T.C.
İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

YAZILIM TANIMLI AĞ DENETLEYİCİSİNE BAĞLI PAKET TRAFİK GÖZETLEME UYGULAMASI VE OPTİMİZASYONU

Şükran DEĞİRMENCİ

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

DANIŞMAN
Doç. Dr. Derya YILTAŞ KAPLAN

Eylül, 2018

İSTANBUL

Bu çalışma, 3.09.2018 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Anabilim Dalı

, Bilgisayar Mühendisliği Programında Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Jürisi

Doç. Dr. Derya YILTAŞ KAPLAN(Danışman)
İstanbul Üniversitesi
Mühendislik Fakültesi

Prof. Dr. Fırat KAÇAR
İstanbul Üniversitesi
Mühendislik Fakültesi

Doç. Dr. Berk CANBERK
İstanbul Teknik Üniversitesi
Bilgisayar ve Bilişim Fakültesi

Dr. Öğr. Üyesi Oğuzhan ÖZTAŞ
İstanbul Üniversitesi
Mühendislik Fakültesi

Dr. Öğr. Üyesi Özgür Can TURNA
İstanbul Üniversitesi
Mühendislik Fakültesi

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, İstanbul Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

ÖNSÖZ

Bu tez çalışması sürecinde yönlendirmeleri ve önerileri ile bana destek olan danışman hocam Doç. Dr. Derya Yıltaş Kaplan'a teşekkürlerimi sunarım. Ayrıca çalışmalarım boyunca karşılaştığım her türlü zorlukta daima yanımda olan ve bana destek çıkan değerli aileme ve sevgili eşime sevgi ve saygılarımı sunarım.

Eylül 2018 Şükran DEĞİRMENCİ

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

TABLO LİSTESİ

Sayfa No

SİMGE VE KISALTMA LİSTESİ

Simgeler **Açıklama**

f(x) : Amaç fonksiyonu
c(x) : Amaç fonksiyonunun iyilik derecesi
x : Kısıtlı tabu hareketleri
X : Bellekteki tüm hareketler
rx : Alınan paketler
tx : İletilen paketler

Kısaltmalar **Açıklama**

API : Application Programming Interface
CPU : Central Processing Unit
DoS : Denial of Service Attack
DDoS : Distributed Denial of Service Attack
GUI : Graphical User Interface
IP : Internet Protocol
ISO : International Organization for Standardization
LAN : Local Area Network
MAC : Media Access Control
MAN : Metropolitan Area Network
MAPE : Mean Absolute Percent Error
MITM : Man In The Middle
ONF : Open Network Foundation
OPENSIG : Open Signaling
OSI : Open Systems Interconnection
SDN : Software Defined Network (Yazılım Tanımlı Ağ)
TCP : Transmission Control Protocol
TLS : Transport Layer Security
UDP : User Datagram Protocol
WAN : Wide Area Network

ÖZET

YÜKSEK LİSANS TEZİ

YAZILIM TANIMLI AĞ DENETLEYİCİSİNE BAĞLI PAKET TRAFİK GÖZETLEME UYGULAMASI VE OPTİMİZASYONU

Şükran DEĞİRMENCİ

İstanbul Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman : Doç. Dr. Derya Yıltaş KAPLAN

Bu çalışma yazılım tanımlı ağ denetleyicisi kullanarak paket trafiğini takip etme amacıyla yapılmıştır. Bu amaç doğrultusunda yazılım tanımlı ağ denetleyicisi olarak floodlight ve çalışma platformu olarak eclipse kullanılmıştır. Doğruluğu artırmak adına beş ayrı topoloji üzerinde gerçekleştirilen takipler sonrasında paket istatistik bilgileri elde edilmiştir. Bu istatistik verileri matlab üzerinde işlenerek istenilen veri setleri oluşturulmuştur. Bu veri setleri matlab üzerinde yer alan nntool vasıtasıyla yapay sinir ağlarında kullanılarak herhangi bir zamandaki paket akışı tahmin edilmiştir. Yapay sinir ağları beş ayrı topoloji üzerinde denenmiş ve doğrulukları MAPE ve R^2 fonksiyonları ile test edilmiştir. Yapılan testler sonucunda doğruluğu en yüksek çıkan topoloji üzerinde optimizasyon işlemleri uygulanmıştır. Bu amaç doğrultusunda doğrusal arama, tabu arama, değiştirilmiş tabu arama ve tabu arama ile tavlama benzetim algoritmasının karışımı olmak üzere toplamda dört ayrı optimizasyon tekniği ile optimizasyon testleri gerçekleştirilmiştir. Sonuç olarak karışım algoritması ile gerçekleştirilen optimizasyon sonucunda en yoğun olan güzergâh en kısa sürede tespit edilmiştir.

Eylül 2018, 86 sayfa.

Anahtar kelimeler: Yazılım Tanımlı Ağ, Yapay Zekâ, Optimizasyon, Yapay Sinir Ağları

SUMMARY

M.Sc. THESIS

APPLICATION AND OPTIMIZATION OF PACKET TRAFFIC MONITORING DEPENDING ON SOFTWARE DEFINED NETWORK CONTROLLER

Şükran DEĞİRMENCİ

İstanbul University

Institute of Graduate Studies in Science and Engineering

Department of Computer Engineering

Supervisor : Assoc. Prof. Dr. Derya YILTAŞ KAPLAN

This study has been done for packet traffic monitoring depending on software defined network controller. For this purpose, floodlight is used as software defined network controller and Eclipse is used as working platform. In order to increase the accuracy, packet statistical information was obtained after follow up on five separate topologies. These statistical data were processed on matlab and the desired data sets were created. These data sets are used in artificial neural networks by nntool on matlab to predict packet flow at any time. Artificial neural networks were tested on five separate topologies and their correctness was tested with MAPE and R^2 functions. As a result of the tests performed, optimizations were applied on topology with the highest accuracy. For this purpose, a total of four optimization techniques have been tested with a combination of linear search, tabu search, modified tabu search and mixture algorithm (tabu search - simulated annealing). As a result, the most intensive route was determined as soon as possible with the optimization algorithm performed with the mixture algorithm.

September 2018, 86 pages.

Keywords: Software Defined Network, Neural Network, Artificial Intelligence, Optimization

1. GİRİŞ

Gelişen teknoloji ile beraber veri merkezleri gittikçe büyümektedir. Büyüyen veri merkezlerinde ise büyük hacimli, karışık ve düzensiz bilgiler yer almaya başlar. Büyük veri içerisinde yer alan bu bilgilerin anlamlı ve değerli olabilmeleri sebebiyle, bu verilerin işlenmesi gerekir. Büyük veri geleneksel yöntemlerle işlenemez, yönetilemez ve depolanamaz yani geleneksel ağ yönetimi yaklaşımı bu aşamada yetersiz kalmaktadır. Daha iyi bir ağ yaklaşımıyla, yeni metotlarla ve daha geniş bir bant aralığıyla bu veriler işlenebilir hale gelmektedir. Yazılım tanımlı ağ (Software Defined Network-SDN) bu ihtiyaçları karşılayan bir yöntem olarak karşımıza çıkmaktadır. SDN yönetim kolaylığını, donanım bağımsızlığını, dinamik, esnek ve ölçeklenebilir ağ mimarisini temin eder. Dolayısıyla bu geniş ve karmaşık ağ yönetimine etkili bir çözüm sunar.

İletişim ağının temel amacı, bilgi paketlerini bir noktadan diğerine aktarmaktır. Ağ içinde birden çok düğüme paket iletimi gerçekleştiği için bu da yoğun bir paket trafik akışına sebep olmaktadır. Bu esnada SDN kullanılarak denetleyici sayesinde, etkin ve verimli paket trafik akışı sağlanabilmektedir. Böylece yoğunluğa ve çeşitliliğe sebep olan trafiğin oluşturduğu karmaşanın önüne geçilerek, daha basit ve yönetimi daha kolay olan bir anlayış sunulmaktadır.

Bu tezde ağlardaki paket trafik akışının yazılım tanımlı ağ denetleyicisine bağlı olarak gerçekleştirilmesi ve burada elde edilen verilerin yapay zekâ optimizasyon teknikleri kullanılarak optimize edilmesi amaçlanmıştır. Elde edilen veriler ışığında yapay sinir ağları kullanılarak tahmin üzerine bir uygulama geliştirmek amaçlanmıştır.

2. GENEL KISIMLAR

2.1. AĞ

2.1.1. Ağ Nedir?

Ağ, kaynakları paylaşmak, dosya değişimi yapmak veya elektronik iletişimlere izin vermek için birbirine bağlı iki veya daha fazla bilgisayardan oluşan yapıdır. Ağ olmadığı durumlarda kaynaklar usb bellekler, CD'ler veya bunlara benzer taşınabilir depolama aygıtları ile başka bir bilgisayara aktarılabilmektedir. Fakat ağa bağlı bilgisayarlarda kaynaklar taşınmaktan ziyade ortak paylaşıma açılmaktadır. Bir ağdaki bilgisayarlar kablolar, telefon hatları, radyo dalgaları, uydular veya kızılötesi ışık ışınları ile birbirlerine bağlanabilmektedir.

Bilgisayar ağları kapsadığı alana göre yerel alan ağları (LAN), metropolitan alan ağları (MAN) ve geniş alan ağları (WAN) olmak üzere üç sınıfa ayrılmaktadır.

Yerel Alan Ağları (LAN): Nispeten küçük bir alanla sınırlı olarak birbirine yakın mesafede bulunan birkaç bilgisayarın bağlanmasıyla oluşan ağ yapısıdır. En küçük bilgisayar ağı modelidir [1]. Küçük yapısından dolayı hızlı bir yapıya sahiptir öyle ki şuan için en hızlı olan ağ yapısı modeli olarak kabul edilmektedir. İçerisinde Ethernet protokolü ve Ethernet anahtarlama cihazı barındırmaktadır [2].

Geniş Alan Ağları (WAN): Geniş coğrafi bölgelerdeki ağları birbirine bağlamak için kullanılmaktadır. Büyük ve daha karmaşık bir yapıya sahiptir. Bundan dolayı yerel alan ağlarına göre daha yavaştır [2]. Bu tür bir küresel ağa bağlanmak için özel transokeanik kablolama veya uydu uplinkleri kullanılabilmektedir.

Metropolitan Alan Ağları (MAN): Bir bölgenin veya bir şehrin iki ayrı yerel alan ağı ile birleşmesi sonucu oluşan ağ türüdür. Metropolitan denmesinin sebebi ise bundan yani ağın şehirleri kapsıyor olmasından kaynaklanmaktadır. Bu ağ çoğunlukla birkaç yüz kilometre mesafeye kadar olan cihazları birbirine bağlamak için kullanılmaktadır [1, 3, 4].

2.1.2. Ağ Topolojileri Nelerdir?

Ağ topolojisi, bir ağın fiziksel veya mantıksal düzenini ifade etmektedir. Farklı düğümlerin yerleşim şekli ve düğümlerin birbirlerine nasıl bağlandığı topoloji oluşturulurken üzerinde

durulan noktalardır. Alternatif olarak, ağ topolojisi verilerin bu düğümler arasında nasıl aktarıldığı olarak da ifade edilebilir.

İki türlü ağ topolojisi vardır. Bunlar fiziksel ve mantıksal topolojilerdir. Fiziksel topoloji, bağlı cihazların ve düğümlerin fiziksel düzenini vurgularken, mantıksal topoloji ağ düğümleri arasındaki veri aktarımı modeline odaklanmaktadır**.** Hem fiziksel hem de ağ topolojileri beş temel modele ayrılmaktadır [5]:

Bus Topoloji: Bir veri yolu ağında, tüm bilgisayarların genellikle tek arabirim üzerinden bir kabloyla birbirine bağlanması sonucu oluşmaktadır. Ana bilgisayar veri yoluna bir elektrik sinyali gönderdiğinde, sinyal veri yoluna bağlı bütün sunucular tarafından alınmaktadır. Sinyal hedefe ulaştığında veya bir sonlandırıcı ile karşılaştığında hattan ayrılmaktadır. Bus topolojisi bazı dezavantajlara sahiptir. Örneğin hatta fiziksel olarak bir kesinti gerçekleşirse bu tüm ağı etkilemekte ve ağın çökmesine sebep olmaktadır. Bir diğer dezavantajı ise veri yoluna eklenen her bilgisayarın ağın yükünü artırıyor olmasıdır. Kısacası bus topolojisinin bakımı ve işletilmesi konusunda bazı problemler bulunmaktadır [1, 3].

Ring Topoloji: Bus topolojisinde olduğu gibi ring topolojisinde de tek bir fiziksel hat bulunmaktadır. Adından da anlaşıldığı üzere ring topolojisi dairesel bir yapıya sahiptir. Halka üzerindeki bir bilgisayar tarafından gönderilen herhangi bir sinyal, halkaya bağlı tüm bilgisayarlar tarafından alınmaktadır. Sinyal hedefe ulaşana kadar tüm bilgisayarları dolaşmaktadır. Günümüzde bu topoloji yerine yıldız topoloji tercih edilmektedir [1, 3].

Star Topoloji: Bu tür topolojilerde, bilgisayarlar tek bir fiziksel ara yüze sahiptir ve her bir bilgisayar ve yıldızın merkezi arasında bir fiziksel bağlantı bulunmaktadır. Yıldızın ortasındaki düğüm, ya bir elektrik sinyalini güçlendiren bir ekipman parçası ya da ağ üzerinden değiştirilen mesajların formatını anlayan bir ekipman parçası gibi aktif bir cihaz olabilir. Merkezi düğümün başarısızlığı ağın başarısızlığına sebep olmaktadır. Ancak, bir fiziksel bağlantı başarısız olursa, ağdan yalnızca bir düğümün bağlantısı kesilir. Bu topolojinin en büyük avantajı ise topolojiye yeni bilgisayarlar ekleyerek büyümeyi kolay bir şekilde gerçekleştirebilmesidir [1, 3].

Tree Topoloji: Bu tür ağlar genellikle çok sayıda müşterinin çok uygun maliyetli bir şekilde bağlanması gerektiğinde kullanılmaktadır. Bus topoloji ve yıldız topolojinin karışımı olarak oluşturulmuştur. Yıldız topolojisindeki ağları birbirine bağlayarak büyük ağlar elde etmektedir. Kurulumu ve düzenlenmesi diğer topolojilere oranla daha zordur [1, 3, 4].

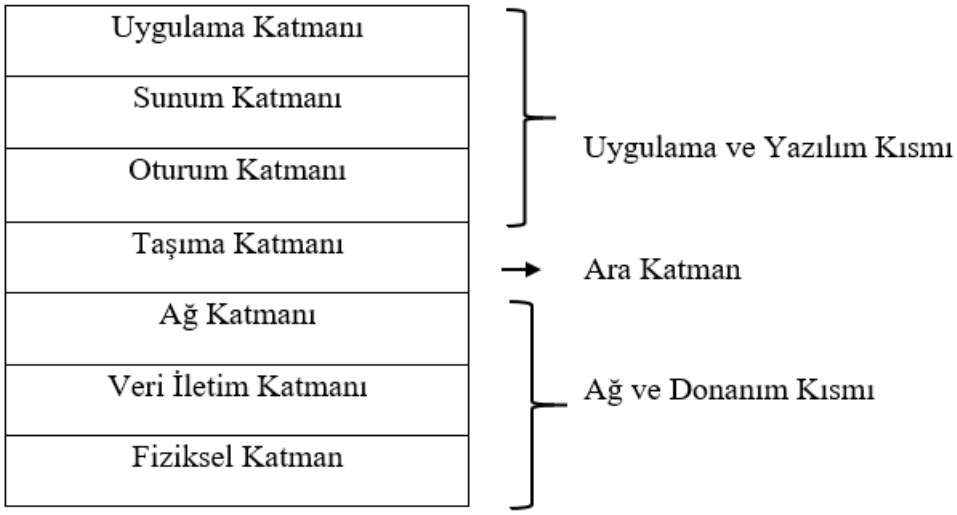
Mesh Topoloji: Ağ üzerinde bulunan bütün bilgisayarların birbirlerine direk olarak bağlanması sonucu oluşmaktadır. Ağdaki bilgisayarların birbirlerine direk bağlı olmalarından dolayı hedefe kısa süre içerisinde varılabilmektedir. İki bilgisayar arasında alternatif pek çok yol bulunmaktadır. Dezavantaj olarak da maliyetinin çok yüksek olduğu belirtilmektedir. [1].

2.1.3. Ağ Protokolleri Nelerdir?

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

2.1.3.1. OSI Referans Modeli

Dünyanın her yerinde bilgisayar ağlarını kullanan pek çok kullanıcı bulunmaktadır. Bu model ulusal ve uluslararası veri iletişimini sağlamak için standartlaştırılmış bir ağ sistemdir. ISO yani "Uluslararası Standardizasyon Örgütü" anlamına gelir ve OSI modeli olarak adlandırılmaktadır. Bu model ağ protokollerinin kim tarafından oluşturulduğuna bakılmaksızın tüm ağ elemanlarının beraber çalışmasına olanak sağlamaktadır. OSI mimarisi yedi katmandan oluşmaktadır. Her katman üstündeki katmanları desteklemek için belirli işlevleri gerçekleştirmekte ve altındaki katmanlara hizmetler sunmaktadır. Bu model, ağ işlemleri için genel bir görünüm sunmakta, bu da bilgisayar ağlarının ayrıntılarını anlamak için bir çerçeve görevi görmektedir. OSI modelinin katmanlı işlevsellik hali protokol yığını olarak adlandırılmaktadır. Protokoller ya da kurallar, çalışmalarını donanım ya da yazılımda ya da çoğu protokol yığınında olduğu gibi hem donanım hem yazılım arasında bir kombinasyon şeklinde yapabilmektedir. Katmanların yapısı gereği alt katmanlar donanım çalışmalarını gerçekleştirirken daha üst katmanlar yazılım işlemlerini gerçekleştirmektedir. En üst katmandan somut bilgi olarak yola çıkan veri, aşağı katmanlara doğru yol aldıkça 0 ve 1'lerden oluşan sinyallere dönüşmektedir [6, 7, 8].



Şekil 2.1: OSI Referans Modeli Katmanları

1. *Fiziksel Katman*: OSI modelinin en düşük tabakasıdır. Bu katmanda fiziksel bağlantı etkinleştirilir, korunur veya devre dışı bırakılır. Ağ üzerinden yapılandırılmamış ham verilerin iletilmesi ve alınması bu katmanda gerçekleşir. Bu işlemde datanın dijital bitlere dönüştürülerek elektrik sinyali veya optik sinyal haline gelmesiyle sağlanır. Veri kodlaması bu katmanda yapılmaktadır. Hub, repeater, kablolar gibi cihazlar bu katmanda bulunmaktadır. Fiziksel katmanda herhangi bir protokol yer almamaktadır [1, 9].
2. *Veri İletim Katmanı:* Bu katman, hattaki veri okumaktan ve yazmaktan sorumludur. Bağlantı hataları bu katmanda algılanmaktadır. Veri iletim katmanı, alttaki fiziksel katman tarafından sağlanan hizmet üzerine kurulmuştur. Bu katmanındaki iki varlık arasında değiştirilen bilgi birimi bir çerçevedir ve bu çerçeve sonlu bir bit dizisidir.
   Bu çerçevelerin iletilmesi ve alınması veri iletim katmanı tarafından yönetilmektedir. Bu katman sırasıyla alınan ve gönderilen çerçeveler için onaylar gönderir. Kabul edilmeyen çerçevelerin yeniden gönderilmesi de bu katman tarafından gerçekleştirilmektedir. Çerçevelerin kapasitesi dolduğunda verici düğümün durması sinyalini vermektedir [3, 9].
3. *Ağ Katmanı:* Ağ katmanı, adres atamasının gerçekleştiği katmandır. Bu katmanda ağda var olan bilgisayarların bilgisayara özel olacak şekilde adreslemesi gerçekleşmektedir. Sinyallerin kanallar arasında yönlendirilmesi bu katmanda gerçekleşmektedir. Bu

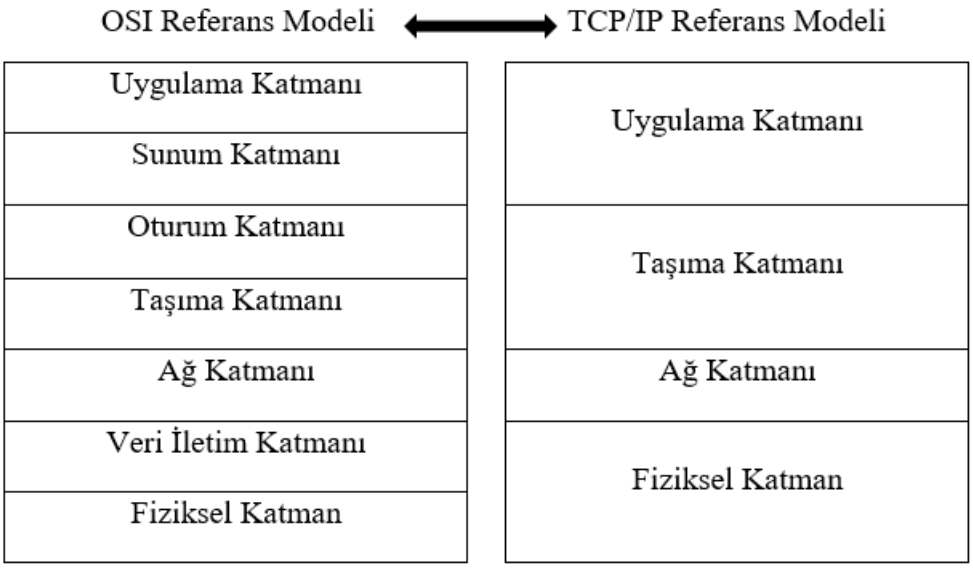
katman bir ağ denetleyicisi gibi davranarak alt ağ trafiğini yönetmektedir. Giden iletilerin paketlere bölündüğü ve gelen iletilerinde parçalanmış olan paketlerin daha yüksek seviyeler için bir araya getirildiği katmandır. Yönlendirme protokolleri bu katmanda yer almaktadır [4, 8].

4. *Taşıma Katmanı:* Bu katman, bilgisayarlar arasındaki uçtan uca gerçekleşen paket iletiminden sorumludur. Bu katmanda veri iletiminin ne şekilde olacağına karar verilir. Taşıma katmanı bir üst katman olan oturum katmanından mesajları alır ve iletiyi daha küçük birimlere dönüştürerek ağ katmanına aktarır. Bu şekilde verilerin ağ katmanında daha verimli bir şekilde işlenmesi sağlanmaktadır. Bu katmanda çoğunlukla kullanılan taşıma protokolleri ise Transmission Control Protocol (TCP) ve User Datagram Protocol（UDP）'dür [2, 8].
5. *Oturum Katmanı:* İki farklı uygulama arasındaki konuşmayı yönetir ve senkronize eder. Verilerin kaynaktan hedefe doğru olan veri akışları işaretlenir ve uygun şekilde yeniden senkronize edilir, böylece mesajlar vaktinden önce kesilmez ve veri kaybı yaşanmaz. İletimde kopmalar olduğu takdirde herhangi bir işaretli veri akışı noktasından başlanarak iletimin kaldığı yerden devam etmesi sağlanmaktadır [2,9].
6. *Sunum Katmanı:* Sunum katmanı, bir uygulamanın ağa gönderilecek verileri nasıl biçimlendirdiğinden sorumludur. Sunum katmanı temel olarak bir uygulamanın iletiyi okumasını veya anlamasını sağlamaktadır. Bu katman verileri alarak uygulama katmanı için verileri elverişli hale getirmektedir [6].
7. *Uygulama Katmanı:* Uygulama katmanı, OSI modelinin en üst tabakasını oluşturmaktadır. Bu katman alınan ve gönderilecek olan veri üzerinde kullanıcının işlem yapabilmesini sağlayacak olan uygulama programlarını içermektedir. Posta hizmetleri, dizin hizmetleri, ağ kaynağı vb. uygulama katmanı tarafından sağlanan hizmetlerden bazılarıdır [7, 8].

2.1.3.2. TCP/IP Referans Modeli

TCP / IP modeli günümüz internet mimarisinde kullanılan ağ modelidir. Protokoller, bir ağ üzerinden mümkün olan her türlü iletişimi yöneten kurallar kümesidir. Bu protokoller, kaynak ile hedef veya internet arasındaki verilerin hareketini tanımlar. TCP/IP'de bu protokollerden en yaygın olarak tercih edilen protokol yığınıdır. OSI modeli genel iletişim modelidir fakat TCP/IP

modeli tüm iletişim için internetin kullandığı sistemdir. TCP/IP modelinin yaygın olarak kullanılmasının pek çok sebebi bulunmaktadır. Farklı işletim sistemleri arasındaki veri alışverişi esnasında problem yaşanmaması, internet bazlı çalışması, üreticiden bağımsız olarak kullanılabilmesi bu sebeplerden bazılarıdır. OSI modelindeki yedi katman TCP/IP modelinde dört katman olarak gruplandırılmıştır. Bu katmanlar şunlardır: Uygulama, Taşıma, İnternet, Ağ Arayüzü [4, 9, 10]



Şekil 2.2: OSI Referans Modeli ve TCP/IP Referans Modeli

1. *Ağ Arayüzü (Fiziksel) Katmanı:* Elektrik sinyali olarak kablo üzerinden gelen sıfır ve birlerden oluşan verinin fiziksel olarak görüntülenmesi bu katmanda gerçekleşmektedir. Ethernet bu görüntülemeyi gerçekleştiren ve fiziksel katmanda çalışan kablolu yerel ağ teknolojisidir. Çerçevelerin hedefe gönderilmesi bu katmanda gerçekleşmektedir. OSI modelinin son iki katmanı olan veri bağı ve fiziksel katmanın ikisine birden karşılık gelmektedir [7, 10].
2. *Ağ Katmanı:* Ağ katmanı, TCP/IP modelinin ikinci katmanını oluşturmaktadır. OSI modelinin üçüncü katmanı olan ağ katmanına karşılık gelmektedir. Bu katman ağ üzerinden paketlerin yönlendirilmesinden sorumludur. Pakete kaynak ve hedef Ip bilgileri eklenerek paketin gideceği yer belirlenir ve paket datagram şekline dönüşür. Bu katmanda, bir üst katmandan gelen datagramlar daha küçük parçalar olan paketlere ayrılmakta ve bu paketlere hedef bilgisayarın adresi eklenmektedir [2, 10].

3. *Taşıma Katmanı:* Veri iletiminin nasıl gerçekleşeceği bu katmanda belirlenmektedir. Verilerin birleştirilmesi, parçalanması işlemleri taşıma katmanı tarafından yapılmaktadır. Taşıma katmanı, iletiyi küçük birimler haline getirerek ağ katmanı tarafından daha verimli bir şekilde işlenmesini sağlamaktadır. Ayrıca taşıma katmanında paketlerin sırayla gönderilmesi işlemi gerçekleştirilmektedir [8].
4. *Uygulama Katmanı:* Bu katman, kullanıcının ağ ile etkileşime girmesini sağlayan protokolleri içermektedir. Gönderilecek olan verinin türüne farklı protokoller çalıştırılır ve uygulamalarla protokollerin haberleşmesi sağlanmaktadır [8].

2.1.4. Ağ Cihazları Nelerdir?

Bir internet ağı kurulurken bilgisayarlarla beraber kurulumda bulunması gereken bazı cihazlar bulunmaktadır. Bu cihazlar ağı genişletmek, ağın güvenliğini sağlamak ve ağda bir hiyerarşi oluşturmak için kullanılmaktadırlar. Yoğun olarak tercih edilen ağ cihazları şu şekildedir: Tekrarlayıcılar (Repeater), Çoklayıcılar(Hub), Köprüler(Bridges), Anahtarlamalı Çoklayıcılar(Switch), Yönlendiriciler (Router)

Tekrarlayıcılar (Repeater): Fiziksel katmanda yer alan tekrarlayıcıların yaptığı iş, çok zayıflamışlar veya bozulmuş olan sinyallerin yenilenmesi işlemidir. Tekrarlayıcılar sinyali yükseltmemektedirler. Sinyalde problem olduğunda, sinyal biti birebir kopyalanır ve orijinal haliyle orijinal gücüyle yeniden oluşturulur. Tekrarlayıcılarda yönlendirme işlemi söz konusu değildir. İki portlu bir cihaz olarak tasarlanmıştır [11].

Çoklayıcılar (Hub): Fiziksel katmanda yer alan çoklayıcılar, ağ cihazlarını fiziksel olarak birbirine bağlamaktadır. Çoklayıcılar verileri filtrelemezler ve veri paketleri ağa bağlı tüm cihazlara gönderilir. Alınan paketlerde bir değişiklik gerçekleştirmeksizin, paketleri diğer cihazlara iletmek üzere tasarlanmışlardır. Bu da veri paketleri için verimsiz bir iletime ve zaman kaybına sebebiyet vermektedir. Aktif ve pasif çoklayıcılar olmak üzere iki türlü çoklayıcı bulunmaktadır. Aktif çoklayıcılar yukarıda bahsettiğimiz tekrarlayıcılarla aynı şeydir. Pasif çoklayıcılara göre daha akıllı bir işleyişleri vardır. Pasif çoklayıcılar sinyalde bir değişiklik yapamazken aktif çoklayıcılar sinyali kopyalayarak ağdaki problemi giderebilmektedir [11, 12].

Köprüler (Bridges): Veri bağı katmanında yer alan köprüler, aynı protokol üzerinde çalışan iki yerel alan ağı arasında bağlantı kurarak bu ağlar arasında iletişimi gerçekleştirmektedir. Köprüler, kaynak ve hedefin MAC adreslerini okuyarak filtreleme işlemi gerçekleştirebilmektedir. Tek girişli ve tek çıkışlı olmak üzere iki portlu bir cihazdır [11, 12].

Anahtarlamalı Çoklayıcılar(Switch): Veri bağı katmanında yer alan anahtarlamalı çoklayıcılar, bir ara belleğe sahip çoklu port köprüsü olarak tasarlanmıştır. Çok sayıda porttan oluşması ağ trafiğinin daha az olmasını sağlamakta ve böylece performansı artırmaktadır. Anahtarlamalı çoklayıcılar, veriyi iletmeden önce hata kontrolü gerçekleştirmektedir. Bu da iletimdeki veriyi artırmaktadır. Çoklayıcılarda olduğu gibi veriyi tüm cihazlara değil de sadece verinin gitmesi gereken adrese yönlendirmektedir. Veri sinyallerinin iletimi bir anahtarda iyi tanımlandığından, ağ performansı da buna bağlı olarak artmaktadır. Tüm bu sebeplerden dolayı günümüzde çoklayıcıların yerine anahtarlamalı çoklayıcılar tercih edilmektedir. Bu cihazların 4-5-8-16-24-26-48 adet porta sahip olan çeşitleri bulunmaktadır [11, 12].

Yönlendiriciler (Router): OSI modelinin ağ katmanında yer alan yönlendiriciler veri paketlerini IP adreslerine göre yönlendiren cihazlardır. Karmaşık ağ trafiğine sahip ağlarda etkin bir şekilde çalışmaktadır. Yönlendiriciler normal olarak LAN'ları ve WAN'ları birbirine bağlar ve veri paketlerinin yönlendirilmesiyle ilgili karar verdikleri, dinamik olarak güncellenen bir yönlendirme tablosuna sahiplerdir. Ayrıca yönlendiriciler yayın akışını sınırlama kabiliyetine de sahiplerdir. Yönlendiriciler sinyalleri güçlendirme, veri iletimini yönetme, uzak mesafelerde bulunan LAN'ları birbirine bağlayabilme özellikleri itibariyle diğer ağ cihazlarının tüm özelliklerini barındırmaktadırlar [11, 12, 13].

2.2. SDN

Bilgisayar ağlarında router, switch, firewall gibi pek çok ağ cihazı bulunmaktadır. Bu cihazların çalışmasından farklı farklı protokoller sorumludur. Gelişen teknoloji ile beraber veri merkezleri de gittikçe büyümekte ve bilgisayar ağ cihazlarının sayısı gittikçe artmaktadır. Bu sebeple ağ yönetimi daha karmaşık ve zor bir hal almaktadır. Geleneksel ağ yönetimi yaklaşımı bu durumda yetersiz kalmaya başlamıştır. Bu sebepten dolayı daha iyi bir ağ yaklaşımına ve yeni metotlara ihtiyaç duyulmaktadır. SDN bunlara çözüm olarak ortaya çıkmıştır. SDN yönetim kolaylığı, donanım bağımsızlığı, dinamik, esnek ve ölçeklenebilir ağ mimarisini temin eder. Dolayısıyla bu geniş ve karmaşık ağ yönetimine etkili bir çözüm sunar [14].

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

2.2.1. SDN Nedir?

Ağ konfigürasyonu ve kurulumu, pek çok ağ elemanının konfigürasyonu konusunda yetenekli personel gerektirmektedir. Ağ düğümleri (anahtarlar, yönlendiriciler, vb.) arasındaki etkileşimlerin karmaşık olduğu durumlarda daha sistem tabanlı bir yaklaşım gereklidir. Günümüzün ağ donanımlarının çoğundaki mevcut programlama arabirimleri ile bunu gerçekleştirmek zordur. Bunu sağlamak için yeni bir ağ modeli gerekir ve bu da SDN'dir.

Açık Ağ Oluşturma Vakfı (ONF) SDN' nin geliştirilmesine, standardizasyonuna ve ticarileştirilmesine ayrılmış kâr amacı gütmeyen bir kuruluştur. ONF, SDN' i en iyi açıklayan tanımı şu şekilde yapmaktadır: SDN, ağ denetiminin iletimden ayrıldığı ve doğrudan programlanabilen yeni bir ağ mimarisidir. Bu tanıma göre SDN, kontrol ve veri düzlemlerinin ayrılması ve kontrol düzlemindeki programlanabilirlik olmak üzere iki özellik ile öne çıkmaktadır. Bununla birlikte, SDN' nin bu iki özelliğinden hiçbiri, aşağıda ayrıntılı olarak açıklandığı üzere, ağ mimarisinde tamamen yeni değildir. SDN temel olarak dört özelliğe odaklanmaktadır:

- Kontrol düzleminin veri düzleminden ayrılması
- Merkezi bir denetleyici ve ağın görünümü
- Kontrol düzlemindeki (denetleyiciler) ve veri düzlemindekiler arasındaki açık ara yüzler
- Ağın harici uygulamalarla programlanabilirliği [15, 16]

2.2.2. SDN Kavramının Gelişimi

SDN terimi son yıllarda ortaya çıkmıştır. SDN mimarisinin tarihsel geçmişi, statik ağların bazı dezavantajlarının iyileştirilmesine dayanmaktadır. Columbia Üniversitesi'nde kurulan "Açık Sinyal Çalışma Grubu (OPENSIG)", 1995 yılında ATM, internet ve mobil ağlarda açık ağ kontrol konularında araştırma yapmaya başlamıştır. Açık programlanabilir ağların tanımlanmasına, uygulanmasına ve denenmesine odaklanmışlardır. Bu araştırmada, araştırmacılar, ağ aygıtlarındaki donanım ve kontrol yazılımının daha güçlü, esnek ve programlanabilir ağ yönetim sistemleri sağlamak için ayrılmış olması gerektiğini belirtmişlerdir. Maalesef, hiyerarşik olarak yerleştirilmiş router-switch mimarisindeki uygulama zorlukları nedeniyle çabaları başarısız olmuştur.

OPENSIG araştırmaları bir dizi yeni araştırmaya sebep olmuştur. IETF çalışma grubu ATM veya MPLS anahtarının kontrol durumunu oluşturmak ve sürdürmek için "Genel Anahtar Yönetim Protokolü (GMSP)" önermiştir. Son sürüm olan GMSPv3, geçiş sistemi kaynakları ve QoS özelliklerini kontrol etmek için hem tek noktaya yayın hem de çok noktaya yayın anahtarı bağlantısı kurabilmektedir.

Programlanabilir anahtar yaklaşımı temel alınarak Teannenhouse tarafından önerilen "Aktif Ağ Yapısı" önerisi ise mevcut paket / hücre formatını korumaktadır. Program seçimini destekleyen ve kullanıcılar yerine ağ yöneticileri tarafından işlevleri yürüten ayrı bir mekanizma sağlamaktadır.

- Greenberg tarafından önerilen 4D protokolünde de ağ kontrolünün işlevleri dört düzleme parçalanmaktadır: Bir ağ yapılandırması oluşturmaktan sorumlu bir karar katmanı
- Ağ durumuyla ilgili bilgileri karar katmanında toplayan ve karar katmanı çıktılarını yönlendiricilere dağıtan bir yaygınlaştırma katmanı
- Cihazların keşfedilmesini sağlayan bir keşif katmanı
- Ağ trafiğini yönlendirmek için bir veri katmanı

Stanford Üniversitesi'nde bir çalışma grubu tarafından geliştirilen " Sane/Ethane Projeleri", Openflow protokolünün ilk versiyonu olarak düşünülebilir. OpenFlow, Akış Tablosundaki girdilerin anahtarın dışındaki bir sunucu tarafından tanımlanmasını sağlar. Sane, kurumsal ağlar

için temiz ve güvenli bir koruma mimarisidir. Ağdaki tüm bağlantıyı yöneten tek bir koruma katmanı tanımlar. Tüm yönlendirme ve erişim kontrolü kararları, erişim denetimi ilkelerine göre yetenekleri dağıtarak hizmetlere erişim izni veren merkezi bir sunucu tarafından yapılmaktadır. Ethane, ağ yöneticilerinin, tüm ağ aygıtlarına uygulanacak bir ağ genelinde ilke tanımlamalarını sağlamaktadır. Ethane tasarımı, merkezi bir denetleyici tarafından gerçekleştirilen SDN' den bazı farklılıklar (adlandırma, ilke beyanı ve güvenlik kontrolleri gibi) içermektedir. SDN, bu uygulamalardan herhangi biriyle sınırlı değildir. SDN bu uygulamaların temelinde var olan platformdur.

Stanford ve California üniversiteleri arasındaki 5 yıllık araştırma iş birliği, Openflow protokolünün ilk sürümünün piyasaya sürülmesi ve 2011'de "Açık Ağ Oluşturma Kuruluşu" nun kurulması ile sonuçlanmıştır [17].

2.2.3. SDN Mimarisi

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

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

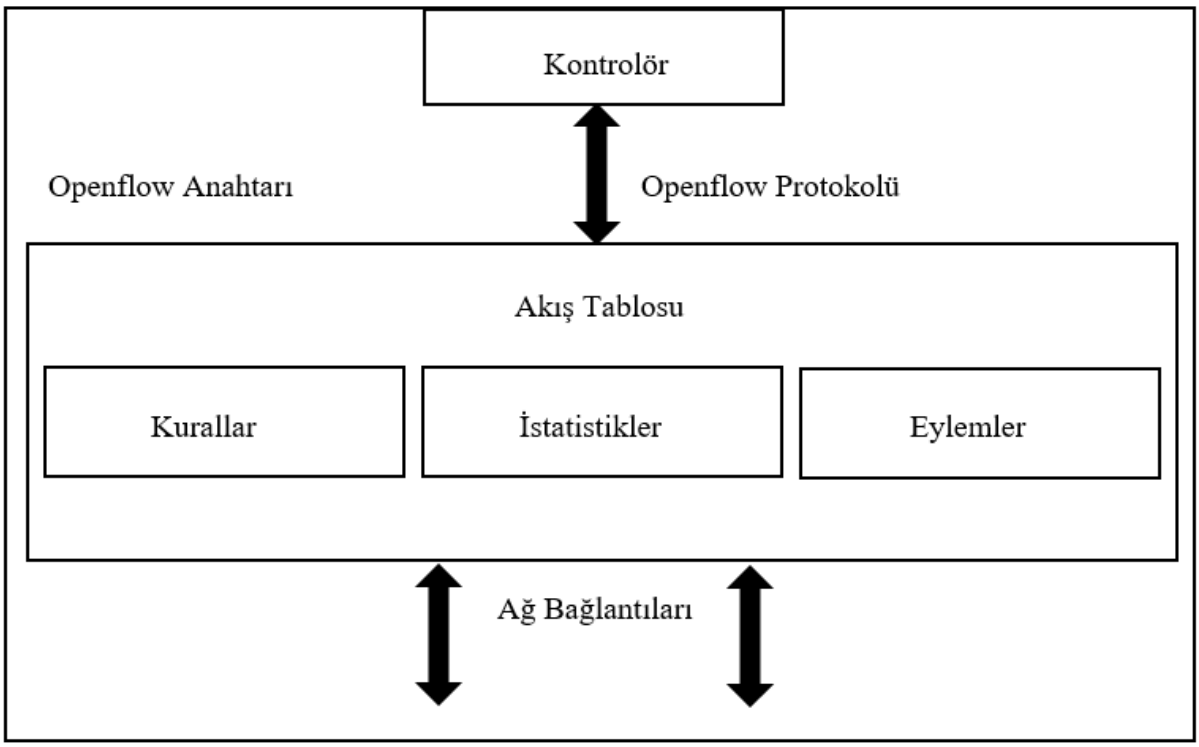
2.2.3.1. OpenFlow Anahtarı

SDN mimarisi temelde Openflow protokolüne bağlıdır. Geleneksel bir switch veya router hem veri hem de denetim yolu işlemini gerçekleştirmektedir. Openflow mimarisinde ağ aygıtı,

işlenmiş paketlerin gerekli yönlendirme bilgilerini içeren bir veya daha fazla akış tablosu içermektedir. Akış girişleri tipik olarak aşağıdaki verileri içerir:

- Eşleşme alanları paket başlığı, giriş limanı ve meta-data gibi bilgileri tutar.
- Sayaçlar akış süresi, paket sayısı ve alınan bayt gibi akış istatistiklerini toplar.
- İşlem setleri, eşleşen paketlerin sil, ilet veya değiştir gibi işlemlerini nasıl ele alacağını tanımlar.

Anahtarın üzerine bir paket geldiğinde, eşleme alanları akış girişi ile karşılaştırılır. Bir eşleşme bulunursa, önceden tanımlanmış işlem yapılır. Hiçbir eşleşme yoksa anahtar, paketin bırakılması ya da denetleyiciye gönderilmesi için ikinci kayıp akış tablosuna bakar. Denetleyici ve anahtar, bir ileti dizisi kullanarak güvenli bir kanal üzerinden iletişim kurar. Denetleyici, anahtarın akış girişlerini silebilen, ekleyebilen veya güncelleyebilen uzak bir cihazdır.



Şekil 2.3: Openflow Anahtarı Çalışma Akış Şeması

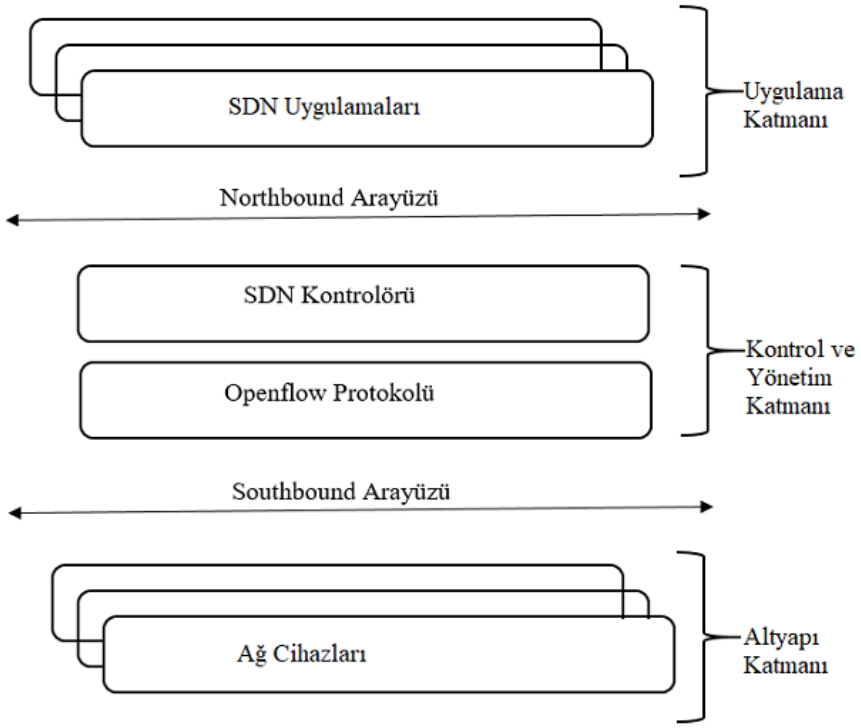
2.2.3.2. Denetleyici Çalışması

SDN'nin kontrol ve yönetim işlemleri merkezi bir denetleyici tarafından gerçekleştirilmektedir. Denetleyici tam ağ topolojisine ve adres bilgisine sahiptir. Bir switch, akış tablolarında olmayan bir paket için denetleyiciye paket gönderim isteği yolladığında, denetleyici bu istekleri alır. Bu modda, denetleyici trafiği dinler, eylemleri alır veya talep üzerine rotaları tanımlar. Openflow denetleyicileri ağ operatörleri tarafından bir kontrol arabirimi ile programlanır.

Şekil 2.4'de gösterildiği gibi, daha büyük ağlar, daha fazla SDN denetleyicisi, yani SDN uygulama katmanı içerecek şekilde yüksek düzeyli bir karar ilkesi katmanı gerektirir. Bu katman, ağ sanallaştırması, trafik mühendisliği, yönlendirme izleme ve hizmet kalitesi servislerinden sorumludur.

Uygulama katmanı ve SDN denetleyiciler arasındaki iletişim, "North Bound" olarak adlandırılan bir dizi uygulama programlama arayüzü (API) tarafından gerçekleştirilir. Benzer şekilde SDN denetleyicisi ve Openflow ağ aygıtları arasındaki arabirime de "South Bound" denmektedir.

SDN mimarisi, karmaşık ağların birleşik ve global görünümünü sağlar ve bu katmanlar arasındaki etkileşimler yoluyla trafik akışları üzerinden ağ yönetimi için güçlü bir kontrol platformu sağlar.



Şekil 2.4: SDN Katmanları

2.2.3.3. Southbound Arayüz Ayarı

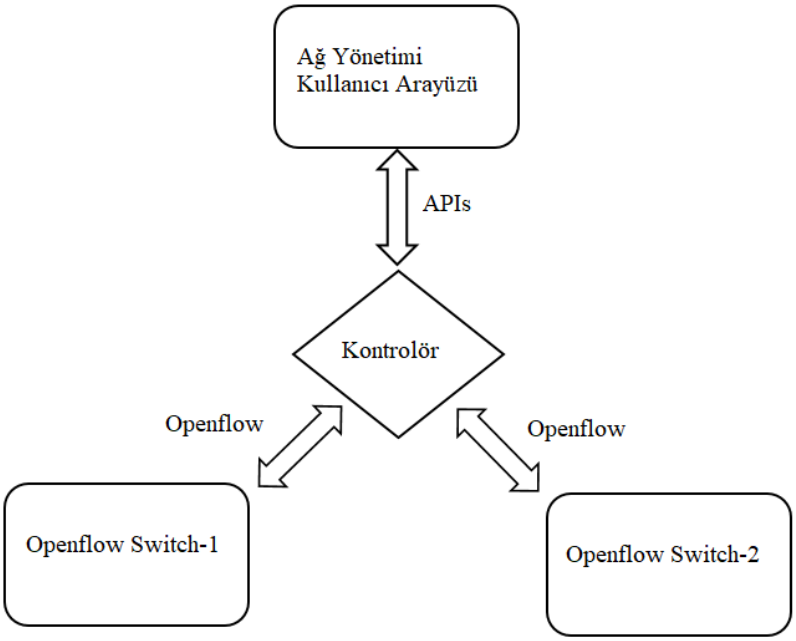
Denetleyici-Switch iletişimi southbound arayüzü ile kontrol edilmektedir. Anahtar ve denetleyici, standart bir ileti kümesi kullanarak TLS bağlantısı üzerinden iletişim kurar. TLS bağlantısı, başlatma anahtarı tarafından varsayılan olarak TCP bağlantı noktası 6653'te bulunan denetleyiciye başlatılır. Anahtar ve denetleyici, site özel bir anahtar tarafından imzalanmış sertifikaları değiştirerek karşılıklı olarak kimlik doğrulamasını yapar. Southbound arayüzü, belirli bir denetleyiciye uygulanan çözüm dikkate alınmaksızın standartlaştırılmıştır ve Openflow en yaygın kullanılan standarttır.

2.2.3.4. Northbound Arayüz Ayarı

Northbound arabirimi, Openflow anahtar parametrelerine hâkim olmak için web tabanlı bir GUI tarafından kontrol edilen bir dizi API' dir. " Open Network Foundation" nın "Northbound Interface Working Group" adlı bir çalışma grubuna sahip olmasına rağmen, bu alanda yeterli standardizasyon sağlanamamıştır.

Açık kaynak olarak piyasaya sürülen ve sürekli gelişen üç önemli northbound arayüzü vardır: Project Floodlight, OpenDaylight ve NOX-POX denetleyicilerinin ikilisi.

Northbound API' leri "Yönetim ve Yönetim Uygulaması" olarak adlandırılır. Bu API 'lere kullanıcı dostu "Grafiksel Kullanıcı Arayüzü (GUI)" ile erişilir [14, 17].



Şekil 2.5: SDN Veri Akış Şeması

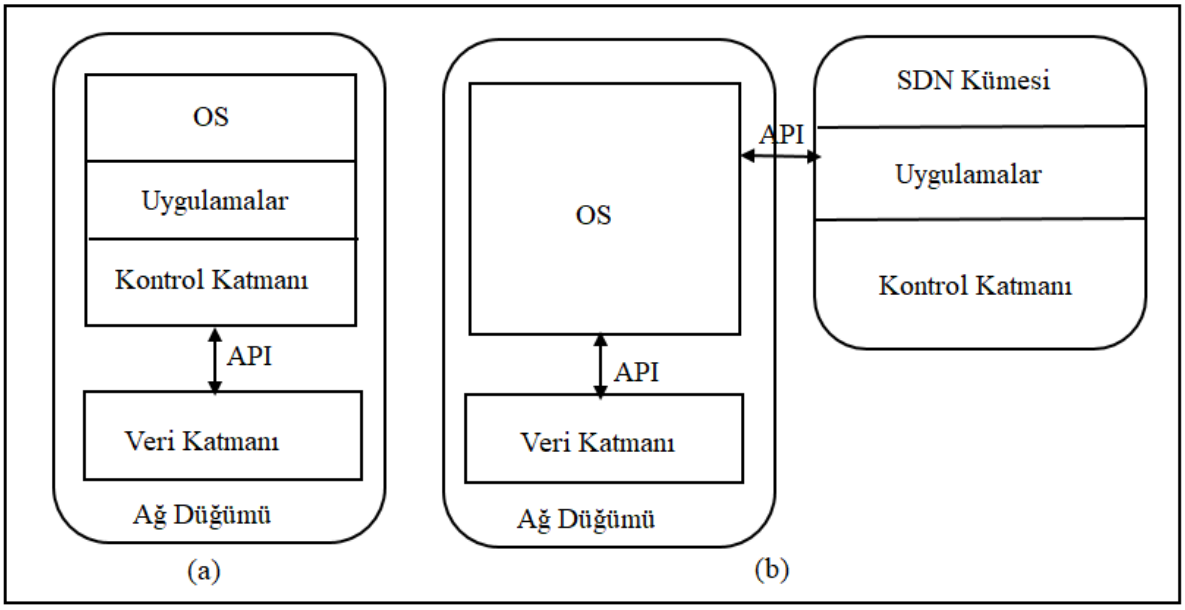
2.2.4. Neden SDN Tercih Edilmeli?

İletişim ağının temel amacı, bilgileri bir noktadan diğerine aktarmaktır. Ağ içinde birden çok düğüme veri dolaşır ve SDN ağ uygulamaları / hizmetleri tarafından sağlanan kontrolle, etkin ve verimli veri aktarımı sağlanır.

SDN mimarisi güvenlik açısından bazı faydaları beraberinde getirmiştir. SDN programlamayı ve ağ üzerinde merkezileştirilmiş kontrolü sağlar. SDN in bu özellikleri güvenlik tehditlerine karşı avantaj oluşturmaktadır. Örneğin ağda bir anomali gerçekleştiği tespit edildiğinde, ilişkili

trafik analiz edilerek denetleyicilere yollanabilir. Saldırıları önlemek için analiz sürecinden sonra var olan kurallar güncellenebilir veya yeni kurallar oluşturulabilir.

SDN ağlarda yaşanan yoğun ve çeşitliliğe sebep olan trafiğin oluşturduğu karmaşanın önüne geçerek, daha basit ve yönetimi daha kolay olan bir anlayış sunmaktadır. Böylece şirketler SDN teknolojilerinin sağladığı bu esnek yapıyı ihtiyaçlarına uygun özelleştirebilmektedirler. SDN, programlanabilir ağ hizmetleri sağlayarak, işletme ve operatör ağlarını yönetmek için toplam maliyeti düşürmekle birlikte, ağ kurulumunu ve operasyonunu basitleştirmek açısından da büyük bir vaatte bulunmaktadır [15].



Şekil 2.6: Geleneksel Ağ (a) ve SDN Ağı(b)

2.2.5. SDN Kaynaklı Problemler

SDN günümüzde pek çok kolaylık sunmakla beraber bazı problemlere de sebep olmaktadır. Bu problemler SDN in dört temel eksikliğinden meydana gelmektedir: performans-esneklik, ölçeklenebilirlik, güvenlik ve birlikte işlerlik

2.2.5.1. Performans-Esneklik

Performans hem verim hem de gecikme göz önüne alındığında, özellikle ağ düğümünün işlem hızını belirtir. Esneklik ise sistemleri öngörülemeyen yeni özellikleri destekleyecek şekilde yeniden uyarlamaktır

Mevcut ağ teknolojilerinin SDN hedeflerine uygun bir şekilde programlanabilir olmasını sağlamak için çeşitli girişimler vardır. Bunların ötesinde, SDN programlanabilirlik ve performans sorunu, 100 GB / sn.' nin ötesinde düğüm bant genişliği elde etmek için hala bir mücadele içerisindedir.

Veri işleme teknolojilerinin programlanabilirlik / performans dengesi göz önüne alındığında, SDN için yalnızca karma bir yaklaşımın etkin bir teknoloji çözümü sağlayacağı açıktır. Temel SDN düğüm işlevleri, güç dağılımı, maliyet ve ölçeklenebilirlik bakımından programlanabilirlik dengelemesine karşı en iyi performansı sağlamak için özelliklere has teknolojiler kullanılmak üzere alt işlev kümelerine ayrılabilir.

SDN belirli bir standardın üzerine oturtulduğu için özelliğe has ve işletmelere has hale getirilmesi konusunda bir performans-esneklik sıkıntısı oluşmaktadır. SDN standart donanım üzerine kurulmuş programlanabilir bir arayüz ile çok satıcılı bir ağ kurabilmeye çalışmaktadır. Böylece bu sorunu ortadan kaldırmayı hedefler.

2.2.5.2. Ölçeklenebilirlik

Bu sorun denetleyici ölçeklenebilirliği ve ağ düğümü ölçeklenebilirliği olarak ikiye ayrılabilir. Burada odaklanılan kısım denetleyici ölçeklenebilirliğidir. Bu sorunda kendi içerisinde üç kısma ayrılır. Birinci sorun, çoklu düğümler ve tek bir denetleyici arasında ağ bilgisini değiştirerek getirilen gecikmedir. İkinci sorun, SDN denetleyicilerinin doğu ve batı yönündeki API' leri kullanarak diğer denetleyicilerle nasıl iletişim kurduklarıdır. Üçüncü zorluk ise, denetleyici arka uç veri tabanının boyutu ve çalışmasıdır. Bizi asıl ilgilendiren kısım güvenlik ile ilgili kısıtlar olduğu için bu sorunların içeriğinden bahsetmeyeceğim. SDN için tam ölçeklenebilirliğe ulaşma yolunda, ağ programlanabilirliğine evrimsel bir yaklaşım gerekmektedir. Örneğin, hibrid mimari ile CPU'nun düğümünde bir dizi sorgu çözülebilir, aksi takdirde işlem denetleyiciye aktarılır. Bu, denetleyicideki veri tabanı boyutunu potansiyel olarak düşürebilir ve aynı anda denetleyici ile düğümleri arasındaki iletişimi azaltır.

2.2.5.3. Güvenlik

SDN' in arayüzünü ve tabakalarını hedef alan çeşitli güvenlik tehditleri vardır. SDN' deki güvenlik tehditleri yedi farklı tehdit vektörü olarak sınıflandırılmaktadır. Bunlardan üçü SDN' e özel tehditlerdir. "DoS, unauthorized access, data leakage, data modification, malicious applications" gibi tehditler hem diğer ağ mimarilerinde hem de SDN' de görülmektedir. SDN' e özel olan saldırılar yazılım denetleyicisini, kontrol katmanı ile veri katmanı arasındaki iletişimi ve kontrol katmanı ile uygulama katmanı arasındaki iletişimi hedef alır.

DoS ve DDos saldırıları veri katmanındaki switchlere karşı saldırır ki bu da kullanılabilirlik ve güvenlik hizmeti servisini hedef alır. Merkezi denetleyici SDN 'in temel özelliklerinden birisidir. Bu özellik ağ üzerinde genel olarak kullanılmasına rağmen, SDN mimarisi güvenliğinde bazı zayıflıklara da sebep olmaktadır. Sahte trafik akışlarının OpenFlow anahtarlarına ve denetleyici kaynaklarına DoS saldırıları yapmak için kullanılabileceği tespit edilmiştir. Saldırgan denetleyiciye çok fazla paket göndererek DoS saldırılarına maruz kalabilir. Benzer olarak, birden fazla saldırgan switchlere sistematik yol içinde büyük miktarda paket yollayabilir. Anahtarların akış tablolarında kuralların tümü bulunmadığından, denetleyiciye çok sayıda sorunun gönderilmesine neden olur. Bu durumda, denetleyici DDoS saldırısına maruz kalacak ve meşru isteklere yanıt veremeyecektir. Denetleyicideki DoS ve DDoS saldırıları, tüm ağın işleyişini olumsuz bir şekilde etkileyebilir. Benzer olarak DoS saldırısı veri katmanında bulunan switchlerde de mümkün olabilir.

Denetleyicinin ele geçirilmesi veya yetkisiz denetleyiciye erişim saldırıları, güvenliğin gizlilik ilkesini hedefler. Denetleyicideki güvenlik açıklarının, tüm ağı tehlikeye atabilecek sonuçları olabilir. Saldırgan yanlış yapılandırılmış, savunmasız denetleyicinin yönetimini ve ağ yönetimini üstlenebilir. Ardından, saldırgan, veri düzlemindeki anahtarları, denetleyiciye gelen trafiğin diğer hedeflere saldırı başlatmak için kullanmasına izin verecek şekilde programlayabilir. Kontrol düzlemi ve uygulama düzlemi arasındaki iletişimi sağlamak için mekanizmalar olmaması durumunda, kötü amaçlı uygulamalar anahtarların akış tablosuna hileli kurallar ekleyebilir. Bu, ağdaki çatışma kurallarına neden olacaktır. Bu nedenle, denetleyiciler ve uygulamalar arasında yetkilendirme ve kimlik doğrulama mekanizmaları oluşturarak güvenilir bağlantı kurulmalıdır.

MITM (Man-in-the-middle) saldırısı denetim katmanı ve veri katmanı iletişimi arasında oluşur, bütünlük ve gizlilik ilkesini hedef alır. Bu saldırı tipinden hem kontrol düzlemi hem de veri düzlemi etkilenir. SDN mimarisinin getirdiği ayrık katmanlardaki iletişimde, şifrelenmemiş protokolleri kullanmak ciddi sonuçlar doğurabilir. MITM saldırısı OSI referans modelinin ikinci katmanında yer alır. MITM, ağdaki sunucular, yönlendirici veya anahtar gibi ağ kaynakları ile ağdaki uç nokta arasındaki trafik akışını dinlemeyi veya değiştirmeyi sağlar. Bu durumda, saldırgan, anahtarlardaki akışları değiştirebilir veya yeni akış kuralları ekleyebilir. İletişim kanalı, şifreleme protokolü olan TLS' yi kullanarak daha güvenli hale getirilebilir. Openflow protokolü varsayılan olarak TLS bağlantısını destekler. Karşılıklı kimlik doğrulama, ağ paketlerinin iletimi için denetleyiciler ve anahtarlar arasında sertifikalar değiştirilerek yapılabilir. Şifreli protokoller kullanıldığından saldırgan mesajların içeriğini görüntüleyemez veya değiştiremez.

Yetkisiz ve kimliği belirsiz uygulamalar, güvenliğin gizlilik ve bütünlük ilkesini hedef almaktadır. Uygulama katmanında çalışan birçok üçüncü parti uygulama vardır. Denetleyici, SDN uygulamaları için soyutlama sağlar ve bu, uygulamaların ağ durumunu okumasını ve yazmasını sağlar. Bu durum, ağın kontrolü için bir sorun teşkil eder. Saldırgan, yetkisiz erişime neden olan uygulamaları kendisini gizlemek ve ağ kaynaklarına erişmek ve ağın çalışmasını değiştirmek için kullanabilir. Denetleyiciye doğrudan bağlı olan yönetimsel bilgisayarlar ağa giriş noktasına erişebilir. Bu bilgisayarlarda bir güvenlik açığı varsa, saldırgan bu güvenlik açıklarını bilgisayarların denetimini ele geçirerek denetleyiciye kolayca erişmek için kullanabilir [19].

Diğer tehdit vektörleri ise şu şekilde sınıflandırılmaktadır:

- Bir hatayı zamanında tespit etmede başarısızlık
- Ağ sorunları sırasında ağın güvenilir bir kurtarma noktası edinme başarısızlığı

2.2.5.4. Birlikte İşlerlik

Ağdaki tüm öğeler ve aygıtlar SDN ile etkinleştirilmiş olsaydı, SDN teknolojisine dayalı tamamen yeni bir altyapının kurulması basit olacaktı. Bununla birlikte, bugün hayati sistemleri

ve işletmeleri destekleyen çok geniş bir ağ tabanı var. Bu ağları yeni altyapıya basitçe "takas etmek" mümkün olmayacak.

SDN' e geçiş için SDN' nin ve eski ekipmanın aynı anda desteklenmesi gerekmektedir. Bu da var olan karmaşayı daha da artıracaktır. Bu aşamada SDN' nin elverişli bir ağ altyapısı oluşturduğunu söyleyemeyiz. SDN konusunda potansiyeli artırmak için SDN' in kısıtlarına daha fazla çaba gösterilmelidir [15, 16].

2.3. OPENFLOW

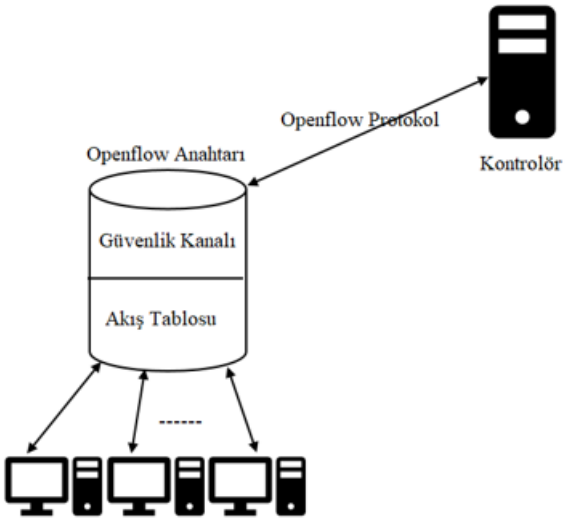
2.3.1. Openflow Nedir?

Open Networking Forum tarafından geliştirilen OpenFlow, SDN uygulamalarındaki yüksek performansından dolayı en çok tercih edilen protokoldür. OpenFlow, bir sunucunun paketleri nereye göndereceğini ağ anahtarlarına bildirmeye imkân tanımaktadır. Geleneksel ağ ile uyum sağlayan OpenFlow, ağ operatörlerine esneklik kazandırarak paket iletiminin kontrol edilmesini mümkün kılar. Kullanıcı, OpenFlow protokolü ile esneklik kazanan ağ operatörü üzerinde kendi tercihlerine göre yazılım yapma imkanına kavuşmuştur.

Geleneksel ağlarda, paket yönlendirmesi (veri yolu) ve üst seviye yönlendirme (kontrol yolu) aynı cihazda gerçekleşirken, OpenFlow protokolü kullanılan ağda veri yolu ve kontrol yolu birbirinden ayrılmaktadır. Veri yolu bölümü OpenFlow anahtarının kendisinde bulunur; ayrı bir denetleyici ise, üst düzey yönlendirme kararlarını verir. Anahtar ve denetleyici arasındaki iletişim ise OpenFlow protokolü aracılığıyla sağlanır. Yazılım tanımlı ağ (SDN) olarak bilinen bu yöntem, ağ kaynaklarının geleneksel ağlarda olduğundan daha etkili bir şekilde kullanılmasını sağlamaktadır [20, 21].

2.3.2. Openflow Mimarisi

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



Şekil 2.7: Openflow Temel Bileşenleri

2.3.2.1. Openflow Anahtarı

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

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

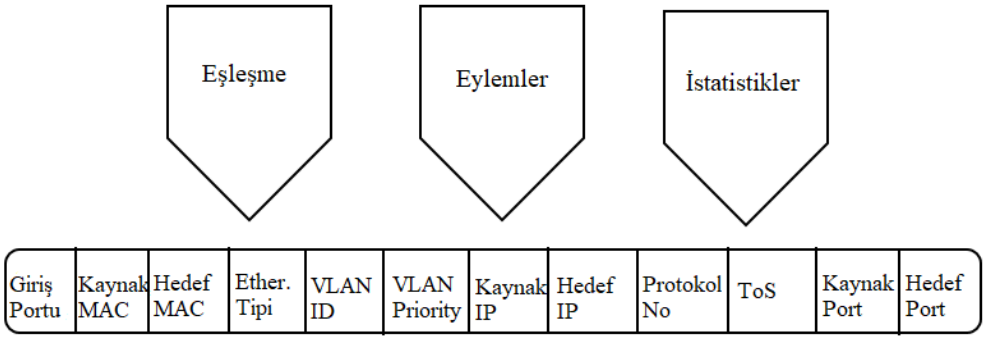
Veri Katmanı: Anahtar giriş portlarına gelen paketlerin yani flowların tutulduğu katmandır. Kontrol katmanından aldığı bilgilere göre flowların iletilmesinden sorumludur.

Yönetim Katmanı: Anahtar üzerindeki yapılandırma ve yönetim işlemleri bu katmanda gerçekleşmektedir [2, 22].

Mevcut ağ anahtarlarında bu üç katman tek bir cihaz üzerinde bulunarak paketlerin iletilmesinden sorumludur. Bu da anahtarlar üzerinde bir sorunla karşılaşıldığında gerekli çözümün uygulanmasını zorlaştırmaktadır. Anahtarın tamamında karmaşık ayarlar yapılmasına hatta bazı durumlarda problemin giderilememesine sebep olmaktadır. OpenFlow anahtarı bu soruna çözüm olarak ortaya çıkmıştır. Openflow anahtarlarında sadece veri katmanı yani flow

iletiminden sorumlu katman bulunmaktadır. Kontrol ve yönetim katmanı anahtarın dışına çıkarılarak, flow iletiminde karar alırken anahtara esneklik kazandırılmıştır. Normalde tek bir cihaz üzerinde gerçekleşen flow iletimi openflow anahtarı, openflow denetleyicisi ve anahtar-denetleyici arası iletişimi sağlayan openflow protokolü olmak üzere üç kısımda gerçekleşmektedir.

OpenFlow anahtarının en temel bileşenleri ise flowlar ve flow tablolarıdır.



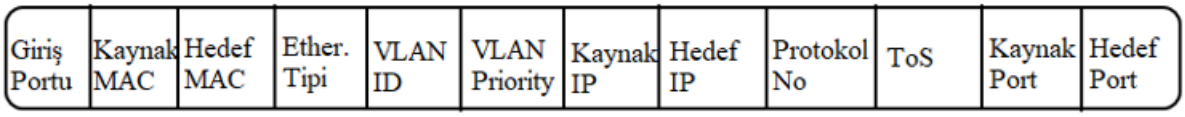
Şekil 2.8: Flow Tablosu ve Bileşenleri

Bir openflow denetleyicisinden bir openflow anahtarına iletilen openflow talimatları, paketleri flow olarak adlandırılmaktadır. Her bir flow, paket eşleme alanları, akış önceliği, çeşitli sayaçlar, paket işleme talimatları, akış zaman aşımları ve çerez kısımlarından oluşmaktadır. Flowlar tablolarda düzenlenmektedir. Flow tabloları openflow anahtarının veri katmanı içerisinde bulunmaktadır. Flow tabloları kurallar, eylemler ve sayaç(istatistik) olmak üzere üç temel kısımdan oluşmaktadır [22, 23].

Kurallar	Eylemler	Sayaç

Şekil 2.9: Flow Tablosunun Genel Kısımları

Kurallar: Flow paketinin kontrol edilmesine olanak sağlayan bilgilerin ve paket başlığının bulunduğu kısımdır. Her bir paket, herhangi bir değerle eşleşen belirli bir değeri veya eşleşme yoksa ANY değerini içerir. Anahtar, IP kaynağı veya hedef alanlar üzerindeki alt ağ maskelerini destekliyorsa, daha doğru sonuçlar veren eşleşmeler belirlenebilir [2, 24].



Şekil 2.10: Akış Girişlerine Karşı Eşleştirme İçin Kullanılan Paketlerden Alanlar

Eylemler: Flow paketinin nasıl işleneceğinin belirlendiği kısımdır. Paketler üzerinde farklı işlemler uygulanabilir. Bir paket düşürülebilir, iletilebilir veya paketin içeriğinde değişiklik yapılabilir. Her bir akış girişi, anahtarın eşleşen paketleri nasıl işleyeceğini belirleyen sıfır veya daha fazla eylemle ilişkilendirilir. Eğer ileriye yönelik bir eylem yoksa paket düşürülür. Eklenen akış girdileri için eylem listeleri, belirtilen akış sırasına göre işlenmelidir. Anahtar, bir eylemin listelenen sırada işlenememesi durumunda akış girişini reddedebilir; bu durumda, anahtar hemen desteklenmeyen bir akış hatasına dönmelidir. İki türlü eylem tipi vardır. Bunlar "Required Action" ve "Optional Action"'dır. Bir openflow anahtarı bu iki türü birden desteklemek zorunda değildir. Bu aşamada da şunu söyleyebiliriz. İki türlü openflow uyumlu anahtar vardır. Bunlardan openflow-only yalnızca required-actionları desteklerken, openflow-enabled anahtarları ise optional-actionları desteklemektedir.

- *Required Action:* OpenFlow anahtarlarının paketi fiziksel portlara ve *ALL, CONTROLLER, LOCAL, TABLE, IN_PORT* gibi sanal portlara yönlendirmesini sağlayan eylemlerdir.
- *Optional Action:* OpenFlow anahtarı isteğe bağlı olarak *NORMAL* ve *FLOOD* gibi sanal portları destekleyebilir [2].

Sayaç: Flow paketleri ile ilgili istatistiksel verilerin tutulduğu kısımdır. Bu veriler flow paketinin takip numarası, zaman aşımı bilgisi veya paketin boyutu olabilir. Sayaçlar, tablo başına, akış başına, port başına ve sıra başına tutulmaktadır [25].

Flow Tablolarında Eşleşme Nasıl Gerçekleşir?

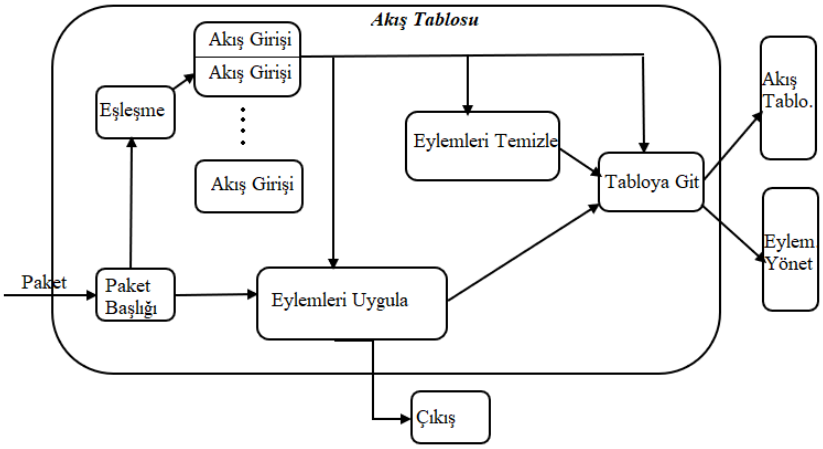
Bir paket alındığında, Openflow Anahtarı, ilk olarak akış tablosunda tablo arama yaparak işe başlar ve pipeline işlemi temel alınarak, diğer akış tablolarında tablo aramaları gerçekleştirilir.

Paket eşleşme alanları paketten çıkarılır. Tablo aramaları için kullanılan paket eşleme alanları paket türüne bağlıdır ve genellikle Ethernet kaynak adresi veya IPv4 hedef adresi gibi çeşitli

paket başlık alanlarını içerir. Paket başlığına ek olarak, eşleşmeler giriş portuna, meta veri alanına ve diğer pipeline alanlarına karşı da yapılabilmektedir. Meta veriler, bir anahtar içindeki tablolar arasında bilgi aktarmak için kullanılabilir. Paket eşleme alanları, paketin geçerli durumda olduğunu gösterir; eğer önceki tabloda uygulanmış olan eylemler *Apply-Actions* talimatını kullanarak paket başlıklarını değiştirdiğinde, bu değişiklikler paket eşleme alanlarına yansıtılmaktadır.

Bir paketin, arama için kullanılan paket eşleme alanlarındaki değerleri, akış tablosu girdisinde tanımlananlarla eşleşiyorsa bu paket akış tablosu girişi ile eşleşiyor demektir. Akış tablosu girişi alanı *ANY* değerine sahipse (alan atlanır), başlıktaki olası tüm değerlerle eşleşme gerçekleşir.

Paket tabloya karşı eşleştirilir ve yalnızca paketle eşleşen en yüksek önceliğe sahip akış girişi seçilmelidir. Tam eşlemeyi belirten bir giriş her zaman en yüksek önceliğe sahiptir. Seçili akış girişi ile ilişkili sayaçlar güncellenmeli ve seçilen akış girişinde bulunan yönerge kümesi uygulanmalıdır. Aynı en yüksek önceliğe sahip eşleşen birden fazla akış girişi varsa, seçili akış girişi açıkça tanımlanmamış demektir. Bu durumda anahtar herhangi bir paketi eşlemede serbesttir. Bir paket için eşleşen bir giriş bulunamazsa, paket güvenli kanal üzerinden denetleyiciye gönderilir [21, 26].



Şekil 2.11: OpenFlow Eşleşme Aşamaları

2.3.2.2. Denetleyici

Yazılım tanımlı ağ oluşturmak için gereken en önemli elemanlardan birisi de denetleyiciler yani denetleyicilerdir. SDN denetleyicileri ağın beynini oluşturmaktadır. Denetleyiciler ağın akıllı olmasını sağlamak için akış denetimini yönetmektedirler. Şöyle ki; denetleyici denetim adına flow tablolarından bir akışı kaldırabilir veya tabloya yeni bir akış ekleyebilir. Bir uçtaki ağ aygıtları ile diğer uçtaki uygulamalar arasında uzanır. Uygulamalar ve cihazlar arasındaki iletişim, denetleyiciden geçmek zorundadır. Denetleyici ayrıca, ağ aygıtlarını yapılandırmak ve uygulama trafiği için en uygun ağ yolunu seçmek için protokolleri kullanır. Denetleyiciler protokolleri temel alarak işlem yaparlar. Protokoller sayesinde anahtarlar ve denetleyici arasında iletişim sağlanır. Yazılım tanımlı ağlarda yaygın olarak kullanılan protokoller *OpenFlow* ve *OVSDB*'dir. OpenFlow protokolüne bağlı olarak çalışan denetleyicileri ise şu şekilde belirtebiliriz: NOX, POX, Beacon, Floodlight, Maestro, NodeFlow, Trema, OpenDaylight

NOX: Pek çok araştırma makalesinde kullanılan NOX denetleyicisi, Stanford Üniversitesi tarafından geliştirilmiş olup, GNU (General Public License) kapsamında lisanslanmıştır. C++ diliyle uyumludur.

POX: NOX denetleyicisinin Phyton'da yeniden yazılmış halidir. Çeşitli platformlarda kullanıma uygundur. APL (Apache Public License) kapsamında lisanslanmıştır.

Beacon: Stanford Üniversitesi tarafından geliştirilmiş olup, java diliyle uyumludur.

Floodlight: Big Switch Networks sponsorluğu ile Beacon'un bir uzantısı olarak üretilmiştir. APL kapsamında lisanslanmıştır. Java diliyle uyumludur.

Maestro: Rice Üniversitesi tarafından geliştirilmiş olup, java diliyle uyumludur. LGPL (Lesser General Public License) kapsamında lisanslanmıştır.

NodeFlow: Java ile uyumlu olan bu denetleyici Node JS kütüphanesine dayanmaktadır. MIT kapsamında lisanslanmıştır.

Trema: NEC tarafından geliştirilmiş olup, C ve Ruby dilleriyle uyumludur. GPL kapsamında lisanslanmıştır.

OpenDaylight: Java diliyle uyumlu olan bu denetleyici, Linux kuruluşu tarafından barındırılmaktadır. Linux'a bağlı değildir, yani işletim sistemi kısıtlaması yoktur. EPL (Eclipse Public License) kapsamında lisanslanmıştır [20, 21].

SDN Kontrolörü	***Kullandığı Programlama Dili***
NOX	C++
Beacon, Floodlight, OpenDaylight, Maestro	Java
POX	Python
NodeFlow	JavaScript
Trema	C

Şekil 2.12: SDN Denetleyicileri ve Kullandıkları Programlama Dilleri

2.3.2.3. Openflow Protokolü

OpenFlow protokolü, her OpenFlow anahtarını bir denetleyiciye bağlayan arabirimdir. Bu ara yüz aracılığıyla, denetleyici anahtarı yapılandırır ve yönetir, anahtardan olayları alır ve anahtardan paket gönderir. Yani bu protokol sayesinde anahtar ve denetleyici arasında iletişim ağı kurulmaktadır. OpenFlow protokolü genellikle TLS kullanılarak şifrelenmekte ve doğrudan TCP üzerinden çalıştırılmaktadır.

OpenFlow protokolü; denetleyiciden anahtara, eş zamansız ve simetrik olmak üzere üç farklı mesaj iletim türünü desteklemektedir. Denetleyiciden anahtara ileti, denetleyici tarafından başlatılır ve anahtarın durumunu doğrudan yönetmek veya incelemek için kullanılır. Eşzamansız ileti anahtar tarafından başlatılır. Şebeke olaylarının denetleyicisini güncellemek için kullanılır ve anahtarın durumunu değiştirir. Simetrik ileti, anahtar veya denetleyici tarafından başlatılır ve talep edilmeksizin gönderilir.

Denetleyiciden Anahtara

Denetleyici / anahtar mesajları, denetleyici tarafından başlatılır ve anahtardan bir yanıt gelebilir de gelmeyebilir de.

Features: Denetleyici özellik isteği göndererek anahtarın kimliğini ve temel özelliklerini isteyebilir; anahtar, anahtarın kimliğini ve temel özelliklerini belirten bir özellik yanıtıyla yanıt vermelidir.

Configuration: Denetleyici, anahtardaki yapılandırma parametrelerini ayarlayabilir ve sorgulayabilir. Anahtar yalnızca denetleyiciden gelen bir soruyu yanıtlar.

Modify-State: Modify-State iletileri, anahtarlardaki durumu yönetmek için denetleyici tarafından gönderilir. Temel amacı, OpenFlow tablolarındaki akış / grup girişlerini eklemek, silmek, değiştirmek ve geçiş portu özelliklerini ayarlamaktır.

Read-State: Read-State iletileri, geçerli yapılandırma, istatistikler ve yetenekler gibi denetleyiciden ve anahtardan çeşitli bilgiler toplamak için kullanılır.

Packet-Out: Bunlar, denetleyicinin anahtardaki belirli bir bağlantı noktasından paketleri göndermek ve Paket-in iletileri yoluyla alınan paketleri iletmek için kullanılır. Paket çıkış iletileri, anahtarda depolanan bir pakete referans veren tam bir paket veya bir arabellek kimliği içermelidir. Mesaj, ayrıca, belirtilen sıraya göre uygulanacak eylemlerin bir listesini içermelidir; boş bir eylem listesine sahip paket bırakılır.

Barrier: Engel talep / cevap mesajları, tamamlanmış işlemler için bildirim almak için denetleyici tarafından kullanılır.

Role-Request: Role-Request mesajları, denetleyicisi tarafından OpenFlow kanalının rolünü ayarlamak veya bu rolü sorgulamak için kullanılır. Bu, çoğunlukla anahtar birden fazla denetleyiciye bağlandığında faydalı olmaktadır.

Asynchronous-Configuration: Zaman uyumsuz yapılandırma iletisi, eş zamansız iletilere ek bir filtre ayarlamak veya bu filtreyi sorgulamak için kullanılır. Bu, çoğunlukla anahtar birden fazla denetleyiciye bağlandığında faydalı olmaktadır.

Eş zamansız

Eş zamansız mesajlar, denetleyiciden istek olup olmamasına bakmaksızın anahtar tarafından gönderilirler. Anahtarlar, paket varış, anahtar değiştirme durumu veya hata belirtmek için denetleyicilere eş zamansız iletiler gönderir.

Packet-in: Paketler veri yolu tarafından alındığında ve denetleyiciye gönderildiğinde OFPT_PACKET_IN iletisini kullanırlar.

Packet-out: Denetleyiciden gelen ve otomatik olarak bir süre sonra süresi dolan mesajlardır.

Flow-Removed: Flow tablosundan çıkarılan bir flow hakkında denetleyiciyi bilgilendiren mesajdır.

Port Status: Denetleyiciye bağlantı noktasında değişiklik olduğu zaman bildirimde bulunur. Anahtarın, port yapılandırması veya port durumu değişiklikleri gibi port durumu mesajlarını denetleyiciye göndermesi beklenir.

Error: Anahtar hata mesajlarını kullanarak denetleyicilere sorunları bildirebilir.

Simetrik

Simetrik mesajlar herhangi bir talep edilmeksizin, denetleyici ve anahtar arasındaki her iki yönde de gönderilir.

Hello: Bağlantı başlatıldığında anahtar ve denetleyici arasında merhaba mesajları değiş tokuş edilir.

Echo: Yankı istek / yanıt iletileri anahtardan veya denetleyiciden gönderilebilir ve bir yankı yanıtı döndürmelidir. Bunlar çoğunlukla bir denetleyici-anahtar bağlantısının canlılığını doğrulamak için kullanılır ve gecikme veya bant genişliğini ölçmek için de kullanılabilmektedir.

Experimenter: Deneyci mesajları, OpenFlow anahtarlarının OpenFlow ileti türü alanında ek işlevler sunması için standart bir yol sağlar. Bu, gelecekteki OpenFlow düzeltmeleri için kullanılan özellikler için hazırlama alanı oluşturmaktadır [25].

2.4. YAPAY ZEKA

2.4.1. Yapay Zekâ Nedir?

Yapay zekâ, insanlar tarafından zekâ gerektirerek yapılan işlemlerin bilgisayarlar tarafından yapılması olarak ifade edilen bir bilim dalıdır. Mathison Turing'in "Makineler düşünebilir mi?" sorusunu ortaya atması ile yapay zekanın temelleri atılmış oldu. McCarthy, Minsky, Claude Shannon ve Nathaniel Rochester otomata teorisi ve sinir ağları gibi konular üzerinde görüşmek üzere 1956 yazında Dartmouth'da bir çalıştay düzenlediler. Bir yaz süren yapay zekâ çalıştayında makinelerin dili nasıl kullandığı, problemleri nasıl çözebildikleri ve kendilerini nasıl geliştirebilecekleri üzerine çeşitli çalışmalarda bulunmuşlardır. Bu çalıştayın

toplanmasında öncü olan McCarthy yapay zekâ adını ortaya atarak yapay zekanın isim babası olmuştur.

Yapay zekâ farklı türde problemlere etkili çözümler getirebilmek için çeşitli alt başlıklara ayrılmıştır. Bunlardan en yaygın olan yapay zekâ türleri ise bilgi tabanlı uzman sistemler, bulanık mantık yaklaşımı, yapay sinir ağları, geleneksel olmayan optimizasyon teknikleridir.

Uzman Sistemler: Uzman sistemler, belirli bir alanda uzmanlaşmış bir veya daha fazla insan uzmanın problem çözmede gerçekleştirdiği performansı gerçekleştirmeyi hedefleyen bilgisayar programlarıdır [27]. Uzman sistemler bilgi tabanlı bir yapıya sahiptir. Bu bilgi tabanı ise alanında uzman kişilerden elde edilen bilgiler sayesinde oluşturulmaktadır. Uzman sistemler "insan" ve "bilgisayar" olmak üzere iki temel unsurdan oluşurlar. İnsan kısmı "Kullanıcı" ve "Bilgi Mühendisi-Uzman" olarak ikiye ayrılır. Bilgisayar kısmında ise uzman sistemleri diğer karar destek sistemlerinden ayıran "Bilgi Tabanı" ve "Çıkarım Mekanizması" özellikleri yer alır [28]. Birçok uzman sistem, yapay zekâ programlama dilleri olan LISP ve PROLOG ile geliştirilmiştir. Uzman sistemler geliştirme sürecinde en az bir uzman ve bir bilgi mühendisinin beraber çalışmasıyla bilgi tabanını oluştururlar. Sistem çalıştığında verilen hipotezleri test edip açıklayabildiği gibi, kendisi de yeni öneriler ileri sürebilir [29].

Uzman sistemler pek çok alanda uygulama alanına sahiptir. Tahmin yapabilme özelliği ile hava tahmini, tahıl tahmini; teşhis yapabilme özelliği ile tıp ve elektronik alanlarında; planlama yapabilme özelliği askeri planlama alanlarında, tamir yapabilme özelliği ile bilgisayar ve otomobil alanlarında; kontrol yapabilme özelliği ile hava trafik kontrolü, savaş kontrolü gibi alanlarda kullanılabilmektedir [27].

Bulanık Mantık: Yapay zekanın bir alt dalı olan bulanık mantık isminde yer alan bulanıklık ifadesine rağmen aslında bir bulanıklığı değil doğru veya yanlış arasındaki ara değerleri ifade etmektedir. Bulanık mantık, netliği tam olmayan ifadelerin incelenmesinin yapıldığı bir yapay zekâ yöntemi olarak tanımlanmaktadır [30].

Bulanık mantık üzerine incelemeler 1960'lı yıllarda Lotfi A. Zadeh tarafından başlatılmıştır. Fakat gerçek anlamda bir uygulama 1970'li yıllara kadar meydana gelmemiştir. 70'li yıllarda Japonlar tarafından bulanık mantık uygulamaları üretilmeye başlanmıştır. ABD'nin de bu üretime katılmasıyla beraber bulanık mantık pek çok alanda uygulama alanı bulmuştur [31]. Karakter ve görüntü tanıma, elektrikli ev aletlerinin (klima, çamaşır makinesi, buzdolabı)

ayarlarının yapılmasında, metro işleyişlerinin ayarlanmasında, robotların yönlendirilmesinde ve daha pek çok alanda bulanık mantıktan faydalanılmaktadır. Bulanık mantığın işleyişi şöyle açıklayabiliriz. Örneğin; bir bulaşık makinesinin çalışma prensibini ele alalım. Bulaşıkların türü, bulaşıkların kirlilik derecesi ve bulaşık miktarına bağlı olarak, yıkama zamanı parametresi değişiklik gösterir. Fakat burada dikkat edilmesi gereken husus net ifadelerden kaçınılmasıdır. Yani bulaşık kirli veya temiz değil de bulaşıklar temiz, az kirli, orta kirli, çok kirli gibi ifadeler kullanılmaktadır. Aynı şekilde diğer parametrelerde de ara değerler kullanarak ihtimaller genişletilir ve daha doğru sonuçlar elde edilir [32].

2.4.2. Yapay Sinir Ağları Nedir?

Yapay sinir ağları kavramı, insan beyninin dijital bilgisayarlardan tamamen farklı bir çalışma prensibine sahip olduğunun anlaşılması üzerine ortaya çıkan bir kavramdır. İnsan beyni karmaşık bir bilgi işleme sistemine sahiptir. Beyin, günümüzde insanın en hızlı kabul edilen bilgisayardan bile hızlı olacak şekilde belirli hesaplamalar gerçekleştirmesini sağlayan, nöron olarak adlandırılan yapısal bir bileşene sahiptir [33]. Sinir ağı, deneyimsel bilginin depolanması ve kullanımı için uygun hale getirildiği doğal bir eğilime sahip olan, basit işlem birimlerinden oluşan, büyük ölçüde paralel dağıtılmış bir işlemcidir. Yapay sinir ağları ise insan beynindeki nöronlar gibi yapay nöronların aynı şekilde bir araya gelerek problemle ilgili bilgi toplayan ve problemi çözen sistemlerdir. Kısaca yapay sinir ağları, yapay nöronların farklı geometrik yollar kullanarak birbirlerine bağlandıkları ağ sistem yapısı olarak tanımlanmaktadır. Bu yapay sistemlerin bir problemle karşılaştıklarında öğrendiklerini kullanarak karar verme yetenekleri vardır**.** Yapay sinir ağlarında bilgi, ağlar tarafından edinilir ve elde edilen bu bilgi nöronlar arasındaki bağlantılar kullanılarak saklanır. Yapay sinir ağlarında ağ, bilgiyi saklamak ve bu bilgiyi işlevsel hale getirmek üzere kullanılan bir işlemci olarak düşünülmektedir [34].

2.4.2.1. Yapay Sinir Ağlarının Gelişimi

İlk yapay sinir ağı nörolog olan Warren McCulloch ve matematikçi Walter Pitts tarafından 1943 yılında oluşturulmuştur. McCulloch ve Pitts, insan beyninin hesaplama yeteneğine dayanan elektrik devreleriyle oluşturulmuş bir sinir ağı modellemişlerdir. 1949 yılında Donald Hebb yapay sinir ağlarından ve özellikle nöronlar arasındaki bağlantılardan kitabında bahsetmiştir. 1958 yılında ise Frank Rosenblatt, basit toplama ve çıkarma işlemleri kullanarak iki katmanlı bir bilgisayar öğrenme ağına dayalı model tanıma algoritması olan perceptron'u oluşturmuştur.

Yapay sinir ağı algoritması hesaplama gücü ve kaynaklara olan ihtiyacından dolayı gelişimini oldukça sessiz bir şekilde gerçekleştirmiştir. 2000'li yıllarla beraber bilgisayarların geliştirilmiş hesaplama kaynakları ve paralellikleri sebebiyle, yapay sinir ağları da gelişimini hızlandırmış ve daha popüler bir hale gelmiştir [35].

2.4.2.2. Yapay Sinir Ağlarının Özellikleri

Doğrusal Olmama: Hücrelerin birleşiminden meydana gelen yapay sinir ağları doğrusal değildir ve bu özellik tüm ağ boyunca geçerlidir. Yapay sinir ağları karmaşık ve doğrusal olmayan problemlerin çözümü için iyi bir araçtır.

Hata Toleransı: Yapay sinir ağlarında hata toleransı oldukça yüksektir. Bunun sebebi ise bilginin sistem etrafında düzenli olarak dağılmasıdır.

Eğitim: Belli bir amaca göre düzenlenmiş olan yapay sinir ağlarındaki sinirsel ağlar kendi değerlerini değiştirebilmekte ve problemin tam çözümü için kendilerini adapte edebilmektedir.

Öğrenme: Yapay sinir ağları doğru sonuçları elde etmek için ağırlıklarını değiştirerek algoritmalar tanımlamaktadır. Ağırlıkların ayarlandığı bu sürece öğrenme denmektedir. Yapay sinir ağlarının bir problemi öğrenebilmesi için, giriş ve çıkış verilerinin, yeterli sayıda örnek içermesi ve öğrenme kümesinin açık bir tanımının olması gerekmektedir.

Genelleme: Genelleme ile yapay sinir ağları, hiç karşılaşmadığı örneklerle ilgili olarak, çalıştıktan ve problemi öğrendikten sonra eğitim sürecinde istenen yanıtı oluşturma yeteneğine sahiptir [34].

2.4.2.3. Yapay Sinir Ağlarının Yapısı

İnsan beyninde bir nöron, giriş ve çıkış kanalları olan ve nöronları birbirine bağlayan soma, dendritler ve aksonlardan oluşmaktadır. Her nöron, dendritler vasıtasıyla diğer nöronlardan elektrokimyasal sinyaller / girdiler almaktadır. Bu elektrokimyasal girdilerin toplamı nöronu aktive etmek için yeterince güçlü olduğunda, nöron sinyali akson boyunca iletir ve bu elektrokimyasal sinyali aksonlara bağlanmış bulunan nöronlara geçirir. Bu şekilde diğer nöron uyarılmış olur. İnsan beynindeki bu biyolojik model yapay sinir ağları için bir model teşkil etmektedir [35]. Biyolojik sinir ağlarındaki sinir hücrelerine benzer yapı yapay sinir ağlarında yapay sinir hücresi olarak yer almaktadır. Yapay sinir hücresi bilgilerin toplanıp diğer sinir

hücresine iletimini toplama fonksiyonu, aktivasyon fonksiyonu ve ağırlık elemanları ile gerçekleştirmektedir. Biyolojik sinir hücresine karşılık gelen yapay sinir hücresi elemanları şekildeki gibi gösterilmektedir:

Sinir Sistemi	Yapay Sinir Ağı
Nöron	İşlem Elemanı
Dentrit	Toplama Fonksiyonu
Hücre Gövdesi	Aktivasyon Fonksiyonu
Akson	Eleman Çıkışı
Sinaps	Ağırlıklar

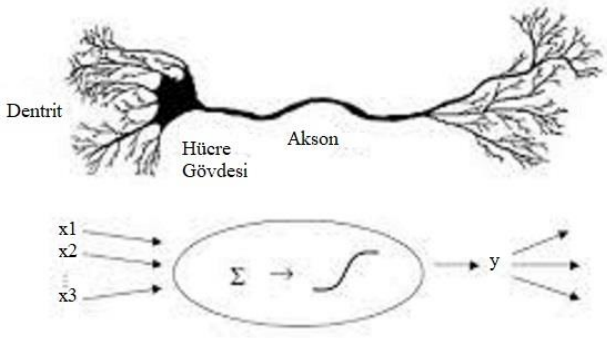
Şekil 2.13: Biyolojik Sinir Ağı ve Yapay Sinir Ağı Bileşenleri

Yapay sinir ağını oluşturan elemanlar birbirine paralel olacak şekilde üç katmanlı bir ağ yapısı oluşturmaktadır [36]. Bu katmanlar girdi katmanı, gizli katman ve çıktı katmanıdır.

Giriş Katmanı: Girilen verilerin ağa tanıtıldığı katmandır. Giriş katmanındaki nöronların sayısı, giriş verilerinin sayısıyla doğru orantılıdır. Giriş katmanında bulunan her bir nöron, bir sonraki katman olan gizli katmana iletilmektedir.

Gizli Katman: Gizli katman ağın temel işlevinin gerçekleştiği katmandır. Bu katmanda, giriş katmanından alınan veriler doğru bir şekilde işlenerek çıktı katmanına iletilmektedir.

Çıkış katmanı: Öğrenme çıktı katmanında gerçekleşmektedir. Ağdaki son katmandır ve gizli katmandan alınan verileri işler ve çıktıyı oluşturur. Elde edilen değerler yapay sinir ağındaki problem için çıktı değerlerini oluşturmaktadır. Ağ tarafından alınan çıktı sayısı nöronların sayısı ile doğru orantılıdır [34].



Şekil 2.14: Biyolojik Sinir Ağı ve Yapay Sinir Ağı [37]

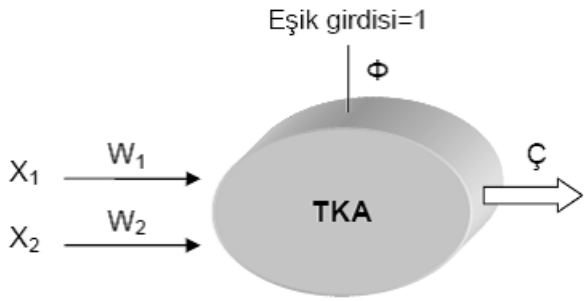
2.4.2.4. Yapay Sinir Ağlarının Çalışma Prensibi

Yapay sinir ağlarının ilk katmanı olan giriş katmanına dışarıdan gelen bilgiler ağın girdi elemanını oluşturmaktadır. Girdiler giriş katmanında ilerlemek suretiyle ara katmanlara iletilmektedir. Ara katman olan gizli katmana gelen girdiler işlenerek çıkış katmanına iletilmektedir. Girdilerin işlenmesi, ağa gelen bilgilerin ne kadar önemli olduklarına ve ağa ne kadar etkileri olduklarına göre ağırlık değerleri ile ifade edilmesi durumuna denmektedir. Ağın doğru sonuç vermesi için ağırlık değerlerinin özenli bir şekilde belirlenmesi gerekmektedir. Ağırlıkların belirlenmesi işlemi en doğru sonucu bulana kadar farklı ağırlık değerleri denenerek bulunmaktadır. Yapay sinir hücresi üzerindeki toplam girdi ise toplama fonksiyonu ile hesaplanmaktadır. Toplama fonksiyonu olarak ağırlıklı toplam yaygın olarak kullanılmaktadır. Bu da ağırlıkları daha önceden belirlenmiş olan girdilerin ağırlıkları ile çarpılıp ardından toplanması ile bulunmaktadır. Dışarıdan gelen bilginin çıktısını belirleme işlemi de aktivasyon fonksiyonu ile gerçekleşmektedir. Çeşitli aktivasyon fonksiyonları bulunmaktadır. Bunlar doğrusal aktivasyon fonksiyonu, adım aktivasyon fonksiyonu, sigmoid aktivasyon fonksiyonu, tanjant-hiperbolik aktivasyon fonksiyonu, eşik değer aktivasyon fonksiyonu ve sinüs aktivasyon fonksiyonudur. En yaygın kullanılan aktivasyon fonksiyonu ise sigmoid fonksiyonudur [36, 38].

2.4.2.5. Yapay Sinir Ağları Modelleri

Yapay sinir ağları ile geliştirilen en genel modeller şunlardır: Tek katmanlı algılayıcılar (perceptron, ADALINE/MADALINE) ve çok katmanlı algılayıcılar (ileri beslemeli, geri beslemeli)

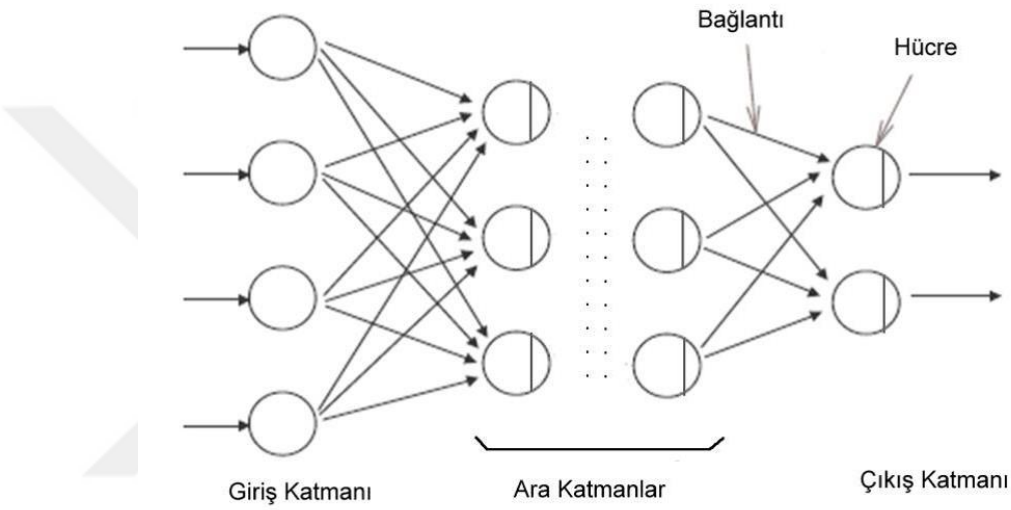
Tek Katmanlı Algılayıcılar: Tek katmanlı algılayıcılar yapay sinir ağlarının temelini oluşturmaktadır. Bu model girişler, çıkış ve bir de aktivasyon ve toplama fonksiyonun bulunduğu ana gövdeden oluşmaktadır. Bu ağlar eşik değeri mantığı ile çalışmaktadır. Girişlerin ağırlıklar ile çarpılıp, toplanması sonucu elde edilen değer eşik değer ile kıyaslanır. Elde edilen sonuçlar göz önüne alınarak iki ayrı sınıf oluşturulur. Bu iki sınıfı birbirinden ayırmak için sınıf ayracı olarak adlandırılan bir yüzey kullanılır. Tek katmanlı algılayıcıların en çok bilinen örnekleri ise perceptron ve ADALINE/MADALINE'dır [36, 37]. modeller öğrenme kuralındaki farklılıklardan dolayı birbirlerinden ayrılırlar. Perceptronlarda ağırlıklar, girişlerin öğrenme katsayısı adı verilen sabit bir sayı ile çarpılıp elde edilen değerin bu ağırlıklara eklenmesi veya çıkartılması sonucu değiştirilmektedir. Giriş değerlerine göre elde edilen sonuçlar ağırlıkları artırma veya azaltma eğilimi göstermektedir. ADALINE modelinde ise gerçekleşen ve olması istenen sonuç arasındaki hata oranına göre ağırlık değerleri değişmektedir. MADALINE modeli ise birden fazla ADALINE modelinin mantıksal operatörler kullanılarak birbirine bağlanması sonucu oluşmaktadır.



Şekil 2.15: Tek Katmanlı Yapay Sinir Ağı [38]

Çok Katmanlı Algılayıcılar: Tek katmanlı algılayıcıların çözüm üretmek de yetersiz kaldığı durumlarda algoritma ve mimari açıdan daha gelişmiş bir yapıya sahip olan çok katmanlı algılayıcılar kullanılmaktadır. Basit yapısı ve hızlı çalışmasından dolayı tek katmanlı algılayıcılar daha cazip görünse de problemler karmaşıklaştığı takdirde bu model yetersiz

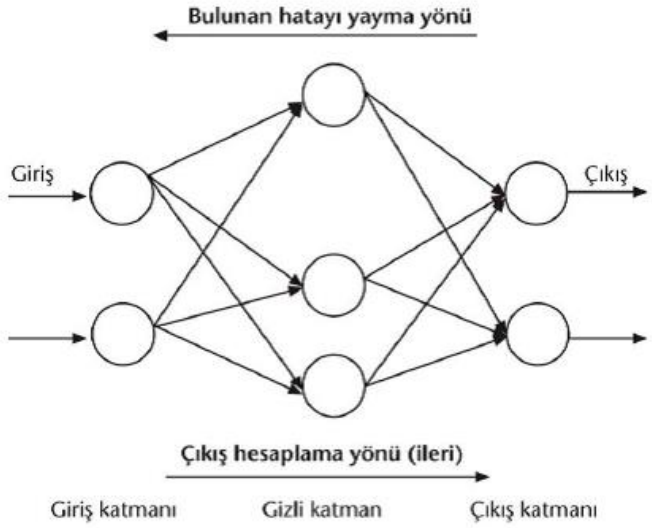
kalmaktadır. Bu sebeple daha geç sonuç üretmesine rağmen karmaşık problemlerde çok katmanlı algılayıcılar tercih edilmektedir. Çok katmanlı algılayıcılar ileri beslemeli ve geri beslemeli ağlar olmak üzere iki farklı modele sahiptir [36, 39]. İleri beslemeli ağlarda tek yönlü bir sinyal akışı gerçekleşmektedir. Bu akış katmanlar üzerinde gerçekleşir. Giriş katmanında hiçbir değişikliğe uğramayan bilgi ara katmana yani gizli katmana iletilir. Dış ortamdan gelen bilgi gizli katman ve çıkış katmanında işlenir ve bu şekilde çıkış verisi elde edilmiş olur.



Şekil 2.16: İleri Beslemeli Çok Katmanlı Yapay Sinir Ağı [38]

Geri beslemeli ağlarda ağırlık değerleri hata oranlarını azaltma gayesi ile değiştirilmektedir. Hesaplanmış olan hata değerleri geriye doğru yayılarak ağırlıklarda değişikliğe sebep

olmaktadır. Bu işlem çıkış katmanından giriş katmanına doğru gerçekleştiğinden dolayı bu modele geri beslemeli ağ denmiştir [36].



Şekil 2.17: Geri Beslemeli Çok Katmanlı Algılayıcılar [38]

2.4.2.6. Yapay Sinir Ağları Uygulamasında Kullanılan Performans Fonksiyonları

Ortalama Mutlak Yüzde Hatası (MAPE /Mean Absolute Percent Error)

MAPE, tahmin problemlerinde tahminin doğruluğunu hesaplamada kullanılan yaygın bir fonksiyondur. Ortalama mutlak yüzde hatası anlamına gelmektedir ve şu şekilde hesaplanmaktadır:

$$MAPE= 100/n \sum| (p_i - a_i) / a_i |$$

Burada "a" değeri belirli bir gözlemdeki gerçek değeri, "p" bu gözlem için modelin tahmin ettiği değeri ve "n" ise gözlemlenen durumların sayısını ifade etmektedir. Fonksiyonun çalışma prensibi ise şu şekildedir:

İlk olarak "$p_i - a_i$" ile hata hesaplanır. Daha sonra hatanın yüzde olarak oransal değeri "$(p_i - a_i) / a_i$ " ifadesi ile bulunur. Bulunan değerin mutlak değeri alınır ve son olarak da ortalama hesaplanır.

Hatanın mutlak değerinin alınmasının sebebi ise şu şekilde açıklanmaktadır: Bazı durumlarda tahmin edilen değerlerdeki fazlalık veya eksiklik miktar olarak birbirine eşit olabilmektedir. Örneğin gerçek değerin 5 fazlası ve 5 eksiği iki ayrı gözlem sonucu elde edilmiş olabilir. Bu durumda ortalama hatayı aldığımızda sonuç sıfır çıkmaktadır ki bu durumda tahmin de hata olmadığı gibi yanlış bir izlenime düşülmektedir. Bu durumdan kurtulmak için hatanın artış veya azalış yönünde olmasıyla ilgilenilmeyip sadece sapma miktarı hesaplanmaktadır. Bu sebeple hatanın mutlak değeri alınmaktadır [40, 41].

R-Kare (R^2 / R-Squared)

R^2, performans ölçütü MAPE gibi tahmin problemlerinde tahminin doğruluğunu hesaplarken kullanılmaktadır. Çok değişkenli durumlarda değişkenler arasındaki ilişkinin bağımlılığını ölçmek için kullanılmaktadır. R^2 fonksiyonu, değişkenler arasında doğrusal bir ilişki olup olmadığını ölçmemize olanak tanımaktadır. Fonksiyonun değeri 0 ile 1 arasındadır. Eğer sonuç sıfıra yakın çıkarsa sonuç doğruluktan uzaklaşmış, bire yakın çıkarsa tahminin doğru olma olasılığı yüksek demektir.

Fonksiyonu oluşturan değişkenler bağımlı ve bağımsız değişkenler olmak üzere ikiye ayrılmaktadır. Bağımsız değişkenler başka bir değişken tarafından etkilenmeyen diğer değişkeni etkileyen onu değiştiren değişken olarak açıklanmaktadır. Bağımlı değişken ise bağımsız değişkenden etkilenen onun değiştirdiği etkilediği değişken olarak tanımlanmaktadır. Tahmin problemlerinden örnek vermek gerekirse, tahmin değişkeni bağımsız değişkeni sonuç değişkeni ise bağımlı değişkeni ifade etmektedir [42].

Yapay sinir ağlarında kullanılan R^2 fonksiyonunu şu şekilde ifade edebiliriz:

SStotal=sum (test_output-mean(test_output))^2

SSerror=sum (network_ciktisi-test_output)^2

R2=1-SSerror/SStotal

İlk olarak test çıktısından test çıktısının ortalamasını çıkartılır ve ardından kareleri alındıktan sonra tüm sonuçlar toplanmaktadır. Elde edilen değere toplam varyans denmektedir.

İkinci olarak yapay ağın ürettiği çıktıdan gerçek çıktı değeri çıkartılır ve ardından kareleri alındıktan sonra tüm sonuçlar toplanmaktadır. Elde edilen değere açıklanan varyans denmektedir.

Son olarak açıklanan varyans toplam varyansa bölünür ve 1'den çıkartılır. Elde edilen sonuç R2 değerini vermektedir [43].

2.5. YAPAY ZEKA OPTİMİZASYON TEKNİKLERİ

2.5.1. Yapay Zekâ Optimizasyon Teknikleri Nedir?

Meta sezgisel olarak da bilinen yapay zekâ optimizasyon teknikleri optimizasyon problemlerinin çözümü için etkili sonuçlar sunmaktadır. Bu teknikler en iyi çözüme ulaşmak için var olan çeşitli hamlelerden en iyi sonuç verecek olanı bulmaya yarayan bilgisayar algoritmalarından oluşmaktadır. Problemlerinin pek çoğunun nihai bir çözümü olmadığından ve bu problemlerin çözümü için gerekli olan sürenin gayet büyük olmasından dolayı, araştırmacılar nihai çözüme yakın bir sonuç veren en hızlı algoritmayı bulmaya çalışmışlardır. Sonuç olarak da yapay zekâ optimizasyon teknikleri elde edilmiştir. Yapay zekâ optimizasyon teknikleri kesin çözüm metotları için bir alt veya üst sınır oluşturmaları nedeni ile de önemlidir. Yapay zekâ algoritmaları kesin bir çözüm üretmemekle beraber muhtemel en iyi çözümü bulmayı sağlamaktadır.

Yapay zekâ optimizasyon teknikleri probleme bağlı algoritmalardır. Yani bir problemde başarılı olurken diğer bir problem için aynı şekilde başarılı olamayabilirler. Yapay zekâ optimizasyon tekniklerinden başlıcaları; tabu arama algoritmaları, tavlama benzetim algoritmaları (simulated annealing), genetik algoritmalar, karınca koloni algoritmaları, bağışıklık algoritmaları ve diferansiyel gelişim algoritmaları sayılabilir.

2.5.2. Tabu Arama Algoritması Nedir?

Fred Glover tarafından 1986 yılında önerilen "tabu arama algoritması", yerel arama metotlarının eksikliklerini gidermek üzere ortaya çıkmıştır. Yerel arama metotları ile bazı karmaşık kombinatoryel problemlere optimal çözümler getirilememekteydi. Tabu arama bu tip problemlere en etkili sezgisel çözümü getirmektedir. Tabu aramanın çalışma prensibi, yerel optimumla karşılaştığında en iyi olmayan bu çözüme izin vererek yerel arama üzerinden takibe

devam etmektir [44]. Bu aramaya "tabu" denmesinin yani "yasaklı arama" olarak ifade edilmesinin sebebi ise bu aramayı yaparken birtakım kısıtlamaların oluşturulmasıyla ilgilidir. Kısıtlamalar ise hafıza yapılarından faydalanılarak oluşturulur [45]. Bu kısıtlar ile normalde kabul edilebilen zor durumlar geçilerek en optimal sonuca ulaşmamız sağlanır. Tabu arama da problem çözme işleminin akıllı olarak nitelendirilebilmesi için uyarlanabilir bir hafıza ve duyarlı bir araştırmanın olması gerekir. Bu özellikler ile probleme özel çözümler üretebilme yetisi edinen tabu arama algoritması, meta-sezgisel bir arama algoritması olarak kabul edilmektedir. Tabu arama algoritması problem çözümünde dört ayrı kriterde yüksek performans sağlamaktadır. Bunlar; elde edilen sonuçların kaliteli olması, sonuçları gereken zamanda hızlı bir şekilde elde etmesi, problem özelliklerine göre değişebilme yani esneklik ve son olarak da veri de meydana gelen değişikliklere karşı duyarlılıktır. Bu özellikler sayesinde tabu arama algoritması hemen hemen her türlü optimizasyon problemine uygulanabilmektedir [46].

2.5.3. Tabu Arama Algoritmasını Oluşturan Elemanlar

$$\begin{aligned} &\textit{Optimize } f(x) \\ &\textit{Min} \quad c(x) \\ &x \in X \end{aligned}$$

Tabu arama algoritmasının işleyişini yanda bulunan denklem ile basit bir şekilde matematiksel hale getirebiliriz. Denklemde yer alan *c(x)* ifadesi, amaç fonksiyonunun iyilik derecesini belli etmektedir. *c(x)* ifadesinin minimum veya maximum olması amaç fonksiyonunun kârının ne kadar iyi olduğunu ifade eder. Amaç fonksiyonunun *(f(x))* iyilik derecesi aranırken tabu arama algoritması gereği bazı kısıtlamalara uymak gerekmektedir. Bu kısıtlamalar *x* ile ifade edilmektedir. Kısıtlamalar yani *x* değeri bellekte tutulan hareketleri göstermekle beraber, *X* ifadesi ise algoritmada var olan tüm hareketleri temsil etmektedir. Hareketlerin bu şekilde ikiye bölünmesi ile çözüm aramada bazı hareketler tabu olarak algılanacak ve engellenecek, diğer hareketlere odaklanılacaktır.

2.5.3.1. Çözüm Uzayı ve Başlangıç Çözümü

Tabu arama algoritmasında optimal bir çözüme ulaşmak için ilk olarak bir başlangıç çözümü üretilmektedir. Bu çözüme rastgele veya problemle ilgili özel olarak geliştirilmiş sezgisel bir algoritmadan faydalanılarak ulaşılır. Başlangıç çözümü elde edildikten sonra bir takım dönüşüm teknikleri kullanılarak çözüme gidilmektedir [47].

2.5.3.2. Komşu Arama Mekanizması

Yerel arama veya tabu aramanın her yinelemesinde, başlangıç çözümüne uygulanabilen yerel dönüşümler, arama uzayında komşu çözümler kümesi olarak adlandırılır. x başlangıç çözümünü, x‘ ise komşu çözümü temsil etmektedir. Komşu çözümlere götüren hareketler deneme hareketi olarak adlandırılır ve bunlar çeşitlilik göstermektedir. Bu hareketlerin çeşitliliğine bağlı olarak da farklı komşuluk yapıları oluşur ve bu komşuluklardan en iyisi seçilir. Eldeki probleme yönelik olarak da çeşitli komşuluk yapıları bulunmaktadır. Komşuluk yapısı tabu arama algoritmasının işleyişindeki en önemli faktörlerden biridir [44, 45].

2.5.3.3. Amaç Değerinin Hesaplanması

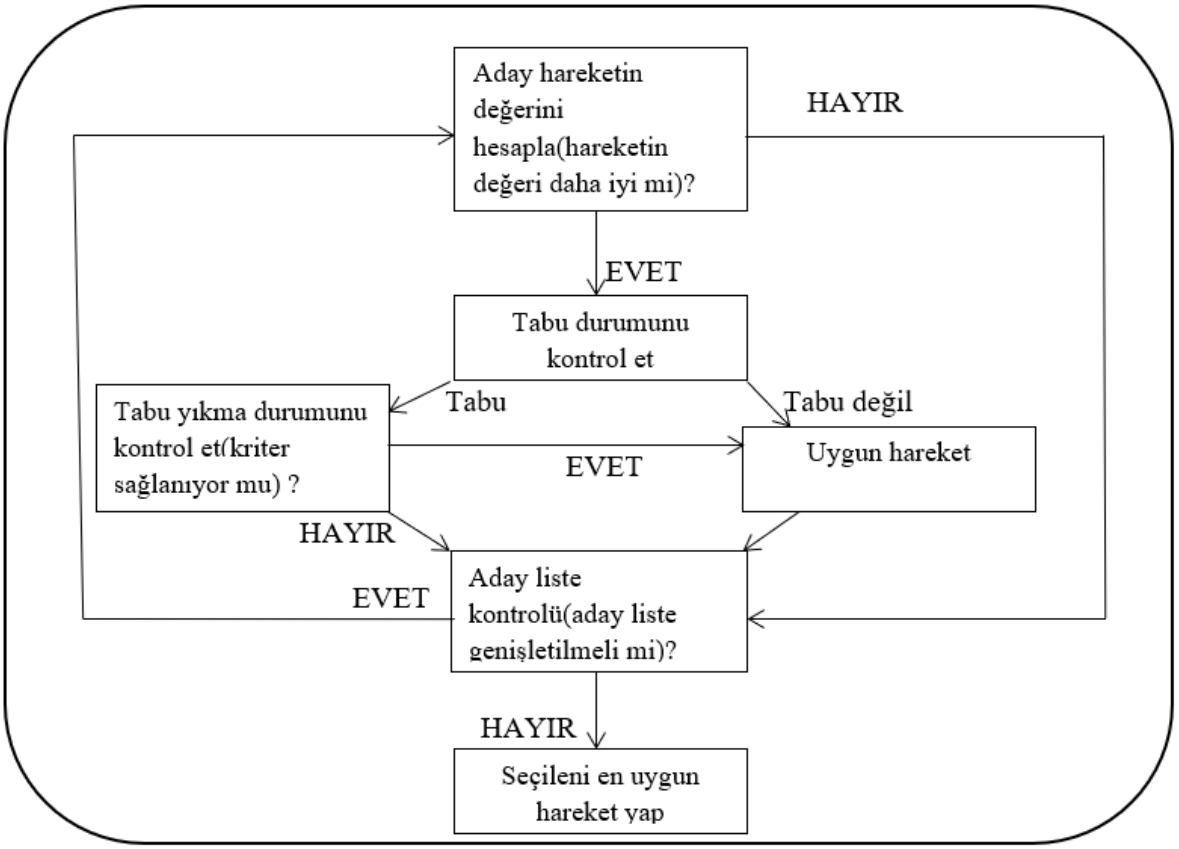
Bir hareketin değeri iyilik derecesi ile ölçülür. Amaç fonksiyonundaki değişiklikler hareketin değerini belirlemektedir. Bir hareketin uygulanmasından önce ve sonra amaç fonksiyonunun değerinde meydana gelen fark fonksiyonun değeri olarak adlandırılır.

2.5.3.4. Tabu Listeleri

Tabu arama algoritmasını yerel arama algoritmalarından farklı kılan en önemli özellik adından da anlaşılacağı gibi aramanın tabular üzerine yani yasaklar üzerine oturtulmuş olmasıdır. Tabu arama iyileştirilmemiş hareketler sayesinde yerel en iyiden kaçınır ve döngüye girmeyi engeller. Tabular, tabu listesi adı verilen kısa bir hafızada saklanırlar. Bu listelerde sabit ve sınırlı miktarda bilgi tutulmaktadır. En çok kullanılan tabu türü mevcut çözüm üzerinde yapılan son hareketlerden oluşmaktadır. Tabu olarak nitelendirilen bir hareket diğer iterasyona kadar tabu listesinde tutulmaktadır. Böylece hareketin tekrarı yani döngü engellenmiş olur. Tabu listesinin aktif olma süresi ne kadar artarsa tabu olarak listede tutulan hareketlerin sayısı da buna paralel olarak artmaktadır. Hareketlerin fazla kısıtlanması ise daha küçük bir alanda komşulukların incelenmesine sebebiyet vermektedir [39, 43]. Tabu listesinin boyutu da en az aktiflik süresi kadar önemlidir. Eğer listenin boyutu küçük seçilirse, tabu olarak seçilen hareketlerin sayısı azalacağı için daha önce denenmiş çözümler ve döngüler meydana gelebilmektedir. Eğer listenin boyutu büyük seçilirse, çok fazla hareket kısıtlanacağı için algoritmada hiçbir hareket gerçekleşemeyebilir ve arama kısa sürede durabilir. Bu yüzden tabu listesinin boyutu seçilirken çok dikkatli olunması gerekmektedir [45].

2.5.3.5. Aday Liste Stratejisi

Aday liste, bir çözüm için gerçekleştirilen hareketlerin oluşturduğu komşulukların alt kümesidir. Komşuluklar çok fazla olduğu takdirde bu listenin oluşturulması aşamasında da gerek zaman gerek maliyet açısından problemler oluşmaktadır. Çok büyük olan komşuluklarda hesaplama zorluğu oluşmaktadır. Bu sorunları gidermek için aday liste stratejileri kullanılmaktadır. Böylece hesaplama zorluğu azalmakla beraber, makul bir süre içerisinde kaliteli çözümlere ulaşılabilmektedir [45, 49].



Şekil 2.18: Aday Seçimi

2.5.3.6. Tabu Yıkma Ölçütü

Tabu arama algoritmasının esnek bir yapıya sahip olduğundan bahsedilmişti. Bu esnekliği daha önceden tabu olarak kabul edilen bir hareketin, uygun bir hareket olarak değiştirilmesiyle ifade edebiliriz. Bu şekilde hareketlerin tabu durumlarında değişikliğe gidilebilir ve esneklik sağlanmış olur. Tabu olarak kabul edilen bir hareket eğer kriterleri sağlıyorsa tabu listesinden

çıkar ve aday listesine geçer. Böylece hareketin tamamen iptaline engel olunabilmektedir. İki türlü kriter bulunmaktadır. Bunlardan ilki amaca dayalı kriter ikincisi ise yokluğa dayalı kriterdir. Amaca dayalı kriterde, tabu olan hareket mevcut çözümden daha iyi bir sonuç üretiyorsa bu hareket artık tabu olmaktan çıkar ve aday listesine alınır [50]. Yokluğa dayalı kriter ise tüm hareketlerin tabu olduğu durumda çözüme en yakın hareketin seçilip aday listesine alınması şeklindedir [51].

2.5.3.8. Durdurma Koşulu

Çözüme ulaşmak için kullanılan algoritma sonsuz bir döngüde çalışıp giderse bu durum çözüm kalitesi açısından değer düşürmektedir. Probleme uygulanan algoritma belirli süre zarfında istenen sonucu üretiyorsa o zaman algoritmanın başarısından söz edilebilir. Yani algoritmanın bir başlangıç ve bitişi olmak zorundadır. Tabu arama algoritmasında algoritmanın sonlandırılması üç farklı durumda olmaktadır. Eğer belirli sayıda bir tekrarlama oluyorsa, amaç fonksiyonu değerinde bir değişme olmaksızın birtakım yinelemelerden sonra ve daha önceden problem için belirlenmiş optimal çözüme veya eşik değerine ulaşıldığında tabu arama algoritması durmaktadır [44, 52].

2.5.3.9. Arama Yoğunlaştırma (Intensification)

Arama yoğunlaştırma, aramanın dar bir alana odaklanmasını temel almaktadır. Bu ifade de dar alan ile bahsedilmek istenen çözüm için umut vadeden bölgelerdir. Yoğunlaştırma, problemi çözüme götürecek olan muhtemel alanın daha ayrıntılı bir şekilde incelenmesi olarak açıklanmaktadır. Tabu arama uygulamalarının pek çoğunda, yoğunlaştırma işlemi kullanılabilmektedir. Fakat her fırsatta bu tekniği kullanmak her zaman iyi sonuçlar vermeyebilir. Çünkü yoğunlaşma tekniğini kullanmadan da iyi sonuçlar alınan bir probleme bu tekniği uygulamak zaman kaybından başka bir şey değildir [44].

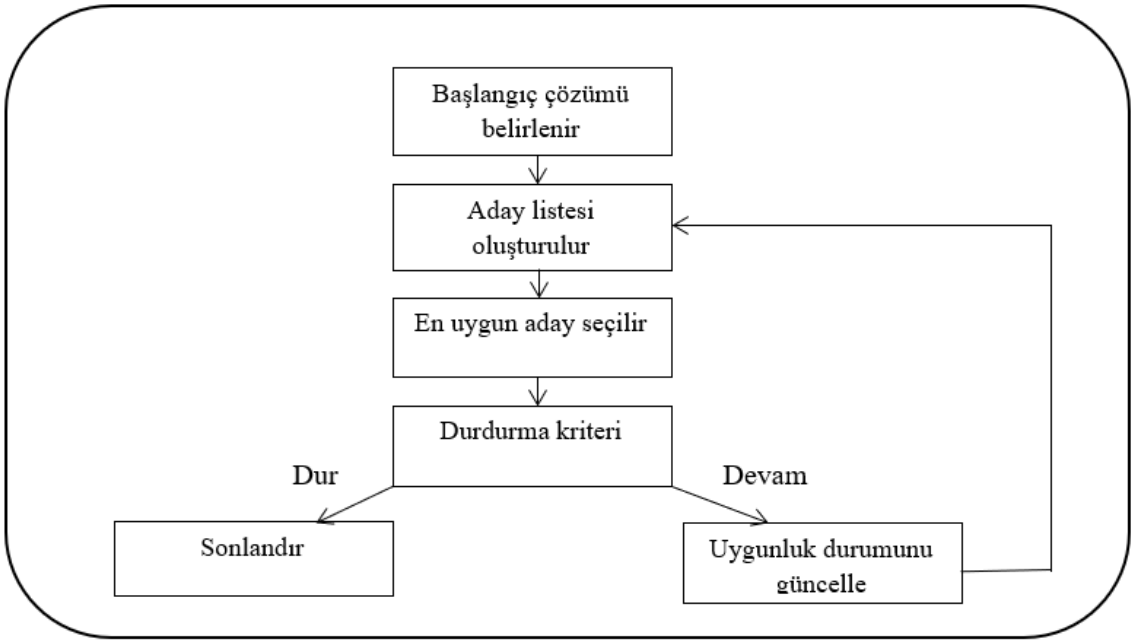
2.5.3.10. Arama Çeşitlendirme (Diversification)

Arama çeşitlendirme, yoğunlaştırmanın aksine belirli dar bir alan üzerinde değil de daha geniş bir alan üzerinde faaliyet göstermektedir. Arama yoğunlaştırma tekniği çoğu zaman iyi sonuçlar vermesine rağmen bazen kısıtlı dar bir alana yoğunlaşmak problemin çözümünde başarısızlığa sebep olabilir. Belki de çözüm diye odaklandığımız alanın çok uzak komşuluklarında bulunan hareket dar alana yoğunlaşıldığı için gözden kaçabilmektedir. Çeşitlendirme, aramayı arama

alanının daha önce keşfedilmemiş alanlarına zorlayarak bu sorunu hafifletmeye çalışan bir mekanizmadır. Bu sebeple çeşitlendirme, mevcut çözümün en uzak komşulukları arasında arama yapılarak gerçekleşmektedir [44].

2.5.4. Tabu Arama Algoritmasının İşleyişi

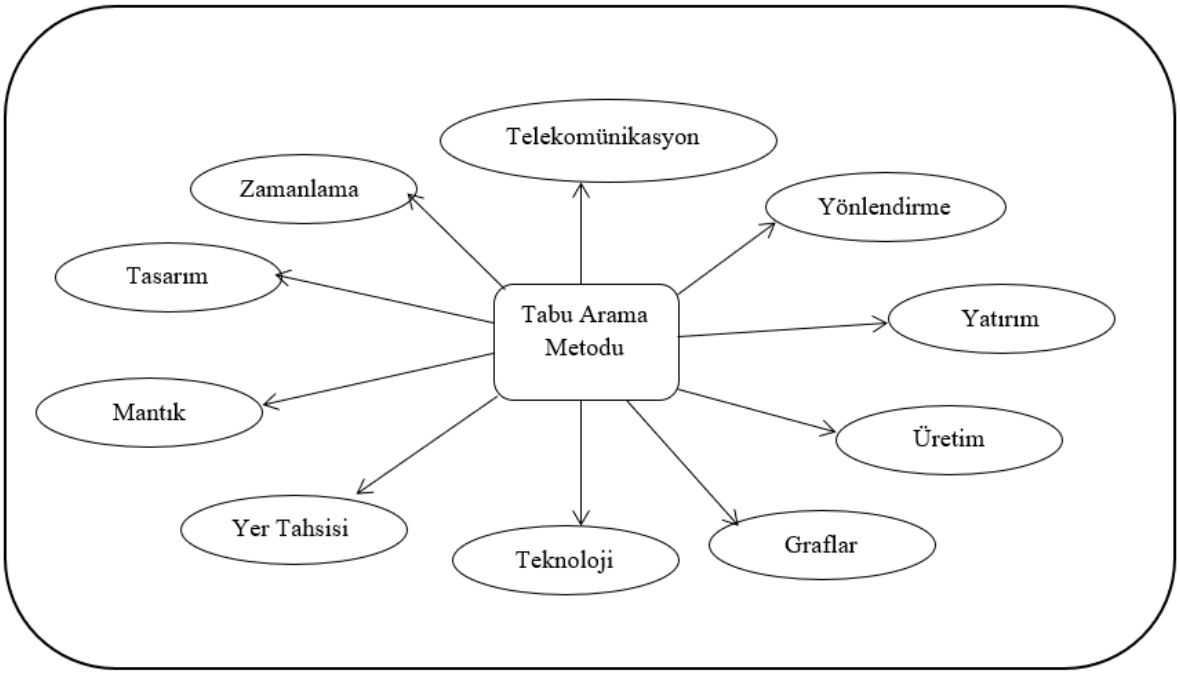
Şimdiye kadar bahsedilen kavramların ışığında tabu arama algoritmasının işleyişini bütün olarak ele alırsak; ilk olarak algoritmaya, probleme uygun bir başlangıç çözümü belirlenerek başlanılır. Bu çözüm rastgele veya problemle ilgili bilgilerden yola çıkarak üretilmiş bir çözüm olabilir. Çözüm, mevcut çözüm ve komşu çözüm olarak hafızada tutulur. Daha sonra amaç fonksiyonunu gerçekleştirmek için gereken hareketler aday listesi ve tabu listesi olmak üzere iki ayrı listede tutulur. Çözüme ulaşmamızı kolaylaştıran hareketler aday listesinde tutulurken, amaç fonksiyonunu çözümden uzaklaştıran hareketler yasaklı listede tutulmaktadır. Bazı durumlarda yasaklı listede olsa bile başka bir iterasyonda fonksiyonu çözüme ulaştıran bir hareket bulunuyorsa tabu yıkma kriterini sağlayarak bu hareket aday listeye alınır. Çözüm için tekrar bir arama yapılır. Bulunan çözüm mevcut çözümden daha iyiyse, o ana kadar ki en iyi çözümse yeni çözüm en iyi çözüm olarak belirlenir. Son olarak da bir durdurma kriteri belirlenir ve bu kriter sağlanana kadar arama devam eder.



Şekil 2.19: Tabu Arama Algoritması

2.5.5. Tabu Arama Algoritması Uygulama Alanları

Tabu arama algoritması yerel arama algoritmalarındaki eksiklikleri gidermek için tercih edilen meta-sezgisel bir algoritmadır. Bu algoritma sayesinde yerel optimal çözümde sıkışıp kalma ihtimali ortadan kaldırılır ve genel maximum çözüme ulaşılabilir. Tabu arama ilk çıktığı günden beri çeşitli sorunların çözümünde diğer alternatif yöntemlerden daha üstün çözümler elde etmiştir. Problem çözme konusundaki pratikliği, problemlere son derece kaliteli çözümler elde etme özelliği ve uygulama kolaylığı bu algoritmanın hızla büyümesine ve yaygınlaşmasına sebep olmuştur [53]. Zamanlama, sıralama, kaynak tahsisi, yatırım planlaması, telekomünikasyon ve daha pek çok alanda tabu arama metodu kullanılarak problemlere etkin ve pratik çözümler getirilmektedir.



Şekil 2.20: Tabu Arama Metodu Uygulama Alanları

Zamanlama kategorisine; akış-zamanlı hücre üretimi, heterojen işlemci zamanlaması, işgücü planlaması, makina çizelgeleme, sıralama ve toplu işleme örnek gösterilebilir.

Tasarım kategorisine; bilgisayar destekli tasarım, hata toleranslı ağlar, ulaşım ağı tasarımı, mimari uzay tasarımı, diyagram tutarlılığı, yerleşim planlaması örnek gösterilebilir.

Mantık kategorisine; olasılıksal mantık, örüntü tanıma, sınıflandırma, veri madenciliği, kümeleme, istatistiksel ayrımcılık, sinir ağı eğitimi örnek gösterilebilir.

Yer tahsisi kategorisine; kuadratik görev, ikinci dereceden yarı tahsis, çok seviyeli genelleştirilmiş tahsis örnek gösterilebilir.

Teknoloji kategorisine; sismik ters çevirme, uzay istasyonu inşaatı, devre hücresi yerleştirme, elektrik güç dağıtımı örnek gösterilebilir.

Graflar kategorisine; grafik bölme, grafik renklendirme, max. planlayıcı grafikler örnek gösterilebilir.

Üretim-Yatırım kategorisine; tedarik zinciri yönetimi, kapasitelenmiş MRP, parça seçimi, esnek imalat örnek gösterilebilir.

Yönlendirme kategorisine; araç yönlendirme, çok modlu yönlendirme, seyyar satıcı, karışık filo yönlendirme örnek gösterilebilir.

Telekomünikasyon kategorisine; çağrı yönlendirme, hub tesisi konumu, senkron optik ağlar, hizmetler için ağ tasarımı örnek gösterilebilir [49].

2.5.6. Tavlama Benzetim Algoritması Nedir?

Kirkpatrick, Gelatt ve Vecchi tarafından oluşturulan tavlama benzetim algoritması olasılık tabanlı sezgisel bir algoritmadır. Katı cisimlerin fiziksel tavlanma işleminden yola çıkılarak oluşturulan tavlama benzetim algoritması, karmaşık optimizasyon problemlerinin çözümünde kullanılmaktadır. Tavlama, katı cismin erime noktasına kadar ısıtılıp ardından cisim kristalize olana kadar yavaş yavaş cismi soğutma işlemidir. Yüksek sıcaklıkta enerji seviyesi de yüksek olan cisimlerin atomları, düzgün yapılı bir kristal haline geçtiği esnada sistemin enerjisi minimum düzeye inmektedir. Sistemin enerjisini düşürmek içinse sıcaklık azaltılmaktadır. Bu soğutma işlemi esnasında çok hızlı davranılırsa, cismin kristalize yapısında bozulma gerçekleşmekte ve cismin yapısında düzensizlik oluşmaktadır. Bu sebeple soğutma işlemi sırasında çok itinalı davranılması gerekmektedir [54].

Tavlama benzetim algoritması, çok değişkenli fonksiyonlarda fonksiyonun en büyük ya da en küçük değerlerinin bulunması amacıyla tasarlanmıştır. Bu algoritma kısa süre de en iyi çözümü

gerçekleştirmektedir. Bu sebeple matematiksel bir modelle ifade edilemeyen problemlerin optimize edilmesi aşamasında tercih edilmektedir.

Tavlama benzetim algoritması algoritmasında amaç fonksiyon değeri azalma eğilimindedir. Fakat yerel minimuma takılı kalmamak için bazı durumlarda bu fonksiyonda yüksek değerlerde kabul görülmektedir. Bu şekilde yerel minimuma takılmayıp, daha iyi global bir çözümü aramaya devam edilmektedir [55].

Tavlama benzetim algoritması elektronik devre tasarımı, görüntü işleme, yol bulma problemleri, seyahat problemleri, malzeme fiziği benzetimi, kesme ve paketleme problemleri, iş/akış çizelgeleme ve buna benzer pek çok problemin çözümünde kullanılmaktadır [56].

3. MALZEME VE YÖNTEM

3.1. FLOODLIGHT

Floodlight, yazılım tanımlı ağ (SDN) ortamında trafik akışlarını düzenlemek için OpenFlow protokolüyle çalışan Big Switch Networks tarafından sunulan bir SDN denetleyicisidir. Yazılım tanımlı ağlarda en yaygın olarak kullanılan OpenFlow, SDN denetleyicisinin (ağın beyinleri) ağda değişiklik yapmak için yönlendirme düzlemiyle yani anahtarlarla konuşmasına izin veren açık iletişim protokolüdür.

Floodlight denetleyiciyi, tüm ağ kurallarını korumak ve ağ trafiğinin nasıl ele alınması gerektiği konusunda gerekli talimatları vermekle sorumludur. Bu denetleyici, işletmelerin değişen ihtiyaçlarına daha iyi uyarlanabilmekte ve işletmelerin ağları üzerinde daha iyi kontrol sahibi olmalarına olanak sağlamaktadır.

Floodlight denetleyicisi başlangıçta Big Switch tarafından OpenDaylight Projesi kapsamında sunulmuş, ancak Haziran 2013'te Big Switch, bu projeden ayrılarak bireysel olarak varlığını sürdürmüştür. Floodlight denetleyicisi hala açık kaynak olsa bile, OpenDaylight Projesi ile ilgisi bulunmamaktadır.

Floodlight denetleyicisi, geliştiricilere bazı avantajlar sağlamaktadır. Yazılım geliştirenlere, yazılımı kolayca adapte edebilme imkânı ve Java ile uyumlu uygulamalar geliştirebilme olanağı sunmaktadır. Hem fiziksel hem de sanal OpenFlow uyumlu anahtarlarla test edilen Floodlight denetleyicisi, çeşitli ortamlarda çalışabilir. Ayrıca OpenFlow uyumlu anahtar gruplarının, geleneksel anahtarlarla bağlı olduğu ağları da destekleyebilmektedir [57].

3.2. MATLAB

MATLAB ismi MATrix LABoratory (Matrix Laboratuarı) kelimelerinden türetilmiş bir ifadedir. 1970'li yılların sonlarına doğru Fortran Linpack ve Eispack programlarını daha elverişli hale getirmek maksadıyla ortaya çıkmıştır.

MATLAB, özellikle mühendisler ve bilim insanları için tasarlanmış bir programlama platformudur. MATLAB 'in en temel özelliği MATLAB' in kendine has programlama dilidir. Bu dil matematiksel hesaplamalar üzerine kurulmuş bir dildir.

MATLAB' i kullanarak veri analizi, algoritma geliştirme, model oluşturma ve yeni uygulamalar yapılabilmektedir.

Endüstride ve akademide milyonlarca mühendis ve bilim adamı MATLAB' i kullanmaktadır. Derin öğrenme ve makine öğrenimi, sinyal işleme ve iletişim, görüntü ve video işleme, kontrol sistemleri, test ve ölçüm, hesaplama finansmanı ve hesaplama biyolojisi gibi uygulamalar için MATLAB tercih edilmektedir [58].

3.3. SANAL MAKİNE

Floodlight denetleyicisi Linux ortamında çalışan bir denetleyicidir. Bu sebepten floodlight ile çalışmaya başlamadan önce çalışma ortamının Linux olarak ayarlanması gerekmektedir. Bunun için de bilgisayarımızın işletim sistemini Linux' a çevirebilir ya da daha kolay bir yöntem olarak floodlight sanal makinesini bilgisayarımıza kurabiliriz. Floodlight sanal makinesi, uygulama geliştirirken ihtiyaç duyduğumuz programları içerisinde barındıran Linux tabanlı sanal bir pakettir. İçerisinde eclipse, mininet ve wireshark programlarını barındırmaktadır.

3.3.1. Eclipse

Eclipse, Java programlama dilini ve C / C ++, Python, PERL, Ruby gibi diğer programlama dillerini kullanarak uygulamalar geliştirmek için oluşturulmuş bir platformdur.

Eclipse platformu eklentilerden oluşmaktadır ve ek eklentiler kullanılarak genişletilebilecek şekilde tasarlanmıştır. Java kullanılarak geliştirilen Eclipse platformu, zengin istemci uygulamaları, entegre geliştirme ortamları ve diğer araçları geliştirmek için kullanılabilmektedir.

Java Geliştirme Araçları (JDT) projesi, Eclipse' in Java IDE olarak kullanılmasına izin veren bir eklenti sağlamaktadır. PyDev, Eclipse' in bir Python IDE, C / C ++ Geliştirme Araçları (CDT) bir eklenti olarak kullanılmasına izin veren bir eklentidir. Eclipse' in C / C ++ kullanarak uygulama geliştirmek için kullanılmasına olanak tanıyan Eclipse Scala eklentisi, Eclipse' in Scala uygulamalarını geliştirmek için bir IDE kullanılmasına izin vermektedir.

Eclipse ücretsiz ve açık kaynak kodlu olması, çoklu dil desteği, kod tamamlama, kolay paylaşılabilme, toplum desteği, genişletilebilir eklentileri gibi özellikleri sayesinde çok fazla tercih edilmektedir [59].

3.3.2. Mininet

Mininet, sanal hostlar, anahtarlar, denetleyiciler ve bağlantılar oluşturan bir ağ emülatörüdür. Mininet hostları, standart Linux ağ yazılımı ile çalıştırılmaktadır. Mininet anahtarları ise yazılım tanımlı ağlar için, son derece esnek ve özel bir yönlendirme gerçekleştiren OpenFlow' u desteklemektedir.

Mininet araştırma, geliştirme, öğrenme, prototipleme, test etme, hata ayıklama gibi görevleri desteklemektedir. Mininet, OpenFlow uygulamalarını geliştirmek için basit ve ucuz bir ağ test yatağı sağlamaktadır. Birden çok eş zamanlı geliştiricinin aynı topolojide bağımsız olarak çalışmasını sağlamaktadır. Fiziksel bir ağ kurmaya gerek kalmadan karmaşık topoloji testini etkinleştirmektedir. Özel topolojileri desteklemekte ve bir dizi parametrik topoloji içermektedir. Programlamaya gerek kalmadan otomatik olarak topoloji oluşturma imkânı sağlamakla beraber topoloji oluşturma ve deneme için basit ve genişletilebilir bir Python API' si de sağlamaktadır.

Mininet, doğru sistem davranışını elde etmenin ve topolojileri denemenin kolay bir yolunu sunmaktadır. Mininet ağları, standart Unix / Linux ağ uygulamalarının yanı sıra gerçek Linux çekirdeği ve ağ yığınını içeren gerçek kodu da çalıştırmaktadır [60].

3.3.3. Floodlight Projesi

Floodlight sanal makinesi yüklendiğinde içerisinde ki eclipse ve mininet programlarına uyumlu olarak Floodlight v1.1 projesi elde edilmektedir. Bu proje java dili ile yazılmış içerisinde pek çok geliştiricinin katkısıyla oluşan çeşitli uygulamalardan oluşmaktadır. Floodlight projesi yalnızca bir OpenFlow denetleyicisi olmaktan ziyade aynı zamanda yerleşik bir uygulamalar topluluğudur. Floodlight projesi, bir OpenFlow ağını kontrol etmek ve sorgulamak için bir dizi ortak işlevsellik gerçekleştirirken, üstündeki uygulamalar ağ üzerinden farklı kullanıcı ihtiyaçlarını çözmek için farklı işler gerçekleştirmektedir [61].

3.3.4. WireShark

Floodlight VM, Wireshark ile donatılmış olarak gelmektedir. Ubuntu GUI' de veya terminal aracılığıyla çalıştırılmaktadır.

Wireshark bir ağ paket analizcisidir. Bir ağ paket analizcisi, ağ paketlerini yakalamaya çalışmakta ve bu paket verilerini mümkün olduğunca ayrıntılı olarak görüntülemeye çalışmaktadır. Wireshark, bugün mevcut olan en iyi açık kaynaklı paket analizörlerinden biridir [62].

3.4. NNTOOL

MATLAB üzerinde oluşturulmuş bir grafiksel ara yüz olan nntool, yapay sinir ağı oluşturma, eğitme ve test etme işlemlerini gerçekleştirmektedir. MATLAB komut kısmına nntool komutunu vererek ara yüze erişmek mümkündür.

3.5. YAZILIM TANIMLI AĞ DENETLEYİCİSİNE BAĞLI PAKET TRAFİK GÖZETLEME UYGULAMA VE OPTİMİZASYONU

Yazılım tanımlı ağ denetleyicisine bağlı olarak paket trafiği gözetlenmesi üzerine uygulama geliştirirken ilk olarak yapılan şey bilgisayara sanal makine yüklenmesidir. Sanal makine yüklendikten sonra ise “Floodlight VM” indirilerek bu sanal makine üzerine kurulmaktadır.

Bu kurulan Floodlight sanal makinesi içerisinde Floodlight v1.0, eclipse, mininet v2.2.0, open vSwitch v3.2.1, wireshark w/openflow barındırmaktadır.

Sanal makine kurulduktan sonra giriş yapmak için “floodlight” hem kullanıcı adı hem de parola olarak kullanılmaktadır.

Sanal makine içerisinde bulunan floodlight projesini eclipse üzerinde çalıştırmadan önce terminal üzerinde aşağıda yer alan komutlar yazılmaktadır:

git clone git://github.com/floodlight/floodlight.git

cd floodlight/

ant

java -jar target/floodlight.jar

Bu komutları yazarak daha üst versiyonda bir floodlight projesine erişim sağlanabilmektedir. Bunu yapmaktaki maksat ise hazır olarak gelen floodlight projesi düşük versiyonda olduğu için

ihtiyaç duyulan istatistik paketlerini içerisinde barındırmamaktadır. Bu komutlar sayesinde daha üst seviye floodlight projesi indirilerek istatistik paketlerine erişim sağlanmaktadır. İstatistik paketlerine erişimin sağlanmasıyla beraber projede ihtiyaç duyulan istatistik verileri için gerekli alt yapı oluşturulmaktadır.

Projede paket trafiği gözetlemek için gerekli olan veriler switch, port numarası, alınan ve iletilen paketlerin miktarı, alınan ve iletilen paketlerin byte cinsinden değeri, düşmeler, çarpışmalar ve bunların gerçekleştiği süre bilgileridir. Zaman saniye cinsinden alınarak daha ayrıntılı takip yapılması sağlanmıştır.

Elde etmek istenilen veriler ilk olarak deneme maksatlı mininet üzerinde hazır bulunan küçük bir topoloji üzerinde denenmiştir.

“sudo mn –controller=remote, ip=10.0.2.15 port=6653”

Yukarıda yer alan komut sayesinde uzak denetleyiciyle bağlantı sağlanmaktadır. Bunun çalışması içinde eclipse veya terminal üzerinden floodlight projesini çalıştırmak gerekmektedir. Küçük topoloji üzerinde yapılan denemede denetleyiciyle bağlantının kurulmasında ve projenin çalışmasında olumlu sonuç aldıktan sonra text dosyasına gerekli istatistik verilerinin yazılması sağlanmıştır.

Doğruluğu artırmak adına çeşitli topolojiler üzerinden istatistik verileri elde edilmiştir. Bunun içinde farklı sayıda switch ve hostlardan oluşan yeni topolojiler oluşturulmuştur. Bunlar ise python ile terminal üzerinde “nano” komutunu yazarak oluşturulmuştur. Topolojiler şu şekildedir:

1 adet tree topoloji (9 switch-64 host)

3 adet circle topoloji (10 switch-36 host, 20 switch-57 host, 30 switch-87 host)

1 adet linear topoloji (45 switch-45 host)

Oluşturulan topolojilerin her biri yirmi yedi saniye çalıştırılarak beş ayrı veri seti elde edilmiştir. Sürenin yirmi yedi saniye seçilmesi ile binlerce satır veri elde edilmiştir ve bu da verilerin işlenmesi sürecinde yeterli görülmüştür. Sürenin arttırılması ile veri miktarında da aşırı bir artış

gerçekleştiği gözlemlendiğinden ve bu da verilerin işlenmesi esnasında problem teşkil edeceğinden yirmi yedi saniyelik süre tercih edilmiştir.

Elde edilen veri setlerinin işlenmesi aşamasında ise MATLAB kullanılmıştır. MATLAB matris işlemleri kullanılarak veri setlerinden yirmi yedi saniyelik veriler alınmıştır. Bu verilerinde switch, port, alınan ve iletilen paket miktarı ve süre sütunları seçilip diğer sütunlar göz ardı edilmiştir.

Projenin uygulama kısmı ise yapay zekâ üzerine kurulmuştur. Burada amaç belirli bir anda belirli switch ve porttan ne kadar paket alınıp iletildiğini tahmin etmeye çalışmaktır. Bu tahmin kısmı ise yapay zekâ yöntemlerinden olan yapay sinir ağları ile gerçekleştirilmiştir. MATLAB tool olan "nntool" ile verilerin yapay sinir ağları üzerinde işlenmesi sağlanmıştır. MATLAB matris işlemleri yardımıyla veri setlerinin yüzde seksenlik kısmı eğitim, yüzde yirmilik kısmı ise test için ayrılmıştır. Veri setlerinin switch, port ve süre sütunları input verisi olarak, alınan paket ve iletilen paket sütunları ise çıktı verisi olarak ayarlanmıştır. nntool üzerinde gerekli kısımlar seçilerek ağlar oluşturulmuştur. İlk oluşturulan ağ üç girdisi iki çıktısı olan bir ağ olarak oluşturulmuştur. Fakat elde edilen tahmin başarısı düşük olduğundan ağlar üç girişli bir çıkışlı olmak üzere tasarlanmıştır. Böylece her bir topoloji için alınan paket ve iletilen paketlere göre ayrı ağlar oluşturulmuştur. Çok karmaşık veri setlerine sahip olunmadığından on nöronluk iki katmanlı yapay sinir ağları yeterli olmuştur.

Beş farklı topoloji için oluşturulan yapay sinir ağlarının herhangi bir andaki tahminlerinin doğruluğu kıyaslanmıştır. Doğrulukları ise MAPE ve R^2 fonksiyonları kullanılarak hesaplanmıştır.

Uygulamanın son kısmı ise optimizasyondan oluşmaktadır. Optimize edilecek problem ise yani optimum sonuç olarak elde edilmek istenen değer ise topolojide hangi güzergahta en fazla yoğunluğun olduğunun tespit edilmesidir. Bunun için ise dört farklı şekilde optimizasyon gerçekleştirilmiş ve sonuçları karşılaştırılmıştır. Bunlardan ilki geleneksel arama metodu olan doğrusal arama, ikincisi yapay zekâ optimizasyon tekniklerinden olan tabu arama, üçüncüsü değiştirilmiş tabu arama ve dördüncüsü ise tavlama benzetim ve tabu arama metotlarının karıştırılması ile oluşturulmuş optimizasyonlardır. Bu dört farklı optimizasyon yöntemi sonucunda paket trafiğinin en yoğun olduğu güzergâh tespit edilmiştir.

```
//sonuç değerini sıfırla;
//başlangıç switchini belirle;
//switchin port değerlerini al;
//sırayla tüm switchleri dolaş;
//port değerlerini al;
//alınan değerleri topla;
//eğer toplam sonuçtan büyükse;
//yeni sonuç değeri toplam olur;
//değilse
//sonuç değeri değişmez;
```

Şekil 3.1: Geleneksel Arama Metodu-Doğrusal Arama Algoritması

Değiştirilmiş tabu arama algoritması diye bahsedilen aslında tabu arama algoritmasından tabu listesi çıkartılarak elde edilen bir algoritmadır. Tabu arama algoritması incelendiğinde tabu listesinin bu problem için gereksiz olduğu ve optimizasyonu yavaşlattığı gözlemlenmiştir. Liste algoritmadan çıkarıldığında performansta artış gözlemlenmiştir. Buradan da yapay zekâ optimizasyon tekniklerinin her ne kadar elverişli olsalar da probleme özgü olmalarından dolayı her zaman başarı gösteremedikleri gözlemlenmiştir. Başka bir problemde örneğin gezgin satıcı probleminde gayet elverişli olan bu algoritma bu problemde performans düşüklüğüne sebebiyet vermektedir.

```
//tabu listesi oluştur;

//birinci switchin ilk portundaki değeri al;

//ikinci switchin ilk portundaki değeri al;

//değerleri topla;

//başlangıç çözümünü toplama eşitle;

//en iyi çözüme başlangıç çözümünü ata;

//birinci switch ile diğer switcler arasındadi portlardaki
değerleri topla;

//toplamı çözüme ata;

//eğer çözüm başlangıç çözümünden küçükse;

//çözümü tabu listesine ekle;

//değilse;

//çözümü en iyi çözüme ata;
```

Şekil 3.2: Tabu Arama Algoritması

İki algoritmanın karışımından meydana gelen algoritmada en iyi performans elde edilmiştir. Algoritmanın ilk kısmı tabu arama algoritmasından gelmektedir. Tavlama benzetim algoritmasında rastgele seçilen başlangıç çözümü yerine, tabu arama algoritmasındaki ilk değeri başlangıç çözümü olarak atamak tercih edilmiştir. Daha sonraki kısımlarda tabu aramaya yer verilmemiştir. Geri kalan kısımda komşu çözümler oluşturulmuş olup amaç fonksiyonundaki değişim gözlemlenmiştir. Yalnız bu kısımda da bir değişikliğe gidilmiştir. Normalde amaç fonksiyonunda azalma söz konusu ise, çözüm mevcut çözüm olurken bu algoritmada tam tersi göz önünde bulundurulmuştur. Amaç fonksiyonunda artış olduğu takdirde çözüm mevcut çözüm olmaktadır.

```
//birinci switchin ilk portundaki değeri al;

//alınan değeri başlangıç çözümüne ata;

//eğer birinci switchin diğer portlarındaki değerler başlangıç
çözümünden büyükse;

//değeri en iyi çözüm1'e ata ve;

//ikinci switche geç;

//ikinci switchin portlarındaki en büyük değeri bul;

//değeri en iyi çözüm2' ye ata;

//en iyi çözüm1 ve en iyi çözüm2' yi topla;

//sonucu toplama eşitle;
```

Şekil 3.3: Tabu Arama Algoritması ve Tavlama Benzetim Algoritması Karışımı

4. BULGULAR

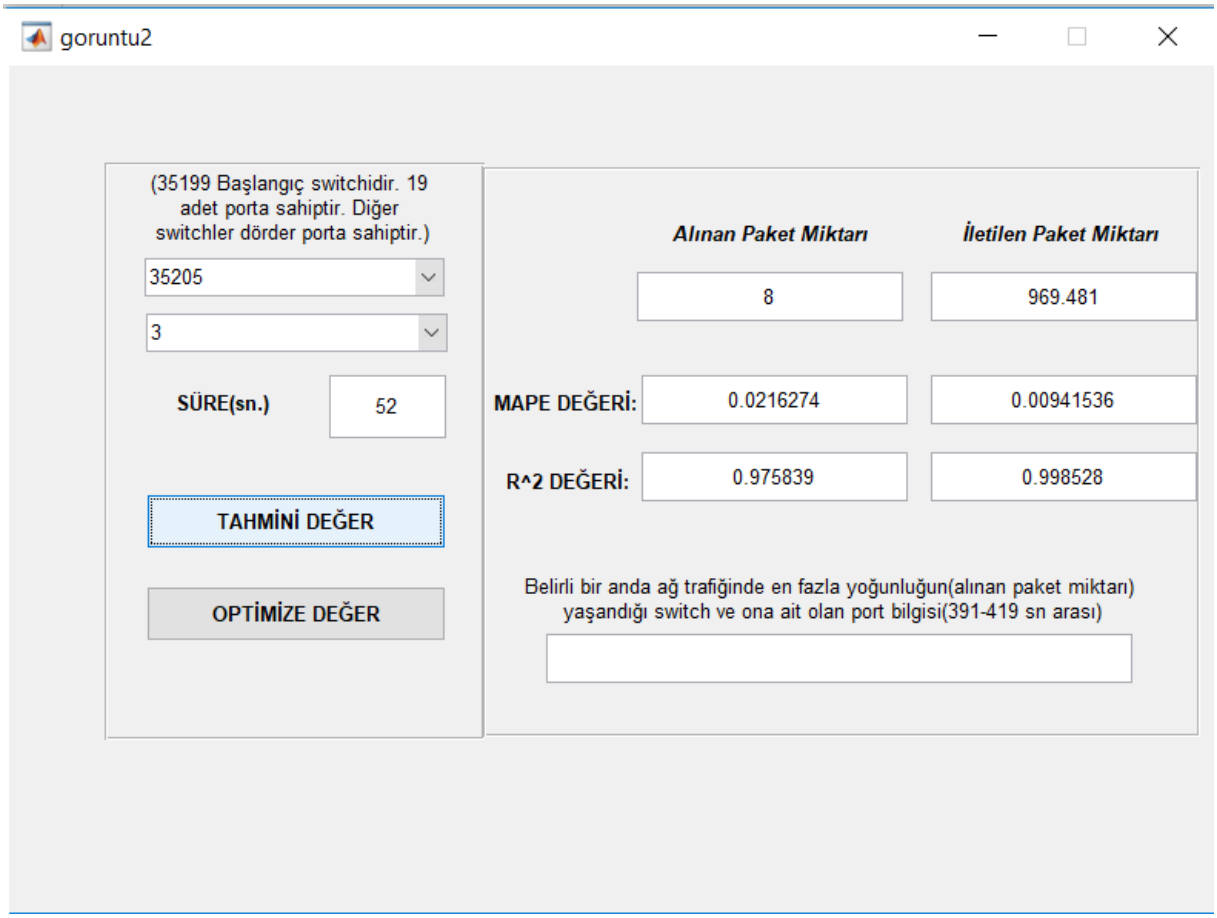
Bu çalışmada sahip olunan bulgular Intel Core i7-7700HQ CPU 2.80GHz, 16 GB RAM, Windows 10/ 64 bit işletim sistemine sahip laptop üzerinde yapılan çalışmalar sonucunda elde edilmiştir. Yapılan çalışma birbirine bağlı üç kısımdan oluşmaktadır. İlk kısım yazılım tanımlı ağ sayesinde ağ topolojileri hakkındaki istatistik verilerini elde edebilme, ikinci kısım elde edilen bu veriler ışığında yapay sinir ağları kullanılarak herhangi bir andaki paket trafiğini tahmin etme ve son olarak üçüncü kısım ise yapay sinir ağları sonuçlarına göre doğruluğu en yüksek olan topoloji üzerinde trafik yoğunluğunu optimize etmektir.

İlk kısım olan yazılım tanımlı ağ kısmında python yardımıyla elde edilen beş farklı topoloji uygulama üzerinde çalıştırılarak denetleyicinin işlerliği gözlemlenmiştir. Denetleyicinin farklı türde topolojiler üzerinde etkin bir şekilde çalıştığı tespit edilmiştir. İlk kısım sonucunda beş ayrı veri seti elde edilmiştir.

Tablo 4.1: Python yardımıyla oluşturulan beş ayrı topolojiye ait bilgiler

TOPOLOJİ TÜRÜ	SWİTCH	HOST
Doğrusal	45	45
Ağaç	9	64
Dairesel	10	36
Dairesel	20	57
Dairesel	30	87

İkinci kısım olan yapay zekâ kısmında yapay sinir ağları kullanılarak bir tahmin uygulaması gerçekleştirilmiştir.



Şekil 4.1: Alınan ve iletilen paket miktarlarına göre yapılan tahmin ve doğruluk oranları

Tahminlerin doğruluk oranları iki farklı fonksiyona göre hesaplanmıştır ve elde edilen sonuçlara göre tahmin başarısı en yüksek olan topolojinin yirmi switch ve elli yedi hostdan oluşan dairesel topoloji olduğu gözlemlenmiştir. Doğruluğu en yüksek olan ikinci topolojinin ise ağaç topolojisi olduğu tespit edilmiştir.

Tablo 4.2: 10 switch-36 host topolojisi için doğruluk oranları

	PAKET TÜRÜ	MAPE	R^2
10 switch-36 host	Alınan (rx)	0.0473	0.9989
	İletilen (tx)	0.0554	0.7559

Tablo 4.3: 20 switch-57 host topolojisi için doğruluk oranları

	PAKET TÜRÜ	MAPE	R^2
20 switch-57 host	Alınan (rx)	**0.0216**	**0.9758**
	İletilen (tx)	**0.0094**	**0.9985**

Tablo 4.4: 30 switch-87 host topolojisi için doğruluk oranları

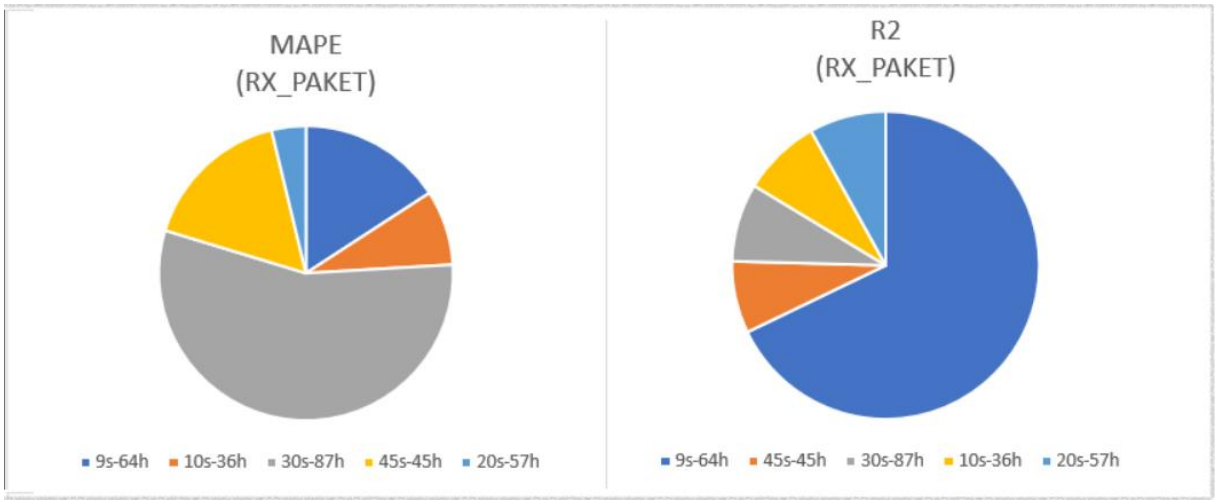
	PAKET TÜRÜ	MAPE	R^2
30 switch-87 host	Alınan (rx)	0.3190	0.9951
	İletilen (tx)	0.0559	0.9851

Tablo 4.5: 45 switch-45 host topolojisi için doğruluk oranları

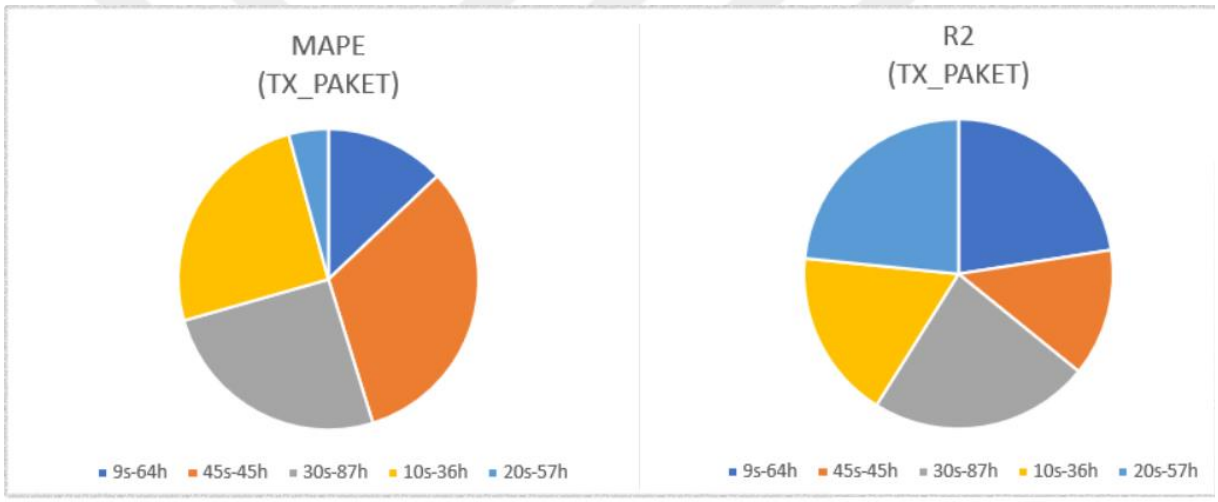
	PAKET TÜRÜ	MAPE	R^2
45 switch-45 host	Alınan (rx)	0.0949	0.9154
	İletilen (tx)	0.0711	0.5685

Tablo 4.6: 9 switch-64 host topolojisi için doğruluk oranları

	PAKET TÜRÜ	MAPE	R^2
9 switch-64 host	Alınan (rx)	**0.0908**	**0.9675**
	İletilen (tx)	**0.0285**	**0.9614**

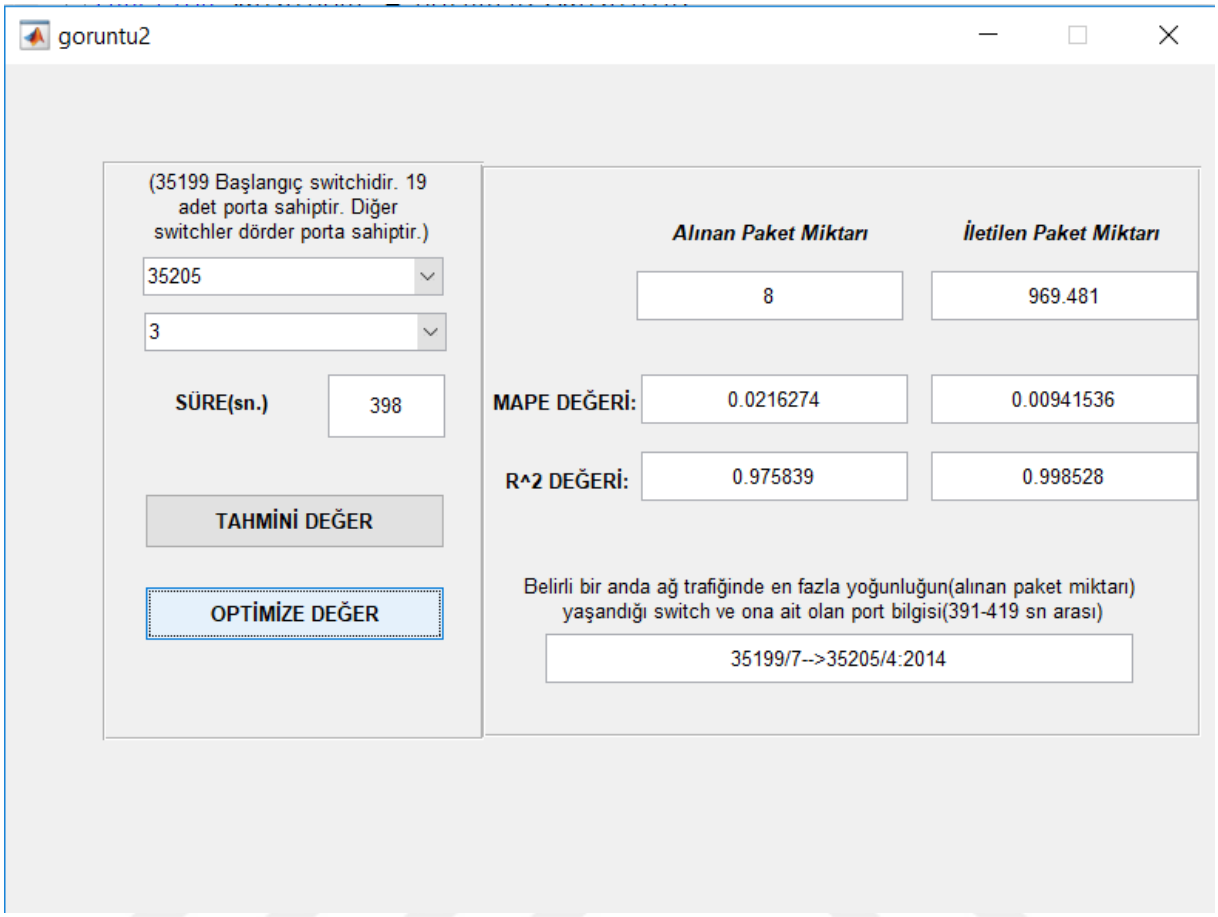


Şekil 4.2: Alınan paket miktarına göre yapılan tahminlerin doğruluk oranları



Şekil 4.3: İletilen paket miktarına göre yapılan tahminlerin doğruluk oranları

Üçüncü ve son olan kısım ise optimizasyon kısmından oluşmaktadır. İkinci kısımda yapılan tahmin işleminin doğruluk oranına göre seçilen en iyi iki topoloji üzerinde paket yoğunluğu en fazla olan güzergahı bulma problemi optimize edilmiştir.



Şekil 4.4: 398. Saniye de ağda en fazla paket akışının olduğu güzergâhın bulunması

Yukarıdaki şekil üzerinden açıklamak gerekirse ağ 398. saniyede optimize edildiğinde, 35199 numaralı switchin 7 numaralı portu ile 35205 numaralı switchin 4 numaralı portu arasındaki güzergahta alınan paket miktarına göre toplamda 2014 adet paket akışı gerçekleşmiştir. Bu en yoğun güzergahı tespit ederken de optimizasyon işleminin gereği olarak en kısa süre de bu sonuç elde edilmeye çalışıldı. Bu amaç doğrultusunda optimizasyon işlemi sırasında dört ayrı optimizasyon yöntemi denenmiştir ve bunların performansı kıyaslanmıştır. Bu yöntemler tabu arama, doğrusal arama, değiştirilmiş tabu arama ve tavlama benzetim-tabu arama karışımı olan fonksiyonlardır. Sonuçlar değerlendirildiğinde en başarılı performansın iki fonksiyonun karışımı olan tavlama benzetim-tabu arama karışımı olan fonksiyon olduğu gözlemlenmiştir. Fonksiyonların işlerliğini test etmek açısından en iyi ikinci doğruluğa sahip olan ağaç topolojisinde de bu fonksiyonlar denenmiştir. Sonuç olarak yine en iyi performansın tavlama benzetim-tabu arama karışımı olan fonksiyona ait olduğu gözlemlenmiştir.

Tablo 4.7: Yirmi switch ve elli yedi hostdan oluşan topolojiye ait optimizasyon performansı

SÜRE	Doğrusal Arama (Geleneksel arama metodu)	Tabu Arama (Yapay zekâ optimizasyon tekniği)	Değiştirilmiş Tabu Arama	Tabu Arama ve Tavlama Benzetim Karışımı
395.sn	0.006152	0.019727	0.017668	**0.000895**
400.sn	0.005415	0.018567	0.016828	**0.000585**
405.sn	0.005975	0.017276	0.014780	**0.000631**
410.sn	0.006417	0.019628	0.018047	**0.000556**
415.sn	0.005671	0.017062	0.015701	**0.000548**

Tablo 4.8: Dokuz switch ve altmış dört hostdan oluşan ağaç topolojisine ait optimizasyon performansı

SÜRE	Doğrusal Arama (Geleneksel arama metodu)	Tabu Arama (Yapay zekâ optimizasyon tekniği)	Değiştirilmiş Tabu Arama	Tabu Arama ve Tavlama Benzetim Karışımı
4.sn	0.007210	0.020124	0.011993	**0.001496**
8.sn	0.004278	0.013395	0.008320	**0.001482**
12.sn	0.003789	0.012859	0.008282	**0.002117**
16.sn	0.003978	0.013196	0.008277	**0.001883**
20.sn	0.004648	0.012556	0.008566	**0.001975**

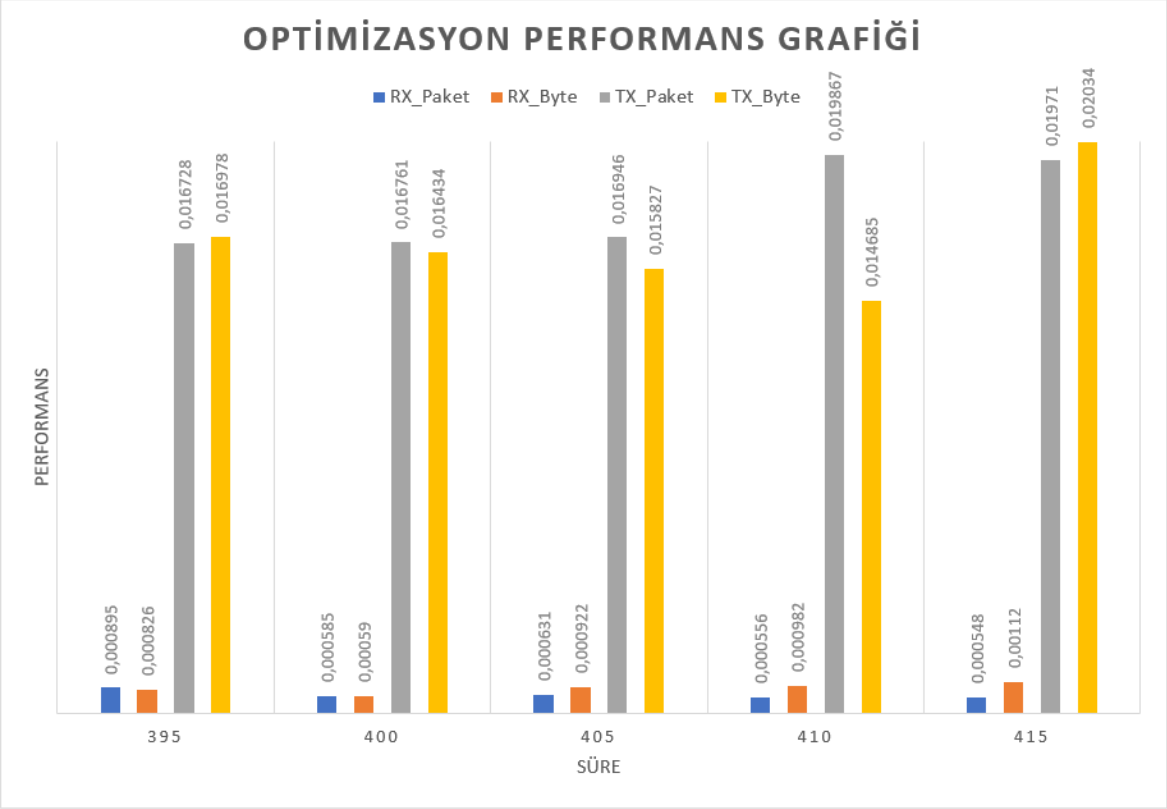
Alınan paket miktarı göz önünde bulundurularak yapılan optimizasyon işlemlerinde hangi algoritmanın daha iyi performans gösterdiği bilgisine ulaşılmıştır. Buradan sonuç olarak

tavlama benzetim ve tabu arama algoritmalarının karışımı olan algoritmanın en iyi performans gösteren algoritma olduğu görüldükten sonra bu algoritma üzerinde farklı değerler ile denemeler yapılmıştır ve performans değerleri incelenmiştir. Şöyle ki portlar arasında alınan paket miktarına göre yoğunluğun yani alınan paket sayısının en fazla olduğu güzergâh tespit edilmişti. Diğer denemelerde ise portlar arasında iletilen paket miktarına, alınan byte miktarına ve iletilen byte miktarına göre de yoğunluğun en fazla olduğu güzergahlar tespit edilmiş ve sonuçlar karşılaştırılmıştır. Bu denemeler karışım algoritması üzerinden beş farklı zamanda gerçekleştirilmiştir. Yapılan denemeler sonucunda güzergahların ve performansların değiştiği gözlemlenmiştir. Bunun sebebini ise alınan ve iletilen paketler üzerinde yapılan denemeleri baz alırsak porta gelen ve porttan çıkan paket miktarının farklılık göstermesi olarak açıklarız. Diğer bir senaryo olarak alınan paket ve alınan byte miktarlarına göre yapılan denemeler sonucunda farklı güzergahlar ve farklı performans sonucu elde edilmesinin sebebi ise her bir paketin taşıdığı bilginin byte cinsinden aynı büyüklükte olmamasıdır.

Tablo 4.9: Yirmi switch ve elli yedi hostdan oluşan daire topolojisine ait dört ayrı kritere göre hesaplanmış optimizasyon performans değerleri

SÜRE	RX_PAKET	RX_BYTE	TX_PAKET	TX_BYTE
395.sn	0.000895	0.000826	0.016728	0.016978
400.sn	0.000585	0.000590	0.016761	0.016434
405.sn	0.000631	0.000922	0.016946	0.015827
410.sn	0.000556	0.000982	0.019867	0.014685
415.sn	0.000548	0.001120	0.019710	0.020340

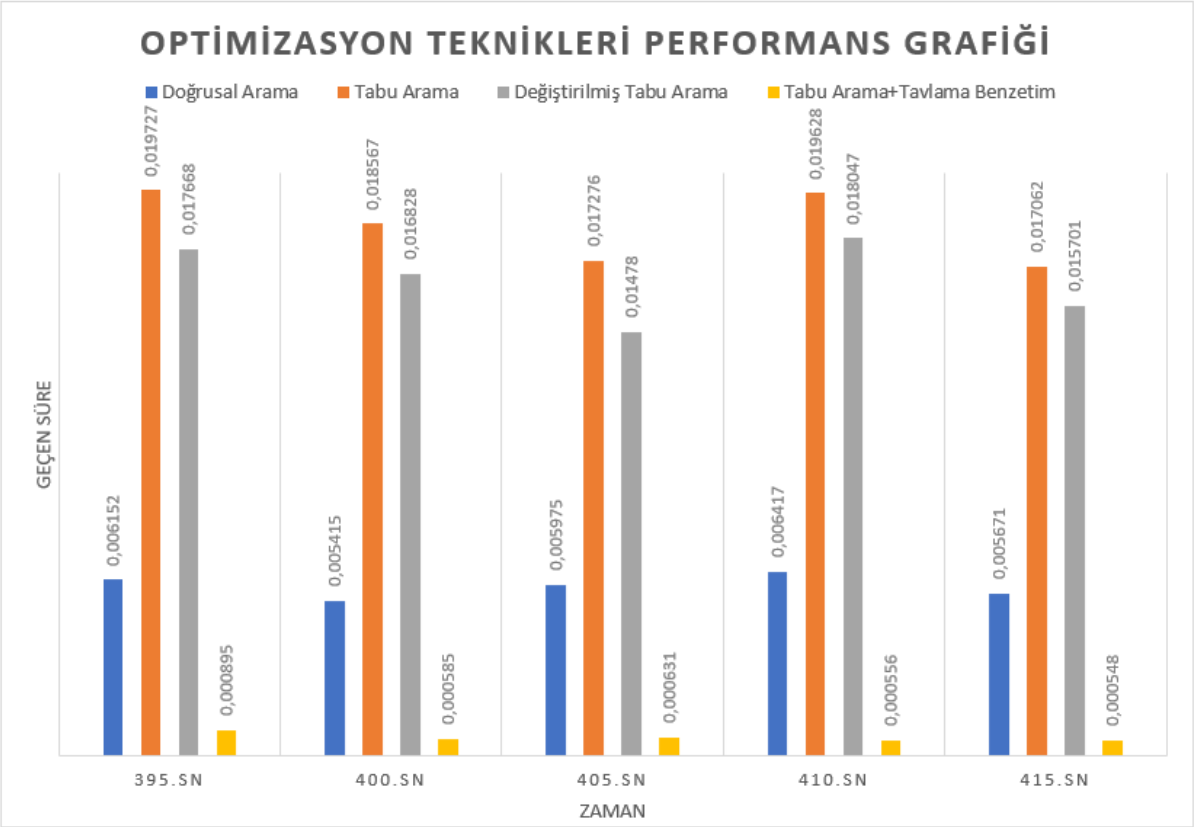
Optimizasyondaki amaç problemin çözümüne giden en doğru yolu en kısa süre gitmektir. Bu amaçla dört ayrı kritere göre hesaplanmış optimizasyon performans değerlerinden en iyi performansın alınan paket miktarına göre yapılan optimizasyon işleminden olduğu sonucuna varılmaktadır.



Şekil 4.5: Yirmi switch ve elli yedi hostdan oluşan daire topolojisine ait dört ayrı kritere göre hesaplanmış optimizasyon performans grafiği

5. TARTIŞMA VE SONUÇ

Büyük ağ topolojilerinde yer alan anlamlı ve değerli veri geleneksel ağ yönetimi yaklaşımıyla işlenemediğinden, bu tezde yazılım tanımlı ağ ile bu verilerin elde edilmesi ve işlenmesi üzerine çalışmalar yapılmıştır. Ağ içinde birden çok düğüme paket iletimi gerçekleştiği için bu da yoğun bir paket trafik akışına sebebiyet vermektedir. Bu esnada yazılım tanımlı ağ denetleyicisi kullanılarak ağ trafik akış bilgileri elde edilmiştir. Bu bilgiler uygulamada veri seti olarak kullanılmıştır. Ağda trafik yoğunluğuna sebebiyet veren paketler, paket sayısı ve byte cinsinden hesaplanmıştır. En yoğun paket trafiğinin yaşandığı güzergâh alınan paket miktarı, iletilen paket miktarı, alınan byte ve iletilen byte miktarlarına göre ayrı ayrı bulunmuştur. Yapay zekâ optimizasyon teknikleri bu dört ayrı kriterin her birine ayrı ayrı uygulanarak en yoğun paket trafiğinin yaşandığı güzergâh başarılı bir şekilde belirlenmiştir. Bu dört kritere göre elde edilen sonuçlar incelendiğinde alınan paket miktarına göre yapılan optimizasyon işleminin performans açısından daha başarılı olduğu görülmüştür. Optimizasyon teknikleri kısmında ise dört farklı optimizasyon tekniği kullanılarak sonuçları değerlendirilmiştir. Bu tekniklerden birisi doğrusal arama metodu yani geleneksel yöntem iken geri kalan üç tanesi ise yapay zekâ optimizasyon tekniklerinden olan tabu arama ve tavlama benzetim algoritması türevleridir. Sonuçlar incelendiğinde geleneksel arama metodu tabu arama metodundan daha iyi performans göstermektedir. Buradan da yapay zekâ optimizasyon tekniklerinin her problem için en iyi çözüm olmadığı sonucuna ulaşılmaktadır. Fakat tabu arama algoritması ve tavlama benzetim algoritmasının bazı özellikleri alınıp karma bir metot geliştirilerek sonuç olarak doğrusal arama metodundan daha iyi bir performans elde edilmiştir.



Şekil 5.1: Yirmi switch ve elli yedi hostdan oluşan daire topolojisine ait dört ayrı optimizasyon tekniğine göre hesaplanmış optimizasyon teknikleri performans grafiği

Son olarak bu çalışma kapsamında yapay sinir ağları kullanılarak ağın herhangi bir anında yaşadığı paket trafik yoğunluğu tahmin edilmiştir. Bu kısım ise alınan paket miktarı ve iletilen paket miktarı göz önünde bulundurularak gerçekleştirilmiştir. Tahminler neticesinde yüksek doğruluk elde edilmiştir.

Yapılan bu çalışma geliştirilmeye elverişlidir. Şöyle ki; ileri aşamalarda yoğunluğu tespit edilen güzergâh üzerinde yazılım tanımlı ağlar kullanılarak trafik yoğunluğunu azaltma gerçekleştirilebilir.

Diğer bir geliştirilmeye açık olan fikir ise optimizasyon üzerinedir. Tezin optimizasyon kısmı sadece belirli zaman aralıklarındaki veriler üzerinde optimizasyon işlemini gerçekleştirmektedir. Yani veri setini oluşturan zaman aralığında en yoğun güzergahı bulmaktadır. Tez ileriki aşamalarda geliştirilerek yapay sinir ağları kısmında yapılan tahmin işlemi gibi herhangi bir anda ağdaki en yoğun güzergâh tahmin edilebilir.

KAYNAKLAR

[1]. CISCO, *Türkçe CCNA Eğitim Notları,* Academytech.

[2]. Yazar, S., 2013, *Ağ Anahtarlarında Openflow Protokolü ve POX Denetleyicisi Kullanımı,* Yüksek Lisans, Trakya Üniversitesi.

[3]. Bonaventure, O., 2011, *Computer Networking: Principles, Protocols and Practice*, The Saylor Foundation.

[4]. Karaca, C., *Ders Notları: Bilgisayar Ağ Sistemleri,* Aksaray Üniversitesi.

[5]. *What does Network Topology mean?*, https://www.techopedia.com/definition/5538/network-topology.

[6]. Simoneau, P., *The OSI Model: Understanding the Seven Layers of Computer Networks,* Global Knowledge.

[7]. Aydilek, İ.B., 2006, *Web Uygulama ve Sunucularının Performans Analizi,* Yüksek Lisans, Selçuk Üniversitesi.

[8]. *Computer Networks: Types Of Network Topology*, https://www.studytonight.com/computer-networks/network-topology-types.

[9]. *Computer Network Topology*, https://www.tutorialspoint.com/data_communication_computer_network/computer_network_topologies.htm.

[10]. İTÜBİDB, 2013, *TCP/IP Protokolü,* İstanbul Teknik Üniversitesi, http://bidb.itu.edu.tr/eskiler/seyirdefteri/blog/2013/09/07/tcp-ip-protokol%C3%BC.

[11]. *Network Devices*, https://www.geeksforgeeks.org/network-devices-hub-repeater-bridge-switch-router-gateways/.

[12]. *Network Devices*, http://www.certiology.com/computing/computer-networking/network-devices.html.

[13]. Ataç, S., *Ağ Temelleri Ders Notu: Bilgisayar Ağı Nedir?,* Dokuz Eylül Üniversitesi Bergama Meslek Yüksek Okulu.

[14]. Akbaş, M.F. ve Karaaslan, E. ve Güngör, C.,2016, A Preliminary Survey on the Security of Software Defined Networks, *International Journal Of Applied Mathematics, Electronics and Computers*, 4 (Special Issue), 184-189.

[15]. Sezer, S. ve Scott-Hayward, S. ve Kaur, P. ve Fraser, B. ve Lake, D. ve Viljoen, N. ve Miller, M. ve Rao, N., 2013, Are We Ready for SDN?, *IEEE Comminications Magazine*, 51 (7), 36-43.

[16]. Xia, W. ve Wen, Y. ve Foh, C. ve Niyato, D. ve Xie, H., 2015, A Survey on Software Defined Network, *IEEE Communication Surveys & Tutorials*, 17 (1).

[17]. Kutay, M. ve Ercan, T., 2016, Yazılım Tanımlı Yerleşke Ağlarının Geliştirilmesi için Bir Derleme, *Selçuk Üniversitesi Mühendislik, Bilim ve Teknoloji Dergisi,* 4, 155-164.

[18]. Niyaz, Q. ve Sun, W. ve Alam, M., 2015, Impact on SDN Powered Network Services under Adversarial Attacks, *The 2015 International Conference on Soft Computing and Software Engineering (SCSE 2015), Procedia Computer Science*, 62, 228 – 235.

[19]. Hogg, S., 2014, SDN Security Attack Vectors and SDN Hardening, *Network World*

[20]. Hamad, D.J., 2015, *Network Management Functionalities of Openflow v1.3 in Software Defined Networking (SDN)Enviroment,* Yüksek Lisans, Kahramanmaraş Sütçü İmam Üniversitesi.

[21]. Alparslan, Ö., 2016, *Veri Merkezlerinde Büyük Boyutlu Verilerin Taşınması Sürecinde Yazılım Tanımlı Ağların (SDN) Kullanımı,* Yüksek Lisans, Gazi Üniversitesi.

[22]. McKeown, N. ve Anderson, T. ve Balakrishnan, H. ve Parulkar, G. ve Peterson, L. ve Rexford, J. ve Shenker, S. ve Turner, J., 2008, *OpenFlow: Enabling Innovation in Campus Networks.*

[23]. IXIA, 2014, *Black Book: SDN/Openflow*, 10.

[24]. *Akış Tabloları,* http://www.netoburus.com/sdn/sdn-4-akis-tablolari.html/.

[25]. Open Networking Foundation, 2013, *OpenFlow Switch Specification,* version 1.3.2.

[26]. Nacshon, L. ve Puzis, R. ve Zilberman, P., *Floware: Balanced Flow Monitoring in Software Defined Networks.*

[27]. Üstkan, S., 2007, *Uzman Sistemler-Genel,* Yönlendirilmiş Çalışma, Sakarya Üniversitesi.

[28]. Babalık, A. ve Güler, İ., 2007, Boğaz Enfeksiyonlarının Teşhis ve Tedavisinde Uzman Sistem Kullanımı, *Selçuk Üniversitesi Teknik Bilimler Meslek Yüksekokulu Teknik-Online Dergi*, 6.

[29]. Bilge, U., *Tıpta Yapay Zekâ ve Uzman Sistemler,* Biyoistatistik ve Tıp Bilişimi AD, Akdeniz Üniversitesi Tıp Fakültesi, Antalya.

[30]. Erdal, C., 2008, *Bulanık Mantık ve Firmaların Başarı Kriterlerinin Tanımlanarak Bulanık Mantık ile Ölçülmesinin Bir Uygulaması,* Yüksek Lisans, Kırıkkale Üniversitesi.

[31]. Altaş, İ.H., 1999, Bulanık Mantık: Bulanıklılık Kavramı, *Enerji, Elektrik, Elektromekanik-3e*, 62, 80-85.

[32]. Tiryaki, A. ve Kazan, R., Bulaşık Makinesinin Bulanık Mantık ile Modellenmesi, *Mühendis ve Makine,* 48 (565).

[33]. Haykin, S., 2009, *Neural Networks and Learning Machines,* Üçüncü Basım, Pearson, New Jersey, ISBN: 978-0-13-147139-9.

[34]. Staub, S. ve Karaman, E. ve Kaya, S. ve Karapınar, H. ve Güven, E., 2015, Artificial Neural Network and Agility, *World Conference on Technology, Innovation and Entrepreneurship,* Procedia - Social and Behavioral Sciences 195 (2015) 1477 – 1485.

[35]. Taşhan, B., 2017, *Road Lane Detection System With Convolutional Neural Network,* Yüksek Lisans, Bahçeşehir Üniversitesi.

[36]. Öztemel, E., 2012, *Yapay Sinir Ağları,* 3. Basım, Papatya, İstanbul, ISBN: 978-975-6797-39-6.

[37]. *Yapay Sinir Ağları Nedir?*, http://kod5.org/yapay-sinir-aglari-ysa-nedir/.

[38]. Şengöz, N., *Yapay Sinir Ağları,* http://www.derinogrenme.com/2017/03/04/yapay-sinir-aglari/.

[39]. Öğücü, O., 2006, *Yapay Sinir Ağları ile Sistem Tanıma,* Yüksek Lisans, İstanbul Teknik Üniversitesi.

[40]. *What is MAPE?*, https://content.nexosis.com/blog/what-is-mape.

[41]. *MSE, RMSE, MAE, MAPE ve Diğer Metrikler*, https://veribilimcisi.com/2017/07/14/mse-rmse-mae-mape-metrikleri-nedir/.

[42]. Tonta, Y., *Regresyon Analizi,* Hacettepe Üniversitesi.

[43]. *R-Squared*, https://www.investopedia.com/terms/r/r-squared.asp.

[44]. Gendreau, M. ve Potvin, J., 2010, *Handbook of Metaheuristics*, İkinci Basım, Springer Science+Business Media.

[45]. Aladağ, Ç.H., 2009, *Yapay Sinir Ağlarının Mimari Seçimi İçin Tabu Arama Algoritması*, Doktora, Hacettepe Üniversitesi.

[46]. Michalska, M. ve Zufferey, N. ve Mattevelli, M., 2016, Tabu Search For Partitioning Dynamic Dataflow Programs, *The International Conference on Computational Science,* Procedia Computer Science, 80, 1577-1588.

[47]. Gürbüz, Ö., 2015, *Tabu Arama Algoritmasının Kuyruk Problemine Uygulanması,* Yüksek Lisans, Hacettepe Üniversitesi.

[48]. Cesaret, B., 2010, *A Tabu Search Algorithm For Order Acceptance And Scheduling Problem,* Yüksek Lisans, Koç Üniversitesi.

[49]. Glover, F. ve Laguna, M. ve Martı, R., 2008, *Tabu Search,* Colorado Boulder Üniversitesi ve Valencia Üniversitesi.

[50]. Arıkan, M. ve Erol, S.,2006, Bir Tabu Arama Uygulaması: Esnek İmalat Sistemleri'nde Parça Seçimi ve Takım Magazini Yerleşimi, *Gazi Üniversitesi, Mühendislik Mimarlık Fakültesi Dergisi,* 21 (2), 221-227.

[51]. Çelik, C.,2008, *İşlerin Bölünebilir Olduğu Paralel Makine Çizelgeleme Problemi İçin Tabu Arama Yöntemi,* Yüksek Lisans, Eskişehir Osmangazi Üniversitesi.

[52]. Aladağ, Ç.H. ve Hocaoğlu, G., 2007, A Tabu Search Algorithm to Solve a Course Timetabling Problem, *Hacettepe Journal of Mathematics and Statistics*, 36 (1), 53-64.

[53]. Glover, F., 1990, *Tabu Search: A Tutorial*, Colorado, 74-94.

[54]. Kalınlı, A., 2003, Elman Ağının Benzetilmiş Tavlama Algoritması Kullanarak Eğitilmesi, *Erciyes Üniversitesi Fen Bilimleri Enstitüsü Dergisi,* 19 (1-2), 28-37.

[55]. Çakır, B., 2006, *Stokastik İşlem Zamanlı Montaj Hattı Dengeleme İçin Tavlama Benzetimi Algoritması,* Yüksek Lisans, Gazi Üniversitesi.

[56]. Kılıçaslan, K., *Isıl İşlem Algoritması,* Doktora Programı, Trakya Üniversitesi.

[57]. *Floodlight*, https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview.

[58]. *Matlab*, https://www.mathworks.com/discovery/what-is-matlab.html.

[59]. *Eclipse*, https://www.tutorialspoint.com/eclipse/eclipse_overview.html.

[60]. *Mininet*, http://mininet.org/overview/.

[61]. *FloodlightProject*, https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343548/The+Controller.

[62]. *WireShark*, https://www.wireshark.org.

EKLER

EK 1. Ağaç topolojisine ait networkten elde edilen veri setinden örnek

Yazılım tanımlı ağ denetleyicisine bağlı olarak çalıştırılan ağlardan onar sütunluk veri setleri elde edilmiştir. Bu veri setlerini oluşturan özellikler switch numarası, port numarası, alınan paket miktarı, iletilen paket miktarı, alınan byte miktarı, iletilen byte miktarı, alınan paketlerde meydana gelen düşmeler, iletilen paketlerde meydana gelen düşmeler, çarpışmalar ve süredir.

switch no	port no	rx_paket	tx_paket	rx_byte	tx_byte	rx düşme	tx düşme	çarpışma	süre
54529	1	6	13	508	2001	0	0	0	2
54528	1	6	20	508	3097	0	0	0	2
54531	1	6	19	508	2990	0	0	0	2
54530	1	6	18	508	2838	0	0	0	2
54536	1	6	14	508	2079	0	0	0	2
54534	1	22	21	4093	3780	0	0	0	2
54532	1	6	19	508	3034	0	0	0	2
54529	2	6	20	508	3258	0	0	0	3
54528	2	7	30	578	4754	0	0	0	3
54531	2	7	30	578	4742	0	0	0	3
54530	2	6	21	508	3187	0	0	0	3
54536	2	6	22	508	3438	0	0	0	3
54534	2	24	25	4412	4816	0	0	0	3
54535	1	6	21	508	3348	0	0	0	2
54532	2	7	29	578	4425	0	0	0	3

EK 2. Ağaç topolojisine ait networkün uygulamada kullanılan veri özellikleri

Veri setindeki bilgilerden switch numarası, port numarası, alınan paket miktarı, iletilen paket miktarı ve süre uygulama esnasında kullanılan sütunlardır.

switch no	port no	rx_paket	tx_paket	süre
54529	1	6	13	2
54528	1	6	20	2
54531	1	6	19	2
54530	1	6	18	2
54536	1	6	14	2
54534	1	22	21	2
54532	1	6	19	2
54529	2	6	20	3
54528	2	7	30	3
54531	2	7	30	3
54530	2	6	21	3
54536	2	6	22	3
54534	2	24	25	3
54535	1	6	21	2
54532	2	7	29	3

ÖZGEÇMİŞ

Kişisel Bilgiler	
Adı Soyadı	Şükran
Doğum Yeri	Değirmenci
Doğum Tarihi	18.06.1991
Uyruğu	T.C. Diğer:
Telefon	05396624773
E-Posta Adresi	skrngurol@gmail.com
Web Adresi	

Eğitim Bilgileri	
Lisans	
Üniversite	Eskişehir Anadolu Üniversitesi
Fakülte	Mühendislik Fakültesi
Bölümü	Bilgisayar Mühendisliği
Mezuniyet Yılı	18.06.2014

Yüksek Lisans	
Üniversite	İstanbul Üniversitesi
Enstitü Adı	Fen Bilimleri Enstitüsü
Anabilim Dalı	Bilgisayar Mühendisliği Anabilim Dalı
Programı	Bilgisayar Mühendisliği