T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YAZILIM TANIMLI AĞ TABANLI GERÇEK ZAMANLI CİHAZ YÖNETİM UYGULAMASI

ANIL DURSUN İPEK – 031890131
BATUHAN ARSLANDAŞ – 032190097

TASARIM PROJESİ

BURSA 2023

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YAZILIM TANIMLI AĞ TABANLI GERÇEK ZAMANLI CİHAZ YÖNETİM UYGULAMASI

ANIL DURSUN İPEK – 031890131
BATUHAN ARSLANDAŞ – 032190097

PROJE DANIŞMANI: DOÇ. DR. MURTAZA CİCİOĞLU

ÖZET

Günümüzde kullanılan geleneksel ağlarda ağın yönetilememesi ve merkeziyetçi bir yaklaşım olmaması temel sorunlardandır. Ağ üzerinde birçok farklı satıcı standartlaşmamış cihazları kullanmakta ve karmaşık, yönetimi zor bir ağ yapısı oluşturmaktadır. Ağın giderek hızlı bir şekilde büyümesi karmaşıklığı arttırmaktadır. Bu soruna çözüm olarak düşünülen ve geleceğin ağ yapılarını oluşturacak olan yazılım tanımlı ağlar SmartDN projesinin temelini oluşturmaktadır. Projede yazılım tanımlı ağların merkeziyetçi, esnek, dinamik ve programlanabilir yapısı kullanılmaktadır. Bu dinamikler ile NSFNET topolojisi üzerinde kurgulanan senaryoda sunucu ve istemci arasında yapay zeka tabanlı yönlendirme algoritması önerilmektedir.

Mininet simülasyon ortamında Floodlight denetleyici ile kurgulanan topolojide 1 istemci, 1 sunucu ve 14 anahtar bulunmaktadır. Gerçekçi bir ağ ortamı sağlamak için ortama farklı alıcılar eklenmekte ve bu alıcılar arasında iperf3 aracı ile TCP trafiği oluşturulmaktadır. Oluşturulan test ortamında ffmpeg aracı ile video iletimi sağlanmakta ve elde edilen RTT, throughput, loss gibi yol bilgileri kayıt altına alınmaktadır. İletilen ve orijinal video arasındaki değişimi gözlemlemek için PSNR ve SSIM hesaplaması yapılmaktadır. Literatürde, oluşturduğumuz sanal ortama uygun veri seti olmaması nedeniyle sürekli olarak farklı koşullarda video iletimi yapılmakta ve elde edilen veriler ile bir veri seti oluşturulmaktadır.

Oluşturulan veri seti ile farklı makine öğrenme teknikleri kullanılarak eğitimler gerçekleştirilmektedir. Gerçekleştirilen eğitimler sonucunda makine öğrenmesi modelleri başarılı bir şekilde trafik düzeyini sınıflandırmaktadır. Burada trafik düzeyi düşük ve yüksek olarak 2 durumda ele alınmaktadır. Seçilen model yönlendirme algoritmasında kullanılmakta ve veri transferi öncesi trafik düzeyine göre aktarılacak yolu seçmektedir. Önerilen algoritma hopcount metrikli Floodlight denetleyici algoritması ile karşılaştırıldığında daha başarılı olmaktadır. Önerilen yaklaşım ile ağ üzerindeki trafik birçok parametre ile farklı boyutlardan ele alınmakta ve tespit edilebilmektedir.

ABSTRACT

In today's traditional networks, the inability to manage the network and the lack of a decentralized approach are fundamental challenges. The use of non-standardized devices from various vendors on the network creates a complex and difficult-to-manage infrastructure. The increasing growth of the network exacerbates this complexity. The solution envisioned for this issue is the foundation of the SmartDN project, which aims to shape the future network structures through Software-Defined Networking (SDN). The project employs the centralized, flexible, dynamic, and programmable nature of SDNs. In a scenario designed on the NSFNET topology, an artificial intelligence-based routing algorithm is proposed between servers and clients.

The topology is implemented using the Floodlight controller in the Mininet simulation environment, consisting of 1 client, 1 server, and 14 switches to provide a realistic network environment. Different receivers are added to the environment, and TCP traffic is generated among these receivers using the iperf3 tool. In the created test environment, video transmission is facilitated with the ffmpeg tool, and metrics such as Round-Trip Time (RTT), throughput, and loss are recorded. To observe the changes between the transmitted and original videos, Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) calculations are performed. Due to the lack of a suitable dataset for the virtual environment in the literature, continuous video transmissions are conducted under different conditions, and the obtained data is used to create a dataset.

Using the generated dataset, training is conducted with various machine learning techniques. The machine learning models successfully classify traffic levels, considering low and high traffic conditions. The selected model is employed in the routing algorithm, choosing the path for data transfer based on the traffic level before the transfer. When compared to the Floodlight controller algorithm using the hop count metric, the proposed algorithm proves to be more successful. This approach allows the traffic on the network to be addressed from various dimensions using multiple parameters, enabling its detection.

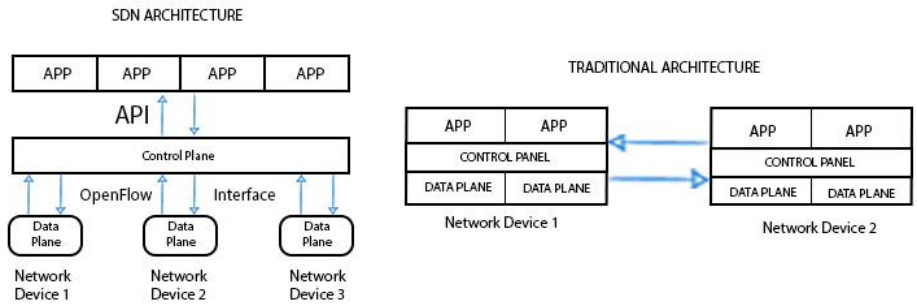
İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZGİLER DİZİNİ

1. GİRİŞ

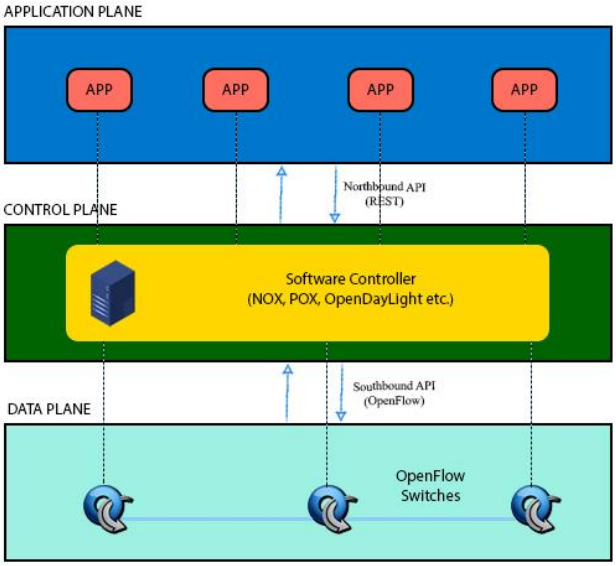
Bilgisayar ağlarının kullanımı gittikçe yaygınlaşmakta ve her geçen gün ağdaki cihaz sayısı parabolik bir artış göstermektedir. Cisco Annual Internet Raporu'na göre 2018 yılında 18,4 milyar olan ağa bağlı cihaz sayısının 2023 sonuna kadar 29,3 milyar olması beklenmektedir [1]. Bu gelişmeler ölçeğinde ağdaki trafik giderek artmakta ve teknolojide oluşan yeni gelişmeler ile bu artışın devam etmesi beklenmektedir. Cihaz sayısındaki bu artış ile günümüzde kullanılan geleneksel ağ sistemlerinde karmaşıklık artmakta ve yönetim zorlaşmaktadır. Bunun en temel sebebi ağdaki her bir cihazın kendi kontrol ve yönetim mekanizmasının olmasıdır. Bu cihazlarda iki farklı düzlem bulunmaktadır. Bunlar veri ve kontrol düzlemidir. Kontrol düzleminde her bir cihaz paket iletimi hakkında genel kararları vermektedir. Veri düzleminde kontrol düzleminde verilen karar uygulamakta ve paketleri fiziksel olarak iletmektedir. Buradaki temel sorun ağ üzerindeki her bir cihazın kendi kararlarını alması ve uygulamasıdır. Ek olarak ağ cihazlarının farklı satıcılar tarafından üretilmesi ve satıcıların kendi yöntemlerini kullanarak cihazları üretmesi ortak bir dil oluşmasını engellemekte ve geliştirme aşamasında satıcı firmalara bağımlılık oluşturmaktadır. Standartlaşmanın olmaması ise gelecekte yapılacak yeni güncellemeleri ağa entegre etmeyi zorlaştırmakta ve maliyeti arttırmaktadır.



Şekil 1. SDN ile geleneksel ağ mimarisi karşılaştırması

Mevcut geleneksel ağların sorunlarını çözmek için 2008 yılında Stanford Üniversitesi'nde yapılan çalışmalarda yazılım tanımlı ağlar (SDN) fikri ortaya atılmıştır. Bu ağların temel çıkış prensibi geleneksel ağlardaki ölçeklenebilirlik, yönetim, esneklik

ve maliyet sorunlarını ele almak ve yenilikçi bir yaklaşım getirmektir. SDN mimarisinde geleneksel ağ cihazlarındaki kontrol ve veri düzlemleri ayrılarak merkeziyetçi bir yaklaşım oluşturulmaktadır. Kontrol düzlemi ağı yöneten ve beyni olan denetleyici tarafından yönetilmektedir. Denetleyiciler işlem gücü çok yüksek bilgisayarlar olmakta ve ağ üzerindeki genel kararları almaktadırlar. Veri düzleminde ise ağdaki anahtarlar (switch), yönlendiriciler (router) ve diğer ara cihazlardan (middlebox) oluşmaktadır. Bu cihazlar kontrol düzleminde alınan kararları uygulamakta ve sadece fiziksel olarak paket iletiminde görev almaktadır. Şekil 1'de geleneksel ağ mimarisi ile yazılım tanımlı ağ mimarisinin karşılaştırılması ele alınmaktadır.[2][18]



Şekil 2. Yazılım tanımlı ağ mimarisi

Yazılım tanımlı ağların mimarisini detaylı olarak incelediğimizde 3 farklı katman görülmektedir. Bu katmanlar altyapı, kontrol ve uygulama katmanlarıdır. Altyapı ile kontrol katmanları arasında güneye giden arayüz bulunmaktadır. Güneye giden arayüzde en yaygın olarak Openflow protokolü kullanmaktadır. Bu protokol anahtarlar ile denetleyici arasında iletişimi kurmaktadır. Protokole göre ağ üzerindeki paket akışları anahtara geldiğinde ilk olarak kendi akış tablosunu kontrol etmektedir. Eğer paket ile ilgili bilgi kendi akış tablosunda yok ise ağdaki denetleyici ile iletişime geçerek denetleyicinin yeni bir kural tanımlamasını istemektedir. Denetleyici paketin durumu

hakkında karar verdikten sonra ilgili anahtarın akış tablosunu güncellemektedir. Protokol genel olarak akış tablolarındaki eşleşme-eylem durumuna göre paket iletimi üzerine kurgulanmaktadır. Kontrol ile uygulama katmanı arasında ise kuzeye giden arayüz bulunmaktadır. Kuzeye giden arayüz ile programcı ile denetleyici arasında bir iletişim kurulmakta ve denetleyicinin programlanabilmesi sağlanmaktadır. Bu arayüz sayesinde programcı ağ üzerinde genel bir görünümüne sahip olabilmekte, ağdaki cihazların mevcut durumunu kontrol edebilmekte ve ağda yeni kurallar belirleyebilmektedir. Günümüzde bu arayüzde hala bir standartlaşma bulunmaması yazılım tanımlı ağların genel sorunlarından biridir. Şekil 2'de Yazılım tanımlı ağların mimarisi görselleştirilerek anlatılmaktadır. [3]

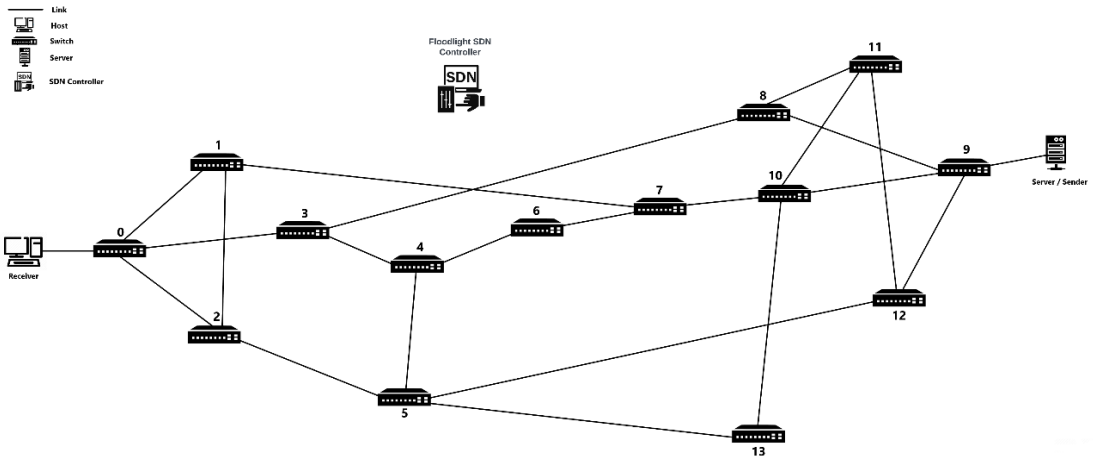
SDN fikrinin ortaya çıkması ile ağ üzerinde global bir görünüm elde edilebilmekte ve ağın programlanarak daha verimli ve optimize hale getirilmesi sağlanabilmektedir. Günümüzdeki ihtiyaçlardan biri de kaynak kullanımını optimize etmek ve kullanıcı hizmet kalitesini arttırabilmektedir. Geleneksel ağların veri merkezleri, içerik dağıtımlı ağlar vb. alanlarda servis hizmet kalitesi incelendiğinde global bir görünüme sahip olmaması nedeniyle yetersiz kaldığı görülmektedir. Geleneksel ağlarda servis kalitesini arttırmak ve ağı optimize etmek için kullanılan yük dengeleme ve yönlendirme metotları bulunmaktadır ancak yapılan araştırmalarda yazılım tanımlı ağlardaki yöntemlere göre daha az verimli olmaktadırlar [4].

Yük dengeleyici ve yönlendirme algoritmaları temel olarak benzer işlevlere sahip olmaktadırlar. Ağ üzerinde yanıt süresini en aza indirmek, kaynak kullanımını optimize etmek, darboğazlardan kaçınmak ve verimi maksimuma çıkarmak gibi temel görevleri bulunmaktadır. Bu çalışmada sanal bir simülasyon üzerinde sunucu ve istemci arasında veri transferi için yapay zeka tabanlı yönlendirme algoritması önerilmektedir. Algoritmanın tasarımında yazılım tanımlı ağların esnek, merkezi yapısı kullanılmaktadır. Yapay zeka modelleri ile dinamik bir şekilde ağın genel görünümü değerlendirilerek yapılmaktadır. Bu çalışma sayesinde ağdaki trafiğin azaltılması, bant genişliği kullanımının optimize edilmesi ve kullanıcı deneyiminin iyileştirilmesi hedeflenmektedir.

Yapay zeka, bilgisayar sistemlerine insan benzeri zeka özellikleri kazandırmayı amaçlayan bir yaklaşım olmaktadır. Yapay zekanın gelişimi, bilgisayarların güçlenmesi

ve büyük veri setlerinin kullanılabilir hale gelmesiyle hız kazanmakta olup makine öğrenimi ve derin öğrenme gibi alt alanlar yapay zekanın başarıyla uygulanmasını sağlamaktadır. Ağ dünyasındaki kullanımına gelince yapay zeka birçok alanda etkisini göstermektedir. Akıllı ev sistemleri, otomatik araçlar ve diğer IoT cihazlarıyla iletişim kurarak daha akıllı ağlar oluşturmaktadır. Ağ güvenliği konusunda saldırıları tespit etme, analiz etme ve önleme konularında önemli bir rol oynamaktadır. Anormal hareketleri algılayarak potansiyel tehditleri belirlemekte ve bu sayede ağları daha güvenli hale getirmektedir. Ayrıca ağ performansını optimize etmekte veri trafiğini yönetmekte, iletim hızını arttırmakta veya enerji kullanımını düzenleyerek ağların daha verimli çalışmasına katkı sağlamaktadır. Yönlendirme algoritmalarında yapay zeka karar modelleri kullanılarak sezgisel algoritmalar geliştirilebilmekte ve ağ üzerinde dinamik trafik yönlendirme sağlanabilmektedir. [5][19]

Çalışmada geliştirme ortamı olarak Floodlight denetleyici ile Mininet kullanılmaktadır. Mininet sanal ağ geliştirme ortamı sunan bir simülasyon aracıdır ve Python API desteği sunmaktadır. Floodlight java dilinde geliştirilmekte olan bir SDN denetleyicisidir. Floodlight REST API ile ağ denetleyici ile programcı arasında dil bağımsız geliştirme ortamı sunmaktadır. Projeyi geliştirme aşamasında trafik oluşturmak için iperf3 aracı kullanılmaktadır. Bu araç sayesinde ağ üzerinde UDP ve TCP trafikleri oluşturulabilmektedir. İçerik dağıtım ağında kullanıcı hizmet kalitesini ölçmek için ffmpeg aracı ile video aktarımı yapılmaktadır. Aktarım sonrasında video ilk hali karşılaştırılmakta ve sonuç olarak bir PSNR (Peak Signal-to-Noise Ratio) ve SSIM (Structural SIMilarity Index) değeri üretilmektedir. Elde edilen veriler ile video kalitesindeki değişim değerlendirilmektedir. Çalışma için kullanılan NSFNET topolojisi ve senaryo şekil 3’te gösterilmektedir.



Şekil 3. Mininet simülasyon ortamında oluşturulan NSFNET ağının şeması

Projemizin sağladığı ana katkılar şunlardır:

1. Kullanıcı Deneyiminin İyileştirilmesi: Veri transferinde kullanıcılara düşük gecikmeli ve daha kaliteli içerik sunulmasına imkan tanımaktadır.
2. Yapay Zeka ile Dinamik Trafik Yönlendirmesi: Ağ dünyasında sanal simülasyon ortamında oluşturulan veri seti ile eğitilen model otomatik karar almakta ve ağın dinamik olarak trafiğin optimize edilmesini sağlamaktadır.
3. Verimli Kaynak Kullanımı: Yapay zeka kaynakları daha etkili bir şekilde tahsis etmektedir. Yükün veri yolları arasında eşit bir şekilde dağıtılması gereksiz kaynak kullanımını önlemektedir.
4. Yeni Bir Veri Seti Oluşturulması: Daha önce mevcutta olmayan bir veri setini Mininet test ortamında oluşturarak mevcut literatüre katkıda bulunmaktadır.

Projemizin sağladığı ana katkıların yanı sıra uygulama esnasında çeşitli zorluklar içermektedir. Bu zorluklar şunlardır:

1. Uygulamanın Sanal Ortamda Gerçekleştirilmesi: Mininet üzerinde geliştirilen uygulamada sanal ortamda ağ cihazları arasında gerçek bir mesafenin olmaması ve sadece atlama sayıları referans alınarak sanal bir iletim yapılması uygulama kısmında bazı zorluklar oluşturmaktadır.
2. Sınırlı Veri Seti Olması: Uygulamada eğitilen modelde sanal ortama uygun veri seti oluşturulması gerekmektedir. Makine öğrenmesi ve derin öğrenme yöntemlerinde veri seti başarımın yüksek olması için büyük önem arz etmektedir.

Literatürde çalışmaya uygun veri seti bulunmaması nedeniyle test ortamında kendi veri setimizi oluşturmamız gerekmektedir.

3. Denetleyici CPU Tüketiminin İhmal Edilmesi: Trafik yönlendirmesi aşamasında yapay zeka modellerinin tahminde bulunması ve ağ trafik istatistik durumlarının anahtarlardan alınması gibi işlemlerde denetleyici ek rol almaktadır. Bu nedenle zamanla denetleyici yükü artmakta ve geleceğe yönelik olası problemler oluşmasına sebep olabilmektedir.
4. Yeterli Kaynak Bulunmaması: Geliştirme esnasında yaptığımız araştırmalara göre literatürde fazla sayıda teorik çalışma bulunmakta ancak uygulamaya yönelik araştırma sayısı az olmaktadır.
5. Yol Bilgisi Toplamada Fazla Zaman Harcanması: Simülasyon ortamında sınırlı sayıda araç kullanılarak ağ bilgisi toplanmaktadır. Bilgi toplama esnasında zaman göz ardı edilmektedir. Bu sorunun temel sebebi simülasyon ortamının belirli kısıtlarının olmasıdır.

Özetlemek gerekirse projemizde Mininet ortamında oluşturulan NSFNET topolojisi üzerinde gerçekleştirilmektedir. Bu topolojide belirlenen istemci ve sunucu arasında veri transferi öncesinde önerdiğimiz yapay zeka tabanlı yönlendirme algoritması çalıştırılmaktadır. Bu sayede kullanıcının servis kalitesi arttırılarak veri aktarımının en düşük trafikli ortamdan yapılması sağlanmaktadır. Bu algoritmanın başarımı hopcount metrikli Floodlight denetleyicisi ile karşılaştırılmakta ve sonuç değerler grafikler ile sunulmaktadır. Bölüm 2'de literatürde ilgili çalışma ile yapılan araştırmaların özeti yer almaktadır. Bölüm 3'te geliştirme için kullanılan araçlar ve geliştirme ortamı aktarılmaktadır. Bölüm 4'te kullanılan materyal ve yöntemler aktarılmakta ve son olarak bölüm 5'te sonuçlar aktarılmaktadır.

2. KAYNAK ARAŞTIRMASI

Ahmet W. [6] çalışmasında, klasik yük dengeleyici algoritmalarından olan eşit maliyetli çok yollu yönlendirme algoritması (ECMP) ile geliştirmiş olduğu bant genişliği yük dengeleyici arasında bir karşılaştırma sunmaktadır. ECMP ağ üzerinde statik olarak çalıştırılmakta ve gelen akışları eşit bir şekilde bir sonraki ağ cihazlarına aktarmaktadır. ECMP'nin en büyük dezavantajı gelen akışları ağın genel durumundan habersiz bir şekilde statik olarak aktarmasıdır. Araştırmada sunulan yaklaşım olan bant genişliği yük dengeleyicisinde ise akışlar denetleyici tarafından alınan bant genişliği istatistiklerine göre dinamik bir şekilde aktarılmaktadır. Bu yaklaşımda anahtarlar ağın genel durumundan haberdardır ve akışları trafik olmayan yöne iletmektedir. 2 algoritma Mininet ortamında Floodlight denetleyici kullanılarak oluşturulan topolojide karşılaştırılmaktadır. Sonuçlara göre yeni önerilen yaklaşım ağdaki verimi (throughput) 30% arttırmaktadır. Ancak önerilen yaklaşımın da bazı zorlukları bulunmaktadır. Yapılan testler küçük bir topoloji üzerinde yapılmaktadır. Daha büyük çaplı ağlara entegre edilebilmesi için algoritmanın geliştirilmesi gerekmektedir. Ek olarak testlerde denetleyici üzerinde CPU yükü ihmal edilmiştir. Yapılan çalışmalarda denetleyici üzerindeki işlem hacminin sınırlı olması ve kaynak kullanım durumlarında gecikmelere sebep olabilmesi bilinen bir sorundur. Bu tarz problemlerin çözümü için araştırmalar yapılması gerekmektedir.

Bilal B. ve Banu U. [7] çalışmalarında yapay zeka teknikleri kullanarak veri merkezlerinde yük dengelemesi üzerine araştırma yapmaktadır. Veri merkezlerinin kullanım yoğunluğu ve veri sayısındaki hızlı artış sonucu geleneksel yük dengeleme yöntemleri yetersiz kalmaktadır. Geleneksel yöntemlere alternatif olarak sunulan yazılım tanımlı ağlar ile veri merkezlerinde dinamik yük dengeleme mümkün olabilmektedir. Bu araştırma yazısında yapay zeka teknikleri (ANN, SVM, ML, DL) kullanılarak bir yük dengeleme hedeflenmiştir. Sunucular arasındaki gecikmelerin azalması ve genel sistem performansının artması beklenmektedir. Yazılım tanımlı ağlar kullanılarak sistem üzerinde oluşturulan topolojide yapılan testler ile bir veri seti oluşturulmuştur. Oluşturulan veri seti 800 adet eğitim 192 adet test verisinden oluşmaktadır. Yapılan eğitimler sonucunda en başarılı model yapay sinir ağları olmaktadır (ANN). Sisteme yapay zekanın entegre olması ile performansta iyileşme görülmektedir.

Ahmed H. Ve Ahmadreza M. [8] yazılım tanımlı ağlarda kullanılan yapay zeka tabanlı yük dengeleyicileri geniş bir bakış açısı ile incelemektedir. İlk olarak yük dengeleyicileri 4 sınıfa ayırmaktadır. Bunlar doğadan ilham alınan yük dengeleyicileri, makine öğrenmesi tabanlı yük dengeleyicileri, matematiksel model tabanlı yük dengeleyicileri ve diğer yük dengeleyicileridir. Her bir başlık altında 2017 yılından sonraki literatürde yapılan çalışmalar detaylı bir şekilde incelenmiştir. Araştırmaların avantajlı ve dezavantajlı olduğu kısımlar vurgulanarak bir karşılaştırma tablosu oluşturulmuştur. Yapılan çalışmalarda yük dengeleme için kullanılan parametreler kullanım sıklığına göre sıralanmakta ve gelecek çalışmalara referans olabilecek şekilde sunulmaktadır. Araştırmalara göre en çok kullanılan 5 ağ parametresi verim (throughput), yanıt süresi (response time), gecikme (latency), iş yükü derecesi (work load degree) ve paket kayıp oranıdır (packet loss ratio). Sonuç olarak önceki çalışmalarda yeteri kadar ağ kalite parametresi (QoS) kullanılmadığı vurgulanmakta ve gelecekteki çalışmalara güzel bir bakış açısı sunmaktadır.

Mohamed I. Hamed ve diğerlerinin [9] çalışmasında geleneksel yük dengeleyicilerin esneklik eksikliği ve ağ akışlarını yönetme gibi sorunlarının olduğu ele alınmakta. Makale, geleneksel yük dengeleyicilerin sorunlarına karşı bir çözüm olarak Yazılım Tanımlı Ağ (SDN) kullanımını anlatmaktadır. Geleneksel yük dengeleyicilerin esneklik eksikliği ve maliyetli donanım gereksinimlerinden dolayı, SDN'in merkezi kontrol ve programlanabilirlik sağlayarak bu tarz problemlerin çözülebileceği öne sürülmektedir. Bant genişliği tabanlı bir yük dengeleme yaklaşımı önerilmekte olup bu yaklaşım ağ taleplerini sunucular arasında sunucuların bant genişliği tüketimini baz alarak dağıtmaktadır. Önerilen yaklaşımın çalışmaları sonucunda Round robin ve Bağlantılar temelli yük dengeleme şemalarına göre daha iyi performans gösterdiğini Mininet simülasyon ortamı ve Raspberry Pi uygulaması altında değerlendirmektedir. Bu çalışma, SDN'in geleneksel yük dengeleyici sorunlarına karşı ekonomik ve esnek bir çözüm sunabileceğini ortaya koymaktadır.

Mosab H. ve arkadaşları [4] yük dengeleme tekniklerini konu alan geniş bir araştırma yazısı yazmışlardır. Başlangıç olarak yazılım tanımlı ağların geleneksel ağlara göre avantajlarından bahsettikten sonra yük dengeleyicilerin yanıt süresini en aza indirmek, kaynak kullanımını optimize etmek, darboğazlardan kaçınmak ve verimi maksimuma çıkarmak gibi temel objektiflerinden bahsetmektedirler. Yük dengeleyicileri

aralarında veri düzlemi yük dengeleyicisi ve kontrol düzlemi yük dengeleyicisi olarak sınıflandırılmaktadırlar. Veri düzlemi sunucu ve bağlantı yük dengeleyici olarak alt başlıklara indirgenmektedir. Kontrol düzlemi mantıksal olarak merkezileştirilmiş, fiziksel olarak dağıtılmış ve sanallaştırılmış yük dengeleyici olarak alt başlıklara indirgenmektedir. Yapılan araştırmalarda yazılım tanımlı ağ tabanlı yük dengeleyicilerin geleneksel yük dengeleyicilerden daha başarılı olduğu saptanmıştır. Ek olarak yük dengeleme yöntemlerinde kullanılan ağ parametreler ile ilgili kısa bir özet sunulmakta ve yazılım tanımlı ağlardaki güvenlik sorunları ele alınmaktadır.

Mohammad R. B. ve diğerlerinin [10] çalışmasında bulut kullanıcılarından gelen hizmet taleplerinin artması ağ üzerinde oluşan yoğunluğu ve yükü oldukça arttırmakta. Bu durum sistemin verimliliğini düşürüp yavaşlamasını neden olur. Sistemin daha verimli olabilmesi için çeşitli yükleri ve artan ihtiyaçları işleyebilmeli ve otomatik ölçüm sistemlerine bağımlılıklarını göstermesi gerekmektedir. Bu konuda yapay zeka optimizasyon tekniklerinin SDN'deki yük dengeleme sorunlarına nasıl etkili bir şekilde çözüm getirebileceğini araştırmak amacıyla Ant Colony Optimization (ACO) ve Particle Swarm Optimization (PSO) gibi iki yapay zeka optimizasyon tekniğini inceleyen bir araştırma sunulmaktadır. ACO ve PSO'nun performansını karşılaştırmak için bir deneyde kullanılan 6 switch ve 8 host içeren bir SDN ağ topolojisi üzerinde durulmakta olup bu analizde PSO'nun daha düşük gecikme ve daha iyi performans gösterdiği gözlemlenmiştir. Mevcut yapay zeka optimizasyon tekniklerinde değişikliklerle performansın artabileceği ve yük dengeleme yeteneklerini geliştirebilecek bir yol göstermektedir. Sonuç olarak SDN'de düğüm ve bağlantı güvenilirliğini değerlendiren bir teknik önerilmektedir.

Ekber Ç. K. [11] 5G ağının eMBB (Enhanced Mobile Broadband) içeriği için kullanıcı erişim protokolüne dayalı yeni bir yönlendirme algoritmasına sahip mimari önermektedir. Yaptığı çalışmaları Mininet ortamında Floodlight denetleyici kullanarak gerçekleştirmektedir. Oluşturduğu CDN topolojisinde 2 adet önbellek sunucu, 1 adet alıcı ve trafik oluşturmak için oluşturulan 6 adet ek alıcı bulunmaktadır. Sunuculardan birisi alıcıya çok yakın konumdadır ancak yol üzerinde TCP veya UDP trafiği bulunmaktadır. Diğer sunucu ise trafiksiz ortamdadır ancak alıcı ile arasında daha fazla mesafe vardır. Yaptığı çalışmada alıcı sunuculardan video içeriği talep etmektedir. Aktarılan videoların aktarıldıktan önce ve sonraki halleri karşılaştırılarak video kalitesindeki değişim

gözlemlenmektedir. Bu aşamada kullanılan kalite parametresi PSNR'dır (Peak Signal Noise Ratio). Çalışmada temel olarak en kısa mesafeyi bulan Djkstra algoritması kullanılmaktadır. Araştırmacı trafik yoğunluğuna göre yönlendirme yapan algoritmasını topolojideki atlama sayısını göz önünde bulunduran geleneksel DNS tabanlı yönlendirme ile karşılaştırmakta ve %60 daha iyi PSNR değeri verdiği sonucuna ulaşmaktadır.

Ayşe N. T. ve Mevlüt E. [12] çalışmasında yazılım tanımlı ağlarda meydana gelen güvenlik tehditlerini makine öğrenmesi algoritmaları ile sınıflandırmaktadır. Mininet üzerinde oluşturulan topolojide DDOS atakları gönderilerek saldırılar gerçekleştirilmiştir. Gelen saldırı paketleri tehdit var (1) ve tehdit yok (0) olarak ikili sınıflandırma yapılmıştır. Mevcut ağ üzerindeki testler ile veri seti oluşturulmuş ve bu veri seti dış kaynaklı başka bir veri seti ile birleştirilerek karma bir veri seti elde edilmiştir. Sınıflandırma işlemlerinde Yapay Sinir Ağları, Rastgele Orman, XGBoost, AdaBoost ve GradientBoost teknikleri kullanılmış ve bu teknikler arasında performans karşılaştırması yapılmıştır. Yapılan karşılaştırmalara göre genel olarak algoritmaların başarımının yüksek olduğu saptanmıştır. Kullandıkları veri seti üzerinde en yüksek doğruluk oranına sahip algoritmanın Rastgele Orman olduğu sonucuna ulaşılmıştır.

Peter B. ve arkadaşları [13] çalışmasında içerik dağıtım ağında (CDN) vekil sunucular arasındaki yükün eşit bir şekilde dağılmasını sağlamak için bir araştırma yapmaktadır. Yaptıkları araştırmada yazılım tanımlı ağların esnek yapısını ve ağın genel durumuna hakim olabilmesini kullanarak bir yapay zeka modeli geliştirilmektedir. Ağ üzerindeki bant genişliğini, atlama sayısı ve yol üzerindeki diğer ağ parametreleri elde edilerek yapay zeka modeline girdi olarak verilmiştir. Elde edilen veriler Naif Bayes, Karar Ağacı, Rastgele Orman, Lojistik Regresyon ve K en yakın komşu algoritmaları ile eğitilmiştir. Eğitim sonuçlarında Karar Ağacı algoritması diğer algoritmalara göre yüksek başarı göstererek 97% doğruluk oranı ile tahmin yapabilmektedir. Araştırma sonuçlarına göre klasik en yakın sunucu yaklaşımı ile yapay zeka destekli yük dengeleme sistemi karşılaştırmakta ve yapay zekanın sistem üzerinde olumlu etkileri görülmektedir.

Thabo S. ve arkadaşları [14] yük dengeleme yöntemleri ile ilgili geniş bir araştırma sunmaktadır. Yaptıkları araştırmalara göre yük dengelemeyi 5 ana başlık altında incelemektedirler. Bunlar denetleyici yük dengeleme, sunucu yük dengeleme, kablosuz bağlantılarda yük dengeleme, iletişim yolu seçimi yük dengeleme, yapay zeka tabanlı yük

dengelemedir. Bu alt başlıklarda genel literatürde yapılan çalışmalar ele alınmakta ve çalışmaların zayıf ve güçlü yönleri tartışılmaktadır. Bunlara ek olarak araştırmanın sonunda projelerde kullanılan performans ölçütleri ve uygulama araçları incelenmiştir. Araştırmalarda %93'ünde gecikme, %87'sinde verim, %100 QoS ve %73'ünde karmaşıklık parametreleri kullanılmaktadır. Uygulama araçları incelendiğinde ise araştırmacıların %55'i önerdikleri yaklaşımları Mininet ortamında test etmektedirler. Ağ trafiği oluşturmak için ise en çok kullanılan araç iperf3 olmaktadır.

Cui C. ve Xu Y. [15] yazılım tanımlı ağların küresel görünümünden yararlanarak farklı alt ağlar arasında yükü eşit dağıtmayı hedeflemektedir. Yaptıkları çalışmada bant genişliği kullanım oranı, paket kayıp oranı, iletim gecikmesi ve atlama sayısı kullanılarak bir yapay sinir ağı geliştirilmektedir. Geliştirilen yapay zeka modeli statik round robin algoritması ile karşılaştırılmaktadır. Sonuçlara ağ gecikmesinin en fazla %19,3 düştüğü gözlemlenmekte ve sistemde iyileşme olduğu saptanmaktadır.

Şükran D. ve Derya Y. [16] yaptıkları çalışmada yazılım tanımlı ağ denetleyicisi olarak Floodlight kullanmıştır. Elde edilen ağ akış bilgileri ile veri seti oluşturulmuştur. RX_PAKET, TX_PAKET, RX_BYTE ve TX_BYTE metrikleri üzerinden en yoğun paket trafiği belirlenmiş ve bu bilgiler yapay zekâ optimizasyon teknikleriyle analiz edilmiştir. Optimizasyon aşamasında, doğrusal arama metodu ve yapay zekâ teknikleri olan tabu arama ve tavlama benzetimi algoritmaları kullanılmıştır. Sonuçlar, geleneksel arama metodunun tabu arama metodundan daha iyi performans gösterdiğini, ancak önerilen karışım algoritmasının doğrusal arama metoduyla kıyaslandığında daha etkili olduğunu göstermektedir. Yapay zeka tekniklerinin her zaman en iyi başarımı vermediği gözlemlenmiştir.

Anteneh A. Gebremariam ve diğerlerinin [17] çalışmasında yapay zeka ve makine öğrenimi teknolojilerinin Yazılım Tanımlı Ağ (SDN) ve Ağ Fonksiyonlarını Sanallaştırma (NFV) temelli ağlardaki karmaşık sorunları çözme hedefinin üzerine odaklanılmıştır. Makalede SDN ve NFV tabanlı ağlarda makine öğrenmesinin başlıca uygulama alanlarını inceleyen kısa bir araştırma sunulmaktadır. Bu uygulama alanları ağ mimarisi, yük dengeleme ve kaynak kullanımı, bozulma tespiti ve yönetimi, ağ yönetimi ve işlemleri, ağ güvenliği ve ihlal tespiti olarak beş geniş kategoride sınıflandırılmıştır. SDN ve NFV'nin merkezi olmayan yapısı ağ bileşenlerinin sayısının çokluğu nedeniyle insanların

tüm bileşenleri yönetmesinin teknik olarak çok zor olacağı için bu durum sorun teşkil etmektedir. Bu durum ağ yönetimi için bir problem yaratmaktadır ve aynı zamanda ağların tek bir hatası ciddi güvenlik açıkları doğurabilme riskini bulundurmaktadır. Makine öğrenmesi ağları yöneterek kendi kendine uyarlanabilen ve kendi kendini yönetebilen bir yol sağlama konusunda yön göstermektedir. Ancak bu alandaki araştırmalar belirlenen bazı zorluklar sebebiyle oldukça azdır. Makalede de bu alandaki çalışmaların sayısının sınırlı olması nedeniyle bu incelemenin kısa bir özeti olduğu belirtilmiştir. Sonuç olarak belirli bir ağın boyutu ve trafiğiyle ilgili farklı makine öğrenmesi algoritmaları kullanılmıştır. Çıkan sonuçlarla birlikte gelecekte ağ elemanlarının hesaplama kaynaklarını ölçmesi planlanmaktadır.

3. GELİŞTİRME ORTAMI VE ARAÇLAR

Proje ücretsiz, açık kaynaklı, hafif ve verimli bir işletim sistemi olması nedeniyle Ubuntu 20.04 işletim sisteminde geliştirilmektedir. Ubuntu VMware Workstation Pro aracı kullanılarak sanal makine üzerine kurulmuştur. Sanal makine için ayırılan kaynak miktarları ve test ortamının parametreleri tablo 1'de paylaşılmaktadır.

Tablo 1. Geliştirme ortamı özellikleri

İsim	**Özellik**
İşletim Sistemi	Ubuntu 20.04
RAM	8 GB
İşlemci	Intel I7 – 4 Çekirdek
SDN Denetleyici	Floodlight
Simülasyon Aracı	Mininet
Topoloji	NSFNET
Trafik yaratıcı	Iperf3, Ping
ML Platformu	Google Colab
Programlama Dili	Python3

3.1 Mininet Simülasyon Ortamı

Mininet, ağ protokollerini ve uygulamalarını test etmek için kullanılan açık kaynaklı bir emülasyon ve test platformudur. Yazılım tanımlı ağlarda yük dengeleme algoritmaları için literatürdeki yapılan çalışmaların 55%'lik kısmı Mininet ortamında yapılmaktadır [14]. Mininet, gerçek ağ donanımını fiziksel olarak kullanmadan, sanal ağ topolojileri oluşturmamıza ve ağ protokollerini simüle etmemize olanak tanır. Fiziksel olarak cihazlara ihtiyaç duymadan sanal bir ağ ortamında çalışacağımız için geliştirmek ve test etmek maliyeti düşürür. Mininet içinde oluşturabileceğiniz bazı temel topoloji çeşitleri Single Switch, Linear, Tree ve Full Mesh olarak örnek gösterilebilir. Bunlara ek olarak sağlamış olduğu Python API ile Mininet üzerinde istenilen şekilde topoloji oluşturulabilir. Oluşturulan topolojiye istenilen ip de denetleyici entegre edilebilir ve

denetleyici yardımı ile yönetilebilir. Remote denetleyici ile Python API kullanarak oluşturmuş olduğumuz topoloji kodu tablo 2'de örnek olarak paylaşılmaktadır. [20]

Tablo 2. Mininet ortamında örnek topoloji oluşturma kodu

```
1: class SingleSwitchTopo(Topo):
2:     for h in range(n):
3:         host = self.addHost('h%s' % (h+1), cpu=0.5/n)
4:         self.addLink(host, singleSwitch, bw=10, delay='5ms', loss=2, max_queue_size=1000, use_htb=True)
5: def run():
6:     topo = SingleSwitchTopo(n=4)
7:     net = Mininet(topo=topo, host=CPULimitedHost, link=TCLink, build=False, switch=OVSKernelSwitch, autoSetMacs=True, waitConnected=True)
8:     remote_ip = "127.0.0.1"
9:     net.addController('c1', controller=RemoteController, ip=remote_ip, port=6653, protocols="OpenFlow13")
10:    net.build()
11:    net.start()
12:    net.pingAll()
13: if __name__ == "__main__":
14:     run()
```

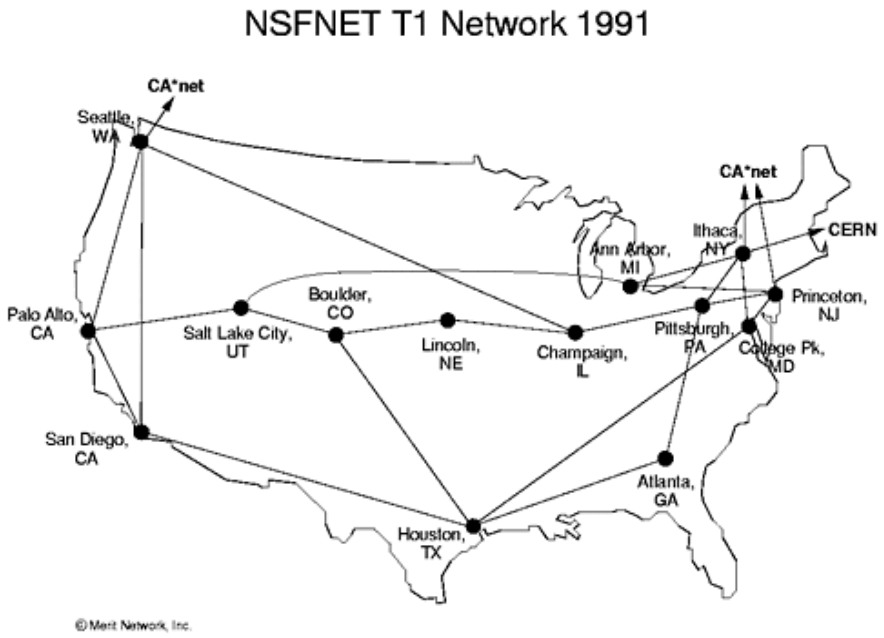
3.2 Floodlight SDN Denetleyici

Floodlight, açık kaynaklı bir SDN (Yazılım Tanımlı Ağ) kontrol yazılımıdır. SDN, ağdaki yönetim ve kontrolün ayrılmasına olanak tanıyan bir yaklaşımı temsil etmektedir. Floodlight, bu ağ kontrolünü sağlamak ve yönetmek için kullanılan bir kontrol düzlemidir. Floodlight, OpenFlow protokolünü destekleyen anahtarları yönetmek, ağ trafiğini kontrol etmek ve ağdaki kaynakları yönetmek için kullanılmaktadır. Ağdaki kaynakların dinamik olarak yönetilebilmesini sağlar, böylece ağ yöneticileri daha esnek ve ölçeklenebilir ağlar oluşturabilirler. Geliştiriciler için SDN'in genel ilkelerinden olan

programlanabilir ağ oluşması için programcılara REST API desteği sunmaktadır. REST API sayesinde dil bağımsız geliştirme yapmak mümkün olmaktadır. [21]

3.3 National Science Foundation Network

NSFNET (National Science Foundation Network), 1985 ile 1995 yılları arasında Amerika Birleşik Devletleri'nde kullanılan bir araştırma ve eğitim ağıdır. NSFNET'in topolojisi, bir dizi ana bilgisayar ve router üzerinden oluşmaktadır. Ağın topolojisi, çekirdek ağ ve bu ağa bağlı ara bağlantı noktaları (IPOP) üzerine kurulmuştur. Bu ağ yapısı, yüksek hızlı geniş alan bağlantıları ve ana bilgisayarlar arasında veri iletimini sağlamak için tasarlanmıştır. NSFNET, özellikle bilimsel ve eğitimsel alanlarda bilgi alışverişi için tasarlanmış bir ağ olup üniversiteler, araştırma kurumları ve devlet kurumları arasında veri iletimini sağlamak amacıyla kullanılmaktadır. TCP/IP protokolü üzerinde çalışılmıştır ve bu çalışma o dönemdeki internetin temel protokol setini oluşturmuştur. NSFNET dönemindeki internetin gelişimi için kritik bir altyapıyı temsil etmekte olup 1995 yılında yerini daha geniş ve modern internet altyapısına bırakmıştır. Şekil 4'te NSFNET ağının genel görünümü aktarılmaktadır. [22]

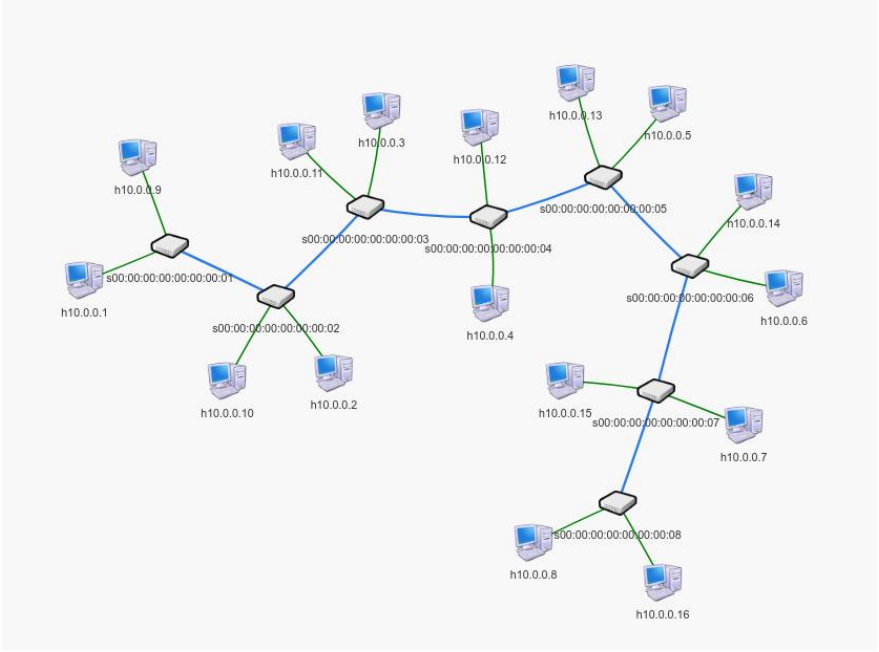


Şekil 4. NSFNET ağının şeması

3.4 İstemciler Arasında Trafik Oluşturma

Ağ ortamının gerçekçilik seviyesini arttırmak için farklı istemciler arasında sürekli olarak trafik oluşturulmaktadır. Bu işlemi otomatik bir şekilde sağmak için bu alanda en çok kullanılan araç olan iperf3 kullanılmaktadır. Iperf3, ağ bant genişliği ölçümü ve ağ performansı testleri yapmak için kullanılan bir araçtır. Sunucu-istemci modeli kullanarak ağ üzerinde veri iletimini simüle etmektedir ve bağlantı hızını, gecikmeyi ve paket kaybını ölçmek için kullanılmaktadır.

Bizim çalışmamızda sunucu ve istemci arasındaki yol üzerinde farklı istemciler konumlandırılmıştır. Bu istemciler arasında TCP paket transferi gerçekleştirilmektedir. Bu işlemin yapılışına örnek vermek için basit bir topoloji gösterilmekte ve terminal ekranları örnekleri ile açıklanmaktadır. Örnek topoloji şekil 5'te paylaşılmaktadır.



Şekil 5. Mininet Ortamında Örnek Lineer Topoloji

Örnek iperf3 ile TCP paketi gönderme işlemi:

Sunucu komutu:

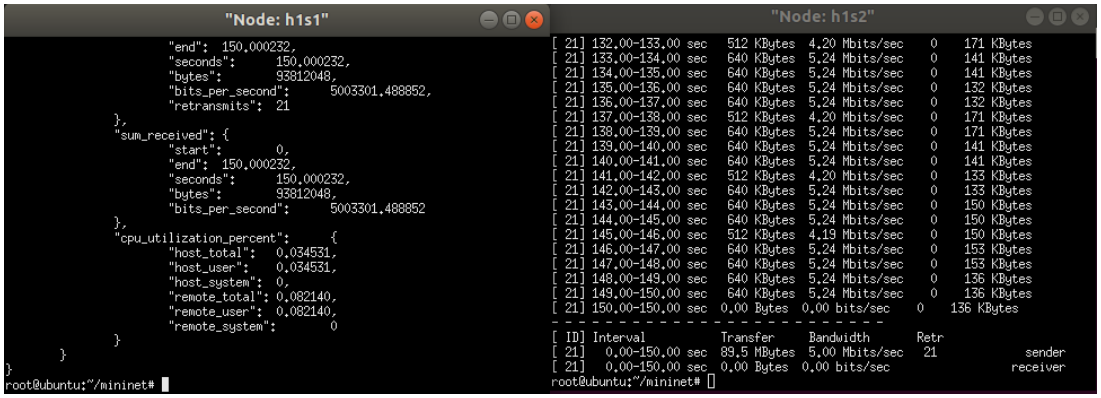
"iperf3 -s -p 5555 -i 1 -1"

(-s) komutu bir sunucu olduğunu ve 5555 portu (-p 5555) üzerinden TCP paketlerini alacağını belirtmektedir ve gelen paketlerin sonucunu ekrana yazdırmaktadır (-I 1)

İstemci komutu:

“iperf3 -c 10.0.0.9 -p 5555 -b 5M -R -t 150 -J”

10.0.0.9 ip adresli sunucunun (-c 10.0.0.9) 5555 portuna (-p 5555) saniyede 10 megabit/saniye hızında (-b 10G) 150 saniye boyunca (-t 150) TCP paketleri gerçek zamanlı (-R) göndermektedir. Gönderilen paketlerin sonuç değerlerini JSON (-J) tipinde geri döndürmektedir.



Şekil 6. İstemci-sunucu arası TCP paketi gönderme örnek terminal ekranları

3.5 Sunucu ve İstemci Arasında Video Aktarımı

FFmpeg, açık kaynaklı bir çoklu ortam çerçevesidir. Ses, video, altyazı ve diğer multimedya dosyalarını kodlama, kod çözme, dönüştürme ve yayınlama işlemleri için bir dizi araç ve kütüphane sağlamaktadır. FFmpeg, birçok farklı multimedya formatını destekler ve geniş bir kullanıcı tabanına sahiptir. Bu çerçeve, komut satırı aracılığıyla kullanılabilen bir dizi komut içerir.

Projede yapılan testlerde ise .mp4 uzantılı bir videoyu FFmpeg kütüphanesinin “ffmpeg -i video.ts output.ts” komutuyla .ts (Transport Stream) uzantısına çevrilmektedir. TS video ve ses verilerini içen bir taşıma formatıdır. Bu çevrilen videoyu istemciden sunucuya göndermek için şu adımlar izlenmektedir:

Sunucu Komutu:

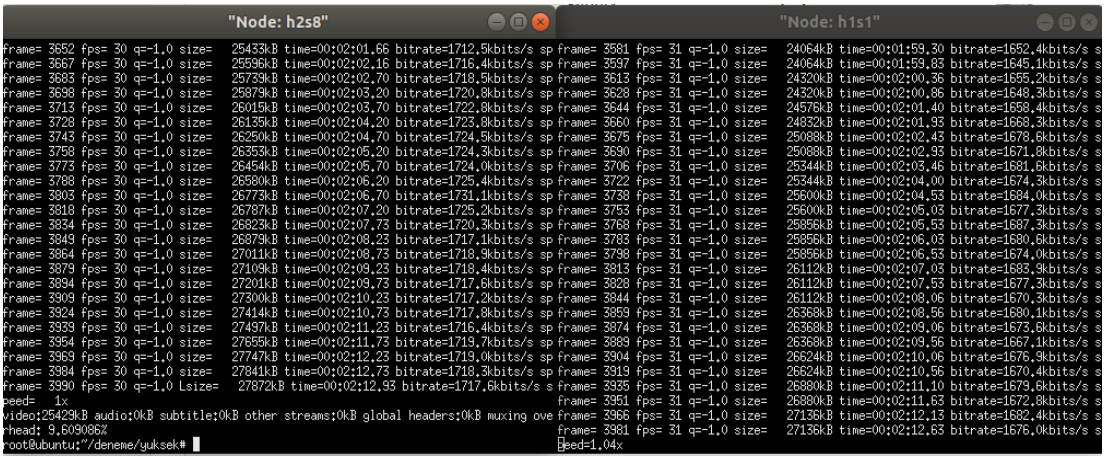
“ffmpeg -i udp://10.0.0.1:1234 -c copy input.ts”

Sunucu “udp://10.0.0.1:1234” ipli adrese (-I udp://10.0.0.1:1234) gelen UDP paketlerini alıp bulunduğu dizine input.ts adında bir dosya oluşturarak içeriğine paketleri kaydeder (-c copy input.ts).

İstemci Komutu:

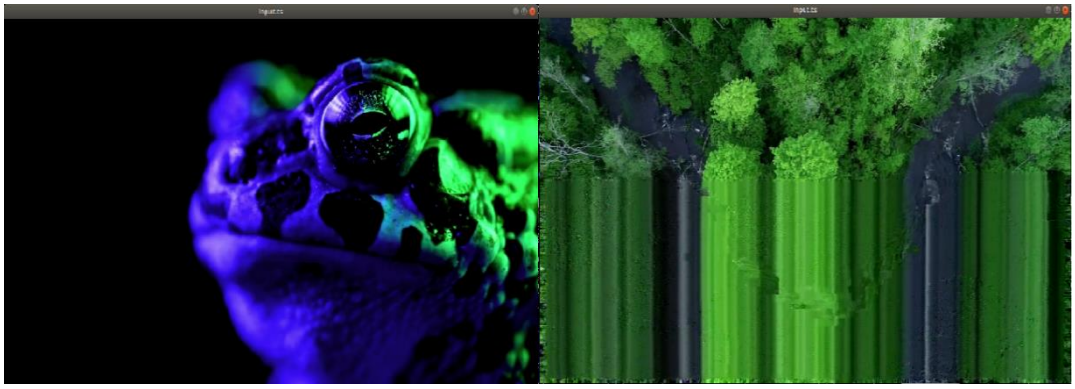
“ffmpeg -re -i video.ts -c copy -f mpegts udp://10.0.0.1:1234”

Dönüştürülen “video.ts” dosyasını (-i video.ts) normal hızda okuyup (-re) videoyu olduğu gibi (-c copy) ts formatında (-f mpegts) “udp://10.0.0.1:1234” adresine aktarmaktadır.



Şekil 7. Sunucu ve alıcı arasındaki video akışının terminal çıktıları

Orjinal videodan (video.ts) gönderilen kare sayısı 3990 iken sunucu tarafından alınan videonun (input.ts) kare sayısı 3881’dir. Gönderim sırasında 9 kare kayıp oluşmuştur.



Şekil 8. Aktarım sonrası elde edilen videodaki normal ve bozulmaya uğramış kareler

3.6 NFStream

NFStream ağ verileri ile çalışmayı kolay ve sezgisel hale getirmek için tasarlanmış bir Python çerçevesidir. [23] Pratik bir şekilde ağ üzerinde veri analizi yapma imkanı sunmaktadır. Projemizde ağ üzerindeki bir arayüz NFStream ile dinlenmekte ve paketler hakkında istatiksel veriler elde edilmektedir. Dinleme esnasında mevcut paketlerin kopyasını oluşturmakta ve ağa herhangi bir şekilde müdahale etmemektedir. Kurmuş olduğumuz yapay zeka modelinde NFStream ile 33 adet özellik elde edilmekte ve çalışmamızın devamında bu özelliklerin ağ üzerindeki trafik düzeyine etkileri incelenmektedir.

Çalışmamızda NFSteamer nesnesi oluşturulurken idle_timeout özelliği 15 saniye ve statistical_analysis True olarak ayarlanmıştır. Diğer özellikler varsayılan olarak kullanılmıştır.

3.7 Ping Komutu

Ping, bağlantı, erişilebilirlik ve ad çözümleme sorunlarını gidermek için kullanılan birincil TCP/IP komutudur. Bu komut bir anasistemden ya da ağ geçidinden ICMP ECHO_RESPONSE almak için bir ICMP ECHO_REQUEST (Internet Denetim İletisi İletişim Kuralı) gönderir. Bu komut 2 cihaz arasındaki bağlantıyı kontrol ve test etmek için kullanılabilmektedir.

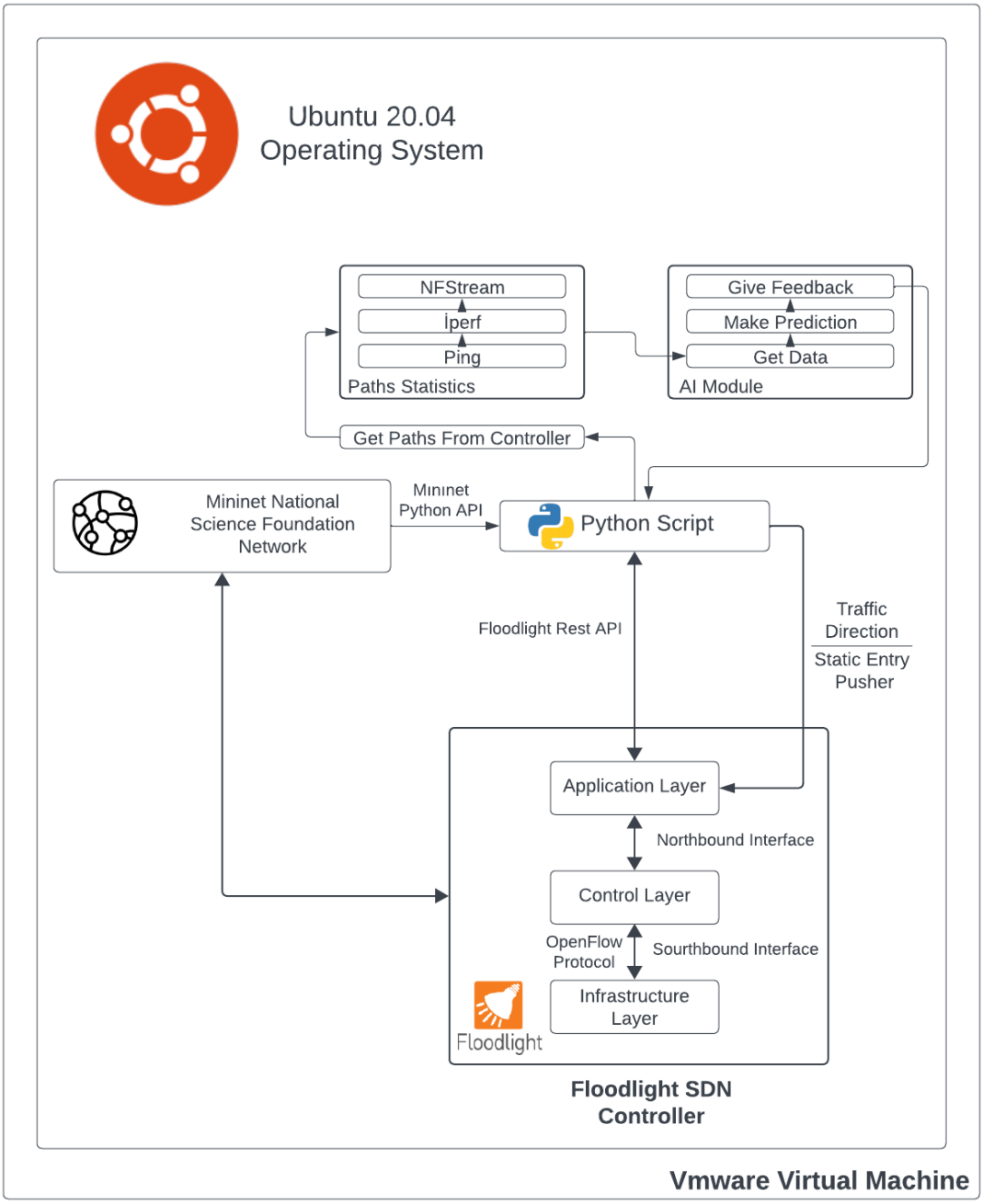
2 cihaz arasında saniyede bir veri paketi göndermekte ve alınan her yanıt için bir satır çıkış yazdırmaktadır. Ek olarak gidiş dönüş sürelerini ve paket kaybı istatistiklerini hesaplamakta ve kısa bir özet çıktı görüntülemektedir. Yaptığımız çalışmada kullanılan ping komutu ve parametreleri bir sonraki satırda örnek olarak paylaşılmaktadır.

“ping 10.0.0.1 -c 5 -f”

- 10.0.0.1: Ping komutunun hedef IP adresidir.
- -c 5: Gönderilecek paket sayısının 5 olduğunu belirtmektedir.
- -f: Ağın durumunu test etmek için tüm paketleri akış olarak göndermekte ve bağlantı durumunu ölçmektedir.

4. MATERYAL VE YÖNTEM

Proje tasarlanırken geliştirilen sistem akış şeması şekil 9'da gösterilmektedir.



Şekil 9. Sistem Akış Şeması

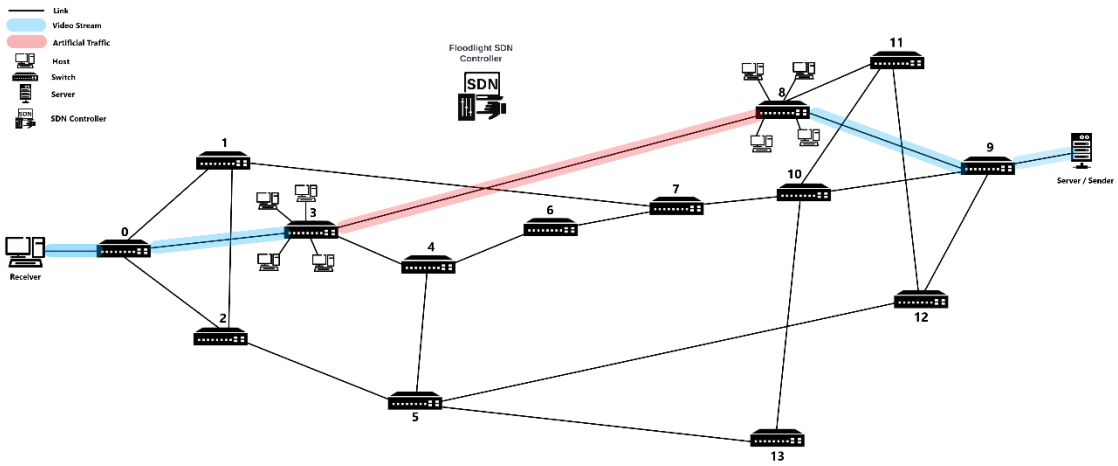
4.1 NSFNET Ağında Video Aktarım Senaryolarının Oluşturulması

Mininet ortamında kurulan NSFNET ağı üzerinde veri elde edilmesi için belirli senaryolar oluşturulmuştur. Senaryolar temel olarak ağ üzerindeki 3 farklı yol üzerinde veri göndermeye dayanmaktadır. Bu yollar üzerinde ise 2 farklı trafik düzeyi oluşturulmuştur. Senaryonun kurgulandığı ağın ana görseli ve cihazların isimlendirilmesi giriş bölümünde şekil 3'te gösterilmektedir.

Ağ üzerinde toplam 14 adet anahtar, 1 adet istemci ve 1 adet sunucu bulunmaktadır. Ağ üzerindeki tüm anahtarları kontrol eden 1 adet de Floodlight SDN Denetleyici bulunmaktadır. Bu ağ cihazları kurgulanan senaryolarda temeli oluşturmaktadır. Senaryolar kurgulanırken ağa farklı istemciler eklenmekte ve aralarında trafik oluşturulmaktadır.

4.1.1 Video Aktarımı Birinci Senaryo

İlk senaryoda istemci ile sunucu arasında en kısa yol temel alınmıştır. Şekil Bahsedilen ağ üzerindeki en kısa yol 0 – 3 – 8 – 9 anahtarlarını takip etmekte ve toplam 3 adet hop ile hedefe ulaşmaktadır. Bizim kurgulamış olduğumuz ilk senaryonun görseli şekil 10'da paylaşılmaktadır.



Şekil 10. İlk video aktarım senaryosunun görselleştirilmesi

Senaryoda alıcı ile gönderici arasında 2 farklı trafik düzeyi oluşturulmakta ve 3 ve 8 numaralı anahtarlara 4'er adet istemci atanmaktadır. Düşük trafikli ortamda seçilen 4 istemci aralarında 2 adet bağlantı kurarak TCP trafiği oluşturmakta ve bant genişliğinin %30 kadarını kullanmaktadır. Yüksek trafikli ortamda 8 adet istemci aralarında 4 adet bağlantı kurarak TCP trafiği oluşturmakta ve bant genişliğinin yaklaşık olarak %90 kadarını kullanmaktadır. Trafik oluşturulduktan sonra ortamda ana istemci ile sunucu arasında video aktarımı olmaktadır. Düşük ve yüksek trafik oluşturmak için eklenen istemci bilgileri tablo 3 ve 4'te açıklanmaktadır.

Tablo 3. Düşük trafikli ortamda örnek trafik bilgileri

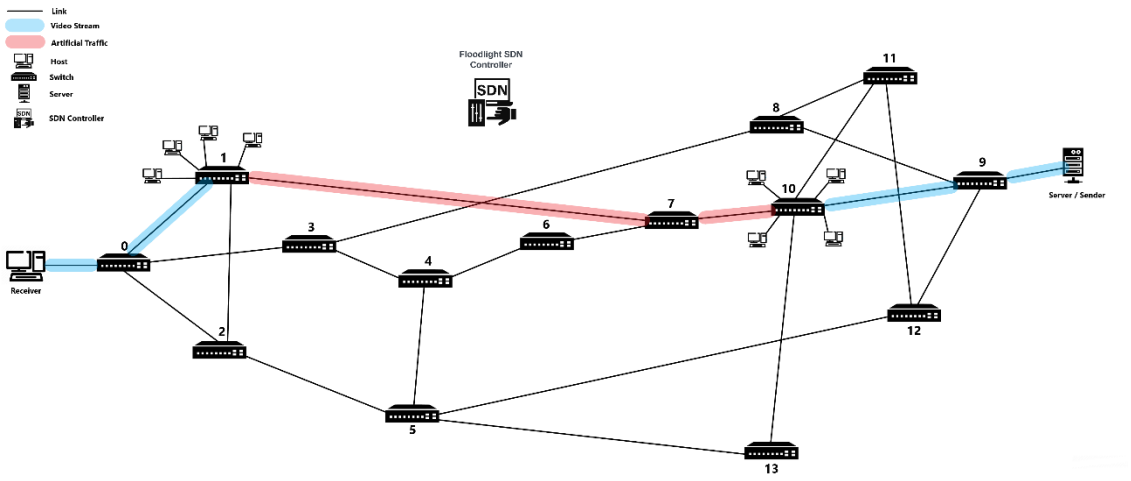
DÜŞÜK TRAFİK			
KAYNAK IP	**HEDEF IP**	**BANT GENİŞLİĞİ**	**PROTOKOL**
10.0.0.9	10.0.0.4	1500K	TCP
10.0.0.17	10.0.0.15	1500K	TCP

Tablo 4. Yüksek trafikli ortamda örnek trafik bilgileri

YÜKSEK TRAFİK			
KAYNAK IP	**HEDEF IP**	**BANT GENİŞLİĞİ**	**PROTOKOL**
10.0.0.9	10.0.0.4	2250K	TCP
10.0.0.17	10.0.0.15	2250K	TCP
10.0.0.18	10.0.0.16	2250K	TCP
10.0.0.21	10.0.0.19	2250K	TCP

4.1.2 Video Aktarımı İkinci Senaryo

İkinci senaryoda hop sayısı bir arttırılarak 4 hop uzunluğunda bir yol tercih edilmektedir. İstemci ile sunucu arasındaki video aktarımı 0 – 1 – 7 – 10 – 9 anahtarlarını takip etmektedir. Bu yol üzerinde de 2 farklı trafik düzeyi oluşturulmakta ve senaryonun görselleştirilmesi şekil 11'de aktarılmaktadır.

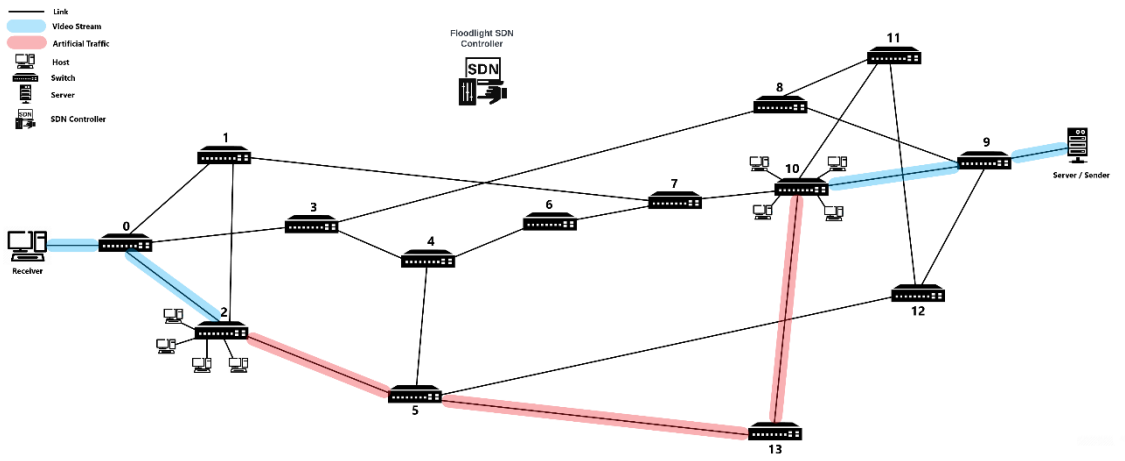


Şekil 11. İkinci video aktarım senaryosunun görselleştirilmesi

İkinci senaryoda 1 ve 10 numaralı anahtarlara 4'er adet istemci atanmaktadır. Düşük trafikli ortamda 4 adet istemci arasında ilk senaryoya benzer şekilde 2 adet bağlantı kurarak TPC trafiği oluşturmakta ve bant genişliğinin %30 kadarını kullanmaktadır. Yüksek trafikli ortamda 8 adet istemci aralarında 4 adet bağlantı kurarak TCP trafiği oluşturmakta ve yaklaşık olarak bant genişliğinin %90 kadarını kullanmaktadır. İki durumda da video aktarımı yapılarak testler gerçekleştirilmektedir.

4.1.3 Video Aktarımı Üçüncü Senaryo

Üçüncü senaryoda hop sayısı bir kez daha arttırılarak 5 hop uzunluğunda bir yol tercih edilmektedir. İstemci ile sunucu arasında video aktarımı 0 – 2 – 5 – 13 – 10 – 9 anahtarlarını takip etmektedir. Diğer senaryolara benzer şekilde bu senaryoda da düşük ve yüksek trafik oluşturulmakta ve aynı sayıda istemci, aynı miktarda bant genişlikleri kullanılmaktadır. Trafik oluşturacak istemciler 2 ve 10 numaralı anahtarlara atanmaktadır. Senaryonun görseli şekilde 12'de aktarılmaktadır.



Şekil 12. Üçüncü video aktarım senaryosunun görselleştirilmesi

4.2 Aktarım Sonrasında Video Kalitesinin Ölçülmesi

Oluşturulan senaryolar üzerinde istemci ile sunucu arasında video aktarımı yapılmaktadır. Aktarılan video 66 saniye uzunluğa ve 1920x1080 piksel çözünürlüğe sahip olmaktadır. Aktarım sonrasında kurgulanan trafiğin video kalitesi üzerinde etkilerinin ölçülmesi için 2 adet kalite parametresi kullanılmaktadır. Bunlar PSNR ve SSIM parametreleridir.

4.2.1 PSNR

PSNR Peak Signal-to-Noise Ratio (Pik Sinyal Gürültü Oranı), video kalitesini değerlendirmek için yaygın olarak kullanılan bir ölçüdür. Bu metrik, orijinal bir videonun üzerinde yapılan işlemler sonrasında elde edilen işlenmiş halinin, format değişiklikleri, sıkıştırma veya ağ üzerinde iletim gibi süreçlere karşı ne kadar başarılı olduğunu ölçer. PSNR hesaplamak için, ilk adım olarak MSE (Ortalama Kare Hata) (E.1.1) hesaplanır, bu da referans alınan orijinal video ile işlenmiş video karelerindeki piksel değerlerinin farklarının karesinin ortalamasıdır. Sonrasında, bu MSE değeri kullanılarak PSNR (E.1.2) Denklemi ile video kalitesi ölçülür.

$$\text{MSE}=\frac{1}{\text{m x n}} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [\, O(i,j) - D(i,j) \,]^2 \qquad \text{(E.1.1)}$$

$$\text{PSNR}= 10.log_{10} \left(\frac{MAX_1^2}{MSE} \right)$$

$$= 20.log_{10}\left(\frac{MAX_1}{\sqrt{MSE}}\right)$$

$$= 20.log_{10}\left(\frac{MAX_1}{\sqrt{MSE}}\right) \quad \text{(E.1.2)}$$

Bu denklemde MAX, piksellerin tepe değerini temsil eder ve genellikle pikseller 8 bit olarak ifade edildiğinde 255 olarak kabul edilir. Bu hesaplamaları ifade eden denklem:

$$\text{PSNR} = 48.1308036087 - 20.log_{10}\left(\frac{MAX_1}{\sqrt{MSE}}\right) \quad \text{(E.1.3)}$$

4.2.2 SSIM

SSIM (Structural Similarity Index), iki görüntü arasındaki benzerliği karşılaştıran bir ölçümdür. SSIM, insan görsel sistemine daha yakın sonuçlar elde etmeyi amaçlar ve görüntüler arasındaki yapısal, parlaklık ve kontrast benzerliği gibi ölçümlerini içerir. -1 ile 1 arasında değerler alır. 1'e yaklaşan değerler iki görüntü arasında yüksek benzerlik olduğunu ifade ederken -1'e yaklaşan değerler düşük benzerlik olduğunu ifade etmektedir

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)}. \quad \text{(E.2.1)}$$

- x ve y, karşılaştırılan iki görüntüdür.
- μx ve μy, görüntülerin piksel değerlerinin ortalama değerleridir.
- σx ve σy, piksel değerlerinin standart sapmalarıdır.
- σxy, x ve y arasındaki kovaryanstır.
- $C1$ ve $C2$, stabilite ve normalleştirme için küçük sabit değerlerdir.

4.3 Yol Bilgilerinin Elde Edilmesi

İstemci ile sunucu arasındaki aktarım yapılacak yol hakkında farklı araçlar kullanılarak bilgiler elde edilmektedir. Bu bilgiler seçilen bağlantı yolunun trafik düzeyini sınıflandırmamız için referans olmaktadır. Ek olarak aktarım öncesi ve sonrası trafik düzeyine göre oluşan etkileri ortaya çıkarmaktadır. Bilgilerin hangi araç ile elde edildiği, isimlendirmesi ve açıklamaları tablo 5'te aktarılmaktadır.

Tablo 5. Veri setinde oluşturmak için elde edilen değişkenler ve açıklamaları

Kullanılan Araç	Değişken	Açıklama
PING	average_rtt	RTT round trip time olarak bilinir. Bir bilgisayarın bir hedefe bir paket gönderip, hedefin bu paketi alıp cevap verene kadar geçen süreyi ifade eder. avarage_rtt ise belirli bir süre içinde gerçekleşen RTT değerlerinin ortalamasıdır.
	packet_loss	Ağda gönderilen paketlerinin hedefe ulaşamaması durumunu ifade eder. Yüzdelik kayıp paket durumunu ifade etmektedir.
IPERF3	bits_per_second (throughput)	1 saniye içinde iletilen veya transfer edilen bit sayısını belirtir.
	bu_ratio	Yol üzerindeki anlık bant genişliği kullanım oranını ifade etmektedir.
	retransmits	Ağ iletişimi sırasında gönderilen paketlerinin kaybolması veya eksik ulaşması durumunda gönderici tarafından tekrar iletilen paket sayısını ifade etmektedir.
	cpu_host_total	Bir bilgisayar sistemindeki toplam CPU kullanımını ifade etmektedir.
	cpu_host_user	Bilgisayar sisteminde CPU kullanımının kullanıcı tarafında gerçekleştirilen işlemlerle ilgili olan kısmını ifade etmektedir.
	cpu_host_system	Bilgisayar sisteminde CPU kullanımının sistem seviyesinde gerçekleştirilen işlemlerle ilgili olan kısmını ifade etmektedir.
	cpu_remote_total	Bir ağ üzerinden erişilen sunucudaki toplam CPU kullanımını ifade etmektedir.
	cpu_remote_user	Ağ üzerinden erişilen bir sunucudaki CPU kullanımının, kullanıcı tarafından gerçekleştirilen işlemlerle ilgili olan kısmını ifade etmektedir.

	cpu_remote_system	Ağ üzerinden erişilen bir sunucudaki CPU kullanımının, sistem seviyesinde gerçekleştirilen işlemlerle ilgili olan kısmını ifade etmektedir.
NFSTREAM	bidirectional_duration_ms	Bir iletişim bağlantısında her iki yönde gerçekleşen akış süresini ifade etmektedir.
	bidirectional_packets	Bir iletişim bağlantısında her iki yönde gerçekleşen veri paketlerini ifade etmektedir.
	bidirectional_bytes	Bir iletişim bağlantısında her iki yönde aktarılan toplam byte miktarını ifade etmektedir.
	src2dst_duration_ms	Ağ üzerinde kaynaktan hedefe gerçekleşen veri transferinin süresini milisaniye cinsinden ifade etmektedir.
	src2dst_packets	Ağ üzerinde kaynaktan hedefe gerçekleşen veri transferi sırasında gönderilen paket sayısını ifade etmektedir.
	src2dst_bytes	Ağ üzerinde kaynaktan hedefe gerçekleşen veri transferi sırasında gönderilen toplam byte miktarını ifade etmektedir.
	dst2src_duration_ms	Ağ üzerinde hedeften kaynağa gerçekleşen veri transferinin süresini milisaniye cinsinden ifade etmektedir.
	dst2src_packets	Ağ üzerinde hedeften kaynağa gerçekleşen veri transferi sırasında gönderilen paket sayısını ifade etmektedir.
	dst2src_bytes	Ağ üzerinde hedeften kaynağa gerçekleşen veri transferi sırasında gönderilen toplam byte miktarını ifade etmektedir.
	bidirectional_min_ps	Bir iletişim bağlantısında her iki yönde gerçekleşen veri transferinin

	minimum paket boyutunu ifade etmektedir.
bidirectional_mean_ps	Bir iletişim bağlantısında her iki yönde gerçekleşen veri transferinin ortalama paket boyutunu ifade etmektedir.
bidirectional_stddev_ps	Bir iletişim bağlantısında her iki yönde gerçekleşen veri transferinin paket boyutunun standart sapmasını ifade etmektedir.
bidirectional_max_ps	Bir iletişim bağlantısında her iki yönde gerçekleşen veri transferinin maksimum paket boyutunu ifade etmektedir.
src2dst_min_ps	Bir iletişim bağlantısında kaynaktan hedefe gerçekleşen veri transferinin minimum paket boyutunu ifade etmektedir.
src2dst_mean_ps	Bir iletişim bağlantısında kaynaktan hedefe gerçekleşen veri transferinin ortalama paket boyutunu ifade etmektedir.
src2dst_stddev_ps	Bir iletişim bağlantısında kaynaktan hedefe gerçekleşen veri transferinin paket boyutunun standart sapmasını ifade etmektedir.
src2dst_max_ps	Bir iletişim bağlantısında kaynaktan hedefe gerçekleşen veri transferinin maksimum paket boyutunu ifade etmektedir.
dst2src_min_ps	Bir iletişim bağlantısında hedeften kaynağa gerçekleşen veri transferinin minimum paket boyutunu ifade etmektedir.
dst2src_mean_ps	Bir iletişim bağlantısında hedeften kaynağa gerçekleşen veri transferinin ortalama paket boyutunu ifade etmektedir.
dst2src_stddev_ps	Bir iletişim bağlantısında hedeften kaynağa gerçekleşen veri

	transferinin paket boyutunun standart sapmasını ifade etmektedir.
dst2src_max_ps	Bir iletişim bağlantısında hedeften kaynağa gerçekleşen veri transferinin maksimum paket boyutunu ifade etmektedir.
bidirectional_min_piat_ms	Bir iletişim bağlantısında her iki yönde gerçekleşen veri transferinin minimum paket varış süresini ifade etmektedir.
bidirectional_mean_piat_ms	Bir iletişim bağlantısında her iki yönde gerçekleşen veri transferinin ortalama paket varış süresini ifade etmektedir.
bidirectional_stddev_piat_ms	Bir iletişim bağlantısında her iki yönde gerçekleşen veri transferinin paket varış sürelerinin standart sapmasını ifade etmektedir.
bidirectional_max_piat_ms	Bir iletişim bağlantısında her iki yönde gerçekleşen veri transferinin maksimum paket varış süresini ifade etmektedir.
src2dst_min_piat_ms	Bir iletişim bağlantısında kaynaktan hedefe gerçekleşen veri transferinin en küçük paket varış süresini ifade etmektedir.
src2dst_mean_piat_ms	Bir iletişim bağlantısında kaynaktan hedefe gerçekleşen veri transferinin ortalama paket varış süresini ifade etmektedir.
src2dst_stddev_piat_ms	Bir iletişim bağlantısında kaynaktan hedefe gerçekleşen veri transferinin paket varış sürelerinin standart sapmasını ifade etmektedir.
src2dst_max_piat_ms	Bir iletişim bağlantısında kaynaktan hedefe gerçekleşen veri transferinin maksimum paket varış süresini ifade etmektedir.

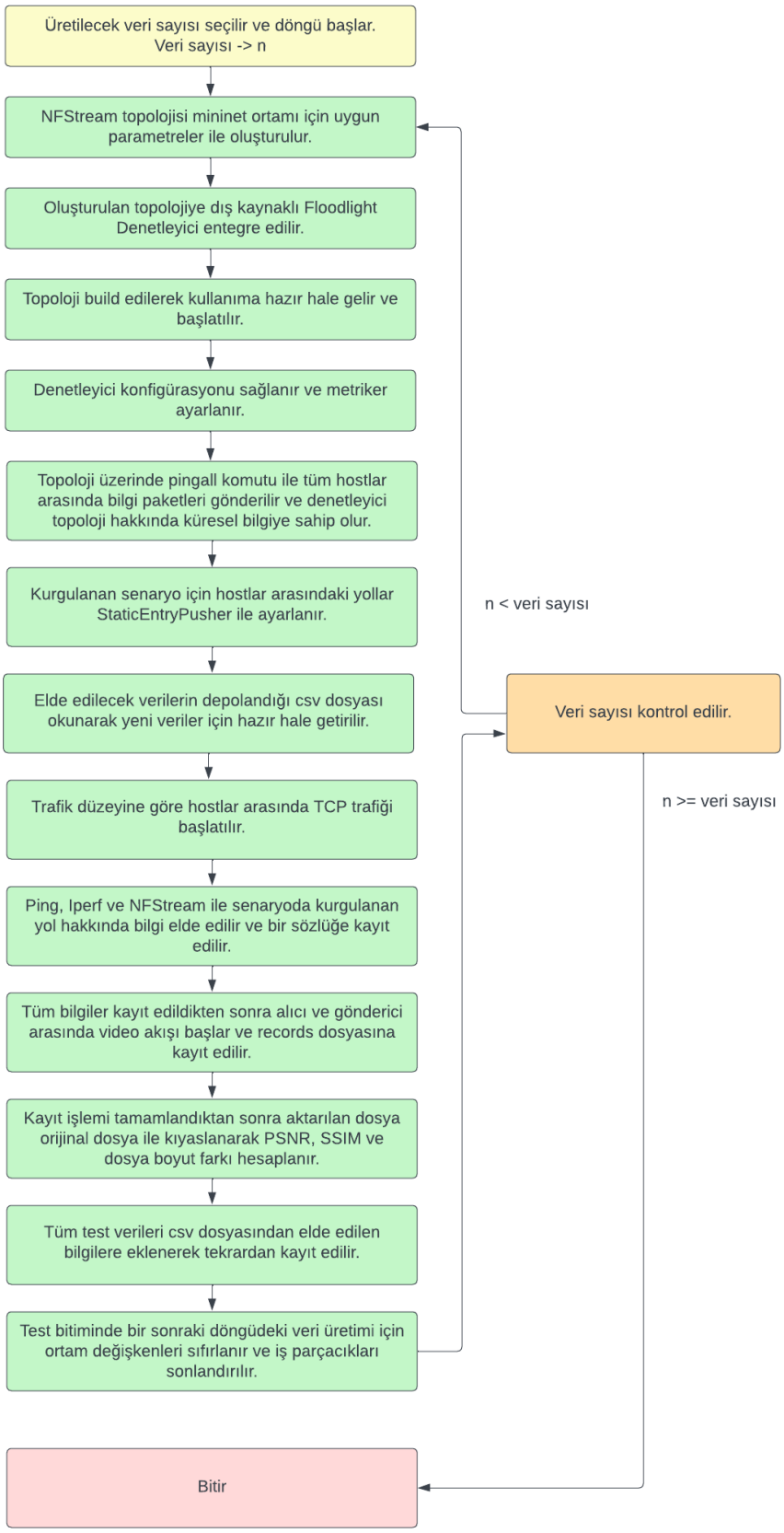
	dst2src_min_piat_ms	Bir iletişim bağlantısında hedeften kaynağa gerçekleşen veri transferinin en küçük paket varış süresini ifade etmektedir.
	dst2src_mean_piat_ms	Bir iletişim bağlantısında hedeften kaynağa gerçekleşen veri transferinin ortalama paket varış süresini ifade etmektedir.
	dst2src_stddev_piat_ms	Bir iletişim bağlantısında hedeften kaynağa gerçekleşen veri transferinin paket varış sürelerinin standart sapmasını ifade etmektedir.
	dst2src_max_piat_ms	Bir iletişim bağlantısında hedeften kaynağa gerçekleşen veri transferinin maksimum paket varış süresini ifade etmektedir.
FFMPEG	psnr	Bir görüntünün kalitesini ölçen bir metriktir. Görüntü sıkıştırma veya iletimi sırasında meydana gelen paket kaybını değerlendirmek için gönderilen görüntüyü, orijinal görüntüyle karşılaştırarak sonuç elde edilir.
	ssim	Aynı PSNR gibi görüntünün kalitesini ölçmeye yarar. PSNR a göre insan gözüne daha uygun bir ölçümdür ve yapısal benzerlik üzerine odaklanır.
LINUX COMMAND	original_file_size	Gönderilen görüntünün orijinal dosya boyutunu ifade eder.
	file_size	Gönderilmiş olan görüntünün dosya boyutunu ifade eder
FLOODLIGHT	hop_count	Veri iletiminin kaç atlayış (hop) yapıldığını ifade eder.

4.4 Kurgulanan Senaryolar ile Veri Üretilmesi

Mininet ortamında kurgulanan senaryolar ile veri üretimi yapılmaktadır. Verilerin otomatik bir şekilde üretilmesi için Python dilinde bir program yazılmıştır. Python dilinin

tercih edilmesinin temel sebebi Mininet ve diğer araçlarının Python dilini desteklemeleri ve kullanımı kolay olmasıdır.

Bahsedilen program içerisinde ağ topolojisi oluşturulmakta, Floodlight Denetleyici konfigürasyon ayarları yapılmakta, senaryo için kullanılacak yollar Floodlight StaticEntryPusher ile ayarlanmakta, yol üzerinde iperf3 ile trafik oluşturulmakta, trafik oluşturulan yolun parametreleri daha önce bahsedilen araçlar ile elde edilmekte, yolun iki ucundaki istemci ve sunucu arasında video akışı yapılmakta ve kayıt edilmekte, kayıt edilen videonun aktarım öncesi ve sonrası kalitesi test edilmekte ve son olarak sistem sıfırlanarak döngüsel bir şekilde çalıştırılmaktadır. Aynı anda birden fazla işlemi yapmak için program içerisinde birçok kez iş parçacıkları oluşturulmaktadır. Genel çalışma özeti bu şekilde olan programın akış şeması şekil 13'te paylaşılmaktadır.



Şekil 13. Veri üretimi programı mantıksal akış şeması

4.5 Yapay Zeka Tabanlı Yönlendirme Algoritması

Günümüzde, veri analizi ve makine öğrenimi, birçok endüstri için kritik bir rol oynamaktadır. Veri setlerinin karmaşıklığı ve hacmi arttıkça, etkili bir şekilde modelleme yapmak ve doğru tahminlerde bulunmak giderek önem kazanmaktadır. Bu bağlamda makine öğrenmesi algoritmaları, özellikle sınıflandırma görevlerinde etkili bir şekilde kullanılmaktadır.

Elde etmiş olduğumuz veriler analiz edilerek trafik düzeyini etkileyen değişkenler ile ikili sınıflandırma modelleri kullanılarak bir eğitim gerçekleştirilmiştir. Eğitim sonuçları ve verinin analizi araştırma sonuçları kısmında detaylı bir şekilde paylaşılmaktadır. Bu bölümde yapay zeka tabanlı yönlendirme algoritmasının mantığı aktarılmaktadır.

İstemci ve sunucu arasında video transferi yapılmadan önce yönlendirme algoritması çalıştırılmaktadır. Algoritma ilk aşamada yapay zeka modelini yüklemekte ve tahminde bulunmak için hazır hale getirmektedir. Yapay zeka modeli sisteme yüklendikten sonra istemci ve sunucu arasındaki 5 en kısa yol Floodlight denetleyicisinden alınmaktadır. Her bir yol StaticEntryPusher ile ayarlanarak gerekli anahtarların akış tablosuna eklenmekte ve 2 cihaz arasındaki akışın belirlenen yol üzerindeki istatistikleri toplanmaktadır. Toplanan istatistikler yapay zeka modeline girdi olarak verilmekte ve bir sonuç trafik düzeyi elde edilmektedir. Elde edilen trafik düzeyleri yol indeksi ile bir sözlüğe kaydedilmektedir. Tüm yollar için aynı işlem yapıldıktan sonra trafik düzeyi en düşük olan yol seçilmektedir. Seçilen yol üzerindeki ilgili anahtarların akış tablosu güncellenerek istemci ve sunucu arasındaki video transferin o yol üzerinden yapılması sağlanmaktadır. Bahsedilen yönlendirme algoritmasının sözde kodu tablo 6'da aktarılmaktadır.

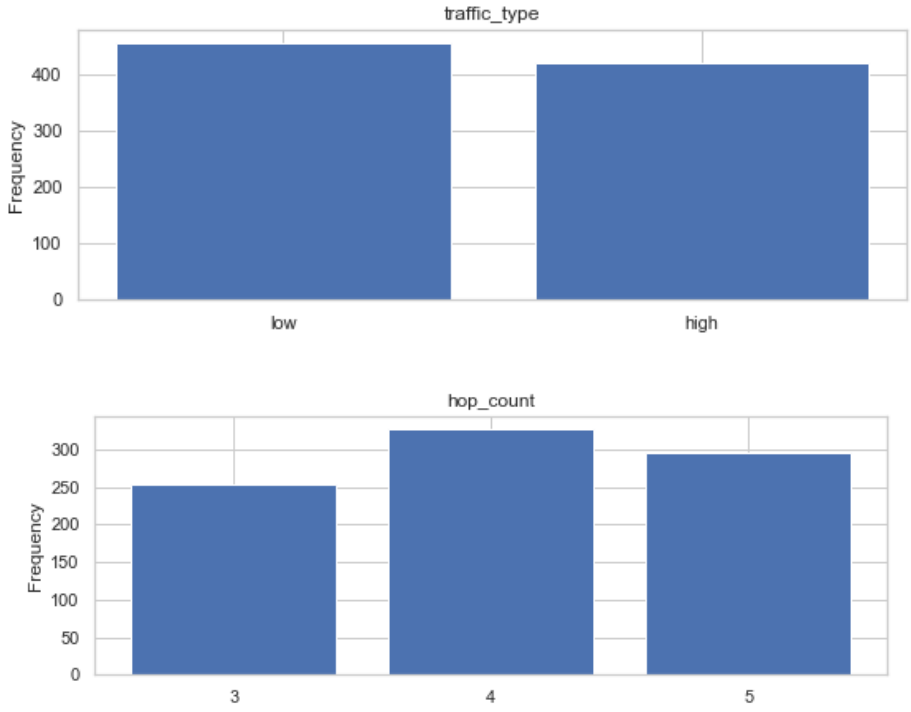
Tablo 6. Yapay zeka tabanlı yönlendirme algoritması sözde kodu

Algoritma:	**Yapay zeka tabanlı yönlendirme algoritması**
Girdi:	İstemci ve sunucu ipv4 adresleri
Çıktı:	Trafik düzeyi en düşük olan yolun akış tablolarına eklenmesi
1:	**function** YönlendirmeAlgoritması(istemci_ipv4, sunucu_ipv4):
2:	model ← YapayZekaModeliYükle()
3:	paths ← EnKısa5YoluGetir(istemci_ipv4, sunucu_ipv4)
4:	pathsTraffic ← Boş dizi tanımla
5:	**for all** paths **do:**
6:	AkışTablosunaEkle(path)
7:	path_statistics ← PathİstatistikleriniTopla()
8:	prediction ← model.prediction(path_statistics)
9:	pathsTraffic.append(prediction)
10:	deleteFlows()
11:	**end for**
12:	path ← TrafiğiEnAzOlanYoluBul(pathsTraffic)
13:	AkışTablosunaEkle(path)

5. ARAŞTIRMA SONUÇLARI

5.1 Veri Analizi Sonuçları

Kurgulanan 3 farklı senaryo ile toplamda 876 adet veri üretilmiştir. Üretilen her bir veri 51 adet değişkene sahip olmaktadır. Bunlardan 50 tanesi aktarım sırasında ve öncesindeki durumları temsil ederken 1 adet değişken ise ortamdaki trafik türünü ifade etmektedir. Verilerin trafik düzeyine ve senaryolardaki hop sayısına göre dağılımı şekil 14'te gösterilmektedir.

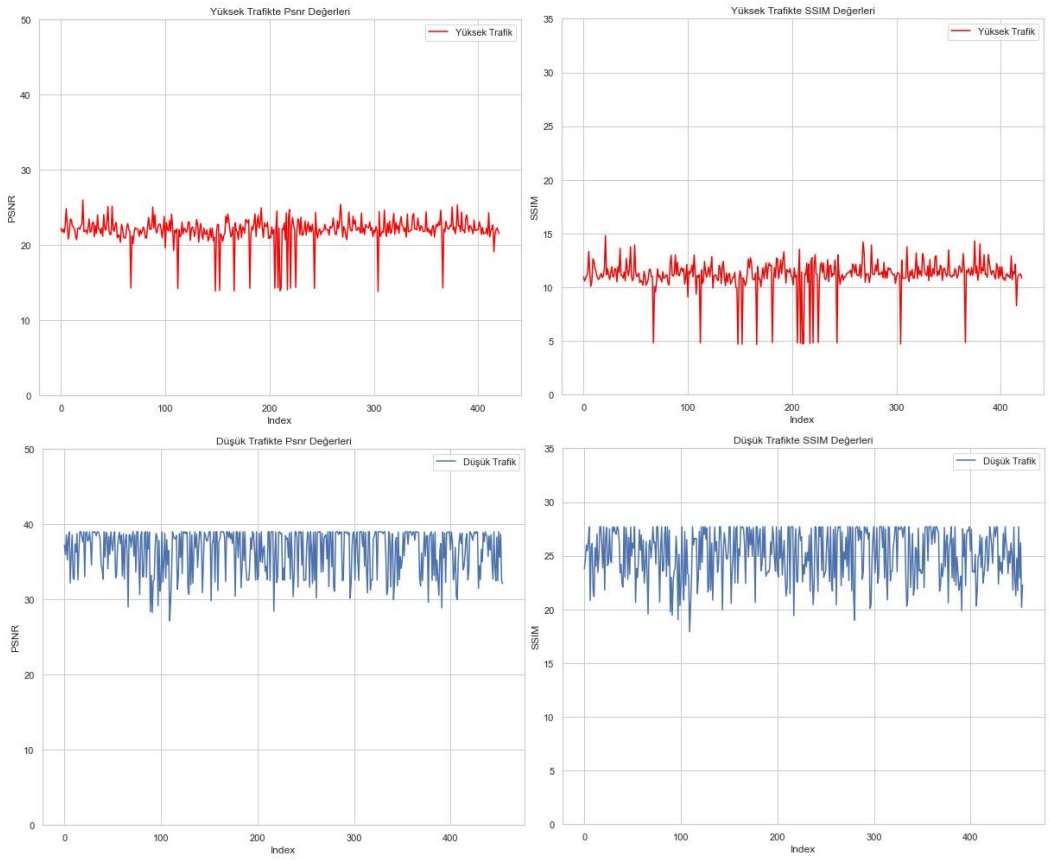


Şekil 14. Verilerin trafik düzeyine ve hop sayısına göre dağılımı

Trafik düzeyi yüksek ve düşük ortamda aktarılan video boyutu değişmekte ve veri kayıpları oluşmaktadır. Veri üretim esnasında kullanılan videonun orijinal boyutu 39478308 bittir. Düşük trafikli ortamda video aktarımı yapıldığında ortalama olarak aktarılan dosya boyutu 39320447.71 bit olmaktadır ve 157860.28 bit kayıp oluşmaktadır. Yaklaşık olarak %0,4 veri kaybı olduğu gözlemlenmektedir. Yüksek trafikli ortamda video aktarımı yapıldığında ortalama olarak aktarılan dosya boyutu 36388202.41 bit

olmaktadır ve 3090105.58 kayıp oluşmaktadır. Yaklaşık %7,82 veri kaybı olduğu gözlemlenmektedir.

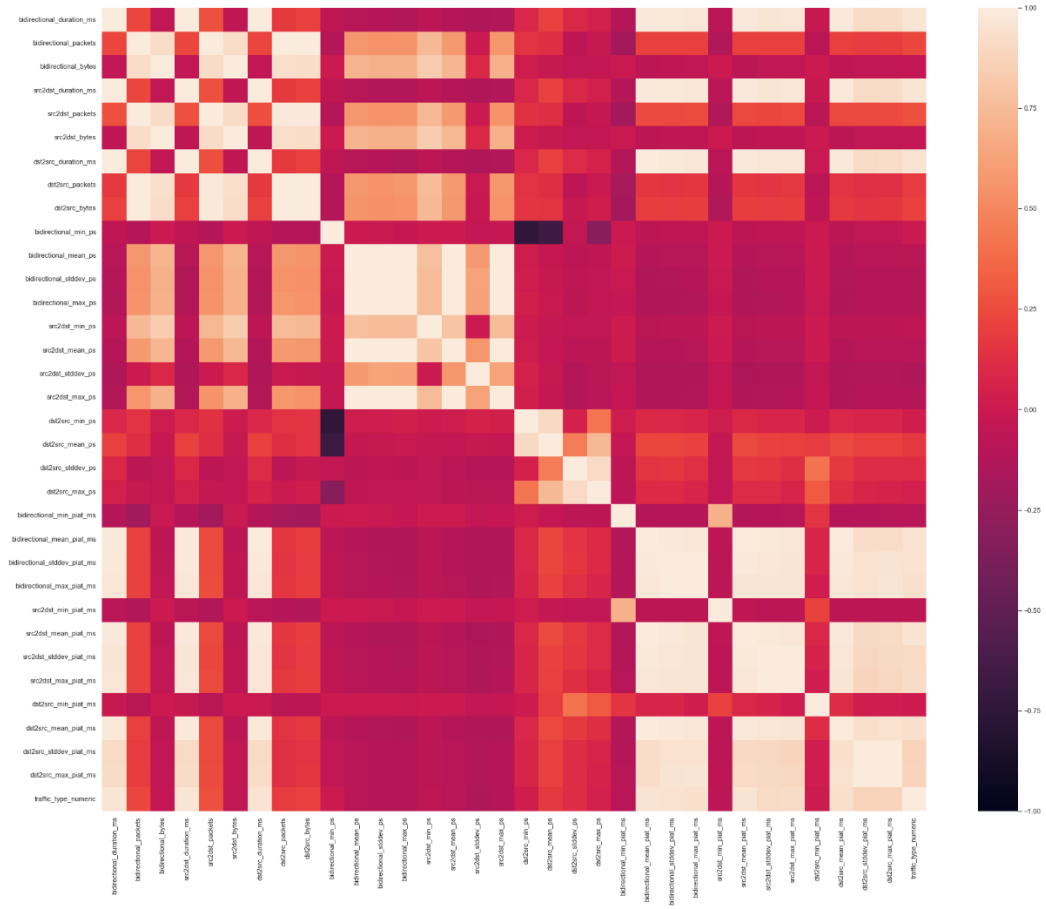
Senaryoda kurgulanan TCP trafiğinin aktarılan video kalitesine etkisini ölçmek için PSNR ve SSIM parametreleri kullanılmaktadır. Bu parametreler videonun orijinal videodan ne kadar farklı olduğu hakkında bize bilgi vermektedir. Yüksek değerler video kalitesinin orijinale yakın olduğunu belirtirken düşük değerler video üzerindeki çerçevelerde bozulma olduğunu göstermektedir. Üretilen verilerde düşük ve yüksek trafikli ortamdaki aktarılan videoların ortalama PSNR ve SSIM değerleri şekilde 15'te gösterilmektedir.



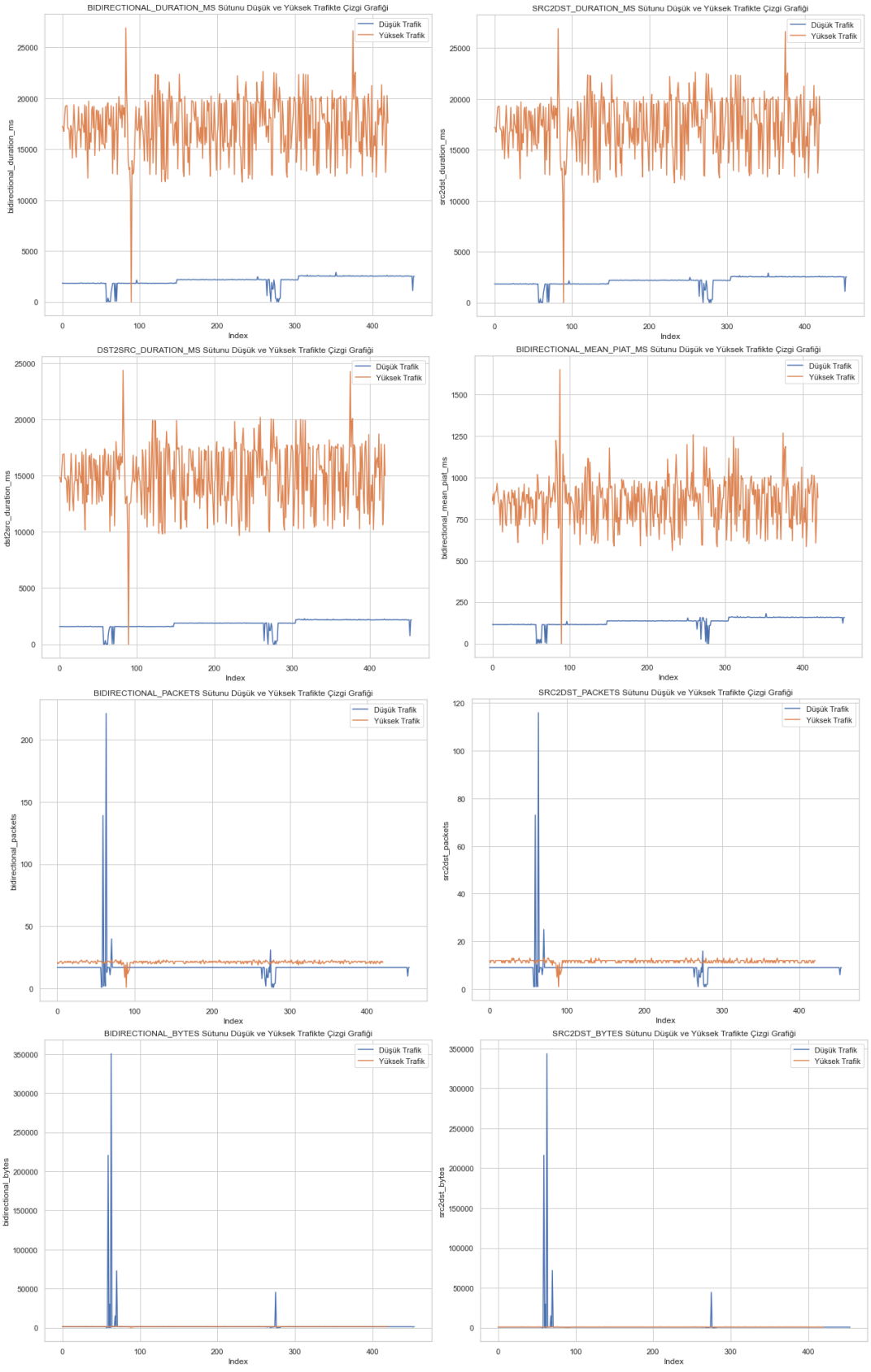
Şekil 15. Trafik düzeyine göre PSNR ve SSIM değerleri

Veri setinde iki cihaz arasındaki bağlantı kalitesinin ölçülmesi için NFStream, iperf3 ve ping araçları kullanılarak bilgi toplanmaktadır. Toplanan veriler incelendiğinde tüm değişkenlerin sonuç değere doğrudan etki etmediği gözlemlenmekte ve etkisi olmayan değişkenler veri setinden çıkarılmaktadır.

NFStream aracı ile toplamda 33 adet değişken elde edilmiştir. Bu değişkenlerin trafik düzeyi korelasyon matriksi oluşturularak incelenmiş ve 12 adet değişkenin sonuç değişkene etki ettiği gözlemlenmiştir. Etkisi olmayan değişkenler genellikle paket boyutları ile ilgili olmaktadır. Değişkenlerin birbirleri arasındaki ilişkiyi gösteren ısı haritası ve örnek çizgi grafikleri şekil 16 ve 17'de paylaşılmaktadır.

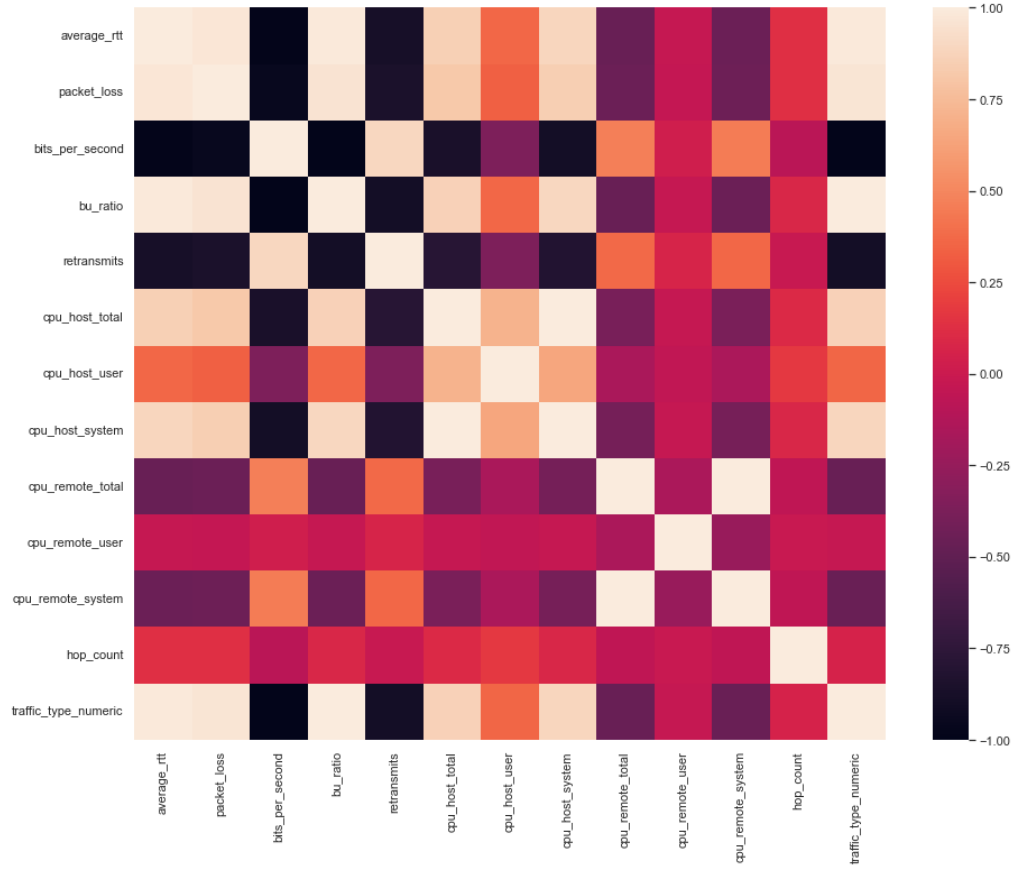


Şekil 16. NFStream verileri korelasyonunu gösteren ısı haritası

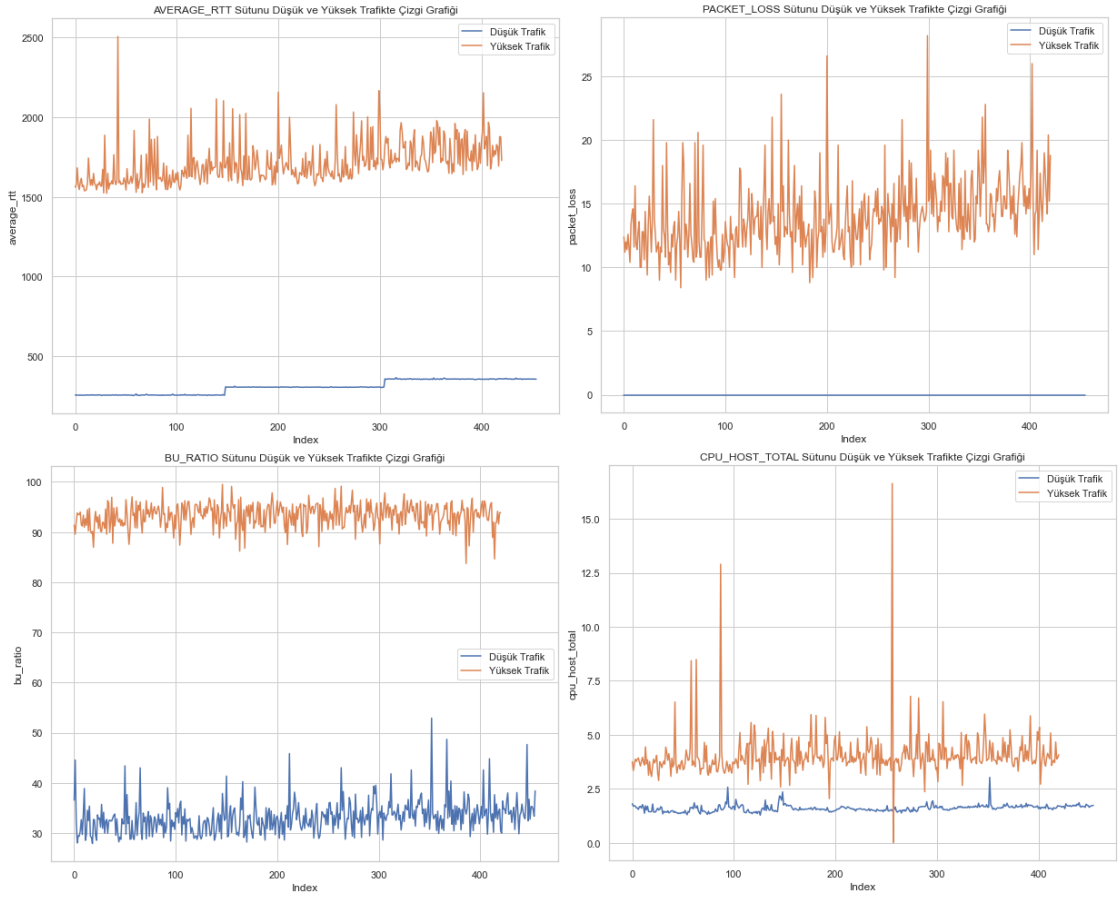


Şekil 17. NFStream değişkenleri trafik düzeyine göre örnek çizgi grafikleri

Diğer araçlar ile toplanan 12 adet değişken incelendiğinde 7 adet değişkenin trafik düzeyine etki ettiği gözlemlenmektedir. Bu değişkenlere ait ısı haritası ve çizgi grafikleri şekil 18 ve 19'da paylaşılmaktadır.

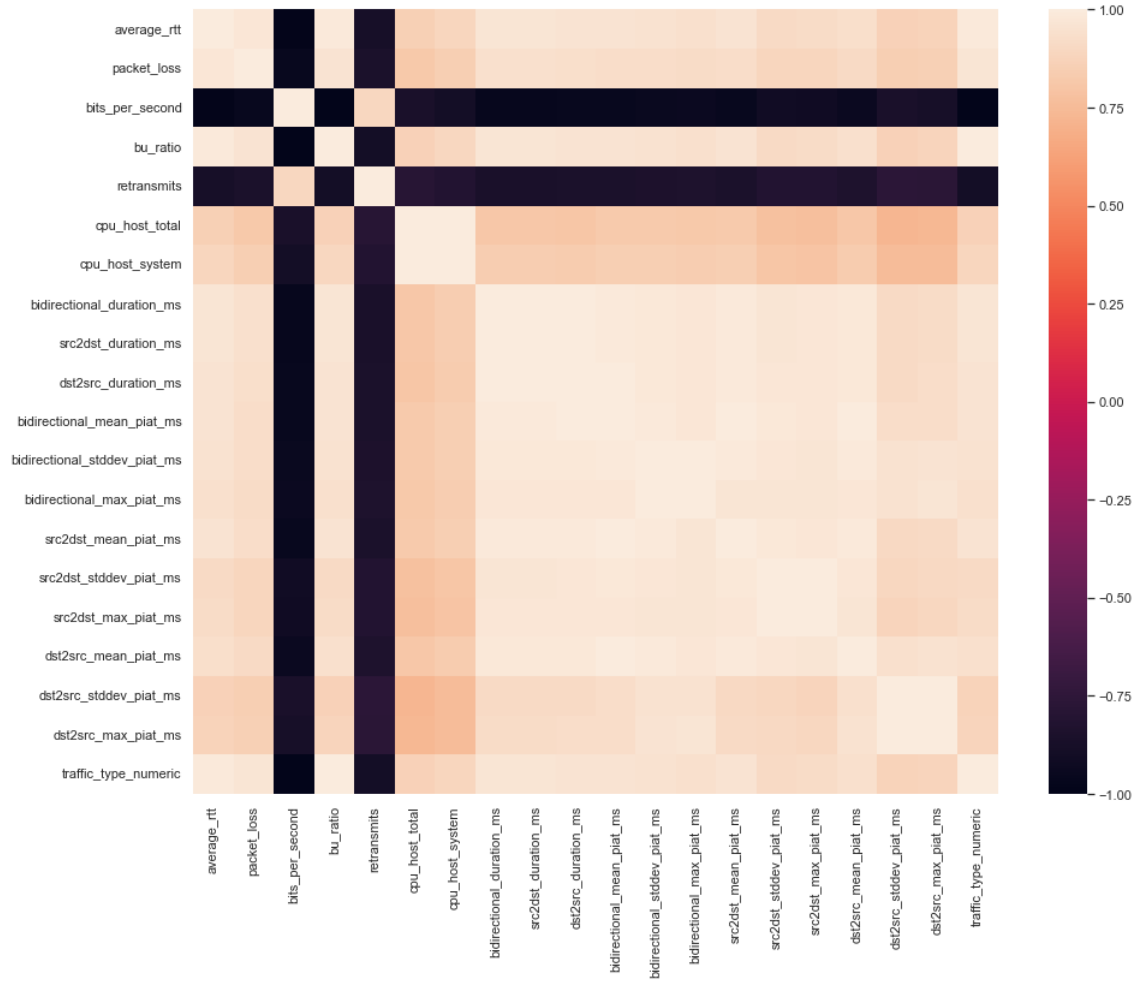


Şekil 18. Iperf3 ve ping değişkenleri korelasyonunu gösteren ısı haritası



Şekil 19. Diğer araçlar ile elde edilen değişkenlerin çizgi grafikleri

Sonuç olarak başlangıçta 50 adet olan değişken sayısı veri analizi sonucu özellik çıkarımı yapılarak 19 adete düşürülmüştür. Sonuç veri setinin ısı haritası ve değişkenleri şekil 20'de paylaşılmaktadır. Şekil incelendiğinde tüm değişkenler arasında yüksek miktarda korelasyon olduğu gözlemlenmektedir. Bu değişkenler trafik düzeyine doğrudan etki etmektedirler.



Şekil 20. Özellik çıkarımı sonrası veri seti ısı haritası

5.2 İkili Sınıflandırma Eğitim Sonuçları

Analiz işlemi sonrasında sonuç veri seti farklı ikili sınıflandırıcı modeller ile eğitilmiştir. Eğitim esnasında veriler %85 eğitim ve %15 test için ayrılmıştır. Eğitim sonrasında modellerin başarımları veri setinin basit olması ve değişkenler arasında net bir ayırımın olması nedeniyle çok yüksek çıkmaktadır. 4 farklı model ile eğitim yapılmıştır. Bu modeller KNN, XGBoost, SVM ve Random Forest'dır.

Modellerin logaritmik kayıp metriği incelendiğinde en düşük değere sahip model KNN modelidir. Modellerin değerlendirme metriğine göre sonuç değerleri tablo x'de gösterilmektedir.

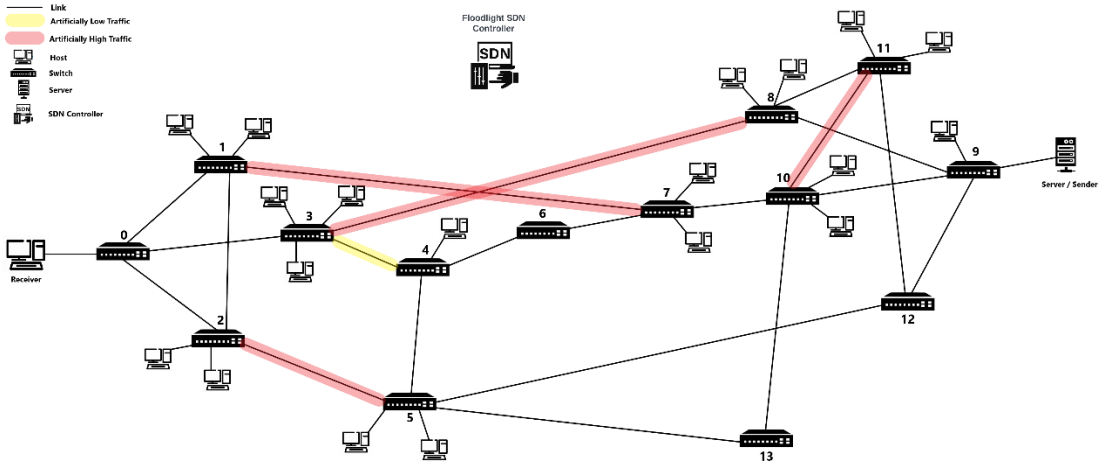
Tablo 7. Eğitim sonrası modellerin logaritmik kayıp değerleri

MODEL	LOGARİTMİK KAYIP
KNN	9.99 x e-16
RANDOM FOREST	0.0015
SVM	0.0018
XGBOOST	0.0023

Modellerin test verisi üzerinde precision, recall ve accuracy değerleri incelendiğinde değerlerin 1 olduğu gözlemlenmektedir. Yapay bir veri seti ile sanal bir ortamda veri üretildiği için veri seti içerisinde gürültü taşımamaktadır. Bu nedenle hatasız bir şekilde sınıflandırma yapabilmektedir. Veri toplama aşamasında verilere gürültü eklenerek yapay veri setinin gerçekçi bir grafik çizmesi sağlanabilir. Modeller eşit seviyede başarım göstermektedir.

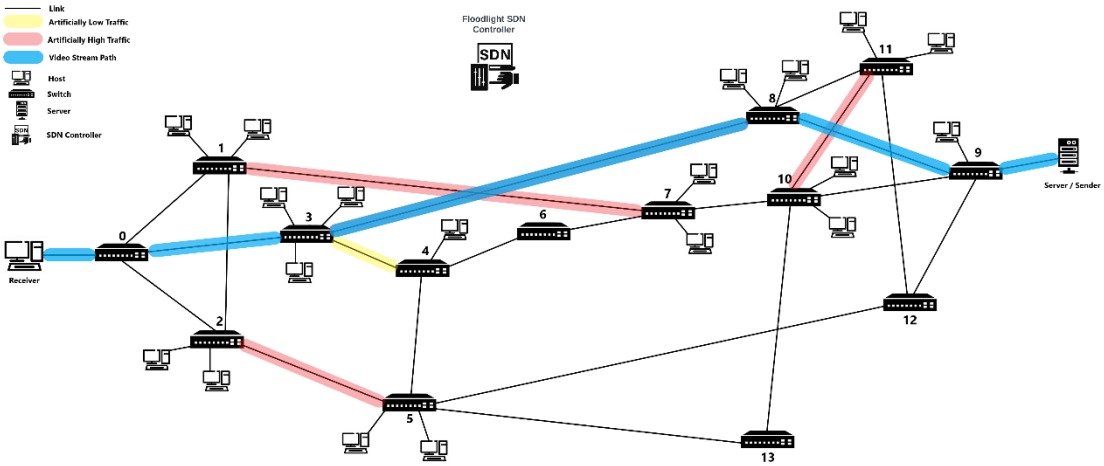
5.3 Yönlendirme Algoritması Başarımı

Bizim tarafımızdan önerilen yapay zeka tabanlı yönlendirme algoritması NSFNET topolojisi üzerinde kurgulanan senaryoda test edilmektedir. Senaryoda trafik oluşturan 19 adet istemci, 1 adet ana istemci (videoyu alan) ve 1 adet sunucu (videoyu gönderen) bulunmaktadır. Bunlara ek olarak ağın beyni olarak nitelendirdiğimiz Floodlight denetleyici metrik olarak hopcount değerine ayarlanmıştır. Şekil 21'de hostlar arasındaki trafik akışları ve dereceleri gösterilmekte ve senaryonun genel bir görselleştirilmesi sunulmaktadır.



Şekil 21. Önerilen algoritmanın başarımını test etmek için kurgulanan senaryonun görselleştirilmesi

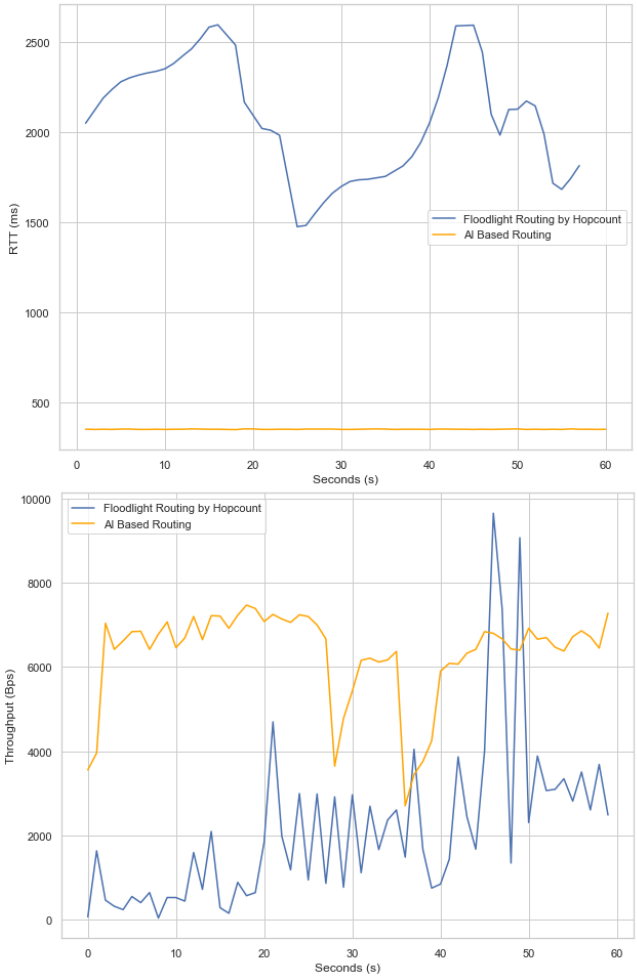
Test aşamasında ilk olarak ağa herhangi bir müdahale etmeden istemci ve sunucu arasında veri transferi yapılmaktadır. Floodlight denetleyici hopcount metriği ile en kısa mesafe üzerinden veri akışını gerçekleştirmekte ve 0-3-8-9 anahtarlarını takip etmektedir. Bu yol üzerinde ağır trafik olması veri akışını olumsuz etkilemekte ve veri kayıpları oluşmaktadır. Daha önce yapılan testlerde denetleyicinin gitmiş olduğu yol üzerinde video gönderildiğinde kalite değerleri düşük çıkmaktadır. Şekil 22'de Floodlight denetleyicisinin veri transferi sırasında izlediği yol aktarılmaktadır.



Şekil 22. Floodlight denetleyicinin hopcount metriği ile veri transferi sırasında takip ettiği yolun görselleştirilmesi

Test aşamasının ikinci kısmında önerilen yönlendirme algoritması test edilmektedir. Test için ilk aşama ile birebir aynı ortam değişkenleri kullanılmaktadır. İstemci ile sunucu arasındaki transfer öncesinde önerilen algoritma çalışmakta ve trafik düzeyi düşük olan yolu bulmaktadır. Aktarım sırasında 0-3-4-5-12-9 anahtarlarını takip etmektedir. Bu yol üzerinde trafiğin daha düşük olması daha başarılı bir veri aktarımı yapılmasına olanak tanımaktadır.

Her iki algoritmanın kurgulanan senaryolarını gerçekleştirdikten sonra test esnasında toplanan veriler karşılaştırılmaktadır. Karşılaştırma için ağ 60 saniye boyunca dinlenmekte ve yol üzerinde RTT ve throughput değerleri kaydedilmektedir. Şekil 23'te kaydedilen verilerin karşılaştırılması çizgi grafiklerinde gösterilmektedir.



Şekil 23. Floodlight hopcount metrikli yönlendirme algoritması ile önerilen algoritmanın RTT ve Throughput değerlerinin karşılaştırılması

İlk test aşamasında hopcount metrikli varsayılan yönlendirme algoritması sonucu ortalama 2063.14 ms RTT ve 2135.90 bps throughput değerleri elde edilmektedir. İkinci aşamada önerilen algoritması ile bu oranlarda iyileşme görülmekte 354.05 ms RTT ve 6313.50 bps throughput değeri elde edilmektedir. Karşılaştırma sonucu olarak önerdiğimiz algoritma başarılı bir şekilde trafik düzeyi az olan yolu tespit etmekte ve veri aktarımını bu yol üzerinden yapmaktadır.

6. TARTIŞMA

Yapılan çalışma, yazılım tanımlı ağların geleneksel ağlara göre avantajlarına odaklanmıştır. Özellikle, veri düzleminde yük dengeleyicilerinin ve yönlendirme algoritmalarının ayrıntılı incelenmesi, bu alandaki gelişmelerin önemini vurgulamaktadır. Yazılım tanımlı ağların, geleneksel yaklaşımlardan daha başarılı olduğu ve performansı artırdığı görülmektedir.

Çalışmada NSFnet topolojisi üzerinde seçilen istemci ve sunucu arasında video akışı sağlanmaktadır. Bu akış esnasında ağ üzerinde kurgulanan trafikler sonucunda bazı veri yolları trafik nedeniyle video kalitesini olumsuz etkilemektedir. Kurgulanan senaryoların gerçekleştirilmesi ile önerilen yapay zeka tabanlı yönlendirme algoritması hopcount metrikli denetleyici yönlendirme algoritmasından daha başarılı olmaktadır.

Çalışma Mininet sanal simülasyon ortamında gerçekleştirilmekte ve gerçek bir ağ üzerinde test edilmemiştir. Sanal simülasyon ortamının maliyet, zaman ve test ortamı için kolaylık sağlaması gibi birçok artısı bulunurken bazı dezavantajlı yönleri bulunmaktadır. Yapay zeka tabanlı yönlendirme algoritması için oluşturulan veri seti tamamen yapay ve bizim oluşturmuş olduğumuz senaryolar üzerinden üretildiği için daha basit ve gerçek zamanlı trafikten farklı olmaktadır. Bu çalışmanın daha verimli olabilmesi için gerçek mesafeler ve gerçek bir ağ ortamında test edilmesi daha doğru sonuçlar alınmasına olanak sağlayacaktır.

Önerilen algoritmanın tasarlanma aşamasında ağ üzerinde trafiği tespit etmek için birçok farklı parametre kullanılmaktadır. Bu parametreler ağ üzerindeki trafik durumunu daha net ve birçok farklı boyuttan tespit etmemize yardımcı olmaktadır. Burada temel amaç trafik düzeyini farklı boyutlardan izleyerek olağan ve olağan dışı durumları daha doğru tespit etmektir. Bu aşamada yapay zekanın ve mevcut bilgisayarların hesaplama kapasitesi önemli rol oynamaktadır.

Sonuç olarak, elde edilen bulgular, yazılım tanımlı ağlarda yönlendirme ve yapay zeka optimizasyonu konularında ileriye dönük çalışmalara rehberlik edebilir. Yazılım tanımlı ağların yenilikçi yaklaşımı ile programcılar ağ üzerinde söz sahibi olabilmekte ve kolaylıkla işlem ve geliştirme yapabilmektedir. Bu alandaki teknolojik gelişmeler, ağ performansını daha da artırabilir.

KAYNAKLAR

[1] Cisco, U. (2020). “Cisco annual internet report (20182023)” https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (1/12/2023)

[2] B. A. A. Nunes, M. Mendonca, X. -N. Nguyen, K. Obraczka and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," in IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1617-1634, Third Quarter 2014, doi: 10.1109/SURV.2014.012214.00180.

[3] F. Hu, Q. Hao and K. Bao, "A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation," in IEEE Communications Surveys & Tutorials, vol. 16, no. 4, pp. 2181-2206, Fourthquarter 2014, doi: 10.1109/COMST.2014.2326417.

[4] Hamdan, M., Hassan, E., Abdelaziz, A., Elhigazi, A., Mohammed, B., Khan, S., ... & Marsono, M. N. (2021). A comprehensive survey of load balancing techniques in software-defined network. Journal of Network and Computer Applications, 174, 102856.

[5] Belgaum, M. R., Ali, F., Alansari, Z., Musa, S., Alam, M. M., & Mazliham, M. S. (2022). Artificial intelligence based reliable load balancing framework in software-defined networks. CMC—Comput. Mater. Contin, 70, 251-266.

[6] Warsama, A. (2020). Traffic Engineering with SDN: Optimising traffic Load-Balancing with OpenFlow.

[7] Babayigit, B., & Ulu, B. (2021). Deep learning for load balancing of SDN-based data center networks. International Journal of Communication Systems, 34(7), e4760.

[8] Alhilali, A. H., & Montazerolghaem, A. (2023). Artificial Intelligence based Load balancing in SDN: A Comprehensive Survey. Internet of Things, 100814.

[9] Hamed, M. I., ElHalawany, B. M., Fouda, M. M., & Eldien, A. S. T. (2017, December). A new approach for server-based load balancing using software-defined networking. In 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS) (pp. 30-35). IEEE.

[10] Belgaum, M. R., Ali, F., Alansari, Z., Musa, S., Alam, M. M., & Mazliham, M. S. (2022). Artificial intelligence based reliable load balancing framework in software-defined networks. CMC—Comput. Mater. Contin, 70, 251-266.

[11] Çetin KÜK, Ekber. "YAZILIM TANIMLI İÇERİK DAĞITIM AĞLARINDA eMBB TRAFİĞİ İÇİN ERİŞİM PROTOKOLÜ TABANLI KONTROLÖR TASARIMI." 2021.

[12] TEMURÇİN, A. N., & ERSOY, M. (2023). AĞ ALTYAPILARINDA YAPAY ZEKA TABANLI AĞ TRAFİK YÖNETİM MEKANİZMALARININ İNCELENMESİ. Uluslararası Sürdürülebilir Mühendislik ve Teknoloji Dergisi, 7(1), 31-40.

[13] Bakonyi, P., Boros, T., & Kotuliak, I. (2020, July). Classification Based Load Balancing in Content Delivery Networks. In 2020 43rd International Conference on Telecommunications and Signal Processing (TSP) (pp. 621-626). IEEE.

[14] Semong, T., Maupong, T., Anokye, S., Kehulakae, K., Dimakatso, S., Boipelo, G., & Sarefo, S. (2020). Intelligent load balancing techniques in software defined networks: A survey. Electronics, 9(7), 1091.

[15] Chen-Xiao, C., & Ya-Bin, X. (2016). Research on load balance method in SDN. International Journal of Grid and Distributed Computing, 9(1), 25-36.

[16] DEGİRMENCİ, S., & YILTAS, D. (2020). YAPAY ZEKÂ TEKNİKLERİYLE YAZILIM TANIMLI AĞ UYGULAMASI. Mühendislik Bilimleri ve Tasarım Dergisi, 8(4), 999-1009.

[17] Gebremariam, A. A., Usman, M., & Qaraqe, M. (2019, March). Applications of artificial intelligence and machine learning in the area of SDN and NFV: A survey. In 2019 16th International Multi-Conference on Systems, Signals & Devices (SSD) (pp. 545-549). IEEE.

[18] Cicioğlu, M., & Çalhan, A. (2017). Yazılım tanımlı ağlar–YTA. Karaelmas fen ve mühendislik dergisi, 7(2), 684-695.

[19] Latah, M., & Toker, L. (2016). Application of artificial intelligence to software defined networking: A survey. Indian Journal of Science and Technology, 9(44), 1-7.

[20] https://mininet.org/walkthrough/ (1/12/2023)

[21]https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview (1/12/2023)

[22] https://en.wikipedia.org/wiki/National_Science_Foundation_Network (10/01/2024)

[23] https://www.nfstream.org/docs/design (10/01/2024)

TEŞEKKÜR

Bu çalışmamızda planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığımız, yönlendirme ve bilgilendirmeleriyle çalışmamızı bilimsel temeller ışığında şekillendiren sayın hocamız Doç. Dr. Murtaza CİCİOĞLU'na sonsuz teşekkürlerimizi sunarız.

ANIL DURSUN İPEK

BATUHAN ARSLANDAŞ

BURSA

2023

ÖZGEÇMİŞ

Anıl Dursun Ipek

+90 531 517 71 25 @ anildursunipek@gmail.com

Education

BURSA ULUDAG UNIVERSITY
BACHELOR OF COMPUTER ENG.
Sep 2021-Present Bursa, Turkey
- GPA: 3.52/4.00
- Fourth-year student

BURSA ULUDAG UNIVERSITY
BACHELOR OF ARCHITECTURE
Sep 2018-Jun 2021 Bursa, Turkey
- GPA: 3.33/4.00
- I changed my major in 2021

Links

GitHub **anildursunipek**
Linkedin **anıl-dursun-ipek**

Certificates

GLOBAL AI HUB;
-Introduction To Deep Learning
-Python For Machine Learning
-Introduction To Database
-Data Analysis
-Introduction To Machine Learning

BTK ACADEMY;
-Programming With JAVA
-Version Controls: Git and GitHub
-Data Science Workshop-Bootcamp
-Python And TF for Data Science

OTHERS;
-Udemy Web Development
-Udeym C Programming
-Turkcell Sql Server 101-201

SOCIAL ACTIVIES;
-Personal Mindfulness Seminars

Skills

PROGRAMMING
Python • Java • C# • HTML • CSS • JavaScript • Angular • Entity Framework • SQL • NoSQL• Ubuntu • Git

Languages

TURKISH
• Native Language

ENGLISH
• Intermediate

Projects

AI BASED ROUTING AND TRAFFIC CLASSIFICATION IN SDN
GRADUATION THESIS I
- Mininet and Floodlight SDN controller were employed to create a simulation environment in software-defined networks. The environment utilized video streaming to generate an artificial dataset, and flow steering was accomplished using the XGBoost machine learning algorithm.

TICKET RESERVATION SYSTEM
- A C# form application developed by a team of 5 individuals, utilizing Entity Framework and MySQL database, to enable users to easily purchase ferry, airplane, and bus tickets for their travel plans.

TOURISM AGENCY WEB SITE
- A web-based project using Angular and ASP.NET WEB API, aimed at enhancing user-friendly creation of holiday and travel plans. It involves manual creation and use of data structures within the system.

AUTOMATIC PLATE NUMBER RECOGNIZER
- I have developed a program using YOLOv5, image processing methods, and various OCR models to enable license plate detection at entrances of locations such as shopping malls and parking lots.

URBAN SOUND CLASSIFICATION USING CNN
KOÇ HOLDING, GLOBAL AI HUB DEEP LEARNING BOOTCAMP
- Ten different city sounds were preprocessed and classified using deep learning techniques by converting them into spectrograms during the preprocessing stage.

Competition

TEKNOFEST TURKISH NLP COMPETITION
HERMES AI TEAM
Feb 2022 – May 2022
- We designed an ensemble model to identify insult and hate speech in Turkish, utilizing both BERT and Electra-based models. Our ensemble model achieved the 7th position at the conclusion of the competition

Honors & Activities

SUMMER WORK AND TRAVEL 2023
Jun 2023 – Oct 2023 New York, U.S.A
I participated in an international cultural exchange program where I enhanced my English language skills and spent time with people from diverse cultures.

HIGH HONOR STUDENT
BURSA ULUDAG UNIVERSITY
I have earned the privilege to receive this document through exceptional performance in the courses within my major.

ZORLU FOUNDATION SCHOLARSHIP
ZORLU FOUNDATION
I earned the scholarship by achieving a ranking in the top 1% in the nationwide university entrance exam in Turkey.

Batuhan Arslandaş

Full-Stack Developer

Profile

I graduated from Bolu Abant İzzet Baysal University, Computer Programming Department. I transferred to Bursa Uludağ University Computer Engineering Department with VTE, and I continue to study in the 3rd grade. Since 2020, I have been working as a Full-Stack Web Developer at Hiraparl Software Company. I am extremely confident in solution-oriented communication, agility, strategy development, and teamwork in business life.

Work experience

present – 2021

Hiraparl Software And Informatics

Full-Stack Developer, Bolu

- I have been working as a Full-Stack Web Developer for approximately 2.5 years at this company, where I did an internship for 6 months at the beginning.

Education

2025 – 2021

Bursa Uludag Universty

Bachelor's Degree, Compoter Engineer (GNO: 3,04/4), Bursa

2021 – 2019

Bolu Abant İzzet Baysal Universty

Associate's Degree, Compoter Programming (GNO: 3,24/4), Bolu

Project Experiences

2022-2024

Proteyon Building Management System (Hiraparl)

I worked on the frontend and backend parts of a mobile and web project that lasted approximately 26 months. (Angular, Java, and MongoDB)

2023

Ticket Reservation Automation System

I worked on the Frontend and Backend parts and designed this project, which lasted 1 month, as a desktop application. (C, SQL, Visual Studio)

2023

Tourism Agency Web Site using Data Structures

It's a 2-month web project that I did by working on the Frontend and Backend parts using data structures. (.NET, Angular, and SQL)

2021

Accounting System (Hiraparl)

A web project was given to me during my 6-month internship in 2020. I worked on the frontend and backend parts. (Angular, Java, MySQL)

Contact

Email
batuhanarslandass@hotmail.com

Phone
+90 539 951 5278

Github
@batuhanarslandas

Languages

- Turkish
- English

Skills

- Java
- Angular
- Sping Boot
- Python
- C
- C++
- TypesScript
- JavaScript
- HTML
- CSS
- SCSS
- MySQL
- MongoDB