
GAZİ ÜNİVERSİTESİ
Gazi
1926

YAZILIM TANIMLI AĞLARDA İÇERİK TABANLI YÖNLENDİRME

Mustafa ATALAY

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

EKİM 2019

Mustafa ATALAY tarafından hazırlanan "YAZILIM TANIMLI AĞLARDA İÇERİK TABANLI YÖNLENDİRME" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Bilgisayar Mühendisliği Ana Bilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Dr. Öğr. Üyesi Mehmet DEMİRCİ

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi ……..……..

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

Başkan: Prof. Dr. Ayhan ERDEM

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi ...……

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

Üye: Doç. Dr. Sevil ŞEN

Bilgisayar Mühendisliği Bölümü, Hacettepe Üniversitesi ...……

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

Tez Savunma Tarihi: 04/10/2019

Jüri tarafından kabul edilen bu çalışmanın Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum

……..…….

Prof. Dr. Sena YAŞYERLİ

Fen Bilimleri Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

……..
Mustafa ATALAY
04/10/2019

YAZILIM TANIMLI AĞLARDA İÇERİK TABANLI YÖNLENDİRME

(Yüksek Lisans Tezi)

Mustafa ATALAY

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Ekim 2019

ÖZET

Sayıları gün geçtikçe artan IOT cihazlarının topladıkları veri miktarı da katlanarak artmaktadır. Bu durum verilerin, veri tiplerine göre özelleşmiş farklı makinalarda işlenmesi ihtiyacını ortaya çıkarmıştır. Makinelerin kendi işleyeceği veri tipinde olmayan verileri ilgili makinelere göndermesi gerekmekte, bunun sonucunda da veri iletimi problemi ortaya çıkmaktadır. Ayrıca makinelerin dışarıya göndereceği veriler için hedef makine bilgisine sahip olmaması da diğer bir problemi oluşturmaktadır. Bu çalışmanın amacı makineler arası ağ yönetimini kolaylaştırmak ve farklı tipteki verilerin ilgili hedeflere yönlendirilmesi yükünü makineler üzerinden almaktır. Bu çalışma kapsamında öncelikle konuya ilişkin alan araştırması yapılmış ve Mininet üzerinde bir ağ topolojisi oluşturularak bu ağa 4 adet ana bilgisayar eklenmiştir. Makineler kendilerinin işleyeceği veri tipinde olmayan verileri, etiketli IP paketleri ile ön tanımlı bir IP adresine gönderip SDN kontrolcü ile de bu etiketli IP paketlerinin ilgili makinelere yönlendirilmesi sağlanmıştır. Farklı hedeflere dağıtımı sağlanan farklı tipteki verilerin birbirleri arasında iletim önceliği olabilmektedir. Bu durum göz önünde bulundurularak QoS yaklaşımı ile veri iletiminde farklı kaynaklara sahip kuyruklar oluşturulmuş ve bu kuyrukların etkinliği analiz edilmiştir. Bir sonraki adımda makineler arasındaki veri iletimi hem TCP hem de UDP protokolleri kullanılarak test edilmiştir. Ayrıca veri iletiminin etkinliğini artırmak için gönderilecek veri tipine ait verilerin, belirli bir sayıya ulaştıktan sonra iletilerek veri iletim süresinin kısaltılması hedeflenmiştir. Gerçekleştirilen testlerin sonuçları dikkate alındığında QoS kullanılarak bazı veri tipleri için garanti altına alınan kaynak kullanımı ile veri iletimi etkinliğinin arttığı gözlemlenmiştir. Ayrıca önerilen pencere boyutu yöntemi için seçilen pencere boyutu parametresinin, veri iletiminde önemli bir etkiye sahip olduğu gözlemlenmiştir. Yapılan bu çalışma ile IoT cihazlarından toplanan farklı tiplerdeki verilerin içeriklerine göre YTA kullanılarak farklı hedeflere yönlendirildiği, içerik tabanlı yönlendirmeye farklı bir yaklaşım sunan etkin bir yönlendirme alt yapısı tasarlanmıştır.

Bilim Kodu	:	9240
Anahtar Kelimeler	:	YTA, Ryu kontrolcü, IPv4 paket etiketleme, Nesnelerin interneti
Sayfa Adedi	:	83
Danışman	:	Dr. Öğr. Üyesi Mehmet DEMİRCİ

CONTENT BASED FORWARDING IN SOFTWARE DEFINED NETWORKS

(M. Sc. Thesis)

Mustafa ATALAY

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

October 2019

ABSTRACT

Number of IOT devices are increasing day by day. This brings the increase in amount of data collected by the IOT devices. These increases have resulted in the need to process the collected data on different hosts specialized in the data types. The hosts require to send the data that is not the type to be processed by them to the relevant hosts. As a result, the problem of data transmission occurs. Furthermore, the fact that the hosts do not have the target host information for the data to be sent out is another problem. The purpose of this study is to facilitate the inter-host network management and to direct the different types of data to the relevant targets through the network. In this study, firstly literature review was done. Then a network topology on Mininet was created and 4 hosts were added to this network. Hosts send non-own type data to a predefined IP address with labeled IP packets. The SDN Ryu controller has been designed to direct IP packets to the respective hosts. The transmission of different types of data, which is distributed to different targets, may be the priority of transmission between each other. Considering this situation, queues with different sources have been formed in QoS approach and the efficiency of these queues have been analyzed. In the next step data transmission between hosts was tested with both TCP and UDP protocols. Furthermore, in order to increase the efficiency of data transmission, a design was made to transmit data of a type after reaching a certain number, and the data transmission time was aimed to be reduced. When the results of the tests are taken into consideration, it is observed that the efficiency of data transmission increases with the use of the sources guaranteed for some data types using QoS. In addition, it has been observed that the selected window size parameter for the proposed window size method has a significant effect on data transmission. With this study, an efficient routing infrastructure was designed in which different types of data collected from IoT devices are routed to different targets using SDN according to their contents and offering a different approach to content based routing.

Science Code	:	9240
Key Words	:	SDN, Ryu controller, IPv4 packet marking, IoT
Page Number	:	83
Supervisor	:	Assist. Prof. Dr. Mehmet Demirci

TEŞEKKÜR

Bu çalışma konusunun belirlenmesi ve yürütülmesindeki katkılarından dolayı değerli danışman hocam Dr. Öğr. Üyesi Mehmet DEMİRCİ'ye, çalışmalarım sürecinde zamanını ve yardımlarını benden esirgemeyen, beni sürekli destekleyen eşime ve aileme teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklamalar**
Gbps	Giga bits per second
Mbps	Mega bits per second

Kısaltmalar	**Açıklamalar**
API	Application Programming Interface
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
CPU	Central Processing Uni
ECN	Explicit Congestion Notification
EIGRP	Enhanced Interior Gateway Routing Protocol
ICPM	Internet Control Message Protocol
IHL	The Internet Header Length
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IoT	Internet of Things
LAGO	Latin American Giant Observatory
MAC	Media Access Control Address
MPLS	Multi-Protocol Label Switching
NDN	Named Data Networking
NGI	New Generation Internet
NVF	Network Function Virtualization
OSI	Open Systems Interconnection
ONF	Open Networking Foundation
OSPF	Open Shortest Path First

Kısaltmalar	Açıklamalar
OVSDB	Open vSwitch Database Management Protocol
POF	Protocol Oblivious Forwarding
SSH	Secure Shell
STP	Spanning Tree Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
ToS	Type of Service
TTL	Time to Live
UCAID	Univ. Corporation for Advanced Internet Development
UDP	User Datagram Protocol
QoS	Quality of Service
XSEDE	Extreme Science and Discovery Environment
vBNS	Very High-perf Speed Backbone Network Service
YTA	Yazılım Tanımlı Ağlar
IIoT	Industrial Internet of Things

1. GİRİŞ

Aklımıza gelebilecek her türlü nesnenin bir şekilde internete bağlanıp diğer cihazlarla iletişim halinde olduğu yapıyı anlatan kavram olan IoT, nesnelerin interneti olarak tanımlanmaktadır. Bu kavram ilk defa 1999 yılında özel bir şirket için yaptığı bir sunumda Kevin Ashton tarafından dile getirilmiştir [1]. O günden bugüne bu konuda yapılan araştırmalar ile IoT cihazlarının kullanım alanları ve kullanılan cihaz sayısı katlanarak artmıştır. Gartner'in yaptığı araştırmaya göre 2020 yılında internete bağlı 26 milyardan fazla cihaz olacağı tahmin edilmektedir [2]. Sayıları giderek artan bu cihazların topladıkları veri miktarları da artmaktadır. Ev otomasyon sistemleri, akıllı giysiler, akıllı spor aletleri, akıllı medikal ürünler gibi çok geniş kullanım alanına sahip bu ürünlerin topladıkları veriler de cihazların kullanım alanlarına göre çeşitlilik göstermektedir. Bu veri türü; bir dağda sıcaklık ölçümü yapan cihazlardan toplanan numerik bir sıcaklık bilgisi, bir akıllı futbol topu üzerindeki cihazdan toplanan top hareket bilgisi veya bitki verimini artırmak amacıyla toprağın nem ölçümü yapan bir cihazdan toplanan nem bilgisi olabilir. Bu bilgilerinin hepsinin bir analiz için toplandığı ve işlenip bunlardan faydalı bir bilgi çıkarılacağı düşünülürse verinin işlenme adımı oldukça önem arz etmektedir. İşlenecek veri miktarı büyüdüğü için verilerin işleneceği makinelere düşen görev de artmaktadır. Bu durum göz önünde bulundurulduğunda farklı türlerdeki verilerin farklı makinelerde işlenmesi ve her bir makinenin bir veri tipini işleyecek şekilde özelleşmesi fikri ortaya çıkmıştır. Böylelikle makineler işleyecekleri verilerin karakteristiklerine göre ayarlanabilir. Bu sayede veri analizi yapılacak makinelerde ihtiyaca yönelik yatırımlar yapılır ve gereksiz donanım kullanımı, dolayısıyla gereksiz maliyetler ortadan kaldırılır. Bir makineye farklı türlerde veri toplayan algılayıcıların bağlı olabileceğini düşünülürse makinelerin kendi işleyeceği türde olmayan verileri, bu veri türünü işleyecek özelleşmiş makinelere yönlendirmesi gerekmektedir. Bu da makineler arası veri iletimi problemini ortaya çıkarmaktadır. Ayrıca makinelerin kendi işleyeceği veri tipinde olmayan verileri göndereceği hedef makine bilgisine sahip olmaması da diğer bir problemi oluşturmaktadır. Son yılların popüler yaklaşımı olan Yazılım Tanımlı Ağlar da bu kısımda devreye girmektedir.

Yazılım Tanımlı Ağlar, ağa gelen paketlerin nasıl yönlendirileceğine karar veren ağın beyni konumundaki kontrol katmanı ile bu kontrol katmanının aldığı kararları uygulayan fiziksel yönlendirme cihazlarından oluşan veri katmanını birbirinden ayırarak esnek bir ağ yönetimi

yapabilmeyi sağlar. YTA, ağ yönetiminde yazılımın gücünü etkili bir şekilde kullanabilme imkânı sunması açısından geleneksel ağ yönetim çözümleri içerisinde önemli bir yere sahiptir. Bu yaklaşım sayesinde ağın merkezi bir konumdan bir kontrol yazılımı ile programlanması sağlanarak ağ yönetimi işleri ile veri iletimi işleri birbirlerinden ayrılır. Bu da ağ yönetimini kolaylaştırma, daha esnek yönlendirme kabiliyeti ve düşük maliyet gibi birçok avantaj sunmaktadır.

Yazılım Tanımlı Ağlarda ağa gelen bir paketin yönlendirilmesi 3 katmandan oluşan bir yapı ile sağlanır. En tepede bulunan uygulama katmanında tüm yönlendirme politikaları belirlenir. Bu katmanın altında paketin yönlendirme kararlarının alındığı kontrol katmanı bulunmaktadır. Uygulama katmanı ile kontrol katmanı arasındaki iletişim bir NorthBound arayüzü ile gerçekleştirilir. En alt katmanda ise kontrol katmanından bir SouthBound arayüzü ile aldığı kontrol katmanı kararlarının uygulandığı veri katmanı bulunmaktadır. Kontrol katmanı ile veri katmanı arasındaki ilk iletişim protokolü, bir SouthBound arayüzü protokolü olan OpenFlow protokolüdür. Bu protokol ağ anahtarlarını programlayabilen standart API'ler sunar. YTA, bu API'ler yardımı ile paket başlıklarındaki çeşitli alanları kullanarak yük dengeleme, filtreleme, maliyet azaltma ve güvenlik iyileştirmeleri gibi farklı amaçlar için hizmet vermektedir.

Bu çalışmanın amacı IoT cihazlarından toplanan farklı türlerdeki verilerin, türlerine göre işlenecekleri farklı makinelere yönlendirilebilmesi ve bu makineler arasındaki veri iletimi alt yapısının, Yazılım Tanımlı Ağların sunduğu avantajlarla oluşturulmasıdır. Makineler, gönderilecek veriler için herhangi bir hedef bilgisine sahip olmadığı için, ön tanımlı bir IP ile gönderilen paketlerin hedef bilgileri, merkezi bir kontrolcü ile dinamik değiştirilmesi gerekmektedir. Ayrıca verilerin gönderileceği makinelerde bir değişiklik olması durumunda bunun ağa kolayca uygulanabilmesi gerekmektedir. Merkezi bir kontrol yapısı bulunmayan, ağ ayarlamalarının manuel ve hataya açık bir şekilde yapıldığı geleneksel ağ yaklaşımları ile bu gereksinimleri karşılamak oldukça zor olacaktır. Bu durumlar göz önünde bulundurularak geleneksel yöntemlerin yerine, YTA ile etkili bir yönlendirme mekanizmasının oluşturulması amaçlanmıştır.

Bu çalışma kapsamında öncelikle YTA kullanılarak IP paket başlıkları üzerinde işlem yapıp tekrar bu paketleri yönlendiren yöntemlerle ilgili bir alan araştırması yapılmış ve ilgili bilimsel kaynaklar taranmıştır. Başlangıçta hedef bilgisi belli olmayan paketlerin ağda

dinamik olarak hedeflerinin belirlenmesi işlemleri genellikle YTA yük dengeleme uygulamalarında kullanılmaktadır. Bu tür uygulamalarda bir hedefe gidecek IP paketleri ön tanımlı bir IP adresine yönlendirilmekte ve YTA kontrolcüsü belirlenen hedef seçim algoritması ile hedefi belirlemektedir. Daha sonra paketin hedef IP adresini değiştirerek bu paketi yönlendirmektedir. Bu çalışmada gerçekleştirilecek uygulamada ise yük dengeleme uygulamalarında olduğu gibi ön tanımlı bir hedef IP adresi ile ağa gönderilen IP paketlerinin hedef bilgileri dinamik bir şekilde değiştirilmektedir. Yük dengeleme uygulamalarından farklı olarak hedef seçimi belirli bir hedef seçim algoritmasına göre olmayıp ilgili paket üzerindeki ToS bit alanında bulunan paketin taşıdığı verinin tipine göre oluşturulmuş etikete bakılarak hedef seçim işlemi yapılmaktadır. Bu işlemleri gerçekleştirebilmek için öncelikle IPv4 paketlerini, paketin taşıdığı veri tipine göre etiketleyen daha sonra ağa gönderilen bu paketleri, YTA Ryu kontrolcüsü kullanarak paketlerin etiketlerine göre ilgili hedeflere yönlendiren bir yöntem oluşturulmuştur. Oluşturulan bu yöntemin Mininet ortamında, Internet2 ağ topolojisinden esinlenilerek kurgulanmış bir ağ topolojisi üzerindeki etkinliği ve performansı simülasyonlarla gösterilmiştir. Ayrıca paketler için farklı iletim kuyrukları oluşturup verinin tipine göre paketler ilgili kuyruklara eklenip QoS işlemleri yapılmıştır.

Yapılan bu tez çalışması ile IoT ve 5G gibi gelişmekte olan teknolojiler için etkili bir içerik tabanlı yönlendirme yöntemi ortaya koyulmuştur. Kullanılan IP paketi etiketleme yöntemi ve ağ topolojisinin karmaşıklığı bakımından mevcuttaki benzer çalışmalar içerisinde önemli bir yere sahip olabilir.

Tez genel hatları ile şu kısımlardan oluşmaktadır. Bölüm 2'de farklı hedef seçim algoritmaları ve farklı YTA kontrolcüleri kullanarak gerçekleştirilen yük dengeleme uygulamalarına yer verilmiştir. Bölüm 3'de YTA mimarisi, geleneksel ağlar ile YTA karşılaştırması, YTA avantajları ile dezavantajları ve YTA'nın ana protokolü olan OpenFlow protokolü hakkında bilgi verilmiştir. Bölüm 4'de bu çalışma kapsamında kullanılmış araçlar ve yöntemler hakkında bilgiler verilmiştir. Bölüm 5'te çalışmanın uygulama kısmı olan paket etiketleme ve yönlendirme işlemlerinden bahsedilmiştir. Bölüm 6'da gerçek veri seti ile önerilen metodun testi, QoS testi gibi farklı testler yapılmış ve sonuçlar verilmiştir. Son olarak Bölüm 7'de genel bir özet ve çalışmayı değerlendiren bir sonuca yer verilmiştir.

2. KAYNAK ARAŞTIRMASI

Sayıları gün geçtikçe artan milyonlarca IoT cihazı tarafından üretilen veriler göz önünde bulundurulduğunda, bu cihazlar tarafından toplanan büyük miktardaki verinin gerçek zamanlı nasıl üstesinden gelineceği önemli bir problem olarak ortaya çıkmaktadır. Bulut teknolojisi; yüksek depolama kapasitesi, hesaplama gücü ve kaynak havuzdan isteğe bağlı kaynak erişimi avantajları ile bu probleme karşı en iyi çözümlerden biridir. Ancak tüm verileri hesaplamak ve saklamak için merkezi bir buluta göndermek, IoT cihazlarının sayısındaki artış da göz önünde bulundurularak tıkanıklık, yüksek gecikme süreleri ve düşük hizmet kalitesi gibi dezavantajları ortaya çıkarmaktadır [3]. Uç hesaplama yöntemi ile hesaplama, depolama ve ağ oluşturma yetenekleri ağın kenarına taşınarak bulut bilişimin bu dezavantajları telafi edilebilmektedir. Toplanan veriler merkezi bir bulut sunucusu yerine ağın kenarlarında buluna uç hesaplama sunucuları tarafından dağıtık bir şekilde işlenir. Bu uç hesaplama sunucuları; hesaplama, işleme ve geçici depolama özelliklerine sahiptirler. Ayrıca bulut sunucuları ile işbirliği yaparak daha fazla işlem, işleme ve depolama görevleri için verileri ilgili bulut sunucularına iletir [4].

Uç hesaplamanın kullanıldığı IoT hizmet ve uygulamaları; genellikle yük dengeleme ve mobilize gibi farklı görev ve hizmetleri yerine getirmek için büyük ölçekli platformlarda kullanılmaktadır. Bu amaç doğrultusunda, uç hesaplama bileşenleri ağın ihtiyaç ve çıkarlarını göz önünde bulundurarak bu görevler için uzmanlaştırılmıştır. Bu hizmetler gerçekleştirilirken tüm veriler iletişim kanalları aracılığıyla uygun sunuculara gönderilmektedir. Bununla birlikte ağ boyutu arttıkça bu tür heterojen verilerin iletilmesi zor olabilir ve bu nedenle gecikme, bant genişliği ve güvenlik bakımından uygulama talepleri dikkate alınarak bu verileri ilgili hedeflere iletmek için uygun yönlendirme mekanizmaları gerekmektedir [5].

Bu tez çalışması için yapılan alan araştırması iki bölümden oluşmaktadır. Öncelikle farklı algılayıcılardan toplanan verilerin bulutta bir sunucuda işlenmeyip bu verileri işleyecek özelleşmiş uç ana makinelerde işlenmesi işlemlerinin genel adı olan uç hesap, diğer bir adla Sis Bilişimin YTA ile birlikte kullanıldığı çalışmalar taranmıştır. Bu çalışmalarda genel olarak yapının ana mantığını anlatılıp altta, ağ katmanında yapılan yönlendirme mekanizmalarından bahsedilmemiştir. Ağ katmanında yapılan yönlendirme ile ilgili

yapılacak işlemler yapı bakımından yük dengeleme uygulamaları ile benzerlik göstermektedir. Yük dengeleme uygulamalarında kaynaktan çıkan paket önce yük dengeleyiciye gelir, burada farklı algoritmalarla paketin gideceği hedef ana bilgisayar belirlenerek paket yeniden yönlendirilir. Bu tez çalışmasında da algılayıcılardan toplanan verilerin ilgili ana bilgisayarlara dağıtımı yapılırken paketler ağa ön tanımlı bir IP adres ile gönderilir ve kontrolcü bu paketlerin gideceği ana bilgisayarları tespit ederek tekrar paketi ilgili ana bilgisayarlara yönlendirir. Bu bakımdan alan araştırmasının ikinci kısmını YTA ile gerçekleştirilmiş yük dengeleyici uygulamaları oluşturmaktadır. Bu bölümde öncelikle YTA ile uç hesaplama uygulamaları ile ilgili alan araştırmalarına yer verilmiş, sonra yine YTA ile gerçekleştirilmiş yük dengeleme uygulamaları ile ilgili alan araştırmalarına yer verilmiştir.

2.1. YTA ile Uç Hesaplama Uygulamaları

Okay ve Özdemir çalışmalarında [5], daha sık gerçekleşen olaylar için yerel aksiyonlar alan uç hesaplama kontrolcüleri ile daha seyrek gerçekleşen küresel olaylar için bulut kontrolcülerinin aksiyon aldığı, YTA tabanlı hiyerarşik bir uç hesaplama mimarisi önermiştir. Önerilen bu mimari, açık kaynak kodlu YTA emülatörü olan Mininet üzerinde gerçekleştirilmiştir. Mininet üzerinde Python tabanlı YTA kontrolcüsü olan POX'u kullanarak OpenFlow protokolü ile anahtarlar ve kontrolcüler arası iletişim sağlanmıştır. Ayrıca kaynaktan hedefe giden en kısa yolu bulmak için Dijkstra algoritması kullanılmıştır. Önerilen mimariyi değerlendirmek için başlangıç seviye bir değerlendirme yapılmıştır. Bu kapsamda Mininet'te 127 anahtar, 2 iletişim ana bilgisayarı ve değişken sayıda kontrolcü ile basit bir YTA ağaç topolojisi oluşturulmuştur. Önerilen yöntemin performansını değerlendirmek için kontrolcü sayısını ve ağdaki diğer bazı parametreler değiştirilerek iki senaryo oluşturulmuştur. Yapılan test sonuçlarına göre çalışmada önerilen hiyerarşik YTA tabanlı ortamın, yönlendirme gecikmesini ve veri iletim yükünü azaltma potansiyeli görülmüştür.

YTA ile uç hesaplama ile ilgili başka bir çalışmada Kaur ve arkadaşları [6]; nesnelerin internetinin ortaya çıkışının, bulut ortamında gerçek zamanlı büyük veri depolama, erişim ve işleme yolunu açtığını belirtmiştir. IoT'de akıllı telefonlar, kablosuz vücut algılayıcıları ve akıllı sayaçlar gibi çeşitli cihazların ürettiği büyük veriler yakın gelecekte çok büyük miktarlara ulaşmaktadır. Bu nedenle bu büyük miktardaki veriyi daha ileri işlemler için uzak bulut platformlarına iletmek ağ tıkanıklıklarına sebep olmaktadır. Bu da IoT'deki çeşitli

uygulamalar için genel hizmet kalitesini etkileyen gecikme sorunlarına yol açmaktadır. Bu zorluklarla başa çıkabilmek için, son zamanların popüler konusu olan kenar hesaplama yaklaşımı gündeme gelmektedir. Kenar hesaplama, rekabetten ziyade bulut bilişimin tamamlayıcısı olarak görülmektedir. Bulut ve son aygıtlar arasındaki işbirliği ve etkileşim, IoT ortamındaki çeşitli uygulamalar için hizmet kalitesini sürdürmenin yanı sıra enerji tüketimini de azaltmaya yardımcı olabilecektir. Ancak, uç aygıtlar ve bulut sunucuları arasında çok sayıda geçiş, temel ağlarda tıkanmaya neden olabilmektedir. Bu problemin üstesinden gelmek için, programlanabilir ve ölçeklenebilir bir ağ paradigması olan YTA ile uygulanabilir bir çözüm ortaya atılmıştır. İncelenen bu çalışmada IoT ortamında büyük veri akışını idare etmek için YTA tabanlı bir kenar bulut etkileşimi sunulmuştur. Burada YTA etkili bir ara katman desteği sağlamaktadır. Önerilen algoritmanın temel iki optimizasyon hedefi olmakla birlikte, bu hedefler; enerji verimliliği ile gecikme arasındaki denge ve enerji verimliliğe ile bant genişliği arasındaki dengedir. Uç hesaplama yapmadan, tüm hesaplama yükü bulut üzerine bindirildiği durumda aşağıdaki sonuçlar ortaya çıkacaktır;

- Uç cihazlardaki kaynakların sınırlı olmasından dolayı, yükün büyük kısmı bulut tarafına kaydırılacaktır.
- Uç düğümdeki cihaz kaynaklarına fakla yüklendiğinde, cihazın tükettiği enerji daha da artacaktır.
- Bulut ve uç düğümler arasında gecikmeye duyarlı ve kaynak odaklı taleplerin sınıflandırılması performansı artırıp enerji tüketimini azaltmaktadır.
- Uç cihazların mobilitesi, kaybolan cihazların izlenebilmesi nedeniyle enerji tüketimi için bir zorluktur.
- Uç cihazlar ile bulut arasındaki çok sayıda geçiş ve iletişim, altta yatan ağlar üzerinde ek bir yüke yol açacaktır.

Yukarıda sıralanan zorlukların üstesinden gelebilmek için incelenen bu çalışmada son teknoloji bir bulut etkileşimi mimarisi sunulmuştur. Uç bulut etkileşim mimarisi üç katmanda incelenmiştir. Bunlar;

1. Bulut hesaplama katmanı
2. Kenar hesaplama katmanı
3. Ağ katmanı

Olmak üzere 3 katmandır. İncelenen çalışmada önerilen yaklaşımın genel mimarisi, kenar bulut etkileşiminin başarılı bir şekilde uygulanmasına yönelik temel olan ağ katmanı tarafından desteklenen katman yazılımına dayanmaktadır. Verimsiz bir ağ, aşırı enerji tüketimine ve yüksek gecikmeye neden olmaktadır. Dahası standart protokoller ve mimarilere dayalı geleneksel ağlara olan bağımlılık, uç ve bulut arasındaki etkileşimin zorluklarının üstesinden gelmede yetersiz kalmaktadır. Bu sebeple de performansı iyileştirerek enerji tüketimini azaltmak için kenar ve bulut aygıtları arasındaki büyük miktarda trafikle başa çıkmak için programlanabilir, ölçeklenebilir, esnek, sağlayıcı bağımsız ve yeniden yapılandırılabilir bir ağ mimarisi ihtiyacı dile getirilmektedir. Bu kapsamda yapılan genel işlemler üç gruba ayrılmıştır. Bunlar; akış sınıflandırma, YTA'nın kontrol katmanı vasıtası ile kontrol mantığının seçimi ve enerji odaklı akış zamanlama ve yönlendirme sağlamak için kontrol mantığının YTA ile yürütülmesi şeklinde sıralanmaktadır. İncelenen bu çalışma, kullanılan yöntemden ziyade uç hesaplamanın önemini anlatması bakımından kritiktir. Önerilen yöntem doğrultusunda gerçekleştirilmiş test sonuçları çalışmada ayrıntılı anlatılmaktadır [6].

Li ve arkadaşları çalışmalarında [7], son yıllarda, Endüstri 4.0 ve Endüstriyel Nesnelerin İnterneti (IIoT) ile güçlendirilmiş akıllı fabrikalar, hem akademi hem de sanayi için önemli bir konu haline geldiği belirtilmiştir. IIoT sistemlerinde, farklı akıllı cihazlar arasında farklı gecikmelerle veri değişim gereksinimi gittikçe artmaktadır. İncelenen çalışmada, geleneksel yöntemlerin sınırlarının üstesinden gelerek bu problemi çözmek için yazılım tanımlı ağların uç hesaplama yöntemlerine dâhil edilmesinin gerekliliği dile getirilmiştir. Bu kapsamda IIoT sistemleri için yazılım tanımlı ağlar ve uç hesaplamanın da dâhil olduğu uyarlanabilir bir iletim mimarisi önerilmektedir. Farklı gecikme sınırlamaları olan veri akışlarına göre gereksinimler; sıradan akış ve önceliği olan akış olarak ikiye ayrılmıştır. Sıradan akış durumunda, nesnelerin internetinde zaman kısıtlamalarını karşılayan tüm yolların bulunacağı bir algoritma tanımlanmıştır. Daha sonra zaman sınırı, trafik yükü dengesi ve enerji tüketiminin toplamı dikkate alınarak yol farkı derecesi hesaplanıp hesaplanan bu değere göre optimum bir rota seçilmektedir. Önceliği olan akış durumunda ise, sıradan akış durumunda kullanılan algoritmanın sunduğu zaman yeterli değilse düşük gecikme süresi elde etmek ve etkili bir iletim yolu oluşturmak için ikinci bir algoritma oluşturulmuştur. Çalışmadaki yaklaşımın ana çalışma süreci aşağıdaki gibi tanımlanmıştır. İlk olarak IIoT ağındaki her sıradan düğüm, çalışan gerçek uygulamalara göre verileri küme başlarına ya da sıradan düğümlere iletilmiştir. Tüm kümede veri toplama işlemi tamamlandıktan sonra her

bir küme başı ilgili bilgileri sink düğümüne yada baz istasyonuna iletmiştir. Ardından sink düğümü bu bilgiyi buluta göndermiştir. İkincil olarak tüm ağın kontrolünün esnek bir şekilde yapılabilmesi için ağdaki tüm cihazların durumu SDN kontrolcüsüne gönderilerek veri tabanında ilgili verinin haritalama işlemi yapılmıştır. YTA uygulama katmanı değiştirildikten ve belirli uygulama işlevi ayarlandıktan sonra, YTA kontrolcüsü parametreleri toplayıp görevleri uç hesaplama sunucularına yüklemiştir. Üçüncül olarak iletim yolunun değiştirilmesi gibi bir görev gerçekleştirileceğinde; uç hesaplama sunucuları, güç ve atlama yolu gibi ağ parametreleri optimize edilir ve optimizasyon sonuçları YTA kontrolcüsü ve IIoT cihazlarına gönderilmiştir. Son olarak, kontrol verilerinin iptalinden sonra IIoT cihazları kendi ağ koşullarını değiştirmektedir. Önerilen çalışma MATLAB ile gerçekleştirebilecek bir simülasyon ortamında oluşturulmuştur. İlgili metotların simülasyonu için çoklu iş parçacıkları kullanılmıştır. Ağ düğümlerinden düzenli olarak ağ parametrelerini toplayan YTA kontrolcüsünü simüle etmek için bu iş parçacıklarından biri kullanılmıştır. Gerçekleştirilen simülasyonlara göre önerilen yöntem; ortalama gecikme, verimlilik, yol farkı derecesi ve indirme süreleri bakımından ilgili yöntemlere göre daha iyi bir performans gösterdiği belirtilmiştir.

2.2. YTA ile Yük Dengeleme Uygulamaları

Alan araştırmasının ikinci kısmını oluşturan bölümde gideceği hedef bilgisi belli olmayan bir IP paketinin, bir OpenFlow anahtarında paket içeriğine bakılarak hedef IP adresinin tespit edilip paketin yönlendirilmesi işlemleri ile ilgili kaynaklar taranmıştır. Ön tanımlı bir IP adresi ile ağa gönderilen paketin, paket içeriğine göre YTA kullanılarak dinamik bir şekilde farklı hedeflere yönlendirilmesi ile ilgili kaynaklar ve farklı çalışmalar incelenmiştir. YTA ile paketlerin hedef IP adreslerini değiştirerek farklı hedeflere yönlendirilmesi işlemi genel olarak YTA yük dengeleme çalışmalarında kullanılmıştır. Yük dengeleme işlemlerinde yönlendirme işlemi, hedef sunucuların yük durumlarına göre yapılmaktadır. Yük dengeleme işlemleri ile bu tez kapsamından yapılan çalışma, hedef ana bilgisayarların yaptığı iş bakımından farklılık göstermektedir. Yük dengeleme çalışmalarında hedef sunucular, aynı hizmeti veren bir sunucu kümesi gibi çalışmaktadır. Örneğin bir Web sayfası hizmeti veren bir sunucu topluluğunun önünde duran bir OpenFlow anahtarı, kendine gelen Web sayfa sorgularını, arkadaki Web sayfalarını sunan sunucuların yük durumlarına göre, paketlerin hedef IP adreslerini değiştirerek farklı sunuculara yönlendirmektedir. Ancak her bir ana bilgisayarın farklı türde veriyi işleyeceği bu tez kapsamındaki çalışmada, ağda bulunan ana

bilgisayarlar hem sunucu hem de istemci görevi görecek şekilde tasarlanmıştır. Bu yönüyle YTA yük dengeleme çalışmaları bu çalışmadan ayrılsa da temel prensip olarak aynı işlemler yapılmaktadır. Bu sebeple yapılan kaynak taraması kapsamında, YTA kullanılarak gerçekleştirilen yük dengeleme çalışmaları; çalışmada kullanılan YTA kontrolcü seçimi, benzetim ortamı için kullanılan ağ topolojisi seçimi ve ağda kullanılan yönlendirme algoritmaları bakımından incelenmiştir [7].

Prakash ve Deepalakshmi [8] sunucu tabanlı dinamik bir yük dengeleyici mimarisi üzerinde çalışmışlardır. Bu çalışmada ağlarda artan trafik yükünü etkili bir şekilde yönetebilmek için yük dengeleme işleminin anahtar role sahip olduğunu belirtip OpenFlow anahtarları ile Floodlight kontrolcü kullanarak alternatif bir yük dengeleme mimarisi önerilmektedir. CPU kullanımı, Memory kullanımı ve aktif bağlantı parametrelerini kullanarak sunucular için bir yük hesabının yapıldığı bu yöntemle gelen isteklerin, hesaplanan bu yük durumuna göre yönlendirilmesi planlanmıştır. Hedef sunucu seçiminin, hedef sunucuların yük durumlarına göre yapıldığı bu yöntemin uygulanması için 2 OpenFlow anahtarın birbirlerine doğrusal bağlandığı bir ağa, 2 sunucu ve 4 istemci eklenerek ağ topolojisi oluşturulmuştur. 2 OpenFlow anahtardan oluşan bu ağda paket yönlendirme işlemini karmaşık hale getirecek bir döngü bulunmadığı için herhangi bir yönlendirme algoritmasına ihtiyaç duyulmamıştır. Çalışmanın sonuncunda da önerilen dinamik yük dengeleme yöntemiyle diğer yük dengeleme yöntemlerine göre daha başarılı test sonuçlarının elde edildiği belirtilmiştir.

Bir başka çalışmada Hamed ve arkadaşları [9] ağ akışlarının yönetiminin zor ve donanım bağımlı cihazlar olduğu için pahalı olan geleneksel yük dengeleyiciler yerine bu limitleri ortadan kaldıracak SDN tabanlı bir yük dengeleyici yaklaşımı üzerinde çalışmıştır. OpenFlow Ryu kontrolcü kullanılarak gerçekleştirilen bu yöntemde hedef sunucu seçimi sunucuların bant genişliği tüketimlerine göre yapılmıştır. Bant genişliği tüketimlerine göre hedef sunucuların belirlendiği bu yöntemin uygulaması için 1 istemci ve 2 sunucu kullanılmıştır. Sadece bir tane OpenFlow anahtara bağlanan bu 2 sunucu ve 1 istemci arasında paket yönlendirme işlemi, sunucu ve istemcilerin bağlı olduğu portlara manuel olarak yapılabileceği için her hangi bir yönlendirme algoritması kullanılmamıştır. Araştırmacıların yaptıkları test sonuçlarına göre önerilen yöntemin, Round-Robin ve bağlantı sayısına göre yapılan dengeleme yöntemleri ile karşılaştırıldığında daha performanslı olduğu belirtilmiştir.

Kaur ve arkadaşları [10] çalışmalarında, ağlarda istemcilerin oluşturduğu trafik yükünün, birden fazla sunucu ile bunların arasında yük dengeleme yapılarak kaldırılabileceğini ifade etmektedir. Diğer bilimsel çalışmalarda olduğu gibi, donanımsal yük dengeleyicilerin pahalılığı dile getirilerek YTA'nın avantajlarından bahsedilmiştir. Bu çalışmada YTA POX kontrolcüsü kullanılarak Round-Robin yük dengeleme yöntemi ve rastgele yönteminin karşılaştırılması yapılmıştır. 1 adet OpenFlow anahtara 6 adet istemci ve 3 adet sunucunun bağlanarak oluşturulduğu ağ topolojisinde, yönlendirme işlemi OpenFlow anahtarın ilgi portlarına manuel olarak yapıldığı için herhangi bir yönlendirme algoritmasına ihtiyaç duyulmamıştır. Çalışma sonuçlarına göre Round-Robin yönteminin rastgele yöntemine göre daha performanslı çalıştığı gözlemlenmiştir. Yukarıda kaynak araştırması kapsamında incelenen yöntemlerin her birinde benzetim çalışmaları Mininet aracı kullanılarak yapılmıştır.

Wang ve arkadaşları çalışmalarında [11], günümüz veri merkezlerindeki çoklu sunucular üzerinde çalışan çevrimiçi hizmetlerin sunumu için istemci isteklerini farklı sunuculara yönlendiren özel yük dengeleyicilerin pahalı ve hata durumlarında tıkanıklık noktası haline geleceğini belirtmiştir. OpenFlow protokolü ile alternatif bir yaklaşımla ağ trafiğinin bir denetleyici tarafından farklı sunuculara dağıtılabileceği dile getirilmiş ancak bununla birlikte her müşteri için ayrı bir kural oluşturmanın denetleyicide büyük miktarda istemci trafiği oluşturacağına dikkat çekilmiştir. Bu yüzden kontrolcünün joker kuralları için anahtar desteğini kullanarak büyük miktardaki veri trafiğini farklı sunuculara daha ölçeklenebilir bir şekilde yönlendirebileceği fikri öne atılmıştır. Bu kapsamda joker kurallarını hesaplayıp mevcut bağlantıları kesmeden yük dengeleme politikalarındaki değişiklikleri otomatik olarak yapan algoritmalar önerilmektedir. Trafik bölümlemesi için önerilen algoritmaya göre; yalnızca OpenFlow anahtarlarında bulunan özelliklere dayanarak istemci trafiği yük dengeleme ağırlıkları ile orantılı olarak bölünmektedir. Aynı TCP bağlantısından başarılı paketlerin aynı sunucuya iletilmesini sağlamak amacıyla anahtara istemci IP adresi ile eşleşme kuralları yüklenmiştir. Bu tez çalışması kapsamında yapılan alan araştırmalarında hedef sunucu seçim işlemi için daha fazla ayrıntıya girilmemiştir. NOX OpenFlow kontrolcüsü kullanılarak gerçekleştirilen bu çalışma, Mininet emülatör ortamında 3 sunucu, 2 OpenFlow anahtar ve bir takım istemciden oluşturulmuştur. Oluşturulan NOX uygulaması ile bu 2 OpenFlow anahtarına kurallar yüklenmiştir. Bu anahtarlardan biri istemci trafiğini çoklama ve tekleme işlemi için ağ geçidi olarak kullanılırken diğer anahtar da sunucular üzerindeki yükü ayırmıştır. Çalışmada ayrıca karmaşık ağ topolojilerinde etkili bir

yönlendirme işlemi için en kısa yol algoritması da kullanılmıştır. Yapılan değerlendirmeye göre sistemin gerçek hedef trafik dağılımındaki değişikliklere uyum sağlayabildiği ve kontrolcüye gönderilen birkaç paketin verimlilik üzerine en az etkiye sahip olduğu belirtilmiştir.

Koerner ve Kao çalışmalarında [12]; yük dengeleyicilerin, büyük ağlarda giriş noktası görevi yaptığı ve ağın kullanılabilirliği ile performansı üzerine büyük etkiye sahip olduğu belirtilerek ağın performansı için önemli bir role sahip olduğunu vurgulamıştır. İncelenen bu çalışmada mevcut yük dengeleyicilerin çoğunlukla özel donanım bileşenleri olarak uygulandığı bunun yanında OpenFlow kontrolcülerine dayanan ve belirli bir donanıma gerek duymadan birden fazla hizmetin yükünü işleyebilen bir yük dengeleyici yaklaşımı benimsenmiştir. Bu yaklaşımla ağı ve yük dengeleme işlevselliğini bütünleştirerek bakım eforunu azaltma amaçlanmıştır. Bu amaç için özel olarak uyarlanmış yük dengeleme algoritmaları sağlanarak verimlilik artırılmaya çalışılmıştır. Farklı kontrolcülere farklı görevler yüklenmiş, örneğin; bir kontrolcü, sunucu düğümleri ve diğer ağ bileşenleri arasındaki standart ağ trafiğini yönetirken, başka bir kontrolcü web sunucularının yük dengelemesini yapmış ve diğer bir kontrolcü de e-posta sunucularının yük dengelemesini gerçekleştirecek şekilde bir tasarım kurgulanmıştır. İlgili çalışmada OpenFlow NOX kontrolcüsü kullanılarak bir yönlendirme algoritması oluşturulmuştur. Yönlendirme algoritması olarak OSPF kullanılan bu algoritmada hedef sunucu seçimi RoundRobin [13] yöntemi kullanılarak yapılmıştır. Paket yönlendirme için ayrı bir yönlendirme algoritması kullanıldığı için topolojiden bağımsız bir tasarım gerçekleştirilmiştir. Kontrolcü, yük dengeleme işlemini, anahtarı farklı ara yüzlere sahip bir cihaz gibi yöneterek yapmaktadır. Bu cihaz ile ilişkili IP adresleri için ARP sorguları cevaplanmış ve uç ana bilgisayarın IP adresi, bu cihazın IP ve MAC adresi ile değiştirilerek istek sunucu havuzunun dışındaki sıradaki ana bilgisayara gönderilmiştir. Tasarlanan bu algoritma üç ana kısımdan oluşmaktadır. Birinci kısımda işlenmeyecek olan LLDP gibi tüm paketler düşürülmektedir. İkinci kısımda ARP isteğini yakalayıp sanal yük dengeleyiciden dönüyormuş gibi ARP isteği anahtardan geldiği yere geri gönderilmektedir. Üçüncü ve son kısımda ise giden ARP cevabına göre hedef IP adres ve MAC adresi ayarlanıp paket ağa gönderilmektedir. İşlenen tüm istemci istekleri; kaynak IP adresi, hedef IP adresi ve port bilgisi sunucudan gelen cevap yazılıncaya kadar bağlantılı listelerde tutulmaktadır. Bu çalışmada uygulanan istemciden gelen ARP isteklerinin manipüle edilmesi yöntemi bu tez çalışmasında kullanılan yönteme oldukça benzemektedir. Servis türüne göre bir yük dengeleme yaklaşımının önerildiği bu

çalışmanın sonuç kısmında önerilen yöntem için dikkate değer avantajlar listelenmiştir.

Long ve arkadaşları çalışmalarında [14], hızla büyüyen ağ uygulamaları ile birlikte veri merkezlerindeki ağlarda yükün dengelenmesinin kilit bir işlem olduğunu vurgulamıştır. Belirli akışların ağ üzerindeki yollarını tanımlayabilmek için her kullanıcıya bu akışlar için programatik bir kontrol sağlayabilen OpenFlow protokolünün, bu yük dengeleme sorununun çözümünde önemli bir aday olabileceği belirtilmiştir. Bununla birlikte ağ yapılanması veri iletimi sırasında değişebileceği göz önünde bulundurularak, OpenFlow'a dayanan mevcut çözümlerin başlatma aşamasında yalnızca statik bir yönlendirme yolu bulunması performans kaybına neden olmaktadır. Buna çözüm olarak da incelenen çalışmada iletim sırasında trafiği dinamik olarak dengeleyebilmek için yeni bir yol geliştirme algoritması olarak tanımlanan LABERIO önerilmiştir. İki farklı ağ mimarisinde yapılan testlere göre önerilen yöntemin RounRobin gibi geleneksel yük dengeleme algoritmalarına göre %13'e kadar iletim süresini azaltarak daha iyi performans gösterdiği tespit edilmiştir. LABERIO temelde meşgul bağlantıların yerine alternatif bağlantıları bulması için oluşturulmuş bir algoritmadır. Bu algoritma ile yük dengelenerek daha yüksek doğruluk elde edilmiştir. Kontrolcüye ana bilgisayar h1'den diğer bir ana bilgisayar h7'ye bir akış isteği geldiği zaman öncelikle akışın geçmesi için geçici olarak en uygun olan başlangıç yolunu hesaplar. Tüm olası yolların içerisinden max-min politikası MMRCS stratejisi [15] kullanılarak hangi yolun kullanılacağına karar verilmektedir. Daha sonra dosya transferi boyunca ağ durumu kontrol edilerek ihtiyaç durumunda LABERIO algoritması uygulanmaktadır. Algoritma, OpenFlow NOX kontrolcüsü kullanılarak gerçekleştirilmiştir. A ve B isimlerinde iki ağ topolojisi oluşturulmuştur. 10 adet anahtar ve 16 adet ana bilgisayarın bağlandığı A topolojisi incelenen bu çalışmada kullanılan genel ağ topolojisidir. B topolojisi ise yapısı farklı ölçekler için genişletilebilir ve ayrıca iki işlem düğümü arasında gereksiz düğümlerin de bulunduğu büyük ölçekli bir ağ topolojisidir. Kullanılan topolojiler için harici yönlendirme algoritmalarının da kullanıldığı bu çalışma karmaşık topolojide gerçekleştirilmesi bakımından önemlidir.

Li ve Pan çalışmalarında [16], veri merkezi ağlarının, veri merkezi içinde yoğun bir şekilde birbirine bağlı ana bilgisayarların veri iletim talebini karşılamak için tasarlandığını ifade etmektedir. Ayrıca ağ topolojisinin ve yönlendirme mekanizmasının performans ve gecikmeler üzerinde önemli bir etkiye sahip oldukları vurgulanmıştır. Günümüzde fat-tree ağı, veri merkezi ağları için en yaygın kullanılan topolojilerden biridir. Ağ yöneticileri

yönlendirme algoritmalarını tasarlarken yük dengeleme yöntemlerini de benimsemektedir. Bununla birlikte veri merkezlerinde yaygın olarak kullanılan fat-tree ağlarında yük dengelemenin de içinde olduğu bir yönlendirme algoritması, geleneksel yaklaşımlarla tam olarak karşılanamamaktadır. Bunun en önemli sebebi her ağ cihazından ağ trafiği istatistiklerini elde etmenin etkili bir yolu olmamasıdır. İlgili çalışmada bu duruma çözüm olarak OpenFlow protokolü kullanılarak merkezi bir denetleyici ile ağ trafik istatistiklerinin toplanması önerilmektedir. Önerilen metot ile veri merkezi ağları için yüksek performans ve düşük gecikme süreli bir yük dengeleyici tasarımı yapılmış ve yük dengeleyicisine dinamik bir yönlendirme algoritması uygulanmıştır. Algoritmanın görevi yaklaşmakta olan ağ akışlarının trafiğini dağıtmak ve her alternatif yolun eşit miktarda trafik yükünü almasını sağlamaktır. Ayrıca algoritmanın, büyük ölçekli ağlara uygulanabilme ve veri akışını dinamik olarak zamanlayabilme yetenekleri de vardır. OpenFlow denetçisi olarak Beacon'ın [17] kullanıldığı çalışma Mininet emülatör ortamında gerçekleştirilmiştir. Geliştirilen OpenFlow kontrolcü programının iki önemli fonksiyonu vardır. Bunlardan biri, port istatistiklerini gözlemlemek, ikincisi ise yeni akışları zamanlamaktır. Port istatistikleri, maksimum bant genişliğine sahip bağlantıyı bulmaya dayanmaktadır. Ancak her bağlantıdaki mevcut bant genişliğini doğrudan kontrolcü veya anahtardan elde edemeyeceği için kontrol cihazı, her bir portta iletilen baytları anahtarlardan periyodik olarak sorgulayıp, artışlarını hesaplamakta ve bu artışları seçim kriteri olarak kullanmaktadır. Yeni akışları zamanlama şu şekilde çalışmaktadır. Her OpenFlow anahtarı kendi akış tablosunu korumaktadır. Herhangi bir paket geldiğinde, anahtar paketin eşleşme bilgilerini akış tablosundaki girişlerle kontrol etmektedir. Bir eşleşme bulursa paket ilgili bağlantı noktasına gönderilir. Aksi takdirde mesaj kontrolcüye gönderilir. Kontrolcü ilgili paket için uygun bir yol bulmaktadır. İlgili çalışma 128 ana bilgisayar ve 80 anahtar ile Mininet ortamında oluşturulmuştur. Oluşturulan bu algoritma, tekrarlayan bir yol aramak için fat-tree ağlarının hiyerarşik özelliğini kullanarak OpenFlow protokolü ile elde edilen gerçek zamanlı trafik istatistiklerine dayanarak kararlar almaktadır. Çalışmanın sonuçlarına göre; oluşturulan algoritmanın, yüksek hızlı veri iletimini sürdürmede ve çeşitli ağ trafiği türlerinde ağ gecikmesini önlemede oldukça başarılı olduğu belirtilmektedir.

Yukarıda incelenen çalışmalar, OpenFlow protokolü kullanılarak gerçekleştirilen yük dengeleme çalışmalarından seçilmiştir. OpenFlow ile gerçekleştirilen yük dengeleme çalışmaları mevcut tezde de kullanılan yöntemle benzerlik göstermektedir. Dolayısı ile bu tez çalışmasında başlangıçta hedef bilgisi olmayan paketlerin ilgili hedeflere yönlendirilmesi

işlemi için OpenFlow yük dengeleme çalışmaları incelenmiştir. Ancak tez çalışmasının geneli dikkate alınarak yapılan işlemin, algılayıcılardan toplanan verilerin her birinin kendi işlenecekleri makinelere dağıtılması işlemi olduğu göz önünde bulundurularak alan araştırması bir başka alana kaydırılmıştır. Algılayıcılardan toplanan her bir tip için özelleşmiş makinelerin olduğu konsept uç hesaplama olarak tanımlanmaktadır. Bu konu ile ilgili incelenen çalışmalara Yazılım Tanımlı Ağlar başlığında yer verilmiştir.

3. YAZILIM TANIMLI AĞLAR

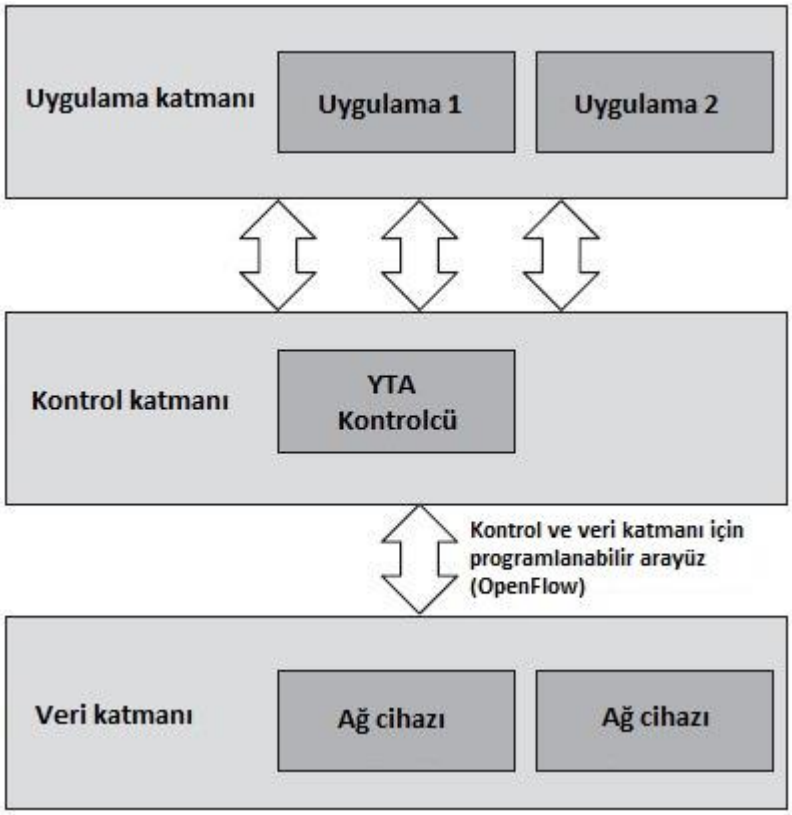
2018 itibarı ile yapılan bir araştırmada verilen istatistiğe göre [18] mevcutta kullandığımız internet ağına, IoT cihazlarının da içinde bulunduğu yaklaşık 23 milyar cihazın bağlı olduğu düşünülürse bu devasa ağın yönetiminin gittikçe zorlaşacağını söylemek mümkündür. Bu kadar büyük bir ağ içerisinde cihazları donanım anahtarları ile birbirlerine bağlamak, mevcuttaki ağ yönetim yüküyle ağdaki hareket kabiliyetini oldukça azaltmaktadır. Yazılım anahtarları kullanan büyük sanal alt yapıları olan şirketler dâhil olmak üzere manuel olarak teker teker yazılım anahtarlarının kurulumu karmaşık ve hataya açık bir işlemdir. Bunların hepsi bir tarafa, ağ trafiğinin nasıl yönetileceğine karar veren kontrol düzlemi ile bu kontrol düzleminin verdiği kararlara göre ağ trafiğini yönlendiren veri düzleminin aynı katmanda bulunması ağ yönetiminde esnekliği azaltarak, ağ altyapısının gelişimine izin vermemektedir [19].

Yazılım tanımlı ağlar, ağ yönetiminde yazılımın gücünü kullanmaya olanak sağlaması açısından mevcut ağ yönetimi çözümleri içerisinde önemli bir yere sahiptir. Yazılım tanımlı ağlar, bir ağın merkezi bir konumdan bir kontrol yazılımı aracılığı ile programlamasını sağlayarak ağ yönetimi işlemlerinin veri iletimi işlemlerinden soyutlanmasını sağlamaktadır. Ağ yönetimini kolaylaştırma, ağ yönetiminde daha hızlı aksiyon alabilme, daha esnek yönlendirme işlemleri ve düşük maliyet gibi daha birçok avantaj sunmaktadır. Bu bölüm içerisinde ağ cihazlarında bulunan kontrol katmanı, veri katmanı ve uygulama katmanı hakkında kısa bir bilgi verilip Yazılım Tanımlı Ağların önerdiği kontrol katmanı ile veri katmanın birbirinden ayrılması ile sağlanan avantajlardan bahsedilmektedir. Yine Yazılım Tanımlı Ağlar ile ortaya çıkan OpenFlow protokolü hakkında bilgi verilerek geleneksel ağlar ile Yazılım Tanımlı Ağların avantaj ve dezavantajları karşılaştırılmakta ve bölüm tamamlanmaktadır.

3.1. Yazılım Tanımlı Ağ Mimarisi

Bir ağda bir paketin yönlendirilme işlemi 3 katmandan oluşan bir yapı tarafından gerçekleştirilmektedir. Bunlar; bu paketin nasıl yönlendirileceğine karar veren kontrol düzlemi, kontrol düzleminin aldığı kararlara göre yönlendirme işlemini yapan veri düzlemi ve tüm yönlendirme işlemleri için politikaları belirleyen uygulama katmanıdır. Ayrıca YTA

mimarisi içerisinde YTA kontrolcüsünün fiziksel cihazlarla iletişimini gerçekleştirdiği SouthBound arayüzü ve kontrolcünün uygulama katmanı ile iletişimini gerçekleştirdiği NorthBound arayüzü bulunmaktadır. Şekil 3.1'de bu katmanlar gösterilmiştir.



Şekil 3.1. Yazılım tanımlı ağ mimarisi

Kontrol Katmanı: Veri katmanı ile uygulama katmanları arasında bulunur. Bu katmanda ağ yönetimi ile ilgili kararlar alınır. Ağ yönetiminin beyni konumundaki bu katman, ağ yönetimi ile ilgili kararları veri katmanındaki cihazlara iletir.

Veri Katmanı: Kablolu veya kablosuz birbirlerine bağlı cihazlarının bulunduğu katmandır [20]. Bu katman ağ trafiğinin iletildiği en alt katmandır. Kontrol katmanının ağ yönetimi ile ilgili verdiği kararlar bu katmanda uygulanır.

Uygulama Katmanı: Bu katmanda; yönlendirme, güvenlik duvarı, yük dengeleme ve izleme gibi Northbound arayüzünün sunduğu ağ yönetimi uygulamalarının bir kümesi bulunur. Bu ağ yönetim uygulamalarının her biri, yönlendirme cihazlarının davranışlarını programlayan Southbound talimatlarına çevrilecek politikaları tanımlar [20].

NorthBound Arayüzü: YTA mimarisinde, Northbound arayüzleri (API'ler) genellikle YTA kontrolcüleri ile ağ üzerinde çalışan hizmet ve uygulamalar arası iletişim kurmak için kullanılan YTA arayüzleridir [21].

SouthBound Arayüzü: Bu arayüzler, kontrol katmanı ve veri katmanı arasında iletişimi gerçekleştiren API'lerden oluşur. OpenFlow, Southbound arayüzü için en bilinen protokollerinden biridir [22].

Bu arayüz ile aşağıdaki özellikler sağlanır:

- Yönlendirme işlemlerinin tümünün program ile kontrolü
- Reklam kabiliyeti
- İstatistiksel raporlama
- Olay bildirimi

OpenFlow protokolü YTA için tek Southbound arayüzü olmayıp; protokol bağımsız yönlendirme (POF) gibi başka arayüzler de vardır [23].

3.2. Yazılım Tanımlı Ağlar Avantaj ve Dezavantajları

Kontrol ve veri düzlemlerini ayırma fikri YTA'nın temel prensibini oluşturmaktadır. Ağın kontrol mantığı ile ağ trafiğinin yönlendirilmesi işlemini yapan katmanların birbirinden ayrılması dikey entegrasyonu artırmaktadır. Kontrol ve veri düzlemlerinin bu şekilde ayrılması ağ anahtarlarının basit birer yönlendirme cihazı olarak kalmaktadır. Ağ yönetimi ise, ağ politikalarını yürüten, ağın yeniden yapılandırılmasını ve gelişimini basitleştiren mantıksal merkezi bir kontrolcüye bırakılmaktadır [19].

Yazılım Tanımlı Ağların kontrol düzlemini ağ cihazlarından ayırıp harici bir varlık haline getirmesi aşağıdaki avantajları ortaya çıkarmıştır [20];

- Kontrol platformu veya ağ programlama dilleri tarafından sağlanan soyutlama işlemleri paylaşılabilir hale geldiği için ağ kontrolcü uygulamalarını programlamak daha kolay hale gelmektedir.

- Kontrol platformu yazılım modülleri yeniden kullanılarak aynı ağ bilgisine sahip uygulamaların daha tutarlı ve daha etkili politika kararları alması sağlanmaktadır.
- Bu kontrolcü uygulamaları ağın herhangi bir yerinden yönlendirme cihazlarını yeniden yapılandırabilir. Bu sayede ağa eklenecek yeni bir fonksiyonun yeri hakkında önceden kesin bir strateji tasarlamaya gerek kalmamaktadır.
- Farklı tip uygulamaların entegrasyonları daha kolay hale gelmektedir. Örneğin, yük dengeleme ve yönlendirme uygulamaları, yönlendirme politikalarına göre öncelikli olan yük dengeleme kararları bir sıraya koyularak birleştirilip bir kontrol politikası oluşturulmaktadır.

3.3. Geleneksel Ağlar ve Yazılım Tanımlı Ağların Karşılaştırılması

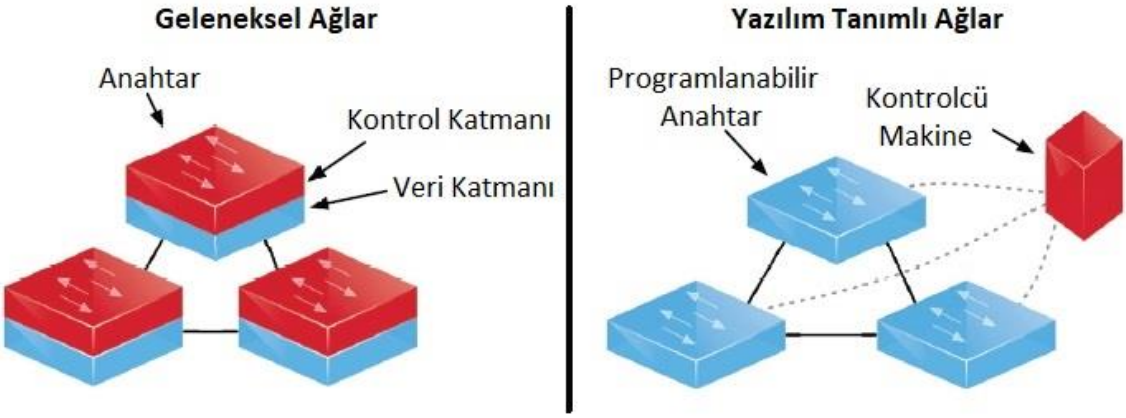
Geleneksel ağlar, kontrol düzlemi için dağıtık bir model kullanılmaktadır. Bu ağlarda ARP, STP, OSPF, EIGRP ve BGP gibi protokoller, her ağ cihazında ayrı olarak çalışmaktadır [24]. Bu ağ cihazları birbirleri ile iletişim halindedir ancak büyük resmi görecek merkezi bir kontrolcü cihaz bulunmamaktadır.

YTA, kontrol düzlemi için merkezi bir kontrolcü modeli kullanılmaktadır. Bu modelde kontrolcü cihaz, kontrol düzleminde bulunurken diğer ağ cihazları veri düzleminde bulunmaktadır. YTA kontrolcüsü kontrol düzleminden alınan kararlarla veri düzleminin yönetiminden sorumludur. Bu düzlemde; OpenFlow, BGP ve OVSDB gibi protokoller kullanılmaktadır [19].

Şekil 3.2'de görüldüğü gibi geleneksel ağlarda kontrolcü ile veri düzleminin ayrılmadığı bir yapıda her bir ağ cihazında bulunan kontrolcünün diğer kontrolcülerle bir bağlantısı bulunmadığı ve birlikte çalışamadıkları için bu tip ağların yapılandırılmasında hataların meydana gelmesi daha olasıdır. Buna karşın YTA'da merkezi bir kontrolcü ile tüm ağ cihazlarının tek bir noktadan yapılandırılması sağlanarak hata payı en aza indirilmektedir [19].

Kontrol ve veri düzleminin aynı fiziksel cihazda bulunduğu durumda, her iki katman tarafından gerçekleştirilecek işlemler için aynı fiziksel cihazın kaynağı kullanılacağından cihaz üzerindeki veri trafiği, CPU ve bellek yükü artacaktır. Şekil 3.2'de görüldüğü gibi kontrol ve veri düzlemlerini ayırıp özel bir sunucu ile kontrol katmanında gözlemleme ve

kontrol işini yaparak ağın etkili bir şekilde yönlendirme kararlarını alabilmesi kolaylaştırılır ve ağın daha az trafik yüküyle düzgün bir şekilde yapılandırılması sağlanır [19].



Şekil 3.2. Geleneksel ağlar ve YTA karşılaştırması [25]

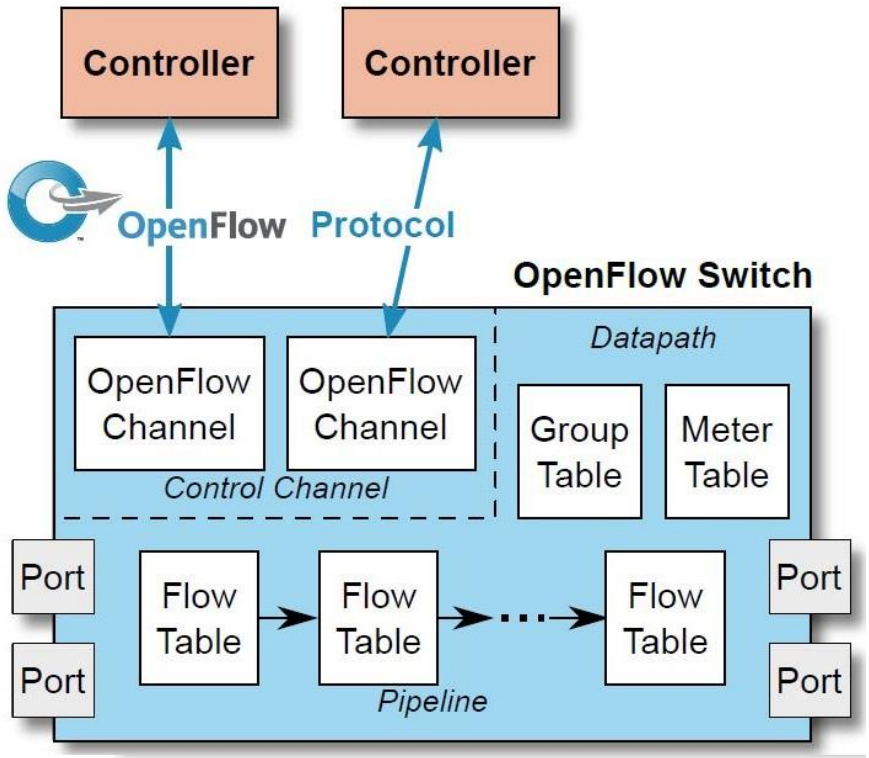
3.4. OpenFlow Protokolü

2000'li yılların ortalarında, internet teknolojilerindeki gelişmelere de paralel olarak araştırmacılar, yeni teknoloji ve protokolleri test edip deneyebilecekleri büyük ağlara ihtiyaç duymaya başlamıştır. NFS (National Science Foundation) tarafından kurulan GENI projesi (National Science Foundation) kapsamında bilişim teknolojileri alanında farklı ihtiyaçlara yönelik tasarımlar yapılıp, çözümler üretilmiştir. OpenFlow Protokolü de bu projelerden biridir [26]. Stanford Üniversitesindeki bir grup araştırmacı, ağ kontrolünü daha kolay hale getirebilmek için yerel alan ağ yönetimi konusunda testlere yoğunlaşmış ve Clean State programını hayata geçirmiştir. Bu program kapsamında ABD'deki tüm üniversiteleri bir birine bağlayan bir ağ oluşturulmuştur [27]. Oluşturulan bu ağ, daha sonra Layer2 ve Layer3 seviye ağ protokollerin yerini alacak protokollerin ortaya çıkmasına yardım etmiştir. OpenFlow protokolünün gelişimi her yeni sürümde yeni özellikler eklenerek devam etmektedir. OpenFlow ekosistemi, sayıları giderek artan OpenFlow özellikli ürün ve çözümleri piyasaya süren firmalarla, OpenFlow üzerinde geliştirilen uygulamaların sayılarındaki artışla ve OpenFlow özellikli ağları kullanan kuruluşların sayısındaki artışla giderek gelişmektedir [28].

OpenFlow anahtar konsorsiyumu 30 Temmuz 2007 tarihinde 0.1.0 sürüm numarası ile ilk referans uygulamasını çıkarmıştır. Daha sonra 31 Aralık 2009 tarihinde, minimum bant

genişliği garantisi için her çıkış portuna çoklu kuyrukların eklendiği OpenFlow 1.0 çıkarılmıştır. Bir sonraki sürüm olan OpenFlow 1.1 versiyonu 28 Şubat 2011'de çoklu tablo işleme özelliği ile piyasaya sürülmüştür. Bundan sonra OpenFlow'u standartlaştırma işlemleri ONF üzerinden devam etmiştir. 2011 yılının aralık ayında OpenFlow 1.2 sürümünü yayınlayan ONF, 2012 yılının şubat ayında IPv6'yı destekler hale gelmiştir. Devam eden zaman içerisinde ONF organizasyonu OpenFlow için yeni sürümler çıkarmaya devam etmiştir. En son 4 Mayıs 2015 tarihinde OpenFlow 1.5.1 sürümü piyasaya sürülmüştür [29].

YTA mimarisinde kontrol katmanı ve dağıtım katmanı arasındaki ilk iletişim protokolü OpenFlow protokolüdür. Bu protokol, ağ cihazlarını programlayabilen standart API'ler sunmaktadır. Gerekli aksiyonlarla birlikte akış tablolarını kullanan Ethernet anahtarlama teknolojisi mantığı ile çalışmaktadır. Kontrol cihazı ile OpenFlow anahtarının iletişim şekli Şekil 3.3'de gösterilmektedir. OpenFlow'un kontrol cihazının iletişimi güvenli kanal üzerinden yapılmaktadır. Ağ anahtarı, bu güvenli kanal üzerinden denetleyici tarafından yapılandırılmakta ve yönetilmektedir. Özel anahtarlarla sertifika alışverişi yaparak kimlik doğrulama işlemi, ilk adım olan iletişimin kurulması adımında gerçekleşmektedir [26].



Şekil 3.3. OpenFlow anahtarı ana yapıları [29]

Şekil 3.3'de de görüldüğü gibi bir OpenFlow mantıksal anahtarı, paket arama ve yönlendirme işlemlerini gerçekleştiren bir veya daha fazla akış tablosu ve bir grup tablosundan oluşmaktadır. Ayrıca harici kontrolcüler için bir veya daha fazla OpenFlow kanalına sahiptir. Anahtar, kontrolcü ile iletişim kurarak, kontrolcü OpenFlow anahtar protokolü üzerinden bu anahtarı yönetmektedir. Kontrolcü bu protokolü kullanarak akış tablolarındaki akış girdilerine ekleme yapabilmekte, bunları güncelleyebilmekte ya da silebilmektedir. Anahtarda bulunan her bir akış tablosu bir dizi akış girdisi içerimektedir. Her bir akış girdisi, eşleşme alanları, sayıcılar ve eşleşen paketlere uygulanacak bir dizi talimatlardan oluşmaktadır [29].

3.4.1. OpenFlow portları

OpenFlow portları, paketleri OpenFlow işleme ile ağın geri kalanı arasında iletimini sağlamak için kullanılan ağ arayüzleridir. OpenFlow anahtarları, OpenFlow bağlantı noktaları üzerinden birbirlerine mantıksal olarak bağlanırlar. Bir paket bir OpenFlow anahtardan diğer bir OpenFlow anahtara gönderilirken sadece ilk anahtardaki OpenFlow çıkış portunu ve ikinci anahtardaki OpenFlow giriş portunu kullanmaktadır [29].

Bir OpenFlow anahtarı, OpenFlow işlemleri için kullanılan bir takım OpenFlow portları sağlamaktadır. Anahtar donanımı tarafından sağlanan ağ arayüzleri ile OpenFlow port seti birbirinin aynısı olmayabilmektedir. Bazı ağ arayüzler OpenFlow için devre dışı bırakılmış olabilmekte ya da OpenFlow için ek bağlantı noktası tanımlanmış olabilmektedir [29].

OpenFlow paketleri giriş portundan alınmakta ve bunları bir çıkış portuna yönlendirecek olan OpenFlow iletim hattı tarafından işlenmektedir. Paket giriş portu, paketin OpenFlow anahtarına alındığı OpenFlow portunu temsil etmekte ve OpenFlow iletim hattı boyunca paketin bir özelliği olarak kullanılmaktadır. Bu giriş port bilgisi paket eşleştirilirken kullanılmaktadır. OpenFlow iletim hattı, paketin ağa nasıl geri döneceğini tanımlayan çıktı eylemini kullanarak paketi bir çıkış portuna gönderebilmektedir [29].

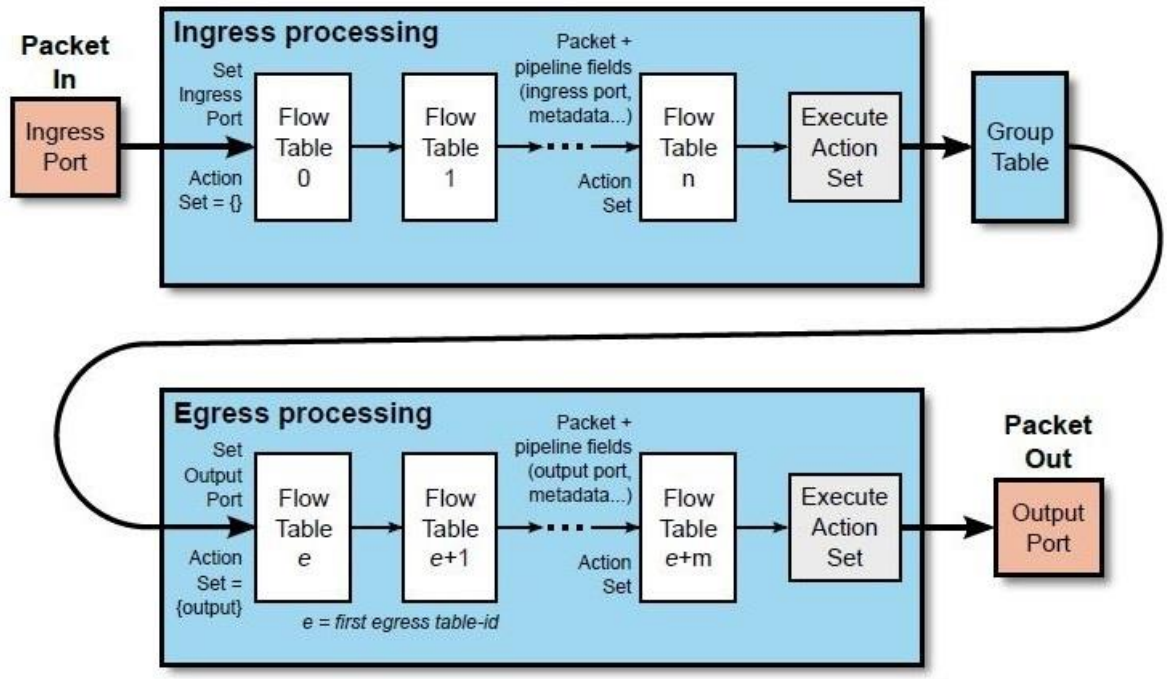
Bir OpenFlow anahtarı, fiziksel portlar, mantıksal portlar ve rezerve portlar olmak üzere üç farklı OpenFlow port tipini desteklemektedir.

3.4.2. OpenFlow tabloları

İletim hattı

Her OpenFlow Mantıksal Anahtarının iletim hattı, bir veya daha çok akış tablosuna sahiptir. Her bir akış tablosu da çoklu akış girdisi içermektedir. OpenFlow iletim hattı işlemi, paketlerin bu akış tabloları ile nasıl etkileşime gireceğini belirlemektedir. Bir OpenFlow anahtarı, bir veya daha fazla sayıda giriş akış tablosuna sahip olması gerekmektedir. Bir akış tablosuna sahip OpenFlow anahtarı olabilmekte ve bu durumda iletim hattı işlemleri oldukça basit hale gelmektedir. İletim hattı işlemleri boyunca bir paketin geçtiği adımlar Şekil 3.4'de gösterilmiştir [29].

Bir OpenFlow anahtarının akış tabloları, paketlerin bu tablolardan geçiş sırasına göre 0'dan başlayarak numaralandırılır. İletim hattı işlemleri, giriş ve çıkış işleme olarak iki aşamada gerçekleştirilir. Giriş işleme, paket OpenFlow anahtarına girdiğinde gerçekleşen ana işlemdir ve bir veya daha fazla akış tablosu içerebilmektedir. Çıkış işleme ise, paketin çıkış portu belirlendikten sonra gerçekleşen işlemdir ve 0 veya daha fazla akış tablosu içerebilmektedir. Bu iki kademenin ayrılması, ilk çıkış tablosuyla belirlenmektedir. İlk çıkış tablosunun numarasından daha düşük numaraya sahip tüm tablolar giriş tablosu olarak kullanılıp eşit veya büyük numaraya sahip olan tablolar kullanılmamaktadır [29].



Şekil 3.4. İletim hattı boyunca paket akışı [29]

İletim hattı işlemi her zaman ilk akış tablosundaki giriş işleme ile başlamaktadır. Paket öncelikle akış tablosu 0'ın akış girdileri ile eşleştirilmektedir. İlk tablodaki eşleşmenin sonucuna bağlı olarak diğer giriş akış tabloları kullanılabilmektedir. Eğer giriş işlemenin sonucu paketi bir çıkış portuna yönlendirmekse, OpenFlow anahtarı çıkış işlemeyi gerçekleştirebilmektedir. Çıkış işleme, isteğe bağlı olup bir anahtar herhangi bir çıkış tablosunu desteklemiyor veya bunları kullanacak şekilde ayarlanmamış olabilmektedir. Eğer geçerli bir çıkış tablosu, ilk çıkış tablosu olarak yapılandırılmadıysa, paket her hangi bir işleme uğramadan çıkış portundan anahtarın dışına gönderilecektir. Eğer geçerli bir çıkış tablosu, ilk çıkış tablosu olarak yapılandırılmışsa, paket bu akış tablosunun akış girdileri ile eşleştirilir ve bu eşlemenin sonucuna bağlı olarak diğer giriş akış tabloları kullanılmaktadır [29].

Akış tabloları ve akış girdileri

Bir akış tablosundaki bir akış girdisinin ana bileşenleri Çizelge 3.1'de verilmiştir [29].

Çizelge 3.1. Akış tablosundaki akış girdisinin ana bileşenleri

Eşleşme Alanları	Öncelik	Sayıcılar	Talimatlar	Zaman Aşımları	Çerezler	Bayraklar

Her bir akış tablosun içerdiği alanlar;

- *Eşleşme Alanları:* Paketleri eşleştirmek için kullanılmaktadır. Bunlar; giriş portları, paket başlıkları ve isteğe bağlı bir önceki tablo tarafından belirtilen meta veri alanlarından oluşmaktadır.
- *Öncelik:* Akış girdilerinin eşleşme önceliğidir.
- *Sayıcılar:* Paketler eşleştiğinde güncellenir.
- *Talimatlar:* Eylem kümesini veya ardışık düzen işlemlerini değiştirmek için kullanılır.
- *Zaman Aşımları:* Anahtar tarafından akışın sona erdirilmesinden önce geçmesi gereken maksimum süre veya boşta bekleme süresidir.
- *Çerezler:* Paket işleme işleminde kullanılmayan, kontrolcü tarafından seçilen mat veri değeridir. Akış istatistikleri, akış değişiklikleri ve akış silme isteklerinden etkilenen akış girdilerini filtrelemek için kontrolcü tarafından kullanılmaktadır.
- *Bayraklar:* Akış girdilerinin yönetilme şeklini değiştirir.

Bir akış tablosu girdisi, eşleşme alanları ve önceliği ile tanımlanmaktadır. Bu öncelik ve eşleşme alanları birlikte belirli bir akış tablosunun bir akış girdisini oluşturmaktadır. Tüm alanları taranmış ve 0 önceliğe sahip bir akış girdisi table-miss akış girdisi olarak ifade edilmiştir [29].

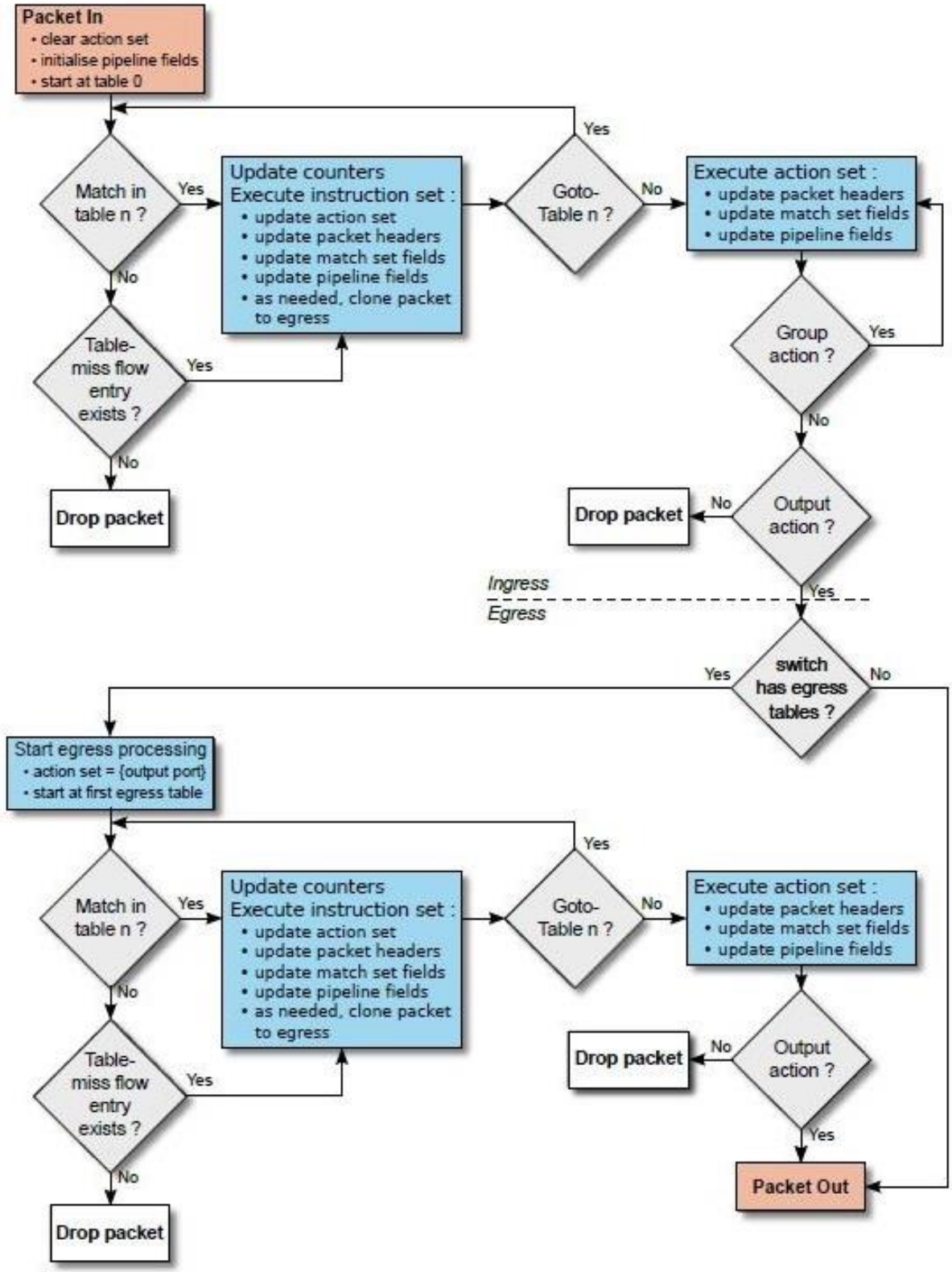
Eşleme

Eşleşme ilk akış tablosunda başlar ve iletim hattındaki diğer akış tablolarına devam edebilmektedir. Akış girdileri paketleri öncelik sırasına göre eşleştirir ve her tablodaki ilk eşleşme girdisi kullanılır. Eşleşen bir girdi bulunursa, bu akış girdisine özel talimatlarla yürütülür. Eğer bir akış tablosunda eşleşme bulunamazsa, nelerin yapılacağı ayrıca tanımlanır. Örneğin eşleşme bulunamadığı durumlarda:

- Paket OpenFlow kanalı üzerinden kontrol cihazlarına iletilebilir.
- Paket atılabilir.

- Paket bir sonraki akış tablosuna eşleşme kontrolü için gönderilir.

Şekil 3.5'de verilen akış diyagramında da görüldüğü gibi bir anahtara giriş portlarından bir paket geldiğinde bu paket için gerçekleştirilecek ilk adım, ilk tablodaki eşleşme durumu kontrolüdür. Eğer bir eşleşme varsa anahtar, sayacı artırır ve belirlenen eylemler listesini uygular. Anahtara gelen her bir paket için özel aksiyonlar gerçekleştirilmektedir [29].



Şekil 3.5. OpenFlow anahtar üzerindeki paket akışının akış şeması [29]

Öncelik paketten başlık alanları ve iletim hattı alanları çıkartılmaktadır. Tablo aramaları için kullanılan paket başlık alanları, paket tipine bağlıdır ve Ethernet kaynak adres veya IPv4

hedef adres gibi çeşitli protokol başlık alanları içermektedir. Paket başlıklarına ek olarak giriş portuna, meta veri alanına ve diğer iletim hattı alanlarına karşı da eşleştirme yapılabilmektedir. Eğer bir akış girdisinin tüm eşleşme alanları ile pakette bu alanlara karşılık gelen başlık alanları ve iletim hattı alanları eşleşiyorsa paket akış girdisi ile eşleştiği anlamına gelmektedir. Eğer akış girdisinde belirtilmeyen bir alan varsa, bu alan paketin başlık alanı veya iletim hattı alanındaki olası tüm değerlerle eşleştirilmektedir. Akış girdileri ile eşleme için kullanılan paket alanları Çizelge 3.2'de verilmiştir [29].

Çizelge 3.2. Eşleşme için kullanılan paket alanları

Alan	Açıklama
IN_PORT	Giriş portu. Fiziksel ya da mantıksal olabilir.
ACTSET_OUTPUT	Eylem kümesi çıkış portu.
ETH_DST	Ethernet hedef adres.
ETH_SRC	Ethernet kaynak adres.
ETH_TYPE	Ethernet tipi, VLAN etiketinden sonra gelir.
IP_PROTO	IPv4 veya IPv6 protokol numarası.
IPV4_SRC	IPv4 kaynak adres.
IPV4_DST	IPv4 hedef adres.
IPV6_SRC	IPv6 kaynak adres.
IPV6_DST	IPv6 hedef adres.
TCP_SRC	TCP kaynak port.
TCP_DST	TCP hedef port.
UDP_SRC	UDP kaynak port.
UDP_DST	UDP hedef port.

Paket, akış tablosundaki akış girdileri ile karşılaştırılmakta ve yalnızca paketle eşleşen en yüksek öncelikli akış girdisi seçilmektedir. Seçilen akış girdisiyle ilişkili sayıcılar güncellenmekte ve seçilen bu akış girdisinde yer alan komut seti çalıştırılmaktadır. Aynı yüksek önceliğe sahip birden fazla eşleşen akış girdisi varsa seçilen akış girdisi tanımsız sayılmaktadır.

Talimatlar

Her bir akış girdisi bir paket girdiyle eşleştiğinde çalıştırılacak bir talimat seti içermektedir. Bu talimatlar, ya bir takım eylemleri içermekte ya da iletim hattı işlemlerini güncellemektedir. Talimatlarda yer alan eylemler, paket yönlendirme, paket güncelleme ve grup tablosu işlemeyi içermektedir. İletim hattı işleme talimatları, daha sonraki işlemler için bilgilerin meta veriler halinde sonraki tablolara gönderilerek tablolar arası iletimi

sağlamaktadır. Eşleşen bir akış girdisiyle ilişkilendirilmiş komut seti bir sonraki tabloyu belirtmiyorsa tablo iletim hattı işlemi durdurularak paket güncellenir ve yeniden yönlendirilir [29].

Eylemler

Eylem, bir paket üzerinde etkili olan bir işlemdir. Bir eylem, paketi bir porta yönlendirebilir, paketi değiştirebilir (hedef IP adresi değiştirme vb.) veya durumunu değiştirebilir (bir sıra ile ilişkilendirme vb.). Eylemlerin çoğu çeşitli parametreler içermektedir. Örneğin, Set-Field eylemi alan tipi ve alan değeri parametreleri ile kullanılmaktadır. Eylemler, bir akış girdisi ile ilişkili bir komut setinin bir parçası olarak veya bir grup girdisi ile ilişkili eylem seti içinde belirtilebilir. Eylemler, paketin eylem kümesi içerisinde bulanabildiği gibi direk pakete de uygulanabilmektedir.

Bir anahtar tüm eylem türlerini desteklemek zorunda değildir. Aşağıda Çizelge 3.3'de gerekli olan eylemler ve isteğe bağlı eylemler, açıklamaları ile verilmiştir. Gerekli eylemler tüm akış tablolarında desteklenmelidir. Kontrolcü ayrıca isteğe bağlı eylemlerin hangilerinin desteklendiği hakkında anahtarda sorgu yapabilmektedir [29].

Çizelge 3.3. Eylemler

Gerekli	Eylem	Açıklama
Evet	Output *port no*	Bu eylem, bir paketi belirlenen OpenFlow portuna yönlendirir. Giriş işleme burada başlar.
Evet	Group *group id*	Paket belirtilen grup kapsamında işlenir.
Evet	Drop	Paket atma işlemi için ayrı bir eylem bulunmayıp eylem kümeleri, çıkış eylemi ve grup eylemi içermeyen paketler atılır.
Hayır	Set-Queue *queue id*	Bu eylem, bir paket için sıra numara ayarlar.
Hayır	Meter *meter id*	Paketi belirlenen ölçere yönlendirir.
Hayır	Push-Tag/Pop-Tag *ethertype*	Mevcut ağlarla entegrasyonu kolaylaştırır.
Hayır	Set-Field *field type value*	Alan tipinde göre başlıkta ilgili alanı bulup verilen değeri bu alana yazar.
Hayır	Copy-Field *src field type dst field type*	Başlıklar ya da iletim hattı alanları arasında veri kopyalama işi yapar.

3.4.3. OpenFlow kanalı

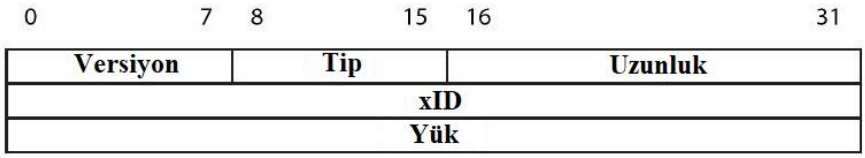
OpenFlow kanalı, tüm ağ anahtarlarını kontrolcüyle birbirlerine bağlayan bir arayüzdür. Bu arayüz, akış tablolarını yapılandırma ve anahtar yönetimi için kullanılmaktadır. Ayrıca paketlerin anahtara gönderilmesi ve anahtardan gelen istekleri kabul edilmesi işlemlerinde kullanılmaktadır. OpenFlow çoğunlukla TLS şifrelemesini kullanırken TCP bağlantısı da kullanabilmektedir. OpenFlow bağlantısı kurulmadan önce gönderici ve alıcıdan oluşan her iki taraf, kullanılacak olan desteklenen en yüksek protokolün bilgisini içeren OFPT_HELLO mesajı göndermektedir. Her iki tarafın da aynı protokol sürümünü desteklemesi durumunda bağlantı sağlanmaktadır. Aksi halde alıcının, OFPT_HELLO_FAILED ve OFPHFC_INCOMPATIBLE alanları işaretli OFPT_ERROR mesajı göndermesi gerekmektedir. Bağlantı kurulduktan sonra el sıkışma işlemi yapılmaktadır. Kontrolcü, anahtar kimliği ve anahtar özelliklerini almak için OFPT_FEATURES_REQUEST mesajı göndermektedir. Bu mesaj türü yalnız OpenFlow başlığından oluşmaktadır. Anahtar da OFPT_FEATURES_REPLY mesajıyla cevap vermektedir [29].

Eğer anahtar, tüm kontrol cihazlarıyla olan bağlantısını kaybederse, hata modlarından birine geçmelidir. Anahtar, hata modlarından hatalı güvenli modda iken kontrolcü için olan tüm paketler atılmaktadır. Eğer hata modlarıdan hatalı bağımsız modda ise anahtar, tüm paketleri OFPP_NORMAL rezerve portundan iletip klasik bir anahtar veya bir yönlendirici gibi davranmaktadır. Bu tip anahtar fonksiyonelliği yalnızca hibrit anahtarlar tarafından desteklenmektedir. OpenFlow protokolünde üç adet mesaj tipi vardır. Bunlar; kontrolcüden anahtara, asenkron ve simetrik mesajlardır. Kontrolcüden anahtara olan mesajlar, kontrol cihazı tarafından başlatılır ve anahtarı yönetmek veya anahtar durumunu incelemek için kullanılır. Asenkron mesajlar, anahtar tarafından başlatılır ve kontrolcüye ağ olaylarıyla birlikte anahtar durumu hakkında bilgiler iletilir. Simetrik mesajlar ise anahtar ya da kontrolcü tarafından başlatılabilir ve istemsiz gönderilir [29].

Anahtar ve kontrolcüler iletişimlerini anahtar tarafından başlatılan 6653 TCP portundan, TLS bağlantısı üzerinden yapmaktadır. Kimlik doğrulama işlemi, özel anahtarla imzalanmış sertifikaların değişimi ile gerçekleştirilmektedir. İletişim için TCP bağlantısı kullanılıyorsa güvenlik hususları dikkate alınmalıdır [29].

3.4.4. OpenFlow başlık alanı

Her OpenFlow mesajı Şekil 3.6'da gösterilen başlık bilgisi ile başlamaktadır.



Şekil 3.6. OpenFlow başlığı [26]

Versiyon bilgisi, kullanılan OpenFlow anahtar protokolünün sürüm bilgisini belirtmektedir. 8 bitle ifade edilen sürüm bilgisinin en önemli biti olan ilk biti sıfırdır ve geri kalan 7 bit protokolün version bilgisini göstermektedir. Uzunluk bilgisi de, mesajın toplam uzunluğunu göstermektedir.

3.4.5. OpenFlow kontrolcüler

Bir OpenFlow kontrolcüsü, OpenFlow protokolünü kullanan bir SDN kontrolcü türüdür. Uygulama trafiğinde en iyi yolu bulabilmek için yönlendirir ve anahtarlar gibi ağ cihazlarını bir birlerine bağlayıp yapılandırır. Bu kontrolcüler, değişen ihtiyaçlara cevap verebilmek için ağ akışlarını değiştirerek, uygulamalar ve cihazlar arası tüm iletişimi etkili bir şekilde yaparak ağ yönetimini basitleştirirler. Ağın kontrol düzlemi operasyonları donanım yerine yazılımla kontrol edildiğinden, ağ yöneticileri ağ trafiğini daha dinamik ve daha ayrıntılı bir şekilde yönetir. Günümüzde en çok kullanılan kontrolcüler Çizelge 3.4'de verilmiştir [26].

Çizelge 3.4. OpenFlow kontrolcüler ve özellikleri

Kontrolcü ismi	NOX	POX	RYU	Floodlight	OpenDaylight
Dil	C++	Python	Python	Java	Java
Performans	Yüksek	Düşük	Düşük	Yüksek	Yüksek
Dağıtık	Hayır	Hayır	Evet	Evet	Evet
Desteklenen OpenFlow versiyonu	1.0	1.0	1.0 - 1.4	1.0 - 1.3	1.3
Bulut desteği	Hayır	Hayır	Evet	Evet	Evet
Öğrenme	Orta	Kolay	Orta	Zor	Zor

4. MATERYAL VE METOT

Bu bölümde öncelikle tüm ağ alt yapısının üzerinde oluşturulduğu Mininet ağ emülatör ortamı, Mininet'te oluşturulan topolojinin görsel çıktısını veren Mininet topoloji görselleştirme aracı ve ağa gönderilen IP paketlerini oluşturan Netcat araçları hakkında bilgi verilmiştir. Daha sonra sırasıyla ağ yönetim işini yapan Ryu kontrolcüsü, çalışmada kullanılan ağ topolojisinin esin kaynağı olan Internet2 2018 Ağ Altyapı Topolojisi ve IPv4 paketinin yapısı irdelenmiştir. Son olarak test işlemlerinde ortaya çıkan ARP seli için kullanılan Yayılan Ağaç Protokolü ve kullanılan En Kısa Yol yönlendirme algoritmasından bahsedilerek bölüm tamamlanmıştır.

4.1. Çalışmada Kullanılan Araçlar

Bu çalışma kapsamında kullanılan araçlar hakkında bilgi verilmiştir.

4.1.1. Mininet ağ emülatörü

Mininet, tek bir makine üzerinde gerçekçi bir sanal ağ oluşturulabilen ve üzerinde gerçek kod çalıştırılabilen bir emülasyon ortamıdır. Tek bir komut çalıştırılarak istenilen ağı saniyeler içinde oluşturabilme olanağı sunan bu ortam, OpenFlow ve YTA sistemlerini geliştirmek ve test etmek için yaygın olarak kullanılmaktadır [30]. İşlem sanallaştırmaya dayalı sanallaştırma kullanan Mininet ortamında her bir işlem bir düğümü, anahtarı, yönlendiriciyi veya kontrolcüyü temsil etmektedir. Oluşturulan tüm ağ elemanları aynı makine üzerinde çalışmasına rağmen birbirlerinden mantıksal olarak tamamen ayrıdır ve her biri gerçek birer fiziksel cihaz gibi çalışmaktadır. Mininet'in sağladığı diğer özellikler aşağıda listelenmiştir;

- Herhangi bir donanım yatırımı yapmadan elde bulunan kaynaklarla çeşitli ağ ortamları oluşturabilmeyi sağlamaktadır.
- Karmaşık ve büyük ağ projelerinin üretime çıkmadan önce test edilip hataların tespit edilebilmesini sağlamaktadır.
- Ağa eklenen her bir düğüme SSH bağlantısı ile bağlanılmaktadır.
- Her bir düğümde Linux işletim sistemine yönelik programlar çalıştırılmaktadır.
- Düğümler arası her bir link için ayrı ayrı bant genişliği ve gecikmeler ayarlanmaktadır.

- Mininet'in üzerinde çalıştığı makinenin donanım kaynakları izin verdiği sürece her türlü ağ topolojisi oluşturulabilmektedir.
- Mininet üzerinde oluşturup test edilen çözümler gerçek cihazlara aktarılmaktadır.

4.1.2. Netcat IP paketi oluşturma aracı

Netcat, IP paketlerini TCP veya UDP kullanarak göndermek veya almak için kullanılan bir yardımcı ağ programıdır. Doğrudan bir komut olarak çalıştırılabildiği gibi diğer programlar ve komut dosyaları tarafından da çalıştırılabilir. Birçok bağlantı türünü desteklediği için ağda hata ayıklama ve araştırma için oldukça kullanışlı bir araçtır [31]. Bu tez çalışması kapsamında Netcat aracı, IP paketlerini oluşturup ağa gönderme ve ağdan gelen paketleri saklamak için kullanılmıştır. IP paketlerinin başlık alanlarını değiştirebilme özelliği sayesinde IP paketleri veri tiplerine göre bu araçla etiketlenmiştir. Çizelge 4.1'de bazı Netcat parametreleri ve açıklamaları verilmiştir.

Çizelge 4.1. Bazı Netcat komut ve parametreleri

Komut	Parametre	Açıklama
Netcat	-4	IPv4 kullan
	-6	IPv6 kullan
	-b	Broadcast yapılsın
	-D	Hata ayıklama soket seçeneğini aktif et
	-I	TCP gelen, tampon alan uzunluğu
	-k	Giriş soketlerini çoklu bağlantı için açık tut
	-l	Gelen bağlantılar için dinleme yap
	-O	TCP gönderilen, tampon alan uzunluğu
	-p	Uzak bağlantılar için yerel port tanımla
	-r	Hedef port seçimi rastgele
	-s	Yerel kaynak adresi
	-T	IP hizmet tipini yapılandır
	-u	UDP modu

Netcat aracı kullanımında özellikle bağlantı tipi –u parametresi ile UDP olarak belirtilmezse bağlantı tipi TCP'dir. Tez çalışması kapsamında da kullanılan bazı örnek komutlar aşağıda verilmiştir.

Örnek 1: 10.0.0.1 yerel adresi üzerinde 3000 numaralı port dinlenir.

```
# netcat -l 10.0.0.1 3000
```

Örnek 2: IP hizmet tipi onaltılık 0xff olarak ayarlanıp 10.0.0.3 uzak adresine 3003 portundan paket gönderilir.

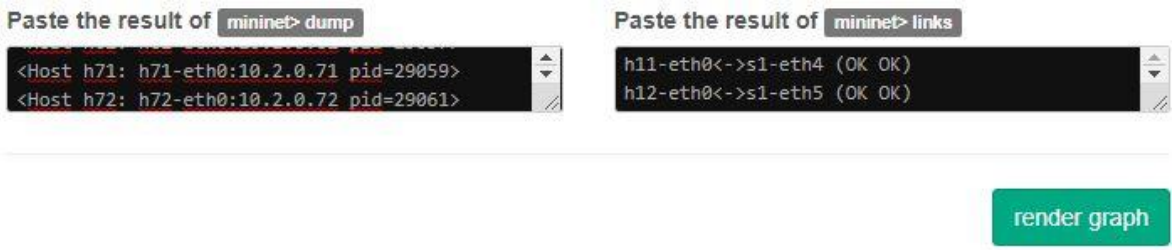
```
# netcat -T 0xff 10.0.0.3 3003
```

4.1.3. YTA Ryu kontrolcüsü

YTA Ryu kontrolcüsü, tamamen Python ile yazılmış, bileşen tabanlı bir yazılım tanımlı ağ çatısıdır. Ryu, geliştiricilerin yeni ağ yönetimi ve kontrol uygulamaları oluşturmasını kolaylaştıran iyi tanımlanmış API'ler içeren yazılım bileşenleri sunar. OpenFlow, Netconf, OF-config vb. gibi network cihazlarının yönetimi için çeşitli protokolleri desteklemektedir. Ryu, OpenFlow versiyonlarından 1.0, 1.2, 1.3, 1.4 ve 1.5 dâhil tüm versiyonları desteklemektedir [32]. Bu tez çalışmasında YTA kontrolcüsü olarak Ryu kontrolcüsü kullanılmıştır. Ryu'nun diğer kontrolcülerle karşılaştırılması Bölüm 3.4.5'de verilmiştir.

4.1.4. Mininet topoloji görselleştirme aracı

Mininet ortamında oluşturulacak ağa anahtarlar, bunlar arasındaki bağlantılar ve bu anahtarların bazılarına bağlanan ana bilgisayarlar eklendikten sonra ağ topolojisi tamamlanmıştır. Ancak Mininet, oluşturulan bu ağa dair görsel bir çıktı vermemektedir. Bu tez çalışması kapsamında kullanılacak topolojinin görsel bir çıktısına da ihtiyaç duyulduğundan harici bir araç için araştırma yapılmış ve Narmox isimli YTA alanında çalışmalar yürüten bir firmanın [33] Mininet'te oluşturulan topolojilerin görsel çıktılarını veren bir araç keşfedilmiştir. Web tabanlı çalışan bu uygulamaya parametre olarak, Mininet ortamında oluşturulan topoloji için "dump" ve "links" komutlarının çıktıları verilmiş ve bu araç topolojinin görsel bir çıktısını üretilmiştir. Şekil 4.1'de Mininet Topoloji Görselleştirme aracına ait bir görüntü verilmiştir. İlgili komutların çıktıları parametre olarak verildikten sonra "Render Graph" butonu ile topolojinin görsel bir çıktısı oluşturulmuştur.



Şekil 4.1. Mininet topoloji görselleştirme aracı

4.2. Kullanılan Yöntemler

Bu çalışma kapsamında kullanılan yöntemler hakkında bilgi verilmiştir.

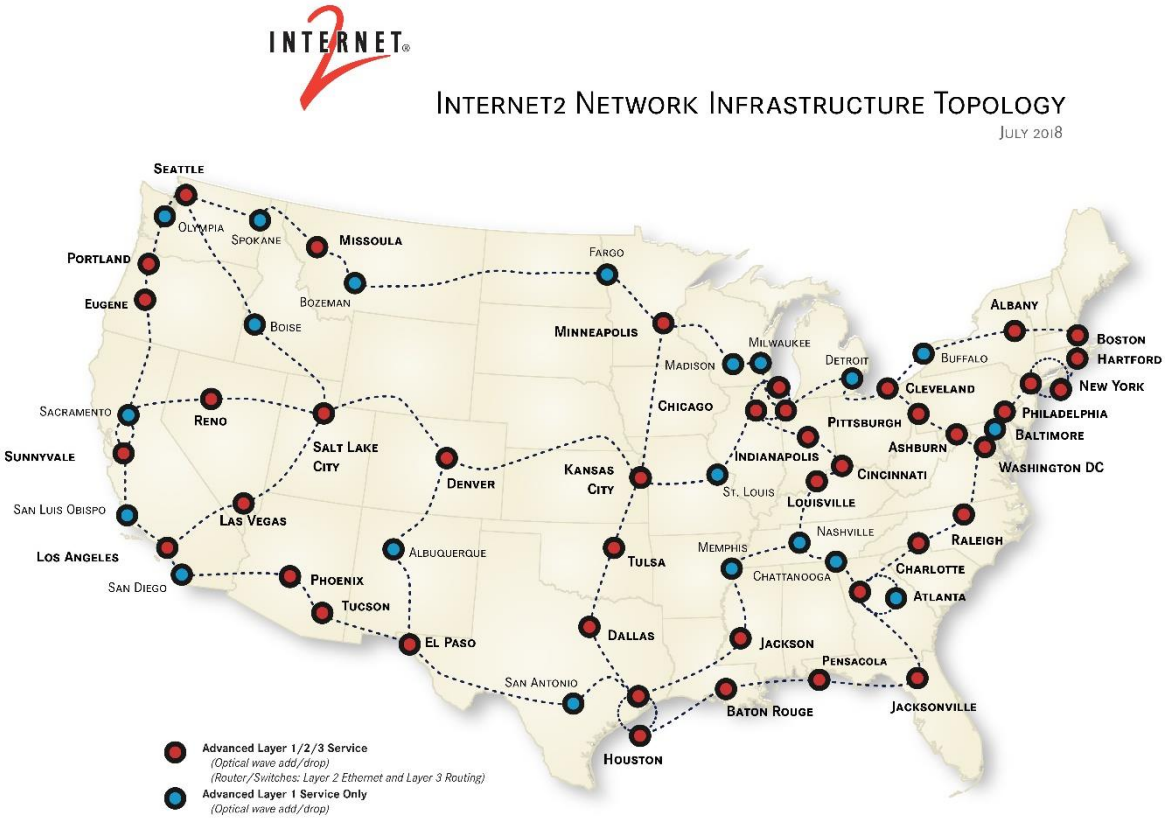
4.2.1. Internet2 ile topoloji oluşturma

Castillo-Velazquez ve ark. [34] çalışmalarında tanımladıklarını göre, Internet2 günümüzde dünyanın en gelişmiş ağlarından biridir. Bu ağ, yeni araştırmalar, yeni teknolojilerin testi ve IPv6, MPLS, QoS, SDN, NVF, NDN vb. gibi yeni nesil internet yaklaşımlarının bütünleştirileceği bir ortam sunmaktadır. Yine bu çalışmada Internet2'nin ABD'deki 10 Gbps ve 100 Gbps arasında hızlara sahip transatlantik bağlantılar sayesinde Kanada'nın CANARIE ağı, Latin Amerika'nın CLARA ağı, Avrupa'nın GEANT ağı ve diğer farklı gelişmiş ağları birbirine bağlayan bir köprü görevi gördüğü belirtilmiştir. Internet2; bulut hizmeti, uzak enstrümantasyon, dinamik veri sanallaştırma, XSEDE (eXtreme Science and Discovery Environment) ve LAGO (Latin American Giant Observatory) gibi dünya çapında birçok proje için altyapı oluşturmaktadır [35]. Internet2, 1996 yılında UCAID (University Corporation for Advanced Internet Development) tarafından NGI (New Generation Internet) projesi kapsamında ortaya çıkmıştır. 1996 yılı başlarında 34 üyeye sahip olan bu oluşum, 2016'da 317'den fazla üyeye sahip hale gelmiştir. Başlangıçta Internet2'nin tescilli bir omurgası olmadığı için üyeleri vBNS (very high-performance speed Backbone Network Service) omurgasına bağlanmak zorunda kalmıştır. Daha sonra 1998 yılında Abilene isminde 622 Mbps iletim kapasitesine sahip yeni omurga piyasaya sürülmüştür [36, 37]. Zamanla bu Abilene omurgasının hızı 1 Gbps'den 2.5 Gbps'ye, 2007 yılında ise 10 Gbps'ye ulaşmıştır. 2008 yılında mantıksal olarak birbirinden ayrı 3 farklı ağa sahip Internet2'nin yeni omurgası ortaya çıkartılmıştır.

Internet2'nin bu 3 katmanından;

- Gelişmiş Seviye 3 IP Yönlendirme katmanı için 9 düğüm,
- Gelişmiş Seviye 2 Optik Anahtarlama katmanı için 9 düğüm,
- Gelişmiş seviye 1 Optik Çekirdek katmanı için 27 düğümden,

Oluşturulmuştur. Bu yeni oluşturulan ağ omurgası 10 Gbps ile hizmet vermeye başlanmıştır. Bu sayede 2014 yılında planlanan, 2016 yılında 100 Gbps'e ulaşma hedefine de zamanında ulaşılmıştır [38]. Eklenen yeni düğümler ile 2018 yılına ait Internet2 ağ topolojisi haritası Şekil 4.2'de gösterilmiştir.



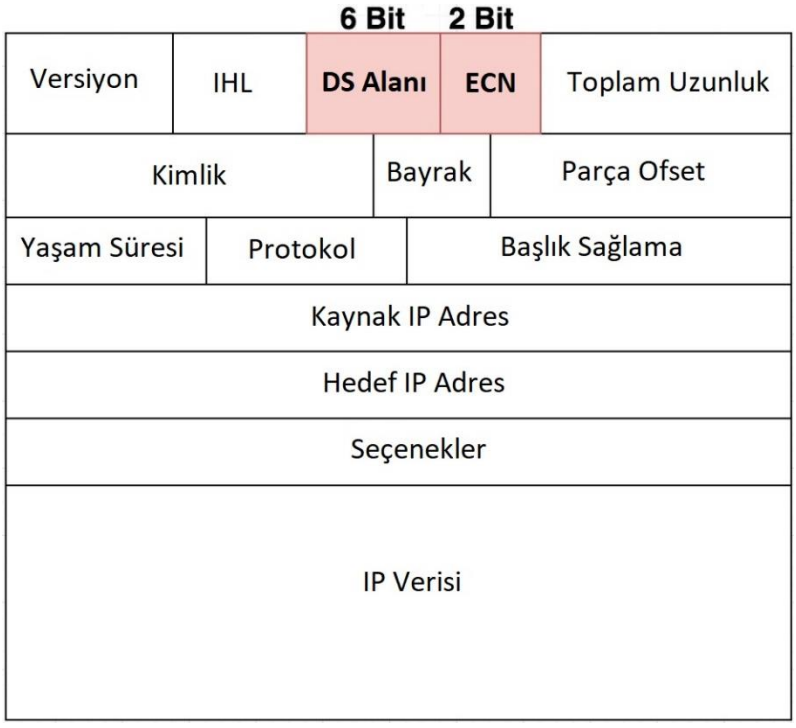
Şekil 4.2. Internet2 ağ altyapı topolojisi [39]

Bu tez çalışması kapsamında 20 adet sadece Gelişmiş Seviye 1 hizmetlerini destekleyen mavi renkle gösterilen düğümler ve 42 adet Gelişmiş Seviye 1, Seviye 2 ve Seviye 3 hizmetlerini destekleyen kırmızı renkle gösterilen düğümlerden oluşan Internet2'nin güncel 2018 yılı haritası üzerinde güncellemeler yapılmıştır. Bu güncellemeler kapsamında ağ topolojisini sadeleştirmek için dallanma oluşturmayan ara düğümlerin çıkarılmasına karar

verilmiştir. Haritada herhangi bir dallanma oluşturmayan Spokane, Missoula, Bozeman, Fargo, Boise, Olympia, Portland, Eugene, Reno, Sacramento, San Luis Obispo, Las Vegas, San Diego, Phoenix, Tucson, Albuquerque, San Antonio, Dallas, Tulsa, Baton Rouge, Jackson, Pensacola, Jacksonville, Chattanooga, Memphis, Louisville, Cincinnati, St. Louis, Indianapolis, Milwaukee, Detroit, Buffalo, Albany, Boston, Hartford, Philadelphia, Baltimore, Ashburn, Pittsburgh, Raleigh ve Charlotte şehirleri haritadan kaldırılarak tez çalışmasında kullanılacak topolojiye alt yapı sağlayacak harita oluşturulmuştur. 5. Bölümde güncellenmiş haritaya ayrıntılı bir şekilde yer verilmiştir.

4.2.2. IPv4 paket yapısı

İnternetin omurga protokolü olan IP protokolü, TCP/IP protokolleri grubunun da ana protokolüdür. Bu protokol OSI modelinin ağ katmanında bulunmaktadır. TCP, UDP veya ICMP kullanılarak gerçekleştirilecek veri iletiminde gerçek veri iletimi, IP protokolü ile yapılmaktadır. Diğer protokollerde de olduğu gibi taşınan veri dışında her bir IP paketine, farklı detaylar içeren kontrol bilgileri eklenmiştir. Bu bilgiler paketin nasıl yorumlanacağı hakkında bilgilerden oluşmaktadır [40]. Bu alanları içeren IP Protokol başlığı Şekil 4.3'de gösterilmektedir.



Şekil 4.3. IPv4 paketi başlık yapısı [40]

IPv4 paketinin başlık alanında bulunan verilerin detayları aşağıdaki gibidir [41]:

- *Versiyon:* Kullanılan internet protokolünün sürüm numarasıdır.
- *IHL:* Tüm IP başlığının boyutudur.
- *DS Alanı:* Hizmet türüdür.
- *ECN:* Rotada görülen tıkanıklık hakkında bilgi taşır.
- *Toplam Uzunluk:* IP paketinin toplam büyüklüğüdür.
- *Kimlik:* Eğer iletim sırasında IP paketi parçalara ayrılırsa, her parça için aynı tanımlama numarası bu alana eklenerek parçanın hangi IP paketine ait olduğu belirtilir.
- *Bayraklar:* IP paketi ağ kaynakları tarafından işlenemeyecek büyüklükte ise bu bayraklara bakılarak IP paketinin bölünüp bölünemeyeceğine karar verilir.
- *Parça Ofset:* Mevcut parçanın, parçalanan IP paketindeki tam yerini belirtir.
- *Yaşam Süresi:* Ağda döngüleri engellemek için her pakete bir TTL değeri eklenir ve bu şekilde ağa gönderilir. Bu değer paketin kaç tane yönlendiriciden geçebileceğini belirtir. Her bir yönlendiricide bu değer bir azaltılır ve bu değer sıfır olduğu zaman paket atılır.
- *Protokol:* Paketin hangi protokole (TCP, UDP, ICMP) ait olduğunu belirtir.
- *Başlık Sağlama:* Paketin hatasız alındığını kontrol etmek için kullanılır.
- *Kaynak IP Adres*: Paketi gönderenin 32 bitlik adresi.
- *Hedef IP Adres:* Paketin gönderildiği hedefin 32 bitlik adresi.
- *Seçenekler:* İsteğe bağlı bir alandır ve IHL alanının 5'ten büyük olduğu durumlarda kullanılır. Bu alan; güvenlik, rota kaydı, zaman damgası vb. durumlarda isteğe bağlı kullanılır.

IP paketi başlık alanında bulunan hizmet tipi (ToS) alanı, 6 bit DS Alanı ve 2 bit ECN alanı olmak üzere 8 bitten oluşmaktadır. 8 bitlik bu ToS alanı QoS için kullanılsa da genellikle servis sağlayıcılar tarafından dikkate alınmamaktadır. Bu sebeple bu çalışmada IP paketlerini etiketlemek için bu 8 bitlik alan kullanılmıştır. Devam eden bölümlerde etiketleme işleminden ayrıca bahsedilmiştir.

4.2.3. Yayılan Ağaç protokolü

Yayılan Ağaç Protokolü (STP), katman 2 anahtarlarda önemli bir protokoldür. Bu protokol kullanılarak bir ağ topolojisi, herhangi bir döngü içermeyen forma dönüştürülmektedir.

Böylelikle ağda yayın seli problemi önlenmektedir [42]. Bu çalışma kapsamında oluşturulan ağ topolojisinde farklı kapalı döngüler bulunduğu için yayın seli problemi ortaya çıkmış ve Yayılan Ağaç Protokolü kullanılarak bu problem ortadan kaldırılmıştır.

4.2.4. Dijkstra En Kısa Yol algoritması

Internet2 ağ topolojisinden esinlenerek oluşturulan ağ topolojisi üzerinde, herhangi bir anahtardan diğer bir anahtara giden yol sayısının birden fazla olduğu durumlar vardır. Ağa gelen bir paketin gönderileceği hedef ana bilgisayara olan farklı yollardan hangisinin kullanılacağını belirleyen farklı yönlendirme algoritmaları bulunmaktadır. Bunlardan birisi de Dijkstra en kısa yol algoritmasıdır. Bu algoritmaya göre ağda bulunan tüm düğümlerin arasındaki yollar için birer maliyet bilgisi vardır ve A noktasından B noktasına gidilirken bu maliyetin toplamda en az olacak şekilde bir en kısa yolun bulunması gerekmektedir. Algoritmanın çalışma adımları incelenen kaynaklarda [43, 44, 45] anlatıldığı şekilde aşağıdaki gibidir;

1. Bir tane başlangıç düğümü seçilir.
2. Boş bir S listesi oluşturulur ve bu liste (mesafe, düğüm) çiftlerinden oluşan düğümlerin mesafe bilgileri tutar. Algoritma çalıştıkça en kısa yol üzerinde bulunan düğümler listeye eklenir.
3. Başlangıç düğümünün mesafesi 0 olarak ayarlanır ve S listesine 1. eleman olarak atanır.
4. Kaynak düğüme komşu olan tüm düğümler için mesafe bilgileri, komşu düğümlere giden yolların maliyet bilgileri ile güncellenir. Güncelleme yapılırken kaynak düğümün mesafesi artı gelinen yolun maliyeti şeklinde hesaplanır. Eğer daha önce mesafe bilgisi güncellenmiş bir düğüm için daha düşük bir mesafe hesaplanırsa, yeni hesaplanan küçük mesafe bilgisi ile düğümün mesafe bilgisi güncellenir.
5. Mesafe bilgileri güncellenen düğümlerden en düşük olan S listesinde bulunmayan düğüm S listesine eklenir.
6. 2. adımı tekrarlayarak hedef düğüm, S kümesi içerisine girinceye kadar işleme devam edilir.

Hedef düğümün mesafe bilgisi güncellenmiş ise güncellenen bu değer kaynaktan hedefe mesafeyi göstermektedir. Ancak işlem sonunda hedef mesafe değeri güncellenmemiş ise kaynaktan hedefe giden bir yol yoktur.

4.2.5. Hizmet Kalitesi (QoS)

İnternetin büyümesiyle son kullanıcılar için web sayfalarında gezinme, mesajlaşma, VoIP, e-posta, sesli ve görüntülü konferans, çevrimiçi oyun ve e-ticaret gibi yeni tip ağ uygulamaları ortaya çıkmıştır. Bu uygulamalar ve hizmetler, internet tarafından iletilmesi gereken kendilerine özgü yeni akışlar ortaya çıkarmıştır. Her bir uygulama ve hizmet için bu akışları başarılı bir şekilde ağ üzerinden iletecek farklı çözümlere ihtiyaç vardır [46]. Örneğin video konferans gibi bazı uygulamaların akışları için belirli bir bant genişliği ayrılırken, VoIP gibi uygulamalar ağ üzerindeki gecikmelere karşı daha hassas olması gerekmektedir. Bu gereklilikleri iyi bir şekilde ele alabilmek için ağda iyi tanımlanmış bir Hizmet Kalitesi (QoS) mekanizması gerekmektedir. Bununla birlikte mevcuttaki Hizmet Kalitesi mekanizmaları yukarıda belirtilen hizmetlerin her birine yeterli katkıyı yapamamaktadır. Ayrıca mevcuttaki QoS çözümleri geleneksel ağların Hizmet Kalitesi sorunlarını çözmede yeterince başarılı olamamaktadır. QoS, tipik olarak bir ağın, seçilen bir ağ trafiği için gerekli hizmetleri sağlama kabiliyeti olarak tanımlanır. QoS'nin temel amacı, aşağıdakileri içeren ancak bunlarla sınırlı olmayan QoS parametrelerine göre öncelik sağlamaktır [47].

- Bant genişliği
- Gecikme
- Kayıp

Hizmet Kalitesi sağlamak için mevcut ağ kaynakları kullanılarak savaşacak farklı uygulama akışlarına ihtiyaç vardır. Bu ağ kaynakları, yüksek öncelikli trafiğin önceliğinin sağlanması için tahsis edilmelidir. Bu süreç genellikle paket iletme ile ilgili doğru kararlar alabilmek için mevcut ağ durumları hakkında bilgi sahibi olmayı gerektirmektedir. Veri katmanı aracılığıyla kontrol katmanını ayırarak ağ yönetimini daha esnek hale getiren YTA ile mevcut ağın genel durumu hakkında bilgi sahibi olmak daha kolay hale gelmektedir. Bu tez çalışması kapsamında algılayıcılardan toplanan farklı tiplerdeki verilerin kendi içlerinde öncelik sıralaması olabileceği ihtimali göz önünde bulundurularak çalışmaya QoS kontrolü de eklenmiştir. IP paketinde bulunan ToS bit alanları, paketin öncelik derecesine göre değiştirilmektedir. YTA kontrolcüsü, ağa gelen paketi ToS bitine göre farklı kuyruklara göndererek, daha önceliği olan verilerin iletimi için yeterli kaynakların verildiğinden emin olmaya çalışmaktadır.

5. YTA RYU KONTROLCÜ İLE İŞARETLİ PAKET YÖNLENDİRME

Yapılan bu çalışmada öncelikle çalışma boyunca yapılacak tüm test ve geliştirme faaliyetlerinin gerçekleştirileceği ağ altyapısını oluşturmak için bir ağ topolojisi belirlenmiştir. Daha sonra belirlenen bu ağ topolojisi Mininet ortamında oluşturulmuştur. Oluşturulan bu ağa yine Mininet ortamında 4 adet ana bilgisayar bağlanmıştır. Teoride bu 4 ana bilgisayara farklı tiplerde veri toplayan algılayıcıların bağlı olduğu varsayılmıştır. Her bir algılayıcı topladığı veriyi bağlı bulunduğu ana bilgisayara göndermektedir. Ana bilgisayar da farklı algılayıcılardan gelen bu verileri ve bu verilerin tiplerini de ekleyerek bunları bir dosyada saklar. Bu işlemden sonra tüm verilerin tutulduğu bu dosyadaki her bir satır için ana bilgisayar, bir Ipv4 paketi oluşturur. Oluşturulan bu IPv4 paketinin ToS bitleri verinin tipine göre ayarlanmakta ve hedef IP adresi olarak ön tanımlı bir IP adres ayarlanarak paket ağa gönderilmektedir. Gönderen ana bilgisayar, gönderilecek IP paketinin kime gönderileceği hakkında bir bilgiye sahip olmadığı için IP paketlerinin hedef IP adres alanları ön tanımlı bir adres olarak ayarlanmaktadır. Ana bilgisayarlar, kendilerine bağlı olan algılayıcılardan gelip diğer ana bilgisayarlara gidecek verileri ağa gönderirken kendi işleyeceği veri tipindeki verileri ağa göndermeyip kendi üstünde bir dosyada saklamaktadır.

Yukarıda başka ana bilgisayara gönderilmek üzere bir paketin, ağa gönderilinceye kadar geçen süreci anlatılmıştır. Ağa gönderilen paketlerin, ilgili hedef ana bilgisayarlara ulaşabilmesi için YTA Ryu kontrolcü ile bir kontrol programı oluşturulmuştur. Bu kontrolcü programı ağa gelen IPv4 paketinin ToS bitlerine bakarak paketin gideceği hedef ana bilgisayarı tespit etmektedir. Tespit edilen bu hedef, ana bilgisayarın IP adresi bu paketin yeni hedef IP adresi olarak ayarlanmakta ve paket bir sonraki anahtara yönlendirilmektedir. Diğer anahtarlarda bir işleme uğramadan yoluna devam eden bu IPv4 paketi hedef ana bilgisayara ulaşarak burada bir dosyaya kaydedilmektedir. Böylelikle ana bilgisayarlardan birine bağlı bir algılayıcıdan çıkan ve bağlı bulunduğu ana bilgisayarda işlenen veri tipine sahip olmayan bir veri, kendi tipinin işleneceği ana bilgisayara ulaşmaktadır. Bir IPv4 paketinin bir algılayıcıdan çıkıp ilgili hedef ana bilgisayara ulaşması için gerçekleştirilen adımlar, alt adımlar şekilde aşağıdaki alt başlıklarda açıklanmıştır.

5.1. Ağ Topolojisi Oluşturma

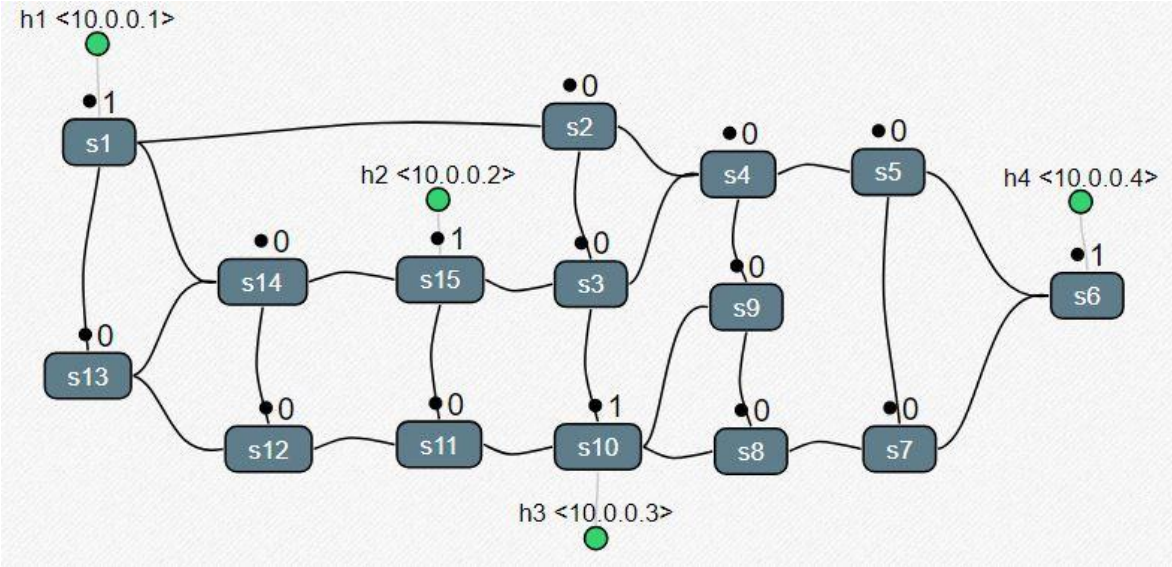
Tüm çalışma boyunca yapılacak test ve uygulama adımlarının gerçekleştirileceği bir ağ altyapısı oluşturmak için öncelikle bir ağ topolojisi belirlenmiştir. Internet2 ağ topolojisinden esinlenilen topolojide öncelikle orijinal ağ haritasındaki dallanma yapmayan iç düğümleri çıkartılmıştır. Internet2 ağ topolojisinin orijinal hali Bölüm 3.4'te verilmiştir. Bu ağ haritasından dallanma yapmayan iç düğümlerin çıkarıldıktan sonraki hali ise Şekil 5.1'de gösterilmektedir.

Şekil 5.1. Internet2 ağ altyapı topolojisi (sadeleştirmek için yeniden düzenlenmiştir)

Dallanma oluşturmayan ara düğümler yönlendirmede farklı yollar oluşturmayacağı için bu düğümleri çıkarak ağ topolojisini sadeleştirme yoluna gidilmiştir.

Bu yeni oluşturulan ağ haritası temel alınarak Mininet ortamında, haritadaki her bir şehir bir anahtara ve şehirlerin arasındaki her bir kesikli çizgi bir bağlantıya karşılık gelecek şekilde bir ağ altyapısı oluşturulmuştur. Bu ağda 15 adet anahtar ve bu 15 anahtarı birbirine bağlayan toplamda 24 adet bağlantı vardır. Ağa eklenen 15 adet anahtarın her biri OpenFlow 1.3 versiyonunu destekleyecek şekilde ayarlanmıştır. Daha sonra oluşturulan bu ağa 4 adet ana bilgisayar bağlanmıştır. h1 olarak isimlendirilen bu ana bilgisayarlardan ilki s1 isimli (Seattle) anahtarına bağlanmıştır. Sırasıyla h2 isimli ana bilgisayar s15 isimli (Denver)

anahtarına, h3 isimli ana bilgisayar s10 isimli (Houston) anahtarına ve son olarak h4 isimli ana bilgisayar s6 isimli (New York) anahtarına bağlanmıştır. Ana bilgisayarların da eklenmesi ile ağın son hali oluşturulmuştur. Şekil 5.2'de ağın son hali gösterilmektedir.



Şekil 5.2. Ana bilgisayarlar eklendikten sonra ağın son hali

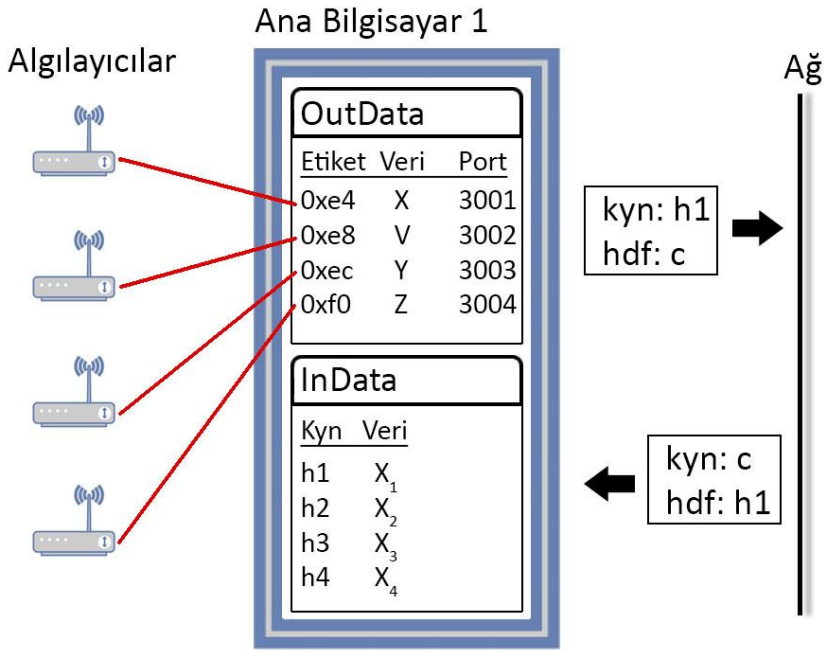
Mininet ortamında oluşturulan ağın Şekil 5.2'de gösterildiği gibi görsel bir hale dönüştürülmesi için "Mininet Topology Visualizer" [33] isminde internet üzerinden çalışabilen bir araç kullanılmıştır. Bu araca girdi olarak aşağıdaki komutların çıktısı verilmekte ve oluşturulan ağ topolojisinin görsel hali çıktı olarak alınmaktadır.

5.2. IPv4 Paketlerinin Oluşturulması ve İletimi

5.2.1. İşaretli paketlerin oluşturulması

Oluşturulan ağa bağlı 4 ana bilgisayarın her biri hem sunucu hem istemci olacak şekilde tasarlanmıştır. Bu bakımdan her bir ana bilgisayar kendisine bağlı algılayıcılardan gelip diğer ilgili ana bilgisayarlara gönderilecek verileri iletirken istemci rolünde, diğer ana bilgisayarlardan işlenmek üzere kendisine gelen verileri alırken sunucu rolünde olacaktır. Bu işlemleri gerçekleştirebilmek için her bir ana bilgisayarda "InData" isimli diğer ana bilgisayarlardan gelen verilerin kayıt edileceği birer dosya oluşturulmuştur. Ayrıca her bir ana bilgisayarda, bu ana bilgisayara bağlı algılayıcılardan toplanan verilerin kayıt edileceği "OutData" isminde birer dosya oluşturulmuştur. Algılayıcılardan toplanan verilerin

tutulduğu “OutData” dosyasında bulunan bir satırdaki ilk kolon verinin tipini ifade eden 8 bitlik onaltılık yapıda bir sayıdan oluşmaktadır. İkinci kolonda ise algılayıcıdan gelen ham veri bulunmaktadır. Örneğin bu veri bir sıcaklık verisi ise en son ölçülen değerin santigrat derece cinsinden değerinin yazdığı varsayılmaktadır. Son kolon olan üçüncü kolonda ise verinin gönderilirken hangi hedef porta gönderileceği bilgisini içermektedir. Bunun yanında diğer ana bilgisayarlardan gelen verilerin kayıt edildiği “InData” dosyasında bulunan bir satırdaki ilk kolon, verinin hangi ana bilgisayardan geldiğini ifade ederken, ikinci kolon ise ana bilgisayarlardan gelen ham veriyi ifade etmektedir. Bu iki dosyanın örnek içeriği Şekil 5.3’de gösterilmektedir.



Şekil 5.3. Ana bilgisayarlarda verilerin tutulan verilerin yapıları

Algılayıcılardan toplanan verilerin saklandığı “OutData” dosyasının içerisindeki ilk kolonda bulunan veri tipinin, 8 bitlik onaltılık bir sayıdan oluştuğu yukarıda belirtilmiştir. Algılayıcılar tarafından toplanan veri tipi sayısını 4 olarak tanımlayıp bu tiplerin her biri, bir ana bilgisayara yönlendirilecek şekilde bir paylaşım yapılmıştır. Her bir ana bilgisayara yönlendirilecek veri tiplerinin onaltılık karşılıkları Çizelge 5.1’de gösterilmiştir.

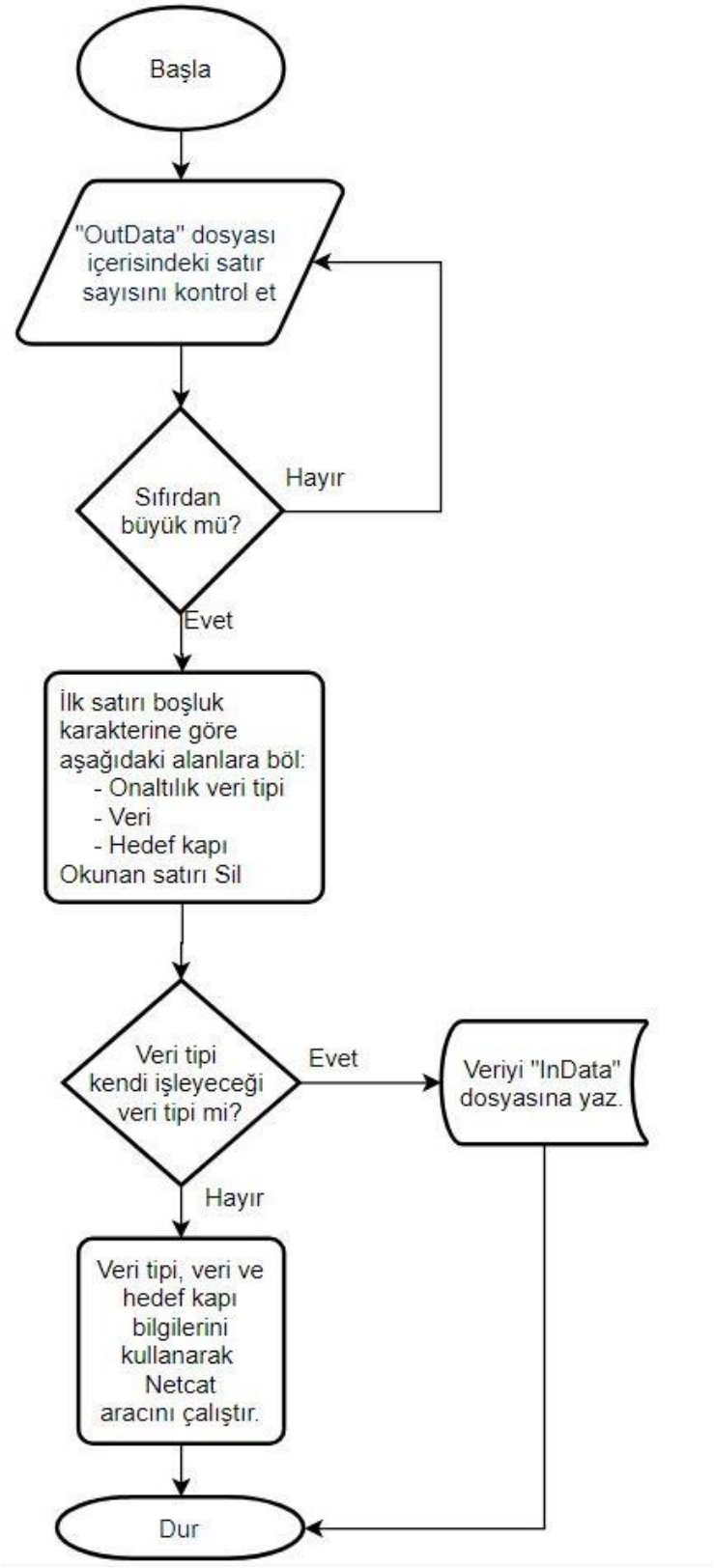
Çizelge 5.1. Ana bilgisayarlar ve bunlara ait veri tipleri

Ana bilgisayar ismi	Onaltılık veri tipi
h1	0xe4
h2	0xe8
h3	0xec
h4	0xf0

Bu veri tipi bilgisi ToS biti şeklinde ayarlandığı için 8 bit şeklinde düşünülmüştür. Bu veri tiplerinin onaltılık karşılıklarının bir özelliği yoktur ve gelişi güzel belirlenmiştir. Bunlar sadece kontrolcü tarafında verinin sınıfını tayin etmek için kullanılmaktadır.

5.2.2. Paketlerin iletimi ve alımı

Bu tez çalışması kapsamında algılayıcılardan toplanan verilerin toplanma aşamasına dair bir çalışma yapılmamıştır. Bu çalışma; algılayıcıların topladıkları verileri, belirli aralıklarla ağa bağlı 4 ana bilgisayara gönderdikleri ve bu ana bilgisayarlar da bu verileri "OutData" dosyasının içerisine veri tipi bilgisini de ekleyerek kaydettikleri varsayımı üzerine kurulmuştur. Bu durum göz önünde bulundurularak her bir ana bilgisayarda algılayıcılar tarafından toplanan verilerin yazıldığı "OutData" dosyasının içeriğini okuyacak bir betik oluşturulmuştur. Python dili ile oluşturulan bu betik Netcat aracını kullanarak "OutData" dosyası içerisinde, okuduğu her bir satır için bir IPv4 paketi oluşturmuştur. Dosyanın içeriğinin okunması, paketin hazırlanması ve gönderilmesi adımlarını anlatan akış diyagramı Şekil 5.4'de gösterilmiştir.



Şekil 5.4. Algılayıcılardan toplanan verilerin iletim için akış şeması

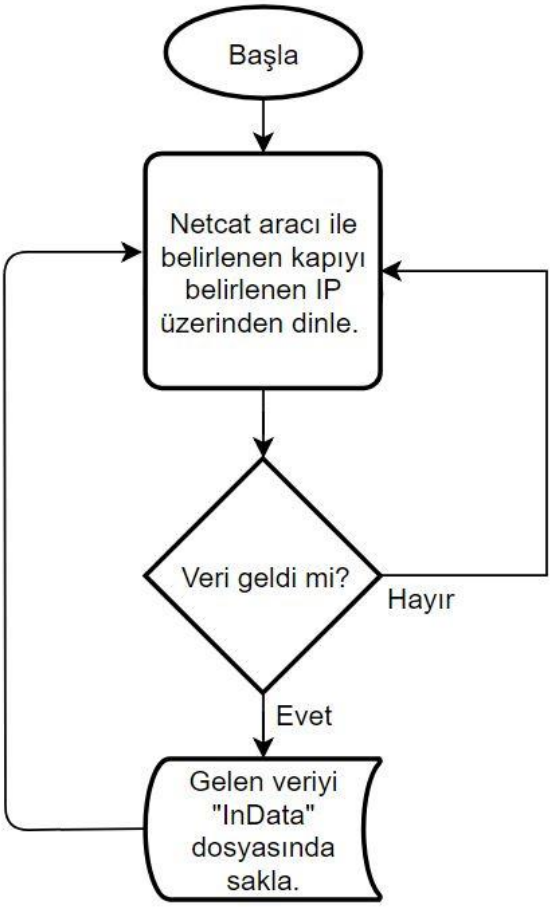
Algılayıcılar tarafından toplandığı varsayılan verilerin, “OutData” dosyası içerisinde belirlenen bir formatta saklandıktan sonra, bu dosya yukarıda akış diyagramı verilen bir

betik için girdi olarak verilmiştir. Bu betik "OutData" dosyasının içeriğini sürekli kontrol etmektedir. Dosya içerisindeki satır sayısı 1'den küçükse, 1 veya 1'den büyük bir değer gelinceye kadar kontrol etmeye devam etmektedir. Eğer satır sayısı 1 veya 1'den büyük bir değer gelirse dosya içerisindeki ilk satırı alır ve boşluk karakterine göre parçalar. Bu parçalama sonucu birinci kolonda verinin tipi, ikinci kolonda verinin kendisi ve üçüncü kolonda da hedef kapı bilgisi şeklinde 3 adet parametre çıkarılmış olmaktadır. Bu adımdan sonra veri, mevcutta üzerinde bulunduğu ana bilgisayarda işlenecek veri tipine ait bir veri mi diye kontrol edilmektedir. Eğer üzerinde bulunulan ana bilgisayarda işlenecek türde bir veri ise parçalamadan sonra ortaya çıkan ikinci kolondaki ham veri yine aynı ana bilgisayar üzerinde bulunan "InData" dosyası içerisine kaydedilmektedir. Ancak veri tipi, mevcut ana bilgisayar üzerinde işlenecek bir veri olmayıp, ağa gönderilmesi gereken bir veri ise, yine parçalama sonucu ortaya çıkan veri tipi, verinin kendisi ve hedef kapı bilgileri kullanılarak Netcat aracı çalıştırılmaktadır. Netcat aracı bu parametreler ile bir IPv4 paketi oluşturmaktadır. Oluşturulan bu IPv4 paketinin veri kısmına parçalama sonucu ortaya çıkan verinin kendisi, ToS biti kısmına parçalama sonucu ortaya çıkan onaltılık veri tipi ve hedef kapı kısmına yine parçalama sonucu elde edilen hedef kapı parametresi eklenerek IPv4 paketi oluşturulmuş olur. Bu parametreler ile çalışan Netcat aracı, gönderilecek veriyi, belirlenen ToS biti ve belirlenen hedef kapı parametreleri ile ağa gönderir.

Netcat aracı ile kendisinin işlemeyeceği veri tiplerini ağa gönderen ana bilgisayar, paketin gideceği hedef ana bilgisayar konusunda bir bilgiye sahip olmadığı için Netcat aracının, ağa göndereceği IPv4 paketinin hedef IP adresi alanına ön tanımlı bir IP adresi yazılır. Bu işlem tüm ana bilgisayarlarda aynı şekilde gerçekleştirilir. Kısaca her bir ana bilgisayar kendi üzerinde bulunan ve algılayıcılar tarafından doldurulan "OutData" dosyası içerisindeki her bir satırı okur ve kendi işleyeceği türde bir veri ise bunu "InData" dosyasına kaydeder. Kendi işleyeceği türde bir veri değilse de bunu yine aynı dosyadan okuduğu veri tipi ve hedef kapı bilgisine göre Netcat aracını kullanarak ön tanımlı bir IP adresi ile ağa gönderir.

Ağa gönderilen IPv4 paketleri, paketler üzerinde bir sonraki alt bölümde anlatılacak kontrolcü işlemleri ile paket ToS bitine göre belirlenen hedef ana bilgisayara yönlendirilecektir. Hedef ana bilgisayarlar, diğer ana bilgisayarlardan aldıkları verileri "InData" isimli dosyada saklamaktadır. Diğer ana bilgisayarlardan gelen verilerin alınması yine Netcat aracı ile gerçekleştirilir. Netcat aracı kullanılarak belirli bir IP adresi üzerinde

belirli bir kapıyı sürekli dinleyecek şekilde farklı bir betik daha oluşturulmuştur. Şekil 5.5'de bu betiğin yaptığı adımların akış diyagramı gösterilmektedir.



Şekil 5.5. IPv4 paketlerinin saklanması için akış şeması

Akış diyagramı verilen bu betik, sürekli yerel IP adresini belirli bir kapı üzerinden dinleyip bir paket geldiği zaman bunu "InData" isimli dosyaya yazar. Daha sonra ilgili kapıyı dinlemeye devam eder ve bu döngü sonsuza kadar devam eder. Sonuç olarak "InData" dosyası içerisinde farklı ana bilgisayarlardan gelen, üzerinde bulunulan ana bilgisayarın işleyeceği türde verilerin bulunduğu bir dosya oluşturulmuş olur. Bu tez kapsamında çalışma sınırlandırılarak bu adımdan sonra gerçekleştirilecek veri işleme adımına dair herhangi bir çalışma yapılmamıştır.

5.3. YTA Kontrolcü İşlemleri

Önceki bölümlerde algılayıcılardan toplanan verilerden, diğer ana bilgisayarlara gönderilecek olan veriler için IPv4 paketi oluşturulup ağa gönderme işlemleri hakkında bilgi verilmiştir. Oluşturulan bu IPv4 paketinin ağa gönderilebilmesi için öncelikle paketin gönderileceği ön tanımlı IP adresine ait MAC adresinin bilinmesi gerekmektedir. Bunun için ilk adım olarak ağa bu ön tanımlı IP adresi için bir ARP sorgusu gönderilmektedir. YTA kontrolcüsünün görevi de burada başlamaktadır. YTA kontrolcüsü, bu ön tanımlı IP adresi için ağa gelen ARP sorgusunu yakalar ve bu ARP sorgusuna cevap olarak yine ön tanımlı bir MAC adresini göndermektedir. Bu işlemle kontrolcü, IPv4 paketini kendi belirlediği hedef MAC adresi ile ağa gönderilmesini sağlamış olmaktadır. Veri tipi etiketi IP paketine eklendiği için kontrolcü hedef ana bilgisayarı belirleyebilmek için kaynaktan gelen IPv4 paketine ihtiyaç duymaktadır. Ön tanımlı MAC adresi ile ağa gelen IP paketi yine kontrolcü tarafından yakalanarak başlık bilgileri parçalanmaktadır. Bölüm 5.2.1'de verilen Çizelge 5.1'de belirtildiği şekilde; kontrolcü, IP paketinin 8 bitlik ToS alanına bakarak paketin hangi tip veri içerdiğine bakmakta ve buna göre de hedef ana bilgisayarı belirlemektedir. Hedef ana bilgisayara ait IP adresi ve MAC adresi bilgileri belirlendikten sonra IPv4 paketinin hedef IP ve MAC adres bilgileri bu belirlenen IP ve MAC adresleri ile değiştirilerek paket ağa gönderilmektedir. Aynı zamanda hedef ana bilgisayardan dönecek cevap paketi için de mevcut anahtarda bir kural daha oluşturulmuştur. Bu kurala göre hedef ana bilgisayarın gönderdiği cevap paketi bu anahtara geldiği zaman, paketin kaynak IP ve MAC adresleri ön tanımlı IP ve MAC adresleri ile değiştirilir ve paket kaynak ana bilgisayara gönderilir. Buradaki amaç kaynak ana bilgisayardan çıkan paketin hedef IP ve MAC adresleri ne ise, bu ana bilgisayara dönecek cevap paketi de aynı IP ve MAC adresi olması zorunluluğudur. Kaynak bilgisayar, paketi ön tanımlı bir IP ve MAC adresine gönderdiği için cevabı da bu IP ve MAC adresinden beklemektedir. Oluşturulan kural ile bu durum sağlanmıştır. Bu çalışma kapsamında oluşturulan ve yukarıda anlatılan adımları içeren bu özgün yöntem için geliştirilen Algoritma -1 aşağıda verilmiştir.

Algoritma 1. İçerik tabanlı yönlendirme algoritması

1: *Başla*
2: *H: ağdaki bir ana makine*
3: *C: ön tanımlı bir IP adresi*

4: *M: ön tanımlı bir MAC adresi*

5: *N: Hedef ana bilgisayar*

6: *Ağdaki H ana bilgisayarından gönderilecek her veri için aşağıdaki adımları uygula*

7: *H, ön tanımlı C IP adresi için ARP sorgusu yap*

8: *Kontrolcü, C için gelen ARP sorgusunu M olarak cevapla*

9: *H, veriyi C ve M adresi ile ağa gönder*

10: *Kontrolcü, paketi yakala, başlık alanlarını ayır*

11: *Kontrolcü, ToS bitine göre yeni hedefi N olarak belirle*

12: *Kontrolcü, paketin hedef bilgilerini N ile değiştirip paketi yönlendir*

13: *N, paketi al*

14: *N, hedefi H olan bir cevap paketi gönder*

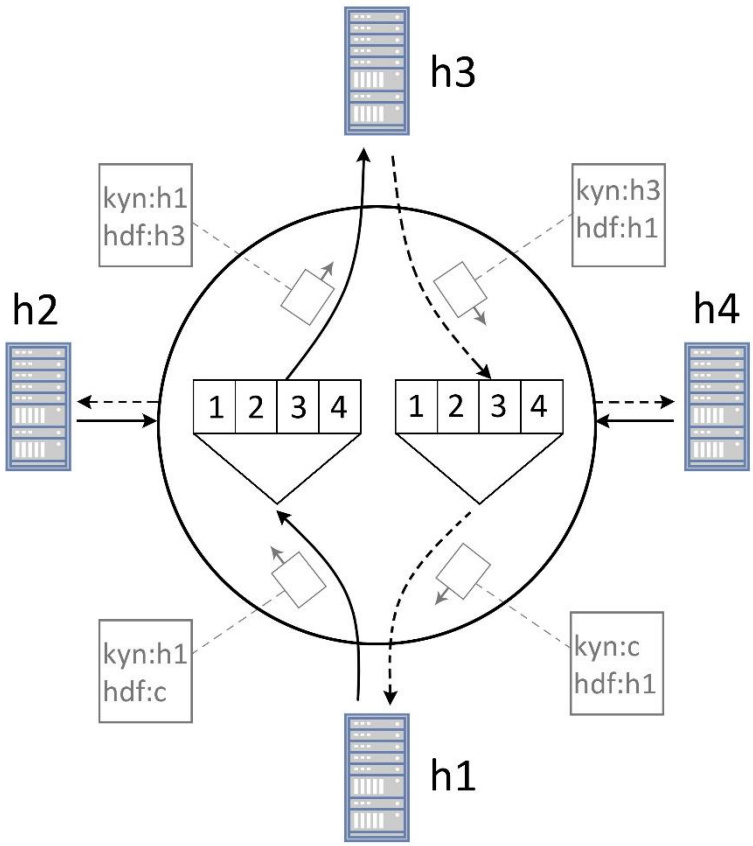
15: *Kontrolcü, N'nin cevabını yakala, paket kaynağını C ve M adresleri ile güncelle*

16: *Kontrolcü, paketi yönlendir*

17: *H, C'den beklenen cevap paketini al*

18: *Bitir*

Yukarıda tanımlanan algoritma ile bir paketin kaynak ana bilgisayardan çıkıp hedef ana bilgisayara varışı ve hedef ana bilgisayardan dönen cevabın kaynak ana bilgisayara gelişi verilmektedir. Şekil 5.6’da bu algoritmanın görselleştirilmiş hali görülmektedir.



Şekil 5.6. IP paketlerinin gönderilmesi ve alınması

Yukarıda verilen yönteme göre h1 ana bilgisayarından çıkan bir IPv4 paketinin yolculuğu aşağıdaki sırada verilmiştir.

1. IP adresi 10.0.0.1 olan h1 ana bilgisayarı, kendisine ait olmayan bir veri için oluşturulan paketi ön tanımlı bir IP adresi olan 10.0.0.50 adresine göndermektedir.
2. h1 ana bilgisayarı 10.0.0.50 IP adresi için bir ARP sorgusu yapmaktadır.
3. h1'den gelen ARP sorgusu kontrolcü tarafından yakalanmaktadır.
4. Kontrolcü, 10.0.0.50 IP adresi için yapılan ARP sorgusuna cevap olarak yine ön tanımlı bir MAC adresi olan AA-BB-CC-AB-BC-CD adresini göndermektedir.
5. ARP cevabını alan h1 ana bilgisayarı, paketi AA-BB-CC-AB-BC-CD hedef MAC adresi ve 10.0.0.50 hedef IP adresi ile ağa göndermektedir.
6. Kontrolcü ön tanımlı hedef IP ve MAC adresine sahip paketi yakalamakta ve başlık alanlarını parçalamaktadır.

7. Kontrolcü parçaladığı başlık alanlarından 8 bitlik ToS alanına bakarak paketin gideceği hedef IP ve MAC adresi tespit etmektedir (Bu örnekte tespit edilen hedefin, 10.0.0.3 IP adresine sahip h3 ana bilgisayarı olduğu varsayılmıştır).
8. Kontrolcü paketin 10.0.0.50 olan hedef IP adres alanını h3'ün IP adresi olan 10.0.0.3 ile AA-BB-CC-AB-BC-CD olan hedef MAC adres alanını da yine h3'ün MAC adresi olan 00-00-00-00-00-03 ile değiştirmekte ve paketi yönlendirmektedir.
9. Paket h3 ana bilgisayarına 10.0.0.1 kaynak IP ve 00-00-00-00-00-01 kaynak MAC adresi ile ulaşmaktadır.
10. h3 ana bilgisayarı, 10.0.0.1 hedef IP ve 00-00-00-00-00-01 hedef MAC adresi ile ağa bir cevap paketi göndermektedir.
11. Kontrolcü yine burada devreye girerek h3'ün gönderdiği cevap paketinin 10.0.0.3 olan kaynak IP adresini 10.0.0.50 ile 00-00-00-00-00-03 olan kaynak MAC adresini AA-BB-CC-AB-BC-CD ile değiştirerek paketi yönlendirmektedir.
12. Paketi 10.0.0.50 IP ve AA-BB-CC-AB-BC-CD MAC adresine gönderen h1 ana bilgisayarı cevap paketini yine aynı IP ve MAC adresinden almış olmaktadır.

Oluşturulan bu yapı Mininet ortamında örnek verileri ile çalıştırılmıştır. Bu kapsamda öncelikle h1 ana bilgisayarında bulunan OutData dosyasının içerisine farklı algılayıcılardan toplandığı varsayılan örnek veriler eklenmiştir. Şekil 5.7'de h1 ana bilgisayarında bulunan OutData dosyasının içerisinde bulunan veriler görülmektedir.

```
ubuntu@sdnhubvm:~/ryu/ryu/matalay/cs/1cs[17:46] (master)$ cat outData
0xe4 "cs1-i" x
0xe8 "cs1-ii" 3002
0xe4 "cs1-i" x
0xf0 "cs1-iiii" 3004
0xe8 "cs1-ii" 3002
0xec "cs1-iii" 3003
0xe4 "cs1-i" x
0xf0 "cs1-iiii" 3004
0xec "cs1-iii" 3003
0xf0 "cs1-iiii" 3004
0xe4 "cs1-i" x
0xe8 "cs1-ii" 3002
0xec "cs1-iii" 3003
0xf0 "cs1-iiii" 3004
0xec "cs1-iii" 3003
0xe8 "cs1-ii" 3002
ubuntu@sdnhubvm:~/ryu/ryu/matalay/cs/1cs[17:46] (master)$
```

Şekil 5.7. h1 ana bilgisayarında bulunan OutData dosyasının içeriği

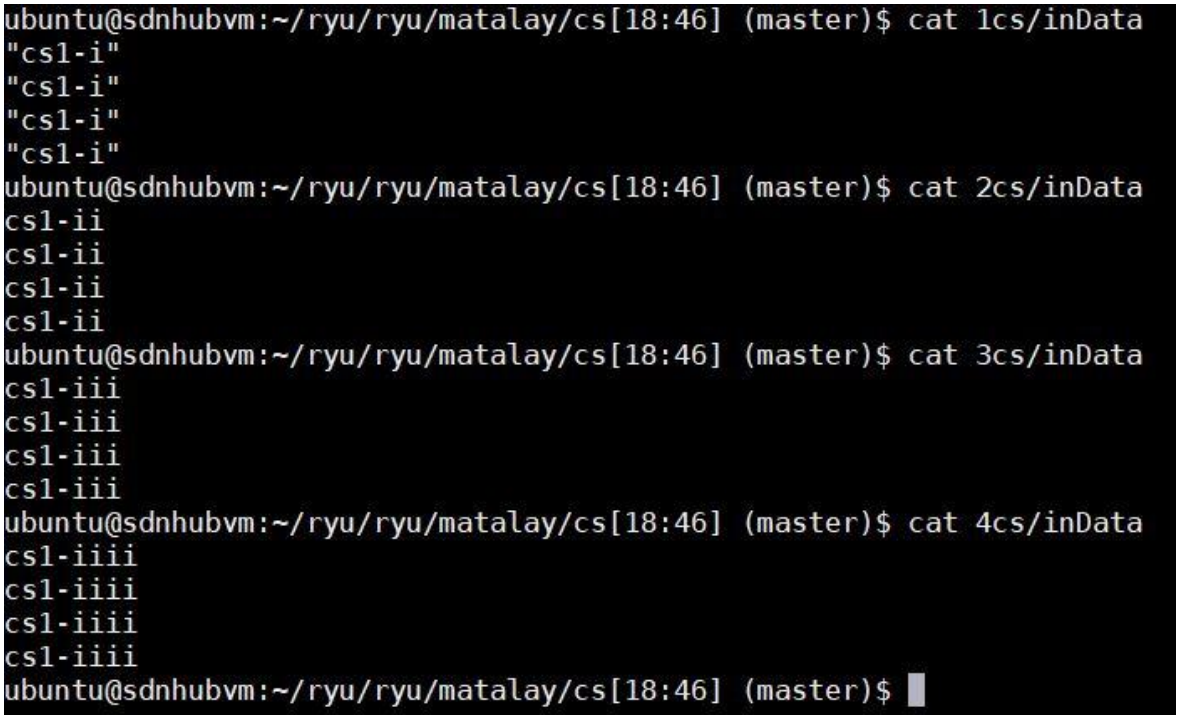
OutData dosyası içerisinde görülen ilk kolon veri tipini gösteren onaltılık bir değerdir. İkinci kolonda bulunan “cs1” ifadesi ile başlayan kolonda gönderilecek verinin kendisi vardır. Bu kısımda gerçek veriler ile çalışılmadığı için veriye, hedef ana bilgisayarda verinin hangi ana bilgisayardan geldiğini gösteren bir etiket eklenmiştir. Bu örnekte veriler h1 ana bilgisayarından diğer ana bilgisayarlara gideceği için etiket “cs1” şeklinde oluşturulmuştur. Bu etiketten “-” karakteri ile ayrılan kısım da verinin tipini belirtmesi için eklenmiştir. Dosya içeriğinde görüldüğü üzere i, ii, iii ve iiii’den oluşan 4 tane veri tipi bulunmaktadır. Bu aşamada gerçek veriler ile çalışılmadığı için bu şekilde bir tanımlamaya gidilmiştir. Aşağıda Çizelge 5.2’de örnek veriler ve açıklamaları verilmiştir.

Çizelge 5.2. Örnek veriler ve açıklamaları

Örnek veri	Veri açıklaması
cs1-ii	h1 ana bilgisayarından toplanan ii tipine sahip bir veri.
cs1-iii	h1 ana bilgisayarından toplanan iii tipine sahip bir veri.
cs1-i	h1 ana bilgisayarından toplanan i tipine sahip bir veri.

Kolay anlaşılabilir olması için bu aşamada nümerik değerler yerine anlamlı ifadeler kullanılmıştır.

Yukarıda algoritmanın bahsedilen algoritmaya göre h1 ana bilgisayar üzerinde bulunan bir betik yardımı ile OutData dosyası içerisindeki satırlar, teker teker okunarak her bir satır için bir IP paketi oluşturularak verilerin ilgili ana bilgisayarlara gitmesi sağlanmaktadır. OutData içerisinde işlenmemiş herhangi bir satır kalmayıncaya kadar bu işleme devam edilmektedir. h1 üzerinde çalışan betik işlemlerini bitirdikten sonra OutData dosyasında bulunan tüm veriler ilgili ana bilgisayarlardaki InData dosyalarına yazılmış olmaktadır. Şekil 5.8’de her bir ana bilgisayarda bulunan InData dosyalarının içerikleri görülmektedir.



Şekil 5.8. Her bir ana bilgisayar üzerinde bulunan InData dosyalarının içerikleri

Başlangıçta her bir ana bilgisayardaki InData dosyalarının içi boşken, OutData'yı okuyan Python betik çalıştırılmıştır. Betik çalışmasını bitirdikten sonra InData dosyalarının içerikleri yukarıdaki gibidir. Burada dikkat edilmesi gereken kısım, başlangıçta h1 ana bilgisayarında karışık bir şekilde tutulan verilerin her biri ilgili olduğu ana bilgisayara gönderilmiş olmasıdır. Dolayısıyla her bir ana bilgisayarda sadece bir tipe ait veriler bulunmaktadır. Yapılan örnekte de "i" tipine sahip veriler h1 ana bilgisayarında, "ii" tipine sahip veriler h2 ana bilgisayarında, "iii" tipine sahip veriler h3 ana bilgisayarında ve "iiii" tipine sahip veriler de h4 ana bilgisayarında InData dosyası içerisinde toplanmıştır.

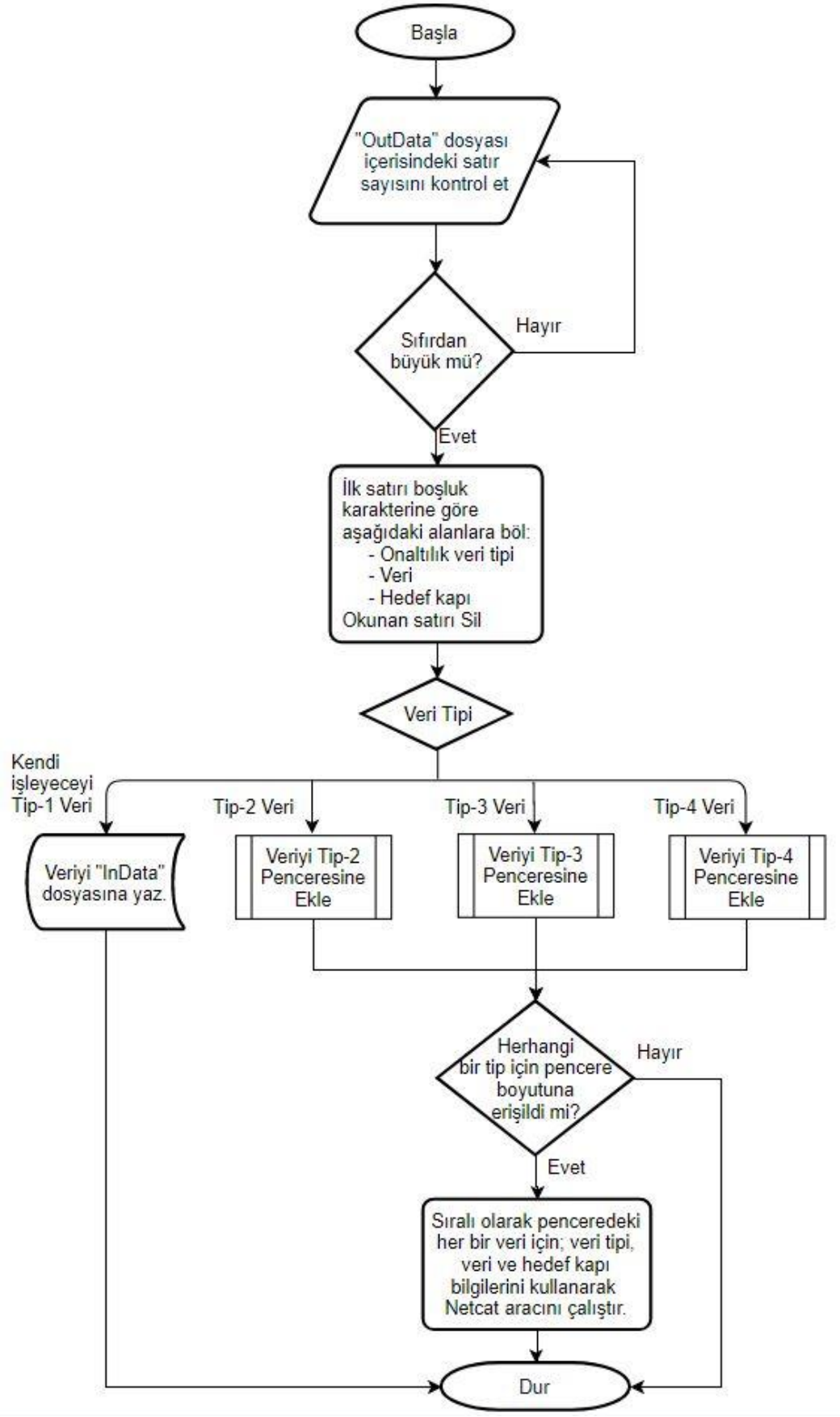
Özetlenecek olursa h1 ana bilgisayarına bağlı farklı tiplerde veri toplayan algılayıcılardan toplanan veriler önce OutData isimli dosyaya yazılmaktadır. Daha sonra bir Python betik yardımı ile bu dosyadaki her bir satır parçalanarak IP paketleri oluşturulmuştur. IP paketleri yine bu dosyada tutulan veri tiplerinin onaltılık karşılıkları ile etiketlenerek ön tanımlı bir IP adres ile ağa gönderilmektedir. Burada ön tanımlı bir IP adres kullanılmasının sebebi gönderici tarafın paketin gideceği IP adresini bilmemesinden kaynaklanmaktadır. YTA kontrolcü, ön tanımlı IP adres ile ağa gelen paketin etiketine bakarak, IP paket için tekrar bir hedef ataması yapmakta ve paketi ağa tekrar göndererek paketin ilgili ana bilgisayara ulaşmasını sağlamaktadır. Böylelikle çalışmanın temel amacı olan, hedef bilgisi belli

olmayan verinin, etiketine bakılarak YTA kontrolcüsü ile ilgili ana bilgisayara yönlendirilmesi sağlanmış olmaktadır.

5.4. Önerilen Pencere Tabanlı Veri Gönderim Metodu

Önceki başlıkta gerçekleştirilen uygulamaların hepsi TCP protokolü ile gerçekleştirilmiştir. Ancak IoT cihazlarının gerçek zamanlı veriler toplayabileceği de göz önünde bulundurularak yukarıda TCP ile gerçekleştirilen adımlar, UDP protokolü ile de gerçekleştirilmiştir. Bu kapsamda IPv4 paket gönderen ve alan Python betikler güncellenmiştir. Güncellenen betiklerle, gönderici tarafında algılayıcılardan toplanan verilerden Netcat aracı ile IPv4 paketleri oluşturulmakta ve toplanan verinin tipine göre oluşturulan IPv4 paketinin ToS bit alanı etiketlenmektedir. Daha sonra bu paket UDP protokolü ile ağa gönderilmektedir. Paketi alan tarafta ise paketler yine Netcat aracıyla UDP protokolü üzerinden alınmakta ve mevcut ana bilgisayar üzerinde saklanmaktadır. Netcat aracının, UDP protokolüyle veri gönderi veya alımının TCP'den tek farkı, fazladan bir parametre kullanılmasıdır. UDP ve TCP protokolleri kullanılarak gerçekleştirilen test ve sonuçlar karşılaştırmalı olarak bu bölüm içerisinde verilmiştir.

Bu bölümde, algılayıcılardan toplanıp diğer ana bilgisayarlara gönderilecek olan verilerin iletim etkinliğini artıracak bir yöntem önerilmiştir. Bu yönteme göre toplanan bir verinin ağa gönderilmesi için aynı tipteki veri sayısının belirli bir sayıya ulaşması beklenmektedir. Herhangi bir tipteki veri sayısı, belirlenen sayıya ulaştığında bu veri grubundaki her bir veri için bir IPv4 paketi oluşturulur ve ilgili tip etiketi ToS bit alanına yazılır. Oluşturulan bu paketler UDP veya TCP protokolü üzerinden sıralı bir şekilde ağa gönderilmektedir. Önerilen yönteme ait akış diyagramı Şekil 5.9’da verilmiştir.



Şekil 5.9. Önerilen pencere yöntemi akış şeması

Bu yönteme göre, OutData dosyasında herhangi bir satır varsa, bu satır boşluk karakterine alanlar ayrılır. İlk kolondaki veri tipi eğer kendi işleyeceği veri tipi ise bu veri direk InData dosyasına yazılır. Diğer veri tipleri için bu tipe sahip veri sayısı belirli bir sayıya ulaşıncaya kadar beklenir. Herhangi bir veri tipine ait veri sayısı belirlenen pencere boyutuna erişir ise penceredeki tüm veriler ilgili etiket ile etiketlenerek sıralı bir şekilde ağa gönderilir. Önerilen bu yöntemin amacı, verinin iletimi için kurulan TCP bağlantı sayısını azaltarak veri iletim süresini kısaltmaktır. Diğer yandan bu yöntemle UDP ile gönderilen paketlerdeki kayıp miktarlarını da azaltma hedeflenmiştir.

5.5. Hizmet Kalitesi İşlemleri

QoS (Hizmet Kalitesi), verilerin türlerine göre verilere birer öncelik atayarak iletilmesini sağlayan bir teknolojidir ve ağda belirli bir iletişim için sabit bir bant genişliği sağlamayı hedefler. Bu tez çalışması kapsamında farklı algılayıcılardan toplanan 4 farklı türdeki verinin iletimi gerçekleştirilmiştir. Bu sebeple ağdaki akış, 5 (Sınıflardan biri tanımsız paketler içindir) farklı QoS sınıfına bölünerek gerçekleştirilmiştir. IP paketinin başlık alanında bulunan 6 bitlik ToS alanına değiştirilerek paketlerin sınıflandırılması sağlanmaktadır. QoS işlemlerinin mevcut algoritmadaki yeri ve sırası aşağıda listelenmiştir. Algoritma - 1'de tanımlanan 11'inci adımdan sonra bir hizmet kalitesi işlemi adımı eklenmiştir. Algoritmanın yeni hali aşağıda verilmiştir.

Algoritma 2. Hizmet kalitesi uygulanmış içerik tabanlı yönlendirme algoritması

1: *Başla*
2: *H: ağdaki bir ana makine*
3: *C: ön tanımlı bir IP adresi*
4: *M: ön tanımlı bir MAC adresi*
5: *N: Hedef ana bilgisayar*
6: *Ağdaki H ana bilgisayarından gönderilecek her veri için aşağıdaki adımları uygula*
7: *H, ön tanımlı C IP adresi için ARP sorgusu yap*
8: *Kontrolcü, C için gelen ARP sorgusunu M olarak cevapla*
9: *H, veriyi C ve M adresi ile ağa gönder*
10: *Kontrolcü, paketi yakala, başlık alanlarını ayır*
11: *Kontrolcü, ToS bitine göre yeni hedefi N olarak belirle*

12: *Kontrolcü, paketin ToS bitinin hizmet kalitesi işlemleri için güncelle*
13: *Kontrolcü, paketin hedef bilgilerini N ile değiştirip paketi yönlendir*
14: *N, paketi al*
15: *N, hedefi H olan bir cevap paketi gönder*
16: *Kontrolcü, N'nin cevabını yakala, paket kaynağını C ve M adresleri ile güncelle*
17: *Kontrolcü, paketi yönlendir*
18: *H, C'den beklenen cevap paketini al*
19: *Bitir*

Tanımlanan bu algoritmaya göre ağa gönderilen bir pakete uygulanacak hizmet kalitesi işlemlerini anlatan bir sıralı adımlar listesi aşağıda verilmiştir.

1. Algılayıcıdan toplanan ve mevcuttaki ana bilgisayarın işlemeyeceği türdeki bir veri için ön tanımlı bir IP adresi ile bir IP paketi oluşturulur.
2. IP paketi ağa gönderilmeden önce ön tanımlı IP adresi için bir ARP sorgusu yapılır. Kontrolcü bu ARP sorgusunu yakalayarak paketin ağa gönderilmesini sağlar.
3. Ağa gelen paket daha önce bir kural tanımlı olmadığı için önce YTA Ryu kontrolcüsüne gider.
4. Kontrolcü gelen paketin başlık kısmını parçalar ve ToS bit alanında saklı etiket bilgisini okuyarak bu etiketli verilerin gönderileceği hedef IP adresi ile paketin hedef IP adresini değiştirir.
5. Bu adım QoS adımıdır. Burada kontrolcüye gelen IP paketinin ToS bit alanı, paketin gideceği hedef IP adresi tespit edildikten sonra QoS'a uygun olarak değiştirilir. Çizelge 5.3'de hangi veri tipi için hangi ToS değerinin yazılacağı verilmiştir.
6. Yukarıda yapılan adımlar aynı tip paketler için tekrar gerçekleştirilmesin diye bir kural oluşturulur ve akış tablosuna yazılır.
7. Son adımda ise paket ilgili porttan ağa gönderilir.

Çizelge 5.3. Veri tiplerine karşılık gelen ToS değerleri ve öncelik atamaları

Veri Tipi	ToS Bit Alanı	
	QoS Öncesi	QoS Sonrası
i	0xe4	0x28
ii	0xe8	0x2c
iii	0xec	0x30
iiii	0xf0	0x34
Tanımsız	0x0	0x0

Algoritmanın QoS uygulanmamış halinde etiketli paket ilk anahtara gelip de hedef ana bilgisayar tespit edildikten sonra ToS bit alanı sıfırlanmaktadır. Bu işlemin amacı, paket her anahtarda aynı işleme maruz kalmasını sağlamaktır. İlk anahtarda hedef tespit edilip paketin hedef IP adresi değiştirildikten sonra paket diğer anahtarlarda bir işleme maruz kalmamaktadır. Algoritmanın QoS uygulanmış halinde ise paket ilk anahtara geldiğinde yine etiket alanına bakılarak hedef tespiti yapılır ve paketin hedef IP adres alanı güncellenir. Daha sonra ToS alanı bu veri tipinin önceliğine göre ayarlanır ve paket ağa bu şekilde gönderilir.

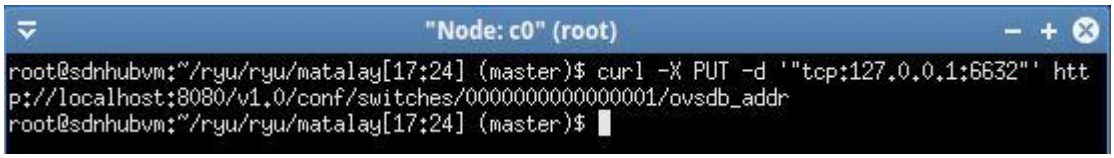
QoS için 5 sınıf oluşturulmuş ve her bir sınıf için bir ToS değeri tanımlanıp paketlere bu QoS sınıf bilgileri yazılmıştır. Ayrıca bir şekilde ToS bilgisi ayarlanmamış paketleri de bir kuyruğa sokmak için veri tipi harici ek bir sınıf daha oluşturulmuştur. Bu işlemden sonra bir sonraki adım olan QoS alt yapısı oluşturma işlemleri yapılmaktadır.

Bu tez çalışması kapsamında internet2 ağ topolojisi örneklenerek toplamda 15 adet anahtar, 4 adet ana bilgisayar ve 1 adet YTA kontrolcüden oluşan bir ağ topolojisi Mininet ortamında oluşturulmuştur. Mininet üzerinde ağ topolojisi oluşturulduktan sonra tez kapsamında hazırlanan YTA Ryu kontrolcüsü de çalıştırılarak QoS için gerekli ayarlamalar yapılmıştır. İlgili adımlar aşağıda anlatılmaktadır.

Öncelikle s1 anahtarının OVSDB'ye erişimi sağlanmak için 6632 numaralı portunu dinleyecek biçimde ayarlanmış, Şekil 5.10’da anahtar port ayarlamaya yer verilmiştir.

Şekil 5.10. Anahtar port ayarlama

Daha sonra yine OVSDB'ye erişebilmek için "ovsdb_addr"'nin ayarlanması gerekmektedir. Bunun için YTA kontrolcüsü üzerinde Şekil 5.11'de verilen ekran alıntısında komut çalıştırılmıştır.



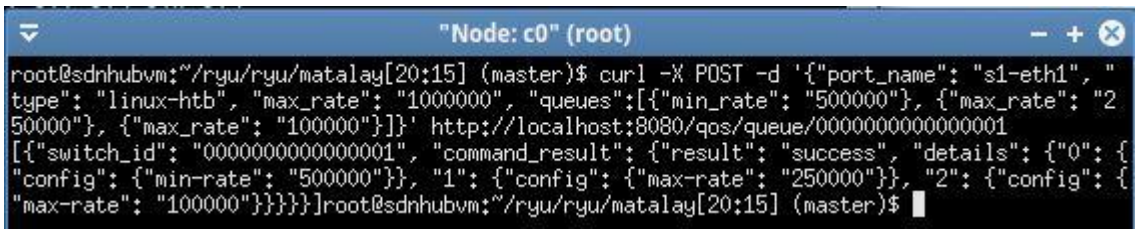
Şekil 5.11. Kontrolcü OVSDB erişimi

Gerçek olmayan verilerle yapılacak test için s1 anahtarına eth4 portundan bağlı olan h1 ana bilgisayarında toplanan algılayıcı verilerinin ilgili ana bilgisayarlara gönderilmesi için 3 farklı kuyruk tanımlanmıştır. Bu kuyrukların her biri Çizelge 5.4'te görüldüğü gibi farklı parametrelerle düzenlenmiştir.

Çizelge 5.4. Tanımlanan kuyruklar ve parametreleri

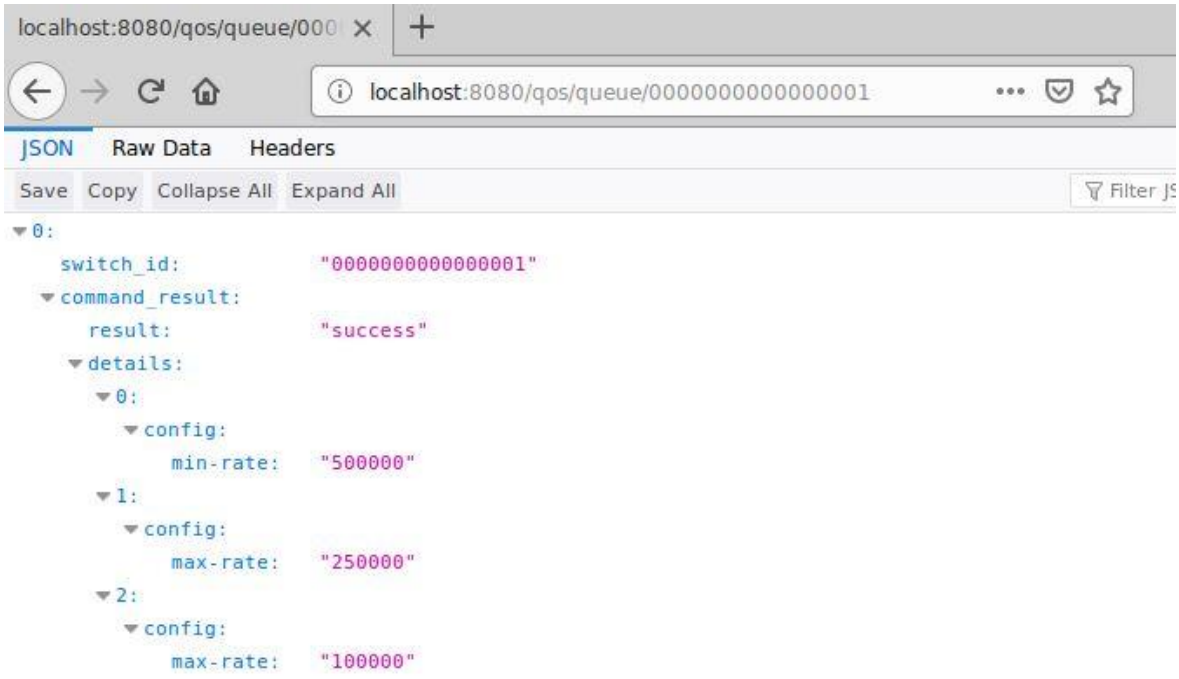
Kuyruk ID	Maksimum Değer	Minimum Değer
0	(1Mbps)	500Kbps
1	250Kbps	-
2	100Kbps	-

Yukarıda maksimum ve minimum değerleri verilen kuyruklar, Mininet üzerinde çalışan kontrolcüye Şekil 5.12'deki gibi eklenmektedir.



Şekil 5.12. Kuyruk ekleme işlemi

Yukarıdaki komutla oluşturulan kuyruk, bir internet tarayıcısı tarafından görülebilmektedir. Şekil 5.13'de s1 anahtarı üzerindeki s1-eth1 portu için tanımlanmış kuyruklar görülmektedir.



Şekil 5.13. Tanımlanmış kuyrukların görüntüsü

Yukarıda kuyrukların belirlenen parametrelerle oluştuğu görülmektedir. Bir sonraki adımda anahtar üzerinde paketlerin DSCP değerlerine bakılarak akışların kuyruklara eklenmesi tanımları yapılmaktadır. Bunun için ToS, DSCP, kuyruk ve QoS bilgilerinin bulunduğu Çizelge 5.5 aşağıda verilmiştir.

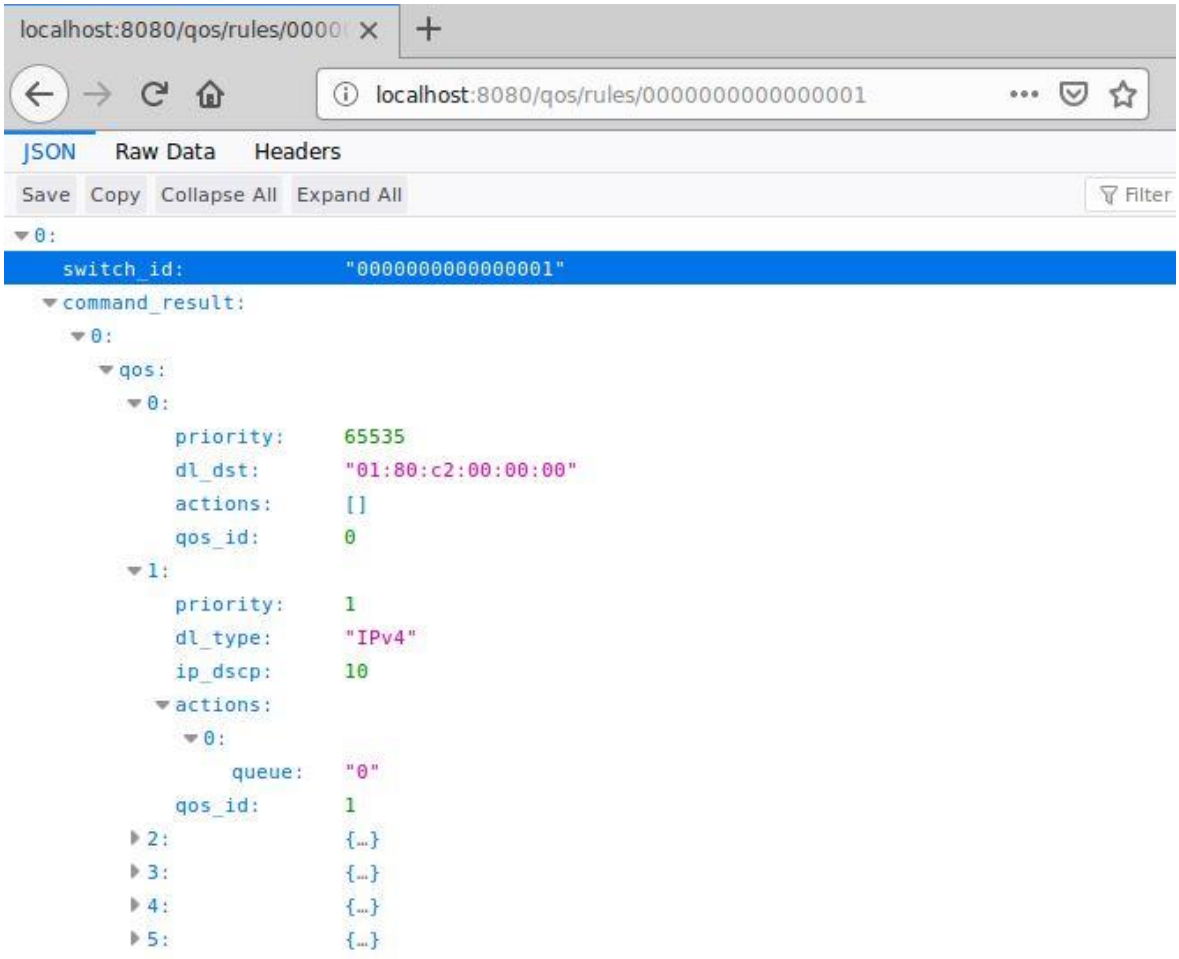
Çizelge 5.5. Kuyruklar için oluşturulmuş QoS kuralları

ToS	DSCP	Kuyruk ID	QoS ID
0x28	10	0	1
0x2c	11	1	2
0x30	12	2	3
0x34	13	0	4
0x0	0	0	5

Çizelge 5.5'te verilen eşleşmeye göre anahtara gelen paketin DSCP bitine bakılarak ilgili kuyruğa atamasını yapan ayarlamalar aşağıdaki gibidir. Daha anlaşılabilir olması açısından ekran görüntüsü yerine c0 kontrolcüsü üzerinde kuyruklara DSCP bitlerine göre paketlerin eklenmesi için oluşturulan komutlar verilmiştir.

```
# curl -X POST -d '{"match": {"ip_dscp": "10"}, "actions":{"queue": "0"}}' http://localhost:8080/qos/rules/0000000000000001
# curl -X POST -d '{"match": {"ip_dscp": "11"}, "actions":{"queue": "1"}}' http://localhost:8080/qos/rules/0000000000000001
# curl -X POST -d '{"match": {"ip_dscp": "12"}, "actions":{"queue": "2"}}' http://localhost:8080/qos/rules/0000000000000001
# curl -X POST -d '{"match": {"ip_dscp": "13"}, "actions":{"queue": "0"}}' http://localhost:8080/qos/rules/0000000000000001
# curl -X POST -d '{"match": {"ip_dscp": "0"}, "actions":{"queue": "0"}}' http://localhost:8080/qos/rules/0000000000000001
```

Yukarıdaki komutlarla s1 anahtarı üzerine gelen IP paketleri DSCP bitlerine göre farklı kuyruklara yönlendirilmektedir. İlgili anahtar üzerinde tanımlanmış kural setleri internet tarayıcısı üzerinden kontrol edilebilmektedir. Aşağıda s1 anahtarı üzerinde tanımlı kurallar görülmektedir. Ekran görüntüsünün çok fazla yer işgal etmemesi için sadece 1 numaralı QoS'a ait bilgiler Şekil 5.14'de ayrıntılı gösterilmiştir. Diğer QoS'ların bilgileri Çizelge 5.5'te tanımlandığı şekildedir.



Şekil 5.14. Tanımlanmış QoS listesi

Kuyruklar oluşturulup bu kuyruklara akışların yönlendirilmesi için kurallar da tanımlandıktan sonra hizmet kalitesi işlemleri tamamlanmıştır. Bu tez çalışmasında oluşturulan topoloji üzerindeki bir ana bilgisayara bağlı bir algılayıcıdan çıkan bir veri, üzerinde bulunduğu ana bilgisayarda işlenmeyecekse ön tanımlı bir IP adresi ile ağa gönderilmektedir. Ağa gelen IP paketinin ToS bit alanındaki etikete göre, kontrolcü bu paket için bir hedef IP adresi tespit ederek elde edilen bu IP adresini, paketin yeni hedef IP adresi olarak değiştirir. Daha sonra paketin ToS bit alanını ilgili verinin tipine göre hangi kuyruğa eklenecekse ona göre bir DSCP değeri ile günceller ve paketi gönderir. Bu DSCP değeri ile paket anahtarlar üzerinde ilgili kuyruklara eklenerek ihtiyacı olan ağ kaynaklarına sahip olmaktadır.

Ağ topolojinin oluşturulmasıyla başlanılan bu bölümde algılayıcılardan toplanan verilerden etiketli IP paketlerinin oluşturulması, YTA Ryu kontrolcü içerisinde yapılan işlemler, önerilen bir metot ve hizmet kalitesi işlemleri anlatıldıktan sonra bu bölümde yapılan

çalışmalar tamamlanmıştır. Bir sonraki bölümde gerçek veriler ile bu bölümde gerçekleştirilen yapı farklı senaryolar ile test edilmiştir.

6. TEST VE DEĞERLENDİRME

Bu bölümde 5. Bölüm'de tanımlanan yapının gerçek veriler ve farklı senaryolar ile testleri yapılmıştır. Bu kapsamda tez danışman hocamın sağladığı Gazi Üniversitesi içerisinde konumlandırılmış farklı algılayıcılardan toplanan verilerin tutulduğu bir veri kullanılmıştır.

6.1. Veri Seti

İçerisinde farklı algılayıcılardan toplanmış, toplamda 151736 adet verinin bulunduğu bir veri seti ile çalışılmıştır. Önceki bölümlerde anlatılan kurguya göre 4 farklı veri tipi ile test işlemleri yapılmıştır. Bu sebeple bu veri seti içerisinden Çizelge 6.1'de sayıları ve tipleri belirtilen veriler bir PowerShell betik yardımı ile filtrelenmiştir.

Çizelge 6.1. Kullanılan veri seti içerisindeki veri tipleri ve adetleri

Veri Tipi	Adet
Sıcaklık	21078
Nem	21078
Gaz	21394
Aydınlatma	12995
Toplam **:** 76545	

Toplam 76545 adet verilen oluşan bu veriler, 14.05.2019 ile 29.05.2019 tarihleri arasında 15 günlük bir süre içerisinde toplanmıştır. Bu verilerin ham hali bir PowerShell betik yardımı ile Çizelge 6.1'de verilen veri tipleri haricindeki verilerin de filtrelendiği aşağıdaki forma dönüştürülmüştür. Orijinal veriden fazla veri tiplerinin çıkarılmış hali Şekil 6.1'de verilmiştir.

Önce

```
1  1,2019-05-14 13:09:27.608261,"coap:gassensor:3:1",3
2  2,2019-05-14 13:09:27.642766,"coap:gassensor:3:1",3
3  3,2019-05-14 13:10:04.678538,"mqtt:celsius:2:25.60",2
4  4,2019-05-14 13:10:04.721824,"mqtt:humidity:1:37.70",1
5  5,2019-05-14 13:16:13.326922,"mqtt:lux:4:65536.00",4
6  6,2019-05-14 13:17:52.728463,"mqtt:lux:4:65536.00",4
7  7,2019-05-14 13:18:57.156269,"mqtt:celsius:2:26.20",2
8  8,2019-05-14 13:18:57.217711,"mqtt:humidity:1:54.20",1
9  9,2019-05-14 13:19:14.290607,"mqtt:lux:4:65536.00",4
10 10,2019-05-14 13:19:27.357961,"coap:gassensor:3:175",3
```

Sonra

```
1  2019-05-14 13:09:27.608261 gassensor 1
2  2019-05-14 13:09:27.642766 gassensor 1
3  2019-05-14 13:10:04.678538 celsius 25.60
4  2019-05-14 13:10:04.721824 humidity 37.70
5  2019-05-14 13:16:13.326922 lux 65536.00
6  2019-05-14 13:17:52.728463 lux 65536.00
7  2019-05-14 13:18:57.156269 celsius 26.20
8  2019-05-14 13:18:57.217711 humidity 54.20
9  2019-05-14 13:19:14.290607 lux 65536.00
10 2019-05-14 13:19:27.357961 gassensor 175
```

Şekil 6.1. Orijinal veriden fazla veri tiplerinin çıkarılmış hali

Daha sonra bu veri seti OutData dosyasındaki veri formatına uygun şekilde güncellenmiştir. Ancak öncelikle belirlenen bu 4 veri tipleri için ToS bitlerine yazılacak onaltılık tabanda belirlenmiş etiketleri ve kullanacakları portlar Çizelge 6.2'de verilmiştir.

Çizelge 6.2. Veri tipleri için atanan onaltılık etiketler ve port bilgileri

Veri Tipi	Onaltılık Etiket	Port
Sıcaklık	0xe4	3001
Nem	0xe8	3002
Gaz	0xec	3003
Aydınlatma	0xf0	3004

Çizelge 6.2'ye referans alınarak veri setinden oluşturulmuş yeni OutData dosyasının içeriği aşağıda Şekil 6.2'de gösterildiği gibidir.

Önce

```
1  2019-05-14 13:09:27.608261 gassensor 1
2  2019-05-14 13:09:27.642766 gassensor 1
3  2019-05-14 13:10:04.678538 celsius 25.60
4  2019-05-14 13:10:04.721824 humidity 37.70
5  2019-05-14 13:16:13.326922 lux 65536.00
6  2019-05-14 13:17:52.728463 lux 65536.00
7  2019-05-14 13:18:57.156269 celsius 26.20
8  2019-05-14 13:18:57.217711 humidity 54.20
9  2019-05-14 13:19:14.290607 lux 65536.00
10 2019-05-14 13:19:27.357961 gassensor 175
```

Sonra

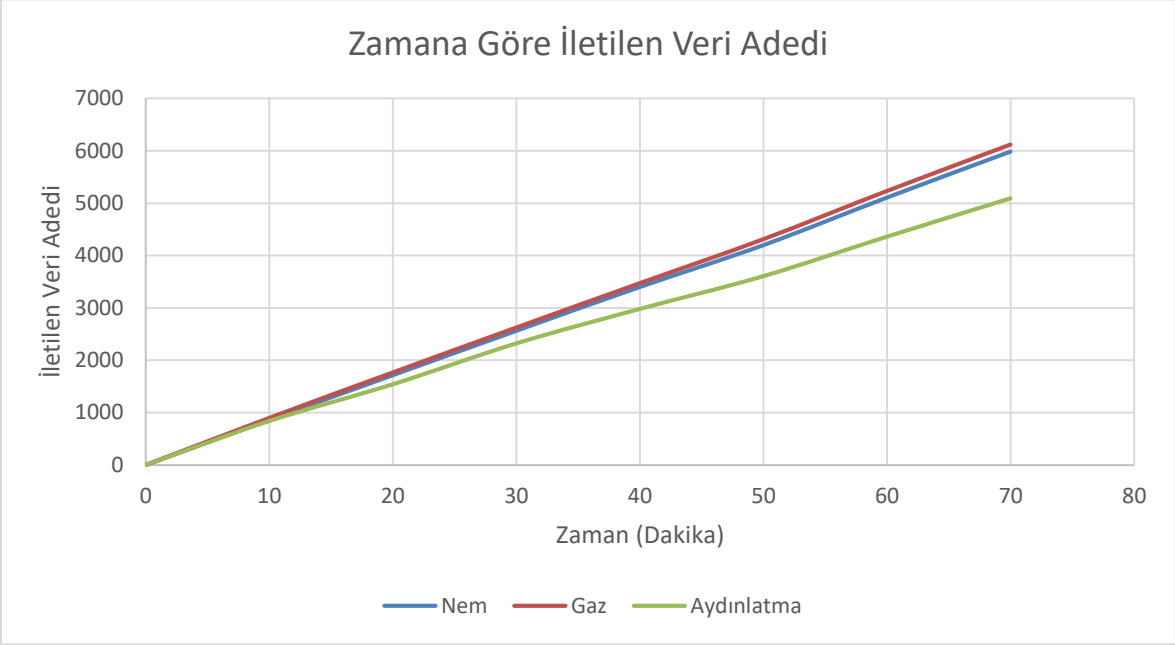
```
1  0xec 1 3003
2  0xec 1 3003
3  0xe4 25.60 3001
4  0xe8 37.70 3002
5  0xf0 65536.00 3004
6  0xf0 65536.00 3004
7  0xe4 26.20 3001
8  0xe8 54.20 3002
9  0xf0 65536.00 3004
10 0xec 175 3003
```

Şekil 6.2. Veri setinden OutData dosya formatına dönüşüm

OutData dosyası, mevcutta kullanılan Python betiklerin işleyeceği formata dönüştürülmüştür. OutData dosyası bu bölümde gerçekleştirilecek testler için kullanıma hazırlanmıştır.

Veri setinden bulunan 151736 adet veriyi ilgili ana bilgisayarlara gönderecek betik çalıştırılmış ancak tüm verilerin gönderilmesi uzun zaman alacağından betiğin çalışması 24342 verinin gönderimi tamamlandıktan sonra betik kesilmiştir. h1 ana bilgisayarı, bu 24342 veriyi ön tanımlı bir IP adresi ile ağa göndermiş ve bu verilerden 6285 tanesi Nem verisi olarak h2 ana bilgisayarına, 6419 tanesi Gaz verisi olarak h3 ana bilgisayarına ve 5354 tanesi Aydınlatma verisi olarak h4 ana bilgisayarına yönlendirilmiştir. 6284 tane Sıcaklık verisi ise üzerinde çalışılan ana bilgisayarda işleneceği için ağa gönderilmeyip yereldeki

InData dosyasının içerisine yazılmıştır. Birim zamanda h1 ana makinesinden diğer ana makinelere gönderilen veri sayısı Şekil 6.3'deki grafikte verilmiştir.

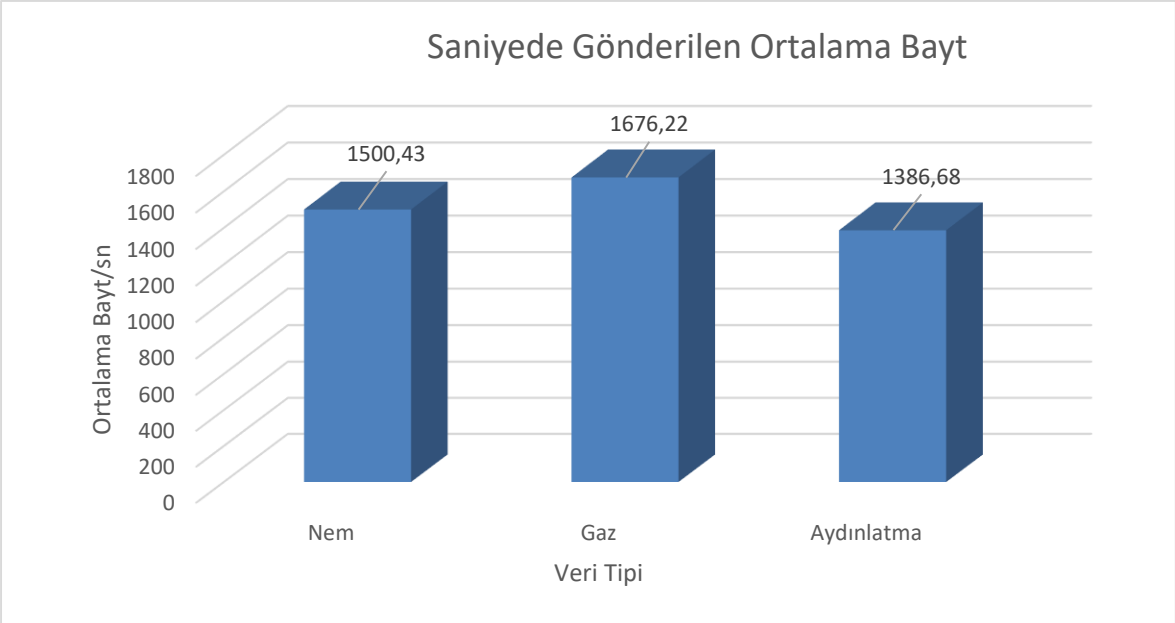


Şekil 6.3. Zamana göre iletilen veri adedi

Buradaki süre IP paketinin ağa çıktıktan sonra hedefe varıncaya kadarki geçen süresi olarak düşülmemelidir. Grafikte belirtilen sürenin içerisinde;

- h1 ana bilgisayarı üzerinde bulunan OutData dosyasının içerisini sürekli okuyarak satırları parçalayıp bu bilgilerden etiketli IP paketini oluşturan Python betiğin çalışması.
- Oluşturulan paketin ağa gönderilerek kontrolcü tarafından hedef tespitinin yapılması ve paketin yeniden yönlendirilmesi.
- Paketin hedefe vardıktan sonra içerisindeki verinin InData dosyası içerisine yazılması işlemleri için harcanan süreler vardır.

Test işlemlerini hızlandırmak adına, veri seti içerisinde bulunan ilk 1000 veri ile test işlemleri devam ettirilmiştir. Test karışıklığını önlemek için veri gönderimi tek taraflı yapılmaktadır. Sıcaklık verisi h1 ana bilgisayarı üzerinde tutulduğu için bu veri tipi ağa gönderilmemektedir. Geri kalan Nem, Gaz ve Aydınlatma verileri için saniyede gönderilen ortalama bayt sayıları Şekil 6.4'te verilmiştir.



Şekil 6.4. Saniyede gönderilen ortalama bayt

Yukarıda veri tipleri için ortalama gönderilen bayt sayıları arasındaki fark, her tipi işleyen ana bilgisayarın ağ üzerinde farklı konumlarda olmasından kaynaklanmaktadır. Burada h1 ana bilgisayarından çıkan farklı tipteki veriler, işlenecekleri ana bilgisayarlara varmak için farklı sayıda anahtardan geçip hedefe varmaktadırlar. Bu sebeple yukarıdaki ortalama gönderilen baytlarda farklılıklar görülebilmektedir.

6.2. Hizmet Kalitesi Testi

Ağa gelen paketlerdeki verilerin önemine göre paketin iletim önceliği değişebilmektedir. Algılayıcılardan toplanıp işlenmesi için ilgili ana bilgisayarlara yönlendirilecek veriler için de farklı öncelikler olabilmektedir. Paketleri anahtarlarda belirli bir önceliğe göre yönlendirebilmek için farklı kaynaklara sahip kuyruklar oluşturulmaktadır. Bu kuyruklara belirli kurallara göre paketler eklenmekte ve yönlendirme işlemleri bu kuyruklar dikkate alınarak yapılmaktadır. Bölüm 5'te kuyrukların hangi kaynaklarla kaynaklar nasıl oluşturulacağı anlatılmıştır. Bu bölümde sadece hizmet kalitesi kullanılarak yapılan yönlendirmelerdeki test sonuçlarına yer verilmiştir.

Öncelikle gerçek veri seti ile test etmeden önce iPerf [48] aracı ile test işlemi yapılmıştır. Kuyruklar için ayrılmış kaynakları doğru bir şekilde test edebilmek için bu yöntem seçilmiştir. Gerçek veri seti ile yapılan test işlemlerine çalışma alt yapısı, oluşturulan Python

betikler ve ortamın çalıştığı gerçek makinenin performansı gibi farklı parametreler etki etmektedir. Bu sebeple tanımlanan kuyrukların performansı ayrık olarak test edilmiştir.

Oluşturulan kuyruklardan kuyruk numarası 1 olan ve maksimum 250Kbps bant genişliği tahsis edilen kuyruk için yapılan test aşağıda Şekil 6.5'te görülmektedir.

```
mininet> h1 iperf -c 10.0.0.2 -p 5002 -u -b 300K --tos 0x2c
------------------------------------------------------------
Client connecting to 10.0.0.2, UDP port 5002
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.1 port 48239 connected with 10.0.0.2 port 5002
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec   369 KBytes   300 Kbits/sec
[  3] Sent 257 datagrams
[  3] Server Report:
[  3]  0.0-12.4 sec   369 KBytes   244 Kbits/sec   9.083 ms    0/  257 (0%)
```

Şekil 6.5. Maksimum 250Kbps bant genişliğine sahip kuyruk

Ana bilgisayar h1'den h2'ye doğru olan iletişim için oluşturulmuş kuyruk için yukarıdaki gibi bir ağ trafiği oluşturulur. Burada ağ trafiği oluşturulurken IP paketleri için ToS bit alanları paketleri kuyruk 1'e yönlendirecek şekilde ayarlanır. 0x2c ToS biti ile s1 anahtarına gelen paketler, 1 numaralı kuyruğa eklenir ve bu kuyrukta bu veri tipi için maksimum 250Kbps bant genişliği ayrılır.

Diğer bir kuyruk olan 2 numaralı, maksimum 100Kbps bant genişliği tahsis edilen kuyruk için yapılan test aşağıda Şekil 6.6'da görülmektedir.

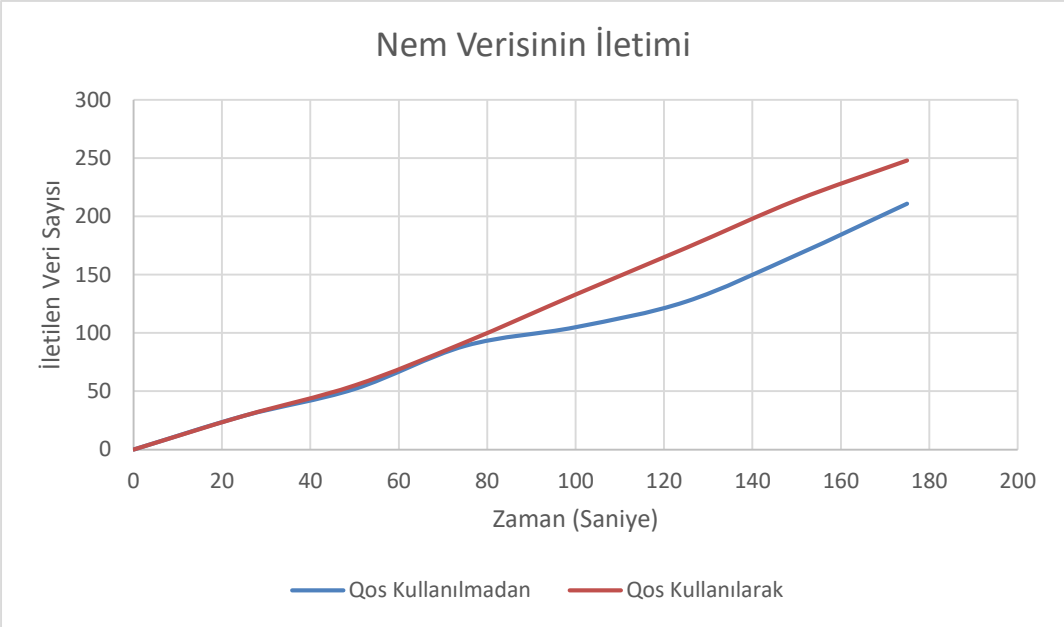
```
mininet> h1 iperf -c 10.0.0.3 -p 5003 -u -b 110K --tos 0x30
------------------------------------------------------------
Client connecting to 10.0.0.3, UDP port 5003
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.1 port 40471 connected with 10.0.0.3 port 5003
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.2 sec   136 KBytes   110 Kbits/sec
[  3] Sent 95 datagrams
[  3] Server Report:
[  3]  0.0-11.4 sec   136 KBytes  98.3 Kbits/sec  14.176 ms    0/   95 (0%)
```

Şekil 6.6. Maksimum 100Kbps bant genişliğine sahip kuyruk

Yine ana bilgisayar h1'den h3 ana bilgisayarına doğru olan iletişim için oluşturulmuş kuyruk için yukarıdaki gibi bir test trafiği oluşturulur. Ağ trafiği için oluşturulan IP paketlerinin ToS bit alanları 0x30 olarak ayarlanıp ağa gönderilir. Ağda 0x30 ToS biti ile s1 anahtarına gelen IP paketleri yukarıda tanımlanan kurallar gereği 2 numaralı kuyruğa koyulur. 2 numaralı kuyruk için maksimum 100Kbps bant genişliği ayrılmıştır. Yukarıda görüldüğü gibi 2 numaralı kuyruğa eklenen trafik 100Kbps'nin üzerine çıkamamıştır.

Yukarıdaki test sonuçları da göz önünde bulundurularak IoT cihazlarından toplanan verilerin önceliğine göre farklı kuyruklar oluşturup bu kuyruklara farklı kaynaklar tahsis edilebilmektedir. Kuyruklara tahsis edilen kaynakların performansını doğru bir şekilde test edebilmek için yukarıda iPerf ile yapılan testlerin benzerleri aşağıda gerçek veriler ile yapılmıştır.

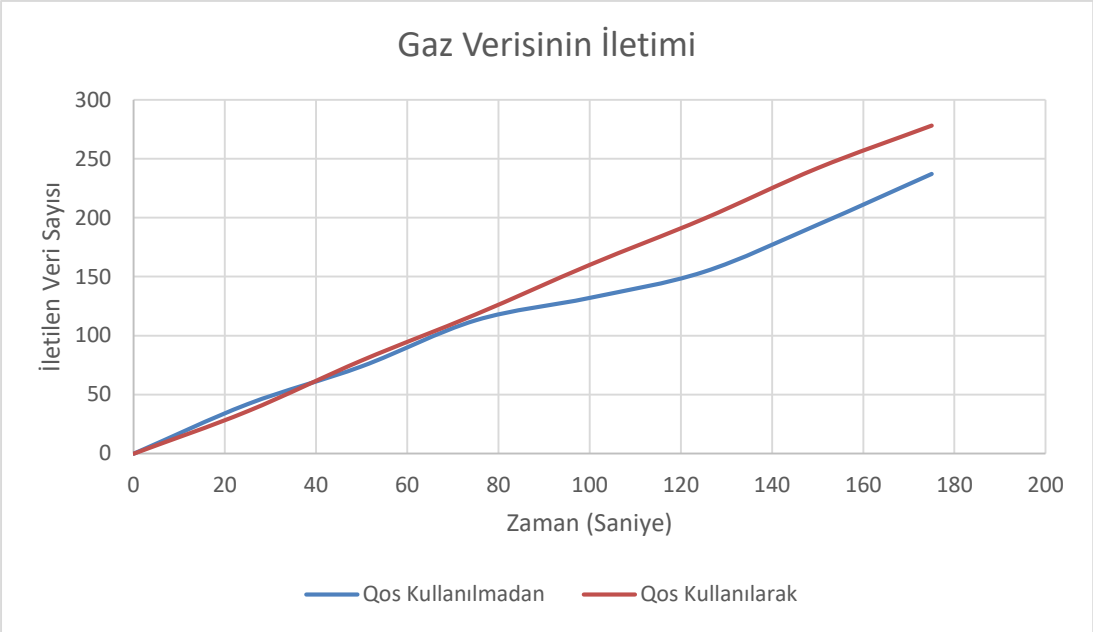
Ana bilgisayar h1'e bağlı bir algılayıcıdan toplayan nem verisi, h1 ana bilgisayarı üzerinde işlenmeyeceği için ön tanımlı bir IP adresi ve ToS biti etiketli halde ağa gönderilmektedir. Kontrolcü gelen nem verisinin etiketine bakarak gideceği hedef IP adresini tespit edip paketi tekrar yönlendirmeden önce nem verisinin iletim önceliğine göre DSCP bit alanını günceller. Daha sonra ilgili porta yönlendirerek işlemi bitir. Daha önce oluşturulan QoS kuralları tarafından bu paket DSCP bitine bakılarak ilgili anahtardaki ilgili kapıdaki ilgili kuyruğa eklenir. Testi yapılan bu yöntemde öncelikle herhangi bir QoS işlemi yapılmadığı durumda nem verisinin iletim durumu ile QoS işlemi uygulandıktan sonra nem verisinin iletim durumu aşağıda Şekil 6.7'de gösterilmektedir.



Şekil 6.7. Nem verisinin QoS ile ve QoS kullanmadan iletimi

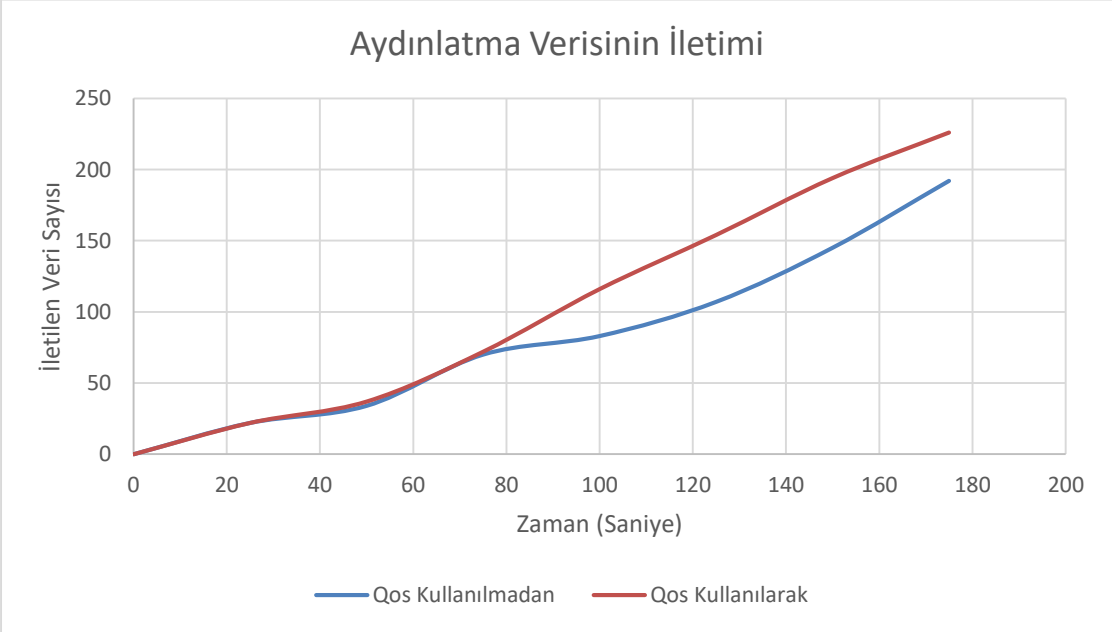
Yukarıdaki grafik için oluşturulan test senaryosunda öncelikle herhangi bir kuyruk oluşturmadan direk ağa gönderilen nem verilerinin hedef ana bilgisayara varması için geçen süre ile belirli bir iletim oranı garanti edilmiş bir kuyruğa eklenerek gönderilen nem verisinin karşılaştırılması görülmektedir. Burada dikkat edilmesi gereken husus belirli bir iletim oranı garanti edildiğinde nem verisinin ilgili hedef ana bilgisayara varış için daha az zaman kullandığıdır.

Yukarıdaki nem verisi için oluşturulmuş test senaryosu gaz ve aydınlatma verileri için de oluşturulmuş ve çıkan grafikler birbirlerine oldukça yakın sonuçlar vermiştir. Aşağıda Şekil 6.8'de gaz verisinin QoS kullanılarak ve QoS kullanılmadan gerçekleştirilmiş test sonuçları verilmiştir.



Şekil 6.8. Gaz verisinin QoS ile ve QoS kullanmadan iletimi

Aşağıda Şekil 6.9'da ise aydınlatma verisi için gerçekleştirilmiş QoS kullanılarak ve QoS kullanılmadan yapılan test sonuçları verilmiştir.

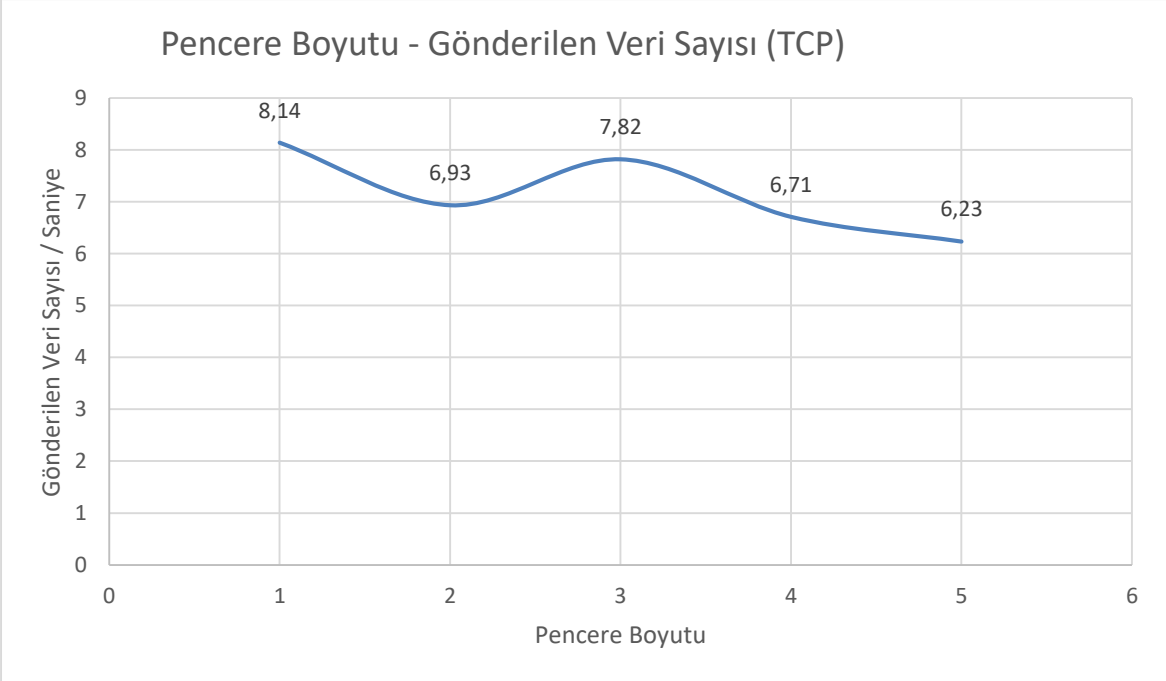


Şekil 6.9. Aydınlatma verisinin QoS ile ve QoS kullanmadan iletimi

Her bir veri tipi için yukarıda verilen grafikler dikkate alındığında QoS kullanılarak paketler için belirli bir iletim oranının garanti edilmesi iletim zamanlarına önemli bir katkı sağlamıştır.

6.3. Önerilen Pencere Yönteminin Testi

Test işlemleri öncelikle TCP protokolü kullanılarak gerçekleştirilmiştir. Önerilen pencere boyutu yönteminin kullanılıp kullanılmamasına bakılmaksızın, TCP protokolü ile gönderilen paketlerde herhangi bir kayıp yaşanmamıştır. Ayrıca önerilen pencere yönteminin saniyede gönderilen paket sayısına pozitif yönde bir etkisi olmadığı gözlemlenmiştir. Hatta pencere boyutu hesaplamalarından kaynaklı bir miktar verim kaybı yaşanmıştır. Ayrıca pencere boyutu farklı parametrelerle test edilmiş ve pencere boyutunun bazı değerler için iletim zamanına olumlu katkı sağladığı görülmüştür. Şekil 6.10'de pencere boyutunun saniyede gönderilen veri sayısına etkisi görülmektedir.

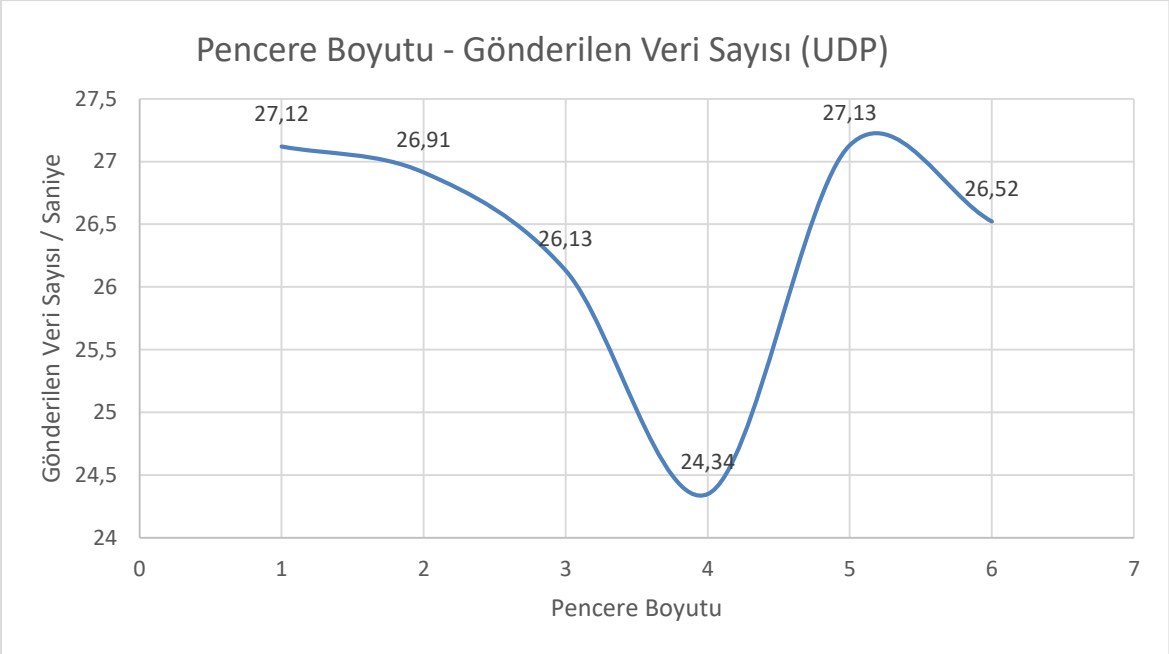


Şekil 6.10. Pencere boyutunun veri iletimine etkisi (TCP)

Pencere boyutu hesaplamak için kullanılan Python betiklerden kaynaklanan gecikme ile pencere boyutu yöntemi saniyede iletilen veri sayısını azaltmıştır. Ancak Python betiklerden kaynaklı paket oluşturulması ve iletilmesi ile ilgili gecikmeler ortadan kaldırılırsa pencere boyutunun veri iletimine katkısı olabilir. 2 ile test edilen pencere boyutu yöntemiyle saniyede gönderilen paket sayısını azalırken 3 pencere boyutu ile gönderilen paket sayısında artış görülmüştür. Daha sonra pencere boyutu arttıkça iletilen veri miktarının da azaldığı görülmektedir. Buradan uygun bir pencere boyutu ile veri iletimine olumlu katkı

sağlanabileceği çıkarılabilir. Gelecekte alt yapı performansı iyileştirilerek bu durum farklı senaryolar ile test edilebilir.

Aynı test işlemleri UDP protokolü kullanarak da gerçekleştirilmiş ve pencere boyutunun belirlenip belirlenmemesine bakılmaksızın UDP ile gönderilen ilk paketlerin hedefe ulaşmadığı gözlemlenmiştir. Pencere boyutu yönteminin kullanılmadığı senaryoda eğer aynı tipten ardışık birden fazla veri yoksa paket kayıp oranlarının %100lere çıktığı gözlemlenmiştir. UDP protokolü kullanılarak gerçekleştirilen pencere boyutunun veri iletimine etkisini gösteren test sonuçları Şekil 6.11'da verilmiştir.



Şekil 6.11. Pencere boyutunun veri iletimine etkisi (UDP)

Yukarıdaki grafikte de görüldüğü üzere hiç pencere boyutunun tanımlanmadığı durumdaki veri iletimi, pencere boyutunun 5 olarak tanımlandığı durumdaki veri iletiminin az da olsa gerisinde kalmıştır. Kullanılan veri setinin de karakteristiğine göre en uygun pencere boyutu ile veri iletim hızında önemli artış sağlanabilir. UDP protokolü ile gönderilen ilk paketlerin de kayıp olduğu göz önünde bulundurulduğunda uygun bir pencere boyutu ile veri iletiminin etkinliği artırılmış olur.

7. SONUÇ VE ÖNERİLER

Bu tez çalışması kapsamında sayıları ve kullanım alanları gün geçtikçe artan IoT cihazlarının topladıkları farklı türlerdeki verileri, bu verilerin işleneceği özelleşmiş farklı ana bilgisayarlara yönlendirebilmek için YTA tabanlı bir yöntem tasarlanmıştır. Bu kapsamda öncelikle Amerika'da onlarca şehri kapsayacak şekilde oluşturulmuş ağ test ve geliştirme çalışmaları için kullanılan Internet2 ağ topolojisinden esinlenilerek bir topoloji oluşturulmuştur. Internet2 ağ topolojisinin orijinal halinde olup da herhangi bir dallanma oluşturmayan ara ağ düğümleri topolojiden çıkarılarak ağ topolojisi sadeleştirilmiştir. Daha sonra tek bir makine üzerinde sanal bir alt yapı sağlayarak çok çeşitli ağ yapıları oluşturup test edilebilecek bir ortam sağlayan Mininet'e bu ağ topolojisi aktarılmıştır. Mininet üzerinde oluşturulan bu ağın farklı düğümlerine, her birine farklı türlerde veri toplayan IoT cihazlarının bağlı olduğu varsayılan 4 adet ana bilgisayar bağlanmıştır. IoT cihazları tarafından toplanıp bağlı oldukları ana bilgisayarlar üzerindeki bir dosyaya yazılan ve mevcut ana bilgisayarlarda işlenmeyecek türdeki verileri diğer ilgili ana bilgisayarlara gönderebilmek için IP paketleri oluşturan bir Python tabanlı betik oluşturulmuştur. Oluşturulan bu betik ile gönderilecek verilerin tutulduğu dosyayı belirli aralıklarla kontrol eden ve gönderilecek her bir veri için Netcat aracı ile IPv4 paketleri oluşturan bir yapı oluşturulmuştur. Netcat aracının sağlamış olduğu IP paket başlık alanlarını güncelleyebilme yeteneği sayesinde ağa gönderilecek paketlerin 8 bitlik TOS bit alanları, paketteki verinin türü ile etiketlenmiş ve yine Netcat aracı ile paket ağa gönderilmiştir. Kontrol düzlemi ve veri düzlemini birbirinden ayırarak ağ yönetimini daha esnek hale getiren YTA Ryu kontrolcüsü ile ağa gelen bu etiketli paket yakalanarak paketin etiketine göre yeni hedef IP adresi tespit edilmiş, paketin hedef IP adresi değiştirilmiş ve paketin hedefe ulaşması sağlanmıştır.

IoT cihazlarından toplanıp ilgili hedeflere gönderilecek farklı türdeki verilerin birbirlerine göre önceliğinin olabilmektedir. Bu yüzden ağa gelen paketler için QoS adımları uygulanarak veri türlerine göre farklı ağ kaynakları tahsis edilmiş kuyruklara sokulmaktadır. Yapılan QoS testlerine göre belirli ağ kaynakları garanti edilerek gerçekleştirilen veri iletimi ile veri iletiminin etkinliği artırılmıştır.

Ana bilgisayarlar arasındaki veri iletiminin etkinliğini artırmak adına bir de yöntem önerilmiştir. Bu yönteme göre IP paketlerini oluşturup ağa gönderen betik, bir türdeki veri sayısının belirli bir sayıya ulaşmasını bekleyip daha sonra bu verileri gönderecek şekilde güncellenmiş ve test işlemleri ile önerilen yöntemin etkinliği değerlendirilmiştir. Pencere boyutu şeklinde tanımlanan bu yöntemde paketin gönderilmesi için ulaşması gereken sayıyı ifade eden pencere boyutu parametresinin seçimi oldukça önem arz etmektedir. Gerçekleştirilen testlerde farklı pencere boyutu parametrelerinin veri iletimine etkileri ayrıntılı irdelenmiştir.

Öncelikle YTA ile ana yönlendirme mekanizmasının oluşturulduğu ardından hizmet kalitesi ve önerilen pencere boyutu yöntemi ile farklı test işlemlerinin gerçekleştirildiği bu tez çalışmasında bir takım zorluklarla karşılaşılmıştır. Bu zorlukların çoğu, YTA ile oluşturulan yöntemin test edilebilmesi için oluşturulan test alt yapısında karşılaşılmıştır. Sınırlı donanım kaynakları ve performans değerlendirmesi yapılmamış Python betiklerle oluşturulan test alt yapısı, önerilen yöntemin testinde yetersiz kalmıştır. Ancak gerçekleştirilen testler, aynı ortamda aynı kaynaklarla gerçekleştirildiği için yöntemin değerlendirilmesi için önemli sonuçlar ortaya koymuştur. Gelecek çalışması olarak daha performanslı yazılım dilleri ile oluşturulmuş, IP paketlerini oluşturan ve etiketli bir şekilde ağa gönderen yardımcı araçlar oluşturularak önerilen yöntemin etkinliği daha detaylı incelenebilir. Böylelikle yönteme geliştirilmesine katkı sağlayacak sonuçlar elde edilebilir.

Yapılan bu çalışma ile IoT ve 5G gibi gelişmekte olan teknolojiler için etkili bir içerik tabanlı yönlendirme yöntemi ortaya koyulmuştur. Kullanılan ağ topolojisinin karmaşıklığı, IP paketlerini etiketleme yöntemi ve paketlerin yönlendirilme algoritmaları bakımından mevcuttaki çalışmalardan ayrışan bu çalışmanın performansını test etmek için farklı senaryolar ile test işlemleri gerçekleştirilmiştir. Gerçek ortamdan toplanan verileri barındıran bir veri seti ile Mininet ortamında TCP protokolü ile gerçekleştirilmiştir. Bu testlere ek, IoT cihazlarının gerçek zamanlı veriler toplayabileceği de göz önünde bulundurularak aynı testler UDP protokolü ile de gerçekleştirilmiştir. Önerilen pencere yöntemi kullanılarak UDP protokolü ile gönderilen paketlerde seçilen pencere boyutu parametresine göre önemli veri iletim oranları elde edilmiştir. Elde edilen bu sonuçlara göre; YTA kullanılarak içerik tabanlı yönlendirme ile IoT sistemlerinde etkili ve gerçek zamanlı veri işlemeyi sağlayabilmek için önerilen yöntemin kullanılabileceği ortaya konulmuştur.

KAYNAKLAR

1. Ashton, K. (2009). That ‘internet of things’ thing. *RFID journal*, *22*(7), 97-114.

2. İnternet: Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017. URL: https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billionconnected-things-will-be-in-use-in-2017-up-31-percent-from-2016. Son Erişim Tarihi: 22.09.2019.

3. İnternet: Jose S. Fog computing and the Internet of Things: Extend the cloud to where the things are. CA, USA, Cisco, White Paper. URL: http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf. Son Erişim Tarihi: 22.09.2019.

4. Bonomi F., Milito R., Zhu J. ve S. Addepalli. (2012). Fog computing and its role in the Internet of Things. in Proc. 1st Ed. Workshop Mobile Cloud Comput. (MCC), 13–16.

5. Okay, F. Y., Ozdemir, S. (2018). Routing in Fog-Enabled IoT Platforms: A Survey and an SDN-Based Solution. *IEEE Internet of Things Journal*, *5*(6), 4871-4889.

6. Kaur, K., Garg, S., Aujla, G. S., Kumar, N., Rodrigues, J. J., Guizani, M. (2018). Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay. *IEEE communications magazine*, *56*(2), 44-51.

7. Li, X., Li, D., Wan, J., Liu, C., Imran, M. (2018). Adaptive transmission optimization in SDN-based industrial Internet of Things with edge computing. *IEEE Internet of Things Journal*, *5*(3), 1351-1360.

8. Prakash, S. W. ve Deepalakshmi, P. (2017). *Server-based Dynamic Load Balancing.* International Conference on Networks & Advances in Computational Technologies (NetACT), 25-28.

9. Hamed, M. I., ElHalawany, B. M., Fouda, M. M. ve Eldien, A. S. T. (2017). *A new approach for server-based load balancing using software-defined networking.* Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), 30-35.

10. Kaur, S., Kumar, K., Singh, J. ve Ghumman, N. S. (2015). *Round-robin based load balancing in Software Defined Networking.* 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2136-2139.

11. Wang, R., Butnariu, D., Rexford, J. (2011). OpenFlow-Based Server Load Balancing Gone Wild. Hot-ICE, 11, 12-12. 110.

12. Koerner, M., Kao, O. (2012). *Multiple service load-balancing with OpenFlow*. In 2012 IEEE 13th International Conference on High Performance Switching and Routing, 210-214.

13. İnternet: RoundRobin. Round Robin Load Balancing Definition. URL: https://avinetworks.com/glossary/round-robin-load-balancing/http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf. Son Erişim Tarihi: 22.09.2019.

14. Long, H., Shen, Y., Guo, M., Tang, F. (2013). LABERIO: *Dynamic load-balanced routing in OpenFlow-enabled networks.* In 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA) 290-297.

15. Bertsekas, D. P., Gallager, R. G., Humblet, P. (1992). *Data networks (2).* New Jersey: Prentice-Hall International.

16. Li, Y., Pan, D. (2013). *OpenFlow based load balancing for fat-tree networks with multipath support.* In Proc. 12th IEEE International Conference on Communications (ICC'13), Budapest, Hungary, 1-5.

17. İnternet: Openflow. What is Beacon? URL: https://openflow.stanford.edu/display/Beacon.html. Son Erişim Tarihi: 22.09.2019.

18. İnternet: Statista. Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025. URL: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/. Son Erişim Tarihi: 19.05.2019.

19. Prajapati, A., Sakadasariya, A., ve Patel, J. (2018). *Software Defined Network: Future of Networking*. 2nd International Conference on Inventive Systems and Control *(ICISC).* 1351-1354.

20. Kreutz, D., Ramos, F. M., Verissimo, P., Rothenberg, C. E., Azodolmolky, S. ve Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, *103*(1), 14-76.

21. İnternet: Sdxcentral. What are SDN Northbound APIs (and SDN Rest APIs)? URL: https://www.sdxcentral.com/networking/sdn/definitions/north-bound-interfaces-api/. Son Erişim Tarihi: 19.05.2019.

22. İnternet: Sdxcentral. What are SDN Southbound APIs? URL: https://www.sdxcentral.com/networking/sdn/definitions/southbound-interface-api/. Son Erişim Tarihi: 19.05.2019.

23. Masoudi, R., ve Ghaffari, A. (2016). Software defined networks: A survey. *Journal of Network and computer Applications*, *67*, 1-25.

24. İnternet: Networklessons. Introduction to SDN (Software Defined Networking). URL: https://networklessons.com/cisco/ccna-routing-switching-icnd2-200-105/introduction-to-sdn-software-defined-networking. Son Erişim Tarihi: 19.05.2019.

25. İnternet: Datacomm. Software Defined Network. URL: https://www.datacomm.co.id/en/telco/sdn/. Son Erişim Tarihi: 25.05.2019.

26. Godanj, I., Nenadić, K. ve Romić, K. (2016). *Simple example of Software Defined Network.* International Conference on Smart Systems and Technologies (SST), 231-238.

27. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J. ve Turner, J. (2008). OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, *38*(2), 69-74.

28. Xia, W., Wen, Y., Foh, C. H., Niyato, D., ve Xie, H. (2014). A survey on software-defined networking. *IEEE Communications Surveys & Tutorials*, *17*(1), 27-51.

29. İnternet: Open Networking Foundation. OpenFlow switch specification version 1.5.1. URL: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf. Son Erişim Tarihi: 25.05.2019.

30. İnternet: Mininet. URL: http://mininet.org/. Son Erişim Tarihi: 19.05.2019.

31. İnternet: Wikipedia. Netcat. URL: https://en.wikipedia.org/wiki/Netcat. Son Erişim Tarihi: 19.05.2019.

32. İnternet: Ryu. URL: https://ryu.readthedocs.io/en/latest/getting_started.html#what-s-ryu. Son Erişim Tarihi: 19.05.2019.

33. İnternet: Mininet Topology Visualizer. URL: http://demo.spear.narmox.com/app/?apiurl=demo#!/mininet. Son Erişim Tarihi: 25.05.2019.

34. Castillo-Velazquez, J. I., Serrano-Martinez, D. J. ve Morales, A. (2017). *Emulation of the connectivity of backbone and management for the layer 3 service of INTERNET2: 2016 topology*. 2017 IEEE 37th Central America and Panama Convention (CONCAPAN XXXVII), 1-4.

35. Gitler, I., ve Klapp, J. (Eds.). (2016). *High Performance Computer Applications:* 6th International Conference, ISUM 2015, Mexico City, 25-32.

36. Rabinovitch, E. (1998). Internet2 [Your Internet Connection]. *IEEE Communications Magazine*, *36*(3), 17-18.

37. Yeung, F. (1997). Internet 2: scaling up the backbone for R&D. *IEEE Internet Computing*, *1*(2), 36-37.

38. İnternet: Internet2. About Intenet2: Accelerating research and education through community-developed technology. URL: https://www.internet2.edu/media/medialibrary/2014/03/17/AboutInternet2.pdf. Son Erişim Tarihi: 25.05.2019.

39. İnternet: Internet2. Internet2 Network Infrastructure Topology. URL: https://www.internet2.edu/media/medialibrary/2019/04/10/I2-Network-Infrastructure-Topology-All-legendtitle.pdf. Son Erişim Tarihi: 25.05.2019.

40. İnternet: Slashroot. Understanding Differentiated Services (TOS) field in Internet Protocol Header. URL: https://www.slashroot.in/understanding-differentiated-services-tos-field-internet-protocol-header. Son Erişim Tarihi: 19.05.2019.

41. İnternet: Tutorialspoint. IPv4 - Packet Structure. URL: https://www.tutorialspoint.com/ipv4/ipv4_packet_structure.htm. Son Erişim Tarihi: 19.05.2019.

42. Chi, P. W., Wang, M. H., Guo, J. W., ve Lei, C. L. (2016). *SDN Migration: An Efficient Approach to Integrate OpenFlow Networks with STP-Enabled Networks.* International Computer Symposium (ICS), 148-153.

43. Jasika, N., Alispahic, N., Elma, A., Ilvana, K., Elma, L., ve Nosovic, N. (2012). *Dijkstra's Shortest Path Algorithm Serial and Parallel Execution Performance Analysis.* 2012 Proceedings of the 35th İnternational Convention MIPRO, 1811-1815.

44. Yin, C., ve Wang, H. (2010). *Developed Dijkstra shortest path search algorithm and simulation.* 2010 International Conference On Computer Design and Applications, V1-116.

45. Internet: Hackerearth. Shortest Path Algorithms. URL: https://www.hackerearth.com/practice/algorithms/graphs/shortest-path-algorithms/tutorial/. Son Erişim Tarihi: 25.05.2019.

46. Liu, Y., Li, Y., Wang, Y., Yuan, J. (2015). Optimal scheduling for multi-flow update in Software-Defined Networks. *Journal of Network and Computer Applications*, *54*, 11-19.

47. Karakus, M., Durresi, A. (2017). Quality of service (qos) in software defined networking (sdn): A survey. *Journal of Network and Computer Applications*, *80*, 200-218.

48. İnternet: iPerf. URL: https://iperf.fr/. Son Erişim Tarihi: 22.09.2019.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ATALAY, Mustafa
Uyruğu : T.C.
Doğum tarihi ve yeri : 24.05.1990, Konya
Medeni hali : Evli
Telefon : 0 (538) 743 48 68
e-mail : atalaymst@gmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Yüksek lisans	Gazi Üniversitesi / Bilgisayar Mühendisliği	Devam ediyor
Lisans	Gazi Üniversitesi / Bilgisayar Mühendisliği	2013
Lise	Konya Meram Muhittin Güzel Kılınç Lisesi	2007

İş Deneyimi

Yıl	Yer	Görev
2017-Halen	Aselsan A.Ş.	Yazılım Mühendisi
2016-2017	TÜBİTAK YTE	Araştırmacı
2013-2015	VAKIFBANK	Uzman Yardımcısı

Yabancı Dil

İngilizce

Yayınlar

1. Atalay M., Demirci M. (2019, 26-28 Nisan). *IP Packet Marking and Forwarding Based on Content Type with SDN,* International Conference on Advanced Technologies, Computer Engineering and Science (ICATCES 2019), Alanya, Türkiye.

Hobiler

Kamp yapma, Yüzme, Kitap okuma

GAZİ GELECEKTİR..