

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YAZILIM TANIMLI AĞLARDA YÜK Dengeleme Probleminin Optimizasyonu

DOKTORA TEZİ

Şeyma AYMAZ

EYLÜL 2023
TRABZON



KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YAZILIM TANIMLI AĞLARDA YÜK DENGELEME PROBLEMİNİN OPTİMİZASYONU

Şeyma AYMAZ

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünde
"DOKTOR (BİLGİSAYAR MÜHENDİSLİĞİ)"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 09 / 08 / 2023

Tezin Savunma Tarihi : 01 / 09 / 2023

Tez Danışmanı : Doç. Dr. Tuğrul ÇAVDAR

Trabzon 2023

ÖNSÖZ

“Yazılım Tanımlı Ağlarda Yük Dengeleme Probleminin Optimizasyonu” isimli bu tez Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı, Doktora Programı’nda hazırlanmıştır.

Tez çalışması kapsamında, Yazılım Tanımlı Ağ tabanlı veri merkezlerinde yük dengeleme konusuna yönelik farklı yaklaşımlar önerilmiştir. Önerilen yaklaşımların içerisinde veri düzleminde uygulanan sunucu tabanlı ve bağlantı tabanlı yük dengeleme yöntemleri mevcuttur. Ayrıca çalışmaya yeni bir yön vermesi açısından akış sınıflandırma konusu incelenmiş ve fil akış tespiti uygulanmıştır. Fil akış tespitinde farklı sınıflandırıcı yöntemleri denenmiş ve başarısını kanıtlayan sınıflandırıcılar fil akış tespitinde kullanılmıştır. Bu yaklaşımla amacımız, akışların türlerine göre yönlendirilmesini sağlamaktır.

Doktora eğitimim ve tez danışmanlığım boyunca ilgi, destek ve tecrübelerini esirgemeyen, yönlendirmeleriyle çalışmamın olgunlaşmasını sağlayan saygıdeğer hocam Doç. Dr. Tuğrul ÇAVDAR’a teşekkürü bir borç bilirim. Çalışma süresince bilgi ve tecrübelerini esirgemeyen Sayın Dr. Öğr. Üyesi Selçuk CEVHER’e ve Sayın Doç. Dr. Gökçe HACIOĞLU’na en içten teşekkürlerimi sunarım.

Öğrenim hayatımda bu noktaya gelmemde en büyük pay sahibi olan değerli babam Mustafa DEMİR’e ve sevgili annem Zahide DEMİR’e şükranlarımı sunarım. Çalışmam boyunca sabır, anlayış ve her türlü fedakarlığı göstererek desteğini bir an olsun esirgemeyen eşim Dr. Samet AYMAZ’a, bu zorlu sürecin her anında pozitif enerjisi ve neşesi ile yorgunluk nedir hissettirmeyen canım oğlum Burak AYMAZ’a sonsuz teşekkürlerimi sunarım. Son olarak varlıklarına şükrettiğim güzel kardeşim ve yakışıklı abim, iyi ki varsınız.

Şeyma AYMAZ

Trabzon 2023

TEZ ETİK BEYANNAMESİ

Doktora Tezi olarak sunduğum “Yazılım Tanımlı Ağlarda Yük Dengeleme Probleminin Optimizasyonu” başlıklı bu çalışmayı baştan sona kadar danışmanım Doç. Dr. Tuğrul ÇAVDAR’ın sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 01/09/2023

Şeyma AYMAZ

İÇİNDEKİLER

	<u>Sayfa No</u>
ÖNSÖZ	III
TEZ ETİK BEYANNAMESİ	IV
İÇİNDEKİLER	V
ÖZET.....	VII
SUMMARY	VIII
ŞEKİLLER DİZİNİ.....	IX
TABLolar DİZİNİ	XI
SEMBOLLER DİZİNİ.....	XIII
1. GENEL BİLGİLER	1
1.1. Giriş.....	1
1.2. Yazılım Tanımlı Ağlar (YTA)	2
1.3. YTA Kullanım Alanları	5
1.4. YTA Yük Dengeleme	14
1.4.1. Sunucu Tabanlı Yük Dengeleme	14
1.4.2. Bağlantı Tabanlı Yük Dengeleme.....	19
1.5. YTA Yönlendirme	21
1.5.1. Fil Akış Tespiti.....	22
1.5.2. Fil Akışların Planlaması.....	24
1.6. Tezin Amacı, Kapsamı ve Sunulan Katkıları.....	28
2. YAPILAN ÇALIŞMALAR, BULGULAR VE İRDELEME.....	29
2.1. Giriş.....	29
2.2. YTA Sunucu Tabanlı Yük Dengeleme	29
2.3. YTA Bağlantı Tabanlı Yük Dengeleme.....	31
2.3.1. Kullanılan Ağ Topolojisi.....	33
2.3.2. Uygulanan Ayrık Parçacık Sürü Optimizasyonu (D-PSO).....	34
2.3.3. Maliyet Fonksiyonun Hesaplanması	39
2.4. YTA Fil Akış Tespiti	42
2.4.1. Kullanılan Veri Setleri	46
2.4.2. Temel Gerçek Etiketleme.....	47
2.4.3. Seçilen Akış Sınıflandırma Yöntemleri	48

2.5.	YTA Fil Akışların Planlanması.....	56
3.	BULGULAR VE İRDELEME	58
3.1.	YTA Sunucu Tabanlı Yük Dengeleme Sonuçları.....	58
3.2.	YTA Bağlantı Tabanlı Yük Dengeleme Sonuçları	61
3.2.1.	İlk Trafik Senaryosu ve Çıktıları.....	62
3.2.2.	İkinci Trafik Senaryosu ve Çıktıları.....	69
3.3.	YTA Fil Akış Tespiti Sonuçları	73
3.4.	YTA Fil Akışların Planlanması Sonuçları	80
4.	SONUÇLAR VE TARTIŞMA.	94
5.	ÖNERİLER.....	97
6.	KAYNAKLAR	98
ÖZGEÇMİŞ		

Doktora Tezi

ÖZET

YAZILIM TANIMLI AĞLARDA YÜK Dengeleme Probleminin OPTİMİZASYONU

Şeyma AYMAZ

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Doç. Dr. Tuğrul ÇAVDAR
2023, 107 Sayfa

Son zamanlarda, Yazılım Tanımlı Ağlarda trafik mühendisliği konusuna yönelik önemli bir ilgi vardır. Özellikle, veri düzleminde sunucu tabanlı ve bağlantı tabanlı yük dengeleme konusu ilgi çekicidir. Sunucu tabanlı yük dengeleme aynı özelliklere sahip sunucular arasında yükün dengelenmesini ele alırken, bağlantı tabanlı yük dengeleme kaynak ile hedef arasında belirlenen yollar arasında en az yüke sahip yolun belirlenmesini ele almaktadır. Tez çalışması kapsamında bu iki konu için de farklı yaklaşımlar önerilmiş ve başarıları kanıtlanmıştır. Bağlantı tabanlı yük dengelemede farklı özelliklere sahip akışların aynı yönlendirme stratejisi ile yönlendirilmesi gerçekçi değildir. Bu nedenle, tez çalışmasında akışlar sınıflandırılarak fil akışlar tespit edilmiştir. Fil akış tespiti için önerilen algılayıcı yedi farklı sınıflandırıcı ile test edilmiş ve yöntemlerin başarısı simülasyon sonuçları ile kanıtlanmıştır. Sonuçlar incelendiğinde Karar Ağaçları, Destek Vektör Makineleri ve Derin Öğrenme yöntemlerinin fil akış tespiti için daha başarılı sonuçlar ürettiği belirlenmiştir. Tez çalışması kapsamında, fil akışlar, bağlantı tabanlı yük dengeleme yaklaşımı için önerilen, yükü dinamik olarak dengeleyen Ayrık – Parçacık Sürü Optimizasyon yöntemi ile yönlendirilmektedir. Fare akışlar ise geleneksel Round Robin veya Rastgele yöntemleriyle yönlendirilmektedir. En başarılı sonuç Derin Öğrenme yöntemine göre fil akış tespiti yaparak fare akışların Round Robin’e göre yönlendirildiği yöntemden elde edilmiştir.

Anahtar Kelimeler: YTA, yük dengeleme, fil akış tespiti, sınıflandırma, derin öğrenme.

PhD Thesis

SUMMARY

OPTIMIZING THE LOAD BALANCING PROBLEM IN SOFTWARE-DEFINED NETWORKS

Şeyma AYMAZ

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Computer Engineering Graduate Program
Supervisor: Assoc. Prof. Tuğrul ÇAVDAR
2023, 107 Pages

Recently, there has been considerable interest in traffic engineering in Software-Defined Networks. In particular, the issue of server-based and connection-based load balancing in the data plane is of interest. Server-based load balancing deals with load balancing between servers with the same specifications, while connection-based load balancing deals with determining the path with the least load among the paths between the source and the destination. Within the scope of the thesis, different approaches have been proposed for these two subjects, and their success has been proven. Routing flows with different characteristics with the same routing strategy in connection-based load balancing is unrealistic. For this reason, in the thesis study, elephant flows were determined by classifying the flows. The proposed framework for elephant flow detection has been tested with seven different classifiers, and simulation results have proved the success of the methods. When the results were examined, it was determined that Decision Trees, Support Vector Machines and Deep Learning methods produced more successful results for elephant flow detection. Within the scope of the thesis, the elephant flows are routed by the Discrete-Particle Swarm Optimization method, which dynamically balances load, which is recommended for the connection-based load balancing approach. Also, Mouse flows are routed by traditional Round-robin or Random methods. The most successful result was obtained from the method in which the mouse flows were routed according to Round Robin by performing elephant flow detection according to the Deep Learning method.

Key Words: SDN, load balancing, elephant flow detection, classification, deep learning.

ŞEKİLLER DİZİNİ

Sayfa No

Şekil 1.1. Geleneksel ağlar mimarisi ile YTA mimarisinin karşılaştırılması [2].....	2
Şekil 1.2. YTA mimarisi [3]	3
Şekil 1.3. YTA'nın çalışması [5]	4
Şekil 1.4. YTA'da trafik ölçümünün zorlukları ve açık konular [28].....	9
Şekil 1.5. YTA'da yük dengelemenin sınıflandırılması [31].....	10
Şekil 1.6. Dağıtık YTA mimarileri [31]	11
Şekil 1.6. (a) Düz (Flat) mimari	11
Şekil 1.6. (b) Hiyerarşik mimari	11
Şekil 1.7. Anahtar göçü [25]	11
Şekil 1.8. Dinamik küme mimarisi [34].....	13
Şekil 1.9. Örnek bir ağ topolojisi	15
Şekil 1.10. Yük dengeleme mimarisi ve önerilen mimari	17
Şekil 2.1. Fat tree ağ topolojisi	33
Şekil 2.2. Örnek bir ağ topolojisi	35
Şekil 2.3. Uygulanan D-PSO algoritmasının akış şeması	36
Şekil 2.4. Önerilen algılayıcının akış şeması	45
Şekil 2.5. Hiperparametreler alanı	48
Şekil 2.6. Lojistik Regresyon [97]	49
Şekil 2.7. Destek Vektör Makineleri [99]	50
Şekil 2.8. Karar Ağaçları [100]	51
Şekil 2.9. Rastgele Ormanlar [102]	52
Şekil 2.10. Naive Bayes [104]	53
Şekil 2.11. K-En Yakın Komşu Algoritması [106].....	53
Şekil 2.12. Derin Öğrenme [110].....	54
Şekil 2.13. Önerilen CNN mimarisi.....	55
Şekil 2.14. Önerilen yönlendirme çerçevesi	57
Şekil 3.1. 4 istemci 2 sunucu 100 istek	59
Şekil 3.2. 4 istemci 2 sunucu 1000 istek	59
Şekil 3.3. 4 istemci 4 sunucu 100 istek	60

Şekil 3.4. 4 istemci 4 sunucu 1000 istek	60
Şekil 3.5. 5 dakikalık simülasyon süresinde RTT değerlerinin karşılaştırılması	63
Şekil 3.6. 10 dakikalık simülasyon süresinde RTT değerlerinin karşılaştırılması ..	64
Şekil 3.7. 5 dakikalık simülasyon süresinde PLR değerlerinin karşılaştırılması	65
Şekil 3.8. 10 dakikalık simülasyon süresinde PLR değerlerinin karşılaştırılması ...	65
Şekil 3.9. Ağ topolojisinin hiyerarşik yapısı	70
Şekil 3.10. Oluşturulan topolojinin tüm linkleri	71
Şekil 3.11. S1 bağlantılarının yük dağılımı	71
Şekil 3.12. S7 bağlantılarının yük dağılımı	72
Şekil 3.13. Topolojideki tüm anahtarların yük dağılımları	73
Şekil 3.14. Hata matrisi.....	75
Şekil 3.15. UNI1 veri seti için F-puanı değeri karşılaştırması	78
Şekil 3.16. UNI2 veri seti için F-puanı değeri karşılaştırması	79
Şekil 3.17. UNI1 veri seti ve UNI2 veri seti için çalışma süresi karşılaştırması	80

TABLÖLAR DİZİNİ

	<u>Sayfa No</u>
Tablo 1.1. YTA’larda trafik mühendisliği kapsamı [25]	8
Tablo 1.2. Literatürde sıklıkla kullanılan fil akış planlama yöntemlerinin karşılaştırması	27
Tablo 2.1. FlowRecorder aracı kullanılarak elde edilen özellikler.	43
Tablo 2.2. Simülasyonlarda kullanılan izlemelerin özellikleri	47
Tablo 3.1. 5 dakikalık simülasyon süresi boyunca Verimlilik değerinin karşılaştırılması	67
Tablo 3.2. 10 dakikalık simülasyon süresi boyunca Verimlilik değerinin karşılaştırılması	67
Tablo 3.3. 5 dakikalık simülasyon süresi boyunca Arabellek Boyutu değerinin karşılaştırılması	68
Tablo 3.4. 10 dakikalık simülasyon süresi boyunca Arabellek Boyutu değerinin karşılaştırılması	68
Tablo 3.5. UNI1 veri seti kullanılarak mevcut fil akış algılama yöntemleriyle genel performans karşılaştırması	77
Tablo 3.6. UNI2 veri seti kullanılarak mevcut fil akış algılama yöntemleriyle genel performans karşılaştırması	78
Tablo 3.7. Fil akış tespiti uygulanmadan yapılan yönlendirmenin RTT değerlerinin karşılaştırılması	82
Tablo 3.8. Fil akış tespiti uygulanmadan yapılan yönlendirmenin Verimlilik (Mbit/sec) değerlerinin karşılaştırılması	82
Tablo 3.9. Fil akış tespiti uygulanmadan yapılan yönlendirmenin Arabellek boyutu (KB) değerlerinin karşılaştırılması	82
Tablo 3.10. Fil akış tespiti sonrası yapılan yönlendirmenin RTT değerlerinin karşılaştırılması (Fil akış (%20) ve Fare akış (%80))	84
Tablo 3.11. Fil akış tespiti sonrası yapılan yönlendirmenin Verimlilik (Mbit/sec) değerlerinin karşılaştırılması (Fil akış (%20) ve Fare akış (%80))	85
Tablo 3.12. Fil akış tespiti sonrası yapılan yönlendirmenin Arabellek Boyutu (KB) değerlerinin karşılaştırılması (Fil akış (%20) ve Fare akış (%80))	86

Tablo 3.13.	Fil akış tespiti sonrası yapılan yönlendirmenin RTT değerlerinin karşılaştırılması (Fil akış (%50) ve Fare akış (%50))	87
Tablo 3.14.	Fil akış tespiti sonrası yapılan yönlendirmenin Verimlilik (Mbit/sec) değerlerinin karşılaştırılması (Fil akış (%50) ve Fare akış (%50))	88
Tablo 3.15.	Fil akış tespiti sonrası yapılan yönlendirmenin Arabellek Boyutu (KB) değerlerinin karşılaştırılması (Fil akış (%50) ve Fare akış (%50))	89
Tablo 3.16.	Oluşturulan birinci ve ikinci trafik için elde edilen en başarılı yöntemlerin kıyaslanması (Derin Öğrenme ile fil akış tespiti, Fil akışların yönlendirilmesi D-PSO, Fare akışların yönlendirilmesi RR).....	91
Tablo 3.17.	Fil akış tespitinin yönlendirmeye etkisi	92

SEMBOLLER DİZİNİ

YTA	Yazılım Tanımlı Ağlar (Software Defined Network)
IoT	Nesnelerin İnternet'i (Internet of Things)
TE	Trafik Mühendisliği (Traffic Engineering)
QoS	Servis Kalitesi (Quality of Services)
D-PSO	Ayrık Parçacık Sürü Optimizasyonu (Discrete Particle Swarm Optimization)
RTT	Gidiş Dönüş Süresi (Round Trip Time)
BW	Bant Genişliği (Bandwidth)
PLC	Kayıp Paket Sayısı (Packet Loss Count)
PN	Port Sayısı (Number of Ports)
RR	Round Robin
GA	Genetik Algoritma (Genetic Algorithm)
ACO	Karınca Kolonisi Optimizasyonu (Ant Colony Optimization)
G-ACO	Genetik-Karınca Kolonisi Optimizasyonu (Genetic-Ant Colony Optimization)
PLR	Paket Kayıp Oranı (Packet Loss Rate)
DVM	Destek Vektör Makineleri (Support Vector Machine)
KNN	K-En Yakın Komşu (K-Nearest Neighbours)
DL	Derin Öğrenme (Deep Learning)
POX	Python OpenFlow eXercise
Mininet	Mini Network

1. GENEL BİLGİLER

1.1. Giriş

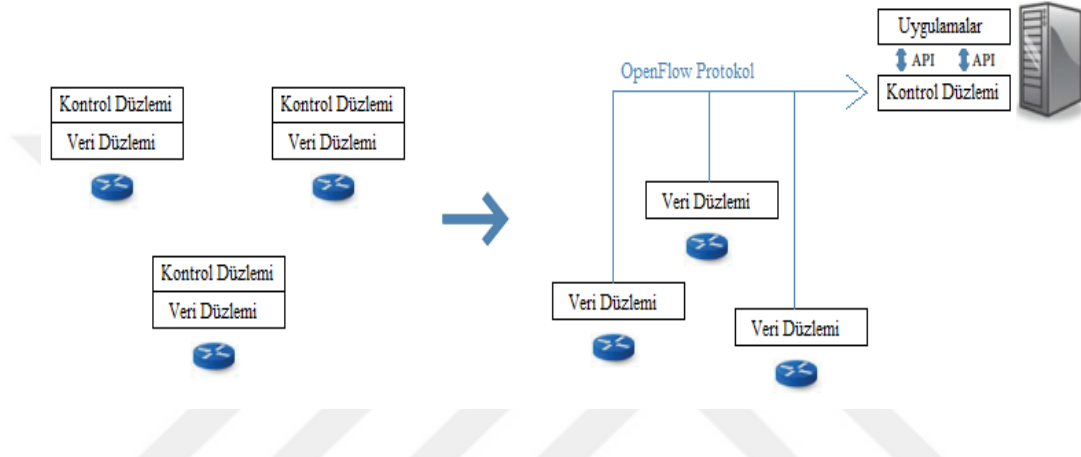
İnternet kullanımı, her geçen gün hızla yaygınlaşmaktadır. Yaygın kullanım ile birlikte, artan ve gelişmekte olan ihtiyaçları geleneksel ağ mimarileriyle karşılamak zorlaşmaktadır. Aynı zamanda geleneksel ağlar; ağ anahtarları, yönlendiriciler, hublar, farklı ağ cihazları, karmaşık protokoller ve arayüzler gibi büyük çeşitlilikte ağ bileşenlerinden oluşmaktadır. Her yeni protokol birtakım sorunları çözmesinin yanında daha karmaşık ve yönetilmesi daha zor ağ yapılarının oluşmasına sebep olmaktadır. Bu durum mevcut ağlarda kurulumdan sonra geliştirilme ve yeni şebeke özelliklerinin yüklenmesini sınırlamaktadır. Bahsedilen ağ problemlerinin üstesinden gelebilmek amacıyla Yazılım Tanımlı Ağlar (YTA) kavramı ortaya çıkmıştır.

YTA kavramı 1990'ların öncesine dayanmakla birlikte, 2006 yılında OpenFlow kavramıyla küresel bazda ilgi görmeye başlamıştır [1]. Son zamanlarda ise, akademik dünyada ve sanayi topluluklarında sıcak bir konu olarak ele alınmaktadır. Günümüz ağ altyapısının sınırlılıklarını ortadan kaldırmaya çalışan YTA, kontrol ve veri düzlemini birbirinden ayırarak, ağda programlama yapılabilmesine imkân tanıyan ve ağın küresel görünümünü sağlayarak değişen durumlara hızlı tepki verebilen yeni bir ağ yaklaşımı olarak karşımıza çıkmaktadır. YTA, basitçe ağ anahtarları ve denetleyici(ler)den oluşmaktadır. Ağ anahtarları basit yapıdadır ve denetleyicinin kendilerine yüklediği kurallar ile akış işlemlerini gerçekleştirmektedirler. Denetleyici ağın küresel görünümüne sahiptir. Böylece her bir paketin hedefine gidebilmesi için gerekli yolu hesaplayabilmektedir. Yani merkezden yönetilebilen bir kontrol düzlemi dinamik, yüksek performanslı, verimli ve geliştirilebilir bir iletişim altyapısı inşa edilebilmesine olanak sağlamaktadır.

YTA oldukça fazla kullanım alanına sahiptir. En önemli kullanım alanlarından birisi yük dengelemedir. Uygulanan düzleme ve mimariye göre çeşitli yük dengeleme stratejileri mevcuttur. Tez çalışması kapsamında YTA'da yük dengeleme konusu farklı stratejilerle ele alınmaktadır.

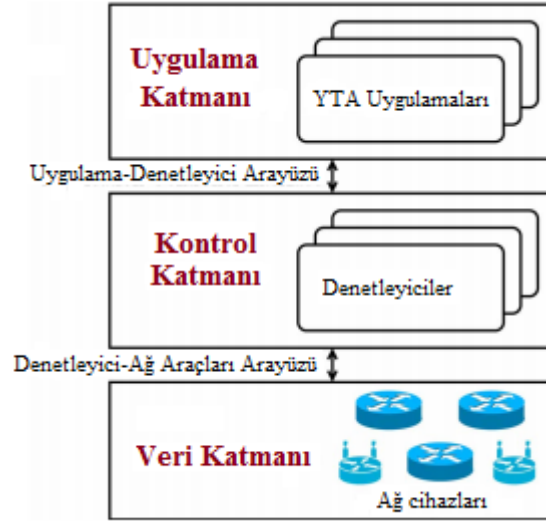
1.2. Yazılım Tanımlı Ağlar (YTA)

Geleneksel ağların karmaşık mimarisi, ağın yönetimini zorlaştırmaktadır. YTA, daha basit yapıda ve ağın küresel görünümüne sahip olduğundan yönetimi geleneksel ağlara göre daha kolaydır. Her iki ağ yapısı da belirli bir mimari üzerine kurulmaktadır. Geleneksel ağlar mimarisi ve YTA mimarisi Şekil 1.1.'de verilmiştir.



Şekil 1.1. Geleneksel ağlar mimarisi ile YTA mimarisinin karşılaştırılması [2]

YTA, basitçe ağ anahtarları ve denetleyici(ler)den oluşmaktadır. Ağ anahtarlarından yönetim mekanizması ayrılarak, bu ağ cihazları daha rutin işler yapan basit cihazlar haline getirilmekte ve denetleyicinin kendilerine yüklediği kurallar ile akış işlemlerini gerçekleştirmektedirler. Denetleyici ağın küresel görünümüne sahip olduğundan her bir paketin hedefine gidebilmesi için gerekli yolu hesaplayabilmektedir. Böylelikle hem sayıca fazla olan ağ cihazlarının maliyetleri düşmekte, hem de yönetim konumunda olan sayıca az cihazların değişimi ve güncellemeleri (bellek ve işlemci gücü artırılması vb.) rahat yapılabilmektedir. YTA mimarisi basit olarak Şekil 1.2. ile verilmektedir.

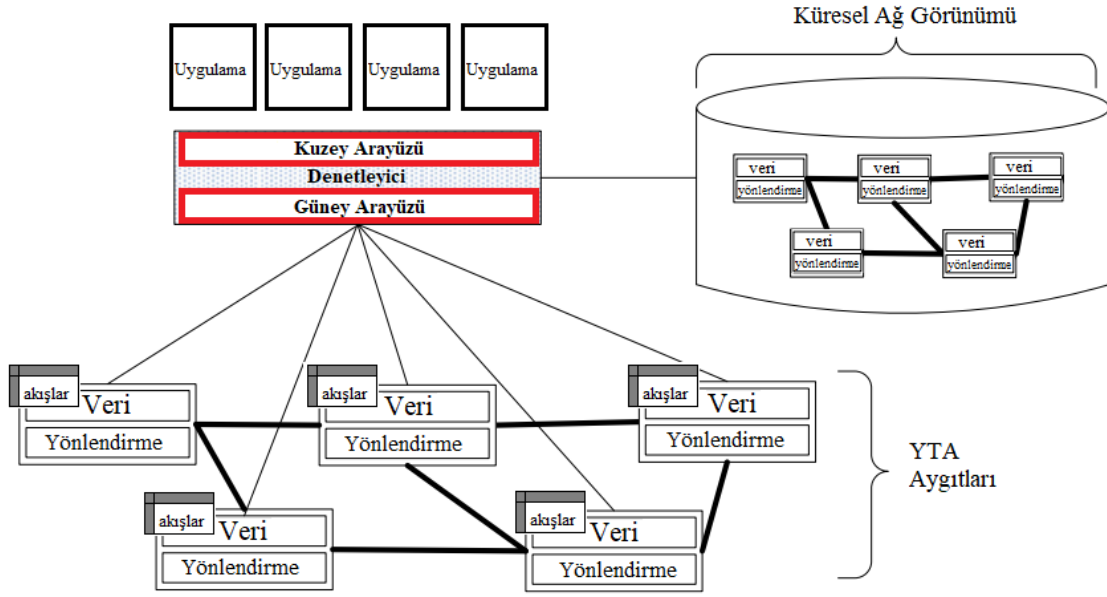


Şekil 1.2. YTA mimarisi [3]

YTA'nın avantajları;

- ✓ Kontrol düzlemiyle veri düzleminin ayrılması,
- ✓ Hedef bazlı yönlendirme yerine akış bilgilerine dayalı yönlendirme yapılması,
- ✓ Ağda programlama yapılabilmesi,
- ✓ Yönetici vasıtasıyla anlık durumlara hızlı tepki verilebilmesi,
- ✓ Cihaz güç tüketimlerinin ve maliyetlerinin azaltılabilmesidir [4].

Yazılım tanımlı ağların çalışabilmesi için kullanılacağı ağa ait denetleyici ya da denetleyiciler bulundurulması gerekmektedir. Yönetici görevindeki bu cihazlarda ağ akışlarına uygulanacak kurallar oluşturularak ağın yönetimi sağlanır. Her paket için hedef-kaynak IP, port ve MAC adresleri, VLAN ID'si ve önceliği, protokol ve zaman bilgilerinden oluşan akış bilgisi, denetleyiciye gelir ve denetleyici, üzerinde bulunan programa göre, bu akış hakkında bir karar verir. Ancak bu işlemlerin daha hızlı yapılabilmesi ve denetleyiciye daha az iş yükü getirmesi amacıyla uygulanacak kurallar, iletim cihazlarında bulunan akış tablolarına yazdırılarak bazı akışlar, denetleyiciye hiç sorulmadan iletim cihazı tarafından eklenir. Denetleyici, zaman içerisinde değişen görevlere göre bu tabloları düzenleme yetkisine sahiptir. YTA'nın çalışması Şekil 1.3.'te gösterilmektedir.



Şekil 1.3. YTA'nın çalışması [5]

Yeni bir ağ anlayışı getiren YTA ile yeni terimler de ortaya çıkmıştır. YTA'yı daha iyi anlayabilmek için bu terminolojiye de hakim olmak gerekir. YTA'nın ortaya çıkardığı en önemli terimler aşağıda verilmiştir [6].

Kontrol Düzlemi: Ağ cihazlarına ne yapılması gerektiğini söyleyen YTA denetleyicileri ve ağ işletim sisteminden oluşmaktadır. Kontrol düzleminin iki görevi vardır; ağ cihazlarına hangi işlemi uygulayacağını söylemek ve ağ cihazları ile bağlantıları hakkında bilgi toplayarak ağ uygulamalarına güncel ağ görünümünü sunmak.

Veri Düzlemi: Fiziksel seviyede veri transferinin yapıldığı ağ aygıtları ve bağlantılarının bulunduğu düzlemdir. Bu düzlemde bulunan basit yapıları cihazlar, kontrol düzleminden aldığı komutlara göre işlem yapmaktadır.

Uygulama Düzlemi: Kontrol düzleminde yer alan ağ denetleyicisinin gereksinim duyduğu ağ uygulamalarının var olduğu düzlemdir. Bu uygulamalara, yük dengeleyici, güvenlik duvarı, yönlendirici, ağ ölçümü gibi uygulamalar örnek verilebilir.

İletim Cihazı: Veri düzleminde bulunan ve üzerinde oluşturulan akış tablosunu kullanarak ağdaki paket iletimlerini gerçekleştiren cihazlardır. Paket iletiminin yanında gelen bir paket için düşürme, başlık bilgisini değiştirme ya da kontrolcüyeye gönderme işlemleri de yapabilmektedir.

Denetleyici: Ağın beynidir. Veri düzleminden aldığı iletim bilgilerini değerlendirir, gerekliyse uygulama düzlemiyle paylaşır ve iletim cihazlarına yapması gereken işlemi bildirir.

Güney (Southbound) ve Kuzey (Northbound) Arayüzleri: Denetleyici ve ağ anahtarları arasındaki arayüz Güney; uygulamalar ve denetleyici arasındaki arayüz ise Kuzey olarak isimlendirilir. Güney arayüzü, programlanabilir anahtarlar ve denetleyici arasındaki protokol ve arayüzü ifade eder. En yaygın kullanılan Güney protokolü OpenFlow'dur. Kuzey arayüzü ise işletimsel görevlerin ve ağ politikalarının nasıl belirleneceğini ve bu politikaların denetleyici tarafından kullanılabilir hale getirilmesini sağlamaktadır.

1.3. YTA Kullanım Alanları

YTA'nın son zamanlarda yaygınlaşmasıyla birlikte kullanım alanlarının da arttığı görülmektedir. Nesnelerin İnterneti (IoT) [7], Araçların İnterneti (IoV) [8], Sis Hesaplama (Fog Computing) [9], sanallaştırma [10], trafik mühendisliği [11], güvenlik [12], yük dengeleme bu kullanım alanlarına örnek olarak verilebilir. Kullanım alanlarına literatürden örnek birkaç çalışma verilmiştir.

Nesnelerin İnterneti (IoT), nesneler ve heterojen cihazlar arasındaki iletişimi destekleyen yeni bir paradigmadır. Heterojen cihazlar arasındaki veri iletimini sağlamak için yönlendirme protokolüne ihtiyaç vardır. Heterojen aygıtların enerji tüketimini göz önüne alacak bir yönlendirme protokolü önerilmiştir. YTA denetleyicisi ile ağda mevcut olan bencil düğümlerin ağa erişimi engellenmiştir. Böylece enerji tasarrufu sağlanmıştır [13].

Sis Hesaplama (Fog Computing), veri işleme, depolama ve analizleri ağ kenarına yakınlaştıran yeni bir teknolojidir. Sis gezginliği destekler, ağ kenarlarına yakındır ve cihazların coğrafi dağılımını destekler. En büyük avantajı ağ kenarlarına olan yakınlığı nedeniyle düşük gecikme sağlamasıdır. Sis Hesaplama teknolojisinin bu faydaları ile YTA'nın sunduğu avantajlar birleştirilerek yeni bir mimari model sunulmuştur [14]. Chen ve arkadaşları Sis Hesaplama ile IoT servislerinin nasıl kullanılacağı ve faydaları ile ilgili bir çalışma yapmıştır. Çalışmada Sis Hesaplamanın sağladığı avantajlardan bahsederken Bulut Hesaplama (Cloud Computing) ile farklarını da vermektedir. Bu farklar; kenar konumunda işleme, gecikmeye duyarlı, coğrafi olarak dağıtılmış, büyük ölçekli algılayıcı ağlardan oluşur ve çok sayıda düğüm içerir, gerçek zamanlı etkileşim sunar, heterojenliği destekler, birlikte çalışılabilirlik sağlar, bulut ile etkileşime girebilir [15]. Chang ve

arkadaşları 2017 yılında Indie Fog diye adlandırdıkları yeni bir yapı önermişlerdir. Bu yapı ile Sis Hesaplamanın bazı sınırlılıklarına çözüm aranmıştır. Indie Fog esnek, uygun maliyetli, IoT desteği olan ve günümüz teknolojisiyle çalışabilen yapıdadır [16]. Biteencourt ve arkadaşları ise Sis Hesaplama kaynak tahsisi ve yönetimi üzerine çalışma yapmışlardır. Çalışmada kullanıcıların gezgin olduğu varsayılmış ve dinamik senaryolara uygun bir algoritma önerilmiştir [17]. Literatürde Sis Hesaplama teknolojisini taşıtlarla (vehicular) kullanan bazı çalışmalarda mevcuttur. Örneğin Sookhak ve arkadaşları kullanılmayan taşıt kaynaklarını, Sis Hesaplama kaynaklarını artırmak için kullanmışlardır. Taşıt kaynaklarını kullanırken taşıt sahiplerine cazip teklifler sunmuşlardır [18]. Başka bir çalışma Hou ve arkadaşları tarafından yapılmıştır. Hareketli ve park halindeki araçların iletişim ve hesaplamasını incelemişlerdir. Önerilen VFC (Vehicular Fog Computing) mimarisi coğrafi olarak dağıtık, gerçek zamanlı yük dengelemeye sahip, büyük ölçekli ve daha hızlı tepki süresi gibi avantajlar sağlamaktadır. Fakat en büyük avantajı link kesintisi meydana gelirse VFC taşıtların güç kaynakları ile çalışmasını sürdürebilmektedir [19]. Jalali ve arkadaşları IoT enerji tüketiminin gelecekte bir darboğaz halini alacağını savunmaktadır. IoT uygulamalarının ve hizmetlerinin Sis yardımıyla güç tüketimini azaltmaya yardımcı olabilecek stratejilerini vermişlerdir. Amaçları literatürdeki enerji tasarruflu IoT uygulamalarının tanıtımını yapmaktır [20].

Sanallaştırma, ağın yönetilebilir bölümlere ayrılarak gerçek karmaşıklığının gizlenmesidir. YTA'da artan trafik ve gezginlik için tercih edilmektedir. Bizanis ve arkadaşları IoT için YTA ve sanallaştırma teknolojilerini birleştiren yayınları incelemişlerdir. IoT ve ağ sanallaştırma teknolojileri ile ölçeklenebilirlik ve çok yönlülük sorunlarının uygun maliyetle aşılabileceğini savunmuşlardır. Sherwood ve arkadaşları ilk kez bu iki teknolojiyi bir arada kullanmışlardır. Çalışmalarında ağ, mantıksal olarak alt ağlara dilimlenmiş ve her bir alt ağ birbirinden bağımsızdır. Bu yapılarını FlowVisor olarak adlandırmışlardır. Çalışmada çoklu denetleyici mantığı vardır. Dilimlenmiş her bir alt ağ kendi denetleyicisi ile yönlendirilmektedir. Alternatif olarak önerilen FlowN yapısında ise tek denetleyici mantığı vardır [21-23]. YTA hipervizör, YTA ağlarının sanallaştırılması için kritik bir bileşendir. YTA hipervizör, fiziksel YTA ağını kendi denetleyicisine sahip birden fazla sanal YTA ağına soyutlamaktadır. Blenk ve arkadaşları hipervizörler ile ilgili yayınları kendi aralarında sınıflayarak bundan sonraki çalışmalara yol göstermeyi amaçlamışlardır [24].

Trafik mühendisliği (TE) yazılım tanımlı ağların kullanıldığı farklı bir uygulama alanıdır. Trafik mühendisliği, aktarılan verilerin davranışını düzenleyerek, tahmin ederek ve

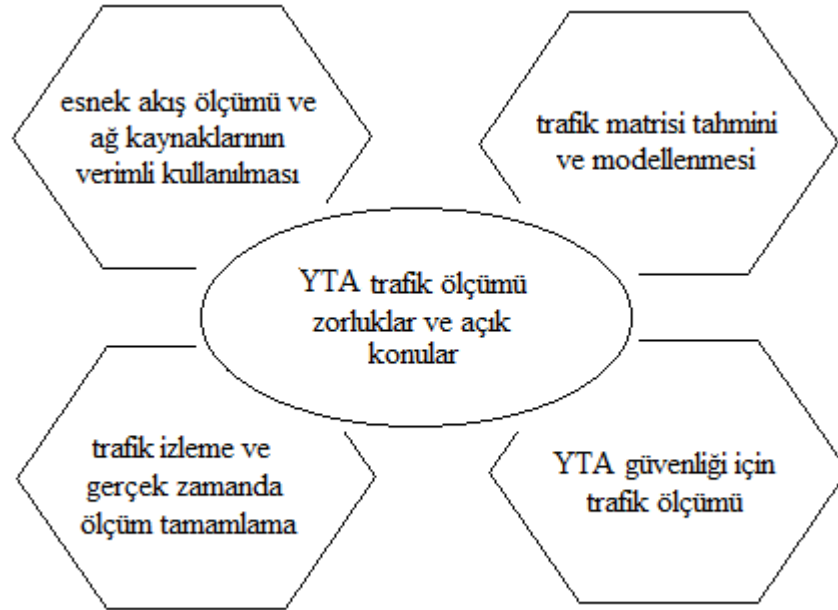
dinamik olarak analiz ederek bir veri ağının performansını optimize etmek için kullanılan önemli bir mekanizmadır. Akyildiz ve arkadaşları mevcut YTA'larda trafik mühendisliği kapsamını ele almışlardır. Bu kapsam Tablo 1.1.'de ifade edilmektedir. YTA'da bir ağ anahtarı, akış girişindeki kurallarla eşleşmeyen bir akış aldığında, akışın ilk paketini denetleyiciye iletmektedir. Buna göre, denetleyici, ağda dengeli trafik yüküne neden olabilecek yeni bir yönlendirme kuralı kurması gerekip gerekmediğine karar verir. Bununla birlikte, bu yönlendirme kuralı yükleme işlemi zaman alabilir ve gecikmeye neden olabilir. Ağ anahtarlarının uç kısımlarında çok sayıda yeni akış toplandığında hem kontrol düzleminde hem de veri düzleminde önemli miktarda yük ortaya çıkabilir. Akış yönetimi, gecikme ve yük dengesi arasındaki dengeyi (tradeoff) göz önüne alarak YTA'daki bu darboğazı önlemeyi amaçlayan çözümleri araştırmaktadır. Ağ güvenilirliğinden emin olmak için, YTA, ağ altyapısında (denetleyiciler, ağ anahtarları ve bağlantılar) başarısızlıklar oluştuğunda, başarısızlığı kurtarma işlemini gerçekleştirme yeteneğine sahip olmalıdır. Bir ağ anahtarı başarısız bağlantıyı tanımlayabilse bile, yeni bir rota oluşturmak için denetleyiciden gelen güncelleştirmelere bağlı kalmak zorundadır. Bir denetleyici başarısız bir bağlantıyı tanımlayana ve ilgili tüm ağ anahtarlarındaki akış tablosu girdilerini güncelleştirene kadar, başarısız olan bağlantıda dolaşması beklenen paket bırakılacaktır. Büyük önemine rağmen, hızlı hata kurtarma (50 ms), YTA için oldukça zor bir iştir. Çünkü merkezi denetleyicinin yeni yollar hesaplaması ve etkilenen tüm ağ anahtarlarını, kurtarma işlemi hakkında hemen bilgilendirmesi gerekmektedir. Hata toleransı, YTA ağlarında hızlı arıza gidermeyi gerçekleştirmek için mevcut çalışmaları araştırmaktadır. Topoloji güncellemesi, planlanmamış olaylar yerine (ağ anahtarı, bağlantı hataları vb.) planlanan değişikliklere (ağ ilkesi kural değişiklikleri vb.) odaklanmaktadır. Genel güncelleme işlemleri her paket veya her akış için düzenlenmektedir. Yani, ağ üzerinden akan her paketin tek bir ağ yapılandırmasına göre işleneceği veya aynı akıştaki tüm paketler ilkenin aynı sürümüyle işleneceği anlamına gelir. YTA denetleyicisinin ağın tutarlılıkla ve gerçek zamanlı olarak nasıl verimli bir şekilde güncelleştirileceği zorluklar arasındadır. YTA trafik analizinde ise YTA mimarilerindeki ağ yönetimi, ağ doğrulama ve hata ayıklama için mevcut ağ izleme araçlarını ele almaktadır [25-26].

Tablo 1.1. YTA’larda trafik mühendisliği kapsamı [25]

Trafik Mühendisliği			
Akış Yönetimi	Hata Toleransı	Topoloji Güncellemesi	Trafik Analizi/Tanımlama
*Ağ anahtarı yük dengeleme	*Veri düzlemi için hata toleransı	*Yeni güncelleme politikaları	*İzleme çerçevesi
*Denetleyici yük dengeleme	*Kontrol düzlemi için hata toleransı	-Ağ anahtarlarındaki tablo girişlerini yedekleme	*Trafik analizi
*Çoklu akış tabloları		-Zaman esaslı yapılandırma	*Ağın değişmez değerlerinin kontrolü
			*Program hatalarının onarılması

Shu ve arkadaşları, trafik ölçümü ve trafik yönetimi olmak üzere iki kısımdan oluşan YTA’da TE için bir referans çerçeve önermişlerdir. Trafik ölçümü, trafik yönetiminin ön şartı olarak, gerçek zamanlı ağ trafiğini izlemekten ve ağı analiz etmekten sorumludur. Trafik yönetimi ise yük dengeleme ve enerji tasarruflu tarifelemeden sorumludur. Ağ yönetiminin amacı makul trafik tarifeleme stratejisi sağlamaktır. Bu işlemi ağ parametrelerini izleyerek yapmaktadır. Ağın her katmanında ağın durumunu ifade eden parametreler mevcuttur [27]. Şekil 1.4., Yassine ve arkadaşlarının ele aldığı YTA’da trafik ölçümünün zorlukları ve açık konularını ifade etmektedir. YTA’larda trafik ölçümünün ana hedefi, çeşitli uygulamaları karşılamak için farklı ayrıntılarla esnek bir akış ölçümü sağlamaktır. Böylece birbirine bağlı heterojen büyük ölçekli ağlarda esnek ayarlarla ağ akışlarının ince-taneli hacmini tahmin etmeyi gerektirir. Buna ek olarak, trafik tahmini, öngörme ve ağ kaynaklarının tahmini gereklidir. Zamanla, ağ kaynaklarının düzgün şekilde kullanılmadığı ve dolayısıyla bağlantı kullanımının optimal olmadığı bazı senaryolar mevcuttur. Örneğin, daha küçük bant genişliği olan yeni hizmetler muhtemelen daha fazla bant genişliği gerektirecek olan gelecekteki hizmetleri dikkate almadan hazırlanırken, en uygun güzergahlar küçük bant genişliği (ilk gelen, ilk hizmet) tüketen hizmetlere verilmektedir. Daha sonra, yüksek bant genişliği servislerini tedarik etmek istediğimizde, kaçınılmaz olarak daha az optimum yollar karşımıza çıkmaktadır. Bu senaryolarda, şebeke operatörlerine bilgi verecek esnek ve dinamik ölçüm şemaları sağlamak için mekanizmalara ihtiyaç duyulur; bu yolla, küçük hizmetlerin dinamik bir şekilde başka yollara dağıtılarak

ağın yönetimini ve yeni yüksek bant genişliği hizmetleri için bağlantı kapasitelerini daha iyi kullanmasını sağlayarak kaynak kullanımını maksimize etmelerine yardımcı olurlar. Trafik matrisleri, bir ağdaki kaynak çiftleri ve hedef arasında akan trafik miktarını yansıtmaktadır. YTA’lardaki OpenFlow protokolü, akış tablolarını sorgulayarak diğer mekanizmaların trafik matrislerini tahmin etmesine izin veren akış tablosu sayaçlarını sağlamaktadır. Trafik matrisleri tarafından sağlanan bilgiler (akışların boyutu vb.), trafikle ilgili sorunları teşhis etmek, trafik anormalliklerini keşfetmek, yönlendirme yapmak ve yük dengelemek için büyük önem taşımaktadır. Yeni nesil ağlar heterojen, büyük ölçekli ve karmaşıktır. Bu nedenle, servisler ağda dinamik olarak hazırlanırken trafik matrisini doğru bir şekilde tespit etmek oldukça zordur. Trafik matrislerini, matematiksel ve istatistiksel yöntemlerle ve çok sayıda trafik matrisi verisini işlemek için ölçeklenebilir algoritmalar yoluyla tahmin ve modelleme mekanizmalarını bulmak hala çözülmesi gereken bir problemdir [28].

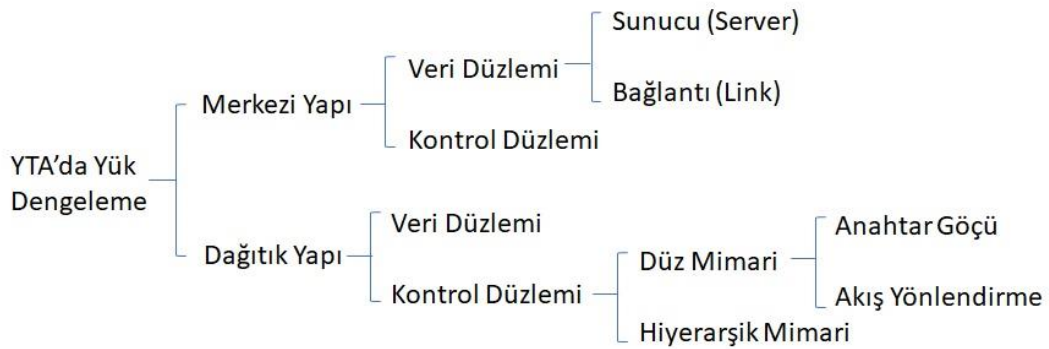


Şekil 1.4. YTA’da trafik ölçümünün zorlukları ve açık konular [28]

İletişim ağı içindeki trafik matrisi tahmini, asıl zorluğu ve çok sayıda ağ optimizasyonu ve yönetim işlevine potansiyel faydası nedeniyle uzun süredir devam eden bir sorundur. Polverini ve arkadaşları seçilen trafik akışı ölçümleri ile kolaylıkla elde edilen bağlantı yükü ölçümlerinin yanı sıra trafik matrisi tahmininin iyileştirilmesini de incelemişlerdir. Akış

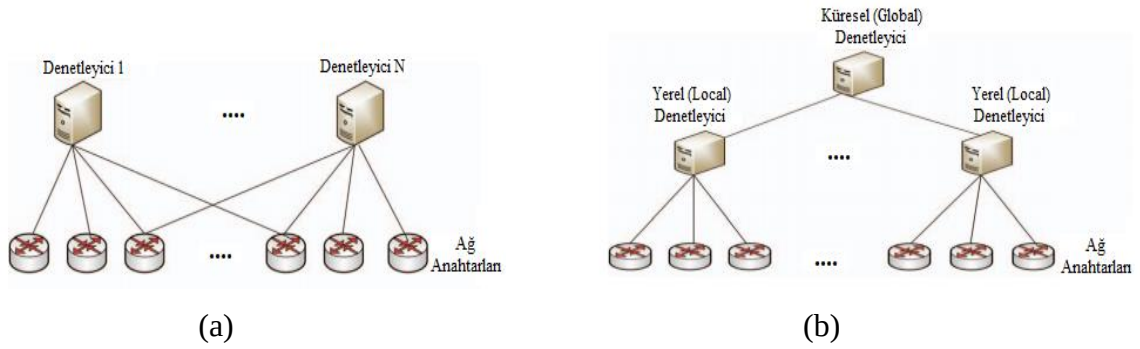
yayılımına dayalı algoritma (flow spread-based algorithm-FSBA) olarak adlandırdıkları algoritma, mevcut yönlendirme tabloları alanını göz önüne alarak ölçüm görevlerini ağ düğümleri arasında adaletli bir şekilde dağıtmaktadır. Ayrıca, FSBA'nın en son teknolojiye benzer yaklaşımlardan daha iyi performans sergilediğini de ispatlamışlardır [29].

Yük dengeleme, YTA'nın en önemli uygulama alanlarından bir diğeridir. Yük dengeleme, gelen trafiği ve kaynakları yönetir, artan talebi karşılamak ve kaynak eksikliklerini gidermek için ağ performansını iyileştirir [30]. Li ve arkadaşı YTA'larda yük dengeleme şemasını sınıflandırmıştır. Bu sınıflandırma Şekil 1.5.'te gösterilmektedir. Merkezi YTA mimarisinde bir denetleyici, çoklu ağ anahtarı ve sunuculardan (server) oluşmaktadır. Bu mimaride yük dengeleme problemi sunucularda ve bağlantılarda meydana gelmektedir. Denetleyici bütün ağı kontrol ederek ağıdaki bütün trafiği yönetmektedir. Ayrıca denetleyici, ağıdaki tüm yükü toplar ve yük dengeleme stratejisine göre sunucu ve bağlantıların yükünü dinamik olarak ayarlar. Böylece ağın performansı artar, sıkışıklık azalır, ağ kaynaklarının kullanımı maksimum olur [31].



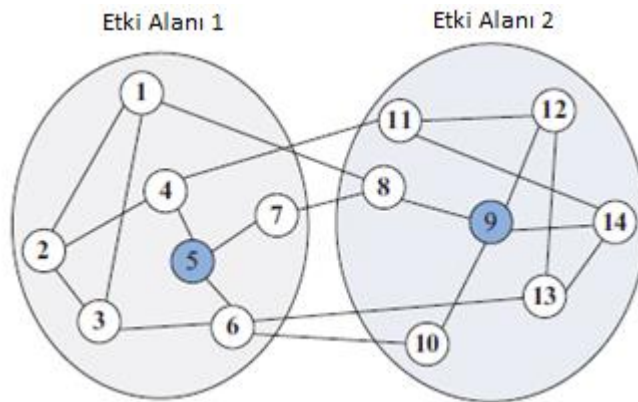
Şekil 1.5. YTA'da yük dengelemenin sınıflandırılması [31]

Dağıtık YTA mimarisinde denetleyiciler iki farklı şekilde konumlandırılabilir: düz (flat) veya hiyerarşik. Düz ve hiyerarşik mimari Şekil 1.6.'da gösterilmektedir. Düz mimaride bütün denetleyiciler aynı seviyededir. Hiyerarşik mimaride ise denetleyiciler farklı seviyelere konumlandırılmıştır [31].



Şekil 1.6. Dağıtık YTA mimarileri: a) Düz (Flat) mimari, b) Hiyerarşik mimari [31]

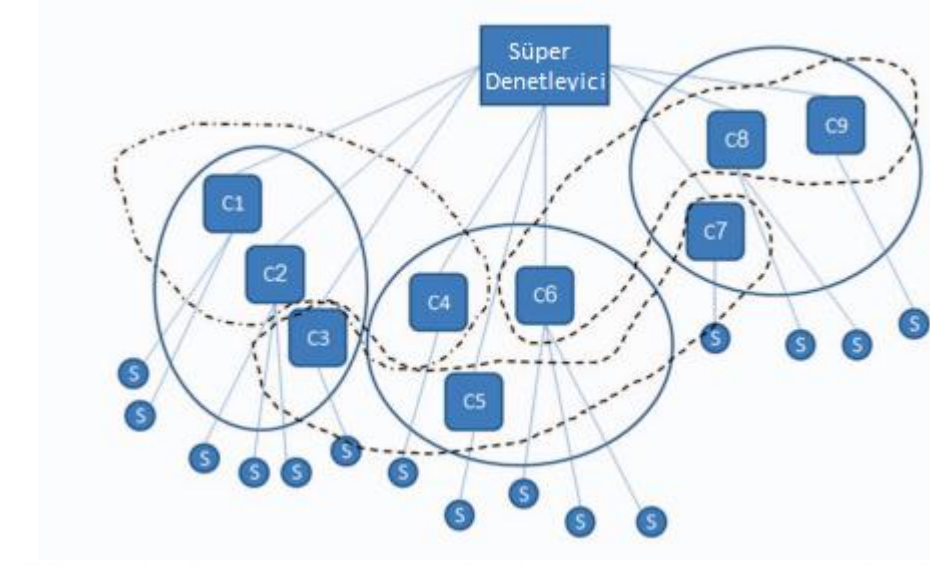
Düz mimaride yer alan yük dengeleme sınıfına dahil olan Anahtar Göçü (Switch Migration) Şekil 1.7.'de ifade edilmektedir. Etki Alanı 1 için denetleyicinin 5 numaralı düğüm, Etki Alanı 2 için ise denetleyicinin 9 numaralı düğüm olduğu varsayılmaktadır. Belirli bir zamanda, Etki Alanı 2'deki denetleyicinin fazla akış alması nedeniyle aşırı yüklendiğini varsayalım. Aşırı yükleme algılandığı zaman anahtar göç mekanizması tetiklenmektedir. Etki Alanı 2'deki denetleyici, komşu denetleyicisiyle birlikte çalışarak geçiş yapmasını istediği anahtarları belirler. Sonuç olarak, 10 numaralı düğüm, Etki Alanı 2'den Etki Alanı 1'e geçirilir. Ancak, geçişten sonra 10 numaralı düğümün, önceki denetleyiciye kıyasla yeni denetleyiciden daha uzakta olması dezavantaj oluşturmaktadır [32]. Bu durumda karşımıza farklı bir problem çıkmaktadır: denetleyicilerin optimum konumlara yerleştirilmesi.



Şekil 1.7. Anahtar göçü [25]

C. Wang ve arkadaşları, anahtar göçü tabanlı karar verme (switch migration-based decision-making - SMDM) çerçevesi önermişlerdir. Bu çerçeve göç masrafları ve yük dengesi oranı arasında bir dengeleme yapmak için oluşturulmuştur. Açgözlü yöntemi temel alan anahtar göçü algoritması, göç verimliliği modelini kullanmak ve dolayısıyla olası göç eylemlerinin seçimini yönlendirmek üzere tasarlanmıştır. Önerilen şema, izleme, yük dengesi tespiti, göç verimliliğini hesaplama, göç stratejisi kararı ve göçü yürütme olmak üzere beş modüle sahiptir. İzleme modülü, gerçek zamanlı yük bilgilerini izler, her denetleyicinin toplam yük değerini hesaplar ve hesaplanan yük verisini yük dengesi algılama modülüne aktarır. Yük dengesi algılama modülü, farklı denetleyicilerin yük çeşitliliğini ölçerek anahtar geçişinin yapılıp yapılmayacağına karar vermektedir. Geçiş verimliliği hesaplama modülü, göç verimliliği ve göç maliyeti arasında dengeleme yapmak için kullanılan bir modeldir. Göç stratejisi karar modülü, göç planının oluşturulmasından sorumludur. Göç yürütme modülü her denetleyiciye yerleştirilir. Görevi, anahtar göçü için göç eylemlerini koordine etmek ve anahtar-denetleyicisi eşleşmesini değiştirmektir [33].

Dinamik yük dengeleme için Sufiev ve arkadaşı yeni bir mimari önermişlerdir. Önerilen mimaride bir tane Süper Denetleyici (Super Controller - SC) ve Sıradan Denetleyicilerden (Regular Controller - RC) oluşan bir kaç küme düşünülmüştür. Her kümede eşit sayıda Sıradan Denetleyici mevcuttur. Şekil 1.8.'de gösterildiği gibi RC'leri kümelere bölmek için farklı olasılık vardır. Çalışmada yük dengelemesi gerektiren iki durum önerilmiştir. Birincisi, kümenin maksimum yük eşiğine dayalı olarak SC tarafından başlatılan kümeler arasındaki dengesizliğin periyodik olarak kontrol edilmesidir. İkincisi, RC'lerden herhangi biri, eşiğine ulaştığında (herhangi bir zamanda olabilir) yükünün bir kısmını aynı küme içerisinde daha az yüklü bir RC'ye aktarmak veya diğer kümelerdeki hangi RC'nin aşırı yükü alabileceğini bilmek için SC'ye bir talep yapmaktır. Yük dengelemeyi sonuç olarak iki farklı düzeyde gerçekleştirmişlerdir; SC'de üst seviyeli işlemler ve RC'de düşük seviyeli işlemler [34].



Şekil 1.8. Dinamik küme mimarisi [34]

Anahtar göçü ve dinamik yük dengeleme problemlerinde mevcut sorunların birisi denetleyicilerin yerleşmesi için uygun konumun nasıl hesaplanacağıdır. Denetleyicilerin yerleşmesi problemini ilk olarak Heller ve arkadaşları ele almışlardır. Ağ gecikmesini minimuma indirmek için k-merkez (k-center) algoritması kullanılmıştır. Çalışmadaki amaç, denetleyici yerleşiminin hem ortalama hem de en kötü durum yayılım gecikmesi üzerindeki etkilerini araştırmaktır [35].

Güner ve arkadaşları yazılım tanımlı mobil ağlarda, denetleyici yerleştirme problemini incelemişlerdir. Çoklu denetleyici yerleştirme ile ilgili dört farklı algoritmayı ele almışlardır.

Kaba Kuvvet Algoritması: Denetleyiciler sırayla tüm düğümlere yerleştirilip aşağıdaki kısıtlara göre uygun konum hesaplanmaktadır. Kısıtların biri ve birkaçı hesaplama için seçilebilir. Her denetleyici için tüm düğümler kontrol edildiğinden hesaplama açısından verimli değildir.

- Düğüm-Denetleyici Arası Ortalama Gecikme
- Denetleyiciler Arası Ortalama Gecikme
- Düğüm-Denetleyici Arası Gecikmelerin En Fazlası
- Denetleyiciler Arası Gecikmelerin En Fazlası

Rastgele Yerleştirme: Denetleyiciler yerleştirilirken hiçbir kısıtlama dikkate alınmaz. Rastgele çalıştığı için başarısının yüksek olması düşük ihtimaldir.

Ağ Çapı Esaslı Yerleştirme (Yarı-Rastgele Yerleştirme): Denetleyiciler yerleştirilirken $L=D/k$ (L = denetleyiciler arası asgari uzaklık, D = topoloji çapı (birbirine en uzak iki düğümün arasındaki mesafe, k = denetleyici sayısı) formülü dikkate alınır. İlk denetleyici rastgele yerleştirilir, sonra düğümler yine rastgele seçilir. Eğer seçilen düğümün yerleştirilen denetleyiciye uzaklığı L 'den büyük ise bir sonraki denetleyici buraya yerleştirilir.

Düğüm Yüğü Esaslı Yerleştirme: Düğümlerdeki trafik yükleri birbirlerinden farklıdır. Algoritma uygulanırken düğümler yüklerine göre sıralanır. Yerleşecek denetleyiciler en az trafik yüküne sahip düğümlerin konumunu seçerler [36].

Şekil 1.5.'te verilen sınıflandırma şemasına dahil olmak istersek, tez çalışması kapsamında merkezi yapılı YTA mimarisi kullanılmıştır. Alt alan olarak veri düzleminde hem sunucu yük dengelemeye hem de bağlantı yük dengelemeye dayalı çalışmalar yapılmıştır. Ayrıca çalışmada sadece yük dengelemeye odaklanılmamış aynı zamanda yönlendirme de yapılmıştır. Yönlendirme yapılırken fil akışlar tespit edilerek verimli bir yönlendirme yapılması amaçlanmıştır. Bu nedenle literatür iki alt başlık olarak detaylı incelenmiştir.

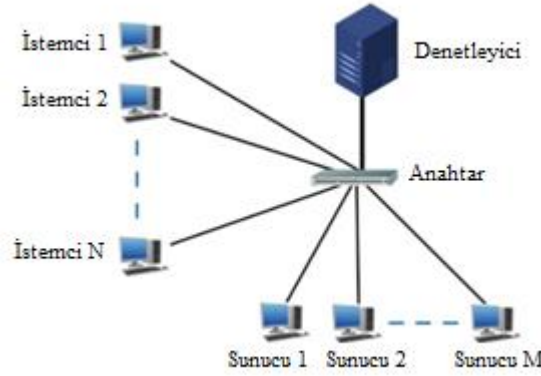
1.4. YTA Yük Dengeleme

YTA'da veri düzleminde uygulanabilen iki farklı yük dengeleme konusu mevcuttur. Bunlar sunucu tabanlı yük dengeleme ve bağlantı tabanlı yük dengelemedir. Tez çalışması kapsamında belirtilen iki yük dengeleme konusu ile ilgili çalışmalar yapılmıştır. Konularla ilgili literatür araştırması bu bölümde aktarılmıştır.

1.4.1. Sunucu Tabanlı Yük Dengeleme

Sunucu tabanlı yük dengeleme şemasında, veri düzlemindeki sunucular arasında yükün eşit olarak dağıtılması hedeflenmektedir. Günümüzde artan internet taleplerine tek sunucu ile yanıt vermek oldukça zordur ve verimliliğin düşmesine neden olmaktadır. Bu nedenle günümüzde aynı işi yapan birçok sunucunun gelen isteklere cevap vermesi alt yapısı sağlanmıştır. Bu durumun sonucu olarak sunucular üzerindeki yüklerin dengeli dağılımı önemli bir konu haline gelmektedir. Şekil 1.9.'da verilen örnek bir ağ topolojisine bakacak

olursak N sayıda istemcinin taleplerini karşılamak üzere aynı özellikte M sayıda sunucu mevcuttur. Sunucu seçimi, aynı anda veya farklı zamanlarda birden fazla istemciden gelen isteklere en hızlı şekilde cevap verecek sunucunun belirlenmesidir. Sunucu yük dengeleme, çoklu sunucular arasından en az meşgul olanı seçme mantığına dayanmaktadır.



Şekil 1.9. Örnek bir ağ topolojisi

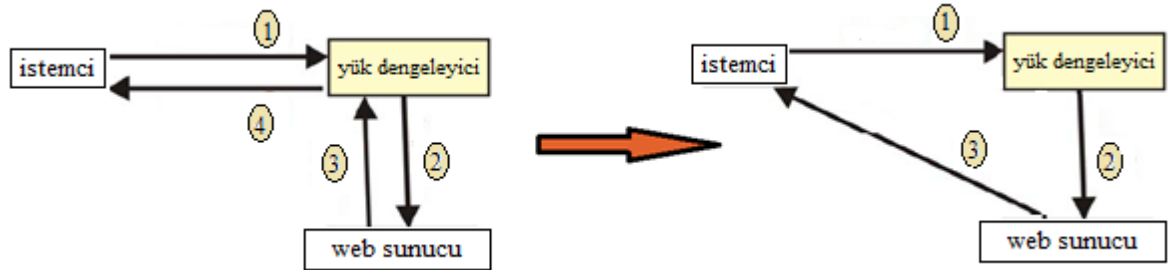
Her sunucu aynı içeriğe sahiptir ve sunucu kümesinin bir parçasıdır. Yük dengeleyicinin sanal IP adresinde bir istek alındığında, bağlantıyı kabul eder ve hedef IP'yi sunuculardan birinin IP adresiyle değiştirir. Yük dengeleyiciden isteği alan sunucu, yük dengeleyicinin IP adresi olarak hedef IP adresiyle bir yanıt gönderir. Ardından, yük dengeleyici sunucudan dönen paketi yakalar ve kaynak IP adresini sanal IP adresiyle eşleştirecek şekilde değiştirir ve ana bilgisayara gönderir. Asıl sorun, hangi sunucunun kullanılacağına yük dengeleyicinin karar vermesi ve bunun yük dengeleme algoritmasına bağlı olmasıdır [37].

Literatürde bu alanda farklı çalışmalar bulunmaktadır. Yapılan çalışmalarda sunucunun kısa tepki süresi farklı mantıklara göre hesaplanmaya çalışılmıştır. Sunucunun yükünü göz önünde bulundurarak az yüklenen sunucuyu seçme mantığına dayalı çok fazla çalışma vardır. Kaur ve arkadaşları Rastgele (Rand), Round Robin (RR) ve yük tabanlı olmak üzere üç tane yük dengeleme tekniği sunmuşlardır. Rand yük dengeleme tekniğine göre cevap için seçilen sunucu rastgele seçilmektedir. RR yük dengeleme tekniğine göre her sunucu sırayla seçilmektedir. Yük tabanlı yük dengeleme tekniğine göre her sunucu için mevcut yük hesaplanıp, minimum yüke sahip sunucu seçilmektedir. Sunulan yük dengeleme

tekniklerinden Rand ve RR yük dengeleme teknikleri ortalama tepki süresi (average response time) ve saniyedeki işlem sayısı (transactions per second) açısından kıyaslanmıştır. Sonuçlar RR yük dengeleme tekniğinin daha iyi olduğunu ortaya koymaktadır. Denetleyici olarak POX kullanılmıştır [38]. Sunucu kapasitesi ve sunucu yanıt süresi gibi parametrelerle sunucu seçim kararını veren çalışmalar da mevcuttur. Sabiya ve arkadaşları ağırlıklı RR tekniğini kullanmıştır. RR tekniğinden farklı olarak her sunucunun kapasitesi dikkate alınmakta ve kapasiteleri oranında istekler gönderilmektedir. Yöntem, RR tekniğiyle ortalama tepki süresi ve saniyedeki işlem sayısı açısından kıyaslanmıştır. Sonuçlar önerilen yöntemin daha başarılı olduğunu göstermektedir. Denetleyici olarak POX kullanılmıştır [39]. Prakash ve arkadaşları dinamik yük dengeleme yöntemi önermişlerdir. Sunucu yükü CPU kullanımı, bellek kullanımı ve mevcut bağlantı sayısı ile belirlenmektedir. Bu parametreler belirli ağırlık katsayıları ile çarpılarak her sunucuya ait yük hesaplanır. Cevap için seçilen sunucu minimum ağırlığa sahip sunucudur. Yöntem, Rand ve RR yük dengeleme teknikleri ile CPU kullanım oranı, bellek kullanım oranı ve verimlilik açısından kıyaslanmıştır. Sonuçlar önerilen yöntemin CPU kullanım oranı ve bellek kullanım oranı açısından dengeli dağılım gösterdiğini, diğer yöntemlere göre daha verimli olduğunu göstermektedir. Denetleyici olarak Floodlight kullanılmıştır [40]. İkrâm ve arkadaşları RR, ağırlıklandırılmış RR, dinamik yük dengeleme ve akış istatistiklerine dayalı yük dengeleme yöntemlerini; verim, tepki süresi, gecikme gibi karşılaştırma metrikleri kullanarak karşılaştırmışlardır. Sonuçlar ağırlıklandırılmış RR yönteminin, RR ve dinamik yük dengeleme yöntemlerinden daha başarılı olduğunu göstermektedir. Fakat akış istatistiklerine dayalı yöntem, test edilen diğer yöntemlere göre daha iyi sonuç vermiştir [41]. Deepika ve arkadaşları yük dengeleme tekniklerini statik yük dengeleme ve dinamik yük dengeleme olarak sınıflandırmışlardır [42]. Statik yük dengeleme yöntemleri uygulama zamanını kısaltır ve iletişim gecikmelerini en aza indirir. RR, Rand, Ana Yönetici (Central Manager) ve Eşik (Threshold) algoritmaları statik yük dengeleme yöntemlerine örnek verilebilir. Ana yönetici yönteminde en az yüke sahip olan köle (slave) işlemci, ana işlemci olarak seçilir ve tüm işi bu işlemci yapar. Ana işlemci diğer işlemcilerin yük bilgilerini toplar ve en iyi kararı vermeye çalışır. Eşik algoritmasında her işlemcide sistem yükünün bir kopyası bulunur ve yeni işlemler için uzak mesajlar göndermeden yerel seçimler yapılmaktadır. Dinamik yük dengeleme yöntemlerinde yük oluştuğunda süreçler dinamik olarak ayarlanır. Merkezi Sıra (Central Queue) ve Yerel Sıra (Local Queue) algoritmaları dinamik yük dengeleme yöntemlerine örnek verilebilir. Merkezi sıra algoritmasında her yeni istek FIFO (First In First Out)

kuyruğunda saklanır. Yerel sıra algoritmasında ise yükün eşik sınırının altına düştüğü bir ana bilgisayar tarafından başlatılan süreç göçü ile tüm yeni süreçler statik olarak tahsis edilmektedir. Statik yük dengeleme yöntemlerinde işlem başarısız olursa başka sunucuya aktarım yoktur yani güvensizdir. Dinamik yük dengeleme yöntemleri; hata toleransı, kaynak kullanımı, bekleme süresi (kuyruk), tepki süresi, verim açısından statik yük dengeleme yöntemlerine göre daha iyidir.

Kaur ve arkadaşının önerdiği yük dengeleme stratejisi trafik tabanlıdır. Her iki saniyede bir, denetleyici anahtarlardan akış istatistik bilgilerini ister. Her sunucunun toplam paket sayısı belirlenir ve en az paketi olan sunucu seçilir. Şekil 1.10.'da klasik yük dengeleme mimarisi ve önerilen yük dengeleme mimarisi gösterilmektedir. Yük dengeleyicinin seçtiği sunucu, istek cevabını yük dengeleyiciye söylemeden direkt olarak istemciye söyler. Böylece yük dengelemenin asıl sorunlarından biri olan gecikme azalır ve ekstra harcanan zaman önlenmiş olur. Yöntem RR yöntemi ile ortalama tepki süresi, saniyedeki işlem sayısı ve ele alınan toplam istek sayısı açısından karşılaştırılmıştır. Sonuçlar önerilen yöntemin daha başarılı olduğunu ortaya koymaktadır [43].



Şekil 1.10. Yük dengeleme mimarisi ve önerilen mimari

Literatürdeki çalışmalar arasında sunucuların bağlı olduğu anahtardan elde edilen akış istatistiklerine göre sunucu seçimi yapan çalışmalar da bulunmaktadır. Akış istatistiklerine bağlı çalışmalardan birisi Kaur ve arkadaşları tarafından 2016 yılında gerçekleştirilmiştir. Yük dengeleme tekniğine göre en düşük akış bağlantısına sahip sunucu seçilmektedir. Her 5 saniyede bir OpenFlow anahtarına akış istatistiklerini öğrenmek için istek mesajı gönderilir ve cevaba göre en düşük aktif bağlantıya sahip sunucu seçilir. Yöntem, RR yük dengeleme tekniği ile tepki süresi (response time), verimlilik (throughput) ve müsaitlik (availability) açısından kıyaslanmıştır. Sonuçlar akış istatistiklerine göre belirlenen yük dengeleme

teknığının daha iyi olduğunu kanıtlar niteliktedir. Denetleyici olarak POX kullanılmıştır [44]. Hamed ve arkadaşları bant genişliği tabanlı bir yük dengeleme tekniği önermişlerdir. Yönteme göre her 14 saniyede bir en düşük bant genişliği tüketen sunucu cevap için seçilir. Yöntem, RR ve bağlantı tabanlı yük dengeleme teknikleri ile kıyaslanmıştır. Bağlantı tabanlı yük dengeleme tekniğinde cevap verecek olan sunucu, minimum aktif bağlantı sayısına göre seçilmektedir. Karşılaştırmalar hem Mininet emülatörü ile hem de gerçek hayatta Raspberry Pi kullanılarak gerçekleştirilmiştir. Önerilen yöntemin daha verimli olduğu ve ortalama tepki süresinin daha kısa olduğu gözlemlenmiştir. Denetleyici olarak Ryu kullanılmıştır [45].

Web sitelerinde artan taleplerle başa çıkmak için Süleymanzade ve arkadaşları yeni bir yaklaşım önermişlerdir. Yaklaşım, karşılık gelen anahtar bağlantı noktasının yanıt süresine ve trafik hacmine bağlı olarak web isteklerini her sunucuya tahsis etmektedir. Merkezileştirilmiş YTA denetleyicisi, sunucular arasındaki yük dağılımının güncel bir görünümünü korumak için bu bilgileri periyodik olarak toplamakta ve gelen kullanıcı isteklerini en uygun sunucuya yönlendirmektedir [46].

Kumari ve arkadaşı literatürdeki bazı yük dengeleme tekniklerinden bahsederek sonraki çalışmalara öncülük etmeyi amaçlamışlardır [47]. Yöntemler karşılaştırma amaçlı olarak performans, kaynak kullanımı, tepki süresi, hata toleransı, gecikme gibi kriterleri esas almaktadırlar. Gerçek zamanlı bir uygulama yapılması planlanıyorsa, statik yük dengeleme teknikleri yerine dinamik yük dengeleme teknikleri tercih edilmelidir. Çoklu denetleyici kullanan topolojilerde yük dengeleme, sunucular arası değil de denetleyiciler arasında olacaksa birbirleriyle haberleşme süreçleri hem iletişim yükünün artmasına hem de bekleme zamanının artmasına sebep olacaktır. Ayrıca çalışmada en fazla kullanılan RR tekniğinin sınırlılıklarını da ele almıştır. Bunlar; gerçek zamanlı test edilmemesi, çoklu denetleyici ile denenmemesi ve sunucuların yükünü dikkate almamasıdır. Çalışma, farklı yöntemlerin sunucu seçim yöntemlerinden de bahsetmiştir. Örnek verirken; kaynak ve hedef arasında bir hash değeri belirlenir. Aynı hash değerine sahip tüm istekler aynı sunucuya gelir. Kaynak ve hedef arasında farklı hash üretilmemesi algoritmanın sınırı olarak göze çarpmaktadır. Farklı bir yöntemle göre seçilecek sunucu isteği kabul mesajını yük dengeleyiciye göndermeden direkt istemciye gönderir. Bu yöntemin amacı tepki süresini minimuma indirmektir.

Literatürdeki çalışmalarda sunucu seçiminin nasıl yapılacağıyla ilgili kullanılan yük dengeleme teknikleri incelenmiş ve inceleme sonuçlarından bazıları aşağıda listelenmiştir;

- ✓ Round Robin Yük Dengeleme,
- ✓ Ağırlıklandırılmış Round Robin Yük Dengeleme,

- ✓ Rastgele Yük Dengeleme,
- ✓ Akış İstatistiklerine Dayalı Yük Dengeleme,
- ✓ Dinamik Yük Dengeleme,
- ✓ Hash Tabanlı Yük Dengeleme,
- ✓ Gecikme Tabanlı Yük Dengeleme,
- ✓ Trafik Tabanlı Yük Dengeleme,
- ✓ Sunucu Yanıt Süresine Bağlı Yük Dengeleme ...vb.

Kullanılacak yük dengeleme yöntemdeki en önemli sınırlama hangi sunucunun seçileceğidir. Literatürde sunucu seçiminde kullanılan birçok parametre vardır. Bu seçim parametrelerinden bazıları aşağıda listelenmektedir;

- Bant genişliği,
- Paket kayıp oranı,
- Hop sayısı,
- Saniyedeki işlem sayısı,
- Ortalama tepki süresi,
- Anahtar bağlantı sayısı,
- Bellek kullanımı,
- İletilen paket sayısı ... vb.

Sunucu seçimi için dikkat edilmesi gereken en önemli husus en doğru sonuca ulaşmak için minimum zaman kullanmaktır. Bu hedefe ulaşmak için sunucu seçiminde hangi parametrelerin kullanılacağı önemlidir. Çalışma için pozitif katkı sağlayan parametreler seçildikten sonra, hedefe ulaşmadaki katkıları belirlenmeli ve yük dengeleme stratejisi uygulanmalıdır.

1.4.2. Bağlantı Tabanlı Yük Dengeleme

Bağlantı tabanlı yük dengeleme yaklaşımlarda, veri düzlemindeki bağlantılar üzerindeki yükün eşit olması hedeflenmektedir. Böylece ağın performansı artmakta, ağ tıkanıklığı azalmakta ve ağ kaynaklarının kullanımı maksimum olmaktadır. Bağlantı tabanlı yük dengeleme yöntemleri de kullandıkları algoritmalarla göre sınıflandırılmaktadır. Bunlar; meta-sezgisel algoritmalar, makine öğrenmesi algoritmaları ve diğerleridir [48].

Literatürde bağlantı yük dengeleme ile ilgili farklı çalışmalar bulunmaktadır. Chen-Xiao ve arkadaşları link yüklerini temel alarak link seçimi yapmaktadır [49]. Burada dikkat edilmesi gereken husus hangi parametrelerin yükü ne derece etkilediğidir. Yöntemde link yükü belirlenirken; paket kayıp oranı (packet loss rate), iletim hopları (transmission hop), bant genişliği kullanım oranı (bandwidth utilization ratio) ve iletim gecikmesi (transmission latency) esas alınmaktadır. Kullanılan parametrelerin yükü etkileme oranları ise geri yayımlı yapay sinir ağları kullanılarak hesaplanmaktadır. Sonuç olarak minimum yüke sahip link seçilmektedir.

Literatürdeki çalışmalardan bir diğeri olan Düşük Maliyetli, Yük Dengeli Rota Yönetimi (Low-cost, Load-balanced Route Management-L2RM) çerçevesi, fat tree veri merkezi ağları için YTA tabanlı yaklaşımlarda önerilen bir rota yönetimidir. L2RM ağ trafiğini izlemekte ve bazı anahtar bağlantılarının ağır yüklerle yüklenip yüklenmediğini kontrol etmektedir. Sonuca göre bir yük sapma parametresi hesaplamaktadır [50]. Farklı bir çalışmada, Wang ve arkadaşlarının geliştirdiği iki farklı algorithmadan birincisi, kontrol bağlantısı yük dengeleme problemini çözmek için tasarlanmış Minimum Maliyetli Akış Yönlendirme tabanlı bir algoritma, diğeri ise düşük gecikmeli yol dağıtım probleminin üstesinden gelmek için önerilen yuvarlama tabanlı bir algoritmadır. Algoritmalar tasarlanırken daha iyi bir Servis Kalitesi (QoS) elde edilmesi hedeflenmektedir [51]. Benzer şekilde, sınırlı yaklaşım faktörlerine sahip yuvarlama tabanlı bir algoritma öneren Yang ve arkadaşları, vswitch (sanal anahtar-virtual switch) dağıtım ve yönlendirme probleminin ortak optimizasyonunu çözmeye odaklanırken [52], Xu ve arkadaşları yuvarlama tabanlı bir rota güncelleme algoritması kullanarak gecikmeye duyarlı rota güncelleme problemini çözmeye odaklanmıştır [53]. Ayrıca, pekiştirmeli öğrenmeye dayalı CFR-RL (Critical Flow Rerouting-Reinforcement Learning- Kritik Akış Yeniden Yönlendirme-Takviye Öğrenimi) yöntemi Zhang ve arkadaşları tarafından ağırlı bağlantı yükünü dengelemek için önerilmiştir [54].

Ek olarak, sezgisel yönlendirme algoritmaları yük dengeleme probleminin çözümünde oldukça etkilidir. Verimli yönlendirme, belirli bir akışın belirli QoS gereksinimlerini karşılayabilecek bir yol bulmak için verimi en üst düzeye çıkarırken maksimum bağlantı yükünü en aza indiren rota seçim algoritmaları gerektirir. Tek başına bant genişliği kısıtlamasıyla bile birden fazla akışı aynı anda yönlendirmek, iyi bilinen bir NP-zor problemdir. Akin ve arkadaşları, belirli bir bölünmez akış kümesinin bant genişliği gereksinimlerini karşılayabilecek birden çok yol bulmak için yeni ve etkili bir buluşsal

çözüm önermektedir [55]. YTA tabanlı Çok Yollu Trafik Planlama Mekanizması (Multi Path Traffic Planning Mechanism-MTSS) [56], özellikle bulut sisteminin ağ tıkanıklığı sorununu çözmek için QoS ve kullanım verimliliğini artırmak için önerilmiştir. Xiaolong ve arkadaşları, MTSS mekanizması ile ağ bağlantısını periyodik olarak izleyerek meşgul bağlantıdaki trafiği dinamik olarak ayarlayan sezgisel bir trafik dengeleme algoritması sunmaktadır. Deneysel sonuçlar, MTSS'nin gecikmeyi ve paket kayıp oranını etkin bir şekilde azalttığını kanıtlamaktadır.

Başka bir çalışmada [57], Xue ve arkadaşları, Genetik Algoritma (GA-Genetic Algorithm)'yı Karınca Kolonisi Optimizasyonu (ACO-Ant Colony Optimization) algoritması ile entegre eden dinamik bir yük dengeleme şeması önermişlerdir. Önerilen şema, ACO yönteminin optimal çözüm için verimli arama kapasitesini ve GA yönteminin hızlı küresel arama özelliğini kullanır. Şemada, önerilen uygunluk fonksiyonu parametresi olarak yol uzunluğu kullanılmıştır. Simülasyon sonuçlarında önerilen şema, optimal rota arama hızı, RTT (Gidiş Dönüş Süresi-Round Trip Time), ve PLR (Paket Kayıp Oranı-Packet Loss Rate) açısından RR ve ACO algoritmaları ile karşılaştırılmakta ve şemanın etkisi gösterilmektedir.

1.5. YTA Yönlendirme

Veri merkezlerinin sağladığı yüksek bant genişliğini verimli bir şekilde kullanmak için trafik yönetiminin önemi bilinmektedir. YTA tabanlı veri merkezlerinde, akış düzeyi yönetimi, veri ve kontrol düzlemleri arasındaki büyük miktarda kontrol mesajı nedeniyle sistem ölçeklenebilirliği ciddi bir sorundur. Son yıllardaki çalışmalarda trafiği etkin şekilde yönetmek için özellikle fil akışı tespiti için trafik sınıflandırmaya odaklanılmıştır. Çünkü, gecikmeye duyarlı fare akışları, ayırım gözetmeyen akış planlaması nedeniyle fil akışlar tarafından sınırlanır. Çok yollu veri merkezleri için trafik yönetimindeki ciddi bir sorun, büyük miktarda veri taşıyan fil akışlarının zamanında tespit edilememesidir. Fil akışı algılama ağ yönetimini güçlendirmek ve ağ performansını iyileştirmek için gerçekleştirilmesi gereken bir stratejidir. Aksi takdirde ciddi ağ tıkanıklıkları kaçınılmazdır.

Fil Akışı (Elephant Flow-EF): Bilgisayar ağlarında, bir fil akışı, TCP (veya başka bir protokol) akışı tarafından kurulan, bayt olarak büyük akışlardır. Fil akışları, sayıca çok fazla olmasalar bile belirli bir süre boyunca tükettikleri toplam bant genişlikleri oldukça yüksektir.

Kısaca; uzun ömürlü, daha az değişken özellikli, trafiğin yüzde olarak daha azını kapsayan ve fazla bant genişliği talep eden akışlardır.

Fare Akışı (Mice Flow-MF): Bilgisayar ağlarında, bir fare akışı, TCP (veya başka bir protokol) akışı tarafından kurulan, bayt olarak küçük akışlardır. Fare akışları sayıca oldukça fazladır, fakat tükettikleri bant genişlikleri küçüktür. Kısa süreli internet akışları örnek olarak verilebilir.

Ağ trafiğinin sınıflandırmasında fil akışı tespiti önemli bir konudur. Fil akışı tespiti sayesinde, tüm akışların işlenmesi engellenerek YTA denetleyicisi üzerindeki yük azaltılır. Denetleyici trafik yönetimini gerçekleştirirken sadece ağ performansını önemli derecede etkileyen fil akışlarını dikkate almaktadır. Yönetimin verimli olmaması ağ arabelleklerin fil akışları ile dolmasına, dolayısıyla da kuyruk gecikmelerine ve paket düşmelerine neden olabilmektedir. Bu nedenle, ağ tıkanıklığını hafifletmek için fil akış tespiti gereklidir [58].

Veri merkezi ağlarında gerçekleştirilen ölçümlere göre;

- Toplam akışların %80'inin birkaç milisaniyeden az sürdüğü ve 10 KB'den daha küçük (fare akışlar) olduğu,
- Trafik hacminin büyük kısmını büyük akışların (fil akışlar, %10 civarı) oluşturduğu gözlemlenmiştir [58].

1.5.1. Fil Akış Tespiti

Literatürde fil akışlarını tespit eden birçok şema bulunmaktadır. Önerilen şemalar arasında Landmark LRU (L-LRU) ön bellek şemasını uygulayan bir çalışma bulunmaktadır [59]. Farklı bir şemada bulanık mantığa dayalı fil akışlarının tespiti için yeni bir kontrol mimarisi olan FuzzyDetec önerilmiştir [60]. Ayrıca Sketches olarak adlandırılan şema sıklıkla fil akışlarını tespit etmek için kullanılır. Böylece fil akışları tespit etmek için gereken hafıza ve hesaplama eforu önemli ölçüde azaltılmış olur. CMS (Count-Min Sketch), akış başına paket sayısını ve bir kümedeki öğelerin sıklığını tahmin etmek için yaygın olarak kullanılır [61]. Sketch ve Space Saving tekniklerini anında bildirim ve örneklenmiş trafik ile çalışacak şekilde uyarlayan diğer bir şema literatürde önerilmiştir [62]. HeavyKeeper olarak tanımlanan algoritmada ise üstel düşüşle sayma stratejisi benimsenmiş ve küçük akışları bozunma yoluyla etkin bir şekilde ortadan kaldırmıştır. Büyük akışlar ise üzerindeki etkiyi en aza indirerek alan-doğruluk dengesine ulaşmıştır [63]. Son olarak gerçek zamanlı ve

dinamik olarak yapılandırmak için şemalarda kullanılan eşik değerinin belirlenmesi üzerine dinamik trafik öğrenme algoritması önerilmiştir [64].

Fil akış tespiti için önerilen çeşitli şemalar farklı uygulama alanları bulmaktadır. Mekânsal bilgi ağının bir parçası olan hava platformlarını üst uydular ve yer cihazlarıyla birbirine bağlayan havadan ağ (Airborne Network-AN)'da dinamik eşik fil akış tespiti problemi ele alınmıştır. Sınırlı ve dinamik olarak değişen bant genişliğine uyum sağlamak için ön sınıflandırılmalı regresyona dayalı akıllı bir yöntem önerilmiştir [65]. Veya LightFDG, dalga boyu bölmeli çoğullama (wavelength division multiplexing-WDM) tabanlı serbest alan optik (free space optical-FSO) bağlantıları aracılığıyla birbirine bağlanan optik kablolu veri merkezi ağlarında (DCN'ler) akış algılama ve akış düzeltmeye yönelik entegre bir yaklaşım önerilmiştir. LightFDG optik olarak her sınıfın akışlarını rack-to-rack (R2R) akışlara dönüştürür. Ardından, R2R-MF ve R2R-EF akışları, sırasıyla fare akışları ve fil akışları sanal topolojilerinin ışık yolları üzerinden ayrı ayrı iletilmektedir [66].

YTA'da fil akışı algılama yöntemi sıklıkla tercih edilmektedir. Bunlardan ilkinde, fil akışı sınıflandırma görevinin denetleyici ve anahtarlar arasında paylaşılmasına izin verilmektedir. Böylece, çoğu fare akışı anahtarlarda filtrelenebilir. Bu durumda çok sayıda sınıflandırma isteği ve kontrolöre sinyal iletimi gönderme ihtiyacı ortadan kalkmaktadır [58]. Farklı bir çalışmada, düşük trafik yükü ile fil akışlarını doğru ve zamanında belirlemek için SDDCN'lerin (software defined data center networks-yazılım tanımlı veri merkezi ağları) sunucu tarafında artımlı öğrenmeyi uygulamaya yönelik olan Ağ Filleri Öğrenicisi ve Çözümleyicisi (Network Elephants Learner and anaLYzer - NELLY) tanıtılmaktadır. Artımlı öğrenme, NELLY'nin değişen ağ trafiği koşullarına uyum sağlamasına ve sınırlı bellek kaynaklarıyla sürekli öğrenme gerçekleştirmesine olanak tanımaktadır [67].

Derin öğrenme yaklaşımları her alanda olduğu gibi YTA'da da ilgi görmektedir. YTA ve derin öğrenmenin birlikte yer aldığı literatürde farklı çalışmalar bulunmaktadır. Bunlardan ilkinde, ince taneli bir akış sınıflandırma yöntemi önerilmektedir. Bu çalışmada, farklı özellikler seçilerek rastgele orman teknolojisi kullanılmıştır. Ayrıca, fil/fare akışlarını tespit etmek için iki seviyeli bir ön sınıflandırma mimarisi önerilmektedir [68]. Farklı bir çalışmada fil akışı algılama yöntemi otomatik algılama yükleme (the autodetect upload-ADU) mekanizmasını benimsemektedir. Ana bilgisayar fil akışını gönderdiğinde, ADU İstemci, YTA denetleyicisine bildirmek için sahte bir kaynak IP adresine sahip paket oluşturur. Böylece ADU Sunucusu fil akışı tanımlamasını tamamlar [69].

YTA'da fil akışlarını tespit etmekle yetinmeyip ayrıca sınıflandırılan akışların verimli bir yönlendirme yapılmasına odaklanan çalışmalar da mevcuttur. Çoğunlukla fil akışı planlamasına odaklanan bu tür yayınlarda, çok az fare akışı planlaması dikkate alınmıştır. Akış tespiti ve yönlendirmeye odaklanan bir çalışmada, verimli bir akış algılama mekanizması önerildikten sonra mevcut bağlantı iş yüküne bağlı olarak fil ve fare akışları için dinamik olarak yolları ayrı ayrı ayarlayan DIFFERENTIATED sChEduling (DIFFERENCE) yaklaşımı önerilmiştir. Bir YTA ağındaki trafiği dengelemek ve mevcut bağlantı kullanım oranı açısından veri akışlarını dinamik olarak yeniden programlamak için algoritma tasarlanmıştır [70]. Farklı bir çalışmada fil akışları yönlendirmek için karınca kolonisi optimizasyonu tabanlı bir teknik önerilmiştir. DPLBAnt adı verilen bu teknik, YTA'da yüksek denetleyici-anahtar yükünü hafifletebilen en kısa yol problemi olarak ifade edilmiştir. Önerilen teknik, ilk olarak hem YTA denetleyicisinde hem de anahtarlarda bir çift sınıflandırıcı kullanarak fil akış tespiti yapar. Ardından, DPLBAnt, tıkanmış bağlantılar için en uygun yolu belirler ve fil akışını buna göre yeniden yönlendirilir [71]. Yönlendirmeye odaklanan farklı bir çalışmada, Verimli Örnekleme ve Sınıflandırma Yaklaşımı (Efficient Sampling and Classification Approach-ESCA) önerilmiştir. ESCA fil akışlarının varış aralığını tahmin eder ve gereksiz örnekleri filtreleyerek örnekleme ek yükünü önemli ölçüde azaltır ve veri akışları arasındaki korelasyonlara dayalı yeni bir denetimli sınıflandırma algoritmasıyla sınıflandırır. Ardından, fil ve fare akışları için yeni bir yük dengeli yönlendirme yaklaşımı LBRouting önerilmiştir [72]. Son olarak, LUNA, akış tamamlanma süresini en aza indirebilen yeni bir sınıf tabanlı yönlendirme stratejisi sunmaktadır. LUNA, akışları boyutlarına göre sınıflandırır. Daha sonra, iletme kurallarını oluşturmak ve her akışı kendi sınıfına göre yönlendirmek için ilişkilendirme kuralları adı verilen bir makine öğrenimi tekniğinden faydalanır [73].

1.5.2. Fil Akışların Planlaması

YTA'da ve geleneksel ağlarda fil akış planlaması fil akış tespiti kadar önemlidir. Bu bölümde tespit edilen fil akışların yönlendirilmesine dayalı literatürdeki yönlendirme şemalarından bazıları sunulmaktadır.

Eşit Maliyetli Çok Yollu Yönlendirme (ECMP-Equal-cost multi-path routing) [74], karma tabanlı rastgele iletmeyi sağlayan, literatürde yaygın olarak benimsenen bir protokoldür. ECMP, paketin tek bir hedefe iletilmesinin, eşit yönlendirme önceliğine sahip

birden çok en iyi yol üzerinden gerçekleştirebildiği bir yönlendirme stratejisidir. Çok yollu yönlendirme, her bir yönlendiricide bağımsız olarak alınan atlama başına yerel bir karar olduğundan, çoğu yönlendirme protokolüyle birlikte kullanılabilir. Birden çok yol üzerindeki trafiğin yükü dengelenerek ağı bant genişliği önemli ölçüde artırabilir. ECMP'nin dezavantajı akış taleplerini dikkate almamasıdır, bu nedenle yalnızca çok sayıda fare akış için iyi çalışmaktadır. Mahout [75], Hedera [76], DevoFlow [77] gibi yönlendirme protokolleri, anahtarların fare akışlarını iletmek için ECMP protokolünü tercih etmektedir.

Literatürdeki diğer bir çalışmada, belirli bir bağlantıda bir tıkanıklık meydana geldiğinde, tıkanıklığı hafifletmek için bir veya birkaç büyük akış dinamik olarak alternatif yollara yönlendirilir. Çalışmanın ana katkısı, IP anahtarları ve yönlendiricilerde herhangi bir değişiklik yapmadan akış başına yeniden yönlendirme sağlayan bir şema tasarlanmasıdır. Tüm akış tabanlı işlemler ve yeniden yönlendirme işlevleri, uç ana bilgisayarlara ve merkezi denetleyicilere yüklenen yazılımlarda uygulanmaktadır. Bu şemaya PROBE (ECMP'ye dayalı Probe ve Reroute) adı verilmiştir. PROBE, alternatif yolları keşfetmek için traceroute benzeri bir yaklaşım kullanır ve akış tabanlı yeniden yönlendirmeyi etkinleştirmek için paket başlıklarını değiştirir. PROBE, mevcut veri merkezi ağlarında kolayca uygulanabilen düşük maliyetli, daha az karmaşıklığa sahip ve uygulanabilir bir şemadır [74].

FlowSeer [78], veri akışı madenciliğini kullanan hızlı, düşük maliyetli bir fil akışı algılama ve planlama sistemidir. Çalışmada başlatılan herhangi bir akışın hızını ve süresini doğru ve hızlı bir şekilde tahmin edebilen akış sınıflandırma modellerinin eğitilmesine imkân tanınmaktadır. Tahmin gerçekleştirmesinde denetleyici ve anahtarlar işbirlikçi olarak çalışmaktadır. FlowSeer daha sonra tahmin edilen bu bilgilerle fil akışlarının yönlendirme politikalarını, taleplere ve dinamik ağ koşullarına göre uyarlayabilmektedir.

Mahout [75], ağ yönetimi için OpenFlow benzeri merkezi denetleyici yaklaşımını izleyen, ancak tasarımını yeni uç ana bilgisayar mekanizmasıyla zenginleştiren, düşük maliyetli ancak etkili bir trafik yönetim sistemidir. Mahout'ta, her uç ana bilgisayarda bulunan dolgu katmanı (shim layer), o ana bilgisayardan kaynaklanan akışları izlemektedir. Bu katman bir fil akışını algıladığında, bant içi sinyalleme mekanizması kullanarak bu akışın sonraki paketlerini işaretlemektedir. Sabit bir eşik değeri belirlenir, gelen akış eğer seçilen eşik değeri geçerse, dolgu katmanı akışın bir fil akış olduğunu belirler ve işaretler. Çalışmada eşik 100 KB olarak belirlenmiştir.

Hedera [76], toplu ağ kaynaklarını verimli bir şekilde kullanmak için çok aşamalı anahtarlama yapısını uyarlanabilir şekilde kullanan dinamik bir planlama sistemidir. Hedera

fil akışı tespiti için periyodik yoklama kullanır. Her t saniyede bir, Hedera denetleyicisi her bir anahtardan akış başına istatistikleri çekmektedir. Büyük kontrol trafiği, ek yüke neden olur ve zayıf ölçeklenebilirliğe sahiptir.

DevoFlow [77], denetim ve küresel görünürlük arasındaki bağlantıyı nazikçe kıran, OpenFlow modelinin bir modifikasyonudur. Gereksiz maliyetler getirmeden yararlı miktarda görünürlük sağlayacak şekilde tasarlanmış ve değerlendirilmiştir. DevoFlow özel yapım anahtar donanımı gerektirir ve anahtarlara ağır bir yük getirir. OpenFlow harika bir konsepttir, ancak özgün tasarımı aşırı maliyete neden olmaktadır. Bu maliyetler; anahtarın kontrol düzlemini çok sık dahil etmenin anahtar uygulama maliyetleri ve OpenFlow denetleyicisini hem akış kurulumlarında hem de özellikle istatistiklerde çok sık dahil etmenin dağıtılmış sistem maliyetleridir.

ESCA [79], iki aşamalı fil akışı tespiti ile verimli bir örnekleme ve sınıflandırma yaklaşımıdır. İlk aşamada ESCA, fil akışlarının varış aralığını tahmin ederek ve bir filtreleme akış tablosu kullanarak fazla örnekleri filtreleyerek örnekleme verimliliğini artırmaktadır. İkinci aşamada, ESCA, örnekleri veri akışları arasındaki korelasyona dayalı olarak yeni bir denetimli sınıflandırma algoritmasıyla sınıflandırmaktadır. Matematiksel analizler, ESCA'nın ilgili şemalardan daha iyi performans gösterdiğini kanıtlamaktadır. Gerçek genel veri merkezi izleri üzerindeki kapsamlı deney sonuçları ayrıca, ESCA'nın daha az örneklenmiş paketler ve daha kısa algılama süresi ile doğru algılama sağlayabildiğini göstermektedir.

Literatürdeki çalışmalar kıyaslandığında fil akış planlama yöntemleri 3 ana başlık altında sınıflandırılabilir. Bunlar; anahtarlardan periyodik istatistik toplanması [80], örnekleme [81] veya pencere tabanlı algoritmalar ve uygulama düzeyindeki değişikliklerdir. Anahtarlardan periyodik istatistik toplanması, küçük ağlarda uygulanabilen yöntemler olarak ele alınmaktadır. Örnekleme veya pencere tabanlı algoritmalar, yüksek overhead içermektedir. Uygulama düzeyindeki değişiklikler ise makine öğrenmesi tekniklerini kullanan yöntemlerdir. Donanımsal veya yazılımsal değişiklikler de olabilmektedir. Tablo 1.2.'de literatürde sıklıkla kullanılan fil akış planlama yöntemlerinin karşılaştırılması verilmektedir.

Tablo 1.2. Literatürde sıklıkla kullanılan fil akış planlama yöntemlerinin karşılaştırması

Yöntem	Kullanılan Sınıflandırma	Kullanılan Yönlendirme	Özellikleri
NELLY [67]			Sunucu taraflı, Artımlı öğrenme (incremental learning).
ECMP [74]			İlk kullanılan yöntem.
Mahout [75]		Çevrimdışı Artan İlk Uygun (Offline increasing first fit)	Sunucu taraflı, Reaktif olarak çalışır, Öğrenmesiz.
Hedera [76]		Benzetilmiş Tavlama (Simulated Annealing) ve Küresel İlk Uygun (Global First Fit)	Akış istatistiklerini kullanır, Reaktif akış tespiti.
DevoFlow [77]			Reaktif akış tespiti, Donanım/yazılım değişikliği.
FlowSeer [78]	Anahtar - C4.5 Denetleyici-CVFDT (Hoeffding tree)	Artan İlk Uygun Akış Zamanlaması (Increasing First Fit Flow Scheduling)	Denetleyici taraflı, Proaktif olarak çalışır, Karma öğrenme.
ESCA [79]	Korelasyon tabanlı C4.5		Denetleyici taraflı, Proaktif olarak çalışır, Toplu öğrenme.

Literatür çalışmaları incelendiğinde fil akış tespiti ve planlaması ile ilgili bazı önemli detaylar aşağıdaki gibi özetlenmiştir;

- Fare akışlar, genellikle ECMP protokolü ile yönlendirilmektedir.
- Fil akışı tıkanıklığa neden olabilir ve gecikmeye duyarlı fare akışlarını etkileyebilir.
- Fil akış tespiti yapılırken, sistem gecikmeye duyarlı ve ölçeklenebilir olmalıdır.
- YTA'da bir fil akışın algılanamaması, bir fare akışın yanlış algılanmasından daha ciddi sonuçlara yol açabilir.
- Fil akışlarının yanlış sınıflandırma maliyeti, fare akışlarının yanlış sınıflandırılma maliyetinden daha ciddidir.
- Fil akışı tespitinin kilit noktası, fil akışları en kısa sürede ve verimli bir şekilde tespit etmektir.

- Anahtarların sınırlı hesaplama gücü nedeniyle, anahtar tarafında sadece kural tabanlı ve karar ağacı tabanlı sınıflandırıcılar kullanılabilir.
- İki aşamalı sınıflandırıcıların (anahtar ve denetleyici taraflı) daha verimli sonuç ürettiği gözlemlenmiştir.
- Fare akışlar bazı çalışmalarda önemsenirken, bazılarında hiç hesabı katılmamıştır.
- Fil akış tespiti için sabit eşik değeri kullanılması, dinamik yapıda olmayı engellemektedir.

1.6. Tezin Amacı, Kapsamı ve Sunulan Katkıları

Tez çalışması kapsamında YTA'da en önemli konulardan biri olan yük dengeleme konusu ele alınmaktadır. Çeşitli yönleriyle değerlendirilen yük dengeleme kapsamlı olarak incelenmiştir. YTA'da merkezi yapıda ve veri düzleminde yük dengeleme, sunucu tabanlı yük dengeleme ve bağlantı tabanlı yük dengeleme olmak üzere ikiye ayrılır. Sunucu tabanlı yük dengeleme, gelen taleplerin aynı özelliklere sahip sunucular arasında dengeli dağılmasını amaçlamaktadır. Tez çalışmasında, bu amacı gerçekleştirmek için özgün bir karar fonksiyonu tasarlanmış ve her zaman en az yüke sahip sunucunun seçilmesi hedeflenmiştir. Bağlantı tabanlı yük dengeleme ise gelen akışların bağlantılar üzerinde dengeli dağılmasını amaçlamaktadır. Literatürden farklı olarak, bağlantı tabanlı yük dengeleme konusu ele alınırken yükün hem bağlantılarda hem de anahtarlarda dengeli dağılması sağlanmıştır. Diğer bir deyişle, uygulanan sezgisel yöntemin maliyet fonksiyonu bağlantılardan ve anahtarlardan istatistiksel veriler olarak hibrit olarak tasarlanmıştır. Ayrıca yöntem, sürekli olarak değişen ağ trafiğine uygun olması için dinamik yapıdadır.

Bağlantı tabanlı yük dengelemede her akışın aynı yönlendirme stratejisini kullanması gerçekçi değildir. Ağ trafiğinde daha az sayıda fakat daha fazla bant genişliği tüketen akışlar olduğu gibi, sayıca daha fazla fakat daha az bant genişliği tüketen akışlar da mevcuttur. Bu nedenle akışların planlanması yapılmadan önce akışlar sınıflandırılmaktadır. Literatürde özellikle fil akış tespiti üzerine çalışmalar var olduğundan, tez çalışması kapsamında da fil akış tespiti konusu irdelenmiştir. Farklı sınıflandırıcılar kullanılarak detaylı analizler yapılmıştır. Analizler sonucunda başarılı olarak değerlendirilen yöntemler akış planlaması yapılmadan önceki süreçte sınıflandırma için kullanılmıştır. Akışlar sınıflandırıldıktan sonra fil akışlar ve fare akışlar için farklı yönlendirme stratejileri uygulanmıştır.

2. YAPILAN ÇALIŞMALAR, BULGULAR VE İRDELEME

2.1. Giriş

Programlanabilir ağlar fikri, YTA'nın ortaya çıkması nedeniyle son zamanlarda önemli bir ivme kazanmıştır. Ağ programlanabilirliği, yeniliği etkinleştirmeyi ve ağ yönetimini önemli ölçüde basitleştirmeyi vaat etmektedir [82]. YTA'da veri düzlemi, kontrol düzlemi ve uygulama düzlemi olmak üzere üç düzlem bulunmaktadır. Veri düzleminde anahtarlar, ana bilgisayarlar ve kontrol düzleminde denetleyici(ler) vardır. Tez kapsamında bu iki düzlemdeki ağ öğelerini içeren ağ topolojisini oluşturmak için Mininet (Mini Network) emülatörü kullanılmaktadır. Denetleyici, ağın genel bir görünümüne sahiptir ve ağı, bu görünümü kullanarak yönetir. Bu tez çalışmasında Python tabanlı açık kaynak kodlu POX (Python OpenFlow eXercise) denetleyicisi tercih edilmiştir. Son olarak YTA mimarisinin en üst düzlemi olan uygulama düzlemi çeşitli uygulamaları desteklemektedir. Tez kapsamında, YTA'nın önemli uygulama alanlarından biri olan yük dengelemeye katkı sağlanmıştır. Hem sunucu tabanlı yük dengeleme hem de bağlantı tabanlı yük dengeleme ele alınmaktadır. Sunucu tabanlı yük dengelemede yükün sunucular arasında dengeli dağıtılması ve dolayısıyla taleplerin en hızlı şekilde cevaplanması amaçlanmaktadır. Bağlantı tabanlı yük dengelemede ise amaç en az yüke sahip bağlantılar üzerinden yönlendirmenin gerçekleştirilmesidir. Diğer bir deyişle, en doğru rotayı bulmak ve verileri en hızlı şekilde hedefe iletmektir. Bu hedef doğrultusunda akış sınıflandırma konusu da incelenmiş ve fil akışların tespiti üzerine çalışmalar yapılmıştır. Akış tespitinin yönlendirme üzerindeki etkisi başarılı bir şekilde kanıtlanmıştır.

2.2. YTA Sunucu Tabanlı Yük Dengeleme

Artan internet taleplerini tek bir sunucu ile karşılamak oldukça zahmetli bir iştir. İşlem yükünü karşılamak için istemci isteklerinin yoğunluğuna bağlı olarak çok sayıda sunucu ve ağ cihazı kullanılması gerekmektedir. Bu durum, kullanıcılar tarafından yapılan isteklerin sunucular üzerinde eşit olarak dağıtılabilmesi, kullanıcıya verilen hizmetin hızlı ve tatmin edici olmasını sağlayacaktır. Yükün sunucular arasında eşit olarak dağıtılabilmesi için yük dengeleme yöntemleri kullanılmaktadır. YTA'da yük dengeleme, belirlenen yük dengeleme

stratejisine bağılı olarak yükün farklı sunucular arasında dağıtılması olarak tanımlanmaktadır. Yük dengeleme stratejisi uygulanırken seçilen sunucu tarafından gelen isteklere en kısa sürede cevap verilmesi hedeflenir. Trafik akışı sürekli değiştiği için tahmin ile yönlendirme çok zordur. Bu nedenle akış geldiğinde seçim kararının bir an önce verilmesi gerekir. Karar verilirken sunucuların bağılı olduğu anahtar portlarından periyodik olarak istatistiksel bilgiler toplanır ve belirlenen karar fonksiyonuna göre sunucu seçilir. Yük stratejisinin başarısını belirleyen en önemli adım, karar fonksiyonunda kullanılan parametrelerin ve bu parametrelerin fonksiyona ne ölçüde katkı sağladığını belirlemektir. Tez çalışması kapsamında sunucu seçimi için kullanılan karar fonksiyonu özgün olarak tasarlanmıştır. Sunucu, değişen trafik akışına uyum sağlamak için tasarlanan karar fonksiyonuna göre dinamik olarak seçilmektedir.

Uygulanacak yük dengeleme stratejisi, yükün dengeli dağılımını sağlayacak ve kullanıcı memnuniyetini en üst düzeye çıkaracak şekilde tasarlanmalıdır. Çalışmada, yük dengeleme stratejilerinin hedefleri doğrultusunda yenilikçi bir yaklaşım geliştirilmiştir. Bu yaklaşım, her bir sunucuyu puanlamaya dayanmaktadır. Sunucunun puanı, yükü kendi üzerine alıp almayacağını belirlemede önemli rol oynamaktadır. Geleneksel yöntemler belirli bir kritere göre yönlendirme yaparken çalışmada önerilen yaklaşım, ağ için önemli olan üç farklı kriteri kullanır. Bu nedenle yükün doğru sunucuya iletilme hızının geleneksel yöntemlere göre daha yüksek olması beklenmektedir.

Önerilen yük dengeleme stratejisinde, özgün olarak tasarlanan karar fonksiyonuna göre sunucu seçimi yapılır. Seçim yapılırken sunucuların bağılı olduğu anahtarların portlarından elde edilen akış istatistiklerine göre karar verilir. Akış istatistikleri, her 10 saniyede bir bağlantı noktalarından toplanmaktadır. Anahtarlardan üç farklı akış bilgisi toplanır;

- bayt miktarı
- paket sayısı
- düşme (drop) sayısı

Bayt miktarının ve paket sayısının yoğun olduğu sunucularda yük fazladır. Aynı zamanda düşen paket sayısı fazla ise bu genellikle yük yoğunluğundan kaynaklanmaktadır. Dolayısıyla bu üç parametrenin seçilen sunucu için minimum değerde olması beklenir. Önerilen yöntemde yükün dengeli dağılımında önemli olan bu parametreler kullanılarak etkin bir puanlama mantığı oluşturulmuştur. Bu üç parametre ile oluşturulan özgün karar

fonksiyonu denklem 2.1 ile ifade edilmekte ve bu fonksiyon yardımıyla puanlama yapılmaktadır.

Paket sayısı gönderilen ve alınan paketlerin toplam sayısını, bayt miktarı gönderilen ve alınan baytların toplam sayısını ve düşen paket sayısı sunucuda düşen paketlerin sayısını ifade etmektedir. Minimum paket sayısına sahip sunucu; $sunucu_x$ olsun, minimum bayt miktarına sahip sunucu; $sunucu_y$ olsun ve minimum düşen paket sayısına sahip sunucu; $sunucu_z$ olsun. Bu durumda minimum değerlere sahip sunucuların skorları bir artırılarak, maksimum skora sahip sunucu seçilmektedir. Bir sunucu için maksimum puan 3 olabilir.

$$Karar\ Fonksiyonu = \max_{\text{skor_sunucu}} \begin{cases} skor(sunucu_x) + 1 \\ skor(sunucu_y) + 1 \\ skor(sunucu_z) + 1 \end{cases} \quad (2.1)$$

Bu alanda daha önce puanlama mantığı kullanılmamıştır. Bu nedenle, bundan sonraki çalışmalara da öncülük etmesi beklenmektedir.

Amaç gelen talebe her zaman minimum sürede cevap vermektir. Bunu yaparken de yükün sunucular arasında eşit olarak dağıtılması esastır. Sunucu seçimi için önerilen bu yeni yaklaşım geleneksel sunucu seçim yöntemleri ile karşılaştırılmıştır. Ve sonuçlar, tasarlanan karar fonksiyonunun yanıt süresini azalttığını ve verimi artırdığını kanıtlar niteliktedir.

2.3. YTA Bağlantı Tabanlı Yük Dengeleme

Bağlantı tabanlı yük dengeleme gerçekleşirken, gecikmeyi azaltma ve ortalama tepki süresini en aza indirme gibi özelliklere sahip olan Ayrık Parçacık Sürü Optimizasyonu (D-PSO) yöntemi tez çalışmasında tercih edilmiştir. D-PSO'nun doğru sonuca hızlı bir şekilde ulaşabilmesi için sürecin en önemli adımı uygunluk fonksiyonun hesaplanmasıdır. Tez kapsamında uygunluk fonksiyonu için maliyet fonksiyonu kullanılmaktadır. Amaç, en az maliyetli yolu seçmektir. Fonksiyon tasarlanırken hem anahtarlardan hem de bağlantılardan periyodik olarak istatistikler alınmaktadır. Fonksiyonda kullanılacak metrik sayısı da en az metrikler kadar önemlidir. Metrik sayısının fazla olması hızı olumsuz yönde etkilerken, metrik sayısının az olması doğru sonuca ulaşmayı olumsuz yönde etkilemektedir. Maliyet fonksiyonunda kullanılan metrikler RTT, bant genişliği, paket kaybı sayısı ve anahtarın sahip olduğu port sayısıdır. Önerilen yöntem, aynı yolu seçen çok fazla paketin neden olduğu

paket tıkanıklığını, artan bekleme sürelerini ve paket kayıplarını ortadan kaldırmaktadır. Ayrıca, dinamik yapıda olduğu için ağda değişen koşullara uyum sağlamaktır. Önerilen dinamik bağlantı yük dengeleme yöntemini, diğer çalışmalardan ayıran en önemli özelliği hem anahtarlar hem de bağlantılar üzerindeki yükü aynı anda dengeleyebilmesidir. Çünkü anahtarlar da yük dengeleme bileşenidir ve görmezden gelinmemelidir.

Simülasyon sonuçları, seçilen yöntemin ve özgün olarak tasarlanan maliyet fonksiyonunun YTA yük dengelemede başarılı olduğunu kanıtlamaktadır. Sonuçlar, önerilen yöntemin daha düşük RTT değerleri ve daha az paket kaybı ile yük dengelemeyi başarılı bir şekilde gerçekleştirdiğini göstermektedir. Karşılaştırmalara ek olarak, anahtarlarda ve bağlantılarda yükün dengeli dağılımını gösteren grafikler, sonuçlarda verilmiştir.

Önerilen çalışmanın temel katkılarını özetlemek gerekirse;

- D-PSO, YTA’da dinamik yük dengelemede yol seçiminde ilk kez kullanılmıştır.
- Eşzamanlı bağlantı ve anahtar yük dengelemesi yapılmış ve simülasyon sonuçları ile başarısı kanıtlanmıştır.
- Bağlantılardan ve anahtarlardan alınan istatistikleri içeren hibrit bir maliyet fonksiyonu tasarlanmıştır.
- Oluşturulan trafik, farklı simülasyon sürelerinde test edilerek simülasyon sonuçlarının zamandan bağımsız olduğu kanıtlanmıştır.
- Düşük bellek gereksinimi ve düşük güç tüketimi ile yük dengeleme sağlanmıştır.
- Başarılı yük dengeleme için kayıpsız veri iletimi simülasyon sonuçları ile kanıtlanmıştır.

Çalışmada, denetleyici olarak POX, ağ topolojisini oluşturmak için Mininet emülatörü kullanılmıştır. POX denetleyici, Python tabanlı açık kaynak kodlu bir YTA denetleyicisidir. POX, YTA anahtarlarıyla iletişim kurmak için OpenFlow veya OVSDB protokolünü kullanan bir çerçeve sağlar. Yeni ağ uygulamalarının hızlı gelişimi için kullanılır. POX [83], web tabanlı GUI desteği sağlayarak uygulama modelinin daha kolay geliştirilmesine yardımcı olmaktadır. Oluşturması ve anlaması daha kolay olan kullanıcı gereksinimleri uygulaması geliştirmek için Python’u destekler. Sonuç olarak, bu araç YTA uygulamalarını programlamak ve araştırmak için gereklidir.

Mininet [84] bir ağ emülatörüdür ve bu emülatör kullanılarak sanal ana bilgisayarlar, anahtarlar, denetleyiciler ve bağlantılardan oluşan bir ağ topolojisi oluşturulabilir. Mininet ana bilgisayarları Linux çekirdeği üzerinde çalışır ve anahtarlar açık akış protokolünü

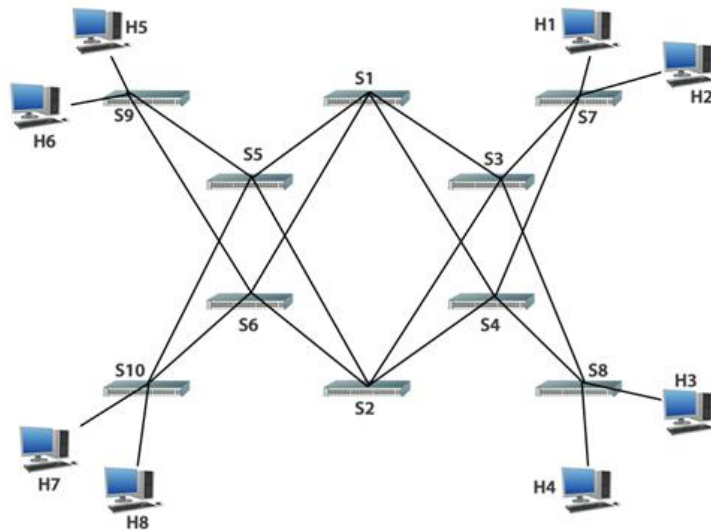
destekler. Python'a dayanmaktadır. Mevcut Mininet'te, önceden tanımlanmış topolojileri kullanmanın yanı sıra özel bir topoloji oluşturulabilir.

Mininet, diğer ağlarla paylaşılabilir ve etkileşime girebilir. Mininet üzerindeki bir ana bilgisayar, işletim sistemi üzerinde çalışan kendi ağı ile temel süreç olarak ifade edilebilir. Her biri, özel bir sahiplik sanal ağ arabirimi, adresler ve yönlendirme tabloları (ARP ve IP gibi), bağlantı noktaları içeren işlemler sağlar. Mininet simülasyonunda, denetleyiciler gerçek veya simüle edilmiş bir ağ üzerinde çalıştırılabilir, yalnızca anahtarların çalıştığı makinenin denetleyiciye bağlanması gerekir [85].

Önerilen yöntemde kullanılan bileşenler alt bölümlerde detaylı olarak anlatılmaktadır.

2.3.1. Kullanılan Ağ Topolojisi

Fat tree ağı evrensel bir şekilde etkili iletişim sağlar. Fat tree ağ topolojisi, çok yollu özellikleri sayesinde veri merkezi ağlarında tercih edilmektedir. Bu topoloji, artan bant genişliği ve hata toleransı gibi temel özellikleriyle dikkat çeker. Ağaç veri yapısı kullanıldığında her dal, hiyerarşideki konumundan bağımsız olarak aynı kalınlığa ve eşit bant genişliğine sahiptir. Ancak fat tree yapısında, hiyerarşinin en üstündeki dalların bant genişliği, alt dallardan daha büyük olur. Bu nedenle üst dallar daha geniş bir kapasiteye sahiptir, yani daha şişmandır [86]. Bu tez çalışmasında, fat tree topolojisi üç katmanlı olarak kullanılmıştır ve Şekil 2.1.'de verilmiştir.



Şekil 2.1. Fat tree ağ topolojisi

Bu ağ topolojisinde, on adet anahtar ve sekiz adet ana bilgisayar bulunmaktadır. S1 ve S2 anahtarları çekirdek (core) anahtarları olarak işlev görmekte, S3, S4, S5 ve S6 anahtarları toplama (aggregation) anahtarlarıdır ve S7, S8, S9 ve S10 anahtarları ise kenar (edge) anahtarlarıdır. Tez çalışmasının amacı doğrultusunda, çekirdek anahtarlar ile toplama anahtarları arasında 1000 Mbps bant genişliği belirlenmiştir. Ayrıca, toplama anahtarları ile kenar anahtarları arasında ise 100 Mbps bant genişliği sağlanmıştır. Bağlantılar iki yönlü iletişimi desteklemekte olup, tüm bağlantıların uzunlukları eşittir.

2.3.2. Uygulanan Ayrık Parçacık Sürü Optimizasyonu (D-PSO)

Parçacık Sürü Optimizasyonu (PSO), kuş sürülerinin hareketlerinden esinlenerek geliştirilmiş sürü tabanlı bir sezgisel optimizasyon algoritmasıdır [87]. PSO, en uygun çözüm etrafında toplanmayı hedefleyerek çözüme ulaşmayı sağlar. Temel olarak, PSO'da sürüdeki bireylerin konumları, sürünün en iyi konumlanmış bireyine yaklaşma stratejisi üzerine kuruludur. Bu yaklaşma hızı rastgele gelişen bir durum olup, genellikle bireyler yeni hareketlerinde önceki pozisyonlarına göre daha iyi bir konuma ilerlerler. Bu süreç, hedefe ulaşılan kadar devam eder ve çözümün gelişimini sağlar. PSO'da hız vektörü (V) ve yeni konum vektörü (X), sırasıyla denklem 2.2 ve denklem 2.3 kullanılarak hesaplanmaktadır.

$$V_{i+1} = V_i + c_1 * rand_1 * (pbest - X) + c_2 * rand_2 * (gbest - X) \quad (2.2)$$

$$X_{i+1} = X_i + V_i \quad (2.3)$$

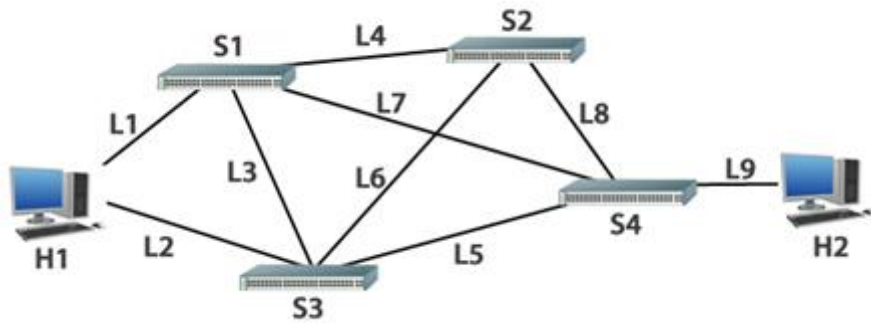
Denklem 2.2'de c_1 ve c_2 değerleri 0 ile 2 arasında değişen sabit değerlerdir, $rand_1$ ve $rand_2$ değerleri ise 0 ile 1 arasında değişen rastgele değerlerdir. Çözüme en yakın parçacık $Pbest$ (en iyi kişisel konum) ile ifade edilirken, tüm parçacıklar arasında çözüme en yakın olan parçacık $Gbest$ (en iyi küresel konum) ile ifade edilmektedir.

YTA yük dengelemede temel amaç, darboğazları önleyerek bağlantılardaki yükü dengeli bir şekilde dağıtmak ve hedefe hızlı iletimi sağlamak için en uygun yolu seçmektir. Yönlendirme, belirli bir akış için en verimli yolu seçerken aynı zamanda bağlantı yükünü en aza indirmeyi hedefler. Bu çalışmada, etkili bir sezgisel çözüm yöntemi önerilmektedir ve seçilen algoritmanın avantajları aşağıda sıralanmıştır [88];

- Enerji tasarrufu
- Gecikmeyi azaltmak
- QoS'yi iyileştirme
- Ortalama yanıt süresini en aza indirme
- Verimi en üst düzeye çıkarmak
- Kullanımı en üst düzeye çıkarmak

PSO, başlangıçta Seyahat Satıcısı Problemi (TSP) gibi ayrık optimizasyon problemlerini çözmek için kullanılmıştır [89]. Ancak, PSO yönteminden farklı olarak D-PSO yöntemi, hız ve konum vektörlerinin hesaplanması açısından farklılık gösterir. D-PSO'da, hız ve konum vektörlerini hesaplamak için kullanılan denklemler, orijinal PSO'da sürekli optimizasyon problemleri için tasarlanmıştır ve bu nedenle ayrık değerlerin kullanıldığı problemler için uygun değildir. D-PSO'da, tam sayı dizileri kullanılarak gerçek sayılar yerine ayrık değerler temsil edilir. Önerilen algoritmada geleneksel toplama ve çıkarma operatörleri uygulanamadığı için hız ve konum güncelleme denklemleri ayrık problemlere göre değiştirilmiştir [90, 91].

Tez çalışmasında, kaynak ana bilgisayardan hedef ana bilgisayara ulaşmak için en az meşgul olan yol araştırılmaktadır. Yön bulma, ağ topolojisine bağlıdır. Bu bölümde, D-PSO'nun uygulanması örnek bir ağ topolojisi üzerinden açıklanmaktadır. Örnek ağ topolojisi Şekil 2.2.'de gösterilmiştir.



Şekil 2.2. Örnek bir ağ topolojisi

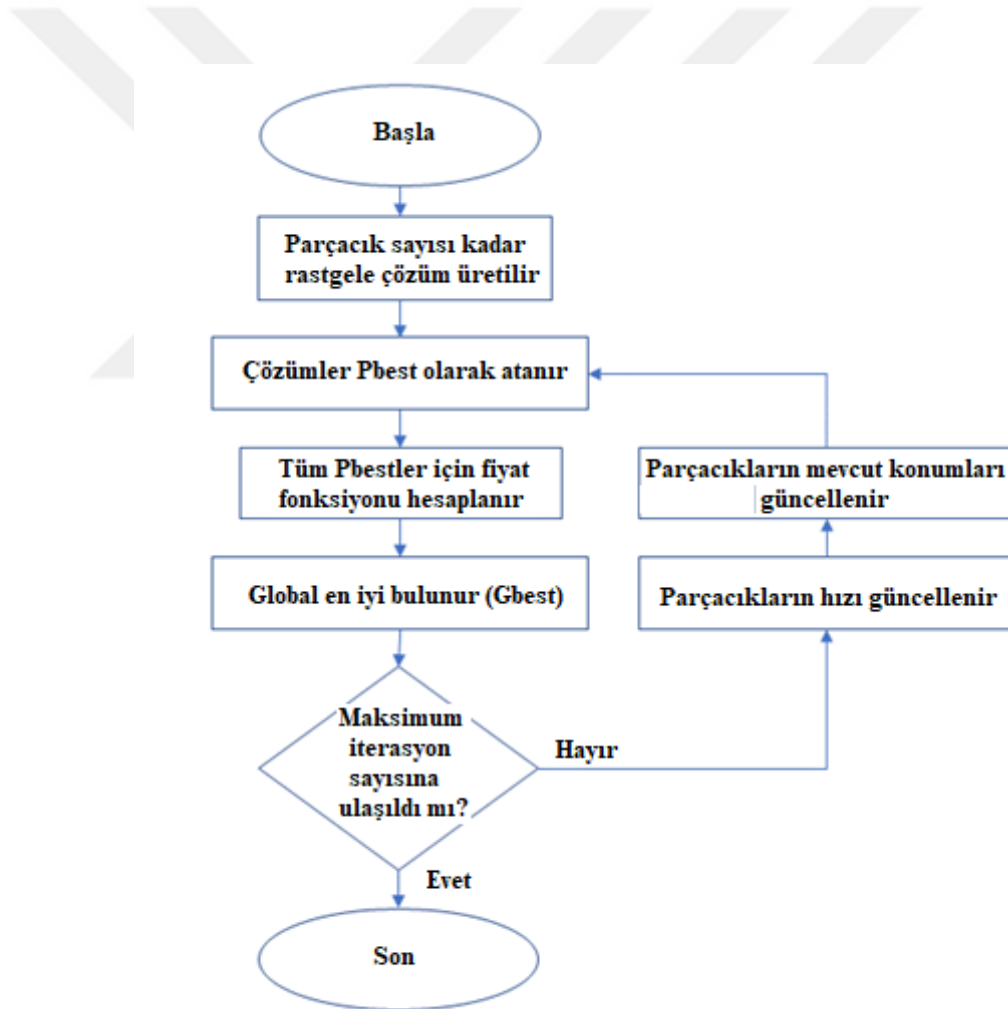
D-PSO algoritmasında, ilk adım olarak ağ topolojisi bir grafa dönüştürülür. Bu graf, $G = (D, K)$ olarak ifade edilen iki kümeden oluşur. D, düğümler kümesini temsil ederken K,

kenarlar kümesini temsil etmektedir. Bu durumda, anahtarlar düğümlere ve bağlantılar kenarlara atıfta bulunmaktadır. Örnek bir topoloji için oluşan kümeler aşağıdaki gibi olabilir.

$$D = \{S1, S2, S3, S4\}$$

$$K = \{L1, L2, L3, L4, L5, L6, L7, L8, L9\}$$

K kümesindeki bazı kenarlar $L1 = (H1, S1)$ ve $L7 = (S1, S4)$ gibi ifade edilmektedir. Ana bilgisayarlar, kaynak ve hedef olarak belirlenir. Kaynak ve hedef arasındaki herhangi bir yol, parçacık adı verilen olası bir çözümü temsil eder. Şekil 2.2.'deki topolojiye göre, örnek parçacıklar $[S1\ S3\ S2\ S4]$, $[S3\ S4]$ vb. olabilir. D-PSO algoritması, tüm yolları tek tek kontrol etmek yerine hızlı bir şekilde çözüme yakınsamayı sağlayarak en uygun çözümü arar. D-PSO algoritmasının akış diyagramı Şekil 2.3.'te gösterilmiştir.



Şekil 2.3. Uygulanan D-PSO algoritmasının akış şeması

Uygulanan D-PSO algoritması akışı, birkaç olası çözümle başlar. Parçacık sayısı, yani olası çözümlerin sayısı, ağa bağlı olarak belirlenmektedir. Daha fazla parçacık daha doğru bir çözüm sağlamaktadır. Fakat hesaplama yükünü artırdığı için işlem süresi de artmaktadır. Bu nedenle, parçacık sayısını belirlemek önemlidir. Şekil 2.2.'de verilen örnek ağ topolojisi için parçacık sayısının iki olduğu varsayılmaktadır. Başlangıç parçacıkları, kaynaktan hedefe giden çözümler arasından rastgele belirlenmektedir. Algoritmanın ilk aşamasında seçilen rastgele çözümler Pbest değerlerini temsil etmektedir. Yani başlangıç parçacıkları Pbest değerlerine eşittir. Rastgele seçilen başlangıç parçacıkları $Pbest_1 = [S1 \ S4]$ ve $Pbest_2 = [S3 \ S2 \ S4]$ olsun.

Daha sonra, tüm Pbest'ler için uygunluk fonksiyonu hesaplanır. Çalışmada, uygunluk fonksiyonu maliyet fonksiyonudur. Yöntemin amacı, en az maliyetli yolu seçmektir. Maliyet fonksiyonu her bir Pbest için hesaplanmaktadır. Hibrit tasarımlı maliyet fonksiyonunun hesaplaması Bölüm 2.3.3'te detaylandırılmıştır. Tüm Pbest'ler arasında en az maliyetli çözüm Gbest olarak adlandırılmaktadır. Örnek için Pbest_2 değeri için maliyet fonksiyonu daha düşük hesaplınsın. Bu durumda, Gbest değeri $[S3 \ S2 \ S4]$ olası çözüme denk gelmektedir. Gbest değeri, denklem 2.4'te temsil edilmektedir.

$$Gbest \in D \quad (2.4)$$

D: Daha iyi Pbest çözümündeki anahtarlar gelip güncellenene kadar, en iyi Pbest çözümündeki anahtarlar kümesi.

Önerilen algoritmada çözüm ararken ki amaç, Pbest değerlerini Gbest değerine yaklaştırmaktır. Parçacıkların konumları bu amaçla fark operatörü yardımıyla yeni Gbest'e göre güncellenir. Buradaki fark operatörü, gerçek değerlerin çıkarılması değildir. Bu fark, Gbest değerinde olup Pbest değerinde olmayan anahtarları içerir ve bu farka hız denir. Hız, denklem 2.5 kullanılarak hesaplanmaktadır.

$$if \begin{cases} Gbest(i) \neq Pbest_k(j) \\ otherwise \end{cases} \quad \begin{matrix} then \\ then \end{matrix} \quad \begin{matrix} V_i += Gbest(i) \\ nothing \end{matrix} \quad (2.5)$$

Denklem 2.5'de ifade edilen i değeri, 0 ile $\vartheta(Gbest)$ arasında değişen sabit bir sayıdır ve j değeri, 0 ile $\vartheta(Pbest_k)$ arasında değişen bir sabit sayıdır. Sabit sayı k, mevcut yolları temsil eder ve 0 ile parçacık sayısı arasında bir değerdir. Dizinin boyutu ϑ ile gösterilir.

Ayrıca hız, küresel en iyi ve k. mevcut en iyi yol sırasıyla V_i , G_{best} ve P_{best_k} ile temsil edilmektedir. Denklem 2.5'e göre P_{best_1} için hesaplanan $V_1 = \{S3, S2\}$, P_{best_2} için hesaplanan $V_2 = \{\}$ (boş küme)'dir.

Parçacığın mevcut konumu hesaplanan hız vektörü ile güncellenir ve yeni konum denklem 2.6 kullanılarak hesaplanır. Güncelleme işleminde hız vektörü ve mevcut konum toplanır. Buradaki toplama geleneksel toplamadan farklıdır çünkü vektörler ayırık değerlerden oluşmaktadır. Hız vektöründeki anahtar kümeleri, olası çözümlerdeki anahtar kümelerine dahil edilir.

$$if \begin{cases} V_i(l) \neq P_{best_k}(j) \\ otherwise \end{cases} \quad \begin{matrix} then \\ then \end{matrix} \quad \begin{matrix} X_i = P_{best_k}(j) + V_i \\ nothing \end{matrix} \quad (2.6)$$

Denklem 2.6'da ifade edilen i değeri, 0 ile $\vartheta(V)$ arasında değişen sabit bir sayıdır. Ve l değeri, 0 ile $\vartheta(V_i)$ arasında değişen sabit bir sayıdır. Sabit sayı k , mevcut yolları temsil eder ve 0 ile parçacık sayısı arasında bir değerdir. Dizinin boyutu ϑ ile gösterilir. Ayrıca, i . hız, k . mevcut en iyi yollar ve i . yeni pozisyonlar sırasıyla V_i , P_{best_k} , X_i ile temsil edilmektedir.

İşlemler sonucunda mevcut konum ve hız vektöründeki anahtarları içeren bir anahtar takımı oluşturulur. P_{best_1} için yeni anahtar grubu $\{S1, S4, S3, S2\}$ anahtarlarını içerirken P_{best_2} için yeni anahtar grubu $\{S3, S2, S4\}$ anahtarlarını içermektedir. Kaynaktan hedefe olan çözüm belirlenen anahtar kümelerinden aranır. Her P_{best} değeri, karşılık gelen anahtar seti tarafından oluşturulan yeni olası çözümle güncellenir. Bu yaklaşımın avantajı, ağdaki tüm anahtarlardan çözüm aramak yerine sadece bu anahtar kümesinden çözüm aranmasıdır. Denklem 2.6'ya göre hesaplanan yeni parçacıklar, yani yeni P_{best} değerleri, $P_{best_1} = [S1 S2 S4]$ ve $P_{best_2} = [S3 S2 S4]$ olsun. Her bir P_{best} değeri güncellenmiş olur.

Önerilen algoritmada durdurma metriği iterasyon sayısıdır ve iterasyon sayısı kadar uygun çözüm arayışı devam etmektedir. Hesaplama sürecinde iterasyon sayısının belirlenmesi hayati önem taşımaktadır ve bu iterasyonlardan sonra elde edilen G_{best} değeri en uygun çözümdür. Algoritma 1, önerilen yöntemin yapısını vermektedir.

Tez çalışmasında, ağ topolojisi olarak fat tree topolojisi kullanılmıştır. Bu topolojinin detayları Bölüm 2.3.1.'de verilmiştir. Önerilen yöntemde başlangıç rastgele parçacık sayısı 20, durdurma kriteri ise 50 olarak seçilen iterasyon sayısıdır. Her 150 saniyede bir belirlenen rotalar akış tablolarından silinir. Rota yeniden hesaplanır, böylece dinamik olarak değişen ağ isteklerine yanıt vermek için dinamik bir rota seçimi sunulmuş olur.

Algoritma 1 Uygulanan D-PSO Yöntemi

Giriş

Ağ topolojisi grafa dönüştürülür,
Graf yardımıyla kaynak ve hedef arasındaki tüm rotalar hesaplanır,
Parçacık sayısı (Pbest) kadar olası yol seçilir.

Çıkış

Gbest değeri

Başla

for i. iterasyon in iterasyon sayısı

Algoritma 2'ye göre tüm Pbest değerlerinin maliyet fonksiyonları hesaplanır

Gbest = Min (Min_maliyet (Pbests), önceki iterasyondaki Gbest)

for j.Pbest in Tüm Pbest değerleri (Tüm Pbest'ler Gbest'e göre güncellenir):

j.Pbest hız değeri (V_j) hesaplanır: $Gbest - j.Pbest$

Yeni konum değeri hesaplanır: $V_j + j.Pbest$

Elde edilen Gbest en uygun çözümdür

Son

2.3.3. Maliyet Fonksiyonun Hesaplanması

Uygunluk değerinin hesaplanması, D-PSO'nun en önemli süreç adımlarından birisidir. Tez çalışmasında uygunluk değerini bulmak için maliyet fonksiyonu kullanılmaktadır. Maliyet fonksiyonu, aday çözümlerden birini girdi olarak alır ve çözümün problem için ne kadar maliyetli olduğunu hesaplamaktadır. Sonuç olarak en az maliyetli aday, çözüm olarak seçilmektedir. D-PSO'da uygunluk değeri iterasyon sayısı kadar hesaplanmaktadır, bu nedenle yeterince hızlı olması önemlidir. Uygunluk değeri ne kadar yavaş hesaplanırsa, yöntem o kadar yavaş sonuca ulaşır. Bu nedenle, maliyet fonksiyonu tasarlanırken seçilen metrikler ve katkı oranları doğru bir şekilde belirlenmelidir. Önerilen yaklaşım, anahtarlar ve bağlantılar üzerinde minimum zamanda dengeli bir yük dağılımı sağlamaktadır. Önerilen yöntemde kullanılan maliyet fonksiyonu bu amaçla belirlenmektedir.

Ağ topolojisinde en uygun rota seçilirken sadece bağlantı istatistiklerine bakmak yeterli olmayabilir; bu nedenle, tez çalışması kapsamında hem bağlantılardan hem de anahtardan alınan istatistikler kullanılarak hibrit yapılı bir maliyet fonksiyonu tasarlanmıştır. Belirli bir anahtara hızlı bir şekilde ulaşmak, her zaman onu hızlı bir şekilde bırakacağımız anlamına gelmez. Anahtarı bırakma zamanı, anahtarın ne kadar meşgul olduğu ve ne kadar çok alternatif yol seçeneği olduğu ile ilgilidir. Maliyet fonksiyonunda kullanılan metrik sayısı da doğru sonuca hızlı bir şekilde ulaşmada etkilidir. Karşılaştırıldığında, metrik sayısının yüksek olması hızı olumsuz etkileyebilir, çok düşük olması ise doğru sonucu elde

etmede bir dezavantajdır. Bu varsayımlara dayanarak, hibrit tasarımı maliyet fonksiyonunu hesaplamak için hem anahtarlardan hem de bağlantılardan dört adet metrik seçilmiştir.

Hibrit tasarımı maliyet fonksiyonunda kullanılan metrikler RTT (Gidiş Dönüş Süresi), BW (Bant Genişliği), PLC (Kayıp Paket Sayısı) ve PN (Port Sayısı)'dır. BW ve PLC, bağlantılardan elde edilen istatistiksel bilgilerdir. BW, bağlantıdaki saniye başına veri miktarını verir. Ağ bağlantısında veriler kaybolduğunda paket kaybı meydana gelir. Bu durum çözüme ulaşmayı yavaşlatır. Ağdaki bir sorunu çözmek için yapılacak ilk şey paket kaybını durdurmak olmalıdır. Paket kaybının birden çok nedeni olabilir. Ancak yetersiz ağ bant genişliği, paket kaybının en önemli nedenlerinden biridir. Bu durum, aynı bağlantı üzerinden çok fazla akış iletişim kurduğunda gerçekleşmektedir. Bu durumda, yeni gelen akışı farklı bir bağlantı seçmeye teşvik etmek için maliyet fonksiyonunun değeri artırılmalıdır.

Ayrıca hibrit tasarımı maliyet fonksiyonunda RTT ve PN anahtarlardan alınan istatistiksel bilgilerdir. RTT, bir sinyali iletmek için geçen süre ile sinyalin alındığını onaylamak için geçen sürenin toplamına eşittir. Bir anahtar için RTT ne kadar fazlaysa, anahtar o kadar yoğun olur. Anahtarın yüksek yoğunluğu, hibrit tasarımı maliyet fonksiyonunu artırarak seçilme olasılığını azaltır. PN, anahtarın sahip olduğu alternatif yolların sayısını temsil etmektedir. Anahtar port sayısı ne kadar yüksek olursa, gelen paketin alternatif yolu o kadar fazla olur ve her bağlantı için yük o kadar düşük olur. Bu bilgiler doğrultusunda hibrit tasarımı maliyet fonksiyonunun hesaplanması Şekil 2.2.'de verilen ağ topolojisi üzerinden anlatılacaktır.

Formül uygulanırken ağ topolojisindeki bağlantılar $\{L1, L2, \dots\}$ ve anahtarlar $\{S1, S2, \dots\}$ dikkate alınır. $[S3 \ S2 \ S4]$ yolu üzerinden ana bilgisayar 1'den (H1) ana bilgisayar 2'ye (H2) bir paket gönderildiğini varsayarsak, tüm yol için hesaplanan hibrit tasarımı maliyet fonksiyonu denklem 2.7'deki gibi ifade edilmektedir. Ayrıca Anahtar 3 ve Anahtar 2 arasındaki hibrit planlı maliyet fonksiyonu hesaplaması denklem 2.8'de verilmiştir. Denklem 2.8'deki $S2$, 2 numaralı anahtarı temsil ederken, 3 numaralı ve 2 numaralı anahtar arasındaki bağlantı $L6$ ile temsil edilmektedir.

$$CF = CF(S3, S2) + CF(S2, S4) \quad (2.7)$$

$$CF(s3, s2) = \frac{w1 * RTT(S2)}{\max(RTT(S3, S2, S4))} + \frac{w2 * BW(L6)}{\max(BW(L6, L8))} + \frac{w3 * PLC(L6)}{\max(PLC(L6, L8))} - \frac{w4 * PN(S2)}{\max(PN(S3, S2, S4))} \quad (2.8)$$

Maliyet fonksiyonuna katkıda bulunan 4 metriğin istatistiksel verileri ayrı ayrı elde edilmektedir. Denklem 2.5'de RTT ile temsil edilen metrik, topolojideki en az yoğun anahtarı algılamak için ağdaki tüm anahtarlara ping mesajları göndererek elde edilir. En düşük RTT değerine sahip anahtarın maliyeti daha düşüktür. İkinci metrik, denklem 2.5'de BW ile temsil edilen bant genişliği metriğidir. BW, hibrit tasarlanmış maliyet fonksiyonuna katkıda bulunurken, bağlantının kullandığı bant genişliği miktarını kullanır, yani düşük bant genişliğine sahip bağlantının maliyeti düşük olacaktır. PLC metriği ile temsil edilen kayıp paket sayısı bir diğer metriktir. TCP paketlerinde kaybolan her paket yeniden gönderildiğinden ve bu durum paket iletme süresini olumsuz etkilediğinden yüksek PLC bir dezavantajdır. Diğer bir deyişle, düşük bir PLC maliyeti düşürecektir. Port sayısı son metrik olarak kullanılır ve denklem 2.5'de PN ile temsil edilmektedir. Anahtar port sayısı ne kadar yüksek olursa, gelen paketin alternatif rotası o kadar fazla olur ve her bağlantı için o kadar az yük oluşur. Başka bir deyişle, yüksek PN, seçilme olasılığını artırarak maliyeti düşürecektir. Ek olarak, her bir metrik farklı bir değer aralığına sahip olduğundan metrik sonuçlarını yakınlaştırmak için normalleştirme yapılmaktadır.

Önerilen yöntemde her bir metrik için belirlenen ağırlık değerleri ve bu ağırlık değerlerinin toplamı denklem 2.9'da gösterilmektedir. Anahtarlardan ve bağlantılardan alınan istatistik değerleri, yöntem sonucu için aynı öneme sahiptir. Yani anahtardan elde edilen istatistiksel katsayılar $w1 + w4$ değeri ile bağlantılardan elde edilen istatistiksel katsayılar $w2 + w3$ değeri birbirine eşittir. Ancak seçilen metriklerin, hedefe ulaşmada aynı katkıyı sağlamadıkları açıktır. Anahtardan elde edilen RTT değeri ve bağlantılardan elde edilen BW değeri diğer parametrelere göre daha fazla öneme sahiptir; bu nedenle bu parametrelerin ağırlıkları biraz daha yüksek verilmiştir. Hibrit tasarımlı maliyet fonksiyonunda RTT ve BW ağırlık değerleri 0.3 olup sırasıyla $w1$ ve $w2$ ile temsil edilmektedir. Ek olarak, PLC ve PN ağırlık değerleri sırasıyla $w3$ ve $w4$ ile temsil edilir ve 0,2 değerine sahiptir.

$$w1 + w2 + w3 + w4 = 1, w1 = w2 = 0.3, w3 = w4 = 0.2 \quad (2.9)$$

Önerilen yöntem dinamik olarak uygulanmaktadır, bu nedenle hibrit tasarımlı maliyet fonksiyonunda kullanılan metrik değerlerinin güncel tutulması hayati önem taşımaktadır. Bu nedenle, istatistikler her 5 saniyede bir güncellenmektedir. Algoritma 2, hibrit tasarımlı maliyet fonksiyonunun hesaplanmasını vermektedir.

Algoritma 2 Hibrit Tasarımlı Maliyet Fonksiyonunun Hesaplanması

Giriş

Tüm anahtarların RTT ve PN değerleri,
Tüm bağlantıların BW ve PLC değerleri,
w1, w2, w3 ve w4 metriklerinin ağırlık değerleri,
Parçacık sayısı (Pbest) kadar olası yol seçilir.

Çıkış

Anahtar ve takip eden bağlantı arasındaki maliyet

Başla

for i. olası çözüm in Pbest:
max (RTT), max (BW), max (PLC) ve max (PN) hesaplanır
j. anahtar ve takip eden bağlantı arasındaki maliyet;

$$CF(j, j + 1) = \frac{w1*RTT(j+1)}{max(RTT)} + \frac{w2*BW(link(j, j+1))}{max(BW)} + \frac{w3*PLC(link(j, j+1))}{max(PLC)} - \frac{w4*PN(j+1)}{max(PN)}$$

Son

2.4. YTA Fil Akış Tespiti

Son zamanlarda, trafik mühendisliği, hizmet kalitesi ve ağ yönetimi gibi uygulamaların yanı sıra akışların sınıflandırılması konusuna da ilgi yüksektir. Akışlarda iki temel sınıf vardır: fil akışlar ve fare akışlar. Özellikle fil akış tespiti için kullanılan sınıflandırma yöntemleri, YTA'da önemli bir rol oynamaktadır. Fil akış tespiti ile veri merkezlerinin sağladığı yüksek bant genişliği etkin bir şekilde kullanılmakta ve ağ üzerinde gecikme olmaksızın trafik yönlendirmesi yapılabilmektedir. Fil akış sınıflandırmasında en önemli hedefler, fil akışlarını mümkün olan en kısa sürede ve yüksek doğrulukla tespit etmektir. Tez çalışmasında fil akış tespitinin doğruluk oranını arttırmayı amaçlayan, yedi farklı sınıflandırıcı ile test edilmiş, yeni bir algılayıcı bu iki sınıfı ayırmaya çalışmıştır. Önerilen algılayıcıda, akışları maksimum doğrulukla hızlı bir şekilde sınıflandırmak için farklı adımlar uygulanmıştır. Literatürde içerisinde fil ve fare akışları birlikte barındıran gerçek zamanlı farklı veri kümeleri bulunmaktadır. UNI1 ve UNI2, sınıflandırıcıları test etmek için kullanılan bu veri kümelerinden bazılarıdır.

İlk olarak, bu veri kümeleri paketlerden oluştuğu için bu paketlerin akışlara dönüştürülmesi gerekir. Bu dönüşümü gerçekleştirmek için FlowRecorder aracı kullanılmaktadır. Bu araç yardımıyla her veri kümesi, akışlardan oluşan yeni bir veri kümesine dönüştürülür. Bu yeni veri kümesi etiket değerleri içermemektedir. Bu nedenle, veri kümelerindeki her bir akışı başlangıçta fare veya fil olarak etiketlemeye ihtiyaç vardır.

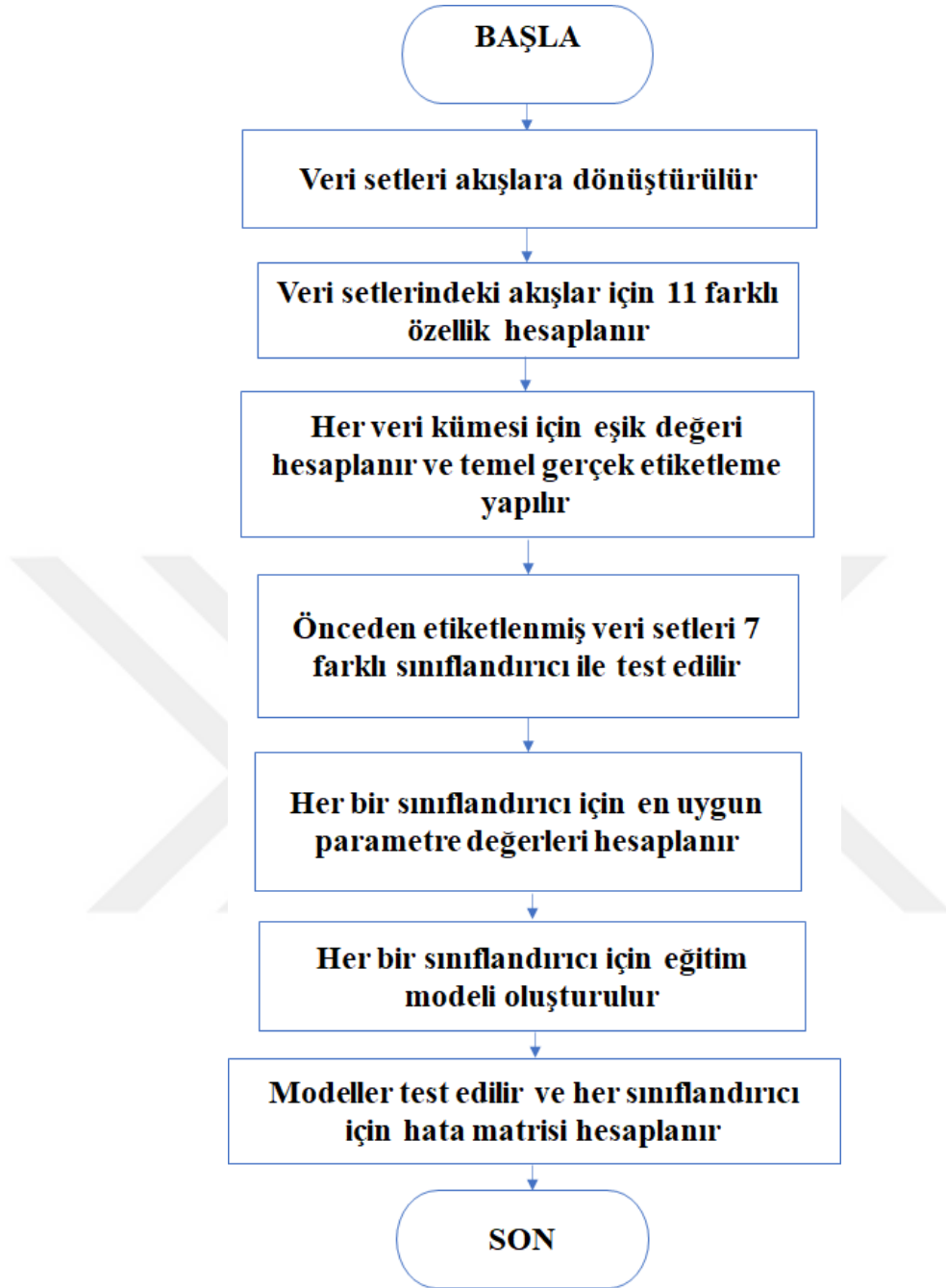
Bu duruma temel gerçek etiketleme denir. Araştırmacılar genellikle bu işlem için statik bir eşik değeri kullanmayı tercih etmektedir. Statik eşik belirlenirken veri setindeki tüm akışlar manuel olarak incelenmekte ve sabit eşik belirlenmektedir. Böyle bir eşik bulma yönteminin pratik veya etkili olması beklenemez. Ayrıca, veriler sürekli güncellendiğinden, statik eşik ile algılama gerçek zamanlı bir ağda başarılı olmayabilir. Bu nedenle literatürden farklı olarak, tez kapsamında önerilen algılayıcıda, değişen ağ durumlarına hızlı bir şekilde uyum sağlayabilen Otsu eşikleme yaklaşımına dayalı dinamik bir eşik kullanılmaktadır. Otsu eşikleme yöntemi, UNI1 ve UNI2 veri kümelerindeki paket boyutu özelliğini kullanılmaktadır. Her veri kümesi için paket boyutu özelliği kullanılarak histogram elde edilir ve bu histogramlar eşikleme yöntemine verilir. Böylece dinamik eşik elde edilir. Daha sonra bu eşik değerleri kullanılarak temel gerçek etiketleme yapılır. Sınıflandırmaya hazır veri setleri, yedi farklı sınıflandırıcı ile test edilmiştir. Bunlar; Lojistik Regresyon, Destek Vektör Makineleri (DVM), Karar Ağaçları, Rastgele Ormanlar, Naive Bayes, K-En Yakın Komşu (KNN) ve Derin Öğrenme (DL) yöntemleridir.

Sınıflandırma yöntemlerinin başarısındaki en kritik faktör veri setlerinden çıkarılan özniteliklerdir. Özellik vektörü veri setini ne kadar iyi temsil ederse, sınıflandırma sonuçları o kadar başarılı olacaktır. Literatürde fil ve fare akışlarının sınıflandırılmasına dayalı yaklaşımlar sadece paket boyutu özelliği ile sınırlıdır. Bu sınırlama, sınıflandırma yöntemlerinin etkinliğini azaltmaktadır. Önerilen algılayıcıda her bir veri seti için zengin bir öznitelik vektörü oluşturulmuştur. Bu özellik vektörü 11 farklı özellik içermektedir. Hesaplanan akış özellikleri Tablo 2.1.'de verilmektedir.

Tablo 2.1. FlowRecorder aracı kullanılarak elde edilen özellikler

Paket Boyu Özellikleri	Varış Zamanı Özellikleri	Diğer Özellikler
- akışın minimum paket boyutu	- akışın minimum paket varış zamanı	- akıştan aktarılan bayt sayısı
- akışın ortalama paket boyutu	- akışın ortalama paket varış zamanı	- akış için iletilen paketlerin sayısı
- akış paket boyutlarının standart sapması	- akış varış zamanlarının standart sapması	- saniye cinsinden akış süresi
- akışın maksimum paket boyutu	- akışın maksimum varış zamanı	

Her bir özelliğin, akışları en iyi şekilde tanımlayabilecek farklı önemleri vardır. Tez çalışması için çıkarılan öznitelik vektörlerini sınıflandırmak için yedi sınıflandırma yöntemi denenmiştir. Farklı sınıflandırıcılar kullanılarak fil akış ve fare akış sınıflandırma başarısının artırılması amaçlanmıştır. Sınıflandırıcıların her biri farklı parametrelerle çalışmaktadır. Optimum parametrelerin belirlenmesi, sınıflandırıcıların başarısında önemli bir faktördür. Önerilen algılayıcıda farklı sınıflandırıcıların maksimum doğruluğunu sağlayan optimum parametrelerini belirlemek için Izgara Arama (Grid Search) Yöntemi kullanılmıştır. Her veri seti için ayrı sınıflandırma modelleri oluşturulur. Veri setlerinden elde edilen test verileri bu modellere beslenir ve doğruluk, kesinlik, geri çağırma ve F-skor metrikleri hesaplanır. Ayrıca, önerilen algılayıcıda sınıflandırıcıların zaman karmaşıklığı da ölçülmüştür. Çünkü gerçek zamanlı bir YTA'da, ağlarda gecikme, sıkışma gibi sorunlara neden olmadan akışların çok hızlı bir şekilde sınıflandırılması gerekmektedir. Böylece trafik yönetimi verimli bir şekilde gerçekleştirilebilir. Önerilen algılayıcının akış şeması Şekil 2.4.'te verilmiştir.



Şekil 2.4. Önerilen algılayıcının akış şeması

Sonuç olarak, fil akışlar algılanamadığında tıkanıklığa neden olabilir ve gecikmeye duyarlı fare akışlarını etkileyebilir. Aynı zamanda herhangi bir fil akışının yanlış algılanması, bir fare akışının yanlış algılanmasından daha ciddi sonuçlara yol açabilmektedir. Bu yüzden akışlara ön etiketleme yaparken değişen ağ verilerine uygun olarak eşik belirlenmektedir. Farklı sınıflandırıcılar kullanarak doğruluk oranının artırılması

ve tespit süresinin kısaltılması amaçlanmaktadır. Ve tüm bunlar gerçekleştirilirken yüksek bant genişliği tüketilmemektedir. Önerilen algılayıcı modelini özetlersek;

- FlowRecorder kullanılarak veri setlerindeki paketler akışlara dönüştürülmüştür.
- Ön etiketlemede dinamik olması için Otsu eşikleme yöntemi kullanılmıştır.
- Karmaşık veri setlerinde de başarıyı artırmak amacıyla özellik vektörü için 11 farklı özellik belirlenmiştir.
- Önerilen algılayıcıda 7 farklı sınıflandırma tekniği test edilerek fil akış tespiti doğruluk oranının artırılması amaçlanmıştır.
- İzgara Arama Yöntemi kullanılarak 7 farklı sınıflandırma yöntemi için ayrı ayrı optimum parametreler belirlenmiştir.
- Yöntem başarısını kanıtlamak için hem hata matrisinden elde edilen karşılaştırma metrikleri hem de zaman açısından değerlendirmek için akış tespit süresi kullanılmıştır.

Çalışmanın literatürdeki diğer yöntemlerle karşılaştırılabilmesi için literatürde sıklıkla kullanılan veri setleri seçilmiştir. Önerilen algılayıcıda denetimli öğrenme kullanıldığından, bu veri kümeleri için ön etiketleme yapılmıştır. Sınıflandırma için makine öğrenmesinde sıklıkla kullanılan 7 farklı yöntem denenmiştir. Amacımız, fil akış tespitini mümkün olan en kısa sürede gerçekleştirirken doğruluğunu artırmaktır. Önerilen algılayıcının başarısını kanıtlamak için literatürde sıklıkla tercih edilen karşılaştırma metrikleri kullanılmıştır. Önerilen algılayıcının detayları alt başlıklarla irdelenmektedir.

2.4.1. Kullanılan Veri Setleri

Veri seti olarak literatürde fil akış tespitine odaklanan yayınlarda sıklıkla tercih edilen UNI1 ve UNI2 veri setleri kullanılmıştır [92]. UNI1 ve UNI2, üniversite veri merkezlerinde yakalanan iki gerçek ağ izleme sonucudur. Bu iki set 3 saatten daha kısadır, ancak fare akış ve fil akış dağılımları, gün boyunca farklı dönemlerde toplanan halka açık olmayan paket izlemelerle benzerdir. Tablo 2.2., UNI1 ve UNI2 olmak üzere iki farklı gerçek izlemeye ait özet bilgileri vermektedir.

Tablo 2.2. Simülasyonlarda kullanılan izlemelerin özellikleri

Veri Setleri	UNI1	UNI2
Paketler	19.85 M	100 M
IPv4 Akışlar	1.02 M	1.04 M
	(TCP and UDP)	(çoğunlukla UDP)
Toplam trafiğin IPv4 yüzdesi	98.98%	99.9%
	(çoğunlukla TCP)	(çoğunlukla UDP)
Süre	65 dakika	158 dakika

FlowRecorder aracı kullanılarak UNI1 ve UNI2 paketleri akış olarak düzenlenmektedir. 11 farklı akış özelliği belirlenmiştir. Hesaplanan akış özellikleri Tablo 2.1.'de verilmiştir. Akış istatistiklerinin yanı sıra FlowRecorder, tüm paket boyutlarının dökümü ve akışları oluşturan yakalama süreleri gibi paket düzeyindeki ayrıntıları da kaydetmektedir [93,94].

2.4.2. Temel Gerçek Etiketleme

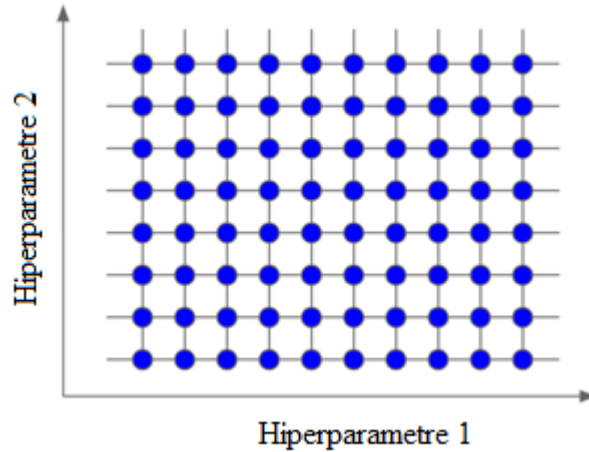
Danışmalı öğrenme yöntemi tercih edildiğinden veri setleri fare akışı ve fil akışı olarak etiketlenmektedir. Fil akışlar için etiket değeri 1 iken, fare akışlar için etiket değeri 0'dır. Etiketleme için sabit eşik değeri kullanan oldukça fazla yayın bulunmaktadır. Fakat dinamik olarak değişen ağ durumu ele alındığında sabit eşik değeri kullanmak, her durumda verimli olmayabilir. Tez çalışması kapsamında eşik değeri Otsu metodu kullanılarak belirlenmektedir. Otsu metodu, iki parçayı sınıflar arası varyansları maksimum olacak ve dolayısıyla birleşik yayılımları minimum olacak şekilde bölmek için en iyi eşik değerlerini hesaplamaya çalışmaktadır. Otsu metoduna göre tüm eşik değerleri için bu iki sınıfın sınıf içi varyans değerleri hesaplanmaktadır. Ve bu değer en küçük olmasını sağlayan eşik değeri, optimum eşik değeridir [95]. Yani yeni gelen her ağ trafiği için eşik değeri farklı olacaktır. Ağ trafik verisi toplandıkça eşik de güncellenecektir.

2.4.3. Seçilen Akış Sınıflandırma Yöntemleri

Veri setlerini sınıflandırmak için uygun sınıflandırıcıyı bulmak en önemli işlem adıımıdır. Tez çalışması kapsamında bu aşamada birden fazla sınıflandırıcı denenmiştir. Bunlar; Lojistik Regresyon, DVM, Karar Ağaçları, Rastgele Ormanlar, Naive Bayes, KNN ve Derin Öğrenme yöntemleridir.

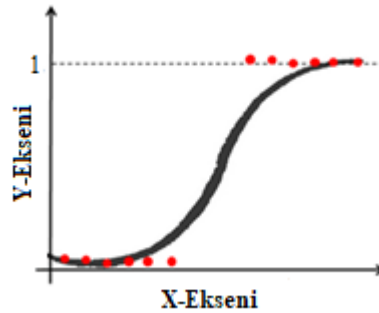
Sınıflandırıcıların başarılarının en önemli etkeni doğru parametrelerin belirlenmesidir. Izgara Arama Yöntemi, tüm parametrelerin kombinasyonlarının doğruluk değerlerini hesaplayarak ilgili veri setini en iyi sınıflandırabilecek parametrelerin seçilmesini sağlamaktadır. Böylece en yüksek doğruluk oranını veren parametreler belirlenmektedir. Çalışma kapsamında sınıflandırıcıların parametre seçimleri için bu yöntem tercih edilmiştir.

Izgara Arama Yöntemi, hiperparametre ayarlama için basit ve kapsamlı bir algoritmadır. Temel olarak, hiperparametreler alanı Şekil 2.5'te gösterildiği gibi ızgaraya aktarılır. Daha sonra, çapraz doğrulama yöntemi ile ızgaranın her bir kombinasyonunun başarısı ölçülmektedir. Çapraz doğrulamada başarıyı maksimize eden ızgara noktası, hiperparametrelerin optimum değerini vermektedir. Izgara Arama Yöntemi, tüm kombinasyonları kapsadığından en iyi parametreleri bulmada oldukça başarılıdır.



Şekil 2.5. Hiperparametreler alanı

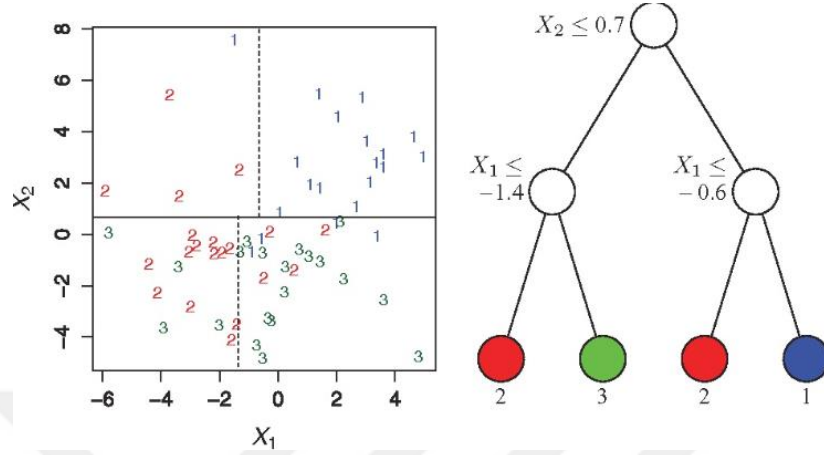
“Lojistik Regresyon”, ikili bağımlı değişkeni modellemek için kullanılan istatistiksel bir modeldir. Regresyon analizinde, lojistik bir modelin parametreleri tahmin edilmektedir. En yaygın lojistik regresyon, ikili sonuç modelleridir. Bu modeller doğru/yanlış, evet/hayır gibi iki değerler alabilmektedir. Matematiksel olarak, ikili lojistik model, iki değer 0 ve 1 olarak etiketlendiği bir gösterge değişkeni tarafından temsil edilmektedir [96]. Lojistik Regresyon Şekil 2.6. ile ifade edilmektedir. Lojistik Regresyon Yöntemi için en önemli parametreler, ceza terimi, durdurma kriteri ve optimizasyon yöntemidir. Ceza terimi, ceza normlarını tanımlamaktadır. Varsayılan olarak l2 (ridge regresyon) seçilmektedir. Durdurma kriteri varsayılan olarak 0.0004 belirlenmektedir. Optimizasyon için çözücü algoritması seçilmektedir. Çözücü, varsayılan olarak lbfgs seçilmektedir. Lbfgs, sınırlı bellekli bfgs’dir, yani bellek kullanımı sınırlıdır ve önceki gradyanlar saklanmaz. Sadece bellek kısıtlamasının izin verdiği ölçüde yeni gradyan değerlerini biriktirir. Önerilen çalışmada sınıflandırıcı parametreleri seçilirken Izgara Arama Yöntemi kullanılmıştır. Yöntem sonucu; ceza terimi olarak l2, durdurma kriteri olarak $1e-4$ ve optimizasyon yöntemi olarak lbfgs belirlenmiştir.



Şekil 2.6. Lojistik Regresyon [97]

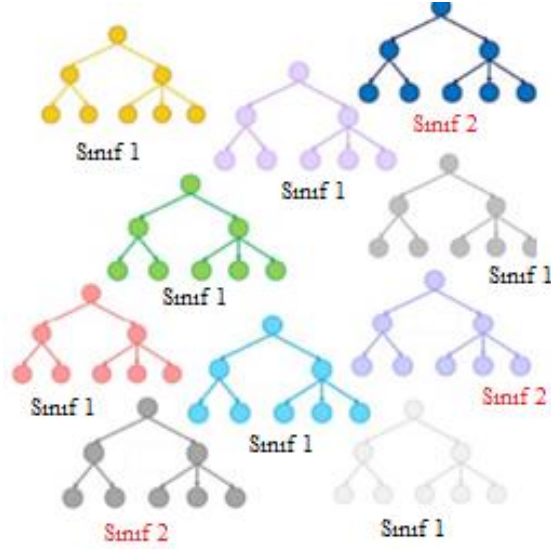
“DVM”, makine öğreniminde kullanılan ikili denetimli bir sınıflandırıcıdır. DVM algoritması, n-boyutlu bir alanı sınıflara ayırmak için çizgiler veya karar sınırlarını kullanmaktadır. DVM, en iyi karar sınırını tanımlayan hiper düzlemin oluşturulmasına yardımcı olan benzersiz noktaları/vektörleri bulur [98]. DVM yönteminde destek noktaların hesaplanması ile ilgili görsel Şekil 2.7. ile temsil edilmektedir. DVM yöntemi için en önemli parametreler, ceza terimi, çekirdek (kernel) ve derecedir. Ceza terimi, ceza normlarını tanımlamaktadır. Varsayılan olarak l2 (ridge regresyon) seçilmektedir. Çekirdek fonksiyonu sayesinde doğrusal olarak ayrılamayan sınıfların ayrılabilir hale getirilerek daha başarılı

seçilirken Izgara Arama Yöntemi kullanılmıştır. Yöntem sonucu; kriter fonksiyonu olarak gini, ayırıcı olarak best ve maksimum derinlik olarak sonsuz belirlenmiştir.



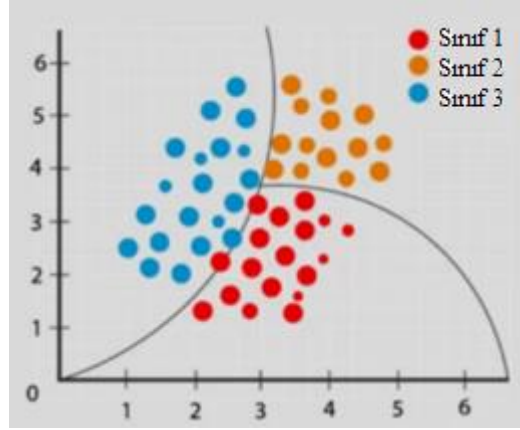
Şekil 2.8. Karar Ağaçları [100]

“Rastgele Ormanlar”, her ağacın bağımsız olarak örneklenen rastgele bir vektörün değerlerine bağlı olduğu sınıflandırıcıdır. Ormandaki tüm ağaçlar için aynı dağılıma sahip olduğu, ağaç tahmin edicilerinin bir kombinasyonudur. Bir ağaç sınıflandırıcı ormanın genelleme hatası, tek tek ormandaki tüm ağaçların gücüne ve aralarındaki korelasyona bağlıdır [101]. Rastgele Ormanlar genellikle karar ağaçlarından daha iyi performans gösterir, ancak doğrulukları eğitim destekli ağaçlardan daha düşüktür. Veri özellikleri performanslarını etkileyebilir. Rastgele Ormanlar yöntemine göre oluşturulan örnek bir yapı Şekil 2.9.’da verilmektedir. Rastgele Ormanlar yöntemi için en önemli parametreler, kriter fonksiyonu, ağaç sayısı ve maksimum derinliktir. Kriter fonksiyonu ile bölünmenin kalitesini ölçme işlevi belirlenmektedir. Kriter olarak gini indeks veya bilgi kazancı (entropi) kullanılmaktadır. Gini indeks her nitelik ikiye bölünerek olası bütün bölünmelerin test edilmesidir. Sonuç olarak gini indeks değeri en küçük olan nitelik seçilmektedir. Maksimum derinlik ile ağacın maksimum derinliği belirlenmektedir. Önerilen çalışmada sınıflandırıcı parametreleri seçilirken Izgara Arama Yöntemi kullanılmıştır. Yöntem sonucu; kriter fonksiyonu olarak gini, ağaç sayısı olarak 100 ve maksimum derinlik olarak sonsuz belirlenmiştir.



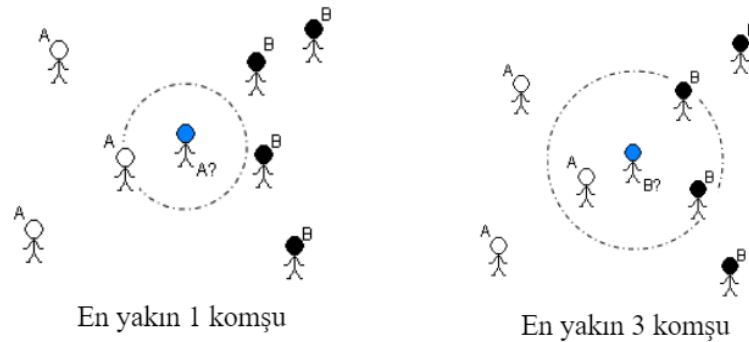
Şekil 2.9. Rastgele Ormanlar [102]

“Naive Bayes”, Bayes teoremine dayanmaktadır. Tahmin edicilerin kendi aralarında bağımsız olduğunu, yani bir sınıftaki bir özelliğin varlığının diğer özelliklerle ilişkisiz olduğunu varsaymaktadır [103]. Bir Naive Bayes sınıflandırıcı, her öznenin biribirinden koşulsal bağımsız olduğu ve öğrenilmek istenen kavramın tüm bu özniteliklere koşulsal bağlı olduğu bir Bayes ağı olarak da düşünülebilir. Örnek bir sınıflandırma Şekil 2.10.’da verilmektedir. Naive Bayes yöntemi için en önemli parametreler, öncelik ve varyans yumuşatmadır. Öncelik, sınıfların önceki olasılıklarını belirtmektedir. Varyans yumuşatma ise hesaplama kararlılığı için varyanslara eklenen tüm özelliklerin en büyük varyans kısmını ifade etmekte ve varsayılan olarak $1e-9$ ’dur. Önerilen çalışmada sınıflandırıcı parametreleri seçilirken Izgara Arama Yöntemi kullanılmıştır. Yöntem sonucu; öncelik olarak 2 ve varyans yumuşatma değeri olarak $1e-9$ belirlenmiştir.



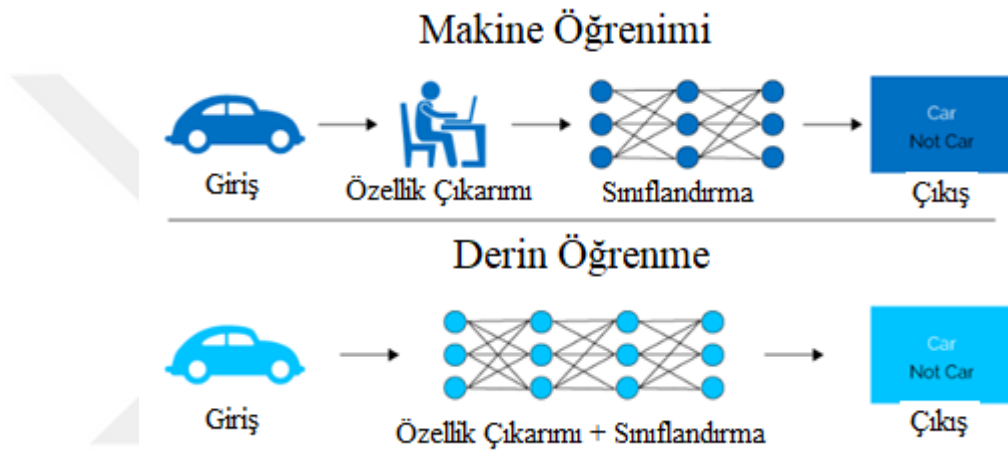
Şekil 2.10. Naive Bayes [104]

“KNN”, hem sınıflandırma hem de regresyon problemlerini çözmek için kullanılan, basit ve uygulaması kolay bir denetimli öğrenme algoritmasıdır. Diğer denetimli öğrenme algoritmalarının aksine, eğitim aşamasına sahip değildir. Örnekler en yakın komşularının sınıfına göre sınıflandırılmaktadır. Komşu sayısının fazla olması genellikle yararlıdır. Tüm işlemler çalışma zamanında yapıldığı için büyük veri setlerinde KNN’in çalışma zamanı performansı düşüktür [105]. KNN yöntemi için en önemli parametre k değerinin, bilinmeyen noktanın en yakın komşularının miktarının, belirlenmesidir. k değerinin 1 ve 3 olarak belirlendiği örnek sınıflandırmalar Şekil 2.11. ile gösterilmektedir. Önerilen çalışmada sınıflandırıcı parametresi seçilirken Izgara Arama Yöntemi kullanılmıştır. Yöntem sonucu; k olarak 3 belirlenmiştir.



Şekil 2.11. K-En Yakın Komşu Algoritması [106]

“DL”, son yıllarda hemen hemen her alanda tercih edilmektedir [107-109]. Derin Öğrenme, bir veya daha fazla gizli katman içeren yapay sinir ağları ve benzeri makine öğrenme algoritmalarını kapsayan bir çalışma alanıdır. Makine öğreniminin yanında veya bir alt küme olarak yapay zeka şemsiyesi altında yer alır. Aradaki fark, makine öğreniminin belirli görevler için geliştirilmiş algoritmaları kullanmasıdır. Derin Öğrenme, daha çok, her katmanın girdi olarak önceki katmandan gelen çıktıyı kullandığı, bir matrisin birden çok katmanına dayanan bir veri temsidir [110]. Makine Öğrenimi ile Derin Öğrenme karşılaştırılması Şekil 2.12. ile temsil edilmektedir.



Şekil 2.12. Derin Öğrenme [110]

Bu çalışmada, akışların bir boyutlu özellik matrislerine göre sınıflandırma yapabileceği bir Evrişimli Sinir Ağı (Convolutional Neural Network-CNN) yapısı tasarlanmıştır. CNN yapısı tasarlanırken doğruluğun yanında zamanı da göz önüne alan derinliği çok fazla olmayan bir ağ önerilmektedir. Önerilen ağ, giriş katmanı, konvolüsyon katmanı, aktivasyon katmanı, normalizasyon katmanı ve çıkış katmanından oluşmaktadır. Ağ girişi, her bir akış için 11 farklı özellik çıkarıldığından 11x1 büyüklüğünde bir boyutlu bir dizidir. İki tane konvolüsyon katmanı bulunmaktadır, ilki 3x1 boyutunda 48 filtreden oluşurken ikincisi 3x1 boyutunda 96 filtreden oluşmaktadır. Aktivasyon fonksiyonu olarak ReLU (Rectified Linear Unit) seçilmiştir. Ayrıca, veri dağılımlarını normalize etmek, ağı ezberlemesini önlemek ve daha hızlı öğrenme sağlayabilmek amacıyla batch normalizasyon katmanı kullanılmıştır. Çıkış katmanında, tek bir çıkış bulunmaktadır. Derin Öğrenme yaklaşımlarında en önemli aşamalardan birisi de parametrelerin seçimidir. Parametreler

seçilirken maksimum doğruluğu en hızlı şekilde veren parametreler seçilmiştir. Bu seçim yapılırken Izgara Arama Yöntemi kullanılmıştır. Optimizasyon yöntemi olarak öğrenme katsayısını adaptif olarak değiştiren ve hızlı yakınsama sağlayan Adam kullanılmıştır. Kayıp fonksiyonu olarak öğrenme işleminin erken bitmesine engel olabilecek binary cross entropy kullanılmıştır. Öğrenme katsayısı ve momentum değeri geri yayılım aşamasında kullanılmaktadır. Parametreler, kayıp fonksiyonunun minimumuna yakınsayacak şekilde güncellenmektedir. Öğrenme katsayısı olarak 0.0001, momentum değeri olarak 0.99 ve batch size olarak 8 belirlenmiştir.

Önerilen CNN mimarisi Şekil 2.13. ile temsil edilmektedir.



Şekil 2.13. Önerilen CNN mimarisi

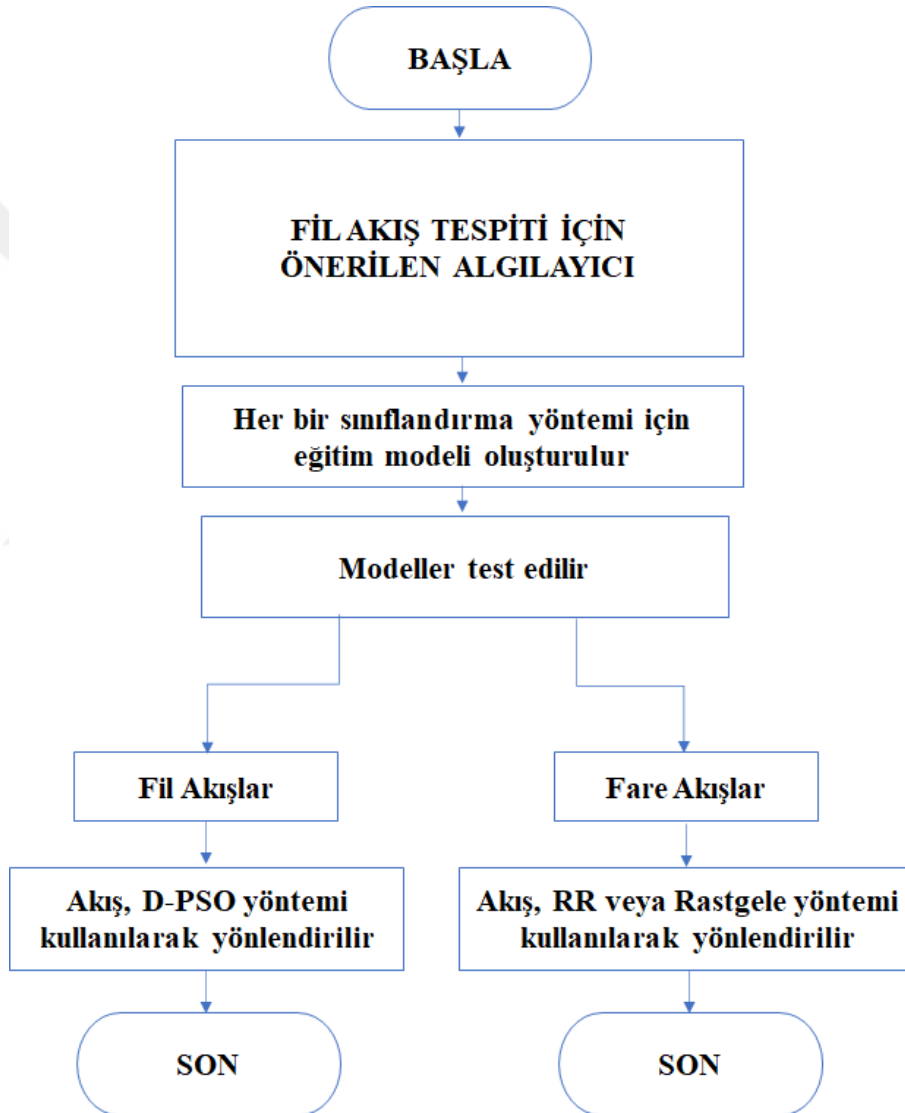
2.5. YTA Fil Akışların Planlanması

Önerilen çalışmada, son kullanıcı uygulamalarından biri olan yük dengelemeye katkı sağlanmıştır. Yük dengeleme yapılırken her akışı aynı kabul ederek aynı yönlendirme stratejisinin uygulanması her zaman başarılı sonuçlar vermeyebilir. Gerçek bir ağda, yol seçiminin uygulamanın performansına çok etki etmeyeceği küçük akışlar olduğu gibi, yanlış yönlendirildiğinde ağı kullanılamaz hale getirebilecek çok büyük akışlar da olacaktır. Bu nedenle yük dengelenirken akış özelliklerine göre yönlendirme yapılması oldukça önemlidir. Literatürde akışlar fil ve fare akışlar olmak üzere iki sınıfa ayrılmaktadır. Çalışmada, her iki akış sınıfı için farklı yönlendirme stratejisi yapılarak ağdan maksimum verim alınması amaçlanmaktadır. Bu noktada maksimum verimin sağlanabilmesi için çok fazla bant genişliği gerektiren fil akışların yönlendirilmesi fare akışlara oranla daha hayatidir.

Tez çalışması kapsamında, tespiti gerçekleştirilen fil akışlar ve fare akışlar için farklı yük dengeleme stratejilerinin uygulandığı yeni bir yaklaşım önerilmektedir. İlk olarak UNI1 ve UNI2 veri setlerindeki örnekler kullanılarak akışların sınıflandırılması sağlanmaktadır. Bu veri setleri paketlerden oluştuğundan akış tespitinde kullanılabilmesi için öncelikle paketler FlowRecorder aracı ile akışlara dönüştürülmektedir. Dönüşüm sonrası fil akışların ya da fare akışların ön etiketlemesi için Otsu eşikleme yöntemi kullanılmaktadır. Ön etiketleme yapılırken sadece paket boyutu özelliği kullanılmaktadır. Yaklaşımdaki diğer bir adım, ön etiketleme yapılan veri setlerinde bulunan akışlar için özellik çıkarımıdır. Önerilen çalışmada karmaşık veri setleri için de uygulanabilir olması nedeniyle her bir akış için 11 farklı özellik çıkarılmaktadır. Sınıflandırma yöntemi olarak Bölüm 2.4.3.'te detaylı olarak anlatılan Karar Ağaçları, DVM ve Derin Öğrenme kullanılmıştır. Maksimum doğruluğu sağlayan optimum parametrelerin belirlenmesi için her bir sınıflandırıcıda Izgara Arama yöntemi kullanılmaktadır.

Akış sınıflandırma işleminden sonra her bir akış belirlenen yönlendirme stratejisine göre yönlendirilmektedir. Fare akışlar, ağın verimini çok fazla etkilemeyen akışlar olduğundan basit bir yönlendirme algoritması ile yönlendirilmesi yeterli olmaktadır. Bu nedenle önerilen çalışmada, fare akışlar geleneksel RR ve Rand yöntemleri kullanılarak yönlendirilmektedir. Fil akışlar, fare akışların aksine yüksek bant genişliği gerektiren yönlendirme hatası olması durumunda ağı kullanılmaz hale getirebilecek akışlardır. Bu akışların yönlendirilmesi için daha komplike yaklaşımlara ihtiyaç duyulmaktadır. Bu nedenle önerilen çalışmada fil akışlar, tez çalışması kapsamında uygulanan D-PSO sezgisel

yöntemi kullanılarak yönlendirilmektedir. Kullanılan D-PSO yaklaşımı hem bağlantıları hem de anahtarlardaki yükü dengeleyebilecek bir yapıya sahiptir. Bu özelliği sayesinde, paket kaybı olmaksızın ağın maksimum verimlilikle kullanılabilmesine olanak sağlamaktadır. Yönlendirmenin başarısı RTT, verimlilik ve arabellek boyutu gibi metrikler ile kanıtlanmaktadır. Önerilen yönlendirme çerçevesinin akış diyagramı Şekil 2.14.'te verilmektedir.



Şekil 2.14. Önerilen yönlendirme çerçevesi

3. BULGULAR VE İRDELEME

3.1. YTA Sunucu Tabanlı Yük Dengeleme Sonuçları

Önerilen yaklaşımın temel amacı, gelen web isteklerine minimum sürede yanıt vermektir. Bu amaçla sunucuların verimli kullanılması önemlidir. HTTP istekleri, karar işlevini ve puanlama mantığını test etmek için kullanılmıştır. İstek oluşturulurken Apache HTTP sunucu kıyaslama aracı kullanılmaktadır. 4 istemciden gelen eşzamanlı istekler, farklı sunucularla cevaplandırılmaktadır. Simülasyon sonuçları için dört farklı test ortamı oluşturulmuştur. Bunlar;

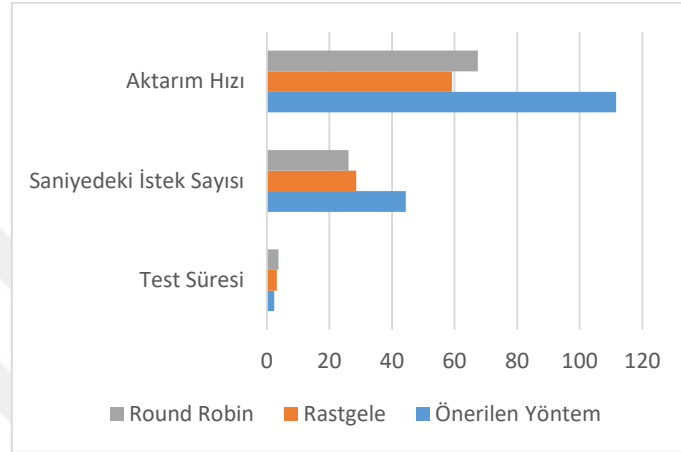
1. test ortamı: Aynı anda 4 istemci tarafından oluşturulan 100 HTTP isteği, 2 sunucu tarafından karşılanır,
2. test ortamı: Aynı anda 4 istemci tarafından oluşturulan 1000 HTTP isteği, 2 sunucu tarafından karşılanır,
3. test ortamı: 4 istemci tarafından aynı anda oluşturulan 100 HTTP isteği, 4 sunucu tarafından karşılanır ve
4. test ortamı: 4 istemci tarafından aynı anda oluşturulan 1000 HTTP isteği, 4 sunucu tarafından karşılanır.

Önerilen yöntem, geleneksel yük dengeleme yöntemleri olan RR Yöntemi ve Rastgele Yöntemi ile karşılaştırılmıştır. Karşılaştırma için üç farklı metrik kullanılmıştır. Kullanılan karşılaştırma metrikleri; aktarım hızı (Mb/s), saniyedeki istek sayısı (adet) ve test süresi (s)'dir. Aktarım hızı, verilerin kaynaktan hedefe ulaşma hızı olarak ifade edilmektedir. Genel olarak, bant genişliği ne kadar büyük olursa, veri aktarım hızı o kadar yüksek olur. Bu nedenle, aktarım hızı ne kadar yüksek olursa, yöntem o kadar başarılı olur. Saniyedeki istek sayısı, sunucuların saniyede işleyebileceği istek miktarını vermektedir. Bu metrik dolaylı olarak verime karşılık düşmektedir. Saniyedeki istek ne kadar büyük olursa, önerilen yöntem sunucular üzerindeki yükü o kadar iyi dengeler. Son olarak, test süresi, seçilen ağ senaryolarında isteklerin yanıt sürelerini ifade etmektedir. Yük dengeleme stratejisi ne kadar iyi olursa, test süresi o kadar düşük olur.

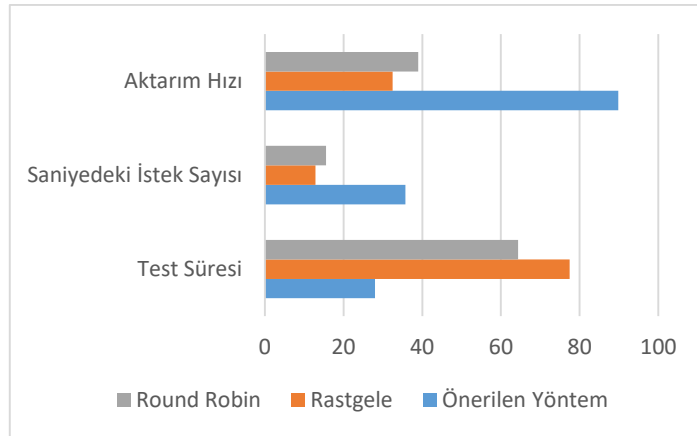
Denetleyici olarak Python programlama dilini destekleyen POX tercih edilmiştir. Bu, yeni ağ uygulamalarının geliştirilmesini daha hızlı hale getirir ve prototip haline getirilebilir [111]. Mininet emülatörü, ağ topolojisini oluşturmak için kullanılmaktadır. Fiziksel bir

bağlantıya ihtiyaç duymadan test etmek istediğimiz topolojiyi oluşturmamıza imkân tanımaktadır. Tercih edilmesinde rol oynayan en önemli özellikleri ise basit ve ucuz bir ağ test ortamı sunmasıdır [112].

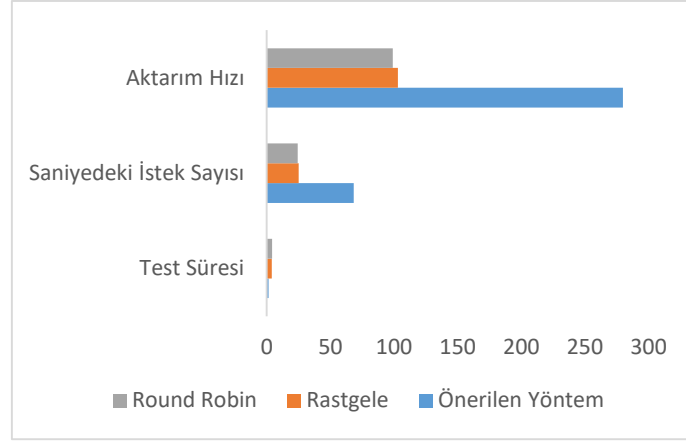
Dört farklı test ortamı için simülasyon sonuçları sırasıyla Şekil 3.1., Şekil 3.2., Şekil 3.3. ve Şekil 3.4. ile gösterilmektedir.



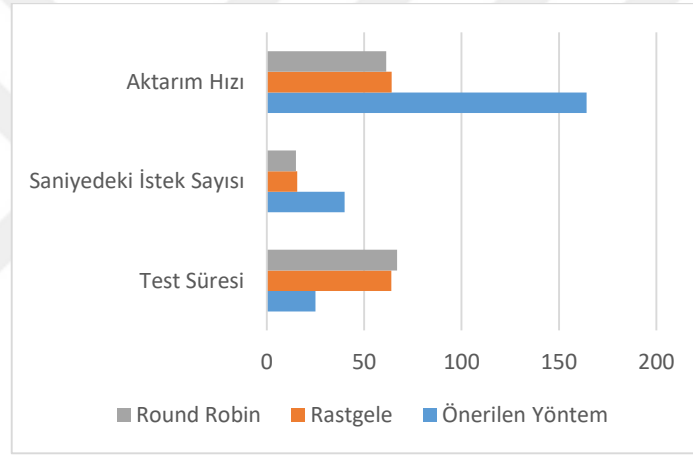
Şekil 3.1. 4 istemci 2 sunucu 100 istek



Şekil 3.2. 4 istemci 2 sunucu 1000 istek



Şekil 3.3. 4 istemci 4 sunucu 100 istek



Şekil 3.4. 4 istemci 4 sunucu 1000 istek

Şekil 3.1., Şekil 3.2., Şekil 3.3. ve Şekil 3.4., farklı senaryolar için simülasyon sonuçlarını göstermektedir. Şekillerde, önerilen yöntem mavi, Rastgele yöntemi turuncu ve RR yöntemi ise gri renk ile gösterilmektedir. Simülasyon sonuçlarına göre tasarlanan yöntem, geleneksel yük dengeleme yöntemlerine göre 4 farklı senaryoda 3 metrik için de başarılı sonuçlar vermektedir. Kullanılan metrikler için aktarım hızı ve saniyedeki istek sayısı ne kadar yüksek olursa, yöntem o kadar başarılı olmuş olur. Test süresi ise ne kadar kısa olursa yöntem o kadar başarılı olur.

Simülasyon sonuçlarından çıkarılabilecek bir diğer yorum, yöntemin istek sayısı ve ağ boyutundan etkilenmediğidir. Yöntem, tüm koşullar için başarılı sonuçlar üretmektedir. Bu, önerilen yöntemin ölçeklenebilir olduğunu kanıtlamaktadır.

Ayrıca, yöntem sonuçları uluslararası bir sempozyumda sözlü olarak sunulmuştur [113].

3.2. YTA Bağlantı Tabanlı Yük Dengeleme Sonuçları

Önerilen çalışmada, bağlantılar ve anahtarlar üzerindeki yükü dengeli dağıtmak için meta-sezgisel tabanlı bir yaklaşım uygulanmıştır. Bu yöntem, literatürdeki YTA yük dengeleme alanındaki üç farklı yaklaşımla karşılaştırılmıştır. Bunlar RR, ACO [114] ve Genetik-Karınca Kolonisi Optimizasyonu (Genetic-Ant Colony Optimization-G-ACO) [57]'dir. Geleneksel RR yöntemi, anahtarlara görevleri dairesel bir sırayla atar ve her anahtara eşit öncelik verir. Bir grafta en uygun yolu arayan ACO yönteminin mantığı, karınca kolonisi ile besin kaynağı arasında bir yol arayan karıncalara dayanmaktadır. Son olarak, G-ACO yöntemi ise YTA'da dinamik yük dengelemesi gerçekleştirmek için GA'yı ACO ile birleştirmektedir. Yöntemlerin yük dengeleme performanslarını ölçmek için literatürde farklı metrikler kullanılmaktadır. Önerilen yöntemin literatürdeki diğer yöntemlerle yük dengeleme performansını karşılaştırmak için RTT, PLR, verimlilik ve arabellek boyutu metrikleri kullanılmaktadır. Bu metrikler yük dengelemenin başarısını ve hızını ortaya çıkarabilecek en uygun metrikler oldukları için tercih edilmiştir.

YTA, yöntemleri değerlendirmek için farklı bileşenlere sahiptir. Bunlar denetleyici, emülatör, kullanılan topoloji ve ağ izleme araçlarıdır. Bu çalışmada, denetleyici olarak POX kullanılmış, fat tree ağ topolojisi Mininet emülatörü yardımıyla oluşturulmuş ve Wireshark ağ izleme aracı olarak kullanılmıştır.

Tez çalışması kapsamında YTA'da bağlantı tabanlı yük dengeleme sağlamak için D-PSO'ya dayalı bir yöntem önerilmektedir. PSO yönteminin en önemli kısmı ve yük dengeleme başarısını doğrudan etkileyen en önemli adım maliyet fonksiyonunun oluşturulmasıdır. Önerilen yöntem, tasarlanan hibrit maliyet fonksiyonu ile en doğru sonuçlara ulaşmaktadır. YTA'da yük dengelemede daha başarılı olmak için yükün bağlantılara daha az bellek ve yüksek verim ile hızlı ve eşit bir şekilde dağıtılması sağlanmalıdır. Önerilen yöntemin bu amaca ulaşmadaki başarısı ölçülmektedir. İlk olarak, yöntemin test edileceği ağ topolojisi hazırlanış ve detayları Bölüm 2.3.1.'de verilmiştir. Ardından, iperf yazılım aracı ile bu topoloji üzerinde trafik oluşturulmaktadır. Iperf yazılım aracı trafik oluşturarak ağ performansını ölçmek amacıyla kullanılmaktadır. Bu araç ile TCP/UDP paketleri ile istenilen paket boyutunda, istenilen sayıda ve eşzamanlı olarak

istenilen port kullanılarak trafik oluşturulabilmektedir. Sonuç olarak, önerilen yöntemle üretilen trafiğin anahtarlara ve bağlantılara dengeli ve hızlı bir şekilde dağıtıldığını kanıtlayacak sonuçlar elde edilmiştir. Tasarlanan topoloji sekiz adet ana bilgisayar ve on adet anahtardan oluşmaktadır. Gönderilen paketlere göre önerilen yöntemde iki farklı senaryo denenmektedir. Senaryolar, rastgele ve düzenli paketlerle oluşturulan senaryolardır. Alt başlıklarda bu iki senaryo için karşılaştırma sonuçları ve yöntemin yük dengeleme başarısını kanıtlayan grafikler verilmiştir.

3.2.1. İlk Trafik Senaryosu ve Çıktıları

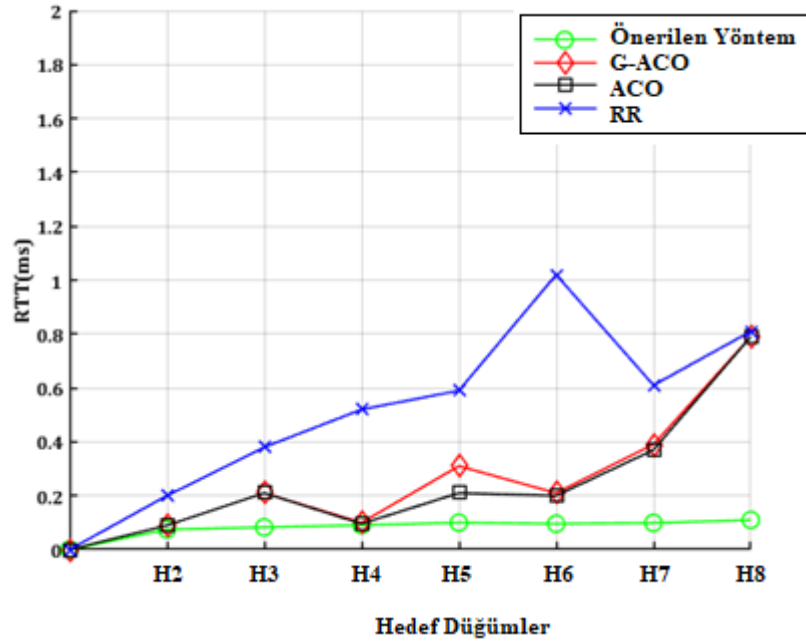
İlk trafik senaryosu oluşturulurken, ana bilgisayar 1'den (H1) diğer tüm ana bilgisayarlara gönderilen her bir yükün bant genişliği 100 Mbps'dir. Hostlar arası veri transferinde güvenilir iletişime dayalı TCP protokolü kullanılmaktadır. Veri aktarımı sırasında ağın performansını ölçmek için literatürde sıklıkla tercih edilen RTT, PLR, verimlilik ve arabellek boyutu metrikleri kullanılmaktadır. Bu metrikler, yük dağılımını, bağlantılarda ve anahtarlarda trafik sıkışıklık durumunu göstermektedir. Ayrıca bu metrikler, yöntemin bellek gereksinimini ve ağın veri aktarım limitini ölçmektedir, bu nedenle yöntemlerin doğru değerlendirilmesi için önemlidir.

Önerilen yöntemde başlangıç parçacıkları rastgele seçilmiştir, bu nedenle yöntemin bir kez uygulanması sonuçların doğruluğunu kanıtlamak için yeterli değildir. Önerilen yöntemin sonuçları karşılaştırmalı olarak belirlenirken, yöntemin 50 kez çalıştırılmasıyla elde edilen sonuçların ortalaması alınmıştır. Önerilen yöntem değerlendirilirken 5 dakika ve 10 dakika olmak üzere farklı simülasyon süreleri oluşturulmuştur. Böylece önerilen yöntemin kısa ve uzun vadeli veri aktarımındaki sonuçları görülebilmektedir. Önerilen yöntem sonuçları, RR, ACO ve G-ACO yöntemleri gibi bu alandaki temel yöntemlerle karşılaştırılır. Metrikler ve karşılaştırmalı sonuçlar bu bölümde ayrıntılı olarak verilecektir.

RTT: Bu metrik, kaynaktan sinyal gönderilmesi ile sinyalin hedefe varması arasında geçen sürenin toplamıdır. Çalışmada RTT değeri belirlenirken ping komutu kullanılmaktadır. RTT değeri, ağın performansını ölçmede önemlidir. Bu metrik doğrudan bağlantılarda tıkanıklık olup olmadığını ve dolayısıyla yük dengelemenin başarısını ortaya koymaktadır. RTT değeri ne kadar küçükse, yük dengeleme için yöntemin o kadar başarılı olduğu söylenebilir. Hesaplaması denklem 3.1'de verilmiştir.

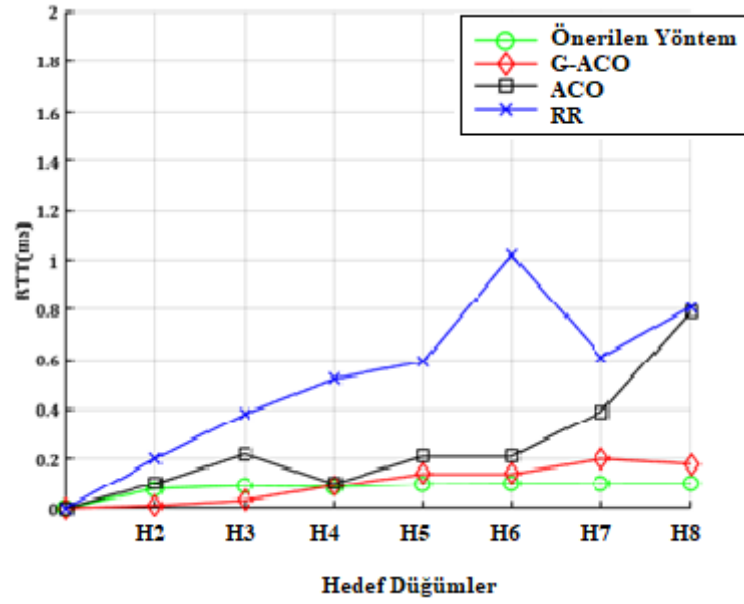
$$RTT = t_{arrival} - t_{start} \quad (3.1)$$

Önerilen yöntem için RTT değerleri Wireshark izleme aracından elde edilmiştir. RTT kullanılarak elde edilen sonuçların karşılaştırmaları sırasıyla 5 dakikalık ve 10 dakikalık simülasyon süresi için Şekil 3.5. ve Şekil 3.6.'da verilmiştir. Önerilen yük dengeleme yöntemleri, ağda sürekli veri alışverişi olurken zamandan bağımsız olarak yüksek performans sağlamalıdır. Bu nedenle önerilen çalışmada farklı simülasyon sürelerinde yöntemlerin sonuçları verilmiştir. Yöntemin farklı simülasyon zamanlarında literatürdeki [57, 114] yöntemlere göre yük dengelemede daha başarılı olduğu kanıtlanmıştır. Yani zamandan bağımsız olarak iyi sonuçlar üretmektedir.



Şekil 3.5. 5 dakikalık simülasyon süresinde RTT değerlerinin karşılaştırılması

Şekil 3.5.'te görüldüğü üzere, önerilen yöntem daha verimli sonuçlar vermektedir. Önerilen yöntem, kaynaktan hedefe ulaşmada literatürdeki diğer yöntemlere göre daha küçük RTT değerlerine sahiptir. Ayrıca kaynak ve hedef arasındaki hop sayısı, farklı yöntemlerin RTT değerlerini olumsuz etkilerken önerilen yöntem hop sayısından bağımsız olarak kararlı sonuçlar üretmektedir.



Şekil 3.6. 10 dakikalık simülasyon süresinde RTT değerlerinin karşılaştırılması

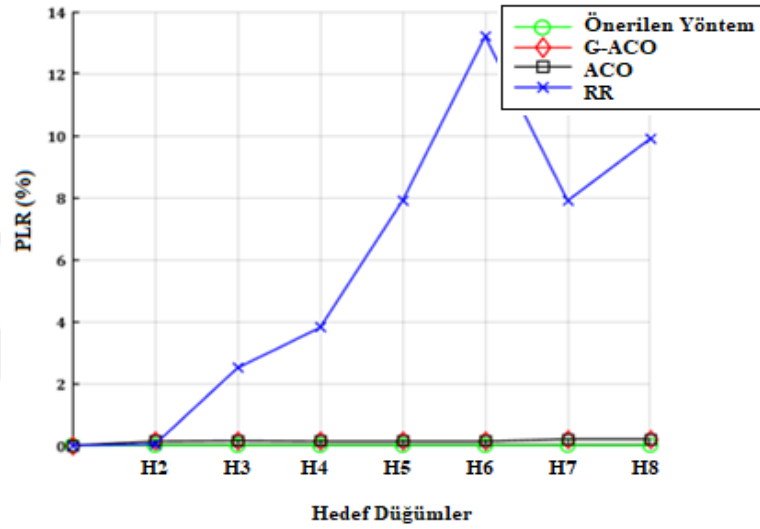
Daha uzun bir simülasyon süresi oluşturularak, RTT metriğine dayalı sonuçlar Şekil 3.6.'da verilmiştir. Buradaki amaç, daha uzun süreli trafik üretilirken artan akış sayısına göre önerilen yöntemin sınırlarını test etmektir. Şekil 3.6. incelendiğinde önerilen yöntemin diğer yöntemlere göre hedef ana bilgisayarların çoğunda daha düşük RTT değerlerine sahip olduğu görülmektedir.

PLR: Performans değerlendirmesi için bir başka metrik olan paket kaybı, hizmet kalitesi ile doğrudan ilişkilidir. Bir iletim aralığında iletilen ve alınan paketlerin toplam sayısı ile hesaplanmakta ve denklem 3.2 ile ifade edilmektedir. Ağdaki bağlantılarda tıkanıklık veya ağ üzerindeki erişilemeyen cihazlarda paket kayıpları artar. Bu nedenle, bu metrik sonuçları ağdaki sıkışıklık miktarı ve araçların kullanılabilirliğini ortaya koymaktadır. Ek olarak, paket kaybı metriği ağın verimliliğini doğrudan etkiler. Birçok ağ uzmanı, paket kaybı arttığında ve ağın verimliliğini etkileyen iletim hızını azalttığında tıkanıklık olduğunu belirtir.

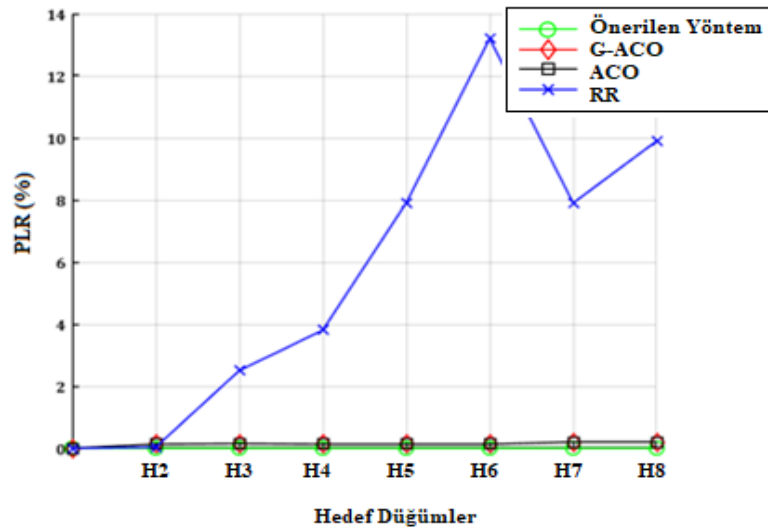
$$PLR = \frac{N^{tx} - N^{rx}}{N^{tx}} \times 100\% \quad (3.2)$$

Denklem 3.2'de N^{tx} iletilen toplam paket sayısıdır ve N^{rx} alınan toplam paket sayısıdır.

PLR metrik sonuçları iperf yazılım aracı kullanılarak elde edilmiştir. Karşılaştırma için kullanılan metrik olan PLR'lerin sonuçları, 5 dakikalık ve 10 dakikalık simülasyon süresi için sırasıyla Şekil 3.7. ve Şekil 3.8.'de verilmiştir. Bu sonuçlar, önerilen yöntemin simülasyon süresi artırılsa bile daha az paket kaybı ile yük dengeleme gerçekleştirdiğini kanıtlamaktadır.



Şekil 3.7. 5 dakikalık simülasyon süresinde PLR değerlerinin karşılaştırılması



Şekil 3.8. 10 dakikalık simülasyon süresinde PLR değerlerinin karşılaştırılması

Sonuçlara bakıldığında RR yük dengeleme yöntemi yüksek paket kayıplarına neden olurken, ACO, G-ACO ve önerilen yöntem birbirine yakın sonuçlar vermektedir. Önerilen yöntem neredeyse hiç paket kaybına neden olmazken, ACO ve G-ACO yöntemleri önerilen yönteme göre daha yüksek PLR değerlerine sahiptir.

Verimlilik (Throughput): Bu metrik, kaynaktan hedefe başarıyla aktarılan verileri doğru bir şekilde ölçmektedir. Bir TCP bağlantısının verimi, RTT, paket kaybı ve maksimum segment boyutundan etkilenmektedir. Mathis ve arkadaşları [115], bu üç çıktı değişkeni arasındaki ilişkiyi tanımlayan bir model (Mathis Denklemi) önerdiler. Mathis Denklemine göre ağdaki paket kayıpları, TCP paketlerinin gönderilme hızını önemli ölçüde azaltır. Bu duruma göre tasarlanan Mathis denklemi, denklem 3.3 ile temsil edilmektedir.

$$throughput = \frac{MSS}{RTT \cdot \sqrt{L}} \quad (3.3)$$

Denklemde, MSS maksimum segment boyutudur ve tez çalışmasında 1460 bayt olarak kullanılmıştır. Ayrıca, denklem 3.3'te gidiş-dönüş süresi RTT olarak ve paket kayıp oranı L ile temsil edilmektedir. Denklem 3.3'e göre, TCP bağlantısı MSS ile doğru orantılı ve RTT ve L'nin karekökü ile ters orantılıdır.

Verimlilik (Mbit/sn) değeri kullanılarak elde edilen sonuçların karşılaştırmaları, 5 dakikalık ve 10 dakikalık simülasyon süresi için sırasıyla Tablo 3.1 ve Tablo 3.2'de verilmiştir. Tablolardaki yöntemler arasındaki maksimum aktarım değerine sahip yöntem koyu yazılmıştır. Önerilen yöntemin her zaman maksimum verim sağladığı Tablo 3.1 ve Tablo 3.2 ile kanıtlanmıştır. Denklem 3.3, çıktı değerinin PLR ile ters orantılı olduğunu gösterir. Önerilen yöntem, kayıpsız veri iletimi sağladığı için literatürdeki diğer yöntemlere göre daha yüksek verimliliğe sahiptir. Ayrıca, çıktı ne kadar büyük olursa, ağın veri iletim limitleri de o kadar yüksek olur. Dolayısıyla sonuçlara bakıldığında önerilen yük dengelemenin kullanıldığı ağlarda veri aktarımının maksimum düzeyde gerçekleştiği söylenebilir.

Tablo 3.1. 5 dakikalık simülasyon süresi boyunca Verimlilik değerinin karşılaştırılması

Yöntemler	Verimlilik (Mbit/sec)							
	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8	Ort.
G-ACO	44992	17951	40493	13062	19282	8370	4132	21183
ACO	44992	17951	42180	19282	20246	8823	4132	22515
RR	42146	2430	1440	880	394	851	572	6959
Önerilen Yöntem	194666	175903	160439	146000	152083	147474	132727	158470

Tablo 3.2. 10 dakikalık simülasyon süresi boyunca Verimlilik değerinin karşılaştırılması

Yöntemler	Verimlilik (Mbit/sec)							
	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8	Ort.
G-ACO	70244	52683	44992	28923	28923	18848	22496	38158
ACO	40493	17135	40493	19282	19282	8370	4132	21312
RR	42146	2430	1440	880	394	851	572	6959
Önerilen Yöntem	189610	160439	171764	150515	146000	147474	146000	158829

Arabellek Boyutu: Arabellek boyutu, ağın gelen verileri işlemesi için geçen süredir. Daha küçük arabellekler, arabellek boyutunu, güç tüketimini ve yönlendiricinin arabelleğini oluşturmak için gereken hızlı bellek miktarını önemli ölçüde azaltabilir. Daha küçük arabellekler, hedef bağlantıda tam bağlantı kullanımını sürdürürken kuyruk gecikmelerini de azaltabilir [116]. Arabellek boyutu metriğinin hesaplanması denklem 3.4’de verilmiştir.

$$B = RTT \times C \quad (3.4)$$

Denklemde B, arabellek boyutunu temsil etmektedir. C ise maksimum bağlantı kapasitesidir ve tez çalışmasında 100 Mbps olarak kullanılmıştır.

Arabellek boyutu metriği kullanılarak elde edilen sonuçların karşılaştırmaları, 5 dakikalık ve 10 dakikalık simülasyon süresi için sırasıyla Tablo 3.3 ve Tablo 3.4’te verilmiştir.

Tablo 3.3. 5 dakikalık simülasyon süresi boyunca Arabellek Boyutu değerinin karşılaştırılması

Arabellek Boyutu (KB)								
Yöntemler	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8	Ort.
G-ACO	1.1	2.6	1.2	3.8	2.6	4.8	9.6	3.7
ACO	1.1	2.6	1.2	2.6	2.4	4.5	9.6	3.4
RR	2.4	4.6	6.3	7.2	12.5	7.4	9.9	7.2
Önerilen Yöntem	0.9	1.0	1.1	1.2	1.2	1.2	1.3	1.1

Tablo 3.4. 10 dakikalık simülasyon süresi boyunca Arabellek Boyutu değerinin karşılaştırılması

Arabellek Boyutu (KB)								
Yöntemler	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8	Ort.
G-ACO	0.7	1.0	1.1	1.7	1.7	2.4	2.2	1.5
ACO	1.2	2.7	1.2	2.6	2.6	4.8	9.6	3.5
RR	2.4	4.6	6.3	7.2	12.5	7.4	9.9	7.2
Önerilen Yöntem	0.9	1.1	1.0	1.2	1.2	1.2	1.2	1.1

Tablo 3.3 ve Tablo 3.4, 100 Mbps bant genişliği için gereken arabellek boyutu karşılaştırma sonuçlarını verilmektedir. Tablolardaki yöntemler arasındaki minimum arabellek boyutu gerektiren yöntem değeri koyu yazılmıştır. Önerilen yöntemin genellikle minimum arabellek boyutu sağladığı tablolarla kanıtlanmıştır. Birkaç değer için ufak farklarla skoru kaçırmış olsa bile, ortalamaya bakıldığında başarısı görülebilir. Düşük bellek kullanımı da gecikmeyi azaltmaktadır, böylece kaynak ve hedef arasındaki akışlar minimum sürede iletilmektedir.

Tüm metrik sonuçları değerlendirildiğinde önerilen yöntemin performansının simülasyon zamanından etkilenmediği görülmektedir. Yani tasarlanan yöntem hem kısa süreli hem de uzun süreli süreçlerde iyi sonuçlar vermektedir. Önerilen yöntemde, değişen ağ koşullarına uyum sağlamak için yol seçimi dinamik olarak yapılmaktadır. Yani belirli bir süre sonunda daha uygun bir rota bulmak için yeniden hesaplamalar yapılır. Önerilen yöntemin bu dört metrik sonucu ile dinamik olarak belirlenen rotanın yük dengeleme

problemi için en uygun çözümü sağladığı kanıtlanmıştır. En uygun yolu bulmak, yükün bağlantılardaki eşit dağılımı ile doğrudan ilgilidir.

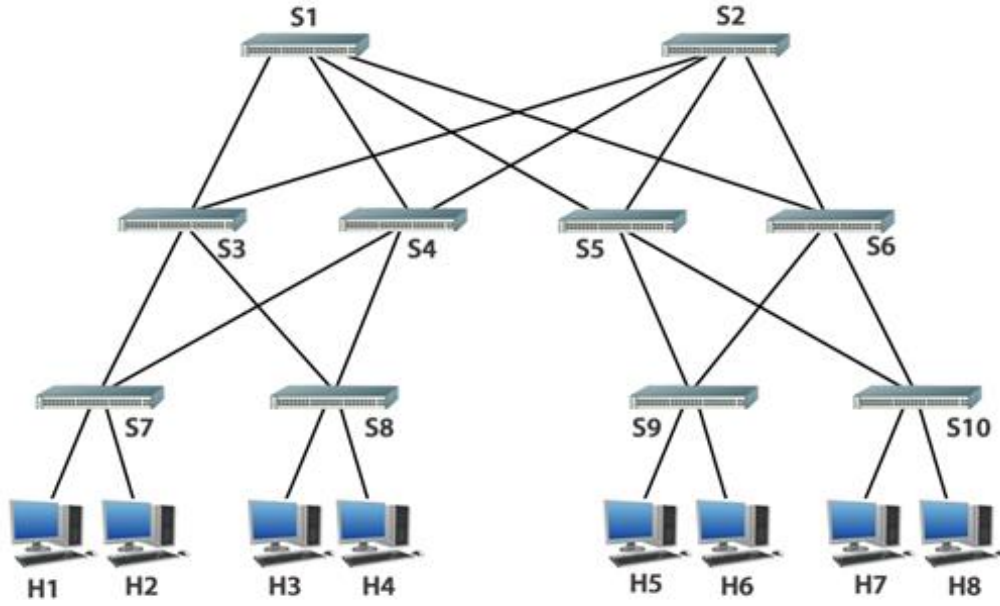
5 dakika ve 10 dakikalık simülasyon sonuçlarındaki düşük RTT değerleri, en az yüklenen rotanın seçildiğini göstermekte, dolayısıyla bağlantılarda sıkışıklığa neden olmadan yükün eşit olarak dağıldığı, sonuçlardan anlaşılmaktadır. Karşılaştırma için kullanılan başka bir metrik PLR'dir. Bu metrik sonuçları, seçilen rotanın ek yüke neden olup olmadığı ile ilgilidir. Sonuç olarak önerilen yöntemin diğer yöntemlere göre en az yüklenen yolu seçtiği görülmektedir ki bu da yükün bağlantılar arasında eşit olarak dağıldığının kanıtıdır. Başka bir karşılaştırma metriği verimliliktir. Sonuçlarda görüldüğü gibi seçilen yollar için verimlilik değeri literatürdeki yöntemlerden daha yüksektir. Dolayısıyla bu durum önerilen yöntemle yük dengeleme yapılırsa ağı yüksek veri aktarım kapasitesine sahip olacağını kanıtlamaktadır. Arabellek boyutu, çalışmada kullanılan son metriktir. Bu çalışma, 100 Mbps bant genişliği sağlamak için gereken arabellek boyutunu ölçmektedir. Sonuçlar, önerilen yük dengeleme yönteminin bu veri aktarımını literatürdeki diğer yöntemlere göre daha az bellek gereksinimi ile gerçekleştirdiğini göstermektedir. Ayrıca düşük bellek gereksinimi, düşük enerji tüketimi ile doğrudan ilişkilidir, bu nedenle önerilen yöntemin daha düşük güç gereksinimleri ile yüksek veri aktarımı gerçekleştirdiği söylenebilir.

Önerilen yöntemin tüm simülasyon sonuçları dikkate alındığında hop sayısından bağımsız olarak kararlı sonuçlar ürettiği görülmektedir. Örneğin, kaynak H1 ana bilgisayarını hedef H3 ana bilgisayarına, hedef H8 ana bilgisayarından daha az hop sayısı ile bağlı olsa da, simülasyon sonuçlarındaki H3 ana bilgisayar ve H8 ana bilgisayarının metrik ölçümleri birbirine yakındır. Ayrıca önerilen yöntemde hibrit tasarımı maliyet fonksiyonunun yük dengeleme problemine çözüm olarak seçilebileceği tüm bu sonuçlarla kanıtlanmıştır.

3.2.2. İkinci Trafik Senaryosu ve Çıktıları

İkinci trafik senaryosu, önerilen yöntemin yükü bağlantılara ve anahtarlara eşit olarak dağıttığını gözlemlemek için geliştirilmiştir. İkinci trafik senaryosu Şekil 2.1.'de verilen ağ topolojisi kullanılarak oluşturulmuştur. Bu senaryoda trafik, H2 ana bilgisayarından diğer ana bilgisayarlara 750 saniye boyunca rastgele ping mesajları gönderilerek oluşturulmuştur. Rastgele ping mesajları, hping yazılım aracı kullanılarak oluşturulmaktadır.

Fat tree ağ topolojisi hiyerarşik yapı kullanılır, bu nedenle ağ topolojisindeki tüm bağlantıların yükünün eşit olması beklenmez. Uygulanacak yöntemler aynı katmandaki bağlantılar üzerindeki yükü dengelemelidir, bu nedenle önerilen yöntemdeki hibrit maliyet fonksiyonu bu amacı gerçekleştirecek şekilde tasarlanmıştır. Ağ topolojisinin hiyerarşik yapısı Şekil 3.9. ile verilmiştir.



Şekil 3.9. Ağ topolojisinin hiyerarşik yapısı

Şekil 3.11. ve Şekil 3.12. bağlantılardaki yükü dengelemede önerilen yaklaşımın başarısını kanıtlamak için verilmiştir. Önerilen yaklaşım için kullanılan topolojinin tüm bağlantıları Şekil 3.10.'da verilmiştir.

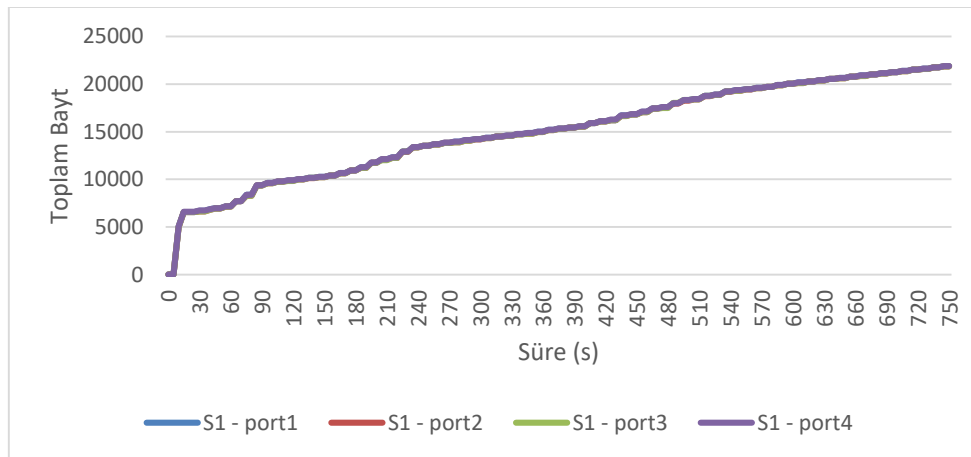
```

mininet> links
s1-eth1<->s3-eth1 (OK OK)
s1-eth2<->s4-eth1 (OK OK)
s1-eth3<->s5-eth1 (OK OK)
s1-eth4<->s6-eth1 (OK OK)
s2-eth1<->s3-eth2 (OK OK)
s2-eth2<->s4-eth2 (OK OK)
s2-eth3<->s5-eth2 (OK OK)
s2-eth4<->s6-eth2 (OK OK)
s3-eth3<->s7-eth1 (OK OK)
s3-eth4<->s8-eth1 (OK OK)
s4-eth3<->s7-eth2 (OK OK)
s4-eth4<->s8-eth2 (OK OK)
s5-eth3<->s9-eth1 (OK OK)
s5-eth4<->s10-eth1 (OK OK)
s6-eth3<->s9-eth2 (OK OK)
s6-eth4<->s10-eth2 (OK OK)
s7-eth3<->h1-eth0 (OK OK)
s7-eth4<->h2-eth0 (OK OK)
s8-eth3<->h3-eth0 (OK OK)
s8-eth4<->h4-eth0 (OK OK)
s9-eth3<->h5-eth0 (OK OK)
s9-eth4<->h6-eth0 (OK OK)
s10-eth3<->h7-eth0 (OK OK)
s10-eth4<->h8-eth0 (OK OK)

```

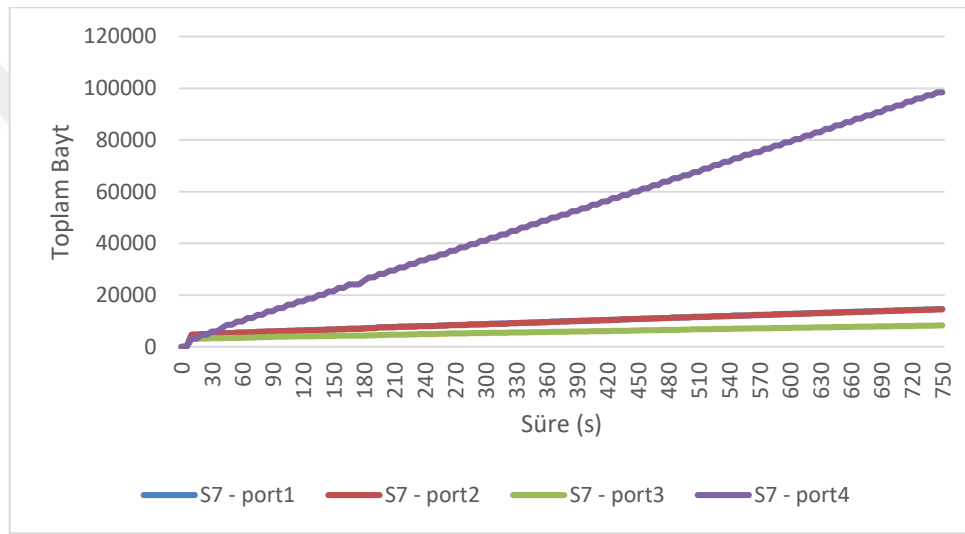
Şekil 3.10. Oluşturulan topolojinin tüm linkleri

Topoloji incelendiğinde bazı bağlantılar dengeli bir dağılım göstermelidir. Örneğin, Şekil 3.9.'a bakıldığında, Anahtar 1 (S1), bir çekirdek anahtardır, bu nedenle tüm bağlantılarındaki yükün eşit olarak dağıtılması beklenir. Şekil 3.11.'de görüldüğü gibi akışlar, Anahtar 1'in dört bağlantısı boyunca eşit olarak dağıtılarak önerilen yöntemin başarılı olduğunu kanıtlamıştır. Tüm topoloji için bu dengeyi korumak, tekniğin temel amacıdır.



Şekil 3.11. S1 bağlantılarının yük dağılımı

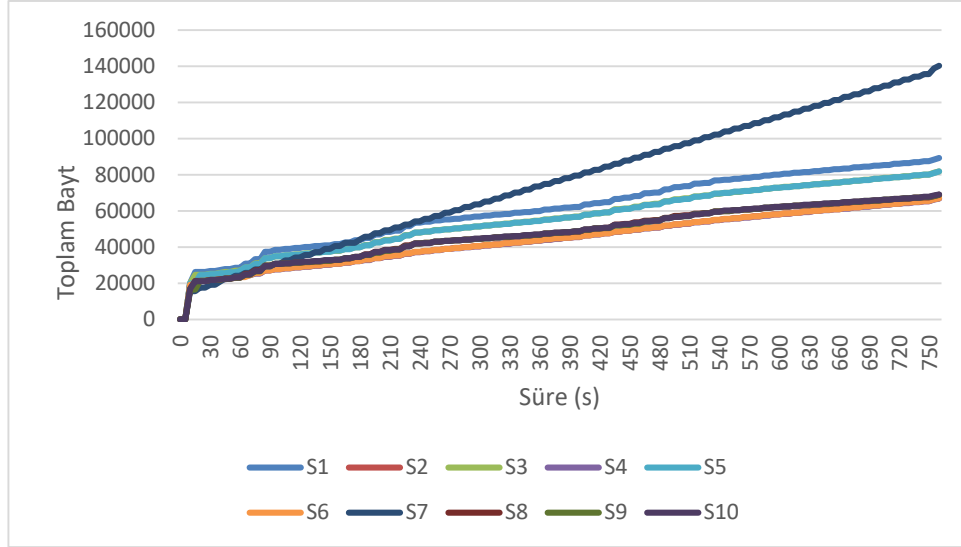
Farklı bir örnek olarak, yükün farklı katmanlardaki dağılımını göstermek için, Anahtar 7 (S7)'nin (kenar anahtarı) bağlantı sonuçları Şekil 3.12.'de verilmiştir. Anahtar 7'nin sahip olduğu birinci ve ikinci portlar üst katmanda yer alırken, üçüncü ve dördüncü portlar alt katmanda yer almaktadır. Birinci ve ikinci portların yük dağılımlarının dengeli olduğu görülmektedir. Üçüncü ve dördüncü portlar ana bilgisayarlara bağlıdır, dolayısıyla dengeli bir dağıtım beklenemez. Dördüncü port H2 ana bilgisayarına bağlıdır ve tüm istekler bu bağlantı üzerinden oluşturulduğu için daha fazla yüklenir. Üçüncü bağlantı noktasının yükü çoğunlukla H1 ana bilgisayarına yapılan isteklere bağlıdır.



Şekil 3.12. S7 bağlantılarının yük dağılımı

Şekil 3.11. ve Şekil 3.12., 750 saniyelik ağ trafiğinden kaynaklanan yük dağılımlarını göstermektedir. Bu örnekler (S1 ve S7'nin bağlantı yük dağılımları), yöntemin yükü hiyerarşik yapının tüm katmanlarına eşit olarak dağıttığını kanıtlamaktadır.

Ayrıca önerilen yöntemde yük dengeleme, anahtar tabanlı ve bağlantı tabanlı olduğundan, anahtarların yük dağılımı da birbirine yakındır. Anahtar üzerindeki yük, anahtara bağlı tüm bağlantı noktalarından alınan toplam veri miktarına eşittir. Verilen topolojideki tüm anahtarların yük dağılımları Şekil 3.13.'te verilmiştir. S7 anahtarındaki yük ne kadar yüksekse, bu anahtardaki H2 ana bilgisayarından o kadar fazla istek üretilir. Şekil 3.13., S7 anahtarı dışındaki anahtarlarda yükün dengeli olarak dağıldığını göstermektedir.



Şekil 3.13. Topolojideki tüm anahtarların yük dağılımları

Şekil 3.13.'te gösterildiği gibi, S7 istek üreten ana bilgisayara bağlı olduğundan dolayı en yüksek yüke sahiptir ve bunu S1 takip etmektedir. S3 ve S5 yaklaşık olarak aynı yüke sahiptir ve bir sonraki en yüksek değere sahiptir. S8, S9 ve S10 sonraki en yüksek yüke sahiptir ve yaklaşık değerlerdir. En ihmal edilebilir yüke sahip, yaklaşık olarak aynı olan anahtarlar S2, S4 ve S6'dır.

Bağlantı tabanlı yük dengelemede oluşturulan birinci ve ikinci trafik senaryoları SCI alan indeksli bir dergide yayınlanmıştır [117]. Ayrıca önerilen çalışma Karadeniz Teknik Üniversitesi Bilimsel Araştırma Projeleri Birimi tarafından desteklenmiştir. Proje numarası: FAY-2021-9701.

3.3. YTA Fil Akış Tespiti Sonuçları

Fil akış tespiti için önerilen algılayıcı, ağdaki akışları fil ve fare olmak üzere iki sınıfa ayırabilen 7 farklı sınıflandırıcıyla denenmiştir. Bunlar Lojistik Regresyon, DVM, Karar Ağaçları, Rastgele Ormanlar, Naive Bayes, KNN ve Derin Öğrenmeye dayalı yöntemlerdir. Başarılı bir sınıflandırma için en kritik parametre özellik seçimidir. Önerilen algılayıcıda literatürdeki çalışmalardan farklı olarak FlowRecorder kullanılarak her bir akış için 11 farklı öznetelik çıkarılmıştır. Bu özellikler, akışları açıkça ayırt edebilen özelliklerdir. Tek bir özelliğin kullanılması, kolayca ayrıştırılabilen veri kümelerinde işe yarar ancak karmaşık

veri kümelerinde sorunlara neden olmaktadır. Literatürdeki çalışmalar sadece paket boyutunu özellik olarak kullanmaktadır. Bu durum önerilen algılayıcıyı literatürdeki çalışmalardan üstün kılmaktadır. Bu yöntemler değerlendirilirken sınıflandırıcıların başarısı kadar önemli olan sınıflandırma süreleri de incelenmektedir. Gelen akışın küçük bir gecikmeyle sınıflandırılması, gerçek zamanlı ağlarda tıkanıklık sorununu en aza indirir. Bu nedenle, sınıflandırıcıların çalışma süreleri çok önemlidir. Literatürde, fil akışının belirlenmesinde sınıflandırıcıların zaman ve doğruluk açısından incelendiği zengin bir çalışma bulunmamaktadır.

UNI1 ve UNI2 veri kümeleri, bir üniversite kampüsünde yapılan bir çalışmadan elde edilmiştir. Yüksek bant genişliği gerektiren bu veri kümelerinde yedekleme, ses, video aktarımı gibi akışların yanı sıra ICMP gibi küçük paketler de bulunmaktadır. Bu nedenle, bu veri kümeleri fil akışı ve fare akışı sınıflandırması için çok uygundur. Veri setleri, %70 eğitim verisi ve %30 test verisi olmak üzere iki bölüme ayrılmıştır. Önerilen algılayıcıda her bir veri seti için ayrı ayrı değerlendirme yapılmaktadır.

Veri setleri paketlerden oluşur ve önce akışlara çevrilmeleri gerekir. FlowRecorder kullanılarak veri kümelerindeki paketler akışlara dönüştürülür ve her akış için 11 farklı özellik çıkarılır. Daha sonra bu veri setleri 7 farklı sınıflandırıcı ile ayrı ayrı sınıflandırılarak sonuçlar elde edilmiştir. Her sınıflandırıcı için en iyi parametreler Izgara Arama Yöntemi ile belirlenmiştir.

Sınıflandırma işleminin başarısını ölçmek için literatürde sıklıkla tercih edilen karşılaştırma metrikleri olan Doğruluk, Kesinlik, Hatırlama ve F-puanı kıyaslama kullanılmıştır. Bu metrikler Şekil 3.14. ile ifade edilen hata matrisiyle elde edilebilen metriklerdir. Ayrıca çalışma süresi de karşılaştırma metriği olarak kullanılmıştır. Gelen akışın fil veya fare olarak sınıflandırılması için geçen süre ağdaki gecikmeyi belirler. Bu nedenle, en iyi sınıflandırıcıyı seçerken sınıflandırma süreleri ve başarısı esastır.

		Tahmin Edilen Değerler	
Gerçek Değerler		Negatif	Pozitif
	Negatif	GN	YP
	Pozitif	YN	GP

Şekil 3.14. Hata matrisi

Şekil 3.14.’te Gerçek Negatif değeri GN ile, Yanlış Pozitif değeri YP ile, Yanlış Negatif değeri YN ile ve Gerçek Pozitif değeri GP ile temsil edilmektedir.

Gerçek Pozitif (GP): Modelin pozitif sınıfı ne ölçüde doğru tahmin ettiğini ölçer. Örneğin, “A” ve “B” sınıflarına sahip bir ikili sınıflandırma probleminde, amacımız “A” sınıfını doğru tahmin etmekse, o zaman gerçek bir pozitif, modelimizin doğru bir şekilde tahmin ettiği “A” sınıfı örneklerinin sayısı olacaktır [118].

Yanlış Pozitif (YP): Model, bir örneğin gerçekte olmadığı bir sınıfa ait olduğunu tahmin ettiğinde, yanlış pozitifler meydana gelir. Örneğin, bir tıbbi tanı modelinin yanlış pozitiflik oranı yüksekse, hastaların gereksiz tedavi görmesine neden olabilir. Yanlış pozitif oranı, pozitif olarak tahmin edilen tüm negatif örneklerin oranıdır [118].

Gerçek Negatif (GN): Modelin negatif olarak doğru bir şekilde öngördüğü sonuçlardır. Örneğin, model bir kişinin hastalığı olup olmadığını tahmin ediyorsa, model kişinin hastalığı olmadığını ve aslında hastalığı olmadığını tahmin ettiğinde gerçek bir negatif olur. Gerçek negatifler, bir sınıflandırma modelinin ne kadar iyi performans gösterdiğini değerlendirmek için kullanılan ölçülerden biridir [118].

Yanlış Negatif (YN): Bir model, bir örneği gerçekten pozitifken negatif olarak tahmin ettiğinde yanlış bir negatif oluşur. Yanlış negatifler, özellikle tıp alanında çok maliyetli olabilir. Örneğin, bir kanser tarama testi, bir hastanın gerçekten kanser olduğunda kanser olmadığını tahmin ederse, bu, hastalığın tedavisiz ilerlemesine neden olabilir [118].

Şekil 3.14.’teki pozitif değerler fil akışları, negatif değerler ise fare akışları temsil etmektedir. Doğruluk, Kesinlik, Hatırlama ve F-puanı kıyaslama için kullanılan metrikler bu bölümde detaylandırılmıştır.

Doğruluk, toplam örnek sayısı içindeki doğru sınıflandırma örneklerinin yüzdesidir. Yani, yapılan tüm tahminlerden gerçek pozitif ve gerçek negatiflerin toplamının oranını ölçmek için kullanılır [118]. Metrik, denklem 3.5 kullanılarak hesaplanmaktadır.

$$Doğruluk = \frac{GP+GN}{GP+YN+GN+YP} \quad (3.5)$$

Kesinlik, gerçek pozitiflerin toplam sayısının, yanlış ve gerçek pozitiflerin toplamına bölümüdür. Yani, yapılan tüm pozitif tahminlerden doğru şekilde gerçek pozitiflerin sayısını ölçmede kullanılır [118]. Metrik, denklem 3.6 kullanılarak hesaplanmaktadır.

$$Kesinlik = \frac{GP}{YP+GP} \quad (3.6)$$

Hatırlama, gerçek pozitiflerin sayısının, yanlış negatiflerin ve gerçek pozitiflerin toplamına bölümüdür. Yani, tüm gerçek pozitif değerlerden doğru bir şekilde gerçek pozitiflerin sayısını ölçmek için kullanılır [118]. Metrik, denklem 3.7 kullanılarak hesaplanmaktadır.

$$Hatırlama_Puanı = \frac{GP}{YN+GP} \quad (3.7)$$

F-puanı, Kesinlik metriği ve Hatırlama metriğinin harmonik ortalamasıdır. Metrik, denklem 3.8 kullanılarak hesaplanmaktadır.

$$F - Puanı = 2 \times \frac{kesinlik \times hatırlama}{kesinlik+hatırlama} \quad (3.8)$$

Ayrıca değerlendirme metriği olarak akış tespit zamanı da kullanılmıştır. Akış tespit zamanı, her bir test verisinin başlangıcından bitişine kadar geçen süredir. Metrik, denklem 3.9 kullanılarak hesaplanmaktadır.

$$T_{koşma} = T_{bitiş} - T_{başlangıç} \quad (3.9)$$

Önerilen algılayıcının sonuçları literatürdeki farklı çalışmalar ile karşılaştırılmıştır. Bu çalışmalar fil akışlarını ve fare akışlarını sınıflandıran nadir yöntemlerdendir. Literatürdeki

çalışmalar genellikle sınıflandırma yaklaşımları yerine sabit eşikler kullanan verimsiz süreçleri içermektedir. Bu nedenle, sınıflandırmaya dayalı akış tespiti için sonuç bulmak oldukça zordur. Sınıflandırma başarısını ölçmek için her bir sınıflandırıcı için Doğruluk, Kesinlik, Hatırlama ve F-puanı metrikleri hesaplanmıştır. Ayrıca önerilen algılayıcıda sınıflandırıcıların zaman karmaşıklığı da incelenmiştir. Lokal ve global ağlarda paketler en kısa sürede hedeflerine ulaşmalıdır. Bu nedenle, sınıflandırıcıların mümkün olan en kısa sürede maksimum doğruluğu bulmaları beklenir. Önerilen algılayıcının Doğruluk, Kesinlik ve Hatırlama metrik sonuçları ve karşılaştırılan çalışmalar UNI1 veri seti ve UNI2 veri seti için sırasıyla Tablo 3.5. ve Tablo 3.6. ile verilmektedir. Tablolar incelendiğinde, Karar Ağaçları, Rastgele Ormanlar ve Derin Öğrenme yöntemlerinin hem UNI1 veri seti hem de UNI2 veri seti için en iyi sonuçları verdiği görülmektedir.

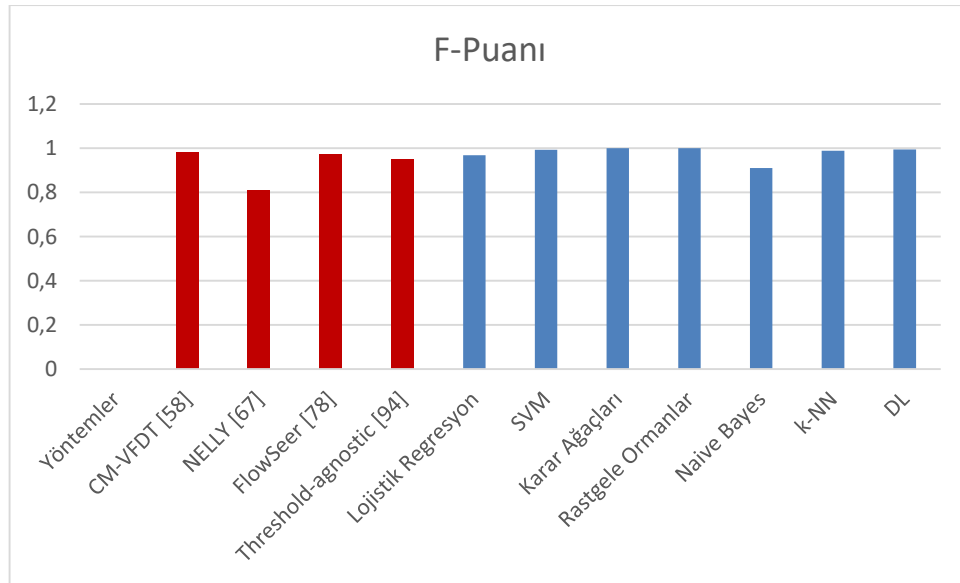
Tablo 3.5. UNI1 veri seti kullanılarak mevcut fil akış algılama yöntemleriyle genel performans karşılaştırması

Veri Seti	UNI1 Veri Seti		
Metrikler	Doğruluk	Kesinlik	Hatırlama
Yöntemler			
CM-VFDT [58]	0.998	0.98	0.99
NELLY [67]	0.78	0.70	0.96
FlowSeer [78]	0.982	0.97	0.98
Threshold-agnostic [94]	0.95	0.98	0.92
Önerilen Algılayıcı ile Lojistik Regresyon	0.995	0.974	0.965
Önerilen Algılayıcı ile DVM	0.999	0.987	0.999
Önerilen Algılayıcı ile Karar Ağaçları	1.000	1.000	1.000
Önerilen Algılayıcı ile Rastgele Ormanlar	1.000	1.000	1.000
Önerilen Algılayıcı ile Naive Bayes	0.984	0.998	0.838
Önerilen Algılayıcı ile KNN	0.998	0.996	0.982
Önerilen Algılayıcı ile DL	0.998	0.996	0.993

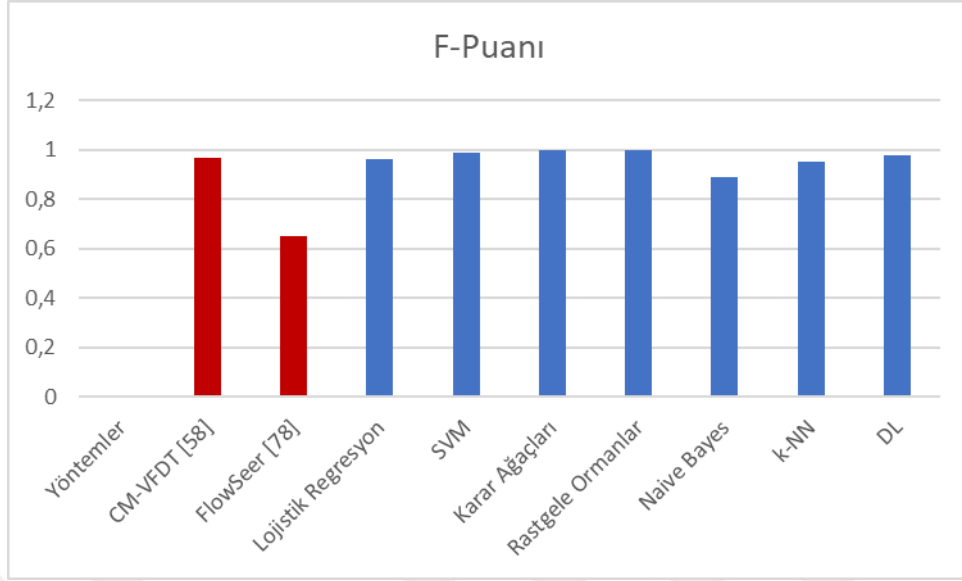
Tablo 3.6. UNI2 veri seti kullanılarak mevcut fil akış algılama yöntemleriyle genel performans karşılaştırması

Veri Seti	UNI2 Veri Seti		
Metrikler	Doğruluk	Kesinlik	Hatırlama
Yöntemler			
CM-VFDT [58]	0.988	0.97	0.98
FlowSeer [78]	0.984	0.57	0.75
Önerilen Algılayıcı ile Lojistik Regresyon	0.997	0.968	0.944
Önerilen Algılayıcı ile DVM	0.999	0.998	0.984
Önerilen Algılayıcı ile Karar Ağaçları	1.000	1.000	1.000
Önerilen Algılayıcı ile Rastgele Ormanlar	1.000	1.000	1.000
Önerilen Algılayıcı ile Naive Bayes	0.993	0.812	0.987
Önerilen Algılayıcı ile KNN	0.997	0.941	0.950
Önerilen Algılayıcı ile DL	1.000	1.000	0.962

Ayrıca çalışmada uygulanan 7 farklı sınıflandırıcı, F-puanı metriği kullanılarak literatürdeki diğer sınıflandırıcılarla karşılaştırılmıştır. UNI1 veri seti ve UNI2 veri seti için karşılaştırma sonuçları sırasıyla Şekil 3.15. ve Şekil 3.16. ile gösterilmektedir.



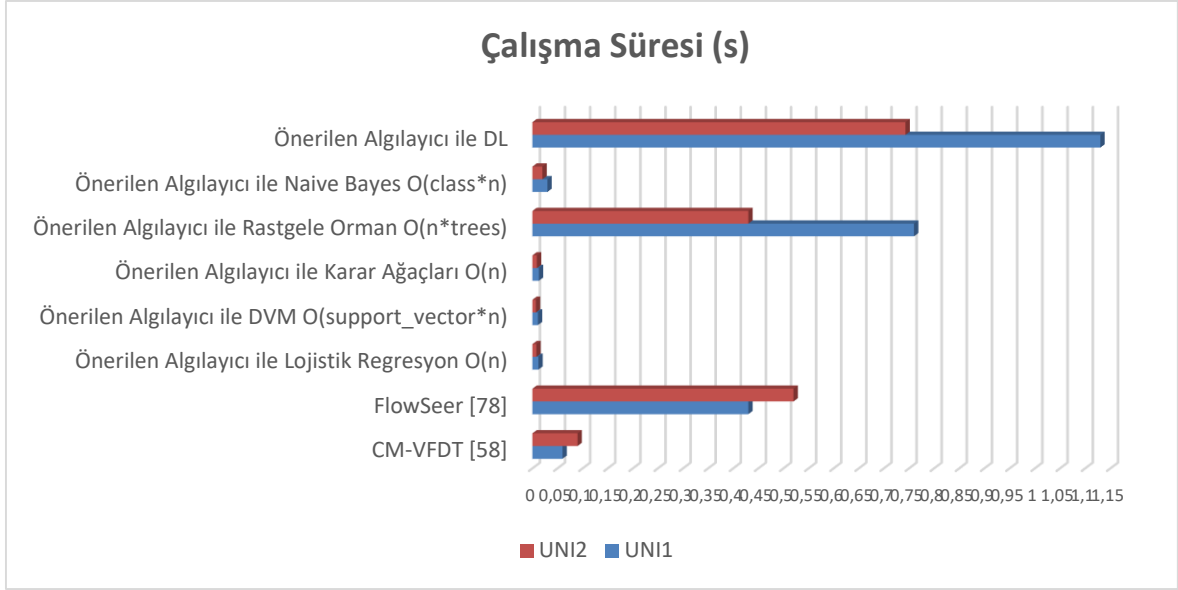
Şekil 3.15. UNI1 veri seti için F-puanı değeri karşılaştırması



Şekil 3.16. UNI2 veri seti için F-puanı değeri karşılaştırması

Şekil 3.15. ve Şekil 3.16.'da literatürdeki sonuçlar kırmızı renk ile, çalışmada uygulanan 7 farklı sınıflandırma yöntemi ise mavi renk ile temsil edilmektedir. UNI1 veri seti daha az karmaşık olduğundan, verilen yöntemlerin çoğu için %95'in üzerinde başarı elde edilmiştir. Çalışmanın deterministik ve daha karmaşık olan UNI2 veri seti için uygulanan sınıflandırıcı yöntemlerinin sonuçları literatürle karşılaştırıldığında, DVM, Karar Ağaçları, Rastgele Ormanlar ve Derin Öğrenme yöntemlerinin başarısı dikkat çekicidir.

Çalışmada uygulanan sınıflandırıcı yöntemlerinin çalışma süreleri literatürdeki diğer tekniklerle karşılaştırılmıştır. UNI1 veri seti ve UNI2 veri seti için çalışma süresi karşılaştırması Şekil 3.17.'de verilmiştir. KNN yönteminin UNI1 veri seti ve UNI2 veri seti için çalışma süresi sonuçları sırasıyla 97.23s ve 13.64 s'dir. Değerler çok yüksek olduğu için, Şekil 3.17.'de hassasiyet açısından dahil edilmemiştir.



Şekil 3.17. UNI1 veri seti ve UNI2 veri seti için çalışma süresi karşılaştırması

YTA'daki en iyi sınıflandırıcı, mümkün olan en kısa sürede akış tespitini tamamlayan sınıflandırıcıdır. Şekil 3.17. incelendiğinde, en iyi çalışma süresine sahip sınıflandırıcı yöntemler Lojistik Regresyon, DVM ve Karar Ağaçlarıdır.

Fil akış tespiti sonuçları SCI alan indeksli bir dergide revizyon sonrası hakem değerlendirmesindedir.

3.4. YTA Fil Akışların Planlanması Sonuçları

Fil akış tespitinin yönlendirmeye etkisini kanıtlamak için önerilen çalışmada iki farklı ağ trafiği oluşturulmuştur. Oluşturulan birinci trafik, H1'den diğer 7 ana bilgisayara aynı anda 10KB'lık (%80) ve 100MB'lık (%20) TCP paketlerin 10 dakika boyunca iletilmesiyle elde edilmektedir. Oluşturulan ikinci trafik ise H1'den diğer 7 ana bilgisayara aynı anda 10KB'lık (%50) ve 100MB'lık (%50) TCP paketlerin eşit sayıda ve 10 dakika boyunca iletilmesiyle elde edilmektedir. Literatürde fil akış oranı %10 ile %25 arası verilmektedir. Tez çalışmasında, iki farklı trafik oluşturularak fil akış oranındaki değişikliğin karşılaştırma metriklerine etkisi gözlemlenmiştir. 10 dakika sonunda yöntemlerin RTT, verimlilik ve arabellek boyutu değerleri hesaplanmaktadır. UNI1 veri setindeki ve UNI2 veri setindeki fil akışlar ve fare akışlar Karar Ağaçları, DVM ve Derin Öğrenme yöntemleriyle ayrı ayrı

sınıflandırılmıştır. Fil akışların tamamı D-PSO yöntemiyle yönlendirilirken, fare akışlar Rand ve RR yöntemleriyle ayrı ayrı değerlendirilmiştir.

RTT değerlerinin karşılaştırılması sonuçları Tablo 3.7., Tablo 3.10. ve Tablo 3.13. ile verilmektedir. Tablo 3.7. ile fil akış tespiti uygulanmadan yapılan yönlendirmenin RTT değerlerinin karşılaştırılması verilmektedir. Tablo 3.10.'da fil akış tespiti sonrası ve fil akış oranının %20 olduğu birinci trafik için yapılan yönlendirmenin RTT değerlerinin karşılaştırılması verilirken Tablo 3.13.'te fil akış tespiti sonrası ve fil akış oranının %50 olduğu ikinci trafik için yapılan yönlendirmenin RTT değerlerinin karşılaştırılması verilmektedir.

Verimlilik değerlerinin karşılaştırılması sonuçları Tablo 3.8., Tablo 3.11. ve Tablo 3.14. ile verilmektedir. Tablo 3.8. ile fil akış tespiti uygulanmadan yapılan yönlendirmenin verimlilik değerlerinin karşılaştırılması verilmektedir. Tablo 3.11.'de fil akış tespiti sonrası ve fil akış oranının %20 olduğu birinci trafik için yapılan yönlendirmenin verimlilik değerlerinin karşılaştırılması verilirken Tablo 3.14.'te fil akış tespiti sonrası ve fil akış oranının %50 olduğu ikinci trafik için yapılan yönlendirmenin verimlilik değerlerinin karşılaştırılması verilmektedir.

Arabellek boyutu değerlerinin karşılaştırılması sonuçları Tablo 3.9., Tablo 3.12. ve Tablo 3.15. ile verilmektedir. Tablo 3.9. ile fil akış tespiti uygulanmadan yapılan yönlendirmenin arabellek boyutu değerlerinin karşılaştırılması verilmektedir. Tablo 3.12.'de fil akış tespiti sonrası ve fil akış oranının %20 olduğu birinci trafik için yapılan yönlendirmenin arabellek boyutu değerlerinin karşılaştırılması verilirken Tablo 3.15.'te fil akış tespiti sonrası ve fil akış oranının %50 olduğu ikinci trafik için yapılan yönlendirmenin arabellek boyutu değerlerinin karşılaştırılması verilmektedir.

Fil akış tespiti uygulanmadan yapılan yönlendirme, akışlar için herhangi bir sınıflandırma yapılmaksızın tüm akışların, aynı yöntem kullanılarak yönlendirilmesidir. Rand, RR ve D-PSO olmak üzere üç farklı yaklaşım denenmiştir. Tablo 3.7., Tablo 3.8. ve Tablo 3.9. incelendiğinde RTT, verimlilik ve arabellek boyutu değerlerinin en iyi olduğu yöntem D-PSO yöntemidir.

Tablo 3.7. Fil akış tespiti uygulanmadan yapılan yönlendirmenin RTT değerlerinin karşılaştırılması

Yöntemler	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8
Rand	0.11	0.24	0.26	0.32	0.39	0.51	0.81
RR	0.1	0.22	0.23	0.28	0.32	0.39	0.53
D-PSO	0.097	0.13	0.16	0.17	0.21	0.22	0.25

Tablo 3.8. Fil akış tespiti uygulanmadan yapılan yönlendirmenin Verimlilik (Mbit/sec) değerlerinin karşılaştırılması

Yöntemler	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8
Rand	474	217	200	163	133	102	64
RR	522	237	227	186	163	133	98
D-PSO	1204123	898461	730000	687058	556190	530909	467200

Tablo 3.9. Fil akış tespiti uygulanmadan yapılan yönlendirmenin Arabellek Boyutu (KB) değerlerinin karşılaştırılması

Yöntemler	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8
Rand	1.3	2.9	3.2	3.9	4.8	6.2	9.9
RR	1.2	2.7	2.8	3.4	3.9	4.8	6.5
D-PSO	1.2	1.6	2	2.1	2.6	2.7	3.1

Fil akış tespiti sonrası yapılan yönlendirme sonuçları için öncelikle akış sınıflandırma yapılmaktadır. Akış sınıflandırma yapılabilmesi için literatürde sıklıkla kullanılan UNI1 veri seti ve UNI2 veri seti tercih edilmiştir. UNI1 veri setindeki ve UNI2 veri setindeki fil akışların tespiti için Karar Ağaçları, DVM ve DL sınıflandırma yöntemleri ayrı ayrı uygulanmıştır. Akış sınıflandırma yapıldıktan sonra fil akışlar ve fare akışlar için ayrı ayrı yönlendirme stratejileri denenmiştir. Fil akışların tamamı D-PSO yöntemiyle yönlendirilirken, fare akışlar Rand veya RR yöntemiyle yönlendirilmişlerdir. Yapılan değerlendirmeler literatürdeki iki farklı yöntemle RTT, verimlilik ve arabellek boyutu metrikleri kullanılarak kıyaslanmaktadır. Karşılaştırma yapılan ECMP [74] ve Hedera [76] yöntemleri fil akış tespiti için sabit eşik değeri kullandığından, bu yöntemler veri setleriyle kullanılmamaktadır. Bu nedenle, literatürdeki bu çalışmalar için Tablo 3.10.-Tablo 3.15.'teki veri seti sütunu X ile temsil edilmektedir.

Fil akış tespiti sonrası ve fil akış oranının %20 olduğu birinci trafik için yapılan yönlendirmenin RTT değerlerinin karşılaştırılması sonuçları Tablo 3.10. ile fil akış tespiti sonrası ve fil akış oranının %50 olduğu ikinci trafik için yapılan yönlendirmenin RTT değerlerinin karşılaştırılması sonuçları ise Tablo 3.13. ile verilmektedir. Tablo 3.10. ve Tablo 3.13. ile temsil edilen RTT değerlerinin karşılaştırma sonuçları incelendiğinde çoğunlukla Derin Öğrenme Yöntemi ile fil akış tespiti yapılan ve fare akışların RR ile yönlendirildiği uygulamalar başarılı sonuçlar üretmektedir.

Fil akış tespiti sonrası ve fil akış oranının %20 olduğu birinci trafik için yapılan yönlendirmenin verimlilik değerlerinin karşılaştırılması sonuçları Tablo 3.11. ile fil akış tespiti sonrası ve fil akış oranının %50 olduğu ikinci trafik için yapılan yönlendirmenin verimlilik değerlerinin karşılaştırılması sonuçları ise Tablo 3.14. ile verilmektedir. Tablo 3.11. ve Tablo 3.14. ile temsil edilen verimlilik değerlerinin karşılaştırma sonuçları incelendiğinde çoğunlukla Derin Öğrenme Yöntemi ile fil akış tespiti yapılan ve fare akışların RR ile yönlendirildiği uygulamalar başarılı sonuçlar üretmektedir.

Fil akış tespiti sonrası ve fil akış oranının %20 olduğu birinci trafik için yapılan yönlendirmenin arabellek boyutu değerlerinin karşılaştırılması sonuçları Tablo 3.12. ile fil akış tespiti sonrası ve fil akış oranının %50 olduğu ikinci trafik için yapılan yönlendirmenin arabellek boyutu değerlerinin karşılaştırılması sonuçları ise Tablo 3.15. ile verilmektedir. Tablo 3.12. ve Tablo 3.15. ile temsil edilen arabellek boyutu değerlerinin karşılaştırma sonuçları incelendiğinde çoğunlukla Derin Öğrenme Yöntemi ile fil akış tespiti yapılan ve fare akışların RR ile yönlendirildiği uygulamalar başarılı sonuçlar üretmektedir.

Tablo 3.10. Fil akış tespiti sonrası yapılan yönlendirmenin RTT değerlerinin karşılaştırılması (Fil akış (%20) ve Fare akış (%80))

Veri Seti	Fil Akış Tespiti	Fil Akış Yönlendirme	Fare Akış Yönlendirme	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8
X	Sabit Eşik	ECMP [74]	Rand	0.098	0.12	0.12	0.11	0.16	0.18	0.17
X	Sabit Eşik	Hedera [76]	Rand	0.11	0.058	0.11	0.18	0.15	0.18	0.11
UNI1	Karar Ağaçları	D-PSO	Rand	0.079	0.088	0.011	0.018	0.016	0.051	0.031
			RR	0.085	0.08	0.01	0.016	0.026	0.040	0.028
UNI2	Karar Ağaçları	D-PSO	Rand	0.018	0.025	0.028	0.018	0.063	0.014	0.079
			RR	0.031	0.082	0.022	0.018	0.014	0.085	0.014
UNI1	DVM	D-PSO	Rand	0.01	0.013	0.014	0.041	0.068	0.02	0.093
			RR	0.094	0.056	0.058	0.029	0.050	0.048	0.078
UNI2	DVM	D-PSO	Rand	0.036	0.012	0.048	0.081	0.013	0.047	0.013
			RR	0.022	0.010	0.041	0.021	0.011	0.039	0.012
UNI1	DL	D-PSO	Rand	0.0091	0.014	0.011	0.018	0.013	0.010	0.011
			RR	0.0098	0.0096	0.010	0.014	0.010	0.018	0.011
UNI2	DL	D-PSO	Rand	0.01	0.012	0.013	0.019	0.02	0.015	0.025
			RR	0.0094	0.011	0.014	0.016	0.018	0.013	0.02

Tablo 3.11. Fil akış tespiti sonrası yapılan yönlendirmenin Verimlilik (Mbit/sec) değerlerinin karşılaştırılması (Fil akış (%20) ve Fare akış (%80))

Veri Seti	Fil Akış Tespiti	Fil Akış Yönlendirme	Fare Akış Yönlendirme	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8
X	Sabit Eşik	ECMP [74]	Rand	1.191.836	973.333	973.333	1.061.818	730.000	648.888	687.058
X	Sabit Eşik	Hedera [76]	Rand	1.061.818	2.013.793	1.061.818	648.888	778.666	648.888	1.061.818
UNI1	Karar	D-PSO	Rand	1.478.481	1.327.272	10.618.181	6.488.888	7.300.000	2.290.196	3.767.741
	Ağaçları		RR	1.374.117	1.460.000	11.680.000	7.300.000	4.492.307	2.920.000	4.171.428
UNI2	Karar	D-PSO	Rand	6.488.888	4.672.000	4.171.428	6.488.888	1.853.968	8.342.857	1.478.481
	Ağaçları		RR	3.767.741	1.424.390	5.309.090	6.488.888	8.342.857	1.374.117	8.342.857
UNI1	DVM	D-PSO	Rand	11.680.000	8.984.615	8.342.857	2.848.780	1.717.647	5.840.000	1.255.913
			RR	1.242.553	2.085.714	2.013.793	4.027.586	2.336.000	2.433.333	1.497.435
UNI2	DVM	D-PSO	Rand	3.244.444	9.733.333	2.433.333	1.441.975	8.984.615	2.485.106	8.984.615
			RR	5.309.090	11.680.000	2.848.780	5.561.904	10.618.181	2.994.871	9.733.333
UNI1	DL	D-PSO	Rand	12.835.164	8.342.857	10.618.181	6.488.888	8.984.615	11.680.000	10.618.181
			RR	11.918.367	12.166.666	11.680.000	8.342.857	11.680.000	6.488.888	10.618.181
UNI2	DL	D-PSO	Rand	11.680.000	9.733.333	8.984.615	6.147.368	5.840.000	778.666	4.672.000
			RR	12.425.531	10.618.181	8.342.857	7.300.000	6.488.888	8.984.615	5.840.000

Tablo 3.12. Fil akış tespiti sonrası yapılan yönlendirmenin Arabellek Boyutu (KB) değerlerinin karşılaştırılması (Fil akış (%20) ve Fare akış (%80))

Veri Seti	Fil Akış Tespiti	Fil Akış Yönlendirme	Fare Akış Yönlendirme	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8
X	Sabit Eşik	ECMP [74]	Rand	1.2	1.5	1.5	1.3	2.0	2.2	2.1
X	Sabit Eşik	Hedera [76]	Rand	1.3	0.7	1.3	2.2	1.8	2.2	1.3
UNI1	Karar	D-PSO	Rand	1.0	1.1	0.1	0.2	0.2	0.6	0.4
	Ağaçları		RR	1.0	1.0	0.1	0.2	0.3	0.5	0.3
UNI2	Karar	D-PSO	Rand	0.2	0.3	0.3	0.2	0.8	0.2	1.0
	Ağaçları		RR	0.4	1.0	0.3	0.2	0.2	1.0	0.2
UNI1	DVM	D-PSO	Rand	0.1	0.2	0.2	0.5	0.8	0.2	1.1
			RR	1.1	0.7	0.7	0.4	0.6	0.6	1.0
UNI2	DVM	D-PSO	Rand	0.4	0.1	0.6	1.0	0.2	0.6	0.2
			RR	0.3	0.1	0.5	0.3	0.1	0.5	0.1
UNI1	DL	D-PSO	Rand	0.1	0.2	0.1	0.2	0.2	0.1	0.1
			RR	0.1	0.1	0.1	0.2	0.1	0.2	0.1
UNI2	DL	D-PSO	Rand	0.1	0.1	0.2	0.2	0.2	0.2	0.3
			RR	0.1	0.1	0.2	0.2	0.2	0.2	0.2

Tablo 3.13. Fil akış tespiti sonrası yapılan yönlendirmenin RTT değerlerinin karşılaştırılması (Fil akış (%50) ve Fare akış (%50))

Veri Seti	Fil Akış Tespiti	Fil Akış Yönlendirme	Fare Akış Yönlendirme	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8
X	Sabit Eşik	ECMP [74]	Rand	0.075	0.11	0.12	0.35	0.2	0.15	0.18
X	Sabit Eşik	Hedera [76]	Rand	0.08	0.06	0.19	0.19	0.18	0.29	0.14
UNI1	Karar	D-PSO	Rand	0.047	0.06	0.065	0.123	0.14	0.119	0.147
	Ağaçları		RR	0.04	0.01	0.06	0.13	0.18	0.126	0.125
UNI2	Karar	D-PSO	Rand	0.075	0.088	0.127	0.143	0.139	0.125	0.23
	Ağaçları		RR	0.09	0.07	0.08	0.1	0.12	0.12	0.2
UNI1	DVM	D-PSO	Rand	0.007	0.054	0.11	0.132	0.129	0.25	0.16
			RR	0.086	0.045	0.019	0.12	0.128	0.13	0.15
UNI2	DVM	D-PSO	Rand	0.073	0.065	0.097	0.14	0.18	0.15	0.29
			RR	0.051	0.062	0.09	0.14	0.128	0.117	0.17
UNI1	DL	D-PSO	Rand	0.083	0.05	0.13	0.15	0.16	0.22	0.15
			RR	0.072	0.043	0.012	0.113	0.1	0.14	0.142
UNI2	DL	D-PSO	Rand	0.01	0.05	0.03	0.7	0.3	0.18	0.22
			RR	0.083	0.035	0.084	0.087	0.12	0.109	0.119

Tablo 3.14. Fil akış tespiti sonrası yapılan yönlendirmenin Verimlilik (Mbit/sec) değerlerinin karşılaştırılması (Fil akış (%50) ve Fare akış (%50))

Veri Seti	Fil Akış Tespiti	Fil Akış Yönlendirme	Fare Akış Yönlendirme	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8
X	Sabit Eşik	ECMP [74]	Rand	1.557.333	1.061.818	973.333	333.714	584.000	778.666	648.888
X	Sabit Eşik	Hedera [76]	Rand	1.460.000	1.946.666	614.736	614.736	648.888	402.758	834.285
UNI1	Karar Ağaçları	D-PSO	Rand	2.485.106	1.946.666	1.796.923	953.469	834.285	981.512	794.557
			RR	2.920.000	11.680.000	1.946.666	898.461	648.888	926.984	934.400
UNI2	Karar Ağaçları	D-PSO	Rand	1.557.333	1.327.272	919.685	816.783	840.287	934.400	507.826
			RR	1.297.777	1.668.571	1.460.000	1.168.000	973.333	973.333	584.000
UNI1	DVM	D-PSO	Rand	16.685.714	2.162.962	1.061.818	884.848	905.426	467.200	730.000
			RR	1.358.139	2.595.555	6.147.368	973.333	912.500	898.461	778.666
UNI2	DVM	D-PSO	Rand	1.600.000	1.799.691	1.199.178	834.285	648.888	778.666	402.758
			RR	2.290.196	1.883.870	1.297.777	834.285	912.500	998.290	687.058
UNI1	DL	D-PSO	Rand	1.407.228	2.336.000	898.461	778.666	730.000	530.909	778.666
			RR	16.290.097	27.162.790	9.733.333	1.033.628	1.168.000	834.285	822.535
UNI2	DL	D-PSO	Rand	11.680.000	2.336.000	3.893.333	166.857	389.333	648.888	530.909
			RR	1.405.535	3.318.181	1.390.476	1.340.987	973.333	1.071.559	981.512

Tablo 3.15. Fil akış tespiti sonrası yapılan yönlendirmenin Arabellek Boyutu (KB) değerlerinin karşılaştırılması (Fil akış (%50) ve Fare akış (%50))

Veri Seti	Fil Akış Tespiti	Fil Akış Yönlendirme	Fare Akış Yönlendirme	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8
X	Sabit Eşik	ECMP [74]	Rand	0.9	1.3	1.5	4.3	2.4	1.8	2.2
X	Sabit Eşik	Hedera [76]	Rand	1	0.7	2.3	2.3	2.2	3.5	1.7
UNI1	Karar	D-PSO	Rand	0.6	0.7	0.8	1.5	1.7	1.5	1.8
	Ağaçları		RR	0.5	0.1	0.7	1.6	2.2	1.5	1.5
UNI2	Karar	D-PSO	Rand	0.9	1.1	1.6	1.7	1.7	1.5	2.8
	Ağaçları		RR	1.1	0.9	1	1.2	1.5	1.5	2.4
UNI1	DVM	D-PSO	Rand	0.1	0.7	1.3	1.6	1.6	3.1	2
			RR	1	0.5	0.2	1.5	1.6	1.6	1.8
UNI2	DVM	D-PSO	Rand	0.9	0.8	1.2	1.7	2.2	1.8	3.5
			RR	0.6	0.8	1.1	1.7	1.6	1.4	2.1
UNI1	DL	D-PSO	Rand	1	0.6	1.6	1.8	2	2.7	1.8
			RR	0.9	0.5	0.1	1.4	1.2	1.7	1.7
UNI2	DL	D-PSO	Rand	0.1	0.6	0.4	8.5	3.7	2.2	2.7
			RR	1	0.4	1	1.1	1.5	1.3	1.5

Oluřturulan iki farklı ađ trafiđi karřılařtırıldıđında ise fil akıř oranının %20 olduđu birinci trafikte RTT, verimlilik ve arabellek boyutu deđerlerinin daha bařarılı olduđu g r lmektedir. Oluřturulan birinci ve ikinci trafik i in DL Y ntemi ile fil akıř tespiti yapılan ve fare akıřların RR ile y nlendirildiđinde elde edilen deđerlerin karřılařtırılması Tablo 3.16. ile verilmektedir.

Ayrıca fil akıř tespitinin y nlendirmeye etkisi ise Tablo 3.17. ile a ık a g r lmektedir. Tablo 3.17.'de verilen fil akıř tespiti uygulanmıř sonu larda, fil akıřların sayısı ve fare akıřların sayısı eřittir. Yani fil akıřların toplam akıřa oranı %50 olarak alınarak en zorlu kořulda bile y ntemin etkisi vurgulanmak istenmiřtir.



Tablo 3.16. Oluşturulan birinci ve ikinci trafik için elde edilen en başarılı yöntemlerin kıyaslanması (Derin Öğrenme ile fil akış tespiti, Fil akışların yönlendirilmesi D-PSO, Fare akışların yönlendirilmesi RR)

Trafik	Değerlendirme Metriği	Veri Seti	H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8
Fil Akış (%20)	RTT	UNI1	0.0098	0.0096	0.010	0.014	0.010	0.018	0.011
		UNI2	0.0094	0.011	0.014	0.016	0.018	0.013	0.02
Fil Akış (%50)		UNI1	0.072	0.043	0.012	0.113	0.1	0.14	0.142
		UNI2	0.083	0.035	0.084	0.087	0.12	0.109	0.119
Fil Akış (%20)	Verimlilik	UNI1	11.918.367	12.166.666	11.680.000	8.342.857	11.680.000	6.488.888	10.618.181
		UNI2	12.425.531	10.618.181	8.342.857	7.300.000	6.488.888	8.984.615	5.840.000
Fil Akış (%50)		UNI1	16.290.097	27.162.790	9.733.333	1.033.628	1.168.000	834.285	822.535
		UNI2	1.405.535	3.318.181	1.390.476	1.340.987	973.333	1.071.559	981.512
Fil Akış (%20)	Arabellek	UNI1	0.1	0.1	0.1	0.2	0.1	0.2	0.1
		UNI2	0.1	0.1	0.2	0.2	0.2	0.2	0.2
Fil Akış (%50)		UNI1	0.9	0.5	0.1	1.4	1.2	1.7	1.7
		UNI2	1	0.4	1	1.1	1.5	1.3	1.5

Tablo 3.17. Fil akış tespiti için yönlendirmeye etkisi

Fil Akış Tespiti	Metrik	Veri Seti	Yönlendirme		H1-H2	H1-H3	H1-H4	H1-H5	H1-H6	H1-H7	H1-H8
			Fil Akış	Fare Akış							
X		X		D-PSO	0.097	0.13	0.16	0.17	0.21	0.22	0.25
DL	RTT	UNI1	D-PSO	RR	0.072	0.043	0.012	0.113	0.1	0.14	0.142
DL		UNI2		RR	0.083	0.035	0.084	0.087	0.12	0.109	0.119
X		X		D-PSO	1.204.123	898.461	730.000	687.058	556.190	530.909	467.200
DL	Verimlilik	UNI1	D-PSO	RR	16.290.097	27.162.790	9.733.333	1.033.628	1.168.000	834.285	822.535
DL		UNI2		RR	1.405.535	3.318.181	1.390.476	1.340.987	973.333	1.071.559	981.512
X		X		D-PSO	1.2	1.6	2	2.1	2.6	2.7	3.1
DL	Arabellek	UNI1	D-PSO	RR	0.9	0.5	0.1	1.4	1.2	1.7	1.7
DL	Boyutu	UNI2		RR	1	0.4	1	1.1	1.5	1.3	1.5

Önerilen çalışmada, farklı sınıflandırıcıların ve farklı yönlendirme stratejilerinin kombinasyonları denenerek geniş bir analiz yapılmıştır. Yapılan bu analiz, gelecekteki çalışmalar üzerinde önemli bir etkiye sahip olacaktır. Tablolar incelendiğinde, statik bir eşik değeri kullanılarak fil akışı tespiti ile yönlendirmenin, sınıflandırıcılar ile fil akışı tespiti ile yönlendirmeye göre daha başarısız olduğu görülmektedir. Ayrıca fare akışlar ve fil akışlar için ayrı yönlendirme stratejileri kullanarak yönlendirmenin sadece ağın kapasitesinin verimli kullanılmasını sağlamakla kalmayıp, daha az bellek gereksinimi ve daha az gecikme ile yönlendirme sağladığı görülmektedir.

DL Yöntemi ile fil akış tespiti yapılan ve fil akışlarının D-PSO ile fare akışlarının ise RR ile yönlendirilmesinin farklı trafik senaryoları için sonuçları, SCI alan indeksli bir dergide yayınlanmıştır [119].

4. SONUÇLAR VE TARTIŞMA

YTA tabanlı veri merkezlerinin sağladığı yüksek bant genişliğini verimli bir şekilde kullanmak için trafik yönetiminin önemi bilinmektedir. Son yıllardaki çalışmalarda trafiği etkin şekilde yönetmek için yük dengeleme konusu ele alınmaktadır. YTA'da veri düzleminde uygulanabilen iki farklı yük dengeleme mevcuttur. Bunlar sunucu tabanlı yük dengeleme ve bağlantı tabanlı yük dengelemedir. Tez çalışması kapsamında belirtilen iki yük dengeleme konusu ile ilgili çalışmalar yapılmıştır. Ayrıca fil akışı tespiti için trafik sınıflandırmaya odaklanılmıştır. Çünkü, çok yollu veri merkezleri için trafik yönetimindeki ciddi bir sorun, büyük miktarda veri taşıyan fil akışlarının zamanında tespit edilememesidir. Fil akışlarının tespiti kadar planlaması da önemlidir. Bu nedenle tez çalışmasında ek olarak fil akışların yönlendirilmesi konusu da ele alınmaktadır. YTA uygulamaları için bir diğer önemli nokta ise uygulama ortamının hazırlanmasıdır. Tez çalışması kapsamında önerilen her bir yeni yaklaşımda ağ topolojisini oluşturmak için Mininet emülatörü ve YTA denetleyici olarak Python programlama dilini kullanan POX tercih edilmiştir.

Tez çalışmasında sunucu seçimi için önerilen yük dengeleme stratejisini benzerlerinden farklı kılan özelliği özgün olarak tasarlanan karar fonksiyonu ve puanlama mantığıdır. Karar fonksiyonu, web isteklerine mümkün olan en kısa sürede yanıt verecek şekilde tasarlanmıştır. Bu fonksiyon dört farklı simülasyon ortamında test edilmiş ve böylece farklı ağ ortamları için önerilen yöntemin sonuçları gözlemlenmiştir. Önerilen yöntem geleneksel yöntemlerle karşılaştırılarak somut sonuçlar elde edilmiştir. Önerilen yük dengeleme stratejisinin aktarım hızını ve saniyedeki istek sayısını artırdığı, test süresini azalttığı simülasyon sonuçlarıyla kanıtlanmıştır. Karşılaştırma sonuçlarına bakıldığında önerilen yöntemin yük dengeleme yöntemlerinin temel amaçlarına ulaşmada diğer yöntemlere göre daha üstün olduğu görülmektedir.

YTA'da bağlantı tabanlı yük dengeleme gerçekleştirilirken amaç en uygun rotayı bulmak ve verileri en hızlı şekilde iletmektir. D-PSO, hızlı yakınsaması ve düşük hesaplama yükü nedeniyle yük dengeleme amaçlarına ulaşmada diğer meta-sezgisel algoritmalarından çok daha uygundur. Bu amaçla D-PSO, probleme uygun olacak şekilde güncellenmiş ve bu alanda ilk kez kullanılmıştır. Yaklaşım uygulanırken, bağlantılardan ve anahtarlardan RTT, BW, PLC, PN gibi istatistikler periyodik olarak alınarak özgün bir hibrit maliyet fonksiyonu tasarlanmıştır. Literatürden farklı olarak, sadece bağlantılar üzerinde yük dengeleme

yapılmamış, eşzamanlı olarak anahtarlar ve bağlantılar üzerinde yük dengeleme yapılmıştır. Veri iletimi sırasında anahtarlar da en az bağlantılar kadar önemlidir. İstatistiklerin periyodik olarak güncellenmesi uygulanan yük dengeleme yönteminin dinamik olmasını sağlamaktadır. Önerilen yöntemi diğer yöntemlerle karşılaştırmak için literatürde sıklıkla kullanılan dört metrik seçilmiştir. Bu metrikler RTT, PLC, arabellek boyutu ve verimliliklerdir. Bu çalışmada kullanılan metrikler, yük dengeleme yapılırken ağ tıkanıklığı, veri aktarım hızı, kullanılan bellek miktarı ve veri aktarımı için gereken güç miktarının hesaplanmasına yardımcı olur. Böylece yöntemlerin yük dengeleme başarısı ortaya çıkar. Bu metriklerle önerilen yöntem, RR, ACO ve G-ACO algoritmaları ile karşılaştırılmıştır. Sonuçlar incelendiğinde önerilen yöntemin yük dengelemede bu yöntemlere göre daha başarılı olduğu görülmektedir. Ayrıca verilen sonuçlarda, bağlantı ve anahtar yük dengelemesinin aynı anda etkin bir şekilde gerçekleştirildiği kanıtlanmaktadır.

Gelen akışın türünü belirlemek, dengeli bir dağıtım için en kritik adımlardan biridir. Bu nedenle tez çalışmasında, fil akışların ve fare akışlarının sınıflandırılması konusunda çalışmalar yapılmıştır. Sınıflandırma yöntemleri, probleme yönelik en uygun modeli oluşturabilmek amacıyla eğitim ve test veri setlerine ihtiyaç duymaktadır. UNI1 veri seti ve UNI2 veri seti, üniversite kampüs ağının izlenmesi sonucu elde edilen veri setleridir ve bu çalışmada kullanılmaktadır. Veri setleri paketlerden oluştuğu için öncelikle FlowRecorder kullanılarak akışlara dönüştürülür. Bu yeni veri kümesi etiket değerleri içermemektedir. Bu nedenle, veri kümelerindeki her bir akışı başlangıçta fare veya fil olarak etiketlemeye ihtiyaç vardır. Araştırmacılar genellikle bu işlem için statik bir eşik değeri kullanmayı tercih etmektedir. Literatürden farklı olarak, tez kapsamında önerilen algılayıcıda, değişen ağ durumlarına hızlı bir şekilde uyum sağlayabilen Otsu eşikleme yaklaşımına dayalı dinamik bir eşik kullanılmaktadır. Otsu eşikleme yöntemi, UNI1 veri setindeki ve UNI2 veri setindeki paket boyutu özelliğini kullanmaktadır. Sınıflandırmaya hazır hale getirilen veri setleri, 7 farklı sınıflandırıcı ile test edilmiştir. Bunlar; Lojistik Regresyon, DVM, Karar Ağaçları, Rastgele Ormanlar, Naive Bayes, KNN ve Derin Öğrenme yöntemleridir. Araştırmacılar, fil ve fare akışlarını sınıflandırırken akışların yalnızca paket boyutu özelliğini kullanır. Kolay sınıflandırılan veri kümeleri için tek bir özellik ile sınıflandırma yeterli görünse de karmaşık veri kümeleri için yeterli olmayacaktır. Literatürdeki bu boşluğu doldurmak için çalışmada önerilen algılayıcıda, her akış için 11 özellik kullanılarak benzersiz ve ölçeklenebilir bir sınıflandırma işlemi gerçekleştirilir. Hesaplanan akış özellikleri; akışın minimum paket boyutu, akışın minimum paket varış zamanı, akıştan

aktarılan bayt sayısı, akışın ortalama paket boyutu, akışın ortalama paket varış zamanı, akış için iletilen paketlerin sayısı, akış paket boyutlarının standart sapması, akış varış zamanlarının standart sapması, saniye cinsinden akış süresi, akışın maksimum paket boyutu ve akışın maksimum varış zamanıdır. Önerilen algılayıcı 7 farklı sınıflandırıcıya uygulanarak literatürde bulunan farklı yöntemlerle kıyaslanmıştır. DVM, Karar Ağaçları, Rastgele Ormanlar ve Derin Öğrenme yöntemlerinin doğruluğu vurgulayan problemler için uygun olduğu, buna karşılık, Lojistik Regresyon, DVM ve Karar Ağaçları yöntemlerinin, zaman ölçüsünü vurgulayan problemler için ideal olduğu kanıtlanmıştır. YTA'daki en iyi sınıflandırıcı, mümkün olan en kısa sürede maksimum doğruluğu sağlayan sınıflandırıcıdır.

Son olarak fil akış tespiti yapılan akışların yönlendirilmesi üzerine çalışılmıştır. Amaç fil akışların tespitinin yönlendirme başarısını artırdığını kanıtlamaktır. Bu amaç için fil akış tespiti için başarılı sayılabilecek 3 yöntem denenmiştir. Bunlar; Karar Ağaçları, DVM ve Derin Öğrenme yöntemleridir. Bu 3 yönteme göre sınıflandırıcı modeli oluşturulduktan sonra, akış türüne göre yönlendirme yapılmaktadır. Fil akışların verimli yönlendirilmesi fare akışların verimli yönlendirilmesinden daha önemlidir. Bu nedenle önerilen çalışmada, anahtar ve bağlantılardaki yükü eş zamanlı olarak dengeleyebilen D-PSO yöntemi fil akışların yönlendirilmesinde kullanılmaktadır. Fare akışlar ise daha basit bir yapıya sahip olan geleneksel RR ve Rand yöntemleri ile yönlendirilmektedir. Fil akış tespiti uygulanmadan yapılan yönlendirme ile UNI1 veri setinde ve UNI2 veri setinde fil akış tespiti sonrası yapılan yönlendirmenin etkileri 3 metrik kullanılarak karşılaştırılmıştır. Kullanılan karşılaştırma metrikleri RTT, verimlilik ve arabellek boyutudur. Fil akış tespiti uygulanmadan yapılan yönlendirmelerde D-PSO yöntemi en başarılı olan yöntemdir. Fakat tüm sonuçlar değerlendirildiği zaman, fil akış tespiti uygulanan çalışmalardaki metrik değerlerinin büyük çoğunluğu fil akış tespiti uygulanmayan yöntemlerden daha başarılıdır. Fil akış tespiti uygulanan çalışmalardaki metrik değerlerinin karşılaştırma sonuçları incelendiğinde ise UNI1 veri seti ve UNI2 veri seti için çoğunlukla Derin Öğrenme Yöntemi ile fil akış tespiti yapılan ve fare akışların RR ile yönlendirildiği uygulamalar başarılı sonuçlar üretmektedir.

5. ÖNERİLER

Tez çalışması kapsamında, Yazılım Tanımlı Ağ tabanlı veri merkezlerinde yük dengeleme konusuna yönelik farklı yaklaşımlar önerilmiştir. Önerilen yaklaşımların içerisinde veri düzleminde uygulanan sunucu tabanlı ve bağlantı tabanlı yük dengeleme yöntemleri mevcuttur. Ayrıca çalışmaya yeni bir yön vermesi açısından akış sınıflandırma konusu incelenmiş ve fil akış tespiti uygulanmıştır. Fil akış tespitinde farklı sınıflandırıcı yöntemleri denenmiş ve başarısını kanıtlayan sınıflandırıcılar fil akış tespitinde kullanılmıştır. Bu yaklaşımla amacımız, akışların türlerine göre yönlendirilmesini sağlamaktır.

Tez çalışmasında, akış sınıflandırma yapılırken akışlar iki adete ayrılmaktadır; fil akışlar ve fare akışlar. Akış sınıflandırma için akış özellikleri artırılarak akışlar daha fazla sayıda sınıfa ayrılabilir. Böylece her akış için sunulan yönlendirme stratejisi sayısı artacak ve akışlar daha detaylı analiz imkânı bulacaktır.

Çalışmada 7 farklı sınıflandırıcı test edilmiş ve başarıları değerlendirilmiştir. Farklı sınıflandırıcılar ile çalışma alanı genişletilebilir. Önerilen CNN mimarisinin başarısı açıktır fakat farklı mimariler de denenerek test edilebilir.

Literatürde sıklıkla kullanılan UNI1 veri seti ve UNI2 veri seti çalışmalarda kullanılmıştır. Gerçek ağ izlerinden oluşan veri setleri oluşturularak açık kaynaklı paylaşımları yapılabilir.

Tez çalışması kapsamında merkezi mimariye sahip POX denetleyicisi kullanılmaktadır. Yöntemlerin başarısı başka denetleyicilerle ve dağıtık mimarilerle test edilebilir.

6. KAYNAKLAR

1. Chen, T., Matinmikko, M., Chen, X., Zhou, X. ve Ahokangas, P., Software defined mobile networks: concept, survey, and research directions, IEEE Communications Magazine, 53,11 (2015) 126- 133.
2. Caraguay, Á., L., V., Peral, A., B., López, L., I., B. ve Villalba, L., J., G., SDN: Evolution and Opportunities in the Development IoT Applications, International Journal of Distributed Sensor Networks, 10,5 (2014) 1-10.
3. Xia, W., Wen, Y., Foh, C., H., Niyato, D. ve Xie, H., A Survey on Software-Defined Networking, IEEE Communication Surveys & Tutorials, 17,1 (2015) 27-51.
4. Kreutz, B., D., Ramos, F., M., V., Verissimo, P., E., Rothenberg, C., E., Azodolmolky, S., ve Uhlig, S., Software-Defined Networking: A Comprehensive Survey, Proceedings of the IEEE, 103,1 (2015) 14-76.
5. Göransson, P. ve Black, C., Software Defined Networks A Comprehensive Approach, 1. Baskı, Elsevier, USA, 2014.
6. Alparslan, Ö., Veri Merkezlerinde Büyük Boyutlu Verilerin Taşınması Sürecinde Yazılım Tanımlı Ağların (SDN) Kullanımı, Yüksek Lisans Tezi, Gazi Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 2016.
7. Farris, I., Taleb, T., Khettab, Y. ve Song, J., A Survey on Emerging SDN and NFV Security Mechanisms for IoT Systems, IEEE Communications Surveys & Tutorials, 21,1 (2019) 812-837.
8. Hou, X., Ren, Z., Wang, J., Cheng, W., Ren, Y., Chen, K., C. ve Zhang, H., Reliable Computation Offloading for Edge-Computing-Enabled Software-Defined IoV, IEEE Internet of Things Journal, 7,8 (2020) 7097-7111.
9. Kadhim, A., J. ve Hosseini Seno, S., A., Maximizing the Utilization of Fog Computing in Internet of Vehicle Using SDN, IEEE Communications Letters, 23,1 (2019) 140-143.
10. Ejaz, S., Iqbal, Z., Azmat Shah, P., Bukhari, B., H., Ali, A. ve Aadil, F., Traffic Load Balancing Using Software Defined Networking (SDN) Controller as Virtualized Network Function, IEEE Access, 7 (2019) 46646-46658.
11. Wang, X., Deng, Q., Ren, J., Malboubi, M., Wang, S., Xu, S. ve Chuah, C., N., The Joint Optimization of Online Traffic Matrix Measurement and Traffic Engineering for Software-Defined Networks, IEEE/ACM Transactions on Networking, 28,1 (2020) 234-247.
12. Chica, J., C., C., Imbach, J., C. ve Vega, J., F., B., Security in SDN: A comprehensive survey, Journal of Network and Computer Applications, 159 (2020) 1-23.

13. Kharkongor, C., Chithralekha, T. ve Varghese, R., A SDN Controller with Energy Efficient Routing in the Internet of Things (IoT), Procedia Computer Science, 89 (2016) 218-227.
14. Tomovic, S., Yoshigoe, K., Maljevic, I. ve Radusinovic, I., Software-Defined Fog Network Architecture for IoT, Wireless Personal Communications, 92 (2017) 181–196.
15. Chen, S., Zhang, T. ve Shi, W., Fog Computing, IEEE Internet Computing, 21,2 (2017) 4-6.
16. Chang, C., Srirama S., N. ve Buyya R., 2017, Indie Fog: An Efficient Fog-Computing Infrastructure for the Internet of Things, Computer, 50,9 (2017) 92-98.
17. Bittencourt, L., F., Diaz-Montes, J., Buyya, R., Rana, O., F. ve Parashar, M., Mobility-Aware Application Scheduling in Fog Computing, IEEE Cloud Computing, 4,2 (2017) 26-35.
18. Sookhak, M., Yu, F., R., He, Y., Talebian, H., Safa, N., S., Zhao, N., Khan, M., K. ve Kumar N., Fog Vehicular Computing Augmentation of Fog Computing Using Vehicular Cloud Computing, IEEE Vehicular Technology Magazine, 12,3 (2017) 55-64.
19. Hou, X., Li, Y., Chen, M., Wu, D., Jin, D. ve Chen, S., Vehicular Fog Computing: A Viewpoint of Vehicles as the Infrastructures, IEEE Transactions on Vehicular Technology, 65,6 (2016) 3860-3873.
20. Jalali, F., Khodadustan, S., Gray, C., Hinton, K. ve Suits, F., Greening IoT with Fog: A Survey, IEEE 1st International Conference on Edge Computing, Haziran 2017, Honolulu, HI, USA, Bildiriler Kitabı, 25-31.
21. Bizanis, N. ve Kuipers, F., A., SDN and Virtualization Solutions for the Internet of Things: A Survey, IEEE Access, 4, (2016) 5591-5606.
22. Sherwood, R., Gibb, G., Yap K., K., Casado, M., Mckeown, N. ve Parulkar, G., Flowvisor: A network virtualization layer, OpenFlow Switch Consortium, Tech. Rep. 2009, 1–13.
23. Drutskoy, D., Keller, E. ve Rexford, J., Scalable network virtualization in software-defined networks, IEEE Internet Computing, 17,2 (2013) 20–27.
24. Blenk, A., Basta, A., Reisslein, M. ve Kellerer, W., Survey on Network Virtualization Hypervisors for Software Defined Networking, IEEE Communications Surveys & Tutorials, 18,1 (2016) 655-685.
25. Akyildiz, I., F., Lee, A., Wang, P., Luo, M. ve Chou, W., A roadmap for traffic engineering in SDN-OpenFlow networks, Computer Networks, 71 (2014), 1–30.
26. Akyildiz, I., F., Lee, A., Wang, P., Luo, M. ve Chou, W., Research challenges for traffic engineering in software defined networks, IEEE Network, 30,3 (2016) 52-58.

27. Shu, Z., Wan, J., Lin, J., Wang, S., Li, D., Rho, S. ve Yang, C., Traffic Engineering in Software-Defined Networking: Measurement and Management, IEEE Access, 4 (2016) 3246-3256.
28. Yassine, A., Rahimi, H. ve Shirmohammadi, S., Software Defined Network Traffic Measurement: Current Trends and Challenges, IEEE Instrumentation & Measurement Magazine, 18,2 (2015) 42-50.
29. Polverini, M., Baiocchi, A., Cianfrani, A., Iacovazzi, A. ve Listanti, M., The Power of SDN to Improve the Estimation of the ISP Traffic Matrix Through the Flow Spread Concept, IEEE Journal on Selected Areas in Communications, 34,6 (2016) 1904-1913.
30. Belgaum, M., R., Musa, S., Alam, M., M. ve Su'ud, M., M., A Systematic Review of Load Balancing Techniques in Software-Defined Networking, IEEE Access, 8 (2020) 98612- 98636.
31. Li, L. ve Xu, Q., Load balancing researches in SDN: A survey, 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC), Temmuz 2017, Macau, China, Bildiriler Kitabı, 403-408.
32. Yao, L., Hong, P., Zhang, W., Li, J. ve Ni, D., Controller Placement and Flow based Dynamic Management Problem towards SDN, IEEE International Conference on Communication Workshop (ICCW), Haziran 2015, London, UK, Bildiler Kitabı, 363-368.
33. Wang, C., Hu, B., Chen, S., Li, D. ve Liu, B., A Switch Migration-Based Decision-Making Scheme for Balancing Load in SDN, IEEE Access, 5 (2017) 4537-4544.
34. Sufiev, H. ve Haddad, Y., A Dynamic Load Balancing Architecture for SDN, IEEE International Conference on the Science of Electrical Engineering (ICSEE), Kasım 2016, Eilat, Israel, Bildiler Kitabı, 1-3.
35. Heller, B., Sherwood, R. ve McKeown, N., The Controller Placement Problem, HotSDN '12: Proceedings of the first workshop on Hot topics in software defined networks, Ağustos 2012, Helsinki, Finland, Bildiriler Kitabı, 7-12.
36. Selvi, H., Güner, S., Gür, G. ve Alagöz, F., The Controller Placement Problem in Software Defined Mobile Networks (SDMN), M. Liyanage, A. Gurtov ve Mika Ylianttila (Eds.), 129-147, Software defined mobile networks (SDMN): beyond LTE network architecture, Wiley, Chichester, 2015.
37. Gajjar, D., Kotak, H., H., Joshi, D., Mithbavkar ve Perigo, L., Round Robin Load Balancer using Software Defined Networking (SDN), Capstone Team Research Project, 1-9, 2016.
38. Kaur, S., Singh, J., Kumar, K. ve Ghumman, N., S., Round-Robin Based Load Balancing in Software Defined Networking, 2nd International Conference on Computing for Sustainable Global Development (INDIACom), Mart 2015, New Delhi, India, Bildiler Kitabı, 2136-2139.

39. Sabiya ve Singh, J., Weighted Round-Robin Load Balancing Using Software Defined Networking, International Journal of Advanced Research in Computer Science and Software Engineering, 6,6 (2016) 621-625.
40. Prakash, S., W. ve Deepalakshmi, P., Server-based Dynamic Load Balancing, International Conference on Networks & Advances in Computational Technologies (NetACT), Temmuz 2017, Thiruvananthapuram, India, Bildiler Kitabı, 25-28.
41. Ikram, A., Arif, S., Ayub, N. ve Arif, W., Load Balancing In Software Defined Networking (SDN), MAGNT Research Report, 5,1 (2018) 298-305.
42. Deepika, Wadhwa, D. ve Kumar, N., Performance Analysis of Load Balancing Algorithms in Distributed System, Advance in Electronic and Electric Engineering, 4,1 (2014) 59-66.
43. Kaur, H. ve Jyoti, N., Traffic Based Load Balancing in Software Defined Networking, International Journal on Computer Science and Engineering (IJCSE), 9,6 (2017) 379-384.
44. Kaur, K., Kaur, S. ve Gupta, V., Flow Statistics Based Load Balancing in OpenFlow, International Conference on Advances in Computing, Communications and Informatics (ICACCI), Eylül 2016, Jaipur, India, Bildiler Kitabı, 378-381.
45. Hamed, M., I., ElHalawany, B., M., Fouda, M., M. ve Eldien, A., S., T., A New Approach for Server-based Load Balancing Using Software-Defined Networking, Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), Aralık 2017, Cairo, Egypt, Bildiler Kitabı, 30-35.
46. Soleimanzadeh, K., Ahmadi, M. ve Nassiri, M., SD-WLB: An SDN-aided mechanism for web load balancing based on server statistics, ETRI Journal, 41,2 (2019) 197-206.
47. Kumari, P. ve Thakur, D., Load Balancing in Software Defined Network, International Journal of Computer Sciences and Engineering, 5,12 (2017) 227-232.
48. Hamdan, M., Hassan, E., Abdelaziz, A., Elhigazi, A., Mohammed, B., Khan, S., Vasilakos, A., V. ve Marsono, M., N., A Comprehensive Survey of Load Balancing Techniques in Software-Defined Network, Journal of Network and Computer Applications, 174 (2021) 102856.
49. Chen-Xiao, C. ve Ya-Bin, X., Research on load balance method in SDN, International Journal of Grid and Distributed Computing, 9,1 (2016) 25-36.
50. Wang, Y. ve You, S., An Efficient Route Management Framework for Load Balance and Overhead Reduction in SDN-Based Data Center Networks, IEEE Transactions on Network and Service Management, 15,4 (2018) 1422-1434.
51. Wang, P., Xu, H., Huang, L., He, J. ve Meng, Z., Control Link Load Balancing and Low Delay Route Deployment for Software Defined Networks, IEEE Journal on Selected Areas in Communications, 35,11 (2017) 2446-2456.

52. Yang, X., Xu, H., Huang, L., Zhao, G., Xi, P. ve Qiao, C., Joint Virtual Switch Deployment and Routing for Load Balancing in SDNs, IEEE Journal on Selected Areas in Communications, 36,3 (2018) 397-410.
53. Xu, H., Yu, Z., Li, X., Huang, L., Qian, C. ve Jung, T., Joint Route Selection and Update Scheduling for Low-Latency Update in SDNs, IEEE/ACM Transactions on Networking, 25,5 (2017) 3073-3087.
54. Zhang, J., Ye, M., Guo, Z., Yen, C., Y. ve Chao, H., J., CFR-RL: Traffic Engineering with Reinforcement Learning in SDN, IEEE Journal on Selected Areas in Communications, 38,10 (2020) 2249-2259.
55. Akin, E. ve Korkmaz, T., Rate-Based Dynamic Shortest Path Algorithm for Efficiently Routing Multiple Flows in SDN, IEEE International Conference on Communications (ICC 2019), Mayıs 2019, Shanghai, China, Bildiler Kitabı, 1-7.
56. Xiaolong, X., Yun, C., Liuyun, H. ve Anup, K., MTSS: Multi-path traffic scheduling mechanism based on SDN, Journal of Systems Engineering and Electronics, 30,5 (2019) 974-984.
57. Xue, H., Kim, K., T. ve Youn, H., Y., Dynamic Load Balancing of Software-Defined Networking Based on Genetic-Ant Colony Optimization, Sensors, 19,2 (2019) 1-17.
58. Hamdan, M., Mohammed, B., Humayun, U., Abdelaziz, A., Khan, S., Akhtar Ali, M., Imran, M. ve Marsono, M., N., Flow-Aware Elephant Flow Detection for Software-Defined Networks, IEEE Access, 8 (2020) 72585-72597.
59. Che, L. ve Qiu, B., Landmark LRU: an efficient scheme for the detection of elephant flows at internet routers, IEEE Communications Letters, 10,7 (2006) 567-569.
60. Pham, M., T., Seow, K., T. ve Foh, C., H., Towards intelligent datacenter traffic management: Using automated fuzzy inferencing for elephant flow detection, International Journal of Innovative Computing, Information and Control, 10,5 (2014) 1669-1685.
61. Murua, J. ve Reviriego, P., Faking Elephant Flows on the Count Min Sketch, IEEE Networking Letters, 2,4 (2020) 199-202.
62. Torres, P., R., García-Martínez, A., Bagnulo, M. ve Ribeiro, E., P., An Elephant in the Room: Using Sampling for Detecting Heavy-Hitters in Programmable Switches, IEEE Access, 9 (2021) 94122-94131.
63. Gong, J., Yang, T., Zhang, H., Li, H., Uhlig, S., Chen, S., Uden, L. ve Li, X., HeavyKeeper: An Accurate Algorithm for Finding Top- k Elephant Flows, IEEE/ACM Transactions on Networking, 27,5 (2019) 1845-1858.
64. Liu, Z., Gao, D., Liu, Y., Zhang, H. ve Foh, C., H., An adaptive approach for elephant flow detection with the rapidly changing traffic in data center network, International Journal of Network Management, 27,6 (2017) 1-13.

65. Liu, P., Lv, N., Chen, K., Tang, L. ve Zhou, J., Regression Based Dynamic Elephant Flow Detection in Airborne Network, IEEE Access, 8 (2020) 217123-217133.
66. AlGhadhban, A., Celik, A., Shihada, B. ve Alouini, M., S., LightFDG: An Integrated Approach to Flow Detection and Grooming in Optical Wireless DCNs, IEEE Transactions on Network and Service Management, 17,2 (2020) 1153-1166.
67. Estrada-Solano, F., Caicedo, O., M. ve Da Fonseca, N., L., S., NELLY: Flow Detection Using Incremental Learning at the Server Side of SDN-Based Data Centers, IEEE Transactions on Industrial Informatics, 16,2 (2020) 1362-1372.
68. Liu, W., X., Cai, J., Wang, Y., Chen, Q., C. ve Zeng, J., Q., Fine-grained flow classification using deep learning for software defined data center networks, Journal of Network and Computer Applications, 168 (2020) 102766.
69. Tang, Q., Zhang, H., Dong, J. ve Zhang, L., Elephant Flow Detection Mechanism in SDN-Based Data Center Networks, Scientific Programming, 2020 (2020) 1-8.
70. Zhang, H., Tang, F. ve Barolli, L., Efficient flow detection and scheduling for SDN-based big data centers, Journal of Ambient Intelligence and Humanized Computing, 10 (2019) 1915–1926.
71. Hamdan, M., Khan, S., Abdelaziz, A., Sadih, S., Shaikh-Husin, N., Al Otaibi, S., Maple, C. ve Marsono, M., N., DPLBAnt: Improved load balancing technique based on detection and rerouting of elephant flows in software-defined networks, Computer Communications, 180 (2021) 315-327.
72. Tang, F., Zhang, H., Yang, L., T. ve Chen, L., Elephant Flow Detection and Load-Balanced Routing with Efficient Sampling and Classification, IEEE Transactions on Cloud Computing, 9,3 (2021) 1022-1036.
73. Yahyaoui, H., Aidi, S. ve Zhani, M., F., On Using Flow Classification to Optimize Traffic Routing in SDN Networks, IEEE 17th Annual Consumer Communications & Networking Conference (CCNC), Ocak 2020, Las Vegas, NV, USA, Bildiler Kitabı, 1-6.
74. Xi, K., Liu, Y. ve Chao, H., J., Enabling flow-based routing control in data center networks using Probe and ECMP, IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Nisan 2011, Shanghai, China, Bildiler Kitabı, 608-613.
75. Curtis, A., R., Kim, W. ve Yalagandula, P., Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection, IEEE INFOCOM, (2011) 1629-1637.
76. Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N. ve Vahdat, A., Hedera: Dynamic flow scheduling for data center networks, 7th USENIX conference on Networked systems design and implementation, Nisan 2010, San Jose, CA, USA, Bildiler Kitabı, 19.

77. Curtis, A., R., Mogul, J., C., Tourrilhes, J., Yalagandula, P., Sharma, P. ve Banerjee, S., DevoFlow: Scaling flow management for highperformance networks, ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'11), Ağustos 2011, Toronto, Ontario, Canada, Bildiriler Kitabı, 254-265.
78. Chao, S., C., Lin, K., C., J. ve Chen, M., S., Flow Classification for Software-Defined Data Centers Using Stream Mining, IEEE Transactions on Services Computing, 12,1 (2019) 105-116.
79. Tang, F., Li, L., Barolli, L. ve Tang, C., An Efficient Sampling and Classification Approach for Flow Detection in SDN-Based Big Data Centers, IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), Mart 2017, Taipei, Taiwan, Bildiler Kitabı, 1106-1115.
80. Farrington, N., Porter, G., Radhakrishnan, S., Bazzaz, H., H., Subramanya, V., Fainman, Y., Papen, G. ve Vahdat, A., Helios: A hybrid electrical/optical switch architecture for modular data centers, ACM SIGCOMM '10, 30 Ağustos-3 Eylül 2010, New Delhi, India, Bildiler Kitabı, 339–350.
81. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks, RFC 3176, 2001.
82. Nunes, B., A., A., Mendonca, M., Nguyen, X., N., Obraczka, K. ve Turletti, T., A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks, IEEE Communications Surveys & Tutorials, 16,3 (2014) 1617-1634.
83. Research, P., J. ve Raj, P., Topology-based analysis of performance evaluation of centralized vs. distributed SDN controller, IEEE International Conference on Current Trends in Advanced Computing (ICCTAC), Şubat 2018, Bangalore, India, Bildiler Kitabı, 1-8.
84. Pal, C., Veena, S., Rustagi, R., P. ve Murthy, K., N., B., Implementation of Simplified Custom Topology Framework in Mininet, Asia-Pacific Conference on Computer Aided System Engineering (APCASE), Şubat 2014, South Kuta, Indonesia, Bildiler Kitabı, 48-53.
85. Oliveira, R., L., S., Schweitzer, C., M., Shinoda, A., A. ve Prete, L., R., Using Mininet for emulation and prototyping Software-Defined Networks, IEEE Colombian Conference on Communications and Computing (COLCOM), Haziran 2014, Bogota, Colombia, Bildiler Kitabı, 1-6.
86. Han, Z. ve Yu, L., A Survey of the BCube Data Center Network Topology, IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS), Mayıs 2018, Omaha, NE, USA, Bildiler Kitabı, 229-231.
87. Özsağlam, M., Y. ve Çunkaş, M., Optimizasyon Problemlerinin Çözümü için Parçaçık Sürü Optimizasyonu Algoritması, Journal of Polytechnic, 11,4 (2008) 299-305.

88. Neghabi, A., A., Navimipour, N., J., Hosseinzadeh, M. ve Rezaee, A., Nature-inspired meta-heuristic algorithms for solving the load balancing problem in the software-defined network, International Journal of Communication Systems, 32,7 (2019) 1-26.
89. Wang, K., P., Huang, L., Zhou, C., G. ve Pang, W., Particle swarm optimization for traveling salesman problem, International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693), Kasım 2003, Xi'an, Bildiler Kitabı, 1583-1585.
90. Abdel-Kader, R., F., Hybrid discrete PSO with GA operators for efficient QoS-multicast routing, Ain Shams Engineering Journal, 2,1 (2011) 21-31.
91. Cai, Q., Gong, M., Shen, B., Ma, L. ve Jiao, L., Discrete particle swarm optimization for identifying community structures in signed social networks, Neural Networks, 58 (2014) 4-13.
92. https://pages.cs.wisc.edu/~tbenson/IMC10_Data.html Data Set for IMC 2010 Data Center Measurement. 5 Nisan 2022.
93. <https://github.com/dnnpkr/flowRecorder> A Network Traffic Flow Feature Measurement Tool. 5 Nisan 2022.
94. Pekar, A., Duque-Torres, A., Seah, W., K., G. ve Rendon, O., M., C., Towards threshold-agnostic heavy-hitter classification, International Journal of Network Management, 32,3 (2022) 1-22.
95. Siddique M., A., B, Arif, R., B. ve Khan M., M., R., Digital Image Segmentation in Matlab: A Brief Study on OTSU's Image Thresholding, International Conference on Innovation in Engineering and Technology (ICIET), Aralık 2018, Dhaka, Bangladesh, Bildiler Kitabı, 1-5.
96. Wright, R., E., Logistic Regression, L. G. Grimm ve P. R. Yarnold (Eds.), 217-244, Reading and Understanding Multivariate Statistics, Washington DC: American Psychological Association, 1995.
97. Krantiwadmare, Analytics Vidhya, Logistic Regression in Machine Learning. <https://medium.com/analytics-vidhya/logistic-regression-in-machine-learning-f3a90c13bb41> 20 Mayıs 2022.
98. Khairi, M., H., H., Ariffin, S., H., S., Latiff, N., M., A., Yusof, K., M., Hassan, M., K., Al-Dhief, F., T., Hamdan, M., Khan, S. ve Hamzah, M., Detection and Classification of Conflict Flows in SDN Using Machine Learning Algorithms, IEEE Access, 9 (2021) 76024-76037.
99. Akca, M., F., Deep Learning Türkiye, Nedir Bu Destek Vektör Makineleri? (Makine Öğrenmesi Serisi-2). [https://medium.com/deep-learning-turkiye/nedir-bu-destek-vekt%C3%B6r-makineleri-makine-%C3%B6%C4%9Frenmesi-serisi-2-94e576e4223e#:~:text=Destek%20Vekt%C3%B6r%20Makineleri%20\(Support%20Vektor,de%20maksimum%20uzakl%C4%B1%20olmas%C4%B1n%C4%B1%20ama%C3%A7lar](https://medium.com/deep-learning-turkiye/nedir-bu-destek-vekt%C3%B6r-makineleri-makine-%C3%B6%C4%9Frenmesi-serisi-2-94e576e4223e#:~:text=Destek%20Vekt%C3%B6r%20Makineleri%20(Support%20Vektor,de%20maksimum%20uzakl%C4%B1%20olmas%C4%B1n%C4%B1%20ama%C3%A7lar) 20 Mayıs 2022.

100. Loh, W., Y., Classification and Regression Trees, WIREs Data Mining And Knowledge Discovery, 1,1 (2011) 14-23.
101. Breiman, L., Random Forests, Machine Learning, 45 (2001) 5–32.
102. Akdağlı, E., Makine Öğrenmesinde Random Forest Algoritması. <https://ece-akdagli.medium.com/makine-%C3%B6%C4%9Frenmesinde-random-forest-algoritmas%C4%B1-a79b044bbb31> 22 Mayıs 2022.
103. Ahmad, A., Harjula, E., Ylianttila, M. ve Ahmad, I., Evaluation of Machine Learning Techniques for Security in SDN, IEEE Globecom Workshops (GC Wkshps), Aralık 2020, Taipei, Taiwan, Bildiriler Kitabı, 1-6.
104. <https://analyticslearn.com/naive-bayes-classifier-unlimited-guide-on-naive-bayes> Naive Bayes Classifier: Unlimited Guide On Naive Bayes. 25 Mayıs 2022.
105. Cunningham, P. ve Delany, S., J., k-Nearest Neighbour Classifiers, ACM Computing Surveys, 54,6 (2021) 1-25.
106. Özdemir, K., Bilge Adam Teknoloji, K-En Yakın Komşu Algoritması (K-Nearest Neighbors Algorithm). <https://medium.com/batech/k-en-yak%C4%B1n-kom%C5%9Fu-algoritmas%C4%B1-k-nearest-neighbors-algorithm-16e5ab69af2e> 25 Mayıs 2022.
107. Liu, Y., Chen, X., Peng, H. ve Wang, Z., Multi-focus image fusion with a deep convolutional neural network, Information Fusion, 36 (2017) 191-207.
108. Zhang, Y., Liu, Y., Sun, P., Yan, H., Zhao, X. ve Zhang, L., IFCNN: A general image fusion framework based on convolutional neural network, Information Fusion, 54 (2020) 99-118.
109. Amin-Naji, M., Aghagolzadeh, A. ve Ezoji, M., Ensemble of CNN for multi-focus image fusion, Information Fusion, 51 (2019) 201-214.
110. Sperling, E., Semiconductor Engineering, Deep Learning Spreads. <https://semiengineering.com/deep-learning-spreads/> 25 Mayıs 2022.
111. Noman, H., M. ve Jasim, M., N., POX Controller and Open Flow Performance Evaluation in Software Defined Networks (SDN) Using Mininet Emulator, 3rd International Conference on Sustainable Engineering Techniques (ICSET 2020), Nisan 2020, Baghdad, Iraq, Bildiriler Kitabı, 012102.
112. <http://mininet.org/overview/> Mininet Overview. 10 Ağustos 2020.
113. Aymaz, Ş., Çavdar, T. ve Aymaz, S., Load Balancing with a New Approach for Dynamic Server Selection in Software Defined Network, 1st International Conference on Electrical-Electronics and Computer Engineering (ICEECE 2021), Temmuz 2021, Trabzon, Türkiye, Bildiriler Kitabı, 85-91.

114. Hsiao, Y., T., Chuang, C., L. ve Chien, C., C., Computer network load-balancing and routing by ant colony optimization, 12th IEEE International Conference on Networks (ICON 2004) (IEEE Cat. No.04EX955), Kasım 2004, Singapore, Bildiriler Kitabı, 313-318.
115. Mathis, M., Semke, J., Mahdavi, J. ve Ott, T., The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm, ACM SIGCOMM, 27,3 (1997) 67–82.
116. Bwalya, B., M. ve Tembo, S., Performance Evaluation of Buffer Size for Access Networks in First Generation Optical Networks, International Journal of Internet of Things, 6,3 (2017) 98-105.
117. Çavdar, T. ve Aymaz, Ş., New Approach to Dynamic Load Balancing in Software-Defined Network-Based Data Centers, ETRI Journal, 45,3 (2023) 433-447.
118. Kumar, A., Data Analytics; Data, Data Science, Machine Learning, AI. <https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/#:~:text=Precision%2DRecall%20score%20is%20a,of%20all%20the%20predictions%20made> 25 Mayıs 2022.
119. Aymaz, Ş. ve Çavdar, T., Efficient Routing by Detecting Elephant Flows with Deep Learning Method in SDN, Advances in Electrical and Computer Engineering, 23,3 (2023) 57-66. <https://doi.org/10.4316/AECE.2023.03007>

ÖZGEÇMİŞ

Şeyma AYMAZ, ilkokulu ve ortaokulu Gazipaşa İlköğretim Okulu'nda ve liseyi Gümüşhane Fen Lisesi'nde tamamlamıştır. Lisans öğrenimini Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü'nde 2012 yılında tamamlamıştır. 2015-2016 eğitim-öğretim yılının bahar döneminde Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda tezli yüksek lisans programını tamamlamış ve takip eden güz dönemi doktora programına başlamıştır. 2013-2022 yılları arasında Karadeniz Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü'nde Araştırma Görevlisi olarak çalışmıştır. 2022 yılında Karadeniz Teknik Üniversitesi, Trabzon MYO, Bilgisayar Teknolojileri Bölümü'ne Öğretim Görevlisi olarak atanmıştır ve görevine hala devam etmektedir. Evli ve bir çocuk annesidir.

Uluslararası hakemli dergilerde yayınlanan makaleler (SCI/SCI-E)

1. Çavdar, T. ve Aymaz, Ş., New Approach to Dynamic Load Balancing in Software-Defined Network-Based Data Centers, ETRI Journal, 45,3 (2023) 433-447. <https://doi.org/10.4218/etrij.2021-0478>
2. Aymaz, Ş. ve Çavdar, T., Efficient Routing by Detecting Elephant Flows with Deep Learning Method in SDN, Advances in Electrical and Computer Engineering, 23,3 (2023) 57-66. <https://doi.org/10.4316/AECE.2023.03007>

Uluslararası hakemli dergilerde süreci devam eden makaleler (SCI/SCI-E)

1. Çavdar, T., Aymaz, Ş. ve Aymaz, S., A framework for elephant flow detection for SDNs based on the classification, Arabian Journal for Science and Engineering, revizyon sonrası hakem değerlendirmesindedir.

Diğer Dergilerde Yayınlanan Makaleler

1. Aymaz, Ş. ve Hacıoğlu, G., Measurement of Energy Consumption of Wsn for Indoor Environment, International Journal of Energy Applications and Technologies, 4,2 (2017) 87-93.

Ulusal ve uluslararası konferanslarda sözlü sunumu yapılan bildiriler

1. Aymaz, Ş., Çavdar, T. ve Aymaz, S., Load Balancing with a New Approach for Dynamic Server Selection in Software Defined Network, 1st International

Conference on Electrical-Electronics and Computer Engineering, Temmuz 2021, Trabzon, Türkiye, Bildiriler Kitabı, 85-91.

2. Aymaz, Ş., Çavdar, T., Aymaz, S. ve Öztürk, E., An Analysis of Load Balancing Strategies with Wireshark in Software Defined Networks, 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Eylül 2018, Malatya, Türkiye, Bildiriler Kitabı, 1-5.
3. Aymaz, Ş., Çavdar, T. ve Dönmez Çavdar, A., Fire Evacuation Route Determination based on Particle Swarm Optimization, 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Eylül 2017, Malatya, Türkiye, Bildiriler Kitabı, 1-6.

