



**T.C.
İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**



YÜKSEK LİSANS TEZİ

**YAZILIM TANIMLI AĞLARDA MAKİNE ÖĞRENME
ALGORİTMALARI İLE TIKANIKLIK KONTROLÜ**

Fatma GÜMÜŞ

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Danışman

Yrd. Doç. Dr. Derya YILTAŞ KAPLAN

Ocak, 2016

İSTANBUL

Bu çalışma 21/01/2016 tarihinde ařağıdaki jüri tarafından Bilgisayar Mühendisliğı Anabilim Dalı Bilgisayar Mühendisliğı programında Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Jürisi:



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



Prof.Dr. Ahmet SERTBAŞ
İstanbul Üniversitesi
Mühendislik Fakültesi



Doç.Dr. Fethullah KARABİBER
Yıldız Teknik Üniversitesi
Elektrik Elektronik Fakültesi



Doç.Dr. Olcay KURŞUN
İstanbul Üniversitesi
Mühendislik Fakültesi



Yrd.Doç.Dr. Tolga ENSARİ
İstanbul Üniversitesi
Mühendislik Fakültesi

ÖNSÖZ

Tez çalışmam boyunca ilgi ve yönlendirmelerinden ötürü değerli hocam, tez danışmanım Yrd. Doç. Dr. Derya YILTAŞ KAPLAN'a, ve yapay öğrenme alanı ile tanışmama vesile olan, bu konudaki eğitimimde büyük emeği geçen değerli hocam Doç. Dr. Olcay KURŞUN'a teşekkür ederim.

Yüksek lisans sürecim boyunca bana destek olan ve her türlü huysuzluğuma sabır gösteren anne ve babama teşekkür ederim.

Ocak, 2016

Fatma GÜMÜŞ

İÇİNDEKİLER

Sayfa No

ÖNSÖZ.....	i
İÇİNDEKİLER	ii
ŞEKİL LİSTESİ.....	iv
TABLO LİSTESİ	vi
SİMGE VE KISALTMA LİSTESİ	vii
ÖZET.....	ix
SUMMARY	x
1. GİRİŞ.....	1
2. GENEL KISIMLAR	4
2.1 YAZILIM TANIMLI BİLGİSAYAR AĞLARI	4
2.1.1 Geleneksel Ağlardaki Yetersizlikler	4
2.1.2 Yazılım Tanımlı Ağ Çözümü.....	6
2.1.3 Yazılım Tanımlı Ağların Bileşenleri	8
2.1.3.1 Veri Düzlemi.....	10
2.1.3.2 Kontrol Düzlemi	12
2.1.3.3 Yönetim Düzlemi	16
2.2 YAZILIM TANIMLI AĞLARDA TRAFİK VE TIKANIKLIK KONTROLÜ	19
2.2.1 Yazılım Tanımlı Ağ Trafik Mühendisliğine Genel Bakış	20
2.2.2 Akış Yönetimi	21
2.2.3 Ağ Uygulaması Seviyesinde Yazılım Tanımlı Ağ Trafik Analizi.....	23
2.3 MAKİNE ÖĞRENMESİ	24
2.3.1 Önemli Kavram ve Tanımlar	24
2.3.2 Makine Öğrenme Teknikleri.....	26
2.3.2.1 Boyut Azaltma	26
2.3.2.2 Öğrenmenin Yapay Sinir Ağları ile Gerçekleştirilmesi.....	30
2.3.3 Öğrenmenin Ölçümü.....	35
2.4 AĞ TIKANIKLIK KONTROLÜNÜN MAKİNE ÖĞRENMESİ İLE SAĞLANMASI	36
2.4.1 Geleneksel Ağlarda Yapılan Çalışmalar	36

2.4.2 Yazılım Tanımlı Ağlarda Yapılan Çalışmalar	40
3. MALZEME VE YÖNTEM	43
3.1 SİMÜLASYON ORTAMI	43
3.2 VERİNİN TOPLANMASI VE DÜZENLENMESİ.....	49
3.3 ALGORİTMA	52
4. BULGULAR	56
5. TARTIŞMA VE SONUÇ	69
KAYNAKLAR	72
EKLER.....	78
EK 1.....	78
ÖZGEÇMİŞ.....	85

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1: Geleneksel ağ cihazı mantıksal şema.	5
Şekil 2.2: NSX Blok Diyagramı. [14]	8
Şekil 2.3: Geleneksel ağ. 1: Ağ uygulamaları. 2: Kontrol düzlemi. 3: Veri düzlemi. [15]	9
Şekil 2.4: YTA. 1: Ağ uygulamaları. 2: Kontrol düzlemi. 3: Veri düzlemi. [15].....	10
Şekil 2.5: Akış tablosu bölümleri.	12
Şekil 2.6: NOX denetleyici ağı. 1: Kenar istemciler. 2: OpenFlow anahtarlar. [17].....	14
Şekil 2.7: Bir HyperFlow demetinden diğerine veri yönlendirme şeması. [19]	15
Şekil 2.8: Sanal yönlendirme aygıtı: "Tek büyük anahtar".	16
Şekil 2.9: YTA trafik mühendisliği bileşenleri. 1: Ölçümler. 2: Kontrol. 3: Optimizasyon	21
Şekil 2.10: Yapay nöron.	31
Şekil 2.11: İleri beslemeli ağ statik YSA.	33
Şekil 2.12: Gecikmeli YSA.	34
Şekil 2.13: Geri beslemeli YSA.	34
Şekil 3.1: OMNET++'ta örnek bir ağ yapısı. 1: Basit modül. 2: Bileşik modül. 3: Tek yönlü kapı. 4: Çift yönlü kapı. 5: Bağlantı.	44
Şekil 3.2: Bir simülasyon boyunca zaman ekseninde gerçekleşen olaylar ve moduleX'in kendine zamanlanmış mesaj göndermesi örneği.	46
Şekil 3.3: Ofomnet OpenFlow Anahtarı.....	48
Şekil 3.4: Ofomnet OpenFlow Denetleyici.	49
Şekil 3.5: Kullanılan dumbbell topolojisi.....	50
Şekil 3.6: Doğrulama topolojisi.....	52
Şekil 3.7: Tıkanıklık öngörü sistemi: 2 öz nitelikli veri setini girdi olarak alan, 2 adım gecikmeli ve 1 gizli katmandan oluşur.	54
Şekil 3.8: NAR ağı yapısı.....	55

Şekil 3.9: Doğrusal olmayan ileri beslemeli model.....	55
Şekil 4.1: Giden paket durum diyagramı.....	58
Şekil 4.2: 1 ara katman ile yürütülen algoritma için öngörü çıktıları.....	62
Şekil 4.3: 1 ara katman ile yürütülen algoritma için performans durumu.....	62
Şekil 4.4: 2 ara katman ile yürütülen algoritma için öngörü çıktıları.....	62
Şekil 4.5: 2 ara katman ile yürütülen algoritma için performans durumu.....	63
Şekil 4.6: 5 ara katman ile yürütülen algoritma için öngörü çıktıları.....	63
Şekil 4.7: 5 ara katman ile yürütülen algoritma için performans durumu.....	64
Şekil 4.8: 10 ara katman ile yürütülen algoritma için öngörü çıktıları.....	64
Şekil 4.9: 10 ara katman ile yürütülen algoritma için performans durumu.....	65
Şekil 4.10: 15 ara katman ile yürütülen algoritma için öngörü çıktıları.....	65
Şekil 4.11: 15 ara katman ile yürütülen algoritma için performans durumu.....	66
Şekil 4.12: Doğrusal olmayan ileri beslemeli model için öngörü çıktıları.....	67
Şekil 4.13: Doğrusal olmayan ileri beslemeli model performans durumu.....	67
Şekil 4.14: NAR modeli için öngörü çıktıları.....	67
Şekil 4.15: NAR modeli için performans durumu.....	68

TABLO LİSTESİ

	Sayfa No
Tablo 2.1: Geleneksel ve karmaşık ağ fonksiyonları.	18
Tablo 2.2: Algoritma - 1	37
Tablo 2.3: Algoritma - 2	38
Tablo 3.1: Standart üretilen istatistikler.	50
Tablo 3.2: Yoğunluk senaryoları.....	51
Tablo 4.1: Öznitelik kümesi.....	57
Tablo 4.2: mRMR sonucu tespit edilen gereksiz ve fazlalık oluşturan öznitelikler.....	59
Tablo 4. 3: Veri kümesinde ilgili bulunan öznitelikler ve ilgi oranları.....	60
Tablo 4.4: Katman sayısının performansa etkisi.....	61
Tablo 4.5: Performans karşılaştırması.....	66
Tablo 4.6: Performans karşılaştırması-2.	68

SİMGE VE KISALTMA LİSTESİ

Simgeler	Açıklama
C	: Sınıf etiketi
D	: Veri kümesi toplam boyut (öznitelik) sayısı
d	: Veri kümesi indirgenmiş boyut (öznitelik) sayısı
d_i	: Beklenen çıktı
E	: Hata
F	: Öznitelik kümesi
f	: Öznitelik kümesi elemandan oluşan veri vektörü
f_F	: F kümesi elemandan oluşan veri vektörü
f_G	: G kümesi elemandan oluşan veri vektörü
F_i	: Öznitelik kümesinde i'inci eleman
G	: Optimal öznitelik altkümesi
$H(X)$	: X değişken kümesinin entropisi
$H(X Y)$	: Y değişkeninin değerlendirilmesinden sonra X'in entropisi
IG	: Bilgi kazanımı
M_i	: i'inci elemanın Markov örtüsü
N	: Veri kümesindeki örnek sayısı
r	: Öğrenme sinyali
s	: Öngörü ufku
S_i	: i'inci elemanın F kümesinden çıkarılmasıyla oluşan tersine eleme kümesi
S_i'	: S_i 'nin en az bir elemanlı altkümesi
SU	: Simetrik bilinmezlik
t	: Zaman birimi değişkeni
w	: Yapay sinir ağı ağırlık vektörün bir elemanı
W	: Yapay sinir ağının ağırlık vektörü
X	: Veri kümesi
y_i	: Öğrenme çıktısı

Kısaltmalar**Açıklama**

ACK	: Onay Sinyali
ACM	: Bilgisayar Makineleri Derneği
API	: Uygulama Programlama Arayüzü
ARP	: Adres Çözümleme Protokolü
DCTCP	: Metalaşmış Veri Merkezi için Etkili Paket Taşıma
DNN	: Derin Sinir Ağları
DNS	: Etki Alanı Ad Sunucusu
ECMP	: Eş Maliyetli Çok Yollu Rota
ECN	: Net Tıkanıklık Bildirme
FIB	: Yönlendirme Bilgisi Temeli
GOK	: Gelecek Olaylar Kümesi
GPS	: Küresel Konumlama Sistemi
GPU	: Grafik İleme Ünitesi
HP	: Hewlett Packard
IP	: İnternet Protokolü
IPv4	: İnternet Protokolü Veriyon 4
IPv6	: İnternet Protokolü Veriyon 6
L3	: Ağ Katmanı
MAC	: Ortam Erişim Yönetimi
mRMR	: En Az Fazlalık En Çok İlgililik
MSE	: Ortalama Kare Hata
NAR	: Doğrusal Olmayan Otoregresif Model
NARX	: Eksojen Girişi ile Doğrusal Olmayan Otoregresif Model
NED	: Ağ Tanımlayıcı
ONF	: Açık Ağ Birliği
P2P	: Uçtan Uca Ağ
QoS	: Servis Kalitesi
RBM	: Kısıtlı Boltzmann Makinesi
RNN	: Yinelenen Sinir Ağı
SIGKDD	: Bilgi Heşfi ve Veri Madenciliği
STCP	: Sensör İletim Kontrol Protokolü
TCAM	: İçeriği Adreslenebilir Üçlü Bellek
TCP	: Transfer Kontrol Protokolü
TSN	: İletim Sekans Numarası
YSA	: Yapay Sinir Ağı
YTA	: Yazılım Tanımlı Ağ

ÖZET

YÜKSEK LİSANS TEZİ

YAZILIM TANIMLI AĞLARDA MAKİNE ÖĞRENME ALGORİTMALARI İLE TIKANIKLIK KONTROLÜ

Fatma GÜMÜŞ

İstanbul Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman : Yrd. Doç. Dr. Derya YILTAŞ KAPLAN

Geleneksel TCP/IP ağ yapısından kaynaklanan çeşitli ağ sorunlarının üstesinden gelmek için alternatif yaklaşım arayışlarına gidilmiştir. Akademi ve sektörde kullanımı ivme kazanan yeni yaklaşımlardan biri yazılım tanımlı ağlardır. Kontrol ve yönlendirme fonksiyonlarının iki farklı alanda gerçekleştirildiği bu yaklaşımda, tıkanıklık kontrolü yöntemleri ile ilgili çalışmalar yenidir. Bu konudaki literatür açığını kapatma yönünde katkı sağlayacak olan bu çalışmada makine öğrenme algoritmaları kullanılmıştır. Simülasyon programında üretilen veri kümesi önerilen tıkanıklık öngörüsü sistemine girdi olarak verilmiştir. Öznitelik seçiminde En Az Fazlalık En Çok İlgililik (mRMR) yönteminden yararlanılmış, eksojen girişi ile doğrusal olmayan otoregresif model (NARX), doğrusal olmayan otoregresif model (NAR) ve doğrusal olmayan ileri beslemeli yapay sinir ağları üzerinde öğrenme aşaması gerçekleştirilmiştir. Çalışma sonucunda, veri kümesini iki boyutlu probleme indirgeyerek ve yapay sinir ağında az sayıda ara katman kullanılarak, yüksek başarımın elde edildiği gösterilmiştir.

Ocak 2016, 95 sayfa.

Anahtar kelimeler: Yazılım tanımlı ağ, tıkanıklık tahmini, öznitelik seçimi, yapay sinir ağı.

SUMMARY

M.Sc. THESIS

CONGESTION CONTROL IN SOFTWARE DEFINED NETWORKS WITH MACHINE LEARNING ALGORITHMS

Fatma GÜMÜŞ

İstanbul University

Institute of Graduate Studies in Science and Engineering

Department of Computer Engineering

Supervisor : Yrd. Doç. Dr. Derya YILTAŞ KAPLAN

To overcome the various network problems resulting from traditional TCP / IP network structure, research on alternative approaches are being conducted. One of the alternative approaches, software defined networking, is increasing its popularity among both academic and related commercial world. Congestion control in software defined networking is a relatively new research area, and this thesis uses machine learning algorithms to address the matter, aiming to contribute towards filling the need of research on the subject. The dataset produced in network simulation program is used as input of the proposed congestion prediction system. On the feature selection stage, minimum redundancy maximum relevance (mRMR) is used. Then, at the learning stage, nonlinear autoregressive model with exogenous input (NARX), nonlinear autoregression models (NAR) and non-linear feed-forward neural networks were performed. The experimental results show that, the high dimensional data reduced into the degree of two, combined with designing neural networks with minimum number of layers results in high prediction performance.

January 2016, 95 pages.

Keywords: Software defined network, congestion prediction, feature selection, artificial neural network.

1. GİRİŞ

Bilgi yönetimi günümüzde, dinamiklik ve esneklik gerektirmektedir. Verinin saklandığı, işlendiği ve istemcilere dağıtıldığı noktalar olan sunucular üzerinde yürütülen uygulamaların her biri farklı kaynak gereksinimlerine sahiptirler. Dahası uygulamalar, sunucununkinden farklı işletim sisteminde yürütülüyor olabilirler. Sanal makine çözümleri, bir işletim sistemi üzerinde başka bir işletim sistemini çalıştırmayı mümkün kılarak sunucuların birden fazla işletim sisteminden verim almasını sağlamıştır. Bununla birlikte, uygulamalara dinamik kaynak sağlamayı olanaklı hale getirmiştir. Bilgi yönetiminin kazandığı bu yeni boyut, fiziksel sunucu yönetimi sınırlarının aşılp sanallaştırma teknolojisinin gelişmesine öncülük etmiştir. Sanallaştırmayı esnek hesaplama (elastic computing) teknolojisi izlemiş, bu yeni gelişme sunucu servis müşterilerine ayrılmış kaynakların, ihtiyaç halinde dinamik olarak artırılması ya da azaltılması imkânını sağlamıştır[1].

Tek bir fiziksel sunucuda, farklı ihtiyaçlara sahip müşteri barındıran veri merkezleri göz önüne alındığında, müşteri servislerinin birbirinden ayrılmasındaki önem görülmektedir. Bunu gerçekleştirmek üzere en çok kullanılan yöntem sanal soyutlamadır. Sektör liderlerinin tercih ettiği bu yöntem[2], farklı protokol ve kümeleme seviyesine sahip uygulamalar için yetersiz kalmaktadır. Allcock ve diğ.[3] çalışmasında olduğu gibi coğrafik olarak dağıtık, bilimsel hesaplamalar için kümelenmiş, özel seviyeler gerektiren uygulamalarda söz konusu yetersizlik daha açık görülmektedir. Bunun ve artan maliyetlerin önüne geçmek için çeşitli protokoller geliştirilmiştir. İhtiyaç oluşukça tasarlanan bu protokoller arasında uyumluluk sağlamak ise, her geçen gün artan sayıları nedeniyle, çok zor hale gelmiştir[4]. Dahası, protokol katmanları arası optimizasyon garantisi çoğu zaman verilememektedir. Wang ve diğ.[5], birçok varsayım yaparak verimliliği azami seviyeye çıkarmalarına karşın sistem kararlılığından büyük ödün vermek zorunda kaldıklarını bildirmişler. Bunun, günümüz bilgi yönetimi yüksek kararlılık gereksinimi göz önünde bulundurulduğunda, azımsanamaz bir maliyet olduğu açıktır.

Günümüz bilgisayar ağları, sanallaştırma ve soyutlamanın karmaşıklığının üstesinden gelebilmelidir. Aynı zamanda protokol uyumsuzluklarını ortadan kaldırmalı, dinamiklik ve ölçeklenebilirlik gereksinimlerini sağlamalıdır. Ayrıca, ideal olarak, ağ aygıt sağlayıcılarına bağımlılığı asgari olmalıdır. Geleneksel TCP/IP paradigması, bu gereksinimleri sağlamada yetersiz kalmıştır. Ağ aygıtlarının gömülü protokolleri ve uygulama esnekliğinin önüne geçen hiyerarşilerine bağımlı olması, yetersizliğin göstergelerindendir. Bahsedilen günümüz ağ ihtiyaçlarının karşılanması amacıyla, aygıt üreticileri, servis sağlayıcıları ve akademi dünyası, daha sağlam ve güvenilir dijital iletişim yapısı sağlayacak yeni bir ağ standardı arayışına girmiştir.

Geleneksel ağ mimarisine en popüler alternatiflerden biri *yazılım tanımlı ağlar*dır. Bölüm 2.1.'de geleneksel ağlardaki yetersizliklerin belirtilmesinin ardından, yazılım tanımlı ağ çözümüne geniş yer verilmiş ve bileşenleri açıklanmıştır. Bu yeni mimaride tıkanıklık kontrolüne ilişkin çalışmalar, Bölüm 2.2.'de ayrıntılı ele alındığı üzere, oldukça yeni ve yetersizdir. Bu tez çalışmasında, konu makine öğrenmesi bakış açısıyla ele alınıp ilgili açığın kapatılması yönünde katkı sağlamak amaçlanmaktadır. Makine öğrenmesi kavram ve tekniklerine yer verildiği Bölüm 2.3.'te, öğrenme ölçümünün de yer aldığı konu, bu tezde tıkanıklık kontrol probleminde kullanılan yöntemler kapsamında anlatılmıştır. Tıkanıklık kontrolünde makine öğrenmesinin kullanılması ilk defa bu tez ile gerçekleştirilmemiştir. Özellikle geleneksel ağlarda bu tür çalışmalar yapılmış, yazılım tanımlı ağlarda ise kullanılmaya başlanmıştır. Bölüm 2.4.'te her iki ağ yapısı üzerinde tıkanıklık kontrolünün makine öğrenmesi ile sağlandığı çalışmalardan örneklerle yer verilmiştir.

Yazılım tanımlı ağlara uygun girişler ile piyasaya sürülen ağ cihazları, geleneksel ağ yapısından gelen klasik tıkanıklık kontrol yöntemlerini (yavaş başlatma gibi) kullanabilmektedir. Tıkanıklık kontrol mekanizmasının her bir cihaz üzerinde çalıştırılması, yazılım tanımlı ağların getirdiği "merkezi kontrol" mantığına aykırı olduğu için etkin bir uygulama değildir. Yazılım tanımlı ağlara özel geliştirilmiş mevcut tıkanıklık kontrol yöntemleri ise derin paket incelemesine (deep packet inspection) ya da akışların ve tabloların etkin kullanılmasına dayanmaktadır. Söz konusu yöntemler ağa hâkim olan trafik karakteristiğine göre kalibrasyon gerektirmektedir. Tezde önerilen tıkanıklık öngörü algoritması ise, yazılım tanımlı ağların yönetim düzleminde

gerçeklenmek üzere geliştirildiği için hem merkezi yapıdadır hem de ağ trafiğinin çeşitliliğinden bağımsız, üst seviyede genel bir çerçeveye sunmaktadır. Bu sayede farklı topoloji ve uygulama alanlarına entegre edilmesi çok daha kolaydır.

Bölüm 3'te malzeme ve yöntem açıklanmıştır. Simülasyon programında tasarlanan ağ yapısında veri toplanmış ve tıkanıklık ölçümü yapılmıştır. Kullanılan simülasyon ortamı Bölüm 3.1.'de incelendikten sonra, verinin toplanması ve düzenlenmesi Bölüm 3.2.'de anlatılmıştır. Bölüm 3.3.'te önerilen tıkanıklık kontrolü yapısının algoritması verilmiştir. Tez çalışmasından elde edilen bulgular Bölüm 4'te açıklanırken, Bölüm 5 ise konu ile ilgili tartışma ve sonuç için ayrılmıştır.

2. GENEL KISIMLAR

2.1 YAZILIM TANIMLI BİLGİSAYAR AĞLARI

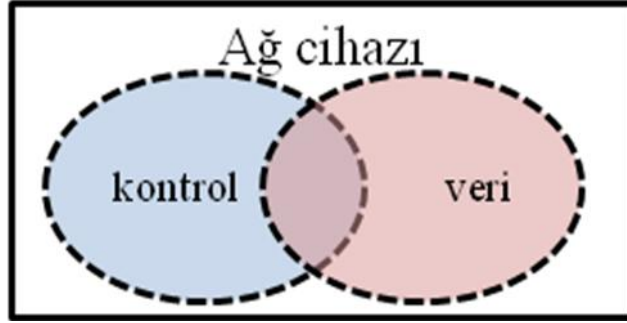
2.1.1 Geleneksel Ağlardaki Yetersizlikler

Geleneksel IP bilgisayar ağları karmaşık olmalarının yanı sıra yönetimsel zorluklar barındırır. Yapılandırma karmaşıklığı, hata dinamikleri, yük değişiklikleri gibi hususları dikkate alarak bir karmaşıklık metriği elde eden Benson ve diğer., arayüzler gibi yeni işlevlerin eklenmesi, var olan bir işlevin güncellenmesi, ve yüksek seviye güvenliğin sağlanması işlemlerinin zorluğunu ölçmüştür[4]. Çalışmaya göre, yapılandırma hataları ağın karmaşıklığı ile doğru orantılı olarak artar. Geleneksel ağlarda karmaşıklığın ölçülmesi için metrik oluşturmak dahi zordur, çünkü çok farklı büyüklükte ve tasarıma sahip olan ağlarda karşılaştırma yapabilmek için, elzem olmayan ağ konfigürasyonlarını yok saymak gerekir.

Bilgisayar ağları fonksiyonel olarak üç düzleme ayrılabilir. Veri düzlemi ağ cihazlarında yer alır ve veriyi yönetir. Kontrol düzlemi, yönlendirme tablolarını doldurmak üzere geliştirilen protokollerden oluşur. Yönetim düzlemi ise, yazılım servisleridir; tipik olarak kontrol düzleminin fonksiyonlarını uzaktan takip etmek ve değiştirmek için kullanılır.

Dikey entegre yapıya sahip olmaları geleneksel ağları karışık hale getirir; dallanmaların takibi zordur. Her bir ağ cihazında kontrol ve veri düzlemi bulunur ve iç içe geçmiştir. Bu, oldukça karmaşık ve çoğunlukla statik mimarilere neden olmuştur [4,6,7]. İnovasyon oldukça zor gerçekleştirilir. Ağda bir cihazda yapılan tek bir konfigürasyon hatası tüm ağın doğru çalışmasını uzun bir süre engelleyebilir. Bunun önüne geçmek amacıyla cihaz üreticileri, cihaza özel çözümler getirmiştir. Bunlar özelleştirilmiş donanım parçaları olabileceği gibi, işletim sistemi ya da ağ uygulaması da olabilir. Sonuçta maliyet artar ve bu, inovasyon ve yeni özelliklerin getirilmesi kararlarının önünde büyük engel oluşturur.

Şekil 2.1'de mantıksal yapısı görünen cihazlarda hem trafiğin nasıl yönetileceğine karar verilir, hem de kontrol düzleminin yönetim kararına göre trafiğin yönlendirilme işlemi yapılır. Günümüz IP ağlarının iç yapıları nedeniyle yeni bir yönlendirme protokolünün tasarım, ölçüm/değerlendirme ve uygulama adımları 5 ile 10 yıl arasında süre gerektirebilir. IPv6'nın yalnızca protokol güncellemesi olduğu halde 10 seneyi aşkın süredir IPv4'ten geçişin tamamıyla bitmemiş olması bu iç içe yapının yol açtığı bir zorluktur.



Şekil 2.1: Geleneksel ağ cihazı mantıksal şema.

Bununla birlikte, Internet mimarisine IP yerine başka bir yapı getirilmesi gibi yeni bir yaklaşımı uygulamak oldukça zor bir görevdir ve pratikte gerçekleştirilemez değişiklik olarak görülmektedir. Ghodsi ve diğer., mimari evrime olanak tanıyan akıllı tasarımlar üzerine yaptıkları çalışmada [6], IP, TCP, DNS ya da soket API'leri gibi internet mimarisinin çekirdek özelliklerini temelli bir değişikliğe sokmanın oldukça zor olduğunu belirtmişler. Çalışmada, mimari evriminin önündeki engellerden bahsedilirken kolay, zor ve karmaşık olmak üzere üç sınıf değişiklik sıralanmıştır. Kolay olarak belirtilen değişim, sunucu işletim sisteminin uygulamalara sunduğu ağ API'sinin (netAPI) değiştirilmesidir. Bu da çok zahmetsiz olmayacaktır, çünkü netAPI uygulamalar içine gömülüdür ve ters yönlü uyumu olmayan değişiklikler, geçmişten gelen tüm uygulamaları derhal kullanılamaz kılacaktır. En zor mimari değişim IP olarak gösterilmektedir. Günümüzde L3, hem evrensel değişim formatı hem de adresleme yapısı sağlar. IPv6 örneğinde olduğu gibi L3'teki bu çift kullanışlılığı değiştirmeden IP'yi yeni bir versiyonu ile değiştirmek beraberinde birçok yeni yapılandırmayı getirir. Uygulamaları yeni adresleri idare edebilecek şekilde değiştirmek, ve yeni versiyonun diğer yeni versiyon ağlarla direkt konuşmasını sağlayacak üst üste binen altyapının

uygulamaya konulması gerekmektedir. Karmaşık değişikliklere bir örnek etki alanları (domain) arası yönlendirmedir. Bu hem etki alanları arası evrensel bir anlaşma ve kabullenme gerektirir, hem de çok yönlü anlam yapıları taşıyan global dağıtık algoritmaların karmaşıklığını barındırır.

2.1.2 Yazılım Tanımlı Ağ Çözümü

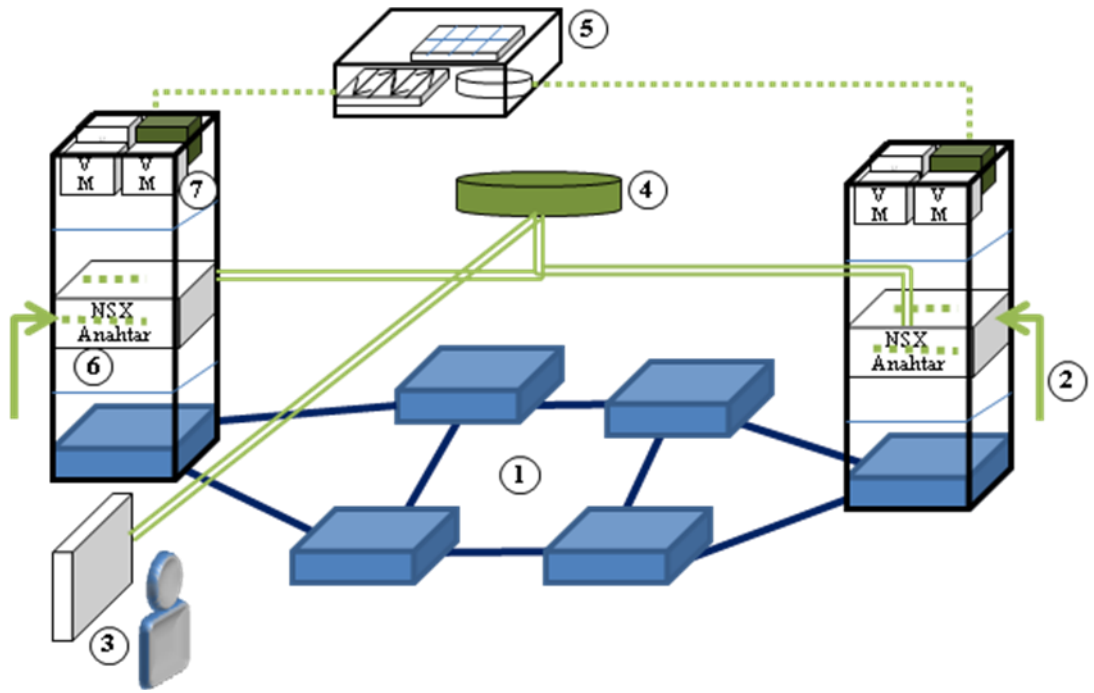
Geleneksel ağların dikey entegre yapısı, yazılım tanımlı mimari ile çözülüp, ayrılır. Böylece ağın kontrol mantığı, trafiği yönlendiren ağ cihazlarından çıkarılmış olur. Kontrol ve veri düzlemlerinin ayrılmasıyla yalnız yönlendirme göreviyle donatılmış ağ cihazları ve mantıksal olarak merkezileştirilmiş kontrol düzlemi bir "denetleyici (controller)" olarak gerçekleştirir.

Denetleyici ile ağ ilkelerinin yürürlüğe konulması, ağın yapılandırılması ve ölçümlemesi daha kolay hale gelir. Kim ve Feamster ağ yönetiminin yazılım tanımlı ağlar (YTA) ile geliştirilmesine yönelik yaptıkları çalışmada farklı tiplerdeki çeşitli ağların yönetim ve konfigürasyonunun daha iyi bir mekanizma sağladığını belirtmişler[7]. Operatörler için yüksek seviye ilkelerin uygulanmasının kolaylığını kanıtlamak üzere Procera adında olay-güdümlü bir YTA denetleyici yapısı tasarlamışlardır.

Mantıksal olarak merkezileştirilmiş kontrolü gerçekleştiren program modeli, fiziksel olarak merkezi yapıda olmasını gerektirmek zorunda değildir. Böyle bir faraziye, yeterli seviyede performans, ölçeklenebilirlik ve güvenilirliğin sağlanmasının önüne geçerdi. Fiziksel merkeziyet yerine, gerçekleştirme aşamasındaki YTA tasarımları fiziksel dağıtık yapıya yönelmektedir. Google'ın dünya çapında veri merkezlerini bağladığı özel WAN yapısı B4, dağıtık YTA tasarımına en büyük uygulama örneklerindendir [8]. B4 için belirlenen gereksinimler: çok miktarda bölgeye dağıtılmış oldukça büyük bant genişliği, ortalama bant genişliğini en yüksek seviyede tutmaya çalışan elastik trafik, ve tüm kenar sunucu ve ağların tam denetimi. B4'un fiziksel dağıtık, mantıksal merkezi YTA mimarisi ile bağlantıların neredeyse %100 verimlilikte çalıştığı ve uygulama akışlarının birden çok yola ayrılıp uygulamanın öncelik/talep dengesinin sağlandığı gözlemlenmiştir.

YTA'nın formal tanımı yapılırken "ilgi alanlarının ayrılması" ön plana çıkar. Bu ayrım, üç kategori olarak gerçekleşir: ağ ilkelerinin tanımı, ilkelerin ağ cihazı donanımında gerçekleşmesi, trafiğin yönlendirilmesi. Esnekliğin sağlandığı bu yapıda ağa yeni soyutlamaların tanımlanması daha kolay olur ve ağ yönetimi kolaylaşır. Dahası, ağ evrimi ve inovasyonunun daha hızlı olması sağlanır.

Açık Ağ Birliği (Open Networking Foundation, ONF), 2011'den bu yana YTA'ların gelişimini izlemektedir [9]. Konsorsiyum, kontrol ve veri düzleminin iletişiminin standartlaştırılması konusunda OpenFlow protokolü ile öncülük etmiştir. YTA ve OpenFlow, akademik çalışmalar olarak başlamıştır [10,11,12]. Protokolde, Ethernet anahtarı temel alınmıştır. İçinde akış tablosu ve akış girdilerinin eklenip çıkartılabileceği standartlaştırılmış bir arayüzü barındırır. Birçok ağ cihazı üreticisi artık OpenFlow API desteğini sağlayan cihazlar piyasaya sürmektedir. YTA üzerine bu hızlı sektör kabullenışı, Google, Facebook, Microsoft ve Verizon gibi akımın gerisinde kalmak istemeyen firmaları ONF'yi desteğe itmiştir [13]. Özellikle bu destek, YTA yaklaşımının olgunlaştığının ve akademik deney seviyesinden çıkıp ticari olmaya başladığının göstergesidir. Bunun yanında daha önce bahsedildiği üzere Google gibi firmalar kendi iç mimarilerinde YTA gerçekleştirmeye başlamıştır, dahası VMWare NSX ürünü ile yaklaşımı yaygınlaştıracak bir adım atmıştır [14]. NSX, yazılım tanımlı veri merkezleri için ağ sanallaştırma platformudur. Önemli özelliklerinden biri, var olan fiziksel altyapıyı üzerine uygulanabilmesi, ve tıpkı firmanın sunucu sanallaştırma ürünleri gibi, fiziksel yapıyı ilke yönelimli yaklaşım kullanarak iş yüklerine bağlanmış ağ ve güvenlik servislerini bir iletim kapasitesi havuzu olarak kullanabilmeyi sağlar. Çok katmanlı ağ ve güvenlik servislerinin hazırlanma zamanının haftalardan dakikalara indiği, ve diğer sanal ağlardan tamamen ayrı bir şekilde her ağın kendine tanımlanmış iş yükleriyle ihtiyacına göre düzenlenebilmesine olanak tanır. Ürün yapısının blok diyagram olarak gösterildiği Şekil 2.2'de, programatik olarak karmaşık işlevselliklerin oluşturulması, yönetilmesi, izlenmesi, silinmesi ya da geri alınmasının sağlandığı yapı görülmektedir.



Şekil 2.2: NSX Blok Diyagramı. [14]

1: Fiziksel altyapı. 2: Dağıtık ağ servisleri. 3: Bulut yönetim platformu. 4: Kümeleme denetleyicisi. 5: Sanal ağ. 6: Hipervizör (misafir işletim sistemi katmanı). 7: Kullanıcı alanı içinde sanal makineler.

2.1.3 Yazılım Tanımlı Ağların Bileşenleri

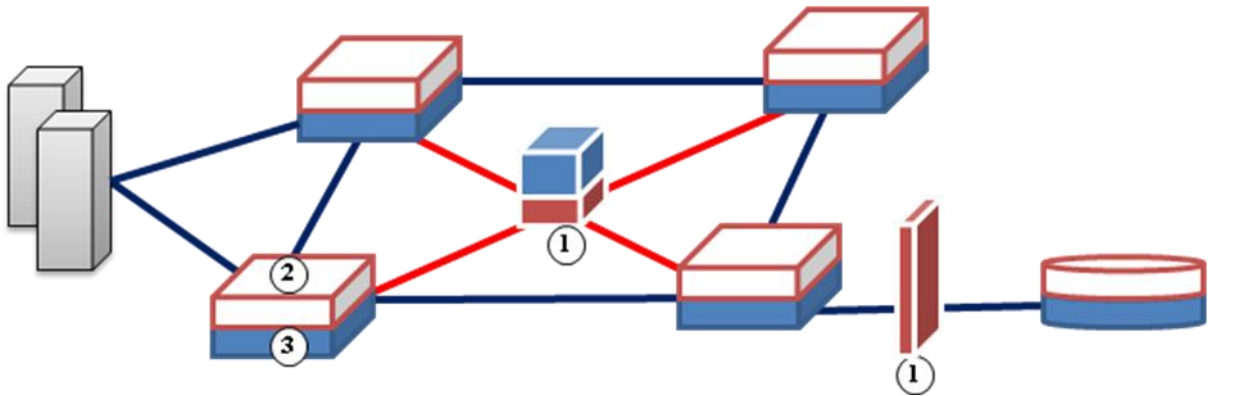
[15]'e göre YTA'lar dört taşıyıcı sütundan oluşur. Birincisi, kontrol ve veri düzlemlerinin ayrıklaştırılarak ilgi alanlarının ayrılmasıdır. İkincisi, geleneksel ağların hedef adres tabanlı yapısı yerine, akış tabanlı yapı kullanmasıdır. Akış, bir dizi veri paketi değeridir, belli bir kriter ve eyleme eşlenirler. Üçüncü sütun, kontrol mantığının harici bir birime dönüşmesidir. Bu birim en yaygın olarak denetleyici adıyla, ya da ağ işletim sistemi olarak görülür. Son olarak, ağ programlanabilir yapıdadır. Programlama denetleyici üzerinde çalışan ağ uygulamaları yoluyla, alt katmanlardaki veri düzlemi cihazlarının etkileşiminin sağlanması ile olur.

[16]'ya göre ise YTA üç temel soyutlama katmanından oluşur: yönlendirme, dağıtma ve şartname. Yönlendirme, alt katmandaki ağ cihazlarının detaylarını gizleyerek ağ uygulaması tarafından istenen yönlendirme davranışına müsaade eder. Dağıtım, YTA uygulamalarını ayrık yapının karmaşıklığından korur. Yönlendirme aygıtları üzerine kontrol komutlarının kurulumundan sorumludur ve ağ uygulamalarına ağın global

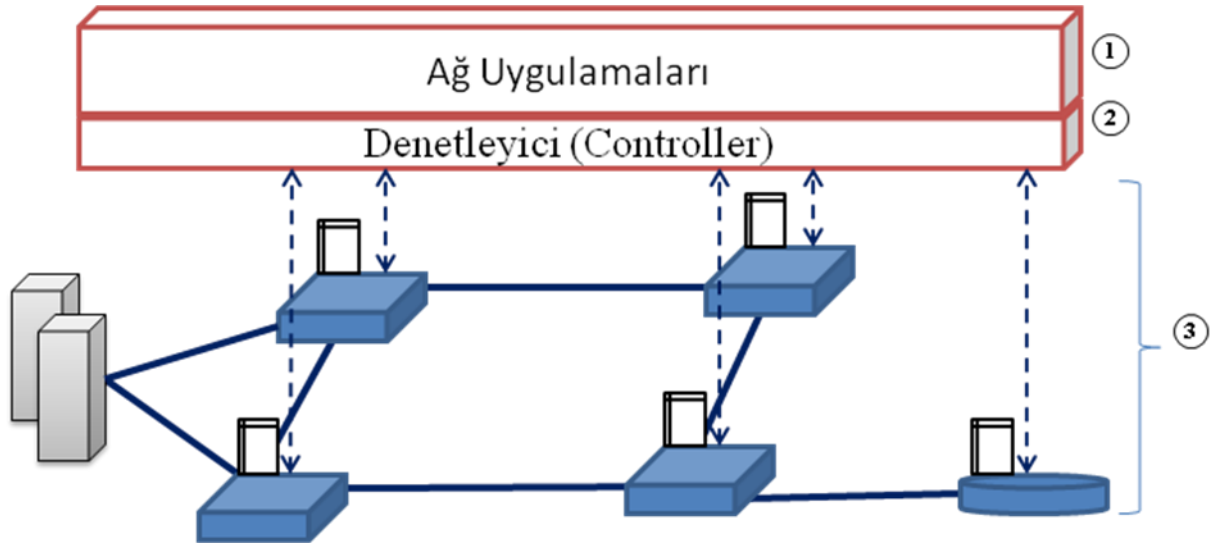
durumunu aktarmak üzere durum bilgisi toplar. Şartname ise, bir ağ uygulamasının istenen ağ davranışını, o davranışı uygulamaktan sorumlu olmayarak, belirtmek içindir.

Kontrol mantığının merkezileştirilmesinin fiziksel olmak zorunda olmadığı Bölüm 2.1.2'de belirtilmiştir. Merkezileşmenin mantıksal gerçekleştirilmesinin diğerine göre bir kısım faydaları bulunmaktadır. Ağ ilkelerinin alt seviyede fiziksel cihaz üzerinde değiştirilmesi cihazın özel konfigürasyonuna hâkim olmayı gerektirir. Bu, hataya çok açıktır. Yüksek seviye programlama dilleri ile gerçekleştirilen değişiklikte ise, fiziksel konfigürasyonu yazılım yönettiği için çok daha iyi başarımla gerçekleştirilebilir. Mantıksal merkezियette, kontrol programı ağ durumundaki istenmeyen değişikliklere direkt karşılık verebilir. Böylece yüksek seviyedeki ilkelerin bütünlüğü korunmuş olur. Ağ durumunun global bilgisini barındıran denetleyici, daha gelişmiş ağ fonksiyonlarının geliştirilmesini sadeleştirir.

Bu tez kapsamında YTA katmanları ayrımında üç ana grup kullanılacaktır: Veri düzlemi, kontrol düzlemi ve yönetim düzlemi. Bunlar [15] esas alınarak alt kategorilere ayrılmış olup, ana katmanların geleneksel ağlar ve YTA'lardaki gösterimi Şekil 2.3 ve Şekil 2.4'te verilmiştir. Yönetim düzlemi ağ uygulamaları, programlama dilleri ve dil tabanlı soyutlamadan oluşur. Kuzey-bağlam arayüzü, ağ işletim sistemi ve ağ hipervizörü kontrol düzlemini oluşturur. Veri düzleminde ise, güney-bağlam arayüz ve ağ altyapısı bulunur. Alt kategorilerden hipervizör ve dil tabanlı soyutlama her YTA yapısında bulunmak zorunda değildir.



Şekil 2.3: Geleneksel ağ. 1: Ağ uygulamaları. 2: Kontrol düzlemi. 3: Veri düzlemi. [15]



Şekil 2.4: YTA. 1: Ağ uygulamaları. 2: Kontrol düzlemi. 3: Veri düzlemi. [15]

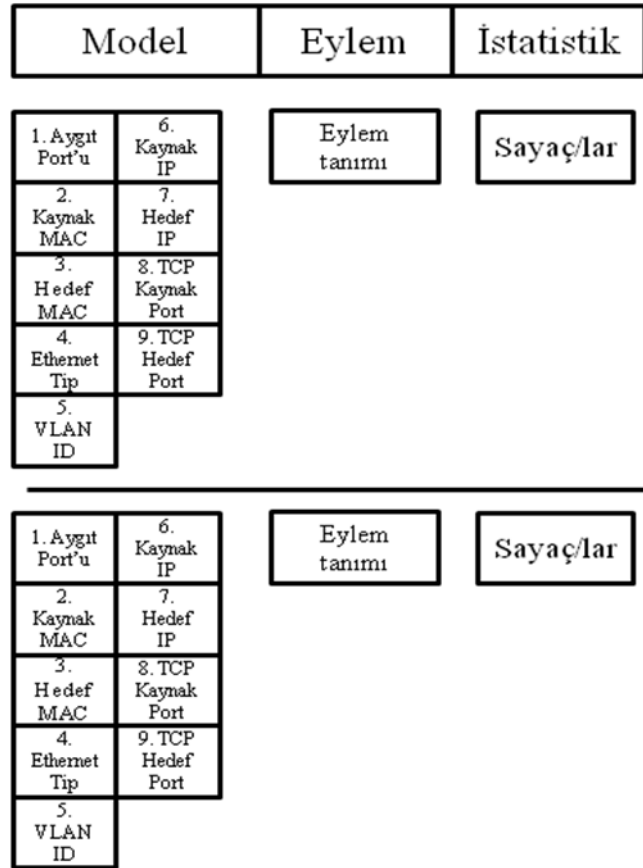
2.1.3.1 Veri Düzlemi

YTA'da veri düzlemi fiziksel ağ altyapısı ve güney-bağlam arayüz olmak üzere iki ana parçadan oluşur. Fiziksel ağ altyapısını oluşturan ağ aygıtları, yalnız yönlendirme işlevi görürler. Bunlar standart arayüzler üzerine inşa edilmiştir. Bu şekilde, farklı veri ve kontrol düzlemi cihazlar arası iletişim, kurulum ve düzenleme bağı sağlanmış olur. Denetleyici birimin heterojen aygıt yönetimini sağlaması da standart arayüzler sayesinde olur, ve bunlar güney bağlam arayüzü oluşturur.

Her arayüz için yeni bir ağ aygıtının tasarımdan pazarlamaya kadar olan sürecinin aylar alabileceği ve böyle bir yatırımın yüksek miktarda finansman ve risk alma gerektirdiği düşünüldüğünde standart bir arayüz gerekliliği daha iyi idrak edilebilir. Bu tezde kullandığımız güney-bağlam arayüz olan OpenFlow gibi standartlar, farklı üreticilerden aygıtların oluşturduğu ağda bunların birlikte çalışabilmesi için büyük önem taşır. Bu arayüzler en basit tabirle köprüdürler: kontrol düzlemi ile fiziksel aygıtları birbirine bağlarlar.

Bir OpenFlow-etkin yönlendirme cihazında akış tabloları bulunur. Akış tablolarında her girdinin üç parçası vardır: eşleme kuralı, eşelenen eylem, eşleşme sayacı. Tabloların bu yapısı, yüksek seviyede basitleştirilmiş bir model sunar. Birçok ticari YTA aygıt cihazının bu protokolü kullanmasının nedenlerinden biri budur.

OpenFlow'da denetleyici, paketler ve ilgili talimatlar için modellerin saklandığı akış tablolarının güncellemelerinden sorumludur. Denetleyici, Şekil 2.5'te içerik şeması gösterilen bu tabloları önceden doldurabilir veya ağ çalışır halde iken dinamik olarak yapabilir. Bir paket bir yönlendirme aygıtı tarafından alındığında, bunu ilk önce akış tablosunda arar. Akış eşlemeleri çeşitli kombinasyonlardan oluşabilir, bit maskeleri gibi operatörler kullanılabilir. Bir eşleşme bulunana veya tüm tabloları tükenene kadar, gelen paket modeli tablolar arasında aranır. Bir eşleşme bulunduğu durumda, eşleşen girdiye tahsis edilen eylem gerçekleştirilir. Örneğin, trafik yükünün dengelenmesi amacıyla tutulan bir sayacın güncellenmesi bir eylemdir. Her paket modelinin tek bir eyleme eşlenmek zorunda olunmadığını belirtmemiz gerekir; bir modele bir dizi eylem eşlenebilir. Eğer tablolarda eşleşen bir girdi bulunmazsa, paket için gerçekleştirilecek eylem denetleyiciye sorgu göndererek öğrenilebilir, paket düşürülebilir ya da hiçbir eylemde bulunulmayabilir. Tüm bu seçenekler, üst seviyedeki programlamanın direktiflerine bağlıdır.



Şekil 2.5: Akış tablosu bölümleri.

Üst seviye programlama direktiflerini içeren denetleyici, daha sonra ayrıntılı inceleyeceğimiz kontrol düzleminde bulunur. OpenFlow protokolü ağın üst seviyelerine bir takım bilgi iletiminde bulunur. Bir link ya da port değişikliği olduğunda yönlendirme aygıtı, denetleyiciye durumu içeren mesaj gönderir. Oluşan istatistikler de yine denetleyicinin bu düzlemden çektiği bilgidir. Diğer bir bilgi oluşturan durum akış modeli ile ilgilidir. Model tabloları eşleşmiyor, ya da denetleyiciye gönderilme eylemiyle eşleşiyorsa üst katmana bilgi iletimi yapılır.

2.1.3.2 Kontrol Düzlemi

Ağ hipervizörü, ağ işletim sistemi ve kuzey-bağlam arayüz kontrol düzlemini oluşturan bileşenlerdir. Hipervizör sanallaştırma ile ilgilidir, ve her YTA mimarisinde kullanılmak zorunda değildir. Sanal makine konseptinde birden fazla işletim sisteminin ana işletim sistemi üzerinde çalıştırılması için ara katman oluşturan hipervizör, YTA'da da benzer görev üstlenir. Özellikle bulut uygulamalarında, her kullanıcı, ana bir kaynaktan

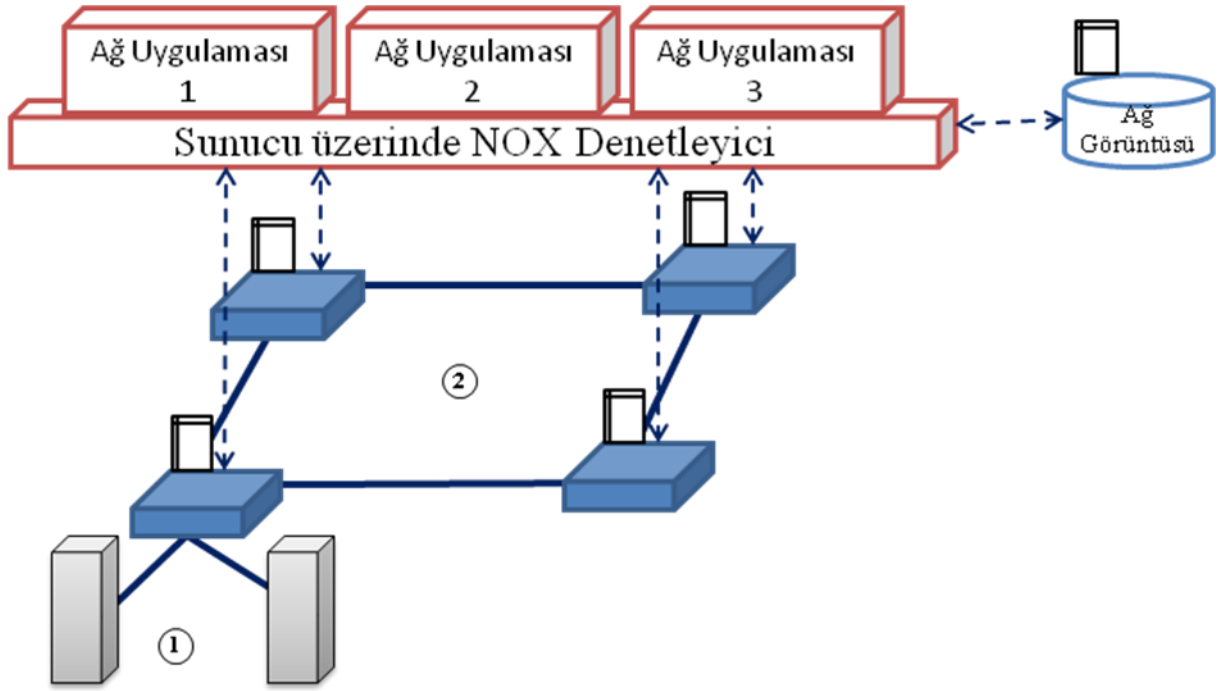
hesaplanıp ayrılan kendi sanal kaynağına sahiptir. Hipervizör, farklı sanal makinaların aynı ana donanım kaynağını paylaşabilmesini sağlar.

Hipervizör gereksinimlerinde dikkat edilmesi gereken bir husus topolojidir. Çeşitli iş yükleri farklı topoloji ve servisler gerektirir. Geleneksel ağlarda tek bir fiziksel topolojinin böyle çeşitli ihtiyaçları karşılaması çok zordur. Adres alanı da aynı şekilde, tek fiziksel topoloji üstüne kurulu sanal ağlar için farklılaştırılması kolay olmayan bir özelliktir. Durum böyle olunca, orijinal topoloji tek bir müşteri ihtiyacına yönelik değiştirilememekte, ya da kolayca taşıma işlemleri gerçekleştirilememektedir.

İdeal olarak ağ alt yapısının üzerine kurulu sanal yapılarda keyfi ağ topolojileri ve adresleme prosedürleri desteklenebiliyor olmalıdır. Dahası, her bir müşteriye eş zamanlı olarak bu destek verilebilmelidir. Taşıma işlemlerinde, ilgili sanal port'lar da otonom bir şekilde ayarlanabilmelidir. Tüm bunları desteklemeyen geleneksel ağ yapısında, taşıma işlemleri aylar alabilmektedir. Bulut sistemler için yapılan son yılda yapılan YTA mantığının ideal yapıyı sağlaması yönündeki çalışmalar umut vericidir.

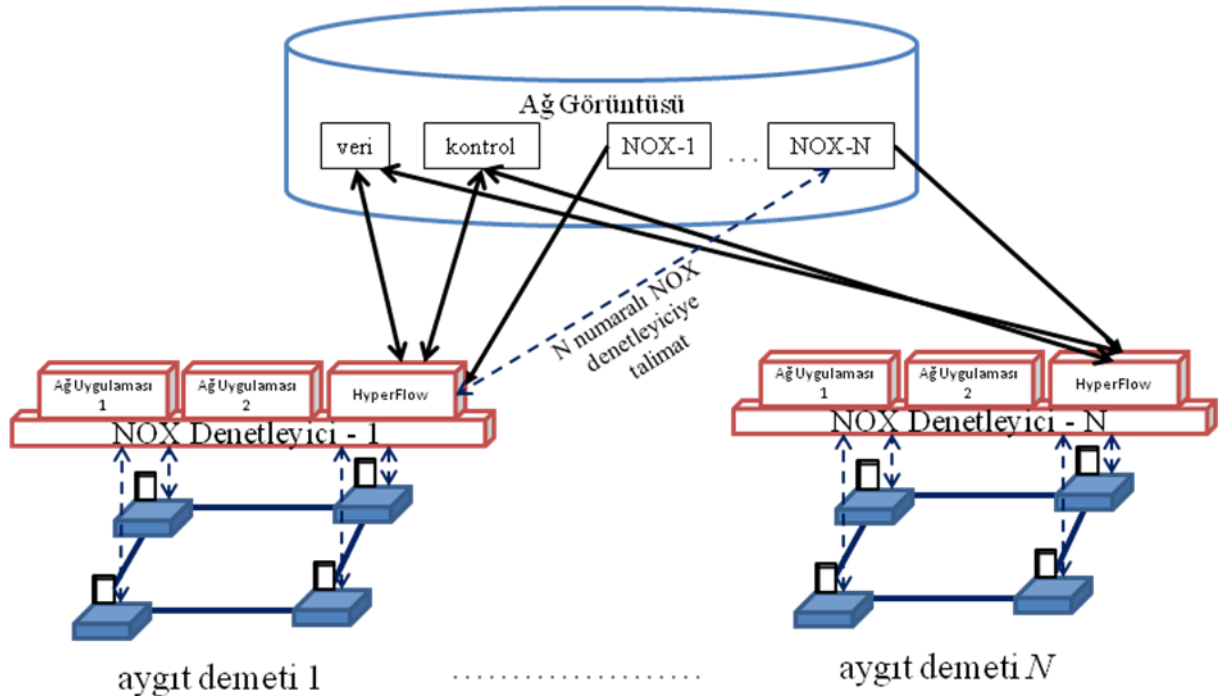
Kontrol düzlemini oluşturan ikinci bileşen ağ işletim sistemi, yani denetleyicidir. Bu bir nevi bilgisayar işletim sistemlerine benzer; üst seviye programlama dilleri ve uygulamalar için alt seviye donanıma erişimi ve kaynak güvenliğini sağlar. Geleneksel ağlarda bu işlev ağ aygıtları üzerinde, alt seviyede gerçekleştirilmektedir. Bundaki önemli bir dezavantaj, aygıtların kendine özgü komut grupları olması yüzünden standartlaşmanın önüne geçmesidir.

YTA, ağ yönetiminin kolaylaştırılması ve problemlerin çözümünde merkeziyet sağlar. Ağ işletim sistemlerinden biri olan NOX [17], OpenFlow protokolü ile birlikte kullanılan en yaygın işletim sistemidir. Genel yapısının Şekil 2.6'da görüldüğü NOX denetleyici ağında, ağ cihazına akış tablolarında eşlenen paket alınması durumunda ilgili sayaçlar güncellenir ve eylemler gerçekleşir. Eşleşmeyen paketler ise direkt denetleyiciye gönderilir. Bunlar genelde bir akıştaki ilk paketlerdir ve NOX bunları ağı izlemek amaçlı global görüntü oluşturmak ve o paketle başlayan akışın hangi rotadan gönderileceğini kontrol amacıyla kullanır.



Şekil 2.6: NOX denetleyici ağı. 1: Kenar istemciler. 2: OpenFlow anahtarlar. [17]

Ağ işletim sisteminin önemli özellikleri arasında üst seviye soyutlamanın gerçekleştirilmesi, geliştiricilere programlama arayüzleri sunması (API) ve ağ operatörü tarafından tanımlanmış ağ konfigürasyonunun desteklenmesi bulunur. Bunlar merkezi ya da dağıtık yapıda bulunabilir. Floodlight [18] gibi merkezi işletim sistemlerinde, tek bir eleman ağdaki tüm yönlendirme aygıtlarını yönetir. Dağıtık sistemler daha dinamiklerdir. Çeşitli çevre koşullarının gereksinimlerine ölçeklenebilir, düğümler demeti ya da fiziksel olarak dağılmış ağ elemanları grubu olarak modellenebilir. HyperFlow [19] örneğin, her demette NOX denetleyici kullanır. Her bir denetleyici, sanki tüm ağı yönetiyormuş gibi, ağın tüm görüntüsüne sahiptir. Şekil 2.7'de görüldüğü üzere, yönlendirici aygıtların her biri kendine en yakın denetleyiciye bağlanmıştır. Denetleyicilerden biri arızalanırsa, bağlı aygıtlar en yakın aktif denetleyiciye bağlanmak üzere ayarlanmıştır. Her denetleyici, kendine bağlı aygıtı yönetir ama diğer denetleyicilerle iletişim halinde olarak dolaylı bir şekilde diğer aygıtları da görür.



Şekil 2.7: Bir HyperFlow demetinden diğerine veri yönlendirme şeması. [19]

Neredeyse tüm denetleyici yapılarında üç katman bulunur. Birincisi uygulama, idare etme ve servislerdir. Yönetim ve kuzey-bağlam arayüz uygulamalarıyla iletişim kurar. İkincisi, çekirdek denetleyici fonksiyonlarıdır: topoloji, istatistik, bildirimler ve aygıt yönetimi, en kısa yol algoritmaları, güvenlik mekanizmaları. Son katman ise güney-bağlam iletişim sağlayıcısıdır. Bu, kişisel bilgisayarlardaki aygıt sürücüsü yönetimi gibi işlev görür, farklı güney-bağlam uygulama arayüzlerinin kullanımına olanak tanır [15].

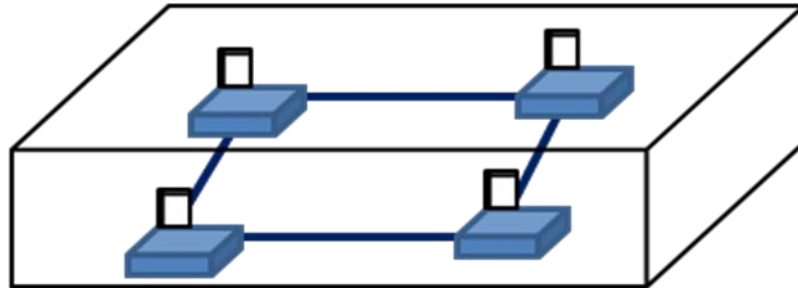
Kontrol düzlemini oluşturan son bileşen kuzey-bağlam arayüzdür. Bir önceki alt bölümde de değinildiği üzere, güney-bağlam arayüzler için standartlaştırma çalışmaları yapılmış ve birçok üretici OpenFlow protokolünü kabul etmiştir. Standart bir kuzey-bağlam arayüzde ise henüz geniş çaplı bir anlaşmaya varılamamıştır. [20]'de yapılan değerlendirmeye göre kuzey-bağlam arayüzlerde bir mutabakata varmak için henüz erkendir. Rothenberg ve diğer. [21], Açık Ağ Birliğinin 2014'te ele almaya başladığı [22] söz konusu standartlaşma başlamadan önce kuzey-bağlam arayüzlerin dikey-yönelimli geliştirilmeleri ele alınmalıdır. Birliğin çalışmasında, kuzey-bağlam arayüzlerin ağ kaynaklarını tanecikli ve dinamik kontrolüne müsaade eden kullanıcı

uygulamalarının geliştirilebilmesi senaryosu bulunmaktadır. Bu senaryo, YTA'ların sınırlarını genişletme potansiyeline sahip olması yönüyle önemlidir.

2.1.3.3 Yönetim Düzlemi

Dil tabanlı soyutlama, programlama dilleri ve ağ uygulamaları yönetim düzlemini oluşturur. Bileşenlerden dil tabanlı soyutlama tüm YTA mimarilerinde bulunmak zorunda olmamakla birlikte, günümüz sanallaştırma mimarileri için büyük önem taşır. Sanallaştırma, modülerite ve aşamalı sanallık boyutları uygulamaları için kaçınılmaz bir yöntem olmaya başlamıştır.

Blok diyagram ile Şekil 2.8'de gösterilen "tek büyük anahtar" modeli, birçok yönlendirme aygıtını içinde barındıran sanal bir aygıttır. Bu şekilde sağlanan bir üst seviye kapsülleme sayesinde, uygulama geliştiricileri birbiriyle etkileşen yönlendirme aygıtlarının alt seviyedeki işleyişlerine hâkim olmak zorunda kalmaz.



Şekil 2.8: Sanal yönlendirme aygıtı: "Tek büyük anahtar".

Topolojinin üst seviyede soyutlaştırılmasına bir örnek uygulama Pyretic'tir [23]. Programcıların denetleyici uygulamaları yazmalarına kolaylık getirmek amacıyla geliştirilmiş projede, tıpkı "assembly" dilinde uğraşmak yerine üst seviye bilgisayar programlama dillerinin kullanımında olduğu gibi Pyretic'in benzer kolaylık sağladığı belirtilmiştir. Frenetic [24] dil projesine bağlı olan bu uygulama, basit, yeniden kullanılabilir, yüksek seviye sanallaştırma ile YTA'larda alt seviyedeki aygıtlar üzerinde otomatik olarak kuralların oluşturulması ve kurulması yapılabilmektedir. Bu dil tabanlı sanallaştırma uygulaması ile paket yönlendirme ilkeleri yazılabilir, ağ durumu izlenebilir ve ağ olaylarına göre dinamik ilke güncellemeleri yapılabilmektedir. Python tabanlı bir platform olan Pyretic, Python dilinin birçok özelliğini üzerinde

barındırmaktadır. Bu özelliğin programcılara gelişmiş uygulamalar yazma imkânı tanıdığı belirtilmiştir.

Başka bir dil-tabanlı soyutlama statik bölümlleme olarak görülür. Bölümlleme, derleyici tarafından, uygulama katmanı direktiflerine uygun olarak gerçekleşir. Böyle bir uygulamada dinamik kontrolü gerçekleştiren hipervizör bileşenine ihtiyaç duyulmaz. Topoloji bölümlere ayrılır, her bölümdeki aygıtların haritası çıkarılır ve her paket bir bölüme atanır.

Yönetim düzleminin ikinci bileşeni programlama dilleridir. OpenFlow'un alt seviye makine diline karşılık geldiğini düşünürsek, bu düzlemdeki programlama dilleri C++ gibi makine dilini iç mekanizmasında idare eden üst seviye dilleri ifade eder. Bunlardan biri olan Frenetic [24], dağıtık ağ aygıtları için tasarlanmıştır. Ağ trafiğinin sınıflandırılması için deklaratif sorgular ve yüksek seviye paket yönlendirme ilkeleri için fonksiyonel kütüphane sunmaktadır. OpenFlow seviyesinde fark alma işlemi bile birden fazla kural ve önceliğin doğru bir şekilde idare edilmesi gerekirken, Frenetic bunları kendi bünyesinde yapar. Yine alt seviyede uygulanması zor olan maskelenmiş kurallar bu programlama dilinde üst seviye komutlar ile gerçekleştirilebilir. Denetleyiciler kendilerine yönlendirme aygıtlarından gönderilen paketleri işlerler. Bu yüzden, denetleyici seviyesinin altında bulunan güney-bağlam arayüzlerde bu paketler için program yazmak oldukça zor ve hataya açıktır. Alt seviyede idare edilmesi zor olan diğer bir husus, dağıtık yapıdaki yönlendirme aygıtlarına gelen paketlerin öncelik sıralamalarıdır. Frenetic gibi genel kullanım için oluşturulmuş diller ve NetCore [25] gibi paket yönlendirme ilkeleri yazmak için özelleştirilmiş diller ile bu işlev çok daha rahat ve hatasız gerçekleştirilebilir. YTA'larda üst seviye programlama dilleri ve özelleştirilmiş kullanım alanları için [15]'teki Tablo 8 kapsamlı bir özet oluşturmaktadır.

Bu düzlemdeki son bileşen ağ uygulamalarıdır. Bunlar ağda temel ve özel fonksiyonelliklerin sağlanması için geliştirilmektedir. Veri düzlemine komutlar olarak iletilip kurulmak üzere oluşturulan kontrol mantığını oluşturur.

YTA'lar geleneksel ağ yapısına yerleştirilebilir ve hem geleneksel hem de daha karmaşık fonksiyonlar gerçekleştirebilir (Tablo 2.1). [26]'ya göre ağ uygulamalarının çeşitlilik potansiyeli, YTA'nın yakın zamanda geniş kitlede kabul görmesinin temel

nedenidir. HP firmasının YTA uygulama mağazası [27,28] diğer aygıt üreticilerine örnek teşkil edecek niteliktedir. HP'nin OpenFlow denetleyicisini satın alan müşterileri mağazaya erişebilmekte, şirket ve diğer kullanıcıların geliştirdiği ağ uygulamalarını dinamik olarak indirip denetleyicilerine kurabilmektedirler. Mağazada yalnız ticari ürünler değil, ticari olmayan geliştirici topluluklarının da geliştirdiği uygulamalar yer almaktadır.

Tablo 2.1: Geleneksel ve karmaşık ağ fonksiyonları.

Geleneksel Ağ Fonksiyonları	Karmaşık Ağ Fonksiyonları
Yönlendirme	Güç tüketimini azaltma
Yük dengeleme	Yük devretme (fail over) ve güvenilirlik
Güvenlik ilkeleri yaptırımı	Uçtan uca servis kalitesi yaptırımı
	Ağ soyutlama
	Kablosuz ağlarda taşınabilirlik yönetimi

Kablosuz ağlarda taşınabilirlik yönetimi, güncel kullanım alanı olması nedeniyle, üzerinde önemli miktarda araştırma yapılmıştır. Kablosuz ağların geleneksel yapısındaki dağıtık kontrol düzlemi yapısı, kablosuz kaynakları bölüştürmesi, devir teslim işlemlerinin uygulanması ve etkili yük dağılımı için yetersiz kalmaktadır. YTA farklı özelliklere sahip kablosuz ağların merkezi denetleyici sistemi sayesinde yönetiminin kolaylaşması açısından önemlidir.

Başka uygulamalara fonksiyonellik sağlayan ağ uygulamaları geliştirilebileceği gibi, YTA özelliklerini geliştirmeye yönelik uygulamalarda getirilebilmektedir. Güvenlik üzerine uygulamalar hem geleneksel ağ açıklarını kapatmaya yönelik hem de YTA'nın kendisinin güvenliğini geliştirmek üzere yazılmaktadır. Ağ taşıma, ağ yönetimi, hata yönetimi, ağ verimliliğinin optimizasyonu için veri merkezi uygulamaları yazılmaktadır. Bu tezde ele aldığımız trafik kontrolü için ise özellikle tıkanıklık oluşumunun engellenmesi yönünde uygulama yazılmaktadır.

2.2 YAZILIM TANIMLI AĞLARDA TRAFİK VE TIKANIKLIK KONTROLÜ

Bölüm 2.1'de ayrıntılı olarak ele alınan YTA özellikleri, ağ trafik izleme ve yönetiminde geleneksel ağlara karşı avantajlı durumdadır. Geleneksel ağlara uygulanan trafik çözümlerinin yetersiz kalma sebepleri arasında günümüz internet uygulamalarının ağ mimarisinin gerçek zamanlı tepki vermesini gerektirmesi ve çok yüksek miktardaki trafik için ölçeklenebilir olması ihtiyacıdır. Buna bağlı olarak gelen başka bir gereksinim ise kaynak veriminin yüksek seviyede izlenmesi yönündedir.

Trafik mühendisliği hem trafik yönetiminin, hem kaynak dağıtımının etkin gerçekleştirilmesi üzerine ağ trafiğinin izlenmesi ve ölçülmesini kapsar. Ağ düğümleri arasındaki bağlantının yeni gelen veriyi taşıyamayacak kadar yoğun olması durumu tıkanıklığı oluşturur. Tıkanıklık kontrol şemaları, tespit yöntemine göre tepkici ve önleyici olarak ikiye ayrılır. Tepkici yöntemler, tıkanıklığın ilk tespiti anında kaynak düğüme trafik akışını azaltması yönünde talimat verme yönelimindedir. Önleyici yöntemlerde ise, tıkanıklığın gerçekleşmesi beklenmez. Giriş kontrolü yapılarak yeni bir bağlantının kabul edilip edilmeyeceği konusunda servis kalitesi gereksinimlerine göre karar verilebilir. Buna alternatif olarak bant genişliği yaptırımı getirilebilir. Buna göre, ağ trafiğinde büyük bant işgal eden büyük veri akışları tespit edilir ve bununla ilgili nasıl bir yöntem izleneceğine karar verilir. Yine başka bir önleyici yöntem trafik sınıflandırmasıdır. Sınıflandırmadan kasıt, veri akışlarına öncelik sıralaması verilmesidir. Öncelikler iki şekilde kullanılabilir. Birincisi acil olan akışlara işlem önceliği verilmesidir, diğeri önem derecesi düşük olan akışların tıkanıklık durumunda düşürülmesidir.

Yapılan bir çalışmada[29] geleneksel ağlardaki tıkanıklık kontrolü yavaş başlatma, tıkanıklıktan kaçınma ve hızlı yeniden iletim ve hızlı toparlanma olarak sıralanmıştır. Birlikte kullanılan yöntemlerden olan yavaş başlatma ve tıkanıklıktan kaçınma için her bağlanma durumuna iki değişken eklenir. Tıkanıklık penceresi (cwnd) göndericinin sınırı, söz verilen pencere (rwnd) ise alıcı için veri sınırını belirler. cwnd-rwnd ikilisinin yanında, yavaş başlatma eşiği (sssthresh) kullanılarak veri akışının düzeni belirlenir. Yavaş başlatma, cwnd'nin basamaklı olarak artırılmasını içerir ve veri akışının en başında ya da çok sayıda veri kaybı olduğu zaman devreye girer. Böylece veri gönderim boyutu ve zamanlamasını ayarlama görevi görür. Hızlı yeniden iletim metodu, ACK

durumuna bakarak gönderilen segment'in kayıp olduğuna karar verir ve zaman aşımını beklemeden segmenti tekrar gönderir. Bu durumda hızlı toparlanma devreye girer ve ACK durumu düzelene kadar yeni veri akışını kontrol eder.

YTA denetleyicileri farklı servis kalitesi gereksinimine sahip trafik akışlarını desteklemek için çeşitli tıkanıklık kontrol ve trafik yönetim prosedürleri içermeli, yük dengesi ve servis kalitesi sağlama mübadelesini iyi yapmalıdır [30].

2.2.1 Yazılım Tanımlı Ağ Trafik Mühendisliğine Genel Bakış

YTA'nın temel akış yönetimine göre, yönlendirme aygıtına gelen veri akışı, tablosundaki hiçbir modele eşlenemiyorsa denetleyiciye gönderilir. İşlem, akışın ilk paketinin aygıt-denetleyici kanalıyla üst katmana iletilmesiyle başlar. Denetleyici, akış için bir yönlendirme yolu hesaplar. Ardından, denetleyici ilgili akış tablolarına planlanan yol ile ilgili güncelleme yapar. Son olarak, akıştaki takip eden tüm paketler - ve eğer o şekilde akış girdisi yapılmışsa diğer başka paketler- veri düzlemine iletilir.

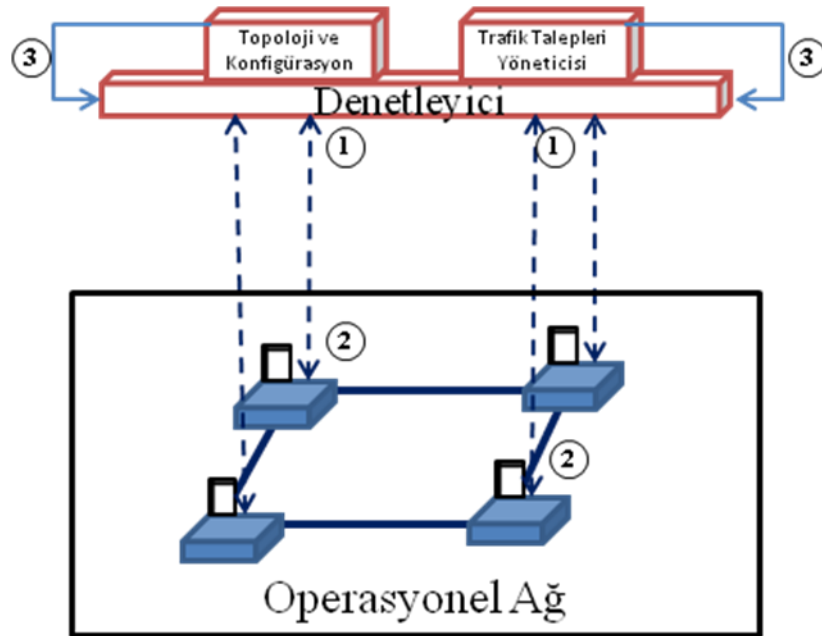
Bu işlem sırasında, eğer toplam trafik yüksek miktarda yeni (tablolarda tanımsız) akıştan oluşuyorsa kontrol ve veri düzleminde ciddi yoğunluk olacaktır. Yeni yönlendirme kuralının oluşturulması da zaman alan bir işlemdir. Bu da gecikmeyi artıracaktır. Bu nedenle, akış yönetimi gecikme ve yük dengesi dikkate alınarak gerçekleştirilmelidir.

Geleneksel ağlarda olduğu gibi YTA'da da hatadan hızlı geri dönme zorlu bir problemdir. Merkezi ağ yönetimi işi biraz kolaylaştırırsa da denetleyicide hata sonucu oluşan rota değişikliklerinin hesaplanması ve ilgili tüm aygıtların anında haberdar edilmesi gerekliliği bulunmaktadır. Bu yapılırken de sınırlı bellek ve akış tablo kaynakları göz önünde bulundurulmalıdır.

Topoloji güncelleme mekanizması planlı değişiklikleri kapsamaktadır. Bunlar ilke değişikliği gibi güncellemelerdir.

YTA'ların geleneksel ağların üzerine kurulabiliyor olması ve aygıt üreticilerinin saf YTA ürünler yerine geleneksel aygıtların YTA etkin port'lar ile satışını yapmaları trafik mühendisliğinin büyük oranda geleneksel ağlardan kalıtsal yolla alındığını göstermektedir (Şekil 2.9). [30]'da sıralandığı üzere YTA'da mevcut trafik mühendisliği

mekanizmaları dört başlık altında toplanır: Akış yönetimi, hata toleransı, topoloji güncelleme ve trafik analizi. Akış yönetimi yönlendirme aygıtı seviyesinde yük dengeleme, denetleyici seviyesinde yük dengeleme ve birden fazla akış tablosu kullanma yöntemlerinden oluşur. Hata tolerans sistemi veri ve kontrol düzleminde ayrı ayrı gerçekleştirilebilir. Topoloji güncelleme ise yeni ilkelerin eklenmesi sırasında yönlendirme aygıtlarında çifte kayıt oluşması ve zaman tabanlı konfigürasyon yöntemlerinden oluşur. Trafik analizi; ağı izlenmesi, değişkenlerin takibi ve programlama hatalarını ayıklamaktan oluşur. Tez kapsamında akış yönetimi önem arz ettiği için, bu ayrı başlıkta incelenecektir.



Şekil 2.9: YTA trafik mühendisliği bileşenleri. 1: Ölçümler. 2: Kontrol. 3: Optimizasyon

Geleneksel ağlardan taşınan metotlar, yüksek izleme gideri ve aygıt kaynak tüketimine yol açar. Curtis ve diğ. [31], bunun önüne geçmek için büyük akışların üç sunucuda tespit edilmesi gerektiğini savunmuştur.

2.2.2 Akış Yönetimi

YTA'da OpenFlow yönlendirici aygıtın gelen pakete karşılık akış tablosunda eşleşen bir girdi bulmadığında bunu denetleyicisine gönderdiği daha önce de belirtilmiştir. Genelde bu paket akışın ilk paketidir ve denetleyici bunun alanlarına bakarak o akışa ya da o akış tipine ne yapılacağına karar verir. Ardından yeni bir akış tablosu girdisinin aygıtlara

eklenmesi talimatı verilir. Eğer yüksek miktarda yeni akış aygıtlarda dizilmişse, kontrol ve veri düzlemi şişebilir.

Bunun önüne geçmek amacıyla üç grup yöntemle başvurulur: aygıt yükü dengeleme, denetleyici yükü dengeleme, çoklu akış tabloları. Akış karması (hashing) yöntemi [32], akışın müsait ağ yolları arasında paylaştırılmasıyla gerçekleşir. Bu yöntem ile programlanmış yönlendirme aygıtı talimatları, bir alt ağda birkaç uygun yönlendirme yolunu içinde barındırır. Bu yöntemin önemli bir kısıtı, bir ya da iki uzun süre ile aygıtı meşgul eden büyük akışın gelmesi durumunda karmaların çakışabilmesi ve aynı tıkanıklık durumunu oluşturabilmesidir.

Aygıt yükünün dengelenmesi için uygulanan bir diğer yöntem, bit maskeleyedir [11]. OpenFlow versiyon 1.0-1.1 32 bit, 1.2-1.3 ise 64-bit maskeleye karakteri kullanımına müsaade eder. Maskeler, benzer model oluşturan akışların tek komutla yönetilmesini sağlar. Bu yöntem tek bir merkezi denetleyici gerektirdiği için ideal değildir. [33]'te belirtildiği üzere, kritik yolların hepsindeki aygıtlara bağlı olup maskeli kuralları işleyecek olan böyle merkezi bir denetleyicide tıkanıklık oluşumu kaçınılmazdır.

Denetleyici yükünün dengelenmesi, çoklu denetleyicinin kullanılıp birkaç denetleyici yerleştirme şemasının barındırılmasıyla mümkündür. Denetleyicilerde mantıksal dağıtıklık, fiziksel dağıtıklık, hiyerarşik yerleşim ve hibrit yerleşim olarak dört grupta yerleşme şeması tasarlanabilir. HyperFlow [19] mantıksal dağıtıklığa örnek bir ağ uygulaması katmanında yönetilen şemadır. HyperFlow ile her bir denetleyicinin karar mekanizmasını uygulama merkezinde toplayarak, kontrol düzleminin veri düzlemine cevap süresini kısaltır. Bu uygulama önemli ölçeklendirme sağlar.

Denetleyicinin fiziksel olarak dağıtık yapıda tutulmasını Onix [34] platformu sağlar. Bu platform birden fazla fiziksel denetleyici cihazına kurulu bir ağ uygulamasıdır. Topolojinin tüm eleman bağlantısını belirten ağ global görüntüsü veri yapısında tutulur. Her denetleyicide aynı ağ görüntüsü paylaşılmak suretiyle yeni akışlar için yol hesaplamaları ilgili fiziksel denetleyicilerde yapılır.

Hiyerarşik yerleşimde, her bir yönlendirme aygıtı tek bir denetleyiciye bağlı, her denetleyici birden fazla ağ aygıtını kontrol edebilir durumda ve denetleyiciler üst seviye

denetleyicilere bağlıdır [35]. Hibrit denetleyici yerleşimleri bu üç yerleşim yapısının en az ikisinin birlikte kullanılmasıyla sağlanır.

Denetleyicilerde oluşabilecek tıkanıklığın önüne geçmede kullanılan üçüncü yöntem çoklu akış tablolarıdır. OpenFlow'un 1.1. versiyonundan itibaren uygulanan bu mekanizma, her akış girdisinin öncelikli olarak eşleneceği tablo ataması ile kurulmuştur. Aygıt önce gelen akışın en yüksek öncelikli olarak eşleneceği tabloyu belirler, sonra ilgili eşlenen alandaki eylemi gerçekleştirir. Eylem, bir başka tabloya geçilmesi olarak atanmış olabilir. Böylece tablo eylemleri işlem sırasına sokularak akışa uygulanır.

2.2.3 Ağ Uygulaması Seviyesinde Yazılım Tanımlı Ağ Trafik Analizi

Trafik analizi ve yönetiminin büyük kısmı uygulama seviyesinde gerçekleştirilir. Bölüm 2.2.2'de belirtilen akış yönetiminin yanı sıra hata toleransı, topoloji güncellemesi ve trafik sınıflandırması önemli uygulama konularıdır. Özellikle veri merkezlerini ilgilendiren güç tüketimi konusu yine trafik analizi altında incelenir. Burada temel dayanak noktası hiyerarşik ağda toplam ağ veriminin artırılmasıdır. Yükün dengelenmesi hem veri hem de kontrol düzleminde gerçekleştirilebilir.

Yine veri merkezleri için sorun oluşturabilecek konu hesaplama yapılarıdır. Özellikle bulut servislerinde müşterilere sağlanan kaynakların hesaplanması sırasında yapılan işlemlerin gecikmesi durumunda, ilgili akışların ağda tıkanıklığa yol açması muhtemeldir. Bunun önüne geçilmek için kullanılan yöntemlerden biri, trafik örüntülerine doğrusal (lineer) ayrımlar uygulamaktır. Bant genişliğinde uygulanan doğrusal ayrımlar, ağı yoran trafik grupları için kullanılır. Bir diğer yöntem tahminleme ile ilgilidir [36]. Burada temel alınan yaklaşım hangi tip akışın hangi yollardan gittiğinin haritalamasını gösteren talep tahminleyicidir.

Trafik analizi, her türlü ağda incelenmesi gereken önemli bir durumdur. YTA yaklaşımı kapsamında yeni uygulamalar ve gelişmelerin yakın zamanda patlak vereceği öngörüsünün yüksek doğrulanma olasılığı taşıdığını görmek açıktır. Biz de tez kapsamında bu konuya eğilerek makine öğrenmesinin tıkanıklık kontrollü ağ uygulamalarında kullanılmasına öncülük edebilecek simülasyon ve hesaplama çalışmamızı gerçekleştirdik.

2.3 MAKİNE ÖĞRENMESİ

Yapay zekâ terimi literatürde ilk olarak 1955 yılında John McCarthy'nin bir araştırma konferansındaki yayınında “öğrenmenin her yönü ya da zekânın herhangi diğer bir özelliği prensipte öyle kesinlikte tanımlanabilir ki bir makine tarafından simüle edilebilir ” ifadesi ile ortaya konmuştur [37]. Makine öğrenmesi kavramı ise takip eden yılda Ray Solomonoff'un araştırma çalışmasında [38] “tümevarımsal çıkarım makinesi” terimi kullanılarak açıklanmıştır. Çalışmadaki tanımda günümüze kadar geçerlilik gösteren unsurlar geçmiş verileri kategorilere ayırma, yeni kategoriler oluşturma, gelecek veri için olabilirlik hesabı ve öngörü test etme olarak sıralanabilir. Solomonoff 1996 tarihli makalesinde [39] algoritmik olasılığın keşfine dair tanık olduğu gelişmeleri belirtmiş ve yapay öğrenmenin çıkışına dair bir nevi zaman çizelgesi oluşturmuştur.

Bu tezde makine öğrenmesi, yeterliliği sağlanmış geçmiş veri ilişkisinden genelleme ya da başka değerlendirme yollarıyla edinilen bilginin, veriyi açıklama ya da gelecek durum tahmini için kullanılması ve doğruluğunun deneysel olarak test edilmesi olarak ele alınmıştır. Takip eden alt bölümlerde makine öğrenmesine dair önemli kavram ve tanımların yanı sıra teknikler ve öğrenmenin ölçümü genel hatlarıyla incelenmiştir.

2.3.1 Önemli Kavram ve Tanımlar

Makine öğrenmesinin formal tanımını yaparken Mitchell [40] performans, görev ve deneyim ifadelerini kullanır: Makinenin öğrenmiş olması, belli bir görevin verilen deneyimden yararlanılarak seçilen performans ölçümüne göre başarısının yükseltilmesi ile sağlanır. Bu, yapay öğrenme sürecini oluşturur. Yapay öğrenme, hesaplamalı istatistik ve optimizasyon alanlarıyla ortak değerler taşır. İstatistik biliminin teori ve metodolojisi, optimizasyon fonksiyonlarının ise girdi-çıkı ilişkisini içerir.

İstatistik uygulamalarında model oluşturma, gözlemsel veri ve saha bilgisinin toplanması ile olur. Amacı, veri birikiminin oluşma sürecini anlamaktır. Uygulamalarda modeller düzenlenir ve parametre değerleri seçimi yoluyla çıkarımlar yapılır. Genelde asimptotik yaklaşımların kullanıldığı istatistik uygulamalarında alternatif bir yaklaşım olarak, bilgisayar hesaplama ve bellek kapasitesinin artmasının olumlu bir sonucu olarak modellerin simülasyonuna dayalı hesaplamalı çıkarım yöntemi kullanımı

yaygınlaşmıştır[41]. Yapay öğrenme ile ortak paydası modelleme ve simülasyon yönünü kapsar.

Bir fonksiyonun tanımlı bir girdi kümesinden kurallı bir şekilde giriş parametrelerinin seçilmesi suretiyle çıktının maksimum ya da minimum bir değerinin bulunması aşamaları optimizasyonun çerçevesini belirler. Yapay öğrenme sürecini de bir optimizasyon fonksiyonu olarak değerlendirmek mümkündür. Giriş parametreleri deneyim bilgisi, fonksiyon görev, çıkış değeri ise performans olarak eşlendiğinde optimizasyon ilişkisi kurulmuş olunur.

Veri madenciliği ve örüntü tanıma, makine öğrenmesinin başlıca uygulamalarıdır. Bilgisayar bilimleri için en eski mesleki kuruluş (1947) olan Bilgisayar Makineleri Derneği'nin (Association for Computing Machinery, ACM) özel ilgi gruplarından Bilgi Keşfi ve Veri Madenciliği (Knowledge Discovery and Data Mining, SIGKDD), yapay zekâ'nın makine öğrenmesi alt bilim dalının öğretim müfredatı için konsensüs oluşturma çalışmaları yapmaktadır. Versiyon 1.0 [42] olarak yayınlanan çalışmaya göre veri madenciliği ve örüntü tanıma arasında çok belirgin çizgi çekilmemiştir. Öğrenme aşamalarına dair birçok ortak noktası bulunan bu iki disiplinin farkı, veri madenciliğinin bilgi keşfine dair süreçleridir. Farklı formattaki veri üzerinde farklı çeşit bilginin çıkarımı ve bunun değerlendirilmesi veri madenciliği sürecini oluşturur. Veri tabanı ve veri yapıları bilgisi gerektirir ve gözetimsiz öğrenmeye daha çok odaklanır.

Makine öğrenme yöntemlerinde ortak olarak kullanılan terimler veri kümesi, veri vektörü, öznitelik ve boyuttur. Veri kümesi, yönteme girdi sağlayacak deneyim bilgisinden oluşur ve matris ile gösterilir. Her bir elemanı ayrık ya da sürekli değişkenlerden oluşabilir. Veri kümesinin her bir satırı veri vektörü olup, öznitelik olarak adlandırılan parametre değerlerinden oluşur. Bir veri kümesindeki veri vektörlerinin öznitelik sayısı, o kümenin boyutunu ifade eder. Buna göre N adet veri vektöründen oluşan D boyutlu bir veri kümesi X , Eşitlik 2.1'deki gibi gösterilir. Bazı öğrenme problemlerinde veri vektörlerinde gruplama yapılır. Gruplama sınıf etiketi atanmak suretiyle yapılır ve sınıf kümesi C ile gösterilir.

$$X = \begin{bmatrix} X_1^1 & X_2^1 & \dots & X_D^1 \\ X_1^2 & X_2^2 & \dots & X_D^2 \\ \dots & \dots & \dots & \dots \\ X_1^N & X_2^N & \dots & X_D^N \end{bmatrix} \quad (2.1)$$

Veri kümesi algoritmaya girdi verilirken üç altkümeyle ayrılır. Birincisi *eğitim kümesi* olarak adlandırılır ve ele alınan problemin bir modelinin oluşturulması için kullanılır. Modelin geçerlilik değerlendirmesini yapmak üzere *doğrulama kümesi* girdi olarak sağlanır. Eğitimin performans ölçümü ise *test kümesi* girdisi üzerinde gerçekleştirilir.

Bilgisayar ağları ve makine öğrenmesinin birlikte çalışılmasının önüne geçebilen nedenler iki disiplindeki metodoloji ve problem tanımı farkıdır. Bu tezde YTA'daki tıkanıklık problemi Bölüm 2.2'de tanımlanmıştır. Makine öğrenmesi metodolojisi de bu probleme uygun olarak ve problemin kapsamında kalacak şekilde ilerleyen alt başlıklarda açıklanmıştır.

2.3.2 Makine Öğrenme Teknikleri

2.3.2.1 Boyut Azaltma

Veri kümesi öznelilikleri kaydedilen deneyim ile ilişkili değerlerdir. Tanımlanan öznelilikler arttıkça, kümenin girdi olarak verildiği makine öğrenme algoritmasındaki hesaplama karmaşıklığı artar. Dahası bu, literatürde *boyutluluk felaketi*[43] olarak tanımlanan duruma götürür. Doğal bir durumun öğrenme problemi düşünüldüğünde, bunun sonsuz dağılımdan gelmiş olabileceği öne sürülebilir. Yani, her bir kombinasyondan özneliliğin olduğu bir veri kümesi oluşur. Buna karşın, veri vektörü sayısı sayılı olduğunda, boyut arttıkça öğrenilebilme gücü düşer. Kısaca, eğitime alınacak veri modelinin karmaşılaşmasının ya da eğitim kümesi üzerine fazla örtüşmesinin önüne geçmek gerekir.

Verinin büyük kısmını daha az öznelilik sayısı kullanarak ifade etmek üzere yapılan işlemler boyut azaltma olarak tanımlanır. X veri kümesi $N \times D$ matris formundaysa, bu $d < D$ olmak üzere $N \times d$ formuna getirilir. Bu işlemi gerçekleştirmek üzere iki ana yordam bulunmaktadır: öznelilik seçimi ve öznelilik çıkarımı. Öznelilik seçimi $F = \{f_1, f_2, \dots, f_D\}$ kümesinden d elemanın seçilmesi, öznelilik çıkarımı ise verinin doğrusal veya doğrusal olmayan yöntemlerle d boyuta izdüşümünün alınması suretiyle

d adet yeni özniteliğin oluşturulmasıdır. Tez çalışmasında kullanılan boyut azaltma yöntemi öznitelik seçimi olduğu için, bu bölümde yalnız bu yordam ayrıntılı anlatılmıştır.

Bu tür altküme oluşturma analizinde ya öznitelikler teker teker ya da altküme bütün olarak değerlendirilir. Bireysel değerlendirme her bir özniteliğin sınıf tanımı için ilişkisiz olup olmadığının ölçülmesidir. Bu şekilde ilişkisiz öznitelikler ayrılırken, fazlalıkların ayrılmasında başarı sağlanmaz. Altküme değerlendirmede belli bir arama stratejisine göre oluşturulan altkümeler puanlanarak birbirleriyle karşılaştırılır. Daha iyi değerlendirilen altküme elde tutularak bir sonraki altküme değerlendirme iterasyonu gerçekleştirilir.

Bir öznitelik altkümesinin ne kadar iyi olduğunu ölçmenin zorluğunun yanında alt kümelerin oluşturulmasında kullanılan yöntemlerin etkileri de göz önünde bulundurulmalıdır. Boyutun binlerle ifade edildiği kümelerde açgözlü (greedy) yöntemler zaman karmaşıklığını idare edilemez düzeye getirebilmektedir. Bu nedenle genelde arama algoritmalarından sezgisel ya da rastgele yöntem izleyenler tercih edilmektedir.

En az sayıda öznitelik kullanılarak, veri kümesinin büyük kısmını ifade edebilen toplam özniteliklerin altkümesi *optimal altküme* olarak adlandırılır[44]. f bir veri vektörünü, C sınıfını ifade etsin. G kümesi F öznitelik kümesinden seçilmiş elemanlardan oluşsun. Buna göre f_F ve f_G ilgili özniteliklerin kullanıldığı veri vektörleridir. İfade etme ölçeği genelde sınıf etiketi kullanılarak sağlanır. Altkümenin o örnek sınıflarının birbirinden ayırt edilebilirliğini koruyacak şekilde oluşturulması gerekir. Tüm bunlar göz önünde bulundurularak optimal altküme olasılık teorisi kullanımı Eşitlik 2.2'deki gibi tanımlanabilir.

$$\min(P(C|F = f_F) \sim P(C|F = f_G)) \quad (2.2)$$

Öznitelik seçiminde kullanılan yöntemlerden biri *tersine eleme*dir. F kümesinden sırayla elemanlar çıkarılarak ayırt edilebilirlik ölçümleri yapılır. *Markov örtüsü*, bu yöntemde kullanılan algoritmalarından biridir. $M_i \subset F$ ($F \not\subset M_i$) olmak üzere,

$$P(F - M_i - \{F_i\}, C | F_i, M_i) = P(F - M_i - \{F_i\}, C | M_i) \quad (2.3)$$

ise, M_i, F_i nin Markov örtüsüdür denir[44].

Öznitelik kümesinden ayıklanmak istenen ilgisiz ve fazla bilgi oluşturan elemanlardır. Bir öznitelik veri kümesinde vektörlerin ilgili sınıf atamaları için hiçbir iyileştirme sağlamıyorsa ilgisizdir. Fazla bilgi ise aynı bilginin birden fazla öznitelik ile belirtilmesi durumunda söz konusudur. Örneğin, bir süre hesabının hem İstanbul hem Londra yerel saatine göre yapılması fazla bilgi oluşturur. İkisi de aynı özelliği ifade eder, yalnız birinin öznitelik kümesinde bulunması yeterlidir. [44]'teki tanıma göre, F_i bir özniteliği belirtmek üzere, tersine eleme ile $S_i = F - \{F_i\}$ kümesi elde edilir. İlgisiz öznitelik

$$P(C|F_i, S_i) = P(C|S_i) \quad (2.4)$$

eşitliğini sağlamaktadır. Yüksek ilgili öznitelik ise,

$$P(C|F_i, S_i) \neq P(C|S_i) \quad (2.5)$$

olduğu anlamına gelir. Bir öznitelik zayıf ilgi oluşturunca ilgisizlik eşitliğini sağlamakla birlikte S_i 'nin en az bir elemanı S_i' için

$$P(C|F_i, S_i') \neq P(C|S_i') \quad (2.6)$$

şartını sağlar.

Bir özniteliğin güçlü ilişkili olması optimal kümede daima bulunması gerektiğini, zayıf ilişkili olması her zaman bulunması gerekmediğini, ilişkisizliği ise gerekliliğin olmamasını ifade eder. Bu kapsamda zayıf ilişkiler fazla bilgiyi ifade eder. Orijinal öznitelik kümesi içinden seçilen F_i 'lerden Markov örtüsüne sahip olanlar çıkarılmak suretiyle optimal altküme oluşturulabilir. Tüm bunlardan yararlanılarak yapılan fazlalık oluşturan öznitelik, zayıf ilişkili ve Markov örtüsü bulunan olarak tanımlanabilir.

Altküme değerlendirme yöntemi ile başarılı optimal altkümeler oluşturulsa da altküme oluşturma adımının karmaşıklığı yüksek olduğundan bunu tercih etmek uygun olmaz.

Sezgisel arama yöntemleri kullanılsa dahi çok büyük boyutlu veri kümelerinde karmaşıklık $O(N^2)$ 'ye varacaktır. Bu, korelasyon tabanlı, bilgi kazanımına dayalı yöntemlerin tercih edilme sebebidir.

Korelasyon iki değer arasındaki bağımlılık ilişkisini ifade eder. Korelasyon analizi doğrusal ya da doğrusal olmayan şekilde yapılabilir. Doğrusal korelasyonda en çok kullanılan yöntem doğrusal korelasyon katsayısıdır. Korelasyon katsayısı iki değişkenin kovaryansının, standart sapmalarının çarpımına bölünmesi ile bulunur ve $[-1,1]$ arasında değer alır. İki değer korelasyonunun olması 1 ya da -1 olarak ölçülürken, korelasyonun sıfır olması değerlerin bağımsız olduğunu gösterir. Doğrusal korelasyon ölçümü sembolik veride uygun değildir.

Doğrusal olmayan korelasyon analizi entropi tabanlıdır. Sistemdeki düzensizliği/belirsizliği ifade eder ve bir X değerinin entropisi Eşitlik 2.7'de verildiği üzeredir.

$$H(X) = - \sum_i P(x_i) \log_2 P(x_i) \quad (2.7)$$

Y değişkeninin değerlendirilmesinden sonra X'in entropisi

$$H(X|Y) = \sum_j P(y_j) \sum_i P(x_i|y_j) \log_2 P(x_i|y_j) \quad (2.8)$$

olarak hesaplanır. Gerçekleşen entropi azalması, bilgi kazanımı (IG) olarak adlandırılır, Eşitlik 2.9'da gösterildiği üzere Y'nin X'e korelasyonunun büyüklüğünü ölçer.

$$IG(X | Y) = H(X) - H(X | Y) \quad (2.9)$$

X ve Y'nin sırasının değişmesiyle sonuçta farklılık olmaması için normalizasyon yapılma yöntemine gidilebilir. Bu işlem ile simetrik hâle gelir ve simetrik bilinmezlik denir (Eşitlik 2.10).

$$SU(X, Y) = 2 \left[\frac{IG(X|Y)}{H(X) + H(Y)} \right] \quad (2.10)$$

$SU(X, Y)=0$ ise değişkenler bağımsız, $SU(X, Y)=1$ ise -yerel saat örneğinde olduğu gibi- değişkenlerden birinin bilinmesi diğerinin hesaplanmasını sağlamaktadır. Bir özneteliğin

sınıf tanımına olan korelasyonu C-korelasyon, iki farklı özneliğin birbirlerine olan korelasyonları ise F-korelasyondur.

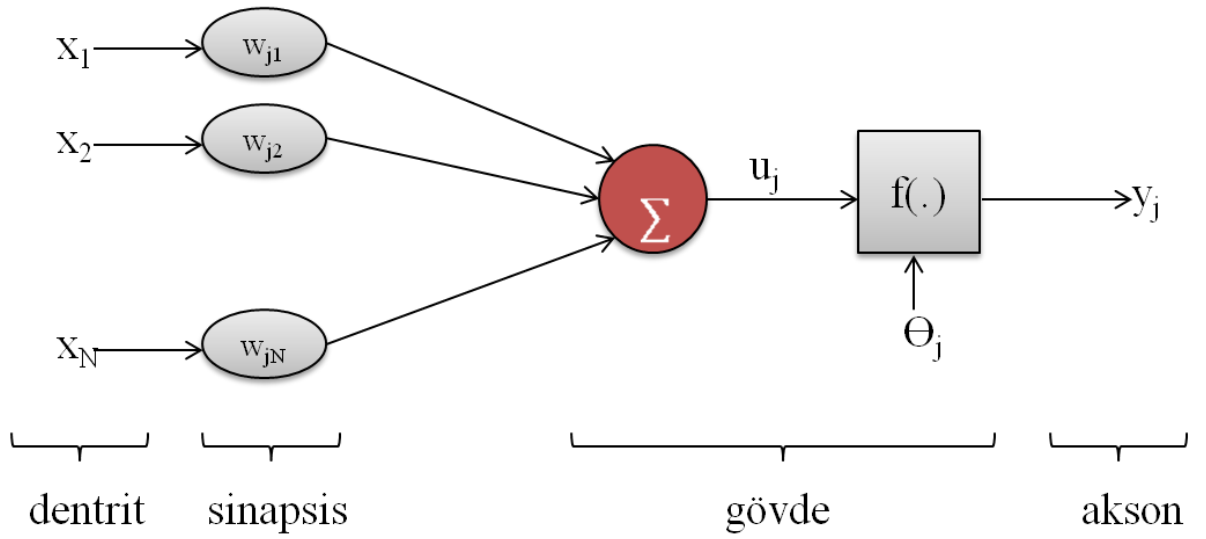
2.3.2.2 Öğrenmenin Yapay Sinir Ağları ile Gerçekleştirilmesi

Biyolojik sinir sisteminin modellenmesi sonucunda yapay sinir ağları (YSA) oluşturulmuştur. Model matematiksel ya da hesaplama dayalıdır. En küçük birimi, biyolojik esin kaynağında da olduğu gibi *nörondur*. En basit hali ile biyolojik nöron diğer nöronlardan gelen sinyali alan dentrit, hücre gövdesi ve başka nörona sinyal ileten akson'dan oluşur. Bir hücrenin aksonu, diğerinin dentrit'ine iletimde bulunur. Bu ikisi arasında, kimyasal iletişimin gerçekleştiği sinapsis bölümü bulunur.

Bir yapay nöron da bir ya da daha fazla girdi alabilir, bunlar üzerinde doğrusal olmayan işlem yapar ve hesabı belirlenen eşik değeri ile kıyaslar. Eşik değeri aşılmışsa hesabı çıktı olarak, varsa bağlı olduğu nörona, yoksa öğrenme çıktısı olarak sisteme gönderir. Eşik değeri yerine ön gerilim (bias) de kullanılabilir. Ön gerilim, hesaba eklenen bir değerdir. Bir yapay nöronun verildiği Şekil 2.10'da, biyolojik nörondaki karşılıkları da gösterilmiştir.

Yaygın gösterimde giriş sinyalleri $X=\{x_1, x_2, \dots, x_n\}$ olarak gösterilirken, ağırlıkları $w_{ji} \in W, j < n$ ile ifade edilir ve i'inci nöronun j. nörona etkisini belirtir. Etki pozitif değer ise *uyarıcı*, negatif ise *sönümleyicidir*. Ağırlıklar, ilgili girdiler ile çarpılarak doğrusal toplam operatörüne ulaşır (Eşitlik 2.11). Aktivasyon fonksiyonunda eşik ya da ön gerilim işlemi gördükten sonra çıktı haline gelir.

$$u_j = \sum_{i=1}^n w_{ij}x_i \quad (2.11)$$



Şekil 2.10: Yapay nöron.

YSA, örüntü tanıma ve veri sınıflandırma gibi problem tanımlarına göre yapılandırılırlar. Yapılandırma, öğrenme işlevinin nasıl gerçekleştirileceğini belirler. Öğrenme, *gözetimli* ya da *gözetimsiz* gerçekleştirilebilir. Gözetimsiz öğrenmede veri vektörlerinin etiketleri yoktur. Sistem yalnız veri kümesini girdi olarak alır. Tercih edilen bir çıktı yoktur, etiket eşlemesi olmadığı için çıktıların doğru/yanlış olduğu yorumu yapılmaz. Sistem bu şekilde veri kümesinde genellikle örüntü keşfi ya da aykırı durumların tespit edilmesi işlemlerini gerçekleştirir. Bu yöntem tez kapsamında kullanılmamıştır. Bu yöntem hakkında ayrıntılar için, Ghahramani[45]'in rehberine başvurulabilir.

Gözetimli öğrenmede ise öğrenme sonucunda beklenen bir çıktı vardır. Bu aykırı tanımlanmış etiketler olabileceği gibi, sürekli değerler de olabilir. Eğitim ve doğrulama kümesine beklenen çıktı verilir ve öğrenme modeli eğitilir. Beklenen çıktının sisteme verilmediği test kümesinin eğitilmiş sistemde elde ettiği sonuçları ile beklenen çıktılar kullanılarak performans değerlendirmesi yapılır (bkz. Bölüm 2.3.3).

YSA'da öğrenme, sinapsis bağlantılarının, yani ağırlıkların, ayarlanması ile gerçekleşir. Bir nöron i için ağırlık vektörü

$$\mathbf{w}_i = [w_{i1} \ w_{i2} \ \dots \ w_{in}]^T \quad (2.12)$$

şeklinde bir sütun vektörü olarak temsil edilir. Öğrenme sinyali, ağırlık, girdi ve beklenen çıktının bir fonksiyonudur (Eşitlik 2.13). YSA'nın kullandığı algoritmaya göre tasarlanmıştır.

$$r = f(w_i, x, d_i) \quad (2.13)$$

Ağırlıkların ayarlanması Δw_i 'nin eklenmesi ile olur. Bu, öğrenmenin bir adım ilerlemesi anlamına gelir. Bu değerin bulunması için c , öğrenme katsayısı, kullanılır. Genel anlamıyla öğrenme hızını belirler. Probleme göre ideal öğrenme katsayısının bulunması gerekir. t öğrenme adımında Δw_i 'nin hesaplanması için genel kural Eşitlik 2.14'te verildiği üzeredir. Ayarlanmış $t+1$ adımındaki ağırlık vektörü ise Eşitlik 2.15 ile gösterilmiştir.

$$\Delta w_i(t) = c \cdot r[w_i(t), x(t), d_i(t)] \cdot x(t) \quad (2.14)$$

$$w_i(t+1) = w_i(t) + \Delta w_i(t) \quad (2.15)$$

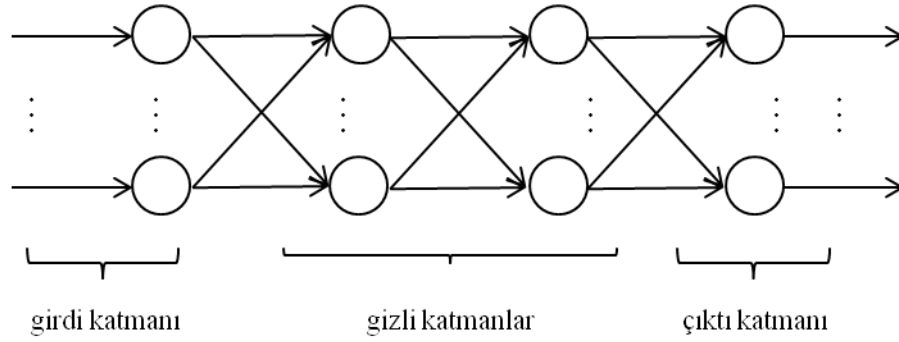
Problem ayrık zamanlı veri üzerinde ise denklem Eşitlik 2.16'da olduğu formda uygulanır.

$$w_i^{k+1} = w_i^k + c \cdot r[w_i^k, x^k, d^k] \quad (2.16)$$

Nöron tek başına da ağ oluşturabilir. Şekil 2.10'daki yapı bir perceptron'dur. Öğrenme sinyali Eşitlik 2.17'deki gibidir. Girdinin bir sınıfa ait olup olmadığını gösteren ikili bir sınıflandırıcıdır.

$$r_i = d_i - y_i \quad (2.17)$$

Nöronlar, birbirine bağlanarak çok katmanlı YSA'yı oluşturur. Bağlantı türlerine göre çeşitli ağ yapıları inşa edilir. Sıra ile birbirilerine bağlanan ve döngü veya gecikme bağlantılarının bulunmadığı ağ yapısı ileri beslemelidir (Şekil 2.11). Nöronlar belli işlem seviyelerinde bir araya gelir. Bu yapılaşmaya katman denir. Algoritmanın işlemeye başladığı ilk katman girdi katmanı, son katman çıktı katmanı olarak adlandırılır. Aradakiler ise gizli katmanlardır. Gizli katmanlar, her YSA yapısında bulunmak zorunda değildir.



Şekil 2.11: İleri beslemeli ağ statik YSA.

Şekil 2.11'deki ileri beslemeli ağ, statik bir sistemdir. Dinamik sistemde, geri besleme ya da gecikme etkenleri bulunur. Temelde, şimdiki duruma göre bir sonraki durumun tahminini yapmak için kullanılır (Eşitlik 2.18). Yalnız bir adım değil, birden fazla adımın tahmini gerçekleştirilecek üzere düzenlenebilir. Bu nedenle *hafızaya* sahiptir, girdiler bir zaman sırasına göre sisteme girer.

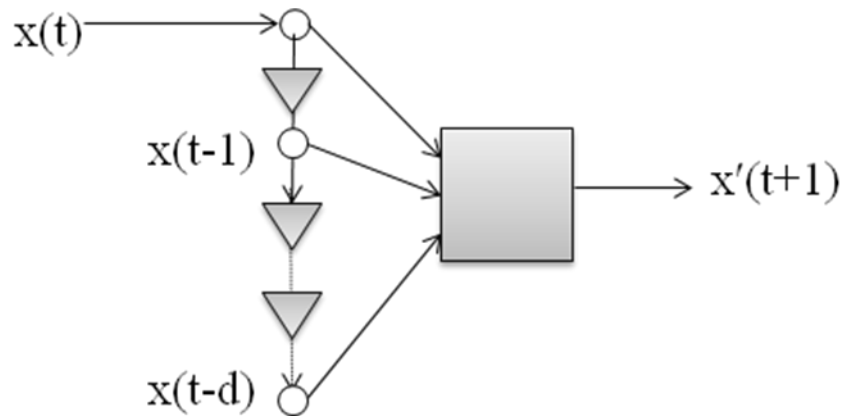
$$y_{n+1} = f(y_n, x_i) \quad (2.18)$$

Burada zaman serilerinden bahsetmek gerekir. Zaman serisi, zamana bağlı bir dizi vektör ya da sabit sayıdan oluşur. Bir işlem çıktısıdır ve verinin sıralaması önemlidir. Sürekli bir sinyaldir, işlenebilmesi belli aralıklarla örneklenmesi gerekir. Uniform bir örnekleme gerçekleştiriyorsak ve örnekleme aralığımız t ise, zaman serisi Eşitlik 2.19'daki gibi gösterilebilir. Probleme uygun örnekleme yapmak gerekir.

$$\{x[t]\} = \{x(0), x(\Delta t), x(2\Delta t), \dots\} \quad (2.19)$$

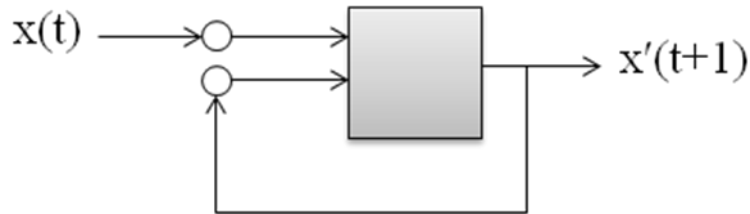
Gelecek değerleri tahmin etme problemi ele alınırken *öngörü ufku* değeri kullanılır[46]. $x[0]$ 'dan $x[t]$ 'ye kadar elimizde bulunan değerleri kullanarak $\hat{x}[t+s]$ değerini bulmak isteriz. s , öngörü ufku parametresidir. Tez kapsamında ele alacağımız problem bir adım ötesini öngörme olduğu için $s=1$ kullanılmıştır. Problemi çözmek için önce genelleştirilmiş bir model kabulü yapılır. Ardından, elimizdeki zaman serisi değerleriyle model eğitilir. Eğitimde her $x[t+1]$ istenen değer, $\hat{x}[t+1]$ ise öğrenme sonucu elde edilen değerdir.

Kullandığımız zaman serisi girdisindeki zamansal veriyi uygulamada kullanmak çok kolay değildir. Çünkü veri kümesindeki ilk değeri öngörmek için bir önceki zaman adımının değeri kümede bulunmamaktadır. Bu nedenle sisteme gecikme mekanizması eklenir (Şekil 2.12). Buna göre, veri kümesinin başından gecikme adımı kadar veri vektörü önceki zaman bilgisi olarak alınır. Öğrenme işlemi boyunca, bu gecikme mekanizması gerçekleştirilir. Yani veri zaman ekseninde kaydırılır. Girdi veri kümesi sonlu elemandan oluşur. Her t anında kırılması belirlenen geçmiş veri miktarını *gömme boyutudur*.



Şekil 2.12: Gecikmeli YSA.

Geri besleme, ağda t anındaki katman çıktısının bir sonraki çıktının hesaplanmasında kullanılmak üzere ağda bir katmana girdi olarak alınması ile oluşur (Şekil 2.13).



Şekil 2.13: Geri beslemeli YSA.

YSA öğrenmesinde ağırlıkların ayarlanması gerekliliğinden bu bölümün başlarında bahsedilmiştir. Bu "ayarlanma", YSA terminolojisinde *ağırlık güncelleme* olarak anılmaktadır. Birçok YSA algoritması ağırlık güncellemede *geri yayımlı öğrenmeyi*

[47] kullanır. Yayınlama ve ağırlık güncelleme adımlarından oluşur. Yayınlama önce ileri beslemeli olarak gerçekleşir. Bir adım ileri besleme çıktısı alındıktan sonra, hedeflenen değer kullanılarak, YSA geriye doğru izlenerek tüm katmanlarda hedef ve çıktı farkı hesaplanır. Hesaplama, ağırlıkların bulunduğu bölüme gelindiğinde sona ermiştir. Bulunan fark sonucu ve girdi aktivasyon değeri çarpılarak her ağırlık düğümü için *gradyan değeri* elde edilir. Gradyanın öğrenme algoritmasında belirlenen bir oran, ağırlık değerinden çıkarılır. Çıkarılan bu miktar *öğrenme oranı* olarak adlandırılır. Oran büyüdükçe öğrenme hızı artmaktadır. Geri yayımlı öğrenme adımları, istenen bir hata değerine ulaşana kadar tekrarlanmalıdır.

2.3.3 Öğrenmenin Ölçümü

Öğrenme ölçümünün esası, hatanın hesaplanmasına dayanır. Hata, hedeflenen sonuç ile alınan çıktı arasındaki fark olarak kabaca tanımlanabilir. Yapay öğrenmede kullanılan çeşitli öğrenme ölçümü yöntemlerinden bu tez kapsamında kullanılan ortalama kare hatadır (MSE, mean squared error). MSE, YSA algoritmaları için uygun bir ölçümdür. Başka öğrenme yöntemleri için, onlara uygun ölçüm metotları kullanılmalıdır.

MSE, olabilecek en az değere indirilerek öğrenme performansının artırılması amaçlanır. [48]'da ayrıntılı bir şekilde MSE değerinin matematiksel çıkarımı anlatılmaktadır. Burada, çıkarım sonuçları üzerinden gidilecektir.

X , n adet veri vektörü ve L adet öznelikten oluşan öğrenme kümesi olsun. Beklenen öğrenme çıktısı Y , öğrenme sonucu oluşan bir çıktı da $\hat{y}$ ise, Y 'nin en az kare hata hesabı

$$\hat{y}(x_1, x_2, \dots, x_L) = E[Y|X_1 = x_1, X_2 = x_2, \dots, X_L = x_L] \quad (2.20)$$

şeklinde yapılır. n çıktı $\hat{Y}$ ve hedef değerler Y kümeleri ise tüm kümenin MSE hesabı

$$\hat{Y} = \hat{y}(X) = \frac{1}{n} \sum_{i=1}^n (\hat{Y}_i - Y_i)^2 \quad (2.21)$$

formuna dönüşür [48]. Öğrenme performansı kapsamında, MSE'nin en aza indirildiği algoritma parametrelerinin kullanılması esastır. Bir başka deyişle, MSE değeri ne kadar az ise öğrenme performansı o kadar iyidir ifadesi yerindedir.

2.4 AĞ TIKANIKLIK KONTROLÜNÜN MAKİNE ÖĞRENMESİ İLE SAĞLANMASI

Veri merkezlerini içeren ağ tıkanıklık durumları birçok sunucu ve istemciyi etkilediğinden, servis operatörleri açısından oldukça önemlidir ve ayrıca incelenmelidir. Veri merkezi üzerinde tıkanıklık oluşturan faktörlerin başında büyük veri akışları (elephant flow) gelir. Bunlar, genelde şirket veritabanlarının yürüttüğü güncelleme işlemlerini kapsamaktadır ve akış başına 100-500 MB büyüklüğündedir.

Veri merkezi tıkanıklığına neden olan bir başka etken küçük veri akışlarıdır (mice flow). Tıkanıklık ilgisinin bir önceki kadar bariz olmamasının nedeni, tek başlarına değil, birbirleriyle olan etkileşimleri ile buna neden olmalarıdır. Birçok sunucu veri merkezine eş zamanlı bir gönderi yaptığında tampon taşması yaşanabilir ya da sunucular birleştirilmek üzere parçalı veri istemine maruz kalabilir. İlgili sunucular mümkün olduğu kadar hızlı bir şekilde cevap verir. Parçalı veri talebinde bulunan bir istemci aldığı veri bloklarını birleştirir ve dış bir istemciye gönderir. Sunucuların hızlı cevap verme gereksinimleri, tüm veri parçalarının hemen hemen hepsinin neredeyse aynı anda cevap vermesi ile sonuçlanabilir.

2.4.1 Geleneksel Ağlarda Yapılan Çalışmalar

Makine öğrenme metodolojisinden yararlanan bir çalışma [49] veri akış istatistiği kullanılarak ağ uygulamalarının sınıflandırılmasını analiz etmiştir. Port ya da yük tabanlı yaklaşımların domine ettiği klasik metodolojilere tercih edilme nedenini, uçtan uca (P2P, peer to peer) uygulamalarının dinamik port numaraları, maskeleyme ve şifreleme kullanmalarının bunları uygulamada büyük analiz ve işlem yükü gerektirmesi olarak göstermiştir. Yarı gözetimli öğrenmenin uygulandığı çalışmada, etiketsiz büyük miktardaki verinin çok daha küçük bir etiketli veri kümesi kullanılarak sınıflandırması yapılmıştır. Gözetimsiz aşamada kümeleme algoritmalarının kullanılması ile etiket listesinde bulunmayan tipteki ağ uygulamalarının tespiti ve gelecekte benzer uygulama akışının yakalanması halinde bu yeni tespit edilen tip ile etiketlenmesi sağlanabilmektedir.

Ağ trafiğinin sınıflandırılmasında genel olarak üç etiket tipi kullanılır: hassas, en iyi trafik, istenmeyen[50]. Bunlar, ilgili veri akışının iletme önceliği ya da iletilmemesinin belirlenmesinde kullanılır. Hassas trafiğin zamanında iletilmesi en büyük önemi taşır.

Hassas akışın dışında kalanlar en iyi trafik olarak işlenir ki bunlar servis kalitesi (QoS) metriklerinde kullanılmamaktadır. İstenmeyen bir bağlantı için iletilmeme ya da önemli ölçüde aksatma yapılmaktadır.

Veri merkezlerindeki tıkanıklık üzerine bir inceleme sunan [51], *Metalaşmış Veri Merkezi için Etkili Paket Taşıma* (DCTCP, Efficient Packet Transport for the Commoditized Data Center) [52] algoritmasına değişiklik önermektedir (Tablo 2.2 ve Tablo 2.3). Veri merkezinde protokol ayırımına gitme üzerine kurulu olan çalışmada veri merkezinin kendi içindeki ile dış iletim protokolleri ayrılmıştır. Bununla beraber, iletim yığınının dış internet iletişiminden kaynaklanan muhtemel bir olumsuz etkilenme durumu ortadan kaldırılmıştır. Bahsedilen değişikliklerin büyük kısmı YTA yapısında hali hazırda mevcuttur. Bu yönüyle, veri merkezlerinde YTA yapısının kullanılmasının gerekliliği ile ilgili bilgi sağlamanın yanında, değindiği gereksinim ve çözümler açısından bu çalışma kayda değer tespitler içermektedir.

Tablo 2.2: Algoritma - 1

VERİ MERKEZİ TIKANIKLIK KONTROL ALGORİTMASI -1 [51]
$(\text{yavaş başlatma eşiği}) = (\text{tıkanıklık penceresi büyüklüğü}) / 2$ $\text{EĞER } (\text{yavaş başlatma eşiği}) < (\text{en yüksek iletim birimi})$ $(\text{yavaş başlatma eşiği}) = (\text{en yüksek iletim birimi})$ $(\text{toparlanma zamanı hedefi}) \leq 1$ $(\text{tıkanıklık penceresi büyüklüğü}) = (\text{yavaş başlatma eşiği})$

Tek bir pencere verisinde bir ya da daha fazla kayıp oluştuğunda, tıkanıklık penceresi büyüklüğü yarılanır ve yavaş başlatma eşiği aynı değere düşürülür. Bir eko alındığında, tıkanıklık kontrol algoritması iki ekstra bilgi parçası verir. Bu iki ekstra bilgi parçası önerilen algoritmanın anahtarlarıdır, bunlar *pencerede işaretçisi* ve *işaretlenen paket sayısı*dır. Pencerde işaretçisi, bir sonraki algoritmanın kullanıp kullanılmayacağını kararda kullanılır ve işaretlenen paket sayısı görmezden gelir. Bir sonraki

algoritmada bu iki değer ile ağ yapısının gönderdiği diğer değerler birleştirilerek kullanılır.

Tablo 2.3: Algoritma - 2

VERİ MERKEZİ TIKANIKLIK KONTROL ALGORİTMASI -2 [51]
EĞER (içerideki pencere) = 0 $(\text{tikanıklık penceresi büyüklüğü}) = (\text{ECN önceki tikanıklık penceresi büyüklüğü}) - ((\text{en yüksek iletim birimi}) * (\text{işaretlenen paket sayısı}))$ $(\text{yavaş başlatma eşiği}) = (\text{tikanıklık penceresi büyüklüğü}) - ((\text{en yüksek iletim birimi}) * (\text{işaretlenen paket sayısı}))$ DEĞİL İSE $(\text{tikanıklık penceresi büyüklüğü}) = (\text{en yüksek iletim birimi}) * (\text{işaretlenen paket sayısı})$ $(\text{yavaş başlatma eşiği}) = (\text{en yüksek iletim birimi}) * (\text{işaretlenen paket sayısı})$

Bir *sensör iletim kontrol protokolü* (STCP, Sensor Transmission Control Protocol) parçasının paket içinde her gönderilişinde, iletimin yapıldığı zamandaki tikanıklık penceresi de kaydedilir. Bir *Net Tikanıklık Bildirme* (ECN, Explicit Congestion Notification) ekosu geldiğinde, modüler tikanıklık kontrolünü çağırmadan önce, *İletim Sekans Numarası* (TSN, Transmit Sequence Number) bulunmaya çalışılır. ECN ekosu yeni bir pencere verisindeyse TSN her zaman bulunacaktır. Eğer TSN bulunduysa, TSN'nin gönderildiği zamandaki tikanıklık penceresi büyüklüğü ağ yapısında ECN için özel alanda (ecn prev cwnd) kaydedilir.

Veri merkezi içindeki ağ tikanıklığı, kullanıcı deneyim kalitesi ve operatör verimliliği ile ters orantılı ilişki gösterir. Optimize edilmesi gereken düğümler yönlendirici ya da anahtarlar olarak göz önüne alındığında bunlar üzerindeki kuyruk dikkat çekmektedir. Küçük paketlerin kuyruklama gecikmesi göz ardı edilebilir sürede iken büyük paketlerin oluşturduğu yük azımsanamaz büyüklüktedir. [53], çok katmanlı yapay sinir ağının

temelini oluşturduğu derin sinir ağları (DNN, Deep Neural Network) ile bu soruna çözüm aramıştır. Yapay sinir ağlarında yaygın olduğu üzere derin ağlar da geri yayılım algoritması kullanılarak eğitilir (bkz. Bölüm 2.3.2). Bunun nedeni gradyan hesaplaması ve algoritma başlangıç değerlerinin rastgele üretilmesinden kaynaklanan yerel minimuma düşme riskidir. Hinton ve diğ.[54] gerçekleştirdikleri çalışmada gözetimsiz ve açgözlü yaklaşım ile bunun üstesinden gelmişlerdir. Hinton'ın bulgularından yararlanılarak geliştirilen yığın otoenkoder [53], önemli link'lerin tıkanıklığını ölçer. Bu da günün belli saatlerindeki tıkanıklık değerlerinin kayıt altına alınması ile olur. İncelenen problem trafik akışlarının uzlamsal-geçici tahmini olarak tanımlanmıştır.

Veri merkezlerinde *Trafik Akışlarının Uzlamsal-Geçici Tahmini Problemi*, bu dokümanın başlığındaki sorunun formal tanımıdır ve şu şekilde ifade edilebilir: X_i^t , t 'inci zaman aralığında i 'inci gözlem anında gözlemlenen trafik akışı olsun. Bir gözlem anı; arayüz sayacı, anahtar CPU yükü ya da bellek verimi veya başka ilgili sensör değeri (geleneksel ağ arayüz sayaçları sensör olarak kabul edilir) olabilir. Ağ topolojisi açıkça temsil edilmemiş fakat daha fazla gözlem anında kodlanmıştır. Gözlemlenen trafik akış verisi $\{X_i^t\}$ sekansı $i = 1, 2, \dots, m$ ve $t = 1, 2, \dots, T$ olarak verildiğinde, problem $(t + \Delta)$ zamanında bir Δ tahmin ufkunda trafik akışının ve dolayısıyla potansiyel tıkanıklığın öngörülmesidir. Aynı zamanda trafik akışları içinde öz geçici ilişkiler göz önünde bulundurmak istenmektedir. Bu nedenle herhangi bir zaman aralığı t 'de trafik akışını ön görmek için önceki zaman aralıklarındaki trafik akışları da kullanılır. Bir r değeri için $X^{t-1}, X^{t-2}, \dots, X^{t-r}$ değerli öngörü elde etmek için zamanda ne kadar geriye gidilmesi gerektiği kesinlik göstermemektedir.

Derin yapay sinir ağlarını kullanan bir diğer çalışma [55], şehir araç trafiğinde oluşan tıkanıklığın günün belli aralıklarındaki tahminine yöneliktir. Bu çalışmanın temel aldığı mimari yapısal olarak bilgisayar ağlarına benzemektedir ve ortak değişkenleri kullanır. Veriler akıllı ulaşım sistemi kullanılarak, belirli düğümler üzerinde ilgili zaman aralıklarında ölçüm yapılarak toplanmıştır. Derin ağların kullanımı, klasik trafik mühendisliğinde yapılan gerçekçi olmayan varsayımların kullanımı gereksinimini ortadan kaldırmış ve parametre kalibrasyonu kısıtlarını gidermiştir. Derin bir Kısıtlı Boltzmann Makinesi (RBM, Restricted Boltzmann Machine) ve Yinelenen Sinir Ağı (RNN, Recurrent Neural Network) mimarisinden vazgeçilerek trafik tıkanıklık evrimi

taksi GPS verilerine dayanarak modellenmiş ve öngöründe bulunulmuştur. Sonuçlar tahmin isabetinin, modelin GPU-tabanlı paralel ortamda gerçekleştirilmesi durumunda, 6 dakikadan az sürede %88'e kadar çıktığını göstermektedir.

Her linkteki trafik durumu, 1 tıkanık ve 0 açık olmak üzere, ikili duruma sınıflandırılmıştır. Link trafik durumu zamanla değişir ve sonunda çeşitli uzunluklarda ikili sekans (binary sequence) haline gelir. Tipik ulaşım ağının boyutu göz önüne alındığında, toplam ağdaki tıkanıklık örüntüsü zaman (zaman adımı sayısı) ve uzay (link sayısı) yüksek boyutlu matrise eş değerdir. Her matris elemanının uzlamsal ve zamansal olarak nasıl değişim geçirdiğini anlamak ağ boyunca trafik tıkanıklığı evrimi tahmini için önemlidir. İlk adım, GPS verisini kullanarak her bir linkin trafik durumunu tanımlamaktır. Tatmin eder kalite sağlamak için, bir ara adım olarak, kayıp ve hatalı veri gözden geçirilecektir. Son olarak, işlenmiş GPS verisi ağ boyunca örüntüleri temsil etmek için çeşitli zaman aralıklarında toplanır.

Daha önce değinildiği üzere, her yol segmentindeki trafik durumu tıkanıklık durumunu belirten ikili değere dönüştürülebilir. Geçmiş trafik örüntüleri ve büyük ölçekli bir ağdaki link sayısı göz önüne alınarak tıkanıklık evriminin tahmini problemi yüksek boyutlu uzlamsal sekans öğrenme problemi haline gelir. RNN-RBM modeli zaman-uzlamsal ağ tıkanıklığı örüntüsünün öngörülmesine uygulanabilir. RNN-RBM modelini etkili bir şekilde gerçeklemek için *küçük yığınlar gradyan azalma optimizasyonu* (mini-batches gradient descent optimization) metodu uygulanır. Yani, eğitim kümesi küçük sayıda örneklerden oluşan gruplara bölünür ve her bir örnek kümesi standart gradyan azaltma optimizasyonu kullanılarak paralel olarak işlenebilir.

2.4.2 Yazılım Tanımlı Ağlarda Yapılan Çalışmalar

YTA trafik incelemesinin gerçekleştirilmesinde kullanılan klasik yöntemler Bölüm 2.2'de verilmiştir. Bu bölümde ise makine öğrenmesi yöntemi kullanılarak sunulan çözümlere dair inceleme yapılmıştır. [56]'da YTA trafik mühendisliği, bant genişliğini veri akışlarına atamak üzere kullanılır.

Mo ve Walrand tarafından tanıtılmış α -adalet konsepti [57] yaygın olarak kullanılmaktadır. Klasik trafik mühendisliği algoritmaları gerçek ortamda saniyelerce, hatta bazen dakikalarca sürmektedir. Bu uzun çalışma zamanları trafik mühendisliğini

çevrimdışı bir araç olarak kullanılmaya zorlamaktadır. Çalışmada, klasik algoritmalara eş değer, fakat çalışma zamanı açısından milisaniye derecesinde performans gösteren ve bu nedenle çevrimiçi araç olarak kullanılabilecek bir trafik mühendisliği algoritması gösterilmiştir. α , $[0, \infty]$ aralığında tanımlı bir parametredir, küçük değerler veri hacmini (throughput) ve büyük değerler adalet değerini iyileştirir. $\alpha \rightarrow \infty$ yaklaştıkça, trafik akış ataması en yüksek-en düşük adaletle yaklaşır.

Bulut servisleri ve/veya sanallaştırma gibi yeni teknolojiler son zamanlarda veri merkezi ağlarını değiştirmiştir. Söz konusu teknolojilerin faydaları açık, yöntemlerin desteklenmesi için fakat anahtarlarda daha çok yönlendirme girişleri (entries) gerektirmesi dezavantajdır. Ne yazık ki, anahtarlarda bulunan yönlendirme girişlerinin sayısı kesin sınırlayıcıdır.

Schwabe ve Karl [10], yönlendirme girişlerini minimize etmek için MAC adreslerine dayalı yöntem sunmuşlardır. YTA'larda hedef MAC adreslerinin evrensel etiket olarak kullanılabileceği ve sunucuların (host) adres çözümleme protokolü (ARP, Address Resolution Protocol) önbelleklerinin giriş etiket tablosu olarak kullanılarak, ağ aygıtlarının yönlendirme tablolarının boyutunu düşürülebileceği gösterilmiştir. Özel bir tip veri merkezi ya da ek donanım kapasitesi gerekmemesi önemli bir avantajdır. Tekniklerinin *yönlendirme bilgisi temeli* (FIB, Forwarding Information Base) boyutlu problemi en düşük sayıda FIB girdileri ile çözebileceğini tüm eşlemeli *eş maliyetli çok yollu rota* (ECMP, Equal-Cost Multi-Path Routing) için açgözlü şema tanıtılarak göstermiştir. Modern bir anahtarda, donanımda yönlendirme hat oranında (line rate) olmak üzere gerçekleşir. Yönlendirmenin anahtar bir bileşeni, içeriği adreslenebilir üçlü bellek (TCAM, Ternary Content-Addressable Memory) kullanan arama tablolarıdır. TCAM'ler enerji ve donanım bakımından yüksek maliyetlidir. Bu, bir anahtarın donanımda saklayabileceği yönlendirme girişlerini sınırlar.

YTA'da anahtarlar paketleri işlemezler, yönlendirirler. Yönlendirme tablolarında bulunmayan bir paketin gelmesi durumunda bunu denetleyiciye gönderirler. Bir ağda birden fazla denetleyici bulunabilir, bunlar da birbirlerine anahtarlar ile bağlanmış olabilir. Denetleyicinin diğer bir görevi istatistik değerler tutmaktır. Halihazırda tutulan istatistiklere yenilerini eklemek mümkündür. Hedef IP adresi gibi başlık bilgisinde bulunan tüm alanlarla ilgili bilgi toplamak mümkündür.

YTA mimarisinin yönetiminde en yaygın olarak kullanılan protokol OpenFlow, anahtar ve denetleyicinin arasında güvenli bir kanal gerektirir. Böylece bir OpenFlow anahtarı, bir ya da daha fazla yönlendirme tablosu ve güvenli kanaldan oluşur. Yönlendirme tablolarında eşleme, sayaç ve yönlendirme yönergelerini içeren bilgi bulunur. Paket başlık bilgisinin çeşitli alanları eşleştirme yapısının oluşturulmasında kullanılabilir.

Eşleme tablosunda bulunmayan paketler denetleyiciye gönderilecek (OFPT_PACKET_IN), burada düşürme ya da işleme/yönlendirme kararı verilecektir. Böylece denetleyici, diğer anahtarlardan da gelen paket bilgisini öncelik ve benzeri kurallar uyarınca işleme alacaktır. Paket işleme genel olarak paket bilgisindeki bir alanın güncellenmesi anlamına gelir. Paket ile yapılacak iş denetleyicide belirlendikten sonra paket yeniden anahtara gönderilir (OFPT_PACKET_OUT).

3. MALZEME VE YÖNTEM

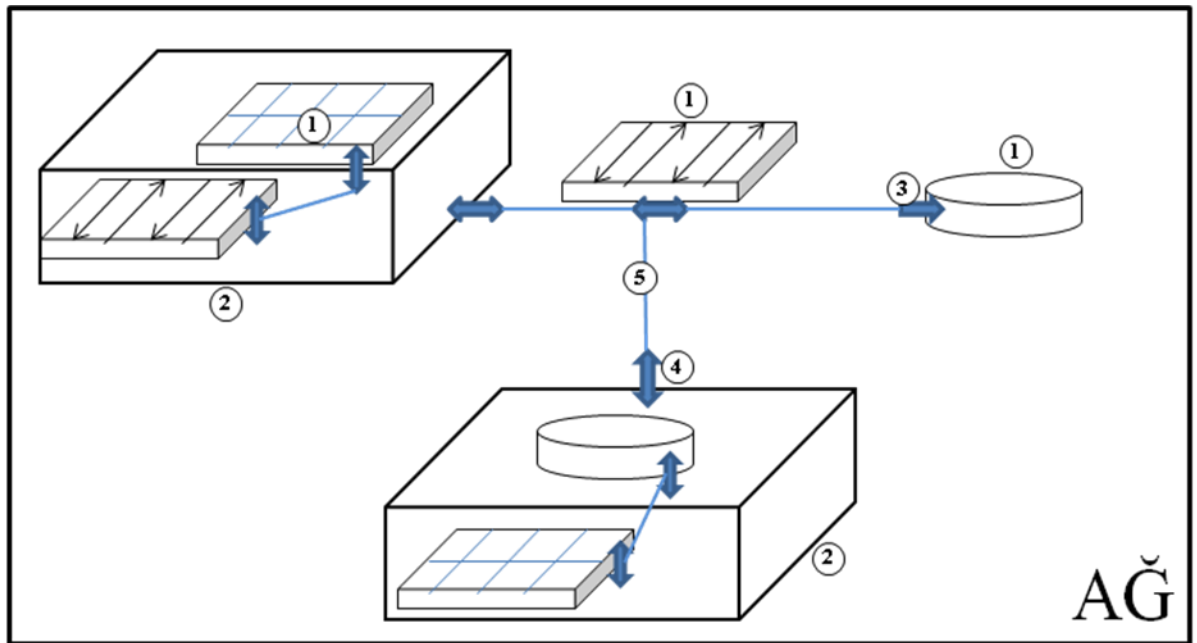
Bölüm 2.1'de belirtildiği üzere, YTA'ların kuzey-bağlam arayüzü, güney-bağlam arayüzünün alt seviye komut dizisini kullanarak kullanıcılara çeşitli ağ uygulamaları yazmak üzere ortam sağlar. Bir çeşit yazılım ekosistemi oluşturur; saldırı tespiti, yük dengeleyici ve yönlendirme algoritmaları gibi ağ yöneticilerinin sıkça başvurduğu araçların geliştirilmesine olanak tanır. Kuzey-bağlam arayüz yolu ile alt katmanlardan veri alınabilir ve güney-bağlam'a bağlı araçların davranışları programlanabilir. Bu tezin konusu olan tıkanıklık kontrolü de kuzeyde yazılmış bir ağ uygulaması senaryosudur. Tıkanıklık kontrolünün sağlanmasında makine öğrenmesi yaklaşımı kullanılırken veri kümesi Omnet++ açık kaynak simülasyon programından çıktı alınmıştır. Tıkanıklık tahmininde kullanılan yinelenen yapay sinir ağı yöntemi olan Eksojen Girişi ile Doğrusal Olmayan Otoregresif Model süreci (NARX) Bölüm 2.3 alt başlıklarında verilmiştir. İlerleyen alt bölümlerde ise simülasyon ortamı, verinin toplanması ve NARX yapısına verilmek üzere düzenlenmesi açıklanmıştır.

3.1 SİMÜLASYON ORTAMI

OMNET++, genişletilebilen modüler bir ayrık olay simülasyonu aracıdır [58]. Simülatör değildir, ancak simülatör oluşturmaya yarayan altyapıyı sağlar. Program ve altındaki modül kütüphanelerinden birçoğu açık kaynaktır. Modüller iç içe tasarlanabilir ve üç seviyeden oluşur. En alt seviyede basit modüller bulunur ve yapısında alt modül bulundurmaz. Bir üst seviye olan bileşik modül ise, basit modüller ve kendine has parametrelerden oluşur. En üst seviyede ise ağ modülü bulunur ki içinde basit ve bileşik modüller bağlanıp ağ topolojisi oluşturulur. NED (ağ tanımlama dili) modüllerinin giriş-çıkışlarını gerçekleştirdiği birim kapıdır (gate). İki modülün iletişime geçmesi için atanan ilişki bağlantıdır. Bu da bir modüldür ve parametreleri 'kanal' değişkeni üzerinde toplanır Şekil 3.1'de OMNET'te bir ağın modül yapısı blok şema olarak verilmiştir.

OMNET'in kendine ait söz dizimi olan programlama dili NED ile modül yapılandırması ve ağ üst seviyede tanımlanır. Programlamanın asıl gerçekleştiği kısım ise NED dosyalarına bağlı C++ sınıf dosyalarıdır. Sınıf dosyaları modülün çeşitli durumlardaki

davranışını belirler ve bunlar fonksiyonlar ile sağlanır. OMNET'teki tüm birimler cObject sınıfından türemiştir. Modüller cModule, kanallar ise cChannel sınıfları olarak gerçekleşir ya da bunlar türetilerek kullanıcı tanımlı versiyonları yazılır. cModule ve cChannel, cComponent sınıfından türemiştir ve kullanıcı modülü türetmede kullanılacak cSimpleModule sınıfı bundan türemiştir. Dikkat edilmesi gereken bir husus, basit modülleri içinde bulunduran bileşik modüllerin cModule'den türetilmesi gerektiridir.



Şekil 3.1: OMNET++'ta örnek bir ağ yapısı. 1: Basit modül. 2: Bileşik modül. 3: Tek yönlü kapı. 4: Çift yönlü kapı. 5: Bağlantı.

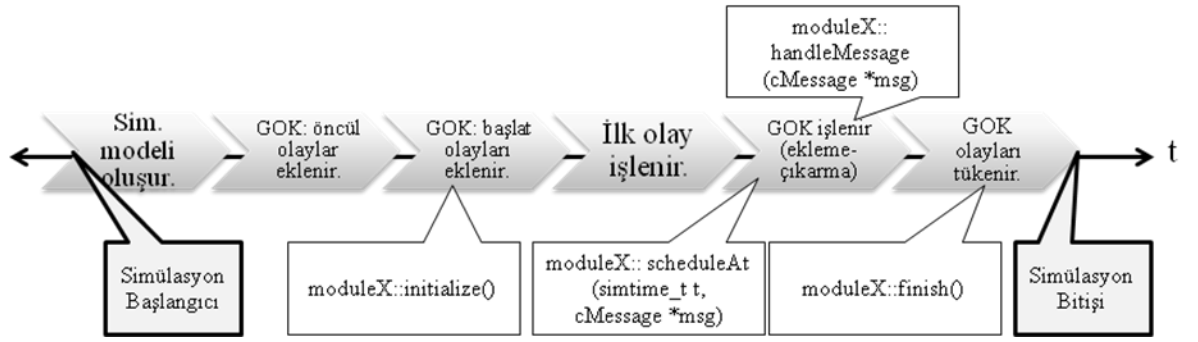
OMNET'in ayrık olay simülasyon aracı olduğu bölüm başında belirtilmiştir. Ayrık olay sistemlerinin bir gereği olarak olaylar (durum değişiklikleri) zaman düzleminde gerçekleşirken birbirinden bağımsızdır. Birbirini takip eden iki olay arasında sistemin hiçbir bölümünde durum değişikliği olmaz. Gelecek Olaylar Kümesi (GOK), programın zaman içinde gerçekleşmesi gereken olayları listelediği veri yapısıdır. Simülasyon başladığında model oluşur ve öncül olaylar GOK'a eklenir. GOK boşaldığında ve simülasyon bitirme şartları sağlandığında kapanış işlemleri yapılır, simülasyon sona erer. Simülasyon çalışmakta iken, GOK'ten olaylar çekilir, zamanlama tablosuna işlenir

ve gerçekleşir. Gerçeklenen olay GOK'e yeni olay girişi yapmak ya da içerideki bir olayı silmek olabilir.

En temel kullanıcı tanımlı basit modül, kalıttan gelen fonksiyonları taşır. Bunlardan bir bölümü, ilgili C++ sınıf dosyasında, üstüne yazılarak modülün işlevi özelleştirilir. Beş temel fonksiyon vardır [59]: initialize (başlat), handleMessage (mesajı yönet), send (gönder), scheduleAt (planla) ve finish (bitir).

Başlat ve bitir fonksiyonlarında modüllerin işlenme sırasını göz önünde bulundurmak gerekir. Modüllerin kendi başlat fonksiyonları, simülasyon başlaması ile GOK'e öncül olayların atanması ile ilk olayın gerçekleşmesi arasındaki zaman diliminde çağrılır. Tüm basit ve bileşik modüller bu fonksiyonun üzerine yazabilir. Bileşik modül ve içindeki basit modüllerin başlatma fonksiyonları çağırılırken izlenen sıra, dıştan içe doğrudur. Yani, önce bileşik modüllerin, sonra altındaki modüllerin başlat fonksiyonu çağırılır.

GOK'teki olaylar herhangi bir hatayla karşılaşılmaksızın tüketildiğinde bitir fonksiyonu tetiklenir. Bileşik modüllerin bitirme fonksiyonu çağırılırken içeriden dışarıya yön izlenir; önce barındırdığı modüllerinin sonra bileşik modülün fonksiyonu çağırılır. Diğer fonksiyonlar, başlat ve bitir arasında geçen süredeki olaylara göre çağrılır. OMNET modülü oluşturunca başlat fonksiyonu yönergeleri işler. Modüle mesaj ulaştığında mesajı yönetme fonksiyonu devreye girecektir. Diğer modüllere mesaj gönderilmek istendiğinde 'gönder' fonksiyon işlemleri gerçekleştirilir. Bitirme fonksiyonu, simülasyon tamamlanma koşullarına ulaştığında tetiklenir. Planla fonksiyonu, modülün kendine mesaj göndermesi işlevini görür. Belirlenen bir t zamanında ilgili öz-mesaj modülün kendisine ulaşacak, mesajı yönet fonksiyonu tarafından işlenecektir. Bir simülasyon süresi boyunca gerçekleşen örnek olaylar şekil 3.2'de zaman ekseninde gösterilmiştir.



Şekil 3.2: Bir simülasyon boyunca zaman ekseninde gerçekleşen olaylar ve moduleX'in kendine zamanlanmış mesaj göndermesi örneği.

Popüler simülasyon model kütüphanelerinden INET [60] , standart protokol modelidir. İnternet yığını, kablolu ve kablosuz bağlantı profillerinden oluşur. YTA kütüphanesi olan Ofomnet[61], INET üzerine kurulmuştur. OpenFlow v.1.0 kullanılarak anahtar, denetleyici ve diğer yardımcı modüller bulunmaktadır. Ofomnet paketi OpenFlow 1.0 versiyonuna göre dizayn edilmiştir. Türetmede kullandığı INET paket versiyonu 2.0, OMNET++ program versiyonu ise 4.2'dir.

Anahtar yapılarında 11 alt modül bulunur. Bunlardan INET kütüphanesi basit modülü olan NotificationBoard (bildirim panosu), kendini barındıran bileşik modüllerin üzerlerinde oluşan olaylarla ilgili birbirini bilgilendirmesinde görev alır. Olaylar, yönlendirme tablolarındaki değişiklikler, arayüz durum ve konfigürasyon değişiklikleri, iç düğümlerde değişiklikleri kapsar. Sunucu ya da yönlendirici modüllerine eklenir. Yalnız bir adet bulunur, olayın olduğu ve olaydan haberdar olmak isteyen modül iletişimini sağlar. Her modülde aynı isme sahip olmak zorundadır, çünkü bu sayede ağdaki herhangi başka bir modüldeki benzeriyle iletişim kurabilir. Bu gerekliliği sağlamak üzere, INET paketinde yapılan sınırlama ile "notificationBoard" olarak adlandırılması zorunlu kılınmıştır[60].

İkinci alt modül InterfaceTable (arayüz tablosu), ağ arayüz (örn: peer-to-peer arayüz) tablolarını tutar, bu tablolara protokolden bağımsız değişkenleri kaydeder. Önceki alt modülde olduğu gibi isim sınırlandırılmasına tabidir. İsmi interfaceTable olmak zorundadır. Alt modüllerden bir diğeri RoutingTable (yönlendirme tablosu)'dur. Tüm fonksiyonelliği C++ dosyası tarafından sağlanır. Tablo dosyaya yazılır ve oradan

okunarak işlemler yapılır. Bu özelliği, programın üzerinde çok fazla giriş-çıkış işlemi yoğunluğuna neden olduğundan yönlendirme simülasyonlarında kullanılabilecek düğüm sayısı oldukça kısıtlıdır.

Yine INET paketinden gelen alt modül Buffer (tampon), mesaj ve paketlerin saklanması, silinmesi ve iletilmesi için kullanır. Kapasite kullanıcı tarafından belirlenebilir. Beşinci alt modül Flow_Table (akış tablosu) akış girdilerinin yapılması, silinmesi ya da tablo içeriğinin okunması işlevlerini görür.

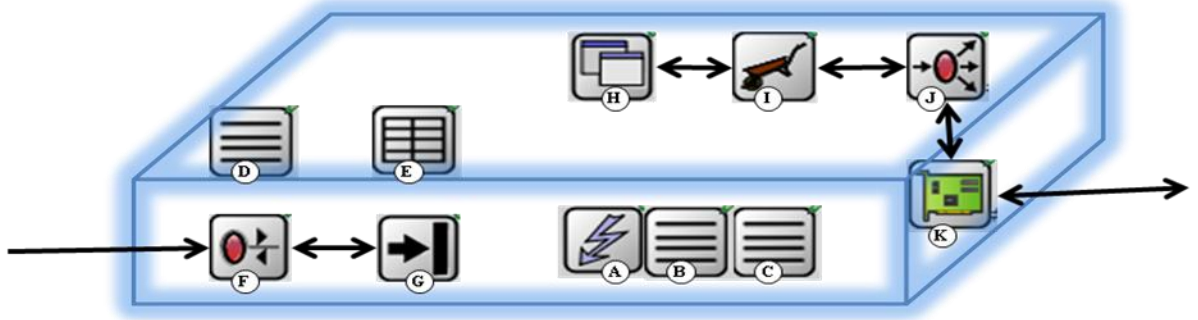
Open_flow_Processing (OpenFlow işleme), Ofomnet'in kendi modülüdür. Giriş ve çıkış yönünde kapısı vardır ve alınan pakete göre sinyal üretiminde bulunur. Üç durumu aktarmak için sinyal kullanır: eşlenen paket bulunamadı, paket gönderilecek, paket istilas (flooding) gerçekleştirilecek. Modül meşgul durumda ise gelen paketi tampona ekler. Aksi halde simülasyon servis zamanına göre paketi kendine zamanlanmış mesaj olarak gönderir. Bu öz-mesaj zamanı geldiğinde işlenmiş olarak kaydedilir. Akış tablosu ve tampon ile etkileşimdedir. Mesaj, bir akışın parçası ise akış bitene kadar mesajları servis süresine göre kendine gönderir.

EtherMAC (Ethernet MAC katmanı), INET'ten alınan başka bir modüldür. Çerçeveleri kabul eder ve bunların iletimini gerçekleştirir. Üst seviye katmanlardan alınan çerçevelerde, kaynak modülün kendisiyse adresini doldurur, çerçeveleri kuyruğa sokar, protokole göre iletimde bulunur ve üst katmanlardan duraklatma mesajı gelmişse buna uyar. Eğer çerçeve ağdaki başka bir modülden direkt gelmişse bu katman çerçeveyi yine protokole uygun olarak kabul eder. Ayrıca veri doğrulama denetimini yapar ve atandığı moda göre kabul ve ret işlemlerini yapar. Yine duraklatma isteklerini karşılar. Yönlendirici aygıt modüllerinde bileşik modülün tamponunu kuyruk için kullanır, sunucularda ise kendi iç kuyruk mekanizması olan txQueue yapısını kullanır. Bu yapı taşıdığı simülasyon hata vererek sonlanır.

Ofomnet paketinde tanımlanmış OFA_switch alt modülü (Şekil 3.3) YTA denetim düzlemine aittir. Direkt TCP modülüne bağlıdır ve denetleyici-yönlendirici arasındaki özel iletişim hattını yönetir. Yönlendirici tablolarında eşlemesi olmayan bir paket aldığı anda bunu denetleyiciye gönderir. Tüm paketi ya da yalnız başlığını göndermek

için `sendCompletePacket` (tüm paketi gönder) parametresine *true* ya da *false* atanır. Bunun yanında, gelen her paket için istatistiki bilgisini denetleyiciye gönderir.

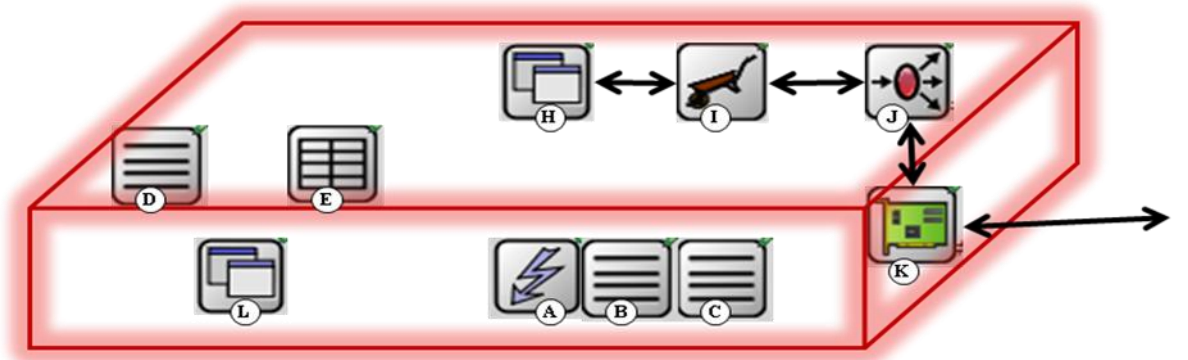
TCP alt modülü, ITCP arayüzü üzerine gerçekleşir. İstemci uygulama, iletim moduna göre paket iletiminde bulunabilir. `TCP_TRANSFER_BYTECOUNT`, fiziki değil yalnız değer olarak belli miktar veri içeren iletimi belirtir. `TCP_TRANSFER_OBJECT` ile istemciden gelen paketler hedefte tekrar üretilir ve `TCP_TRANSFER_BYTESTREAM` ile gelen paket verisi saf haliyle iletilir. TCP aynı zamanda uygulamaya önemli değişikliklerin bilgisini döner. `NetworkLayer` (ağ katmanı) alt modülü iletim katmanı işlemlerini yerine getirir, `EthernetInterface` (Ethernet arayüzü) ise denetleyici kapısı olarak kullanılır.



Şekil 3.3: Ofomnet OpenFlow Anahtarı.

A: Bildirim panosu. B: Arayüz tablosu. C: Yönlendirme tablosu. D: Tampon. E: Akış tablosu. F: Ethernet MAC katmanı. G: OpenFlow işleme modülü. H: OpenFlow anahtarlama modülü. I: TCP. J: Ağ katmanı. K: Ethernet arayüzü.

Ofomnet denetleyicisi içindeki `OFA_controller` alt modülü (Şekil 3.4), kontrol katmanı işlemlerini yapar. Gelen mesajları, servis süresine göre, kendine zamanlanmış olarak gönderir. Mesajın başlığı gelmişse onunla ilgili, tümü gelmişse tümüyle ilgili istatistik kaydı tutar. Ardından mesajı algoritmaya göre bir sunucuya gönderir. `CtrlBehavior` alt modülü ise gelen mesajın üzerinde gerçekleştirilecek algoritmanın uygulandığı yapıdır.



Şekil 3.4: Ofomnet OpenFlow Denetleyici.

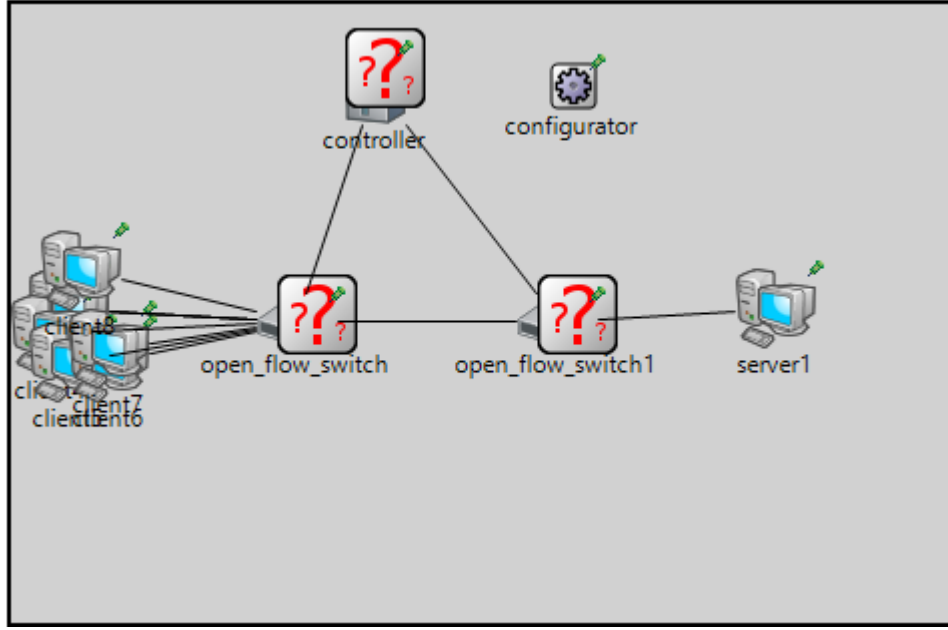
A: Bildirim panosu. B: Arayüz tablosu. C: Yönlendirme tablosu. D: Tampon. E: Akış tablosu. H: OpenFlow anahtarlama modülü. I: TCP. J: Ağ katmanı. K: Ethernet arayüzü. L: Denetleyici davranış modülü.

INET'ten türetilen modüller sayesinde denetleyici ağ üzerinde gerçekleşen birçok olayı istatistik bilgisi olarak kayıt altına alır.

3.2 VERİNİN TOPLANMASI VE DÜZENLENMESİ

n adet akışın üretilip sink görevi gören tek bir düğüme yönlendirildiği topoloji “dumbbell” olarak adlandırılır. Tüm akışlar bağlantıda sağlanan sonlu bant genişliğini kullanmak için rekabet eder. Gerçek ağlarda kaynak ve hedef düğümler arasında birçok başka düğüm bulunur. Tipik olarak tıkanıklık kontrol mekanizmalarında tıkanıklık oluşturan bağlar dışında kalan ara bağlantılar göz ardı edilir[62].

Şekil 3.5'te, sistemi eğitmek için kullanılan dumbbell topolojisi görülmektedir. Bağlantılar, verimlik ölçümü istatistiği üretmek üzere INET kütüphanesinin ThruputMeteringChannel sınıfından türetilmiştir. Bağlantı, 8 adet istatistik verisini simülasyonun her anında üretir (Tablo 3.1). Ortalama değerler, simülasyon başından ölçüm yapıldığı andaki zaman süresince geçen veri üzerinde hareketli ortalama yöntemi ile yapılır. Simülasyon sona erdiğinde sonuç dosyasına son değer kaydedilir. Bağlantı kanallarında 1 μ s gecikme ve 10 Mbps aktarım hızı modellenmiştir.



Şekil 3.5: Dumbbell topolojisi.

Tablo 3.1: Standart üretilen istatistikler.

PARAMETRE	AÇIKLAMA
N	Paket sayısı
V	Veri hacmi (byte)
p	Güncel paket/sn oranı
u	Güncel kanal verimi (%)
P	Ortalama paket/sn
B	Ortalama bant genişliği
b	Güncel bant genişliği
U	Ortalama kanal verimi (%)

Farklı sayıda akış üreteçleri ile 5 yoğunluk senaryosu tasarlanmıştır (Tablo 3.2). Yoğunluk senaryolarında 10'ar oturum gerçekleşmektedir. Büyük veri akışı üreteçleri

olarak tasarlanan istemcilerin her biri, her oturumda 10 GB akış gerçekleştirmektedir. Senaryolarda tüm tıkanıklık kontrol mekanizmaları devre dışı bırakılmıştır.

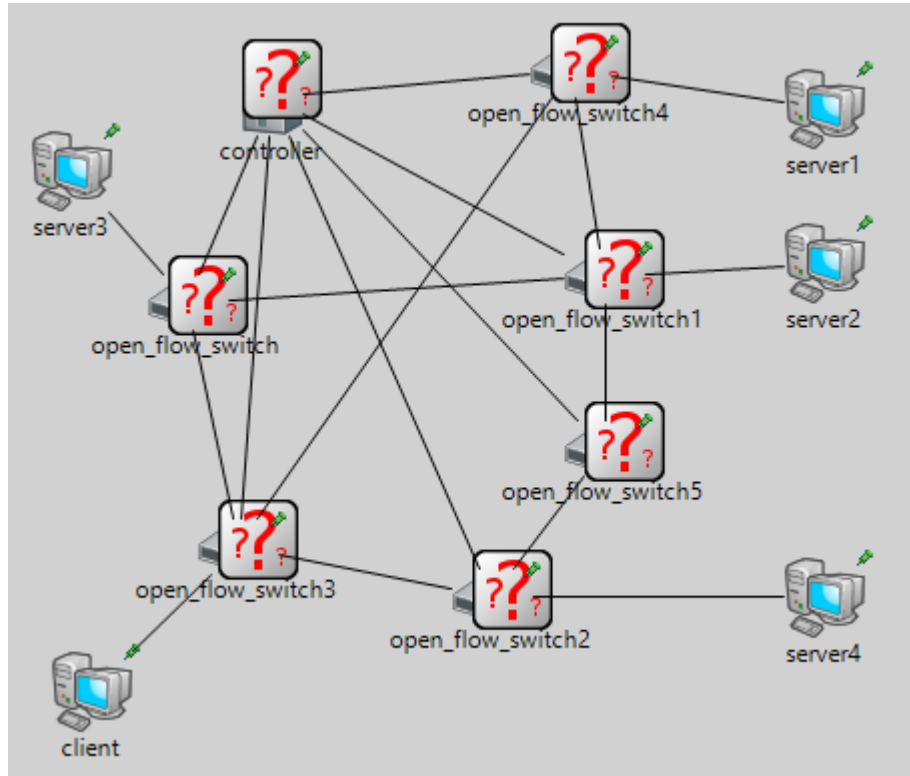
Tablo 3.2: Yoğunluk senaryoları.

SENARYO İNDEKSİ	ÜRETEÇ SAYISI
1	2
2	3
3	4
4	6
5	8

Yöntemin test edilmesi için ikinci bir topoloji oluşturulmuştur (Şekil 3.6). Eşitlik 3.1'de komşuluk matrisi görülmektedir. 6 anahtardan oluşan topolojide, bunlardan 5'ine üreteç bağlanmıştır. Üreteç, uniform dağılımdan [1GB,4GB] aralığında değer üretir. Önceki topolojide olduğu gibi, tüm tıkanıklık kontrol mekanizmaları devre dışı bırakılmıştır. Yönlendirme, Dijkstra'nın en kısa yol algoritması ile sağlanmıştır. İstemci üretecin (client) doğrudan bağlantısı olmayan 2 numaralı sunucuyla (server2) olan veri alışverişi incelenmiştir. Anahtarların sayısı ve birbirlerine olan bağlantı sayıları artırılarak genişletilen yönlendirme sisteminin akış tabloları eşlemesi hesaplama maliyeti önceki topolojiye göre oldukça yüksektir. Böylece, pratikte karşılaşılabilecek bir senaryonun oluşturulması amaçlanmıştır.

Simülasyon boyunca, giden paketlerin durumu ve sistemde oluşturdukları çıktılar Bölüm 4'te açıklanmaktadır.

$$KM = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (3.1)$$



Şekil 3.6: Doğrulama topolojisi.

3.3 ALGORİTMA

Önerilen tıkanıklık öngörü sistemi girdisini oluşturan verinin zaman serisi olduğu önceki bölümlerde belirtilmiştir. Zaman serileri Eşitlik 2.19'daki gibi örneklendirilerek veri kümesi oluşturulur. OMNET++, bir ayrık zamanlı simülasyon oluşturma programı olduğu için örnekleme kendi içinde gerçekleştirir. Sistemimizde, belli miktarda paketin ulaştırılmasının tamamlanması sürecinde geçen zaman üzerinden örneklenen veri simülasyondan çekilir. Verinin örneklenmesi sırasında Bölüm 4'te açıklanan istatistik değerleri de oluşturulur. Veri kümesinde öznitelikler bu değerlerin etiketlerinden oluşur.

Tez kapsamında işlenen öngörü sistemine algoritma sırası ile bakıldığında, örnekleme, istatistik bilgisi oluşturma, ölçekleme, öznitelik seçimi ve YSA öğrenmesi adımlarının izlendiği görülmektedir. Bu adımlar hakkında bilgiler genel özellikleri ile Bölüm 2.3'te verilmiştir. Bu bölümün ilerleyen kısmında öngörü kapsamında adımların özelleşmiş halleri anlatılmaktadır. Verinin simülasyon ortamında işleniş şekli Bölüm 4'te ayrıntılı anlatılmıştır.

Tahminleme sistemine simülasyondan çekilen saf veri verilmemiştir. Veriler çok büyük aralıklarla değişmektedir. Aralıkları ölçeklenen veri, optimizasyon algoritmalarında çok daha hızlı yakınsamaya uğrar. Simülasyondan elde edilen veri kümesinde değerler [0,1] aralığına Eşitlik 3.2 kullanılarak ölçeklenmiştir.

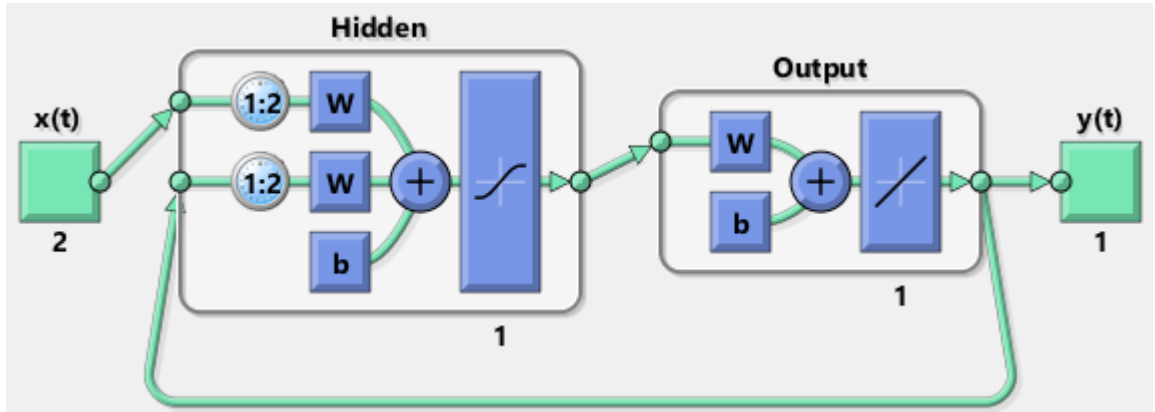
$$X_{ölçeklenmiş} = \frac{X - \min(X)}{\max(X) - \min(X)} \quad (3.2)$$

Boyut azaltmanın gerekliliği ve yöntemi Bölüm 2.3.2.1'de açıklanmıştır. Öznitelik seçimi yapılırken En Az Fazlalık En Çok İlgililik (Redundancy Maximum Relevance, mRMR) [63] yöntemi kullanılmıştır. Yöntem çıktılarına göre ilgili öznitelikler ile oluşturulmuş veri kümesi, öğrenme algoritmasına girdi olarak sağlanmıştır.

mRMR, öznitelik seçimi problemine fazlalık oluşturan altkümeleri öznitelik kümesinden çıkararak işleme başlar. Sınıf bilgisi ile yüksek korelasyonu sağlayıp ilgisi en yüksek öznitelikler seçim grubunda kalır, diğerleri kümeden çıkarılır.

Otoregresif modeller, çok küçük zaman aralıklarında gerçekleşen olayların modellenmesinde kullanılır. Modele göre, çıktı değişkeni stokastik olarak bir önceki değerlerine bağlıdır. NARX[64] da eksojen -yani dışarıdan girdi alan- girdi değişkeni olarak bir sonraki zaman adımının öngörüsünü gerçekleştirir (Şekil 3.7). Buna göre model, şimdiki zaman adımındaki değerlere bakarak ve geçmiş zaman adımını kullanarak bir sonraki değeri elde etmeye çalışır. Eşitlik 3.3'te olduğu gibi ifade edilen yöntemde, y çıktı değeri, x ise zaman serisine sağlanan veri kümesi girdi değerlerini göstermektedir. Gecikme adımı belirlenerek, ilgili değerler sisteme verilir ve MSE yöntemi ile oluşan hatalar hesaplanır.

$$y(t) = f(x(t-1), x(t-2), \dots, y(t-1), y(t-2), \dots) \quad (3.3)$$



Şekil 3.7: Tıkanıklık öngörü sistemi: 2 öznelikli veri setini girdi olarak alan, 2 adım gecikmeli ve 1 gizli katmandan oluşur.

Öngörü sisteminin eğitiminde Levenberg–Marquardt algoritması kullanılmıştır [65-67]. Bu algoritma, doğrusal olmayan en az kare problemlerinde kullanılır. Hızlı hesaplanabilir ve kararlı olması bu algoritmanın seçilme nedenidir. Ayrıca, küçük ve orta büyüklükte problemler için uygun olduğu literatürde belirtilmektedir [65]. Tezin uyguladığı senaryoda, bir ağ problemi kapsamında belli aralıklarla toplanan küçük ve orta büyüklükteki veri üzerinde çalışılarak son öngörü değeri sürekli güncellenecektir.

Oluşturulan öngörü modelinin performansının hesaplanmasının yanı sıra performans karşılaştırması iki farklı yapay sinir ağı algoritmasıyla sağlanmıştır: Doğrusal olmayan otoregresif model [68] (NAR, Şekil 3.8) ve doğrusal olmayan ileri beslemeli model (Şekil 3.9).

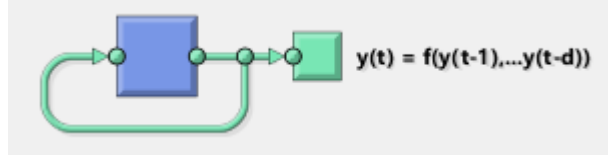
NARX'tan farklı olarak NAR, yalnız hedef verisini belirlenen bir gecikme adımıyla girdi olarak alır. Dışarıdan başka bir giriş kabul etmez. Hedef çıktının geçmişini kullanarak bir sonraki adım için bir tahmin çıktısı verir.

Karşılaştırma için kullanılan diğer yöntem, doğrusal olmayan ileri beslemeli model, dışarıdan girdi almasıyla NARX'a benzer. Farklılaştığı nokta, hedef bilgisini girdi olarak almamasıdır.

Tahminleme sisteminde kullanılan girdi kümesi yine aynı şekilde eğitim-doğrulama-test olarak ayrılmış, eşit katman sayısı kullanılmıştır. Tek fark, girdi parametresi olarak

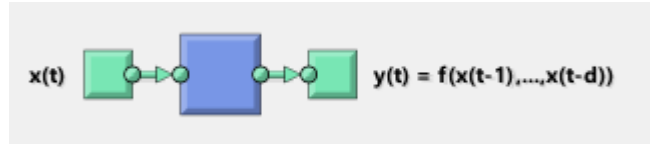
kullanılan değerler ve çıktı fonksiyonudur. NAR modeli tahminleme çıktısı Eşitlik 3.4'te ve ileri beslemeli model tahminleme çıktısı Eşitlik 3.5'te verilmiştir.

$$y(t) = f(y(t-1), y(t-2)) \quad (3.4)$$



Şekil 3.8: NAR ağı yapısı.

$$y(t) = f(x(t-1), x(t-2)) \quad (3.5)$$



Şekil 3.9: Doğrusal olmayan ileri beslemeli model.

4. BULGULAR

OMNET++ üzerinde YTA ağının OpenFlow v.1.0 protokolü kullanılarak Şekil 3.5 topolojisi üzerinde, Bölüm 3.2'de anlatılan yöntemle simülasyonu yapılmış, denetleyici üzerinde toplanan istatistik bilgisi çekilmiştir. (7.4017×10^4) sn süren simülasyondan elde edilen istatistik bilgisinden 24 öznitelik oluşmuş (Tablo 4.1), toplam 48 bağlantı ölçümü yapılmıştır.

Simülasyon ortamında, düğümlerden çıkan paket 4 durum değeri alır: Boşta, duraklamada, beklemede ve iletimde. BOŞTA durumundaki bir paket çerçevesi direkt İLETİMDE durumuna geçemez. Önce DURAKLAMADA veya BEKLEMEDE durumuna geçerek paketin son hali ile ilgili sistemde istatistik bilgisi olarak kaydedilmesi üzere veri üretir. Boştaki çerçeve sırasıyla duraklama ve bekleme safhalarından geçerek iletme başlayabilir. Duraklayan bir çerçeve direkt iletim durumuna varamaz. İletime hazırlık safhası olan BEKLEMEDE durumunda mutlaka bulunması gerekir.

DURAKLAMADA durumu aslında paket iletimi için bir hayalet aşamadır. Kuyruktaki çerçevelerin idaresini gerçekleştirir, o anda iletilen çerçeve ile ilgili değil, sonraki ile ilgilidir. Bu yüzden, burada geçirilen süre ve bu durum ile ilgili diğer verinin kaydedilmesi tıkanıklık sorununu çözmek için önemlidir. Aşağıda, her bir öznitelik anlatılırken bu durumdaki birimler için hesaplanan değerler de belirtilmiştir.

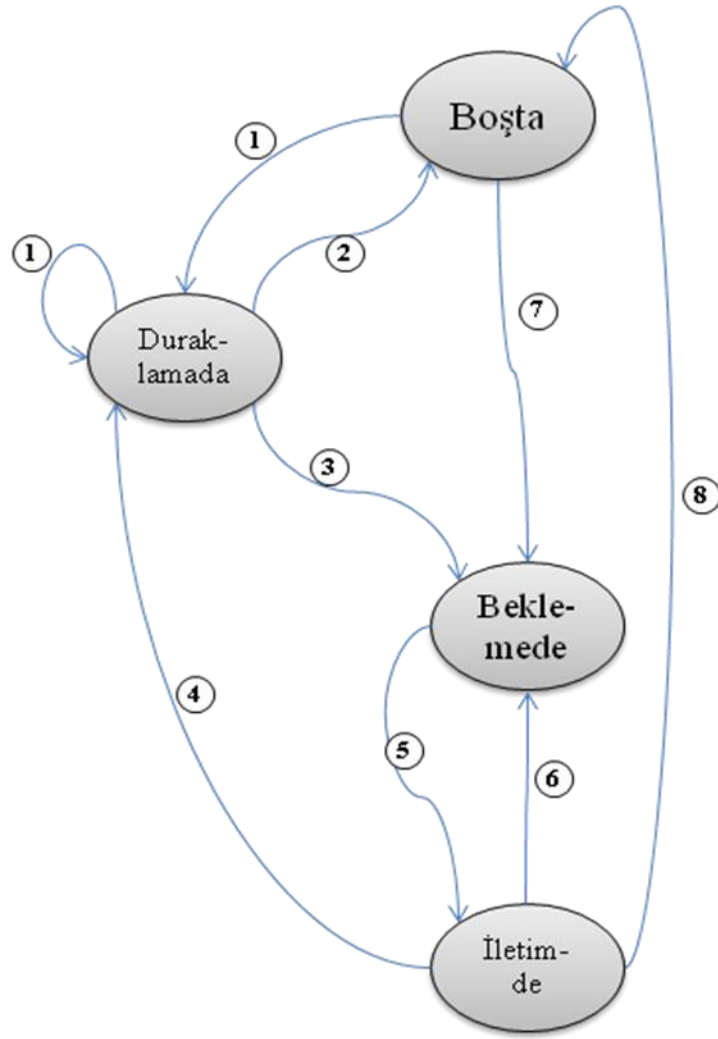
Yoğun veri iletiminin gerçekleştiği ağlarda paketlerin DURAKLAMADA durumu üzerinde öz-çevrim yapıp paket iletim sistemini meşgul etmesi beklenen bir sonuçtur. Duraklama zamanı sona erdiğinde çerçevenin düşerek sistemi BOŞTA durumuna ya da işleme alınarak BEKLEMEDE durumuna getirdiği görülmektedir.

Tablo 4.1: Öznitelik kümesi.

1. alınan bit/sn	2. gönderilen bit/sn	3. bit hatalı paket sayısı	4. bit hatalı gelen toplam bayt
5. MAC bağlı olmadığı için düşen toplam paket sayısı	6. MAC bağlı olmadığı için düşen toplam paket miktarı (bayt)	7. adres hatalı paket sayısı	8. adres hatalı gelen toplam bayt
9. alınan pencere/sn	10. gönderilen pencere/sn	11. çift yönlülük	12. çıkan paket sayısı
13. çıkan paket toplam bayt	14. PAUSE üniteleri sayısı	15. PAUSE üniteleri toplam bayt	16. yüksek katmandan gelen paket sayısı
17. yüksek katmandan gelen toplam bayt sayısı	18. başarıyla alınan pencere sayısı	19. başarıyla alınan pencere toplam bayt	20. toplam simülasyon zamanı
21. duraklayan paket üniteleri için duraklama süresi toplam adet	22. duraklayan paket üniteleri için duraklama süresi toplam saniye	23. başarılı gönderilen toplam veri çerçevesi sayısı	24. başarılı gönderilen toplam veri çerçevesi miktarı (bayt)

Özniteliklerden ikisi gönderilen ve alınan veri miktarını gösterir. Simülasyon boyunca denetleyicinin yönettiği anahtarlar üzerinden giden ve ona gelen bit sayısı kayıt altına alınmıştır. Kayıtlar, veri akış hızını bulmak için kullanılmıştır. Ulaşan bit/sn, ve gönderilen bit/sn olarak öznitelik olarak yapılandırılmıştır.

Bir pencere bit hatasıyla alındığında bilgisi kaydedilir. Bu değer toplam bayt ve pencere sayısı olarak otomatik kaydedilmektedir. Paket, MAC bileşeni bağlı olmadığı için düşüyse bu ve benzer durumlarda toplam bayt ve paket sayısı olarak sistemde tutulacaktır. Paketi olan modülün MAC adresi, pakette belirtilenden farklı ise o paket düşürülür. Bu da toplam bayt ve paket sayısı olarak alınmaktadır. Giden paket durum diyagramı Şekil 4.1'de verilmiştir.



Şekil 4.1: Giden paket durum diyagramı.

- 1: DURAKLAMADA olan çerçeve alındı. 2: Duraklama zamanı sona erdi. 3: Kuyruktaki mesajlar için duraklama zamanı sona erdi.
 4: İletim bitti, DURAKLAMADA olan çerçeve alındı. 5: Bit zamanı sona erdi. 6: Kuyruktaki mesajlar için iletim bitti,
 DURAKLAMADA olan çerçeve yok. 7: Üst seviye katmandan çerçeve alındı. 8: İletim sona erdi, kuyruktaki mesaj yok.

DURAKLAMADA durumu dışında başarıyla alınan veri pencereleri, ayrı bir parametre olarak değerlendirilmektedir. Bu durumu belirten veri penceresi gönderme hızı ve veri penceresi alma hızı da öznitelikler arasındadır.

Çift yönlülük Boolean bir değerdir. Verinin aynı anda gidiş ve geliş yönlerinde çalışıp çalışmadığını belirtir. Başarılı olarak alınıp alınmadığına bakılmaksızın gönderilen (modülden çıkan) paketler de hem adet, hem toplam veri büyüklüğü olarak kaydedilir. DURAKLAMADA durumu gerçekleştiğinde bunun ne kadar sürdüğü hesaplanır. Bu DURAKLAMADA üniteleri de yine iki ölçüm değeriyle saklanır.

Veri, ağda daha yüksek bir katmandan geliyorsa o veri penceresinin bildirimi yapılır. Veri pencereleri başarıyla alındığında toplam değerleri için bilgi kaydeder. DURAKLAMADA çerçevesi gönderildikten sonra, geçen duraklama süresi duraklayan paket üniteleri için toplam adet ve saniye olarak saklanır. Veri çerçevesi başarılı olarak gönderildikten sonra toplam başarılı gönderme sayısı bir artırılır ve paket büyüklüğü toplam paket büyüklüğüne eklenir.

mRMR boyut azaltması için sınıf bilgisine ihtiyaç duyulmaktadır. Bunun için yine toplanan istatistiklerden "kanal verimi" kullanılmıştır. Kanal verimi, bağlantının yoğunluğunu gösteren parametredir. Bu parametre [0,100] arası değer alır; 0 iken boş, 100 iken tam kapasite dolu olduğunu gösterir. Eşik değer olarak 0,5 verilmiş, veri kümesi 0,5 verimliliğe eşit veya bundan yüksek (1) ve 0,5 verimliliğin altında (0) olarak ikili sınıf yapısına getirilmiştir. Öznitelik seçimi algoritması uygulandığında birçok özneliliğin fazla, bazılarının da gereksiz olduğu görülmüştür (Tablo 4.2).

Tablo 4.2: mRMR sonucu tespit edilen gereksiz ve fazlalık oluşturan öznelilikler.

GEREKSİZ	FAZLA
3.bit hatalı paket sayısı	1.alınan bit/sn
4.bit hatalı gelen toplam bayt	2.gönderilen bit/sn
5.MAC bağlı olmadığı için düşen toplam paket sayısı	7.adres hatalı paket sayısı
6. MAC bağlı olmadığı için düşen toplam paket (bayt)	8.adres hatalı gelen toplam bayt
11. çift yönlülük	9.alınan pencere/sn
14. PAUSE üniteleri sayısı	13. çıkan paket toplam bayt
15.PAUSE üniteleri toplam bayt	16. yüksek katmandan gelen paket sayısı
21. duraklayan paket üniteleri için duraklama süresi toplam adet	17. yüksek katmandan gelen toplam bayt sayısı

Tablo 4.2 (devam): mRMR sonucu tespit edilen gereksiz ve fazlalık oluşturan öznitelikler.

GEREKSİZ	FAZLA
22. duraklayan paket üniteleri için duraklama süresi toplam saniye	18.başarıyla alınan pencere sayısı
	19. başarıyla alınan pencere toplam bayt
	20.toplam simülasyon zamanı
	23. başarılı gönderilen toplam veri çerçevesi sayısı
	24. başarılı gönderilen toplam veri çerçevesi miktarı (bayt)

mRMR sonucuna göre "çıkan paket sayısı", veri kümesini 0,65 oranında ilgi ile temsil etmekte, "gönderilen pencere/sn" ise 0.058 oranında ilgililik eklemektedir. Diğer tüm özniteliklerin ya gereksiz, ya da fazla olduğu anlaşılmıştır. Bu yüzden, öngörü sisteminde yalnız bu iki öznitelik kullanılmıştır (Tablo 4.3).

Tablo 4. 3: Veri kümesinde ilgili bulunan öznitelikler ve ilgi oranları.

ÖZNİTELİK	mRMR İLGİLİLİK ORANI
12. çıkan paket sayısı	0,65
10.gönderilen pencere/sn	0,058

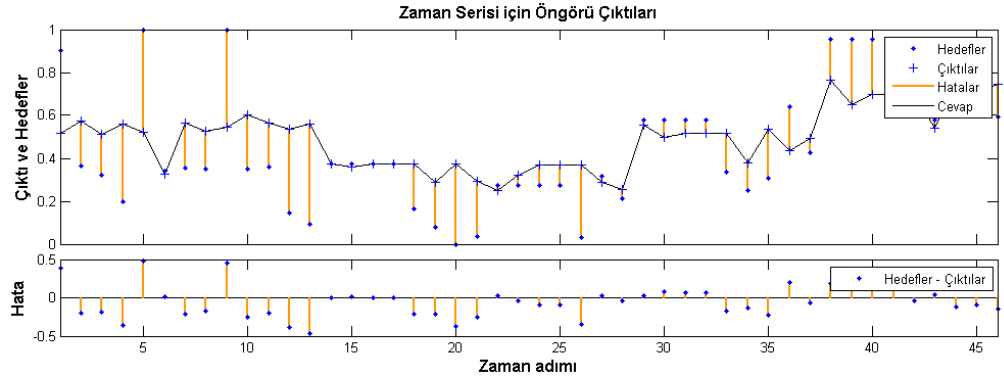
YTA'da trafik analizi ve yönetiminin büyük kısmı, YTA'nın uygulama seviyesinde gerçekleştirildiği Bölüm 2.2 ve alt başlıklarında belirtilmiştir. Tezde geliştirilen tıkanıklık öngörü yöntemi de kontrol düzleminde, bir ağ uygulamasına entegre edilmesi senaryosuyla hazırlanmıştır. Bu yüzden, mümkün olduğu kadar az parametre kullanıp hesaplama karmaşasını azaltmak hedeflerden biridir. Öznitelik sayısını 2'ye indirdikten

sonra NARX yapay sinir ağında da performanstan fazla ödün vermeden parametre azaltmasına gidilmesi uygun görülmüştür.

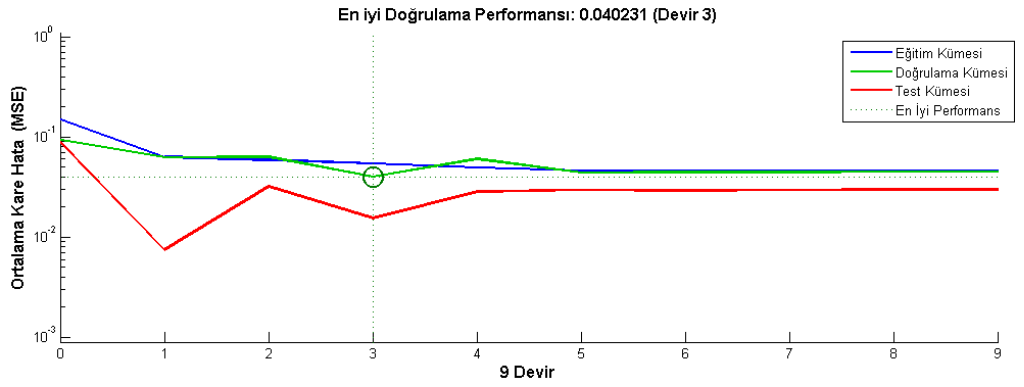
Verinin fazla örtüşmesi, yapay sinir ağı gizli katmanlarının artmasıyla ortaya çıkabilen bir durum ve sistemde öğrenme yerine ezberleme oluşturan bir olgudur [68]. Fazla gizli katman kullanımının bir başka olumsuz yanı ise hesaplama karmaşıklığını artırmasıdır. Bu nedenle, sistemde ara katman sayısını en aza indirmek hedeflenmiştir. Sistem $h=\{1,2,5,10,15\}$ ara katman değerleri ile çalıştırılmış ve performans ölçümü yapılmıştır. Tablo 4.4'te özetlenen sonuçlar Şekil 4.2-4.11'de her h_i için grafiklerde ayrıntılı olarak verilmiştir.

Tablo 4.4: Katman sayısının performansa etkisi.

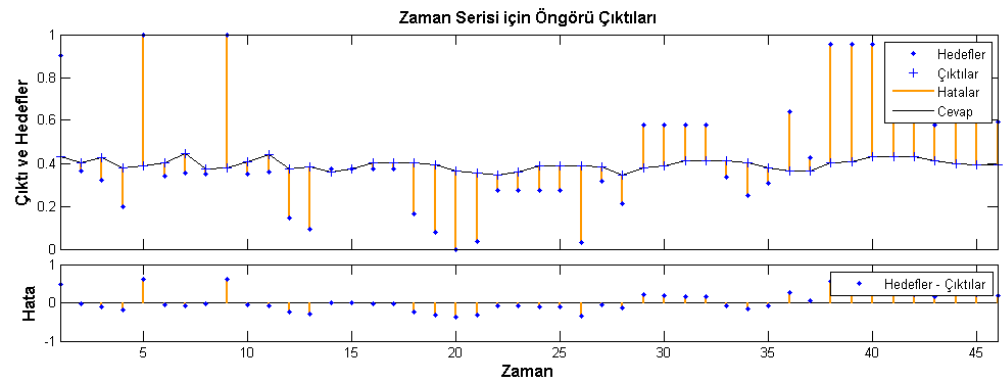
ARA KATMAN SAYISI	SÜRE (sn)	DOĞRULAMA KÜMESİNDE EN İYİ PERFORMANS (MSE)	SONUÇ PERFORMANSI (MSE)
1	1,0477	0,040	0,0503
2	1,1355	0,048	0,0426
5	0,9371	0,013	0,0346
10	1,0786	0,024	0,0559
15	0,6932	0,063	0,0265



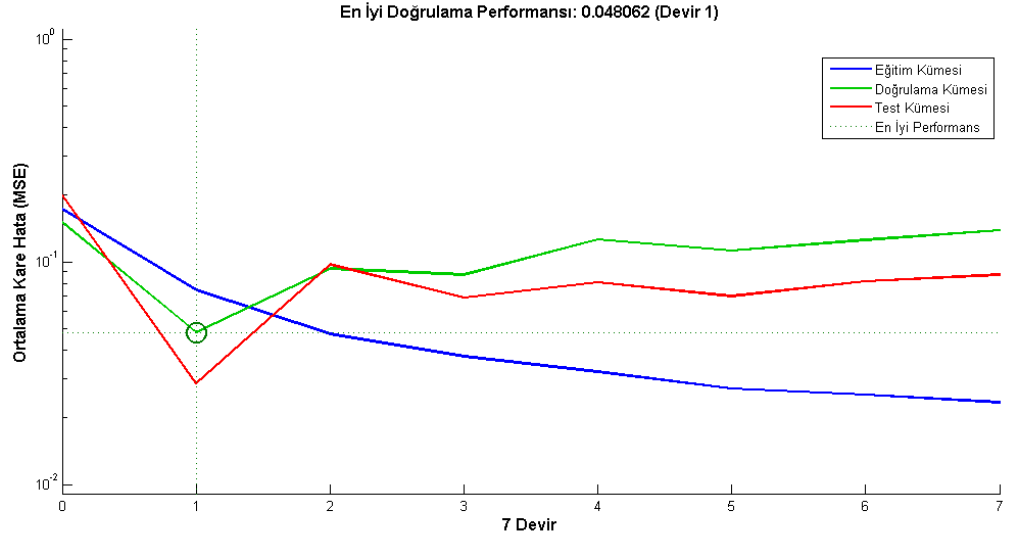
Şekil 4.2: 1 ara katman ile yürütülen algoritma için öngörü çıktıları.



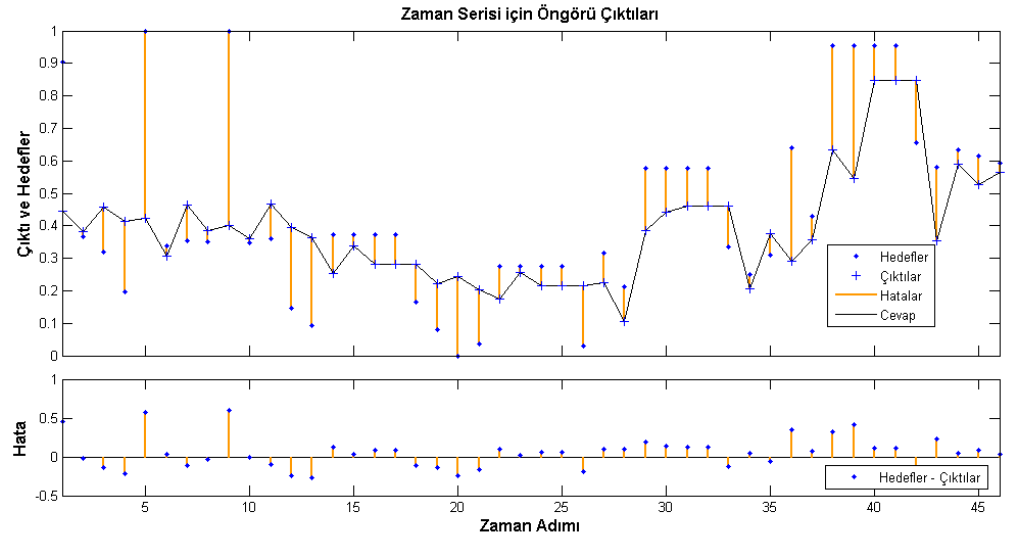
Şekil 4.3: 1 ara katman ile yürütülen algoritma için performans durumu.



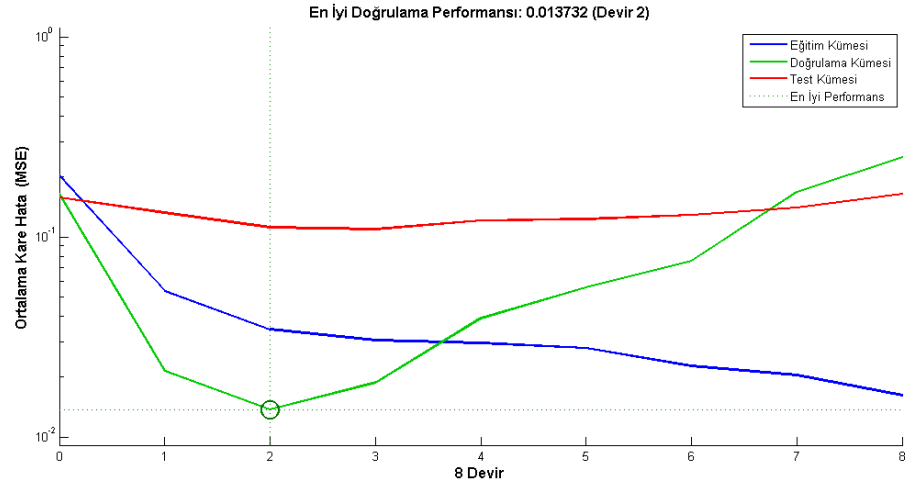
Şekil 4.4: 2 ara katman ile yürütülen algoritma için öngörü çıktıları.



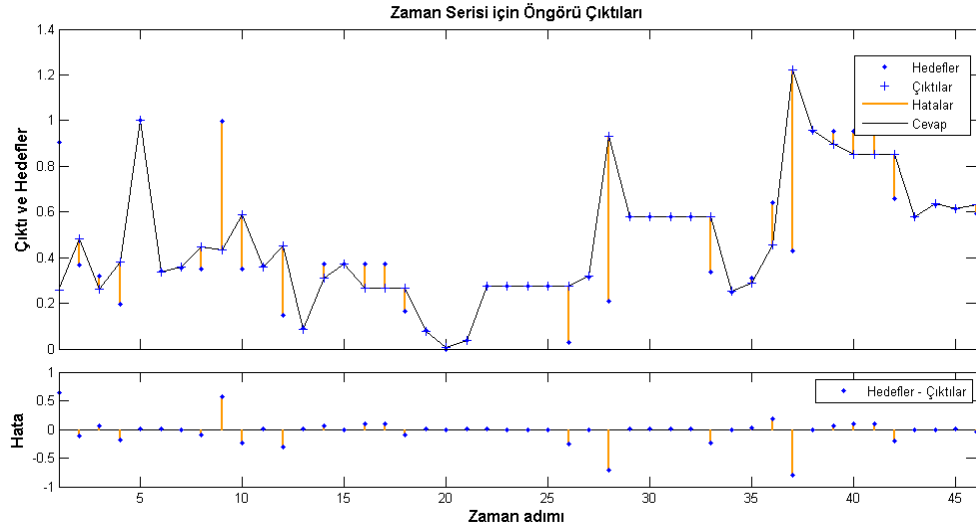
Şekil 4.5: 2 ara katman ile yürütülen algoritma için performans durumu.



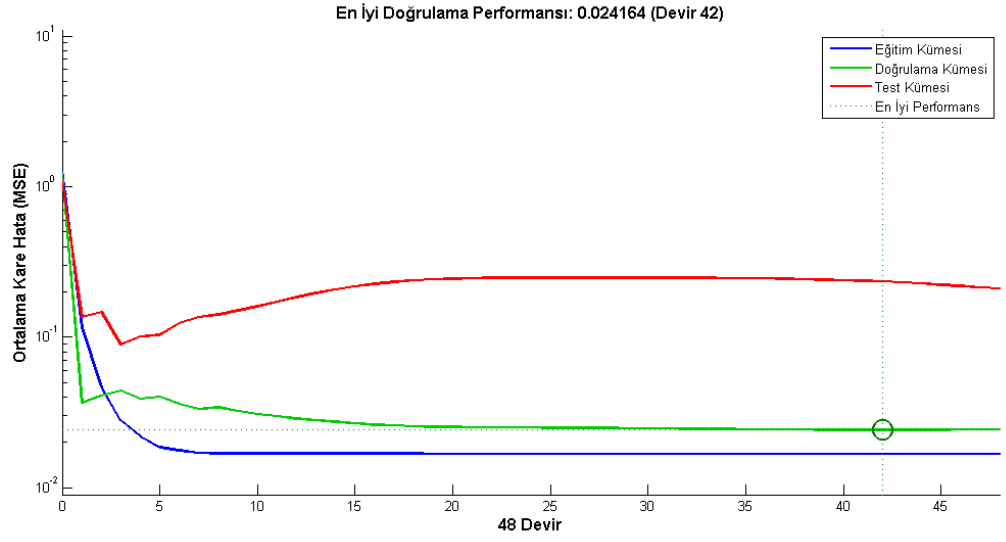
Şekil 4.6: 5 ara katman ile yürütülen algoritma için öngörü çıktıları.



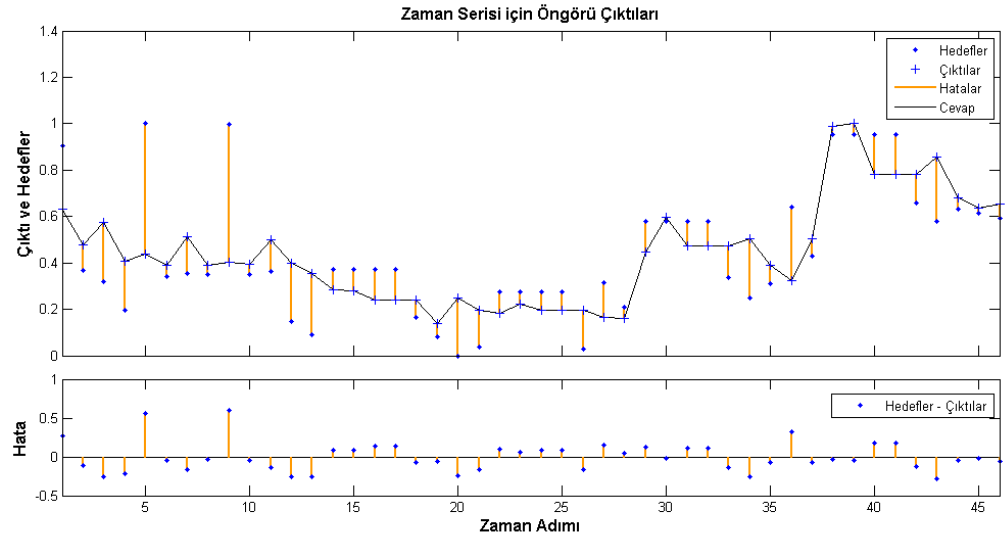
Şekil 4.7: 5 ara katman ile yürütülen algoritma için performans durumu.



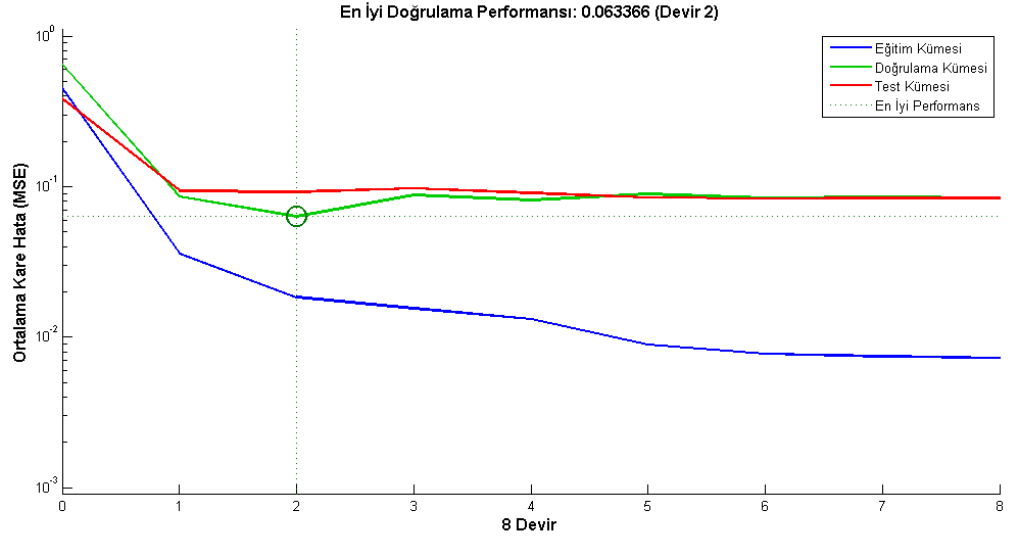
Şekil 4.8: 10 ara katman ile yürütülen algoritma için öngörü çıktıları



Şekil 4.9: 10 ara katman ile yürütülen algoritma için performans durumu.



Şekil 4.10: 15 ara katman ile yürütülen algoritma için öngörü çıktıları.

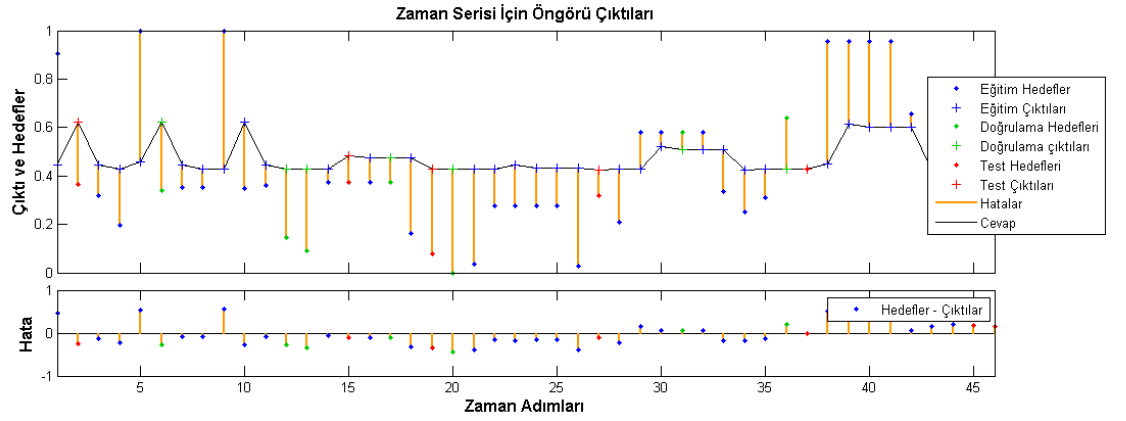


Şekil 4.11: 15 ara katman ile yürütülen algoritma için performans durumu.

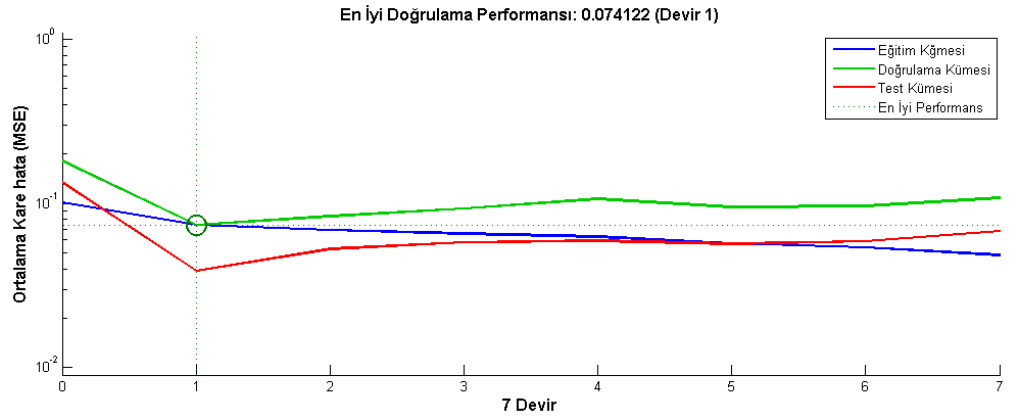
NAR ve doğrusal olmayan ileri beslemeli model üzerinde karşılaştırılma yapıldığında Tablo 4.5'teki sonuçlar ve Şekil 4.12-4.15'teki grafikler elde edilmiştir. Saklı katman sayısına 1 ve gecikme adımına 2 değerleri verilmiştir.

Tablo 4.5: Performans karşılaştırması.

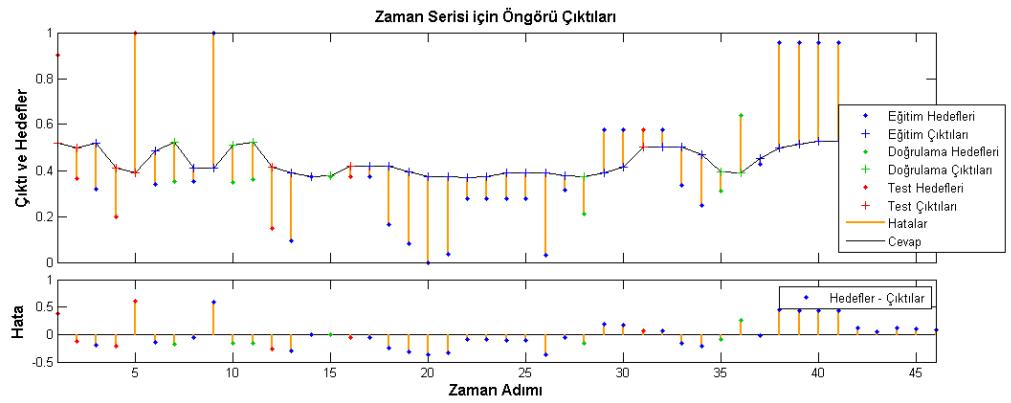
ALGORİTMA	DOĞRULAMA KÜMESİNDE EN İYİ PERFORMANS (MSE)	DOĞRULAMA KÜMESİNDE SONUÇ PERFORMANSI (MSE)
NARX	0,040	0,050
NAR	0,025	0,064
Doğrusal olmayan ileri beslemeli	0,074	0,054



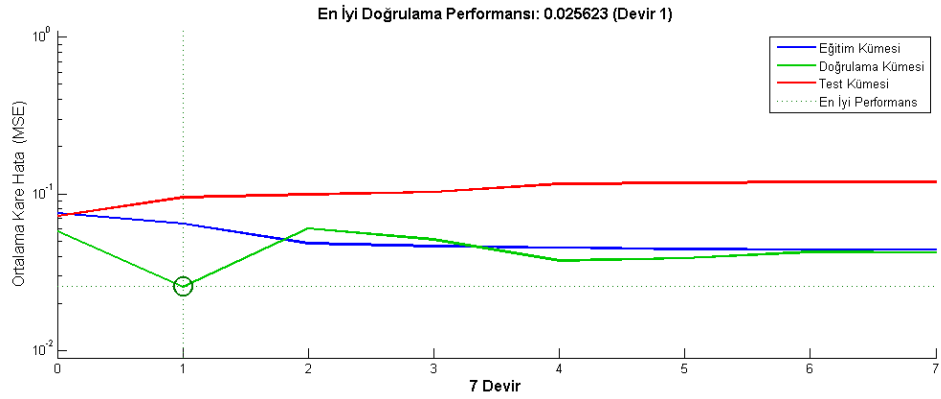
Şekil 4.12: Doğrusal olmayan ileri beslemeli model için öngörü çıktıları.



Şekil 4.13: Doğrusal olmayan ileri beslemeli model performans durumu.



Şekil 4.14: NAR modeli için öngörü çıktıları.



Şekil 4.15: NAR modeli için performans durumu.

Şekil 3.6 ile gösterilen doğrulama topolojisinde simülasyon sırasında oluşan istatistik bilgisinden yalnız Tablo 4.3'teki indirgenmiş, iki öznitelik toplanmıştır. Bu haliyle simülasyon süresi, daha karmaşık yönlendirme şeması olduğu halde yaklaşık 44,59 kat azalmıştır. Çok fazla miktarda veri kaydı tutmanın hesaplama zamanına etkisi açıkça görülmektedir.

Tüm algoritma parametrelerine önceki topolojideki değerler verilmiş, tek ara katman kullanılmıştır. NARX, NAR ve doğrusal olmayan ileri beslemeli YSA algoritmaları performans sonuçları karşılaştırılmıştır.

Tablo 4.6: Performans karşılaştırması-2.

ALGORİTMA	DOĞRULAMA KÜMESİNDE EN İYİ PERFORMANS (MSE)	DOĞRULAMA KÜMESİNDE SONUÇ PERFORMANSI (MSE)
NARX	0,026	0,033
NAR	0,031	0,060
Doğrusal olmayan ileri beslemeli	0,071	0,115

5. TARTIŞMA VE SONUÇ

Kontrol ve yönetim düzlemlerinin dikey entegre yapıda bulunduğu geleneksel ağlarda, her ağ cihazında bulunan bu iki düzlem iç içe geçmiştir. Karmaşıklığın yüksek olduğu ve literatürde çoğunlukla statik mimarilere neden olduğu görülmekte olan geleneksel ağların inovasyon maliyeti yüksek çıkmaktadır. Geleneksel ağda görülen bu zorlukların önüne geçmek için YTA yaklaşımı ile düzlem ayırımına gidilmiştir. Yönlendirme görevi ağ aygıtlarında, kontrol görevi ise aygıtların üst seviyesinde yer alan merkezi denetleyicide gerçekleştirilir. YTA programlama ve soyutlamayı mümkün kıldığı için esnek bir yapı sunar. İlgili alanlarını birbirinden ayırdığı için inovasyon daha hızlı gerçekleşir. Hedef adreslemesi yerine akış tabanlı yapının kullanıldığı YTA'da trafiğin yönlendirilmesi, gelen akışın ağ aygıtlarındaki akış tablolarında eşlenip ilgili işlemin gerçekleştirilmesi şeklinde olur. Denetleyici üzerinde çalışan ağ uygulamaları yoluyla alt bağlamlardaki aygıtların yönetilmesi, ağın programlanması gerçekleştirilir.

Tıkanıklık kontrolü, ağ trafik akışının gözlemlenmesi ile gerçekleştirilir. OMNET++ simülasyon programında oluşturulan topolojiler üzerine rastgele büyüklükte veri üreten düğümler yerleştirilmiş, OpenFlow versiyon 1.0'ın uygulandığı anahtar ve denetleyici elemanları bağlantıları gerçekleştirilmiştir. Simülasyonda, veri paketlerinin ağda dolaşması sırasında oluşturduğu değişimler kaydedilmiş ve istatistik bilgisi olarak toplanıp veri kümeleri oluşturulmuştur. Veri kümeleri üzerinde gerçekleştirilen ölçekleme ve öznitelik seçimi işlemleri sonucunda, kümenin 2 adet öznitelik ile büyük oranda açıklanabildiği hesaplanmıştır. Bu işlemler sonucu oluşan 2 boyutlu veri vektörleri tıkanıklık öngörüsü için NARX, NAR ve doğrusal olmayan ileri beslemeli YSA'lara verilmiştir. MSE performansı incelendiğinde, NARX'ın diğer iki yöntemle göre toplam performansta daha iyi sonuç verdiği kaydedilmiştir.

Öngörü sistemimizde hesaplama yükünü mümkün olduğu kadar azaltmak isteriz. Bu şekilde, ağda sonraki adım tahmini daha hızlı gerçekleşebilecektir. Doğrulama topolojisinde, yalnız indirgenmiş öznitelik kümesi istatistiği ağdan toplandığında, simülasyon süresinde 44 kattan fazla azalma görülmüştür. Hesaplama yükünün önemli

bir kısmı YSA üzerindedir. Bu yüzden, bunun yapısını da mümkün olduğu kadar indirgemek isteriz. 1, 2, 5, 10 ve 15 ara katman ile NARX çalıştırıldığında toplam performans sonuçları standart sapmasının 0.0118 olduğu hesaplanmıştır. Tablo 4.4'teki her ara katman sayısı senaryosu için gözlemlenen toplam performans sonuçları incelendiğinde ara katman sayısının artmasıyla performansta yaklaşık standart sapma miktarı kadar iyileşme olduğu görülmektedir. Buna göre, tahmin isabetinin çok hassas olması gereksiniminin bulunmadığı ağ servislerinde tek ara katman kullanılarak performanstan az bir miktar feragat edilebilir. Ara katman sayısı 10 olduğunda görülen performans sonucunun, veri kümesinin alt kümelerine ayrılırken rastgele seçim yapılmasından kaynaklanan örnek dağılımının bir sonucu olduğu göz önünde bulundurulmalıdır.

Tez kapsamında gerçekleştirilen çalışma YTA'da trafik yönetiminin kuzey-bağlam ağ uygulamaları seviyesinde programlanması senaryosu ile yola çıkmıştır. Önerilen tıkanıklık kontrol yöntemi ile ağda öngörülen tıkanıklığın aşılması yönünde önlem alınması mümkündür. Belirlenen bir tıkanıklık eşiğinin aşıldığı öngörü çıktılarında, istenmeyen durumun önüne geçmek için ağda ek bir denetleyiciyi aktif hale getirme, yönlendirme yapısını değiştirmek üzere aygıtların akış tablolarına müdahale etme ya da soyutlama yaklaşımının kullanıldığı mimaride ağ aygıtı ya da denetleyicinin sanal nesne olarak çoğaltılması yoluna gidilebilir. Bunların içerik olarak, Bölüm 2'de taraması yapılan ürün ve yaklaşımlar ile gerçekleştirilebilir olduğu görülmektedir.

Önerilen sistem bir kısım kısıtlar ile sınırlıdır. Bunlardan ilki simülasyon ortamının yetersiz olanaklarıdır. Ağda iletilen verinin sanal olarak, içerik bilgisi bulunmadan üretiliyor olması toplanabilen öznitelik sayısını sınırlamaktadır. Gelecek çalışmalarda, multimedya verisi gibi günümüz ağlarında yoğun trafiğe neden olabilen veri içeriğinden öznitelik kümesine eklenmek üzere veri çekilmesi değerlendirilen bir senaryodur. Genişletilen veri kümesi, yine önerilen ölçekleme ve öznitelik seçimi adımlarından geçirilerek veri kümesini daha iyi açıklayan öznitelik alt kümesi bulunulabilir. Ardından, önerildiği üzere öğrenme ve öngörü adımları izlenecektir. Gelecekteki çalışmada, fiziksel laboratuvar koşullarında YTA yapısı kurularak ağdaki veri alışverişini daha gerçekçi örnekleyen bir sistem üzerinde çalışılmasının gerekli olduğu düşünülmektedir. Bu şekilde, özellikle bir takım özel tipteki veri alışverişi hizmeti veren

veri merkezleri modellenerek sektöre uygun öznitelik seçimi yapıp tıkanıklığın servise özel parametreleri belirlenebilecektir. Zaten YTA'nın ilgi alanında bulunan çeşitli tıkanıklık kontrol ve trafik yönetim prosedürlerinin özelleştirilmesi gerekliliği yönüyle böyle bir veri kümesinin oluşturulması gereklidir.

Bir diğer kısıt, nispeten yeni bir teknoloji olan YTA'nın kuzey-bağlam arayüzü ile ilgili standartlaşma eksikidir. Bölüm 2'de belirtildiği üzere, bu konuda bir anlaşmaya varılması için henüz erkendir. Yapılan ön çalışmalarda üzerinde çalışılan ağ kaynaklarının kullanıcı uygulamaları seviyesinde direkt kontrol edilebilirliğine dair senaryo, tezde önerilen yöntem çıktısının kullanılmasına esneklik sağladığı yönüyle ümit vericidir.

KAYNAKLAR

- [1]. Nadeau, T. D., & Gray, K., 2013, *SDN: software defined networks*, O'Reilly Media, Inc.
- [2]. Cisco, *Designing Secure Multi-Tenancy into Virtualized Data Centers*, http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/Virtualization/secureldg.html. [Ziyaret tarihi: Kasım 2015]
- [3]. Allcock, B., Bester, J., Bresnahan, J., Chervenak, A. L., Foster, I., Kesselman, C., ... & Tuecke, S., 2002, Data management and transfer in high-performance computational grid environments. *Parallel Computing*, 28(5), 749-771.
- [4]. Benson, T., Akella, A., & Maltz, D. A., 2009, Unraveling the Complexity of Network Management. *In NSDI*, 335-348.
- [5]. Wang, J., Li, L., Low, S. H., & Doyle, J. C., 2005, Cross-layer optimization in TCP/IP networks. *Networking, IEEE/ACM Transactions on*, 13(3), 582-595.
- [6]. Ghodsi, A., Shenker, S., Koponen, T., Singla, A., Raghavan, B., & Wilcox, J., 2011, Intelligent design enables architectural evolution, *In Proceedings of the 10th ACM Workshop on Hot Topics in Networks* (p. 3). ACM.
- [7]. Kim, H., & Feamster, N., 2013, Improving network management with software defined networking, *Communications Magazine*, IEEE, 51(2), 114-119.
- [8]. Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., ... & Vahdat, A., 2013, B4: Experience with a globally-deployed software defined WAN. *In ACM SIGCOMM Computer Communication Review* (Vol. 43, No. 4, pp. 3-14). ACM.
- [9]. Open Networking Foundation, 2013, *Software Defined Networking: A New Norm of Networks*, <https://www.opennetworking.org/ja/sdn-resources-ja/sdn-library/whitepapers>, [Ziyaret tarihi: Kasım 2015].
- [10]. Schwabe, A., & Karl, H., 2014, Using MAC addresses as efficient routing labels in data centers. *In Proceedings of the third workshop on Hot topics in software defined networking*, 115-120.
- [11]. McKeown, Nick, et al., 2008, OpenFlow: enabling innovation in campus networks, *ACM SIGCOMM Computer Communication Review* 38.2, 69-74.
- [12]. N. Feamster, J. Rexford, and E. Zegura, 2013, *The road to SDN*, Queue, vol. 11, no. 12, pp. 20:20–20:40.

- [13]. ONF, 2014, *Open networking foundation*, <https://www.opennetworking.org/>, [Ziyaret tarihi: Kasım 2015].
- [14]. VMware, Inc., 2013, *VMware NSX Virtualization Platform*, <https://www.vmware.com/products/nsx/>, [Ziyaret tarihi: Kasım 2015].
- [15]. Kreutz, D., Ramos, F. M., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., & Uhlig, S., 2015, Software-defined networking: A comprehensive survey. *proceedings of the IEEE*, 103(1), 14-76.
- [16]. S. Schenker, 2011, *The Future of Networking, and the Past of Protocols*, <http://www.youtube.com/watch?v=YHeyuD89n1Y>, [Ziyaret tarihi: Kasım 2015].
- [17]. Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., & Shenker, S., 2008, NOX: towards an operating system for networks, *ACM SIGCOMM Computer Communication Review*, 38(3), 105-110.
- [18]. Project Floodlight, *Floodlight is an Open SDN Controller*, 2016, <http://www.projectfloodlight.org/floodlight/>, [Ziyaret tarihi: Ocak 2016].
- [19]. Tootoonchian, A., & Ganjali, Y. (2010, April). HyperFlow: A distributed control plane for OpenFlow. In *Proceedings of the 2010 internet network management conference on Research on enterprise networking* (pp. 3-3). USENIX Association.
- [20]. Dix, J., *Clarifying the role of software-defined networking northbound APIs*, 2013, <http://www.networkworld.com/news/2013/050213-sherwood-269366.html>, [Ziyaret tarihi: Kasım 2015].
- [21]. Rothenberg, C. E., Chua, R., Bailey, J., Winter, M., Correa, C. N., de Lucena, S. C., ... & Nadeau, T. D., 2014, When open source meets network control planes, *IEEE Computer Special Issue on Software-Defined Networking*, 47(11), 46-54.
- [22]. ONF, *SDN architecture*, 2014, https://www.opennetworking.org/images/stories/downloads/sdnresources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf, [Ziyaret tarihi: Kasım 2015].
- [23]. Reich J., Monsanto C., Foster N., Rexford J., and Walker D., 2013, Modular SDN Programming with Pyretic, *USENIX ;login*, vol. 38, no. 5.
- [24]. Foster, N., Harrison, R., Freedman, M. J., Monsanto, C., Rexford, J., Story, A., & Walker, D., 2011, Frenetic: A network programming language. In *ACM SIGPLAN Notices*, Vol. 46, No. 9, 279-291.
- [25]. Monsanto C., Foster N., Harrison R., and Walker D., 2012, A compiler and run-time system for network programming languages, *SIGPLAN Not.*, vol. 47, no. 1, 217-230.
- [26]. E. Reinecke, 2014, Mapping the future of software-defined networking, 2014, [http://www.dimensiondata.com/Global/Documents/Dimension%20Data's%20Pr%](http://www.dimensiondata.com/Global/Documents/Dimension%20Data's%20Pr%20)

C3%A9cis%20-%20Mapping%20the%20future%20of%20software-defined%20networking.pdf, [Ziyaret tarihi: Kasım 2015].

- [27]. Hewlett-Packard Company (HP), Hp launches industry's first SDN App Store, 2013, <http://www8.hp.com/h20195/V2/GetDocument.aspx?docname=4AA5-1871ENW&cc=us&lc=en>, [Ziyaret tarihi: Kasım 2015].
- [28]. Hewlett-Packard Company (HP), SDN App Store, 2013, <http://www8.hp.com/us/en/networking/sdn/sdnapps.html>, [Ziyaret tarihi: Kasım 2015].
- [29]. Allman, M., Paxson, V., & Blanton, E., 2009, TCP congestion control, *Network Working Group, Standards Track* (No. RFC 5681).
- [30]. Akyildiz, I. F., Lee, A., Wang, P., Luo, M., & Chou, W., 2014, A roadmap for traffic engineering in software defined networks, *Computer Networks*, <http://dx.doi.org/10.1016/j.comnet.2014.06.002>.
- [31]. Curtis, A. R., Kim, W., & Yalagandula, P., 2011, Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection, *In INFOCOM, Proceedings IEEE*, 1629-1637.
- [32]. Hopps, C. E., 2000, Analysis of an equal-cost multi-path algorithm, RFC, Tech. Rep., November 2000.
- [33]. Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., Curtis, A. R., & Banerjee, S., 2010, Devoflow: Cost-effective flow management for high performance enterprise networks. *In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, (p. 1)..
- [34]. Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., ... & Shenker, S., 2010, Onix: A Distributed Control Platform for Large-scale Production Networks, *In OSDI*, Vol. 10, 1-6.
- [35]. Hassas Yeganeh, S., & Ganjali, Y., 2012, Kandoo: a framework for efficient and scalable offloading of control applications. *In Proceedings of the first workshop on Hot topics in software defined networks*, 19-24.
- [36]. Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., & Vahdat, A., 2010, Hedera: Dynamic Flow Scheduling for Data Center Networks. *In NSDI*, Vol. 10, 19-19.
- [37]. Myers, A., *Stanford's John McCarthy, seminal figure of artificial intelligence dies at 84.*, 2010, <http://news.stanford.edu/news/2011/october/john-mccarthy-obit-102511.html>, [Ziyaret tarihi: Kasım 2015].
- [38]. Solomonoff, R. J., 1957, An inductive inference machine. *In IRE Convention Record, Section on Information Theory*, Vol. 2, 56-62.

- [39]. Solomonoff, R. J., 1997, The discovery of algorithmic probability. *Journal of Computer and System Sciences*, 55(1), 73-88.
- [40]. Mitchell, T. M., 2006, *The discipline of machine learning* (Vol. 17). Carnegie Mellon University, School of Computer Science, Machine Learning Department.
- [41]. Gentle, J. E., Härdle, W. K., & Mori, Y., 2012, How computational statistics became the backbone of modern data science. In *Handbook of Computational Statistics*, 3-16.
- [42]. Chakrabarti, S., Ester, M., Fayyad, U., Gehrke, J., Han, J., Morishita, S., ... & Wang, W., 2006,. Data mining curriculum: A proposal (Version 1.0). *Intensive Working Group of ACM SIGKDD Curriculum Committee*.
- [43]. Friedman, J. H., 1997, On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data mining and knowledge discovery*, 1(1), 55-77.
- [44]. Yu, L., & Liu, H., 2004, Efficient feature selection via analysis of relevance and redundancy, *The Journal of Machine Learning Research*, 5, 1205-1224.
- [45]. Ghahramani, Z., 2004, Unsupervised learning. In *Advanced Lectures on Machine Learning*, Springer, 72-112.
- [46]. Touretzky, D., & Laskowski, K., 2006, Neural Networks for Time Series Prediction. *Artificial Neural Networks*.
- [47]. McCaffey, C., *Test Run - Neural Network Back-Propagation for Programmers*, <https://msdn.microsoft.com/en-us/magazine/jj658979.aspx>, [Ziyaret tarihi: Kasım 2015].
- [48]. Oppenheim, A. V., & Verghese, G. C., 2010, *Signals, systems, and inference. Class notes for*, 6. pp.139-160, Chapter 8.
- [49]. Erman, J., Mahanti, A., Arlitt, M., Cohen, I., & Williamson, C., 2007, Offline/realtime traffic classification using semi-supervised learning. *Performance Evaluation*, 64(9), 1194-1213.
- [50]. Wikipedia, *Traffic Classification*, https://en.wikipedia.org/wiki/Traffic_classification, [Ziyaret tarihi: Ocak 2016].
- [51]. Stewart, R. R., Tüxen, M., & Neville-Neil, G. V., 2011, An investigation into data center congestion control with ecn. In *2011 Technical BSD Conference (BSDCan 2011)*, Ottawa, Canada.
- [52]. Alizadeh, M. et al., Data Center TCP, <http://web.mit.edu/6.033/www/papers/dctcp-talk.pdf> , [Ziyaret tarihi: Kasım 2015].

- [53]. Meyer, D., *Can Congestion in Data Center Networks Be Predicted By Of Time Of Day?*, 2014, <http://www.1-4-5.net/~dmm/papers/cp.pdf>, Ziyaret tarihi: Kasım 2015
- [54]. Hinton, G. E., & Salakhutdinov, R. R., 2006, Reducing the dimensionality of data with neural networks, *Science*, 313(5786), 504-507.
- [55]. Yu, Y., Chen, Q., & Li, X., 2014, Distributed Collaborative Monitoring in Software Defined Networks. *arXiv preprint arXiv:1403.8008*.
- [56]. McCormick, B., Kelly, F., Plante, P., Gunning, P., & Ashwood-Smith, P., 2014, Real time alpha-fairness based traffic engineering. *In Proceedings of the third workshop on Hot topics in software defined networking*, 199-200.
- [57]. J. Mo and J. Walrand, 2000, Fair end-to-end window-based congestion control, *IEEE/ACM Transactions on Networking* , 8(5):556-567.
- [58]. Varga, A., and R. Hornig, Omnet++ community site, <http://www.omnetpp.org> [Ziyaret tarihi: Kasım 2015].
- [59]. Varga, A., Omnet++ user manual., <http://www.omnetpp.org/doc/manual/usman.html>. [Ziyaret tarihi: Kasım 2015].
- [60]. Varga A., 2012, INET Framework for OMNeT++ Manual, <https://omnetpp.org/doc/inet/api-current/inet-manual-draft.pdf>, [Ziyaret tarihi: Kasım 2015].
- [61]. Klein, D., & Jarschel, M., 2013, An OpenFlow extension for the OMNeT++ INET framework. *In Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, 322-329.
- [62]. Hespanha, J. P., Bohacek, S., Obraczka, K., & Lee, J., 2001, Hybrid modeling of TCP congestion control. In *Hybrid systems: Computation and control* (pp. 291-304)..
- [63]. Peng, H., Long, F., & Ding, C., 2005, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8), 1226-1238.
- [64]. Billings, S. A., 2013, Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains. John Wiley & Sons.
- [65]. Yu, H., & Wilamowski, B. M., 2011, Levenberg-marquardt training. *Industrial Electronics Handbook*, 5, 12-1.
- [66]. Levenberg, Kenneth, 1944, A method for the solution of certain non-linear problems in least squares: 164-168.

- [67]. Marquardt, D. W. ,1963, An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2), 431-441.
- [68]. Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R.,2012, *Improving neural networks by preventing co-adaptation of feature detectors*. arXiv preprint arXiv:1207.0580.

EKLER

EK 1. Dumbbell Topolojisinden Üretilen Veri Kümesi

Tıkanıklık durumu ve 1-4 numaralı öz nitelikler aşağıda verilmiştir.

Örnek No.	TIKANIKLIK	Öznitelik-1	Öznitelik-2	Öznitelik-3	Öznitelik-4
1	1	563957000000	563679000000	0	0
2	0	157169	387941	0	0
3	1	563679000000	563957000000	0	0
4	0	79865	197093	0	0
5	0	171206	422595	0	0
6	0	237454000000	961859	0	0
7	1	609863000000	60923000000	0	0
8	0	133650	329876	0	0
9	0	106910	263862	0	0
10	0	109782	270952	0	0
11	1	60923000000	609863000000	0	0
12	0	114252	281988	0	0
13	0	89237	220231	0	0
14	0	219786	542573	0	0
15	0	18644000000	755209	0	0
16	0	302085000000	302085000000	0	0
17	0	302085000000	302085000000	0	0
18	0	302085000000	302085000000	0	0
19	0	302085000000	302085000000	0	0
20	0	183445	452856	0	0
21	0	351997	868973	0	0
22	0	351732	868318	0	0
23	0	159306000000	645297	0	0
24	0	258121000000	258121000000	0	0
25	0	258121000000	258121000000	0	0
26	0	258121000000	258121000000	0	0
27	0	258121000000	258121000000	0	0
28	0	293573	724738	0	0
29	0	219991	543071	0	0
30	0	244649000000	990994	0	0
31	0	396399000000	3964000000	0	0
32	0	3964000000	396399000000	0	0
33	0	3964000000	3964000000	0	0
34	0	3964000000	3964000000	0	0

Örnek No.	TIKANIKLIK	Öznitelik-1	Öznitelik-2	Öznitelik-3	Öznitelik-4
35	0	183616	453272	0	0
36	0	352323	869770	0	0
37	0	235102	580377	0	0
38	0	196770	485718	0	0
39	0	350948000000	142158000000	0	0
40	1	568634000000	568634000000	0	0
41	1	568634000000	568634000000	0	0
42	1	568634000000	568634000000	0	0
43	1	568634000000	568634000000	0	0
44	0	164237	405401	0	0
45	0	315127	777914	0	0
46	0	210285	519084	0	0
47	0	245993	607236	0	0
48	0	289288	714124	0	0

5-8 no'lu öznitelikler aşağıda görülmektedir.

Örnek No.	Öznitelik-5	Öznitelik-6	Öznitelik-7	Öznitelik-8
1	0	0	0	0
2	0	0	232663000000	27686600000000
3	0	0	0	0
4	0	0	255014000000	30346400000000
5	0	0	228605000000	27203700000000
6	0	0	0	0
7	0	0	0	0
8	0	0	239464000000	28495800000000
9	0	0	247195000000	29415900000000
10	0	0	246364000000	29317000000000
11	0	0	0	0
12	0	0	245072000000	29163200000000
13	0	0	252305000000	30023900000000
14	0	0	752847	89588200000000
15	0	0	0	0
16	0	0	0	0
17	0	0	0	0
18	0	0	0	0
19	0	0	0	0
20	0	0	803945	95668900000000
21	0	0	566948	67466200000000
22	0	0	257925	30692800000000
23	0	0	0	0
24	0	0	0	0
25	0	0	0	0

Örnek No.	Öznitelik-5	Öznitelik-6	Öznitelik-7	Öznitelik-8
26	0	0	0	0
27	0	0	0	0
28	0	0	309023	3677340000000
29	0	0	10831000000	12888800000000
30	0	0	0	0
31	0	0	0	0
32	0	0	0	0
33	0	0	0	0
34	0	0	0	0
35	0	0	11342000000	13496800000000
36	0	0	897199	10676600000000
37	0	0	106187000000	12636200000000
38	0	0	192378000000	22892800000000
39	0	0	0	0
40	0	0	0	0
41	0	0	0	0
42	0	0	0	0
43	0	0	0	0
44	0	0	197487000000	23500800000000
45	0	0	173788000000	20680600000000
46	0	0	190255000000	22640100000000
47	0	0	184646000000	21972700000000
48	0	0	177846000000	21163500000000

9-12 no'lu öznitelikler aşağıda görülmektedir.

Örnek No.	Öznitelik-9	Öznitelik-10	Öznitelik-11	Öznitelik-12	Öznitelik-13
1	7694	7694	1	569502000000	5217800000000
2	306972	307	1	227211	1454150000000
3	7694	7694	1	569502000000	52152200000000
4	156	156	1	115456	7389180000000
5	334	334	1	247503	1584020000000
6	1879	1879	1	139051000000	21969500000000
7	8253	8253	1	610826000000	56425200000000
8	261	261	1	193210	1236540000000
9	209	209	1	154554	9891460000000
10	214	214	1	158706	1015720000000
11	8253	8253	1	610826000000	56366700000000
12	223	223	1	165168	1057080000000
13	174	174	1	129005	8256320000000
14	429	429	1	154517	9889090000000
15	1475	1475	1	530940	8388720000000

Örnek No.	Öznitelik-9	Öznitelik-10	Öznitelik-11	Öznitelik-12	Öznitelik-13
16	5900	5900	1	212376000000	13592100000000
17	5900	5900	1	212376000000	13592100000000
18	5900	5900	1	212376000000	13592100000000
19	5900	5900	1	212376000000	13592100000000
20	358	358	1	128968	825395000000
21	687	687	1	247466	1583780000000
22	687	687	1	154514	988890000000
23	1260	1260	1	283477	4478850000000
24	5041	5041	1	113391000000	7257020000000
25	5041	5041	1	113391000000	7257020000000
26	5041	5041	1	113391000000	7257020000000
27	5041	5041	1	113391000000	7257020000000
28	573	573	1	128965	825376000000
29	430	430	1	154519	988922000000
30	1936	1936	1	696067	10997700000000
31	7742	7742	1	278427000000	17819300000000
32	7742	7742	1	278427000000	17819300000000
33	7742	7742	1	278427000000	17819300000000
34	7742	7742	1	278427000000	17819300000000
35	359	359	1	128970	825408000000
36	688	688	1	247468	1583800000000
37	459	459	1	165133	1056850000000
38	384	384	1	154529	988986000000
39	2777	2777	1	111641000000	1763900000000
40	11106	11106	1	446564000000	28580100000000
41	11106	11106	1	446564000000	28580100000000
42	11106	11106	1	446564000000	28580100000000
43	11106	11106	1	446564000000	28580100000000
44	321	321	1	128980	825472000000
45	615	615	1	247478	1583860000000
46	411	411	1	165143	1056920000000
47	480	480	1	193185	1236380000000
48	565	565	1	227186	1453990000000

14-18 arası öznitelikler aşağıda görülmektedir.

Örnek No.	Öznitelik-14	Öznitelik-15	Öznitelik-16	Öznitelik-17
1	0	0	569502000000	52152200000000
2	0	0	227174	3589270000000
3	0	0	569502000000	5217800000000

Örnek No.	Öznitelik-14	Öznitelik-15	Öznitelik-16	Öznitelik-17
4	0	0	115417	1823530000000
5	0	0	247467	3909890000000
6	0	0	139050000000	8899230000000
7	0	0	610826000000	5636670000000
8	0	0	193173	3052050000000
9	0	0	154516	2441280000000
10	0	0	158668	2506880000000
11	0	0	610826000000	5642520000000
12	0	0	165131	2608990000000
13	0	0	128967	2037610000000
14	0	0	154513	2441260000000
15	0	0	530937	339800000000
16	0	0	212376000000	1359210000000
17	0	0	212376000000	1359210000000
18	0	0	212376000000	1359210000000
19	0	0	212376000000	1359210000000
20	0	0	128964	2037590000000
21	0	0	247463	3909870000000
22	0	0	154513	2441260000000
23	0	0	283475	1814240000000
24	0	0	113391000000	7257020000000
25	0	0	113391000000	7257020000000
26	0	0	113391000000	7257020000000
27	0	0	113391000000	7257020000000
28	0	0	128964	2037590000000
29	0	0	154513	2441260000000
30	0	0	696063	4454800000000
31	0	0	278427000000	1781930000000
32	0	0	278427000000	1781930000000
33	0	0	278427000000	1781930000000
34	0	0	278427000000	1781930000000
35	0	0	128964	2037590000000
36	0	0	247463	3909870000000
37	0	0	165127	2608960000000
38	0	0	154514	2441270000000
39	0	0	111641000000	714500000000
40	0	0	446564000000	2858010000000
41	0	0	446564000000	2858010000000
42	0	0	446564000000	2858010000000
43	0	0	446564000000	2858010000000
44	0	0	128964	2037590000000
45	0	0	247464	3909870000000
46	0	0	165128	2608970000000

Örnek No.	Öznitelik-14	Öznitelik-15	Öznitelik-16	Öznitelik-17
47	0	0	193170	3052030000000
48	0	0	227172	3589260000000

19-24 arası öznitelikler aşağıda görülmektedir.

Örnek No.	Öznitelik-19	Öznitelik-20	Öznitelik-21	Öznitelik-22	Öznitelik-23	Öznitelik-24
1	521780000000 0	740169	0	0	5695020000 00	521522000000 00
2	145415000000 0	740169	0	0	227174	358927000000 0
3	521522000000 00	740169	0	0	5695020000 00	521780000000 0
4	738918000000	740169	0	0	115417	182353000000 0
5	158402000000 0	740169	0	0	247467	390989000000 0
6	219695000000 00	740169	0	0	1390500000 00	889923000000 0
7	564252000000 00	740169	0	0	6108260000 00	563667000000 00
8	123654000000 0	740169	0	0	193173	305205000000 0
9	989146000000	740169	0	0	154516	244128000000 0
10	101572000000 0	740169	0	0	158668	250688000000 0
11	563667000000 00	740169	0	0	6108260000 00	564252000000 00
12	105708000000 0	740169	0	0	165131	260899000000 0
13	825632000000	740169	0	0	128967	203761000000 0
14	988909000000	359953	0	0	154513	244126000000 0
15	838872000000 0	359953	0	0	530937	339800000000
16	135921000000 00	359953	0	0	2123760000 00	135921000000 00
17	135921000000 00	359953	0	0	2123760000 00	135921000000 00
18	135921000000 00	359953	0	0	2123760000 00	135921000000 00
19	135921000000 00	359953	0	0	2123760000 00	135921000000 00
20	825395000000	359953	0	0	128964	203759000000 0
21	158378000000 0	359953	0	0	247463	390987000000 0
22	988890000000	224919	0	0	154513	244126000000 0
23	447885000000 0	224919	0	0	283475	181424000000 0
24	725702000000 0	224919	0	0	1133910000 00	725702000000 0

Örne k No.	Öznitelik-19	Öznitelik- 20	Öznitelik- 21	Öznitelik- 22	Öznitelik-23	Öznitelik-24
25	725702000000 0	224919	0	0	1133910000 00	725702000000 0
26	725702000000 0	224919	0	0	1133910000 00	725702000000 0
27	725702000000 0	224919	0	0	1133910000 00	725702000000 0
28	825376000000	224919	0	0	128964	203759000000 0
29	988922000000	359623	0	0	154513	244126000000 0
30	109977000000 00	359623	0	0	696063	445480000000 0
31	178193000000 00	359623	0	0	2784270000 00	178193000000 00
32	178193000000 00	359623	0	0	2784270000 00	178193000000 00
33	178193000000 00	359623	0	0	2784270000 00	178193000000 00
34	178193000000 00	359623	0	0	2784270000 00	178193000000 00
35	825408000000	359623	0	0	128964	203759000000 0
36	158380000000	359623	0	0	247463	390987000000 0
37	105685000000 0	359623	0	0	165127	260896000000 0
38	988986000000	402088	0	0	154514	244127000000 0
39	176390000000 0	402088	0	0	1116410000 00	714500000000
40	285801000000 00	402088	0	0	4465640000 00	285801000000 00
41	285801000000 00	402088	0	0	4465640000 00	285801000000 00
42	285801000000 00	402088	0	0	4465640000 00	285801000000 00
43	285801000000 00	402088	0	0	4465640000 00	285801000000 00
44	825472000000	402088	0	0	128964	203759000000 0
45	158386000000 0	402088	0	0	247464	390987000000 0
46	105692000000 0	402088	0	0	165128	260897000000 0
47	123638000000 0	402088	0	0	193170	305203000000 0
48	145399000000 0	402088	0	0	227172	358926000000 0

ÖZGEÇMİŞ



Kişisel Bilgiler

Adı Soyadı	Fatma GÜMÜŞ
Uyruğu	T.C.
Doğum tarihi, Yeri	13.09.1988, Karabük
E-mail	Fa.gumus@gmail.com

Eğitim

Derece	Kurum/Anabilim Dalı/Programı	Yılı
Yüksek Lisans	İ.Ü. Fen Bilimleri Enstitüsü/ Bilgisayar Mühendisliği	2016
Yüksek Lisans	İ.Ü. Sosyal Bilimler Enstitüsü/ Uluslararası İşletmecilik (İ.Ö)	Devam etmekte
Lisans	İ.Ü. Mühendislik Fakültesi/ Bilgisayar Mühendisliği	2013
Lisans	Anadolu Üniversitesi/ İşletme (Açık Öğretim)	2013
Lise	İstanbul Açık Öğretim Lisesi	2005

Makaleler / Bildiriler

Gumus, F., Sakar, C. O., Erdem, Z., & Kursun, O. (2014, August). Online Naive Bayes classification for network intrusion detection. In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on* (pp. 670-674). IEEE.