
GAZİ ÜNİVERSİTESİ
Gazi
1926

YAZILIM TANIMLI AĞLARDA TELEMETRİK YÖNTEMLE GERÇEK ZAMANLI SALDIRI TESPİTİ VE ÖNLEME

Çağdaş KURT

DOKTORA TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OCAK 2020

Çağdaş KURT tarafından hazırlanan "YAZILIM TANIMLI AĞLARDA TELEMETRİK YÖNTEMLE GERÇEK ZAMANLI SALDIRI TESPİTİ VE ÖNLEME" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Bilgisayar Mühendisliği Ana Bilim Dalında DOKTORA TEZİ olarak kabul edilmiştir

Danışman: Prof. Dr. O. Ayhan ERDEM

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ...……..……..

Başkan: Doç. Dr. Sevil ŞEN AKAGÜNDÜZ

Bilgisayar Mühendisliği Ana Bilim Dalı, Hacettepe Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ……..……

Üye: Doç. Dr. Hüseyin POLAT

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ...……

Üye: Doç. Dr. İbrahim Alper DOĞRU

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ...……

Üye: Dr. Öğr. Üyesi Mustafa YENİAD

Bilgisayar Mühendisliği Ana Bilim Dalı, Ankara Yıldırım Beyazıt Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ...……...

Tez Savunma Tarihi: 27 / 01 / 2020

Jüri tarafından kabul edilen bu tezin Doktora Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

……..……

Prof. Dr. Sena YAŞYERLİ

Fen Bilimleri Enstitüsü Müdürü

ETİK BEYAN

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

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

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

Çağdaş KURT

27 / 01 / 2020

YAZILIM TANIMLI AĞLARDA TELEMETRİK YÖNTEMLE
GERÇEK ZAMANLI SALDIRI TESPİTİ VE ÖNLEME
(Doktora Tezi)

Çağdaş KURT

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Ocak 2020

ÖZET

Bu tez çalışmasında, Yazılım Tanımlı Ağlarda gerçek zamanlı anormallik ve saldırı tespiti yapan ve ayrıca tespit edilen saldırıları önleyen bir model geliştirilmiştir. Model, anormallik tespitini doğru yapabilmenin yanı sıra bu anormallik tespit modulüne veri sağlayacak ölçüm yöntemine yoğunlaşmış; gerçek zamanlı, düşük maliyetli, model tabanlı bir ölçüm metodu tasarlanmış ve kodlanmıştır. Yaygınlaşan IoT altyapıları ile çok fazla cihazın geniş bant genişlikleriyle ağlara bağlanabilir olması sebebiyle ağ ölçümü bu tezde büyük veri bakış açısıyla ele alınmış, geliştirilen model tabanlı akış ölçümü mekanizması ile ölçeklenebilirlik arttırılmaya çalışılmıştır. Böylesi yüksek miktarda gelen ölçüm verisi içinden normal ve anormal trafik örüntülerini yüksek doğrulukta ve düşük gecikmeyle ayırt edebilmek amacıyla üstel düzleştirme metodu geliştirilerek kullanılmıştır. Bu çalışma, model tabanlı akış ölçümü ve üstel düzleştirme yöntemlerini temele almaktadır. Ölçüm verilerinin modellenmesinde YANG, serileştirilmesinde GPB, taşınmasında gRPC, kodlanmasında Python, anormallik tespitinde Holt'un tahmin algoritması doğruluğu arttırmak ve yanlış alarmları azaltmak amacıyla adaptif hata sabiti, servis seviyesi kontrolü ve kademeli devreye alma yöntemleriyle geliştirilerek kullanılmıştır. Tüm geliştirmeler gerçek ağ trafikleri üzerinde uygulandıktan sonra sistemin anormallik tespit etme başarısı %92 olarak bulunmuştur.

Bilim Kodu	:	92407
Anahtar Kelimeler	:	Yazılım tanımlı ağlar, Model tabanlı akış ölçümü, Anormallik tespiti, Gerçek zamanlı saldırı önleme, Ağ ölçümü
Sayfa Adedi	:	65
Danışman	:	Prof. Dr. O. Ayhan ERDEM

REAL-TIME ANOMALY DETECTION AND MITIGATION
USING STREAMING TELEMETRY IN SDN
(Ph. D. Thesis)

Çağdaş KURT

GAZİ UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

January 2020

ABSTRACT

In this thesis, real-time anomaly detection and mitigation system is developed for Software Defined Networks. In addition to anomaly detection accuracy, the model mainly focuses on the measurement method with granular, real-time, low over-headed, model-driven streaming telemetry abilities that provides metrics to the next module. Owing to the fact that IoT infrastructures are becoming common and popular with many devices in highly available bandwidths, the measurement approach is addressed as a big data problem in this study. The scalability of the model driven streaming telemetry measurement mechanism has been tried to be increased. Exponential Smoothing method is used in order to provide high-accuracy and low-latency in recognizing normal and abnormal traffic patterns in such a huge amount of streamed data. The underlying approaches of this study are modeled streaming telemetry and exponential smoothing. YANG data model for modeling, GPB for encoding, gRPC for transport, Python for coding have been used in flow measurement part. Holt's prediction algorithm is powered by adaptive error constant, service check and gradual activation features to increase high-accuracy and decrease false-positives. The success rate of the developed system that is emulated in real traffic is calculated as 92% after all enhancements are applied.

Science Code	:	92407
Key Words	:	Software defined networks, Model-based streaming telemetry, Anomaly detection, Realtime anomaly mitigation, Network measurement
Page Number	:	65
Supervisor	:	Prof. Dr. O. Ayhan ERDEM

TEŞEKKÜR

Bu tezin hazırlanma sürecinde ve tüm yükseköğrenimim boyunca bana yol gösteren, değerli destek ve katkılarıyla bugünlere gelmeme vesile olan saygıdeğer danışman hocam Prof. Dr. O. Ayhan ERDEM'e, yine tüm yükseköğrenim hayatıma değerli dokunuşları ile hem tez hem de mesleki gelişimime katkı sağlayan kıymetli hocam Doç. Dr. Hüseyin POLAT'a, tez sürecindeki akademik katkı ve desteklerini esirgemeyen tez izleme komitesi üyesi değerli hocam Dr. Öğr. Üyesi Mustafa YENİAD'a en içten teşekkür ve saygılarımı sunarım.

Yoğun doktora öğrenimim süresince değerli desteğini ve sonsuz anlayışını esirgemeyen hayat arkadaşım Merve KURT'a, maddi ve manevi sonsuz destekleriyle bugünlere gelişimin mimarı olan canım aileme ithaf ediyorum.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

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

Simgeler	Açıklamalar
KB	Kilobyte
ms	Milisaniye
sn	Saniye

Kısaltmalar	Açıklamalar
API	Application programming interface
ARIMA	Autoregressive integrated moving average
BSS	Business support system
CLI	Command line interface
CPU	Central processing unit
CUSUM	Cumulative sum
DNS	Domain name server
D/DoS	Distributed denial of service
GARCH	Generalized autoregressive conditional heterosked
GPB	Google protocol buffers
gRPC	Google remote procedure call
HTTP	Hyper text transfer protocol
IETF	Internet engineering task force
IDS	Intrusion detection system
IOS	Internetwork operating system
IP	Internet protocol
IPS	Intrusion prevention system
JSON	Javascript object notation
MAD	Median absolute deviation
MIB	Management information base

Kısaltmalar	Açıklamalar
MİB	Merkezi işlem birimi
MSE	Mean squared error
NAT	Network address translation
NETCONF	Network configuration protocol
OSS	Operational support system
ONF	Open network foundation
REST	Representational state transfer
RPC	Remote procedure call
SDN	Software defined networks
SNMP	Simple network management protocol
SOAP	Simple object access control
SOM	Self organizing map
SSH	Secure shell
TCP	Transmission control protocol
TLS	Transport layer security
UDP	User datagram protocol
XML	Extensible markup language
W3C	World wide web consortium
YANG	Yet another next generatation
YTA	Yazılım tanımlı ağlar
ZSVT	Zaman serisi veri tabanı

1. GİRİŞ

Günümüzde bilişim teknolojileri altyapısına sahip her kuruluş için ağın tasarlandığı şekilde ve yüksek başarımla çalıştığından emin olmak operasyonel, stratejik ve mali açılardan oldukça önemlidir. Gelişen yazılım dünyasıyla birlikte uygulamalar daha gerçek zamanlı ve düşük gecikmeli erişim altyapılarına ihtiyaç duymaktadır. Ağ başarımına yönelik servis kalitesi, trafik mühendisliği, kısıtlı bant genişliği gibi unsurların ve paket kaybı, gecikme, dalgalanma gibi başarım ölçütlerinin ihtiyaca en uygun şekilde kullanılabilmesi için ağ ölçümü kritik öneme sahiptir. Tüm bunlara ek olarak, bilişim teknolojileri altyapsılarına yapılan yatırımların oldukça yüksek maliyetleri bulması sebebiyle bu yatırımların etkin kullanımı ve kaynak planlama açısından ağ ölçüm ve izleme araçları faydalı bilgiler sağlamaktadır. Ayrıca, faturalama sistemleriyle birleşiminde ölçüm teknolojileri kritik öneme sahiptir. Ağ izleme ve ölçüm altyapılarının tüm bu sayılan faydaları ve önemi sebebiyle ölçüm doğruluğunu, gerçek zamanlılığını, genişleyebilirliği, programlanabilirliği ve ağın bir kısmında oluşan kısa süreli veya küçük hacimli trafikleri saptama başarısını arttırmaya, ölçüm maliyetini ise düşürmeye yönelik bu alanda birçok çalışma yapılmaktadır [1-5].

Bilişim altyapılarında izleme ve ölçüm mekanizmalarının en temel görevlerinden bir diğeri ağda güvenliği arttırmak, ağda olup biten olaylardan haberdar etmek, trafik ağırlıklarının belirlenmesini kontrol dışı/olağandışı trafiklerin oluştuğu ana en yakın sürede fark edilmesini ve engellenmesini sağlamaktır. Tüm bu gereksinimler için ağ izleme ağ yöneticilerine oldukça değerli bilgiler sağlamakta, ağ izleme sistemleri anormallik ve saldırı tespitinde başlıca araçlardan olmaktadır.

Saldırıların gerçek zamanlı veya gerçek zamanlıya yakın zaman diliminde tespit edilebilmesi, izleme yönteminin aynı hızda ve doğrulukta cevap verebilir olmasıyla doğru orantılıdır. Ağda oluşan anormalliklerin doğru ve hızlı tespit edilip buna göre aksiyon alınabilmesi için kullanılacak izleme/ölçüm yönteminin metrik toplayıcı tipi, akış veri yapısı, izleme mimarisinin çekme veya itme tabanlı olması, örneklem yoluyla veya olay tabanlı tetiklenmesi gibi özelliklerinin oldukça önemli olduğu görülmektedir. Ayrıca, bant genişliği, merkezi işlem birimi (MİB, CPU), akış tabloları gibi ağ kaynaklarının ve paket kaybı, gecikme, dalgalanma gibi başarım ölçütlerinin gerçek zamanlı durum tespiti

sayesinde, kaynak tüketme veya ağın olağan işleyişini etkileme saldırılarında önemli birer uyarıcı olduğu, bu kapsamda Yazılım Tanımlı Ağ (YTA) yaklaşımının ağdaki olaylara ilişkin gerçek zamanlı, doğru, düşük maliyetli çözümler sunduğu görülmektedir.

Birçok farklı tip ve desende veri üreten cihazların ağa dahil olması, büyük miktarlarda izleme verisi ve trafik örüntüsü oluşturmakta, bu büyük veri içinden değerli bilgilerin üretilmesi önemli bir sorun haline gelmektedir [2]. Kontrol ve veri düzlemlerinin ayrılması prensibine dayanan YTA kavramı, güvenlik duvarları, erişim, ağ izleme ve ölçüm gibi eskiden oldukça zor ve karmaşık olan bu tarz işlevleri çok daha kolay ve pratik hale getirmektedir. Bunun yanı sıra YTA'ların hızla yaygınlaşması, genişleyen ağların denetim ve işletimi için izleme/yönetim yazılımlarına olan ihtiyacı doğurmaktadır.

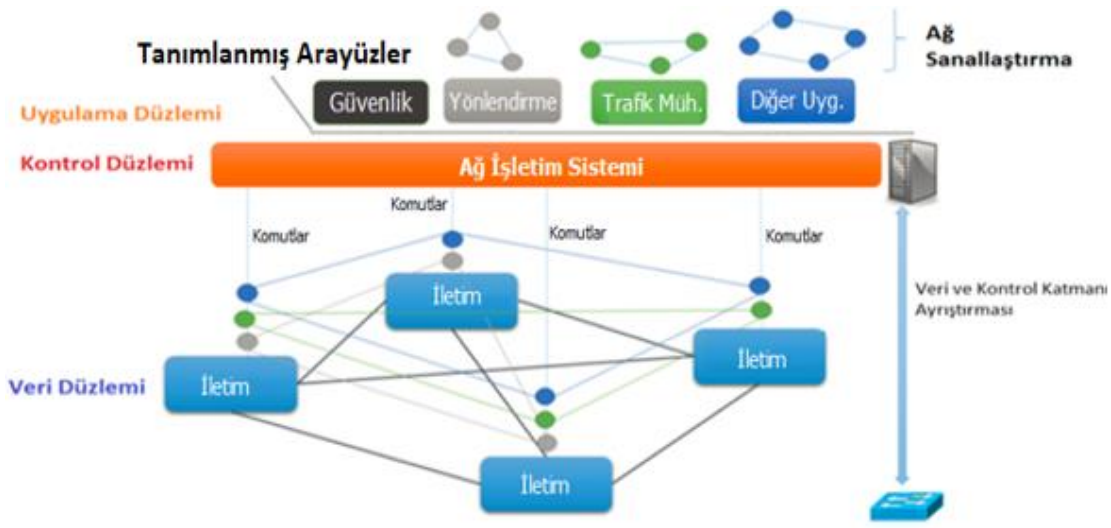
Tüm bu ihtiyaçlar çerçevesinde yapılan araştırmalar gösteriyor ki; anormallik ve saldırı tespitinde kullanılan izleme yönteminde aranan verim beklentilerinin yanı sıra, yüzlerce sistem ve servisin çalıştığı geniş ölçekli ağ ortamlarında üretilen ölçüm verilerinin tekilleştirilmesi ve yapısallaştırılması başarıma, programlanabilirlik, ölçeklenebilirlik ve gerçek zamanlılığa etki etmektedir. Veri akışına veri yapıları ile getirilen modelleştirme, serileştirme ve bu verilerin kullanacak sistem için hazır halde sunulması özellikleri sayesinde ağ cihazlarının çok yüksek boyut ve miktarlardaki ölçüm ve trafik bilgileri cihaz dışına gecikmesiz ve programlanabilir olarak çıkarılabilmektedir. Modelli, yapısal verilerin depolanması ve işlenmesi gerçek zamanlılık ve çok-paydaşlılık gerektiren günümüz haberleşme sistemleri ihtiyaçları için önemli bir çözüm sunduğu görülmektedir [3].

Bu tez çalışmasında, büyük veri gereksinimleri doğrultusunda doğru ölçüm yapabilen, çok geniş ölçekte uygulanabilir, yazılım yetenekleri ile programlanabilir ağ ölçüm yöntemi ihtiyacına yönelik "*Yazılım Tanımlı Ağlarda Telemetrik Yöntemle Gerçek Zamanlı Saldırı Tespiti ve Önleme*" sistemi geliştirilmiş, geliştirilen sistem saldırı ve anormallik tespitine uygulanarak saldırıların gerçek zamanlı ve doğru olarak saptanabilmesi amaçlanmıştır. Tezin 2. bölümünde Yazılım Tanımlı ağ kavramı ele alınmış, bu alanda yaygın kullanılan ölçüm metotları ve anormallik tespit yaklaşımları detaylıca incelenmiştir. 3. bölümde geliştirilen sistemde kullanılan protokol, teknik ve teknolojilere yer verilmiş, 4. bölümde önerilen sistem tüm bileşenleri ve geliştirmeleriyle birlikte anlatılmıştır. 5. bölümde geliştirilen sistemin testi için oluşturulan benzetim ortamı hakkında detaylar ve sistemin test sonuçları verilmiştir. Son bölümde test sonuçları ve çalışma başarısı ele alınmıştır.

2. LİTERATÜRDE YER ALAN ÖNCEKİ ÇALIŞMALAR

ONF'ye (Open Network Foundation) göre YTA, doğrudan programlanabilen ve ağ kontrolünün iletimden ayrıştırıldığı bir yapıyla ortaya çıkan ağ mimarisi olarak tanımlanmaktadır [1]. Ağ üzerindeki kontrol ve veri katmanlarının birbirinden ayrılması prensibine dayanır. Daha önce her bir ağ donanımında dağıtık olarak yer alan kontrol fonksiyonu, yeni mimaride merkezileştirilmiş, uçlar ise kontrol katmanından aldığı sonuçlara göre iletim görevlerini yerine getiren, akıllı olmayan bir misyon yüklemiştir.

Başka bir ifadeyle YTA, fiziksel katmanda bulunan ağ mimarisi bileşenlerini soyutlar, böylece ağ mimarisi mantıksal ya da sanal bir kaynak çiftliği haline gelir. Günümüz tanımında YTA'ya ait kontrol katmanı, veri katmanı ve uygulama katmanı olmak üzere 3 temel unsur bulunmaktadır. Bu katmanlar birbirine programlanabilen arayüzler (Application Programming Interface, API) ile bağlanır. Kontrol katmanı, cihaz üzerinde çalışan protokollerin kararlarının belirlendiği merkezi kısımdır. Yönlendirme, güvenlik duvarı, servis kalitesi, trafik mühendisliği, yük dengeleme gibi fonksiyonların çıktılarının üretildiği katmandır. Veri katmanı, kontrol katmanından alınan bilgi ve kurallara göre trafiğin anahtarlanmasından sorumludur. OpenFlow [2], bu iki katmanın birbiriyle haberleşmesinde yaygın kullanılan protokoldür. Uygulama katmanı ise veri ve kontrol katmanlarının yönetimine ilişkin yazılımların bulunduğu, en üstte bulunan, ağ servislerinin yönetiminden sorumlu katmandır. YTA'nın katmanlı mimarisi Şekil 2.1'de gösterilmiştir.

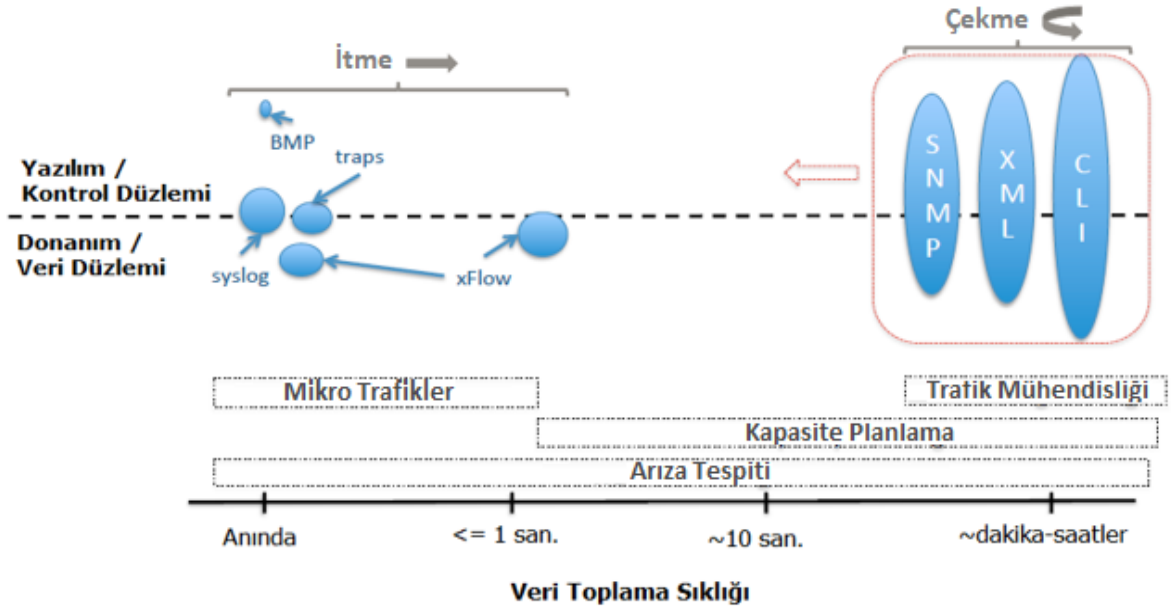


Şekil 2.1. YTA katmanlı mimarisi

YTA mimarisi, altyapıların tamamını tek bir kontrol panelinde birleştirip yönetilmesini ve otomatize edilmesini sağlamaktadır. Bu sayede gerek yeni teknolojilerin uygulanmasını gerekse mevcut sistemler üzerinde yapılması gereken değişiklikleri çok daha hızlı gerçekleştirme imkânı oluşmaktadır. Ayrıca ağ ve sistem yöneticilerinin işlerini kolaylaştırırken son kullanıcıların da ağ servislerine çok daha hızlı erişebilmeleri hedeflenir. YTA kavramı, güvenlik duvarları, erişim, ağ izleme ve ölçüm gibi eskiden oldukça zor ve karmaşık olan bu tarz işlevleri çok daha kolay ve basit bir hale getirmektedir. Geleneksel ağ ölçüm yöntemlerinin hemen hemen hepsi uç cihazlardaki değişkenlerde tutulan yönetim bilgi birimi (Management Informatin Base, MIB) değerlerin biriktirilip işlenmek üzere merkezdeki sunuculara gönderilmesi, daha sonra bu değerlerin analiz edilerek ağın durumuna ilişkin çıkarım yapma esasına dayanmaktadır [3].

2.1. Ölçüm Yöntemleri

Geleneksel ağ ölçüm protokollerinden en bilineni olan basit network yönetim protokolü (Simple Network Management Protocol, SNMP) [3] "çekme" mimarisiyle çalıştığından talep ve cevap paketleri MİB'da kaynak tüketmekte, bu aynı zamanda talep-cevap zaman döngüsünden küçük olan anormallik, saldırı vb. ağ olaylarının yakalanmasına engel olmaktadır. Geleneksel ağ ölçüm yöntemlerine ilişkin zaman/olay saptama grafiği Şekil 2.2'de verilmiştir.



Şekil 2.2. Geleneksel ölçüm yöntemlerinin zaman bazlı karşılaştırması [4]

Şekil 2.2'den görüleceği üzere komut satırı arayüzü (command line interface, CLI), SNMP gibi çekme tabanlı yöntemler yaygın olarak daha geniş zamanlı uygulamalarda kullanılırken küçük yoğunluklu veya hızlı gelişen, saldırı, anormallik, gerçek-zamanlılık gereksinimi gerektiren olayların tespitinde yetersiz kalmaktadır. Ölçüm verilerine ulaşmak için uç cihaza talep gönderme şeklinde kullanılan çekme-tabanlı geleneksel yöntemlerin bir diğer dezavantajı da bu talepleri düzenli ve pratik hayatta en yüksek frekansla 5 dakikalık aralıklarla yapabiliyor olmasıdır. Yaygınlaşan gerçek-zamanlı uygulama sistemleri ve saldırı tipleri 5 dakikalık zaman döngülerinden çok daha kısa sürelerde aksiyon almayı zorunlu kılmaktadır. Örneğin, gerçek zamanlı ses ve video iletim uygulamalarının çoklu kullanıcıların yer aldığı karmaşık ağ ortamlarında bant genişliği düzenlemelerini saniyeler mertebesinde yapabilmesi ihtiyacı günden güne artmaktadır. Ya da ağda kısacık bir zaman döngüsünde gerçekleşen saldırı veya anormalliğin tespiti için 5 dakikalık bir döngü o değişikliğin tespitini zorlaştırabilmektedir.

Çekme ve itme tabanlı ağ izleme mimarilerinin farkları Çizelge 2.1'de verilmiştir. Akış tabanlı mimaride ölçüm değerleri izlenen sistemler tarafından düzenli sıklıkta merkezi toplama birimine gönderilir. sFlow, StatsD, collectd, Ganglia, Graphite mimarinin örnekleridir. Çekme tabanlı mimaride ise merkezi toplama birimi ölçüm değerlerini izlenen sistemlerden düzenli sıklıkta talep eder.

Rastgele trafik örnekleme yöntemi kullanan sFlow [5] yaklaşımı gerçek zamanlıya yakın veri sağlayabilmektedir. Ağ anormalliklerinin tespitinde [6], yönlendirme protokollerinin iyileştirilmesinde [7] ve daha birçok çalışmada [8-13] yaygın kullanılan bir yöntemdir. Dezavantajı ise rasgele alınan örneklerin her zaman gerçek ağ durumunu vermeyebilmesidir.

Cisco firmasının bir geliştirmesi olan NetFlow, sFlow'a benzer şekilde ağdan akış örneklemleri alarak ağ durumuna ilişkin çıkarımlar yapmaktadır. Büyük ölçekli akışların elde edilmesinde ve trafik durumlarının analiz edilmesinde yaygın kullanılan bu yaklaşımın da dezavantajı küçük ölçekli akışların tespit edilmesinde sapma oranlarının yüksek olmasıdır.

Çizelge 2.1. Çekme ve itme tabanlı ölçüm yöntemlerinin karşılaştırması

Özellik	İtme Tabanlı Ölçüm	Çekme Tabanlı Ölçüm
Keşif	Ajan, ölçüm verilerini çalışmaya başladığı andan itibaren ve sürekli olarak yollar. Keşif hızı ajanların sayısından bağımsızdır.	Yeni ajanların keşfi, toplayıcının izlediği ajan adres uzayını sürekli genişletmesine bağlıdır. Keşif hızı bu işlemin periyodik aralıklarına ve adres uzayının büyüklüğüne bağlıdır.
Ölçekleme	Veri gelmesi görevi ajanlara dağılmış durumdadır. Merkezi toplayıcı sadece ölçüm verileri için dinleme ve gelen verileri depolama yapar. Sabit büyüklükteki ölçüm değerlerini gönderen ajanlar düşük yükte çalışır. Ölçüm verisi oluşur oluşmaz gönderim yapılır, haberleşmede durum denetimi yapılmaz.	Toplayıcının iş yükü izlediği uç cihazların sayısıyla doğru orantılı olarak artar. Ek olarak, talep mesajları ve cevaplardan oluşan oturumların yönetilmesi gerekir. Ajan tarafında taleplerin okunması ve işlenmesi yükü vardır. Ajanlar ayrıca tekrar istenen/istenecek metriklerin durumunu takip etmek zorundadır.
Güvenlik	Ajanlar ağ trafiğini dinlemediğinden, çalışma mantıkları gereği dışsal bir tetiklemeye ihtiyaç duymadıklarından, dışarıdan gelecek ataklara karşı doğal olarak güvenlidir.	Veri gönderimi için dışarıdan tetikleme beklediklerinden uzak erişime dayalı saldırılara ve dağıtık servis dışı bırakma (D/DoS) ataklarına müsaittir.
Yönetim Zorluğu	Ajanlar için tek ihtiyaç olan bilgi gönderim sıklığı ve toplayıcı adresidir. Güvenlik duvarlarının, ajanlardan toplayıcıya olacak şekilde tek yönlü iletişim için ayarlanması yeterlidir.	Toplayıcının ölçüm verisi alacağı ajanların listesini, kullanıcı ad/şifre bilgilerini ve istekte bulunacağı ölçüm verilerini bilmesi gerekir. Güvenlik duvarlarının çift yönlü haberleşmeye izin vermesi gereklidir.
Gecikme	Düşük yük getirmesi ve dağıtık mimarisi sebebiyle itme mimarisi daha sık veri gönderimine ve yönetim yazılımının değişiklikler karşısında hızlı aksiyon almasına olanak verir. Bu mimaride yazılan birçok UDP temelli protokol düşük gecikmeli metrik taşımasına yardım eder.	Ölçüm verilerinin belli periyotlarla sunucu tarafından elde edilmesi ağ başarımı ve değişiklikler hakkında gecikmeli bilgi edinmeye sebep olabilmektedir. Çift yönlü haberleşme mimarisi ve kimlik doğrulama mekanizması gecikmeyi arttıran diğer etmenlerdir.
Esneklik	Toplayıcı, istediği bir zamanda istediği ölçüm verisini talep edebilir.	Önceden tanımlı ölçüm listesi sebebiyle nispeten az esnektir.

YTA'ların yukarıda bahsedilen programlanabilirlik, katmanların ayrılığı gibi yenilikçi yaklaşımları ile ağ ölçümünde yaygın karşılaşılan problemler neticesinde geliştirilen, yazılım tanımlı ağ izleme ve ölçüm yöntemlerinden bazıları aşağıda verilmiştir.

FlowSense [14], OpenFlow kontrol paketlerini analiz ederek, sıfır maliyetle ölçüm yapan, pasif bir ağ ölçüm metodudur. *PacketIn* ve *FlowRemoved* paketlerini, bunlar arasındaki zamanı ve bu paketlerin içerdiği akış-başlangıç, akış-bitiş, akış-sayaç verilerini kullanır. Hesaplamaları *FlowRemoved* paketi geldikten sonra yapabildiğinden bir akış tamamlanmadan hesaplama yapamamaktadır.

PayLess [15], ölçüm verilerinin toplanmasında akış büyüklüğüne göre dinamik olarak değişen toplama sıklığı yaklaşımı kullanır. Sürekli olarak aynı zaman aralıklarında veri çekmediğinden ölçüm maliyetleri düşmektedir. Yöntem, düşük maliyetli, gerçek zamanlı ölçüm ile düşük zaman sapmalı ve kesinliği yüksek veriler sağlamaktadır.

OpenNetMon [16], gecikme, paket kaybı ve bant genişliği gibi parametrelerin akış-tabanlı olarak ölçülebilmesine odaklanmış olup bunu yaparken MİB gibi kaynakların kullanımını en aza indirmeyi amaçlamaktadır. Bu ölçümlerin yapılabilmesi için OpenNetMon aktif ölçüm yaklaşımına göre ağa izleme paketleri basar.

FlowTrApp [17], trafik matriks ölçümünü hedef alan, tüm akışların büyüklüğünü, yönlerini ve kaynak tüketimlerini hesaplamaya çalışan bir yaklaşımdır. Akış tabanlı bu bilgiler ağın geneline ilişkin değerli bilgiler sağlamaktadır. Çoğu diğer yazılım tanımlı yaklaşım gibi ölçüm verilerinin ağdan toplanmasında OpenFlow protokolünü kullanır.

FleXam [18], güvenlik mühendislerine ağlarıyla ilgili paket seviyesinde bilgi sağlayabilecek bir sistem oluşturma dinamiğiyle geliştirilmiş, OpenFlow tabanlı bir ağ ölçüm yaklaşımıdır. Yazarlara göre FleXam port-tarama atakları ve büyük akış tespiti, atak tespit ve atak önleme gibi güvenlik fonksiyonlarını bünyesinde barındırmaktadır.

2.2. Anormallik Tespit Yöntemleri

Bu bölümde, bu alanda yapılmış olan anormallik tespit yöntemleri ve bu yöntemlerle birlikte kullanılan izleme/ölçüm yaklaşımları incelenmiştir. İncelemeler, saldırı ve anormallik

tespitinin gerçek zamanlı veya gerçek zamanlıya yakın şekilde yapılmasının kullanılacak ölçüm tekniği yetenekleriyle bağıntılı ve mümkün olduğunu göstermektedir. Bant genişliği, MİB, akış tabloları gibi ağ kaynaklarının ve paket kaybı, gecikme ve dalgalanma gibi başarım ölçütlerinin gerçek zamanlı durum tespiti sayesinde, kaynak tüketme veya ağın olağan işleyişine müdahale etme saldırılarında önemli birer uyarıcı olduğu, bu kapsamda YTA yaklaşımının ağdaki olaylara ilişkin gerçek zamanlı, doğru, düşük maliyetli çözümler sunduğu görülmektedir.

Anormalliğin tanımına ilişkin birçok kavram yer almakla birlikte tüm ifadeler "beklenen normal dışındaki davranış/oluş örüntüleri" ortak tanımında kesişmektedir [19-24]. Teknolojiden sağlığa, psikolojiden bilime her alanda anormallik durumları söz konusu olmakta, her boyutta anormallik tespiti önem arz etmektedir. Benzer şekilde, ağ davranışındaki normal dışı durumların oluş zamanına en yakın sürede tespiti anormalliğin önüne geçmede büyük rol oynar. Anormallik tespit sistemlerinin en temel amacı doğru tespitin mümkün olan en fazla oranda, yanlış alarm yoğunluğu ile saldırı tespit süresinin en düşük oranda olmasıdır [25].

Anormallik tespitinde, kaynakların normal durum davranışı istatistiksel, kural tabanlı ve yapay sinir ağları gibi doğrusal olmayan yöntemler kullanılarak tespit edilir ve profil olarak adlandırılan özet bilgiler oluşturulur. Kaynakların davranışındaki tüm değişimler düzenli olarak değerlendirilerek ilgili politikanın da güncellenmesi sağlanır. Kaynağın da davranışında ani bir değişim görüldüğünde bir "anormallik" olduğu öne sürülerek uyarı üretilir. Entropi, makine öğrenmesi, ilişki analizi, veri madenciliği yöntemleri bu amaçla kullanılan yaklaşımlardan bazıları olup bu yöntemler kullanılarak yapılan bazı çalışmalar aşağıda yer almaktadır.

Giotis ve arkadaşları [26], OpenFlow ve sFlow'u birleştirerek geliştirdikleri anormallik tespit ve önleme sistemlerini gerçek üniversite ağında test etmek için bu alanda yaygın kullanılan, entropi tabanlı bir algoritma [27] tercih etmişlerdir. Algoritma, özel bir veri yığını içindeki rastgeleliği ve düzensizliği ölçerek 0 ile 1 arasında ağırlıklandırarak veri kümesini herhangi bir andaki durumunu sayısallaştırır. Sistemin düzensizliği arttıkça entropi değeri de doğru orantılı olarak artar. Belli bir değerin üzerinde çıkan entropi oranı, sistemde anormalliğin göstergesi olarak kabul edilir. Çalışmada sFlow örneklem tabanlı metrik

toplama yöntemi ile OpenFlow temelli anormallik tespit yöntemlerinin başarımı karşılaştırılmış, her ikisinin birleştirilip kullanıldığı hibrit çözüm başarılı bulunmuştur.

Zhang [28], yaptığı çalışmada ağ anormalliğinin tespiti için makine öğrenmesi yaklaşımı kullanmıştır. Çalışmada değişken veri akışına uyumu ve anormallik tespit gecikmelerini düşürmeye çalışmış, bu amaçlar merkezi ve dağıtık mimaride çalışabilecek, parametrik olmayan durumları analiz edebilecek, değişkenliğe adapte olabilen, farklı tipteki uygulamalarda çalışabilen bir algoritma geliştirmiştir. Bu alanda, normal ve anormal trafiği ayırt etmek için makine öğrenmesi algoritmaları kullanan birçok çalışma mevcuttur [29-32].

Maglaris ve arkadaşları [33] kenar yönlendiricilerden alınan istatistikleri incelemek ve anormallikleri tespit etmek amacıyla, çift yönlü sayan sketch algoritmasını [34] kullanmışlardır. Bu algoritma hafıza verimliliği için geniş veri yığınlarının bir özetini tutarak asimetrik veri haberleşmesinin oluştuğu trafik örüntülerini keşfetmektedir. Ölçüm verilerini almak için ise sFlow protokolü kullanılmaktadır.

Jankowski ve arkadaşları [35] OpenDaylight denetleyicisi üzerinde geliştirdikleri kendini organize eden harita (SOM) [36] yöntemiyle anormallik ve saldırı tespit etmeye çalışmışlardır. SOM, yüksek boyutlu verilerin analizini ve görsel ifade edilmesini sağlayan bir yapay sinir ağı modeli olup öğreticisiz öğrenme yapısı üzerine kurulmuştur. Yüksek boyutlu verilerin iki boyutlu örgü üzerine doğrusal olmayan istatistiksel yöntemlerle haritalanmasını gerçekleştirir. Belli bir metrik göz önüne alınarak birbirine yakın olarak ifade edilen vektörler SOM haritası üzerinde birbirine yakın harita birimleriyle ifade edilir. Bu haritanın uzağında oluşan davranışlar anormallik olarak tariflenir.

Mehdi ve arkadaşları [37] saldırı tespit ve anormallik yöntemlerinde kullanılan entropi, rastgele ilerlemeli eşik değeri algoritması [38], trafik sınırlama ve ağ trafiği anırmallik tespiti [39] yaklaşımlarının YTA'larda kullanılabilirliğini doğruluk, atak saptayabilme, farklı ağ tiplerindeki başarı, gerçek zamanlılık yönlerinden incelemiştir.

Braga ve arkadaşları [40] akış metriklerini OpenFlow yardımıyla aldıktan sonra, akış başına ortalama paket sayısı, akış başına ortalama boyut, ortalama akış süresi, birim zamandaki akış sayısı, farklı port kullanım oranı ve birim zamandaki tekil akış sayısı vektörlerinden oluşan *6-değişkenli* SOM yaklaşımı ile trafiğin anormal olup olmadığını analiz eder. Çalışmaları

aynı veri setini kullanan diğer çalışmalara göre daha düşük maliyetli olduklarını ifade etmektedir.

Sahri ve arkadaşları [41] akış büyüklüğü, akış sayısı, aynı hedef IP'ye gelen farklı akış sayısı, farklı hedef porta gelen akış sayısı ve farklı kaynak-hedef çiftlerini içeren, *5-değişkenli* öznitelik analizi ile YTA ağlarda anormallik tespiti ve sınıflandırması yapmayı amaçlamışlardır. Aktif akışlar benzer birçok yaklaşımda olduğu gibi OpenFlow protokolündeki *Packet_In* mesajlarından elde edilmektedir. Bu paketler adaptif yöntemle, akış yoğunluğuna bağlı olarak değişen oranlarda alınmaktadır.

Özçelik ve arkadaşları [42] GENI test ortamını ve OpenFlow protokolünü kullanarak artımsal toplam (Cusum) algoritmasının [43] başarımını test etmişlerdir. Algoritma, şimdiki zaman ve uzun dönem geçmiş arasındaki farkı hesaplayarak anormallik saptamaya çalışmaktadır. Eğer şimdiki zaman ortalaması geçmiş zaman ortalamasından çok hızlı değişiyorsa kullanılan algoritmasının katsayı değeri yükselir. Katsayının belirli bir eşik değerinin üzerine çıktığı zamanlarda ağda olağan dışı durumların oluştuğu kabul edilir. Çalışmada kullanılan GENİ test ortamı, gerçek üniversite ağında normal ağ trafiğine zarar vermeden test yapabilme imkânı vermiştir. Çalışmaları ayrıca, YTA yaklaşımının ağ kullanımını düşürdüğünü, bununla birlikte atak saptama yeteneğini arttırdığını göstermiştir.

Anormallik tespit algoritmalarının en çok kullanıldığı saldırı türleri arasında D/DoS atakları gelmektedir. D/DoS tespiti imza tabanlı veya anormallik tespit algoritmalarıyla yapılabilmektedir. İmza tabanlı yaklaşım, sürekli güncellenen bir imza veri tabanı ile ilgili trafiği karşılaştırarak benzerlik bulmaya çalışır. Belirli bir orandaki benzerlik aynı atakla karşılaşıldığı anlamına gelir. Öte yandan anormallik tespit algoritmalarında, ağdan trafik istatistikleri ve ölçüm verileri (gelen paket oranı, akış istatistikleri, trafik yoğunluğu, gelen ip paketlerinin entropisi vs…) incelenerek ani değişimler için alarm üretilir. Anormallik tespiti yaklaşımları imza tabanlı yaklaşımlara nazaran daha çok yanlış alarm üretebilmekle birlikte, daha önceden görülmemiş sıfırıncı gün atakların yakalanmasında da imza tabanlı yaklaşıma göre daha yüksek başarıya sahiptir. Bu alanda anormallik tespiti yoluyla D/DoS saptama ve önlemeye yönelik birçok çalışma yapılmıştır [44-52].

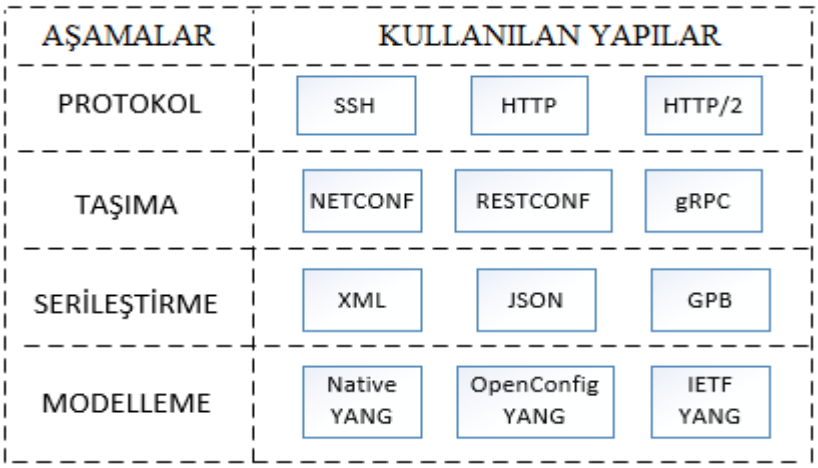
3. YÖNTEM VE ARAÇLAR

Bu bölümde tezin kuramsal altyapısını oluşturan teknolojiler, protokol ve metotlar detaylı olarak anlatılmıştır.

3.1. Model Tabanlı Mimari

Cihaz ile etkileşimde günümüze kadar kullanılan geleneksel yaklaşımların (SNMP, SSH, Telnet vs.) düzensiz veri yapısı programlanabilirlik, ölçeklenebillirlik, başarım ve okunabilirliği yapılan bütün geliştirme ve düzenlemelere rağmen aşılamaz bir sorun haline getirmiştir.

Ağ ölçüm ve izlemesine sürekli ölçüm özelliği katan temel fonksiyonlardan ilki çekme yaklaşımından itme yaklaşımına geçmek iken bir diğer kavram veri akışına veri yapıları ile getirilen serileştirme ve bu verilerin toplayıcı tarafında istenen biçime dönüştürülme özellikleridir. Ağ cihazlarının çok yüksek boyut ve miktarlardaki ölçüm ve trafik bilgilerinin cihaz dışına formatlı şekilde serileştirilerek aktarılması ve standart protokollerle taşınması yaklaşımına model tabanlı telemetri (MTT) denir. Yeni nesil ölçüm ve otomasyonla birlikte anılan bu yaklaşım için Yet Another Next Generation (YANG), Google Protocol Buffers (GPB), Google uzaktan ordam çağrısı (gRPC) gibi veri yapıları kullanılmakta olup model tabanlı ölçüm protokol yığını Şekil 3.1'deki gibidir.

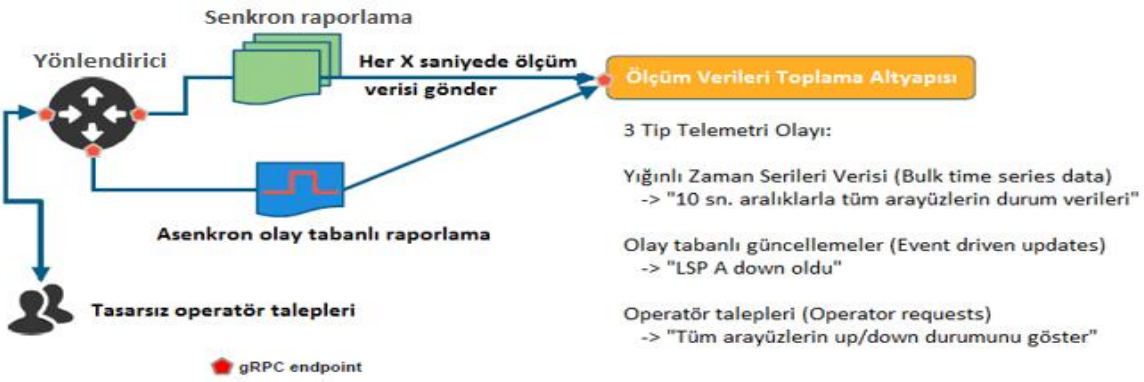


Şekil 3.1. Model tabanlı ölçüm protokol yığını [4]

3.1.1. Akış ölçümü

NASA raporlarında telemetri, "ölçüm verisi üreten ağ aygıtlarıyla bu verilerin toplandığı ortamlar arasındaki otomatize olmuş iletişim" olarak tanımlanır [53-54]. Sürekli ölçüm'de de aynı iletişimin sürekli, kesintisiz, ölçeklenebilir ve verimli olarak yürütülmesi, ölçüm verisi almak isteyen operatörün verileri farklı üreticilerin farklı kodlama ortamlarından bağımsız olarak tek tipte talep edebileceği, modellenmiş veri yapısı ve ortak tanımlanmış arayüzler sunan bir yaklaşım modellenmektedir.

Geçtiğimiz 30 yıl boyunca ağ izleme operasyonu kısıtlı bant genişliğine sahip altyapılarda talep tabanlı olarak, en iyi düzenlemeler yapıldığı takdirde bile en fazla 5 dakikalık aralıklarla veri çekerek yürütülmüştür. Ağda olup biten olayları anlamak ve düzenli aralıklarla güncellemelerden bağımsız olarak uçtan uca gerçek zamanlı görünürlük yeteneği kazanmak için geleneksel, talep tabanlı ağ izleme metotların yerine geniş ölçekli altyapılar için yüksek kullanılabilirlik sunan, programlanabilirlik ve otomasyon özelliklerinin kullanılabildiği akış ölçüm yaklaşımının kullanımı yaygınlaşmaktadır. Akış ölçümü, binlerce kaynaktan belirli bir biçimde sürekli olarak gelen ölçüm verilerinin parça parça ve yığınlı işlemeden ziyade gerçek zamanlı işleme yaklaşımıdır. Ölçüm verilerinin gönderimi için geleneksel yaklaşımlarda olduğu gibi herhangi bir talep beklenmez. Ağ olay ve durum verileri sürekli ve gerçek zamanlı olarak veri işleme sunucularına gönderilir. Ağ hakkında ince ayrıntılı ve yüksek veri sıklığı sağlayan bu yaklaşım ağ olaylarını, başarımı ve güvenlik özelliği olarak anormallikleri izlemede önemli rol oynar. İşlenmek üzere gelen yoğun ölçüm verileri, ağ izleme sürecini, karar vermeyi iyileştirmek için büyük veri kümelerinin hızlı bir şekilde çıkarılmasını ve analiz edilmesini sağlayan büyük veri düzlemine taşır [55]. Akış ölçümü olayları Şekil 3.2'de verildiği gibidir.

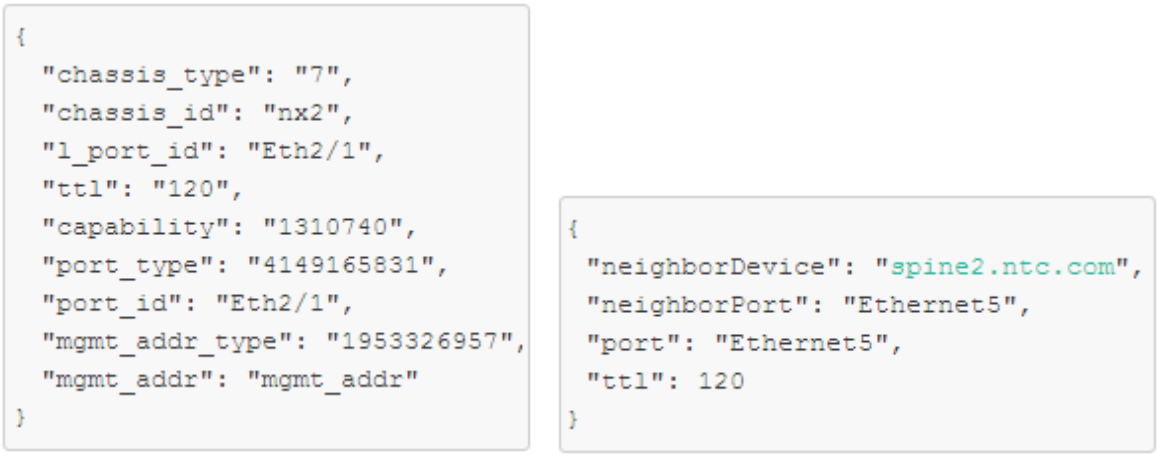


Şekil 3.2. Akış ölçümü olayları

3.1.2. Veri modelleme

YTA'nın temel amaçları ağ operasyonlarına otomasyon ve programlama yetenekleri kazandırmak, bunu yaparken üretici bağımlılığını azaltmak/ortadan kaldırmak, açık kaynak kodlu bir ortam kurmaktır. Model tabanlı yaklaşımı oluşturan NETCONF, YANG, GPB gibi yeni nesil yaklaşımlar bu kuramın uygulamaya geçmiş yöntemleridir. Mevcutta komut satırı arayüzü, SNMP gibi geleneksel yöntemler yapısal olmayan, her üreticinin özelindeki ayarlamaların cihazlara ulaştırılmasına hizmet eder. Bu da ağ yapılarında otomasyonun gelişmesi önündeki en zorlu engellerdendir. YANG veri modelleme dili tam bu noktada, cihazlardan ölçüm verilerinin çıkarılmasında esnek, ölçeklenebilir, yapısal bir çatı oluşturarak üreticiden bağımsız bir ortam sağlar.

Aynı taşıma ve serileştirme yöntemleri kullanılsa bile içerisinde taşınan verinin biçiminin farklı olması sebebiyle istenen başarım ve otomasyon sağlamak mümkün olmayacaktır. Örneğin geleneksel SSH/komut satırı arayüzü komutları yapısal olmayan ve yığınsal veri şeklinde veri taşırken REST API'lerde veri JSON şeklinde taşınır. Her üreticinin JSON biçimi bile birbirinden farklı olabilmektedir. Şekil 3.3, Cisco ve Arista üreticilerinin komşuluk komutlarındaki farklılığı göstermektedir.



a. NX-API ile Cisco JSON modellemesi b. eAPI ile Arista JSON modellemesi

Şekil 3.3. Farklı üreticilerin JSON modellemesi örneği

Verinin nasıl taşındığından daha önemlisi verinin nasıl modellendiği, iletildiği üst katmanlarda nasıl hızlı çözümleneceği olmaktadır. Veri modelleri tam bu noktada tüm üreticilerin ihtiyaç duyduğu ortak yapıyı oluşturmada çözüm sağlamaktadır. Ağ endüstrisi

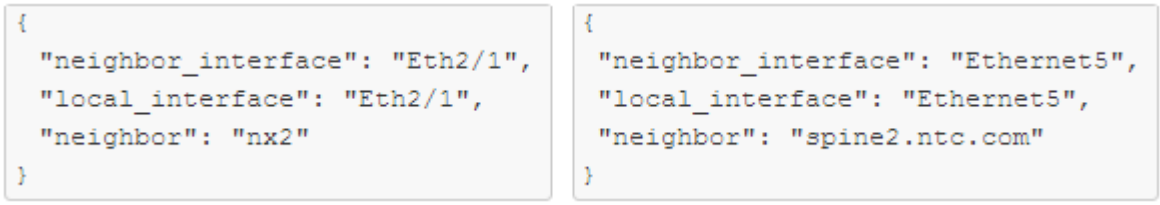
son zamalarda ağın telemetri, yönetim ve otomasyon yetenekleri için YANG veri modelleme dilinde [55-58] mutabık kalmıştır. YANG modelinin üreticiye özel, IETF ve OpenConfig olmak üzere 3 tipi geliştirilmiştir.

Üreticiye özel YANG modelde, her üretici kendi cihazlarını konfigüre etmek ve cihazdan veri çekmek için kendi özel YANG modülünü yayınlar. Temelde aynı dil yapısı ve semantik kullanılsa da her üretici kendi özel düğümlerini oluşturur. Komut sastırı arayüzü yerine üretici bağımlı bu model kullanılabilir.

IETF YANG modeli, IETF tarafından, ağ cihazlarının ayarlanması için ortak bir veri modeli oluşturmak amacıyla geliştirilmiştir. Modelin kullanılabilmesi için üreticilerin bu modeli desteklemesi gerekmektedir.

OpenConfig YANG modeli, ağ operatörleri için aşılmaz bir bariyer olan standart arayüz ve veri modeline sahip olma sorununun üstesinden gelmek ve esnek programlama yeteneklerini ağ ortamlarına aktarmak için Google, AT&T, British Telekom, Microsoft, Facebook, Comcast, Verizon, Yahoo, Apple, Deutsche Telekom, TeraStream, Bell Canada gibi firmaların ortak katılımıyla geliştirilmiş veri yapısı sunar.

MTT yaklaşımı ile birlikte üst katmanda geliştirilen uygulamalar alt katmanda hangi üreticinin donanımının çalıştığından bağımsız olarak Şekil 3.4'te örneklendiği şekilde yönetim yapabilecektir.



a. MTT yaklaşımlı Cisco modellemesi b. MTT yaklaşımlı Arista modellemesi

Şekil 3.4. MTT yaklaşımı ile üretici normalizasyonu sonrası durum

3.1.3. Veri serileştirme

Bu alanda yaygın kullanılan 3 temel veri serileştirme yöntemi aşağıda verilmiştir.

Genişleyebilir İşaretleme Dili (XML)

İnterneti kullanarak veri alışverişi yapan sistemler arasındaki veri iletişimini standart hale getirmek için Geniş Dünya Ağı Konsoriyumu (World Wide Web Consortium, W3C) tarafından tasarlanan genişletilebilir işaretleme dilidir. Bir yazılımda veya veri tabanında kullanılan verilerin başka bir sisteme taşınması sırasında karşılaşılan, iletilen bilgi kümesindeki veri yapısının, diğer sistemdeki yapıya uygun hale getirilmesi işleminin uzun zaman alması, karmaşık süreçler içermesi ve tasarlanan sistem ve fonksiyonların işlem sonucunda başka alanlarda kullanılması mümkün olmaması gibi problemleri aşmak amacıyla standart bir veri iletim teknolojisinin geliştirilmesi gerekliliği üzerine XML dili ortaya çıkmıştır. HTML gibi katmanlı bir işaretleme dili olan XML'de kullanılan etiketler, HTML'den farklı olarak operatör tarafından belirlenir. Veriler belli bir yapıda derlenip, başkalarının bu verileri kullanmasına imkan tanıyacak hale getirilir [57].

XML dili günümüzde veri iletimiyle alakalı her alanda sıklıkla kullanılmaktadır. Web siteleri için site haritası oluşturma, veri tabanlarının aktarılması, yazılım paketleri içerisindeki bağımlılıkların tanımlanması, finansal verilerin iletimi, dosya sistemlerinin oluşturulması, bilimsel içeriklerin depolanması gibi birçok alanda XML teknolojisi kullanılmaktadır.

Javascript Nesne Gösterimi (JSON)

Başlangıçta Javascript uygulamaları için, XML dilinin veri alışverişi sırasında büyük ve yavaş kalması üzerine geliştirilmiştir. Daha sonra birçok sistem tarafından desteklenmiş, okunması ve kullanımı kolay, XML'e alternatif olmuş, veri değişim ve içerik kategorileme dilidir. JSON'un temel amacı veri alış verişi yaparken daha küçük boyutlarda veri alıp göndermektir. XML'deki gibi uzun kodlar yerini anlaşılır ve sade kodlara bırakmıştır. Bu yaklaşım dosya boyutlarının düşmesine, başarımın artmasına yardımcı olmuştur. Programlama dilinden bağımsız bir veri değişim arayüzü sağlar [58].

Google Protocol Buffers (Protobufs) (GPB)

Verinin ağ üzerinde hareket edebilir olma özelliği sebebiyle bu dolaşımı sırasında ortam bağımsızlık, okunabilirlik, genişletilebilirlik ve başarım gibi ölçütler eşliğinde

serileştirilmesi gerekir. XML ve JSON'dan sonra karşımıza çıkan diğer serileştirme yöntemi Google tarafından geliştirilen Google Protocol Buffers veya protobufs'tır [59-60]. Bu model, Google'nin kendi uygulamalarındaki sistemler arasında veri değiş tokuş işlemleri için ele aldığı, programlama dillerinden bağımsız bir serileştirme standardıdır.

Bir programlama dilinde oluşturulan nesneler daha sonra kullanabilmek için dosyalara serileştirerek yazılır ama bu serileştirme sistemi muhtemelen programlama diline özgü olur. Örneğin Java'da serileştirilen bir nesne Python kullanarak okunamaz. Python'un serileştirme yapısı ile Java'nınki farklıdır. Bu durum karşısında Google'da çalışan mühendisler farklı programlama dillerinde evrensel olarak kullanılabilecek bu serileştirme yöntemini geliştirmişlerdir.

JSON ve XML'e göre en büyük avantajlarından biri veriye metin yerine ikili serileştirme yapılıyor olması sebebiyle daha hızlı olması ve daha az yer tutmasıdır. Aynı zamanda, metin çözümleme olmadığından dolayı hafıza kullanımı açısından daha etkili olduğu ifade edilmektedir. İkili serileştirmeyi kullanan protokol açık kaynak olarak sunulmakta olup C++, C#, GO, Java, Python, Javascript gibi diller tarafından da desteklenmektedir. JSON veya XML kadar okunabilir olmasa da MİB, hafıza ve bant genişliği kullanımının önemli olduğu uygulamalarda tercih edilebilir olmaktadır.

Sistemin çalışma mantığı şöyledir; öncelikle serileştirme yapılacak nesnelerin ait olacağı sınıf yapısını protobuf dilinde tanımlanır. Ardından protobuf derleyicisi yüklenir ve kullanılacak programlama dili belirtilir; derleyici de buna göre seçilen dil için bir sınıf tanımı oluşturur. Bu sınıf yapısı kullanılarak bir nesne serileştirip, başka bir dilde tekrardan kullanılabilir. Bunun için de tekrardan nesnenin açılacağı dilde protobuf sınıf tanımını derlenir ve o dosya çalışmaya ekleyerek nesne okunur. Serileştirme işlemi esasında o nesneyi evrensel bir karakter katarına çevirdiğinden, bu karakter katarı kullanılarak gerçekleştirilebilecek herhangi bir iletişim işleminde protobuf kullanılabilir. Örneğin, Java diliyle yazılmış bir socket sunucusuna Python istemci üzerinden bir nesneyi serileştirip gönderilebilir.

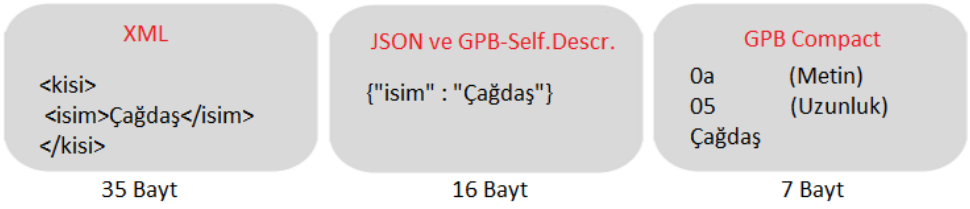
Compact GPB ve self-describing GPB olmak üzere 2 tipi vardır. Compact olan tipinde değişkenler birer sayısal değer olarak tutulur. .proto dosyaları sayesinde bu sayısal değerlerin karşılığı olan anlamlı ifadelere ulaşılır. Self-describing GPB ise daha çok JSON'a

benzer yapıda olup değişkenler anlaşılır metin halindedir. Şekil 3.5'te *protobuf* tipleri gösterilmiştir.

GPB - Compact Tipi	GBP - self_describing Tipi
1: GigabitEthernet0/0/0/0 50: 449825 51: 41624083 52: 360333 53: 29699362 54: 91299 …	(interfaceName : GigabitEthernet0/0/0/0 GenericCounters (PacketsSent: 449825 BytesSent: 41624083 PacketsReceived: 360333 BytesReceived: 29699362 MulticastPacketsReceived: 91299 …

Şekil 3.5. Protobuf tipleri

Veri serileştirme yöntemlerinin gösterim ve kapasite karşılaştırmaları Şekil 3.6'daki gibidir.



a. XML b. JSON/GBP_self-desc. c. GPB_compact

Şekil 3.6. Veri serileştirme yöntemlerinin gösterim ve kapasite karşılaştırması

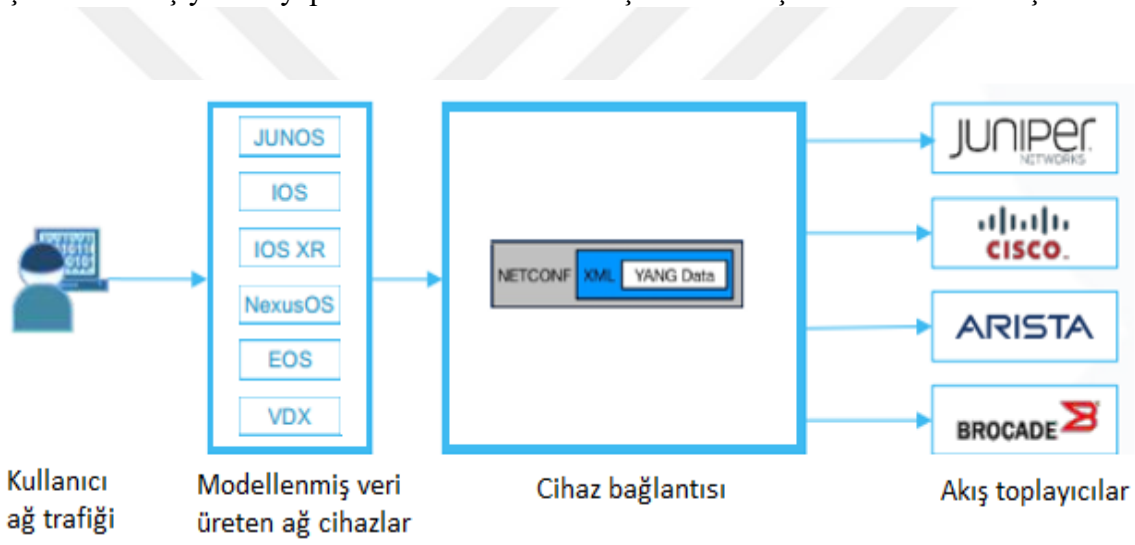
3.1.4. Veri taşıma protokolleri

Günümüz ağ operasyonlarının otomatize edilmesi ve bu sayede yapılandırma, yönetim, test, kurulum işlemlerinin otomatik olarak yapılması için çözümler aranmaktadır. Taşıma protokolleri bu amaca yönelik çözüm üretmektedir. API'ler, iki uygulamanın birbiriyle konuşmasına izin veren aracı yazılımlardır. Diğer bir deyişle, programcıya ait isteği, servis sağlayıcısından talep eden ve ardından programcıya yanıt gönderen yazılım parçacıklarıdır. API, uygulamaların ve tanımların birbirlerinden ödün vermeden değişmelerine izin veren uygulamalardan bağımsız işlevleri tanımlar. Bu nedenle bir API, program geliştirmeyi kolaylaştırır. Çok sayıda API türü vardır. Örneğin, Java API'leri veya sınıflar içindeki nesnelerin Java programlama dilinde birbirleriyle konuşmasına izin veren arayüzler mevcuttur. Program odaklı API'lerin yanı sıra, Basit Nesne Erişim Protokolü (Simple Object Access Protocol, SOAP), Uzaktan Yordam Çağrısı (Remote Procedure Call, RPC) ve

Temsili Durum Transferi (representational state transfer, REST) gibi web API'leri vardır. Halka açık ve programlanabilir 15.000 API ve şirketlerin dahili ve harici yeteneklerini genişletmek için kullandığı binlerce özel API'ler var. Ağ teknolojilerinde kullanılan belli başlı API'ler aşağıda verilmiştir.

NETCONF

NETCONF [61], IETF tarafından geliştirilen YANG veri modeliyle yapısal hale getirilen verinin taşınmasından ve ağ ayarları yönetiminden sorumlu olan protokoldür. NETCONF, ayar değişikliklerini doğrulayabilir, saptayabilir, cihazlardan verileri ve olay tetiklemeleri işlemlerini taşıyabilir yapıdadır. NETCONF iletişim modeli Şekil 3.7'de verilmiştir.



Şekil 3.7. NETCONF iletişim modeli

RESTCONF

RESTCONF [62], http üzerinde çalışan, XML veya JSON biçiminde yapısal verileri taşıyabilen, YANG veri modelleme dilini destekleyen, üretici bağımsız programlanabilir ağ yönetim arayüzü sağlayan bir protokoldür. GET, POST, PUT gibi HTML kodlarını ve NETCONF'un veri depolarını kullanır. NETCONF gibi RESTCONF da YTA mimarisinde uygulamaların birbirleriyle bağlantısında kullanılır.

TCP/UDP

Taşıma katmanı, uygulama ve ağ katmanları arasında mantıksal bir bağlantı kurularak verilerin taşınmasından sorumlu katmandır. TCP/IP'de tanımlanan, en yaygın kullanılan taşıma katman protokollerinden ikisi TCP ve UDP'dir. Paket kayıplarının yaşanmasının engellenmesi, paketlerin sıralı bir şekilde gitmesinin önemli olduğu, güvenirlilik ve kararlılık isteyen altyapılarda TCP, uygulamaların gerçek zamanlılık ve yüksek hız gereksinimlerinin olduğu, tekrar gönderimlerin istenmediği durumlarda UDP yaygın kullanılmaktadır. Birbirlerine göre avantaj ve dezavantajları olmakla birlikte her iki protokol de uygulama gereksinimlerine göre konumlandırılırlar.

gRPC

Google'nin 2015 Şubat'ında açık kaynak haline getirdiği gRPC [63], uzaktan yordam çağrısı altyapısının Google tarafından gerçeklenmiş halidir. RPC, genelde ağ paylaşımlı bilgisayarlar üzerinde programcının kodlama olmadan çalıştırmak için bir alt yordam veya metot sağlayan bilgisayar programına izin veren mikro servisler ve süreçler arası iletişim teknolojisidir. Düşük gecikmeli, yüksek ölçeklenebilirlikli, dağıtık haldeki sunucu ve istemcilerin iletişimi için tasarlanmış, kimlik doğrulama, yük dengeleme, loglama izleme özellikleri barındıran bir servistir.

Herhangi bir uygulama sunucu istemci iletişimine ihtiyaç duyuyor ise bu RPC servisi kullanılabilir. Google tarafından özelleştirilmiş olan gRPC de, yüzlerce farklı programlama dili ve sunucu mimarisi arasındaki iletişimi sağlayan, açık kaynak kodlu bir API'dir. Açık kaynak kodlu hale getirildikten hemen sonra Cisco, Juniper ve Arista gibi büyük telekom üreticileri tarafından ağ cihazlarının yapılandırılmalarında ve ölçüm verilerinin cihazlardan çekilmesinde gRPC'yi kullanmaya başlamıştır. Python, Node.js, Go, Ruby, C++, Java, Objective-C ve daha birçok programlama dillerini sorunsuz destekleyen gRPC, taşıma katmanında HTTP/2 teknolojisini kullanır. Sunucu-istemci arasındaki iletişimin gecikme süresini azaltmak üzere geliştirilen bu taşıma protokolünde, tek bir TCP bağlantısı üzerinden birden çok oturum taşınabilir, veriler ikili serileştirilerek iletilir, paket başlıkları da sıkıştırılarak iletilen veri boyutları küçültülür, sunucu itme tekniği sayesinde akış ölçümümimarisinde olduğu gibi gecikmenin en az seviyelere inmesi sağlanır. Sunucu-istemci mimarisinin bu alandaki en güncel çözümü olan HTTP/2, yayınlama/abone olma

modeli, tüketicilerin talep ettikleri veriye üye olarak o veriyi sürekli olarak edinebilmeleri temeline dayanan telemetri mimarisi için başarım, hız, düşük maliyet, ölçeklenebilirlik faydaları sağlamaktadır.

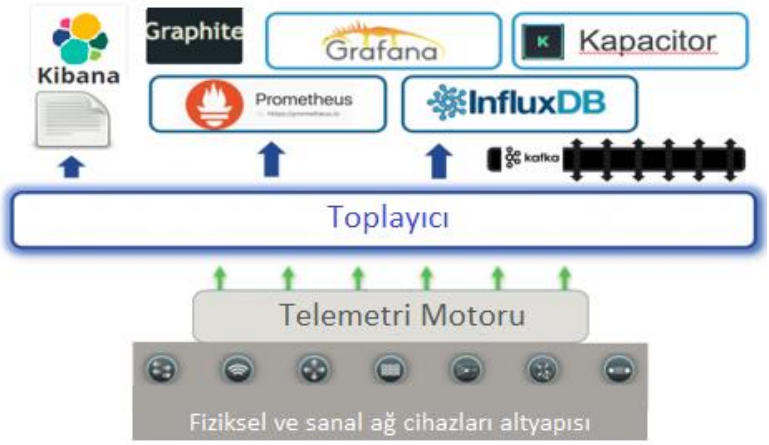
3.2. Model Tabanlı Telemetri Uygulama Mimarisi

Tüm işlem ve süreçlerin dijitalleştiği günümüzde sistemlerin ürettiği veri ve log miktarı çok ciddi büyüklüklere ulaşmaktadır. Bu büyük verinin ele alınması için birçok yaklaşım olmakla birlikte en yaygın kullanılanlardan birisi Topla-Depola-İşle modelidir. Bu modelde yüzlerce farklı kaynaktan, onlarca farklı tür ve tipteki veri toplayıcılar vasıtasıyla bir işlem hattında toparlanarak ihtiyaç duyan sistemlere sunulur. Verinin sunulduğu kaynaklardan en temeli depolanmak üzere iletildikleri veri tabanlarıdır. Çalışılan sistemin özelliklerine bağlı olmakla birlikte gerçek zamanlı büyük veri ihtiyacına yönelik Zaman Serisi Veri Tabanları (ZSVT) geliştirilmiştir. Veriler bu veri tabanlarında tutulur ve ağ olaylarının görselleştirilmesi, saldırı tespiti ve engellenmesi, ağ trafiğinin yönetilmesi ve gerekli reaktif/proaktif aksiyonların alınması, veri madenciliği yöntemleriyle ağ davranışına ilişkin değerli örüntülere ulaşılması, alarmların üretilmesi gibi amaçlarla işlenmek üzere diğer uygulamalarla API'ler aracılığıyla paylaşılmaktadır.

Bu alanda yaygın kullanılan toplayıcı, veri tabanı ve işleme çözümleri ve bu çözümlerin detayları aşağıda incelenmiştir.

3.2.1. Toplayıcı

Toplayıcı, ağda bulunan uygulamalar, servisler, cihazlar tarafından üretilen ölçüm, akış, trafik ve durum bilgilerinin üretildiği kaynaktan dış ortama aktarılması için kullanılan bileşen olup veri kaynakları ile diğer sistemler arasındaki mantıksal konumu Şekil 3.8'de verilmiştir. Bu alanda birçok toplayıcı bulunmakla birlikte her biri 2 temel mantıkta çalışmaktadır. Noktadan noktaya ismiyle geçen birinci yöntemde kaynaktan çekilmek istenen veriler sıralanarak bir kuyruk oluşturur, kuyruktaki mesajlar sırayla talep eden sistem veya sistemlerle paylaşılır. Ancak bir veri yalnızca bir sistem tarafından tüketilebilir ve tüketildikten sonra kuyruktan silinir. Yayınlama/abone olma modelinde ise tüketiciler talep ettikleri veriye üye olarak o veriyi sürekli olarak edinebilirler. Noktadan noktaya'nın aksine yayınlama/abone olma modelinde bir veriye birden çok tüketici üye olabilir.



Şekil 3.8. Toplayıcının mantıksal konumu

Toplayıcıya gelecek veriler cihazlardan direkt alınabileceği gibi mesaj kuyruklama sistemleri üzerinden de temin edilebilir. Bu amaçla kullanılan teknolojilerden birkaçı Apache Kafka [64], ActiveMQ [65], RabbitMQ [66]'dir. Apache Kafka, Linkedin tarafından başlangıçta kendi sistemleri için geliştirilen ve 2011 yılında halka sunulmuş, açık kaynak kodlu bir mesajlaşma sistemidir. Dağıtık mimarideki sistemlere uygun, sistemlerin ürettiği büyük ve gerçek zamanlı veriyi hızlı, sorunsuz ve ölçeklenebilir şekilde toplamaya ve bu veriye gerçek zamanlıya yakın bir gecikmeyle erişmeye yönelik geliştirilen Kafka, verileri temelde log kaydına benzer bir yapıda tutan ve bu kayıtları diğer sistemlere mesajlaşma kuyruğu şeklinde sunan bir yapıdır. Yüksek başarımı ve düşük gecikme zamanı ile gerçek zamanlı veri akışı sağlar.

Bu alandaki bazı toplayıcılar aşağıda verilmiştir.

Pipeline

Cisco'nun yeni nesil ağ işletim sistemi olan XR versiyonunun telemetri verilerini cihaz dışına çıkarmak için geliştirdiği, yayınla/abone ol mimarisinde çalışan model tabanlı verileri taşıyabilen, Go dilinde yazılmış açık kaynak kodlu veri yoludur. TCP, gRPC/HTTP2, UDP, Apache Kafka gibi farklı giriş veri yapılarını ve taşıma katmanlarını destekler. TLS şifreleme seçeneği ile cihazlardan alınacak veriler şifleri olarak taşınabilir. Compact GPB, GPB K/V ve JSON veri serileştirme yöntemlerini destekler. Zaman serisi veri tabanları ile uyumlu olan Pipeline [67], verileri JSON nesnesi gibi metin dosyasına yazabilir veya Kafka gibi veri yollarına aktarabilir.

Logstash

Logstash [68], farklı kanallardan veri toplayıp, yapılandırma seviyesinde filtrelerle belirli kurallara göre veriyi parçalamayı sağlayan ve farklı tipterde kanallara dağıtabilen gerçek zamanlı ve açık kaynaklı bir veri toplama motorudur. TCP/UDP soketleri ve log dosyaları gibi basit kaynaklar dışında çeşitli streaming protokolleri, Kafka, log4j, redis araçları, github, heroku, twitter ve irc servisleri gibi pek çok kaynaktan beslenebilir. Tüm bu kaynaklardan alınan verileri belirlenen kurallar çerçevesinde işledikten sonra eposta olarak gönderilebilir veya buradan çıkan verileri Elasticsearh, Kibana, Graphite, statsd gibi veri işleme araçlarına iletilebilir. Apache Kafka, RabbitMQ, Amazon SQS gibi mesaj sistemlerini destekler.

OpenNTI

Juniper tarafından gerçek zamanlı verileri toplamak için geliştirilen model tabanlı, açık kaynak kodlu bir toplayıcıdır. Docker konteynır mimarisiyle paketlenmiş olup pakette toplayıcı, InfluxDB zaman serisi veri tabanı ve Grafana grafik arayüzü araçlarıyla birlikte gelmektedir. OpenNTI [69], Juniper cihazlarından talep edilen verileri çekmek için, SNMP yerine geliştirilmiştir. Python betiklerine arayüzü PyEZ API'si ile sağlar ve XML serileştirme tipini destekler.

sFlow

sFlow [70], InMon firması tarafından, sürekli akan ölçüm verilerini toplamak için geliştirilmiştir. İlk 3 toplayıcıdan farklı olarak, YANG gibi model tabanlı veri yapısı yerine kendine özel geliştirilmiş bir veri yapısı kullanır. Yine kendine özel olan XDR encoding yapısını UDP üzerinden taşır. Talep edilen metrikleri istatistiksel örneklem yoluyla temin ederek veri akışını sürekli olarak sağlar. Farklı örneklem oranları ile trafik hassasiyeti ayarlanabilir.

Telegraf

InfluxData firması tarafından metrik toplama arayüzü olarak Go diliyle geliştirilen Telegraf [71], StatsD, collectd, Kafka gibi mesaj kuyruklama servislerinden veri alabilir, OpenTSDB,

Datadog, InfluxDB, Graphite gibi veri tabanı, servis ve mesaj yollarını destekler. Stream edilecek ölçüm verileri için itme ve çekme mimarisinde çalışabilmektedir. GPB serileştirme yöntemini ve gRPC taşıma protokolünü destekleyen Telegraf Juniper firmasının OpenNTI projesinde Kafka ile birlikte kullanılmaktadır.

3.2.2. Zaman serisi veri tabanı

Bir zaman diliminde bir ya da daha fazla değişkenin değişimlerinin gruplanması, analize hazır halde sunulması ihtiyacı, çekme mantığıyla düzenli aralıklarla verinin çekilmeyip itme modelinde abone olunan değerlerin düzenli zaman aralıklarında ardışık ve sürekli aktığı sürekli ölçüm ölçüm yöntemi için en önemli gerekliliklerdendir. ZSVT, bu ihtiyaç için geliştirilmiş olup, geleneksel veri tabanlarının aksine veriyi zamanla dizinleyerek karmaşık örüntüsü olan gerçek zamanlı veri için yüksek başarımlı dizinleme, depolama ortamı ve bunların işleneceği birimlere API'lerle arayüz sağlamaya çalışır. ZSVT yeni olmamakla birlikte haberleşme teknolojilerinde patlayan büyük veri kavramıyla birlikte yüzlerce sistemden gelen zamana dayalı ölçüm, durum, istatistik verilerinin gerçek zamanlı olarak işlenmesi için kullanılmaya başlanmıştır.

Bu alanda geliştirilen bazı zaman serisi veri tabanları aşağıda verilmiştir.

Prometheus

Açık kaynak kodlu izleme ve alarm sistemi olarak geliştirilen Prometheus [72] çok boyutlu veri modellerinin dizinlemesi için kullanılmaktadır. Verileri zaman serisi olarak depolayan Prometheus Sunucusu, uygulama kodu oluşturmak için istemci kütüphanesi, kısa süreli işlemek ve gerçek zamanlılığı desteklemek için Push GW yapısı ve önceden ayarlanmış eşik değerlerini geçen trafikler için alarm üretmek üzere Alarm Yöneticisi gibi temel bileşenler ile bu bileşenlerle birlikte çalışabilecek arayüzlerden oluşmaktadır. Akış verilerini zaman damgalı değer olarak tutan Prometheus, sorgu sonuçlarını da geçici zaman serileri şeklinde oluşturabilir. Go, Java, Python, Ruby gibi istemci kütüphanelerini destekler [72].

Elasticsearch

Açık kaynak kodlu Elastik projesinin tüm-metin arama ve analitik motoru olan Elasticseach [73], yüksek yoğunluklu verilerin gerçek zamanlıya yakın halde depolanlasında, aranması ve analiz edilmesinde kullanılır. Verileri zaman serisi şeklinde değil döküman olarak tuttuğu için verinin dökümandan çıkarılıp aranabilir hale getirilmesine kadar 1 saniyeye varan gecikmeler yaşanabilmektedir [74]. Bununla birlikte standart RESTful API'leri ve JSON'u kullanan Elasticsearch Curl, Java, C#, Python, Ruby gibi programlama/betik dillerini destekler.

InfluxDB

Go programlama diliyle yazılmış olan InfluxDB [75], Telegraf toplayıcısı, InfluxDB zaman serisi veri tabanı, Chronograf veri görselleştirme arayüzü ve Kapacitor motorundan oluşan TICK yığınının parçası olarak, açık kaynak kodlu şekilde servis edilmektedir. Sahip olduğu Python istemcisi ile kullanıcıların kendine özel geliştirdikleri kodları veri tabanı üzerinde hızlıca uygulayabilmelerine olanak verdiği gibi, gerçek zamanlı sürekli ölçüm işleme motoru olan Kapacitor [76] sayesinde veri toplayıcıdan ZSVT'na gelirken eş zamanlı olarak analitik veya anormallik tespiti gibi fonksiyonlara yönlendirilerek gerçek zamanlı işleme yapılabilir.

OpenTSDB

Ağ, işletim sistemi ve uygulamaların ürettiği metrikleri depolayarak zamanın bir fonksiyonu olarak dizinlemek ve diğer sistemlere servis etmek için Apache HBase altyapısıyla, Java ve Hadoop teknolojileri kullanılarak geliştirilmiş bir veri tabanı olan OpenTSDB [77], çok yüksek yoğunluktaki gerçek zamanlı veriyi, verideki küçük değişimleri kaybetmeyecek şekilde işlemeyi hedeflemektedir. Altyapısında kullanılan Hadoop mimarisi sebebiyle, verinin dağıtık ve ölçeklenebilir şekilde kaybolmadan saklanabilmesi üzerine yoğunlaşmaktadır.

Yaygın kullanılan açık kaynak kodlu birkaç zaman serisi veri tabanının karşılaştırmalı analizi Çizelge 3.1'de verilmiştir.

Çizelge 3.1. Zaman serisi veri tabanlarının karşılaştırma tablosu [78]

	DalmatinerDB	InfluxDB	Prometheus	OpenTSDB	Elasticsearch
Web Sitesi	www.dalmatiner.io	www.influxdata.com	www.prometheus.io	http://opentsdb.net	www.elastic.co/products/elasticsearch
Desteklenen Ölçüm	Metrik	Metrik, olay	Metrik	Metrik	Metrik, olay
Yedeklilik	Kümeleme	2 sunucuya çifte yazma	2 sunucuya çifte yazma	Kümeleme	Kümeleme
Operasyonel Karmaşıklık	Orta	Düşük	Düşük	Yüksek	Orta
Kayıt şekli	Sabit aralıklı	Olay tabanlı	Sabit aralıklı	Sabit aralıklı	Olay tabanlı
Teknoloji	Erlang, PostgreSQL	Golang	Golng	Java, Hadoop	Java
Desteknenen Veri Tipleri	Float62, int56	İnt64, float64, string	Float64	Int64, float32	String, Int32, Int64, Float32
Metrik Alma Süresi	Milisaniye	Nanosaniye	Milisaniye	Milisaniye	Milisaniye
Yazma Başarımı	2,5 milyon metrik/ saniye	470K metrik/ saniye	800K metrik/ saniye	32K metrik/ saniye	30K metrik/ saniye
Sorgu Başarımı	15 milyon metrik/ saniye	-	-	128K metrik/ saniye	120K metrik/ saniye
Sorgulama Dili	DQL	InfluxQL	PromQL	Arama	Query DSL
Giriş	Tcp, OpenTSDB(metin) Graphite(metin) Prometheus(metin) InfluxDB (http)	InfluxDB (http) InfluxDB (udp) OpenTSDB(metin) OpenTSDB (https) Graphite(metin)	Metin	http, tcp(metin)	http
Çıkış	htpp, tcp, metin	http	http	http	http
Sorgu Dili Fonksiyonelliği	3/5	4/5	5/5	3/5	3/5
Sorgu Dili Kullanılabilirliği	4/5	5/5	4/5	1/5	3/5
Kurumsal Destek	Var	Var	Var	Yok	Yok
Destek Ekibi Büyüklüğü	Dar	Geniş	Geniş	Orta	Geniş
Lisans	MIT	MIT	Apache 2	GPLv3	Apache 2

3.2.3. Veri işleme

Cihaz, servis ve uygulamalardan model tabanlı olarak üretilen, toplayıcılar üzerinden elde edilerek zaman serisi veri tabanlarında tutulan büyük ve devamlı akan verinin işlenmesi aşamasında en çok uygulanan yöntemlerden ikisi verinin görselleştirilmesi ve

programlanmasıdır. Sürekli ölçüm yaklaşımında yaygın kullanılan birçok araç aşağıda verilmiştir.

Grafana

Grafana [79], InfluxDB, OpenTSDB, Prometheus, Elasticsearch, CloudWatch gibi çeşitli kaynaklardan alınan zaman serisi biçimindeki gerçek zamanlı verilerin görselleştirilmesi, ön-tanımlı eşik değerlerine göre alarmların üretilmesi ve başka işleme ortamlarıyla birlikte çalışabilmesi amacıyla geliştirilen, açık kaynak kodlu kontrol panelidir.

Kibana

Elastic projesinin veri görselleştirme aracı olarak JavaScript ve HTML dilleriyle geliştirilen, browser tabanlı Kibana [80], Linux, Windows ve MacOS işletim sistemlerinde çalışabilmektedir. Veri bölgesel koordinat bilgisine sahipse haritalama yapabilmekte, frekansa bağlı yoğunluk haritası çıkarabilmekte, zaman serisi verilerini görselleştirebilmektedir.

Graphite

Kurumsal ölçekli ağ izleme aracı olarak 2006 yılında geliştirilmeye başlanan Graphite [81], Apache 2.0 lisansı altında açık kaynak edilmiştir. Zaman serisi verilerini inceleyen Carbon, verileri kısa süreli depolamaya yarayan veri tabanı kütüphanesi olan Whisper ve verileri görselleştirme arayüzü Graphite Webapp olmak üzere 3 bileşenden oluşmaktadır. Geniş ölçekli gerçek zamanlı grafikleştirme yapabilmektedir.

Chronograf

Telegraf-InfluxDB-Chronograf-Kapacitor yığınının kullanıcı arayüzü bileşeni olan Chronograf [82], zaman serisi veri tabanlarından aldığı verileri gerçek zamanlı olarak görselleştirebilmektedir. Önceden oluşturulmuş kontrol panelleri kullanılabileceği gibi kullanıcıya özel arayüzlerin tanımlanmasına da imkan verir. Chronograf, rol tabanlı kullanıcı kontrolüne ve role özel arayüzlerin gösterimine imkan verir.

Python

YTA mimarisinin en önemli özelliklerinin başında ağ davranışının 3.parti programlarla kontrol edilebilmesi gelmektedir. Python programlama dilinin en önemli özelliklerinden birisi, C ve C++ gibi dillerin aksine derlenmeye gerek olmadan çalıştırılabilmesi sayesinde yapılan değişikliklerin hızlıca deney ortamına alınabilmesidir. Python programlama dilinin basit ve temiz söz dizimi sayesinde hem program yazmanın hem de başkası tarafından yazılmış bir programı okumanın başka dillere kıyasla kolay olması Google, Youtube, Yahoo, Facebook gibi büyük şirketlerin bu dile olan yakınlığını arttırmıştır.

Modülerliği, okunabilirliği desteklemesi, ortam bağımsız nesne yönelimli, yorumlanabilir, interaktif bir betik dili olması, olağandışı durumlar, dinamik yazım, oldukça yüksek dinamik veri türleri ve sınıfları ile birlikte çalışabilmesi, birçok sistem çağrısına ve kütüphanesine uygun olan birden fazla arayüze sahip olması, programlanabilir arayüz ihtiyacı gibi uygulamalarda genişletilmiş dil olarak kullanılabilmesi, sınırlı kaynaklara sahip ağ ortamları için portatif olabilmesi, Unix, Mac, MS-DOS, Windows, Windows NT ve OS/2 gibi birçok işletim sistemlerinde çalışabilmesi gibi özellikleri ile Python yaygın kullanılan programlama dilidir [89].

3.2.4. Sanallaştırma ortamı

Yapılacak geliştirmeler için fiziksel ortam kullanılabileceği gibi, maliyet, tekrar edilebilirlik, ölçeklenebilirlik gibi ihtiyaçlar sebebiyle bu çalışmalarda genelde kaynakların bir bütün oluşturacak şekilde bir arada kullanılabileceği sanallaştırma ortamlarına ihtiyaç duyulur. Sahip olunan sanal makine ve üreticiye özel cihaz işletim sistemi imajlarının birlikte çalışabileceği, açık kaynaklar kullanılarak geliştirmelerin yapılabileceği güncel birkaç sistem aşağıda verilmiştir.

Vagrant

Vagrant [83], sanal makina programları (virtualbox, vmware) üzerine kurulan işletim sistemlerini konsoldan yöneterek, yazılımcılara gerçek ortamla birebir aynı geliştirme ortamı sağlamak için geliştirilmiş sanal makine yöneticisi, başka bir deyişle sanallaştırma

uygulamasıdır. Birçok üreticinin sanal makine olarak çıkardığı ortamları bir çatı altında birleştirmek için kullanılır.

VIRL

Cisco firması tarafından, ağ sanallaştırma ve birleştirme ortamı olarak geliştirilen VIRL [84], işletim sistemleri, sanallaştırma ortamları ve konteynır yapılarını destekler. Bu araç yardımıyla kullanıcılar ağlarında gerçekleştirecekleri değişiklikleri düşük laboratuvar maliyetleri ile riskten bağımsız olarak test edebilmektedir.

Grafik Ağ Simülatörü 3

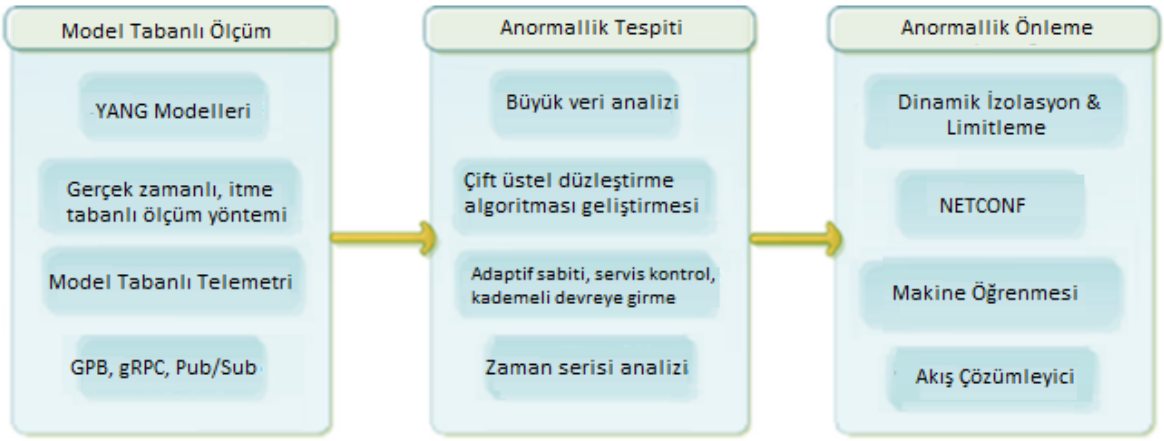
"Graphical Network Simulator" kelimelerinin baş harfinden oluşan GNS3 [85] açık kaynak kodlu bir emülatör programıdır. Ağ üreticilerinin cihazlarındaki işletim sistemlerini emüle ederek gerçek bir cihazda çalışıyor gibi yapılandırmaya imkan veren GNS3, aynı zamanda VMware gibi sanallaştırma yazılımlarının yarattığı sanal makineleri kullanarak mimariler oluşturabilir, bu sanal makineleri birbirleri ile haberleştirebilir. Hem ağ ortamına hem de kullanıcı ortamına emüle edilebildiği için yaygın kullanılan bir lab/sanallaştırma ortamıdır.

Mininet

Mininet [86] emülatörü, özellikle YTA'ların ortaya çıkmasından sonra bu konuya özel geliştirilmiş, OpenFlow tabanlı YTA'lar oluşturulmasına, tekrar tekrar test edilmesine, gerçeklenmesine olanak sağlayan açık kaynak kodlu bir emulasyon projesidir. Tek bir makine üzerinde çalışan gerçek çekirdek, anahtar ve uygulama kodu olan gerçekçi bir sanal ağ oluşturur. Bu tez çalışmasında ölçüm verilerinin gerçek zamanlı elde edilmesi için itme temelli sürekli ölçüm modeli kullanılacaktır. Mininet'te yer alan vSwitch sanal anahtarı YANG gibi veri modellerini desteklemediğinden bu çalışma kapsamında kullanılmayacaktır.

4. YAZILIM TANIMLI AĞLARDA TELEMETRİK YÖNTEMLE SALDIRI TESPİT VE ÖNLEME SİSTEMİ

Geliştirilen tez çalışmasının modüler gösterimi Şekil 4.1'deki gibidir. Buna göre; ilk modül sürekli ölçüm yaklaşımıyla tasarlanmış model tabanlı ölçüm metriklerinin gerçek zamanlı ve itme mimarisinde cihaz dışına alınmasından sorumludur. Ölçüm metrikleri işlenmek üzere denetleyiciye geldiğinde geliştirilen özel bir algoritma büyük veri üzerinde zaman serisi analizi yaparak anormallik tespiti yapmakta, anormallik tespit edildiği taktirde bir sonraki modül saldırı önleme aksiyonlarını otomatik olarak işletmektedir.



Şekil 4.1. Önerilen çözümün modüler gösterimi

Bundan sonraki kısımda yapılan geliştirmeler detaylı olarak anlatılmaktadır.

4.1. Özelleştirilmiş Toplayıcı

Pipeline toplayıcısı, Cisco'nun yeni nesil ağ işletim sistemi olan XR versiyonunun telemetri verilerini cihaz dışına çıkarmak için geliştirdiği, yayınla/abone ol mimarisinde çalışan, model tabanlı verileri taşıyabilen, Go dilinde yazılmış açık kaynak kodlu veri yoludur. TCP, gRPC/HTTP2, UDP, Apache Kafka gibi farklı giriş veri yapılarını, GPB K/V ve JSON veri serileştirme yöntemlerini ve UDP, TCP, gRPC gibi taşıma katmanlarını destekler. Ayrıca; TLS seçeneği ile cihazlardan alınacak verilerin şifreli olarak taşınmasına olanak verir.

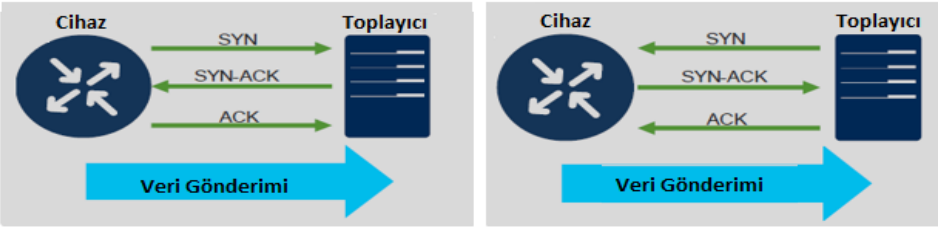
Yapılan çalışmaya özel büyük veri setine en uygun protokol yığınını belirlemek için tüm ayarlamaları içeren başarım testleri uygulanmış, sonuçları Çizelge 4.1'de verilmiştir. Yapılan testler, gRPC + GPB (Compact) ikilisi, oluşan veri boyutu sebebiyle ölçüm

maliyetini en aza indirgediğini, bant genişliği tüketimde verimlilik ve yüksek hız sağladığını, büyük veri yoğunluğu altında yüksek güvenilirlik ve başarımla çalıştığını göstermiştir. Açık kaynak kodlu RPC framework olan gRPC'nin diğer 2 önemli faydası tercihe dayalı TLS şifrelemesi ve kaydolma haberleşmesini desteklemesidir. Tüm bu sebeplerle bu çalışmada gRPC + GPB (Compact) ikilisi kullanılmıştır.

Çizelge 4.1. Protokollerin başarım karşılaştırması

Taşıma Protokolü	Encoding Yöntemi	Veri Modeli	1 dk'da oluşan Veri Boyutu (KB)	BW Verimliliği (D / O / Y)	1 akışın sunucuya varma sür. (ms)	Hız (1/Gecikme) (D / O / Y)	Güvenilirlik (D/Y)
UDP	GPB (Compact)	YANG	40	Yüksek	0,01895	Yüksek	Düşük
UDP	KVGPB (Self-Desc.)	YANG	260	Orta	0,06942	Orta	Düşük
UDP	JSON	YANG	290	Düşük	0,75839	Düşük	Düşük
TCP	GPB (Compact)	YANG	60	Yüksek	0,57751	Orta	Yüksek
TCP	KVGPB (Self-Desc.)	YANG	305	Orta	0,85356	Düşük	Yüksek
TCP	JSON	YANG	320	Düşük	0,95675	Düşük	Yüksek
gRPC	GPB (Compact)	YANG	45	Yüksek	0,02653	Yüksek	Yüksek
gRPC	GPBKV (Self-Desc.)	YANG	280	Orta	0,04163	Yüksek	Yüksek
gRPC	JSON	YANG	310	Düşük	0,06032	Orta	Yüksek

Bu alanda, kaydetme ve kaydolma şeklinde 2 temel taşıma modeli bulunmaktadır. Kaydetme yönteminde akış verilerini aktarmak üzere iletişimi yönlendirici başlatır. Kaydolma yönteminde ise bu 3-yönlü el sıkışma işlemi toplayıcı tarafından başlatılır. Her iki yöntem için de unutulmaması gereken; eşleşme tamamlandığında sürekli ölçüm yaklaşımına uygun olarak veri akışının tek yönlü olduğu, herhangi bir sorgu-cevap yapısının olmadığıdır. Ağda güvenlik duvarı gibi yapıların olması veya gRPC ile cihaz ayarlamalarının yapılacağı durumlarda kaydolma haberleşmesi, çok geniş ağlarda ölçeklenebilirliği arttırmak için ise kaydetme yöntemi kullanılabilir. Şekil 4.2, bu taşıma başlatma modellerinin ilk paket alışverişlerini göstermektedir.



a. Kaydetme b. Kaydolma

Şekil 4.2. 2 temel taşıma başlatma modeli

Seçilecek modele göre toplayıcı ve cihaz tarafından yapılacak düzenlemeler Şekil 4.3'te gösterilmiştir.

TOPLAYICI

```
type = grpc
encap = gpb
listen = IP:port
tls = false
```

CİHAZ

```
telemetry model-driven
 destination-group Topla1
  address-family IP port port
  encoding self-describing-gpb
  protocol grpc no-tls

 sensor-group olcum
  sensor-path cisco-IOS-XR-shellutil-oper:system-time/uptime

 subscription olcum
  sensor-group-id olcum sample-interval 5000
  destination-group Topla1
```

a. Kaydetme

TOPLAYICI

```
type = grpc
encoding = gpbkv / gpbcompact
encap = gpb
server = IP:57500
subcriptions = olcum
tls = false
```

CİHAZ

```
grpc
 port 57500

telemetry model-driven
 sensor-group olcum
  sensor-path cisco-IOS-XR-shellutil-oper:system-time/uptime

 subscription olcum
  sensor-group-id olcum sample-interval 5000
```

b. Kaydolma

Şekil 4.3. 2 temel taşıma başlatma modeli yapılandırması

Her iki yöntem de denenmiş başarım yönünden önemli bir fark gözlenmemiştir. Bu sebeple bu çalışmada yayınlama/abone olma mimarisi olan kaydolma yöntemi kullanılmıştır.

Tüm bu bilgiler ışığında Pipeline toplayıcısı, Şekil 4.4'teki gibi ayarlanmıştır. Buna göre streaming telemetry taşıma protokolü gRPC, serileştirme yöntemi olarak GPB-Compact, kapsülleme olarak GPB kullanılmış, gecikmeyi azaltmak için şifreleme devre dışı bırakılmıştır.

TOPLAYICI

```
[mymdtrouter]
stage = xport_input
type = grpc
encoding = gpbcompact
encap = gpb
server = 172.16.16.1:57500
subcriptions = health
tls = false
```

Şekil 4.4. Toplayıcının özelleştirilmiş yapılandırması

Model tabanlı verinin cihazdan dışarı alınabilmesi için cihaza abone olacak toplayıcıyla birlikte model tabanlı ağ cihazında da bunlara karşılık gelecek düzenlemelerin yapılması gerekmektedir. Şekil 4.5, Internet İşletim Sistmi (IOS) XRv'de yapılan ayarları göstermektedir. 1.kısım gRPC protokolünün çalışacağı port numarasını, 2. kısım hangi metriklerin gönderileceğini, 3. kısım ise hangi sıklıkta gönderileceğini ifade etmektedir.

CİHAZ

```
grpc
 port 57500

telemetry model-driven
 sensor-group health
  sensor-path Cisco-IOS-XR-infra-statsd-oper:
              infra-statistics/interfaces/interface/latest/generic-counters
 subscription health
  sensor-group-id health sample-interval 4000
```

Şekil 4.5. Model tabanlı ağ cihazının yapılandırması

Ayarlar her iki tarafta da başarıyla girildiğinde cihaz üzerinden durumun Aktif, aynı şekilde Toplayıcı tarafında da oturumun gRPC kaydolma şeklinde başladığı ve akışların alındığı Şekil 4.6'da görülmektedir.

```
INFO[2018-09-16 13:00:40.976896] gRPC starting block
INFO[2018-09-16 13:00:40.977505] gRPC: Start accepting dialout sessions
INFO[2018-09-16 13:01:09.775514] gRPC: Receiving dialin stream
```

a. Toplayıcı tarafındaki başarı gRPC oturumu

```
#show telemetry model sub health
Subscription:  health
-------------
  State:       ACTIVE
  Sensor groups:
  Id: health
    Sample Interval:      4000 ms
    Sensor Path:          Cisco-IOS-XR-infra-statsd-oper:
                          infra-statistics/interfaces/interface/latest/generic-counters
    Sensor Path State:    Resolved

  Destination Groups:
  Group Id:               health
    Destination IP:       172.16.16.16
    Destination Port:     48667
    Encoding:             gpb
    Transport:            dialin
    State:                Active
    No TLS
    Total bytes sent:     5723
    Total packets sent:   4
    Last Sent time:       2018-09-16 13:02:28.2143492698 +0000
```

b. Cihaz tarafındaki başarılı gRPC oturumu

Şekil 4.6. Yayınalam/abone olma yapısının iki taraflı tesisi

XRv ile toplayıcının bulunduğu sunucu arasındaki bağlantı Wireshark ile izlendiğinde HTTP/2 üzerinde çalışan gRPC taşıma protokolü ile taşınan, GPB ile ikili olarak serileştirilmiş YANG modelli ölçüm verileri görülebilmektedir. gRPC HTTP/2 protokolü ve onun altında da TCP ile taşınır. Bu sebeple Şekil 4.7'deki Wireshark çıktısında Ethernet Başlığı => IP Başlığı => TCP Başlığı => HTTP2 => DATA şeklinde bir akış görülmektedir.

```
    50 59.206852    172.16.16.1     172.16.16.16    HTTP2     750 DATA
    51 59.207002    172.16.16.16    172.16.16.1     TCP        66 57500 → 64421 [ACK] Seq=32 Ack=11619 Win=53440 Len=0 …
    52 59.207479    172.16.16.16    172.16.16.1     TCP        66 57500 → 64421 [ACK] Seq=32 Ack=15191 Win=60608 Len=0 …
    53 63.218545    172.16.16.1     172.16.16.16    TCP      1510 64421 → 57500 [ACK] Seq=15191 Ack=32 Win=45056 Len=14…
    54 63.218644    172.16.16.1     172.16.16.16    TCP      1510 64421 → 57500 [ACK] Seq=16635 Ack=32 Win=45056 Len=14…
    55 63.218697    172.16.16.1     172.16.16.16    TCP      1510 64421 → 57500 [ACK] Seq=18079 Ack=32 Win=45056 Len=14…
    56 63.218746    172.16.16.1     172.16.16.16    HTTP2     750 DATA

> Frame 50: 750 bytes on wire (6000 bits), 750 bytes captured (6000 bits) on interface 0
> Ethernet II, Src: PcsCompu_cc:74:f5 (08:00:27:cc:74:f5), Dst: PcsCompu_c1:54:c8 (08:00:27:c1:54:c8)
> Internet Protocol Version 4, Src: 172.16.16.1, Dst: 172.16.16.16
> Transmission Control Protocol, Src Port: 64421, Dst Port: 57500, Seq: 14507, Ack: 32, Len: 684
> [4 Reassembled TCP Segments (5016 bytes): #47(1444), #48(1444), #49(1444), #50(684)]
> HyperText Transfer Protocol 2
```

Şekil 4.7. Sürekli ölçüm verisinin benzetim ortamındaki Wireshark çıktısı

4.2. Veri Modelleme

Mevcutta komut satırı arayüzü, SNMP gibi geleneksel yöntemler yapısal olmayan, her üreticinin özelindeki ayarların cihazlara ulaştırılmasına hizmet ederken bu durum aynı zamanda ağ altyapılarında otomasyonun önündeki en zorlu engellerdendir. YANG veri modelleme dili tam bu noktada, cihazlardan ölçüm verilerinin çıkarılmasında ve cihazların konfig edilmesinde esnek, ölçeklenebilir, yapısal, bir çatı oluşturmaktadır. Bununla birlikte, InfluxDB gibi zaman serisi veri tabanları verileri tek katmanlı yapıda işleyebilmektedir. Bu sebeple çok katmanlı düzendeki, YANG modelli ölçüm verilerinin yapısal fakat tek katmanlı biçime dönüştürülmesi gerekmektedir. Bu amaçla cihazdan toplayıcıya YANG modellemesiyle alınan ölçüm verileri veritabanına gönderilmeden önce denetleyici tarafından Şekil 4.8'deki gibi JSON biçimine dönüştürülmüştür. Bu dönüşüm için açık kaynak kodlu pyang [87] yazılımı kullanılmıştır.

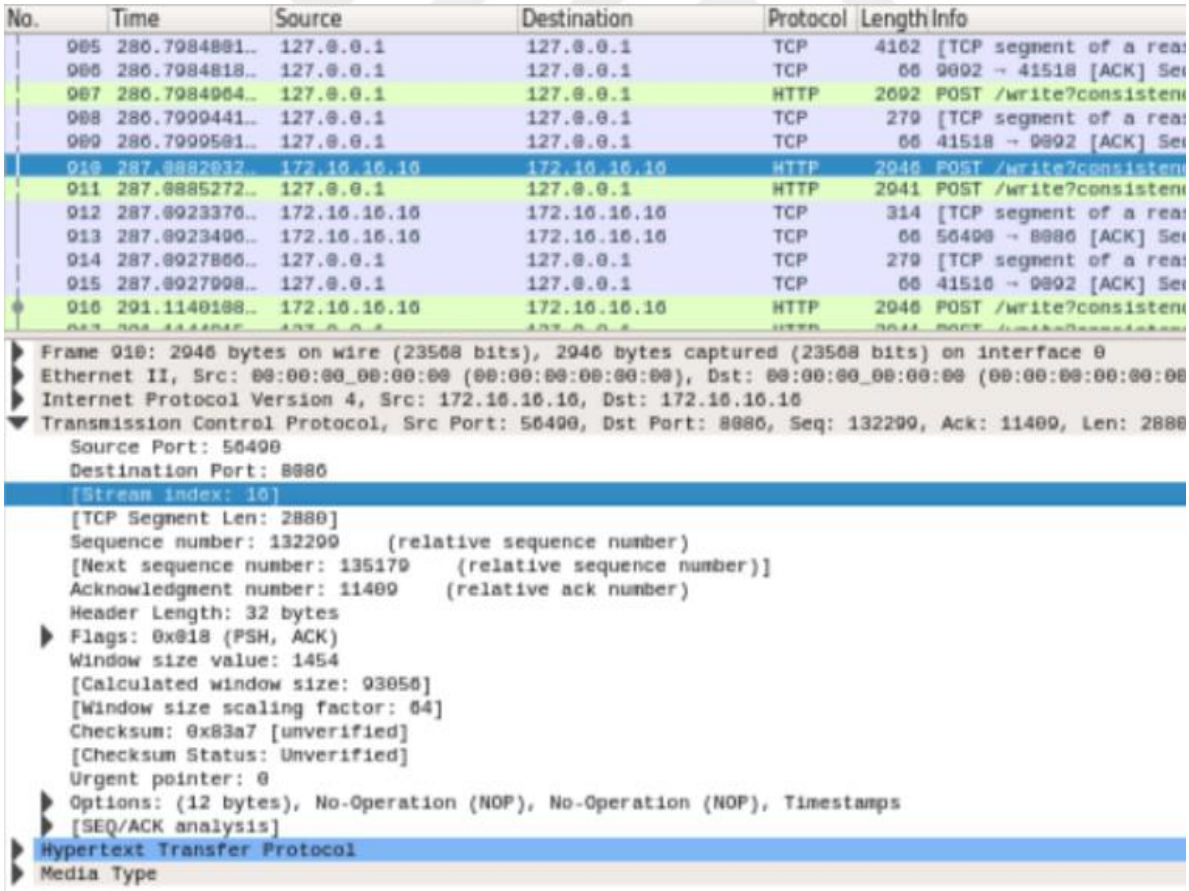
```
{
"basepath" : "Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters",
"spec" : {
        "fields" :
                [
                {"name" : "interface-name", "tag" : true},
                {"name" : "packets-received"},
                {"name" : "bytes-received"},
                {"name" : "packets-sent", "track": true},
                {"name" : "bytes-sent"},
                {"name" : "output-drops"},
                {"name" : "output-queue-drops"},
                {"name" : "input-drops"},
                {"name" : "input-queue-drops"},
                {"name" : "input-errors"},
                {"name" : "crc-errors"},
                {"name" : "input-ignored-packets"},
                {"name" : "output-errors"},
                {"name" : "output-buffer-failures"},
                {"name" : "carrier-transitions"}
                ]
        }
}
```

Şekil 4.8. Ölçüm metriklerinin JSON biçimine dönüştürülmesi

Şekil 4.9.a, ZSVT'nındaki kayıtları, Şekil 4.9.b ise toplayıcıdan ZSVT'nın 8086 portuna giden verileri göstermektedir. Akış ölçümü yaklaşımının en önemli getirilerinden biri olan gerçek zamanlı veri toplayabilme özelliği Şekil 4.9.a'da veri tabanı kayıtlarındaki zaman damgasında, ayarlandığı şekilde 4 saniye olarak görülmektedir.

```
cagdas@osboxes:~$ curl -G 'http://localhost:8086/query?pretty=true' --data-urlen
code "db=mdt_db" --data-urlencode "q=SELECT \"packets-sent\" FROM \"Cisco-IOS-XR
-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters
\" WHERE \"interface-name\"='GigabitEthernet0/0/0/1'"

                    [
                        "2018-11-03T12:39:12.716Z",
                        680
                    ],
                    [
                        "2018-11-03T12:39:16.727Z",
                        717
                    ],
                    [
                        "2018-11-03T12:39:20.737Z",
                        753
                    ],
                    [
                        "2018-11-03T12:39:24.747Z",
                        797
                    ],
                    [
                        "2018-11-03T12:39:28.759Z",
                        837
                    ],
                    [
                        "2018-11-03T12:39:32.769Z",
                        873
                    ],
                    [
                        "2018-11-03T12:39:36.797Z",
                        909
                    ]
                ],
                "partial": true
            }
        ]
```

a. ZSVT Kayıtları



b. Toplayıcıdan ZSVT'na giden veri

Şekil 4.9. Toplayıcı-ZSVT yapılandırması

4.3. Anormallik Tespit Yöntemi

Model tabanlı akış ölçümü yaklaşımıyla cihazlardan toplayıcı yardımıyla toplanan ve zaman serisi şeklinde depolanan veriler, Python betik dilinde kodlanmış özel bir anormallik tespit algoritmasıyla gerçek zamanlı olarak analiz edilmiştir. Yanlış alarmları azaltmak ve ölçüm doğruluğunu arttırmak amacıyla algoritma servis seviyesi kontrolü ve kademeli devreye girme yöntemleriyle desteklenmiştir.

Ağ anormallikliklerinin saptanmasında uygulanan en temel yöntemlerden biri ağın normal davranışının bir modelini çıkarmak ve bu normal eğrisindeki ani değişim ve sapmaları yakalamaktır. Bu yöntemle sıfırıncı gün atakları çok kolay yakalanabilirken, yöntemin olumsuz yanı yüksek hacimde gelmeyen saldırıların saptanma ihtimalinin düşük olmasıdır. Bu çalışmada uygulamaya özel ataklar kapsam dışı tutulmuş, saldırı tipi olarak yüksek hacimli D/DoS atakları baz alınmıştır. Yüksek hacimli ataklar, saldırılan sisteme çok yüksek miktarlarda trafik, paket veya istek göndermek yoluyla bant genişliği ve sunucunun diğer kaynaklarını tüketerek servisi erişilmez hale getirmeyi hedefler. Bu saldırılar botnet veya zombi adındaki son kullanıcılar üzerinden sistematik şekilde uygulanır. UDP, ICMP gibi yüksek hacim atak anormallikleri sistem tarafından algılanabilmektedir. Geliştirilen algoritma ve diğer anormallik tespit mekanizmaları aşağıda verilmiştir.

Zaman serilerinde tahminleme için bu alanda, otomatik gerileyen entegre hareketli ortalama (AutoRegressive Integrated Moving Average, ARIMA), sinir ağları, üstel düzleştirme yöntemi, Holt-Winters metodu, aritmetik hareketli/ağırlıklı hareketli ortalama, genelleştirilmiş koşullu otomatik gerileyen varyans (Generalized AutoRegressive Conditional Heteroskedasticity, GARCH), durum uzayı modeli, dinamik regresyon modeli gibi birçok algoritma kullanılmış olup bazılarının kıyaslaması Çizelge 4.2’de verilmiştir.

Çizelge 4.2. Bazı tahminleme yöntemlerinin karşılaştırması

Metot ve Modeller	Avantajları	Dezavanatajları
ARIMA (AutoRegressive Integrated Moving Avr.) [88]	Gerçekçi aralıklar kullanılır. Yorumlanabilirliği yüksektir Önyüklemesiz tahminleme yapabilir	Katı kuralları ve varsayımları vardır Otomatize etmesi zordur
Sinir Ağları [89]	Otomasyonu oldukça kolay ve hızlıdır Nonlineer örüntülerde başarılıdır Çok değişkenli serilerde iyi sonuç verir	Yorumlanabilirliği düşüktür Sağlıklı aralıkları belirlemek zordur. Çok fazla veri gerektirir
Doğrusal Regresyon [90]	Farklı zaman serisi bileşenleri hesaplanabilir Yorumlanabilirliği yüksektir	Sapan verilere aşırı duyarlıdır Çok fazla varsayım kullanılır
Üstel Düzleştirme [91]	Rasgele değişen eğilimlerde çok başarılıdır Büyük veride başarım artmaktadır Tüm veriler ele alınır, son verilerdeki değişimler ve sıçramalar daha ağırlıklıdır	Başarım verinin çoklığuyla doğru orantılılır Alfa ve beta sabitlerinin başarımı deneyseldir
Holt-Winters [91]	Sezonlu verilerde çok başarılıdır Büyük veri ile uyumludur	Alfa ve beta sabitlerinin başarımı deneyseldir Sezonsal olmayan verilerde başarım düşüktür
Dinamik Lineer Modeli [93]	Belirsizlik noktalarında iyi sonuç vermektedir Bileşenlerin değişkenliği kontrol edilebilmekte	Uyumsuzluk(holdout) hatası yüksektir Eğitme ve test için çok süreye ihtiyaç duyulur
Hareketli Ortalama Yöntemi [94]	Güncel veriyi etkili kullanmaktadır Eski veriyi modelden düşürmektedir Güncel bilgilerin etkisinin çok yoğun olduğu veri setlerinde başarım yüksektir	Sezonlu veya eğilim içeren veride etkili değildir Sadece lineer analizde başarılıdır

Eğilim yapısının zaman içinde rasgele değişebilmesi, başarımın büyük veri ile doğru orantılı olması, hem deterministik hem belirli bir düzen içinde seyretmeyen, belirsiz değişkenli, stokastik eğilimli seriler için başarılı sonuçlar üretebilmesi Holt Üstel Düzleştirme Yönteminin önemli avantajlarıdır. Ayrıca, verideki son değişime duyarlılığın fazla olması, bununla birlikte Hareketli Ortalama Yönteminin aksine eski verileri de modelden düşürmeden değerlendirmeyi güncelleştirerek öngörü sağlaması gibi özellikleri sebebiyle bu çalışmada zaman serisi analizlerinde en önde gelen yöntemlerden biri olan Holt metodu geliştirilerek kullanılmıştır. Holt metodunda kullanılan eşitlikler aşağıdaki gibi olup Eş. 4.1; t döneminde üstel düzeltilmiş tahmini, Eş. 4.2; t döneminde üstel düzeltilmiş eğilimi; Eş. 4.3; t+1 dönemindeki tahmini ifade etmektedir. Eşitliklerde kullanılan α ; hata düzleştirme katsayısını $(0 \leq \alpha \leq 1)$, β ; eğilim düzeltim sabitini $(0 \leq \beta \leq 1)$, Y_t ise t dönemindeki gerçek değeri ifade etmektedir.

$$F_t = \alpha Y_{t-1} + (1-\alpha)(F_{t-1} - T_{t-1}) \quad (4.1)$$

$$T_t = \beta(F_t - F_{t-1}) + (1-\beta)T_{t-1} \quad (4.2)$$

$$F_{t+1} = F_t + T_t \quad (4.3)$$

4.3.1. Adaptif hata sabiti

Yöntemde hata düzleştirme katsayısı olarak α ve eğilim tahmini için β düzleştirme katsayısı kullanılmaktadır. Hata payları, α katsayısıyla düzgünleştirilerek modele dahil edilir. Model, son dönemki tahmin ve bir önceki dönemki tahmin ile bir önceki dönem yapılan hatanın belli bir oranının toplamı şeklinde çalışmaktadır. Eğer α = 0 ise bir önceki dönemki hata hesaba katılmamakta; daha önceki dönemlerdeki hatalar yoğun kullanılmakta, 1'e yaklaştıkça bir önceki dönemki hatadan daha yoğun biçimde yararlanılmakta; daha önceki hataların etkisi düşük olmaktadır. α sabitinin tahmine etkisi Şekil 4.10'da verildiği gibidir.

$$F_t = \alpha Y_{t-1} + \alpha(1-\alpha)A_{t-2} + \alpha(1-\alpha)^2 A_{t-3} + \ldots$$

α=	Ağırlıklar		
	Bir önceki dönem	İki önceki dönem	Üç önceki dönem
	α	α(1 - α)	α(1 - α)²
α= 0.10	10%	9%	8.1%
α= 0.90	90%	9%	0.9%

Şekil 4.10. α sabitinin tahmine etkisi

Kestirim metodunun başarısı, özellikle α katsayısının en az hata içerecek şekilde belirlenmesiyle doğru orantılı olduğundan hatayı minimize eden α sabiti mevcut algoritmada deneme yanılma yoluyla tespit edilmektedir. Hataları toplama ya da çıkarma yoluyla toplam hatanın belirlenmesi yanıltıcı olabileceğinden hataların ölçümünde Karelerin Toplamının Ortalaması (Ortalama Hata Kare-MSE) ve Mutlak Değerlerin Ortalaması (Ortalama Mutlak Sapma-MAD) kullanılmaktadır. Geliştirilen bir arayüz ile, 8 farklı trafik örüntüsü için farklı α değerlerine ilişkin MSE değerleri hesaplanmış, Çizelge 4.3'te verilmiştir. MSE'nin en küçük olduğu α değeri en başarılı tahminleme yapacaktır.

XRv cihazından alınan ölçüm veri setleriyle yapılan testin sonuçlarında en düşük hatalı α değerinin veri setine göre değişkenlik gösterdiği saptanmıştır. Buna göre, geliştirme olarak algoritmaya her çalıştırma öncesi ön-gereklilik olarak dinamik öğrenme fonksiyonu eklenmiş ve bu öğrenme sürecinde dinamik olarak hesaplanan α sabitinin kullanılması planlanmıştır. Geliştirmenin testi için sabit ve dinamik öğrenilen α değerleri 50 farklı trafik

örüntüsüne uygulanmıştır. Sabit α değerinin kullanıldığı durumda sistemin anormallikleri doğru tespit etme başarısı %72 iken, öğrenme sürecinde trafiğe özel hesaplanan α değeri kullanıldığında doğruluk oranı %90 olarak hesaplanmış, sistemin başarısına, ön-gerekli dinamik öğrenme ve trafik karakteristiğine özel α sabitinin hesaplanmasının olumlu etki ettiği görülmüştür. β düzleştirme katsayısı için 0'a yakın değerlerin çalışmaya konu streaming veri seti için daha doğru sonuçlar verdiği, β sabiti yükseldikçe sistemin kararsız çalıştığı görülmüştür. Bu sebeple bu çalışmada β = 0,1 alınmıştır.

Çizelge 4.3. En iyi α değerinin belirlenmesi

α	Veri Seti	MSE	Veri Seti	MSE	Veri Seti	MSE	Veri Seti	MSE
0,1		45,31206		202,56599		45,07392		74,42366
0,2		17,16515		100,77906		20,38659		79,00651
0,3		18,47332		101,55808		27,50245		74,32236
0,4		24,35423		110,06223		26,45672		73,90621
0,5	**1**	25,84032	**3**	103,80753	**5**	28,35256	**7**	81,23417
0,6		25,92343		99,888421		20,39526		84,24567
0,7		26,58093		101,69095		20,01225		88,94521
0,8		27,41404		105,45252		36,25626		78,23453
0,9		28.03894		109,22218		43,34256		77,34325
0,1		192,6747		373,98163		174,2456		134,22332
0,2		93,93952		166,49653		173,5625		156,24214
0,3		93,27323		216,97842		198,4221		170,13221
0,4		110,9525		192,96568		203,4497		140,42261
0,5	**2**	102,0282	**4**	169,21582	**6**	207,4623	**8**	132,45355
0,6		94,52845		169,96245		194,1134		141,46325
0,7		93,62936		176,23187		180,5267		144,53576
0,8		95,62932		183,58942		179,4198		160,46396
0,9		98,06085		189,54315		175,2455		154,56622

Yoğun miktarda akan büyük ölçüm verisi, birçok anormallik tespit algoritmasında gerçek zamanlılıktan ödün verme sebebi iken Holt Üstel Düzleştirme Yöntemi temelli algoritmamızda işlenecek veri ne kadar yoğunsa sistemin başarısı da o oranda artmaktadır. Sürekli ölçüm yaklaşımı hat hızında ölçüm verisi ürettiğinden kullanılan algoritmanın sistemin tasarım amaçlarıyla uyuştuğu, başarımı arttırdığı gözlenmiştir. Algoritmadaki güncellemelere ek olarak servis kalitesi kontrolü ve kademeli devreye girme özellikleri sistemin başarısını arttıran diğer geliştirmelerdir.

4.3.2. Servis kalitesi kontrolü

Anormallik tespit algoritmasına göre saldırı olduğu tespit edildiğinde servisin ayakta olup olmadığı kontrol edilerek yanlış alarmların önüne geçmek amaçlanmıştır. Aynı yerel alan

ağında (YAA) bulunan kullanıcı ve sunucular için saldırı olmadığı durumlarda ortalama erişim süresi 30ms seviyelerinde ölçülmüştür. Atak anında servis kesintisi olduğunda bu ortalamaların yükseldiği, çoğu zaman paket kaybı olduğu gözlemlenmiştir. Saldırı anında servis kalitesi kontrol edilerek servisin düşüp düşmediğine bakılarak son aksiyona karar verilir. Şekil 4.11'de bu amaçla geliştirilen servis kalitesi kontrol kodunun çıktısı yer almaktadır. Buna göre, sunucunun servis veren 80 portuna 6 adet TCP paketi gönderilmiş, gelen cevaplara göre servis başarı oranı ve paket erişim süreleri en küçük, en yüksek, ortalama cinsinden hesaplanmıştır. Hesaplanan başarı oranı bir sonraki modülde devreye girmeye karar vermede kullanılmaktadır.

Sunucu	Port	Basarili	Kayıp	Basari Orani	Minimum	Maximum	Ortalama
10.10.10.10	80	6	0	100.00%	23.72ms	28.53ms	26.12ms

Şekil 4.11. Servis kalitesi kontrol çıktısı

4.3.3. Kademeli devreye girme

Tespit algoritmasında anormallik algılandıktan sonra servis kalitesi kontrol modülünde servis kesintisinin oranı hesaplanmıştı. Bu kısımda anormallik önleme politikalarının hangi başarı oranında devreye gireceği sisteme ağ yöneticisi tarafından bildirilmektedir. Bu modül, değişken karakteristiklere (patlamalı trafiğin yoğun ve olağan olduğu, tepe değerlerin çok fazla görülmediği, belli saat veya günlerde patlamaların olduğu vs.) sahip ağlarda önemli esneklik ve güvenilirlik sağlamaktadır.

Modelde, yanlış alarm, hafif servis kesintisi, yoğun servis kesintisi ve servis çalışmıyor olmak üzere 4 kademe belirlenmiştir. Servis başarı oranı %90 ve yukarısındaysa saldırı olmadığı, alarmın yanlış üretiğine karar verilip herhangi bir önleme politikası işletilmez. Servis başarı oranı %70-90 aralığındaysa düşük servis kesintisi, %50-70 aralığındaki servis başarı oranı yoğun servis kesintisi, %50'den küçük başarı oranı ise servisin artık çalışmadığı şeklinde yorumlanır. Ağ yöneticisinin belirleyeceği hassasiyet derecesine göre anormallik önleme aşamasına geçilir veya aksiyon alınmaz. Şekil 4.12, ağ yöneticisine açılan kademe giriş arayüzünü göstermektedir.

```
Bu çalışmada 4 Saldırı Seviyesi belirlenmiştir.

[1] Servis Başarı Oranı >=90%           --> FALSE POSITIVE

[2] Servis Başarı Oranı >=70% ve <90%   --> DÜŞÜK SERVİS KESİNTİSİ

[3] Servis Başarı Oranı >=50% ve <70%   --> YOĞUN SERVİS KESİNTİSİ

[4] Servis Başarı Oranı <50%            --> SERVİS ÇALIŞMIYOR

Korumanın hangi servis başarı seviyesi itibariyle devreye girmesini istersiniz ? [1/4] :
```

Şekil 4.12. Kademeli devreye girme yetkili arayüzü

4.4. Anormallik Önleme

Atak önleme için yapılan geliştirmeler aşağıda verilmiştir.

4.4.1. Atak önleme politikaları

Kademeli devreye girme kararı neticesinde koruma modülünün devreye girmesi aşamasına gelindiğinde saldırgan IP'lerin her biri için cihazın saldırı alan arayüzüne karantinaya alma politikaları hiçbir müdahaleye gerek olmadan otomatik olarak girilmektedir. Karantinaya alma : i) saldırgan IP adreslerinin bir süre boyunca bant genişliğinin ağ servislerinin çalışmasına engel olmayacak bir değere indirilmesi ii) bu adreslerin ağa erişimin tamamen izole edilmesi veya iii) bant genişliği kısıtlanan saldırganın saldırısının devam etmesi halinde belli bir süre sonra tamamen izole edilmesi şeklinde olabilmektedir.

4.4.2. Akış çözümleme

Geliştirilen bir Python kodu ile cihaz üzerindeki akışlar denetleyici üzerine alınıp çözümlenmektedir. Bu işlem sonrasında aynı akışlar burada birleştirilerek herbir kaynak IP adresi için trafik ağırlıkları hesaplanmaktadır. Şekil 4.13. bu şekilde hesaplanan akışları göstermektedir. 10.2.0.10, 10.0.0.10 ve 172.16.16.16 adreslerinden gelen trafikler gerçek kullanıcılardan, 80.80.80.80, 77.77.77.77 ve 150.150.150.150 kaynak adresli trafikler ise DDoS atağı yapan sahte IPlerden oluşmaktadır. Geliştirilen kodun akışlardaki tüm IP adreslerine ilişkin trafik ağırlıklarını doğru olarak saptayabildiği, saldırgan IP'lerin akıştan otomatik olarak çekildiği görülmektedir.

```
SrcIf         SrcIPaddress     DstIf         DstIPaddress     Pr SrcP DstP  Pkts
Gi1/0         80.80.80.80      Fa0/0         10.10.10.10      06 05EF 0050   14K
Gi1/0         77.77.77.77      Fa0/0         10.10.10.10      06 060A 0050   11K
Gi1/0         150.150.150.150  Fa0/0         10.10.10.10      06 068F 0050   8K
Gi1/0         10.2.0.10        Fa0/0         10.10.10.10      01 0000 0800   922
Gi1/0         10.0.0.10        Fa0/0         10.10.10.10      01 0000 0800   199
Gi1/0         172.16.16.16     Fa0/0         10.10.10.10      01 0000 0800   135
```

Şekil 4.13. Saldırgan IP'lerin akıştan çekilmesi ve trafik ağırlıklarının belirlenmesi

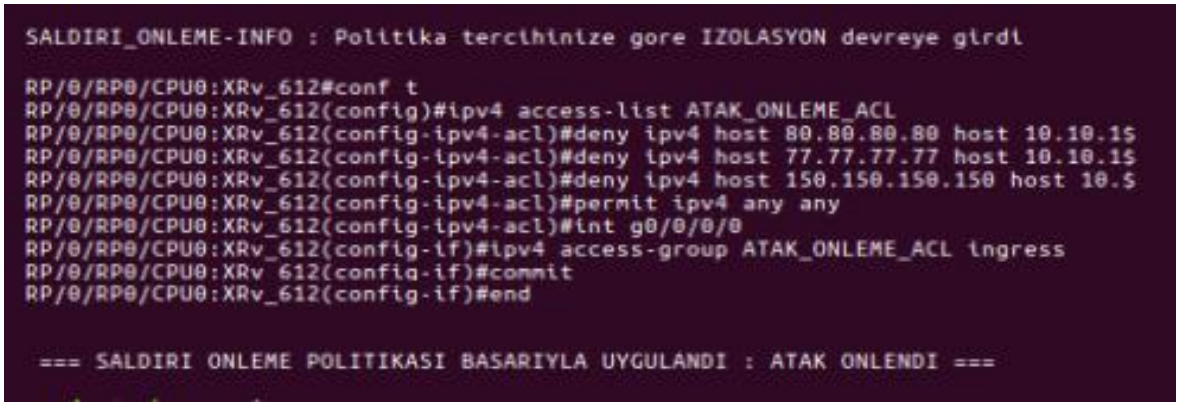
4.4.3. Dağıtık DoS ataklarına karşı koruma

Saldırgan IP'lerin ağ trafiğinden otomatik olarak çekilebilmesi sayesinde saldırı yapan tüm IP'ler tespit edilebilmekte, sistem tüm saldırgan IP'ler için koruma politikalarını işletebilmektedir. DDoS yapan IP'lerin sıralı veya belirli bir altağdan olma kısıtı bulunmamakta, geliştirilen kod 32bit'lik tüm IPv4 havuzu için aynı başarılı sonuçları vermektedir. Şekil 4.14.'te akıştan dinamik olarak çekilen trafik ağırlıklarının sınıflandırılması için K-Means kümeleme algoritması kullanılmıştır. Bu sayede akışlar "saldırgan" ve "gerçek trafik" olarak sınıflandırılabilmektedir. Kümeleme işlemi sonrasında yüksek trafik yaratan 3 sahte IP bir kümeye, gerçek IP'ler ise diğer kümeye ayrıştırılmıştır.

```
PARSER-INFO : Saldirgan IP'lerin tespiti icin PARSER devreye girdi

PARSER-INFO : Flowlar icin cihaza baglanti saglandi

PARSER-INFO : Flowlar okundu

PARSER-INFO : Flowlar parse edildi

PARSER-INFO : Trafic agirliklari hesaplandi :


             IPLER  PAKETLER
5      80.80.80.80     14000
4      77.77.77.77     11000
2  150.150.150.150      8806
1        10.2.0.10       922
0        10.0.0.10       199
3     172.16.16.16       135

PARSER-INFO : K-Means Normal ve Anormal IP ayristirmasi tamamlandi : [1 1 1 0 0 0]

PARSER-INFO : Saldirgan IP'ler tespit edildi : ['80.80.80.80', '77.77.[redacted].150.150.150']
```

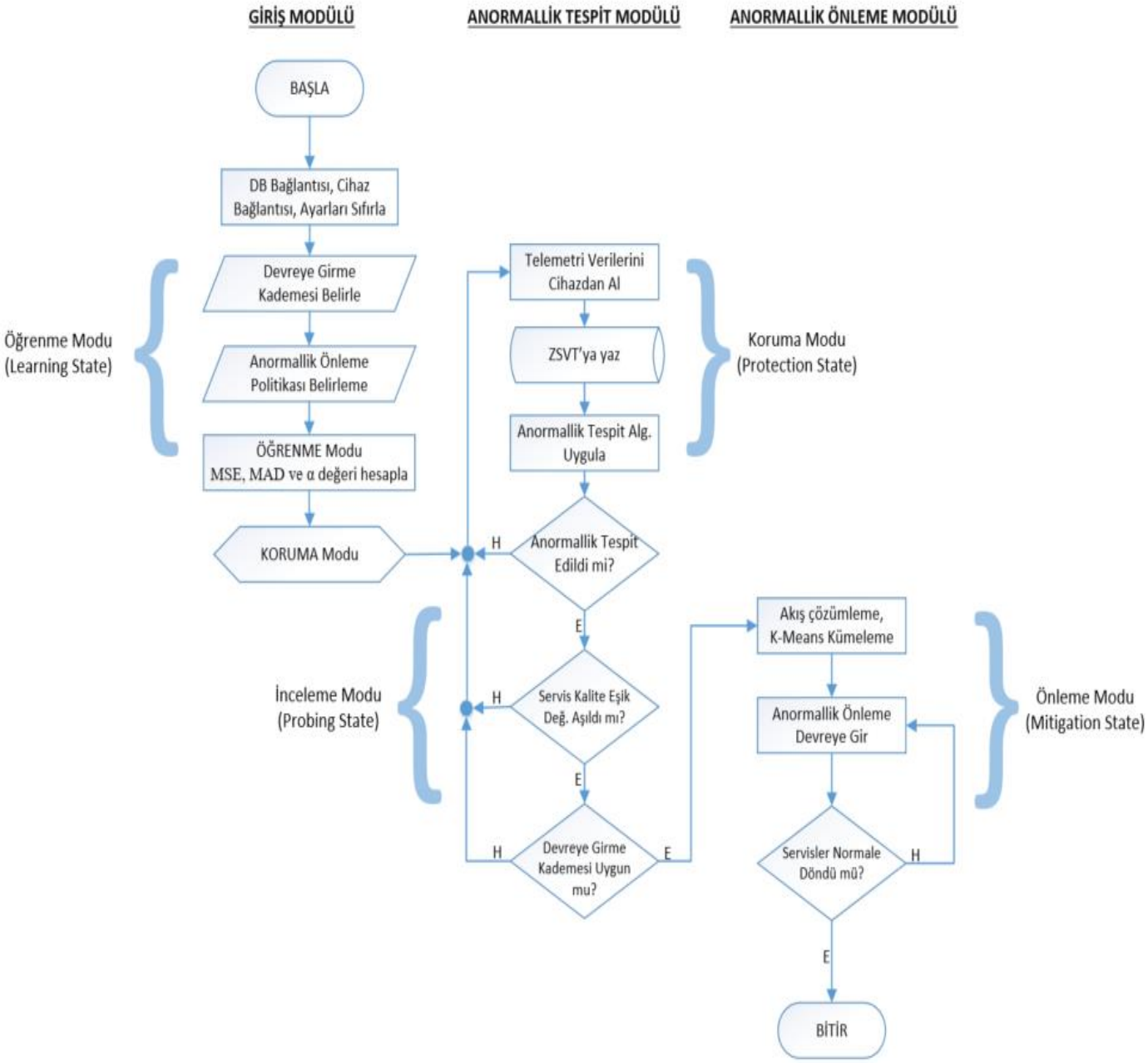
Şekil 4.14. K-Means ile normal ve anormal IP ayrıştırmasının yapılması

Ağ yöneticisinin politika tercihine göre izolasyon veya bant genişliği sınırlama algoritmalarından birisinin saptanan saldırgan IP'ler için devreye girmesi, Şekil 4.15. gösterilmektedir. Politika devreye girdikten hemen sonra normal kullanıcıların sunucu ile iletişiminin gerçek zamanlı olarak normale döndüğü görülmektedir.



Şekil 4.15. Saldırgan IP'ler için atak önleme politikasının devreye girmesi

Tez çalışmasının bir bütün olarak algoritma akış diyagramı Şekil 4.16'da verilmiştir. İlk olarak toplayıcı, model tabanlı cihaza ve ZSVT'na güvenli bağlantı sağlamakta, ön yükleme yapılandırmaları tamamlanmaktadır. Daha sonra ağ yetkilisinden devreye girme kademesi ve anormallik önleme politikasını girmesi istenmektedir. Daha sonra Holt'un algoritmasına eklenen dinamik öğrenme modu ile α değeri hesaplanmakta ve Koruma modu döngüsüne geçilmektedir. Cihazdan her 4 sn.'de gelen ölçüm verileri ZSVT'na yazılmakta, buradaki veriler daha sonra anormallik tespit algoritması tarafından kullanılmaktadır. Anormallik tespit edildiğinde servis kalitesi eşik değeri ve devreye girme kademe denetlemeleri yapılmaktadır. Tüm kontroller neticesinde anormallik tespit edildiyse önleme modu devreye girmektedir. Geliştirilen akış çözümleyici ile ağ trafiğinden IP'ler dinamik olarak çekilmekte, bu veri setine K-Means algoritması uygulanarak saldırgan IP'ler ayrıştırılmaktadır. Daha sonra belirlenen bu saldırgan IP'lere ağ yetkilisi tarafından belirlenen anormallik önleme kuralı uygulanmaktadır. Bu kural uygulama döngüsü, ağ servisleri normale dönene kadar devam etmektedir.



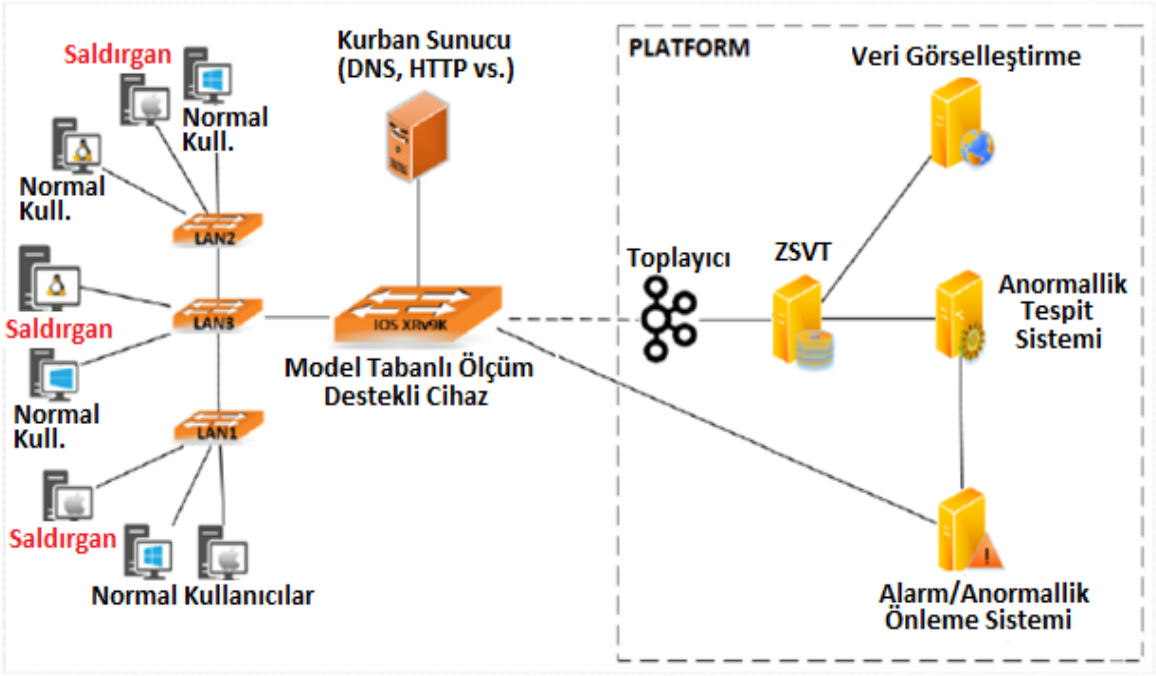
Şekil 4.16. Tez çalışmasının akış şeması

5. GELİŞTİRİLEN SİSTEMİN BENZETİMİ VE TESTİ

Bu bölümde geliştirilen sistemin testi için oluşturulan mimari ve benzetim ortamı detayları ile yapılan testler verilmiştir.

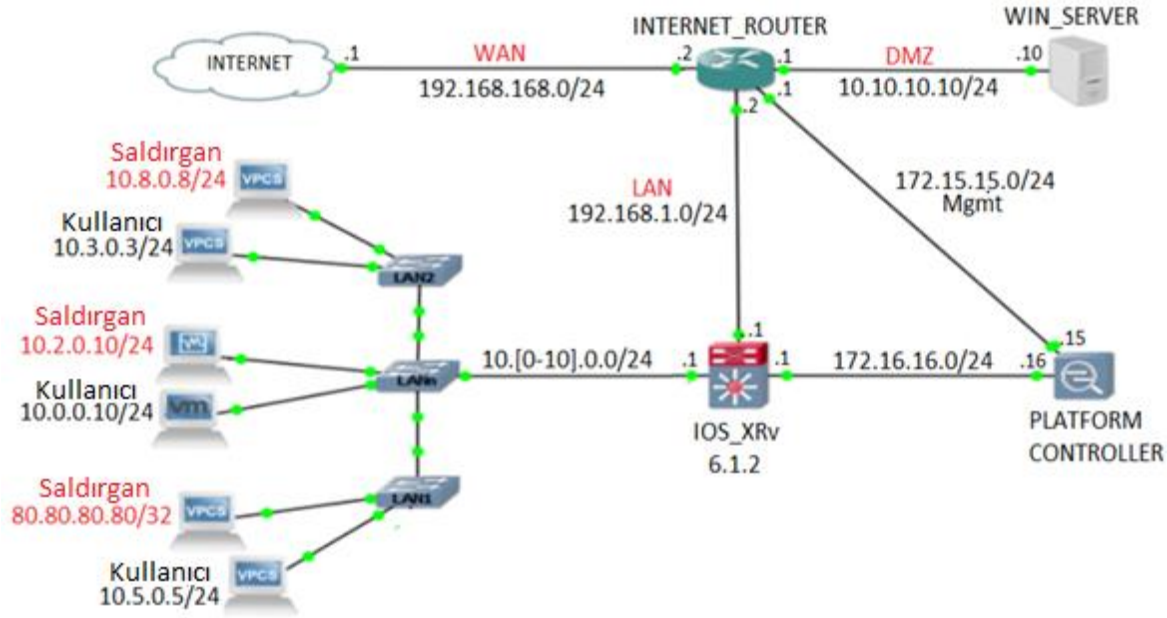
5.1. Mimari ve Benzetim Ortamı

Bu tez çalışmasında, saldırı ve anormallik tespitinin gerçek zamanlı veya gerçek zamanlıya yakın şekilde yapılmasına ilişkin geleneksel ağ ölçüm yöntemleri dışında, YTA yaklaşımı kullanılarak bir yöntem geliştirilmesine odaklanılmış, YTA mimarisinde geliştirilecek metotta model-tabanlı, sürekli ölçüm yönteminin kullanılması planlanmıştır. Bu kapsamda oluşturulan mantıksal mimari Şekil 5.1'de, fiziksel mimari Şekil 5.2'de verilmiştir.



Şekil 5.1. Mantıksal mimari

Ağ istemcileri anahtarlar yardımıyla toplama anahtar olan model tabanlı ölçüm destekli cihazda toplanmıştır. Ağa http ve DNS servisi sağlayan sunucu da bu toplama anahtarına bağlanmıştır. İçerisinde toplayıcı, ZSVT, anormallik tespit sistemini barındıran geliştirilen sistem sanal makine olarak mimariye dahil edilmiştir.



Şekil 5.2. Fiziksel mimari

Mimaride, ölçüm verilerinin model tabanlı ve gerçek zamanlı olarak Toplayıcı'ya gönderimine olanak vermesi, canlı ortamı uçtan uca gerçeklemesi, gRPC, TCP/UDP taşıma protokollerini ve GPB, JSON gibi serileştirme yöntemlerini desteklemesi gibi özellikleri sebebiyle telemetri destekli ağ cihazı olarak IOS XRv 6.1.2 [88] yazılımı kullanılmıştır. Donanıma gereksinim duymayarak tüm fonksiyonlarıyla sanal ortamda çalışabilen, sanal makine tabanlı bu sistem, Cisco tarafından üretilmiş yeni nesil ağ işletim sistemidir. Kontrol ve veri düzlemlerinin ayrıştırılması prensibini temele alan YTA mimarsine tam uyumlu olarak geliştirilmiş olan IOS XRv, kontrol düzlemindeki tüm fonksiyonları simulator veya emulator olarak değil, tamamen gerçek ortamda canlı sistem olarak çalıştırmakta, veri düzlemi olarak da x86 mimarisindeki sanal makinenin fiziksel arayüzlerini kullanmaktadır. Sistemin sanallaştırma kaynağı için 4 GB RAM ve i5-7200 2,50GHz MİB atanmıştır.

Saldırgan ve normal kullanıcılar VirtualBox ortamında sanallaştırılmış birer işletim sistemi olarak IOS XRv sanal cihazına bağlanmıştır. Saldırgan PC kaynağı için Windows 7 Enterprise İşletim Sistemi, i5-7200 2,50GHz MİB 1GB RAM, normal kullanıcı kaynağı için Ubuntu 14.04 versiyonu ile i5-7200 2,50GHz MİB, 1GB RAM kullanılmıştır.

Toplayıcı-ZSVT-İşleme birimlerini ve YTA kontrölcüsünü tek bir sanal makinede toplayarak sistemler arası paket gecikmesini en aza indirmek ve bu sayede gerçek zamanlılık

başarımını en yukarıya çıkarmak maçlanmıştır. Kısaca “Platform” olarak isimlendirilecek bu sistem için Ubuntu 16.04.04 versiyonlu işletim sistemi kullanılmıştır.

Kullanıcıların oluşturduğu trafiklerin ölçüm verileri model tabanlı olarak sistemin ilk bileşeni olan toplayıcıya gelmektedir. Üye olunan metriklerin ağ cihazı üzerinden itme mimarisinde çekilebilmesi için üretilmiş olması, açık kaynaklı ve kullanıcı tarafından geliştirilebilmesi, geniş ölçekli ağ ortamları için elverişli ve kararlı olması gibi özellikleriyle bu tez kapsamında toplayıcı olarak Pipeline projesinin çekirdeği kullanılmıştır.

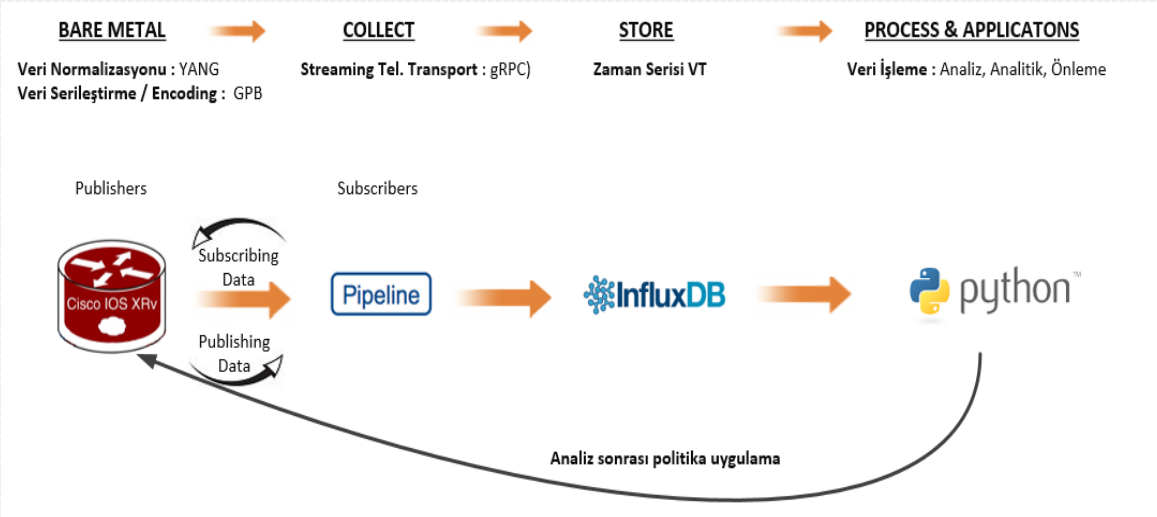
Toplayıcı tarafından alınan metrikler ZSVT’na iletilmekte, veriler burada veri yığını formundan zamanın bir işlevi şekline dönüştürülmektedir. Metin veya sayısal verinin bu şekilde dizinlemesi, karmaşık örüntüsü olan gerçek zamanlı veriye yüksek hızda erişim ve işleme olanağı sağlanmaktadır. Sahip olduğu Python istemcisi ile kullanıcıların kendine özel geliştirdikleri kodları veri tabanı üzerinde hızlıca uygulayabilmelerine olanak vermesi, operasyonel işletim kolaylığı, metrik ölçüm hassasiyetinin mikro saniyeler mertebesinde olması, bağıntı ve tüm işlemleri RAM üzerinde yapması gibi özellikleri sebebiyle bu çalışmada ZSVT olarak InfluxDB 1.6.4 kullanılmıştır.

Depolanan verilerin işlenmesi için kodlama ortamı olarak Python 2.7.12 [94] kullanılmıştır. Python programlama dilinin en önemli özelliklerinden birisi, C ve C++ gibi dillerin aksine derlenmeye gerek olmadan çalıştırılabilmeleri sayesinde yapılan değişikliklerin hızlıca deney ortamına alınabilmesidir. Modülerliği desteklemesi, ortam bağımsız nesne yönelimli, yorumlanabilir, interaktif bir betik dili olması, dinamik veri türleri ve sınıfları ile birlikte çalışabilmesi, birçok sistem çağrısına ve kütüphanesine uygun olan birden fazla arayüze sahip olması gibi özellikleri Python programlama dilinin diğer bazı avantajlarıdır [94].

Ağa alan adı servisi (Domain Name Server, DNS) ve http (Hyper Text Transfer Protocol) servisleri sunmak üzere Windows Server 2012 R2 işletim sistemli sunucu kullanılmıştır. Bu sunucuya 2048 GB RAM, 2,71GHz MİB atanmıştır.

Uç kullanıcılar, IOS XRv ağ işletim sistemi, tez kapsamında geliştirilen Platform ve saldırı yapılacak sunucu bileşenlerinin tamamı birer sanal makine şeklinde oluşturulmuştur. Sanal makine ortamı olarak yapılan çalışmalarda yaygın kullanılan VirtualBox 5.2.12 [95] ve sanal makineleri bir ortamda birleştirmek için GNS3 [96] emülatörü kullanılmıştır.

Çalışmanın uygulama yığını Şekil 5.3'te verilmiştir.

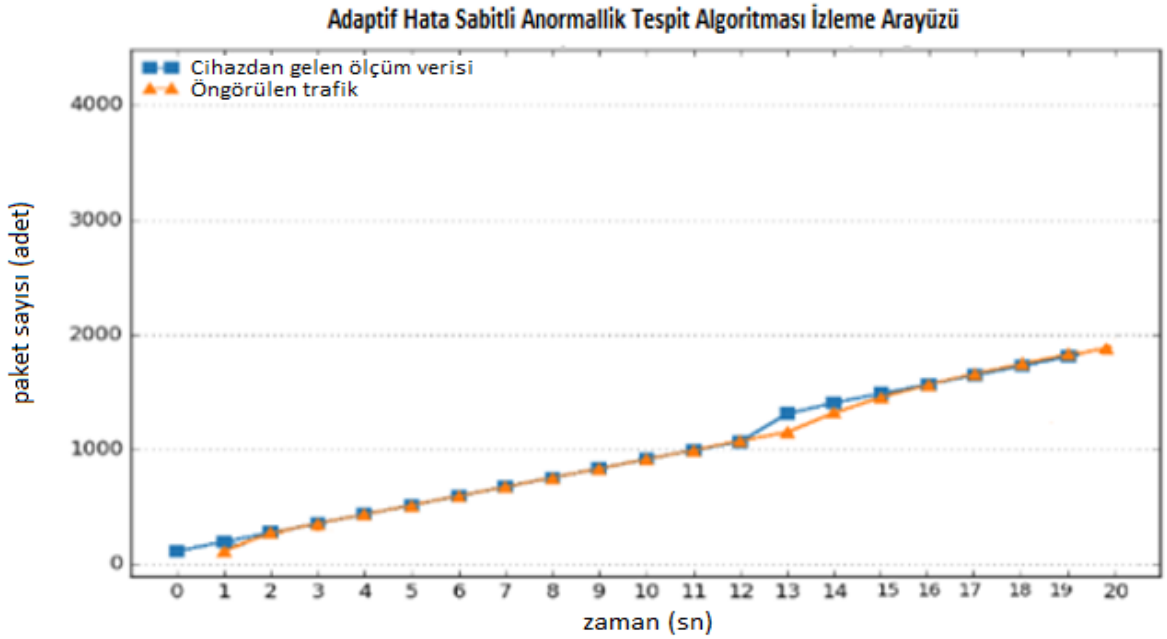


Şekil 5.3. Tezin uygulama yığını

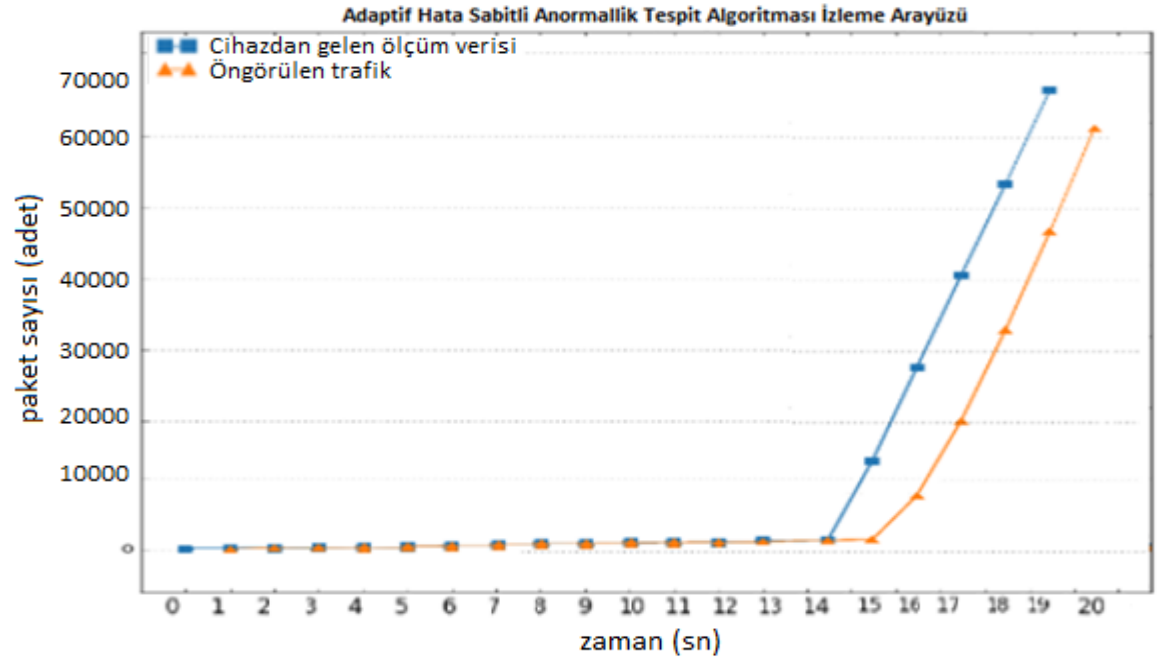
5.2. Sistemin Testi

Koruma modundayken ağ trafiği sürekli olarak izlenmektedir. Trafikte bir anormallik tespit edildiği an itibaren detaylı inceleme aşamasına geçilmektedir. Servis kesintisinin uzamaması ve aynı zamanda sistemin yanlış alarm hatalarının en düşük olması için detaylı inceleme aşaması 5-10 sn. olarak belirlenmiştir. Bu süre boyunca saldırının devam ettiği tespit edildiğinde servis kalitesi eşik değer sorgulaması ve devreye girme kademesi seçimine göre aksiyon alınmaktadır. Saldırının başladığı andan itibaren ortalama 5 sn. içerisinde saldırı önleme mekanizması tetiklenmekte, sonrasında 3-5 sn.'lik servis kalitesi sorgulamasının ardından 3-5 sn. içinde atak önleme mekanizması sonuçlanmaktadır. Yapılan tekrarlı testlerde saldırı başladığı andan itibaren ortalama 12 sn. içerisinde sistemin saldırıyı önlediği tespit edilmiştir.

Şekil 5.4 saldırı öncesi ve saldırı anındaki izleme arayüzü çıktılarını göstermektedir. Mavi çizgi sürekli ölçüm ile cihazdan alınan modelli ölçüm verisini, turuncu çizgi ise geliştirilen algoritma ile öngörülen bir sonraki veriyi işaret etmektedir. Şekil 5.4.a'nın 12-15 saniyelerinde küçük dalgalanmaların yaşandığı, algoritmanın bu dinamik duruma uyum sağladığı görülmektedir. Şekil 5.4.b'de ise sisteme gerçek saldırı uygulanmıştır. Bu durumda ölçülen ile tahminlenen veri miktarı arasındaki farkın belirli bir süre kapanmadığı için geliştirilen tez sistemi bunu saldırı olarak algılamaktadır.



a. Normal trafik



b. Anormal trafik

Şekil 5.4. Saldırı izleme arayüzü

Sanal ortamda gerçek kodlarla geliştirlen sistem, YTA'nın getirdiği faydalar neticesinde gerçek ortama çok hızlı şekilde uygulanabilmiştir. Bu kapsamda Internet_Router üzerinde gerekli gerçek IP yönlendirme ve ağ adres dönüşümü (Network Address Translation, NAT) işlevleri aktif edilmiş, bu çalışmanın konusu olmayan fakat ağın en dışında bulunan güvenlik cihazında alan adı sunucusu (Domain Name Server, DNS) ve hiper metin iletim protokolü

(Hyper Text Transfer Protocol, http) servisleri için gerekli izinler verilmiştir. Algoritmanın çalışma mantığı değişmediğinden anormallik tespit başarımında ve geliştirilen sistem ile IOS XRv sanal işletim sistemi aynı yerel alan ağında(YAA) bulunduğundan tepki sürelerinde bir fark gözlenmemiştir.

Holt metodunda kullanılan eşitlikler aşağıdaki gibi olup Eş. 5.1; tespi oranını, Eş. 5.2; yanlış tespit oranını, Eş. 5.3; doğruluk oranını, Eş. 5.4; hassasiyeti, Eş. 5.5; geri çekme değerini ve Eş.5.6; F1 değerini vermektedir.

$$\text{Tespit Oranı} = TP / (TP + FN) \quad (5.1)$$

$$\text{Yanlış Tespit Oranı} = (FN / (TN + FN) + (FP / (TN + FP))/2 \quad (5.2)$$

$$\text{Doğruluk} = (TP + TN) / (TP + FP + FN + TN) \quad (5.3)$$

$$\text{Hassasiyet} = TP / (TP + FP) \quad (5.4)$$

$$\text{Geri çekme} = TP / (TP + FN) \quad (5.5)$$

$$F1 = 2* ((\text{Hassasiyet}*\text{Geri Çekme})/(\text{Hassasiyet} + \text{Geri Çekme})) \quad (5.6)$$

Formüllerde kullanılan TP; atak olarak tespit edilen atak örüntüsünü, FP; atak olarak tespit edilen temiz trafiği, FN; temiz olarak tanımlanan atak örüntüsünü, TN; temiz olarak tespit edilen temiz trafiği ifade etmektedir.

Sistem, farklı trafik örüntülerinden oluşturulan senaryolar ile toplamda 100 kez kontrollü olarak test edilmiştir. Testlerde kullanılacak verileri oluşturmak için Scapy [97] uygulaması kullanılmıştır. Her biri 5 dakikalık uzunluğunda olan veri setleri, HTTP, DNS, TCP-SYN, TCP-ACK ve UDP protokollerinin birleşimi şeklindeki trafiklerin aktığı fiziksel arayüzlerin gelen paket sayaç değerleridir. Oluşturulan verilerin tekrar tekrar kullanılabilmesi için Tcpreplay [98] aracı kullanılmıştır. Testlerde kullanılan ölçüm metriklerini içeren veri setlerinden bazıları Çizelge 5.1.'de verilmiştir.

Çizelge 5.1. Testlerde kulanılan veri seti örnekleri

Veri Seti-1	92, 510, 980, 1320, 1900, 2310, 2740, 3178, 3512,…
Veri Seti-2	1400, 2950, 4250, 6750, 7800, 9300, 10620, 11480,…
Veri Seti-3	6500, 16000, 23000, 32500, 51000, 60000, 67000,…

Testlerin %75'inde HTTP, DNS, TCP-SYN, TCP-ACK ve UDP taşkını atakları uygulanmış, %25'inde ise herhangi bir atağın olmadığı, değişken örüntüler gösteren günlük, olağan trafikler sisteme uygulanmıştır. Geliştirilen sistemin %25'lik ataksız kısımda hiç yanlış alarm üretmediği görülmüştür. Özellikle servis-kontrol modülünün bu sonuca etki ettiği

görülmektedir. Uygulanan 75 saldırı trafiğinin 67'sini sistem başarıyla tespit edebilmiştir. Bu sonuçlar birlikte değerlendirildiğinde geliştirilen sistemin normal ve anormal trafiği tespit etme başarısı %92 (100 veri setinin 92'sinde doğru tespit) olarak bulunmuştur. Sistemin doğruluk hesaplamaları Çizelge 5.2.'de gösterilmektedir.

Çizelge 5.2. Önerilen sistemin doğruluk hesaplamaları

Veri Setleri	Tespit Oranı (%)	Yanlış Tespit (FN veya FP) Oranı (%)	Doğruluk	Hassasiyet	Geri çekme	F1
Anormal Trafik	89	11	-	-	-	-
Normal Trafik	100	0	-	-	-	-
Toplam	92	8	0,92	1	0,89	0,94

Gerçek zamanlılığın testi için veri setleri farklı atak tiplerine göre ayrıştırılmış, her atak tipinde 10 adet veri seti sisteme uygulanmıştır. Saldırıların ortalama tespit süreleri Çizelge 5.3.'de verilmiştir. Görüleceği üzere, tüm ataklar 1 saniyenin altında tespit edilebilmiştir.

Kısa süreli atakların tespiti için süresi kademe kademe düşürrülen ataklar altımda gelitirilen sistemin davranışı test edilmiştir. Testler, 0,7 sn'den uzun süren atakların sistem tarafından sorunsuz olarak saptanabildiğini göstermektedir. 0,7-0,4 sn aralığındaki atakların tutarlı şekilde tespit edilemediği görülmüştür.0,4 sn'nin altındaki ataklar sistem tarafından tespit edilememektedir. Test sonuçları Çizelge 5.4.'te verilmiştir.

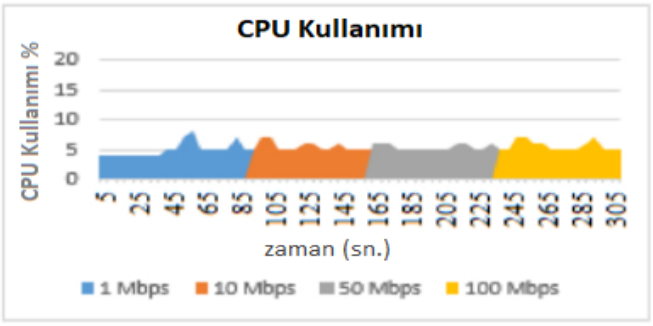
Çizelge 5.3. Atak tespit süreleri

Atak Tipi	Ortalama Atak Tespit Süreleri (sn)
TCP-SYN Seli	0,948
TCP-ACK Seli	0,989
UDP Seli	0,772
HTTP Seli	0,874
ICMP Seli	0,831

Çizelge 5.4. Kısa süreli atak tespiti

Atak Süresi (sn)	Tespit Edilme Durumu
61-300	Başarılı
31-60	Başarılı
1-30	Başarılı
1-0,70	Başarılı
0,69-0,40	Kararsız
0,39-0	Başarısız

Geliştirilen sistemin yarattığı CPU yükünü test etmek için ölçüm verilerinin sürekli olarak iletildiği denetleyici gözlenmişir. 40. sn'ye kadar herhangi bir atak uygulanmamış, CPU yükünün %4 civarlarında olduğu görülmüştür. 40. sn'dem itibaren atak uygulanmış ve bant genişliği kontrollü olarak arttırılmış, ortalama CPU maliyetinin %2 oranında olduğu görülmüştür.



Şekil 5.5.Farklı bant genişliklerinde CPU yükü

Testler esnasında aşağıdaki senaryo ve durumlar incelenmiştir:

1- Normal kullanıcılar aynı anda farklı tipte yoğun trafik oluşturmuş, sistemin yanlış alarm verip vermediği, bu trafikleri engelleyip engellemediği incelenmiştir.

2- Saldırı uygulanmaya başladıktan ne kadar süre sonra sistemin bu saldırıyı tespit edebildiği incelenmiştir. Tüm test bittiğinde bu sürelerin ortalaması hesaplanmıştır.

3- Saldırı başladıktan hemen sonra, sistem saldırı önleme mekanizmasını henüz devreye sokmamışken saldırı kontrollü olarak durdurulmuş, sistemin tepkisi incelenmiştir.

4- Saldırı önlendikten ne kadar süre sonra normal kullanıcıların ağ kaynaklarına kesintisiz erişebildiği incelenmiştir.

5- Sistem öğrenme modundayken farklı karakteristiklere sahip trafikler uygulanmış, koruma moduna geçtiğinde benzer ve farklı karakteristikteki trafikler uygulanarak sistemin davranışı incelenmiştir.

6- Hangi tip atakların geliştirilen sistem tarafından tespit edilebildiği incelenmiştir.
7- Geliştirilen adaptif α sabitinin sistem başarısına etkisi incelenmiştir.

8- Geliştirilen kademeli devreye alma fonksiyonunun sistem başarısına etkisi incelenmiştir.

9- Model tabanlı ölçüm mimarisi yöntemleri test edilerek maliyet, gerçek zamanlılık yönlerinden incelenmiştir.

10- Taşıma başlatma yöntemleri aynı ve farklı trafik örüntüleri için denenmiş, başarıma etkisi olup olmadığı incelenmiştir.

11- Sistem başarımı dış ortama kapalı laboratuvar ortamında ve gerçek ağ trafiğinde olmak üzere her iki şekilde de denenmiştir.

Geliştirilen sistemin bu alanda yapılan diğer çalışmalarla karşılaştırması Çizelge 5.5.'te verilmiştir. Testlerde, aynı ortam ve veri setleri kullanılmıştır.

Çizelge 5.5. Çalışmaların karşılaştırması

	Örneklem ve OpenFlow Tabanlı			Yalnızca OpenFlow		Akış Tabanlı Telemetri Tabanlı		
ÖZELLİKLER	Giotis [6]	FlowTr. [17]	OpenW. [28]	Mehdi [37]	LigthW. [40]	Cheng [45]	DenStr. [49]	Önerilen Sistem
Yanlış Alarm Oranı (FP veya FN)	%39,3	Bilgi Yok	Bilgi Yok	%30	%0,59	%22	%0,4	%11
Yanlış Tespite Karşı Direnç	Yok	Yok	Yok	Yok	Yok	Yok	Var	Var
Atak Tespit Süresi	30 sn.	48 sn.	60 sn.	23 sn.	26 sn.	12 sn.	16 sn.	1 sn.
Atak Önleme Süresi	30 sn.	10 sn.	120 sn.	Bilgi Yok	Yok	20 sn.	30 sn.	10 sn.
Küçük Süreli Atak Tespiti (<30 sn.)	Yok	Var	Yok	Yok	Var	Var	Var	Var
CPU Yükü	%25	%19	%20	%17,54	Bilgi Yok	Bilgi Yok	Bilgi Yok	%2
Doğruluk	%69	%63	Bilgi Yok	%90	%98.61	%79	%99	%92

Akışların tamamını değil de öğrneklem temelli olarak bir kısmını kullandıklarından, örneklem ve OpemFlow tabanlı yaklaşımlarda atak tespit etme doğrıluğunun diğer iki yaklaşıma göre düşük olduğu, ayrıca yanlış atak tespitine karşı herhangi bir koruma mekanizmalarının olmadığı görülmektedir. Ek olarak, diğer çalışmalara göre daha yüksek CPU yükü ve hesaplama maliyeti içerdikleri tespit edilmiştir.

Yalnızca OpenFlow mekanizmasını kullanan yöntemlerin örneklem tabanlı yöntemlere kıyasla daha yüksek doğruluğa sahip oldukları, küçük süreli atakları daha yüksek oranda

tespit edebildikleri görülmüştür. Bu çalışamaların dezavantajları, genelde anormallik tespitinde kullanılmaları, atak önleme için herhangi bir mekanizmalarının bulunmamasıdır.

Ölçeklenebilirlik, birlikte çalışabilme, programlanabilirlik özelliklerinden dolayı akış tabanlı ölçüm yöntemlerinin popülerliği ve kullanım oranları her geçen gün artmaktadır. Bu tür ölçüm yöntemleri tüm ölçüm verilerini kullandıklarından kısa süreli atakların tespiti kolayca mümkün olmaktadır. Model tabanlı mimari ve abone olma temeline dayanan ölçüm mekanizmaları sayesinde diğer sistemlerle birleştirilmeleri oldukça kısa sürmektedir. Bu alanda yapılan model tabanlı ölçüm yöntemleri incelendiğinde, önerilen çözümün atak tespit ve atak önleme süreleri yönünden oldukça gerçek zamanlıya yakın olduğu görülmektedir. Ataklar 1 saniyede tespit edilemikte, tüm kontrol mekanizlmalarndan sonra yaklaşık 10 sn. içinde önlenebilmektedir. Ayrıca, geliştirilen yanlış alarm önleme modülü sayesinde başarımı yüksek yanlış tespit oranı yakalanabilmiştir. Holt'ın tahminleme algoritmasına eklenen geliştirmeler sayesinde doğruluğu yüksek, farklı trafiklere karşı uyum ve esnekliğin fazla olduğu bir sistem sunulmuştur.

6. SONUÇ VE ÖNERİLER

Bu tezde, YTA'larda kullanılmak üzere telemetrik yöntemle gerçek zamanlı anormallik tespiti ve önleme yapan bir ortam geliştirilmiştir. Sistem üzerinde kullanılan tüm fonksiyon ve araçlar YTA mimarisine uygun olarak sanallaştırılmış, sanal ortamda yapılan tüm geliştirmeler aynı zamanda gerçek ağ ortamlarına da uygulanabilmiştir. Hem sanal ortamda hem de gerçek ağ altyapısında sistemin hedeflenen başarım artışlarını sağladığı görülmüştür.

Bu tez çalışması, kapsam olarak ağ ölçümü, güvenlik ve YTA olmak üzere 3 disiplini temel almaktadır. Çalışmalarımız ağ ölçümü alanında detaylı alan taraması ile başlamış, gerçek zamanlılık, doğruluk, düşük maliyetlilik, küçük hacimli/kısa süreli trafiklerin saptanabilirliği, programlanabilirlik, ölçeklenebilirlik gibi gelişim alanları belirlenmiştir. YTA'nın sunduğu faydalar bu kapsamda incelenmiş, YTA tabanlı olan izleme/ölçüm yöntemleri için alan taraması yinelenmiştir. Ölçüm yönteminde hedeflenen geliştirmelerin faydasını daha net görebilmek ve gerçek hayat sorunlarına çözüm sunmak amacıyla uygulama alanı olarak anormallik ve saldırı tespiti problemi ele alınmış, bu kapsamda yapılan çalışmalar incelenerek kullanılan algoritma ve yöntemlerdeki sorunlar ve gelişim alanları belirlenmiştir. Geliştirilecek ölçüm yöntemi ve anormallik tespit yaklaşımı belirlendikten sonra saldırı önleme için kullanılacak yaklaşımlar incelenmiş, saldırgan trafiğini kısıtlama veya ağdan tamamen arındırma gibi yöntemlerin istenen sonucu verdiği saptanmıştır.

Ağ ölçümü kapsamında belirlenen gelişim alanları neticesinde ölçüm metodunda model tabanlı mimarinin uygulanabilirliği ve olası faydaları detaylıca irdelenmiştir. Araştırmada, ölçüm metriklerinin akışına model tabanlı mimari ile getirilen modelleme, serileştirme, taşıma ve bu verilerin kullanacak ortam için hazır halde sunulması özellikleri sayesinde ağ cihazlarının çok yüksek boyut ve miktarlardaki ölçüm ve trafik bilgileri cihaz dışına gecikmesiz ve çoklu programlanabilirlik özellikleriyle çıkarılabileceği görülmüştür. Model tabanlı akış ölçümü mimarisinde, Open, Native ve IETF gibi YANG veri modelleri ile modellenen, ikili ve metin yapısında olabilen XML, JSON ve GPB gibi serileştirme yöntemleri ile biçimlendirilen veriler NETCONF, RESTCONF ve gRPC gibi taşıma protokolleri ile hızlı ve güvenli şekilde taşınmaktadır. Yüzlerce sistem ve servisin çalıştığı geniş ölçekli ağ ortamlarında verilerin bu yöntemlerle tekilleştirilmesinin ve

yapısallaştırılmasının başta saptanan başarım, programlanabilirlik, ölçeklenebilirlik gibi ihtiyaçlara faydalar getireceği belirlenmiş ve bu kapsamda özelleştirilmiş toplayıcı tasarlanmıştır.

Anormallik tespiti kapsamında belirlenen gelişim alanları neticesinde, model tabanlı sürekli ölçüm yaklaşımıyla cihazlardan toplayıcı yardımıyla toplanan ve zaman serisi şeklinde depolanan veriler, Python betik dilinde kodlanmış özel bir anormallik tespit algoritmasıyla gerçek zamanlı olarak analiz edilmiştir. Eğilim yapısının zaman içinde rasgele değişebilmesi, başarımın büyük veri ile doğru orantılı olması, hem deterministik hem stokastik eğilimli seriler için başarılı sonuçlar üretebilmesi, verideki değişime duyarlılığın ayarlanabilmesi gibi özellikleri sebebiyle bu çalışmada zaman serisi analizlerinde en önemli yöntemlerden biri olan Holt Üstel Düzleştirme Yöntemi geliştirilerek kullanılmıştır. Çalışmada uygulamaya özel ataklar kapsam dışı tutulmuş, saldırı tipi olarak yüksek hacimli D/DoS atakları baz alınmıştır.

Anormallik önleme kapsamında 2 temel politika belirlenmiştir: i) bant genişliği sınırlama ii) izolasyon. Bu iki politika ayrı ayrı uygulanabileceği gibi her ikisi sıralı olarak ard arda da uygulanabilmektedir. Bant genişliği sınırlama; saldırgan IP adreslerinin bir süre boyunca bant genişliğinin ağ servislerinin çalışmasına engel olmayacak bir değere indirilmesi şeklinde, izolasyon ise saldırgan adreslerin ağa erişimin tamamen kesilmesi şeklinde uygulanmaktadır.

Bu tez çalışmasındaki tüm uygulama bileşenleri YTA mimarisine uygun olarak sanallaştırılmış yazılımlardan oluşmakta, fiziksel bir ağ bileşenine ihtiyaç duyulmamaktadır. Bununla birlikte oluşturulan sistem gerçek zaman ortamıyla birleştirilmiş, gerçek ortam trafiklerine uygulanmıştır.

Ağ cihazları ve son kullanıcıların kullandığı tüm işletim sistemleri VirtualBox ortamında sanallaştırılmış ve GNS3 üzerinde oluşturulan bir yapı birleştirilmiştir. Ölçüm verilerinin model tabanlı ve gerçek zamanlı olarak toplayıcıya gönderimi için telemetri destekli ağ cihazı olarak Cisco firmasının lisanslı ağ işletim sistemi olan IOS XRv yazılımı kullanılmıştır. Toplayıcı-ZSVT-İşleme birimlerinin tamamı "Platform" adı verilen, Linux tabanlı bir işletim sistemi üzerinde çalıştırılmıştır. Toplayıcı olarak Pipeline Ağ Telemetri projesi, ZSVT için InfluxDB ve verilerin işlenmesi için kodlama ortamı olarak Python dili

kullanılmıştır. Son olarak ağ servisini örneklemek amacıyla üzerinde DNS ve HTTP servisleri çalışan bir Windows sunucu ağa sanal olarak dahil edilmiş, saldırganın bu sunucuya saldırı yapması yoluyla geliştirilen sistemin başarımı test edilmiştir.

Birçok farklı saldırı senaryosu ve ağ davranışı kontrollü olarak denenmiş, geliştirilen ölçüm yönteminin ve anormallik tespit sisteminin gerçek zamanlılık, doğruluk, düşük maliyetlilik, küçük hacimli/kısa süreli trafiklerin saptanabilirliği, programlanabilirlik, ölçeklenebilirlik gibi yönlerden başarılı sonuçlar ürettiği gözlemlenmiştir.

Bu çalışmada, protokol bağımsız, yüksek hacimli ataklar dikkate alınmıştır. Sonraki çalışmalarda, yüksek hacimli olmayan fakat uygulama özelinde geliştirilmiş ataklar incelenebilir. Ayrıca, YTA kavramı kontrol ve veri düzlemlerini birbirinden ayrıştırdığı için ve merkezileştirdiği için kontrol düzlemi ataklara açık hale gelmektedir. Bu sebeple denetleyicinin ataklardan korunmasına yönelik çalışmalar yapılabilir. Bunlara ek olarak, koruma yeteneklerini ve doğruluğu arttırmak amacıyla yapay zekâ temelli, makine öğrenmesi gibi algoritmalar kullanılabilir.

KAYNAKLAR

1. Wickboldt, J. A., De Jesus, W. P., Isolani, P. H., Both, C. B., Rochol, J. (2015). Software-defined networking: management requirements and challenges. *IEEE Communications Magazine*, 53(1), 278-285.

2. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Petrson, L., Rexford, J., Shenker, S., Turner, J. (2008). OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication*, 38(2), 69-74.

3. Case, J., Fedor, M., Schoffstall, M., Davin, J. (1990). Simple Network Management Protocol (SNMP). *IETF*, RFC 1157.

4. İnternet : Ten Lessons From TelemetryCisco Systems, URL:https://www.nanog.org/sites/default/Lessons.pdf, Son Erişim Tarihi : 01.11.2019.

5. Phaal, P., Panchen, S., McKee, N. (2001). InMon corporation's sFlow: A method for monitoring traffic in switched and routed networks. *IETF*, RFC 3176.

6. Giotis, K., Androulidakis, G., Maglaris V. (2014). Leveraging SDN for efficient anomaly detection and mitigation on legacy networks. *3. European Workshop on Software Defined Networks Dergisi*, 3, 85-90.

7. Huang, L., Zhi, X., Gao, Q., Kausar, S., Zheng, S. (2016*). Design and implementation of multicast routing system over SDN and sFlow.* 8. IEEE International Conference on Communication Software and Networks (ICCSN*)*, 8, 524-529.

8. Liu, G. P., Zuo, W., Li, C. L., OuYang, Z. Y. (2007). Communication network traffic anomaly monitor solution on sflow. *Journal of Computer Engineering*, 33(6), 61-63.

9. Zhui, H. X., Chen, H. C. (2010). Research on sflow technology in network traffic data monitoring. *Computer & Digital Engineering*, 38, 110-113.

10. Guo, F. H., Liu, S. F. (2015). Distributed network traffic analysis system based on sflow. *Journal of Science University*, 53(5), 987-990.

11. Han, C. Q. (2013). Research on network security situation fusion analysis method based on sflow and SNMP. *Harbin: Habin Egineering University*, 9-17.

12. Liu, B. (2009). Study on sampling stream based network behavior analysis. *China Computer and Comunication*, 8, 47-48.

13. Zhang, H. L., Wang, H.(2007). A method of network traffic analysis based on sflow. *Computer Engineering and Science*, 29(8), 61-63.

14. Yu, C., Lumezanu, C., Zhang, Y., Singh, V., Jiang, G., Madhyastha, H. V. (2013). *FlowSense: Monitoring network utilization with zero measurement cost.* In Proceedings of the 14th IPAM, 14, 31– 41.

15. Chowdhury, S., Bari, M, Ahmed, R., Boutaba,R. (2014). *PayLess: A low cost network monitoring framework for software defined networks.* In Proceedings of the 14th IEEE/IFIP NOMS, 14, 1–9.

16. Adrichem, N. L. M., Doerr, Kuipers, C. F. A. (2014). OpenNetMon: Network monitoring in OpenFlow software-defined networks. *IEEE NOMS*, 14, 1-8.

17. Buragohain C, Medhi N. (2016). *FlowTrApp: An SDN based architecture for DDoS attack detection and mitigation in data centers.* In Proceedings of the 3rd International Conference on Signal Processing and Integrated Networks, 13, 519-524.

18. Shirali, S. Ganjali, Y. (2013). *Efficient implementation of security applications in OpenFlow controller with FleXam.* 2013 IEEE 21st Annual Symposium on High-Performance Interconnects, 13, 49-54.

19. Spence, C., Parra, L., Sajda, P. (2001). *Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model.* In Proceedings of the IEEE Workshop on Mathematical Methods in Biomedical Image Analysis IEEE Computer Society, 3, 20-24

20. Kumar, V. (2005). Parallel and distributed computing for cybersecurity. *IEEE Distributed Systems*, 6, 10-19.

21. Kreutz, D., Ramos, F. M. V., Verissimo, P. (2013). *Towards secure and dependable software-defined networks.* In Proceedings of the 2nd ACM SIG-COMM Workshop Hot Topics in Software Defined Networks. HotSDN'13, 13, 55-60.

22. Ballard, J. R., Rae, I., Akella, A. (2010). Extensible and scalable network monitoring using OpenSafe. *In Proceedings of INM/WREN*, 10, 8-19.

23. Adrichem, N. L., Doerr, C., Kuipers, F. A. (2014). OpenNetMon: Network monitoring in OpenFlow software-defined networks. *In Proceedings of IEEE NOMS*, 14, 1–8.

24. Sterbenz, J. P. (2010). Resilience in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks Journal*, 54(8), 1245–1265

25. Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2), 222–232.

26. Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., Maglaris, V. (2014). Combining OpenFlow and sFlow for an effective and scalable anomaly detection on SDN environments, *Computer Networks Journal*, 62, 122-136.

27. Lakhina, A., Crovella, M., Diot, C. (2005). Mining anomaly using traffic feature distributions. *ACM SIGCOMM*, 217–228.

28. Zhang, Y. (2013). *An adaptive flow counting method for anomaly detection in SDN*. In Proceedings of the 9th ACM CoNEXT, 4, 25-30.

29. Sheen, S., Rajesh, R. (2008). Network intrusion detection using feature selection and Decision tree classifier. *TENCON*, 10, 1-4.

30. Tarek, A., Adel, B., Michael, R. (2004). *Protocol analysis in intrusion detection using decision tree*. In Proceedings of the International Conference on Information Technology: Coding and Computing, 2, 20-24.

31. Shun, J.M., Hanet, A. (2008). *Network intrusion detection system using neural networks*. ICNC '08. The Fourth International Conference, 5, 242-246.

32. Jiu-Ling Z., Jiu-Fen Z., Jian-Jun L. (2005). Intrusion detection based on clustering genetic algorithm. *Journal of Machine Learning and Cybernetics*, 6, 3911- 3914.

33. Vasilis, M., Kostas, G., Androulidakis, G. (2014). *Leveraging SDN for efficient anomaly detection and mitigation on legacy networks*. in Proceedings of the 3rd European Workshop on Software-Defined Networks, 3, 10-24.

34. Liu, H., Sun, Y., Kim, M. S. (2011). A scalable DDoS detection framework with victim pinpoint capability, *Journal of Communications*, 6(9), 660-670.

35. Jankowski, D., Amanowicz, M. (2015). Intrusion detection in software defined networks with self-organized maps, *in Proceedings of IEEE*, 81, 3-9

36. Kohonen, T. (1990). The self-organizing maps. *IEEE*, 78(9), 1464–1480.

37. Mehdi, S. A., Khalid, J., Khayam, S. A. (2011). Revisiting traffic anomaly detection using software defined networking. *Springer-Verlag*, 161–180.

38. Schechter, S. E., Jung, J., Berger, A. W. (2004). *Fast detection of scanning worm infections*. In Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection, 44-48.

39. Mahoney, M.V. (2003). *Network traffic anomaly detection based on packet bytes*. In Proceedings of the 2003 ACM symposium on Applied computing. 346-350.

40. Braga, R., Mota, E., Passito, A. (2010). *Lightweight DDoS flooding attack detection using NOX/OpenFlow*. In Proeedings of the 35th Ann. IEEE Conference Local Computer Networks, *35*, 408-415.

41. Sahri, N.M., Okamura, K. (2015). *Adaptive anomaly detection for SDN*. In Proceedings of the Asia-Pacific Advanced Network, 40, 57-63

42. Ozcelik, Ilker, Fu, Y. Brooks, R. R. (2013). *DoS detection is easier now*. GENI Research and Educational Experiment Workshop, 2(01), 50-55.

43. Rabby. B. (2001). *A novel approach to detection of denial-of-service attacks via adaptive sequential and batch-sequential change-point detection methods.* IEEE Workshop Information Assurance and Security, 220-226

44. Chen, Y. Hwang, K. (2006). *Collaborative change detection of ddos attacks on community and ISP networks*. In IEEE International Symposium on Collaborative Technologies and Systems, *6*, 401–410.

45. Cheng, J., Xu, R., Tang, X., Sheng, V., Cai, C. (2018). An abnormal network flow feature sequence prediction approach for DDoS attacks detection in big data environment. *Computers, Materials and Continua*, 55(1), 95-119.

46. Carl, G., Brooks, R. R., Rai, S. (2006). Wavelet based denial of service detection. *Computers and Security*, 25(8), 600-615.

47. Callegari, C., Giordano, S., Pagano, M., Pepe, T. (2012). Wavecusum: Improving cusum performance in network anomaly detection by means of wavelet analysis. *Computers Camp Security*, 31(5), 727-735

48. Jin, S., Yeung, D. (2004). *A covariance analysis model for ddos attack detection*. In Communications, 2004 IEEE International Conference, 4(4), 1882-1886.

49. Putina, A., Rossi, D., Bifet, A., Barth, S., Pletcher, D. (2018). *Telemetry-based stream-learning of BGP anomalies.* In Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks (Big-DAMA '18) ACM; 26, 15-20.

50. Nychis, G., Sekar, V., Andersen, D. G., Kim, H., Zhang, H. (2008). *An empirical evaluation of entropy-based traffic anomaly detection*. In Proceedings of the 8th ACM SIGCOMM conference on Internet measurement, 151–156.

51. Chen, Y., Hwang, K., Ku, W. S. (2007), Collaborative detection of ddos attacks over multiple network domains. *IEEE Transactions on Parallel and Distributed Systems*, 18, 1649–1662.

52. Lee, K., Kim, J., Kwon, K. H., Han, Y., Kim, S. (2008), DDoS attack detection method using cluster analysis. *Expert Systems with Applications*, 34(3), 1659-1665.

53. İnternet : NASA Telemetry Report, URL:https://spinoff.nasa.gov/spinoff1997/ct12.html, Son Erişim Tarihi : 01.11.2019.

54. İnternet : NASA Telemetry Report, URL:https://spinoff.nasa.gov/spinoff1996/53.html, Son Erişim Tarihi : 01.11.2019.

55. İnternet : YANG Veri Modelleme Dili, RFC 6020, URL:https://tools.ietf.org/html/rfc6020, Son Erişim Tarihi : 01.11.2019.

56. İnternet : A YANG Data Model for Interface Management, RFC 7223, URL:https://tools.ietf.org/html/rfc7223, Son Erişim Tarihi : 01.11.2019.

57. İnternet : A YANG Data Model for System Management, RFC 7317, URL:https://tools.ietf.org/html/rfc7317, Son Erişim Tarihi : 01.11.2019.

58. İnternet : A YANG Data Model for SNMP Configuration, RFC 7407, URL:https://tools.ietf.org/html/rfc7407, Son Erişim Tarihi : 01.11.2019.

59. İnternet : Google Protocol Buffers, URL:https://developers.google.com/protocol-buffers/, Son Erişim Tarihi : 01.11.2019.

60. İnternet : GPB, URL:https://blogs.cisco.com/sp/streaming-telemetry-with-google-protocol-buffers, Son Erişim Tarihi : 01.11.2019.

61. İnternet : NETCONF, Network Configuration Protocol, RFC 4741, URL:https://tools.ietf.org/html/rfc4741, Son Erişim Tarihi : 01.11.2019.

62. İnternet : RESTCONF, Restconf Protocol, RFC 8040, URL:https://tools.ietf.org/html/rfc8040, Son Erişim Tarihi : 01.11.2019.

63. İnternet : gRPC, Introducing gRPC, a new open source HTTP/2 RPC Framework, URL:https://developers.googleblog.com/2015/02/introducing-grpc-new-open-source-http2.html, Son Erişim Tarihi : 01.11.2019.

64. İnternet : Apachi Kafka, URL:https://kafka.apache.org/, Son Erişim Tarihi : 01.11.2019.

65. İnternet : ActiveMQ, URL:http://activemq.apache.org/, Son Erişim Tarihi : 01.11.2019.

66. İnternet : RabbitMQ, URL:https://www.rabbitmq.com/, Son Erişim Tarihi : 01.11.2019.

67. İnternet : Pipeline, URL:https://github.com/cisco/, Son Erişim Tarihi : 01.11.2019.

68. İnternet : Logstash, URL:https://www.elastic.com/logst, Son Erişim Tarihi : 01.11.2019.

69. İnternet : Open-NTI, URL:http://open-nti.io/, Son Erişim Tarihi : 01.11.2019.

70. İnternet : sFlow, URL:https://sflow.org/collectors.php/, Son Erişim Tarihi : 01.11.2019.

71. İnternet : Telegraf, URL:https://www.influxdata.com, Son Erişim Tarihi : 01.11.2019.

72. İnternet : Prometheus, URL:https://prometheus.io, Son Erişim Tarihi : 01.11.2019.

73. İnternet : Elasticsearch, URL:https://www.elastic.com, Son Erişim Tarihi : 01.11.2019.

74. İnternet : Elasticsearch dosyalama yapısı sebebiyle gecikme yaşanması, URL:https://www.elastic.co/guide/en/elasticsearch/reference/current/_basic_concepts.html, Son Erişim Tarihi : 01.11.2019.

75. İnternet : InfluxDB, URL:www.influxdata.com/, Son Erişim Tarihi : 01.11.2019.

76. İnternet : Kapacitor, URL:www.influxdata.com, Son Erişim Tarihi : 01.11.2019.

77. İnternet : OpenTSDB, URL:http://opentsdb.net/, Son Erişim Tarihi : 01.11.2019.

78. İnternet : Zaman serisi veri tabanlarınıın karşılaştırma tablosu, URL:https://docs.google.com/spreadsheets/d/1sMQe9oOKhMhIVw9WmuCEWdPtAoccJ4a-IuZv4fXDHxM/edit#gid=0, Son Erişim Tarihi : 01.11.2019.

79. İnternet : Grafana, URL:https://grafana.com/, Son Erişim Tarihi : 01.11.2019.

80. İnternet : Kibana, URL:https://www.elastic.co/kibana/, Son Erişim Tarihi : 01.11.2019.

81. İnternet : Graphite, URL:https://graphiteapp.org/, Son Erişim Tarihi : 01.11.2019.

82. İnternet : Chronograf, URL:www.influxdata.com/, Son Erişim Tarihi : 01.11.2019.

83. İnternet : Vagrant, URL:https://www.vagrantup.com/, Son Erişim Tarihi : 01.11.2019.

84. İnternet : VIRL, URL:http://virl.cisco.com/, Son Erişim Tarihi : 01.11.2019.

85. İnternet : GNS3, URL:https://www.gns3.com/, Son Erişim Tarihi : 01.11.2019.

86. Mininet, B., Lantz, B. H., McKeown, N. (2010). A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. ACM Hotnets, 20–21.

87. İnternet : Pyang, URL:https://github.com/mbj46/pyang, Son Erişim Tarihi : 01.11.2019.

88. Asteriou, D., Hall, S. (2011), *ARIMA Model and the Box-Jenkins Methodology*, Applied Econometrics, 265-286.

89. Lindley, D.V. (1987). *Regression and correlation analysis,* New Palgrave: A Dictionary of Economics, 4, 120-23.

90. McCulloch, W. S., & Pitts, W. A. (1943). *A logical calculus of the ideas immanent in nervous activity,* Buttetin of Mathematics and Biophysics, 5, 115-133.

91. Holt, C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages*, International Journal of Forecasting*, 20 (1), 5-10.

92. Winters, P. (1960). Forecasting sales by exponentially weighted moving averages, *Management Science*, 6, 324–342.

93. Lamon, E. C., Carpenter, S. R, Stow, C.A. (1998). *Forecasting Pcb Concentrations in Lake Michigan Salmonids: A Dynamic Linear Model Approach*, Ecological Applications, 8, 659–68.

94. Molugaram, K., Rao S. (2017). *Statistical Techniques for Transportation Engineering*. Butterworth-Heinemann Elsevier Yayınevi, 463-489.

95. Dobesova, Z. (2011). *Programming language Python for data processing*. International Conference on Electrical and Control Engineering, 4866-4869.

96. İnternet : VirtualBox, URL:www.virtualbox.org/, Son Erişim Tarihi : 01.11.2019.

97. İnternet : GNS3, URL:https://www.gns3.com/, Son Erişim Tarihi : 01.11.2019.

98. İnternet : Scapy, URL: www.github.com/secdev/scapy , Son Erişim Tarihi : 01.11.2019.

99. İnternet : Tcpreplay, URL: www.github.com/appneta , Son Erişim Tarihi : 01.11.2019.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: KURT, Çağdaş
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 06.07.1988, İzmir
Medeni hali	: Evli
Telefon	: 0 (555) 712 59 90
e-mail	: cagdaskurt@gmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Doktora	Gazi Üniversitesi / Bilgisayar Mühendisliği	Devam Ediyor
Yüksek lisans	Gazi Üniversitesi / Bilgisayar Bilimleri	2013
Lisans	Gazi Üniversitesi / Bilgisayar Sistemleri	2011
Lise	İzmir Cumhuriyet N.S.İ. A.M.L.	2006

İş Deneyimi

Yıl	Yer	Görev
2019 - Halen	ING Bank Nederland	DC & Netw. Dev-Ops Müh.
2016 - 2019	Turkcell	Kurumsal Çözümler Yön.
2013 - 2016	Oyak Teknoloji	Ağ Çözümleri Danışmanı
2011 - 2013	Türksat	Omurga Ağ Mühendisi

Yabancı Dil

İngilizce

Yayınlar

1. Kurt, Ç., Erdem O.A. (2012). Öğrenci Başarısını Etkileyen Faktörlerin Veri Madenciliği Yöntemleriyle İncelenmesi, *Politeknik Dergisi*, 15(2), 111-116.
2. Kurt, Ç., Özdemir, S. (2013). User Identity Protection in Wireless Local Area Networks, *Gazi University Journal of Science*, 26(4), 563-570.
3. Kurt, Ç., Erdem O.A. (2019). Real-Time Anomaly Detection and Mitigation using Streaming Telemetry in SDN, *Turk J Elec Eng & Comp Sci*, https://doi.org/10.3906/elk-1909-112

Hobiler

Kültür gezileri, Ortaçağ Avrupa Mimarisi, Basketbol

GAZİ GELECEKTİR...