Test ortamı ve istatistik modülü verileri

Switch1'de bulunan Dscp_remark modundaki Meter, kendisine gelen paketlerden 400 Kbps'i aşanları yeniden işaretleyerek DSCP değerlerini 0x30 (AF12) ve 0x32 (AF13) dönüştürmüştür. Böylelikle paketlerin farklı trafik önceliklerine sahip olmaları sağlanmıştır. Bant genişliği oranı olan 400 Kbps'yi aşmayan AF11 paketleri ise direkt olarak Switch2'ye gönderilmiştir.

Switch2'de Meter bulunamamaktadır. Switch2, kendisine gelen 400 Kbps bant genişliğine sahip AF11 paketlerini 2 numaralı porttan Switch3'e, 800 Kbps'lik yeniden işaretlenmiş AF12 ve AF13 paketlerini ise 3 numaralı porttan Switch3'e yönlendirmiştir. Böylelikle paketlerin DSCP değerlerine göre trafik bölmelendirme işlemi gerçekleştirilmiştir.

Switch3'te Meter bulunmamaktadır. Switch3, kendisine gelen AF11 sınıfından paketleri herhangi bir işlem yapmadan Switch5' e yönlendirmiştir.

Switch4'te bulunan Drop modundaki Meter 400 Kbps'lik bir bant genişliği garantisi vermektedir. Bu sınırı aşan paketler düşürülmektedir. Switch2'den gelen 800 Kbps'lik yeniden işaretlenmiş AF12 ve AF13 paketleri Meter'dan geçirildiğinde bant genişliği oranını aşan 400 Kbps'lik kısım düşürülecektir. İlk önce düşürülme önceliği yüksek olan AF13 paketleri daha sonra ise AF12 paketleri düşürülecektir.

Switch5'te Drop modunda ve 5 Mbps bant genişliği değerine sahip iki adet Meter bulunmaktadır. Bu Meter'lardan birincisi Switch3'ten gelen AF11 paketlerinin miktarının ölçülmesi için kullanılırken ikincisi ise Switch4'ten gelen AF12 ve AF13 paketlerinin miktarlarının ölçülmesi için kullanılmıştır. Switch5'e, Swtich3'ten AF11 paketleri ve

Switch4'te yapılan Drop işlemi sonrasında AF13 paketlerinin tamamı düşürüldüğü için sadece AF12 paketleri ulaşabilmiştir. Tablo 4.3'te Floodlight kontrolörde geliştirdiğimiz Meter istatistik modülünden alınan test ortamı verilerini gösterilmektedir.

Tablo 4.3 Elde edilen Meter istatistikleri

Meter İstatistik Tablosu					
Switch	**Meter Id / Meter Mod**	**Paket sayısı**	**Bit Miktarı (MByte)**	**Düşürülen / Yeniden işaretlenen paket ve bit miktarı**	
				Paket sayısı	**Bit Miktarı (MByte)**
Switch 1	1 / Remark	1032	1.488	692	0.998
Switch 4	1 / Drop	692	0.998	363	0.523
Switch 5	1 / Drop	340	0.491	-	-
Switch 5	2 / Drop	329	0.474	-	-

İstatistik modülünde elde edilen değerlere bakıldığında: Host1 --> Host2 yönünde gönderilen 1032 paketten 692 tanesinin Switch1'deki Meter tarafından yeniden işaretlendiği, 363 tanesinin Swhitch4'deki Meter tarafından düşürüldüğü ve toplamda 669 paketin Host2 ye ulaştığı görülmüştür. Gönderilen 1200 Kbps bant genişliğindeki trafiğin; Switch1'de önceliklendirildiği, Switch2'de bölmelendirildiği, Switch4'te ise sınırlı seviyede bir bant genişliğinin garanti edilebildiği görülmüştür.

5. SONUÇLAR ve ÖNERİLER

Hizmet kalitesi bilgisayar ağlarında farklı hizmet ihtiyaçlarına sahip uygulamaların trafiklerinin önceliklendirilerek gerekli kaynakları kullanabilmelerini sağlamak için gereklidir. Yazılım tanımlı ağlarda hizmet kalitesi sağlamak için kontrol düzlemi kullanılmaktadır. Mantıksal olarak merkezi yapıda olması ve tüm ağı tek noktadan yönetiyor olması bütünsel bir hizmet kalitesi politikasının uygulanmasını kolaylaştırmaktadır. Kontrolör ile ağ cihazları arasındaki iletişim genel olarak OpenFlow protokolüyle sağlanmaktadır.

Bu çalışmada yazılım tanımlı ağlarda hizmet kalitesi yapıları oluşturmak için kullanılan, OpenFlow1.3 protokolü ile desteklenmeye başlanılan, Meter özellikleri incelenmiş ve bu özellikler kullanılarak hizmet kalitesi yapılarının nasıl oluşturulabileceği ile ilgili olarak uygulamalar yapılmıştır. Ayrıca Floodlight Kontrolör için Meter ve istatistik modülleri oluşturulmuştur.

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.

Dscp_remark modu ile paketlerin başlık kısımlarında yer alan DSCP bitleri yeniden işaretlenmiştir. Yeniden işaretleme işlemiyle paketlerin düşürülme öncelikleri arttırılmaktadır. Bu da paketlerin farklı öncelik seviyelerine göre kategorize edilmelerini sağlamıştır. Yeniden işaretleme işlemi, trafik önceliklendirme ve trafik bölmelendirmesi olmak üzere iki farklı hizmet kalitesi yapısı oluşturmada kullanılabilmektedir.

Dscp_remark modunda Meter tanımlanırken belirtilen bant genişliği değerini aşan paketler yeniden işaretlenmektedir. Tanımlanan bant genişliği içerisinde kalan paketler en yüksek önceliğe sahiptir. Bant genişliği değerini aşan paketlerin ise düşürülme öncelikleri arttırılmaktadır. Dolayısıyla ağda oluşabilecek tıkanıklık durumlarında ilk önce düşürülme önceliği yüksek olan paketler düşürülecektir. Bu durumda yeniden işaretlenmeyen paketlerin ağ trafiğinde öncelikli olduğu görülmektedir. Dscp_remark modu ile paketlere önceliklendirme yapılarak bazı hizmetler için paket kaybı, gecikme, gecikme farkı gibi sorunların en aza indirilebileceği sonucuna varılmıştır.

DSCP değerleri akış tablolarında eşleşme kriteri olarak kullanılabilmektedir. Akış tablosundaki kayıtlar ile paketlerin başlık kısmında yer alan DSCP değerlerinin eşleşmesi durumunda paketler akış tablolarında ki kurallara göre farklı portlara yönlendirilebilmektedir. Bu da Dscp_remark modu sayesinde trafiğin yönlendirilebileceğini ortaya koymaktadır.

Yeniden işaretleme işlemlerindeki dezavantaj ise sadece AF(Assured Forwarding) sınıfı içerisinde işaretleme işleminin yapılabilmiş olmasıdır. Best Effort ve Expedited Forwarding (EF) sınıflarıyla bir işaretleme işlemi yapılamamıştır. Bu durumun sebebi ise kullanmış olduğumuz yazılımsal switch'in özelliklerinin yeterli olmayışıdır.

Meterlar ile birlikte gelişmiş kuyruk yapıları kullanılarak daha detaylı hizmet kalitesi kontrolü sağlanabilir. Bu yapı hizmet kalitesi destekli yönlendirme algoritmaları kullanıldığında daha verimli çalışacaktır.

KAYNAKLAR

[1] S. KEMP, The State Of Digital In April 2019, 25 April 2019. URL(erişim tarihi:12.05.2019) https://wearesocial.com/blog/2019/04/the-state-of-digital-in-april-2019-all-the-numbers-you-need-to-know.(2019).

[2] OpenFlow Switch Specification Version 1.3.0, ONF, Open Networking Foundation, 25 June 2012. URL(erişim tarihi:05.09.2017) https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf.

[3] RFC-2998 , Integrated Services Over Diffserv Networks,November 2000 URL(erişim tarihi:12.04.2019 https://tools.ietf.org/html/rfc2998).

[4] RFC-4594, DiffServ Hizmet Sınıfları Konfigürasyon Rehberi, Agust 2006. URL(erişim tarihi:12.04.2019) https://tools.ietf.org/html/rfc4594

[5] H. Krishna, N. L. M. V. Adrichem ve F. A. Kuipers, Providing Bandwidth Guarentees with OpenFlow, Symposium on Communications and Vehicular Technologies (SCVT), Mons/Belgium, 2016.

[6] H. Liaoruo, S. Qingguo, Z. Feng, C. Xiaoyu ve S. Wenjuan, Non-uniform bandwidth reservation for real-time streaming applications based on Meter table, IEEE 17th International Conference on Communication Technology (ICCT), Chengdu/China, 2017.

[7] T.-N. Lin, Y.-M. Hsu, S.-Y. Kao ve P.-W. Chi, OpenE2EQoS: Meter-based Method for End-to-End QoS of Multimedia Services over SDN, IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Valencia/Spain, 2016

[8] N. Kitsuwan ve E. Oki, Implementation of Traffic Splitting Using Meter Table in Software-Defined Networking, The Journal of Engineering, pp. 662-665, 2017

[9] SDN (Yazılım Tanımlı Ağlar) tanımı, ONF, Open Networking Foundation, URL(erişim tarihi: 5.01. 2019) https://www.opennetworking.org/sdn-definition.

[10] Floodlight Kontrolör, Project Floodlight, URL(erişim tarihi:05.01.2019) http://www.projectfloodlight.org/floodlight

[11] R. Izard, The Controller, 26 April 2016. URL(erişim tarihi: 10.01.2019) https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343548/The+Controller

[12] Floodlight Kurulum Rehberi, URL(erişim tarihi:10.09.2017) https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343544/Installation+Guide#InstallationGuide-Introduction.

[13] Floodlight Müdül Ekleme İşlemleri, URL(15.09.2017) https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343513/How+to+Write+a+Module.

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

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

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

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

[18] 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

[19] 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.

[20] Mininet, Network Emulatorü, URL(erişim tarihi: 15.03.2018) http://www.mininet.org.

[21] Mininet Geliştirme Platformu, URL(erişim tarihi:20.03.2018) https://github.com/ mininet

[22] Mininet Temel Özellikleri, URL(erişim tarihi: 22.03.2018) https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.

[23] Mininette Trafik oluşturma, Iperf Tool, URL(erişim tarihi:25.03.2018) http://iperf.fr

[24] Iperf Tool User Documentation, URL(erişim tarihi:30.03.2018) https://iperf.fr/iperf-doc.php

[25] J. Yan, H. Zhang, Q. Shuai ve B. L. a. X. Guo, HiQoS: An SDN-based multipath QoS solution, China Communications, Cilt 5, No. 12, pp. 123-133, 2015

[26] H. Krishna, Providing End-to-end Bandwidth Guarantees With OpenFlow, M.Sc. Thesis , Delft University of Technology, Delft,Netherland, pp. 10-13 , 2016

[27] RCF-2205, Resource ReSerVation Protocol (RSVP), IETF, Internet Engineering Task Force , URL(erişim tarihi: 12.02.2019) https://tools.ietf.org/html/rfc2205.

[28] S. Pillai, Understanding Differentiated Services (TOS) field in Internet Protocol Header, 20 November 2017. URL(erişim tarihi: 20.04.2019) https://www.slashroot.in/understanding-differentiated-services-tos-field-internet-protocol-header.

[29] RFC-791, Internet Protocol Features, IETF, Internet Engineering Task Force , URL(erişim tarihi: 12.02.2019) https://tools.ietf.org/html/rfc791#page-11.

[30] RFC-2474, Definition of DS Field, IETF, Internet Engineering Task Force , URL(erişim tarihi:13.02.2019) https://tools.ietf.org/html/rfc2474

[31] RFC-3168, The Addition of ECN to IP, IETF, Internet Engineering Task Force , September 2001, URL(erişim tarihi :14.02.2019) https://tools.ietf.org/html/rfc3168

[32] RFC-2597, Assured Forwarding PHB, IETF, Internet Engineering Task Force, URL(erişim tarihi: 16.02.2019) https://tools.ietf.org/html/rfc2597

[33] RFC-3246, An Expedited Forwarding PHB, IETF, Internet Engineering Task Force, URL(erişim tarihi: 16.02.2019) https://tools.ietf.org/html/rfc3246

[34] Meter Özelliklerinin Kullanmı, Project Floodlight, URL(erişim tarihi: 10.04.2018) https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/15040523/How+to+Use+OpenFlow+Meters

[35] OpenFlowJ Loxigen Library, Project Floodlight, URL(erişim tarihi: 22.04.2018) https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343547/How+to+use+OpenFlowJ-Loxigen#HowtouseOpenFlowJ-Loxigen-FlowMods

[36] StatikEntryPusher API, Project Floodlight, 2016, URL(erişim tarihi: 17.06.2018) https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343518/Static+Entry+Pusher+API.

[37] Floodlight İstatistik Modülü, Project Floodlight, URL(erişim tarihi:17.01.2019) https://github.com/floodlight/floodlight/tree/master/src/main/java/net/floodlightcontroller/statistics..

[38] Floodlight İstatistik Modülü Oluşturma, Project Floodlight, URL(erişim tarihi: 18.01.2019) https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/21856267/How+to+Collect+Switch+Statistics+and+Compute+Bandwidth+Utilization.

[39] Floodlight İstatistik Modülü , GitHub Platformu, URL(erişim tarihi: 18.01.2019) https://github.com/floodlight/floodlight/blob/master/src/main/java/net/floodlightcontroller/statistics/StatisticsCollector.java.

ÖZ GEÇMİŞ

Kişisel Bilgiler

Adı, soyadı	: ALİ PAK
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 13.06.1985 / Kahramanmaraş
Medeni hali	: Evli
Telefon	: 0 (505) 464 11 13
E-posta	: alipak13@yahoo.com.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Marmara Üniversitesi, Teknik Eğitim Fakültesi Bilgisayar ve Kontrol Öğretmenliği	2009
Ortaöğretim	Kahramanmaraş Anadolu Teknik Lisesi Bilgisayar Yazılım	2003

İş Deneyimi

Yıl	Yer	Görev
2011- ...	Milli Eğitim Bakanlığı	Bilişim Teknolojileri Öğretmeni

Yabancı Dil

İngilizce

Yayınlar

1. Ali PAK, İbrahim Taner Okumuş, International Symposium on Advanced Engineering Technologies (ISADET 2019) Mayıs 2-3, 2019, Kahramanmaraş, Turkey: Yazılım Tanımlı Ağlarda OpenFlow Meter Özelliklerinin Hizmet Kalitesi İçin Kullanımı ve Meter İstatistiklerinin Toplanması

Hobiler

Müzik, Fotoğrafçılık, Seyahat etmek