T.C.
GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK
LİSANS
TEZİ

YAZILIM TANIMLI AĞLARDA HİZMET DIŞI BIRAKMA SALDIRILARININ TESPİTİ VE ENGELLENMESİ

NAİL GÖKSEL

BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

TEMMUZ 2019


GAZİ ÜNİVERSİTESİ
Gazi
1926

YAZILIM TANIMLI AĞLARDA HİZMET DIŞI BIRAKMA SALDIRILARININ TESPİTİ VE ENGELLENMESİ

Nail GÖKSEL

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEMMUZ 2019

Nail GÖKSEL tarafından hazırlanan "YAZILIM TANIMLI AĞLARDA HİZMET DIŞI BIRAKMA SALDIRILARININ TESPİTİ VE ENGELLENMESİ" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Bilgisayar Mühendisliği Ana Bilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Dr. Öğr. Üyesi Mehmet DEMİRCİ

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. .....

Başkan: Doç. Dr. Hacer KARACAN

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. .....

Üye: Dr. Öğr. Üyesi Pelin ANGIN

Bilgisayar Mühendisliği Ana Bilim Dalı, Orta Doğu Teknik Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. .....

Tez Savunma Tarihi: 31/07/2019

Jüri tarafından kabul edilen bu tezin Yüksek Lisans 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.

Nail GÖKSEL

31/07/2019

YAZILIM TANIMLI AĞLARDA HİZMET DIŞI BIRAKMA SALDIRILARININ TESPİTİ VE ENGELLENMESİ

(Yüksek Lisans Tezi)

Nail GÖKSEL

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Temmuz 2019

ÖZET

Yazılım tanımlı ağlarda (YTA) kontrolcüyü hedef alan hizmet dışı bırakma (Denial of Service-DoS) saldırıları, kontrolcünün önemi nedeniyle oldukça tehlikelidir. Bu tezde, YTA'daki DoS saldırılarının etkileri karakterize edilmiş ve olası önlemler tartışılmıştır. DoS saldırılarının kontrolcü tarafındaki etkilerine odaklanılmış ve bir simülasyon ortamında packet_in mesajı sayısının değişimi üzerinde durulmuştur. Sonuçlara göre düğümleri sadece packet_in mesajı sayısı temelinde birbirinden ayırmak saldırganları tespit etmede yanıltıcı olabilmektedir. Bunun yerine, saldırganları normal kullanıcılardan ayırmak için packet_in sayısının iletilen paket sayısına oranının kullanılmasının daha iyi bir yöntem olduğu görülmüştür. Buna ek olarak farklı saldırgan sayılarındaki adalet durumu da ölçülmüştür. Sonuçlar, Jain's Index'in benzetim ortamında anomaliyi tespit etmek için entropiden daha iyi bir yöntem olduğunu göstermiştir. Ayrıca, Jain's Index'i temel alan bir DoS önleme sistemi geliştirilmiştir. Önerilen önleme altyapısı ile saldırganların iyi huylu düğümlere oranının %10'a kadar olduğu durumlarda saldırganlar tespit edilebilmiştir. Devamında ise, önleme altyapısının gerçek ağ verilerinin bulunduğu bir ortamdaki etkinliği gösterilmiştir. Sonuçlarda önleme altyapısının veri iletiminde %52'ye varan oranda iyileşme sağladığı görülmüştür.

Bilim Kodu : 92407
Anahtar Kelimeler : Yazılım tanımlı ağlar, hizmet dışı bırakma, adil dağıtım
Sayfa Adedi : 75
Danışman : Dr. Öğr. Üyesi Mehmet DEMİRCİ

DETECTION AND PREVENTION OF DENIAL OF SERVICE ATTACKS IN SOFTWARE DEFINED NETWORKS

(M. Sc. Thesis)

Nail GÖKSEL

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

July 2019

ABSTRACT

Denial-of-service (DoS) attacks targeting the controller in software-defined networks (SDN) are dangerous due to the importance of the controller. In this thesis, we characterize the effects of DoS attacks in SDN and discuss potential countermeasures. We concentrate on the controller-side effects of DoS attacks and present our experimental results on how packet_in message counts change in a simulation scenario. Our results imply that differentiating hosts based on only packet_in counts may be misleading for detecting attackers. Instead, packet_in to transmitted packet count ratio is better for distinguishing attackers from normal users. In addition, we measure fairness values with different attacker counts. Our results show that Jain's index is better than entropy in terms of detecting anomaly in our simulation environment. We also developed a DoS prevention system based on Jain's Index. According to our prevention scheme's results, we were able to detect attacker hosts in an environment that attacker to benign host ratio is up to 10%. We then proposed our prevention scheme's results and effectiveness in an environment with real network data. Our results show that our prevention scheme led up to a 52% of recovery in transmitted data.

Science Code : 92407
Key Words : Software-defined networks, denial-of-service, fairness
Page Number : 75
Supervisor : Asst. Prof. Dr. Mehmet DEMİRCİ

TEŞEKKÜR

Bu tez çalışmasında, beni çalışmalarımı tamamlamam konusunda yüreklendiren Bilgisayar Mühendisliği Bölüm Başkanı Prof.Dr. Şeref SAĞIROĞLU'na, desteklerini esirgemeyen tez danışmanım Dr.Öğr.Üyesi Mehmet DEMİRCİ'ye, yoğun çalışmalarım esnasında bana sabır gösteren eşim Sezen ERGÜL GÖKSEL'e ve sevgili kızıma teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

RESİMLERİN LİSTESİ

Resim **Sayfa**

SİMGELER VE KISALTMALAR

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

Simgeler	**Açıklamalar**
MB	Mega byte
mbps	Mega bit per second
pps	Packets per second
RTT	Round trip time

Kısaltmalar	**Açıklamalar**
BFS	Breadth first search
CMT	Concurrent multipath transfer
DDoS	Distributed denial of service
DNS	Domain name sevice
DoS	Denial of service
FTP	File transfer protocol
HTTP	Hyper text transfer protocol
ICMP	Internet control message protocol
IDS	Intrusion detection system
IP	Internet protocol
MAC	Media access control
OSI	Open systems interconnection
SCTP	Stream control transmission protocol
SDN	Software defined networks
SMTP	Simple mail transfer protocol
TCP	Transmission control protocol
TLS	Transport layer security
UDP	User datagram protocol
YTA	Yazılım tanımlı ağlar
VANET	Vehicular ad-hoc networks

1. GİRİŞ

Yazılım tanımlı ağlar (YTA), merkezi yönetimi nedeniyle ağ yöneticilerinin yükünü hafifletecek ve ağ sistemlerini daha kararlı hale getirecek olmaları nedeniyle son yıllarda birçok araştırmada ilgi odağı haline gelmiştir. Her bir ağ cihazının ağ topolojisini keşfetmesini ya da ağ yöneticilerinin cihazlara konfigürasyon bilgisi girmesini mecbur bırakmak yerine YTA sistemi, ağ davranışını yönetme sorumluluğunu merkezi bir kontrolcüye vererek veri ve kontrol katmanlarını birbirinden ayırmak üzere inşa edilmiştir.

YTA, bütün ağın tek bir noktadan izlenmesi ve yönetilebilmesini sağlaması nedeniyle önemli avantajlar sağlamaktadır. Programlanabilir yapısı geliştirme ve yeniliklerin daha kolay biçimde uygulanmasına olanak tanır. Açık kaynak altyapısı sebebiyle, üretici bazlı yazılımların bulunduğu ortamlara nazaran uygulama kolaylığı ve anlaşılabilirlik sağlar [1]. YTA mimarisi yeni bir yöntem olarak üzerinde geliştirmeler yapmaya açık bir alandır. Bunun yanında her yeni yöntemin getirdiği yenilikler kadar üzerinde henüz çalışılmamış eksikliklerin ve güvenlik zafiyetlerinin olması da oldukça yüksek bir ihtimaldir. Bu tez çalışmasında YTA mimarisi, belirli noktalarda zafiyeti olduğu görülen güvenlik boyutuyla ele alınmıştır.

YTA mimarisinde yönetim merkezinin tek veya bir kaç dağıtık noktada olması, avantajlarının yanısıra güvenlik sorunlarını da beraberinde getirmektedir. Bu güvenlik sorunlarının en önemlilerinden biri ise Hizmet Dışı Bırakma (Denial of Service - DoS) saldırılarıdır. Merkezi kontrolcünün devre dışı kalmasıyla tüm ağ yapısının devre dışı bırakılma ihtimali saldırganların büyük oranda ilgisini çekmektedir. YTA, bugüne kadar kullanılan ağ altyapılarıyla birlikte çalışma özelliğine sahiptir ve mevcut ağ cihazlarıyla entegre edilebilme özellikleri mevcuttur. Bu nedenle mevcut DoS saldırılarının birçoğu, geleneksel mimaride uygulandığı biçimiyle YTA'da da uygulanabilmektedir. Ancak etkileri, geleneksel ağ altyapılarına nazaran çok daha tehlikeli boyutlara ulaşabilecek niteliktedir.

YTA temelindeki DoS saldırılarında saldırılar; veri katmanı (anahtar), kontrolcü, uygulama veya aradaki bağlantı katmanını hedef alma durumlarına göre sınıflandırılmaktadırlar. Bazı durumlarda saldırılar bu birimlerden birine yönelik iken, bazı durumlarda ise birçok birimi

birden etkileyen saldırılar yapılabilmektedir. Bu tez çalışmasında DoS saldırılarının YTA özelinde incelenmesi amaçlanmış, bu nedenle YTA bileşenlerinin kullanıldığı DoS saldırıları üzerine odaklanılmıştır.

Literatürde YTA üzerinde uygulanan DoS saldırıları incelendiğinde, bu saldırıların amacının genel olarak, akış tablolarının oluşturulması esnasında kontrolcü ve anahtarlar arasında meydana gelen iletişimden faydalanmak olduğu gözlenmiştir. Bu iletişim ise günümüzde büyük oranda OpenFlow protokolü kullanılarak gerçekleştirilmektedir [2]. Bu nedenle, bu tez çalışmasında, OpenFlow protokolünde anahtarların herhangi bir pakete ait akış bilgisine kendi akış tablolarında ulaşamadıklarında yönlendirme bilgisini kontrolcüye sordukları packet_in mesajlarından faydalanılarak gerçekleştirilen DoS saldırıları üzerine odaklanılmıştır.

YTA üzerindeki DoS saldırılarına ilişkin savunma yöntemleri son yıllarda aktif bir araştırma alanı konumundadır. Bu saldırılara ilişkin bazı önlemler önceki ağ altyapılarında kullanılan geleneksel tespit mekanizmalarını kullanırken, bazı yöntemler ise saldırıları tespit etmek veya önlemek için YTA altyapısının karakteristik özelliklerini kullanmaktadır.

YTA özelinde DoS saldırılarına yönelik birtakım ilerlemeler kaydedilmiş olsa da, saldırganların devre dışı bırakılması için bir belirteç olarak kabul edilen eşik değerlerinin nasıl belirlendiği ve yoğun ağ trafiği altında packet_in mesajlarındaki dağılımın nasıl gerçekleştiği gibi noktalarda birtakım eksiklikler görülmüştür. Bu tez çalışmasında, literatürde yukarıda bahsedilen eksikliklerin giderilmesi amacıyla, YTA'daki DoS saldırılarının packet_in mesajları üzerindeki etkileri detaylı olarak incelenmiş ve geliştirilen bir çözüm yöntemi sunulmuştur.

Bu tez çalışması şu şekilde düzenlenmiştir: 2. Bölümde YTA ve OpenFlow protokolü genel çerçevede ele alınmış ve geleneksel ağ alt yapılarından farkları ve getirdikleri faydalar özetlenmiştir. 3. Bölümde DoS saldırılarına yönelik literatürde gerçekleştirilen çalışmalar özetlenmiş, DoS saldırılarına karşı geliştirilen yöntemler özetlenip karşılaştırılarak eksiklikler ve katkı yapılabilecek noktalar belirlenmiştir. 4. Bölümde DoS saldırılarının YTA üzerindeki etkileri ele alınarak örneklerle gösterilmiş ve buna karşı olarak geliştirilen yöntemin hangi temeli kapsadığı belirtilmiştir. Ayrıca geliştirilen yönteme çerçeve teşkil edecek algoritma ve tasarlanan sistemin bileşenleri ortaya

konulmuştur. 5. Bölümde ise deneysel sonuçlar gösterilmiştir. Bu kapsamda üzerinde çalışılan benzetim ortamı, ağ altyapısının yapısı, düğümler arasındaki ilişkiler özetlenmiş ve geliştirilen tespit ve önleme sistemine ait sonuçlar paylaşılmıştır. Sonuç ve Öneriler bölümünde ise tez kapsamında yapılan çalışmalar özetlenerek tartışılmış ve gelecekte yapılabilecek çalışmalara ilişkin fikir verilmiştir.

2. YAZILIM TANIMLI AĞLAR

Geleneksel ağ mimarisi bugüne dek başarılı biçimde uygulansa da, bir takım problemleri de beraberinde getirmiştir. Bu problemlerin en önemlisi kuşkusuz yönetim zorluğudur. Yönetilmesi gereken ağın boyutu genişledikçe konfigürasyonu yapılması gereken cihaz sayısı artmakta, bu durum ise ağ yöneticilerinin yükünü artırmaktadır. Yönetimsel zorluğunun yanı sıra güvenlik bakımından da takip edilmesi gereken cihaz sayısının artmasıyla birlikte açıklar artabilmekte, sistemin bir bütün olarak gözlenmesi zorlaşmaktadır.

YTA, bu ve benzeri sorunlara çözüm olarak geliştirilmiş bir altyapı olması sebebiyle son yıllarda önem kazanmış bir çalışma alanıdır. Ağın yönetim sorumluluğunun tek bir birime verilmesi ile birlikte YTA, yönetimsel ve güvenlikle ilgili sorunların önüne geçmeyi amaçlamaktadır. Bu bölümde YTA mimarisi ve geleneksel mimari ile olan farkları açıklanmış, YTA'nın üzerine inşa edildiği OpenFlow protokolünün özelliklerine değinilmiştir.

2.1. YTA Mimarisi

Geleneksel ağ altyapıları her bir ağ cihazının topoloji keşfi yapması ve ağ dışındaki düğümler için yönlendiricilere başvurması üzerine inşa edilmiştir. Geleneksel ağ altyapıları günümüzde büyük çoğunlukla OSI referans modeli baz alınarak geliştirilmiştir [3]. Bu model, ağ altyapılarındaki haberleşmeye bir standart getirmek amacıyla geliştirilmiştir. OSI modelinde ağ cihazların ve uygulamalarının faaliyetlerinin üzerine inşa edildiği 7 adet katman bulunmaktadır. Bu katmanlar ve işlevleri aşağıdaki gibidir:

- Uygulama katmanı: Sistemler üzerinde çalıştırılan uygulamaları ifade etmektedir. Bu katman OSI protokolünün en üst katmanıdır ve hiçbir katmana hizmet etmeyen tek katmandır. Örnek vermek gerekirse dosya transfer protokolü (file transfer protocol -ftp), smtp (simple mail transfer protocol - smtp), http (hyper text transfer protocol - http) gibi uygulamalar bu katmanda çalışmaktadır.
- Sunum katmanı: Veri sunumunda (syntax) uygulamalar arasında doğacak farkları ortadan kaldırarak, farklı uygulamar arasındaki verilerin uyumlu olmasını sağlar.

Örneğin farklı dosya formatları verilerin farklı şekilde ifade edilmesiyle meydana gelir.

- Oturum katmanı: Uygulamaların birbirleri ile görüşmelerini sağlayan katmandır. Örnek vermek gerekirse farklı bilgisayarlar birbirleri ile dosya transferi başlatırken karşılıklı oturum açmaları ve işlem sonrasında bu oturumları kapatmaları gerekmektedir. Bu esnadaki oturum açma ve kapatma işlemleri oturum katmanında gerçekleştirilir.
- Taşıma katmanı: Bu katman, uygulamalar tarafından üretilen verileri ağ altyapısına iletim için hazırlayan ilk katmandır. Bu katmanda uygulamalardan gelen veriler paket olarak ifade edilen parçalara ayrılır. Taşıma katmanı ayrıca güvenilir veri transferinden sorumludur. Verilerin zamanında iletilip iletilmediğinin kontrolü bu katmanda gerçekleştirilir. TCP ve UDP protokolleri bu katman üzerinde çalışır.
- Ağ katmanı: Ağ paketlerinin farklı ağlar arasında yönlendirilmesini sağlayan katmandır. Günümüzde yaygın olarak kullanılan IP protokolü bu katman üzerinde çalışmaktadır. Yönlendirici cihazlar, kendilerine gelen paketleri farklı ağlara iletirken bu katmanda işlem gerçekleştirmektedirler.
- Veri bağı katmanı: Verilerin fiziksel ortama taşınmadan önce çerçeve olarak ifade edilen parçalara bölünmesini sağlayan katmandır. Bu katmanda fiziksel katmanda meydana gelebilecek sinyal seviyesindeki hataların kontrolü yapılır ve cihazlar fiziksel adresleri (mac) ile ifade edilirler. Geleneksel ağ altyapılarındaki anahtarlar, aynı ağ içerisinde birbirlerini tanımlamak ve veri iletimi için bu katman üzerinde çalışırlar.
- Fiziksel katman: Verilerin, cihazların birbiri ile iletişimde kullandıkları fiziksel altyapıya iletilmesini ifade etmektedir. Bu katman, üst katmandan gelen veriyi, fiziksel ortama iletilmesi amacıyla mekanik, radyo, elektriksel, optik ve benzeri sinyallere çevirme işlevini üstlenir.

OSI referans modeli üzerine inşa edilen ağ altyapıları, faaliyetlerini bu katmanlı mimari üzerine yürütmektedirler. Bu model, verilerin organize edilmesi ve iletilmesi için kapsülleme işlemi uygulamaktadır. Her bir katman, bir üst katmandan gelen veriyi alır, kendi verilerini ekleyerek bir alt katmana gönderir. Veriler bu şekilde en alt katmana iletildikten sonra nihai halini almış olur.

Yukarıda ifade edilen mimaride veri bağı katmanında çalışan ağ cihazları, topoloji keşfi yapmakta, her bir arayüzüne bağlı bulunan cihazların listesini tutmakta, bu cihazlardan gelen paketler eğer aynı ağ içerisinde ise kaydettiği topoloji bilgisine göre ilgili

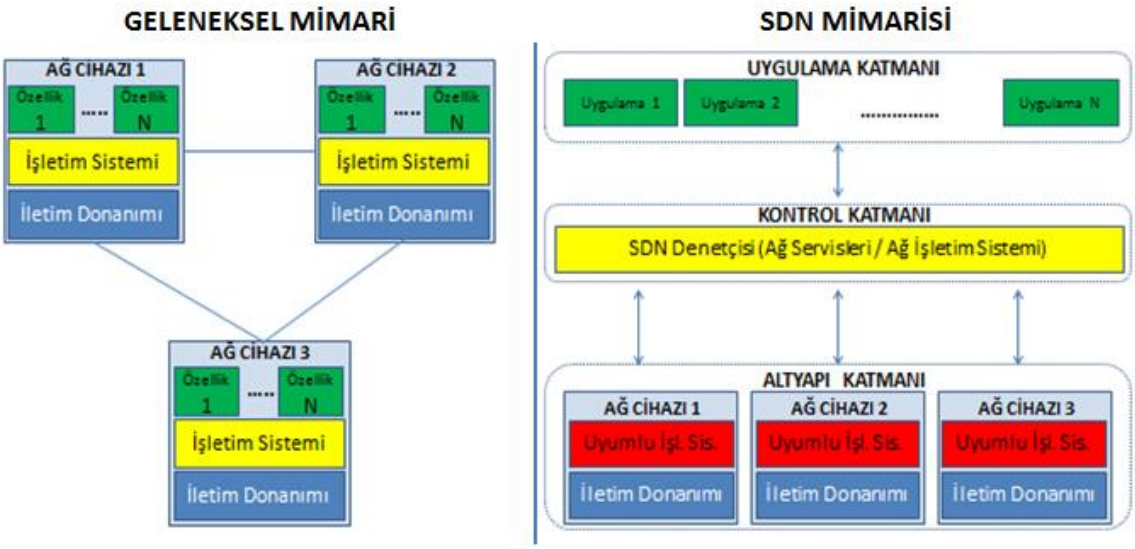
arayüzünden göndermektedir.

Gönderilecek paket farklı bir ağda ise durum biraz daha farklı bir hâl almaktadır. Bu sefer paketlerin hedef ağa gönderilmesi için yönlendirici cihazlara ihtiyaç duyulmaktadır. Bu cihazlar ya kendileri topoloji keşfi yapmakta ya da ağ yöneticileri, hangi ağa veri gönderirken hangi arayüzün kullanılacağı bilgisi daha önceden cihazlara girmektedir. Verilerin yönlendiriciye bir ağ yöneticisi tarafından girildiği durumlarda, ağ üzerinde gerçekleşecek herhangi bir değişiklik cihazlarda konfigürasyon değişiklikliğine yol açmakta ve bu durum maliyet artışına neden olmaktadır.

Yazılım Tanımlı Ağlar, ağ altyapılarında uygulanan yeni bir yaklaşımdır ve geleneksel mimaride yukarıda adreslenen sorunlara çözüm getirmek amacıyla tasarlanmış bir mimaridir. Yazılım Tanımlı Ağlar, ağ altyapılarına getirdikleri farklı bakış açısıyla aşağıdaki avantajları sağlamaktadırlar:

- Altyapı değişikliklerinde, programlanması gereken cihaz sayısının az olması sebebiyle daha kolay adaptasyon sağlar.
- Tüm ağın merkezi noktadan izlenebilmesini ve yönetilebilmesini sağlar.
- Programlanabilir altyapısı sayesinde optimizasyon ve servis kalitesi konusunda iyileştirmelerin daha kolay yapılabilmesini sağlar.
- Açık kaynak altyapısı sebebiyle, üretici bazlı yazılımların bulunduğu ortamlara nazaran uygulama kolaylığı ve anlaşılabilirlik sağlar [1].

Şekil 2.1.'de geleneksel ağ mimarisi ile YTA mimarisi arasındaki farklar görülmektedir. Geleneksel mimaride her bir cihaz paket yönlendirme donanımının yanı sıra yönlendirmeye karar veren işletim sistemi ve yazılımları da üzerinde bulundurmaktadır. YTA yapısında ise anahtarlar sadece yönlendirme işlemini yapmaktan sorumludurlar. paketlerin hangi arayüzden gönderilmesi gerektiğine dair karar verme gibi bir görevleri bulunmamaktadır. Yönlendirmeye karar verme işlemini kontrol katmanı olarak adlandırılan katmandaki kontrolcü birim gerçekleştirmektedir.



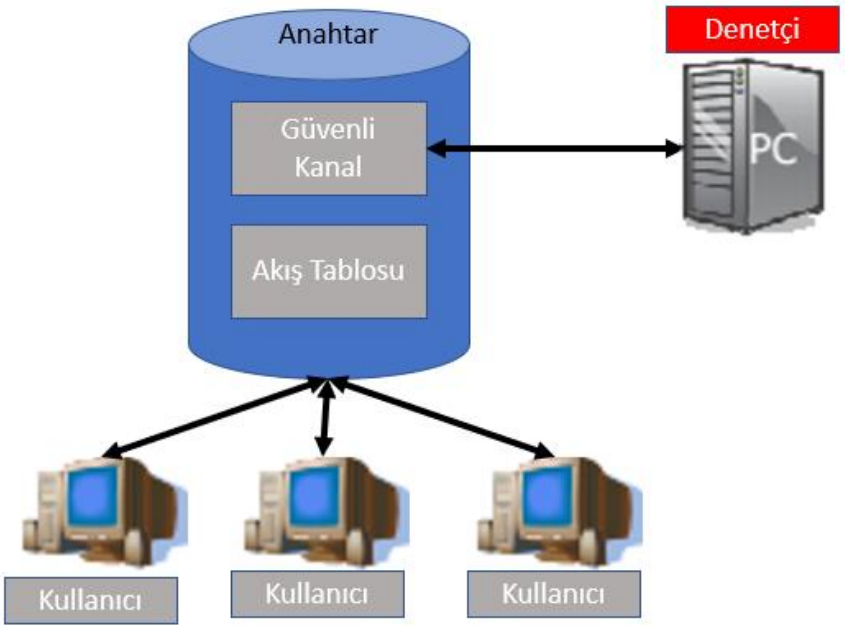
Şekil 2.1. Geleneksel ağ mimarisi ile YTA mimarisinin karşılaştırılması

YTA'da topoloji kontrolü ve cihazların yönlendirilmesi merkezi bir noktadan yapılmaktadır [2]. Anahtar cihazlarının YTA'da yönlendirmeye dair kendi kendine herhangi bir karar vermesi mümkün değildir. Tüm yönlendirme yönetimi, merkezi kontrolcü birim tarafından gerçekleştirilmektedir. YTA'da üç bileşen bulunur:

- Kontrolcü: Ağ ortamındaki yönlendirme kurallarını belirleyerek cihazlara bildiren birimdir. Anahtarlar akış tablolarında bulunmayan bir kaynak ile karşılaştıklarında yönlendirme bilgisi için kontrolcüye danışmakta ve gerekli yönlendirme bilgisi kontolcü tarafından anahtara iletilmektedir.
- Anahtar: Kontrolcü tarafından yönetilen anahtarlar üzerinde bir veya daha fazla akış tablosu bulunmaktadır. Tablodaki her bir girdide, kaynağa ait bilgiler (IP, MAC adresi, Port vb.), istatistiksel manada kullanılacak sayaç bilgisi ve hedef port bilgisi bulunmaktadır. Anahtar ve kontrolcü arasındaki bağlantı güvenliği ise TLS protokolü kullanılarak sağlanmaktadır.
- Kanal: Anahtar ve kontrolcü arasındaki iletişimin sağlandığı ortamdır. Bu ortam sayesinde kontrolcü, anahtar üzerinde gerekli yönlendirme mesajlarını gönderebilir veya ondan gerekli bilgileri alır.

2.2. OpenFlow Protokolü

OpenFlow ilk kez McKeown ve arkadaşları tarafından ortaya konulmuş ve YTA'da temel olarak kullanılan bir protokolüdür. YTA mimarisinde OpenFlow protokolü, Şekil 2.2.'deki mimari temel alınarak geliştirilmiş, anahtarlar ve kontrolcü arasındaki iletişim altyapısını teşkil eden bir protokoldür. Kontrolcü ve anahtarlar, bu protokolde belirtilen kurallar kapsamında birbirleri ile iletişim kurmaktadırlar [2].



Şekil 2.2. OpenFlow protokolüne ait altyapı [2]

OpenFlow protokolünde her bir anahtar, arayüzlerine gelen paketleri yönlendirmeden önce, akış tablosuna bakmakta ve gelen bilgilere dayalı bir yönlendirme kaydı olup olmadığını kontrol etmektedir. Akış tablosu ise aşağıdaki bilgileri içermektedir:

- Eşleşme alanları: Anahtara gelen bir paketin, akış tabloasundaki girdiler ile karşılaştırılmasını sağlayan bilgilerini içermektedir. Bu bilgiler, IP adresi, fiziksel (mac) adresi, port bilgisi gibi paketi tanımlamaya yarayan verileri içermektedir. Anahtarlar kendilerine gelen paketlerdeki bu bilgilere bakarak akış tablosundaki veriler ile karşılaştırma yapmaktadır.
- Öncelik: Akış girdileri arasındaki eşleşme önceliğini ifade etmektedir. Bir paket, akış tablosunda birden fazla girdi ile eşleştiğinde, öncelik değeri en yüksek olan kayda göre

işlem yapılmaktadır.

- Sayaç: Herhangi bir paket, akış tablosundaki bir girdi ile eşleştiğinde bu değer arttırılmaktadır.
- Komutlar: Bahse konu girdiyle eşleşen paketlere yapılacak işlemler belirtilmektedir. Bu işlem paketin anahtarın hangi arayüzünden yönlendirileceği bilgisini içerebileceği gibi, düşürülmesi işlemini de içerebilmektedir. Böylelikle istenmeyen kaynaklardan gelen paketler, ağın diğer kısımlarına ulaşmadan anahtar düzeyinde engellenebilmektedir.
- Zaman aşımı: Akış girdisinin tablodan ne zaman silineceğini belirtmektedir. Bu girdi iki zaman değerinden oluşur:
- Kesin zaman aşımı (Hard timeout): Bu değer ile belirtilen zaman sonrasında, akış tablosundaki girdi her şekilde silinir.
- Boşta kalma zaman aşımı (Idle timeout): Tablodaki girdi, bu değer ile belirtilen zaman süresince herhangi bir paket ile eşleşmediğinde o girdi silinir.
- Çerezler: Paket iletiminde kullanılmayan ancak kontrolcünün paketleri filtrelemede kullanacağı verileri yazmasını sağlayan bir alandır.
- Bayraklar: Akış girdilerinin işleme konulmasında kullanılan bayrak işaretleridir.

Anahtara gelen paketlerin akış tablosundaki girdilerle eşleşmesini sağlayan paket başlığı Şekil 2.3.'te gösterilen şu bilgileri içermektedir: paketin geldiği anahtar arayüzü (port), ağda bir bölümlendirme yapıldıysa numarası, veri bağı katmanına ait kaynak, hedef adres ve tür bilgileri, ağ katmanına ait kaynak, hedef ve tür bilgileri ile iletim katmanına ait kaynak ve hedef kapı bilgisi [2].

Gelen Port	Ağ ID	Ethernet			IP			TCP	
		KA	HA	Tür	KA	HA	Tür	KA	HA

Şekil 2.3. OpenFlow protokolüne göre paket tespitinde kullanılan başlık bilgileri

Eğer bir paket, anahtardaki akış tablosunda yok ise anahtar tarafından yapılacak işlem kontrolcüye bir packet_in mesajı göndermektir. Bu mesaj gönderilirken paketin kendisi anahtarın önbelleğinde tutulur ve kontrolcüye bu paketin başlık bilgileri gönderilir. Eğer anahtarın önbelleği yoğunluk nedeni ile dolmuş ise packet_in mesajı ile birlikte veri paketinin tamamı gönderilir.

Kontrolcü birim kendisine gelen packet_in mesajındaki bilgilere bakmakta ve kaynak ile hedef bilgilerini alarak, ağın topolojisine ve önceden eklenen kurallara göre hangi işlemin yapılması gerektiğine karar vermektedir. Bu işlem paketin belirli bir arayüzden yönlendirilmesi olabileceği gibi, güvenlik gibi çeşitli nedenlerle paketin düşürülmesi (drop) işlemi de olabilmektedir. Verilen karara göre kontrolcü tarafından anahtara bir cevap paketi gönderilmekte ve yapılacak işlem belirtilmektedir. OpenFlow spesifikasyonuna göre packet_in isteği sonrasında kontrolcü tarafından anahtara gönderilen mesajlar aşağıdaki gibi olmalıdır [4]:

- OFPFC_ADD: Akış tablosuna belirtilen verilerle kayıt oluşturulmasını sağlayan bir mesajdır.
- OFPFC_MODIFY: Belirtilen bilgilerle eşleşen akış girdilerinin güncellenmesini sağlar. Örneğin: OFPFF_RESET_COUNTS bayrağı ile gönderilen OFPFC_MODIFY mesajı, akış girdisindeki sayaçların sıfırlanmasını sağlar.
- OFPFC_DELETE: Belirtilen bilgilerle eşleşen akış girdilerinin silinmesini sağlar.

Kontrolcüden gelen cevabı alan anahtar akış tablosunu aldığı bilgilere göre güncellemektedir. Güncelleme işlemi esnasında kontrolcü tarafından belirlenen kayda ait kesin zaman aşımı ve boşta kalma zaman aşımı değerleri de tabloya girilmektedir. Bu değerlerden boşta kalma zaman aşımı, akış girdisinin ne kadar süre hiç kullanılmaz ise silineceğini belirtirken, kesin zaman aşımı değeri kaydın her durumda silinmesi gereken zaman aşımını belirtmektedir. Aynı başlık bilgilerine sahip başka mesajlar anahtara geldiğinde, akış tablosuna bakılacak ve bu kez akış tablosunda eşleşen bir kayıt bulunması sebebiyle kaydın karşılığındaki işlem gerçekleştirilecektir.

OpenFlow protokolünde kullanılan anahtar-kontrolcü ikilisi, tasarım olarak belirli sorunların önüne geçse de, birtakım saldırı tiplerine karşı açık hedef haline gelmektedir. Saldırılar bu mimarinin bileşenlerinin bir kısmını ya da hepsini hedef alacak biçimde olabilmektedirler. Bu saldırıların önemli bir kısmını DoS saldırıları teşkil etmektedir. 3. Bölümde bu saldırıların nasıl gerçekleştirilebileceği, etkileri ve olası çözümleri özetlenmiştir.

3. YTA'DAKİ DoS SALDIRILARINA YÖNELİK LİTERATÜRDEKİ ÇALIŞMALAR

3.1. YTA'daki DoS Saldırılarının Sınıflandırılması

YTA, bugüne kadar kullanılan ağ donanımlarına bir eklenti olarak görülmektedir ve büyük oranda mevcut ağ altyapısını kullanmaktadır. Bu nedenle mevcut saldırıların birçoğu yazılım tanımlı ağlarda da kullanılabilmektedir. Hizmet Dışı Bırakma Saldırıları geleneksel ağ altyapılarının birçok katmanında farklı şekillerde uygulanabilmektedir.

YTA temelinde DoS saldırılarında ise saldırılar anahtar, kontrolcü veya aradaki bağlantı katmanını hedef alma durumlarına göre sınıflandırılmaktadırlar. Bazı durumlarda saldırı bu birimlerden birine yönelik iken, bazı durumlarda birçok birimi birden etkileyen saldırılar yapılabilmektedir.

YTA altyapısı DoS saldırılarına karşı bir kurban olarak incelenecek olursa, hangi YTA bileşenlerinin DoS saldırılarına hedef teşkil ettiğinin belirlenmesi gerekmektedir. YTA katmanlarından her biri ayrı bir DoS saldırısına hedef olabilmektedir. Yan, Yu, Gong ve Li bu saldırıları aşağıdaki şekilde gruplandırmıştır [5].

- Uygulama katmanı saldırıları: Kontrolcü üzerinde çalışan uygulamalara veya bu uygulamalar ile kontrolcü arasındaki iletişime yönelik gerçekleşmektedir.
- Kontrol katmanı saldırıları: Kontrolcünin kendisini veya, kontrol katmanının iletişimde bulunduğu diğer katmanlar ile iletişimini sekteye uğratma yönünde gerçekleşmektedirler.
- Altyapı katmanı saldırıları: Anahtarlara veya anahtar-kontrolcü iletişimini sekteye uğratmak amacıyla gerçekleştirilmektedir.

Li, Meng ve Kwork ise DoS saldırılarını anahtar ve kontrolcü bazlı olarak gruplandırmışlardır [6].

- Anahtar bazlı saldırılar: Anahtara gönderilecek farklı hedef adreslere sahip çok sayıda paketin anahtar akış tablosunu dolduracağı ve bunun da normal kullanıcılara ait yönlendirme işlemlerinin gecikmesine neden olacağı belirtilmiştir.

- Kontrolcü bazlı saldırılar: Kontrolcüye çok sayıda istek gönderilerek işlem kapasitesinin hedef alınabileceği ve bu sayede tüm ağda bir yavaşlama meydana getirilebileceği ifade edilmiştir.

Shu, Wan, Li ve diğerleri ise YTA altyapısındaki saldırıları aşağıdaki şekilde gruplandırmaktadır [7].

- Altyapı katmanı saldırıları: Anahtara farklı hedef adresleriyle gönderilecek birçok paket ile gerçekleştirilen flooding saldırılarının anahtar önbelleği ile akış tablosunu doldurmasına neden olabileceği belirtilmiştir.
- Kontrol katmanı saldırıları: Altyapı katmanında gerçekleştirilen saldırılara benzer şekilde yapılacak flooding saldırılarının birçok packet_in mesajı üretilmesine neden olduğu ve bunların da kontrolcünün işlem kapasitesini ve anahtar-kontrolcü iletişim kapasitesini ciddi manada düşüreceği belirtilmiştir.

YTA ortamındaki DoS saldırıları alanında yukarıda özetlenen araştırmalar incelendiğinde önemli bir bölümünün altyapı katmanındaki saldırılardan meydana geldiği görülmektedir. Bu saldırılar açıklanacak olursa:

Veri katmanında saldırgan tarafından ağın farklı noktalarına birçok bağlantı isteği gönderilerek, anahtarın akış tablosunda ve önbelleğinde gereksiz birçok girdi oluşturulması sıkça uygulanan bir yöntemdir. Bu yöntem uygulandığında anahtarın akış tablosu veya önbelleği dolduğundan, gereksiz girdileri silme ve yeni yönlendirme isteğini kontrolcüye göndererek cevabını bekleme süresince iyi niyetli isteklerin gereksiz yere bekletilmesi durumu meydana gelebilmektedir. Bu saldırılar gerçekleştirilirken ağın topolojisinin tespit edilmesi amacıyla tarama (scanning) ve saldırgan kimliğini gizleyip başka düğümleri taklit etme amacıyla aldatma (spoofing) yöntemleri de birlikte kullanılabilmektedir [6].

Yukarıda açıklanan DoS saldırıları anahtarı etkilediği kadar kontrolcüyü ve aradaki bağlantı kanalını da etkilemektedir. Anahtarlar kendi tablolarında bulunmayan her bir adres için kontrolcüye packet_in mesajı göndermekte, kontrolcü tarafından gerekli hesaplamalar yapıldıktan sonra paketin iletileceği yol bilgisi anahtara gönderilmektedir. Kötücül olarak yaratılan her bir packet_in mesajı kontrolcünün işlem kapasitesini düşürmekte ve aradaki bağlantı kanalının kapasitesini boşa harcamaktadır [7].

Literatürdeki çalışmalarda YTA'da altyapı katmanında kontrolcü, anahtar veya aradaki bağlantı arayüzünü tüketme üzerine yapılan saldırılar geleneksel ağ altyapılarındaki flooding yöntemleri uygulanarak gerçekleştirilmiştir.

Örnek olarak SYNFlood saldırısı ile hedefe TCP_SYN mesajı göndererek bağlantı isteğinde bulunulmaktadır. TCP Protokolü gereğince kurban bu isteğe SYN_ACK ile cevap vermekte ve bağlantı için kaynak ayırmakta, ayrıca karşı taraftan gelecek ACK paketini beklemektedir. Ancak saldırgan tarafından hiçbir zaman gönderilmeyen ACK paketi ve sürekli farklı adreslerden gönderilen bağlantı istekleriyle kurbanın kaynakları tüketilebilmektedir. YTA ağında farklı hedeflere gönderilen TCP_SYN mesajları hem hedefin kaynaklarını tüketmesine neden olacak, hem de iletim yönü bilinmeyen paketler için kontrolcüyü meşgul edecek; aynı zamanda anahtar tablosunu boş yere dolduracaktır [8,9].

Bir başka saldırı tipi olan UDPFlood saldırısında ise hedefe farklı portlardan birçok UDP paketi gönderilmekte ve hedefin bu portlarla ilgili uygulama olup olmadığını araştırmasına neden olmaktadır. YTA ağında farklı ip adresleriyle farklı hedeflere birçok UDP paketinin gönderilmesi anahtar kuyrukları ile kontrolcü kaynaklarının boşa tüketilmesine neden olmaktadır [10].

Başka bir saldırı tipi olan ICMP Flooding saldırısında ise saldırgan tarafından hedefe ICMP Echo Request paketleri gönderilerek cevap istenmektedir. Birçok hedef ICMP protokolünün gereğini yerine getirerek bu pakete cevap vermektedir. YTA ağında bu işlem farklı ip adresleriyle ağdaki birçok hedefe gönderilerek yukarıdaki yöntemlere benzer biçimde anahtar ve kontrolcü kaynaklarının tüketilmesi sağlanmaktadır [11].

Yukarıdaki yöntemlerden hangisi uygulanırsa uygulansın, tüm saldırılardaki genel amaç YTA'nın zayıf yönü olarak tabir edilebilecek, anahtarların akış tablosu ve bu tablonun oluşturulması aşamasındaki anahtar-kontrolcü arasındaki mesajlaşmadan faydalanmaktır. Literatürdeki çalışmalar incelendiğinde flooding saldırılarının, kontrolcü üzerinde packet_in mesajları sebebiyle yarattığı etkinin önemli olduğu anlaşılmaktadır.

3.2. DoS Saldırılarına Karşı Geliştirilen Yöntemler

YTA'da saldırı için geliştirilmiş çeşitli yöntemler olduğu gibi, bu saldırılara karşı geliştirilmiş savunma yöntemleri de bulunmaktadır. Specht ve Lee yaptıkları incelemede DoS saldırılarına karşı geliştirilen yöntemleri şu şekilde sınıflandırmaktadırlar: Saldırıların Tespiti, önlenmesi, etkisinin azaltılması, tamamen etkisizleştirilmesi, saptırılması ve atak sonrası inceleme [12].

- Saldırıların tespiti: DoS saldırısının gerçekleşmesiyle birlikte bu saldırının oluştuğunun tespit edilmesi yöntemlerini kapsamaktadır.
- Saldırıların önlenmesi: Saldırı gerçekleşmeden önce yapılan faaliyetleri kapsar. Amacı faaliyetin gerçekleşmesinin engellenmesidir.
- Etkisinin azaltılması: Saldırı gerçekleştikten sonra, tamamen engellenemese bile ağa verdiği zararların en aza indirilmesi ile ilgili yöntemlerdir.
- Etkisizleştirilmesi: Saldırının tamamen ortadan kaldırılması faaliyetlerini kapsar.
- Saptırılması: Saldırganlara yanlış hedef gösterilerek asıl hedefin korunmasıdır.
- Atak sonrası inceleme: Saldırı sonrasında saldırı kaynağının ve yolunun tespiti, kayıtların incelenmesi gibi faaliyetlerdir.

Literatürde YTA alanında yapılan DoS saldırılarına ilişkin çözüm yöntemleri yukarıdaki başlıklar göz önüne alınarak incelendiğinde tespit, etki azaltma, önleme ve saldırıya yanıt verme faaliyetlerinin önem kazandığını söylemek mümkündür.

3.2.1. Saldırı tespit yöntemleri

Chen ve Yu tarafından yapılan çalışmada YTA'da kullanılan Saldırı Tespit (Intrusion Detection) yöntemlerinin kontrolcü üzerinde bulundurulmasının yaygın olduğu, ancak bu yöntemin kontrolcü üzerine fazlaca yük getireceği ve tek nokta problemi yaratacağı ileri sürülmüş ve hesaplamanın dağıtık hale getirilmesi için Yapay Sinir Ağı modelindeki her bir nöron, birer YTA anahtarı olarak düşünülmüştür. ICMP Flood, SYN Flood, UDP Flood ve DNS Reflection saldırılarının tespiti için aşağıdaki özellikler kullanılmıştır.

- ICMP Paketlerinin diğer paketlere oranı,

- UDP Paketlerinin diğer paketlere oranı
- SYN_ACK paketlerinin diğer paketlere oranı özellik olarak kullanılmıştır.

Sonuçlar incelendiğinde başarı oranları ICMP Flooding saldırısı için %96.3, SYN Flood için %94.8, UDP Flood için %95.8 ve DNS Reflection saldırısı için %93.1 olarak ölçülmüştür [13].

Braga, Mota ve Passito tarafından yapılan bir çalışmada, NOX yönlendiricisi ve Özdüzenleyici Haritalar (Self Organising Map-SOM) kullanılarak DoS saldırısının tespiti gerçekleştirilmiştir. Tespit yönteminde sadece akış tablolarındaki bilgilerden faydalanılmıştır. Çalışma biçimine bakılacak olursa, öncelikle kontrolcü tarafından akış bilgileri toplanmakta, özellik seçim modülü tarafından önemli özellikler ayıklanmakta ve SOM ile sınıflandırılmaktadır. Sınıflandırmada kullanılan özellikler aşağıdaki gibidir:

- Akış başına ortalama byte miktarı.
- Akış başına ortalama paket sayısı.
- Akış tablosunda bir akışın ortalama kalma zamanı.
- Karşılığı olan akışların tüm akışa oranı. (DoS ataklarında karşılığı olmayan paketler çoğunluktadır.)
- Karşılıksız akışların artış oranı. (DoS ataklarında bu oran artmalıdır.)
- Farklı olan port sayısındaki artış.

Yukarıdaki özellikler kullanılarak yapılan eğitim sonucunda sistemdeki tespit oranları en düşük durumda %98.57 en iyi durumda %99.11 olarak verilmiştir [14].

Cui ve diğerleri tarafından tespit işlemi iki aşamalı olarak gerçekleştirilmiştir. Kontrolcüye gelen packet_in mesajlarının sayısı tutularak bu sayılarda fazla bir artış var ise exact-STORM adlı algoritma ile değişimin anormal olup olmadığına bakılmaktadır. Eğer packet_in sayısında anomali bulunur ise yapay sinir ağı ile sınıflandırma yapılarak saldırı olup olmadığına bakılmaktadır. Geliştirilen yöntem avantaj olarak periyodik kontrol yerine packet_in mesajıyla yapılan tetiklemenin daha efektif sonuçlar doğurduğu öne sürülmektdir. YSA sınıflandırmasında kullanılan özellikler aşağıdaki gibidir.

- Akış başına toplam paket sayısı
- Akış başına toplam byte miktarı
- Akışın tabloda kalma süresi
- Akış başına saniyedeki paket sayısı
- Akış başına saniyedeki byte miktarı

Sonrasında diğer çalışmalarda görülmeyen bir yöntem uygulanmış ve ağdaki atak yolunun tespit edilmesine çalışılmıştır. Her bir anahtar ayrı ayrı YSA ile tekrar sınıflandırılarak saldırı yolu bulunmuştur [15].

Shirali-Shahreza ve Ganjali ise paket içeriğini temel alarak kontrol yapılması durumunda örneklemenin hangi aralıklarla yapılacağına yönelik bir çalışma gerçekleştirmişlerdir. Çalışmada ele alınan noktalar, kaç pakette bir örnekleme yapılacağı ve örneklemenin kaç farklı paketle yapılacağı, belli bir sayıdan fazla pakete sahip akışların örneklemeye dahil edilmemesi ve akışların belirli sayıdaki ilk paketlerinin bağlantı kurmaya yaraması sebebiyle bu paketlerin de örneklemeye dahil edilmemesi durumları olmuştur. Çalışmada elde edilen sonuçlar, tüm paketlerin örneklemeye dahil edildiği durumlara göre yoğunluğun büyük oranda azaltıldığını göstermektedir [16].

YTA altyapısının avantajlarından biri, merkezi yönetimi sayesinde güvenlik ile ilgili bilgilerin tek noktadan idare edilmesindeki kolaylıktır. Bu duruma ilişkin bir çalışma Flowguard altyapısı ile gerçekleştirilmiştir. Kontrolcü ile entegre güvenlik duvarına daha önceden akış bilgileri girilmekte ve paketlerin izledikleri yollar takip edilerek güvenlik duvarı kurallarına aykırı bir durum olup olmadığına bakılmaktadır [17].

Buragohain ve Medhi tarafından geliştirilen FlowTrApp ile Normal kullanıcıların kullanım esnasındaki verileri SFLOW-RT yazılımı ile toplanmış ve akış süresi ve akıştan belli bir zamanda geçen veri miktarı için min ve max aralıkları belirlenmiştir. Belli sunuculara giden paketler SFLOW-RT ile gözden geçirilerek belirlenen değerlerin arasında olup olmadığına bakılmış, değilse kötücül olarak sınıflandırılmıştır [18].

Dao, J. Park, M. Park ve Cho tarafından yapılan çalışmada sayaç değerlerine bağlı olarak sınıflandırma gerçekleştirilmiştir. Kaynağın bağlantı sayısı daha önce belirlenen k

değerinden büyük ise ve bağlantı başına gönderdiği paket sayısı sayısı n değerinden küçük ise kötücül olarak tanımlanmıştır. Bağlantı sayısı k değerinden küçük ise normal kullanıcı olarak görülmüştür [19].

Operetta sisteminde ise SynFlood saldırılarına karşı Avant-Guard ile belirtilen yönteme benzer bir sistem geliştirilmiştir. Geliştirilen yöntemde istemcilerin SYN istek sayıları tutulmaktadır. İstek yapan taraf hedeften kendisine gelen SYN_ACK paketi sonrası geriye ACK paketi göndermez ise belli bir eşik değerinden sora kara listeye eklenmektedir. Ardından anahtarlara ait akış tablolarına bu ip adresine ait drop kuralı eklenmektedir. ACK paketi gelir ise beyaz listeye eklenmekte ve anahtarlarda akış kuralı belli bir zaman aşımı değeriyle birlikte girilmektedir. Sistemin AVANT-GUARD'dan farkı ise, AVANT-GUARD kaynak ve hedef arasında bir aracı gibi çalışıp, güven ilişkisini sağladıktan sonra TCP_SYN mesajlarını direkt hedefe iletip bağlantıyı birleştirirken, OPERETTA ise güven ilişkisini kurduktan sonra kaynağa RST mesajı yollayıp anahtara da ilgili akış kaydını girerek kaynak ve hedefin tekrar mesajlaşmasını sağlamasıdır. Yapılan deneyde 10 saldırgan ile normalde 2800 civarında olan SYN isteği sayısı 100 civarında bir sayıya indirgenebilmiştir [9].

Y. Qian, You ve K. Qian tarafından yapılan çalışmada ise daha önce bahsedilen çalışmalara benzer biçimde kara liste uygulaması yapılmıştır. Geliştirilen yöntemde kontrolcü tarafında kendisine gelen her bir packet_in mesajında, eğer kaynağın (MAC ve IP adresine göre) saniyedeki paket sayısı veya saniyedeki byte miktarı belirlenen eşik değerini aşıyorsa kara listeye eklenmektedir. Sonrasında kaynak için anahtarlara blok kuralı eklenmekte ve kaynağa ait akış girdileri anahtarlardan silinmektedir. Geliştirilen yönteme ait sonuçlar incelendiğinde avantajların olduğu kadar olumsuz sonuçlarında meydana geldiği belirtilmiştir. Anahtarlardaki akış girdisi sayısı DoS saldırısının tepe noktasında normalde 900 iken bu yöntemle 100'e düşmüştür. Ancak normal kullanıcılar için bant genişliği değerinde bir iyileşme görülmez iken paket kayıpları az da olsa artmıştır [20].

Dao, Kim ve Cho ise Qian ve diğerleri tarafından yapılan çalışmaya benzer bir kara liste uygulaması gerçekleştirilmiştir. Çalışmada öncelikle bir kaynağın normal durumda ne kadar istek paketi göndermesi gerektiğini belirlemek için güvenilir adres veri tabanındaki ip adreslerinden gelen packet_in istek sayılarının ortalaması alınmaktadır. Sonrasında

bulunan ortalama değer kullanılarak k1 ve k2 limitleri belirlenmekte ve bu değerler periyodik zaman aralıklarında güncellenmektedir. Saldırganlar, DDoS saldırılarında k1 değerinden düşük sayıda istek mesajları gönderirken, DoS saldırılarında k2 değerinden daha fazla istek mesajı göndermektedirler. Saldırı zamanında bu değerlerin altında ve üstünde olan bağlantılar akış tablolarından silinmekte ve şüpheli görülerek gönderdikleri veri paketi miktarına bakılmaktadır. Veri paketi miktarının önceden belirlenen eşik değerinin altında ve üstünde olması durumunda kaynak adres güvenilir veri tabanında ise oradan silinmekte, akış tablolarına da kaynak ip adresine ait engelleme kaydı düşülmektedir. Bir önceki yöntemle karşılaştırıldığında packet_in isteği ve veri paketinin birbiri ile ilişkili olarak ele alınmasının daha iyi sonuçlar verebileceği değerlendirilmektedir [21].

SECOD sisteminde ise kara liste uygulaması farklı katmanlarda gerçekleştirilmiştir. Packet_in mesajı sayısı temelinde anahtar, anahtar arayüzü ve IP adresi temelinde üç farklı eşik değeri uygulaması yapılmıştır. Anahtar arayüzü bazında tutulan packet_in mesajı sayıları eşik değerlerini dinamik olarak güncellemek amacıyla kullanılmıştır. Bu eşik değerleri aşıldığında ise sistem, öncelikle ilgili anahtarda herhangi bir akış kuralı ile eşleşmeyen paketlere dair drop kuralı eklemekte, eğer saldırı halen devam ediyorsa bu kez kontrolcü tarafında ilgili anahtarın isteklerini gözardı etmektedir. Sonuçlara bakıldığında veri iletiminde %99,72 oranında bir iyileşme sağlandığı belirtilmiştir [22].

Carvalho ve arkadaşları tarafından yapılan çalışmada entropi tabanlı bir tespit mekanizması uygulanmıştır. Entropi hesaplaması yapılırken, kontrolcüye gelen istek paketlerindeki her bir IP adresinin diğer IP adreslerine olan oranı hesaplanmış, bu oran temelinde entropi ölçümü gerçekleştirilmiştir. Çalışmadaki eksik yönler ele alınacak olursa, eşik değeri sabit ve ağ yöneticisi tarafından belirlenecek şekilde ayarlanmıştır. Bu durum değişen ağ yapıları için uygulanabilir görünmemektedir. Ayrıca entropi haricinde başka bir yöntem ile karşılaştırma da yapılmamıştır [23].

3.2.2. Saldırı etkisinin düşürülmesine dair yöntemler

Floodlight kontrolcüsünde [24] anahtar başına packet_in sayısı için belli bir eşik değeri verildiği ve bu değer aşıldığında eğer aynı MAC adresinden iki farklı adrese hedef olan packet_in mesajı gelirse o MAC adresi ya da port için 5 saniye boyunca engelleme kuralı

koyulduğu belirtilmiştir [10]. Bu yöntem iki yönden dezavantajlı bulunmuştur. İlk olarak eşik değerinin sabit olmaması gerektiği, dinamik olması gerektiği belirtilmiştir. Bu durum ağ yoğunluğunun değişebileceği göz önüne alındığında olması gereken bir özelliktir. Bir mac adresi veya anahtar arayüzünün kısıtlanmasıyla eğer o arayüze birden çok kullanıcı bağlı ise hepsinin cezalandırılacağı bildirilmiştir ki bunda haklılık payı mevcuttur.

MLFQ (Multi Layer Fair Queing) sisteminde ise IP aldatma yöntemiyle birlikte uygulanan UDP Flooding saldırılarının kontrolcü kaynaklarını tüketmesinin önüne geçilmeye çalışılmıştır. Geliştirilen sistemde kontrolcü üzerinde dinamik olarak genişletilip daraltılan kuyruk yapısı tasarlanmış ve bu kuyruklar anahtar bazında tahsis edilerek önceliklendirme yapılmıştır. Anahtarlara ait kuyruklar flooding nedeniyle dolduğunda, bu kuyruklar port bazında alt kuyruklara ayrılmışlardır. Böylece kontrolcü tarafında kuyruğun dolmasından kaynaklanan sorunların önüne geçildiği belirtilmiştir. Sistem üzerine geliştirildiği Floodlight kontrolcüsüne göre daha başarılı RTT zamanı sunmaktadır [10].

Flowsec sisteminde OpenFlow protokolünün 1.3.0 versiyonunun bir özelliği olan Meter tabloları kullanılarak, anahtar başına bant genişliği tüketimi ölçülmüş, belli bir oranı geçen anahtarın kontrolcüye olan bant genişlikleri yarı oranında düşürülmüştür. Sonuçlarda saldırı anında ortalama bant genişliği tüketimi 7 Mbps seviyesinde iken önerilen yöntem ile 3,5 Mbps olarak ölçülmüştür [11].

FlowFence sisteminde ise anahtarlar üzerindeki bağlantılardaki yoğunluk durumları izlenmiştir. Anahtarlar çıkış kanallarında %80 den fazla yoğunluk tespit ederse kontrolcüye bilgi vermektedir. Kontrolcü bilgi veren anahtara ait bağlantı üzerinden veri gönderen diğer anahtarlardan istatistik bilgisi almaktadır. Bu bilgide kaynak ve hedef ip adresleri ile akış süresi bilgileri bulunur. Sonrasında kontrolcü söz konusu bağlantı için akış başına ortalama bant genişliği değerini bulur. Ortalama değerin altındakiler arta kalan bant genişliği miktarı ile ödüllendirilir. Ortalama üstündekiler ceza formülü kullanılarak cezalandırılır ve bant genişlikleri düşürülür. Yapılan deneylerde 3 saldırganın bulunduğu bir ortamda normal kullanıcıların normal zamanda 40 Mbps olan bant genişliği kullanım miktarı saldırı durumunda 0'a yakın iken FlowFence ile 20 Mbps olarak ölçülmüştür [25].

Yau, Lui, Liang ve Yam ise kontrolcü üzerindeki yükün maksimum ve minimum noktası belirlenerek trafiğin bu aralıkta tutulmasına yönelik bir çalışma gerçekleştirmişlerdir. Gelen trafik maksimum değerin üzerinde ise bant genişliği tüm uç anahtarlarda yarıya düşürülmüştür. Eğer trafik minimum değer altında ise altı ise bant genişliği tüm uç anahtarlarda belli oranda arttırılmaktadır. Trafik yoğunluğu min-maks aralığına gelene kadar bu işlem devam ettirilmiştir [26].

Saifullah ise yukarıda belirtilen çalışmadaki eksik yönleri ortaya koymuştur. Dezavantaj olarak rastlantısal olarak iyi niyetli kullanıcıların yoğunlukta bulunduğu anahtarların da bant genişliklerinin yarıya düşürülmesiyle adaletsiz bir cezalandırma yöntemi olacağı belirtilmiştir. Bu duruma çözüm olarak ağırlıklandırmalı bir sistem öne sürülmüştür. Anahtarlar BFS ağaç yapısına göre düşünülmüştür. Öncelikle her bir anahtar kendine bağlı tüm bağlantıların (istemci ya da anahtar olabilir) en düşük bant genişliğini alarak bunu istemci sayısı ile çarpmakta ve toplam bant genişliğini bulmaktadır. Böylece tüm anahtarlar kendilerine bir bant genişliği sınırı koymaktadırlar. Kontrolcü tüm anahtarlardan bant genişliği bilgisini alır. Toplam bant genişliği beklediğinden düşük ise kalan farkı istemci sayısına bölerek istemciler arasında paylaştırmaktadır. Paylaştırılan değer tüm anahtarlar tarafından hesaplanarak bant genişliği değeri arttırılmaktadır. Böylece bant genişliğini fazlaca kullanan istemcilerin ağdaki etkisi azaltılmış olmakta ve istemci sayısı fazla olan anahtarların bant genişliğinden daha adil bir pay alması sağlanmaktadır. Çalışmada herhangi bir uygulama veya karşılaştırma sonucu sunulmamıştır [27].

SDNManager isimli sistemde ise akış bilgileri ağ üzerinde sürekli izlenmekte ve ileriki döneme yönelik olarak bant genişliği tahmininde bulunulmaktadır. Bant genişliği tahmininde bulunulurken genellikle ekonometri alanında uygulanan koşullu otoregresif heteroskedastik model benimsenmiş, bu model kapsamında zaman serisi bazında değişimler ölçülerek ağın toplam bant genişliğindeki volatilite tahmin edilmeye çalışılmıştır. Tahmin yapıldıktan sonra, mevcut döneme ait tahmin edilen bant genişliğinin üzerinde olan düğümler, bant genişliğini aştıkları oranda cezalandırılmışlardır [28].

SoftGuard sisteminde düşük oranda (low rate) DoS saldırılarına yönelik bir çözüm yöntemi geliştirilmiştir. Düşük orandaki saldırının tespiti için her bir anahtara izleme

kuralları eklenmiş ve bu kurallar üzerinden geçen veri miktarı periyodik olarak ölçülerek toplam bant genişliği sürekli izlenmiştir. Bant genişliğinde belirgin bir düşüş meydana geldiğinde ise saldırganın tespit edilmesi aşamasına geçilmektedir. Tespit aşamasında ortalama bant genişliğinin üzerindeki anahtar arayüzleri, daha önceden belirlenen bant genişliğindeki düşüşe benzerlikleri var ise kayıt altına alınmaktadır. Saldırganlarla mücadele yöntemi olarak ise ilgili akışa dair drop kuralı eklenmesi veya bant genişliğinin düşürülmesi önerilmiştir [29].

3.2.3. Saldırı önleme yöntemleri

DoS saldırılarında sıklıkla kullanılan IP aldatma (spoofing) olayına çözüm olarak getirilen SAVI altyapısında veri bağı seviyesinde kimliklendirilen istemcilerin IP adreslerinin doğru aralıkta olup olmadığı kontrol edilmektedir. Eğer alması gereken IP aralığından farklı bir IP'ye karşılık gelen istemci mevcut ise IP aldatma durumu tespit edilmektedir [30].

Yao, Bi ve Xiao trafından yapılan çalışmada SAVI altyapısı YTA altyapısına uyarlanmıştır. Üzerinde SAVI algoritması çalışan OpenFlow anahtarları güvenilir ağda kabul edilmiştir. Güvenilir ağ haricinden gelen paketlerin kaynak ve hedeflerine bakılarak, ip aldatması olup olmadığına karar verilmiştir. Eğer kaynak anahtar güvenilir ağ dışından ancak paket içeriğindeki ipler güvenilir ağ içinden ise aldatma var denilmiştir. Böylece YTA ağı dışından gelen paketlerin de güvenilirliğinin sağlandığı belirtilmiştir [31].

AVANT-GUARD sistemi SYNFlood saldırılarına karşılık olarak geliştirilmiş bir yöntemdir. Geliştirilen sistem kontrolcü ve anahtar arasında bir katman görevi görmekte ve TCP_SYN mesajlarının hedefe ulaşmasından önce kontrolcü tarafından kontrol edilerek onaylanması sağlanmaktadır. Kontrolcü ise bu duruma karar verirken SNORT benzeri araçlarla kontrol yaparak DoS saldırısı olup olmadığına karar vermektedir. Sonuçlarda SYNFlood saldırısında koruma olmadan ağda tıkanıklık meydana gelirken, AVANT-GUARD ile yanıt zamanları 0.399'dan 0.401 saniyeye düşmüştür [8].

OF-Guard sisteminde ise packet_in mesajı sayısı baz alınarak şüpheli bir durum olup olmadığına karar verilmekte, eğer packet_in mesaj sayısı belli bir değerin üzerinde ise bu

kaynaktan gelen paketler bir önbellek sunucusuna yönlendirilmektedir. Burada gelen mesaj detaylı incelenerek ağda daha önceden oluşturulan packet_in mesajlarından türetilen olası mesajlar karşılaştırılmakta, uyumluluk görülmediği taktirde saldırgan olarak sınıflandırılmakta ve engellenmektedir [32].

OF-Guard'ın devamı niteliğinde olan FloodGuard sisteminde ise kontrolcü üzerindeki yönlendirme programı incelenerek girdi ve çıktı olasılıkları önceden hazırlanmış ve gelen packet_in mesajları bu olasılıklarla karşılaştırılmıştır. Uygun olmayan mesajlara ait kaynaklar engellenmiştir. Yapılan deneylerde sistemin FloodGuard olmadan çalışan versiyonda DoS saldırısı saniyede 150 paket olduğunda iletim oranı yarı yarıya düşerken FloodGuard ile DoS saldırısı saniyede 200 paket olduğunda iletim oranı 8,4 Mbps seviyesinden yalnızca 8.3 seviyesine inmiştir [33].

3.2.4. Saldırıya yanıt verme yöntemleri

DoS saldırısında saldırı tespitinin yapılmasının ardından bazı çalışmalarda saldırının tamamen ortadan kaldırılmasına yönelik işlemler de gerçekleştirilmiştir [8,9,15,18,20,21]. Bu işlemlerin saldırıyı tamamen ortadan kaldırdığını söylemek mümkün olmasa da, tespit sonrasında gerçekleştirilen faaliyetler ile buna yönelik hareket edilmektedir.

Saldırıya yanıt verme yöntemleri birçok farklı şekilde gerçekleştirilebilmektedir. Specht ve Lee, çalışmalarında DoS saldırılarına yanıt verme yöntemlerini, saldırı kaynaklarının ortadan kaldırılması olarak tanımlamaktadırlar. Saldırıyı gerçekleştiren servislerin veya cihazların kapatılması gibi faaliyetler bu kapsama girmektedir. Bu durum saldırı kaynağının tam olarak tanımlanabilmesi ile mümkündür, bir başka deyişle tespit işleminin doğru biçimde gerçekleştirilmesi gereklidir [12].

DoS saldırılarının YTA üzerindeki etkileri ele alındığında, mümkün olan etkisizleştirme yöntemleri saldırı kaynaklarının kapatılması veya bu kaynaklara ait trafiğin devre dışı bırakılmasıdır. DoS saldırılarının genellikle dağıtık şekilde ve birçok kaynaktan yapıldığı düşünüldüğünde kaynağın kapatılması her zaman mümkün değildir. Bu nedenle trafiğin devre dışı bırakılması daha olası bir seçenektir.

YTA'nın sunduğu merkezi yönetim altyapısının faydası bu noktada dikkat çekmektedir. Tespit yöntemleri başlığında açıklanan yöntemlerle saldırı olarak sınıflandırılan durumların tümünde engelleme işlemi anahtarlardaki akış tabloları kullanılarak gerçekleştirilmiştir. Tespit işleminin ardından akış tablolarına ilgili kaynağa ait drop kuralları eklenmiştir. Böylelikle anahtar tarafından saldırgan olarak sınıflandırılan adreslere ait veri paketi ile karşılaşıldığında bu paket otomatik olarak düşürülecektir. Drop kuralı konulmasının yanısıra, DoS saldırısının oluşturduğu akış tablolarındaki gereksiz girdilerin temizlenmesi de saldırı etkilerinin giderilmesini sağlayan başka bir tekniktir.

3.2.5. Özet ve değerlendirmeler

Tez kapsamında incelenen kaynaklardan elde edilen sonuçlar Çizelge 3.1.'de özetlenmiştir.

Çizelge 3.1. Tez kapsamında incelenen kaynakların karşılaştırması

Kaynak	Amaç	Yöntem	Faydalanılan Özellik	Sağlanan Fayda
Chen & Yu [13]	Tespit	Makine Öğrenmesi-Sınıflandırma	ICMP Oranı UDP Oranı SYN-ACK Oranı	%931-96.3 başarım
Braga et al.[14]	Tespit	Makine Öğrenmesi-Sınıflandırma	Paket sayısı Veri miktarı Akış süresi Karşılık oranı Port farklılığı	%98.57-99.11 başarım.
Hu et al.[17]	Tespit+Engelleme	Akış yolu izleme	Kaynak IP Kaynak Port Protokol	9.01 ms içerisinde tespit ve engelleme

Çizelge 3.1. (devam) Tez kapsamında incelenen kaynakların karşılaştırması

Buragohain & Mehdi[18]	Tespit	Ölçme+Eşik Değeri	Akış miktarı Akış süresi	Flood trafiği 60-95 Mbps arasında tutulmuştur.
Dao et al.[19]	Tespit	Ölçme+Eşik Değeri	Bağlantı sayısı Bağlantıdaki paket sayısı	Akış girdisi ve Packet_in sayısı %50 oranında azalma.
Fichera et al.[9]	Tespit+Engelleme	Ölçme+Eşik Değeri	SYN paket sayısı	SYN isteği sayısı %96.5 oranında azalma
Qian et al. [20]	Tespit+Engelleme	Ölçme+Eşik Değeri	Paket sayısı Veri miktarı	Akış tablosu yoğunluğu %89 oranında azalma.
Dao et al.[21]	Tespit+Engelleme	Ölçme+Eşik Değeri	Packet_in sayısı Veri miktarı	SBF, DFA ve POX denetçisisden daha başarılı.
Wang et al. [22]	Tespit+Engelleme	Ölçme+Eşik Değeri	Packet_in sayısı	Veri iletiminde %99,72 iyileşme.
Carvalho et al. [23]	Tespit	Entropi ölçümü	IP adres dağılımı	-
Zhang et al.[10]	Etki azaltma	Kuyruk+ Önceliklendirme	-	Denetçiye göre daha başarılı RTT
Kuerban et al.[11]	Etki azaltma	Bant genişliği kısıtlama	Veri miktarı	Bant genişliği kullanımı 7 Mb iken 3.5 Mbps düşüş
Piedrahita et al.[25]	Etki azaltma	Bant genişliği kısıtlama	Veri miktarı	Saldırı etkisi %50 oranında azalma
Yau et al.[26]	Etki azaltma	Bant genişliği kısıtlama.(%50)	Veri miktarı	Yaklaşık %50 oranında başarılı iletim.

Çizelge 3.1. (devam) Tez kapsamında incelenen kaynakların karşılaştırması

Wang et al. [28]	Etki azaltma	Bant genişliği kısıtlama.	Veri miktarı	-
Xie et al. [29]	Etki azaltma	Engelleme+Bant genişliği kısıtlama.	Veri miktarı	-
Saifullah [27]	Etki azaltma	Bant genişliği kısıtlama.(ağırlıklı)	Veri miktarı	-
Yao et al.[31]	Önleme	Kaynak doğrulama	IP adresi	SAVI yöntemine göre daha yüksek başarım.
Shin et al.[8]	Önleme	Entegre Güvenlik Duvarı	SYN paket içeriği	Tıkanıklık durumu yerine %0.5 oranında yavaşlama
Wang et al.[32]	Önleme	Örnekleme veri tabanı	Packet_in sayısı Anomali tespiti	-
Wang et al.[33]	Önleme	Örnekleme veri tabanı	Packet_in mesajı	İletimin %50 düşmesi yerine %2 düşüş

Literatürdeki çalışmalar incelendiğinde, geleneksel ağ altyapılarındaki DoS saldırılarının YTA'da da uygulanabildiği görülmektedir. YTA'da anahtarların akış tabloları ile önbellekleri ve denetçinin işlem gücünü hedef alacak saldırıların ağda ciddi zaafiyete yol açabileceği görülmektedir. Bu nedenle anahtarların akış tabloları, önbellekleri ve denetçiyi korumaya alacak yöntemlerin geliştirilmesinin önemli bir konu olduğu değerlendirilmektedir.

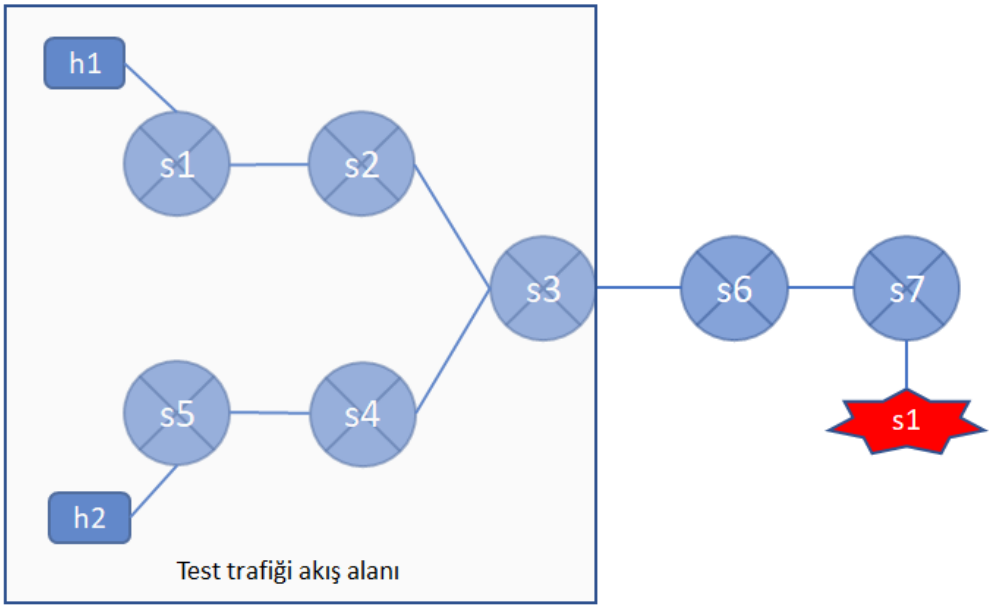
Hizmet Dışı Bırakma Saldırıları ve uygulanabilecek karşı yöntemler YTA temelinde incelendiğinde, geleneksel ağ altyapılarında DoS saldırılarına karşı uygulanan tespit, engelleme ve mücadele yöntemlerinin (Saldırı tespit sistemleri vb.) aynı biçimiyle YTA altyapısında da kullanılabildikleri ve bu alanda yapılan çalışmaların belli bir olgunluk seviyesine ulaştığı görülmektedir. Bu bakımdan DoS saldırılarının YTA üzerindeki etkileri ve karşı tedbirler bakımından yapılacak bir çalışmada, YTA'nın getirdiği yeni özelliklerin kullanılmasının literatüre yenilik getireceği değerlendirilmiştir. Bu bakımdan YTA bileşenlerini kullanarak yapılacak tespit, önleme ve mücadele gibi yöntemlerin literatüre sağlayacağı katkı bakımından daha önemli olduğu sonucuna varılmış ve araştırmaların bu yönde sürdürülmesi kararlaştırılmıştır.

4. ÖNERİLEN YÖNTEM

Bölüm 3'te belirtildiği üzere, YTA bileşenleri kullanılarak gerçekleştirilen bir tespit ve önleme yönteminin literatürdeki diğer çalışmalardan farklılık teşkil ederek literatüre katkı sağlayabileceği düşünülmüştür. Bu bölümde öncelikle DoS saldırılarının YTA üzerindeki etkileri araştırılmış ve geleneksel ağlardan farklı olarak hangi YTA bileşenlerini ne şekilde etkilediği ortaya konulmuştur. Sonrasında ise saldırı etkisinin azaltılması veya saldırının tamamen engellenmesi için uygulanacak yöntem detaylı olarak açıklanmıştır.

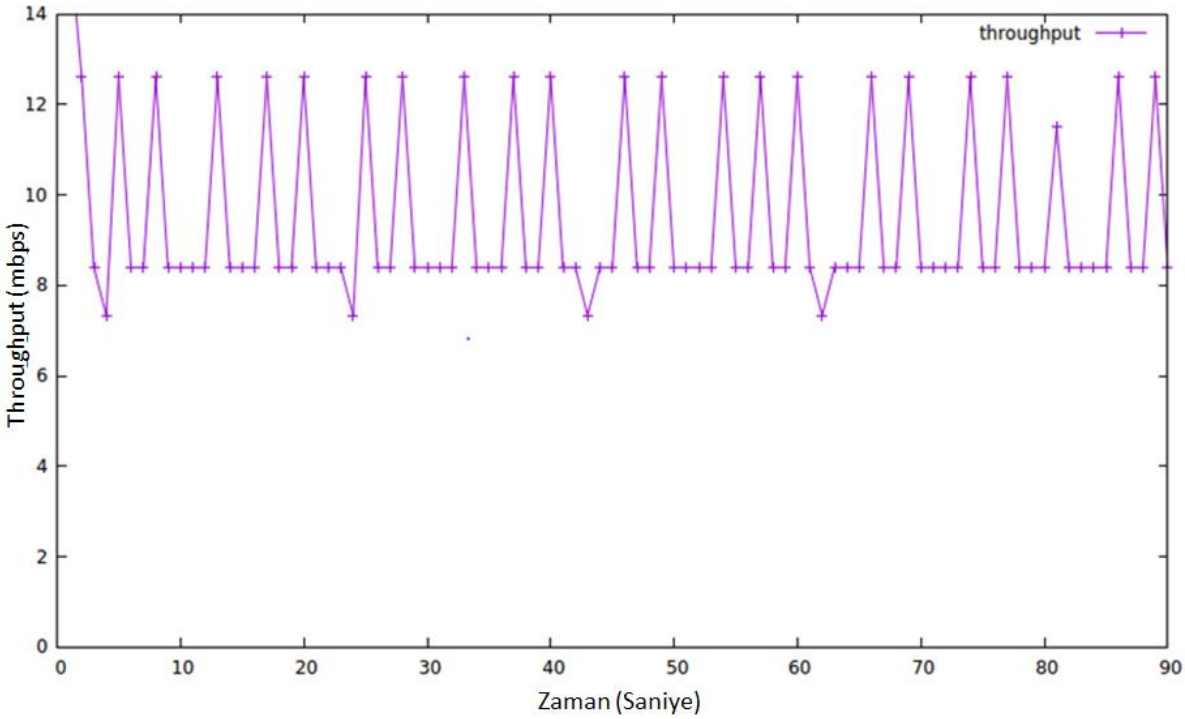
4.1. DoS Saldırılarının YTA'daki Etkileri

YTA temelindeki DoS saldırılarının en büyük etkisi, kontrolcü üzerinde fazla sayıda packet_in mesajı yaratılıyor olması ve kontrolcünün gereksiz yere meşgul edilmesidir. Bu etkilerin daha iyi anlaşılabilmesi açısından test ortamında saldırı durumu ve normal durumda iki farklı düğüm arasındaki veri iletim hızı (throughput) ölçülmüştür. Saldırı tipi olarak literatürde sıklıkla kullanılan ICMP flooding saldırı yöntemi seçilmiştir. Etkilerin ölçülmesi için oluşturulan topoloji Şekil 4.1'deki gibidir. Topolojide h1 ve h2 normal düğümleri temsil etmektedir. İki düğüm arasındaki trafik literatürde sıkça kullanılan iperf yazılımı kullanılarak oluşturulmuştur [34]. h1 ve h2 düğümleri arasındaki trafik, şekilde de görüldüğü üzere s1-s2-s3-s4-s5 rotasını takip etmektedir. Saldırgan, hedef IP adreslerini rastgele biçimde seçtiğinden bu istekler kontrolcüye iletilmekte, kontrolcü ise hedef adres h1 ve h2 düğümleri değil ise ağda böyle bir düğüm olmadığına dair S7 anahtarına cevap vermektedir. Bu nedenle normal düğümler arasındaki veri akışı büyük ölçüde ICMP paketleri ile çakışmayacak biçimde iletilmektedir.

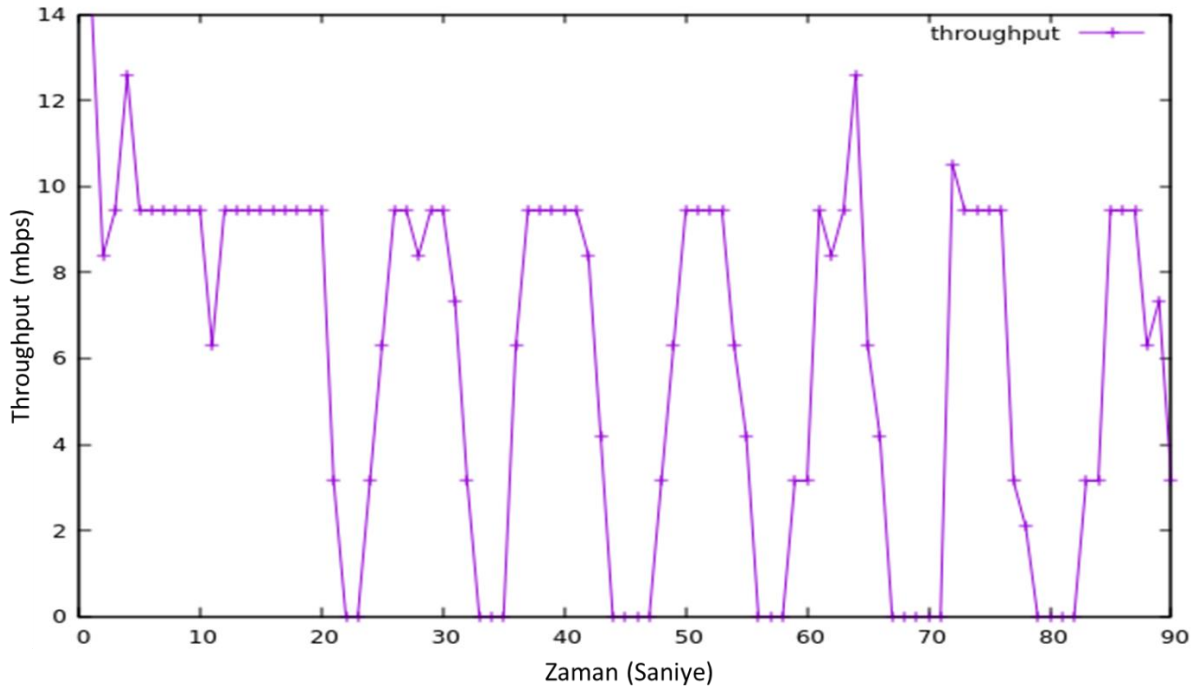


Şekil 4.1.Deney topolojisi

Ölçüm işleminin gerçekleştirildiği topolojide saldırgan düğüm ve normal düğümler birbirinden ayrı noktalara konuşlandırılmıştır. Bunun nedeni, saldırganın gerçekleştirdiği flooding saldırısının normal düğümlerin veri iletimini etkilememesinin istenmesidir. Bunun yerine saldırgan, iletim ortamında en düşük seviyede engellemeyle kontrolcü üzerine odaklanmakta ve kontrolcüyü hizmet dışı bırakmaya çalışmaktadır.



Şekil 4.2. Saldırgan bulunmadığı durumda iletilen veri miktarı



Şekil 4.3. Ağda bir saldırgan bulunduğu durumda iletilen veri miktarı

Şekil 4.2. ve Şekil 4.3. incelendiğinde iki durum göze çarpmaktadır. İletilen veri miktarının üst seviyesi iki durumda da 8-10 mbps aralığında birbirine yakın durumdadır. Normal durumda iletilen veri miktarı 8 mbps seviyesinin biraz üzerinde iken zaman zaman yukarı yönlü hareketler göze çarpmaktadır. Ağda bir adet saldırganın bulunduğu topolojide ise veri miktarı 9,5 mbps düzeyindedir ancak bunun üzerinde bir hareket fazla mevcut değildir. İki durumdaki üst seviyedeki veri iletim miktarının birbirine yakın olmasının nedeni saldırgan ve normal düğümlerin konumları itibarıyla birbirinin veri iletimini en az düzeyde etkiliyor olmalarıdır.

Göze çarpan ikinci husus ise zaman içerisinde iletilen veri miktarındaki kesintilerdir. Şekil 4.2.'de iletilen veri miktarının seviyesinde çok kısa süreli düşüşler görülmektedir. Bu düşüşler, anahtarlardaki akış girdi süresine bağlı olarak değişmektedir. Deney ortamında kesin zaman aşımı ve boşta kalma zaman aşımları eşit şekilde 10 saniye olarak belirlenmiştir. Şekil 4.2.'deki grafikte de görüldüğü üzere 10 saniyelik aralıklarla veri iletiminde kısa süreli kesintiler meydana gelmiştir. Şekil 4.3.'e bakıldığında bu aralıkların giderek uzayan biçimde devam ettiği görülmektedir. Bu durum ICMP flooding saldırısının kontrolcüyü meşgul etmesiyle meydana gelen bir durumdur. Başlangıçta kesintiler kısa

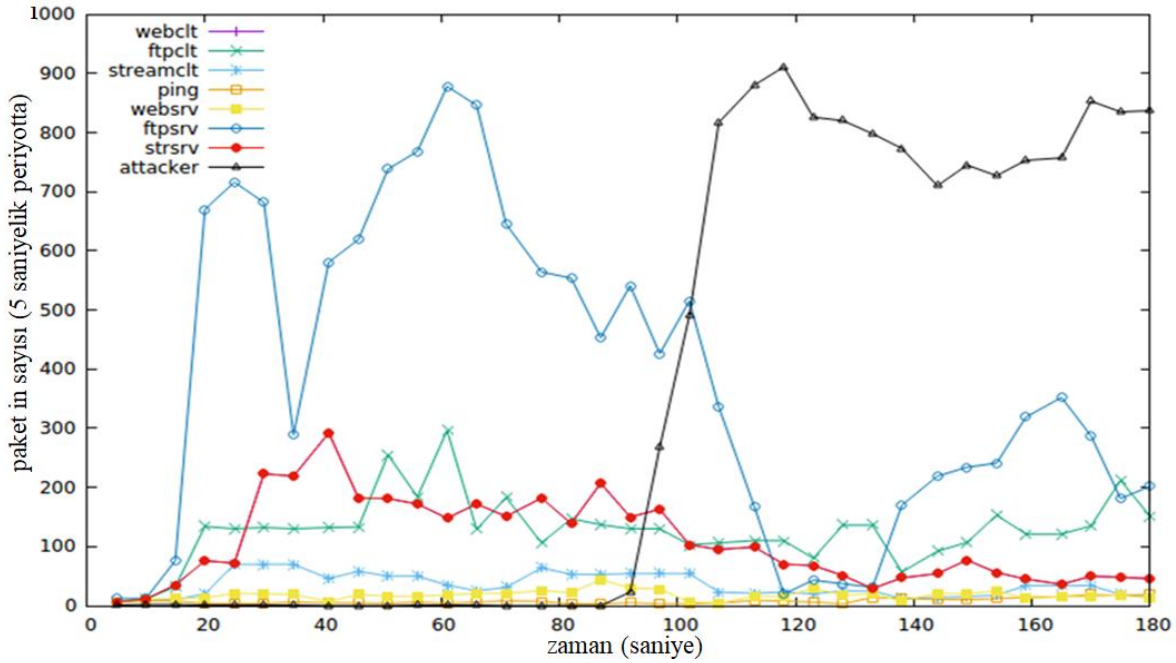
süreliyken, packet_in mesajlarının kontrolcüde birikmesiyle kontrolcü normal düğümlere cevap veremez duruma gelmektedir.

Yukarıda özetlenen problem, packet_in mesaj sayısındaki artışın YTA'da kontrolcü üzerinde ciddi bir zafiyete yol açabildiğini göstermektedir. Problemin etkisi ve YTA'nın ana bileşenlerinden olan packet_in mesajları kullanılarak yapılıyor olması sebebiyle, bu tez çalışmasında kullanılan DoS tespit mekanizması, packet_in sayılarındaki anomaliyi tespit etmek üzere kurulmuştur. Bu yöntem, herhangi bir düğümün gönderdiği packet_in mesaj sayılarına bağlı olarak belirlenebileceği gibi, daha farklı parametreler ile de desteklenebilir. Bunun için öncelikle normal trafik akışı olduğu zamanda packet_in sayılarının ve varsa birlikte kullanıldığı parametrelerin ölçülmesi ve sonrasında ise saldırı zamanında bu değerdeki değişimlerin normal trafik akış zamanındaki durumuyla karşılaştırılması gerekmektedir.

Bu tez çalışmasında geliştirilen tespit mekanizması, normal kullanıcılar ile saldırganlar arasında packet_in sayısına bağlı bir farklılık aramaktadır. Farklılığın hangi ölçüm yöntemiyle tespit edildiği ve neden packet_in / iletilen paket sayısı (tx paket) oranının kullanıldığı devam eden başlıklarda detaylı olarak anlatılmıştır.

4.2. Anomali Tespiti İçin Doğru Özelliğin Seçimi

DoS saldırılarının tespitine kontrolcünün bakış açısından yaklaşmamız sebebiyle, öncelikle bu saldırıların yalnızca packet_in sayıları kullanılarak tespit edilip edilemeyeceği sorgulanmıştır. Şekil 4.4.'te, http, ftp, video streaming, icmp trafiklerinin bulunduğu ortamda; bu trafikleri üreten düğümlerin ve DoS saldırısı gerçekleştiren düğümlerin yarattığı packet_in mesajı sayılarının grafiği gösterilmiştir.

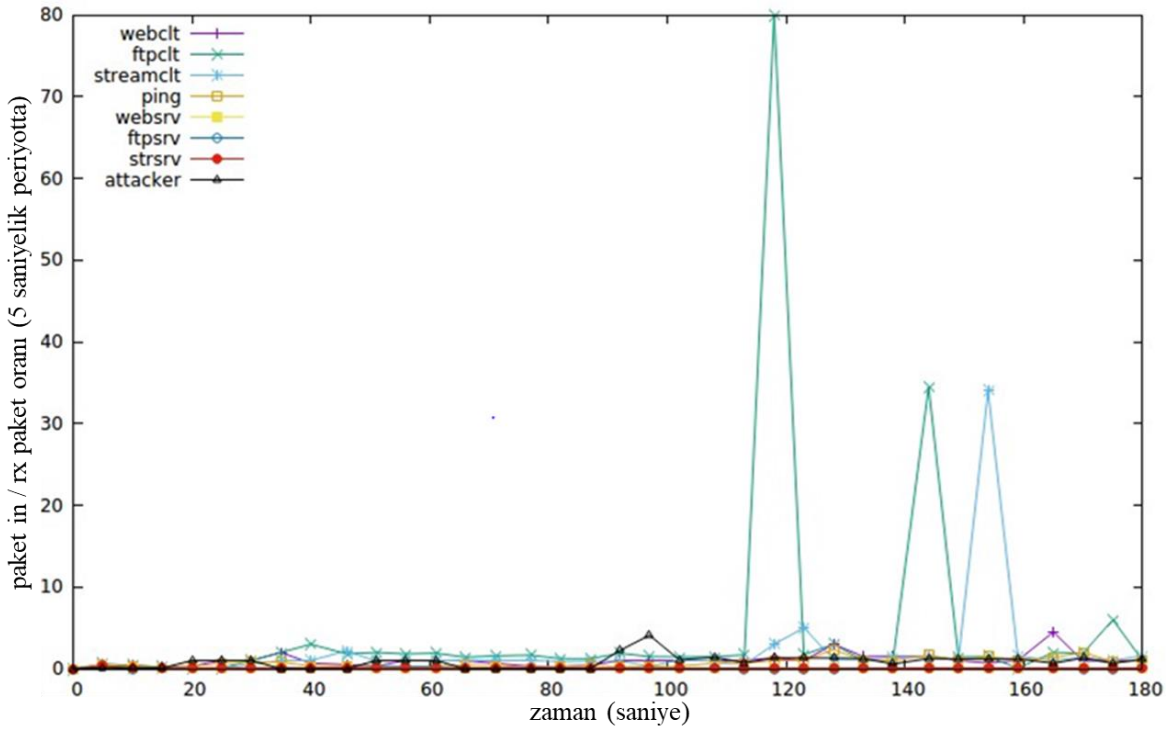


Şekil 4.4. Ağ benzetimindeki packet_in sayılarının değişimi

Şekil 4.4.'te de görüldüğü üzere, ağ trafiğine bağlı olarak, normal düğümlerin de kontrolcüye fazlaca packet_in mesajı gelmesine sebep olabildikleri görülmektedir. Yukarıdaki örnekte dosya transfer sunucusunun, kısa süre içerisinde farklı noktalara veri transfer etme ihtiyacı sebebiyle saldırgan düğüme benzer bir davranış biçimi gösterdiği gözlenmiştir.

Yukarıdaki senaryoyu yorumlamak gerekirse; anahtarlar, kontrolcü tarafından kendilerine gönderilen cevaplar geciktikçe veya kendilerine ulaşmadığında, bu cevapları beklemek yerine tekrar tekrar packet_in mesajı gönderme yoluna gitmektedir. Bu senaryoya bakıldığında, anahtarların, üzerlerindeki normal düğümlerin fazla veri transferi ihtiyaçları olduğunda da kontrolcüye yüksek sayıda talep gönderdiklerinden, packet_in mesajlarının tek başına DoS saldırılarını ayırt etmek için uygun bir parametre olamayacağı görülmüştür.

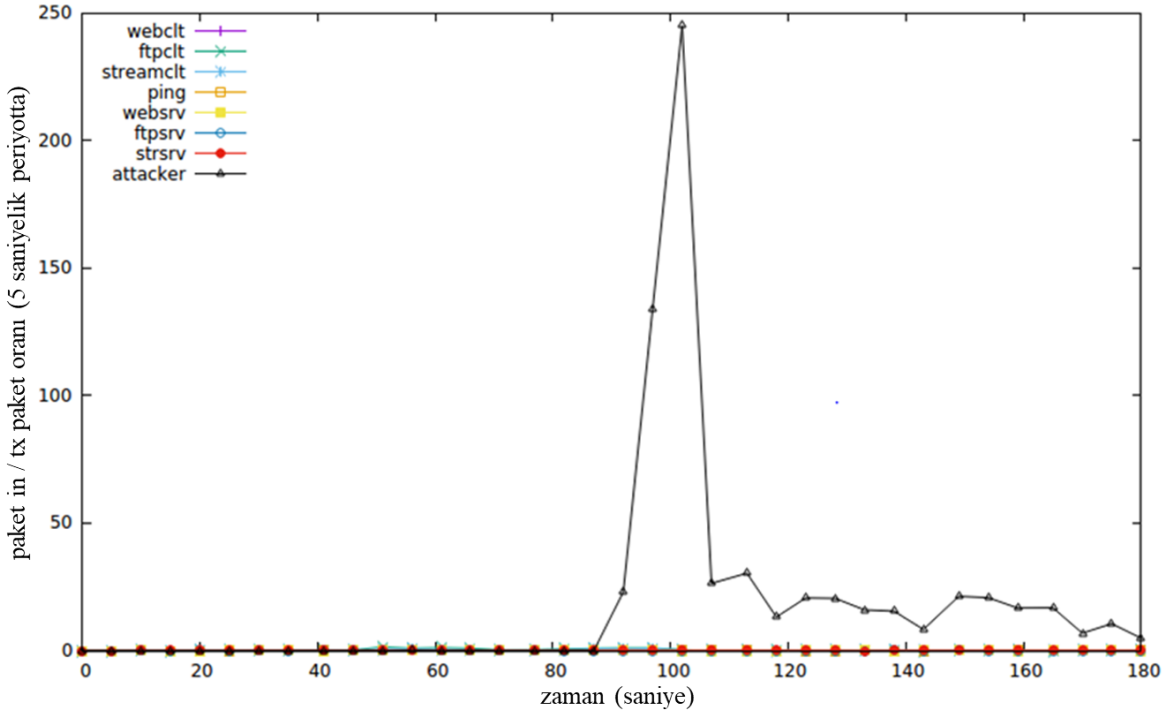
Packet_in mesajı sayısının tek başına bir belirteç olamayacağının görülmesi üzerine bu değer ile birlikte farklı parametrelerin kullanılarak bir tespit yöntemi geliştirilip geliştirilemeyeceği araştırılmıştır. Şekil 4.5.'de düğümlere ait packet_in sayısının alınan paket sayısına (rx) bölünmesi sonucunda ortaya çıkan durum gösterilmektedir.



Şekil 4.5. Ağ benzetimindeki packet_in / rx paket oranının değişimi

Sonuçlara bakıldığında, fazlaca veri indirmesi yapan düğümlerin saldırganlardan daha yüksek değerlere ulaşabildiği gözlenmiştir. Örnekte dosya ve video transfer istemcilerinin packet_in / rx paket değerlerinin diğer düğümlerden ve saldırganlardan daha yüksek noktalara ulaştığı görülmektedir.

Şekil 4.4. ve Şekil 4.5'te belirtilen iki örnek, veri transfer ihtiyacı arttığında sadece packet_in sayısına veya packet_in / rx paket oranına göre bir ayrımın mümkün olmadığını göstermiştir. Şekil 4.6.'da ise iletilen paket sayısı (tx) kullanılarak ulaşılan packet_in / tx paket oranına ait sonuçlar gösterilmiştir.



Şekil 4.6. Ağ benzetimindeki packet_in / tx paket oranının değişimi

Sonuçlara bakıldığında packet_in / tx paket oranının saldırgan düğümü diğer düğümlerden ayırt edebildiği görülmüştür. Bu grafiğin daha detaylı biçimde yorumlanabilmesi amacıyla değişen saldırgan sayıları karşısında packet_in / tx paket oranının davranışı Çizelge 4.1.'de detaylı biçimde özetlenmiştir.

Çizelge 4.1.'de, normal düğüm değerleri arasındaki maksimum aralık olarak ifade edilen değer, iki normal düğüm arasında packet_in / tx paket değeri oranının maksimum hangi seviyede olabileceğini belirtmektedir. Bu duruma bir örnek vermek gerekirse:

Örnek:

Belli bir zaman periyodunda,

Düğüm 1: 5 packet_in mesajı gönderiyor ve buna karşılık 10 paket transfer ediyor ise packet_in / tx paket oranı 1/2'dir

Düğüm 2: 5 packet_in mesajı gönderiyor ve karşılığında 1000 paket transfer ediyor ise packet_in / tx paket oranı 1/200'dür.

Buradan Düğüm 1 ve Düğüm 2 arasındaki oran 100 olarak bulunur.

Bu değerin ölçülmesindeki amaç, herhangi iki normal düğüm arasındaki veri transferi miktarından kaynaklanan farkın, bu düğümlerin birbirleri arasındaki packet_in /tx paket oranının hatalı pozitif bir biçimde saldırı olarak tanımlanıp tanımlanmayacağını görmektir. Çizelge 4.1.'de görüldüğü üzere bu oran en yüksek seviyede 24,43 olarak ölçülmüştür. Normal düğümler arasındaki ortalama uzaklık ise, tüm normal düğümler arasında ölçülen aralıkların aritmetik ortalamasını ifade etmektedir.

Çizelge 4.1. Çeşitli saldırgan sayısına bağlı olarak packet_in / tx paket değişimi

Saldırgan Sayısı	Normal Düğüm Değerleri Arası Max. Aralık	Normal Düğümler Arası Ortalama Aralık	Saldırgan – Normal Düğümler Arası Ortalama Aralık
0	8.2	3.25	-
1	12.2	3.5	642
2	21	3.8	181
3	19	3.9	176
4	19	3.9	79
5	20	4.6	73
10	24.43	4.8	18.04
15	16.9	4.06	14.25
20	19.16	3.71	18.62
25	18.99	4.14	15.43

Saldırganlar ve normal düğümler arasındaki ortalama aralık ise Şekil 4.6.'da görülen grafiğin sayısal olarak ifade edilmiş halidir. Bu değer, saldırganlar arasındaki en düşük ve normal düğümler arasındaki en yüksek packet_in / tx paket değeri oranını ortaya koymaktadır. Bu değerin ölçülmesindeki amaç, saldırgan sayısından bağımsız olarak saldırganlar ile normal düğümler arasında bir ayrım noktasının belirlenip belirlenemeyeceğinin görülmesidir. Çizelge 4.1.'deki değerlere bakıldığında, 1 saldırganlı durumda, en düşük paket_in/ tx paket değerine sahip saldırgan ile en yüksek packet_in / tx paket değerine sahip normal düğüm arasındaki oranın 642 olduğu görülmüştür. Bu oranın

normal düğümler arasında en fazla 24,43 olarak ölçülmesi sebebiyle oldukça ayırt edici bir değere sahip olduğu görülmektedir. Bu değer saldırgan sayısı arttıkça düşmektedir. Bunun nedeni ise saldırganların anahtarlar ve kontrolcü arasındaki iletim kanalını yoğunlaştırarak sınır kapasitesine ulaştırmalarıdır. 5 saldırgana kadar ayırt ediciliğin yüksek olduğu, 10 saldırgandan itibaren ise ayırt etme imkânının mümkün olmadığı görülmüştür.

Çizelge 4.1.'deki değerler yorumlandığında packet_in / tx paket oranının saldırganları ayırt etmede bir parametre olarak kullanılabileceği değerlendirilmiş ve bu yöntemin geliştirilmesine çalışılmıştır. Bu amaçla sonraki başlıklarda detayları verilmiş olan Jain's Index ve entropi değerleri packet_in / tx değeri üzerinden hesaplanmıştır.

Yukarıda da bahsedildiği üzere, herhangi bir eşik değerinin tespit edilebilmesi için öncelikle normal kullanım durumunda (ağda bir saldırgan olmadığında) tespit mekanizmasında kullanılan packet_in / tx paket oranının nasıl seyrettiğinin bilinmesi gerekecektir. Bu durum ağdaki kullanıcıların davranışlarına göre, günün çeşitli zaman dilimlerine göre, kullanılan yazılımların özelliklerine göre değişebilmektedir. Bu nedenle bu değerlerdeki değişiminin genel bir fonksiyon ile ifade edilerek karşılaştırılmasının faydalı olabileceği değerlendirilmiştir.

Bu çeşit bir fonksiyonun kullanılması araştırılırken, saldırganların davranışının ağda bir adaletsizlik durumu yaratabileceği değerlendirilmiş ve bu nedenle adil dağıtım ölçüm yöntemlerinin ağ üzerinde uygulanarak sonuçlarının görülmesine karar verilmiştir. Bu nedenle literatürde sık karşılaşılan Jain's Index ve entropi yöntemleri üzerinde araştırma yapılmıştır.

4.3. Jain's Index

Jain's Index, ilk kez R. Jain tarafından ortaya atılmış bir adil dağıtım değerlendirme yöntemidir [35]. Bu yöntem, sayısal bir değerlendirmeyle sistemde paylaşılan kaynakların adil olarak dağıtılıp dağıtılmadığına dair bir ölçü sunmaktadır. Jain's Index, aşağıdaki gereksinimleri karşılamak üzere geliştirilmiştir:

- Popülasyon bağımsızlığı: Endeks değeri, sonlu veya sonsuz herhangi bir sayıda istemci arasında kullanılabilmektedir.

- Boyut ve ölçüm biriminden (metrik) bağımsızlık: Dağıtılan kaynak sayısının ve ölçüm biriminin endeks değeri üzerinde bir etkisi olmamalıdır.
- Sınırlılık: Endeks 0-1 aralığında bir değer ile adilliği değerlendirmelidir. Değer 1'e yaklaştıkça, sistemin daha adil olduğu anlamına gelmelidir.
- Süreklilik: Dağıtımdaki herhangi bir değişiklik, endeks değeri üzerinde bir değişim göstermelidir.

Yukarıdaki gereksinimler bağlamında, Jain's Index, Eş. 4.1 ile tanımlanmıştır:

$$f(x) = \frac{\left[\sum_{i=1}^{n} x_i\right]^2}{n \sum_{i=1}^{n} {x_i}^2} \tag{4.1}$$

Eş. 4.1 ile tanımlanan sistemde x değeri adil dağıtımı hedeflenen kaynağı ifade etmektedir. X değeri uygulamadan uygulamaya farklı kaynak çeşitlerini temsil edebilmektedir. Örnek olarak, bilgisayar ağlarında aşağıdaki birimler, Jain's Index'in değerlendirme konusu olabilir [35]:

- Cevap verme zamanı
- İletilen veri miktarı
- Harcanan güç miktarı
- Taleplerin karşılanma oranı

Yukarıdaki örnekler birçok alanda çeşitlendirilebilir ve bilgisayar ağlarının konusu dışında da kullanılabilir. Bu tez çalışmasında da farklı bir ölçüm birimi kullanılarak adil dağıtım durumu değerlendirilmeye çalışılmıştır.

Eş. 4.1'deki fonksiyon ile tanımlanan sistem, 0-1 aralığında sonsuz sayıda değer alabilmekte, en adil durumda 1 değerini, en adaletsiz durumda ise 0 değerini almakta ve yukarıda belirtilen bütün gereksinimleri karşılamaktadır. Bu yöntemin nasıl işlediği R. Jain'in çalışmasında [35] verdiği örnekler ile açıklanacak olursa:

Örnek:

20 TL'nin 100 kişi arasında dağıtıldığı bir sistem düşünülecek olursa,

Her bir kişiye 20 kuruş verildiğinde:

$x_i = 0{,}2, i = 1{,}2, \ldots\ldots, 100\ olacaktır.$

Buna göre Eş. 4.1'deki yerlerine yazıldığında sistemin adalet değeri 1.0 olacaktır ve bu da tamamen adil bir sistemi işaret etmektedir.

Yukarıdakinden farklı şekilde bir dağıtım yöntemi güdüldüğünü varsayacak olursak, örneğin 10 istemciye 2'şer TL verilip diğerlerine hiç dağıtım yapılmadığında,

$$x_i = \begin{cases} 0 & i = 11{,}12{,}13{,}\ldots\ldots\ldots\ldots{,}100 \\ 2 & i = 1{,}2{,}3{,}\ldots\ldots\ldots\ldots\ldots\ldots{,}10 \end{cases} olacaktır.$$

Yukarıdaki değerler formül Eş. 4.1'deki yerlerine yazıldığında sistemin adalet değeri 0.10 olarak değerlendirilmiştir. Bu alanda literatürde yapılan çalışmaları örneklendirmek gerekirse:

Hussain ve arkadaşları [36] VANET üzerinde yaptıkları kanal erişim mekanizmasına dair çalışmalarında, geliştirdikleri sistemin efektifliğini ölçmek için Jain's Index kullanmışlardır. İletilen veri miktarı esas alınarak sistemin adil olup olmadığı gösterilmiştir. Arianpoo ve Leung [37]'de benzer biçimde, CMT-SCTP protokolünün efektifliğini ölçmek için Jain's Index'ı kullanmışlardır. Amer, Busson ve Lassous [38] ise kablosuz ağlarda erişim noktaları arasındaki kanal paylaşımının optimize edilmesine yönelik çalışmalarında, geliştirilen yöntemin efektifliğini Jains Index ile ölçmüşlerdir. Literatürde yukarıdaki çalışmalardan alınan sonuca göre Jain's Index, genellikle geliştirilen yöntemlerin başarılı olup olmadığını göstermek amacıyla kullanılmıştır. Ölçüm birimi olarak ise genellikle iletilen veri miktarının kullanıldığı görülmüştür.

4.4. Entropi

Bilgi entropisi ilk kez Shannon [39] tarafından ortaya konulmuş, sistemlerin düzensizliğini ifade etmek amacıyla geliştirilmiş matematiksel bir yöntemdir.

P kaynağının n adet tüketici arasında paylaştırılması, $P = (p_{1,}p_2, \ldots\ldots, p_n)$ olmak üzere, ve $0 \leq p_i \leq 1$ olacak şekilde,

$$p_i = \frac{x_i}{\sum_{i=1}^{n} x_i} \tag{4.2}$$

şeklinde ifade edilir. Buradan entropi değeri,

$$H(P) = \sum_{i=1}^{n} (p_i log_2 p_i) \tag{4.3}$$

ifadesiyle elde edilir.

Shannon'un entropi ölçümü, temelde adil dağıtım yöntemlerinden ziyade, sistemlerin düzensizliğini ölçmek amacıyla kullanılmaktadır ancak birçok çalışma, adil bir dağıtım durumunu temsil etmesinden dolayı entropiyi sistemlerin dağıtım adaletini ifade etmek amacıyla da kullanmaktadır. Entropide ifade edilen düzensizlik, genellikle rastgelelikle doğru yöndedir. Sistemin rastgeleliğinin artması ile birlikte istemcilere eşite yakın derecede kaynak dağıtılacağı varsayımı ile entropi değerinin artması beklenir.

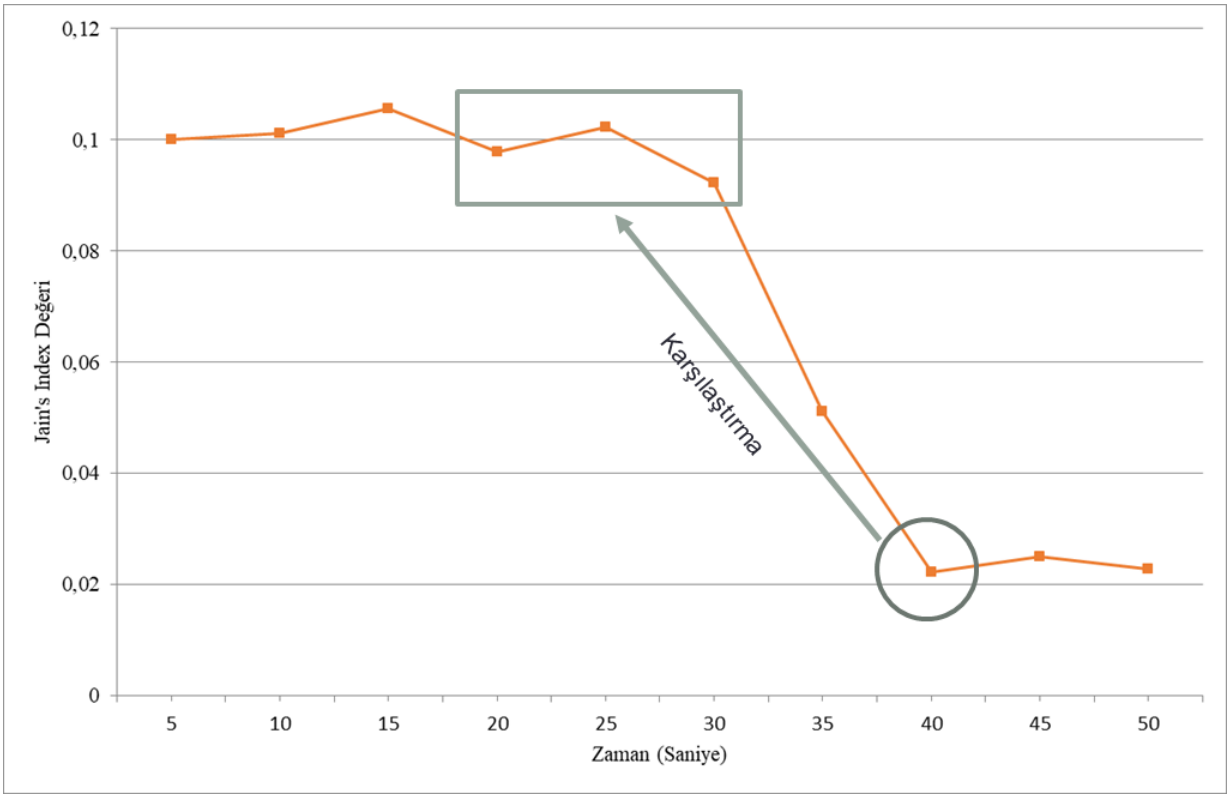
Ağ altyapılarında entropi ölçümünün anomali tespiti amacıyla birçok çalışmada kullanıldığı görülmüştür. Müter ve Asaj [40] taşıt ağlarındaki saldırılara yönelik entropi tabanlı bir tespit yöntemi geliştirmişler ve özetle flooding saldırısında araç içi iletilen sinyallerin entropi değerleri ölçülerek saldırı anındaki değişimlerinden tespit işlemi gerçekleştirilmiştir. Ma ve Chen [41]'de yaptıkları çalışmada DDoS saldırılarına karşı ağ trafiğinde entropi ölçümü kullarak gerçekleştirmişlerdir. Uygulamada kaynak ve hedef IP adreslerini kullanarak entropi değerleri hesaplanmış ve sonraki yöntemlere kaynak olarak kullanılmıştır.

4.5. Kullanılan DoS Tespit ve Önleme Algoritması

Yukarıdaki başlıklarda belirtilen yöntemler, sistemde ölçülen parametrelere göre adil bir dağıtım olup olmadığını sorgulamak için kullanılmıştır. Eğer adil bir dağıtım yok ise, kontrolcünün bu kez çalışma şeklini değiştirerek potansiyel saldırganları engelleme durumuna geçmesi gerekmektedir. Adil dağıtım olup olmadığına karar verme aşamasında ise entropiden daha iyi sonuç vermesi nedeniyle Jain's Index kullanılmıştır. Jain's Index ve entropi ölçümlerine ilişkin karşılaştırma sonuçları başlık 4.6.'da verilmiştir.

Algoritma 4.1., bu tez çalışmasında DoS saldırılarının tespit edilmesi ve önlenmesi için kurulan altyapıyı özetlemektedir. Kontrolcü üzerinde çalışan algoritmada, Jain's Index değerindeki değişim, belirlenen eşik değerinin üzerine çıktığında sistem en son durumu sonraki karşılaştırmalarda kullanmak üzere kaydetmekte ve engelleme aşamasına geçmektedir. Yapılan deneylerde, Jain's Index değerindeki değişim hesaplanırken, değişimin belli bir periyottaki ortalamasının alınmasının daha sağlıklı sonuç verdiği gözlenmiştir. Deneylerde ayrıca, mevcut Jain's Index değeri ile ondan önceki değer atlanılarak elde edilen üç ölçümün ortalamasından oluşan karşılaştırmanın da tespit doğruluğu açısından daha iyi sonuç verdiği görülmüştür. Bunun nedeni Jain's Index değerindeki kademeli düşüşün tespit doğruluğunu engellemesinin önüne geçmektir.

Bu altyapının daha iyi anlaşılması için Şekil 4.7.deki örnek incelenebilir. Şekil 4.7.'de örnek Jain's Index değerleri üzerinden tespit işleminin nasıl geçekleştirildiği gösterilmektedir. Görüldüğü üzere 40. saniyedeki tespit işlemi, 20,25 ve 30. saniyelerdeki ölçümlerin ortalamaları ile 40. saniyedeki ölçümün birbirine oranı alınarak yapılmaktadır. Böylelikle 35. saniyedeki ara değerin tespit doğruluğunu azaltmasının önüne geçilmektedir. Ayrıca karşılaştırma aşamasında üç faklı değerin ortalamasının alınarak anlık oluşabilecek dalgalanmaların önüne geçilmesi amaçlanmıştır.



Şekil 4.7. Jain’s Index değeri karşılaştırmasında uygulanan altyapı

Engelleme aşamasına geçildiğinde Jain’s Index değeri artık sadece saldırı öncesindeki durumu ile karşılaştırılmakta ve bu değerdeki değişim eşik değeri altına inene kadar sistem engelleme durumunda kalmaya devam etmektedir. Engelleme aşmasında kontrolcü, en yüksek packet_in sayısı / tx paket oranına sahip düğümleri sırasıyla engellemektedir. Bu engelleme işlemi sonrasında sistemin kararlı hale gelebilmesi için on saniye boyunca beklenilmekte ve Jain’s Index değerindeki değişim halen eşik değeri üstünde ise sıradaki düğüm için engelleme kuralı girilmektedir.

Algoritma: DoS Tespit ve Önleme Algoritması

```
1:  Düğüm Listesi: H
2:  Zaman Penceresi Değişim Listesi: W
3:  Düğümlere Ait Veri Kuyruk Listesi: X
4:  Jain's Index Değerleri Listesi: J[5]
5:  Mevcut Jain's Index Değeri: J_c
6:  Zaman Penceresi: T
7:  Tespit Eşik Değeri: Y
8:  Engelleme Modu = False
9:  while true do
10:     for h E H do
11:         X[h].push(packet_in/tx paket)
12:         W[h] = X[h][0] – X[h][T]
13:     end for
14:     Jain's Index Değerini Hesapla(W): J_c
15:     if Donduruldu(J) == false then
16          J.push (J_c)
17      end if
18      Ortalama(J[2], J[3], J[4]) / J_c: K
19      if K >= Y then
20          if Engelleme Modu == False then
21              Engelleme Modu = True
22              Dondur(J)
23          end if
24          Sırala W
25          W içerisindeki en yüksek değerli düğümü belirle:D
26          if en son drop kuralı üzerinden geçen süre >= 10 then
27              Drop kuralı oluştur(D)
28          end if
29      end if
30      bekle(5)
31  end while
```

Şekil 4.8. Kullanılan DoS tespit ve önleme mekanizmasının çalışma şekli

4.6. Jain’s Index ve Entropi Ölçüm Sonuçları

Bu başlıkta Algoritma 4.1.’de belirtilen tespit algoritması temel alınarak Jain’s Index ve entropi değerlerinin değişimleri verilmiş ve tespit aşamasında hangisinin daha iyi sonuç vereceğine karar verilmiştir. Bu amaçla, Bölüm 5’te detayları verilmiş olan topolojide 1, 2, 3, 4, 5, 10, 15, 20 ve 25 saldırganın bulunduğu ve 50 normal düğümün bulunduğu ortamda Jain’s Index ve entropi sonuçlarını gösteren deneyler gerçekleştirilmiştir. Çizelge 4.2.’de deney ortamında elde edilen Jain’s Index değerleri, Çizelge 4.3.’te aynı ortamdaki entropi ölçüm sonuçları, Çizelge 4.4.’te ise bu değerlerdeki değişim oranları özetlenmiştir. Sonuçlar değerlendirilirken odaklanılan noktanın Jain’s Index ve entropinin değerinden ziyade, değişim oranı olduğunun altının çizilmesinde fayda vardır. Zira deneylerde istemci profilinin ağ üzerindeki dağılımı, istemcilerin davranış biçimlerindeki değişiklikler gibi çeşitli etkenlerin bu değerleri farklı aralıklara çekebildiği görülmüştür ancak bu deneylerin hepsinde de saldırı öncesi ve sonrasındaki değerler arasında Çizelge 4.4.’tekine benzer değişimler mevcuttur.

Çizelge 4.2. Jains’s Index değerinin saldırgan sayısına bağlı değişimi

Saldırgan Sayısı	Anlık Değişim		Ortalama Değişim	
	Önce	Sonra	Önce	Sonra
1	0.161	0.024	0.274	0.015
2	0.104	0.04	0.260	0.028
3	0.132	0.062	0.274	0.042
4	0.140	0.073	0.287	0.049
5	0.128	0.071	0.260	0.061
10	0.143	0.124	0.286	0.159
15	0.153	0.111	0.289	0.202
20	0.134	0.095	0.292	0.251
25	0.131	0.092	0.279	0.257

Çizelge 4.2.'deki anlık değişim, Jain's Index'in saldırıdan hemen önce ve hemen sonraki değerlerini, ortalama değişim ise saldırıdan önce ve sonraki değerlerinin ortalamasını göstermektedir.

Çizelge 4.3.'te ise aynı topolojideki entropi değerleri gösterilmiştir. Sonuçlara bakıldığında, saldırı öncesi ve sonrasında Jain's Index ve entropi değerlerinin düşüş eğilimi gösterdiği, ancak saldırgan sayısı arttıkça saldırı öncesi ve sonrasındaki farkların da düştüğü gözlenmiştir.

Çizelge 4.3. Entropi değerinin saldırgan sayısına bağlı değişimi

Saldırgan Sayısı	Anlık Değişim		Ortalama Değişim	
	Önce	Sonra	Önce	Sonra
1	3.651	1.545	4.672	0.334
2	2.771	1.555	4.642	1.300
3	3.591	2.179	4.721	1.726
4	4.058	2.589	4.808	2.026
5	3.724	2.625	4.732	2.313
10	4.356	4.017	4.94	3.644
15	4.562	3.946	4.914	3.778
20	4.84	4.139	4.982	4.135
25	4.314	3.776	4.817	4.056

Jain's Index ve entropi değerleri benzer bir eğilim göstermelerine rağmen, tanımlanan problemde hangisinin daha başarılı olduğunun belirlenmesi amacıyla Çizelge 4.4.'te sonuçları belirtilen karşılaştırma gerçekleştirilmiştir. Sonuçlarda Jain's Index ve entropinin saldırı öncesi ve sonrasındaki değişimleri oransal olarak gösterilmiştir.

Çizelge 4.4. Jain’s Index ve entropi değerlerinin saldırı öncesi ve sonrasındaki değişiminin karşılaştırması

Saldırgan Sayısı	Anlık Değişim Oranı Önce / Sonra		Ortalama Değişim Oranı Önce/Sonra	
	Jain’s	Entropy	Jain’s	Entropy
1	6.708	2.363	18.267	13.988
2	2.600	1.782	9.286	3.571
3	2.129	1.648	6.524	2.735
4	1.918	1.567	5.857	2.373
5	1.803	1.419	4.262	2.046
10	1.153	1.084	1.799	1.356
15	1.378	1.156	1.431	1.301
20	1.411	1.169	1.163	1.205
25	1.424	1.142	1.086	1.188

Sonuçlara bakıldığında Jain’s Index’in saldırı sonrasında entropi değerine nazaran daha büyük bir düşüş sergilediği gözlenmektedir. Bu durum ise bu tezde ele alınan probleme yönelik olarak Jain’s Index’in saldırı durumunu tespit etmek için entropiden daha iyi bir yöntem olduğu sonucuna varılmasına neden olmuştur.

4.7. Kontrolcü Üzerinde Çalışan Tespit ve Önleme Sistemi

Kullanılacak tespit ve önleme mekanizmasında Jain’s Index’in daha başarılı olduğunun anlaşılmasının ardından, bu temelde kontrolcü üzerinde bir tespit ve önleme sistemi uygulanmıştır.

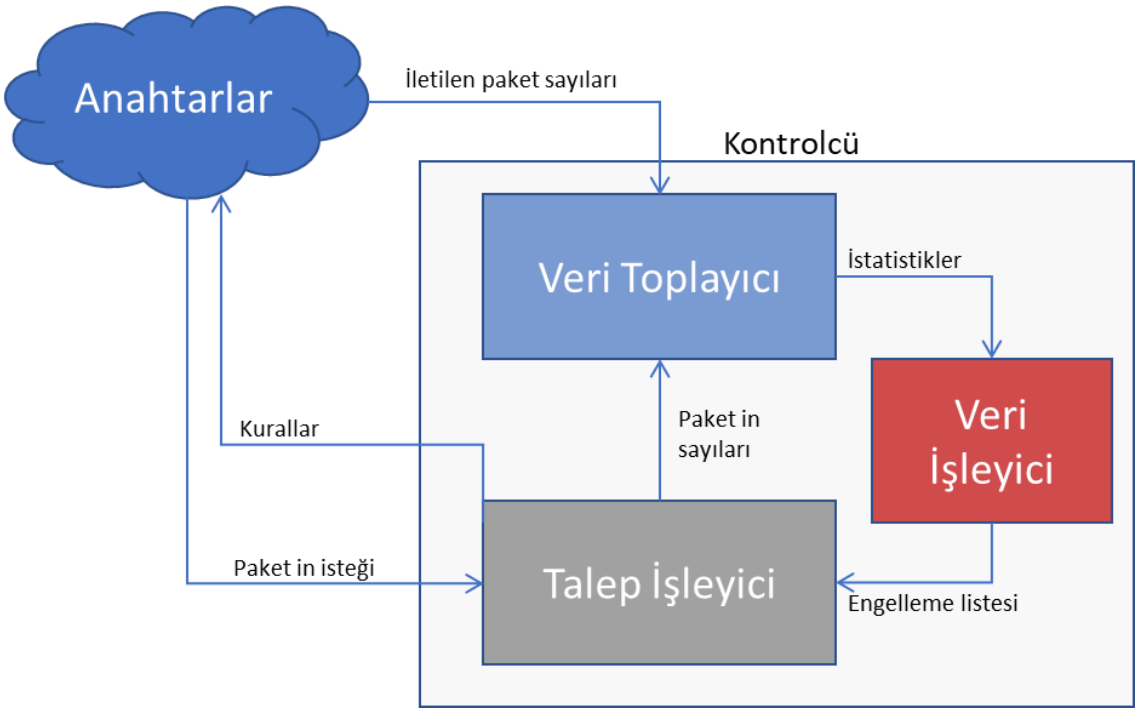
Kontrolcü üzerindeki sistemin birimlerinden olan Veri Toplayıcı, sistemden iki farklı veri toplamaktadır. Bunlardan ilki anahtarlardan gelen her bir arayüz başına iletilen paket sayılarıdır. Veri Toplayıcı birim anahtarlar üzerindeki istatistikleri toplayarak bunların uç

birimlere ait olup olmadığını kontrol etmekte ve öyle ise uç birim bazında tuttuğu kayıt tablosunu güncellemektedir. Güncelleme periyodunun anahtarlardan alınan istatistiklerinin fazla veri trafiği yaratmayacak ve senkronizasyon sorununa neden olmayacak kadar uzun, ancak verilerin güncelliğini sağlayacak kadar kısa olması gerekmektedir. Yapılan deneylerde 5 saniyelik periyodun bu bakımdan en uygun süre olduğu görülmüştür. Veri toplayıcı ikinci verisi olan packet_in sayısını ise Talep İşleyici birimden almaktadır.

Diğer bir birim olan Veri İşleyici, Veri Toplayıcı'dan aldığı verileri işleyerek Jain's Index değerini hesaplamaktadır. Hesaplama sonucunda eğer mevcut Jain's Index değerindeki değişim Çizelge 4.4.'te ortalama değişim kısmında belirtilen eşik değerlerinin üzerinde ise saldırgan tespit aşamasına geçmektedir. Saldırgan sayısı maksimum beş olduğundan, Çizelge 4.4 referans alınarak bu eşik değeri 4 olarak belirlenmiştir. Saldırgan tespit durumuna geçildikten sonra sistem tüm düğümleri packet_in / tx paket oranına göre sıralamakta ve en yüksek değere sahip düğümü engelleme listesine girmektedir. Bir düğüm engelleme listesine girdikten sonra o düğüme ait OpenFlow protokolüne ait drop kuralının anahtarlara gönderilmesi, bu kuralın anahtarlara ulaşması ve anahtarların bu kuralı işleme alması süresince bir miktar daha süre geçmesi gerektiğinden ve bu dönemde ağın saldırganın devre dışı bırakılmasıyla bir miktar dengeye kavuşması beklendiğinden, Veri İşleyici birim engelleme listesine bir düğümü kaydettikten sonra 10 saniye boyunca yeni bir engelleme kaydı girmemektedir.

Bir başka birim olan Talep İşleyicinin ise ana fonksiyonu kontrolcüye gelen packet_in mesajlarını işleme koymaktır. Bu mesajlara karşılık olarak yönlendirme bilgisinin anahtarlara iletilmesinden sorumludur. Saldırı durumunda çok fazla sayıda packet_in mesajı geldiğinden, kısa sürede müdahale edilmediği takdirde anahtarlar arasında yayılan saldırı trafiği, giderek artan sayıda packet_in mesajı iletilmesine sebep olmaktadır. Bu nedenle yapılan deneylerde, saldırganların engellenmesi işleminin Talep İşleyici içerisinde gerçekleştirilmesinin en iyi sonucu verdiği gözlenmiştir. Herhangi bir anahtara drop kuralı gönderildiğinde, Talep İşleyici bunu engelleme listesine kaydetmektedir. Bunun nedeni ise drop kuralı eklense dahi anahtarların sırada bekleyen packet_in mesajlarını göndermeye devam etmesidir. Böyle bir durumda eğer her bir isteğe karşılık drop kuralı eklenirse, drop kuralları da ağda aşırı derecede bir yoğunluğa neden olmaktadır. Bu nedenle yapılan deneylerde, bir anahtara bir düğümle ilgili bir drop kuralı gönderildiğinde, aynı düğüme ait aynı anahtardan gelen diğer packet_in mesajlarının göz ardı edilerek işleme alınmadan

bırakılmasının daha iyi sonuç verdiği görülmüş ve engelleme aşaması bu şekilde uygulanmıştır. Şekil 4.8.'de sistemi oluşturan bileşenler ve bunlar arasındaki ilişkiler özetlenmiştir.



Şekil 4.9. Kontrolcü üzerinde çalışan tespit ve engelleme sistemi

5. DENEYSEL SONUÇLAR

5.1. Ağ Altyapısı ve Benzetim Ortamı

Tez çalışmasında kullanılan benzetim ortamına ait özellikler bu bölümde tanımlanmıştır. Devam eden başlıklarda belirtilen benzetim yazılımlarının üzerinde çalıştığı donanım, Intel T5800 işlemci, 2 GB hafıza, 70 GB sabit disk kapasitesine sahiptir. Tüm yazılımlar Ubuntu 16.04 işletim sistemi üzerine kurulmuştur.

5.1.1. Ağ altyapısının benzetimi

Ağ altyapısının benzetiminin gerçekleştirilmesinde kullanılan ortam ve araçlara detaylı olarak değinilecek olursa; benzetim ortamı seçimi yapılırken aşağıdaki gereksinimler göz önüne alınmıştır:

- YTA desteğinin bulunması,
- Literatürde yaygın kullanımı bulunan güvenilir bir ortam olması,
- Açık kaynak olması; böylece herhangi bir uygulanabilirlik kısıtı getirmemesi,
- Çalışılacak donanıma uygun olması.

Yukarıdaki kriterler değerlendirildiğinde, Mininet [42] simülasyon ortamının, açık kaynak olması, literatürde çok sık kullanılmış olması, YTA üzerindeki araştırmalar için geliştirilmiş olması, donanım uygunluğu sebepleri nedeniyle tez çalışması için uygun bir geliştirme ortamı olduğu görülmüştür.

Mininet, günümüzde donanımlar ile oluşturulan gerçek ağ ortamının benzetiminin gerçekleştirilebilmesi için geliştirilmiş bir benzetim ortamıdır. Mininet temelde, üzerinde anahtarların, kontrolcünün ve diğer donanımların benzetiminin yapılacağı altyapıyı sağlamaktadır. Bu altyapı her biri linux işletim sisteminin bir kopyası olarak çalışan düğümler, anahtarlar, kontrolcüler ve bunların birbirine bağlandığı eklemlerden oluşur. Mininet bu eklemlerin bağlantı hızları, gecikmeleri, kayıpları gibi parametreleri yöneterek gerçek bir ağ altyapısının benzetimini gerçekleştirir. Mininet ağırlıklı olarak Python programlama dili kullanılarak geliştirilmiştir.

Örnek vermek gerekirse, tez çalışmasında uygulanan 16 anahtarlı topoloji mininet üzerinde Resim 5.1.'de belirtilen kod yapısı kullanılarak uygulanmıştır:

```
class Topoloji( Topo ):
        def __init__( self, anahtarSayisi):
                Topo.__init__( self )
                anahtarlar = {}

                for i in range(anahtarSayisi):
                        anahtar = self.addSwitch( 's' + str(i))
                        anahtarlar.setdefault(str(i),anahtar)

                self.addLink('s0','s3',bw=100, delay='5ms')
                self.addLink('s1','s3',bw=100, delay='5ms')
                self.addLink('s2','s3',bw=100, delay='5ms')
                self.addLink('s3','s4',bw=100, delay='5ms')
                self.addLink('s3','s14',bw=100, delay='5ms')
                self.addLink('s3','s15',bw=100, delay='5ms')
                self.addLink('s4','s5',bw=100, delay='5ms')
                self.addLink('s4','s6',bw=100, delay='5ms')
                self.addLink('s4','s7',bw=100, delay='5ms')
                self.addLink('s7','s8',bw=100, delay='5ms')
                self.addLink('s7','s9',bw=100, delay='5ms')
                self.addLink('s6','s10',bw=100, delay='5ms')
                self.addLink('s6','s11',bw=100, delay='5ms')
                self.addLink('s5','s12',bw=100, delay='5ms')
                self.addLink('s5','s13',bw=100, delay='5ms')

def agTest():
        anahtarSayisi = 16
        topo=Topoloji(anahtarSayisi)
        net=Mininet(topo=topo,controller=None, link=TCLink, autoSetMacs=True)
        c0 = RemoteController('c0',ip='127.0.0.1',port=6633,bw=100, delay='5ms')
        net.addController("c0",controller=RemoteController,ip='127.0.0.1',port=6633)
        net.start()


if __name__=='__main__':
        agTest()
```

Resim 5.1. Mininet ortamında örnek bir topoloji oluşturma işlemi

Yukarıdaki kod parçasında, sistem Topoloji sınıfı vasıtası ile bir ağ altyapısı oluşturmaktadır. Bu sınıf içerisinde sisteme parametre olarak girilen kadar anahtar eklemekte ve anahtarlar ise addLink() metodu vasıtasıyla, belirtilen bant genişliği ve gecikme süresi ile birbirlerine eklenmektedir. Tez çalışmasında, bu örnekte olduğu gibi her bir anahtar birbirine 100Mbps hızında ve 5ms gecikme ile sağlanmıştır. Anahtarlar birbirine bağlanırken rastgele bir yöntem de kullanmak mümkündür. Ancak tez çalışmasında yapılacak deneylerin birbiri ile karşılaştırılması gerektiğinden ve bu nedenle topolojinin her çalıştırma esnasında aynı kalması istendiğinden, anahtar bağlantıları manuel olarak gerçekleştirilmiştir.

Kontrolcü tanımlaması ise RemoteController sınıfı kullanılarak gerçekleştirilmiştir. Bu sınıf içerisinde kontrolcüye ait bant genişliği ve gecikme bilgileri parametre olarak verilirken, bunun yanında kontrolcüye ait kapı bilgisi de belirtilmektedir.

5.1.2. Kontrolcü seçimi

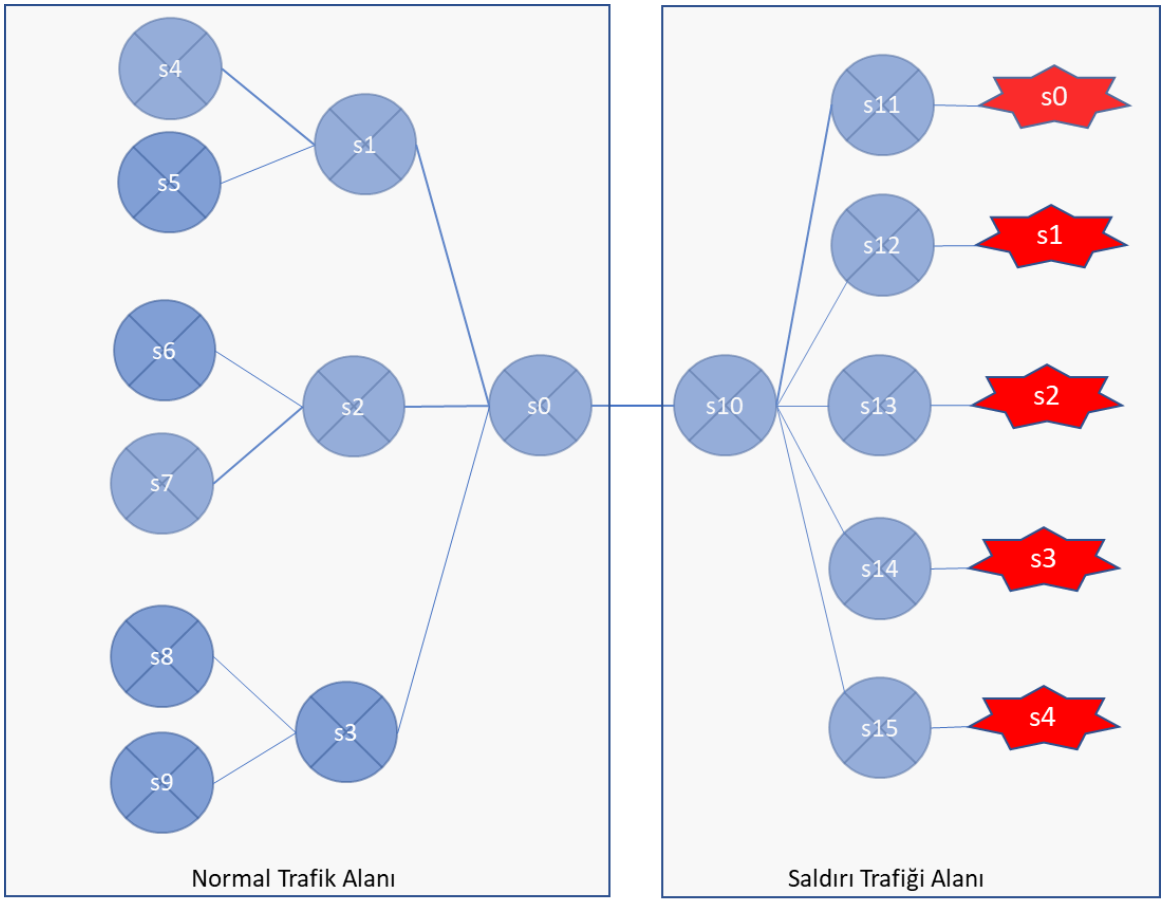
Ağ altyapısı benzetim ortamının seçimi kadar kontolcü seçimi de önemli bir unsur olarak görülmüştür. Literatürde kullanılan birçok kontrolcü versiyonu mevcuttur. Bazıları ticari iken, bazıları açık kaynak olan kontrolcü seçiminde ise aşağıdaki hususların değerlendirilmesi önemli görülmüştür:

- Açık kaynak olması,
- Geliştirmede kullanılacak programlama dili,
- Literatürde kullanılmış olması,
- Esnek ve hızlı olması.

Yukarıdaki gereksinimler değerlendirildiğinde, Ryu [43] kontrolcüsünün denenmesi uygun görülmüş, test sonuçlarının başarılı olması ve tez gereksinimlerini karşıladığının görülmesiyle bu kontrolcü ile deneyler gerçekleştirilmiştir. Tez çalışmasının ağırlıklı kısmının yürütüldüğü kısım da Ryu kontrolcüsü olmuştur. Çünkü YTA'da yönetimsel işlemlerin tek noktadan gerçekleştirilerek ağ yönetiminin kontrolcüye bırakılması sonucunda her tür faaliyet kontrolcü üzerinde denetlenebilir hale gelmiştir.

5.1.3. Topoloji

Mininet üzerinde çalışılan topoloji, 16 anahtar ve 50 adet düğümden meydana gelecek şekilde kurgulanmıştır. Bu 50 düğüm, normal çalışma özelliklerini göstermekte olup, kötücül özellik gösteren düğümler bu düğümler arasından seçilmemiştir. Deney ortamında uygulanan topoloji, Şekil 4.1.'dekine benzer, ancak onun biraz daha genişletilmiş bir versiyonu şeklinde uygulanmıştır. Deney topolojisi Şekil 5.1.'deki gibi kurulmuştur.



Şekil 5.1. Deneylerin gerçekleştirildiği ağ topolojisi

Şekil 5.1.'deki topoloji, deney ortamındaki anahtar dağılımını yansıtmaktadır. 50 adet normal düğüm ise normal trafik akış alanı içerisindeki anahtarlara, anahtar başına 5'er düğüm olacak şekilde dağıtılmıştır.

5.1.4. Ağ trafiği

Normal zamandaki akış trafiğinin benzetiminin gerçekleştirilebilmesi için mümkün olduğunca gerçeğe yakın bir ortam yaratılmasına çalışılmıştır. Mininet simülasyon ortamı, altyapısı sebebiyle işletim sistemi üzerinde her bir düğümün ayrı bir işletim sistemi kopyasına benzer biçimde çalışmasına izin vermektedir. Bu özellikten faydalanılarak her bir düğüme aşağıdaki gibi farklı roller tanımlanmıştır:

Dosya transfer sunucusu düğümler

Bu düğümler dosya istemcisi düğümlere dosya transferi yapmak amacıyla python programlama diline ait, MIT açık kaynak lisansı ile geliştirilmiş “pyftpdlib” kütüphanesini kullanmaktadır [44]. Ağ topolojisinin oluşturulduğu kaynak kodda, bir dosya sunucu basit şekilde Resim 5.2.’deki biçimde oluşturulmaktadır:

```
net['h26'].cmd\
('(nohup python -m pyftpdlib --directory=/home/nail \
--port=21 --write & > /dev/null &)')
ftpServers.append(26)
```

Resim 5.2. Dosya transfer sunucusunun aktif hale getirilmesi

Dosya transfer istemcisi düğümler

Dosya transfer istemci düğümleri, dosya sunucudan belirli zaman aralıklarında dosya kopyalama veya yazma işlemiş gerçekleştiren düğümlerdir. İstemci rolünde çalışan düğümler, bir bölümü Resim 5.3.’de belirtilen python programını çalıştırmaktadırlar:

```
while True:
        randNum = randint(0,1)

        randLoopCount = randint(1,MAXRANDOMLOOP)

        if randNum == 0:
                uploadFiles(randLoopCount)
        elif randNum == 1:
                downloadFiles(randLoopCount)
```

Resim 5.3. Dosya istemcilerinin işleyişi

Bu programın diğer bölümlerinde ftp protokolü üzerinden dosya indirme ve yükleme işlemleri yapılmaktadır.

Video transfer sunucusu düğümler

Bu düğümler video transfer istemcilerine kaynak sağlamaktadırlar.

```
command = '(nohup vlc -I dummy --repeat ~/Videos/1 --sout \
\'#standard{access=http,mux=asf,dst=10.0.0.46:8081}\' & > /dev/null &)'
net['h46'].cmd(command)
streamingServers.append(46)
```

Resim 5.4. Video transfer sunucusunun aktif hale getirilmesi

Resim 5.4.'teki kod parçacığı, ftp sunucu için yapılan işlemi aynı şekilde, bu kez hedef düğüm üzerinde video transfer sunuculuğu yapmaya yarayan Vlc Media Player programını çalıştırmaktadır [45].

Video transfer istemcisi düğümler

Bu düğümler ise video transfer sunucusu düğümden video indirme işlemi gerçekleştirmektedir.

```
def startStreaming():
        while True:
                randomVideoNumber = randint(1,int(videoCount))

                rep = os.system('nohup mplayer -user-agent mnnet-' \
                + localHost + '# -dumpfile /dev/null -dumpstream http://' \
                + remoteHost + ':808' + str(randomVideoNumber) + ' & > /dev/null &')

                time.sleep(randint(SLEEPMIN,SLEEPMAX))
```

Resim 5.5. Video istemcilerinin işleyişi

Yaptıkları işlem ise Resim 5.5.'te örneği görüldüğü gibi, Mplayer [46] programını kullanarak belirli aralıklarla video transfer sunucuları üzerinden video transfer etmektir.

Http sunucusu düğümler

Bu düğümler ise üzerlerinde barındırdıkları örnek web sayfalarını web istemcilerine sunmaktadır. Diğer sunucu düğümlere benzer şekilde çalıştırılan web sunucu düğümleri Resim 5.6.'daki kod parçacığı ile başlatılmaktadır.

```
net['h6'].cmd\
('(nohup python /home/nail/websites/httpserver.py --dir \
/home/nail/websites/ & > /dev/null &)')
webServers.append(6)
```

Resim 5.6. Http sunucusunun aktif hale getirilmesi

Düğüm üzerinde çalıştırılan python betiği ise, python'un SimpleHTTPServer kütüphanesini kullanarak isteklere cevap vermektedir.

Http istemcisi düğümler

Bu düğümler ise diğer sunucu-istemci ilişkilerine benzer şekilde, bu kez sunucudan belirli aralıklarla rastgele web sayfası isteyecek şekilde tasarlanmıştır. Web sayfalarını rastgele zaman aralıklarıyla, [47]'teki betiğin, http sunucusundaki web sayfalarını isteyecek şekilde düzenlemiş halini çalıştırarak istemektedirler. Web istemcisi Mininet üzerinde Resim 5.7.'deki gibi aktif edilmiştir:

```
hostNum = 47
net['h'+str(hostNum)].cmd\
('(nohup python /home/nail/web-traffic-generator/gen_v2.py 10.0.0.' \
+ str(hedef) + ' & > /dev/null &)')
webClients.append(hostNum)
```

Resim 5.7. Http istemcilerinin işleyişi

ICMP paket transferi yapan düğümler

Bu düğümler, kendileri ile aynı gruptaki diğer düğümler ile birbirleri arasında ICMP paket transferi gerçekleştirmektedir. Düğümler Resim 5.8.'deki kod parçacığını çalıştırarak birbirlerine ICMP paketi göndermektedirler.

```
while True:
        randInt = randint(SLEEPMIN,SLEEPMAX)
        pingCount = randint(PINGMIN,PINGMAX)

        rep = os.system('ping -c ' + str(pingCount) + ' ' + remoteHost)

        time.sleep(randInt)
```

Resim 5.8. ICMP istemcilerinin işleyişi

Saldırgan düğümler

Saldırı trafiğinin, kontrolcü üzerinde yoğunluk yaratacak derecede etkiye sahip bir yöntem olması gerekmektedir. Böyle bir etkiyi yaratabilmesi için ise, farklı hedef adreslerine paket gönderme işlemi yaparak kontrolcüye packet_in mesajlarının gitmesini sağlaması gereklidir. Bu durumun etkileri Bölüm 4'te detaylı olarak açıklanmıştır. Literatürdeki

çalışmalar incelendiğinde ve sonrasında yapılan deneylerde hping3 [48] aracının böyle bir saldırı için uygun olduğu görülmüş ve deneylerde Resim 5.9.'da örneği verilen kod parçacığı ile uygulanmıştır:

```
def startAttack():
        command = 'hping3 -1 --interface h' + hostNum + '-eth0 ' + \
        ipRange + ' --rand-dest --rand-source --icmp --flood'
```

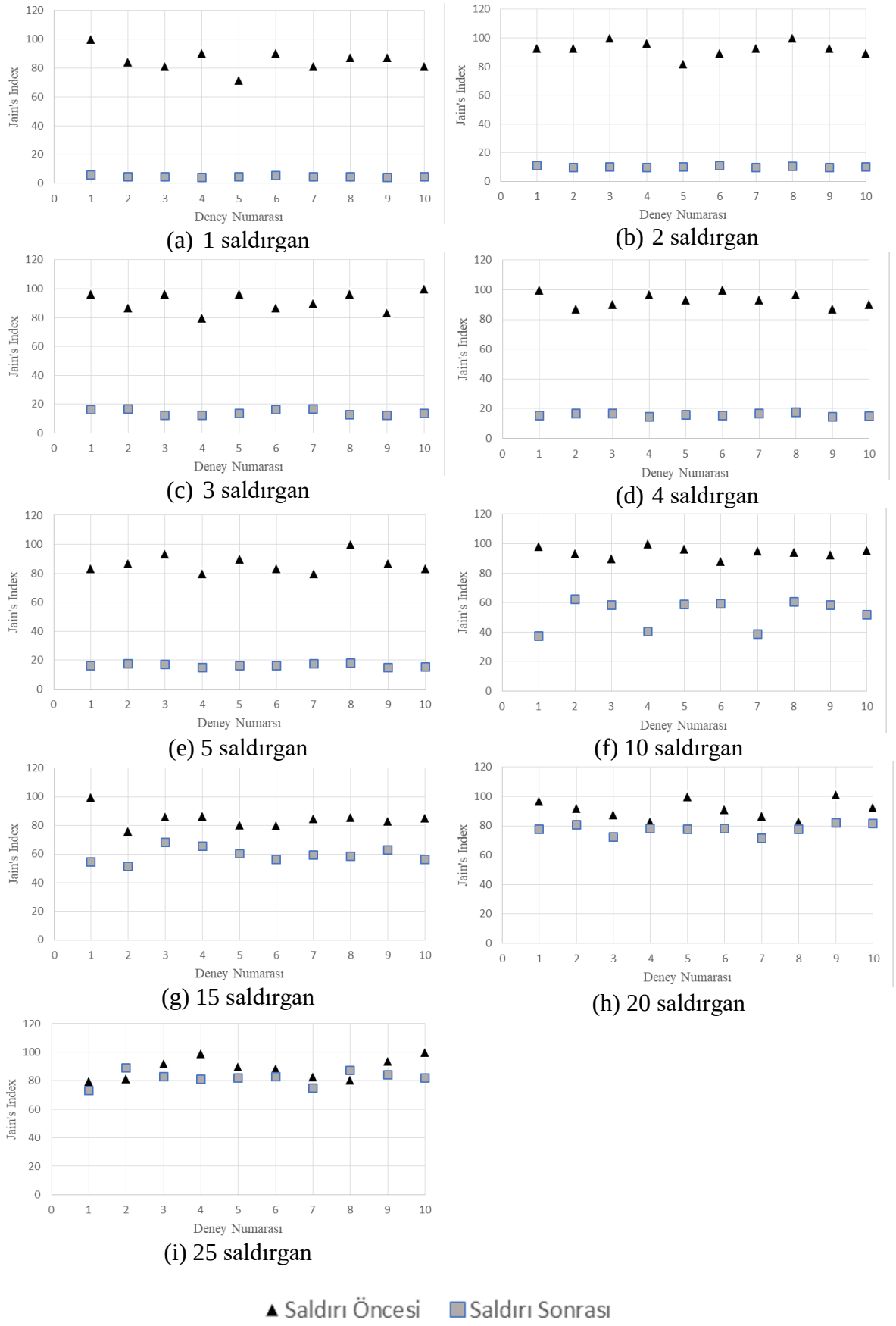
Resim 5.9. Saldırgan düğümlerin işleyişi

5.2. Jain's Index'e Ait Sonuçlar

Bu başlıkta saldırı durumuna ilişkin anomali tespitinde daha belirleyici olduğu görülen Jain's Index'e ilişkin detaylı grafiksel sonuçlar paylaşılmıştır. Deney sonuçlarına ait veriler değerlendirilirken aşağıdaki kriterler göz önüne alınmıştır:

- Her bir deneyde en yüksek Jain's Index değeri 100 kabul edilerek değerler en yüksek değere göre normalize edilmiştir.
- Karşılaştırmalar Jain's Index değerinin saldırı öncesi ve sonrası değişimi üzerinden yapılmıştır.

Şekil 5.2.'de Jain's Index'in deney ortamında saldırgan sayısına bağlı olarak saldırı öncesi ve sonrasındaki değer değişimleri gösterilmektedir. Jain's Index değerinin düğümlerin davranış profillerine, topolojideki dağılım oranlarına göre farklı aralıklarda seyredebilmesi nedeniyle, bundan sonraki çalışmalara referans olabilmesi açısından daha önce de belirtildiği üzere değer değişimlerinin gösteriminde gerçek değerlerin normalize edilmiş hâlleri kullanılmıştır.



Şekil 5.2. Saldırgan sayısına bağlı olarak Jain's Index'in değişimi

Şekil 5.2.'de grafiksel olarak ifade edilen 1,2,3,4,5,10,15,20 ve 25 saldırganlı ortamlardaki Jain's Index'in saldırı öncesi ve sonrasındaki ortalama değerlerine bakıldığında her bir grafik için değerlendirme yapmak faydalı olacaktır.

Şekil 5.2. (a)'da tek bir saldırganın bulunduğu ortamda saldırı öncesi ve sonrasındaki değerler arasındaki farkın oldukça yüksek olduğu görülmektedir. Saldırı öncesi Jain's Index değerinin ortalaması saldırı sonrası ortalamasına bölündüğünde önceki değerin saldırı sonrasındaki değerin 18,27 katı olduğu görülmüştür. Bu değerin bir hayli yüksek olması sebebiyle saldırı durumunu tespit etmek için yeterli görülmüştür. Şekil 5.2. (b)'de iki saldırganlı ortamda ise Jain's Index'in saldırı öncesindeki ortalama değeri, saldırı sonrasındakinin 9,29 katı olarak ölçülmüştür. Bu değer de saldırganı tespit edebilmek açısından yeterli görülmüştür.

Saldırgan sayısına bağlı olarak değişimler Şekil 5.2. (c-e)'de belirtildiği şekilde 3, 4 ve 5 saldırgan için sırasıyla 6,52, 5,86 ve 4,26 kat olarak ölçülmüştür. Bu değerler göz önüne alındığında 3,4 ve 5 saldırgan için de saldırı durumunun tespit edilebileceği değerlendirilmiştir. Bu değer değişimi Şekil 5.2. (f-i)'de belirtildiği şekilde 10,15,20 ve 25 saldırgan için sırasıyla 1,80, 1,43, 1,16 ve 1,01 olarak ölçülmüştür. Ancak bu değer farklarının saldırı öncesi ve sonrası ortalama değerler olduğunu belirtmekte fayda vardır. Aynı senaryoların anlık değişimleri değerlendirilecek olursa, 1,2,3,4 ve 5 saldırgan için sırasıyla 6,71, 2,60, 2,13, 1,92 ve 1,80 olarak ölçüldüğü görülmüştür. Yani Jain's Index için ortalama değerler tespit için yeterli iken, anlık değişimler arasındaki farkın daha az olması sebebiyle durum tespiti için yeterli görülmemiştir.

Anlık değişimin ve ortalama değişim arasındaki davranış farkı yorumlandığında Jain's Index'in saldırı başladıktan hemen sonra keskin bir düşüşten ziyade kademeli olarak azaldığı sonucuna varılmıştır. Saldırının bir an önce tespit edilebilmesi için Jain's Index'teki değişimin mümkün olduğunca çabuk farkedilmesi amaçlandığından, uzun bir zaman penceresinde gözlem yapıp karar vermek yerine değişimin daha kısa bir zaman penceresinde izlenmesi ve kararın buna göre verilmesinin gerekli olduğu değerlendirilmiş, bu sebeple Jain's Index'in 5 saldırganın üzerindeki değişimi efektif bir tespit işlemi için yeterli görülmemiştir.

Saldırgan sayısına bağlı olarak genel bir değerlendirme yapılması gerekirse, Jain's Index'in saldırı öncesi ve sonrası değeri arasındaki farkın saldırgan sayısı arttıkça azaldığı görülmektedir. Jain's Index, hesaplama yapısı gereği, sistemdeki değerler birbirine yaklaştıkça 1'e yaklaşan bir davranış biçimi göstermektedir. Bu durum ise sistemin adil bir dağıtım sergilediğine işaret etmektedir. Saldırgan sayısı arttıkça, saldırganların fazlaca packet_in mesajı gönderilmesine neden olması ve buna karşılık az miktarda veri transferi nedeniyle packet_in / tx paket oranları biçimleri birbirine benzediğinden, sistem içerisinde birbirine benzer davranış sergileyen düğümlerin sayısı artmakta ve bu da Jain's Index değerinin saldırı sonrasında belirgin bir biçimde düşmesini engellemektedir.

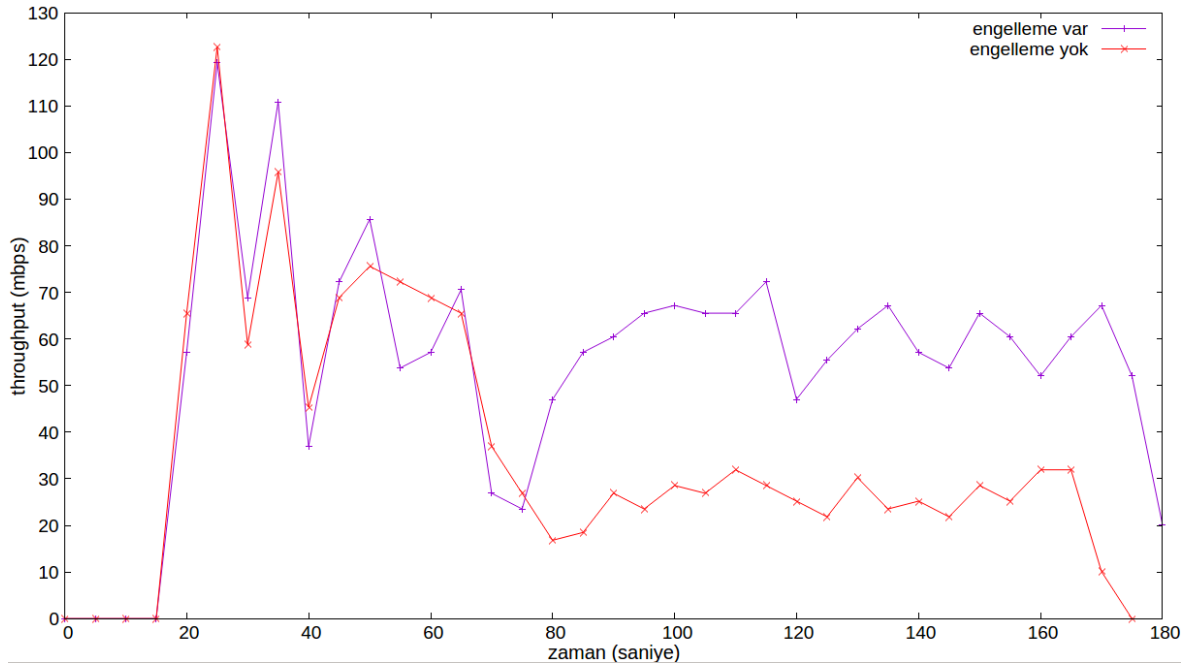
5.3. Önerilen Tespit ve Engelleme Altyapısına İlişkin Sonuçlar

Bu bölümde önceki başlıklarda Jain's Index'in anomali tespitinde belirli bir saldırgan oranına kadar başarılı olduğu sayısal değerlerle ortaya konulmuştur. Bu başlık altında ise, önerilen çözüm yönteminin ağda nasıl sonuçlar yarattığı grafiklerle gösterilmiştir. Önerilen çözüm yönteminin yarattığı sonuçlar, veri iletim hızı ve Jain's Index değeri temelinde değerlendirilmiştir. Deneylerde aşağıdaki kısıtlar belirlenmiştir:

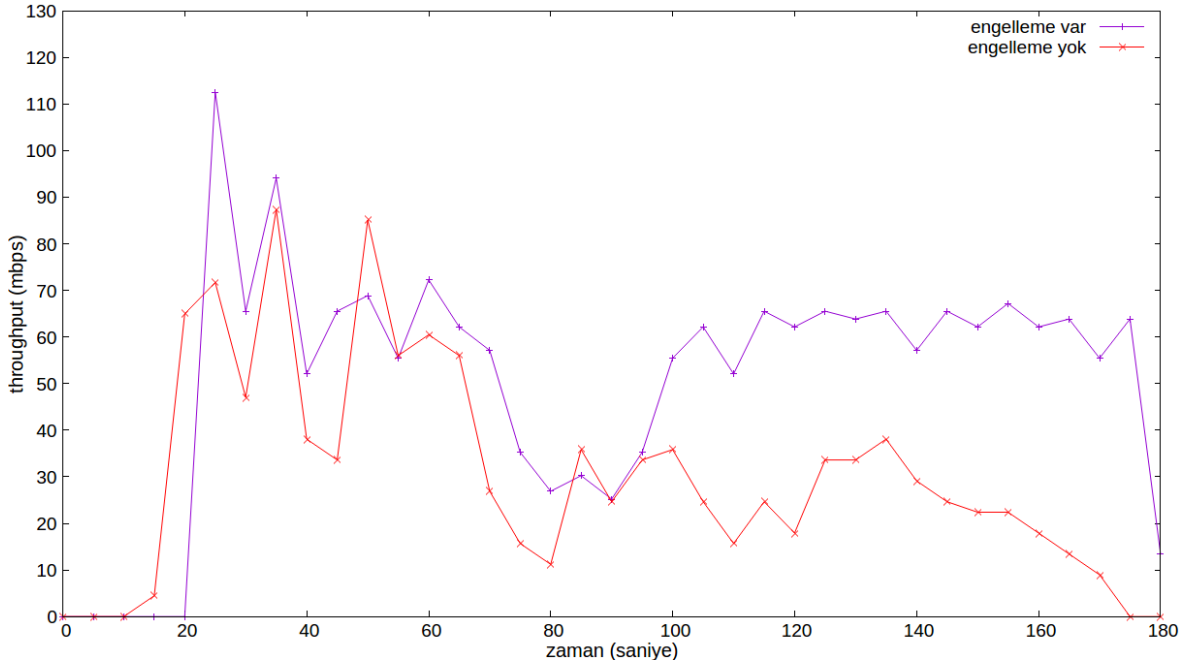
- Tespit sınırının saldırgan düğüm / normal düğüm oranı üzerinden %10 olarak belirlenmesi nedeniyle, ağdaki saldırgan sayısı 5 ile sınırlandırılmıştır.
- Her bir deney adımında benzetim önce 60 saniye boyunca saldırı olmadan çalıştırılmış, sonrasında ise saldırı işlemi başlatılmıştır.
- Grafiklerde saldırı öncesi ve sonrasında ağda iletilen veri miktarı birlikte gösterilmiştir.
- Grafiklerde ayrıca, tespit mekanizmasının uygulandığı ve uygulanmadığı durumlar birlikte gösterilerek karşılaştırma yapılabilmesi sağlanmıştır.

5.3.1. Veri iletim hızına ilişkin sonuçlar

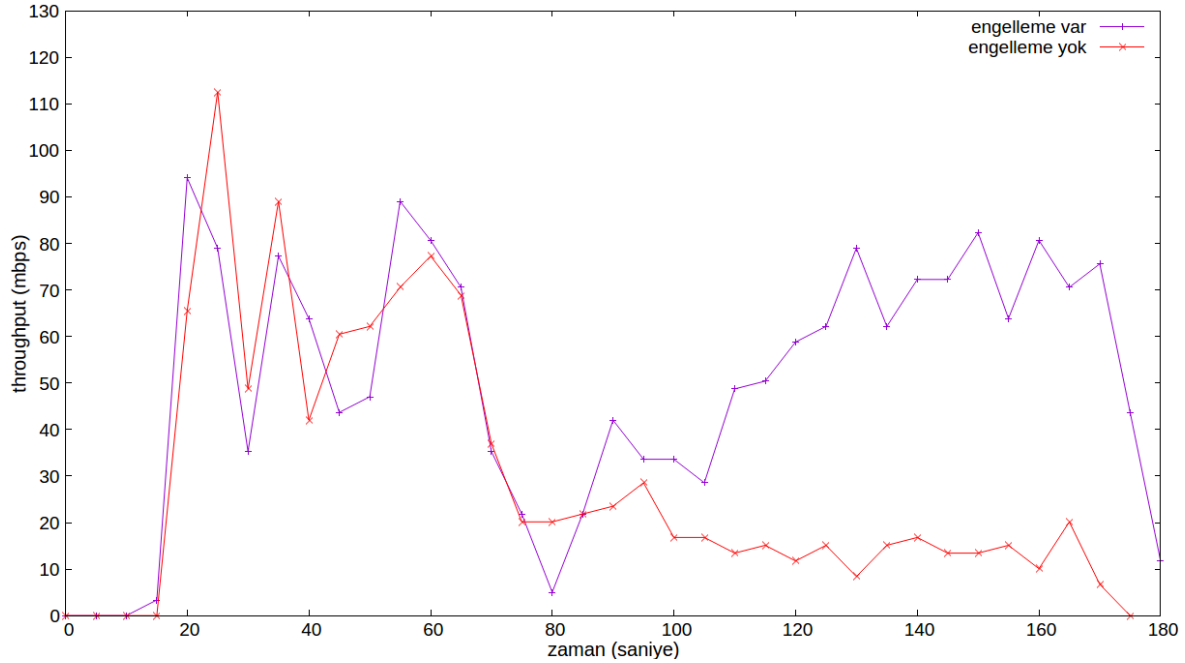
Bu başlık altında önerilen yöntem uygulandığında sistemdeki veri iletim hızında (throughput) yaşanan değişim grafiklerle (Şekil 5.3.–Şekil 5.7.) gösterilmiştir. Grafiklerdeki sonuçlar önerilen yöntemin veri iletim hızına olan katkısını açık şekilde ortaya koymaktadır.



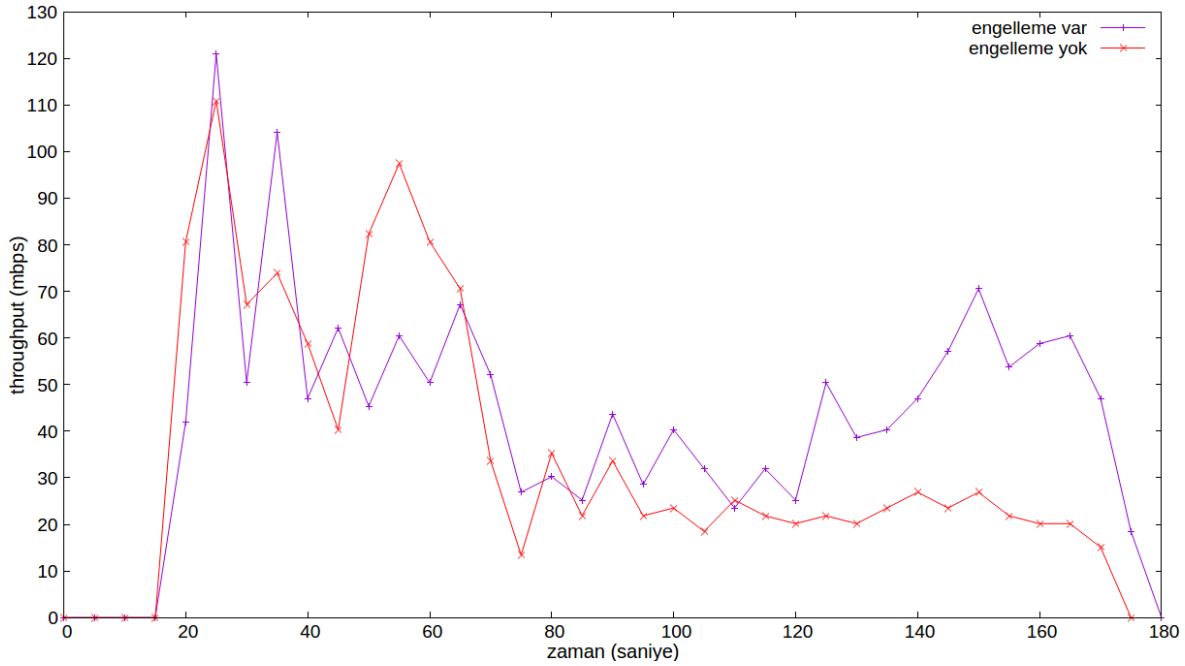
Şekil 5.3. Bir saldırganlı ortamda engelleme sisteminin veri iletim hızına etkisi



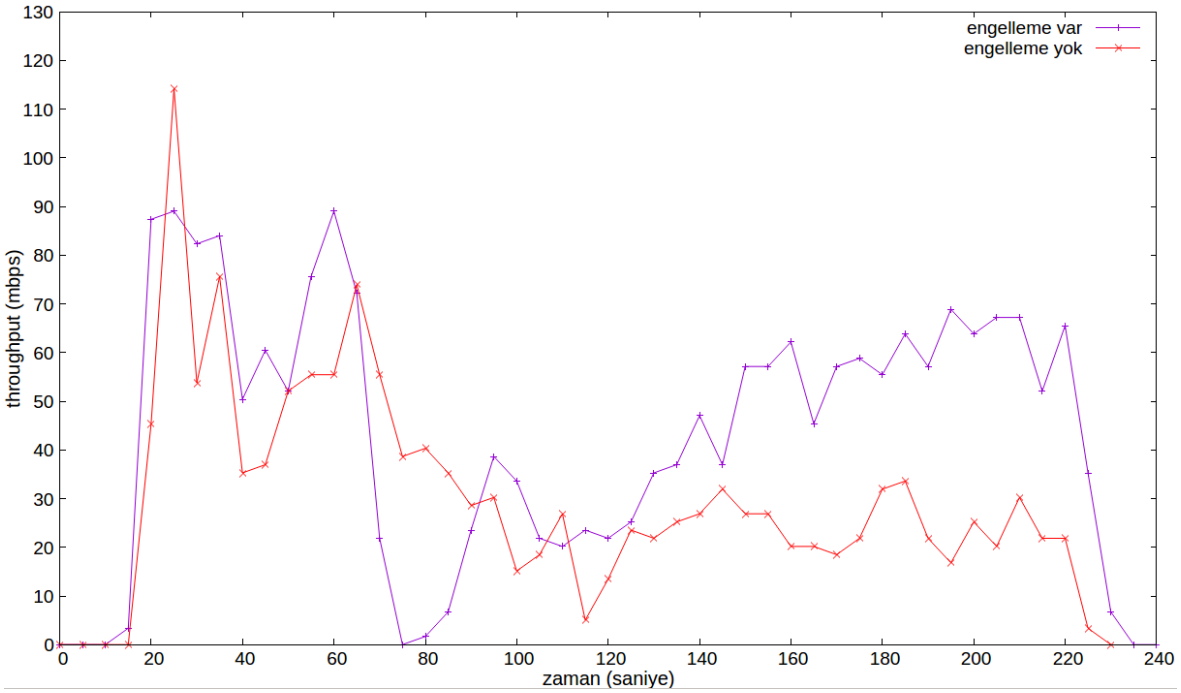
Şekil 5.4. İki saldırganlı ortamda engelleme sisteminin veri iletim hızına etkisi



Şekil 5.5. Üç saldırganlı ortamda engelleme sisteminin veri iletim hızına etkisi



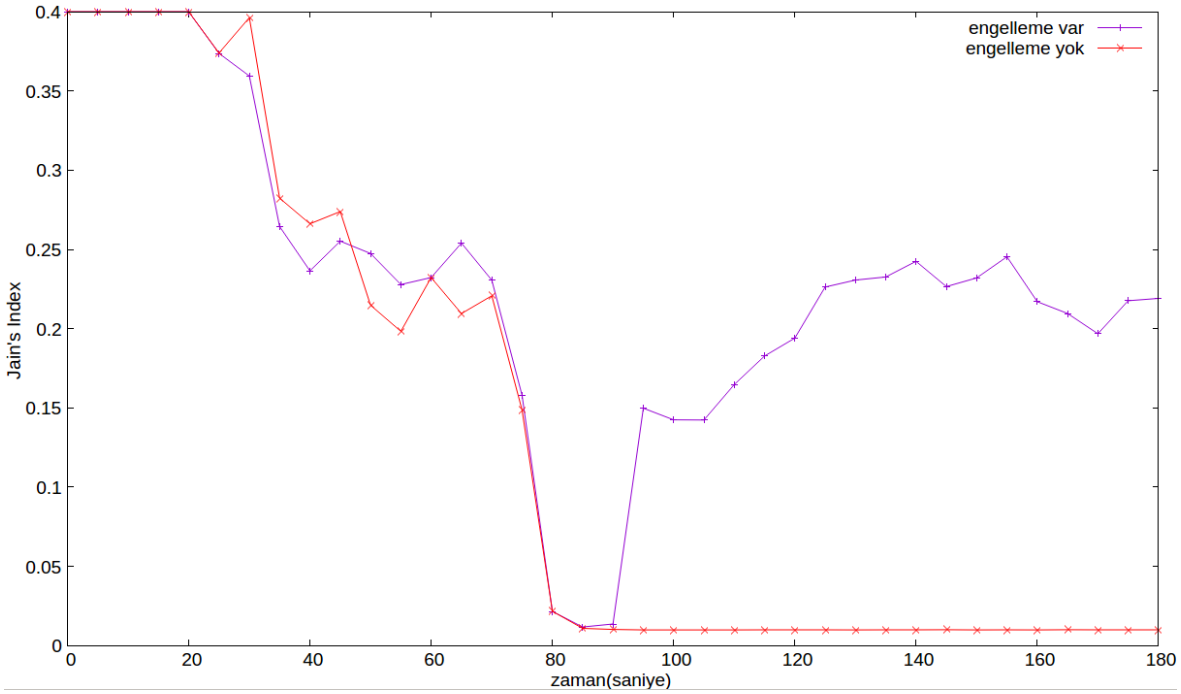
Şekil 5.6. Dört saldırganlı ortamda engelleme sisteminin veri iletim hızına etkisi



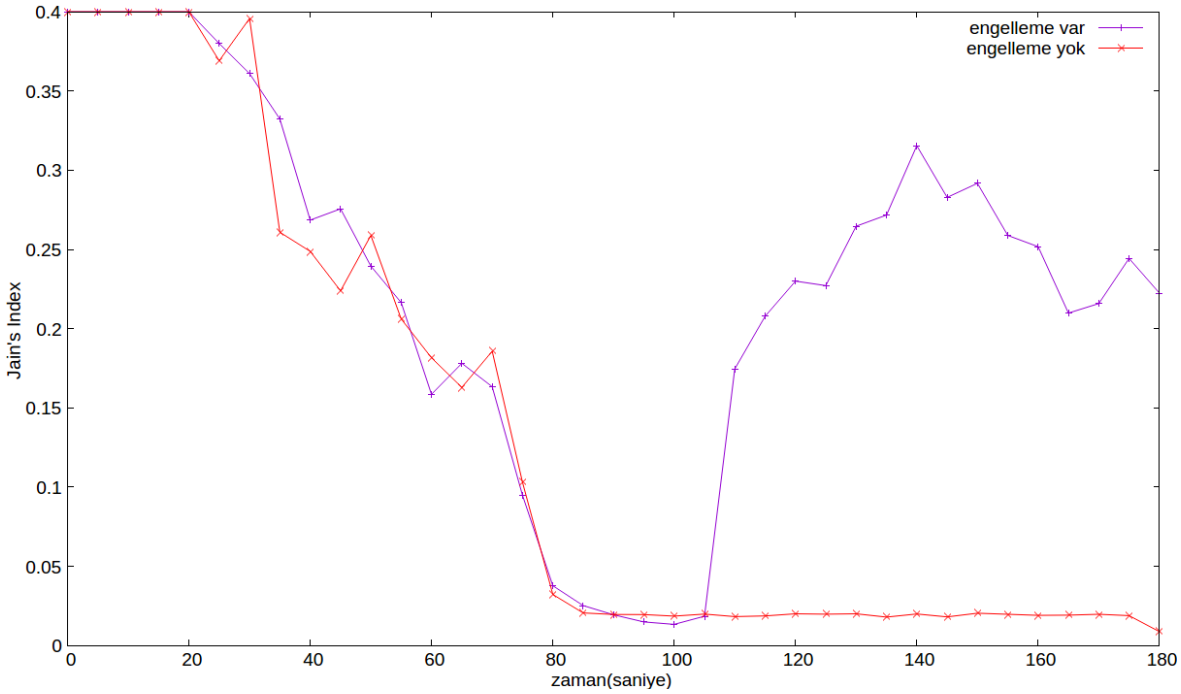
Şekil 5.7. Beş saldırganlı ortamda engelleme sisteminin veri iletim hızına etkisi

5.3.2. Jain's index değerine ilişkin sonuçlar

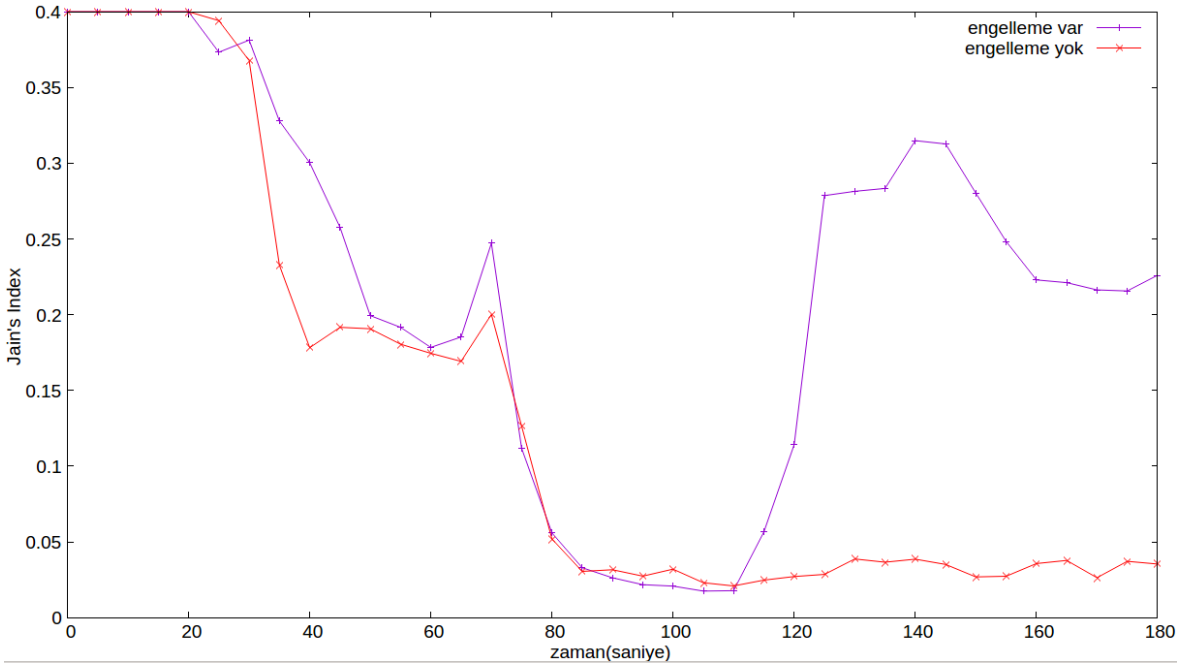
Bu başlık altında Jain's Index değerinin saldırgan sayısına bağlı olarak tespit ve engelleme mekanizmasının aktif olup olmadığı durumlarda nasıl değiştiği grafiklerle (Şekil 5.8.–Şekil 5.12.) gösterilmiştir.



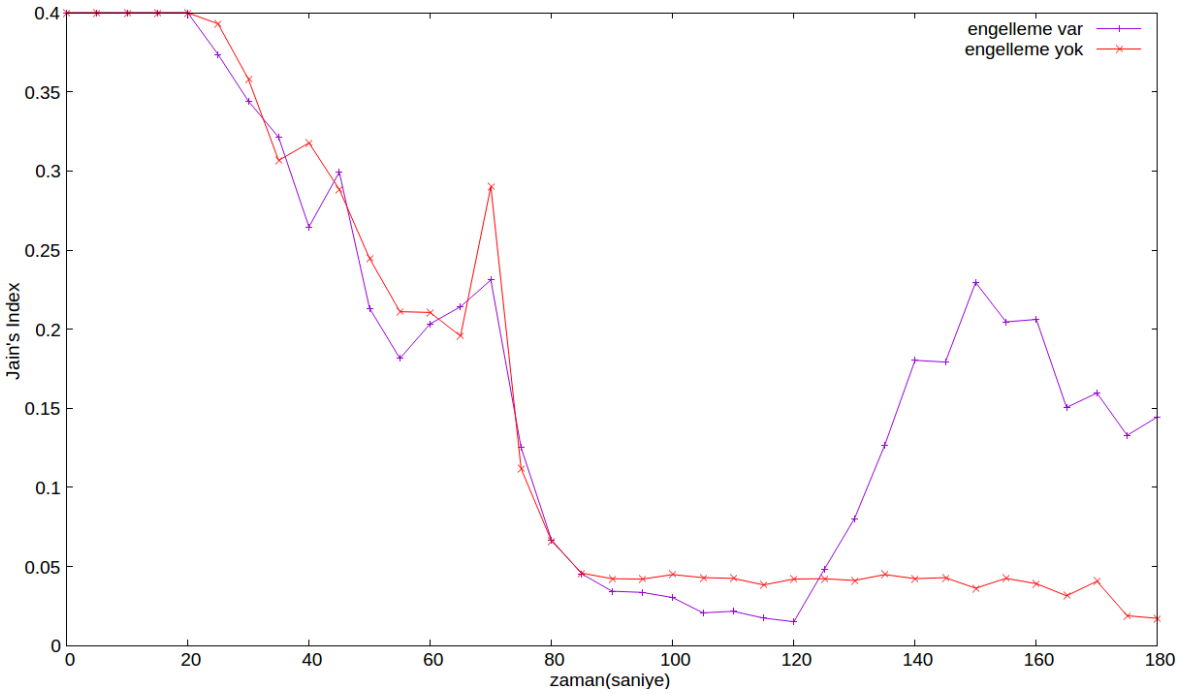
Şekil 5.8. Bir saldırganlı ortamda engelleme sisteminin Jain's Index değerine etkisi



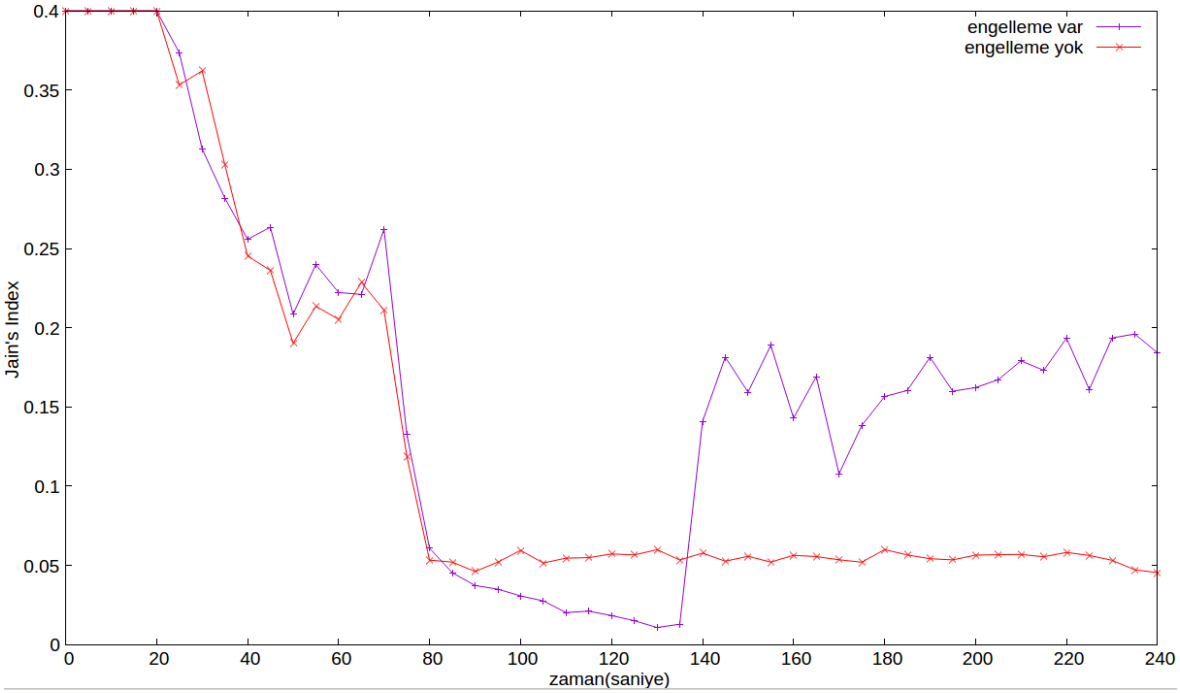
Şekil 5.9. İki saldırganlı ortamda engelleme sisteminin Jain's Index değerine etkisi



Şekil 5.10. Üç saldırganlı ortamda engelleme sisteminin Jain's Index değerine etkisi



Şekil 5.11. Dört saldırganlı ortamda engelleme sisteminin Jain's Index değerine etkisi



Şekil 5.12. Beş saldırganlı ortamda engelleme sisteminin Jain's Index değerine etkisi

5.4. Tespit ve Önleme Mekanizmasının Veri İletimi ve Jain's Index Değeri Üzerindeki Etkisine İlişkin Değerlendirme

Bu başlıkta, önerilen tespit ve engelleme mekanizmasının veri iletiminde ve Jain's Index değeri üzerinde ne gibi değişimlere neden olduğu yorumlanmıştır. Çizelge 5.1.'de, önerilen sistemin uygulandığı ve uygulanmadığı durumlarda 5.3.1 başlığında grafikleri verilen, iletilen veri miktarının saldırgan sayısına bağlı olarak değişimi gösterilmiştir.

Çizelge 5.1. Önerilen tespit ve engelleme sisteminin veri iletiminde sağladığı iyileşme

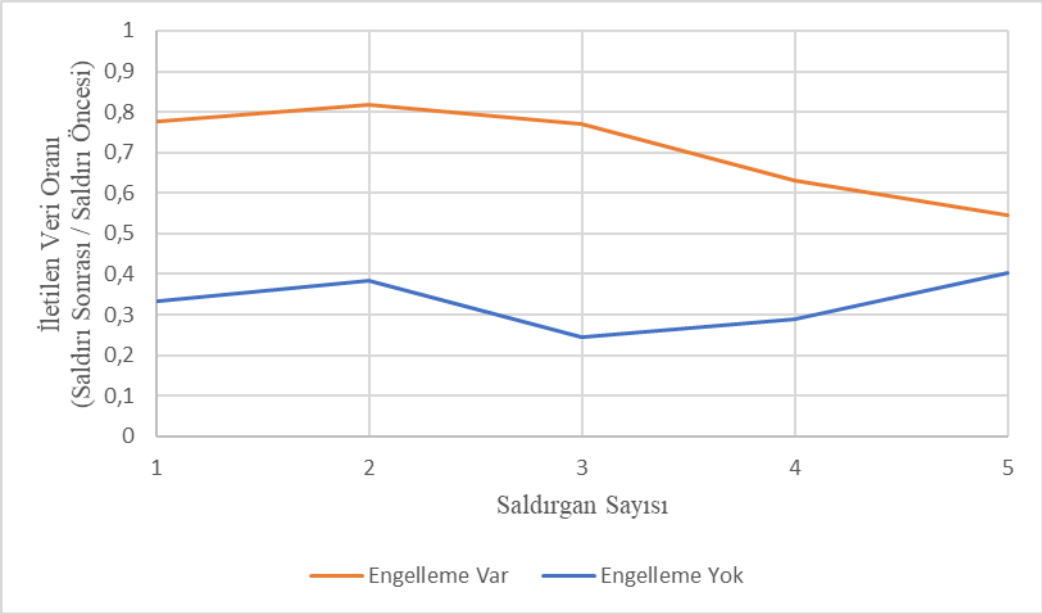
Saldırgan Sayısı	Engelleme Var			Engelleme Yok		
	Önce İletilen (mbps)	Sonra İletilen (mbps)	Değişim %	Önce İletilen (mbps)	Sonra İletilen (mbps)	Değişim %
1	73,25	56,89	22,34	73,92	24,56	66,77
2	64,14	52,48	18,18	56,96	21,92	61,52
3	68,04	52,40	22,99	69,72	17,12	75,44
4	65,02	41,00	36,94	76,27	22,16	70,95
5	74,26	40,40	45,59	59,81	24,18	59,58

Çizelge 5.1.'de sayısal değerleri verilen senaryolara ilişkin saldırgan sayısı bazında değerlendirme yapılacak olursa: saldırının başlangıcından önceki periyotta, sisteme ait veri iletim hızının tüm durumlarda 56,96 ila 76,27 mbps değerleri arasında olduğu gözlenmiştir. Saldırı başladıktan sonra ise, engelleme sisteminin aktif olmadığı durumda üç saldırgana kadar ortalama veri iletim hızı düşme eğilimi göstermiş, üç saldırgandan sonra ise artışa geçmiştir. Bu durum saldırı etkisi açısından üç saldırganın sistemde en fazla zafiyet yaratan durum olduğunu göstermektedir. Zira bu seviyede veri iletim hızının saldırı öncesine nazaran %75,44 oranında düştüğü gözlenmiştir. Saldırgan sayısı üçün üzerine çıktığında ise saldırganlar veri iletim performansını daha da azaltmaktan ziyade, anahtarlar ve kontrolcü arasındaki altyapının iletim kapasitesini sınır seviyesine ulaştırdığından daha fazla negatif etki gösterememektedirler.

Engelleme sisteminin aktif olduğu durumda ise, saldırının başlamasından sonraki periyotta en yüksek veri iletim hızı tek saldırganlı durumda gerçekleşmiştir. Bunun nedeni ise saldırganın tespit edilmesinin ardından sistemin engelleme durumuna geçmesi, ilgili anahtarlara drop kurallarını eklemesiyle birlikte sistemin tekrar hızla eski seviyesine geri dönmesinden kaynaklanmaktadır.

Saldırgan sayısı arttıkça, engelleme sisteminin aktif olduğu durumda sistem her bir saldırganı engelledikten sonra saldırgana ait etkinin ağ üzerinden kalkmasını on saniye boyunca beklediğinden, diğer saldırganlar bu süre boyunca ağı etkilemeye devam etmektedir. Saldırgan sayısının artması sistemin toparlanma süresini uzattığından, saldırı sonrasındaki ortalama veri hızı saldırgan sayısıyla ters orantılı hareket etmektedir. Ancak, her durumda, tüm saldırganlar etkisiz hale getirildikten sonra sistemdeki nihai veri iletim hızı, saldırı öncesindeki seviyelerine geri dönmektedir. Bu da engelleme sisteminin saldırgan sayısı arttıkça etkisini geç gösterse bile nihai durumda veri iletim hızını normal seviyelerine geri döndürdüğünü göstermektedir.

Saldırı sonrası ve öncesi arasındaki veri iletim hızı oranının saldırgan sayısına bağlı olarak değişimi Şekil 5.8.'de özetlenmiştir. Engelleme sisteminin aktif olduğu durumda saldırgan sayısının artmasıyla veri iletim hızı oranında yaşanan düşüş, daha önce de bahsedildiği üzere sistemin saldırganları kademeli biçimde engellemesinden kaynaklanmaktadır.



Şekil 5.13. İletilen veri miktarındaki değişimin karşılaştırılması

Önerilen sisteme ait sonuçlar Jain's Index değeri temelinde değerlendirilecek olursa, Jain's Index değerinin saldırı esnasında tespit için yeterli düşüşü gösterdiği grafiklerde gözlenmektedir. Saldırgan sayısı arttıkça, Jains's Index değerinin saldırı sonrasındaki düşüşünün sınırlandığı gözlenmiş, ancak düşüş miktarı tespit için yeterli seviyelerde meydana gelmiştir. Saldırının tespitinden sonra anahtarlara drop kuralları eklendikçe, saldırgan sayısının birden fazla olduğu durumlarda Jain's Index değeri genellikle düşüş eğilimi göstermiştir. Bu durum, daha önce de belirtildiği üzere, Jain's Index'in hesaplama yapısı gereği saldırgan sayısı azaldıkça sistemde adil dağılımın azaldığı sonucunun çıkarılmasından kaynaklanmaktadır. Tüm durumlarda, saldırganların tamamı engellendiğinde, Jain's Index değeri hızlı şekilde yükselmekte ve saldırı öncesindeki seviyelerine geri dönmektedir.

6. SONUÇ VE ÖNERİLER

Bu tez çalışmasında, DoS saldırılarının YTA üzerindeki etkileri incelenmiş ve bir çözüm yöntemi geliştirilmiştir. Öncelikle DoS saldırılarının YTA üzerindeki etkileri ortaya konulmuştur. Kontrolcü birimin çalışma yapısı sebebiyle DoS saldırılarına karşı açık bir hedef olduğu açıklanmış, problemin tanımı deney ortamında yapılarak etkileri detaylı biçimde ortaya konulmuştur.

Tanımlanan problemin çözümüne ilişkin literatürde yapılan çalışmalar incelenmiş, bunun sonucunda literatürde eşik değeri kullanılarak uygulanan bazı çalışmaların yeteri kadar ayrıntılı olmadığı sonucuna varılmıştır. Bu nedenle kapsamlı bir çalışma ile DoS saldırılarının YTA bileşenleri üzerindeki etkilerinin incelenmesi ve buna bağlı olarak bir tespit ve önleme yönteminin geliştirilmesinin faydalı olacağı sonucuna varılmıştır. Bu kapsamda tespit ve önleme yönteminin geliştirileceği ortam, literatürdekilerden farklı olarak mümkün olduğunca gerçek hayata yakın olarak tasarlanmış ve gerçek trafik üretecek bir altyapı kurgulanmıştır.

Tez çalışması kapsamında DoS saldırısının YTA'da hangi parametreleri daha fazla etkilediği araştırılmış ve sadece packet_in değerine bakılarak yapılacak bir değerlendirmenin yanlış yönlendirmelere yol açabileceği görülmüştür. Bunun yerine packet_in sayısının farklı değerler ile birlikte değerlendirilmesi yoluna gidilmiş ve packet_in sayısının iletilen paket sayısına oranının saldırgan düğümleri tespit etmede daha başarılı olduğu görülmüştür.

Tespit sisteminin tasarımında, belli bir eşik değerinden bağımsız bir yöntem tasarlanması yoluna gidilmiştir. Bu kapsamda literatürde YTA alanında daha önce üzerinde çalışılmayan bir yöntem üzerine araştırma yapılmış ve DoS saldırılarının YTA üzerinde bir adaletsizlik durumu yaratacağı düşüncesiyle Jain's Index ve entropi değerleri üzerinden değerlendirme yoluna gidilmiştir. Bu iki yöntem ile yapılan deneylerde Jain's Index'in saldırı durumundaki değişiminin entropiden daha ayırt edici olduğu sonucuna varılmış ve tespit işlemi Jain's Index değeri üzerinden gerçekleştirilmiştir. Yapılan deneylerde saldırgan düğüm / normal düğüm oranının %10'a kadar olduğu durumlarda tespit işleminin başarıyla gerçekleştirildiği görülmüştür. Saldırgan oranının %10'un üzerine çıkmasıyla

birlikte Jain’s Index değerindeki değişimler daha yakın seviyelere gelmekte ve tespit işlemi güvenilirliğini yitirmektedir.

Jain’s Index temelinde uygulanan tespit mekanizmasının ardından bir engelleme mekanizması tasarlanmış ve en fazla packet_in / tx paket oranına sahip düğümlerin sırayla engellenmesi metodu uygulanmıştır. Engelleme işlemi uygulanırken, drop kuralları eklenmesinin yanı sıra, kontrolcü tarafından tutulan drop listesine göre engellenmiş düğümlerin birdahaki packet_in mesajı göndermelerinde göz ardı edilmeleri sistem performansını artıran önemli bir etken olmuştur. Engelleme sistemine ait sonuçlara bakıldığında, sistemin veri iletimindeki saldırı öncesi ve sonrasına ilişkin farkı %52’ye varan seviyede iyileştirdiği görülmüştür.

Bu tez çalışmasında gerçekleştirilen çalışmanın literatüre faydası özetlenecek olursa, kontrolcü üzerinde sadece packet_in sayısına bağlı bir kara liste oluşturulmasıyla kurulacak sistemlerin hatalı yönlendirmelere sebep olabileceği ilk kez ortaya konulmuş ve örneklerle de desteklenmiştir. Ayrıca packet_in sayısıyla birlikte farklı parametrelerin kullanılmasının fayda sağlayabileceği de deney sonuçlarıyla ortaya konulmuştur. Kontrolcü üzerinde uygulanacak engelleme sistemlerinde drop kuralı ekleme aşamasında önemli bir performans zafiyeti belirlenmiş ve bunu aşmak amacıyla da yeni bir yöntem önerilmiştir.

Bu tez çalışması birkaç bakımdan gelecekte gerçekleştirilecek çalışmalara öncülük edebilecek niteliktedir. Ağdaki adaletsizlik durumunu tespit için kullanılan Jain’s Index veya entropiden daha farklı yöntemler kullanılabilir veya packet_in / tx paket oranı haricinde farklı parametrelerle tespit işlemi gerçekleştirilebilir. Geliştirilen sistemin anahtarlardan istatistik bilgisi toplamayı gerektirmesi sebebiyle, kontrolcünün yönettiği ağın dışarısından gelen saldırılarda paket iletim istatistiklerinin toplanması mümkün olmayabilir. Buna ek olarak saldırganlar düşük oranda (low rate) DoS saldırıları gerçekleştirip herhangi bir mekanizma ile tespit olmayan saldırı yöntemleri kullanabilirler. Kontrolcünün saldırı durumunu veya saldırganı tespit edemediği böyle durumlarda, tespit yapılamasa dahi kontrolcünün kaynaklarını adil biçimde dağıtarak hayatta kalması gerekebilir. Bu tür problemlerde adil dağıtım algoritmalarının performansının karşılaştırılmasının da literatüre katkı sağlayacağı değerlendirilmektedir.

KAYNAKLAR

1. İnternet: Open Networking Foundation. Software-defined networking (SDN) definition. Web: http://www.opennetworking.org/SDN-definition/ adresinden 21 Nisan 2019'da alınmıştır.

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

3. Zimmermann, H. (1980). OSI reference model-the ISO model of architecture for open systems interconnection. *IEEE Transactions on communications*, 28(4), 425-432.

4. İnternet: Open Networking Foundation. OpenFlow Switch Specification Version 1.5.1. Web: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf adresinden 21 Nisan 2019'da alınmıştır.

5. Yan, Q., Yu, F. R., Gong, Q., Li, J. (2016). Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Communications Surveys & Tutorials*, 18(1), 602-622.

6. Li, W., Meng, W., Kwork, L. (2016). A survey on OpenFlow-based Software Defined Networks: Security challenges and countermeasures, *Journal of Network and Computer Applications*, 68, 126-139.

7. Shu, Z., Wan, J., Li, D., Lin, J., Vasilakos, A., Imran, M. (2016). Security in Software-Defined Networking: Threats and Countermeasures, *Mobile Networks and Applications*, 21(5), 764-776.

8. Shin, S., Yegneswaran, V., Porras, P., Gu, G. (2013). *Avant-guard: Scalable and vigilant switch flow management in software-defined networks.* In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, Berlin, Germany.

9. Fichera, S., Galluccio, L., Grancagnolo, S. C., Morabito, G., Palazzo, S. (2015). OPERETTA: An OPenflow-based Remedy to Mitigate TCP SYNFLOOD Attacks against Web Servers. *Computer Networks*, 92, 89-100.

10. Zhang, P., Wang, H., Hu, C., Lin, C. (2016). On Denial of Service Attacks in Software Defined Networks. *IEEE Network*, 30(6), 28-33.

11. Kuerban, M., Tian, Y., Yang, Q., Jia, Y., Huebert, B., Poss, D. (2016). *FlowSec: DoS Attack Mitigation Strategy on SDN Controller.* 2016 IEEE International Conference on Networking, Architecture and Storage (NAS), Long Beach, USA.

12. Specht, S. M., Lee, R. B. (2004). *Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures.* In ISCA International Conference on Parallel and Distributed Computing Systems, 543-550, San Francisco, USA.

13. Chen, X., Yu, S. (2016). A Collaborative Intrusion Detection System against DDoS for SDN. *IEICE Transactions on Information and Systems*, 99(9), 2395-2399.

14. Braga, R., Mota, E., Passito, A. (2010). *Lightweight DDoS flooding attack detection using NOX/OpenFlow in local computer networks (LCN)*, 2010 IEEE 35th Conference on Local Computer Networks, Denver, USA.

15. Cui, Y., Yan, L., Li, S., Xing, H., Pan, W., Zhu, J., Zheng, X. (2016). SD-Anti-DDoS: Fast and efficient DDoS defense in software-defined networks, *Journal of Network and Computer Applications*, 68, 65-79.

16. Shirali-Shahreza, S., Ganjali, Y. (2013). *Efficient implementation of security applications in openflow controller with flexam.* 2013 IEEE 21st Annual Symposium on High-Performance Interconnects (HOTI), San Jose, USA.

17. Hu, H., Han, W., Ahn, G., J. Zhao, Z. (2014). *FLOWGUARD: Building robust firewalls for software-defined networks.* In Proceedings of the third workshop on Hot topics in software defined networking. Chicago, USA.

18. Buragohain, C., Medhi, N. (2016). *FlowTrApp: An SDN based architecture for DDoS attack detection and mitigation in data centers.* 2016 3rd International Conference on Information Science and Control Engineering, Beijing, China.

19. Dao, N. N., Park, J., Park, M., Cho, S. (2015). *A feasible method to combat against DDoS attack in SDN network.* 2015 International Conference on In Information Networking (ICOIN), Siem Reap, Cambodia.

20. Qian, Y., You, W., Qian, K. (2016). *OpenFlow flow table overflow attacks and countermeasures.* 2016 European Conference on Networks and Communications (EuCNC), Athens, Greece.

21. Dao, N. N., Kim, J., Park, M., Cho, S. (2016). Adaptive Suspicious Prevention for Defending DoS Attacks in SDN-Based Convergent Networks. *PloS one*, 11(8), 28-35.

22. Wang, S., Chandrasekharan, S., Gomez, K., Kandeepan, S., Al-Hourani, A., Asghar, M. R., Zanna, P. (2018). *SECOD: SDN sEcure control and data plane algorithm for detecting and defending against DoS attacks.* 2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, Taiwan.

23. Carvalho, R. N., Bordim, J. L., Alchieri, E. A. P. (2019*). Entropy-Based DoS Attack Identification in SDN*. In 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Rio de Janeiro, Brazil.

24. İnternet: Floodlight Project. (2018). Web: http://www.projectfloodlight.org/floodlight/ adresinden 21 Nisan 2019'da alınmıştır.

25. Piedrahita, A. F. M., Rueda, S., Mattos, D. M., Duarte, O. C. M. (2015). *FlowFence: a denial of service defense system for software defined networking.* In Global Information Infrastructure and Networking Symposium (GIIS), Thessaloniki, Greece.

26. Yau, D. K., Lui, J., Liang, F., Yam, Y. (2005). Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *IEEE/ACM Transactions on Networking (TON)*, 13(1), 29-42.

27. İnternet: Saifullah, A. (2009). Defending against distributed denial-of-service attacks with weight-fair router throttling. *Washington University Open Scholarship*. Web: https://openscholarship.wustl.edu/cse_research/23/ adresinden 21 Nisan 2019'da alınmıştır.

28. Wang, T., Chen, H., Cheng, G., Lu, Y. (2018). SDNManager: A safeguard architecture for SDN DoS attacks based on bandwidth prediction. *Security and Communication Networks*, 14(2), 16-32.

29. Xie, R., Xu, M., Cao, J., and Li, Q. (2019). *SoftGuard: Defend Against the Low-Rate TCP Attack in SDN.* In 2019 IEEE International Conference on Communications (ICC), Shanghai, China.

30. İnternet: Source Address Validation Improvements. (2008). Web: http://tools.ietf.org/wg/savi adresinden 21 Nisan 2019'da alınmıştır.

31. Yao, G., Bi, J., Xiao, P. (2011). *Source address validation solution with OpenFlow/NOX architecture.* 2011 19th IEEE International Conference on Network Protocols, Vancouver, Canada.

32. Wang, H., Xu, L., Gu, G. (2014). *OF-GUARD: A DoS attack prevention extension in software-defined networks.* In Proceedings of the 2014 Open Networking Summit (ONS), California, USA.

33. Wang, H., Xu, L., Gu, G. (2015). *Floodguard: A DoS attack prevention extension in software-defined networks.* 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Rio de Janeiro, Brazil.

34. İnternet: Iperf. Web: http://iperf.fr adresinden 21 Nisan 2019'da alınmıştır.

35. Jain, R., Chiu, D., Hawe, W. (1984). A quantitative measure of fairness and discrimination for resource allocation in shared systems. *Eastern Research Laboratory Digital Equipment Corporation.* 13, 14-18.

36. Hussain, S. A., Iqbal, M., Saeed, A., Raza, I., Raza, H., Ali, A., Baig, A. (2017). An efficient channel access scheme for vehicular ad hoc networks. *Mobile Information Systems*, 15, 27-33.

37. Arianpoo, N., Leung, V. C. (2017). A smart fairness mechanism for Concurrent multipath transfer in SCTP over wireless multi-hop networks. *Ad Hoc Networks*, 55, 40-49.

38. Amer, M., Busson, A., Guérin Lassous, I. (2016). *Association optimization in Wi-Fi networks: Use of an access-based fairness.* In Proceedings of the 19th ACM international conference on modeling, analysis and simulation of wireless and mobile systems, Malta.

39. Shannon, C.E. (1948) A mathematical theory of communication. *Bell System Technical Journal,* 7, 379-423.

40. Müter, M., Asaj, N. (2011). *Entropy-based anomaly detection for in-vehicle networks*. 2011 IEEE Intelligent Vehicles Symposium, Baden-Baden, Germany.

41. Ma, X., Chen, Y. (2014). DDoS detection method based on chaos analysis of network traffic entropy. *IEEE Communications Letters*, 18(1), 114-117.

42. Lantz, B., Heller, B., McKeown, N. (2010). *A network in a laptop: rapid prototyping for software-defined networks.* Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Philadelphia, USA.

43. İnternet: Ryu SDN Framework. Web: https://osrg.github.io/ryu/ adresinden 21 Nisan 2019'da alınmıştır.

44. İnternet: Pyftpdlib. Web: http://pypi.org/project/pyftpdlib adresinden 21 Nisan 2019'da alınmıştır.

45. İnternet: Videolan Organisation. Vlc Media Player. Web: https://www.videolan.org/vlc adresinden 21 Nisan 2019'da alınmıştır.

46. İnternet: Mplayer Organisation. Mplayer. Web: https://www.mplayerhq.hu adresinden 21 Nisan 2019'da alınmıştır.

47. İnternet: Web-traffic-generator. Web: http://github.com/ecapuano/web-traffic-generator adresinden 21 Nisan 2019'da alınmıştır.

48. İnternet: Hping3. Web: http://www.hping.org/hping3.html adresinden 21 Nisan 2019'da alınmıştır.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : GÖKSEL, Nail
Uyruğu : T.C.
Doğum tarihi ve yeri : 25.091985, Eskişehir
Medeni hali : Evli
Telefon : 0 (541) 611 23 49
e-mail : nail.goksel@gazi.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Yüksek lisans	Gazi Üniversitesi / Bilgisayar Mühendisliği	Devam ediyor
Lisans	Gazi Üniversitesi / Bilgisayar Mühendisliği	2011

İş Deneyimi

Yıl	Yer	Görev
2017-Halen	Hazine ve Maliye Bakanlığı	Bilgisayar Mühendisi
2006-2017	TSK	Bilişim Uzmanı

Yabancı Dil

İngilizce

Yayınlar

Göksel, N., Demirci, M. (2019). *DoS Attack Detection using Packet Statistics in SDN*. 2019 International Symposium on Networks, Computers and Communications (ISNCC). İstanbul.

Hobiler

Yüzme, Fitness

GAZİ GELECEKTİR..