KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DOKTORA TEZİ

İÇERİK DAĞITIM AĞLARINDA SENKRONİZASYON ZAMANININ PROFILE HIDDEN MARKOV MODEL İLE KESTİRİMİ

FİDAN KAYA GÜLAĞIZ

KOCAELİ 2018

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

DOKTORA TEZİ

İÇERİK DAĞITIM AĞLARINDA SENKRONİZASYON ZAMANININ PROFILE HIDDEN MARKOV MODEL İLE KESTİRİMİ

FİDAN KAYA GÜLAĞIZ

Dr.Öğr.Üyesi Suhap ŞAHİN
Danışman, Kocaeli Üniversitesi .....

Prof.Dr. Celal ÇEKEN
Jüri Üyesi, Sakarya Üniversitesi .....

Doç.Dr. Sevinç İLHAN OMURCA
Jüri Üyesi, Kocaeli Üniversitesi .....

Prof.Dr. Nevcihan DURU
Jüri Üyesi, Kocaeli Üniversitesi .....

Doç.Dr. Cihan KARAKUZU
Jüri Üyesi, Bilecik Şeyh Edebali Üniversitesi .....

Tezin Savunulduğu Tarih: 03.05.2018

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışması, günlük davranışı belirli bir örüntüye sahip, ana bulut sunucu temelli içerik dağıtım ağlarında gerçekleştirilecek senkronizasyon işleminin zamanının tespiti için yeni bir yöntem geliştirmek amacıyla gerçekleştirilmiştir.

Doktora eğitimim süresince desteğini esirgemeyen, tezimin her aşamasında sorunlarımı dinleyerek, görüşleri ile çalışmalarıma katkıda bulunan ve yoğun akademik yaşamında değerli zamanını her türlü problemimi çözmeye ayıran tez danışmanım saygı değer hocam Dr. Öğr. Üyesi Suhap ŞAHİN'e içtenlikle teşekkür ederim.

Tez çalışmam boyunca gösterdiği destek ve anlayış için Öğr. Gör. Dr. Onur Gök'e,

Tez çalışmama bilgi ve tavsiyeleri ile katkıda bulunan Dr. Öğr. Üyesi Alev MUTLU'ya, tez ilerleme jürim Prof. Dr. Celal Çeken'e ve Doç. Dr. Sevinç İLHAN OMURCA'ya,

Çalışmalarım boyunca desteğini ve yardımlarını esirgemeyen, aynı laboratuvarı paylaştığım çalışma arkadaşlarım Arş. Gör. Mehmet Ali ALTUNCU, Arş. Gör. Hikmetcan ÖZCAN'a ve başta Ata NİYAZOV olmak üzere Gömülü ve Algılayıcı Sistemler Araştırma Laboratuvarı çalışanlarına,

Akademik çalışmalarım sırasında, birçok aşamada beni destekleyen çalışma arkadaşlarım Arş. Gör. Derya ÜNLÜ'ye, Arş. Gör. Ekin EKİNCİ'ye, Arş. Gör. Abdurrahman GÜN'e, Arş. Gör. Furkan GÖZ'e ve Arş. Gör. Süleyman Eken'e,

Maddi ve manevi desteklerini tüm hayatı boyunca esirgemeyen annem Lütfiye KAYA'ya, babam Sebahattin KAYA'ya ve tez çalışmam boyunca gösterdiği sonsuz destek ve anlayış için eşim Kadir GÜLAĞIZ'a teşekkürü borç bilirim.

Mayıs – 2018 Fidan KAYA GÜLAĞIZ

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

α	:	İleri değişkeni
a_{ij}	:	i durumundan j durumuna geçiş olasılığı
$\hat{a}$	:	Tahmini geçiş olasılıkları
A	:	Geçiş olasılıkları kümesi
A_{ij}	:	q_i durumundan q_j durumuna yapılan geçişlerin sayısı
$b_j(m)$	:	Saklı Markov Modelde S_j durumunda m gözleminin görülme olasılığı
$\hat{b}$	:	Tahmini gözlem olasılıkları
B	:	Gözlem olasılıkları kümesi
β	:	Geri değişkeni
c	:	Bulanık C-Ortalamalar yöntemi için küme sayısı
D_s	:	Profile Hidden Markov modelde s. silme durumu
$e_i(x)$	:	Profile Hidden Markov Modelde qi durumunda x çıktısının oluşma olasılığı
$E_i(x)$	:	Profile Hidden Markov Modelde q_i durumunda x çıktısının oluşma sayısı
ε	:	Hata değeri
$\xi_t(i,j)$	:	O ve λ verildiğinde t anındaki S_i durumundan t+1 anındaki S_j durumuna geçme ihtimali
δ	:	En yüksek olasılıklı yol
I_s	:	Profile Hidden Markov modelde s. ekleme durumu
J	:	Amaç fonksiyonu
k	:	K-Ortalamalar yöntemi için küme sayısı
K	:	Gözlem dizisi sayısı
λ	:	Saklı markov modelin parametre kümesi
λ^*	:	Saklı markov modelin tahmini parametre kümesi
m	:	Bulanıklık indeksi
M	:	Gözlem alfabesindeki simge sayısı
M_s	:	Profile Hidden Markov modelde s. eşleme durumu
μ	:	Aitlik değeri
n	:	Veri kümesindeki eleman sayısı
N	:	Durum sayısı
P	:	Olasılık
π	:	Başlangıç olasılıkları kümesi
$\hat{\pi}$	:	Tahmini başlangıç olasılıkları
ψ	:	Mevcut durumdan önceki yüksek olasılıklı bir önceki durumu saklayan değişken
q_t	:	t anındaki durum
r	:	İterasyon sayısı
$\gamma_t(i)$	:	O ve λ verildiğinde t anında S_i durumunda olma ihtimali
s	:	Profile Hidden Markov Modelde eşleme durumlarının sayısı.
S	:	Durum kümesi
t	:	Zaman
T	:	Probleme özgü gözlem dizisi uzunluğu

V	:	Simge kümesi
v_m	:	Simge kümesindeki m. simge
O	:	Gözlem dizisi
O_t	:	Gözlem dizisindeki t. gözlem
Q	:	Durum dizisi
U	:	Üyelik matrisi
X	:	Veri kümesi
w	:	Şekil parametresi
x	:	Profile Hidden Markov Model'de oluşan çıktılar
y	:	Ölçek parametresi
z_j	:	j. kümenin merkezi

Kısaltmalar

ACID	:	Atomicity, Consistency, Isolation, Durability (Bölünmezlik, Tutarlılık, İzolasyon, Süreklilik)
API	:	Application Programming Interface (Uygulama Programlama Arayüzü)
BASE	:	Basically Available, Soft State, Eventual Consistency (Temel Olarak Erişilebilir, Esnek Durum, Nihai Tutarlılık)
BSON	:	Binary JavaScript Object Notation (İkilik Javascript Nesne Notasyonu)
CAP	:	Consistency, Availability, Partition Tolerance (Tutarlılık, Erişilebilirlik, Bölümleme Toleransı)
CCDN	:	Cloud Based Content Delivery Network (Bulut Temelli İçerik Dağıtım Ağı)
CDN	:	Content Delivery Network (İçerik Dağıtım Ağı)
CSMA	:	Carrier Sense Multiple Access (Çoklu Erişimde Hat Kontrolü)
EM	:	Expectation Maximization (Beklenti Maksimizasyonu)
FCM	:	Fuzzy C-Means (Bulanık C-Ortalamalar)
GCM	:	Google Cloud Messaging (Google Bulut Mesajlaşma)
HTTP	:	Hypertext Transfer Protocol (Hiper-Metin Transfer Protokolü)
IaaS	:	Infrastructure as a Service (Servis Olarak Altyapı)
IBM	:	International Business Machines
IEEE	:	Institute of Electrical and Electronics Engineers (Elektrik ve Elektronik Mühendisleri Enstitüsü)
IP	:	Internet Protocol (İnternet Protokolü)
JSON	:	JavaScript Object Notation (Javascript Nesne Notasyonu)
k-NN	:	k-Nearest Neighbor (k-En Yakın Komşu
LAN	:	Local Area Network (Yerel Alan Ağı)
MSE	:	Mean Squared Error (Ortalama Karesel Hata)
MVCC	:	Multi-Version Concurrency Control (Çok Sürümlü Eşzamanlılık Kontrolü)
NoSQL	:	Not Only Sql
OLT	:	Optical Line Termination (Optik Hat Sonlandırıcı)
OMNET	:	Objective Modular Network Testbed (Objektif Modüler Ağ Simülatörü)
ONU	:	Optical Network Unit (Optik Ağ Ünitesi)

OPNET	:	Optimized Network Engineering Tools (Optimize Edilmiş Ağ Mühendisliği Aracı)
PaaS	:	Platform as a Service (Servis Olarak Platform)
PDF	:	Probability Density Function (Olasılık Yoğunluk Fonksiyonu)
PHMM	:	Profile Hidden Markov Model
POP	:	Points of Presence (Uzak Erişim Noktaları)
QoS	:	Quality of Service (Servis Kalitesi)
RAM	:	Random Access Memory (Rastgele Erişimli Bellek)
RTT	:	Round Trip Time (Gidiş Dönüş Süresi)
SaaS	:	Software as a Service (Servis Olarak Yazılım)
Sack	:	Selective Acknowledgement (Seçici Alındı)
SMM	:	Saklı Markov Modeli
SQL	:	Structured Query Language (Yapılandırılmış Sorgulama Dili)
SSE	:	Sum Of Squared Error (Karesel Hatalar Toplamı)
TCP	:	Transmission Control Protocol (İletim Kontrol Protokolü)
WAN	:	Wide Area Network (Geniş Alan Ağı)
WBAN	:	Wireless Body Area Network (Kablosuz Vücut Alan Ağı)
XML	:	Extensible Markup Language (Genişletilebilir İşaretleme Dili)

İÇERİK DAĞITIM AĞLARINDA SENKRONİZASYON ZAMANININ PROFILE HIDDEN MARKOV MODEL İLE KESTİRİMİ

ÖZET

İçerik dağıtım ağları yerel ve geniş alan ağlarında kaynakların verimli olarak paylaşımını sağlarlar. Böyle mimarilerde farklı sunucularda saklanan verilerin tutarlılığının sağlanması önemli bir konudur. Bu durumun üstesinden gelebilmek için senkronizasyon mekanizmalarına ihtiyaç duyulur. Verilerin depolanması amacıyla kullanılan veri tabanı yönetim sistemlerine bağlı olarak kullanılabilecek senkronizasyon yöntemleri çeşitlilik göstermektedir.

Bu tez çalışmasında verileri CouchDB'de saklayan ana bulut sunucu temelli içerik dağıtım ağları için Bulanık C-Ortalamalar ve Profile Hidden Markov Model yöntemleri kullanılarak yeni bir poll temelli senkronizasyon yöntemi tanımlanmıştır. Davranışı belirli bir örüntüye sahip olan ağlar üzerinden elde edilen günlük veri seti, Bulanık C-Ortalamalar kümeleme algoritması aracılığıyla boş, normal ve yoğun olarak isimlendirilen üç kümeye ayrılmıştır. Elde edilen kümeler üzerinden modellenecek mimariye ait genel davranış örüntüsü Profile Hidden Markov Model yöntemi kullanılarak tanımlanmıştır.

Önerilen yöntemin verimliliği ve doğruluğu OPNET üzerinde farklı mimariler kullanılarak gösterilmiştir. Elde edilen sonuçlar hem CouchDB senkronizasyon yöntemi ile hem de literatürde mevcut farklı poll yöntemleri ile karşılaştırılmıştır. Elde edilen sonuçlara göre özellikle ağın yoğun olduğu zaman dilimleri için önerilen yöntem ile yerel ağlarda CouchDB Poll yönteminden iki kat daha fazla kullanıcıya hizmet verilebilmektedir. Aynı zamanda kullanıcıların verilere erişim süresi yarıya düşürülürken paketlerin toplam yeniden iletim sayısı da azaltılmıştır.

Anahtar Kelimeler: İçerik Dağıtım Ağları, Örüntü Analizi, Profile Hidden Markov Model, Veri Senkronizasyonu.

ESTIMATION OF SYNCHRONIZATION TIME IN CONTENT DELIVERY NETWORKS WITH PROFILE HIDDEN MARKOV MODEL

ABSTRACT

Content delivery networks enable efficient sharing of resources across local and wide area networks. In such architectures, ensuring the consistency of data stored in different servers is an important issue. To get ahead of this problem synchronization mechanisms are needed. Synchronization methods that can be used depending on database management systems used to store data varies.

In this thesis, a new poll based synchronization mechanism has been defined for the main cloud server based content delivery networks that store data in CouchDB using Fuzzy C Means and Profile Hidden Markov Model methods. Through the Fuzzy C Means clustering algorithm, the daily data set obtained from networks with a certain pattern of behavior is divided into three clusters as idle, normal and busy. The general behavior pattern of the architecture to be modeled through the obtained clusters is defined using the Profile Hidden Markov Model method.

The efficiency and correctness of the proposed method is demonstrated using different architectures on OPNET. The results were compared with both the CouchDB synchronization method and the different poll methods available in the literature. When compared to CouchDB's synchronization mechanism, number of simultaneous users that the proposed method can respond is twice of CouchDB. At the same time, the total number of retransmissions of packets has been reduced while the data access time of users reduced by half.

Key Words: Content Delivery Networks, Pattern Analysis, Profile Hidden Markov Model, Data Synchronization.

GİRİŞ

Dağıtık bir sistem, ağ üzerindeki birden çok işlemi koordine etmek için bir dizi protokolü işleten uygulamadır; böylece, tüm bileşenler tek bir görevi ya da bir görevin küçük parçalarını gerçekleştirmek için birlikte çalışırlar (Prince ve Sloman, 1981). Böyle bir sistem uzaktaki kullanıcıların kaynaklara kolay ve ölçeklenebilir bir şekilde erişebilmesini sağlar.

İnternet teknolojilerinin gelişmesiyle, hizmetlere olan talep ve veri boyutu her geçen gün katlanarak artmaktadır. Bunun sonucu olarak ağ trafiğindeki kısıtlamalar veri iletimi sırasında verimsiz iletişim ve yüksek gecikmeler gibi sorunlara sebep olmaktadır. Bu problemlerin üstesinden gelebilmek ya da problemlerin etkisini azaltabilmek için içerik dağıtım ağı (CDN) mimarileri son yıllarda yaygın olarak kullanılmaktadır (Lin ve diğ., 2011; Wang ve diğ., 2011). Bu tür mimarilerde, kullanıcı bir yerel sunucunun kapsama alanı içinde değilse, doğrudan ana sunucu ile iletişim kurarken, yerel sunucunun kapsama alanında bulunuyorsa, yerel sunucu üzerinden ana sunucu ile iletişim kurar. Kullanıcılar bulundukları konuma göre yerel sunucu veya ana sunucu aracılığıyla depolanan verilere ulaşır. Böyle bir mimari de aynı veri üç farklı konumda bulunacaktır. Bunlar; son kullanıcı cihazları, yerel sunucular ve ana sunuculardır (Eken ve diğ., 2013; Eken ve diğ.(2014a, 2014b)). Veriler aynı anda farklı konumlarda bulunacağından verilerin tutarlılığının sağlanması gerekmektedir. Bu nedenle dağıtık dosya senkronizasyonu (eşleme) mimarilerine ihtiyaç vardır.

CDN'de kullanılabilecek üç farklı dosya senkronizasyonu mimarisi bulunmaktadır. Bunlar; master-slave, master-master ve multi-master olarak sıralanabilir. Master-slave mimarisinde bir ana sunucu tüm yazma işlemlerini üstlenir ve güncellemeleri diğer sunuculara gönderir. Master-master mimarisinde iki ana sunucu vardır. Herhangi bir sunucu da ekleme güncelleme veya silme gibi işlemler varsa bu işlem diğer veri tabanına da yansıtılır. Üçüncü mimaride multi-master mimarisidir ve

mimaride ikiden fazla master sunucu bulunur. Bu mimari master-master mimarisinin özel bir halidir (Bhatnagar, 1997).

Bazı ağ mimarilerinde hem ana sunucu hem de yerel sunucular senkronizasyon işlemini başlatabileceğinden, böyle mimarilerde master-master ya da multi-master senkronizasyon alt yapısına ihtiyaç duyulmaktadır. Bu tür ağ mimarileri, altyapı ve sunucular arasındaki ağ trafiğinden dolayı çeşitli kısıtlamalara sahiptir. Bu kısıtlamalar bant genişliği, paket iletim hızı, aynı anda yanıtlanabilecek kullanıcı sayısı olarak listelenebilir. Bunlar etkili veri aktarımı ve veri trafiği ile ilgili sorunlara neden olur. Sonuç olarak, internet bağlantı hızının azalması ve veri boyutunun artması sistem verimliliğini olumsuz etkileyecektir. Bu durumda, işlem masrafı ve veri erişim süreleri de artacaktır. Verilerin veri tabanlarında saklandığı düşünüldüğünde uygun veri tabanın belirlenmesi, belirtilen problemlerin çözümü açısından önemlidir. Verinin daha etkili iletilebilmesi ve senkronizasyonu için son zamanlarda esnek mimarileri nedeniyle Not Only Sql (NoSQL) veri tabanları yaygın olarak tercih edilmektedir. (Eken ve diğ., 2014c; Li ve diğ. (2013a, 2013b)).

Artan ve yapısal olmayan verileri işlemek, ilişkisel veri tabanı yönetim sistemleriyle zordur (Chickerur ve diğ., 2015; Chopade ve Dhavase, 2017; Li ve diğ., 2013b). Dolayısıyla, NoSQL veri tabanları kullanmak, ilişkisel veri tabanlarından daha iyi bir performans getirecektir. Verileri saklamak için geliştirilmiş pek çok NoSQL veri tabanı yapısı bulunmaktadır. Bunlardan en bilinenleri MongoDB, Cassandra, Redis, HBase, CouchDB ve Neo4j olarak sıralanabilir. Multi-master mimarisinde senkronizasyonu destekleyen birkaç veri tabanı bulunmaktadır (Anderson ve diğ., 2010; Kavak ve diğ., 2014; Tesoriero, 2013). Bunlardan bir tanesi de CouchDB'dir. CouchDB bu işlemi etkili olarak gerçekleştirmektedir (Gupta ve Rani, 2016). Bu nedenle tez çalışmasında CouchDB veri tabanı kullanılmıştır. CouchDB veri tabanında üç farklı senkronizasyon tetikleme yöntemi bulunmaktadır (Anderson ve diğ., 2014). Bunlar;

- continuous yöntemi: Bu yöntemde güncellemeler gerçek zamanlı olarak, sunucular arasında sürekli kurulan bağlantılar üzerinden sunuculara gönderilir. İşlem push mantığı ile gerçekleşir. Değişikler açık bağlantı üzerinden gerçekleştiği gibi gönderilir.

- polling yöntemi: Kontrol sıklığına göre her yerel sunucu, veride değişiklik olup olmadığını kontrol etmek için ana sunucuya poll yöntemini kullanarak kontrol paketi gönderir.
- long polling yöntemi: Poll yönteminin farklı bir türüdür. Sunucu cevabı göndermeden önce başka bir değişiklik olacak mı diye belirli bir süre bekler. Böylece hiçbir şey değişmeden yapılacak gereksiz poll işlemlerinden kaçınmayı sağlar.

Polling yöntemi, long polling yöntemi ve continuous yöntemine göre basit olmasına rağmen, veriler gerçek zamana yakın bir periyot ile senkronize edildiğinde sunucularda aşırı yoğunluk, gönderilen kontrol paketleri nedeniyle gereksiz bant genişliği kullanımı ve eş zamanlı cevap verilebilecek kullanıcı sayısında azalma gibi çeşitli problemler ortaya çıkmaktadır. Burada bahsedilen poll işleminde istekler istemciden sunucuya gönderilir ve cevap olarak sunucudan bilgi alınır. Poll yöntemi geleneksel bir yöntemdir ve genelde bir zaman dilimi içerisinde planlanmış olan işlerin yürütülmesi için kullanılmaktadır. Bu uygulamalara örnek olarak verideki gecikmenin göz ardı edilebildiği uygulamalar, büyük boyutlu veri transferleri ya da çapraz veri aktarımları gerektiren uygulamalar gösterilebilir. Push yönteminde ise sunucu bilginin değiştiğine dair istemciyi bilgilendirir ve bazen de bilgilendirme mesajının bir parçası olarak güncellenen bilginin de aktarımı sağlanır. Push yöntemi daha çok gerçek zamanlı uygulamalar için kullanılmaktadır. Cep telefonu uygulamalarında yer alan güncellemelerin yüklenmesi ya da verinin tek bir sunucuda işlendiği uygulamalar push yönteminin kullanıldığı uygulamalara örnek olarak gösterilebilir.

Williams ve diğ. (2013) poll ve push işlemlerinin her birinin ağa getirdiği yükü veriye erişme maliyeti ve sunucu zamanı açısından test etmişlerdir ve sürekli yapılan poll işlemi sunucu aktivitesine sebep olduğunu bu yüzden ağa bir maliyet getirdiğini tespit etmişlerdir.

Kenyon (1960) tarafından gerçekleştirilen çalışmada poll işleminde elde edilen verinin kullanışlılığı ile oluşturulan trafik arasında bir ilişki bulunduğu tespit edilmiştir ve oluşan trafiği olumsuz yönde etkileyen faktörler;
- poll işlemi uygulanan cihazların sayısı,
- her poll işlemi için ihtiyaç duyulan objelerin sayısı,

- poll işleminin periyodu ve
- mevcut bant genişliği ve tıkanıklık olarak tespit edilmiştir.

Ikeda ve Kitayama (2009) pasif optik erişimli ağlardaki İletim Kontrol Protokolü (TCP) verimliliğini arttırmak için adaptif bir poll algoritması ile işlem yapan dinamik bant genişliği tahsisi algoritması önermişlerdir. Önerilen yöntemde poll periyodu ihtiyaç duyulan bant genişliği ve TCP gidiş dönüş süresi (RTT) parametreleri göz önünde bulundurularak elde edilmiştir. Yapılan benzetimler sonucunda önerilen adaptif bir poll yönteminin, TCP paketlerinin iletmek için bant genişliğini etkin bir şekilde kullandığını göstermişlerdir.

Jacob ve Jacob (2014) kablosuz vücut alan ağları (WBAN) için planlanmış bir polling mekanizması geliştirmişlerdir. WBAN'lar çoklu erişimde hat kontrolü (CSMA) olarak adlandırılan protokol ile ortama erişim sağlamaktadır. IEEE 802.15.6 standardında polling yönteminin uygulanması kullanıcıya bırakılmıştır. Çalışmada da kesikli zamanda erişim sağlayan bir polling yöntemi önerilmiştir. Performans değerlendirilmesi yapılırken servis kalitesi (Qos) parametreleri de göz önünde bulundurulmuştur. İletim yapacak düğümlerin bekleme süreleri Karn's algoritmasının mantığına göre tahmin edilmeye çalışılmıştır. Karn's algoritmasında g olarak adlandırılan ağırlık parametresinin doğru belirlenmesi durumunda daha gerçekçi bekleme sürelerinin elde edilebileceği gösterilmiştir.

Lv ve diğ. (2015) optik erişimli ağlarda veri iletimi için kullanılan poll işleminin planlanması amacıyla yük uyarlamalı enerji verimliliğini arttıracak bir yöntem önermişlerdir. Yapılan çalışma ile düğümlere paket iletimi için uygulanan poll işlemlerinin sırasının ağdaki yoğunluğa göre belirlenmesini sağlamışlardır. Böylece bazı düğümler daha uzun süreler pasif durumda kalmıştır ve ağdaki enerji daha verimli olarak kullanılmıştır. Geleneksel şema ile yazarlar tarafından geliştirilen yöntem karşılaştırıldığında önerilen yöntemin enerji verimliliğini arttırdığı gözlemlenmiştir.

Tandon ve Motani (2017) tarafından ise merkezi bir sunucu etrafında toplanmış ve poll yöntemi ile iletim yapan çok kullanıcılı mimarilerde ortalama paket gecikmesini azaltmaya yönelik bir yöntem önerilmiştir. Bu amaçla ileriye yönelik hata düzeltme şemaları olarak bilinen şemalardan yararlanmışlardır. Önerilen poll temelli çoklu

erişim şeması ile fiziksel katmanın, bir uygulamanın özel gereksinimlerine göre uyarlanabileceği ve böylece ağ performansını geliştirilebileceği gösterilmiştir.

Bulut bilişim mimarisinde de poll ya da push tekniklerinden hangisinin kullanıldığı önemlidir. Fang ve diğ. (2004) poll yaklaşımının zamanlanmasını sağlayacak bir mimari önermişlerdir. Çalışmada ACR olarak adlandırılan bir poll zamanını planlama algoritmasından bahsedilmektedir. Algoritma da sunucudan istekte bulunulan her sayfa için bir kuyruk yapısı ve her sayfa için kaçırılan erişim isteklerinin sayısını tutan bir değişken tanımlanmıştır. Kuyrukta o sayfa için bekleyen istekler yer almaktadır. Böylece her sayfa için bir kritiklik ya da önemlilik durumu elde edilmeye çalışılmıştır.

Saxena ve diğ. (2004) mobil cihazlardaki veriye erişim süresini kısaltmak için hibrit bir yöntem önermişlerdir. Yöntemde bir sonraki push zamanı ve poll zamanını ayrı ayrı tahmin etmeye çalışan olasılık temelli bir yaklaşım sunmuşlardır. Geliştirilen yöntem ile yüksek sistem yükü altında veriye erişim süresi kısaltılmıştır. Böylece sadece push ile çalışan mekanizmalar daha iyi performans elde edilmiştir. Yapılan çalışma ile veriye erişim için zamanlanmış mekanizmaların faydalı olduğunu göstermişlerdir.

Li ve diğ. (2013a) tarafından gerçekleştirilen farklı bir çalışmada poll yönteminin zayıflığını gidermek için, bilgi yönetimi sistemlerine entegre olan Android Google bulut mesajlaşma (GCM) hizmetine dayanan bulut temelli bir uygulama geliştirmişlerdir. Önerilen yöntem ile verilerin senkronizasyonundaki etkinliğin arttırdığını, çevrimiçi trafiğin azaltıldığını ve Android terminallerinin güç tasarrufu sağladığını göstermişlerdir.

Carvalho ve diğ. (2014) tarafından yapılan çalışmada veri senkronizasyonu için poll ve push yöntemlerinin kullanımının enerji tüketimi açısından değerlendirilmesi yapılmıştır. Çalışma sonunda uygulamanın 40 dakika içerisinde birden fazla istek gönderme zorunluluğu yoksa poll tekniğinin kullanılmasının uygun olduğu aksi durumda mobil cihazlar için push tekniğinin kullanımının daha az enerji tüketimine sebep olduğu gösterilmiştir.

Bu tez çalışmasında, içerik dağıtım ağlarında sunucular arasında gerçekleştirilecek olan senkronizasyon işlemini sistemin yükünü arttırmadan, eşleme için en uygun zamanı belirleyerek gerçekleştirecek bir poll yaklaşımı geliştirilmesi planlamıştır. Yöntemde, ağ trafiğinin genel davranışı modellenerek, trafiğin en uygun olduğu zamanlarda senkronizasyon paketleri gönderilmektedir. Yöntem dağıtık yerel sunucu-ana sunucu mimarilerine uyumlu ve verinin olabildiğince sık aralıklarla eşlenmesini sağlayan esnek bir mimariye sahiptir. Ağın yoğunluk durumuna göre poll periyodu dinamik olarak değiştirilmektedir. Böylece hem iç ağda hem de yerel sunucular ile ana sunucular arasındaki paket iletim gecikmeleri ile paketlerin yeniden iletim sayısının azaltılması ve iç ağlarda eş zamanlı olarak cevap verilebilecek kullanıcı sayılarının arttırılması hedeflenmiştir.

Tez çalışmasının ilk bölümünde dağıtık sistemler ve dağıtık sistemlerde senkronizasyon işleminin nasıl gerçekleştirildiği konusunda bilgi verilecektir. İkinci bölümde bir ağı modellemek için kullanılabilecek parametrelerden bahsedilecektir. Üçüncü bölümde dağıtık mimarilerde senkronizasyon için kullanılabilecek veri tabanı türlerinden ve tez çalışması kapsamında tercih edilen veri tabanı çeşidinden bahsedilecektir. Dördüncü bölümde önerilen poll yönteminde kullanılan kümeleme algoritmaları hakkında bilgi verilecektir. Beşinci bölümde bir ağın genel davranışını modellemek için kullanılan Profile Hidden Markov Model (PHMM) yöntemine değinilecektir. Uygulama bölümü olan altıncı bölümde kampüs ağları için PHMM yöntemi ve Bulanık C-Ortalamalar (FCM) yöntemini kullanan yeni bir poll mekanizması önerilecektir. Önerilen yöntemin testleri Optimize Edilmiş Ağ Mühendisliği Aracı (OPNET) (URL-2, 2018) yazılımı kullanılarak gerçekleştirilmiştir. Testler farklı sayıda yerel sunucuya sahip kampüs mimarileri için farklı senaryolar altında yapılmıştır. Aynı zamanda farklı poll periyotları altında da yöntemin davranışı analiz edilmiştir. Önerilen yöntem hem CouchDB içerisinde yer alan klasik poll yöntemi ile hem de literatürdeki farklı uygulamalarda kullanılan poll yöntemleri ile karşılaştırılmıştır. Yöntemin kampüs ağları için bir günlük davranışı da modellenmiştir ve CouchDB mekanizması ile karşılaştırılmıştır. Yukarıda bahsedilen problemler göz önüne alındığında, birçok açıdan başarıma sahip bir poll yöntemi ortaya çıkarılmıştır. Sonuçlar ve öneriler bölümünde, yapılan çalışmalardan elde edilen sonuçlar genel hatlarıyla literatürdeki benzer çalışmalarla

karşılaştırılarak, çalışmanın bilime ve günümüz teknolojisine sağlayabileceği katkıları tartışılacaktır. Ayrıca ileriye dönük yapılabilecek çalışmalar için önerilerde bulunulacaktır.

Yapılan ulusal ve uluslararası literatür araştırmalarında farklı mimariler için önerilen poll yöntemleri incelendiğinde ağın genel davranışını PHMM yöntemini ve kümeleme algoritmalarını kullanarak modelleyen bir çalışma görülmemiştir. Bu nedenle, bu tez çalışmasının oldukça özgün bir değere sahip olduğu ve önerilen poll yönteminin hem senkronizasyon için hem de genel davranışı belirli bir örüntüye sahip olan ağların günlük davranış örüntüsünü belirlemek için literatüre önemli katkılar sağlayacağı düşünülmektedir.

1. DAĞITIK SİSTEMLER VE SENKRONİZASYON

Dağıtık sistemler en basit haliyle kullanıcılarına uyumlu tek bir sistem olarak gözüken bağımsız bilgisayarlar topluluğudur (Tanenbaum ve Steen, 2007). Böyle bir modelde yer alan bilgisayarlar eylemlerinin kontrolünü ve birbirleri arasındaki iletişimi gönderilen mesajlar yoluyla sağlarlar (George ve diğ., 2011). Dağıtık sistemlerin en yaygın olarak kullanıldığı alanların belirgin örnekleri internette yapılan aramalar, internet üzerinden çok oyunculu olarak oynanan çevrimiçi oyunlar ve finansal ticaret piyasaları olarak verilebilir. Dağıtık sistemlerin temelde gerçekleştirmesi gereken dört ana özelliği bulunmaktadır. Bunlar;

- Kolay erişilebilirlik
- Kaynakların bir ağ üzerinde dağıtık olduğunu belli etmemek
- Açıklık ve
- Ölçeklenebilirlik olarak sıralanabilir.

Burada kolay erişilebilirlik olarak bahsedilen özellik dağıtık sistemlerin ana hedefidir ve kullanıcıların uzak kaynaklara erişimini, bu kaynakların kontrollü ve etkili bir şekilde paylaşımını sağlar. Burada kaynak olarak bahsedilen terim yazılım ya da donanım olarak her türlü kaynağı kapsamaktadır. İkinci özellik olan şeffaflık ise farklı açılardan kullanıcıların karşısına çıkabilmektedir. Kullanıcıların kaynağın nerede olduğunu bilmemesi, kaynağın farklı bir konuma taşındığından haberdar olmamaları ya da kaynağın aynı anda kaç kullanıcı ile paylaşıldığını bilmemeleri şeffaflık özelliğinin örneği olarak verilebilir. Açıklık olarak bahsedilen özellik sisteme kolaylıkla alt bileşenleri eklenebilmesi ve mevcut bileşenlerin kolay bir şekilde ayarlanabilmesini tanımlamaktadır. Ölçeklenebilirlik özelliği ise üç farklı yönden tanımlanmaktadır. Bunlardan birincisi sisteme daha fazla kullanıcı ya da kaynağın kolayca eklenebilmesi, ikincisi coğrafi olarak kullanıcı ve kaynakların uzak noktalarda bulunabilmesi ve son olarak farklı yönetimsel organizasyonlara ayrılsa bile kolay bir şekilde yönetilebilmesi olarak açıklanabilir (Tanenbaum ve Steen, 2007).

Yukarıda bahsedilen özelliklerinin belirli ölçülerde sağlanabilmesi için farklı kaynaklarda yer alan verilerin tutarlı olması gerekmektedir. Bu nedenle verilerin senkronizasyonu gereklidir. Verilerin senkronizasyonu için dağıtık sistemlerde yer alan bilgisayarların (düğümlerin) topolojilerinin doğru olarak belirlenmesi gereklidir. Alt bölümlerde ilk olarak senkronizasyonu yapılacak sunucular arasında bağlantı için kullanılabilecek düğüm topolojilerinden ve senkronizasyon işleminden detaylı olarak bahsedilecektir.

1.1. Dağıtık Sistemlerde Ağ Bağlantı Topolojileri

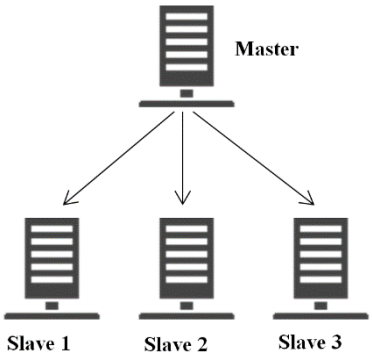
Dağıtık sistemlerde senkronizasyonu yapılacak düğümlerin aralarındaki bağlantılar temelde üç farklı topolojide olabilir (Bhatnagar, 1997). Bunlar;

- master-slave,
- master-master
- multi-master topolojileri olarak sıralanabilir.

Kullanılan senkronizasyon altyapısına göre bu üç temel topolojiden farklı topolojiler de ortaya çıkabilmektedir. Bağlantı topolojilerinde yer alan düğümler işlem yetkilerine göre master ya da slave olarak isimlendirilmektedir. Alt bölümlerde bağlantı topolojilerin açıklamaları verilmiştir.

1.1.1. Master-slave topolojisi

En basit ağ bağlantı topolojisidir. Bir master düğüm ve bir ya da daha çok slave düğümden oluşmaktadır. Bir master düğüm ve üç slave düğümden oluşan örnek bir topoloji Şekil 1.1'de gösterilmiştir. Burada master olarak adlandırılan düğüm güncellemeleri kaydeder ve daha sonra bu güncellemeleri slave düğümlerine iletir. Slave düğümler güncellemeleri tamamladıklarında master düğümü işlemin tamamlandığına dair bilgilendirirler. Burada güncellemelerin iletimi tek yönlü olarak (ana düğümden köle düğümlere doğru) sağlanmaktadır. Tüm yazma işlemleri master düğüm tarafından gerçekleştirilirken, okuma istekleri slave düğümler tarafından karşılanır.

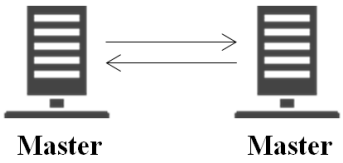


Şekil 1.1. Master-slave senkronizasyon topolojisi

Böyle bir mimari az sayıda yazma ve çok sayıda okuma işlemi yapıldığı zamanlarda etkilidir. Birden çok slave düğüm kullanımı ile iş yükünün slave bilgisayarlar arasında pek çok sunucuya dağıtılması sağlanır. Böylece yatay olarak ölçeklendirme sağlanmış olur.

1.1.2. Master-master topolojisi

Master-master topolojisinde toplamda iki adet sunucu bulunmaktadır. Her iki sunucuda master düğümdür. Şekil 1.2'de örnek bir master-master topolojisi gösterilmiştir. Bu topoloji daha çok coğrafi olarak farklı konumlarda bulunan, aynı veri tabanı üzerinde çalışan ve her iki sunucuda da yazma işlemine ihtiyaç duyulan mimarilerde tercih edilmektedir. Düğümlerin her ikisi de hem yazma hem de okuma işlemlerini gerçekleştirebilmektedir.

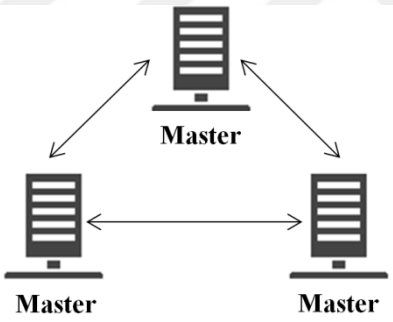


Şekil 1.2. Master-master senkronizasyon topolojisi

Böyle mimarilerin tercih edildiği geniş alan ağlarında (WAN) kısa bağlantı kopukluklarının tolere edilebilir olması gerekmektedir. Bağlantı kopukluğu durumunda her iki tarafta da güncel veriye erişilemeyebilir, bağlantı normale döndüğünde her iki düğümde birbiriyle eş duruma gelecektir.

1.1.2. Multi-master topolojisi

Multi-master topolojisi, master-master topolojisinin n adet master sunucudan oluşan özel bir halidir. Burada ikiden fazla master sunucu bulunmaktadır. Bu sunucuların her biri paylaşılan veri tabanı üzerinde okuma ve yazma işlemlerine yetkilidir. Coğrafi olarak farklı konumlarda farklı sunucular yer aldığından ve bu sunuculardaki veriler eş olduğundan veriye erişim yüksek ağ gecikmeleri olmadan gerçekleştirilebilir. Master-master ve multi-master mimarileri veriye erişimde ağ gecikmelerini azaltırken beraberinde aynı veri tabanına eş zamanlı yazma sonucu ortaya çıkan çakışma sorununu ya da maliyet gibi sorunları beraberinde getirebilmektedir. Çakışma sorunu kullanılan çeşitli çakışma önleme ve çakışma çözümleme algoritmaları aracılığıyla çözülmektedir (Ajila ve Al-Asaad, 2011; Son ve Kouloumbis, 1992). Şekil 1.3'te üç master sunucudan oluşan bir multi-master sunucu mimarisi en basit haliyle gösterilmiştir.



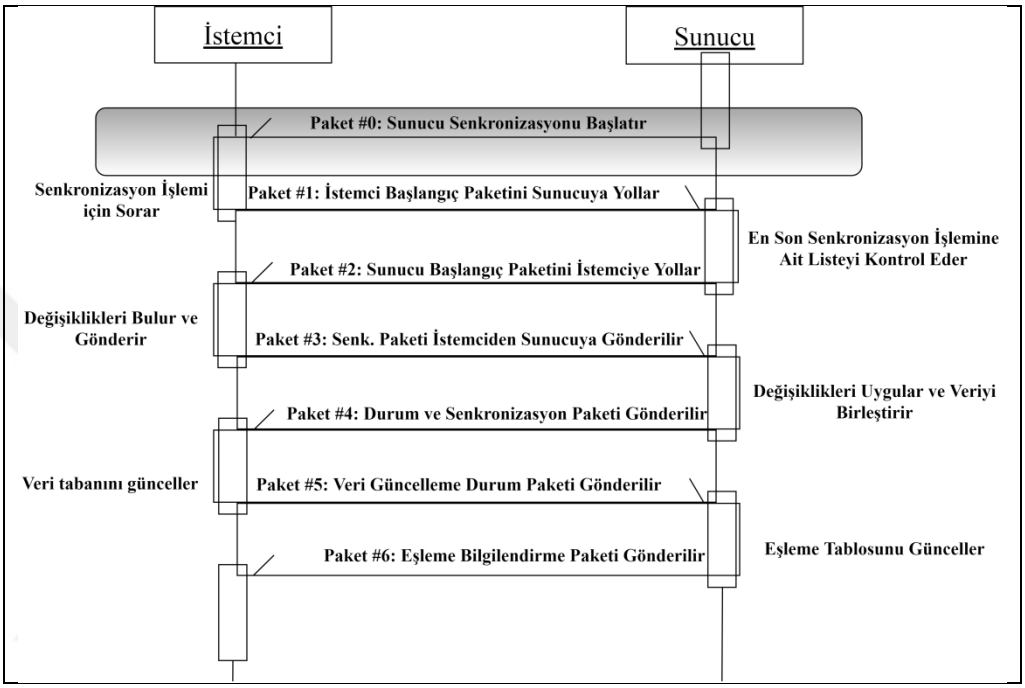
Şekil 1.3. Multi-master senkronizasyon topolojisi

Birinci bölümde bahsedilen tüm mimarilerde verilerin tutarlı olabilmesi için senkronizasyon işlemine ihtiyaç duyulmaktadır. Bir sonraki bölümde senkronizasyon işlemi ve gerekliliği detaylı olarak açıklanacaktır.

1.2. Dağıtık Sistemlerde Senkronizasyon

Dağıtık sistemlerde, aynı veri fiziksel olarak farklı konum veya makinalarda yönetilmektedir. Bu veriler her bir makinada birbirinden bağımsız olarak değişikliğe uğrayabilirler, bu da farklı versiyonlarda veriler oluşmasına neden olur. Farklı versiyonlardaki veriler tutarlı bir biçimde birleştirilerek yönetilmelidirler. Bu nedenle veri senkronizasyonuna ihtiyaç duyulur. Verilerin tutarlı hale getirilebilmesi için gerçekleştirilen işlemler senkronizasyon olarak tanımlanabilir. Senkronizasyon

işlemi yukarıda bahsedilen topolojiler doğrultusunda gerçekleştirilir. Sistemdeki bileşenlerden biri veya bir kaçı bu senkronizasyon işlemini başlatır ve kontrol eder. Şekil 1.4'te eş veri tabanlarına sahip bir istemci ve bir sunucu arasında gerçekleştirilen senkronizasyon işlemine ait zaman diyagramı en basit haliyle gösterilmiştir.



Şekil 1.4. İstemci ile sunucu arasındaki senkronizasyon işlemi adımları

Şekilde senkronizasyonu başlatan düğüm sunucu olarak, senkronizasyon isteğinin gönderildiği düğüm ise istemci olarak tanımlanmıştır. Senkronizasyon yöntemi seçiminde, değişikliklerin iletim yönü dikkate alınarak, poll ya da push temelli yöntemlerden bir tanesi tercih edilir. Push temelli yaklaşımda güncellemeleri ana sunucu gerçekleştirir ve ana sunucudan tek yönlü olarak istemcilere iletilir. Poll temelli yaklaşımda ise senkronizasyonu başlatan düğüm, istemciler ya da sunucular olabilir (Schneider, 1993). Bu nedenle kullanılan ağ topolojisine uygun yöntemin seçilmesi gerekir.

Veri senkronizasyonuna ihtiyaç duyulan pek çok farklı dağıtık sistem mimarisi bulunmaktadır. Bu mimarilerden bir tanesi de içerik dağıtım ağlarıdır. İçerik dağıtım ağları kullanıcılara yüksek performans ve kullanılabilirlik sağlamak amacıyla coğrafi olarak farklı konumlara yerleştirilmiş yerel sunucular ve merkezi sunuculardan oluşan özel bir dağıtık mimaridir. Verilerin farklı konumlarda bulunması nedeniyle

senkronizasyon işlemine ihtiyaç duymaktadırlar. Bir sonraki bölümde içerik dağıtım ağlarından (CDN) detaylı olarak bahsedilecektir.

1.3. Dağıtık Mimarilerde Bulut Temelli İçerik Dağıtım Ağları

İnternetin kullanımının yaygınlaşması ve dijital dünyanın sürekli büyümesi nedeniyle, dijital ortamdaki verilerin 2020'ye kadar 44 zetabayta ulaşması beklenmektedir (IDS, 2018; Wang ve diğ., 2010). Günümüzde boyutu bu denli artmakta olan verilere internet kullanıcıları, ortam ve cihaz kısıtlaması olmadan kolay ve hızlı bir şekilde erişmek istemektedir. Bu nedenle içerik sağlayıcılar kullanıcıların ihtiyaçlarını karşılayacak teknolojilere ve veri dağıtım mimarilerine ihtiyaç duymaktadır. Bu amaçla geliştirilmiş ve yaygın olarak kullanılan içerik dağıtım mimarilerinden bir tanesi içerik dağıtım ağlarıdır (Wang ve diğ., 2010). Benzer şekilde verilere hızlı ve etkin erişim sağlamak amacıyla ortaya çıkmış teknolojilerden bir tanesi de bulut bilişim teknolojisidir. CDN mimarisinin kullanıcıların ihtiyaçlarına göre bulut bilişim teknolojisi ile entegre edilmesi mümkündür. Altbölümlerde bulut bilişim teknolojisi, CDN mimarisi ve CDN mimarisinin bulut bilişim altyapısı ile kullanımı açıklanmıştır.

1.3.1. Bulut bilişim

Bulut bilişim en temel tanımıyla ölçeklenebilir, QoS garantili, basit ve yaygın şekilde erişilebilen, kişiselleştirilebilir ucuz bilgisayar altyapıları sağlayan ağ destekli hizmetler kümesidir (Wang ve diğ., 2010). Farklı bir tanıma göre ise bulut bilişim, pek çok uzak bilgisayar üzerinde tutulan verilere kullanıcıların internet aracılığıyla erişimi, bu veriler üzerinde işlem yapmaları ve verileri birbirleriyle paylaşabilmeleri için oluşturulmuş bir teknolojidir (Sevli, 2011). Bir anlamda bulut bilişim, bulut tabanlı internet teknolojilerinin en genel tanımlamasıdır. Herhangi bir servis alt yapısı gerektirmeden, sanallaştırma teknolojisi aracılığıyla kullanıcılara uygulama ve servislerin sunulmasını amaçlar.

Bulut bilişim servis olarak yazılım (SaaS), servis olarak platform (PaaS) ve servis olarak altyapı (IaaS) olarak adlandırılan üç temel hizmet modeli sunmaktadır. SaaS modeli servis olarak yazılım altyapısı sunmaktadır. Yani bulut altyapısı sağlayıcısı tarafından bulut sunucu üzerinde bulunan yazılımlara kullanıcıların ya da kurumların

erişimi sağlanmaktadır. Kullanıcılar erişmiş oldukları yazılım uygulamasını kurmak, güncellemek gibi pek çok işlemden kurtulmuş olurlar aynı zamanda lisanslama ve yazılım maliyetleri de düşürülmüş olur (Ebem, 2013).

İkinci servis modeli olan PaaS, servis altyapısı olarak platform sunulması anlamına gelmektedir. Yani burada yazılımın çalışması için gerekli olan platform sunulmaktadır. Buradaki platform işletim sistemi, veri tabanı alt yapısı, programlama dili çalıştırmak için gerekli alt yapı gibi farklı servisleri içermektedir (Çam, 2012). Aslında kullanıcıya yükleyeceği uygulama için gerekli teknolojik altyapı sağlanmaktadır ve kullanıcının yalnızca kendi kurduğu uygulama üzerinde yönetim izni bulunmaktadır (Yüksel, 2012).

Üçüncü servis modeli olan IaaS, altyapı olarak servis anlamına gelmektedir. Burada oluşturulan sanal sunucular ile kullanıcılara bulut sunucu hizmeti sağlanmaktadır. IaaS bir veri merkezi olarak düşünülebilir. Kullanıcılar veri merkezi oluşturmak için yazılım, sunucu, alan hizmeti almak yerine bunu bulut altyapısının sunduğu sanallaştırma hizmeti ile elde etmektedirler. Burada kullanıcılara kullandıkları kadar ödeme imkânı da sunulduğu için geleneksel veri merkezlerine göre oldukça düşük maliyetli bir hizmet sunulmaktadır (Çam, 2012; Yüksel, 2012; Wang ve diğ., 2013). Yani IaaS ile kullanıcılara aslında donanım altyapı hizmeti sunulmaktadır (Çam, 2012).

Bulut bilişimin sunduğu servisler göz önünde bulundurulduğunda pek çok avantajı bulunmaktadır. Bunlardan bazıları kullanıcıların uygulamalara web üzerinden erişmesi nedeniyle donanım maliyetini düşürmesi, işlemlerin bulutta gerçekleştirilmesi sebebiyle daha iyi performans sunması, SaaS modeli sayesinde yazılım maliyetlerinin düşürülmesi, limitsiz depolama ortamı sunması, işletim sistemlerinden bağımsız olarak verilere kolayca erişim imkânı ve grup çalışmasına imkân vermesi olarak sıralanabilir (Çam, 2012; Yüksel, 2018). Burada bahsedilen avantajları nedeniyle bulut bilişim teknolojisi pek çok farklı ağ altyapısı ile entegre olarak sunucu hizmeti sağlamaktadır. Bunlardan bir tanesi de içerik dağıtım ağlarıdır. Alt bölümlerde içerik dağıtım ağları ve bulut teknolojisi temelli içerik dağıtım ağlarından bahsedilecektir.

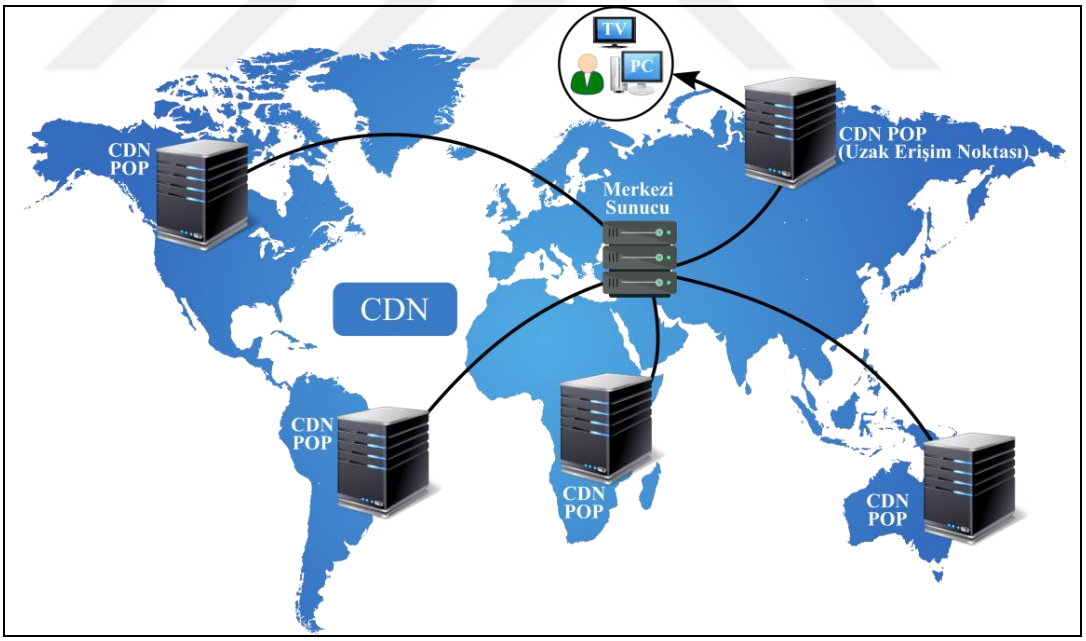
1.3.2. İçerik dağıtım ağları

İçerik dağıtım ağları, içeriklerin kullanıcılara hızlı ve etkili bir şekilde ulaştırılabilmesi için oluşturulmuş özel bir dağıtık ağ mimarisidir. (Douglis ve Kaashoek, 2001). CDN'lerin temelde sunmuş oldukları dört temel hizmet bulunmaktadır. Bunlar;

- Kullanıcı isteklerinin en uygun ve yakın sunucuya yönlendirilmesi,
- Ana sunucu içeriğinin uzak erişim noktalarındaki sunucularda da bulundurulması,
- Kullanıcılara uygun olacak şekilde içerik dağıtım hizmeti sağlanması,
- Ağ bileşenleri ve erişime sunulan içeriğin yönetimi için servislerinin sunulması

olarak sıralanabilir (Pathan ve Buyya, 2007).

Şekil 1.5'te genel bir CDN mimarisi gösterilmiştir. Genel olarak bir CDN mimarisi merkezi bir sunucu, bir istek yönlendirme mekanizması ve uzak erişim noktaları (POP) olarak isimlendirilen uzak erişim noktalarından oluşmaktadır (Wang ve diğ., 2015).



Şekil 1.5. Örnek bir CDN mimarisi

Merkezi sunucu POP'lardaki sunuculara göre çok daha güçlü bir sunucudur. Merkezi sunucu içerisinde, erişim noktalarına dağıtılacak olan içeriğin tamamı barındırılır. POP'lar Şekil 1.5'te de gösterildiği gibi coğrafi olarak farklı noktalarda bulunmaktadırlar. Uzak erişim noktalarındaki sunucular POP sunucular olarak ifade

edilmektedir. Bu sunucuların amacı kullanıcı isteklerine cevap vermektir. İstek yönlendirme mekanizmaları ise istemcilerin isteklerini QoS parametrelerini göz önünde bulundurarak, istemcileri en uygun POP sunucuya yönlendirmekten sorumludur (Wang ve diğ., 2015).

CDN mimarisinin pek çok avantajı bulunmaktadır. Kullanılan POP sunucular sayesinde hem uzak erişim noktalarında hem de merkezi sunucudaki yük azaltılmış ve dengelenmiş olur, kullanıcılar kendilerine en yakın olan sunucular aracılığıyla içeriklere erişeceklerinden, erişim süresi tek bir merkezi sunucu ile hizmet veren ağlara göre oldukça düşüktür, herhangi bir sunucu erişilemez durumda olduğunda diğer POP sunucular üzerinden hizmet vermeye devam edilir ve tutulan istatistikler sayesinde kullanıcılara detaylı raporlama sunulabilir. Bunların yanında CDN mimarisinin en önemli dezavantajları maliyet ve bazı organizasyon mimarileri açısından çoklu sunucu yönetiminin pratik olmaması olarak sıralanabilir.

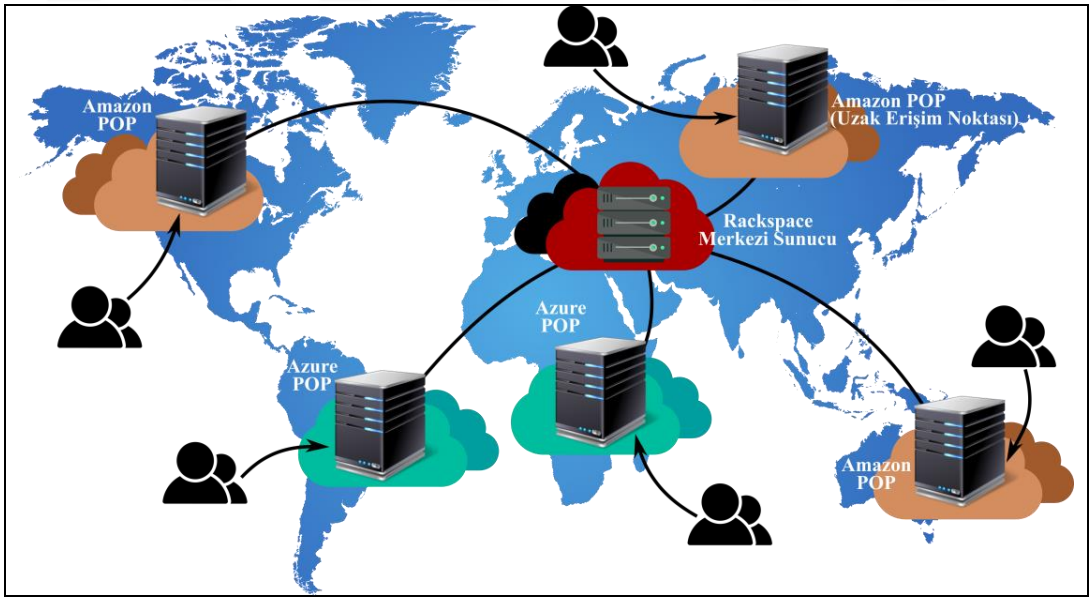
1.3.3. Bulut temelli içerik dağıtım ağları

CDN'lerin içeriklerin kullanıcılara ulaştırılması konusunda önemli faydaları olmuştur. Ancak CDN altyapısını kullanarak içerik dağıtımı yapan içerik sağlayıcılarının pek çoğu üçüncü parti CDN sağlayıcıları aracılığıyla bu işlemi gerçekleştirmektedir. Bu durumda sunulan hizmet coğrafi olarak içerik sağlayıcılarının altyapı kısıtlamalarına maruz kalmaktadır (Wang ve diğ., 2015). Bunlara ek olarak CDN hizmeti almanın maliyeti, depolama ve bant genişliği ile ilgili kısıtlamalar nedeniyle bulut temelli içerik dağıtım ağı (CCDN) olarak isimlendirilen bulut temelli içerik dağıtım ağları tasarlanmaya başlanmıştır (Chen ve diğ., 2012). Bir CCDN, içerik sağlayıcılarının tercihlerine bağlı olarak bir veya daha fazla bulut sunucusunda içeriklerin depolanıp eşleştirilmesine olanak sağlamaktadır (Yin ve diğ., 2010).

CCDN mimarilerinin tercih edilmesinin temelde dört nedeni bulunmaktadır. Bunlardan ilki bu mimariler ile sunulan kullandığın kadar öde olarak isimlendirilen ödeme yapısıdır. Kullanıcılar klasik CDN mimarisinde olduğu gibi fiziksel olarak sunucuları satın almadıkları için ihtiyaçları kadar alan tahsisi yaparlar ve hizmet kullanımları oranında da ödeme yaparlar. İkinci avantajı iletim gecikmesini azaltabilmek amacıyla bulut sağlayıcı aracılığıyla kaynaklarının kolayca

iyileştirilmesidir. Bir diğer avantajı küçük bölgelerde sunucu konumlandırılması gerektiğinde CDN'de olduğu gibi fiziksel olarak altyapı ihtiyacının olmaması ve bölgedeki mevcut bulut sunucuların kolaylıkla kullanılabilmesidir. Dördüncü avantajı ise sunucularda dinamik yük dengelemesi sağladığı için farklı altyapı ihtiyacı olan uygulamaların aynı CDN içerisinde çalıştırılmasını desteklemesidir (Wang ve diğ., 2015).

CDN ve bulut bilişim yapılarının teknik olarak bir araya getirilmesinin pek çok farklı örneği vardır (Ling ve diğ., 2013; Ilie ve Datta, 2016; Papagianni ve diğ., 2013; Wang ve diğ., 2011). Bazı CCDN mimarileri sadece merkezi sunucuyu bulut sunucu olarak kullanır ve geri kalan mimariyi klasik CDN altyapısını kullanarak oluşturur. Bazı CCDN mimarilerinde ise tüm sunucular bulut sunucu olarak hizmet verir. Bu mimarinin örneği Şekil 1.6'da gösterilmiştir. Burada master-slave topolojisi vardır. Ana bulut sunucudan içerik diğer POP sunuculara aktarılır. POP sunucular, bir içeriğe ihtiyaçları olduğunda, ana sunucu ile haberleşirler ve gerekli içeriği temin ederler. Şekilde POP sunucuların her biri bir bulut sağlayıcı ile gösterilmiştir. Ana sunucu her açıdan merkezi bir yönetim noktası konumundadır (Wang ve diğ., 2015).



Şekil 1.6. Örnek bir bulut temelli CDN mimarisi

Tez çalışması kapsamında merkezi bir bulut sunucuya sahip CCDN mimarisi benzeri bir mimari için senkronizasyon planlama mekanizması önerilmiştir. Böyle bir mimaride verilerin aktarımı sırasında ağın analizini yapabilmek ve mimariyi tam

olarak modelleyebilmek için ağ üzerindeki arka plan trafiğinin ve ağ analiz parametrelerinin doğru olarak belirlenmesi gerekmektedir. Tezin ikinci bölümünde arka plan trafiğinin ağlar üzerindeki etkisi ve ağ analizi için kullanılabilecek parametrelerin detaylı incelemesi yapılmıştır.

2. ARKA PLAN TRAFİĞİ VE AĞ ANALİZİ

Dağıtık mimarilerde gönderilecek olan veriler yerel alan ağı (LAN) ortamından çıktıktan sonra ana sunucuya ulaşmak için internet bulutundan geçmek zorundadır. İnternetin çok büyük ve pek çok farklı ağ tarafından paylaşılan bir ortam olduğu düşünülürse arka plandaki trafiğin uygulamamız üzerinde ciddi bir etkisi olacaktır. Bu sebeple geliştirilecek olan dağıtık sistem mimarisinin tasarlanması sürecinde arka plan trafiğinin etkisinin de göz önünde bulundurulması gerekmektedir. Arka plan trafiği modellendikten sonra ağın durumunun analizi için belirli analiz parametrelerine ihtiyaç duyulmaktadır. Tezin bu bölümde ilk olarak arka plan trafiğinin ağ üzerindeki etkisinden daha sonra ağ analizi için kullanılabilecek parametrelerden ve ağı modellemek için kullanılan olasılık dağılımlarından bahsedilecektir.

2.1. Arka Plan Trafiği ve Ağ Üzerinde Etkisi

Ağların benzetimi sırasında kullanılan protokollerin, uygulamaların ve kullanıcıların etkisini doğru bir şekilde değerlendirebilmek için temsili bir trafik oluşturmak çok önemlidir. Simülatörler aracılığıyla modelleme yapılırken dikkate alınması gereken iki tip trafik bulunmaktadır. Bunlardan ilki hedef uygulama için modellenecek olan ön plan trafiği ve diğeri de ağdaki diğer uygulamalar tarafından oluşturulan arka plan trafiğidir. Arka plan trafiği hedef uygulamanın üzerinde önemli bir etkiye sahiptir. Çünkü arka plan trafiği ağ kaynaklarının kullanımı konusunda hedef uygulama ile yarışır. Bu da hedef uygulamanın davranışını önemli ölçüde etkileyecektir (Li, 2014).

Dağıtık mimarilerdeki sistemleri geliştirirken arka plandaki trafiğin nasıl ya da hangi boyutta bir etkisinin olduğu pek çok farklı araştırmacı tarafından analiz edilmiştir. Venkatesh ve Vahdat (2008) tarafından yapılan bir çalışmada gerçek zamanlı olarak düşünülürse arka plandaki trafik yarışının servis ve protokol davranışını etkilediği gösterilmiştir. Uygulamalar arka plan trafiği ile yarışmak zorunda kaldıklarında normalden farklı davranmaktadırlar. Bu nedenle farklı uygulamaların hem sentetik arka plan trafiği hem de gerçek arka plan trafiği altında davranış analizini

yapmışlardır. Bu analiz süresince sentetik trafik üretmek için Constant Bit Rate ve Poisson dağılımları kullanılmıştır. Üretilen trafiklerin üç uygulama davranışı üzerinde analizi yapılmıştır. Elde edilen sonuçlara göre; arka plan trafiğinin bant genişliğinin tahmini gibi uygulamalarda gerçekten etkili olduğu tespit edilmiştir. Belirli bir seviyede arka plan trafiği varken herhangi bir zamandaki meşguliyet karakteristiğinin uygulamanın bireysel karakteristiğine karşılık geleceği anlaşılmıştır. Genel olarak hiper-metin transfer protokolü (http) uygulamalarının arka plan trafiğinin çok yoğun olmasına duyarlı olduğu tespit edilmiştir ve buradaki en önemli etkenin transfer edilen nesnelerin boyutu olduğu da belirtilmiştir. Multimedia uygulamaları http uygulamaları kadar arka plan trafiğine duyarlı değildir ancak bant genişliğini tahmin etmek için kullanılan araçlar da http uygulamaları gibi arka plan trafiğine duyarlıdır. Sonuç olarak her uygulamanın trafikteki meşguliyetten uygulamanın türüne bağlı olarak belirli düzeylerde etkileneceği tespit edilmiştir.

Venkatesh ve Vahdat (2009) ilerleyen yıllarda daha gerçekçi arka plan trafiği elde edebilmek için yeni bir çalışma yapmışlardır. Çalışmada uygulamalara özgü yapısal trafik modellerinin başarılı bir şekilde oluşturulabileceği gösterilmiştir. Yeni trafik oluşturulurken özgün trafiğe ait;

- Paketlerin gönderim sıklığı ve varış zamanı
- Paket boyutlarının dağılımı
- Akışların karakteristikleri ve
- Hedef internet protokolü (IP) ve hedef port adresleri dikkate alınmıştır.

Yapılan çalışma ile farklı uygulama, ağ ve kullanıcı koşullarında üretmiş oldukları trafiğin gerçeğe uygun olduğunu göstermişlerdir.

Nahum ve diğ. (2001) tarafından yapılan farklı bir çalışmada WAN koşullarının sunucuların performansları üzerinde ciddi bir etkiye sahip olduğu gösterilmiştir. Bu çalışma ile kayıp paketlerin sunucuların performansını yüzde elli azalttığı ve cevap zamanını da aynı oranda arttırdığı gösterilmiştir. Gerçek hayatta WAN ortamında yer alan her sunucu yüksek paket kayıpları ve gecikmeler gibi sorunlarla karşılaşmaktadır bu sebeple sunucuların performansları değerlendirilirken WAN karakteristiklerinin de göz önünde bulundurulması gerekmektedir. Bu yüzden

çalışmada ilk olarak sunuculardaki yoğunluğa sebep olabilecek parametreler belirlenmiştir ve temelde üç parametre üzerinde durulmuştur. Bunlar;

- Dosya boyutu, dosya tipi ve dosyaya erişim sıklığı
- İstek oranı ve isteklerin varış zamanı
- Kullanılan protokolün versiyonu olarak belirlenmiştir.

Çalışmada paket kayıplarının daha çok internetin meşgul olduğu zamanlarda gerçekleştiğine değinilmiştir. Yönlendiricinin paketleri düşürmesi için ilk olarak tıkanıklık durumunu fark etmesi ve depolama alanının da kısıtlı olması gerekmektedir. Sınırlı olan kaynaklar (yönlendirici depolama alanı gibi) belirli bir zaman periyodunda erişilemez olarak kalacaktır. Bu sebeple de hem ağ performansında hem de sunucu performansında düşüşler meydana gelecektir. Çalışma sonucunda şu tespitler yapılmıştır. Verim açısından değerlendirildiğinde; yüksek paket kayıplarının düşük verim değerleri altında görüldüğü gözlemlenmiştir. Sunucunun birim zamanda iletilen veri miktarı açısından doygunluğa ulaşabilmesi için yükünün arttırılması gerekir ve yüksek RTT değerleri düşük verim değerlerine karşılık gelmektedir. Bu yüzden paketlere gecikme eklemek elde edilecek verim değeri üzerinde yüksek etkiye sahiptir. Sonuçlar cevap zamanı açısından değerlendirildiğinde ise yükün artması durumunda iş kuyruklarının uzadığı ve isteklerin servis için daha uzun süreler beklemek zorunda kaldığı görülmüştür. Sunucunun yükü az olduğu durumlarda da cevap zamanının ağ tarafından sınırlandırılabileceği görülmüştür. (Burada sunucu yükünün artması durumuna örnek olarak; istek sayısının artması ya da büyük boyutlu dosyaların transferlerinin yapılması gösterilebilir.)

Aynı zamanda TCP'nin farklı versiyonları kayıp durumlarında farklı tepkiler vermektedir. Bu tepkiler genelde RTT ye göre belirlenmektedir. TCP New Reno versiyonu kayıp paketleri kurtarma işini TCP Reno'dan daha hızlı gerçekleştirmektedir. Ancak TCP Seçici Alındı (Sack), Reno ya da New Reno kullanmak, sunucunun tüm kapasitesinde ciddi değişikliklere ya da azalmaya sebep olmamaktadır. Çünkü sunucular paket gönderimi için bir kapasiteye sahiptir. TCP versiyonunu değiştirmek paketlerin ağdaki dağılımını etkiler ancak sunucunun genel kapasitesinde ciddi değişikliklere yol açmaz.

Eylen ve Bazlamaçı (2015) tarafından paket iletimindeki tek yönlü gecikmeyi saat eşleştirmesine ihtiyaç duymadan ölçebilecek bir sistem geliştirilmeye çalışılmıştır. Yapılan çalışmada gerçek trafik benzeri trafikler elde edebilmek için arka plan trafiğine ihtiyaç olduğundan bahsedilmiştir. Bu nedenle çalışmada kullanılan deneme paketlerine rastgele gecikmeler ekleyebilmek için Poisson dağılımı kullanılarak arka plan trafiği üretilmiştir. Trafik üç farklı oranda üretilmiştir. Buradaki amaç tıkanıklığı sağlamaktır. Birinci durum "heavy load\high load" olarak adlandırılan yüksek yüklü trafik durumudur. Bu durumda aktarımın yapılacağı link aşırı yüklü durumdadır. Yani ağa linkin kapasitesinden daha fazla bir arka plan trafiği enjekte edilmiştir. Bu durumda tıkanıklık oluşacaktır ve kuyruk boyutu artacaktır. İkinci durum "low load" olarak adlandırılan durumdur. Böyle bir durumda göndericinin gönderim kuyruğu boş olacak şekilde paket gönderimi yapılmaya çalışılmıştır. Bu durumda arka plan trafiğinin yoğunluğu link kapasitesine göre çok azdır. Böyle bir oranda trafik oluşturulmasının amacı gecikmedeki ani değişimleri yakalayabilmektir. Üçüncü durum ise "slience rate" olarak adlandırılan trafik oranıdır. Bu durumda oluşturulan trafik oranı linkin kapasitesinin yarısı dolu olacak şekilde ayarlanmıştır. Bu durumun oluşturulmasının sebebi de gecikmeye uğramayan paketlerin varlığını tespit edebilmektir. Oluşturulan üç tip trafik ile gerçek trafik modellenmiştir ve önerilen yöntemin analizi arka plan trafiği altında daha doğru bir şekilde yapılmıştır.

Levesque ve Tipper (2015) tarafından gerçekleştirilen farklı bir çalışmada ise asimetrik gecikme şartları altında noktadan noktaya eşleme zamanının tahminini doğru bir şekilde sağlayabilmek için yine arka plan trafiği oluşturulmuştur. Çünkü ağdaki değişken kuyruk gecikmelerinin en temel sebebi ağın genel trafik yüküdür. Trafik yükü kayda değer bir oranda az olursa eşleme hataları da o oranda azalacaktır ve trafik yükünün artmasıyla birlikte eşleme hataları da artacaktır. Bu hatalar özellikle asimetrik trafik yükleri altında daha da belirgin şekilde artmaktadır. Burada oluşturulan trafik daha çok tıkanıklık durumunu elde edebilmek için Pareto dağılımı kullanılarak Objektif Modüler Ağ Simülatörü (Omnet++) yazılımı ile sağlanmıştır. Yine geliştirilen yöntemin analizi farklı trafik koşulları altında test edilmiştir ve arka plan trafiğinin etkisi belirgin bir şekilde gösterilmiştir.

Yapılan çalışmalar incelendiğinde arka plan trafiğinin hem uygulamalar hem de sunucular üzerinde ciddi bir etkisinin olduğu açık bir şekilde görülmektedir. Bu

etkinin boyutu pek çok farklı parametreye göre değişmektedir. Bu nedenle uygulama analizlerinin gerçeğe uygun olarak yapılabilmesi için arka plan trafiğinin mutlaka göz önünde bulundurulması gerekmektedir.

2.2. Ağ Analizi İçin Kullanılacak Parametrelerin Tespiti

Simülasyonu yapılacak ağların modellenmesi gerçekleştirildikten sonra farklı zamanlardaki ağ yoğunluğu analizinin gerçekleştirilebilmesi için farklı yoğunluklar altındaki davranışının incelenmesi gerekmektedir. Bu amaçla ağın analizi için kullanılabilecek doğru parametreler tespit edilmelidir. Ağlardaki paket kayıpları ve yeniden iletimler dolayısıyla ağın durumundaki ani değişiklikler genellikle ağ trafiğinin yoğun olduğu zaman dilimlerinde gerçekleşmektedir. Trafikte aşırı yoğunluğun oluştuğu noktalarda da tıkanıklık meydana gelmektedir.

Mevcut TCP iletim protokollerinin farklı versiyonları tıkanıklık durumunu farklı yollarla tespit etmektedirler (Venkataramani ve diğ., 2002). TCP Reno, New Reno, Tahoe ve SACK versiyonları paket kayıplarını tıkanıklık sinyali olarak kullanmaktadır. Ancak bir paket kaybının olduğunun fark edilmesi, tıkanıklık için çok uzun bir süre olabilir. Bu sürede yönlendiricinin tamponunun dolması ile daha fazla paket kaybı gerçekleşebilir. TCP'nin mevcut versiyonlarına alternatif olarak çıkarılan TCP Vegas versiyonu diğerlerinden farklı olarak RTT değerlerine bağlı olarak tıkanıklık tespiti yapmaya çalışır. TCP Vegas ta RTT ye bağlı olarak beklenen verim ve gerçek verim hesabı yapılır. Buradan elde edilen sonuçlara göre çerçeve boyutu hesaplanır. Ancak çerçeve boyutu hesabında kullanılan parametrelerin doğru olarak seçilememesi durumunda oluşacak paket kayıpları önlenemez. TCP Vegas tıkanıklık kuyruğunda 1-3 aralığında paket tutulmasına izin vermektedir. Arka plan trafiği çoğaldığında bu durum gereksiz bir engellemeye sebep olacaktır.

TCP de yer alan tıkanıklık tespiti ve tıkanıklık önleme mekanizmalarının daha etkin kullanılabilmesi için tıkanıklık olma durumunu uygulama seviyesinde tespit edip tıkanıklık oluşmadan çözümler üretebilecek mekanizmalara ihtiyaç vardır. Tıkanıklık durumuna uygulama katmanında sunulacak çözümler üç sınıfa ayrılabilir (Ren ve diğ., 2014). Bunlar;

- Sunucuların istemcilere cevap olarak her istekte göndereceği verilerin boyutlarının arttırılması, (Böylece gönderilen cevaplar daha büyük boyutlarda gönderilir ve gönderilecek paketlerin sayısı azaltılabilir.)
- Verilerin aktarımlarının birbirleriyle çakışmayacak şekilde planlanması ve
- Veri aktarımının zamanlanması (Bu duruma örnek olarak istemciler arasında jeton kullanımı verilebilir. Sadece jetona sahip olan istemcilerin veri göndermesi sağlanarak eş zamanlı iletim yapacak kullanıcılar sınırlandırılabilir.) olarak sıralanabilir.

Burada bahsedilen çözümlerin uygulanabilmesi için tıkanıklık durumunun oluşmadan ya da oluştuktan sonra kısa bir süre içerisinde fark edilmesi gerekmektedir. Bu nedenle tıkanıklık durumunun tespiti için kullanılacak parametrelerin doğru bir şekilde tespit edilmesi gerekir.

Cisco tarafından uygulama seviyesinde tıkanıklık yönetimi için geliştirilmiş Netflow (Cisco IOS Release 12.4, 2011), Sflow (Cisco Release 7.x, 2015) gibi yazılımlar bulunmaktadır. Bu yazılımlar tıkanıklık kontrolü için;

- Cevap zamanı,
- Ağ kaynaklarının erişilebilirliği,
- Uygulama performansı,
- Video verilerinin transferi için jitter,
- Bağlantı zamanı,
- Verim ve
- Paket kayıplarını parametre olarak kullanmaktadır.

Bunların yanında heterojen bir yapıya sahip ağlarda tıkanıklık tespiti için kullanılabilecek olan parametreler de Floyd (2018) tarafından aşağıdaki gibi listelenmiştir;

- Verim: Bir noktadan başka bir noktaya belirli bir zamanda aktarılan veri miktarıdır. Birimi bps'dir.
- Gecikme: Verinin bitlerinin bir noktadan başka bir noktaya iletilene kadar geçen süredir. Temelde dört farklı gecikmenin bir araya gelmesi ile hesaplanır. Bunlar; yönlendiricinin paket başlığını işlemesi için geçen süre, yönlendirme kuyruğunda

paketin harcadığı süre, paketin bitlerini linke aktarmak için gereken süre ve paket hedefe ulaşana kadar geçen süre.

- Kayıp Paketler: Bilgisayar ağında transfer edilen paketlerden hedefe varmada başarısız olanları temsil eder.
- Cevap Zamanı: Sunucudan istekte bulunulup, cevabın ilk karakteri istemci bilgisayarında gözükene kadar geçen süredir. Bazen RTT olarak da kullanılır.

Bu parametreler tıkanıklık tespiti ya da tıkanıklık önleme amaçlı kullanılan genel parametrelerdir. Veri tabanlarının eşlenmesi sürecinde yapılan işlem toplu veri transferi olarak adlandırılmaktadır ve burada aktarılan veriler büyük boyutlu verilerdir. Bu süreçte standart tıkanıklık kontrol algoritmalarından (Beş farklı tıkanıklık önleme mekanizması bulunuyor. Bunlar; tıkanıklık önleme, yeniden iletim zaman aşımı, yavaş başlangıç, hızlı yeniden iletim ve hızlı kurtarma) yeniden iletim zaman aşımı ve hızlı kurtarma yöntemlerinin mutlaka kullanılması gerekmektedir (Mathis, 2018). Çünkü aktarılan veriler büyük boyutludur ve tıkanıklık durumunda paket kayıpları daha fazla olabilir. Bu sebeple uygulama katmanına yerleştirilecek mekanizmalar ile bu kayıpların daha da azaltılması ya da hiç kayıp olmaması sağlanabilir.

Bu tez çalışmasında simülasyonu yapılacak ağlar için uygun arka plan trafiği oluşturulduktan sonra farklı trafikler altında gönderilen senkronizasyon paketlerine ait gerekli ağ analiz parametrelerinin kaydedilmesi gerekmektedir. Bu aşamada yukarıda bahsedilen parametrelerden dört tanesi kullanılmıştır. Bunlar; verim, gecikme, yeniden iletim sayıları ve cevap zamanı olarak belirlenmiştir. Bu parametreler Cisco tarafından da tıkanıklık tespiti için kullanılan standart parametrelerdir.

2.3. Ağ Modellemede Kullanılan Olasılık Dağılımları

Tez çalışmasının bu bölümünde, tez kapsamında oluşturulan dağıtık ağ mimarisinde bulunan LAN'lara arka plan trafiği eklemek ve gönderilen senkronizasyon paketlerini modellemek için kullanılan olasılık dağılımlarının detaylı açıklaması yapılacaktır.

2.3.1. Pareto dağılımı

Pareto dağılımı 1848 yılında İtalyan bilim adamı Vilfredo Pareto tarafından ortaya atılmıştır (Walck, 2007). Sürekli değişkenler için tanımlı bir olasılık modelidir. Pareto dağılımı ekonomi, şehir boyutları, çevreyle ilgili olaylar, güvenilirlik çalışmaları, ağ paket analizi, ağ davranışı modelleme çalışmaları gibi pek çok farklı alanda yapılacak modellemelerde önemli bir rol oynamaktadır. Pareto dağılımı sürekli bir dağılımdır. Dağılıma ait olasılık yoğunluk ve birikimli dağılım fonksiyonları, dağılımın ortalama ve varyans formülleri sırasıyla Eşitlik (2.1), Eşitlik (2.2), Eşitlik (2.3) ve Eşitlik (2.4)'te gösterilmiştir;

$$f(x;w,y)=\begin{cases}\dfrac{wy^{w}}{x^{w+1}}, & x\geq y>0,\ w>0\\ 0, & \text{aksi durumda}\end{cases} \tag{2.1}$$

$$F(x;w,y)=1-\left(\frac{y}{x}\right)^{w},\ x\geq y>0, w>0 \tag{2.2}$$

$$E[X]=\frac{wy}{w-1},\ \ w>1 \tag{2.3}$$

$$Var[X]=\frac{wy^{2}}{(w-2)(w-1)^{2}},\ w>2 \tag{2.4}$$

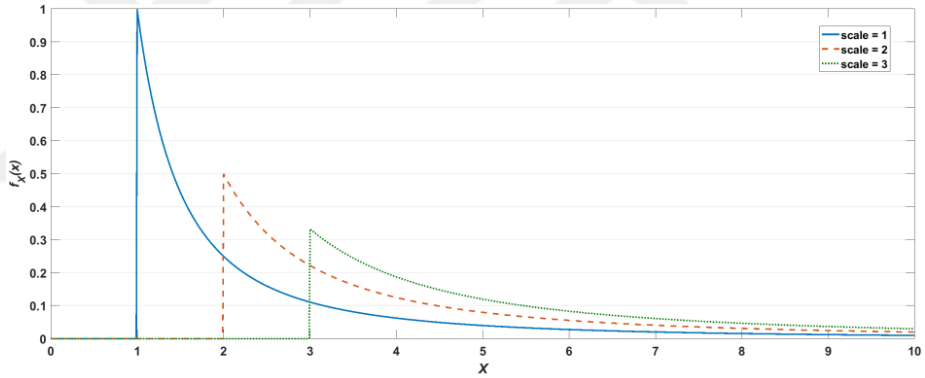
Rastgele sürekli bir X değişkeni için Pareto dağılımına ait olarak verilen formüllerdeki w değişkeni, w> 0 koşuluna sahip şekil parametresini ve y değişkeni, y > 0 koşuluna sahip ölçek parametresini temsil etmektedir (Dayyeh ve diğ., 2013). Dağılıma ait olasılık yoğunluk fonksiyonunun grafiği kayak pistine benzer. Kayak pisti tabiri aslında verinin dağılımdaki eşitsizliği ve aşırı derecede büyük, uç değerlerinde dağılımda bulunması sebebiyle kullanılmıştır. Farklı ölçek ve şekil değerleri için yoğunluk fonksiyonunun değişim grafiği sırasıyla Şekil 2.1 ve Şekil 2.2'de gösterilmiştir. Ölçek parametresi olasılık yoğunluk fonksiyonun sol kenarının konumunu ayarlar. Şekil parametresi de grafiğin eğimini belirler. Şekil 2.1'de şekil parametresi sabit (1 iken) iken ölçek parametresinin 1,2 ve 3 değerleri için başlangıç noktasının değişimi tam olarak görülmektedir. Şekil 2.2'de de ölçek parametresi 1

olarak belirlenmişken şekil parametresinin 1,2 ve 4 değerleri için grafiğin başlangıç noktasının aynı olduğu ve grafiklerin eğimlerinin farklı olduğu görülmektedir.

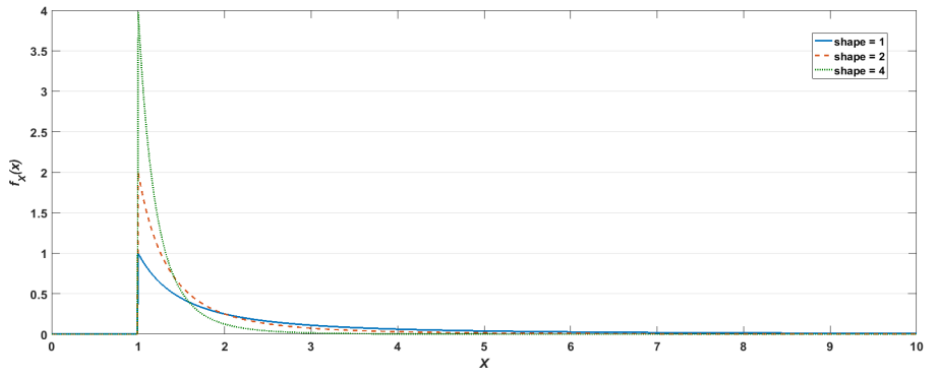
Genelleştirilmiş Pareto dağılımına uygun sayılar üretmek için matlab programında aşağıdaki fonksiyonlar bulunmaktadır;

- gprnd(k,sigma,theta)
- gprnd(k,sigma,theta,m,n,...)
- gprnd(k,sigma,theta,[m,n,...])

Burada k olarak gösterilen parametre şekil parametresini, sigma, ölçek parametresini ve theta değeri eşik parametresini göstermektedir. Eşik parametresi ölçek/şekil olarak hesaplanmaktadır. gprnd(k,sigma,theta) fonksiyonu ile tek boyutlu dizi olarak sayı üretilirken, gprnd(k,sigma,theta,m,n,...) ve gprnd(k,sigma,theta,[m,n,...]) fonksiyonları mxn boyutunda rastgele sayı üretimi yapılabilir.



Şekil 2.1. Pareto dağılımına ait farklı ölçek değerleri için olasılık yoğunluk fonksiyonu değişim grafiği



Şekil 2.2. Pareto dağılımına ait farklı şekil parametreleri için olasılık yoğunluk fonksiyonu değişim grafiği

2.3.2.Weibull dağılımı

Weibull dağılımı sürekli bir olasılık dağılımıdır. 1887 yılında İşveç matematikçi Waloddi Weibull tarafından ortaya atılmıştır (Walck, 2007) ve 1951 yılında detaylı olarak açıklanmıştır. Weibull dağılımı çoğu kez normal dağılım yerine kullanılmaktadır. Fabrikalarda mal teslim zamanlarının modellenmesinde, radar sistemlerinde sinyal seviyelerinin dağılımını modellemede, hava durumu tahminlerinin modellenmesinde, rüzgâr hızı dağılımlarını modellemede ve TCP protokolü ile paket iletimi yapan yoğun kuyruk yapısı olmayan ağları modellemede yaygın olarak kullanılmaktadır. Weibull dağılımına ait olasılık yoğunluk ve birikimli dağılım fonksiyonları, dağılımın ortalama ve varyans formülleri sırasıyla Eşitlik (2.5) , Eşitlik (2.6), Eşitlik (2.7) ve Eşitlik (2.8)'de gösterilmiştir.

$$f(x;w,y)=\begin{cases}\frac{w}{y}\left(\frac{x}{y}\right)^{w-1}e^{-(x/y)^{w}}, & x\geq 0,\ y\in(0,\infty),\ w\in(0,\infty)\\ 0, & x<0\end{cases} \tag{2.5}$$

$$F(x;w,y)=1-e^{-(x/y)^{w}} \tag{2.6}$$

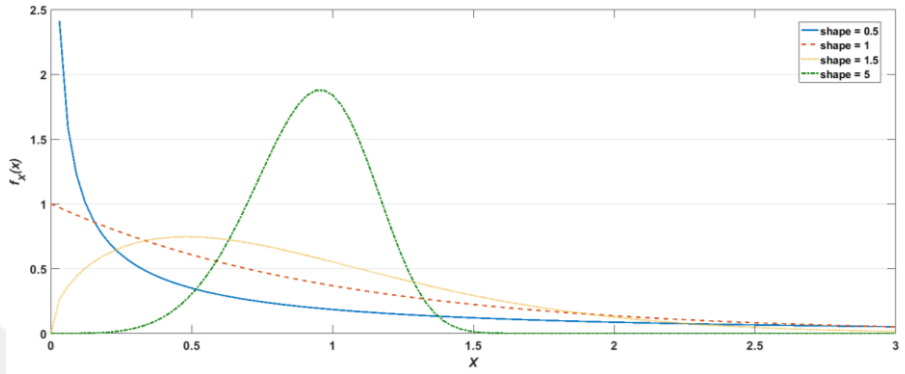
$$E(X)=y\Gamma\left(1+\frac{1}{w}\right) \tag{2.7}$$

$$Var(X)=y^{2}[\Gamma\left(1+\frac{2}{w}\right)-\Gamma^{2}\left(1+\frac{1}{w}\right)] \tag{2.8}$$

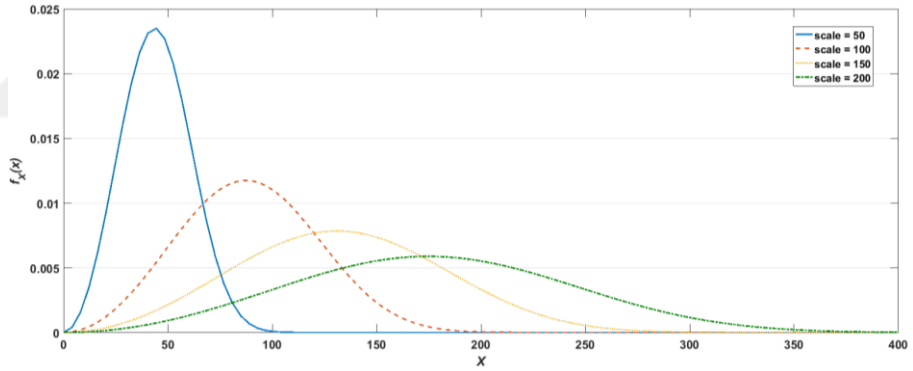
Formüllerde yer alan w değişkeni şekil parametresini ve y değişkeni ölçek parametresini temsil etmektedir. Weibull dağılımı için birikimli olasılık fonksiyonu üstel bir fonksiyondur. Şekil 2.3 ve Şekil 2.4'te sırasıyla farklı şekil ve ölçek değerleri altında Weibull dağılımına ait olasılık yoğunluk fonksiyonunun grafikleri gösterilmiştir.

Grafiklerden de anlaşılacağı gibi Weibull dağılımındaki şekil ve ölçek parametreleri dağılım üzerinde farklı açılardan benzer etkiye sahiptir. Ölçek parametresinde yapılacak değişiklikler olasılık yoğunluk fonksiyonu (pdf) grafiğinin yatay eksendeki ölçek değişikliklerine karşılık gelir. Şekil parametresi sabitken ölçek parametresinin arttırılması pdf eğrisinin yatay eksende uzamasına, azaltılması da yatay eksende

daralmasına sebep olur. Şekil parametresi aynı zamanda yamaç olarak da adlandırılır. Olasılıktaki regresyon eğrisinin eğimine eşittir. Yukarıda da bahsedildiği gibi şekil parametresinin aldığı bazı değerler dağılımın farklı dağılımlara indirgenmesine sebep olur (Nielsen, 2011). Aynı zamanda şekil parametresi pdf fonksiyonun şekli üzerinde Şekil 2.3'ten de anlaşılacağı gibi belirgin bir etkiye sahiptir.



Şekil 2.3. Weibull dağılımına ait farklı ölçek değerleri için olasılık yoğunluk fonksiyonu grafiği



Şekil 2.4. Weibull dağılımına ait farklı şekil parametreleri için olasılık yoğunluk fonksiyonu grafiği

Weibull dağılımına uygun sayılar üretmek için matlab programında aşağıdaki fonksiyonlar bulunmaktadır;

- wblrnd(A,B)
- wblrnd(A,B,m,n,...)
- wblrnd(A,B,[m,n,...])

Fonksiyonlarda yer alan A değişkeni her sürekli X değeri için ölçek parametresini ve B değişkeni de şekil parametresini temsil etmektedir. Buradaki şekil ve ölçek parametreleri mutlaka pozitif değerler almalıdır. wblrnd(A,B) fonksiyonu tek boyutlu

dizi olarak rasgele Weibull sayıları üretirken, wblrnd(A,B,m,n,...) ve wblrnd(A,B,[m,n,...]) fonksiyonları da m x n boyutunda rastgele sayılar dizisi üretir.

2.3.3. Poisson dağılımı

Poisson dağılımı 1837' de Fransız matematikçi Sim´eon Denis Poisson tarafından ortaya atılmıştır ve onun ismi ile adlandırılmıştır (Walck, 2007). Poisson dağılımı istatistik temelli pek çok uygulamayı modellemek için önemli bir dağılımdır. Örnek olarak bir saat aralığında belirli bir internet sitesine gelen bağlantıların sayısı, her bir on dakika içinde bir telefon merkezine gelen telefonların sayısı, bir kavşaktan belirli bir zaman aralığında gelen araçların sayısı verilebilir. Poisson dağılımının odaklandığı olay sayısı rassal bir değişkendir ve sayılabilen bir olaydır. Poisson dağılımı Poisson süreci ile birlikte ortaya çıkar ve kesikli bir dağılımdır, dağılıma ait olasılık kütle fonksiyonu Eşitlik (2.9)'da, birikimli dağılım fonksiyonu Eşitlik (2.10)'da, ortalama ve varyans değerleri ise Eşitlik (2.11)'de gösterilmiştir.

$$f(x;\lambda) = \frac{\lambda^x e^{-\lambda}}{x!}, \ \lambda \in (0,\infty) \tag{2.9}$$

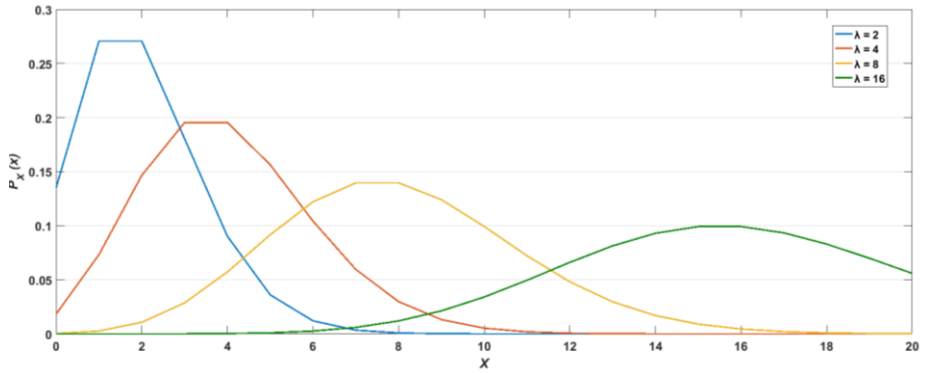
$$F(x;\lambda) = \sum_{i=0}^{x} \frac{\lambda^i e^{-\lambda}}{i!} \tag{2.10}$$

$$E[X] = \lambda \ , Var[X] = \lambda \tag{2.11}$$

Eşitlikte yer alan x değişkeni $x \geq 0$ koşulunu sağlayan bir tamsayıyı ve λ değişkeni de pozitif gerçel değerler alan sabit aralıkta ortaya çıkan olayların sayınının beklenen değerini (ortaya çıkmanın ortalama sayısını) ve e değeri doğal logaritma tabanını (e = 2,71828) temsil etmektedir. Yukarıdaki formülde x tamsayıda olayın ortaya çıkma olasılığı verilmiştir. Şekil 2.5'te farklı λ değerleri için olasılık kütle fonksiyonu ve Şekil 2.8'de farklı λ değerleri için birikimli dağılım fonksiyonu grafikleri verilmiştir.

Olasılık kütle fonksiyonuna ait grafikteki iki ardışık x değerleri karşılaştırıldığında dağılımın $x + 1 < \lambda$ koşulu sağlanana kadar artmaya devam ettiği görülmektedir. Koşul sağlandıktan sonra da olasılık değeri sıfıra kadar azalır. Düşük λ değerleri için çarpık sonuçlar üretebilir. Poisson dağılımı λ değeri sonsuza giderken λ ortalama değeri ve λ varyansı ile normal dağılım gibi davranır. λ değeri Poisson dağılımı için

hem ortalama değer hem de varyansı temsil etmektedir. Ortalama değeri ve varyansı aynı olan tek dağılım Poisson dağılımıdır.



Şekil 2.5. Poisson dağılımına ait farklı lambda değerleri için olasılık kütle fonksiyonu değişim grafiği

Poisson dağılımına uygun sayılar üretmek için matlab programında aşağıdaki fonksiyonlar bulunmaktadır.

- poissrnd(lambda)
- poissrnd(lambda,m,n,…)
- poissrnd(lambda,[m,n,…])

poissrnd(lambda) fonksiyonu ortalama parametresi olan lambda (λ)'yı kullanarak Poisson dağılımına uygun sayılar üretir. Buradaki lambda değeri bir vektör, bir matris ya da çok boyutlu bir dizi olabilir. Elde edilecek sayıların miktarı λ'nın büyüklüğü kadardır. poissrnd(lambda,m,n,…) veya poissrnd(lambda,[m,n,…]) fonksiyonları da m x n boyutunda rastgele Poisson sayılar dizisi üretir. Bu fonksiyonlar için λ parametresi elde edilen m x n dizisi ile aynı büyüklükte bir dizi ya da skaler olabilir. Poisson dağılımına uygun rastgele tamsayılar üretmenin diğer bir yolu da Knuth (2016) tarafından bir algoritma kullanılarak ifade edilmiştir. Bu algoritmaya ait sözde kod Şekil 2.6'da gösterilmiştir.

Algoritma : Poisson Rassal Sayı Üretimi

$L = e^{-\lambda}$;
$x = 0,\ p = 1$;
do
    $x = x + 1$;
    [0,1] aralığında düzgün dağılıma sahip rastgele bir tamsayı (u) üret;
    $p = p \times u$;
while: $p \geq L$
return x - 1**;**

Şekil 2.6. Poisson dağılımına uygun rastgele sayı üretimi için sanal kod

Tez çalışmasının üçüncü bölümünde senkronizasyonu yapılacak verilerin saklanabileceği veri tabanlarından bahsedilecektir. Bölüm 3.3'te tez kapsamında kullanılan CouchDB veri tabanı hakkında detaylı bilgi verilecektir.

3. NoSQL KAVRAMI VE CouchDB

Veri tabanları en genel tanımıyla birbirleriyle ilişkili verilerin depolandığı ortamlardır. İlk zamanlarda dosya sistemleri olarak karşımıza çıkan veri tabanları zamanla bilgisayar ortamına taşınmıştır ve verileri saklamak, yönetmek ve bu verilere erişebilmek için veri tabanı yönetim sistemleri yazılımları geliştirilmiştir (Uzunbayır, 2015). International Business Machines (IBM) tarafından 1970'li yıllarda geliştirilen ilişkisel veri tabanı yönetim sistemleri son 40 yıldır verileri depolamak amacıyla etkin bir şekilde kullanılmaktadır (Eren, 2017; Taha, 2017). Ancak günümüzde internet teknolojisindeki gelişmelerin bir sonucu olarak yapılandırılmış, yarı yapılandırılmış ve yapılandırılmamış türdeki verilerin miktarı sürekli artmaktadır (Taha, 2017). İlişkisel veri tabanları bu verileri yönetmekte her zaman en iyi çözümü sunamamaktadır (Eren, 2017). İlişkisel veri tabanlarına alternatif olarak ortaya çıkmış NoSQL veri tabanları farklı türdeki verileri saklamak amacıyla 2009 yılından günümüze kadar yaygın olarak kullanılmaktadır ve bu veri tabanları homojen olmayan veriler üzerinde çalışabilmektedirler (Eren, 2017). Çalışmanın bundan sonraki bölümünde ilişkisel veri tabanları hakkında kısa bilgi verilecektir daha sonra NoSQL veri tabanı kavramı ve tez çalışmasında kullanılan NoSQL bir veri tabanı olan CouchDB veri tabanı yönetim sistemi detaylı olarak açıklanacaktır.

3.1. İlişkisel Veri Tabanı Yönetim Sistemleri

1970'lerde ortaya çıkan ilişkisel model ve 1980'lerde ortaya çıkan işlemsel model ilişkisel veri tabanı yönetim sistemlerinin temelini oluşturur. İlişkisel model veriler arasındaki ilişkilerden yararlanarak verinin bütününe erişilmesini sağlarken, işlemsel model gerçekleştirilen işlemlerin güvenli şekilde tamamlanmasını sağlamaktadır (Taha, 2017). İlişkisel veri tabanı yönetim sistemlerinde veriler tablolarda tutulurlar. Tablolar ise satırlar ve sütunlardan oluşur. Satırlar tablolarda yer alan her bir kayıta karşılık gelirken, tablolar dikey sütunların bir araya getirilmesiyle oluşturulmaktadır. İlişkiler ise farklı tablolardaki alanlar üzerinden tanımlanırlar (Gözüdeli, 2003).

İlişkisel model temelde ilişki ve özellik olarak ifade edilen iki kavram ile temsil edilir ve yapılandırılmış veriler üzerinde işlem yapar. Özellik veri tabanında yer alan bir nesnenin tipini belirlerken, ilişki ise nesnelerin arasındaki bağlantıyı tanımlar (Uzunbayır, 2015). 1970 yılında James Nicholas Gray tarafından veri tabanı işlemleri için bölünmezlik, tutarlılık, izolasyon ve süreklilik olarak isimlendirilmiş dört özellik seti tanımlanmıştır. Theo H¨arder ve Andreas Reuter 1983 yılında bu özellikleri bölünmezlik, tutarlılık, izolasyon, süreklilik (ACID) olarak isimlendirmiştir (H¨arder ve Reuter, 1983; Uzunbayır, 2015) . İlişkisel veri tabanlarında bir işlemin başarılı sayılabilmesi için bu dört özelliği karşılıyor olması gerekmektedir (Uzunbayır, 2015).

Burada bahsedilen bölünmezlik kavramı bir transaction işleminin ya hep ya hiç mantığı ile çalıştırılmasını temsil etmektedir (Uzunbayır, 2015). İşlemler sırasında gerçekleştirilmesi gereken adımlardan herhangi birinde sorun olursa o zamana kadar yapılan işlemlerde geçersiz sayılır ve başlangıç durumuna dönülür (Taha, 2017). Transaction'ın başarılı olabilmesi için tüm işlemlerin başarılı şekilde tamamlanması zorunludur.

İkinci özellik olan tutarlılık kavramı aynı veri tabanı üzerinde yapılan işlemlerden tutarlı sonuçların alınmasını tanımlamaktadır. Yani tutarlı bir sistem aynı girişlere karşılık aynı çıkışları vermelidir (Uzunbayır, 2015).

Üçüncü özellik olan izolasyon kavramı veri tabanı içerisinde işlem yapan transaction'ların birbirlerinden izole olmaları gerektiğini tanımlar. Birden fazla transaction aynı alan üzerinde eş zamanlı işlemler gerçekleştiriyorsa her biri yaptığı değişikliği kendi oturumu içerisinde yapmalıdır. Birinin işlemi bitene kadar ortak işlem yapılan alan diğerlerinden izole edilmelidir (Taha, 2017; Uzunbayır, 2015).

Son özellik olan süreklilik kavramı tamamlanmış olan bir transaction'ın oluşacak sistem hatalarından etkilenmemesi durumunu temsil eder. Bu durum oluşabilecek hatalara karşı tutulan log dosyaları aracılığıyla sağlanmaktadır. Böylece veri tabanı sürekli olarak bir geri yükleme noktasından en son mevcut durumunu koruyarak devam edebilecek durumda tutulur (Uzunbayır, 2015).

İlişkisel veri tabanı yönetim sistemleri, büyük veri kavramının ortaya çıkması ve yapılandırılmamış veriler ile işlem yapılması gerektiği durumların ortaya çıkmasıyla

hızla büyüyen bu verilere karşı etkili bir şekilde çalışmak ve bu verilere uyum sağlayabilmek konusunda zorluk yaşamaya başlamışlardır. Çoklu birleştirme işlemleri, iç içe geçmiş sorgular gibi operasyonlar ilişkisel veri tabanlarının performansını düşürmüştür (Uzunbayır, 2015). Böylece NoSQL veri tabanlarına ihtiyaç duyulmuştur.

3.2. NoSQL Kavramı ve NoSQL Veri Tabanları

NoSQL kavramı ilk kez Carlo Strozzi tarafından 1998 yılında ortaya atılmıştır. Unix benzeri bir işletim sistemi üzerinde çalışan ilişkisel bir veri tabanı yönetim sistemi geliştirmiştir ve geliştirdiği sistemin diğer Yapılandırılmış Sorgulama Dili (SQL) temelli ilişkisel veri tabanı sistemlerinden farklı olduğunu belirtmek amacıyla bu adı kullanmıştır (Dasgupta, 2018). 2009 yılında Eric Evans Not Only Sql anlamına gelen günümüzdeki NoSQL kavramını ortaya atmıştır. (Uzunbayır, 2015).

Moniruzzaman ve Hossain (2013) tarafından yapılan tanıma göre "NoSQL veri tabanları büyük boyutlu verileri saklamak ve çok sayıda sunucu üzerinde paralel veri işlemek amacıyla geliştirilmiş dağıtık, ilişkisel olmayan veri tabanlarıdır". NoSQL veri tabanları pek çok sunucu üzerinde veriyi yatay olarak ölçeklendirmek (yatay ölçeklendirme kavramı birden fazla düğümün işbirliği yapması kavramına karşılık gelmektedir.), eşlemek ve bölümleyebilmek için;

- İlişkisel veri tabanlarından daha basit ara yüzleri,
- İlişkisel veri tabanlarındaki ACID modelinden daha basit olan temel olarak erişilebilir, esnek durum, nihai tutarlılık (BASE) modelini,
- Etkin indeksleme ve rastgele erişimli bellek (RAM) kullanımını,
- Farklı kayıtlara uygun esnek şema modellerini desteklemektedir (Cattell, 2010).

Dağıtık sistemlerde veri tabanlarında verileri saklarken gerekli olan özellikleri tanımlamak amacıyla Eric Brewer tarafından tutarlılık, erişilebilirlik, bölümleme toleransı (CAP) (Brewer, 2000) teoremi olarak isimlendirilen bir teorem tanımlanmıştır. Bu teoreme göre dağıtık sistemlerdeki veri tabanlarının aynı anda tutarlılık, erişilebilirlik ve bölümleme toleransı özelliklerini sağlaması mümkün değildir. Buradaki tutarlılık kavramı dağıtık bir mimarideki veri tabanlarında veriye erişilmek istendiğinde tüm veri tabanlarında verinin en güncel halinin

bulundurulmasını tanımlar. Erişilebilirlik; dağıtık mimariye gelen tüm isteklerin mutlaka cevaplanması durumunu tanımlar. Son özellik olan bölümleme toleransı da dağıtık mimarideki düğümlerin iletişimde kopukluk olsa bile mimarinin çalışmaya devam etmesini tanımlar.

İlişkisel veri tabanlarındaki ACID kavramına karşılık olarak NoSQL veri tabanlarında da BASE olarak adlandırılan özellikler bulunmaktadır. ACID ve BASE kavramları veri saklama yaklaşımlarıdır ve CAP teoremi ile ilişkilidirler. CAP teoreminin gerekliliklerini karşılarlar. BASE yaklaşımı üç temel özelliğe sahiptir. Bunlar; temel olarak erişilebilirlik, esnek durum ve nihai tutarlılık olarak sıralanabilir. İlk özellik olan temel erişilebilirlik sisteme gelen her isteğin başarılı ya da başarısız olarak cevaplanmasını tanımlar, bu durum CAP teoremindeki erişilebilirlik özelliğini karşılamasını sağlar (Uzunbayır, 2015). İkinci özellik olan esnek durum sistemin durumu ve verinin sürekli olarak değişebilir olması anlamına gelir. Bazen bir veri tabanında değişiklik olmasa bile tutarlılığı sağlamak için veri ve sistemin durumu değişebilir (Uzunbayır, 2015). Son özellik olan nihai tutarlılık ise yeterli zaman olması durumunda dağıtık sistemin her düğümünde bulunan verinin tutarlı olacağı anlamına gelir. NoSQL kavramı ile CAP teoremi esnetilerek BASE kavramı ortaya çıkmıştır ve yatay ölçeklenebilirlik ile performans kazancı sağlanmıştır.

NoSQL veri tabanlarını sınıflandırmanın pek çok farklı yolu vardır. Ancak en temel şekilde sınıflandırıldığında beş ana kategoriye ayrılmaktadır. Bunlar aşağıdaki gibi sıralanabilir (Ulaş, 2018):

- Doküman tabanlı veri tabanları: Daha çok yarı yapılandırılmış verilerin depolanması için geliştirilmişlerdir. Dokümanları veri tabanları içerisinde saklayarak, kullanıcıların dokümanlar üzerinde işlem (doküman üzerinde ekleme, silme, güncelleme) yapmasına olanak verirler. Bu tür veri tabanlarında veriler doküman olarak isimlendirilen nesnelerde tutulurlar (Ulaş, 2018). Bu doküman nesneleri javascript nesne notasyonu (JSON), genişletilebilir işaretleme dili (XML), ikilik javascript nesne notasyonu (BSON) gibi farklı formatlarda olabilir (Uzunbayır, 2015). Dokümanların içerisinde kullanılan kodlama türüne uygun biçimde çok sayıda alan bulunabilir. Daha çok, elektronik ticaret siteleri ve içerik yönetim sistemleri gibi farklı yapıda verileri içeren uygulamalar için tercih edilir. (Ulaş, 2018). Doküman

tabanlı veri tabanlarına örnek olarak CouchDB (Anderson ve diğ., 2010), MongoDB (Cohodorow ve Dirolf, 2010) ve RavenDB (Ritchie, 2013) verilebilir.

- Anahtar-değer tabanlı veri tabanları: Temel veri modeli olarak anahtar-değer dizilerini kullanır. Veriler, anahtar-değer çifti topluluğu olarak temsil edilir, bu nedenle her anahtar tekildir (Uzunbayır, 2015). Veriler anahtar ve hash tabloları aracılığıyla saklandığı (Taha, 2017) için okuma ve yazma işlemlerini hızlı bir şekilde gerçekleştirirler. Bu nedenle okuma ve yazma işlemlerinin sık gerçekleştirildiği uygulamalarda kullanılırlar. (Ulaş, 2018). Anahtar-değer tabanlı veri tabanlarına örnek olarak Oracle NoSQL veri tabanı (Alam ve diğ., 2014) ve Redis (Macedo ve Oliveira, 2011) verilebilir.
- Çizge tabanlı veri tabanları: Bu veri tabanları, aralarındaki ilişkiler grafik olarak iyi temsil edilebilen veriler için tasarlanmıştır. Burada varlıkları temsil etmek için düğümler, ilişkileri temsil etmek için kenarlar, düğümler ve kenarlar arasında bilgi vermek için özellikler kullanılır (Uzunbayır, 2015; Ulaş, 2018). Bu tür veri tabanlarına uygun veriler sosyal ağ verileri, yol haritaları, ağ topolojileri içeren veriler olabilir. Bu tür veri tabanlarına örnek olarak Neo4j (Bruggen, 2014) ve OrientDB (Tesoriero, 2013) verilebilir.
- Kolon tabanlı veri tabanları: Her bir veri saklama bloğu bir kolona ait verileri içermektedir. Veriler bu şekilde sakladığında güncellemeler sırasında aynı kolonda yer alan verilere toplu olarak erişim sağlanır. Yüksek okuma yazma performansı ve yüksek erişilebilirlik için tasarlanmışlardır. Bu tür veri tabanlarına örnek olarak HBase (George, 2011) ve Cassandra (Carpenter ve Hewitt, 2016) verilebilir.

Tez çalışması kapsamında oluşturulan dağıtık ağ mimarisi eğitim dokümanlarının saklanması ve dokümanlar üzerinde güncellemelerin yapılması amacıyla oluşturulmuştur. Dağıtık mimari multi-master senkronizasyonuna ihtiyaç duyan yapılandırılmamış doküman tabanlı veriler içermektedir. CouchDB doküman tabanlı veri tabanları arasında multi-master mimarisinde senkronizasyonu etkin bir şekilde gerçekleştirmektedir. Bu nedenle çalışma kapsamında CouchDB veri tabanı kullanılmıştır. Bir sonraki bölümde CouchDB veri tabanının verileri saklama mimarisi ve senkronizasyon işlemini nasıl gerçekleştirdiği konusunda detaylı bilgi verilmiştir.

3.3. CouchDB

CouchDB doküman tabanlı kullanımı kolay ve ölçeklenebilir bir mimariye sahip NoSQL bir veri tabanıdır. İlk olarak 2005 yılında Damien Katz tarafından yayınlanmıştır ve 2008 yılında Apache Software Foundation kurumunun projesi olarak geliştirilmeye devam edilmiştir. Earlang dili kullanılarak yazılmıştır. CouchDB' de veriler JSON formatında depolanır ve Uygulama Programlama Arayüzü (API) olarak RESTful HTTP'yi kullanır. Sorgular MapReduce tekniği ile Javascript kullanılarak gerçekleştirilir (Anderson ve diğ., 2010; Eren, 2017).

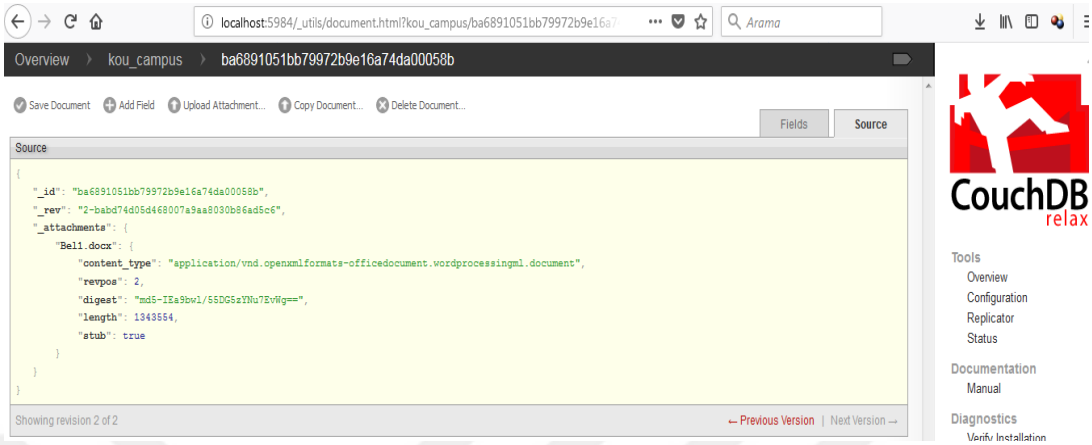
CouchDB verileri dokümanlar olarak saklar ve içerisinde yer alan her doküman birbirinden bağımsızdır. Veri tabanları içerisinde pek çok doküman barındırmaktadır. Dokümanlar CouchDB'nin ana birimidir. Her doküman içerdiği veriye göre ihtiyaç duyulduğu kadar alan ve eklerden oluşmaktadır. Doküman içerisindeki her alan benzersiz olarak isimlendirilir ve herhangi bir alan sınırlanması olmadan farklı tipteki verileri saklayabilirler. Bu nedenle her doküman kendi verilerine özgü bir şemaya sahiptir (Deka ve Bakshi, 2014). Dokümanlar ayrıca, veri tabanı sistemi tarafından korunan meta verileri içerir. Aşağıda Tablo 3.1'de (Anderson ve diğ., 2010) JSON veri yapısına uygun olarak bir CouchDB veri tabanı içerisinde saklanan dokümanın sahip olabileceği alanlarının en genel şeması gösterilmiştir.

Tablo 3.1. CouchDB veri tabanlarında saklanan dokümanların alanları

Alan Adı	**Tanımlama**
_id	Dokümanın id'sini tanımlar.
_rev	Revizyon numarası
_attachments	Dokümanın ekleri
filename	Eklere ait dosya adı bilgisi
content_type	MIME türünde dosyanın uzantısı
Data	Dosya içeriğinin Base64'e göre kodlanmış içeriği

Şekil 3.1'de kou_campus veri tabanı içerisindeki bir dokümanın JSON yapısı gösterilmiştir. Şekilde görülen ekran CouchDB Futon ara yüzüdür. Burada bulunan _id alanı kou_campus veri tabanında bulunan dokümanın benzersiz bir id'si olmasını sağlamaktadır. Bu alan kullanıcı tarafından belirlenebilir, belirlenmemesi durumunda CouchDB tarafından otomatik olarak şekilde de görüldüğü gibi tekil bir id ataması yapılır. İkinci alan olan _rev numarası dokümanın kaç kez güncellendiğini gösteren bir numaradır (Anderson ve diğ., 2010). Böyle bir alana ihtiyaç duyulmasının sebebi

CouchDB'nin çok sürümlü eşzamanlılık kontrolü (MVCC) tabanlı bir modele sahip olmasıdır.



Şekil 3.1. Örnek bir CouchDB veri tabanı içerisindeki dokümana ait ekran görüntüsü

MVCC'de çakışmaları önlemek amacıyla yazma işlemleri sırasında dokümanların kilitlenmesi durumu gerçekleşmez. Bunun yerine dokümanlar versiyonlar ile saklanırlar. Eğer bir doküman üzerinde değişiklik yapılırsa dokümanın eski ve yeni olarak iki versiyonu oluşturulur ve her iki versiyonda saklanır. Okuma işlemini gerçekleştiren kullanıcılar da her zaman okuma isteği yapıldığı anda veri tabanında bulunan en güncel versiyona erişmiş olurlar ancak okuma sırasında yapılan değişikleri göremezler (Brown, 2012).

CouchDB'de veri tabanlarında meydana gelen değişikliklerin takibi için CouchDB changes API kullanılır. Bu API aracılığıyla bir veri tabanında yer alan doküman üzerinde yapılan değişikliklerin sıralanmış bir listesi elde edilebilir. _change bir JSON objesidir ve id, sequence ve changes olmak üzere üç alandan oluşur. Id alanı daha öncede bahsedildiği gibi dokümanın id'sini, sequence alanı yapılan güncelleme işlemine ait sıra numarasını ve changes alanı ise dokümanın revizyon numarasını ve doküman hakkında diğer bilgileri (silindi vb.) içeren bir listedir. Changes API aracılığıyla bir CouchDB veri tabanında yapılan değişiklikler API'de yer alan son sequence değerinden itibaren listelenebilir (Anderson ve diğ., 2010; Kavak ve diğ., 2014). CouchDB'de senkronizasyon işlemi temelde beş adımda gerçekleştirilmektedir (URL-1, 2018). Bunlar aşağıda listelenmiştir;

1. Kaynak veri tabanı ve hedef veri tabanından senkronizasyonu yapılacak veri tabanlarına ait tekil id'ler alınır.

2. Bu tekil id'ler hedef veri tabanında özel bir doküman içerisine kayıt edilir. Bu işlem ile son senkronizasyona ait kaynak sıra numarası ve kontrol noktası elde edilir.
3. Son kontrol noktasına göre Changes API tarafından kaynaktaki geçerli revizyonların bir listesi döndürülür.
4. Toplanan dokuman\revizyon listeleri kaynak veri tabanından alınır. Bu sonuç kaynakta olup hedefte olmayan revizyonların listesini içerecektir.
5. Revizyonlar hedef veri tabanında gerçekleştirildikten sonra kaynak veri tabanında bir sonraki senkronizasyon için yeni kontrol noktası belirlenir.

CouchDB'de senkronizasyon işlemleri tek yönlü olabildiği gibi çift yönlü olarak da gerçekleştirilebilmektedir. Burada kullanılabilecek olan üç farklı güncelleme bildirim yöntemi bulunmaktadır (Anderson ve diğ., 2010). Bunlar;

- Continuous,
- Polling ve
- Long polling olarak sıralanabilir.

Yukarıda bahsedilen her yöntemin kullanımının uygun olduğu senaryolar vardır. Continuous yöntemi isminden de anlaşılabileceği gibi sürekli olarak açık tutulan bir bağlantı üzerinden çalışmaktadır. Açılan bu bağlantı CouchB'nin değişiklik API si olan CouchDB's Changes API'sini sürekli olarak dinlemektedir ve eşlemesi sağlanacak olan veri tabanlarından herhangi birinde değişiklik meydana geldiğinde otomatik olarak eşleme işlemini başlatmaktadır. Yani bu yöntem ile işlem yapıldığında gerçek zamana yakın olarak eşleme sağlanmaktadır (Anderson ve diğ., 2010). Bu işlem canlı eşleme olarak da adlandırılmaktadır ve push mantığı ile çalışmaktadır. Yani değişiklik meydana geldiğinde verinin doğrudan karşı tarafa iletilmesi mantığı ile çalışır. Veri güncellemelerindeki gecikmelerin tolere edilemeyeceği uygulamalar için uygundur. Sunucular üzerinde çok sayıda açık bağlantı olacağından bunların eş zamanlı olarak yönetimini gerektirir.

İkinci güncelleme bildirim mekanizması olan polling yöntemi değişiklik olup olmadığına dair belirli periyotlarla sunucuların tetiklenmesini gerektirir. CouchDB'de varsayılan olarak polling seçeneği seçilidir. Belirlenen kontrol periyodu düşük olduğunda değişikliklerin gerçek zamana yakın olarak senkronizasyonu sağlanabilir ancak sunucuların yoğun olduğu durumlarda çok sık bağlantı açma

kapatma durumu sunucuya ek yük getirmektedir. Polling yöntemi üç yöntem arasında en kolay uygulanan yöntemdir (Anderson ve diğ., 2010).

Üçüncü güncelleme bildirim mekanizması olan long polling, polling işleminin farklı bir versiyonudur. Sunucuya poll işlemi için talepte bulunduğunda sunucu cevabı göndermeden önce başka bir değişiklik olacak mı diye belirli bir süre bekler. Böylece hiçbir şey değişmeden yapılacak gereksiz poll işlemlerinden kaçınmayı sağlar. Polling işleminde belirli aralıklarla sürekli poll yapılırken long polling bu süreyi biraz daha uzun tutar ve değişiklikleri toplu göndermeyi hedefler. Böylece hiçbir değişiklik olmadan yapılacak poll işlemlerinin önüne geçmeyi hedefler (Anderson ve diğ., 2010). Burada da eş zamanlı açılan bağlantıların yönetimini sağlamak gerekmektedir.

Doküman tabanlı veri tabanları arasında multi master senkronizasyonunu destekleyen birkaç veri tabanı bulunmaktadır (Anderson ve diğ., 2010; Kavak ve diğ., 2014; Tesoriero, 2013). CouchDB'de bunlardan en yaygın olarak kullanılanıdır (Grolinger ve diğ., 2013) ve bu işlemi çok basit olarak gerçekleştirir (Kavak ve diğ., 2014). Tez çalışmasında kullanılacak mimari de multi master senkronizasyonu gerektiren bir mimaridir. Senkronizasyonu yapılacak veriler de dokümanlardır. Bu nedenlerle tez çalışması kapsamında verileri saklamak için CouchDB veri tabanı tercih edilmiştir. CouchDB veri tabanında yer alan mevcut üç senkronizasyon tetikleme yöntemine alternatif olacak farklı bir poll yöntemi önerilmektedir. Bu yöntem akıllı bir poll yöntemi olacaktır. Yöntem aracılığıyla ağın durumuna göre poll süresinin periyodu değiştirilecektir. Böylece ağın boş olduğu durumlarda poll yöntemi etkin şekilde kullanılacaktır. Yoğunluk olan durumlarda kontrol süresi arttırılacağından sunucuya poll işlemi yüzünden ek yük gelmesi önlenecektir. Continuous ve long polling yöntemlerinde olduğu gibi sürekli açık bağlantılar olmayacağından bağlantı yönetimi için de çaba harcanmayacaktır. Trafiğin ya da ana sunucuya paket gönderiminin yoğun olduğu zamanlarda yerel sunucular eşleme için zaman harcamayacaktır. Böylece iç ağda cevap verilecek kullanıcı sayısı arttırılacaktır. Yoğunluk nedeniyle oluşacak gecikmeler ve yeniden iletimlerin önüne geçilecektir.

4. VERİ ÖN İŞLEME VE KÜMELEME YÖNTEMLERİ

Veri madenciliği en basit tanımıyla büyük miktardaki veriden faydalı bilginin elde edilmesi işlemidir. Veri madenciliği süreci yedi temel adım ile tanımlanmıştır. Bunlar veri temizleme, veri bütünleştirme, veri indirgeme, veri dönüştürme, veri madenciliği yönteminin uygulanması, örüntünün elde edilmesi ve sunum olarak sıralanabilir (Han ve diğ., 2012). Burada bahsedilen ilk dört adım veri ön işleme adımlarını oluşturmaktadır. Sonraki adım olan veri madenciliği yönteminin uygulanması aşaması için geliştirilmiş pek çok yöntem bulunmaktadır. Yöntemler gerçekleştirdikleri işlemlere göre kategorilere ayrılmaktadır. Kümeleme yöntemleri de bu kategorilerden bir tanesidir. Bu bölümde tez çalışması kapsamında uygulanan veri ön işleme yöntemleri ve kümeleme yöntemlerinden bahsedilecektir.

4.1. Veri Ön İşleme

Veri temizleme, veri bütünleştirme, veri indirgeme ve veri dönüştürme işlemlerinin tamamı veri ön işleme olarak adlandırılmaktadır. Bu işlemlerin açıklamaları (Özkan, 2008) aşağıda verilmiştir.

- Veri temizleme; verideki kayıp, hatalı ya da tutarsız değerlerin çeşitli yöntemlerle temizlenmesi işlemidir.
- Veri bütünleştirme; farklı kaynaklardan veriler alınarak kullanılacaksa, bunların birbirleriyle uyumlu olacak şekilde türlerinin dönüştürülmesi işlemidir.
- Veri indirgeme; çok sayıda özellik ya da çok büyük boyutlarda veri içeren veri setlerinde sonucu değiştirmeyecek ya da büyük oranda etkilemeyecek olanların belirlenmesi ve özellik\veri sayısının azaltılması işlemidir.
- Veri dönüştürme; veri içerisinde farklı tipte ya da farklı değer aralıklarında değerler alan özellikler olduğunda bazı özelliklerin sonuca etkisi daha fazla olacaktır. Bu durumu önlemek amacıyla yapılan işlemlere veri dönüştürme denir. Aynı zamanda özelliklerde yer alan çok uç değerlerin etkisinin azaltılması içinde uygulanır.

Tez çalışması kapsamında veri temizleme ve veri dönüştürme işlemleri uygulanmıştır. Verideki kayıp değerlerin elde edilmesi için K-En Yakın Komşu (k-NN) algoritması ve verilerin normalleştirilmesi için Min-Max Normalleştirmesi kullanılmıştır. Alt bölümlerde kullanılan yöntemlerin detaylı açıklamaları yapılmıştır.

4.1.1. K-En Yakın Komşu yöntemi

k-NN algoritması (Fix ve Hodges, 1951) yaygın olarak kullanılan sınıflandırma algoritmalarından bir tanesidir. Yöntem sınıflandırma haricinde kayıp verilerin elde edilmesi için de kullanılmaktadır. Tez çalışması kapsamında hem sınıflandırma hem de kayıp değerleri elde etmek amacıyla kullanılmıştır. Bu nedenle her iki kullanım şekli de aşağıda açıklanmıştır.

Algoritmanın amacı verileri, kendilerine yakın veri kümelerini kullanarak sınıflandırmak ya da veri içerisindeki kayıp değerleri elde etmektir. Burada yer alan k değeri komşu sayısını ifade etmektedir. k değerinin probleme uygun olarak belirlenmesi gerekmektedir.

Sınıflandırmadaki kullanımında algoritma, ilk olarak sınıflandırılacak örneğin öğrenme kümesinde yer alan örneklere olan uzaklığının hesaplanması ile başlar. Bu aşamada herhangi bir uzaklık bağıntısı kullanılabilir. Tez çalışması kapsamında Öklid uzaklığı kullanıldığı için yalnızca Öklid uzaklığına ait formül Eşitlik 4.1'de (Kantardzic, 2011; Özkan, 2008) gösterilmiştir. Eşitlik (4.1)'de i ve j noktaları arasındaki Öklid uzaklığı hesaplanmıştır. Burada her x noktasının p adet özelliğe sahip olduğu kabul edilmiştir. x_{iu} değeri x_i noktasına ait u. özelliği ve x_{ju} değeri x_j noktasına ait u. özelliği temsil etmektedir.

$$d(i, j) = \sqrt{\sum_{u=1}^{p} (x_{iu} - x_{ju})^2} \qquad (4.1)$$

Şekil 4.1'de sınıflandırma için k-NN yönteminin kullanımına ait sözde kod gösterilmiştir. Veri setinin $X=\{x_1, x_2, x_3, \ldots, x_n\}$ şeklinde tanımlandığını kabul edelim, x_f değeri sınıflandırılacak yeni örneği temsil etsin. x_f örneğinin, öğrenme kümesindeki tüm örneklere olan uzaklığı hesaplandıktan sonra, x_f örneğine en yakın olan k örnek elde edilir. Daha sonra k örnek en fazla hangi sınıfa ait veri içeriyorsa, x_f örneğinin de o sınıfa ait olduğu kabul edilir.

Algoritma 1: K-En Yakın Komşu Algoritması (Sınıflandırma)

Komşu sayısının belirlenmesi k;
Eğitim verisi: $X=\{x_1, x_2, x_3, \ldots, x_n\}$
Y: X verisine ait sınıf etiketleri;
x_f : Sınıf etiketi bilinmeyen örnek;
for i = 1 to n do
  Eşitlik (4.1)'e göre $d(x_i, x_f)$ değerini hesapla ve G kümesine ata;
end
G kümesindeki değerleri küçükten büyüğe sırala;
En üstteki k adet veri içerisinde en çok geçen sınıf etiketini (Y_i) x_f' ye ata.

Şekil 4.1. Sınıflandırma için K-En Yakın Komşu algoritmasına ait sözde kod

Yöntem kayıp verilerin elde edilmesi için kullanıldığında ilk olarak veri setindeki veriler kayıp değer içerenler ve kayıp değer içermeyenler olarak ikiye ayrılır. Daha sonra kayıp değer içeren bir örnek için kayıp değer içermeyen tüm verilere olan uzaklıklar hesaplanır. Elde edilen uzaklıklardan en küçük olan k tanesi seçilir. Seçilen k örneğin ortalaması alınır ve kayıp özelliğin değeri elde edilir. Şekil 4.2'de bu işleme ait sözde kod gösterilmiştir.

Algoritma 2: K-En Yakın Komşu Algoritması (Kayıp Veri)

Komşu sayısının belirlenmesi k;
Eğitim verisi: $X=\{x_1, x_2, x_3, \ldots, x_n\}$
Kayıp değer içeren veri kümesi: $X'=\{x'_1, x'_2, x'_3, \ldots, x'_g\}$;
Kayıp değer içermeyen veri kümesi: $X''=\{x''_1, x''_2, x''_3, \ldots, x''_{n-g}\}$, $X'UX'' = X$;
for i = 1 to g do
  for j = 1 to n-g do
    Eşitlik (4.1)'e göre $d(x'_i, x''_j)$ değerini hesapla ve G kümesine ata;
  end
  G kümesindeki değerleri küçükten büyüğe sırala;
  En üstteki k adet verinin ortalamasını al, x'_i'deki kayıp değer/değerlerin yerine ata;
End

Şekil 4.2. Kayıp veriler için K-En Yakın Komşu algoritmasına ait sözde kod

4.1.2. Min-Max Normalleştirmesi

Bu yöntemde isminden de anlaşılabileceği gibi ilk olarak veri içerisindeki ilgili özelliğe ait en büyük ve en küçük değerlerin belirlenmesi gerekir. Daha sonra bu değerler arasındaki fark alınarak diğer değerler bu değere göre normalleştirilir. Bu işlemin nasıl yapılacağı Eşitlik (4.2)'de (Özkan, 2008) gösterilmiştir. Buradaki amaç verileri 0-1 aralığındaki değerlere dönüştürmektir.

$$X_{normal} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (4.2)$$

Eşitlikte yer alan X_{normal} değeri normalleştirilmiş verileri, X_{min} değeri ilgili özelliğe ait en küçük değeri, X_{max} değeri ilgili özelliğe ait en büyük değeri temsil etmektedir. Bu bağlantı normalleştirilecek X değerlerine uygulanarak veriler 0-1 aralığına dönüştürülür.

4.2. Veri Madenciliğinde Kümeleme

Kümeleme işlemi verinin karakteristiğine bağlı olarak verinin benzer bölümlerinin gruplanmasıdır (Özkan, 2008). Bu amaçla geliştirilmiş pek çok farklı yöntem bulunmaktadır. Tez çalışması kapsamında K-Ortalamalar ve Bulanık C-Ortalamalar yöntemleri kullanıldığı için bu bölümde bu iki kümeleme yönteminin detaylı açıklaması yapılmıştır.

4.2.1. K-Ortalamalar yöntemi

K-ortalamalar (MacQueen, 1967) kümeleme algoritması, verilerin kümelenmesi problemini çözen en basit denetimsiz öğrenme algoritmalarından bir tanesidir (MacQueen, 1967). Yöntem basit olarak verilen bir veri setini k adet kümeye parçalamayı amaçlar.

Algoritmada ilk olarak k adet kümenin merkezinin belirlenmesi gerekmektedir. Küme merkezlerinin belirlenmesi farklı başlangıçlarda farklı sonuçlara sebep olacağından önemli bir konudur. En iyi belirleme şekli küme merkezlerinin birbirinden olabildiğince uzak olacak şekilde belirlenmesidir. Küme merkezleri belirlendikten sonra veri setinde yer alan her bir noktanın küme merkezlerine olan uzaklıklarına göre bir kümeye ataması gerçekleştirilir. Bu aşamada herhangi bir uzaklık bağıntısı kullanılabilir. Tez çalışması kapsamında Eşitlik (4.1)'de gösterilen Öklid bağıntısı kullanılmıştır.

Eşitlik (4.1) kullanılarak $X=\{x_1, x_2, x_3, \ldots, x_n\}$ olarak tanımlanan veri setinde yer alan n adet veri için en yakın oldukları kümelere atamalar yapılır. Böylece veri seti $\{C_1,C_2,\ldots,C_k\}$ şeklinde k adet kümeye ayrılmış olur ve algoritmanın ilk adımı tamamlanır. Sonraki aşamada oluşan kümelere göre yeni küme merkezlerinin

hesaplanması gerekir. Bu işlem her kümede yer alan elemanların ortalaması alınarak Eşitlik (4.3)'e göre gerçekleştirilir.

$$z_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_{ij} \qquad \forall j = 1, 2, ...k, \tag{4.3}$$

Eşitlik (4.3)' de yer alan z_j değeri j. kümede yer alan elemanların ortalama vektörünü ve aynı zamanda yeni küme merkezini temsil etmektedir. n_j değeri j. kümede yer alan eleman sayısını ve x_{ij} değeri de j. kümede yer alan i. elemanı göstermektedir (Özkan, 2008). Yeni küme merkezleri hesaplandıktan sonra verilerin yeniden kümelere ataması gerçekleştirilir ve kümelerde herhangi bir değişiklik meydana gelmeyinceye kadar bu işlemlere devam edilir. K-Ortalamalar yönteminde her iterasyonda Eşitlik (4.4) ile gösterilen kümeler için hata değerinin azalması beklenir (Özkan, 2008). Yöntemin amacı bu amaç fonksiyonunu minimize etmektir. K-Ortalamalar algoritmasına ait sözde kod Şekil 4.3'te gösterilmiştir.

Algoritma 3: K-Ortalamalar

Küme sayısının belirlenmesi k;
Küme merkezlerinin atanması:
  $Z=\{z_1, z_2, \ldots z_k\}$;
Repeat
  for j = 1 to n do
    for i = 1 to k do
      Eşitlik (4.1)'e göre d_{ij} değerinin hesaplanması
    end
    Verinin en yakın olduğu kümeye atanması
  end
  Eşitlik (4.3)'e göre yeni küme merkezlerinin hesaplanması
Until (Küme merkezlerinde değişiklik olmayana kadar **)**

Şekil 4.3. K-Ortalamalar algoritmasına ait sözde kod

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n_j} (x_{ij} - z_j)^2 \tag{4.4}$$

Eşitlik (4.4)'de iç toplam ile küme içi değişmeler elde edilmektedir. Daha sonra her küme için elde edilen küme içi değişmeler toplanarak Karesel Hatalar Toplamı (SSE) değeri elde edilmektedir. Eşitlikte bu değer J sembolü kullanılarak gösterilmiştir. Küme içi değişmeler de ilgili kümedeki elemanların her birinin küme merkezine olan uzaklığının karesi alınarak toplanmasıyla elde edilmektedir.

4.2.2. Bulanık C-Ortalamalar yöntemi

FCM algoritması (Dunn, 1974) kümeleme işlemini bulanık olarak gerçekleştiren ve yaygın olarak kullanılan kümeleme algoritmalarından bir tanesidir (Türe ve Başer, 2015). Yöntem verilerin iki ya da daha fazla kümeye ait olabilmesine imkân vermektedir. Kullanmış olduğu bulanık mantık sebebiyle veriler kümelere [0,1] aralığında değişen bir aitlik değeri ile atanabilirler. Bir veri noktasının tüm kümelere olan aitlik değerlerinin toplamı 1 olmalıdır. Veri noktasının bulunduğu kümenin üyelik değeri, diğer kümelerin üyelik değerinden daha büyük olacaktır. Veri noktalarının kümelere aitlik değerleri ve bulanık küme merkezleri sırasıyla Eşitlik (4.5) ve Eşitlik (4.6)'da gösterildiği gibi hesaplanabilir.

$$\mu_{ij} = 1 / \sum_{u=1}^{c} \left(\frac{\left\| x_i - z_j \right\|}{\left\| x_i - z_u \right\|} \right)^{(2/m-1)}, \qquad \mu_{ij} \in [0,1] \tag{4.5}$$

$$z_j = (\sum_{i=1}^{n} (\mu_{ij})^m x_i) / (\sum_{i=1}^{n} (\mu_{ij})^m), \qquad \forall j = 1,2,...c, \quad m \in [1,\infty] \tag{4.6}$$

Eşitlik (4.6)'da yer alan n değeri veri setindeki veri noktalarını temsil etmektedir. Eşitlik (4.5)'de yer alan z_j değeri j. kümenin merkezini, m değeri bulanıklık indeksini, c değeri oluşturulacak küme sayısını ve μ_{ij} değeri de i. verinin j. kümeye aitlik değerini göstermektedir. FCM algoritmasının ana hedefi Eşitlik (4.7)'de gösterilen amaç fonksiyonunu minimize etmektir.

$$J(U, Z) = \sum_{i=1}^{n} \sum_{j=1}^{c} (\mu_{ij})^m \left\| x_i - z_j \right\|^2 \tag{4.7}$$

Eşitlik (4.7)' deki , $||x_i - z_j||$ değeri i. veri noktası ve j. küme merkezi arasındaki Öklid Uzaklığını göstermektedir. Veri kümesi X değişkeni kullanılarak $X=\{x_1, x_2, x_3, \ldots, x_n\}$ şeklinde temsil edilebilir. X veri kümesi U kümeleme matrisi ile c sayıda kümeye parçalanacaktır. U matrisi her bir kümede yer alan verilerin üyelik derecelerinden oluşan bir matristir. Kümeleme işlemi sonrası elde edilecek U matrisi Eşitlik (4.8)' deki gibi gösterilebilir. Yöntemin amacı $||U^{(r+1)} - U^{(r)}||$ değerini, belirli bir tekrarlama sonucunda daha önce belirlenen bir hata değerinin (ε) altına çekmektir.

$$U = \begin{bmatrix} \mu_{11} & \mu_{21} & \cdots & \mu_{c1} \\ \mu_{12} & \mu_{22} & \cdots & \mu_{c2} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{1n} & \mu_{2n} & \cdots & \mu_{cn} \end{bmatrix} \tag{4.8}$$

FCM algoritmasına ait sözde kod Şekil 4.4'te verilmiştir. Burada yer alan ε değeri, 0 ve 1 aralığında değişen bir hata değeridir ve r değeri de tekrarlama sayısını göstermektedir. Yöntem ε koşulu sağlanana kadar ya da J(U,Z) değeri minimize edilene kadar U matrisini ve küme merkezlerini hesaplamaya devam eder.

Algoritma 4: Bulanık C-Ortalamalar

Küme sayısının belirlenmesi *c*;
İterasyon sayısının tanımlanması r=0;
Küme merkezlerinin rastgele olarak atanması:
  $Z=\{z_1, z_2, \ldots z_c\}$;
Repeat
  for j = 1 to c do
    for i = 1 to n do
      /*Veri noktalarına ait üyelik değerlerinin hesaplanması*/
      Eşitlik (4.5)'e göre μ_{ij} değerlerinin hesaplanması
    end
  end
  Mevcut küme merkezlerinin saklanması;
  for j= 1 to c do
    Eşitlik (4.6)'ya göre yeni küme merkezlerinin elde edilmesi.
  end
Until ($\|U^{(r+1)} - U^{(r)}\| < \varepsilon$ ya da min. J değeri elde edildi mi? **)**

Şekil 4.4. Bulanık C-Ortalamalar algoritmasına ait sözde kod

4.3. Veri Ön İşleme ve Kümeleme Yöntemlerinin Değerlendirilmesi

Tez çalışması kapsamında gerçekleştirilen kümeleme işlemlerinin değerlendirilmesi amacıyla dört farklı ölçüt kullanılmıştır. Bunlar küme merkezlerinin benzerliklerinin toplamı, ortalama küme içi benzerliklerinin toplamı, karesel hatalar toplamı ve işlem süresi olarak belirlenmiştir. Bu bölümde ölçütlerin açıklamaları verilmiştir. Kullanılan ilk iki ölçüte ait formüller sırasıyla Eşitlik (4.9), Eşitlik (4.10)'da gösterilmiştir. Karesel hatalar toplamı daha önce Eşitlik (4.4)'te açıklandığı için bu bölümde açıklanmamıştır.

$$\text{Merkez benzerlikleri toplamı} = \sum_{i=1}^{c-1} \sum_{j=i+1}^{c} d(z_i, z_j) \tag{4.9}$$

$$\text{Küme içi benzerlikler toplamı} = \sum_{u=1}^{c} n_u \left(\frac{1}{{n_u}^2} \sum_{x_i, x_j \in C_u} d(x_i, x_j) \right) \tag{4.10}$$

Eşitliklerde yer alan c değeri küme sayısını, z_i değeri i. kümenin merkezini $d(z_i, z_j)$ değeri i. küme merkezinin j. küme merkezine olan uzaklığını, n_u değeri u. kümenin eleman sayısını ve $d(x_i, x_j)$ değeri C_u kümesinde yer alan i. elemanın, j. elemana olan uzaklığını temsil etmektedir.

Kümele işleminin amacı küme merkezlerinin benzerliklerinin oldukça düşük olmasını, küme içi benzerliklerin ise oldukça yüksek olmasını sağlamaktır. Bu nedenle kullanılan yöntemlerin karşılaştırılması amacıyla küme merkezlerinin benzerliklerinin toplamı ve ortalama küme içi benzerliklerin toplamı yöntemleri tercih edilmiştir. Bunlara ek olarak yöntemlerin işlem süreleri de hesaplanmıştır.

Verilerde bulunan kayıp değerlerin elde edilmesi için kullanılan yöntemlerin karşılaştırılması için Eşitlik (4.11)'de gösterilen ortalama karesel hata (MSE) yöntemi kullanılmıştır. Eşitlikte yer alan $\overline{x_i}$ değeri ilgili özelliğe ait kayıp değer içermeyen verilerin ortalamasını, x_i' değeri kayıp verinin, uygulanan yöntem ile tahmin edilen değerini ve g değeri de veri seti içerisindeki kayıp değer sayısını temsil etmektedir.

$$\text{MSE} = \frac{1}{g} \sum_{i=1}^{g} d(\overline{x_i}, x_i')^2 \tag{4.11}$$

5. MARKOV MODELLERİ

Veri madenciliğinde zaman içerisinde meydana gelen sıralı gözlemleri temsil etmek için markov modelleri kullanılır (Ramage, 2018). Markov modelleri, modeli oluşturan veri kümesindeki veri dizilerinin, birbirinden bağımsız olmadığını ve olasılıksal bir süreç tarafından meydana getirildiğini kabul eder (Alpaydın, 2013). Bir markov zincirinde o anki durumu bildiğimizde, sonraki durumların daha önceki durumlardan bağımsız olduğu ancak o anki durumun, sonraki durumları belirleyecek bütün bilgiye sahip olduğu kabul edilir (Koyun ve Kaymakçı, 2015).

Bu bölümün devamında olasılıksal bir süreç tarafından oluşturulan dizgileri modellemek için kullanılan Kesikli Markov Modeli ve Markov Modeli'nin farklı bir uzantısı olan Saklı Markov Modeli'nden bahsedilecektir.

5.1. Kesikli Markov Modeli

Kesikli Markov Modeli markov özelliğine sahip bir süreci temsil etmektedir. Buradaki sistem t olarak ifade edilen her bir anda, belirli bir olasılık dağılımına bağlı olarak S sembolü ile ifade edilen durumlar kümesindeki bir durumdan başka bir duruma geçiş yapar. Modeli kurulacak problemde N farklı durum olduğu kabul edilirse durum kümesi S_1, S_2, S_3, … , S_N olarak temsil edilir. t değişkeni problemin türüne göre bir zamanı, bir sıra indisini vb. temsil edebilir ve t=1,2,3, şeklinde değerler alır. Burada herhangi bir t anında bulunulan durum qt değişkeni kullanılarak temsil edilebilir. Yine herhangi bir t anında S_i durumunda bulunulma olasılığı ise $P(q_t = S_i)$ ile gösterilir. Bu tanımlamalar doğrultusunda her t anında durumlar arasındaki geçiş olasılığı Eşitlik (5.1) ve Eşitlik (5.2)'de gösterildiği gibi ifade edilecektir. Bu alt bölüm (Alpaydın, 2013)'den derlenmiştir.

$$P(q_{t+1} = S_j \mid q_t = S_i, q_{t-1} = S_k, ...) \tag{5.1}$$

$$P(q_{t+1} = S_j \mid q_t = S_i, q_{t-1} = S_k, ...) = P(q_{t+1} = S_j \mid q_t = S_i) \tag{5.2}$$

Kesikli Markov Modeli birinci dereceden bir Markov modelidir. Bu modelde bir sonraki durum sadece bir önceki duruma bağlıdır. Bu durum Eşitlik (5.2)' de gösterildiği gibi ifade edilebilmektedir. Formülde verilen olasılık değeri bir durumdan başka duruma geçmek için hesaplanan olasılık değerini ifade etmektedir ve bu değer geçiş olasılığı olarak adlandırılır. Geçiş olasılığı a_{ij} sembolü ile gösterilir. Birinci dereceden bir markov modelinde t anında S_i durumunda bulunan bir dizinin t+1 anında S_j durumunda olma olasılığı Eşitlik (5.3)' de gösterilmiştir.

$$a_{ij} = P(q_{t+1} = S_j \mid q_t = S_i) \tag{5.3}$$

Geçiş olasılığı bir olasılık değerini ifade etmektedir ve bu nedenle negatif olamaz. Aynı zamanda bir durumdan yapılabilecek tüm geçişlerin olasılıklarının toplamının 1 olması gereklidir. Bu durumda bir olaydan diğer olaylara geçişi sağlayan tüm geçiş olasılıklarının toplamı da Eşitlik (5.4)'de gösterildiği gibi 1 olacaktır. Kesikli Markov Modeli'nde yer alan tüm durumlara ait geçiş olasılıkları iki boyutlu bir dizi ile temsil edilebilir. Bu dizi $A=[a_{ij}]$ olarak ifade edilir. Durumlar ve durumlara ait geçiş olasılıkları grafiksel olarak özdevinirler kullanılarak temsil edilebilir. Kesikli Markov modelinin tam olarak ifade edilebilmesi için gerekli olan bir başka parametre ise her duruma ait başlangıç olasılıklarıdır. N adet durum olduğu düşünülürse N adet başlangıç olasılığı olacaktır. Başlangıç olasılıkları Eşitlik (5.5)'te gösterildiği gibi π sembolü ile tanımlanır. Bu olasılıkların toplamının da Eşitlik (5.6)'da gösterildiği gibi 1 olması gerekir.

$$a_{ij} \geq 0 \text{ ve } \sum_{j=1}^{N} a_{ij} = 1 \tag{5.4}$$

$$\pi_i \equiv P(q_1 = S_i) \tag{5.5}$$

$$\sum_{i=1}^{N} \pi_i = 1 \tag{5.6}$$

Üzerinde işlem yapılacak bir veri setimiz olduğunda yukarıda bahsedilen geçiş olasılıkları, başlangıç değerleri ve A matrisi elde edilebilir. Veri setinde her biri T uzunluğunda K tane gözlem dizisinin bulunması durumunda başlangıç olasılıkları verilmiş olan örnek dizi setindeki durumlardan kaç tanesinin ilgili durum ile

başladığına bakılarak belirlenir. Bu işlemin nasıl yapılacağı Eşitlik (5.7) ile gösterilmiştir.

$$\hat{\pi}_i = \frac{\{S_i \text{ ile Başlayan Gözlemler}\}}{\{\text{Gözlem Sayısı}\}} = \frac{\sum_{k=1}^{K} (q_1^k = S_i)}{K} \tag{5.7}$$

a_{ij} olarak temsil edilen örnek bir geçiş için geçiş olasılığı; S_i durumundan S_j durumuna olan geçişlerin sayısının, S_i durumundan yapılan tüm geçişlerin sayısına bölünmesi ile elde edilir. Geçiş olasılıklarının matematiksel formülü Eşitlik (5.8)'de gösterilmiştir.

$$\hat{a}_{ij} = \frac{\{S_i\text{'den } S_j\text{'ye Geçişlerin Sayısı}\}}{\{S_i\text{'den Yapılan Tüm Geçişlerin Sayısı}\}} = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T-1} (q_t^k = S_i \text{ ve } q_{t+1}^k = S_j)}{\sum_{k=1}^{K} \sum_{t=1}^{T-1} (q_t^k = S_i)} \tag{5.8}$$

Kesikli Markov Modeli'nde bir t anında hangi durumda olduğu bilinir ancak markov modelin farklı bir sürümü olan Saklı Markov Modeli'nde (SMM) ise t anındaki durum bilinmez. Ancak sistemin bu duruma gelene kadar vermiş olduğu çıktılar bilinir. Bir sonraki bölümde SMM detaylı olarak anlatılacaktır.

5.2. Saklı Markov Modeli

SMM, Kesikli Markov Modeli'nin özel bir halidir. Burada Kesikli Markov Modeli'nde olduğu gibi bir durum dizisi gözlenmez. Bunun yerine durum dizisinin gözlem dizisinden elde edilmesi gerekir. Durumların gözlemlerden elde ediliyor olması, modeli saklı yapan durumdur. Bir gözlem dizisini üretmiş olan pek çok farklı durum dizisi olabilir. Ancak bu durum dizilerinin her birinin olasılığı farklıdır.

SMM'de kullanılan temel örneklerden bir tanesi Alice ve Bob örneğidir. Alice ve Bob farklı bölgelerde yaşayan iki arkadaştır. Her gün birbirlerini arayarak o gün yaptıkları aktiviteler hakkında bilgi almaktadırlar. Bob'un gün içerisinde yaptığı üç aktivite bulunmaktadır. Bunlar; parkta yürüyüş, ev temizliği ve alışveriştir. Bunlardan hangisini yapacağı o gün havanın durumuna göre değişmektedir. Alice, Bob'un ona telefonda günlük olarak yaptığını anlattığı aktivitelere göre havanın durumunu tahmin etmeye çalışmaktadır. Havanın durumu "Yağmurlu" ya da

"Güneşli" olarak iki farklı değer alabilmektedir. Sonuç olarak Alice, Bob'un aktivitelerini gözlemleyerek saklı olan havanın durumunu SMM'de de yapıldığı gibi tahmin etmeye çalışmaktadır (Liu ve diğ., 2011).

SMM'de parametre kümesini tanımlarken, Kesikli Markov Model'den farklı olarak gözlemlerden durumlar elde edileceği için bir gözlem kümesi tanımlanır. Bu gözlem kümesini V sembolü ile temsil edersek gözlem kümesindeki simgeler ($V= \{v_1, v_{2,\ldots,} v_m\}$, m = 1,…., M) şeklinde gösterilebilir (Alpaydın, 2013).

Eşitlik (5.9)'da yer alan $b_j(m)$ değeri herhangi bir t anında S_j durumundayken v_m gözleminin görülme olasılığını göstermektedir. Burada Kesikli Markov Modeli'nden farklı olarak, durumlar değil gözlemlerle ilgilenilmektedir ve gözlem dizisini belirtmek için O değişkeni kullanılmaktadır (Alpaydın, 2013). SMM'de kullanılan temel semboller Tablo 5.1' de (Alpaydın, 2013) listelenmiştir.

$$b_j(m) \equiv P(O_t = v_m \mid q_t = S_j) \tag{5.9}$$

Tablo 5.1. SMM'de kullanılan temel semboller

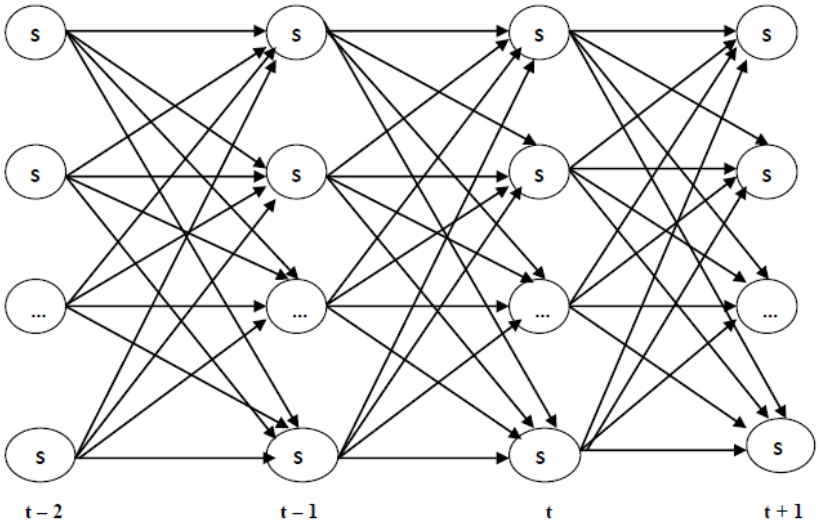
N: Durum Sayısı	$S = \{S_1, S_2, S_3, ..., S_N\}$
M: Gözlem değişkenlerinin kümesi	$V = \{V_1, V_2, V_3, ..., V_M\}$
Geçiş Olasılıkları	$A = [a_{ij}],\ a_{ij} \equiv P(q_{t+1} = S_j \mid q_t = S_i)$
Gözlem Olasılıkları	$B = [b_j(m)],\ b_j(m) \equiv P(O_t = v_m \mid q_t = S_j)$
Başlangıç Olasılıkları	$\pi = [\pi_i],\ \pi_i \equiv P(q_1 = S_i)$

SMM'nin en genel gösterimi ise λ sembolü kullanılarak yapılmaktadır. λ= (A, B, π) SMM'nin parametre kümesini ifade etmektedir (Alpaydın, 2013). Aynı zamanda SMM'nin en genel yapısı trellis diyagramı olarak Şekil 5.1'de (Ibe, 2009) gösterilmiştir.

SMM'de cevap bulunması gereken üç problem vardır. Bunlar;

1. Bir gözlem dizisinin gerçekleşme olasılığının bulunması,
2. Bir gözlem dizisi verildiğinde bu gözlem dizisini en yüksek elde etme olasılığına sahip durum dizisinin bulunması,

3. Bir eğitim (öğrenme) veri seti verildiğinde, bu veri seti aracılığıyla model parametrelerinin elde edilmesi.



Şekil 5.1. Saklı Markov Model'e ait trelis diyagramı

Birinci durumda bahsedilen problemin çözümü için Forward algoritması, ikinci durumda bahsedilen problemin çözümü için Viterbi algoritması ve son durumun çözümü için Forward-Backward algoritması, Expectation Maximization ya da BaumWelch algoritması ile birlikte kullanılmaktadır.

Birinci problem için elimizde bir $O=\{O_1, O_2, \ldots, O_T\}$ gözlem dizisi ve $Q=\{q_1, q_2, \ldots, q_T\}$ durum dizisi olduğunu düşünelim. Bu durumda O gözlem dizisinin görülme olasılığını koşullu olasılık formülünü ($P(O|\lambda)$) kullanarak hesaplayabiliriz (Alpaydın, 2013; Büyüktatlı, 2013; Ibe, 2009). Bu olasılığı özyinelemeli olarak hesaplayabilmek için $\alpha_t(i)$ olarak tanımlanmış bir ileri değişkeni olduğunu kabul edelim. (Alpaydın, 2013). Bir gözlem değeri için kısmi olasılık değeri, bu gözleme ulaşana kadar takip edilen tüm yolların ve bu yollardaki durumlarda görülen gözlem olasılıklarının çarpımı ile hesaplanır. Kısmi olasılıkların hesaplanmasında iki durum göz önünde bulundurulur. Bunlardan ilki t=1 durumu ve diğeri de t>1 durumudur.

t=1 anında gözlenmiş herhangi bir yol bulunmamaktadır. Bu durumda t=1 için hesaplama Eşitlik (5.10)'da (Alpaydın, 2013; Ibe, 2009) gösterildiği gibi yapılacaktır. Başlangıç olasılıkları hesaplandıktan sonra geri kalan olasılık değerleri Eşitlik (5.11)'de (Alpaydın, 2013; Ibe, 2009) gösterildiği gibi hesaplanabilir. Bu durumda

kısmi olasılıklar en genel haliyle Eşitlik (5.10) ve Eşitlik (5.11) kullanılarak hesaplanabilir;

$$t = 1, \qquad \alpha_1(i) = \pi_i b_i(O_1) \tag{5.10}$$

$$t > 1, \qquad \alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_j(O_{t+1}) \tag{5.11}$$

$\alpha_t(i)$ değerlerini gözlem dizisinin sonuna kadar hesapladığımızda tüm gözlemin görülme olasılığını elde etmiş oluruz (Ibe, 2009). Bu hesaplamanın nasıl yapılacağı Eşitlik (5.12)'de (Alpaydın, 2013; Ibe, 2009) gösterilmiştir.

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{5.12}$$

SMM'nin çözüm bulduğu bir diğer durum ise bir gözlem dizisi verildiğinde bu gözlem dizisini elimizdeki modele göre çıktı olarak verebilecek olasılığı en yüksek olan durum dizisinin bulunmasıdır. Bu problemin çözümü için Viterbi algoritması kullanılmaktadır. Viterbi algoritması, Forward algoritması ile benzer bir algoritmadır. Algoritmaların tek farkı Forward algoritması her adımda toplam alarak işlem yaparken, Viterbi algoritması her adımda maksimum olan değeri seçerek işlem yapmaktadır (Büyüktatlı, 2013). Viterbi algoritmasının en genel hali Eşitlik (5.13)'de (Alpaydın, 2013; Ibe, 2009) gösterilmiştir. Burada yer alan $\delta_t(i)$ değişkeni verilen modele göre t anında S_i durumu ile sonlanan ilk t gözlem için mevcut tüm yollardan en yüksek olasılıklı olan yolun olasılığını temsil etmektedir. Tüm gözlemi verecek en yüksek olasılıklı durum dizisi özyinelemeli olarak verilen gözlem dizisinin sonuna kadar her adımda en yüksek olasılıklı durumu seçerek hesaplanır. Hesaplamanın nasıl yapılacağı Eşitlik (5.13) ve Eşitlik (5.14)'te gösterilmiştir (Alpaydın, 2013; Ibe, 2009).

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} p(q_1 q_2 \ldots q_{t-1}, q_t = S_i, O_1 \ldots O_t \mid \lambda) \tag{5.13}$$

$$\delta_t(j) = \max_i \delta_{t-1}(i) a_{ij} . b_j(O_t) \tag{5.14}$$

SMM yönteminde çözüm bulunması gereken en önemli ve son problem ise verilen bir eğitim seti üzerinden model parametrelerinin öğrenilmesidir. Bu işlem için en yaygın olarak kullanılan yöntem Forward-Backward algoritmasıdır. Burada amaç sistemi optimize eden parametre değerlerini elde etmektir. Parametrelerin optimize edilmesi için kullanılabilecek yöntemlerden bir tanesi Baum Welch algoritmasıdır. Elimizdeki eğitim setine ait tüm yolların bilinmesi durumunda parametreleri tahmin etmek kolaydır ancak eğitim veri setinde oluşabilecek tüm yollar bilinmiyorsa Baum Welch gibi bir iteratif optimizasyon yöntemine ihtiyaç vardır.

Burada Forward algoritmasında kullanılan ileri değişkeninin yanında aynı zamanda geriye doğru işlem yapmamızı sağlayan bir algoritmaya da ihtiyaç vardır. Bu algoritmada kullanılan kısmi olasılık formülü ve β geri değişkeni Eşitlik (5.15)'te (Alpaydın, 2013; Ibe, 2009; Büyüktatlı, 2013) gösterilmiştir.

$$\beta_t(i) = P(O_{t+1}...O_T \mid q_t = S_i, \lambda) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \tag{5.15}$$

λ değişkeni ile SMM'nin parametrelerini tanımlamıştık. En uygun model parametrelerinin elde edilebilmesi için λ^* değişkeni tanımlanmış olsun. Bunun yanında $\xi_t(i,j)$ olasılığı da art arda gelen i ve j durumları arasındaki geçiş olasılığını göstermek için Eşitlik (5.16)'daki gibi tanımlanabilir. (Alpaydın, 2013; Ibe, 2009).

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda)$$

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_k \sum_u \alpha_t(k) a_{ku} b_u(O_{t+1}) \beta_{t+1}(u)} \tag{5.16}$$

Eşitlik (5.16)'da yer alan $\alpha_t(i)$ ileriye doğru ilk t gözleme ait olasılığı hesaplamaktadır. Daha sonra $a_{ij} b_j(O_{t+1})$ ile t anında S_i durumundan t+1 anında S_j durumuna geçme olasılığı hesaplanmaktadır. Daha sonra gözlemdeki t+1 anından T anına kadarki gözlem değerleri de $\beta_{t+1}(j)$ ile hesaplanmaktadır (Alpaydın, 2013).

Eşitlik 5.16'daki tanımlamayı yaptıktan sonra t anında S_i durumunda olma olasılığını Eşitlik (5.17)'de tanımlandığı gibi gösterebiliriz (Alpaydın, 2013; Ibe, 2009).

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{5.17}$$

K adet gözlem dizisi olduğu ve bu dizilerinin birbirlerinden bağımsız olduğunu düşünürsek. Tahmini model parametreleri Eşitlik (5.18), Eşitlik (5.19) ve Eşitlik (5.20) (Alpaydın, 2013) kullanılarak hesaplanabilir.

$$\hat{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P(O^k \mid \lambda)} \sum_{t=1}^{T_k-1} \xi_t^k(i,j)}{\sum_{k=1}^{K} \frac{1}{P(O^k \mid \lambda)} \sum_{t=1}^{T_k-1} \gamma_t^k(i)} \tag{5.18}$$

$$\hat{b}_j(m) = \frac{\sum_{k=1}^{K} \frac{1}{P(O^k \mid \lambda)} \sum_{t=1}^{T_k} \gamma_t^k(j) 1(O_t^k = v_m)}{\sum_{k=1}^{K} \frac{1}{P(O^k \mid \lambda)} \sum_{t=1}^{T_k} \gamma_t^k(j)} \tag{5.19}$$

$$\hat{\pi}_i = \frac{\sum_{k=1}^{K} \frac{1}{P(O^k \mid \lambda)} \gamma_1^k(i)}{\sum_{k=1}^{K} \frac{1}{P(O^k \mid \lambda)}} \tag{5.20}$$

SMM farklı alanlarda uygulaması olan bir yöntemdir. Bu alanlardan bir tanesi de bilgisayar ağları ve bulut bilişimdir. Bu alanda da pek çok araştırmacı tarafından yapılmış farklı çalışmalar bulunmaktadır. Abdlhamed ve diğ. (2016) bulut bilişim altyapısını kullanan sistemler için SMM yöntemini kullanarak atak tespiti yapmışlardır. Çalışmada ağ ortamındaki verilerin boyutunun büyük olduğundan, ağda oluşan aşırı yükleme ve tıkanıklıklar sebebiyle güvenliği sağlamak için veri madenciliği ve atak tespit sistemlerine gerek olduğundan bahsedilmiştir. Çalışmanın sonucunda bulut bilişim ortamı için SMM'nin uygun olduğu gösterilmiştir. Kholidy ve diğ. (2014) tarafından bulut bilişim mimarisinde atak tespiti için bir framework geliştirilmiştir. Geliştirilen frameworkte yer alan üç modelden iki tanesi SMM'yi kullanarak başarılı bir şekilde atak tespiti yapmaktadır. Yu-Ting ve diğ. (2014) tarafından yapılan farklı bir çalışmada ise ağ güvenliği sağlamak için ağın durumunu modelleyen bir SMM oluşturulmuştur. Joshi ve Phoha (2005) tarafından yapılan çalışmada ağdaki anormalliklerin tespiti için SMM kullanılmıştır. Çalışmada normal durumu ve atak durumunu modellemek için iki ayrı model oluşturulmuştur.

Sistemin %79 doğruluk oranı ile çalıştığı gözlemlenmiştir. Jain ve Abouzakhar (2012) tarafından yapılan başka bir çalışmada atak tespiti için SMM yöntemi kullanılmıştır. Elde edilen model farklı atak tipleri üzerinde test edildiğinde başarı oranının atak tipine bağlı olarak %76 ile %99 aralığında olduğu görülmüştür. Che ve Ji (2010) tarafından yapılan bir çalışmada atak tespiti için sistem çağrılarından ve SMM'den yararlanılmıştır. Oluşturulan model küçük boyutlu bir veri seti üzerinde test edilmiştir ve başarılı olduğu gösterilmiştir.

Literatürde yer alan çalışmalar incelendiğinde Markov Modeli yönteminin daha çok ağın anormal davranışını modellemek için kullanıldığı görülmektedir. Bu tez çalışmasında ilk olarak SMM ile ağın genel durumu modellenmeye çalışılmıştır. Bu amaçla ikinci bölümde bahsedilen dört parametre durum olarak düşünülmüştür ve bu parametreler arasında geçiş yapılarak model oluşturulmaya çalışılmıştır. Ancak bu çalışmada kullanılacak parametreler literatürde yer alan çalışmalardaki parametrelerden farklı olarak birbirleriyle doğrudan ilişkili parametrelerdir. Parametrelerin durum olarak belirlenmesi halinde her kontrol anında bir parametreden diğerine mutlaka geçiş olacaktır. Dolayısıyla bu şekilde bir modelin kurulması yanlış olacaktır. Yine farklı bir çözüm olarak senkronizasyonu yap ya da yapma şeklinde iki farklı durum kullanılarak modelin kurulması düşünülebilir. Bu durumda da örnekleme periyodunun bir saat olarak alındığı kabul edilirse, bir saat önceki verilerle oluşturulan bir model üzerinden, bir saat sonraki durumun tahmin edilmeye çalışılması da doğru sonuçlar vermeyecektir. Ağın farklı saat dilimlerindeki davranışı farklı modellerde olabileceğinden böyle bir modelin kullanılması da uygun olmayacaktır.

Yukarıda bahsedilen sebeplerle tez çalışması kapsamında biyolojik dizinlerin hizalamasında kullanılan farklı bir markov model türü olan PHMM yöntemi kullanılmıştır. Yöntemin detaylı açıklaması bir sonraki bölümde verilmiştir.

5.3. Profile Hidden Markov Model

Krogh ve diğ. (1994) tarafından çoklu dizi hizalaması için Saklı Markov Model'den daha uyumlu olacak bir yöntem önerilmiştir. Burada bahsedilen çoklu dizi hizalaması ifadesi en az üç tane, protein dizisi gibi biyolojik dizinin hizalanmasıdır. Proteinler pek çok aminoasidin bir araya gelmesi ile oluşan aminoasit dizileridir. PHMM'de

hizalamada kullanılan dizilerin ortak bir aileye sahip olduğu düşünülür ve aralarındaki bağlantı bir model ile ortaya çıkarılmaya çalışılır. Buradaki problem bir protein dizisi verildiğinde bu protein dizisini üreten yani bu protein dizisinin ait olduğu aileyi temsil eden modeli elde etmektir (Mount, 2004).

Bu amaçla geliştirilen yöntem Profile Hidden Markov Model olarak ifade edilmektedir. Bu yöntemin SMM'den en temel farklılığı eğitim kümesinde yer alan dizilerin sıraya bağımlı olmasıdır ve dizinin herhangi bir sırasında yer alan amino asitin birden fazla değer alabilmesidir. PHMM'de durumları ifade etmek için zaman dilimleri ya da dizinin indisleri kullanılmaktadır. PHMM, SMM'den farklı olarak bir başlangıç durumu ve sonlandırma durumuna sahiptir. Yine SMM'den farklı olarak modelde sürekli ileriye doğru gidiş vardır. Yöntem hiçbir zaman geriye doğru bir adım içermez. PHMM'de her bir adımda bulunulan durumların ifade edilebileceği üç farklı durum çeşidi bulunmaktadır. Bunlar eşleme durumu, ekleme durumu ve silme durumu olarak ifade edilmektedir. Tablo 5.2'de bu durumların şekilsel gösterimi verilmiştir.

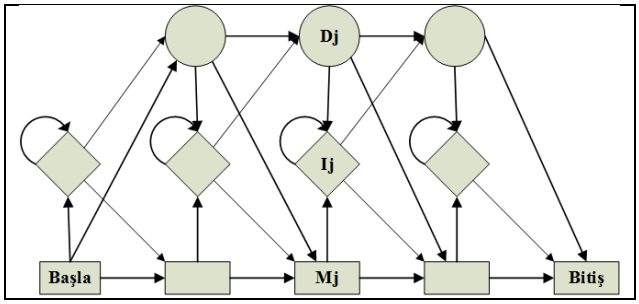
Tablo 5.2. PHMM durum listesi

◇	Ekleme Durumu
▭	Eşleme Durumu
○	Silme Durumu

Bir PHMM bu üç farklı durumu da aynı anda içermek zorunda değildir. Farklı problemler için farklı durum dizilerini içeren modeller oluşturulabilir. Şekil 5.2'de PHMM' in bu üç durumu da içeren basit bir örneği gösterilmiştir.

Şekilde yer alan eşleme durumları hizalanmış protein dizilerinde probleme özgü olarak tanımlanmış kurallara uyan sütunları temsil etmektedir. Eşleme olarak adlandırılan durumlarda her durumda sadece bir aminoasit bu konuma gelebilmektedir. Ancak ekleme olarak adlandırılan durumlarda ekleme durumunun bulunduğu konuma birden fazla aminoasit gelebilmektedir. Silme durumu da eşleme durumunu içermeyen sıralıları temsil etmek için kullanılan bir durumdur. Eşleme durumlarında model sürekli (bitiş durumunu görene kadar) ileri yönde ilerlemektedir. Silme durumu eşleme durumundan atlamak için kullanılmaktadır. Eşleme

durumlarının sayısı aslında bize modelin uzunluğunu vermektedir ve model kurulurken karar verilmesi gereken ilk problem eşleme durumlarının sayısı olacaktır.



Şekil 5.2. PHMM'deki tüm durumları içeren basit bir PHMM örneği

Problemde yer alacak durumların sayısı tespit edildikten sonra SMM'de olduğu gibi, burada da her duruma ait geçiş olasılıkları ve elde edilecek çıktılara ait olasılık değerleri, verilmiş olan bir eğitim veri seti üzerinden hesaplanmalıdır. PHMM'de yer alan silme durumu, ekleme ve eşleme durumlarından farklı bir durumdur. Silme durumuna, diğer durumlardan geçiş yapılabilmesine rağmen, silme durumundayken herhangi bir çıktı oluşmaz.

PHMM matematiksel olarak 5 parametre ({Q, V, P(i), A, B}) kullanılarak ifade edilir. Burada yer alan Q = $\{q_1, q_2, \ldots, q_n\}$ durumlar kümesini, V; çıktı alfabesini, P(i); t zamanında q_i durumunda bulunma olasılığını, A; geçiş olasılıkları kümesini, a^t_{ij}; t anında q_i durumundayken t+1 anında q_j durumunda bulunma olasılığını, B; çıktı olasılıkları kümesini ve $e^t_i(x)$; t anında q_i durumundayken x çıktısının oluşma olasılığını ifade etmektedir. Herhangi bir eğitim veri seti üzerinden hesaplanacak geçiş ve çıktı olasılıklarının formülü sırasıyla Eşitlik (5.21) ve Eşitlik (5.22)'de gösterilmiştir (Durbin, 1998).

$$a_{ij} = \frac{A_i j}{\sum_{j'} A_i j'} \tag{5.21}$$

$$e_i(x) = \frac{E_i(x)}{\sum_{x'} E_i(x')} \tag{5.22}$$

Eşitlik (5.21)'de yer alan a_{ij}; q_i durumundan q_j durumuna geçiş olasılığını ve $A_i j$ ifadesi ise q_i durumundan q_j durumuna yapılan geçişlerin sayısını ifade etmektedir.

Eşitlik (5.22)'de yer alan $e_i(x)$ ifadesi q_i durumundayken x çıktısının oluşma olasılığını ve $E_i(x)$ ifadesi de q_i durumundayken x çıktısının oluşma sayısını göstermektedir. Verilen veri seti üzerinden yukarıdaki formüller kullanılarak model kurulur ve başlangıç değerleri elde edildikten sonra oluşturulacak iki boyutlu bir geçiş tablosu ile modelde yer alacak olasılıklar tablo üzerinden dinamik olarak hesaplanır. Geçiş tablosu S X S boyutlu bir tablo olacaktır. Örnek olarak modelde iki eşleme durumu, iki ekleme durumu ve iki silme durumu olduğu düşünülürse, geçiş tablosu Tablo 5.3'de gösterildiği gibi olacaktır. Tablonun dinamik programlama ile doldurulması sayesinde yapılacak gereksiz hesaplamaların önüne geçilmektedir. Her adımda hesaplanan değerler tabloda tutularak bir sonraki adımda bu değerlere ihtiyaç olması durumunda tabloda kayıtlı olan değerler kullanılmaktadır.

Tablo 5.3. PHMM için örnek geçiş tablosu

	(Başla)	M_1	M_2	I_1	I_2	D_1	D_2	Bitiş
(Başla)								
M_1								
M_2								
I_1								
I_2								
D_1								
D_2								
Bitiş	0	0	0	0	0	0	0	0

Model kurulduktan sonra Bölüm 5.2'de bahsedildiği gibi yeni bir sıralının bu model tarafından üretilme olasılığı Forward algoritması kullanılarak ve belirli bir çıktı sıralısına karşılık gelen, model üzerinden elde edilebilecek en muhtemel durum dizini ya da yol Viterbi algoritması (Viterbi, 1967) kullanılarak elde edilebilir. (Durbin, 1998; Forney, 1973). Tez çalışması kapsamında model üzerinden elde edilebilecek en genel yolun bulunması amaçlanmaktadır. Bu nedenle PHMM için Forward algoritmasına bu bölümde değinilmeyecektir. Algoritmanın çalışma mantığı bölüm 5.2'de detaylı olarak açıklanmıştır.

Eşitlik (5.23)'de Viterbi Algoritmasının PHMM' de bulunan üç duruma göre düzenlenmiş hali verilmiştir. Eşitliklerde yer alan $\delta_s^M(t)$ değeri; t anında M_s: eşleme

durumu ile biten en yüksek olasılıklı yolu, $\delta_s^I(t)$ değeri; t anında I_s: ekleme durumu ile biten en yüksek olasılıklı yolu ve $\delta_s^D(t)$ değeri; t anında D_s: silme durumu ile biten en yüksek olasılıklı yolu ifade etmektedir. Eşleme durumunda yer alan $e_{M_s}(x_t)$ ifadesi t anında s: eşleme durumunda x_t çıktısının oluşma olasılığını ve ekleme durumundaki $e_{I_s}(x_t)$ ifadesi de t anında s: ekleme durumunda x_t çıktısının oluşma olasılığını göstermektedir. Silme durumunda herhangi bir çıktı oluşmadığı için silme durumuna ait eşitliklerde çıktı olasılığı yer almamaktadır. Burada yer alan s değişkeni modelin uzunluğunu yani modeldeki eşleme durumlarının sayısını ve T değişkeni de protein dizisinin ya da örneğin uzunluğunu temsil etmektedir. Olasılığı hesaplanacak duruma göre ilgili eşitlik seçilerek en yüksek olasılıklı yol elde edilir.

$$\begin{aligned}
\delta_s^M(t) &= e_{M_s}(x_t) + \max\begin{cases}\delta_{s-1}^M(t-1)a_{M_{s-1}M_s},\\ \delta_{s-1}^I(t-1)a_{I_{s-1}M_s},\\ \delta_{s-1}^D(t-1)a_{D_{s-1}M_s};\end{cases}\\
\delta_s^I(t) &= e_{I_s}(x_t) + \max\begin{cases}\delta_{s-1}^M(t)a_{M_sI_s},\\ \delta_{s-1}^I(t)a_{I_sI_s},\\ \delta_{s-1}^D(t)a_{D_sI_s};\end{cases}\\
\delta_s^D(t) &= \max\begin{cases}\delta_s^M(t-1)a_{M_{s-1}D_s},\\ \delta_s^I(t-1)a_{I_{s-1}D_s},\\ \delta_s^D(t-1)a_{D_{s-1}D_s};\end{cases}
\end{aligned} \tag{5.23}$$

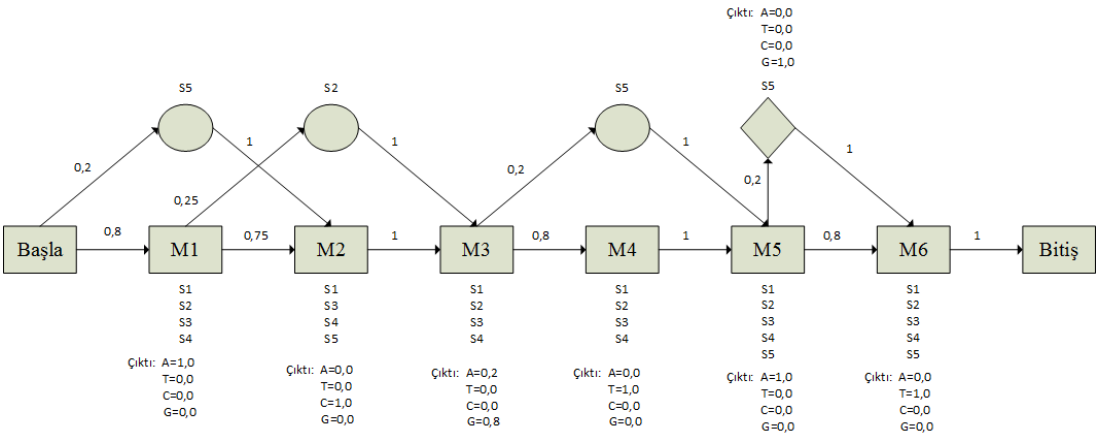
PHMM'ye ait basit bir örmek (Yolal, 2018) aşağıda açıklanmıştır. Model oluşturulmasında kullanılacak protein eğitim veri seti Tablo 5.4'de gösterilmiştir. Bu model üzerinden örnek bir PHMM elde etmeye çalışırsak, aşamalar aşağıdaki gibi olacaktır:

İlk olarak yapılması gereken, verilmiş olan kural dizileri üzerinden eşleme durumlarının belirlenmesidir. Buradaki kuralın, "hizalanmış sütunlarda yer alan kayıp aminoasit sayısı (- sayısı) ikiden fazla ise bu hizayı eşleme durumu temsil edemez" şeklinde tanımlandığını kabul edelim. Bu durumda bu kurala uyan alanlar korunmamış alan olarak ifade edilir. Kayıp aminoasit sayısı iki ya da ikiden az ise bu alanlar korunmuş alan (yani eşleme durumu) olarak ifade edilir. Bu durum bu

örnekteki protein sıralıları için tanımlanmış bir kuraldır. Farklı örnekler için probleme özgü kurallar belirlenebilir. Belirlenen kural üzerinden örnek veri kümesi incelendiğinde protein sıralılarındaki 1., 2., 3., 4., 5. ve 7. sütunların korunmuş (eşleme), altıncı sütunun ise korunmamış alan olduğu görülmektedir. Bu durumda veri setimizin altı tane eşleme durumu bulunacaktır. Daha önce tanımladığımız ekleme durumu korunmamış alan olarak ifade edilen alanlara ekleme işlemini, silme durumu ise korunmuş olarak tanımlanan alanlardan silme işlemini ifade etmektedir. Korunmamış olarak belirlenmiş bir sütunda yer alan "–" değerleri önemsiz olarak ifade edilen sütunlardır. Bu tanımlardan sonra her eşleme durumunun üstüne bir ekleme durumu, her ekleme durumunun üstüne de bir silme durumu yerleştireceğiz. Son durum olan bitiş durumunun üstünde herhangi bir sembol yer almayacaktır. Eşitlik (5.21) ve Eşitlik (5.22) kullanılarak modeli oluşturmak istersek oluşan şekil ve çıktılar Şekil 5.3'te gösterildiği gibi olacaktır.

Tablo 5.4. Örnek protein eğitim veri seti

S1: Birinci Sıralı:	A	C	G	T	A	-	T
S2: İkinci Sıralı:	A	-	G	T	A	-	T
S3: Üçüncü Sıralı:	A	C	A	T	A	-	T
S4: Dördüncü Sıralı:	A	C	G	T	A	-	T
S5: Beşinci Sıralı:	-	C	G	-	A	G	T
	M1	M2	M3	M4	M5		M6



Şekil 5.3. Örnek veri setine ait PHMM

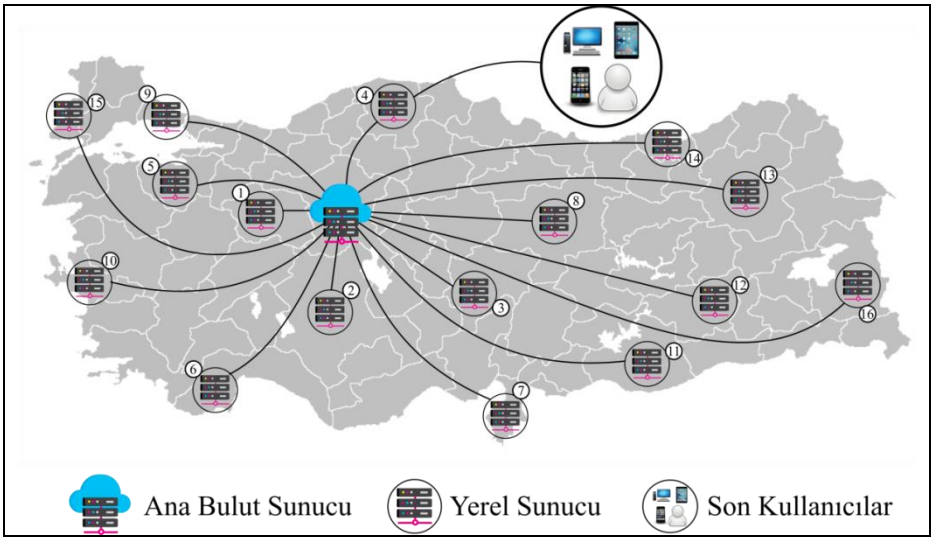
Örneğin başlangıç durumundayken sıralı ilk sütunu düşünürsek S1, S2, S3 ve S4 sıralılarının ilk sütunu kayıp değildir ve M1 durumuna geçer olasılığı ise toplamda 5

durumda 4 tanesi M1 durumuna geçtiğinden 4/5'ten 0,8 olarak hesaplanır. S5 ise ilk sütunda kayıp aminoasit içerdiği için 1/5'ten 0,2 olasılık ile silme durumuna geçer. Benzer şekilde ikinci sütun için işlem yapıldığında silme durumdaki S5 sıralısının 1 olasılık ile M2 durumuna geçiş yaptığı M1 durumundaki S1, S2, S3 ve S4 sıralılarından S1, S3 ve S4 sıralılarının 3/4'ten 0,75 olasılık ile M2 durumuna ve S3 sıralısının 1/4'ten 0,25 olasılık ile ikinci sütununda kayıp aminoasit içerdiğinden silme durumuna geçiş yaptığı görülmektedir.

Bu şekilde her sütun için hesaplamalar yapılarak yukarıdaki model elde edilmiştir. Modelde her bir durumdan diğer durumlara olan geçiş olasılıklarının toplamının bir olduğu görülmektedir. Model kurulurken her duruma ait sıralı dizilerinin de saklanması gerekmektedir. Bu nedenle her adımda ilgili sıralılar saklanmıştır ve diğer duruma geçiş yapılırken sadece bu sıralılar ile işlem yapılmıştır. Model olasılıksal hesaplamalar ile kurulduktan sonra Eşitlik (5.23)'de verilen Viterbi algoritması kullanılarak verilen bir çıktı dizisini üretecek en muhtemel durum dizisi hesaplanabilir.

6. SENKRONİZASYON ZAMANINI BELİRLEME

Tez çalışması kapsamında günlük davranışı belirli bir örüntüye sahip olan ağlar (okul, kampüs ağları vb.) için bir senkronizasyon planlama yöntemi önerilmiştir. Şekil 6.1'de önerilen yöntemin kullanılabileceği örnek bir bulut temelli CDN mimarisi gösterilmiştir. Buradaki mimaride merkezde yer alan sunucu ana bulut sunucuyu temsil etmektedir. Türkiye'nin farklı illerine yerleştirilen sunucular ise farklı üniversitelere ait yerel sunucuları temsil etmektedir. Her yerel sunucu kendi ağına ait verileri NoSQL bir veri tabanı olan CouchDB'de saklamaktadır. Her ağın saklamış olduğu veriler o üniversiteye aittir ve bu veriler aynı zamanda bulut sunucuda da merkezi olarak saklanmaktadır. Böyle bir mimaride veriler, kullanıcı cihazları, yerel sunucular ve bulut sunucu olmak üzere üç farklı konumda tutulmaktadır. Bu nedenle zaman içerisinde farklı konumlarda bulunan bu verilerin eşlenmesi gerekmektedir. Bu mimaride senkronizasyon işlemi, hem bulut sunucu hem de yerel sunucular tarafından çift yönlü olarak başlatılabilmektedir.



Şekil 6.1. Bulut temelli örnek bir dağıtık sistem mimarisi

Çalışmada Şekil 6.1'de gösterilen mimariye benzer bulut tabanlı CDN mimarileri için en uygun senkronizasyon zamanını belirleyecek bir poll yöntemi önerilmiştir. Yöntemin detayları ve elde edilen sonuçlar alt bölümlerde sunulmuştur.

6.1. Profile Hidden Markov Model Tabanlı Poll Yöntemi (PHMM Poll)

Senkronizasyon için uygun zamanı belirlemek amacıyla Profile Hidden Markov Model temelli PHMM Poll olarak isimlendirilen farklı bir poll yöntemi önerilmiştir. Önerilen yöntemin temel adımları aşağıda listelenmiştir;

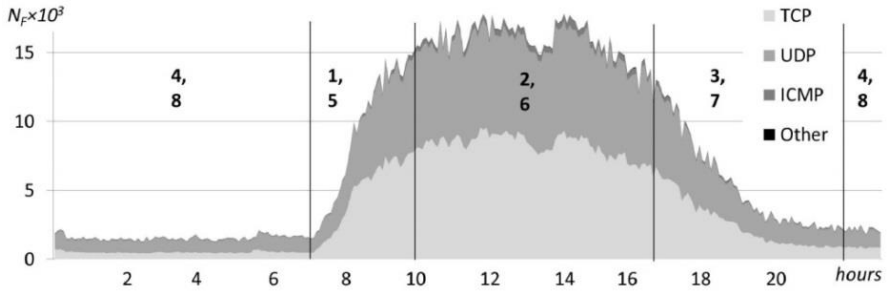
- İlk olarak modellenecek ağa ait veriler bir gün boyunca bir dakikalık periyotlarla kaydedilmiştir. Kaydedilen veri kümesi dört parametre içermektedir. Bunlar; ana bulut sunucunun yükü, cevap zamanı, gecikme ve paket yeniden iletim sayısıdır.
- Daha sonra bir günlük veri seti boş, normal ve yoğun olarak isimlendirilen üç farklı kümeye ayrılmıştır. Bu aşamada Bulanık C-Ortalamalar yöntemi kullanılmıştır.
- Sonraki adımda modellenecek her zaman aralığı için yeniden 10 gün boyunca birinci adımda verilen dört parametre kaydedilmiştir ve veriler 1-NN (Chatziantoniou ve diğ., 2013; Wan ve diğ., 2015) sınıflandırma yöntemine göre kümelere atanmıştır.
- Modellenecek her zaman aralığı için üçüncü adımda elde edilen 10 günlük küme örüntüleri kullanılarak 2 durumlu bir PHMM oluşturulmuştur. Buradaki iki durumdan biri senkronizasyonun yapılmasını, diğeri ise senkronizasyonun yapılmamasını temsil etmektedir.
- Her zaman dilimine uygun en iyi durum dizisini elde edebilmek amacıyla Viterbi algoritması elde edilen modeller üzerinde uygulanmıştır.

Alt bölümlerde ilk olarak önerilen yöntem basit bir senaryo üzerinden detaylı olarak açıklanmıştır. Daha sonra yöntemin farklı senaryolar, farklı poll periyotları, farklı ağ mimarileri altındaki davranışı incelenmiştir ve sonuçlar açıklanmıştır.

6.2. PHMM Poll Yöntemi Detayları

Çalışmanın bu bölümünde Şekil 6.1 ile gösterilen mimari modellenmiştir. Şekilde numaralandırılan LAN'lar üniversiteler ait kampüs ağlarıdır. Böyle bir mimaride ana sunucuda oluşan yoğunluk, yerel sunucularda oluşan yoğunluk ile benzer bir örüntüye sahip olacaktır. Bu nedenle herhangi bir kampüs ağına ait genel örüntünün modellenmesi gerekmektedir. Akademik bir ağın içerisinde bulunan kullanıcıların bir günlük davranışına göre analizi Şekil 6.2'de (Garsva ve diğ., 2014) gösterilmiştir. Şekilden de anlaşılacağı gibi örnekler bir dakikalık periyotlarla alındığında, bütün bir günün davranışını modellemek için $2 \times 24 \times 60 = 2880$ durumlu bir model oluşacaktır.

Böyle bir model hem karmaşık olacaktır hem de zayıf bir performans gösterecektir. Bu nedenle benzer örüntüye sahip zaman dilimlerinin belirlenmesi ve her bir zaman dilimi için farklı modellerin oluşturulması daha doğru olacaktır.



Şekil 6.2. Bir kampüs ağındaki günlük ortalama ağ trafiği dağılımı

Şekil 6.2'de akademik bir ağın günlük yoğunluğu kullanımına göre 4 parçaya ayrılmıştır. Şekilden de anlaşılacağı gibi ağ gece saatlerinde kullanıcılar tarafından kullanılmamaktadır. 22:00-7:00 (4 ve 8 numaralı) saat aralıklarında sadece bazı planlanmış görevler gerçekleştirilmektedir. 07:00-10:00 (1 ve 5 numaralı) zaman dilimlerinde ağın kullanımında artış olduğu görülmektedir. 10:00-16:30 (2 ve 6 numaralı) zaman dilimlerinde ağ yoğunluğu en üst seviyelere çıkmaktadır ve 16:30-22:00 aralığında (3 ve 7 numaralı) kullanıcılar üniversiteden ayrıldığı için ağ trafiğinde bir düşüş görülmektedir. Ağın iç trafiğindeki artış ve azalış ile paralel olarak senkronizasyonu yapılacak dosya boyutları ve sunucuların yoğunlukları da artış ve azalış gösterecektir ve hafta içi farklı günlerdeki aynı zaman dilimlerinde, ağın davranışı benzer olacaktır. Bu nedenle farklı günlerde aynı zaman dilimlerine ait olacak şekilde toplanmış veri örnekleri, aynı ataya ait protein dizileri gibi düşünülebilir ve PHMM kullanılarak modellenebilir. Elde edilen modellerden o zaman dilimi için ağın en genel davranışı çıkarılabilir.

Çalışmanın bu bölümde önerilen yöntemin adım adım modellenebilmesi için 15 dakikalık bir senaryo üzerinden önerilen PHMM Poll yöntemi ile modelin kurulması açıklanacaktır. Bu 15 dakikalık zaman dilimi Şekil 6.2'de 17:30-17:45 zaman aralığına karşılık gelmektedir. Bu zaman dilimi 3 ve 7 ile numaralandırılan yoğun durumdan ağın boş olduğu duruma geçişi temsil eden aralığa karşılık gelmektedir.

İlk olarak CouchDB' nin normal poll işlemi aralığı olan bir dakikalık periyotlarla herhangi bir yerel sunucudan ana sunucuya gönderilen senkronizasyon paketlerine ait

ana sunucu yükü, gecikme, cevap zamanı ve yeniden iletim sayısı değerleri bir gün boyunca kaydedilmiştir ve bu değerler FCM algoritmasında eğitim verisi olarak kullanılmıştır. Bu parametreler üzerinden ağa ait boş normal ve yoğun kümeleri belirlenmiştir. Elde edilen kümelerin merkezleri Tablo 6.1'de gösterilmiştir.

Tablo 6.1. Günlük veri setinden elde edilen küme merkezleri

Kümeler	**Ana Sunucu Yükü (bytes/sn.)**	**Cevap Zamanı (sn.)**	**Gecikme (sn.)**	**Paket Yeniden İletim Sayısı**
Küme 1 (Yoğun)	584012,601	34,675670	32,379840	4,285
Küme 2 (Normal)	454340,051	12,308833	18,391633	3,176
Küme 3 (Boş)	48782,330	0,181096	0,165582	1,013

Kümeler belirlendikten sonra farklı günlerde aynı saat dilimine ait veriler on gün boyunca kaydedilmiştir. Böylece o saat diliminin yoğunluğunu modelleyen farklı örüntüler elde edilmiştir. Tablo 6.2'de birinci güne ait olarak 17:30 – 17:45 zaman aralığında elde edilmiş 15 dakikalık örnek veri seti gösterilmiştir.

Tablo 6.2. Birinci günün 17:30-17:45 aralığına ait örnek veri seti

Ana Sunucu Yükü (bytes/sn.)	**Cevap Zamanı (sn.)**	**Gecikme (sn.)**	**Paket Yeniden İletim Sayısı**	**Küme Etiketi**
177766,783	38,427528	34,794197	5	B
177285,283	12,516582	11,870396	3	N
28970,216	0,284594	0,292030	1	I
145917,733	36,348720	32,875405	4	B
189638,349	36,747030	33,256682	4	B
79446,916	16,324005	17,854945	2	N
68749,699	0,293006	0,273204	1	I
26890,483	0,301438	0,288924	1	I
19687,549	0,28459	0,292030	1	I
13161,0	0,302518	0,285106	1	I
15521,133	0,294566	0,274699	1	I
17382,716	0,288884	0,293961	1	I
41291,183	6,319901	5,926780	2	I
28410,866	6,877206	6,467860	2	I
24161,516	0,296272	0,293031	1	I

Tablo 6.2'de aynı zamanda son sütunda her veri satırının hangi kümeye ait olduğu da gösterilmiştir. Bu aşamada her örneğin uygun kümeye atanması için 1-NN

algoritması kullanılmıştır. Yoğun kümesini temsil etmesi B harfi, normal kümesini temsil etmesi N harfi, boş kümesini temsil etmesi için I harfi kullanılmıştır.

Tablo 6.3'de on gün boyunca 17:30-17:45 aralığına ait toplanan verilerin kümelenmiş hali gösterilmiştir. Tabloda da görüldüğü gibi 10 farklı 15 elemanlı örüntü elde edilmiştir. Tabloda yer alan G1, G2,…, G10 ifadeleri alınan örneğin hangi güne ait olduğunu göstermektedir. Yani G1 birinci güne ait örüntüyü temsil etmektedir. Çalışmada bu sıralılar PHMM'de kullanılan protein dizileri gibi düşünülmüştür. PHMM'de modeli elde etmek için kullanılacak sıralı sayısı için bir alt sınır belirtilmemiştir ancak sayının çok küçük olması örüntünün tam olarak elde edilememesine, çok büyük olması da uzun işlem sürelerine sebep olacaktır. Bu nedenle tez çalışmasında sıralı sayısı ne çok küçük ne de çok büyük olacak şekilde 10 olarak belirlenmiştir.

Tablo 6.3. 10 günlük 17:30-17:45 aralığına ait kümelenmiş veri seti

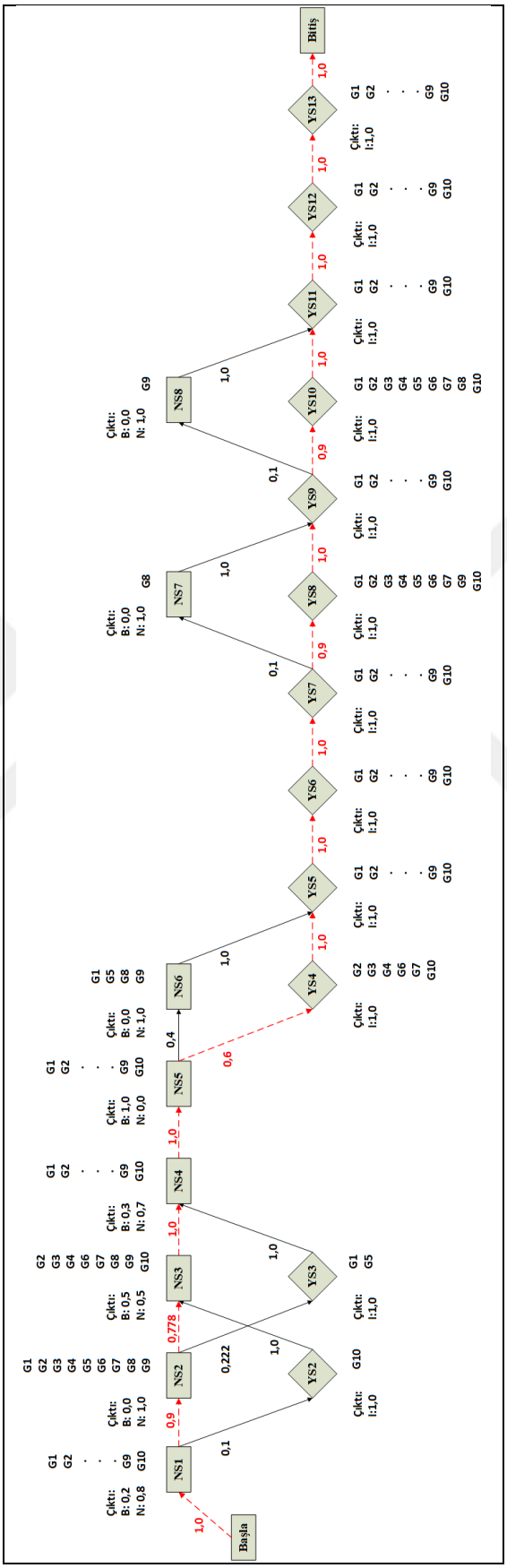
		Dakika														
		1.	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**	**10.**	**11.**	**12.**	**13.**	**14.**	**15.**
Günler	G1	B	N	I	B	B	N	I	I	I	I	I	I	I	I	I
	G2	N	N	N	B	B	I	I	I	I	I	I	I	I	I	I
	G3	N	N	B	B	B	I	I	I	I	I	I	I	I	I	I
	G4	B	N	B	B	B	I	I	I	I	I	I	I	I	I	I
	G5	N	N	I	B	B	N	I	I	I	I	I	I	I	I	I
	G6	N	N	N	B	B	I	I	I	I	I	I	I	I	I	I
	G7	N	N	B	B	B	I	I	I	I	I	I	I	I	I	I
	G8	N	N	N	B	B	N	I	I	I	N	I	I	I	I	I
	G9	N	N	B	B	B	N	I	I	I	I	I	N	I	I	I
	G10	N	I	N	B	B	I	I	I	I	I	I	I	I	I	I

Tablo 6.3'teki sıralılar kullanılarak elde edilen model Şekil 6.3'te gösterilmiştir. Şekilden de anlaşılabileceği gibi PHMM algoritması senkronizasyon problemine uygun olacak şekilde düzenlenmiştir. Klasik PHMM algoritmasındaki üç durumdan farklı olarak burada iki farklı durum bulunmaktadır. Bunlar NS ve YS sembolleri kullanılarak gösterilmiştir. NS durumu senkronizasyonun yapılmama durumunu, YS durumu ise senkronizasyonun başlatılma durumunu göstermektedir. NS durumu, PHMM' de yer alan eşleme durumunun sembolü ile ve YS durumu da ekleme durumunun sembolü ile gösterilmiştir. Sıralanmış olan her kolonda YS ya da NS

durumlarına geçiş yapılabilmektedir. Tablo 6.3'te sıralanmış olan B ve N sembolleri NS durumuna yapılacak geçişleri, I sembolleri ise YS durumuna yapılacak geçişleri göstermektedir. Yani NS durumunda oluşabilecek iki farklı çıktı bulunmaktadır, bunlar B ve N sembolleridir. YS durumunda da oluşabilecek tek çıktı vardır. O da I sembolüdür. Bu sembol de zaten senkronizasyonun başlatılabileceğini göstermektedir.

Şekil 6.3'te gösterilen modelde yer alan geçiş olasılıkları Eşitlik (5.21) kullanılarak ve çıktı olasılıkları da Eşitlik (5.22) kullanılarak elde edilmiştir. Elde edilen olasılık değerleri ve her durumu temsil eden gün değişkenleri şekilde detaylı olarak gösterilmiştir. Modelin uzunluğu örnekleme periyodunun (15 dakika) poll periyoduna (1 dakika) bölünmesiyle elde edilebilir. Tablo 6.3'te yer alan her kolon mevcut durumdan bir sonraki duruma geçişi hesaplamak için kullanılmaktadır. Örneğin ilk sütun başlangıç durumundan NS1 durumuna geçişi sağlar. İkinci sütun NS1 durumundan NS2 ve YS2 durumlarına geçişi sağlar. Üçüncü sütun NS2 ve YS2 durumlarından NS3 ve YS3 durumlarına geçişi sağlar ve bu şekilde modelin sonuna kadar devam eder.

Başlangıç durumundayken geçiş yapılabilecek YS1 ve NS1 olarak ifade edilen iki farklı durum vardır. Ancak Tablo 6.3'te ilk sütunda gözlenen I sembolü bulunmadığından 1 olasılık ile tüm durumlar NS1 durumuna geçiş yapmaktadır. YS1 durumuna geçiş olasılığı da sıfırdır. Bu nedenle Şekil 6.3'te bu durum gösterilmemiştir. NS1 durumundayken on günün ikisinde B sembolü, sekizinde N sembolü gözlenmiştir. Bu nedenle B sembolünün gözlenme olasılığı 0,2 ve N sembolünün gözlenme olasılığı da 0,8 olacaktır. NS1 durumundayken mevcut 10 gözlemden sadece G10'da I sembolü gözlenmektedir (Tablo 6.3'te ikinci sütun) ve YS1 durumuna geçiş yapılmaktadır. Bu nedenle NS1 durumundan YS1 durumuna geçiş olasılığı 0,1 (1/10) olacaktır. Diğer günlerde B ya da N sembolü gözlemlendiğinden NS1 durumundan NS2 durumuna geçiş olasılığı 0,9 (9/10) olacaktır. NS2 durumdayken hiç B sembolü gözlemlenmediğinden burada B sembolü görülme olasılığı 0 ve N sembolü görülme olasılığı 1 olacaktır. YS1 durumu için de I sembolünün görülme olasılığı 1 olacaktır.



Şekil 6.3. Tablo 6.3'teki sıralılara göre elde edilen model

Senkronizasyonu başlatma işlemi sadece YS durumlarında gerçekleştiğinden ve bu durumlarda sadece I sembolü gözlendiğinden, YS durumlarında I sembolünün görülme olasılığı her zaman 1, B ve N sembolünün görülme olasılığı ise 0 olacaktır. Benzer şekilde NS durumdan hiçbir zaman I sembolü görülmeyeceğinden, NS durumu için I sembolünün gözlem olasılığı her zaman 0 olacaktır. Yukarıda Tablo 6.3'teki birinci ve ikinci sütunlar için geçiş ve sembollerin çıktı olasılıklarının nasıl hesaplandığı açıklanmıştır. Diğer sütunlar içinde Eşitlik (5.21) ve Eşitlik (5.22) kullanılarak hesaplamalar yapılmıştır.

Model elde edildikten sonra herhangi bir çıktı dizisi verilmeden, modele ait olasılığı en yüksek durum dizisinin elde edilmesi gerekmektedir. Böylece ağın işlem yapılan zaman dilimindeki genel davranışı elde edilecektir. Bu amaçla Viterbi algoritması kullanılmıştır. Viterbi algoritmasının Disjkstra algoritması gibi en iyi yolu bulmak için kullanılabileceği daha önce yapılan çalışmalarda (Quach ve Farooq, 1994; Wolf ve dig., 1989) gösterilmiştir. Bu çalışmada da Viterbi yöntemi, en iyi yol algoritması mantığı ile kullanılmıştır. Böylece hem modele en çok uyan durum dizisi hem de bu durum dizisinde oluşabilecek en muhtemel çıktı dizisi Şekil 6.4'te gösterilen sözde kod kullanılarak elde edilmiştir ve çıktılara göre senkronizasyon işlemi planlanmıştır.

Algoritma 2: Viterbi Algoritması

$\delta_0(0) = 1$;
$ptr_0(0) = 0$;
for t = 1 **to** *T* **do**
    $\delta_t(j) = \max_j(e_j)\max_i(\delta_{t-1}(i)a_{ij})$;
    $ptr_t(j) = \text{argmax}_i(\delta_{t-1}(i)a_{ij})$;
end
En İyi Skor: $p^* = \max_i(\delta_T(i))$;
Geri İzlemenin Başlaması: $q_T^* = \text{argmax}_i(\delta_T(i))$;
for t = T-1 **to** 1 **do**
    $q_{t-1}^* = ptr_t(q_t^*)$;
End

Şekil 6.4. PHMM Poll modeli üzerinden en iyi yolu bulmak için kullanılan Viterbi algoritması

Kodda yer alan $\delta_t(j)$ ifadesi t anındaki j durumu ile biten en yüksek olasılıklı yolu göstermektedir. Modelin başlangıç ve bitiş durumları hariç T uzunluğunda olduğu

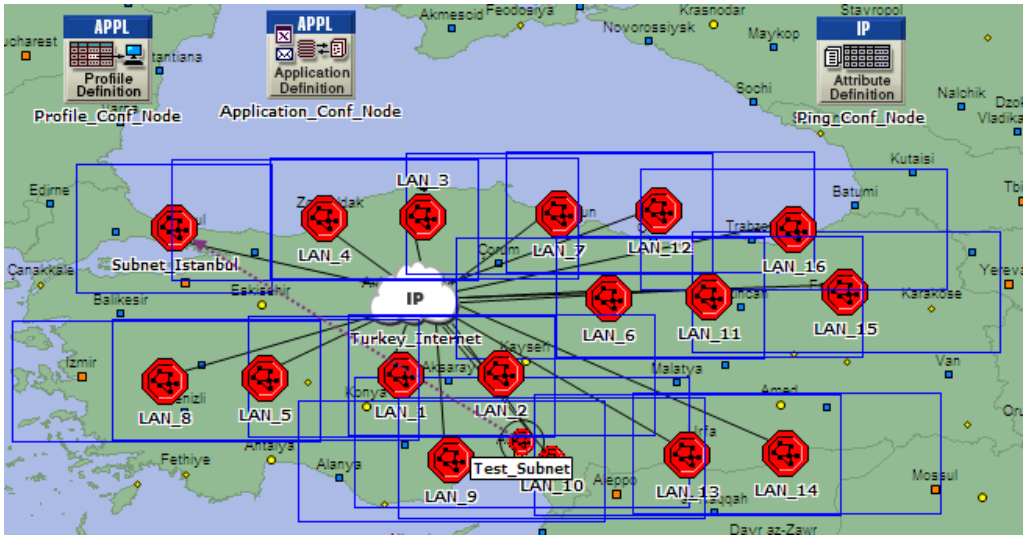
kabul edilmiştir. t değişkeni modeldeki t. adımı göstermektedir. Kodun en başında başlangıç durumunu ifade eden $\delta_0(0)$ olasılığına 1 atanmıştır çünkü model her zaman başlangıç durumu ile başlamak zorundadır. $ptr_t(j)$ ile ifade edilen işaretçi her j durumu için t anındaki, en yüksek olasılıklı durumu saklamaktadır. Benzer şekilde q işaretçisi ile de geri dönüş adımında en yüksek olasılıklı durum dizisinin tamamı elde edilmektedir. Yani q bize modelden elde edilebilecek en muhtemel durum dizisini verecektir. p^* olasılığı da q yolunun (durum dizisinin) modelden elde edilebilme olasılığını verecektir. Burada yer alan T değişkeni de son durumu ifade etmektedir.

Bu durumda Şekil 6.3 ile gösterilen model üzerinden elde edilebilecek en olası durum dizisi Şekilde kırmızı kesikli çizgiler kullanılarak gösterilmiştir. Buna göre en olası durum dizisi yani q:"NS1 NS2 NS4 NS5 YS4 YS5 YS6 YS7 YS8 YS9 YS10 YS11 YS12 YS13" şeklinde olacaktır. Bu durum dizisine karşılık gelen çıktılar dizisi de "N N B N B I I I I I I I I I I" olacaktır. Elde edilen çıktı dizisinde "I" yani boş olarak belirlenen zaman dilimleri senkronizasyonun yapılacağı zamanlara karşılık gelmektedir.

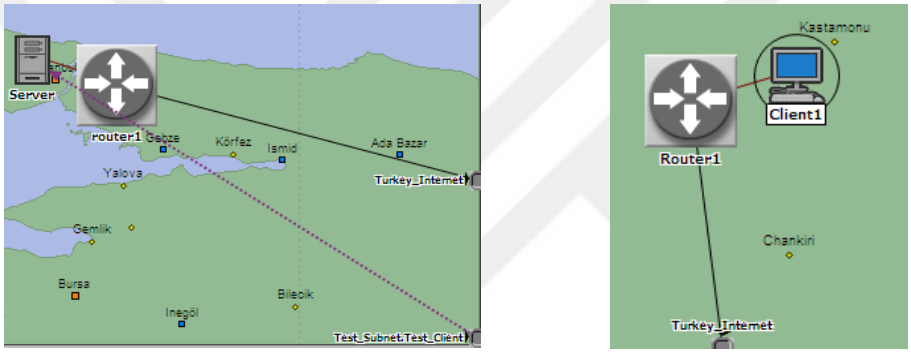
6.3. Yöntemin Test Edilmesi

6.3.1. Ağ trafik modeli ve benzetim koşulları

Çalışma kapsamında önerilen yöntemin test edilebilmesi için Şekil 6.1'de gösterilen örnek mimarinin benzeri Şekil 6.5'te gösterildiği gibi OPNET (URL-2, 2018) ortamında modellenmiştir. Şekilde 16 farklı kampüse ait LAN mimarisi internet bulutu aracılığıyla ana sunucuya bağlanmıştır. Her LAN'da o ağa ait verileri saklamak için Şekil 6.6(b)'de gösterildiği gibi bir yerel sunucu konumlandırılmıştır. Yerel sunucular ve ana sunucu internet bulutuna PPP-DS3 kablo kullanılarak bağlanmıştır. Bu bağlantı türünün erişim hızı 44,7 Mbps'dir. İnternet bulutundaki gecikme ve kayıp oranı standart WAN koşullarına (Isobe ve diğ., 2011) uygun olarak sırasıyla 0,001 saniye ve 0,1% belirlenmiştir.



Şekil 6.5. OPNET üzerinde oluşturulan 16 LAN içeren örnek mimari



(a) (b)

Şekil 6.6. Ana sunucu ve örnek bir istemcinin bağlantıları

Şekil 6.2'de akademik bir ağın kullanımı dört parçaya ayrılmıştır ve bu parçalar 1' den 8'e kadar numaralandırılmıştır. Şekildeki 1,2,3 ve 4 ile numaralandırılan bölgeler ilgili zaman dilimlerindeki gelen trafiği, 5,6,7 ve 8 ile numaralandırılan bölgeler ise ilgili zaman dilimlerindeki giden trafiği göstermektedir. Bu zaman dilimlerine ait istatistikler Tablo 6.4 (Garsva ve diğ., 2014) ve Tablo 6.5'de (Garsva ve diğ., 2014) verilmiştir. Şekil 6.5'te yer alan her kampüste gün içerisinde oluşacak yoğunluğu modellemek amacıyla LAN'lardan internet bulutuna erişim olan kablolara çift yönlü olarak arka plan trafiği eklenmesi gerekmektedir. Arka plan trafiği Tablo 6.4 ve Tablo 6.5'e uygun olacak şekilde üç farklı oranda eklenmiştir.

Bunlardan ilki Şekil 6.2'de 2 ve 6 ile numaralandırılan zaman dilimlerinde uygulanan yüksek yük durumudur. Burada linke, linkin kapasitesini neredeyse dolduracak kadar bir trafik enjekte edilmiştir. İkinci durum düşük yük durumudur.

Bu durum Şekil 6.2'de 1,5,3 ve 7 ile numaralandırılan zaman dilimlerine karşılık gelmektedir. Bu durumda da linkin doluluk oranı kapasitesinin yaklaşık yarısı kadar olacak şekilde ayarlanmıştır. Üçüncü durum sıfır yük olarak ifade edilen durumdur. Bu durum Şekil 6.2'de 4 ve 8 ile numaralandırılmış zaman dilimlerine karşılık gelmektedir. Bu zaman dilimlerinde link neredeyse tamamen boş durumdadır. Böylece gün içerisinde gönderici kuyruğunda oluşan doluluk, belirli bir eşik değerine ulaşıldıktan sonra kuyruğun boyutundaki azalma ve kuyruğun boş olduğu durum tam olarak modellenmiş olacaktır (Eylen ve Bazlamaçcı, 2015). Tablo 6.4 ve Tablo 6.5'te verilen istatistikler incelendiğinde de yoğunluğa geçiş zamanında arka plan trafiğinin linkin yarısını dolduracak kadar olduğu ve yoğunluk durumunda da trafiğin neredeyse linkin tamamını dolduracak kadar olduğu görülmektedir.

Tablo 6.4. Bir kampüs ağının dört farklı zaman aralığı için trafik değerleri

Gelen	**Paket, %**	**Zaman**	**Uzunluk(Saat)**	**Paket, %**	**Giden**
1	6,96	7:00-10:00	3	4,36	5
2	36,58	10:00-16:30	6,30	23,05	6
3	16,48	16:30-22:00	5,30	10,78	7
4	0,79	22:00-7:00	9	1,01	8

Tablo 6.5. Bir kampüs ağında gelen ve giden paketlere ait istatistikler

Parametreler	**Gelen**	**Giden**
Toplam Trafik, $B \times 10^9$	1136,158	225,358
Toplam Akış	25214174	23948612
Toplam Paket	1304	650
Bir Akıştaki Ortalama Paket Sayısı	444	440
Ortalama Paket Boyutu, B	3101	1174

Eklenen arka plan trafiği ile her bir LAN'daki iç yoğunluk sebebiyle yerel sunuculardan gönderilen senkronizasyon paketlerinde oluşacak gecikmeler modellenmiştir. Gecikmelerin rastgele olması açısından Poisson dağılımı kullanılmıştır. Şekil 6.7'de matlab'da üç farklı durum için oluşturulan arka plan trafiği dosyalarına ait örnek matlab kodu gösterilmiştir.

Yüksek Yük Durumu	Düşük Yük Durumu	Sıfır Yük Durumu

x=0;
A=poissrnd(50000000,1,60);
yaz=fopen('dosya','w+');
fprintf(yaz,'seconds bits/second\n');
 	
for i=1:60
t = num2str(A(i));
t2=num2str(x);
t3='\t';
t5='\n';
t4 = strcat(t2,t3,t,t5);
fprintf(yaz, t4);
x=x+60;
end
	x=0;
	B= poissrnd(900000,1,60);
	yaz=fopen('dosya','w+');
	fprintf(yaz,'seconds bits/second\n');
	
	for i=1:60
	t = num2str(B(i));
	t2=num2str(x);
	t3='\t';
	t5='\n';
	t4 = strcat(t2,t3,t,t5);
	fprintf(yaz, t4);
	x=x+60;
	end
		x=0;
		C=poissrnd(30000000,1,60);
		yaz=fopen('dosya','w+');
		fprintf(yaz,'seconds bits/second\n');
		
		for i=1:60
		t = num2str(C(i));
		t2=num2str(x);
		t3='\t';
		t5='\n';
		t4 = strcat(t2,t3,t,t5);
		fprintf(yaz, t4);
		x=x+60;
		end

Şekil 6.7. Eklenen arka plan trafiğine benzer Matlab kodu

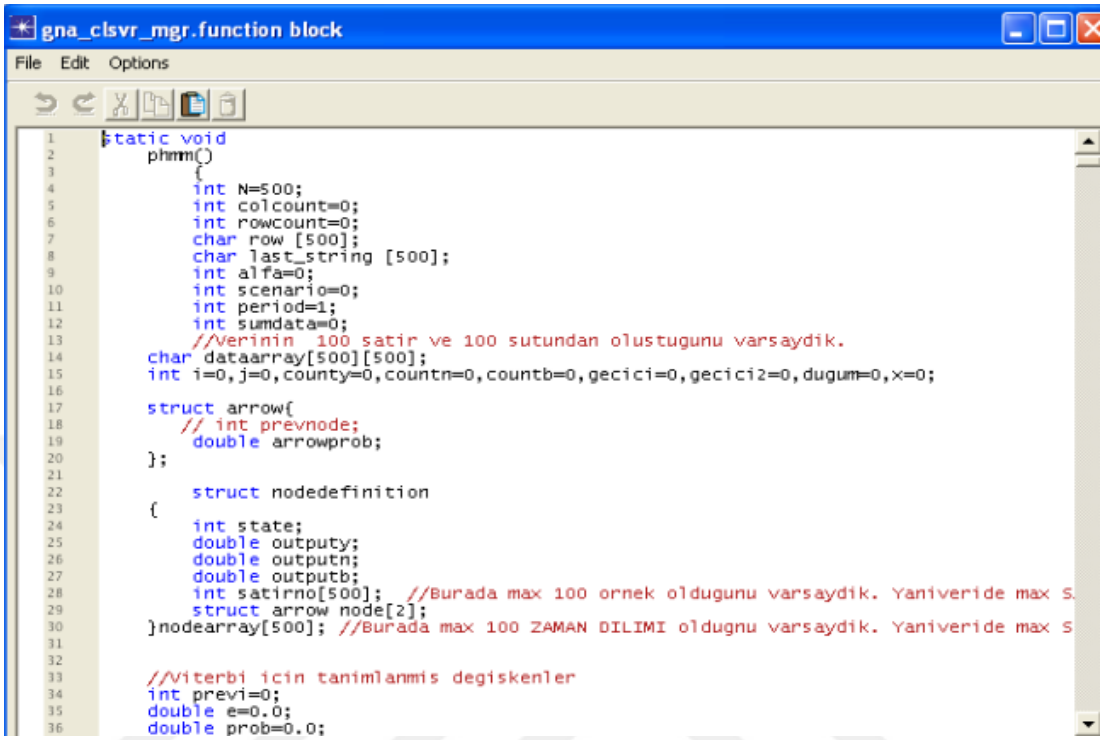
Tablo 6.6. Gönderilen senkronizasyon paketlerine ait detaylı bilgi

Zaman Dilimi	Model	Olasılık Yoğunluk Fonksiyonu	Parametreler
Yüksek Yük	Pareto	$f(t) = w\, y^{w} t^{-w-1}$	w:1,1 y:[1048576] byte
Düşük Yük	Pareto	$f(t) = w\, y^{w} t^{-w-1}$	w:1,1 y:[209715] byte
Sıfır Yük	Weibull	$f(t) = \frac{w}{y}\left(\frac{t}{y}\right)^{w-1} e^{-(t/y)^{w}}$	w: 0,07897 y : 0,014

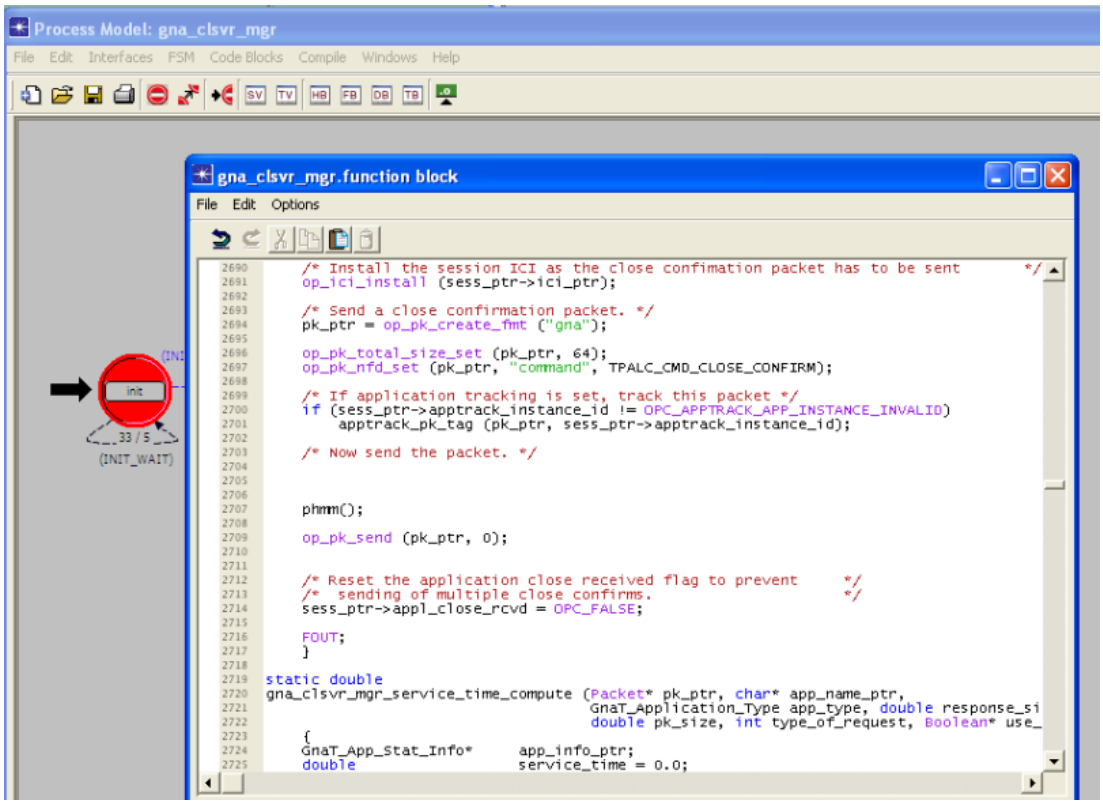
Tablo 6.6'da yerel sunuculardan eş zamanlı ve bir dakikalık periyotlarla gönderilen senkronizasyon paketlerinin üretilmesi için kullanılan dağılımlar verilmiştir. Kullanılan modellerin belirtilen zaman dilimlerine uygun olduğu Garsva ve diğ. (2014) tarafından gösterilmiştir. Farklı ağ yoğunlukları için kullanılan dağılımların uygun olduğu Aduba ve Sadiku (2002) tarafından yapılan çalışmada da gösterilmiştir.

Yukarıda belirtilen dağılımlara kullanılarak arka plan trafiği ve senkronizasyon paketleri modellendikten sonra önerilen PHMM Poll yöntemine ait C kodu, OPNET içerisinde Şekil 6.5'te gösterilen 16 yerel sunucunun uygulama katmanında çalışacak şekilde ilgili katmanın fonksiyon bloğuna Şekil 6.8'de gösterildiği gibi aktarılmıştır. Yazılan phmm() fonksiyonu Şekil 6.9'da gösterildiği gibi uygulama katmanından paket gönderimini yapan op_pk_send(pk_prt,o) fonksiyonundan önce çağırılmıştır. Böylece paket gönderimleri yapılmadan uygulama katmanından her bir yerel

sunucunun paket gönderim zamanlaması yapılmıştır. Modele göre uygun olan zaman dilimlerinde paket gönderimini planlayacak csv uzantılı dosyalar oluşturulmuştur.



Şekil 6.8. Geliştirilen modele ait kodun OPNET içerisindeki görünümü

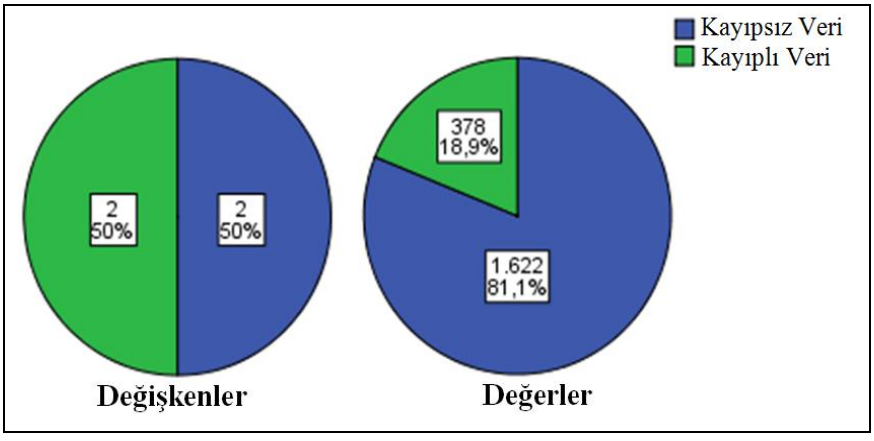


Şekil 6.9. PHMM Poll fonksiyonunun OPNET içerisinde çalıştırılması

6.3.2. Günlük verinin kümelenmesi

Şekil 6.5'te Test_Subnet olarak gösterilen LAN' da yer alan yerel sunucu ağa ait eğitim verisi toplamak amacıyla kullanılmıştır. Bir gün boyunca bu yerel sunucudan gönderilen senkronizasyon paketlerinden elde edilen yük, gecikme, cevap zamanı ve yeniden iletim sayısı verileri FCM algoritması kullanılarak boş, normal ve yoğun kümelerini elde etmek amacıyla kümelenmiştir. (Çalışma kapsamında modellenen mimari Şekil 6.2'de de gösterildiği gibi üç farklı yoğunluk geçişine sahiptir. Bu nedenle kümeleme işlemi için küme sayısı modellenen ağa uygun olarak üç seçilmiştir. Farklı ağ mimarileri için uygun küme sayılarının ağa özgü olarak belirlenmesi gerekecektir.)

Şekil 6.10'da bir günlük elde edilen verinin içeriğine ait kayıp değer oranının özellik ve değerler bakımından grafiksel gösterimi verilmiştir. Şekilde yer alan birinci grafik kayıpların hangi özelliklerde bulunduğunu göstermektedir. Kayıp verilerin büyük çoğunluğu yeniden iletim sayısı ve gecikme parametrelerinde meydana gelmektedir. Veri setinde dört özellik bulunduğu düşünülürse kayıplar bu özelliklerin ikisinde meydana geldiğinden, özellikler bakımından kayıp oranı ilk grafikte %50 olarak gösterilmiştir. İkinci grafik ise değer bazında kayıp oranını göstermektedir. Yani verinin sekiz saatlik, dört yüz seksen satırdan oluştuğunu düşünürsek, her satırda dört özellik bulunduğundan toplam değer sayısı (480×4) 1920 olacaktır. Bu değerlerin %18,9'luk bir bölümü kayıptır. Bu nedenle kümeleme işlemi öncesinde ilk olarak kayıp değerlerin elde edilmesi gerekmektedir.



Şekil 6.10. Günlük veri seti içerisindeki kayıp veri istatistikleri

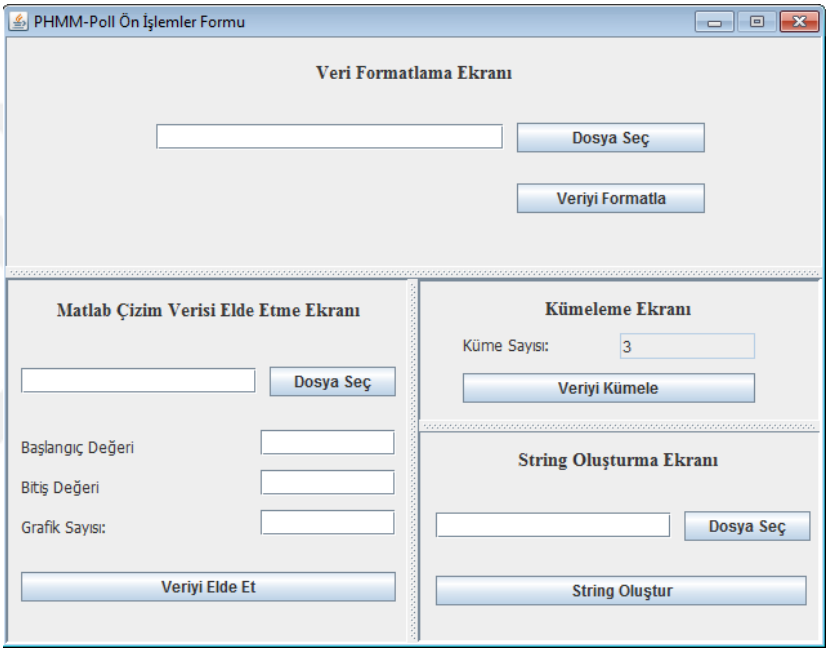
Çalışma kapsamında kayıp değerleri elde etmek amacıyla k-NN yöntemi kullanılmıştır ancak Tablo 6.7'de kullanılabilecek farklı yöntemlerin (EM, Doğrusal Regresyon, k-NN) karşılaştırılması da yapılmıştır. Tablo incelendiğinde aynı ağ koşulları altında her üç yönteminde birbirine çok yakın hata oranları ile sonuç verdiği görülmektedir. Farklı mimariler için sonuçlar değerlendirildiğinde, iki yerel sunucu içeren mimaride k-NN yöntemi (beş ya da on komşu değeri) kullanılarak, dört yerel sunucu içeren mimaride yine k-NN yöntemi (on komşu değeri) kullanılarak, sekiz sunucu içeren mimaride doğrusal regresyon yöntemi kullanılarak ve on altı sunucu içeren mimaride k-NN yöntemi (beş komşu değeri) kullanılarak en düşük hatayla kayıp verilerin elde edildiği görülmektedir. Genel olarak farklı mimarilerde k-NN yönteminin daha başarılı olması ve daha önce benzer ağ veri setleri üzerinde yapılan karşılaştırma çalışmasında da (Kaya Gülağız ve diğ., 2016) komşu değeri beş olduğunda hata değerinin düşük olması sebebiyle tez çalışmasında kayıp verileri elde etmek amacıyla k-NN yöntemi tercih edilmiştir. Komşu değeri de beş olarak belirlenmiştir ve tüm özelliklere ait kayıp değerler elde edilmiştir.

Tablo 6.7. Kayıp değerleri elde etmek amacıyla kullanılabilecek yöntemlerin karşılaştırılması

2 Sunucu			**4 Sunucu**		
Yöntem		MSE	Yöntem		MSE
EM		41096, 9340	EM		34855,8966
k-NN	k:2	41096,9125	k-NN	k:2	34855,8881
	k:5	**41096,8957**		k:5	34855,8795
	k:10	**41096,8957**		**k:10**	**34855,8671**
Doğrusal Regresyon		41096,9117	Doğrusal Regresyon		34855,8746
8 Sunucu			**16 Sunucu**		
Yöntem		MSE	Yöntem		MSE
EM		161619,4074	EM		254068,5132
k-NN	k:2	161619,3370	k-NN	k:2	254068,1051
	k:5	**161619,3210**		**k:5**	**254068,0433**
	k:10	161619,3226		k:10	254068,0518
Doğrusal Regresyon		**161619,2715**	Doğrusal Regresyon		254068,9940

Daha sonra özellikler üzerinde normalleştirme işlemi gerçekleştirilmiştir. Günlük elde edilen veri seti içerisindeki özelliklerden birincisi olan yük özelliği, diğer özelliklere göre daha büyük değerler almaktadır. Bu nedenle normalleştirme işlemi

yapılmaması durumunda sonucu büyük oranda belirleyen özellik birinci özellik olacaktır. Tüm özelliklerin sonuca katkısını sağlamak amacıyla özellikler 0-1 aralığında normalleştirilmiştir. Veri üzerindeki ön işlemleri, kümeleme işlemini ve grafik çizimlerini kolaylaştırmak amacıyla Şekil 6.11'de gösterilen form ekranı tasarlanmıştır. Form ekranında yer alan "Veri Formatlama Ekranı" bölümü veri kümelenmeden önce yapılacak ön işlemlerin tamamını gerçekleştirmektedir. Bu bölümün altında yer alan "Kümeleme Ekranı" ise verilerin kümelenmesi işlemini gerçekleştirmektedir.



Şekil 6.11. PHMM Poll ön işlemler formu

Tez çalışmasında kümeleme işlemi için FCM algoritması kullanılmıştır ancak FCM algoritmasına ek olarak K-Ortalamalar ve İteratif K-Ortalamalar yöntemleri kullanılarak da kümeleme işlemleri gerçekleştirilmiştir. Elde edilen sonuçlar Tablo 6.8'de gösterilmiştir. Kümeleme işlemlerini değerlendirmek amacıyla dört farklı ölçüt kullanılmıştır. Bunlar; küme merkezleri benzerliklerinin toplamı, ortalama küme içi benzerlikleri toplamı, SSE ve işlem süresi ölçütleridir. Kümeleme işleminin başarılı olabilmesi için küme merkezlerinin benzerliklerinin olabildiğince düşük, kümeler içi benzerliklerin ise olabildiğince yüksek olması gerekmektedir. Bu durum göz önünde bulundurularak yöntemler farklı sayıda yerel sunucu içeren mimariler altında karşılaştırıldığında küme merkezleri benzerliklerinin en düşük olduğu yöntemin, FCM olduğu görülmektedir.

Tablo 6.8. Kullanılan kümeleme yöntemlerinin karşılaştırılması

	Performans Metrikleri	**2 Yerel Sunucu**	**4 Yerel Sunucu**	**8 Yerel Sunucu**	**16 Yerel Sunucu**
K-Ortalamalar	Merkez Benzerlikleri Toplamı	495,652082	494,490703	492,967292	490,147238
	Ortalama Küme İçi Benzerlikler Toplamı	490,891945	488,605042	485,744104	480,135893
	Karesel Hatalar Toplamı	38,862109	28,659601	38,561916	47,340986
	İşlem Süresi	0,092076 sn.	0,136711 sn.	0,126519 sn.	0,133056 sn.
İteratif K-Ortalamalar	Merkez Benzerlikleri Toplamı	496,977628	494,495249	492,967292	490,147238
	Ortalama Küme İçi Benzerlikler Toplamı	493,074836	488,6038169	485,744104	480,135893
	Karesel Hatalar Toplamı	32,612065	28,661217	38,561916	47,340986
	İşlem Süresi	5,900728 sn.	6,351052 sn.	6,248692 sn.	6,343927 sn.
Bulanık C-Ortalamalar	Merkez Benzerlikleri Toplamı	466,370915	494,235723	489,723510	492,235751
	Ortalama Küme İçi Benzerlikler Toplamı	470,536663	489,004301	483,807572	485,230261
	Karesel Hatalar Toplamı	43,265943	28,243302	36,506945	47,283295
	İşlem Süresi	0,146 sn.	0,169 sn.	0,158 sn.	0,230 sn.

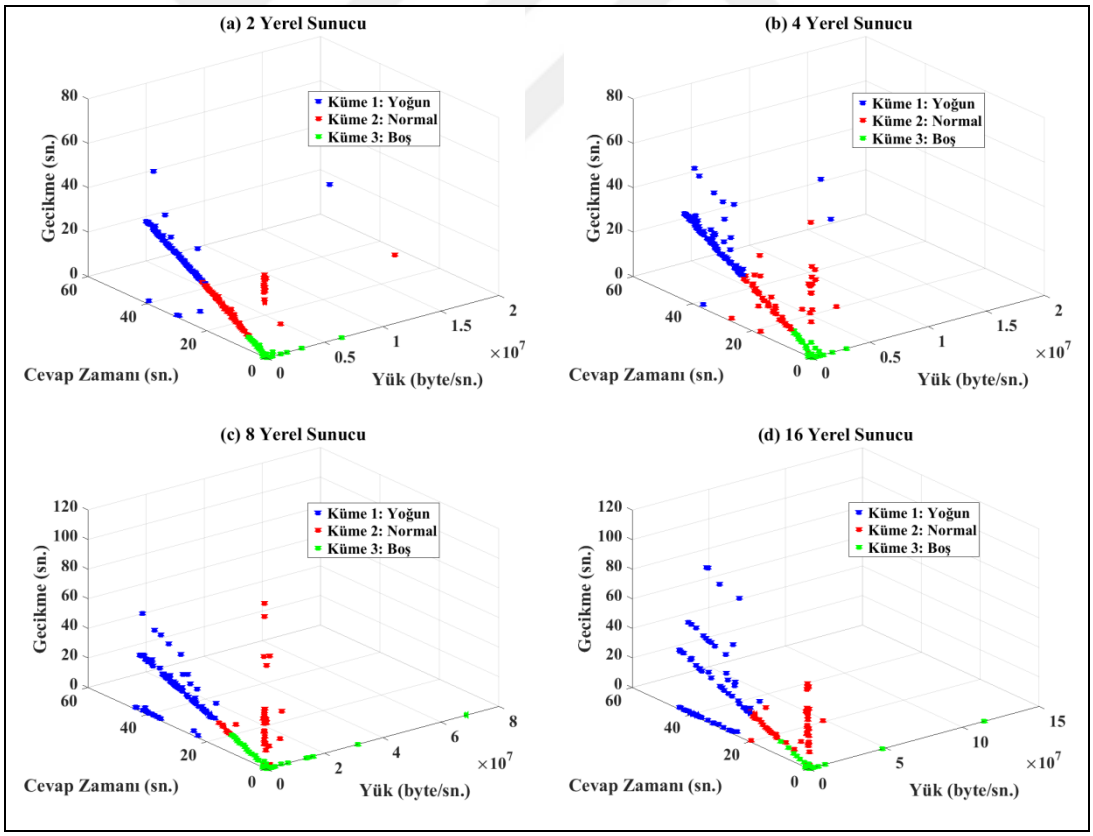
Küme içi benzerlikler bakımından yöntemler karşılaştırıldığında en yüksek değerlerin iki ve sekiz yerel sunucu içeren mimarilerde İteratif K-Ortalamalar yöntemiyle, dört ve on altı sunucu içeren mimarilerde ise FCM yöntemiyle elde edildiği görülmektedir. Karesel hatalar toplamı bakımından yöntemlerin genel hata oranları karşılaştırılırsa iki yerel sunucu içeren mimari dışında tüm mimarilerde en düşük hata değerine sahip yöntemin FCM yöntemi olduğu, iki yerel sunucu içeren mimari de ise İteratif K-Ortalamalar yönteminin daha düşük hata değeri ile işlem yaptığı görülmektedir.

Genel olarak yöntemler değerlendirildiğinde FCM algoritmasının diğer yöntemlerden daha düşük hata değerleriyle kümeleme işlemini gerçekleştirdiği görülmektedir. Aynı zamanda İteratif K-Ortalamalar yöntemi, K-Ortalamalar yönteminden daha başarılıdır. Yöntemler işlem süresi bakımından karşılaştırıldığında K-Ortalamalar yönteminin en düşük işlem süresi ile sonuçları elde ettiği görülmektedir. Yine işlem süresi bakımından İteratif K-Ortalamalar yöntemi verilen aralıktaki tüm küme değerleri için kümeleme işlemini tekrarlandığından diğer iki yöntemden yaklaşık beş kat daha fazla işlem süresine ihtiyaç duymuştur. FCM algoritması hem düşük hata değerleriyle işlem yapması hem de işlem süresi bakımından K-Ortalamalar yöntemiyle yakın zamanlarda sonuç üretebilmesi nedeniyle tez çalışması kapsamında tercih edilmiştir. Bunlara ek olarak ani yoğunluk değişimlerinin sık olduğu ağlarda günlük olarak toplanacak verilerde yüksek çeşitlilik olacaktır. Bu durumda veri setinde bulunan bazı veriler birbirine çok yakın değerler alabilir. Bu nedenle bu verileri birden fazla kümeye atamak gerekebilir. Bu durum da göz önünde bulundurulduğunda ağ verilerin kümelenmesi sürecinde FCM algoritması gibi bulanık bir algoritmanın tercih edilmesi daha doğru olacaktır.

Şekil 6.5'te 16 farklı LAN bulunmaktadır ancak sunucu sayısının kümeleme işlemi ve ağın yoğunluğu üzerindeki etkisini gözlemleyebilmek için LAN sayısının 2,4 ve 8 olduğu mimariler içinde sonuçlar incelenmiştir. Şekil 6.12(a)'daki genel mimaride 2 LAN (dolayısıyla 2 yerel sunucu) ve bir ana sunucu olan, Şekil 6.12(b)'de 4 LAN ve bir ana sunucu olan, Şekil 6.12(c)'de 8 LAN ve bir ana sunucu olan, Şekil 6.12(d)'de de 16 LAN ve bir ana sunucu olan mimariler üzerinden elde edilen bir günlük veriler ile FCM algoritması kullanılarak elde edilen kümeleme sonuçları gösterilmiştir.

Şekillerde yoğun kümesi mavi, normal kümesi kırmızı ve boş kümesi yeşil renkli olarak gösterilmiştir.

Sonuçlar gösterilirken kümeleme işleminde kullanılan 4 parametreden üçü (yük, gecikme ve cevap zamanı parametreleri) kullanılmıştır. Yeniden iletim sayısı değeri ağdaki gecikme ve yük değerlerine paralel olarak değiştiğinden grafiklerde kullanılmamıştır. Kümeleme sonuçları incelendiğinde düşük yük, gecikme ve cevap zamanı değerleri boş kümesinde, yoğunluğun ve paketteki gecikmelerin yüksek olduğu değerler ise normal ve yoğun kümelerinde konumlanmıştır. Sunucu sayısı az iken kümelerin sınırları daha nettir ancak sunucu sayısı arttığında sınırlar biraz daha iç içe geçmiştir. Yerel sunucuların sayısı arttığında aynı anda gönderilen paketlerin sayısı da arttığından parametrelerin aldığı değerlerde de çeşitlilik olmuştur ve kümelerin sınırları genişlemiştir. Ancak her durumda FCM algoritması kümeleme işlemini doğru olarak gerçekleştirmiştir.



Şekil 6.12. FCM algoritması kümeleme sonuçları

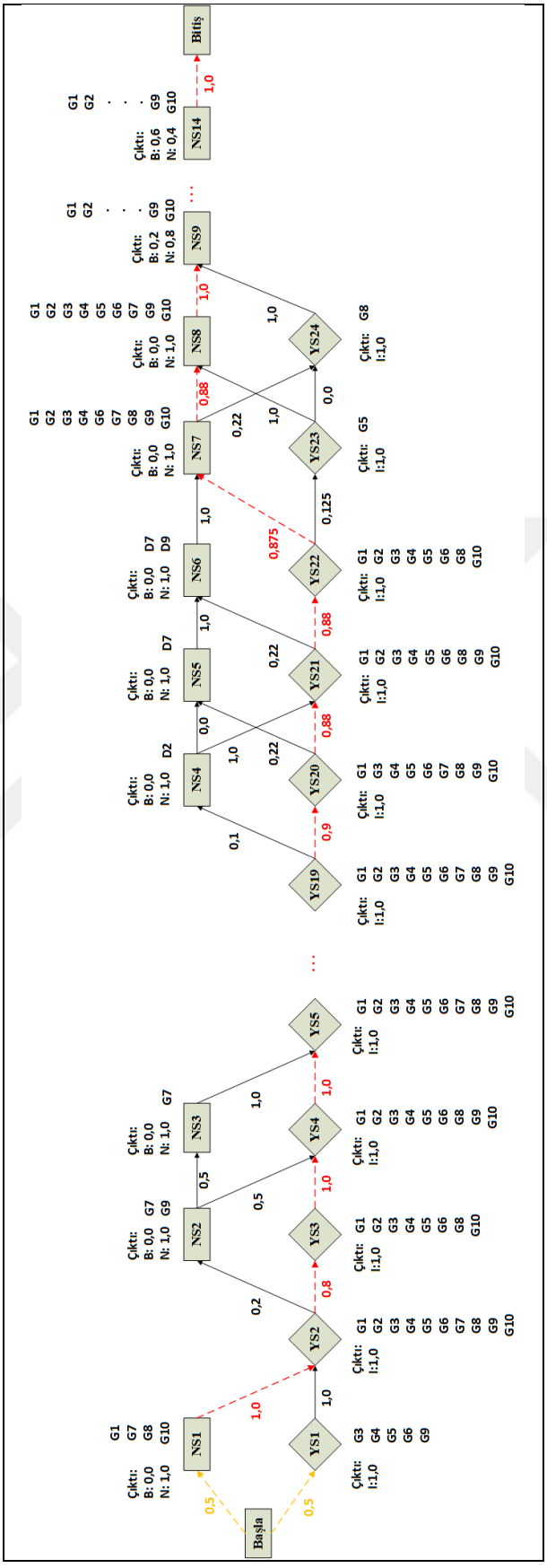
6.3.3. Farklı senaryolar altında yöntemin değerlendirilmesi

Kümeleme işlemi tamamlandıktan sonra gün içerisinde oluşabilecek farklı senaryolar üzerinden ağın genel örüntüsü elde edilmiştir. Bu amaçla kampüs ağını tam olarak modelleyebilmek için ağın günlük davranışı üç farklı senaryo üzerinden değerlendirilmiştir. Bu senaryolardan birincisi Şekil 6.2'de 1 ve 5 ile numaralandırılan, ağın boş olduğu durumdan yoğunluğa geçiş durumudur. İkinci senaryo Şekil 6.2'de 2 ve 6 ile numaralandırılan, ağda sürekli yoğunluğun olduğu durumdur. Üçüncü senaryo ise Şekil 6.2'de 3 ve 7 ile numaralandırılan, yoğunluktan ağın boş olduğu duruma geçiş senaryosudur. Şekil 6.2'de 4 ve 8 ile numaralandırılan zaman dilimlerinde ağ tamamen boştur. Bu zaman dilimlerinde modelin davranışı CouchDB Poll mekanizması ile aynı olacağından modellenmesine gerek yoktur. Bu nedenle belirtilen zaman dilimleri ayrı bir senaryo olarak ele alınmamıştır. Bu üç senaryo için CouchDB Poll yöntemi ve PHMM Poll yöntemi ana sunucuda oluşan yoğunluk, LAN_2'de bulunan yerel sunucudan yapılan yeniden iletimlerin sayısı ve ana sunucu ile yerel sunucular arasındaki linkte oluşan gecikme parametrelerine göre karşılaştırılmıştır.

Tablo 6.9'da birinci senaryo için elde edilmiş on günlük verinin kümelenmiş ve hizalanmış hali gösterilmiştir. Tablo 6.9'deki verilere göre elde edilen model ve senaryoya ait sonuçlar sırasıyla Şekil 6.13 ve Şekil 6.14'te gösterilmiştir.

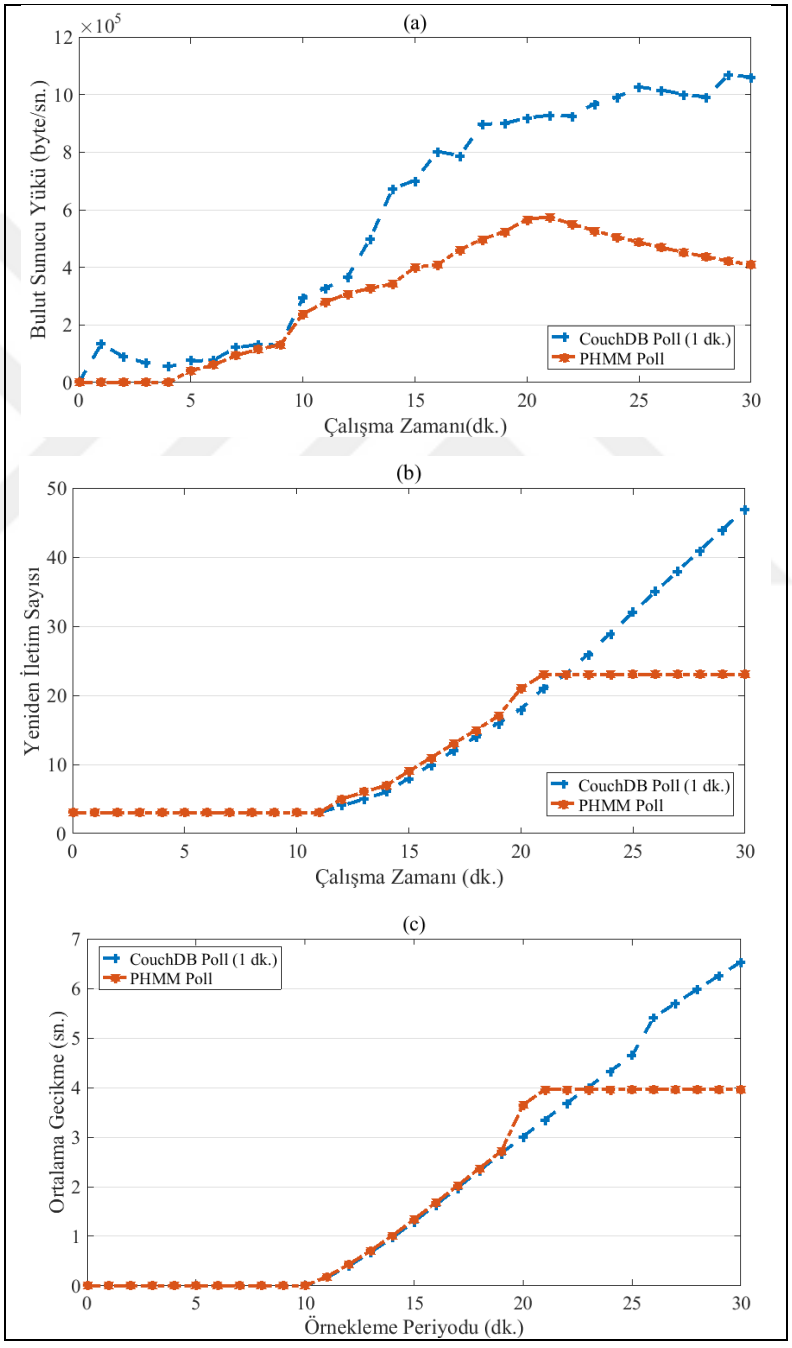
Tablo 6.9. Birinci senaryoya ait on günlük kümelenmiş veri örüntüleri

		Dakika														
		1.	**2.**	**3.**	**4.**	**5.**	**…**	**19.**	**20.**	**21.**	**22.**	**23.**	**24.**	**25.**	**…**	**30.**
Günler:	G1	N	I	I	I	I	…	I	I	I	N	N	N	N	…	B
	G2	N	I	I	I	I	…	N	I	I	N	N	N	N	…	B
	G3	I	I	I	I	I	…	I	I	I	N	N	N	B	…	B
	G4	I	I	I	I	I	…	I	I	I	N	N	B	N	…	N
	G5	I	I	I	I	I	…	I	I	I	I	N	N	N	…	N
	G6	I	I	I	I	I	…	I	I	I	N	N	N	N	…	N
	G7	N	I	N	N	I	…	I	N	N	N	N	N	N	…	B
	G8	N	I	I	I	I	…	I	I	I	N	I	B	N	…	N
	G9	I	I	N	I	I	…	I	I	N	N	N	N	B	…	B
	G10	N	I	I	I	I	…	I	I	I	N	N	N	N	…	B



Şekil 6.13. Birinci senaryoya göre elde edilmiş PHMM Poll modeli

Birinci senaryoda ağın boş olduğu durumdan yoğunluğa geçişi 30 dakikalık bir zaman dilimi için örneklenmiştir. Senaryoda ilk beş dakika için ağdaki paket oranı sıfır yük zamanına göre belirlenmiştir. Daha sonra sırasıyla düşük yük ve yüksek yük durumları uygulanmıştır. Modele göre örneklemenin yapıldığı 22. dakikaya kadar ağın durumu senkronizasyon yapmaya uygundur. 22. dakikadan sonra yoğunluk tespit edilmiştir ve 30. dakikaya kadar senkronizasyon yapılmayacaktır.



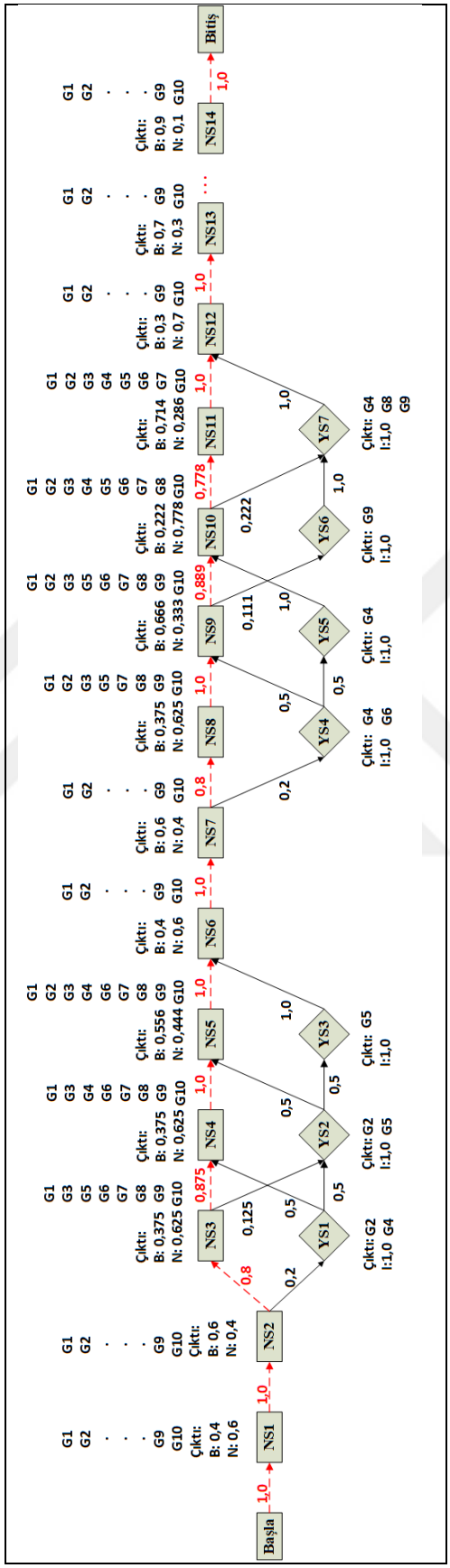
Şekil 6.14. Birinci senaryonun ana sunucu yükü (a), yeniden iletim sayısı (b) ve gecikme (c) değerlerine göre performans grafikleri.

Şekil 6.14'te yarım saatlik zaman dilimi için CouchDB Poll yöntemi ve PHMM Poll yöntemi karşılaştırılmıştır. PHMM Poll yöntemi ağda belirli bir tıkanıklık durumu oluşana kadar (22. Dakikaya kadar) klasik poll yöntemi ile benzer bir davranış göstermiştir. Ağdaki gecikme 4 saniye civarına geldiğinde yöntem ağın yoğun olduğunu fark etmiştir ve senkronizasyon işlemini bir sonraki boş duruma kadar ertelemiştir. Yöntemler birinci senaryo için karşılaştırıldığında PHMM Poll yönteminin yerel sunuculardan yapılacak gereksiz yeniden iletimleri yarı yarıya düşürdüğü gözlemlenmiştir. Aynı zamanda ana sunucunun yük dengesini belli bir seviyede tutarak hem LAN'daki kullanıcıların hem de ana sunucuya bağlı olan kullanıcıların verilere daha düşük gecikmelerle erişmesini sağlamıştır. Yeniden iletim sayısının düşük olması bant genişliğinin daha az tüketilmesini ve aynı hatta iletim yapan kullanıcıların daha düşük gecikmelerle iletim yapmasını sağlamıştır.

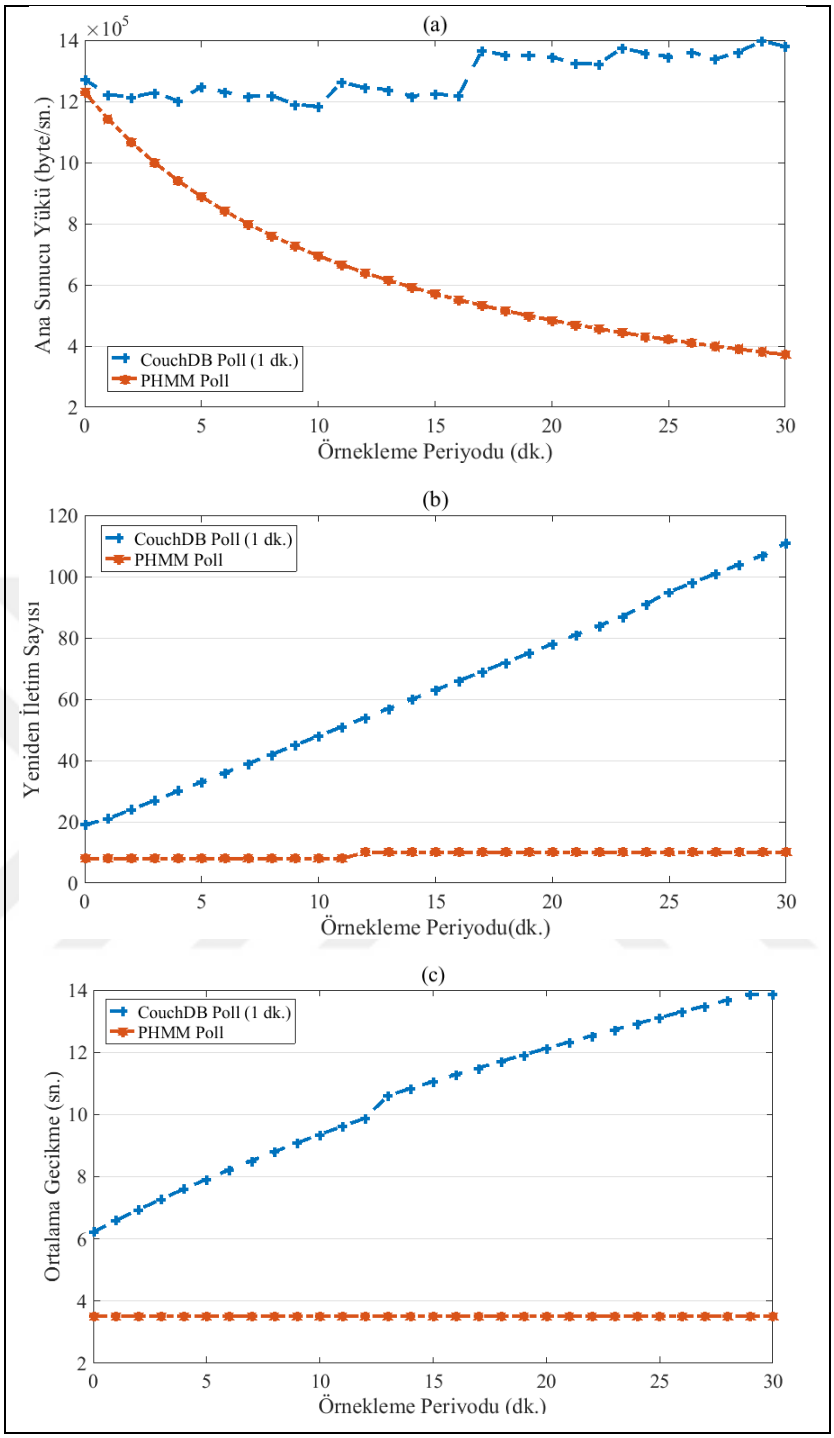
Tablo 6.10'da ikinci senaryo için elde edilmiş on günlük verinin kümelenmiş ve hizalanmış hali gösterilmiştir. Burada ağın sürekli yoğun olduğu durum 30 dakikalık bir zaman dilimi için örneklenmiştir. Simülasyon boyunca paket iletimi sürekli yüksek yük durumuna göre belirlenmiştir. Tablo 6.10'daki verilere göre elde edilen model ve simülasyon sonuçları sırasıyla Şekil 6.15 ve 6.16'da gösterilmiştir. Modele göre örneklemenin yapıldığı 30 dakika boyunca ağ yoğun durumdadır ve senkronizasyon yapılmayacaktır.

Tablo 6.10. İkinci senaryoya ait on günlük kümelenmiş veri örüntüleri

		Dakika														
		1.	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**	**10.**	**11.**	**12.**	**13.**	**…**	**30.**
Günler:	G1	N	B	N	N	B	B	N	N	B	N	N	B	B	**…**	B
	G2	B	N	I	I	B	N	B	N	N	N	B	N	B	…	B
	G3	B	N	N	N	N	B	B	B	B	B	N	N	N	…	N
	G4	N	B	I	N	N	N	N	I	I	N	I	N	N	…	B
	G5	N	N	N	I	I	N	N	N	B	N	B	N	N	…	B
	G6	N	B	B	B	N	B	B	I	N	B	B	N	B	…	B
	G7	B	B	N	N	B	N	B	N	B	N	B	B	B	…	B
	G8	N	N	B	N	B	N	B	B	B	N	I	N	B	…	B
	G9	N	B	B	N	N	B	N	B	N	I	I	N	B	…	B
	G10	B	B	N	N	B	N	B	N	B	N	B	B	B	…	B



Şekil 6.15. İkinci senaryoya göre elde edilmiş PHMM Poll modeli



Şekil 6.16. İkinci senaryonun ana sunucu yükü (a), yeniden iletim sayısı (b) ve gecikme (c) değerlerine göre performans grafikleri

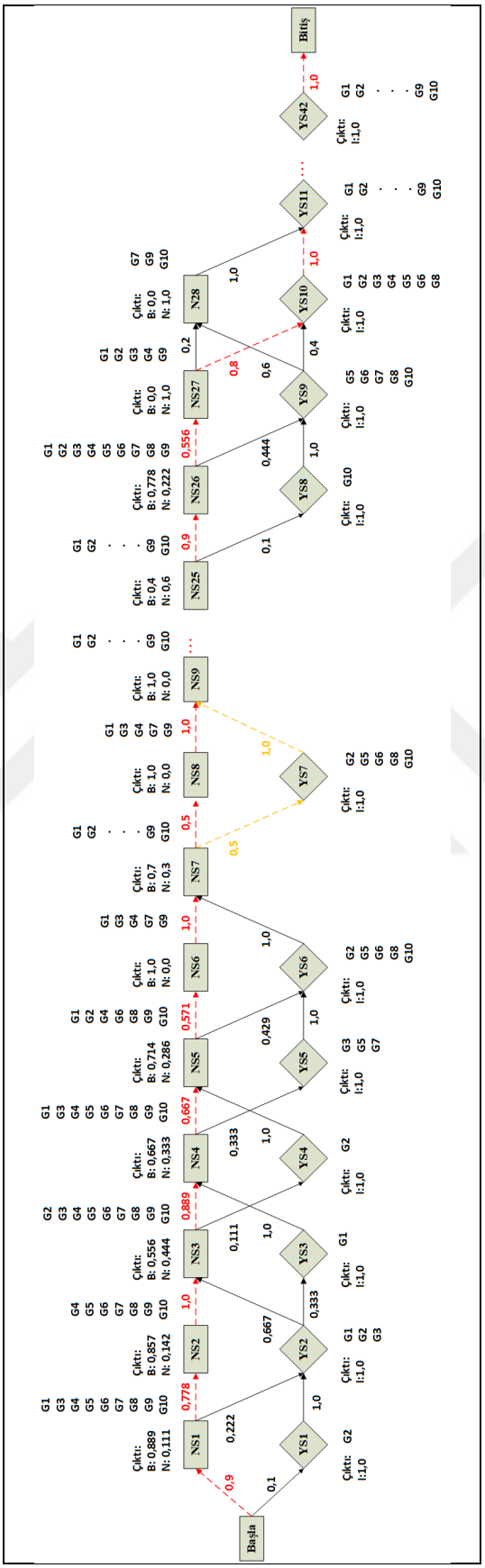
İkinci senaryoya ait sonuçlar incelendiğinde klasik CouchDB yönteminde ana sunucu ve yerel sunucular arasında iletilen paketlerdeki gecikme Şekil 6.16(c)'de de gösterildiği gibi sürekli artmaktadır. Gecikmeye bağlı olarak yeniden iletimler de artmaktadır. PHMM Poll yöntemi ise senkronizasyon işlemini yoğunluk süresince

gerçekleştirmediğinden hem yeniden iletimlerin hem de linkteki gecikmenin belirli bir seviyede kalmasını sağlamıştır. Sunucuya da mevcut yoğunluk öncesi kuyrukta bekleyen işlemleri tamamlaması için zaman tanımıştır ve ana sunucunun yoğunluğunun artmasını önlemiştir.

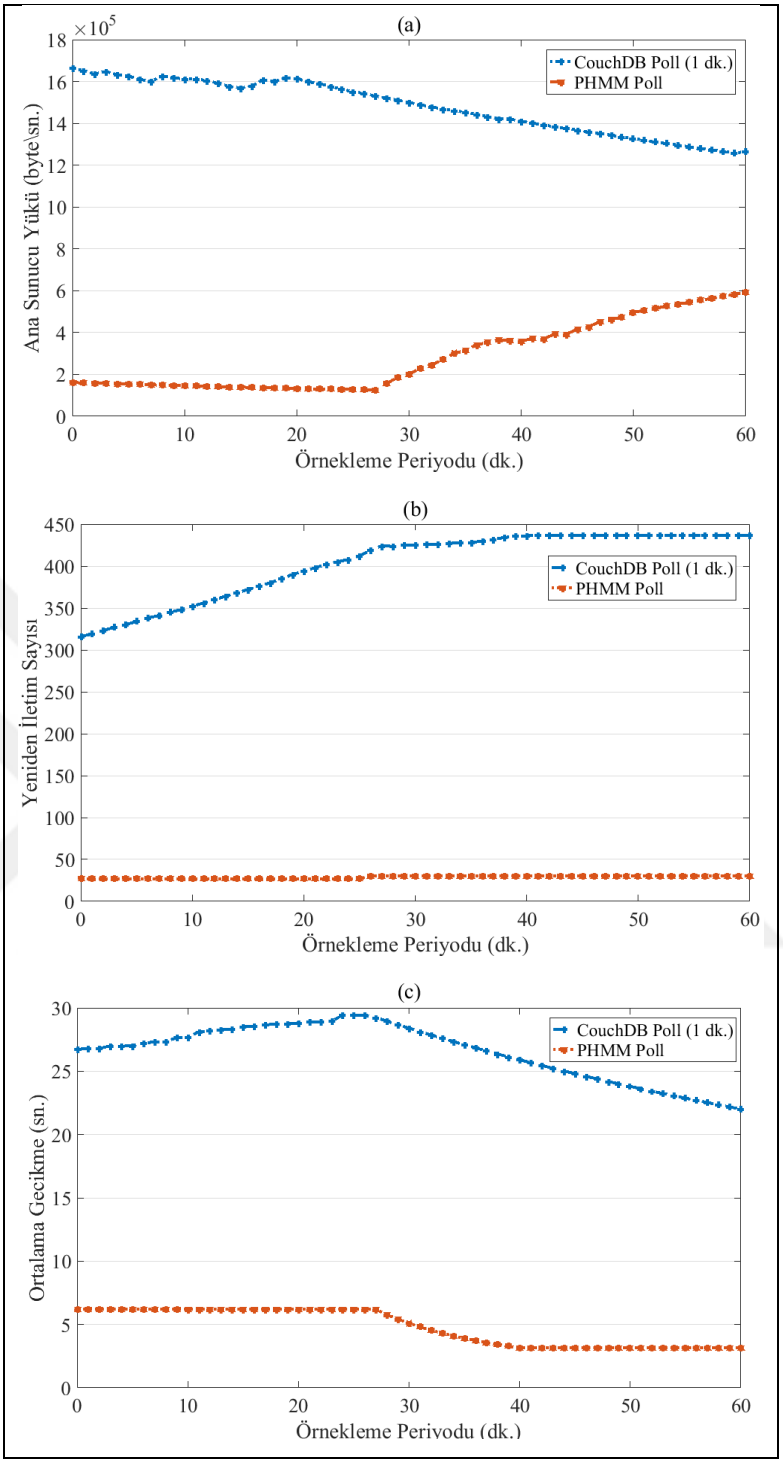
Üçüncü senaryo için elde edilmiş on günlük verinin kümelenmiş ve hizalanmış hali Tablo 6.11'de gösterilmiştir. Verilere göre elde edilmiş model de Şekil 6.17'de gösterilmiştir. Bu senaryo için örnekleme periyodu bir saat olarak belirlenmiştir. Senaryoda yoğunluk sebebiyle senkronizasyon için bekleyen verilerin boyutu diğer senaryolara göre daha büyük olacaktır. Yoğunluk sürecinin uzunluğuna göre senkronizasyon için bekleyen verilerin boyutu da artacaktır. Ağın boş zaman diliminde yapılan ilk gönderimde veri boyutu büyük olacağından sunucular arasındaki linkin yoğunluğu azalsa bile ana sunucunun yoğunluğu belli bir süre daha devam edecektir. Bu durumu tam olarak örnekleyebilmek için senaryonun süresi diğer senaryolara göre uzun tutulmuştur. Altmış dakikalık periyodun, ilk yirmi dakikası ağın yoğun olduğu durumu temsil etmektedir bu zaman diliminde paket gönderimi yüksek yük durumuna göre yapılmıştır. Daha sonra sırasıyla düşük yük ve sıfır yük durumları uygulanmıştır. Üçüncü senaryoya ait sonuçlar Şekil 6.18'de gösterilmiştir. Model yirmi yedinci dakikaya kadar ağın durumunu yoğun olarak tespit ettiğinden yirmi yedinci dakikadan sonra senkronizasyon işlemine başlamıştır. Yoğunluk durumu yirmi dakika olmasına rağmen sistemdeki yükün azalması ve modelin bu durumu fark etmesi için yedi dakikalık ek bir zamana ihtiyaç olmuştur.

Tablo 6.11. Üçüncü senaryoya ait on günlük kümelenmiş veri örüntüleri

		Dakika																
		1.	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**		**25.**	**26.**	**27.**	**28.**	**29.**		**60.**
Günler:	G1	B	I	I	B	B	B	N	N	B	…	N	B	N	I	I	…	I
	G2	I	I	B	I	B	I	B	I	B	…	N	B	N	I	I	…	I
	G3	B	I	B	B	I	I	B	B	B	…	N	B	N	I	I	…	I
	G4	B	B	N	B	B	B	N	B	B	…	N	B	N	I	I	…	I
	G5	B	B	N	B	I	I	B	I	B	…	N	N	I	I	I	…	I
	G6	B	B	N	B	B	B	N	I	B	…	B	N	I	I	I	…	I
	G7	N	N	B	N	I	I	B	B	B	…	B	B	I	N	I	…	I
	G8	B	B	B	N	B	I	B	I	B	…	N	B	I	I	I	…	I
	G9	B	B	B	N	N	B	B	B	B	…	B	B	N	N	I	…	I
	G10	B	B	N	B	N	I	B	I	B	…	B	I	I	N	I	…	I



Şekil 6.17. Üçüncü senaryoya göre elde edilmiş PHMM Poll modeli



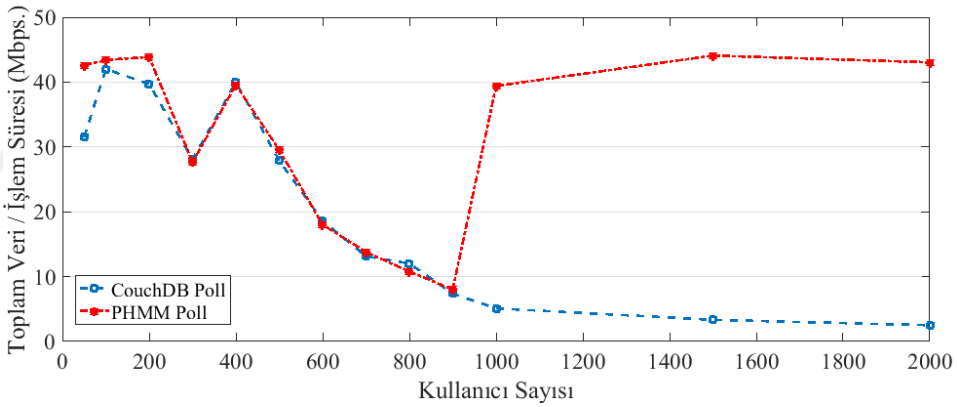
Şekil 6.18. Üçüncü senaryonun ana sunucu yükü (a), yeniden iletim sayısı (b) ve gecikme (c) değerlerine göre performans grafikleri

Üçüncü senaryo için iki yöntem karşılaştırıldığında yoğun zaman dilimi boyunca gönderilmeyen senkronizasyon paketlerinin ilk boşlukta toplu olarak gönderilmesinin ana sunucuda oluşturduğu yoğunluğun klasik Poll yöntemine göre çok düşük olduğu

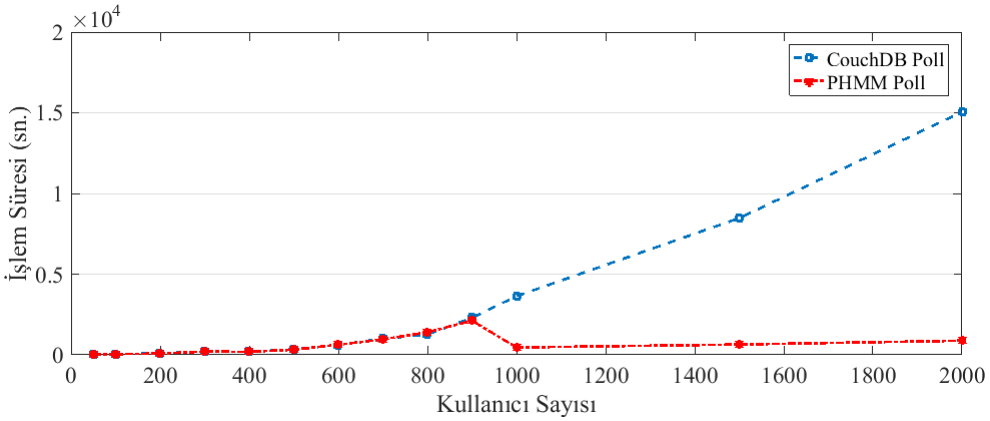
görülmüştür. Bir dakikalık periyotlarla sürekli yapılan poll işlemi yerel sunucu ile ana sunucu arasındaki bağlantıda ortalama 30 saniye gecikmeye sebep olurken önerilen yöntemde bu süre 5 saniye civarındadır. Tüm senaryolar ele alındığında benzer şekilde önerilen yöntemde yapılan toplam yeniden iletimlerin sayısı 50 civarındayken, CouchDB Poll yönteminde bu sayı dokuz kat fazladır.

6.3.4. Kullanıcı sayısı bakımından yöntemin değerlendirilmesi

Tüm ağın gün içerisindeki davranışına ek olarak, her bir LAN'ın yoğunluk oluşana kadar cevap verebileceği kullanıcı sayısı ve CouchDB senkronizasyon mekanizmasının sisteme getireceği yükün analizi de yapılmıştır. Sonuçlar Şekil 6.19 ve Şekil 6.20'de gösterilmiştir. Burada elde edilen grafikler daha önce gerçek zamanlı olarak yapılan çalışma (Kaya Gülağız ve diğ., 2018) temel alınarak elde edilmiştir.



Şekil 6.19. Kullanıcı sayısı ve verim arasındaki ilişki



Şekil 6.20. Kullanıcı sayısı ve işlem süresi arasındaki ilişki

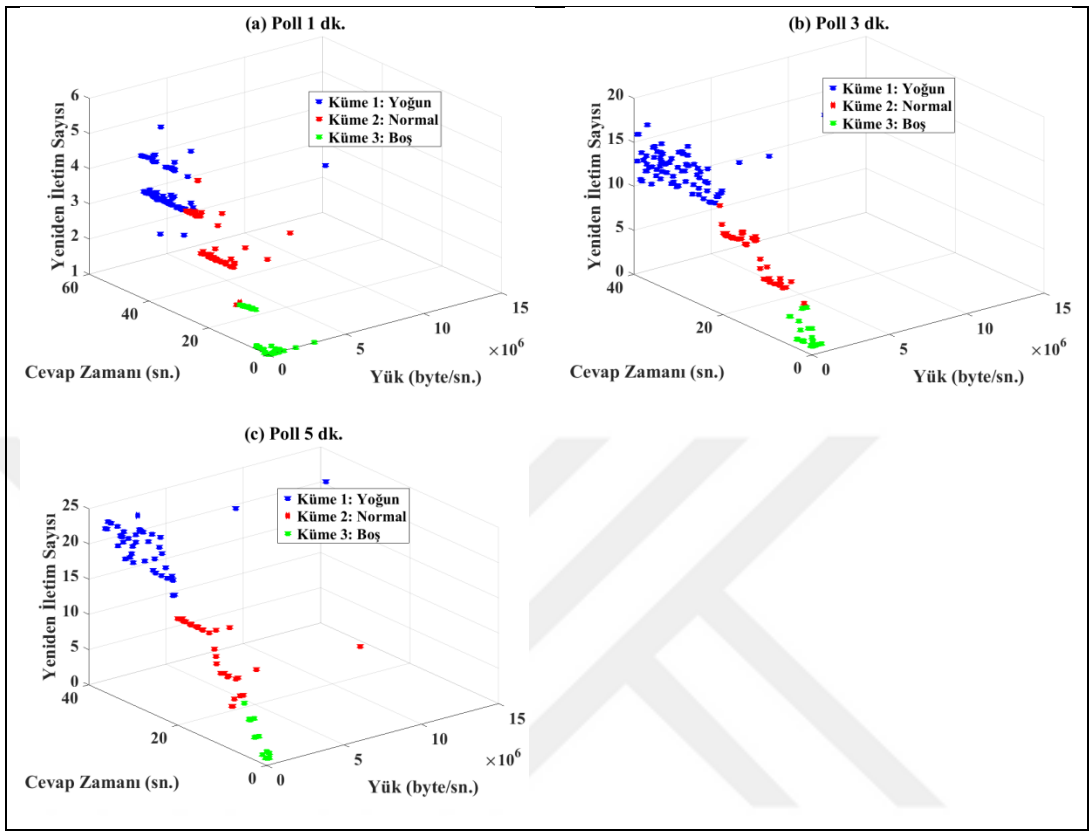
Şekil 6.19'da 2 numaralı LAN'ın kullanıcı sayısına göre verimlilik değeri gösterilmiştir. CouchDB gönderilen verinin boyutuna da bağlı olarak, senkronizasyon işlemi yapılmadığında kullanıcılara hızlı bir şekilde cevap verebilmektedir. Kullanıcılar sistemde aktif olarak işlem yaparken senkronizasyon işlemi de başlatılırsa Şekil 6.20'de de görüldüğü gibi kullanıcılara ortalama cevap verme süresi yaklaşık sekiz kat artmaktadır. PHMM Poll mekanizması, ortalama cevap süresi 2000-3000 milisaniye civarındayken yoğunluğu tespit etmektedir ve eş zamanlı işlem yapan kullanıcı sayısı 700-800 civarındayken arka planda bir dakikalık periyotlarla senkronizasyon işlemini de gerçekleştirmektedir. Kullanıcı sayısı 900-1000 aralığındayken, senkronizasyon işlemini bir sonraki boş zaman dilimine kadar ertelemektedir. Geliştirilen bu mekanizma ile yoğun zaman dilimlerinde senkronizasyon işlemi gerçekleştirilmediğinden LAN içerisinde cevap verilebilecek kullanıcı sayısı klasik CouchDB Poll mekanizmasının yaklaşık iki katına çıkarılmıştır. Şekil 6.19'da da görüldüğü gibi önerilen yöntem kullanıcı sayısı artsa bile LAN içerisindeki tüm kullanıcılara yüksek verimle hizmet verebilmektedir.

6.3.5. Farklı poll periyotları altında yöntemin değerlendirilmesi

Daha önce yapılan çalışmalarda poll periyodu bir dakika olarak belirlenmiştir ve sonuçlar üç farklı senaryo için analiz edilmiştir. Önerilen yöntemin ne kadar verimli olduğu gösterilmiştir. Çalışmanın bundan sonraki bölümünde farklı poll periyotları için sistemin davranışı analiz edilmiştir. Şekil 6.21'de üç farklı poll periyodu için elde edilen günlük verilerin kümeleme işlemine ait sonuçlar gösterilmiştir.

Poll periyodu arttığında günlük olarak alınan örnek sayısı da azalmıştır. Bu nedenle grafiklerden de görüldüğü gibi aynı zaman aralığına düşen veri azalmıştır ve değerler daha belirgin olarak ayrılmıştır. Aynı zamanda poll periyodu arttığında sunucuya senkronizasyon anında toplu gönderilen verinin boyutu da artmaktadır. Böylece özellikle arka plan trafiğinin düşük yük ve yüksek yük olduğu durumlarda, yeniden iletimlerin artmasına sebep olmuştur. Poll periyodunun artmasıyla modelin hassasiyeti azalmıştır ve yoğunluk durumunu daha geç fark etmeye başlamıştır. Periyot arttığında kümeler arasındaki geçişler daha belirgin hale gelmiştir. Poll periyodu bir iken verideki yeniden iletimler 1-2-3-4-5 şeklinde değişirken, periyot

üç olduğunda değişim 1-3-4-6-9-10 şeklinde, periyot beş olduğunda 1-3-5-8-10-15-20 olarak gözlemlenmiştir.

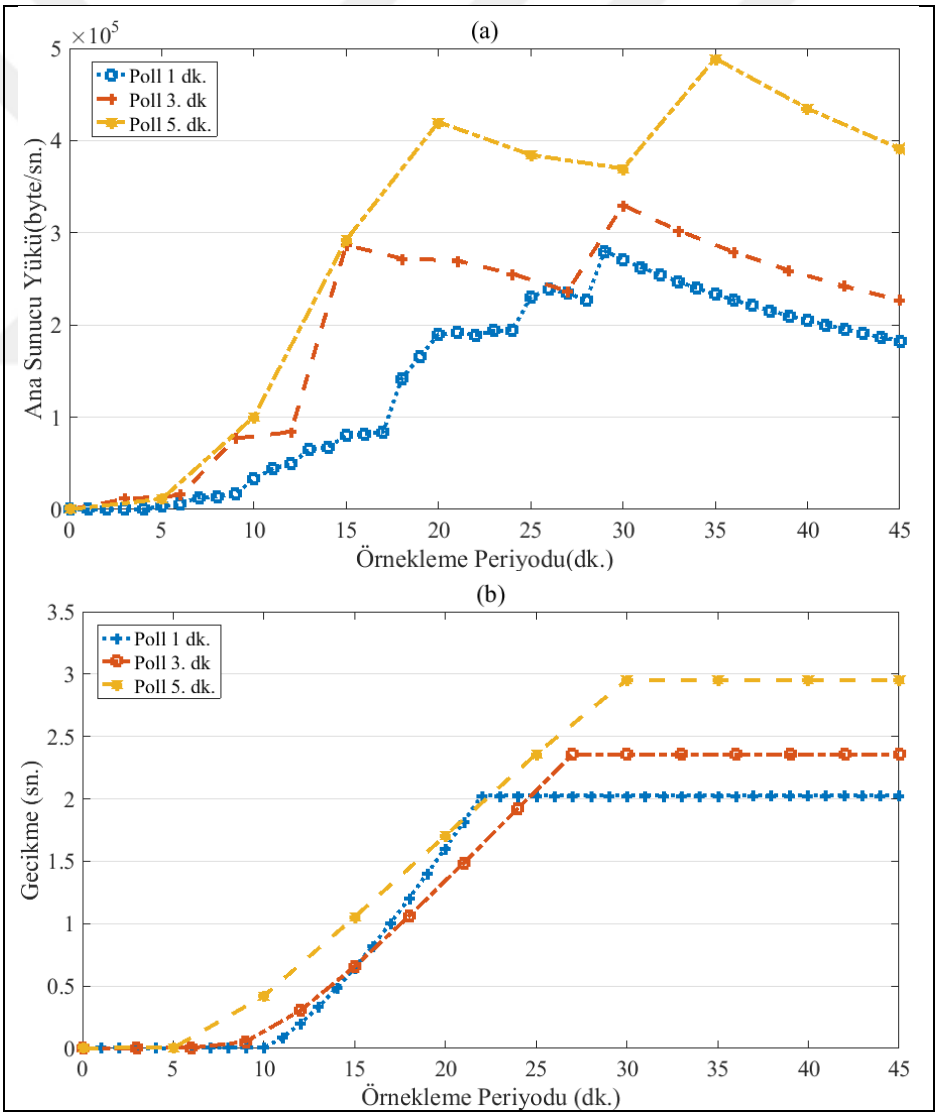


Şekil 6.21. Farklı poll periyotları için günlük veriye ait kümeleme sonuçları

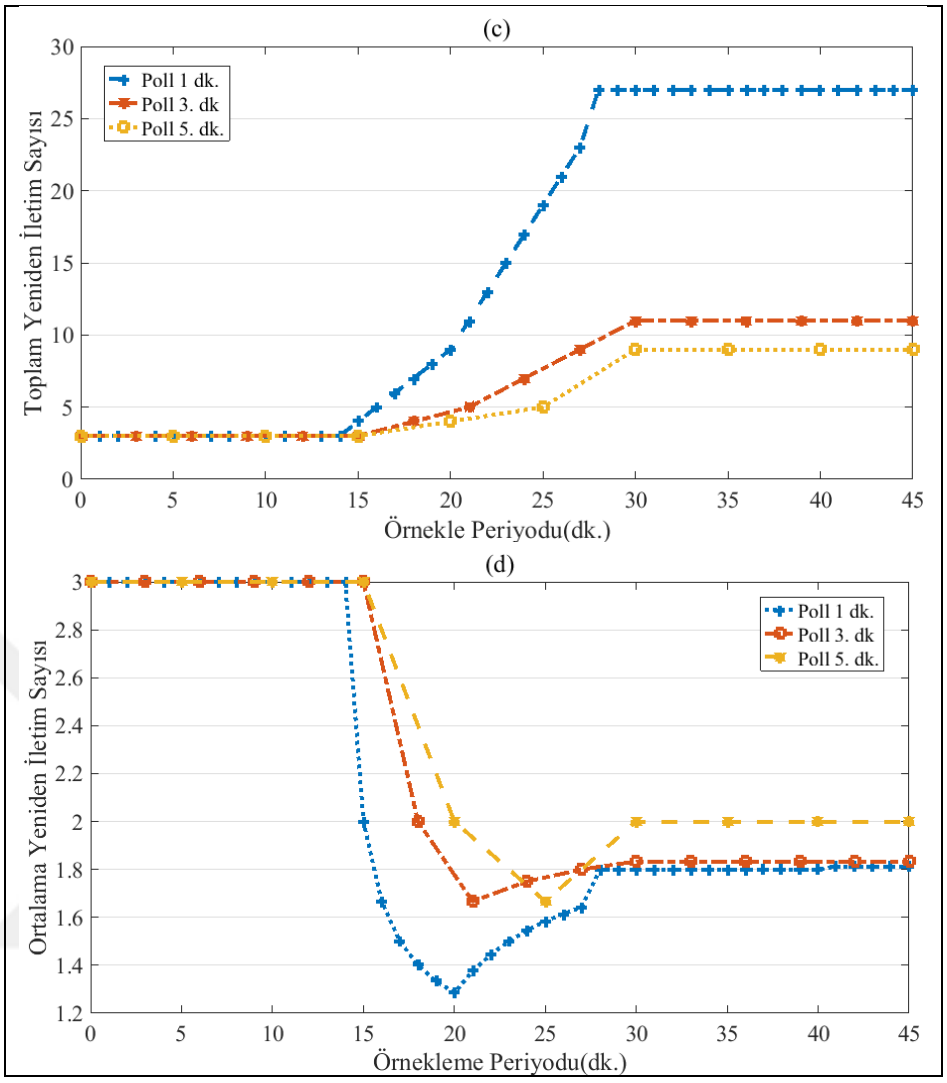
Şekil 6.22'de ilk senaryo olan boş-normal-yoğun senaryosu için elde edilen grafikler gösterilmiştir. Poll periyodu arttığında Şekil 6.22(a)'da da görüldüğü gibi sunucuya gönderilen senkronizasyon paketleri her periyotta poll 3 için poll 1'in yaklaşık üç katı, poll 5 için poll 1'in yaklaşık beş katı olacaktır. Ana sunucu da oluşan yük miktarı da bu oranlar ile benzer oranlarda artış göstermiştir. Şekil 6.22(c)'de zamana göre yerel sunuculardan yapılan toplam yeniden iletim sayısı, Şekil 6.22(d)'de ise yerel sunucudan yapılan ortalama yeniden iletimler gösterilmiştir. Yeniden iletimlerin ortalamasına bakıldığında en düşük yeniden iletim sayısı en düşük poll periyodunda elde edilmiştir.

Bunlara ek olarak Şekil 6.22'de toplam yeniden iletimlerin ana sunucu yoğunluğu ile ters orantılı olduğu görülmektedir. Bu durumun sebebi poll periyodu sık olduğunda aynı zaman dilimi içerisinde daha fazla paket gönderimi yapılmasıdır. Yani yoğunluk durumuna göre iletim yapmayı, ilk olarak poll 1, sonra poll 3, son olarak da poll 5

periyodu ile iletim yapan senaryolar durduruyor. Bu durumda yoğunluğu, poll 1 ile çalışan sistem yaklaşık olarak 25. dakikada, poll 3 ile çalışan sistem yaklaşık olarak 27. dakikada ve poll 5 ile çalışan sistem yaklaşık olarak 30. dakikada fark ediyor. Ancak poll periyodu sebebiyle ilk 25 dakikada periyodu bir olan sistem 25 kez, periyodu 3 olan sistem 9 kez ve periyodu 5 olan sistem yaklaşık 6 kez iletim yapmış oluyor. Böyle bir durumda periyodu bir olan sistem yoğunluğu daha erken fark etse bile belirtilen zaman diliminde yoğunluğun başlangıç anlarında periyodu 3 ve 5 olan sistemlere göre toplamda daha fazla iletim yapmış oluyor. Bu nedenle hangi poll periyodunun daha avantajlı olduğunu anlamak için zamana göre ortalama yeniden iletim sayısına bakılması daha doğru olacaktır.



Şekil 6.22. İlk senaryonun farklı poll periyotları altında ana sunucu yükü (a), gecikme (b), toplam (c) ve ortalama (d) yeniden iletim sayısı değerlerine göre performans analizi

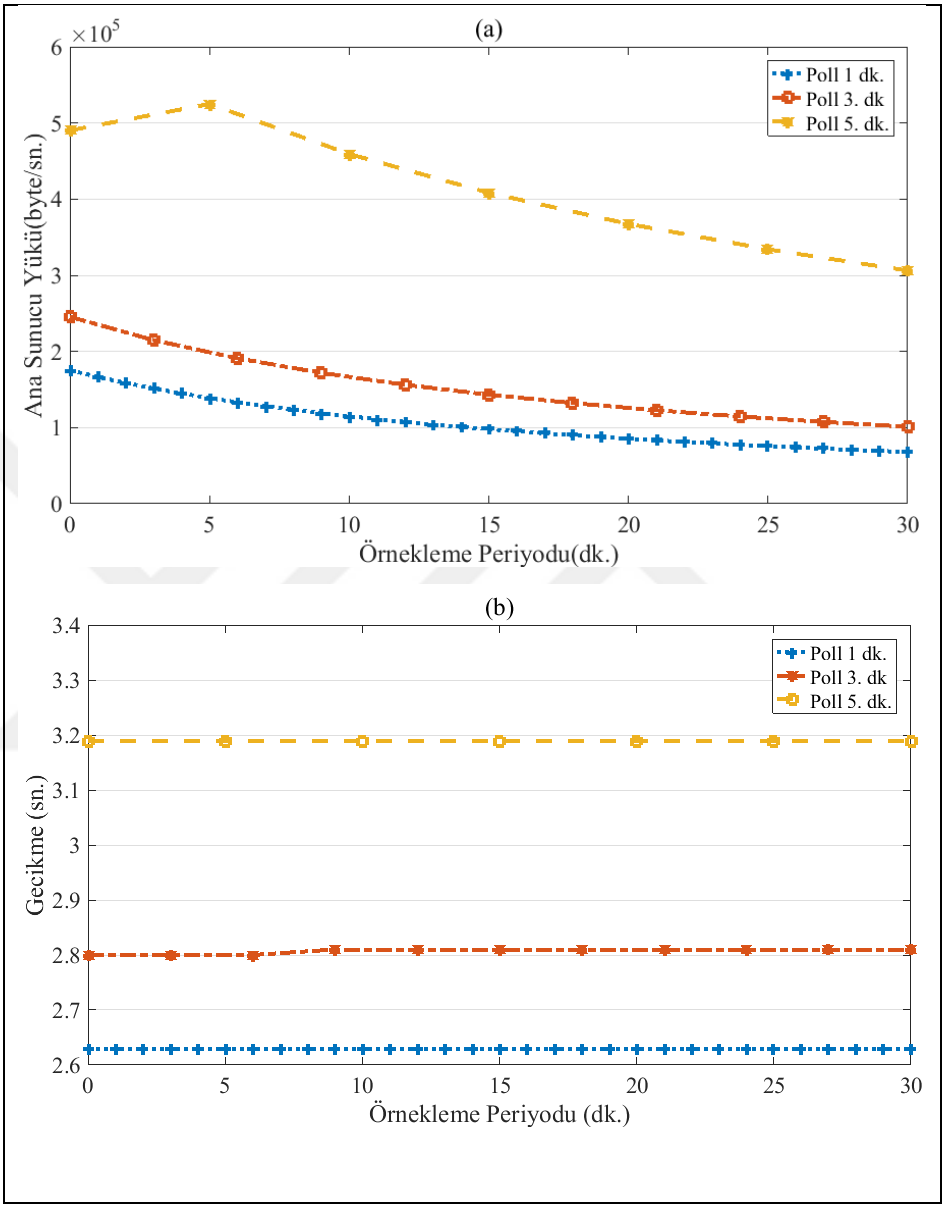


Şekil 6.22. (Devam) İlk senaryonun farklı poll periyotları altında ana sunucu yükü (a), gecikme (b), toplam (c) ve ortalama (d) yeniden iletim sayısı değerlerine göre performans analizi

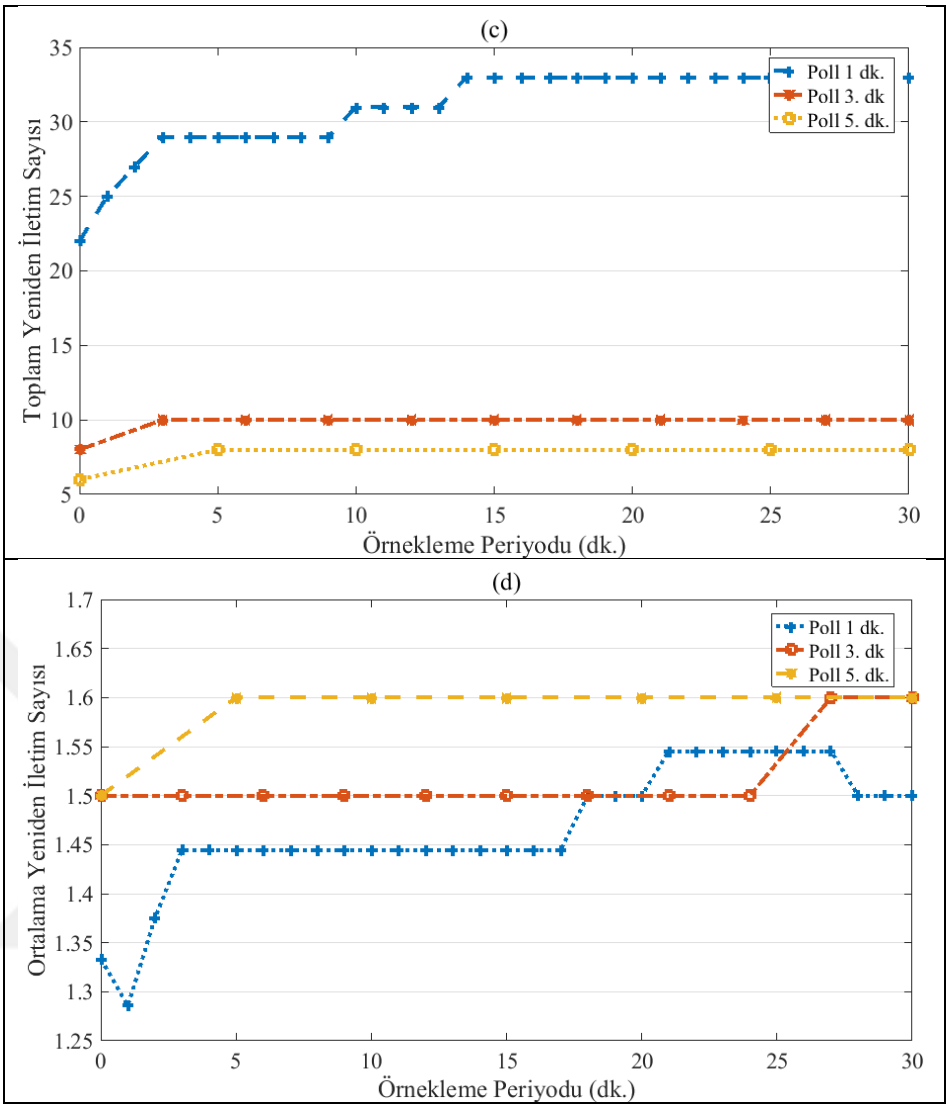
Ortalama yeniden iletim grafiği Şekil 6.22(d)'de gösterilmiştir. Grafiğe göre yapılan yeniden iletimlerin ortalamasına bakıldığında en düşük yeniden iletim sayısı en düşük poll periyodunda elde edilmiştir. Şekil 6.22(b)'den de görüleceği gibi hattaki gecikmenin de yoğunluk ve ortalama yeniden iletimler ile paralellik gösterdiği görülmektedir.

Şekil 6.23'te ikinci senaryo olan yoğun durum senaryosu için elde edilen grafikler gösterilmiştir. Birinci senaryodan sonra yoğun senaryosu çalıştırıldığında bu durumda hiç paket iletimi yapılmadığından ana sunucudaki yoğunluk zaman içerisinde azalmaktadır. En fazla yoğunluğa sahip olan poll 5 dk. periyodunda, yoğunluğun azalması diğer poll periyotlarına göre daha fazla zaman almaktadır.

Yeniden iletim sayısı ise tüm poll periyotları için ilk senaryoda kaldığı şekilde sabitlenmektedir. Sunucu ile yerel sunucular arasındaki hatlarda iletim olmadığı için gecikme değeri de, tüm poll periyotları için ilk senaryodaki değerlerde sabit kalmıştır.

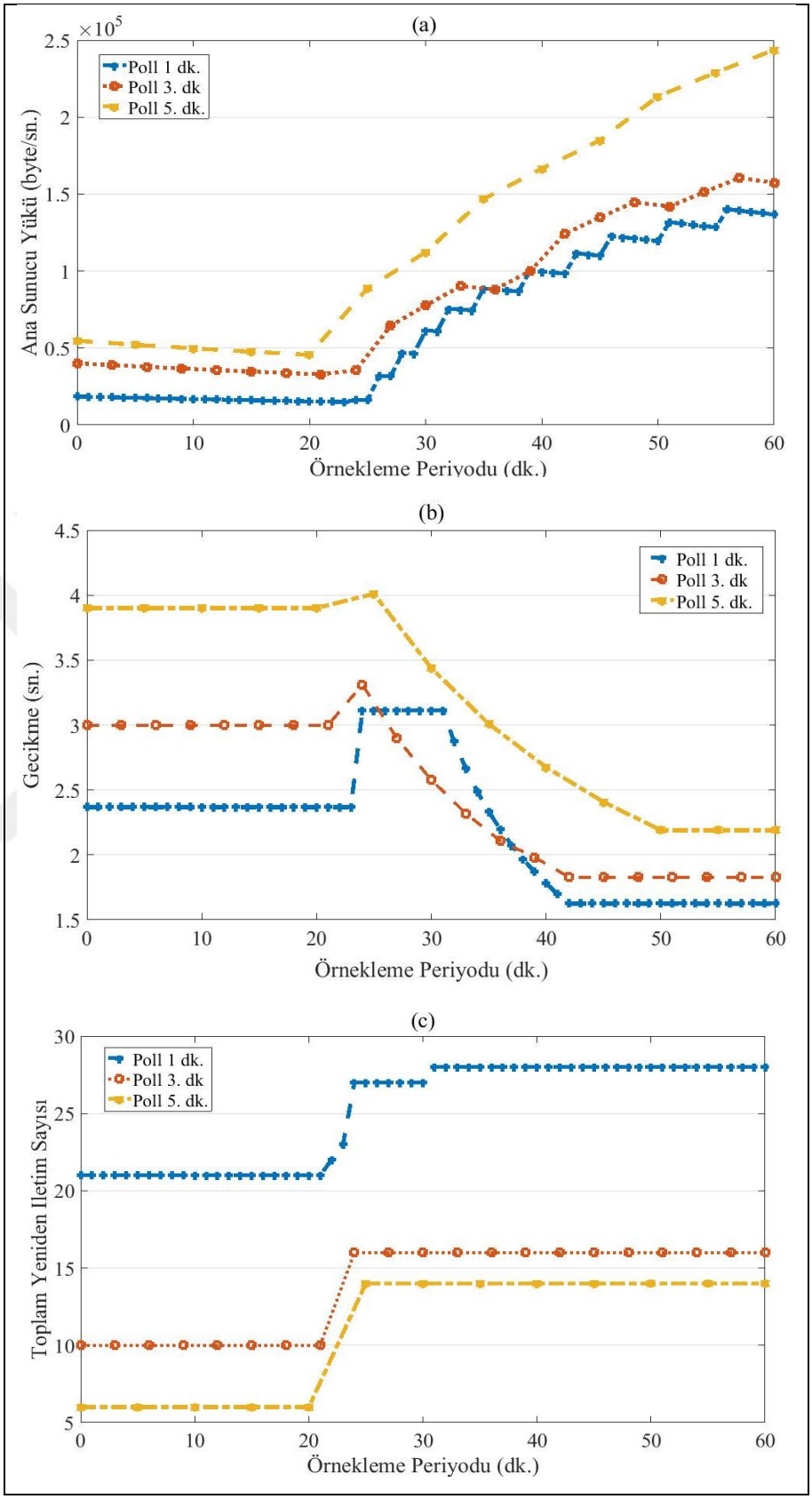


Şekil 6.23. İkinci senaryonun farklı poll periyotları altında ana sunucu yükü (a), gecikme (b), toplam (c) ve ortalama (d) yeniden iletim sayısı değerlerine göre performans analizi

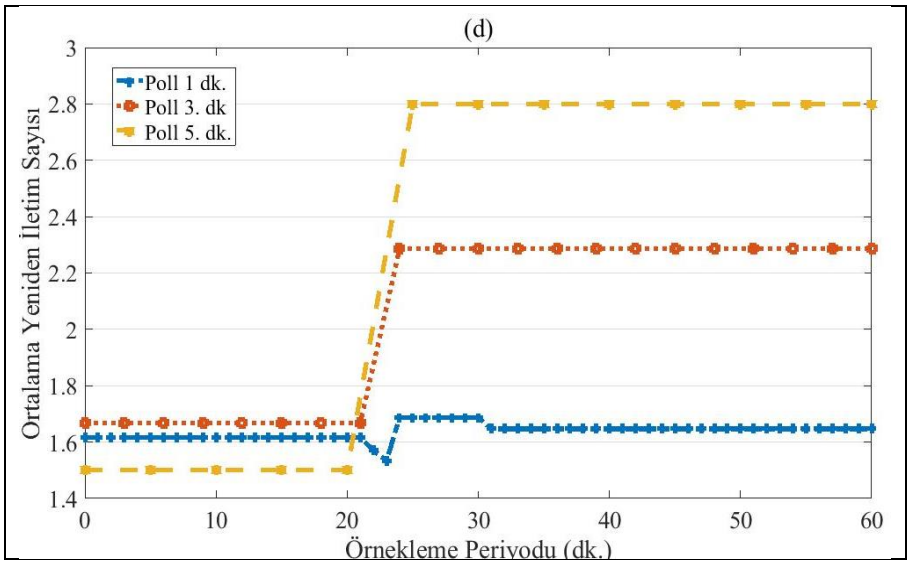


Şekil 6.23. (Devam) İkinci senaryonun farklı poll periyotları altında ana sunucu yükü (a), gecikme (b), toplam (c) ve ortalama (d) yeniden iletim sayısı değerlerine göre performans analizi

Yoğun senaryosundan sonra üçüncü senaryo da ana sunucunun yükü Şekil 6.24'te de gösterildiği gibi zaman içerisinde artış göstermiştir. En fazla artış yine poll 5 periyodunda olmuştur. Yine bu senaryoda ki yoğun durumundan normal ve bos durumuna geçişlerde en fazla ortalama yeniden iletim poll 5 senaryosunda olmuştur. Gecikme değeri de sunucudaki yoğunluk ve ortalama yeniden iletim sayılarıyla paralellik göstermiştir.



Şekil 6.24. Üçüncü senaryonun farklı poll periyotları altında ana sunucu yükü (a), gecikme (b), toplam (c) ve ortalama (d) yeniden iletim sayısı değerlerine göre performans analizi

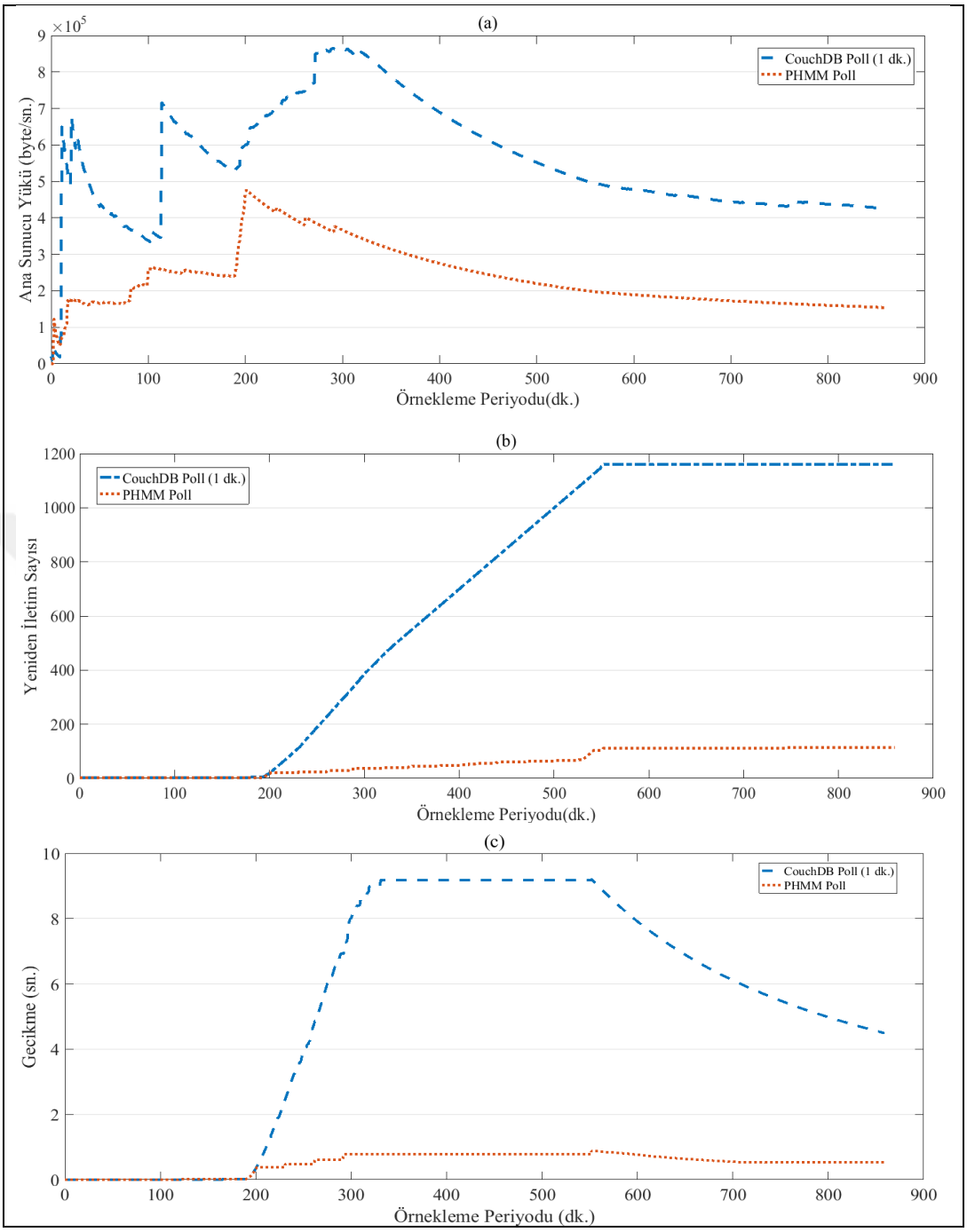


Şekil 6.24. (Devam) Üçüncü senaryonun farklı poll periyotları altında ana sunucu yükü (a), gecikme (b), toplam (c) ve ortalama (d) yeniden iletim sayısı değerlerine göre performans analizi

Üç senaryo genel olarak analiz edildiğinde poll periyodu arttığında hassasiyetin azaldığı görülmektedir. Bunun temel sebebi de periyodun artmasıyla günlük olarak alınan örneklerin azalmasıdır. Poll periyodu arttığında toplam yeniden iletim sayısında ise azalma olduğu görülmektedir. Burada da verinin toplu olarak gönderilmesi ve periyodun daha düşük olması sebebiyle yoğun zaman dilimine denk gelen gönderimlerin azalması etkili olmuştur. Ancak zamana göre ortalama yeniden iletimler incelendiğinde bir dakika olan poll periyodunun daha düşük bir ortalama yeniden iletim sayısına sahip olduğu görülmektedir. Poll periyodunun artışı hassasiyeti düşürürken toplam yeniden iletim sayısını da düşürmekte ancak sunucunun anlık yoğunluğunu daha fazla arttırmaktadır.

6.3.6. Yöntemin bir günlük davranışının analizi

Bu zamana kadar yapılan çalışmalarda CouchDB nin bir dakikalık periyodu ile önerilen bir dakikalık poll periyodu karşılaştırılmıştır. Daha sonra farklı poll periyotları için önerilen yöntemin analizi yapılmıştır. Bu bölümde de akademik bir ağa uygun olacak şekilde bir günlük olarak önerilen yöntemin analizi yapılmıştır. Elde edilen sonuçlar Şekil 6.25'te gösterilmiştir.



Şekil 6.25. Önerilen yöntemin günlük performans analizi

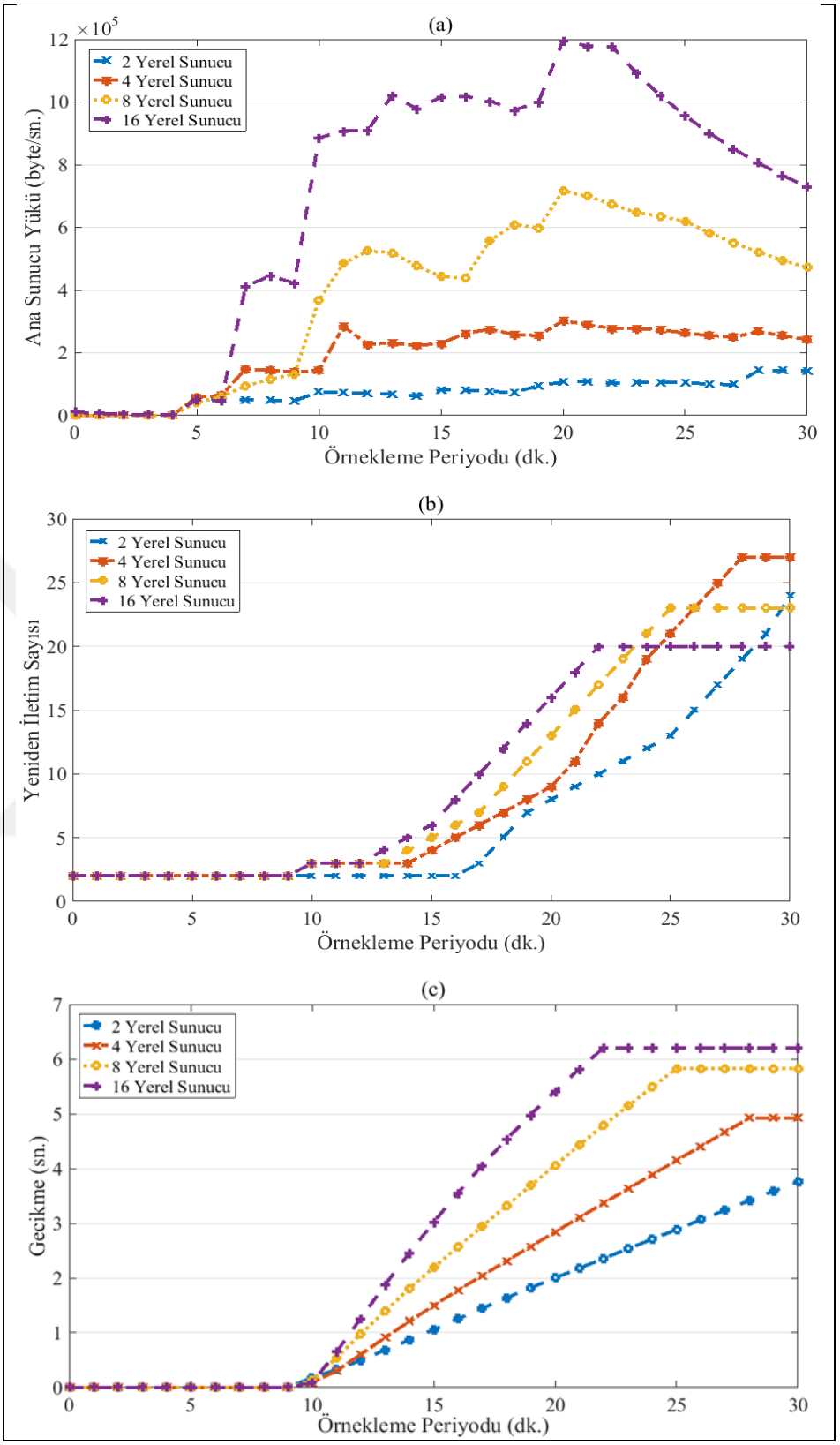
Üç senaryo kullanılarak yapılan analizlerden farklı olarak günlük analizde yoğunluğun uzun sürmesi durumu da göz önünde bulundurulmuştur ve çok uzun süren yoğunluk durumları için senkronizasyonun yapılmaması durumuna bir üst sınır tanımlanmıştır. Bu süre daha önce yapılan çalışmalar (Kaya Gülağız ve diğ., 2018; Kavak ve diğ., 2014) referans alınarak otuz dakika olarak belirlenmiştir. Yani yoğunluk durumu otuz dakikadan fazla devam ettiğinde her otuz dakikada yoğunluk

olsa bile verilerin çok uzun süre eşlenmemiş durumda kalmasını önlemek amacıyla otomatik olarak senkronizasyon işlemi başlatılmıştır. Önerilen yöntem günlük olarak CouchDB Poll periyodu ile karşılaştırıldığında ana sunucudaki yoğunluğu Şekil 6.25(a)'da da görüldüğü gibi yaklaşık olarak yarı yarıya düşürmüştür. Bir gün içerisinde yapılan toplam yeniden iletimler önerilen yöntemde CouchDB Poll periyodunun onda biri kadardır. Yine günlük yoğunluğun en fazla olduğu anlarda CouchDB Poll yöntemindeki gecikme 9-10 saniye civarındayken önerilen yöntemde bu süre 1 saniyenin üstüne çıkmamıştır.

6.3.7. Farklı mimariler altında yöntemin analizi

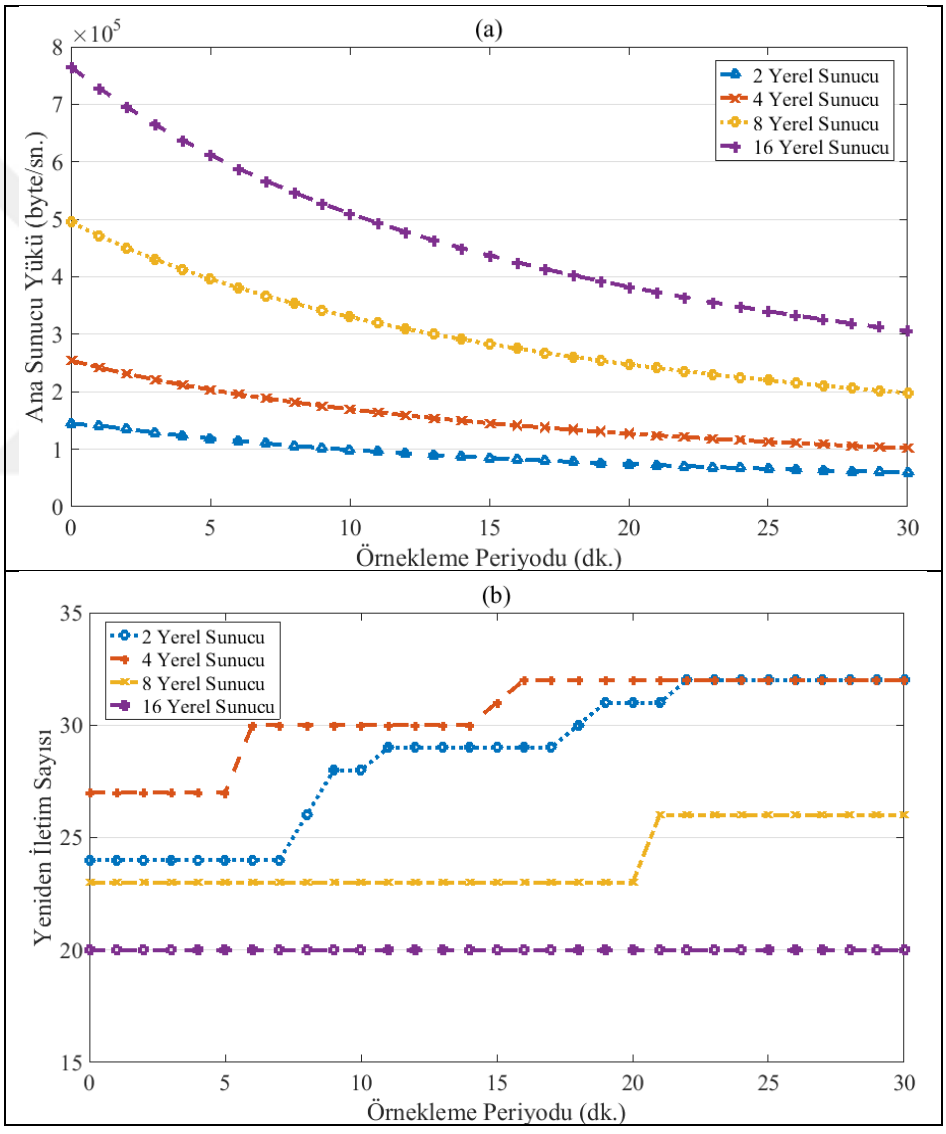
Bundan önceki bölümlerde verilen analiz sonuçları on altı tane yerel sunucu içeren mimariler üzerinden elde edilmiştir. Tez çalışmasının bu bölümünde sırasıyla iki, dört, sekiz ve on altı sunucu içeren mimariler için daha önce bahsedilen üç farklı senaryo üzerinden önerilen yöntemin davranışı analiz edilmiştir.

Birinci senaryoya ait farklı mimariler için ana sunucu yükü sonuçları Şekil 6.26(a)'da gösterilmiştir. Sonuçlar incelendiğinde yerel sunucu sayısı arttıkça ana sunucudaki yükün de orantılı olarak arttığı görülmektedir. Yeniden iletim sayıları grafiği Şekil 6.26(b)'de gösterilmiştir. Buradan elde edilen sonuçların ana sunucu yoğunluğu ile ilişkili olduğu görülmektedir. Yerel sunucu sayısı arttıkça ana sunucudaki yoğunluk daha hızlı artış göstermektedir. Bunun sonucu olarak yerel sunucu sayısı arttığında, PHMM Poll yöntemi sistemdeki tıkanıklık durumu da daha erken fark etmektedir. Bu nedenle yerel sunucu sayısı arttıkça senkronizasyon paketlerinin gönderimi daha erken zamanlarda durdurulmaktadır. Gecikme değeri de yeniden iletim sayısı ile paralellik göstermektedir. Yani on altı yerel sunucu içeren sistemin gecikmesi 21. dakika da sabitlenirken, diğerleri(sekiz, dört ve iki yerel sunucu içerenler) yaklaşık olarak 25., 28. ve 35. dakikalarda sabitlenmektedir.

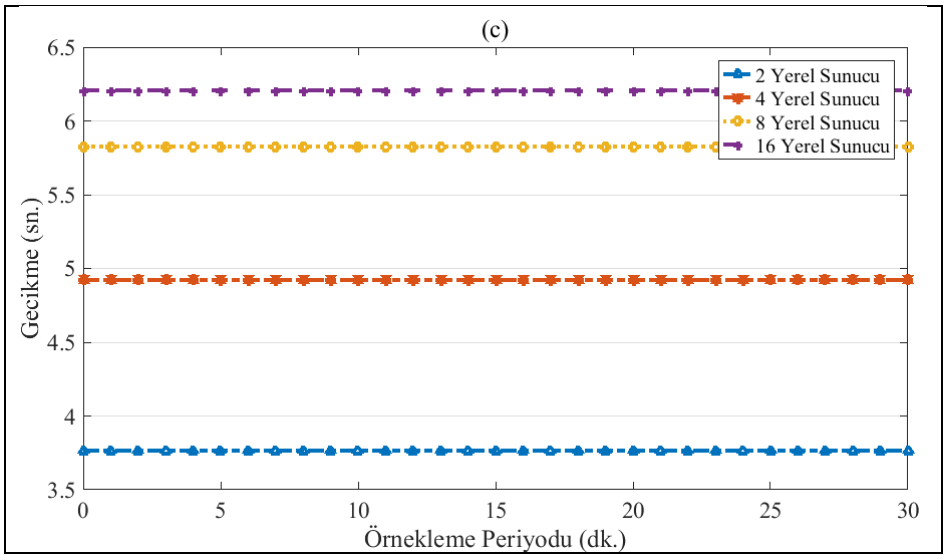


Şekil 6.26. Farklı mimariler için birinci senaryonun ana sunucu yükü (a), yeniden iletim sayısı (b) ve gecikme (c) değerlerine göre analizi

Şekil 6.27'de ikinci senaryo olan yoğun senaryosu için sonuçlar gösterilmektedir. Şekil 6.26(a)'da fazla yerel sunucu içeren sistemin yoğunluk durumunu da daha erken fark ettiğini belirtmiştik ancak yoğunluk durumunu daha erken fark etse bile ana sunucudaki mevcut yükün azalması için daha fazla zamana ihtiyaç duymaktadır. Bu durum Şekil 6.27(a)'da gösterilmiştir. Şekil 6.27(b)'de yeniden iletim sayısı ve Şekil 6.27(c)'de gecikme grafikleri gösterilmiştir. Bu değerlerin her bir mimari için ilk senaryodaki değerlerde sabit kalmıştır. Çünkü bu senaryo da yoğunluk fark edildiği için senkronizasyon paketlerinin gönderimi durdurulmuştur.



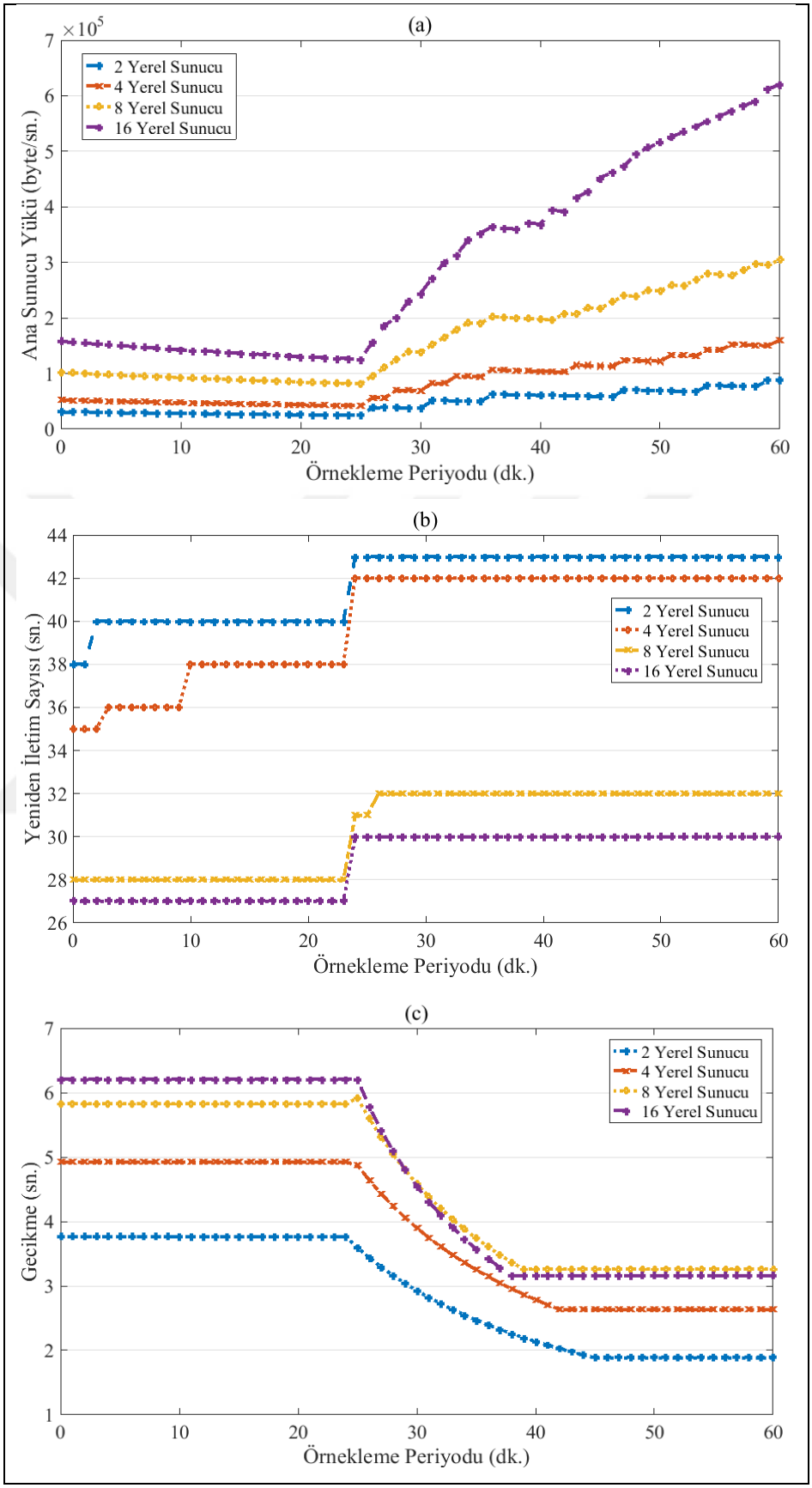
Şekil 6.27. Farklı mimariler için ikinci senaryonun ana sunucu yükü (a), yeniden iletim sayısı (b) ve gecikme (c) değerlerine göre analizi



Şekil 6.27. (Devam) Farklı mimariler için ikinci senaryonun ana sunucu yükü (a), yeniden iletim sayısı (b) ve gecikme (c) değerlerine göre analizi

Şekil 6.28'de üçünü senaryo olan yoğunluk sonrası normal ve boş durumlarının gerçekleşmesi modellenmiştir. Birinci senaryonun tam tersine burada yoğunluk durumundan çıkışı ilk olarak iki yerel sunucu içeren mimari tespit etmektedir. Daha sonra sırasıyla dört, sekiz ve on altı yerel sunucu içeren mimariler boş durumu tespit edip senkronizasyona başlamaktadır. İki ve dört yerel sunucu içeren sistemler yoğunluğu daha geç fark edip iletimi daha geç durdurmaktadır aynı zamanda yoğunluktan çıkışı da daha erken fark edip daha erken senkronizasyona başlamaktadır. Sonuç alarak az sayıda LAN içeren mimariler daha uzun süreler iletimde kalmaktadır. Yoğunluk anına yakın zamanlarda da daha fazla iletim yapmaktadırlar. Bu nedenle az sayıda yerel sunucu içeren mimarilerde toplam yeniden iletim sayısı daha fazla olmaktadır. Üçüncü senaryo için gecikme grafiğinin ise ana sunucu yoğunluğu ile paralellik gösterdiği görülmektedir.

Önerilen yöntem üç senaryo için genel olarak incelendiğinde yerel sunucu sayısındaki artışa paralel olarak, yöntem kümelemeyi mimarilere uygun olarak yapmaktadır. Dolayısıyla mimariye uygun olarak yoğunluk durumunu geç ya da erken fark etmektedir. Böylece senkronizasyon için uygun olan zamanı her mimariye uyumlu olarak tespit edebilmektedir.



Şekil 6.28. Farklı mimariler için üçüncü senaryonun ana sunucu yükü (a), yeniden iletim sayısı (b) ve gecikme (c) değerlerine göre analizi

6.3.8. PHMM Poll yönteminin literatürdeki benzer poll yöntemleriyle karşılaştırılması

Bu bölümde önerilen yöntemin verimliliğini göstermek amacıyla PHMM Poll yöntemi, literatürde yer alan, poll yöntemi ile paket iletimi yapan farklı çalışmalarla karşılaştırılmıştır. Literatürde kampüs ağları ya da ağ örüntülerini elde etmek için geliştirilmiş poll yöntemleri bulunmadığından, farklı ağların analizi amacıyla kullanılmış olan poll yöntemleri göz önünde bulundurulmuştur. Karşılaştırma sonuçları Tablo 6.12'de gösterilmiştir.

Ikeda ve Kitayama (2009) tarafından pasif optik erişimli ağlarda TCP verimliliğini arttırmak amacıyla dinamik bir bant genişliği tahsisi algoritması önerilmiştir. Optik erişimli ağlar Optik Ağ Ünitesi (ONU) ve Optik Hat Sonlandırıcı (OLT) olarak isimlendirilen iki ana birimden oluşmaktadır. OLT'ler ONU'ların veri iletimlerinin planlanmasından sorumludurlar. Burada yapılan veri iletimleri poll yöntemi kullanılarak gerçekleştirilmektedir. OLT'ler poll işlemini gerçekleştirmeden önce ONU'lardan gelen bilgi mesajları doğrultusunda bant genişliği tahsisini de yapmaktadırlar. Çalışmada RTT değerini ve ihtiyaç duyulan bant genişliğini dikkate alan adaptif bir poll yöntemi önerilmiştir. Test işlemleri sırasında merkezi bir sunucu etrafında 32 adet ONU içeren ve iletim hattı kapasitesi 10 Gb/sn. olan bir mimari kullanılmıştır. Protokol TCP NewReno olarak ayarlanmıştır. Önerilen adaptif poll yöntemi temelli algoritma ile pasif optik erişimli ağ mimarisindeki standart bant genişliği tahsisi yapan yöntem farklı yük değerleri altında karşılaştırıldığında yük değeri 0-200 Mb\sn. aralığındayken gecikmedeki azalmanın yaklaşık %72-78 aralığında olduğu, yük değeri 800Mb\sn. üzerine çıktığında yaklaşık %50-63 aralığında olduğu tespit edilmiştir. Gecikmedeki azalma oranları çalışmada verilen grafikler üzerinden yaklaşık olarak hesaplanmıştır.

Lv. ve diğ. (2015) tarafından yine bir önceki çalışmada açıklanan pasif optik erişimli ağlarda veri iletimi için kullanılan poll işleminin planlanması amacıyla yük uyarlamalı enerji verimliliğini arttıracak bir yöntem önerilmiştir. Çalışmada enerji verimliliğinin sağlanması için ONU'ların poll sırasının ağdaki yük ile orantılı olarak belirlenmesi gerektiğinden bahsedilmiştir. Düşük ağ yükleri altında daha düşük poll periyotlarının kullanılması gerektiği, yüksek ağ yükleri altında ise poll periyodunun

uzun tutulmasınının iletim yapan düğümlerin enerji verimliliğini arttırdığı gösterilmiştir. Bu nedenle her başarılı iletim sonrasında bir sonraki iletim zamanı, ortalama bant genişliği ihtiyacı ve paket iletim zamanı göz önünde bulundurularak belirlenmiştir. Yük değeri yaklaşık 800 Mb\sn. iken ortalama paket gecikmesi geleneksel şema ile benzerdir ancak ağdaki yük 800Mb\sn. üzerine çıktığında önerilen yöntem enerji verimliliğini arttırırken ortalama paket gecikmesini de yaklaşık %50-60 aralığında arttırmıştır. Ortalama gecikmedeki artışın yaklaşık değeri yazarlar tarafından verilen karşılaştırmalı grafikteki ek paket gecikmesi değerlerine göre elde edilmiştir.

Tandon ve Motani (2017) merkezi bir sunucu tarafından uygulanan poll işlemi ile istemcilere paket iletiminin gerçekleştirildiği bir mimaride, ortalama paket gecikmesini azaltmaya yönelik olarak hata düzeltme şemalarını kullanmıştır. Böylece poll ile iletim yapan mimarilerdeki paket hata olasılıklarının, dolayısıyla ortalama paket gecikmesi ve paketlerin yeniden iletim sayısının da azaltılması sağlanmıştır. Çalışmada gürültü\güç seviyesinin farklı değerleri için ortalama paket gecikmesindeki iyileşme grafikleri gösterilmiştir. İstemci sayısı dört ve kullanıcı başına paket iletim oranı saniyede 1000 paket iken hattaki gürültünün düşük olduğu durumlarda (SNR değeri yaklaşık 4 dB civarındayken) paket gecikmesindeki azalma yaklaşık %70-78 aralığındadır. Paket gecikmesindeki ortalama azalma değeri yazarlar tarafından verilen ortalama paket gecikmesi grafiğinden yaklaşık olarak elde edilmiştir.

Elde edilen sonuçlara göre önerilen yöntem diğer yöntemlerle her mimariye ait en fazla iyileşmenin olduğu durumlar altında karşılaştırıldığında, gecikme miktarındaki ortalama azalma bakımından daha başarılıdır. PHMM Poll yöntemine ilişkin olarak verilen ortalama verim artışı ve ortalama verim azalması miktarları on altı yerel sunucu içeren bir günlük karşılaştırma grafikleri üzerinden en düşük ve en yüksek değerler göz önünde bulundurularak elde edilmiştir. PHMM Poll yöntemi gecikmeyi azaltmasına ek olarak yoğunluğun olmadığı zamanlarda yaklaşık %7 lik bir verim artışı sağlarken, yoğunluğun yüksek olduğu zamanlarda bu değer yaklaşık %55 olarak tespit edilmiştir.

Tablo 6.12. PHMM Poll yönteminin literatürdeki yöntemlerle karşılaştırılması

Çalışma	**Kullanılan Test Ortamı**	**Simülasyon Parametreleri**		**Verim** (Ortalama Artış)	**Gecikme** (Ortalama Azalma)
Ikeda ve Kitayama (2009)	Pasif Optik Erişimli Ağlar	İletim Hattı Kapasitesi: Düğüm Sayısı: Yayılım Gecikmesi: Protokol:	10 Gb/sn (Kullanıcı başına 300 Mb/sn den 1 Gb/sn) 32 adet optik ağ birimi ve bir adet yönetici düğüm 0,0001 sn. TCP NewReno	-	%72-78 %50-63
Lv ve diğ. (2015)	Pasif Optik Erişimli Ağlar	İletim Hattı Kapasitesi: Düğüm Sayısı: Yayılım Gecikmesi: Protokol	10 Gb/sn. 3 adet optik iletim düğümü ve bir adet yönetici düğüm 0,0001 sn. TCP	-	%50 -60 (Ortalama Artış)
Tandon ve Motani (2017)	Merkezi Bir Sunucu Tarafından Yapılan Poll İşlemi ile Paket İletimi Yapan Çok Kullanıcılı Ağlar	İletim Hattı Kapasitesi: Düğüm Sayısı: Yayılım Gecikmesi: Protokol:	Belirtilmemiş N adet iletim yapan düğüm ve 1 ana sunucu 0,0001 sn. ARQ kullanan protokoller	-	%70-78
PHMM Poll (2018)	Merkezi Sunucusu Bulut Bilişim Temelli İçerik Dağıtım Ağları	İletim Hattı Kapasitesi: Düğüm Sayısı: Yayılım Gecikmesi: Kayıp Oranı Protokol:	Simülasyon: 44,7 Mb/sn Gerçek Zamanlı 350 Mb/ sn. 2 – 4 – 8 - 16 adet iletim yapan düğüm ve 1 ana sunucu 0,0001 sn. % 0,1 TCP Reno	%7-55	%80-85

7. SONUÇLAR VE ÖNERİLER

Teknolojinin gelişmesiyle birlikte elektronik ortamda saklanan verilerin boyutu da hızla artmaktadır. Bu nedenle verilerin tek merkezde depolanması, kullanımı ve yönetimi güçleşmektedir. Bu sorunların çözümü olarak dağıtık mimariler yaygın olarak kullanılmaktadır. Dağıtık mimarilerin en temel kullanım alanlarından bir tanesi de içerik dağıtım ağlarıdır. İçerik dağıtım ağları sayesinde kullanıcıların kendilerine coğrafi olarak yakın sunuculardan hizmet alması sağlanmaktadır. Böylece kullanıcıların verilere daha kısa sürelerle erişimi mümkündür. Donanım ve depolama maliyeti gibi kısıtlamalar nedeniyle içerik dağıtım ağları bulut temelli olarak da oluşturulabilmektedir. İçerik dağıtım ağlarında verilerin etkili iletimi için veri saklama amacıyla kullanılacak veri tabanlarının seçimi de önemlidir.

Veri tabanları ilişkisel ve ilişkisel olmayan veri tabanları olmak üzere ikiye ayrılırlar. İlişkisel veri tabanları yapılandırılmış veriler üzerinde işlem yaparken başarılıdır. Ancak yarı yapılandırılmış ve yapılandırılmamış veriler üzerinde istenilen düzeyde başarılı değildir. Bu nedenle homojen olmayan veriler için NoSQL veri tabanları tercih edilmektedir. İçerik dağıtım ağlarında eş veriler birden fazla sunucudaki veri tabanlarında saklanmaktadır. Aynı anda farklı sunucularda tutulan verilerin tutarlılığını sağlamak amacıyla senkronizasyon mekanizmalarına ihtiyaç duyulmaktadır.

Bu tez çalışmasında merkezi bulut sunucu temelli içerik yönetim ağları için bir senkronizasyon yöntemi önerilmiştir. Modellenen mimari multi-master senkronizasyonuna ihtiyaç duymaktadır ve senkronizasyonu yapılacak olan veriler doküman türünde homojen olmayan bir yapıdadırlar. Multi-master senkronizasyonunu desteklemesi ve doküman tabanlı homojen olmayan verileri yönetebilmesi sebebiyle NoSQL bir veri tabanı yönetim sistemi olan CouchDB tercih edilmiştir.

CDN'de ana sunucunun ya da yerel sunucuların konumlandırıldığı LAN'larda ağ trafiği ve altyapıdan kaynaklanan bant genişliği, paket iletim hızı, eş zamanlı cevap

verilebilecek kullanıcı sayısı gibi kısıtlamalar mevcuttur. Bu sebeplerle senkronizasyon için ağların en uygun olduğu zaman dilimlerinin belirlenmesi ve senkronizasyonun bu zamanlarda gerçekleştirilmesi gerekmektedir. Bu çalışmada CouchDB'de varsayılan olarak kullanılan senkronizasyon tetikleme mekanizması olan poll yöntemine alternatif, senkronizasyon işlemini sunucuların yükünü arttırmadan, en uygun zamanı belirleyerek gerçekleştirecek bir poll yöntemi önerilmiştir. Böylece senkronizasyona ihtiyaç duyan dağıtık yerel sunucu-ana sunucu mimarilerine uyumlu ve verinin olabildiğince sık aralıklarla eşitlenmesini sağlayacak esnek bir mimari elde edilmiştir.

Önerilen PHMM Poll yöntemi OPNET ortamında kampüs ağlarına ait ağ trafik istatistikleri kullanılarak modellenmiştir. Yöntemin etkinliği ve kullanılabilirliği üç farklı senaryo üzerinden gösterilmiştir. Senaryolar kampüs ağlarının gün içerisinde göstermiş olduğu farklı davranış örüntülerine göre oluşturulmuştur.

Tez çalışmasının ana katkıları aşağıdaki gibi maddeler halinde sıralanabilir:

- Günlük davranışı belirli bir örüntüye sahip olan ağların davranış örüntüleri kümeleme yöntemleri ve PHMM yöntemini kullanarak elde edilmiştir.
- Önerilen yöntem OPNET ortamında farklı sayılarda yerel sunucu içeren LAN'lar kullanılarak test edilmiştir. Yöntemin farklı sayıda LAN içeren mimariler ile uyumlu olarak ağ yoğunluğunu tespit edebildiği gösterilmiştir.
- Gün içerisindeki ağ trafiği üç farklı senaryo üzerinden bir dakikalık poll periyodu kullanılarak klasik CouchDB Poll yöntemi ile karşılaştırılmıştır ve yöntemin üstünlüğü gösterilmiştir. Tüm senaryolar üzerinden özellikle toplam paket yeniden iletim sayıları karşılaştırıldığında PHMM Poll yönteminin CouchDB'den yaklaşık dokuz kat daha iyi performans gösterdiği görülmüştür.
- CouchDB Poll ve PHMM Poll yöntemlerinin bir dakikalık poll periyodu altında bir günlük performans analizleri de gerçekleştirilmiştir. Elde edilen simülasyon sonuçlarına göre önerilen yöntemin ana sunucu yoğunluğu bir gün boyunca CouchDB Poll yönteminin ortalama yarısı kadardır. Gün içerisindeki yoğunluğun en fazla olduğu anlarda CouchDB Poll yöntemindeki gecikme 9-10 saniye civarındayken önerilen yöntemde bu süre 1 saniye civarındadır.

- Yöntem farklı poll periyotları altında da test edilmiştir. Poll periyodundaki artışın modelin hassasiyetini azaltırken toplam yeniden iletim sayısını da düşürdüğü ancak sunucunun anlık yoğunluğunu daha fazla arttığı tespit edilmiştir.
- OPNET ortamında gerçekleştirilen testlere ek olarak her bir ağ içerisindeki cevap verilebilecek kullanıcı sayısı Jmeter aracı ve CouchDB veri tabanı kullanılarak tespit edilmiştir. Yöntemin kullanıcı sayısı artsa bile LAN içerisindeki kullanıcılara yüksek verimle hizmet verebildiği gösterilmiştir.
- Yöntem literatürdeki farklı ağ yapılarında kullanılan poll yöntemleri ile karşılaştırıldığında gecikme miktarındaki ortalama azalma bakımından daha başarılı olduğu tespit edilmiştir.

Önerilen yöntemin en önemli eksikliği ağdaki yoğunluğun çok uzun süre devam etmesi durumunda eş verileri içeren sunucuların senkronizasyonunun uzun süre gerçekleştirilememesidir. Bu nedenle yöntemin kullanıldığı ağ mimarisine uygun olarak art arda gerçekleştirilen senkronizasyon işlemleri arasındaki bekleme süresine bir üst sınır belirlenmesi gerekmektedir. Bölüm altıda gerçekleştirilen günlük senkronizasyon işlemi sırasında, bu bekleme süresi otuz dakika olarak belirlenmiştir. Bu değer daha önce gerçekleştirilen test işlemleri doğrultusunda tanımlanmıştır ve yöntemin eksikliği kampüs ağları açısından giderilmiştir.

Günlük ağ verisinin kümelenmesi aşamasında farklı kümeleme yöntemlerinin karşılaştırılması yapılmıştır. Yöntemler genel olarak kümeleme işlemini benzer hata değerleri ile gerçekleştirmiştir ancak genel ağ davranışı ani yoğunluk değişimleri içeren mimarilerde seçilen parametreler kullanılarak elde edilecek günlük veri kümelerinde yakın değerler olacaktır. Böyle bir durumda kümlerin belirlenmesi aşamasında bazı değerlerin birden fazla kümeye ait olması daha doğru olacaktır. Özellikle bu tür davranışa sahip ağlara ait kümeleme işlemleri için bulanık kümeleme algoritmalarının tercih edilmesi daha uygundur.

Tez çalışması kapsamında senkronizasyon periyodunu belirlemek için kullanılan poll yöntemi sms sunucular, mail sunucular, web sunucular gibi pek çok farklı alanda farklı amaçlar için kullanılmaktadır. Yöntemin kullanıldığı alanlar genel olarak bir davranış örüntüsüne sahiptirler. Bu çalışmanın devamında yöntemin farklı uygulama alanlarında da kullanılabilirliği gösterilebilir. Poll işlemi ilişkisel olmayan veri

tabanları tarafından kullanıldığı gibi Sql, Oracle gibi pek çok ilişkisel veri tabanı üzerinde de uygulanmaktadır. Bu çalışmanın devamında önerilen yöntemin ilişkisel veri tabanları ile kullanımı ve performansı test edilebilir.

KAYNAKLAR

Abdlhamed M., Kiyafet K., Shi Q., Hurst W., A System for Intrusion Prediction in Cloud Computing, *International Conference on Internet of things and Cloud Computing*, Cambridge, United Kingdom, 22-23 March 2016.

Aduba C. N., Sadiku M. N. O., Simulation and Analysis of Different Traffic Models for ATM Networks, *IEEE SoutheastCon 2002*, Columbia, USA, 5-7 April 2002.

Ajila S. A., Al-Asaad A., Mobile Databases - Synchronization & Conflict Resolution Strategies Using SQL Server, *IEEE International Conference on Information Reuse & Integration*, Las Vegas, NV, 3-5 August 2011.

Alam M., Muley A., Joshi A., Kadaru C., *Oracle NoSQL Database:Real-Time Big Data Management for the Enterprise*, 1st ed.,Oracle Press, USA, 2014.

Alpaydın E., *Yapay Öğrenme,* 2. Baskı , Boğaziçi Üniversitesi Yayınevi, İstanbul, Türkiye, 2013.

Anderson J. C., Lehnardt J., Slater N., *Couchdb: The Definitive Guide*, 1st ed., O'REILY, United States of America, 2010.

Bhatnagar P. K., *Digital Network Synchronization: Basic Concepts, in Engineering Networks for Synchronization, CCS 7, and ISDN: Standards, Protocols, Planning and Testing*, 1st ed., Wiley-IEEE Press, New York, 1997.

Brewer E. A., Towards Robust Distributed Systems, *Annual ACM Symposium on Principles of Distributed Computing*, New York, USA, 16-19 July 2000.

Brown M. C., *Getting Started with CouchDB*, 1st ed., O'REILY, United States of America, 2012.

Bruggen R. V., *Learning Neo4j*, 1st ed., Packt Publishing, Birmingham, UK, 2014.

Büyüktatlı F., Şirketlerdeki Erken Uyarı Göstergeleri İle Saklı Markov Modeli Üzerine Bir Uygulama, Yüksek Lisans Tezi, Akdeniz Üniversitesi, Sosyal Bilimler Enstitüsü, Antalya, 2013, 344447.

Carpenter J., Hewitt E., *Cassandra: The Definitive Guide: Distributed Data at Web Scale*, 2nd ed., O'REILY, United States of America, 2016.

Carvalho S. A. L., Lima R. N., Silva-Filho A. G., A Pushing Approach for Data Synchronization in Cloud to Reduce Energy Consumption in Mobile Devices, *Brazilian Symposium on Computing Systems Engineering*, Manaus, Brazil, 3-7 November 2014.

Cattell R., Scalable SQL and NoSQL Data Stores, *ACM SIGMOD Record*, 2010, **39**(4), 12-27.

Chatziantoniou E., Allen B., Velisavljević V., An HMM-Based Spectrum Occupancy Predictor for Energy Efficient Cognitive Radio, *IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications*, London, United Kingdom, 8-11 September 2013.

Chickerur S., Goudar A., Kinnerkar A., Comparison of Relational Database with Document-Oriented Database (MongoDB) for Big Data Applications, *2015 8th International Conference on Advanced Software Engineering & Its Applications (ASEA)*, Jeju, Korea, 25-28 November 2015.

Che Z., Ji X., An Efficient Intrusion Detection Approach Based on Hidden Markov Model and Rough Set, *International Conference on Machine Vision and Human-Machine Interface (MVHI)*, Kaifeng, China, 24-25 April 2010.

Chen F., Guo K., Lin J., La Porta T., Intra-Cloud Lightning: Building CDNs in the Cloud, *IEEE Conference on Computer Communications*, Orlando, FL, USA, 25-30 Mart 2012.

Chopade M. R. M., Dhavase N. S., Mongodb, Couchbase: Performance Comparison for Image Dataset, *2nd International Conference for Convergence in Technology (I2CT)*, Mumbai, India, 7-9 April 2017.

Cisco IOS Release 12.4, NetFlow Configuration Guide, *Cisco Systems*, San Jose, USA, 2011.

Cisco Release 7.x, Cisco Nexus 9000 Series NX-OS System Management Configuration Guide, *Cisco Systems*, San Jose, USA, 2015.

Cohodorow K., Dirolf M., *Couchdb: The Definitive Guide*, 1st ed., O'REILY, United States of America, 2010.

Çam H., Türkiye'deki Üniversitelerde Bulut Bilişim Teknolojisinin Uygulanabilirliğinin Teknoloji Kabul Modeli Yaklaşımıyla Belirlenmesi, Doktora Tezi, Atatürk Üniversitesi, Sosyal Bilimler Enstitüsü, Erzurum, 2012, 319751.

Dayyeh W. A., Assrhani A. Ibrahim K., Estimation of the Shape and Scale Parameters of Pareto distribution Using Ranked Set Sampling, *Statistical Papers*, 2013, **54**(1), 207-225.

Dasgupta N., Practical Big Data Analytics: *Hands-on techniques to implement enterprise analytics and machine learning using Hadoop, Spark, NoSQL and R*, 1st ed., Packt Publishing, Birmingham, UK, 2018.

Deka G. C., Bakshi S., *Handbook of Research on Securing Cloud-Based Databases with Biometric Applications*, 1st ed., IGI Global, United States of America, 2014.

Douglis F., Kaashoek M. F., Scalable Internet Services, *IEEE Internet Comput.*, 2001, **5**(4), 36-37.

Dunn, J. C., A Fuzzy Relative ISODATA Process and its Use in Detecting Compact Well-Separated Clusters, *Journal of Cybernetics*, 1974, **3**(3), 32-57.

Durbin R., Eddy S., Krogh A., Mitchison G., *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, 1st ed., Cambridge, United Kingdom, 1998.

Ebem Ş., Kamu Bilişim Sistemleri Açısından Bulut Bilişimin Teknik, Yönetim ve Hukuki Boyutlarıyla İncelenmesi: Bilgi Teknolojileri ve İletişim Kurumu İçin Öneriler, Uzmanlık Tezi, Bilgi Teknolojileri ve İletişim Kurumu, Ankara, 2013, Yayın No: 0154.

Eken S., Kaya F., Çomak K., Sayar A., Kavak A., Şahin S., Fatih Projesinde Proxy Sunucunun Doğru Konumlandırıldığının Kontrol Edilmesi ve Tabletlerin Yerini Tespit için Bir Metot, *Elektrik-Elektronik, Bilgisayar ve Biyomedikal Mühendisliği Sempozyumu*, Bursa, Türkiye, 27-29 Kasım 2014.

Eken S., Kaya F., İlhan Z., Sayar A., Kavak A., Kocasaraç U., Şahin S., Analyzing Distributed File Synchronization Techniques for Educational Data, *International Conference on Electronics, Computer and Computation (ICECCO)*, Ankara, Turkey, 7-9 November 2013.

Eken S., Kaya F., Sayar A., Kavak A., A Method for Localization of Computational Node and Proxy Server in Educational Data Synchronization, *International Wireless Internet Conference-Symposium on Wireless and Vehicular Communication*, Lisbon, Portugal, 13-14 November 2014.

Eken S., Kaya F., Sayar A., Kavak A., Doküman Tabanlı NoSQL Veritabanları: MongoDB ve CouchDB Yatay Ölçeklenebilirlik Karşılaştırması, *7. Mühendislik ve Teknoloji Sempozyumu*, Ankara, Türkiye, 15-16 Mayıs 2014.

Eren K., Bulut Bilişim Teknolojileri Ve NoSQL Veritabanlari Kullanarak Türkiye'de Terör Olaylarinin İncelenmesi, Yüksek Lisans Tezi, Sakarya Üniversitesi, Sosyal Bilimler Enstitüsü, 2017, 460665.

Eylen T., Bazlamaçı C. F., One-Way Active Delay Measurement with Error Bounds, *IEEE Trans. Instrum. Meas.*, 2015, **64**(12), 3476 – 3489.

Fang Q., Vrbsky S. V., Dang Y., Ni W., A Pull-Based Broadcast Algorithm that Considers Timing Constraints, *Workshops on Mobile and Wireless Networking/High Performance Scientific, Engineering Computing/Network Design and Architecture/Optical Networks Control and Management/Ad Hoc and Sensor Networks/Compil*, Montreal, Canada, 15-18 August 2004.

Fix E., Hodges J L., Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties, *California University*, 21-49-004, 1-21, 1951.

Floyd S., Metrics for the Evaluation of Congestion Control Mechanisms, Network Working Group, https://tools.ietf.org/html/rfc5166 (Ziyaret tarihi: 3 Şubat 2018).

Forney G.D., The Viterbi Algorithm, *Proceedings of the IEEE*, 1973, **61**(3), 268 – 278.

Garsva E., Paulauskas N., Grazulevicius G., Gulbinovic L., Packet Inter-Arrival Time Distribution in Academic Computer Network, *Elektronika ir Elektrotechnika*, 2014, **20**(3), 87-90.

George, C., Dollimore, J., Kindberg, T., Blair G., *Distributed Systems: Concepts and Design*, 5th ed., Addison-Wesley, Boston, 2011.

George L., *HBase: The Definitive Guide*, 1st ed., O'REILY, United States of America, 2011.

Gözüdeli Y., *Veritabanı Programlama*, 1st ed., BYTE (Acar Yayıncılık), Mecidiyeköy, İstanbul, 2003.

Grolinger K., Higashino W. A., Tiwari A., Capretz M. A., Data Management in Cloud Environments: NoSQL and NewSQL Data Stores, *J Cloud Comp*, 2013, **2**(22), 1-24.

Gupta S., Rani R., A Comparative Study of Elasticsearch and CouchDB Document Oriented Databases, *International Conference on Inventive Computation Technologies (ICICT)*, Coimbatore, India, 26-27 August 2016.

Han J., Kamber M., Pei J., *Data Mining Concepts and Techniques*, 3rd ed., Elsevier, USA, 2012.

H¨arder T., Reuter A., Principles of Transaction Oriented Database Recovery, *ACM Comput Surv*, 1983, **15**(4), 287-317.

Ibe O. C., *Markov Process for Stochastic Modelling*, 1st ed., Elsevier, USA, 2009.

IDC, Data Growth, Business Opportunities, and IT Imperatives, EMC Digital Universe with Research & Analysis, Dell EMC, https://www.emc.com/collateral/analyst-reports/idc-digital-universe-2014.pdf (Ziyaret Tarihi: 4 Şubat 2018)

Ikeda H., Kitayama K., Dynamic Bandwidth Allocation with Adaptive Polling Cycle for Maximized TCP Throughput in 10G-EPON, *J. Lightwave Technol.*, 2009, **27**(23), 5508-5516.

Ilie D., Datta V. V. K. S., On Designing a Cost-Aware Virtual CDN for the Federated Cloud, *International Conference on Communications*, Bucharest, Romania, 9-10 June 2016.

Isobe T., Ito D., Akashi D., Tsutsumi S., RADIC-TCP: High-Speed Protocol Applied for Virtual Private WAN, *18th International Conference on Telecommunications*, Ayia Napa, Cyprus, 8-11 May 2011.

Jacob A. K., Jacob L., A Discrete Time Polling Protocol for Wireless Body Area Network, *IEEE International Advance Computing Conference (IACC)*, Gurgaon, India, 21-22 February 2014.

Jain R., Abouzakhar N. S., Hidden Markov Model Based Anomaly Intrusion Detection, *International Conference for Internet Technology and Secured Transactions*, London, UK ,10-12 December 2012.

Joshi S. S., Phoha V. V., Investigating Hidden Markov Models Capabilities in Anomaly Detection, *43rd Annual Southeast Regional Conference*, Kennesaw, Georgia, 18 – 20 March 2005.

Kantardzic M., *Data Mining: Concepts, Methods and Algorithms*, 2nd ed., Wiley, USA, 2011.

Kavak A., Şahin S., Sayar A., Kaya F., Eken S., Khan S. A., Yerel Bulut Sunucu Tabanlı Etkin Bir Veri Saklama ve Içerik Dağıtımı Yönetimi Sistemi, *Tübitak*, 113E033, 1-76, 2014.

Kaya Gülağız F., Eken S., Kavak A., Sayar A., Idle Time Estimation for Bandwidth-Efficient Synchronization in Replicated Distributed File System, *International Arab Journal of Information Technology*, 2018, **15**(2), 177-185.

Kaya Gülağız F., Gök O., Kavak A., A Comparison of Imputation Techniques Using Network Traffic Data, *International Journal of Computer Applications*, 2016, **142**(7), 25-29.

Kenyon T., *Data Networks: Routing, Security and Performance Optimization*, 1st ed., Digital Press, United States of America, 1960.

Kholidy H. A., Erradi A., Abdelwahed S., Attack Prediction Models for Cloud Intrusion Detection Systems, *2nd International Conference on Artificial Intelligence, Modelling and Simulation*, Madrid, Spain, 18-20 November 2014.

Knuth D. E., *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 3rd ed., Addison-Wesley, Boston, USA, 2016.

Koyun A., Kaymakçı Ö. T., Bir Tramway Hattının Güvenilirlik Analizi, *Gazi Üniv. Müh. Mim. Fak. Der.,* 2015, **30**(4), 615-626.

Krogh A., Brown M., Mian I. S., Sjolander K., Haussler D, Hidden Markov Models in Computational Biology Applications to Protein Modelling, *J.Mol. Biol.*, 1994 **235**(5), 1501-1513.

Levesque M., Tipper D., Improving the PTP Synchronization Accuracy Under Asymmetric Delay Conditions, *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication*, Beijing, China, 11-16 October 2015.

Li P., Chen Y., Li T., Wang R., Sun J., Implementation of Cloud Messaging System Based on GCM Service, *IEEE International Conference on Computational and Information Sciences*, Shiyan, Hubai, China, 21-23 June 2013.

Li T., Background Traffic Modeling for Large-Scale Network Simulation, Doctoral Thesis, Florida Inernational University, College of Engineering and Computing, Florida, 2014, 10.25148/etd.FI14040803.

Li Y., Manoharan S., A Performance Comparison of SQL and NoSQL Databases, *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, Victoria, Canada, 27-29 August 2013.

Lin C. F., Leu M. C., Chang C. W., Yuan S. M., The Study and Methods for Cloud Based CDN, *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Victoria, Canada, 23-26 August 2011.

Ling L., Xiaozhen M.,Yulan H., CDN Cloud: A Novel Scheme for Combining CDN and Cloud Computing, *International Conference on Measurement, Information and Control*, Harbin, China, 16-18 August 2013.

Liu Y., Lee S. H., Chon T. S., Analysis of Behavioral Changes of Zebrafish (Danio rerio) in Response to Formaldehyde Using Self-Organizing Map and a Hidden Markov Model, *Ecol. Model.*, 2011, **222**(14), 2191-2201.

Lv Y., Jiang N., Qiu K., Xue C., Energy-Efficient Load Adaptive Polling Sequence Arrangement Scheme for Passive Optical Access Networks, *IEEE J. Opt. Commun. Netw.*, 2015, **7**(6), 516-524.

MacQueen J. B., *Some Methods for classification and Analysis of Multivariate Observations*, Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, 1st ed., University of California Press, USA, 1967.

Macedo T., Oliveira F., *Redis Cookbook*, 1st ed., O'REILY, United States of America, 2011.

Mathis M., A Framework for Defining Empirical Bulk Transfer Capacity Metrics, Network Working Group, https://tools.ietf.org/html/rfc3148 (Ziyaret tarihi: 2 Şubat 2018)

Moniruzzaman A., Hossain S. A., NoSQL Database: New Era of Databases for Big Data Analytics-Classification, Characteristics and Comparison, *IJDTA Journal*, 2013, **6**(4), 1-13.

Mount D. M., *Bioinformatics: Sequence and Genome Analysis,* 2nd ed., Cold Spring Harbor Laboratory Press, New York, USA, 2004.

Nahum E. M., Roşu M. C., Seshan S., Almedia J., The Effects of Wide-Area Conditions on WWW Server Performance, *ACM Sigmetrics Conference on Measurement and Modelling of Computer Systems*, Cambridge, MA, 16-20 June 2001.

Nielsen M. A., Parameter Estimation for the Two-Parameter Weibull Distribution, Master Thesis, Brigham Young University, Physical and Mathematical Sciences; Statistics, 2011, 2509.

Özkan Y., *Veri Madenciliği Yöntemleri,* 1. Basım, Papatya Yayıncılık, İstanbul, Türkiye, 2008.

Papagianni C., Leivadeas A., Papavassiliou S., A Cloud-Oriented Content Delivery Network Paradigm: Modeling and Assessment, *IEEE Transactions on Dependable and Secure Computing*, 2013, **10**(5), 287-300.

Pathan A. M. K., Buyya R., A Taxonomy and Survey of Content Delivery Networks, *Grid Computing and Distributed Systems Laboratory, University of Melbourne*, 4, 1-2, 2007.

Prince S. M., Sloman M. S., Communication Requirements of a Distributed Computer Control System, *IEE P-Comput Dig T*, 1981, **128**(1), 21-34.

Quach T., Farooq M., Maximum Likelihood Track Formation with the Viterbi Algorithm, *33rd Conference on Decision and Control*, Florida, USA, 14-16 December 1994.

Ramage D., Hidden Markov Models Fundamentals, Lecture Notes, http://cs229.stanford.edu/section/cs229-hmm.pdf, (Ziyaret tarihi: 18 Şubat 2018)

Ren Y., Zhao Y., Liu P., Dou K., Li J., A Survey on TCP Incast in Data Center Networks, *Int J Commun Syst*, 2014, **27**(8), 1160-1172.

Ritchie B., *RavenDB High Performance*, 1st ed., Packt Publishing, Birmingham, UK, 2013.

Saxena N., Pinotti C. M. and Das S. K., A Probabilistic Push-Pull Hybrid Scheduling Algorithm for Asymmetric Wireless Environment, *IEEE Global Telecommunications Conference Workshops*, Dallas, TX, USA, 29 November-3 December 2004.

Schneider F., *Distributed Systems*, 2nd ed., ACM Press/Addison-Wesley Publishing, New York, USA, 1993.

Sevli O., Bulut Bilişim ve Eğitim Alanında Örnek Bir Uygulama, Yüksek Lisans Tezi, Süleyman Demirel Üniversitesi, Fen BilimleriEnstitüsü, 2011, 295236.

Son S. H., Kouloumbis S, Replication Control for Distributed Real-Time Database Systems, *12th International Conference on Distributed Computing Systems*, Yokohama, Japan, 9-12 June 1992.

Taha I. A., A Comprehensive Comparison of NoSQL and Relational Database Management Systems, Master Thesis, Çankaya University, Graduate School of Applied and Natural Science, 2017, 460827.

Tandon A., Motani M., A Cross-Layer Approach to Reducing Packet Delay in Polling-Based Multiuser Systems, *IEEE T Veh Technol*, 2017, **66**(2), 1506-1518.

Tanenbaum A. S., Steen M. V., *Distributed Systems Principles and Paradigms*, 2nd ed., Pearson Prentice Hall, USA, 2007.

Tesoriero C., *Getting Started with OrientDB,* 1st ed., Packt Publishing, Birmingham, UK, 2013.

Türe H., Başer F.,Bulanık C-Ortalama Kümeleme Algoritması ile Ülke Risk Değerlendirmesi, *Ekonometri ve İstatistik*, 2015, **23**, 16-33.

Viterbi A., Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm, *IEEE Trans. Inf. Theory*,.1967, **13**(2), 260-269.

Yu-Ting D., Hai-Peng Q., Xi-Long T., Real-Time Risk Assessment Based on Hidden Markov Model and Security Configuration, *International Conference on Information Science, Electronics and Electrical Engineering*, Sapporo, Japan, 26-28 April 2014.

Ulaş A., Hangi NoSQL Veritabanı?, Kod Bilgi Sistemleri ve Yazılım, https://kodcu.com/2016/03/hangi-nosql-veritabani/ (Ziyaret Tarihi: 8 Şubat 2018).

URL-1: http://docs.couchdb.org/en/1.6.1/replication/protocol.html (Ziyaret Tarihi: 19 Şubat 2018).

URL-2: https:// www.riverbed.com/ gb/products/steelcentral/ opnet.html?redirect= opnet (Ziyaret Tarihi: 26 Şubat 2018).

Uzunbayır S., A Comparison Between Relational Database Models And NoSQL Trends On Big Data Design Challenges Usıng A Social Shopping Applicatıon, Master Thesis, Izmir University of Economics, Graduate School of Natural and Applied Science, 2015, 395452.

Venkataramani A., Kokku R., Dahlin M., TCP Nice: A Mechanism for Background Transfers, *5th Symposium on Operating Systems Design and Implementation*, Boston, Massachusetts, 9-11 December 2002.

Venkatesh K., Vahdat A., Evaluating Distributed Systems: Does Background Traffic matter?, *USENIX Annual Technical Conference*, Boston, Massachusetts, 22-27 June 2008.

Venkatesh K., Vahdat A., Swing: Realistic and Responsive Network Traffic Generation, *IEEE/ACM Trans. Netw.*, 2009, **17**(3), 712-725.

Walck C., Hand-Book on Statistical Distributions for Experimentalists, *Particle Physics Group Fysikum University of Stockholm*, SUF–PFY/96–01, 133-137, 2007.

Wan C., Freitas A. A., Magalhães J. P., Predicting the Pro-Longevity or Anti-longevity Effect of Model Organism Genes with New Hierarchical Feature Selection Methods, *IEEE/ACM Trans Comput Biol Bioinform*, 2015, **12**(2), 262-275.

Wang L., Chen D., Hu Y., Ma Y., Wang J., Towards Enabling Cyber Infrastructure as a Service in Clouds, *Comput Electr Eng*, 2013, **39**(1), 3-14.

Wang L., Laszewski G. V., Younge A. J., He X., Kunze M., Tao J., Fu C., Cloud Computing: a Perspective Study. *New Gener Comput*, 2010, **28**(2), 137-146.

Wang M., Jayaraman P. P., Ranjan R., Mitra K., Zhang M., Li E., Khan S., Pathan M., Georgeakopoulos D., An Overview of Cloud Based Content Delivery Networks: Research Dimensions and State-of-the-Art, Editors: Hameurlain A., Küng J., Wagner R., Sakr S., Wang L., Zomaya A., *Transactions on Large-Scale Data- and Knowledge-Centered Systems - Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 131-158, 2015.

Wang Y., Wen X., Sun Y., Zhao Z. and Yang T., The Content Delivery Network System Based on Cloud Storage, *International Conference on Network Computing and Information Security*, Guilin, China, 14-15 May 2011.

Wiesmann M., Pedone F., Schiper A., Kemme B., Alonso G., Understanding Replication in Databases and Distributed Systems, *IEEE International Conference on Distributed Computing Systems*, Taipei, Taiwan, 10-13 April 2000.

Williams N., Yeo K. C., Azam S., Cost Effective Analysis of Web Based Transmission, *IEEE Symposium on Computational Intelligence for Communication Systems and Networks*, Singapore, 16-19 April 2013.

Wolf J. K., Viterbi A. M., Dixon G. S., Finding the Best Set of K Paths Through a Trellis with Application to Multitarget Tracking, *IEEE Trans. Aerosp. Electron. Syst.*, 1989, **25**(2), 287-295.

Yin H., Liu X., Min G., Lin C., Content Delivery Networks: A Bridge Between Emerging Applications and Future IP Networks, *IEEE Netw.*, 2010, **24**(4), 52-56.

Yolal M., Profile Hidden Markov Model, Youtube, https://www.youtube .com/watch?v=rebr-6st_rM (Ziyaret Tarihi: 18 Şubat 2018)

Yüksel H., Bulut Bilişim El Kitabı, http://www.bilisimlife.net/KonuOku/1575-Bulut%20Bili?im%20El%20Kitaby%20E-Kitap.html (Ziyaret Tarihi: 3 Şubat 2018).

EKLER

Ek-A

Aşağıda kullanılan yöntemlere ve önerilen yönteme ilişkin kod bölümleri verilmiştir;

Knn kayıp değerleri elde etme

```
while ((line = br.readLine()) != null) {
 if (satir != 0) {
  String[] words = line.split("\t");

  if (words[1].contains(na) && words[2].contains(na) &&
      words[3].contains(na) && words[4].contains(na))
      {last_line = words[0] + "\t" + "0.0" + "\t" +
       "0.0" + "\t" + "0.0" + "\t" + "0.0";
  } else {
   for (int i = 0; i < words.length; i++) {
    if (words[i].contains(na)) {
     dizi[indis][i] = -1.0;
    } else {
     dizi[indis][i] = Double.parseDouble(words[i]);
    }
   }
   indis++;
  }
 } else {
  first_line = line;
 }
 satir++;
}

br.close();
double ornek_araligi = dizi[1][0] - dizi[0][0];
Double[][] kayipli = new
Double[lines][attribute_count];
Double[][] kayipsiz = new
Double[lines][attribute_count];
int kayipli_say = 0;
int kayipsiz_say = 0;
boolean kontrol = false;
for (int i = 0; i < indis; i++) {
 kontrol = false;
 for (int j = 1; j < 5; j++) {
  if (dizi[i][j] == -1.0) {
   kontrol = true;
   break;
  }
 }
 if (kontrol == true) {
  kayipli[kayipli_say] = dizi[i];
  kayipli_say++;
```

```
  } else {
   kayipsiz[kayipsiz_say] = dizi[i];
   kayipsiz_say++;
  }
 }
double toplam = 0.0;
double mesafe = 0.0;
for (int i = 0; i < kayipli_say; i++) {
 double[][] dist = new double[kayipsiz_say][2];
 for (int j = 0; j < kayipsiz_say; j++) {
  toplam = 0;
  mesafe = 0;
  for (int s = 1; s < 5; s++) {
   if (kayipli[i][s] != -1.0) {
    toplam += Math.pow(kayipli[i][s] -
    kayipsiz[j][s], 2);
   }
  }
  mesafe = Math.sqrt(toplam);
  dist[j][0] = mesafe;
  dist[j][1] = (double)(j);
 }
 bubblesort(dist, kayipsiz_say);
 for (int s = 1; s < 5; s++) {
  if (kayipli[i][s] == -1.0) {
   double sum = 0.0;
   double ort = 0.0;
   for (int x = 0; x < k; x++) {
    int tmp = (int) dist[x][1];
    sum += kayipsiz[tmp][s];
   }
   ort = sum / k;
   if (s == 4)
    kayipli[i][s] = 1.0;
   else {
    kayipli[i][s] = ort;
   }
  }
 }
}
```

Bulanık C-Ortalamalar Algoritması:

Verilerin Kümelere olan aitliklerinin hesabı:

```
for(counter=0; counter<iterasyon; counter++){
    for(i=0; i<n; i++){
        for(j=0; j<c; j++){
            if(veri[i]==center[j]){
                M[i][j]=1;}
```

```
                else{
                    for(k=0; k<c; k++){
                        x=oklid(i,j);
                        if(veri[i]==center[k])
                            y=2;
                        else
                            y=oklid(i,k);
                        z=x/y;
                        denklem += pow(z,(2/(m-1))); }
                    M[i][j]=1/denklem;
                    denklem=0;
                }
            }
        }
        for(j=0; j<c; j++){
            for(i=0; i<n; i++){
                for(t=0; t<s-1; t++)
                    x1[0][t]+=veri[i][t]*pow(M[i][j],m);
            }
            for(i=0; i<n; i++){
                x2+=pow(M[i][j],m);
            }
            for(t=0; t<s-1; t++){
                center[j][t]=x1[0][t]/x2;
            }
        }
        objective();
}
```

Uzaklik hesaplama fonksiyonu:

```
double oklid (int i,int j) {
    double sonuc=0.0;
    double distance=0.0;
    for(t=0;t<4;t++){
        sonuc+=pow(fabs(veri[i][t]-center[j][t]),2);
    }
    distance=sqrt(sonuc);
    return distance;
}
```

Amaç fonksiyonu:

```
void objective (){
    double  sum=0.0;
    int i=0,j=0;
    for(i=0;i<n;i++){
        for(j=0;j<c;j++){
            sum+=pow(M[i][j],m)*pow(oklid(i,j),2);
        }
```

```
    }
    J[counter]=sum;
}
```

Profile Hidden MArkov Model Algoritması:

Modelin Kurulması

```
if ((fp1 = fopen ("ornek.txt", "r")) == NULL)
{
    printf("Dosya acma hatasi!");
    exit(1);
}
else
{
    while(!feof(fp1))
    {
        fgets(row,N,fp1);
        colcount=0;
        while(row[colcount]!=NULL)
        {
            dataarray[rowcount][colcount]=
            row[colcount];
            colcount++;
        }
        rowcount++;
        if(rowcount==10)
            break;
    }
    fclose(fp1);
    rowcount=rowcount-1;
    colcount=colcount-1;
    for(i=0; i<=colcount;)
    {
        county=0;
        countn=0;
        countb=0;
        if(i==0)
        {
            for(j=0; j<rowcount; j++)
            {
                if(dataarray[j][i]=='Y')
                {
                    county++;
                    nodearray[1].satirno[gecici]=j;
                    gecici++;
                }
                else if(dataarray[j][i]=='N')
                {
                    countn++;
```

```
                    nodearray[1].satirno[gecici]=j;
                    gecici++;
                }
                else if(dataarray[j][i]=='B')
                {
                    countb++;
                    nodearray[2].satirno[gecici2]=j;
                    gecici2++;
                }

            }
            nodearray[1].satirno[gecici]=-1;
            nodearray[2].satirno[gecici2]=-1;
            gecici=0;
            gecici2=0;
            nodearray[0].node[0].arrowprob=
            (double)(county+countn)/rowcount;
            nodearray[0].node[1].arrowprob=
            (double)(countb)/rowcount;
        }
        else if(i<(colcount))
        {
            dugum++;
            if((dugum % 2)!=0)
            {
                nodearray[dugum].state=3;
                x=0;
                while(nodearray[dugum].satirno[x]!=-1)
                {
                    if(dataarray[nodearray[dugum].
                    satirno[x]][i-1]=='Y')
                    {
                        county++;
                    }
                    if(dataarray[nodearray[dugum].
                    satirno[x]][i-1]=='N')
                    {
                        countn++;
                    }
                    if(dataarray[nodearray[dugum].
                    satirno[x]][i]=='Y'||
                    dataarray[nodearray[dugum].
                    satirno[x]][i]=='N'){
                        nodearray[dugum+2].
                        satirno[gecici]=
                        nodearray[dugum].satirno[x];
                        gecici++;
                    }
                    else
                    {
```

```
                        nodearray[dugum+3].
                        satirno[gecici2]=
                        nodearray[dugum].satirno[x];
                        gecici2++;
                    }
                    x++;
                }
                if((county+countn)!=0)
                {
                    nodearray[dugum].outputy=
                   (double)county/(county+countn);
                    nodearray[dugum].outputn=
                   (double)countn/(county+countn);
                }
                else
                {
                    nodearray[dugum].outputy=0.0;
                    nodearray[dugum].outputn=0.0;
                }
                nodearray[dugum].outputb=0;
                if((gecici+gecici2)!=0)
                {
                    nodearray[dugum].node[0].arrowprob=
                   (double)gecici/(gecici+gecici2);
                    nodearray[dugum].node[1].arrowprob=
                   (double)gecici2/(gecici+gecici2);
                }
                else
                {
                    nodearray[dugum].node[0].
                    arrowprob= 0.0;
                    nodearray[dugum].node[1].
                    arrowprob= 0.0;
                }
            }
            else
            {
                county=0;
                countn=0;
                countb=0;
                nodearray[dugum].state=2;
                nodearray[dugum].outputy=0.0;
                nodearray[dugum].outputn=0.0;
                nodearray[dugum].outputb=1.0;
                x=0;
                while(nodearray[dugum].satirno[x]!=-1)
                {
                    if(dataarray[nodearray[dugum].
                    satirno[x]][i]=='Y'||
                    dataarray[nodearray[dugum].
```

```
                    satirno[x]][i]=='N')
                    {
                        nodearray[dugum+1].
                        satirno[gecici]=
                        nodearray[dugum].satirno[x];
                        gecici++;
                        county++;
                    }
                    else
                    {
                        nodearray[dugum+2].
                        satirno[gecici2]=
                        nodearray[dugum].satirno[x];
                        gecici2++;
                        countb++;
                    }
                    x++;
                }
                nodearray[dugum+1].satirno[gecici]=-1;
                nodearray[dugum+2].satirno[gecici2]=-1;
                if((county+countb)!=0)
                {
                    nodearray[dugum].node[0].arrowprob=
                    (double)county/(county+countb);
                    nodearray[dugum].node[1].arrowprob=
                    (double)countb/(county+countb);
                }
                else
                {
                    nodearray[dugum].node[0].
                    arrowprob= 0.0;
                    nodearray[dugum].node[1].
                    arrowprob= 0.0;
                }
            }
        }
        else if(i==colcount)
        {
            dugum++;
            if((dugum % 2)!=0)
            {
                nodearray[dugum].state=3;
                x=0;
                while(nodearray[dugum].satirno[x]!=-1)
                {
                    if(dataarray[nodearray[dugum].
                    satirno[x]][i-1]=='Y')
                    {
                        county++;
                    }
```

```
                    if(dataarray[nodearray[dugum].
                    satirno[x]][i-1]=='N')
                    {
                        countn++;
                    }

                    x++;
                }
                if((county+countn)!=0)
                {
                    nodearray[dugum].outputy=
                   (double)county/(county+countn);
                    nodearray[dugum].outputn=
                   (double)countn/(county+countn);
                }
                else
                {
                    nodearray[dugum].outputy=0.0;
                    nodearray[dugum].outputn=0.0;
                }

                nodearray[dugum].outputb=0;
                nodearray[dugum].node[0].
                arrowprob= 1.0;
                nodearray[dugum].node[1].
                arrowprob= 1.0;
            }
            else
            {
                county=0;
                countn=0;
                countb=0;

                nodearray[dugum].state=2;
                nodearray[dugum].outputy=0.0;
                nodearray[dugum].outputn=0.0;
                nodearray[dugum].outputb=1.0;
                x=0;
                nodearray[dugum].node[0].
                arrowprob= 1.0;
                nodearray[dugum].node[1].
                arrowprob= 1.0;
            }
        }
        if(dugum==i*2 )
        {
            i++;
            gecici=0;
            gecici2=0;
        }
```

```
    }
    i=(colcount*2)+1;
    nodearray[i].state=1;
}
```

Viterbi Algoritması

```
if ((fp3 = fopen ("cikti.txt", "a+")) == NULL)
{
    printf("Dosya acma hatasi!");
    exit(1);
}
else
{
    for(; i<colcount; i++)
    {
        if(nodearray[j].node[0].arrowprob>=
        nodearray[j].node[1].arrowprob)
        {
            prob=nodearray[j].node[0].arrowprob;
            if(i==0)
                j=i+1;
            else
            {
                if(j%2==0)
                    j=j+1;
                else
                    j=j+2;
            }
            if(nodearray[j].outputy>=
            nodearray[j].outputn)
            {
                e=nodearray[j].outputy;
                last_string[i]='Y';
                fprintf(fp3,"Y");
                printf("%c", last_string[i]);
            }
            else
            {
                e=nodearray[j].outputn;
                fprintf(fp3,"N");
                last_string[i]='N';
                printf("%c", last_string[i]);
            }
            if(i==0)
            {
                prevv=1;
                v=e*prob*prevv;
            }
            else
```

```
                {
                    prevv=v;
                    v=e*prob*prevv;
                }
            }
            else
            {
                prob=nodearray[j].node[1].arrowprob;

                if(i==0)
                    j=i+2;
                else
                {
                    if(j%2==0)
                        j=j+2;
                    else
                        j=j+3;
                }
                e=1;

                if(i==0)
                {
                    prevv=1;
                    v=e*prob*prevv;
                }
                else
                {
                    prevv=v;
                    v=e*prob*prevv;
                }
                fprintf(fp3,"B");
                last_string[i]='B';
                printf("%c", last_string[i]);
            }
        }
        fclose(fp3);
}
```

KİŞİSEL YAYIN VE ESERLER

Kaya Gülağız F., Gök O., Şahin S., Identifying the General Pattern of the Academic Computer Networks Based on Users Daily Behaviors, *Karaelmas Fen ve Mühendislik Dergisi*, 2018, Kabul edildi.

Kaya Gülağız F., Eken S., Kavak A., Sayar A., Idle Time Estimation for Bandwidth-Efficient Synchronization in Replicated Distributed File System, *The International Arab Journal of Information Technology*, 2018, **15**(2), 177-185.

Kaya Gülağız F., Şahin S., Comparison of Hierarchical and Non-Hierarchical Clustering Algorithms, *International Journal of Computer Engineering and Information Technology*, 2017, **9**(1), 6-14.

Kaya Gülağız F., Gök O., Estimation of Synchronization Time in Cloud Computing Architecture, *International Conference on Mobile Networks and Management*, Abu Dhabi, United Arab Emirates, 23-24 October 2016.

Kaya Gülağız F., Gök O., Kavak A., A Comparison of Imputation Techniques using Network Traffic Data, *International Journal of Computer Applications*, 2016, **142**(7), 25-29.

Kaya Gülağız F., Kavak A., A Study Based on Clustering Techniques in Data Mining Comparison of Farthest First Clustering Density Based Clustering K means and K Medoids Algorithms, *World Conference on Innovation and Computer Science*, Antalya, Turkey, 12-14 May 2016.

Kaya Gülağız F., Eken S., Kavak A., Sayar A., Estimation of Idle Time for Synchronization: A Case Study on Fatih Project, *23th Signal Processing and Communications Applications Conference (SIU)*, Malatya, Turkey, 16-19 May 2015.

Eken S., **Kaya F.**, Sayar A., Kavak A., Doküman Tabanlı NoSQL Veritabanları: MongoDB ve CouchDB Yatay Ölçeklenebilirlik Karşılaştırması, *7. Mühendislik ve Teknoloji Sempozyumu*, Ankara, Türkiye, 15-16 Mayıs 2014.

Eken S., **Kaya F.**, Sayar A., Kavak A., Tracking a Single Node's Availability for Communication by Means of Observing Local System Resources**,** IEEE International Symposium on *Innovations in Intelligent Systems and Applications (INISTA)*, Alberobello, Italy, 23-25 June 2014.

Eken S., **Kaya F.**, Güngör A, Sayar A., Kavak A., Sahin S., Bulut Vekil Sunucu ve Tabletler Arasında Veri Senkronizasyonu için RTT Kullanarak Farklı Kümeleme Metotlarının Değerlendirilmesi, *Elektrik – Elektronik – Bilgisayar ve Biyomedikal Mühendisliği Sempozyumu*, Bursa, Türkiye, 27-29 Kasım 2014.

Eken S., **Kaya F.**, Zana İ., Sayar A., Kavak A., Kocasaraç U., Şahin S., Analyzing Distributed File Synchronization Techniques for Educational Data**,** *International*

Conference on Electronics, Computer and Computation (ICECCO), Ankara, Turkey, 7-9 November 2013.

ÖZGEÇMİŞ

Fidan Kaya Gülağız 1988 yılında Kütahya'nın Tavşanlı ilçesinde doğdu. İlk, orta ve lise öğrenimini Kütahya'da tamamladı. 2006 yılında girdiği Kocaeli Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nden 2010 yılında, bölüm birincisi olarak mezun oldu. 2011 yılında başladığı Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Bilgisayar Bilimleri Anabilim Dalı'ndaki Yüksek Lisans eğitimini 2013 yılında tamamladı. 2013 yılından beri Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Bilgisayar Bilimleri Anabilim Dalı'nda doktora eğitimine devam etmektedir. Aynı zamanda 2012 yılından itibaren Kocaeli Üniversitesi Bilgisayar Mühendisliği bölümünde Araştırma Görevlisi olarak çalışmaktadır.