KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

GELECEK NESİL KABLOSUZ AĞLAR İÇİN SANALLAŞTIRMA VE YAZILIM TANIMLI AĞ İLE DİNAMİK SERVİS KALİTESİ YÖNETİMİ

ÖZLEM ÇUBUKÇU

KOCAELİ 2024

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

GELECEK NESİL KABLOSUZ AĞLAR İÇİN SANALLAŞTIRMA VE YAZILIM TANIMLI AĞ İLE DİNAMİK SERVİS KALİTESİ YÖNETİMİ

ÖZLEM ÇUBUKÇU

Prof. Dr. Adnan KAVAK **Danışman, Kocaeli Üniversitesi**	.....
Prof. Dr. Kerem KÜÇÜK **Jüri Üyesi, Kocaeli Üniversitesi**	.....
Dr. Öğr. Üyesi Nur Banu ALBAYRAK **Jüri Üyesi, Kocaeli Sağlık ve Teknoloji Üniversitesi**	.....

Tezin Savunulduğu Tarih: 24.06.2024

ETİK BEYAN VE ARAŞTIRMA FONU DESTEĞİ

Kocaeli Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- Bu tezin bana ait, özgün bir çalışma olduğunu,
- Çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı,
- Bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi,
- Bu çalışmanın Kocaeli Üniversitesi'nin abone olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere uygun olduğunu,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı,

beyan ederim.

■ Bu tez çalışmasının herhangi bir aşaması hiçbir kurum/kuruluş tarafından maddi/alt yapı desteği ile desteklenmemiştir.

☐ Bu tez çalışması kapsamında üretilen veri ve bilgiler ..... tarafından ..... no'lu proje kapsamında maddi/alt yapı desteği alınarak gerçekleştirilmiştir.

Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

Özlem ÇUBUKÇU

YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI

Fen Bilimleri Enstitüsü tarafından onaylanan lisansüstü tezimin tamamını veya herhangi bir kısmını, basılı ve elektronik formatta arşivleme ve aşağıda belirtilen koşullarla kullanıma açma izninin Kocaeli Üniversitesi'ne verdiğimi beyan ederim. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet hakları bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki makale, kitap, tebliğ, lisans, patent gibi çalışmalarda kullanımı, danışmanımın isim hakkı saklı kalmak koşuluyla ve her iki tarafın bilgisi dâhilinde bana ait olacaktır.

Tezin kendi özgün çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim kurulu tarafından yayınlanan ***"Lisanüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"*** kapsamında tezim aşağıda belirtilen koşullar haricinde YÖK Ulusal Tez Merkezi/ Kocaeli Üniversitesi Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

☐ Enstitü yönetim kurulu kararı ile tezimin erişime açılması mezuniyet tarihinden itibaren 2 yıl ertelenmiştir.

☐ Enstitü yönetim kurulu gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihinden itibaren 6 ay ertelenmiştir.

■ Tezim ile ilgili gizlilik kararı verilmemiştir.

Özlem ÇUBUKÇU

ÖNSÖZ VE TEŞEKKÜR

Bu tez, 5G ağlarında servis kalitesinde süreklilik sağlanması amacıyla bir SDN (Yazılım Tanımlı Ağ) uygulamasının geliştirilmesi ve analiz edilmesi üzerine yapılan kapsamlı bir araştırmanın ürünüdür. Bu çalışma, 5G ağlarının hızla gelişen ekosisteminde, servis kalitesi yönetimi ve ağ yönetimi alanlarında katkı yapmayı hedeflemektedir.

Tez çalışmamda bana desteğini esirgemeyen, çalışmalarıma yön veren değerli hocam Prof. Dr. Adnan KAVAK'a sonsuz teşekkürlerimi sunarım.

Ayrıca, akademik çalışmalarım boyunca bilgi ve tecrübelerinden yararlandığım Prof. Dr. Kerem KÜÇÜK'e, yüksek lisans çalışmam boyunca bana inanan ve destekleyen başta sevgili eşim Aykut ÇUBUKÇU olmak üzere, hayatım boyunca desteklerini esirgemeyen sevgili aileme ve bana her zaman başarma inancı aşılayan anneme içten teşekkürlerimi sunarım.

Haziran – 2024 Özlem ÇUBUKÇU

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar

3GPP : 3rd Generation Partnership Project (3. Nesil Ortaklık Projesi)
4G : Forth-Generation (Dördüncü Nesil)
5G : Fifth-Generation (Beşinci Nesil)
5GC : 5G Core Network (Beşinci Nesil Çekirdek Şebeke)
5QI : 5G QoS Identifier (Beşinci Nesil Servis Kalitesi Tanımlayıcısı)
6G : Sixth-Generation (Altıncı Nesil)
AF : Application Function (Uygulama Fonksiyonu)
AMF : Access and Mobility Management Function (Erişim ve Hareketlilik Yönetimi Fonksiyonu
API : Application Programming Interface (Uygulama Programlama Arayüzü)
ARP : Allocation and Retention Priority (Tahsis ve Tutma Önceliği)
AUSF : Authentication Server Function (Kimlik Doğrulama Sunucusu Fonksiyonu)
DN : Data Network (Veri Ağı)
DNN : Data Network Name (Veri Ağı Adı)
eMBB : Enhanced Mobile Broad Band (Geliştirilmiş Mobil Geniş Bant)
ETSI : European Telecommunications Standards Institute (Avrupa Telekomünikasyon Standartları Enstitüsü)
FQDN : Full Qualified Domain Name (Tam Nitelikli Alan Adı)
GBR : Guaranteed Bit Rate (Garanti Edilmiş Bit Hızı)
GTP-U : GPRS Tunneling Protocol - User Plane (GPRS Tünelleme Protokolü)
GUTI : Globally Unique Temporary Identifier (Geçici Kullanıcı Tanımlayıcı)
HTTP : Hypertext Transfer Protocol (Hiper Metin Transfer Protokolü)
HTTP 2 : Hypertext Transfer Protocol Sürüm 2 (Hiper Metin Transfer Protokolü Sürüm 2)
IMSI : International Mobile Subscriber Identity (Uluslararası Mobil Abone Kimliği)
IoT : Internet-of-Things (Nesnelerin İnterneti)
IP : Internet Protocols (İnternet Protokolleri)
K : Encryption Key (Şifreleme Anahtarı)
MANO : Management and Orchestration (Yönetim ve Orkestrasyon)
MCC : Mobile Country Code (Mobil Ülke Kodu)
MNC : Mobile Network Code (Mobil Şebeke Kodu)
MEC : Mobile Edge Computing (Mobil Uç Bilişim)
MBR : Maximum Bit Rate (Maksimum Bit Hızı)
mMTC : Massive Machine Type Communications (Yoğun Makine Tipi İletişim)
NAS : Non-Access Stratum (Erişim Dışı Katman)
NF : Network Function (Ağ Fonksiyonu)
NFV : Network Function Virtualization (Ağ Fonksiyonu Sanallaştırma)
NFVI : Network Function Virtualization Infrastructure (Ağ Fonksiyonu Sanallaştırma Altyapısı)
NFVO : Network Function Virtualization Orchestrator (Ağ Fonksiyonu Sanallaştırma Orkestratörü)
NGAP : Next Generation Application Protocol (Gelecek Nesil Uygulama Protokolü)
NRF : Network Repository Function (Ağ Depolama Fonksiyonu)

NSSF : Network Slice Selection Function (Ağ Dilimi Seçme Fonksiyonu)
ONF : Open Network Foundation (Açık Ağ Oluşturma Vakfı)
ONOS : Open Network Operating System (Açık Ağ İşletim Sistemi)
OPc : Operator Code (Operator Kodu)
OVS : Open vSwitch (Açık Sanal Anahtar)
PCF : Policy Control Function (Politika Kontrol Fonksiyonu)
PDR : Packet Delay Budget (Paket Gecikme Bütçesi)
PDU : Protocol Data Unit (Protokol Veri Birimi)
PER : Packet Error Rate (Packet Hata Oranı)
PFCP : Packet Forwarding Control Protocol (Paket Yönlendirme Kontrol Protokolü)
PLMN : Public Land Mobile Network (Kamu Alanı Mobil Ağı)
QFI : QoS Flow Identifier (Servis Kalitesi Akış Tanımlayıcısı)
QoS : Quality of Service (Servis Kalitesi)
RAN : Radio Access Network (Radyo Erişim Ağı)
SCTP : Stream Control Transmission Protocol (Akış Kontrollü İletim Protokolü)
SDN : Software-Defined Network (Yazılım Tanımlı Ağ)
SIM : Subscriber Identity Module (Abone Kimlik Modülü)
SMF : Session Management Function (Oturum Yönetim Fonksiyonu)
SMSC : Short Message Service Center (Kısa Mesaj Servisi Merkezi)
SUCI : Subscription Concealed Identifier (Abone Gizli Kimliği)
TCP : Transmission Control Protocol (İletim Kontrol Protokolü)
TN : Transport Network (İletim Ağı)
UE : User Equipment (Kullanıcı Ekipmanı)
UDP : User Datagram Protocol (Kullanıcı Datagram Protokolü)
UDR : Unified Data Repository (Birleşik Veri Deposu)
UDM : Unified Data Management (Birleşik Veri Yönetimi)
uRLLC : Ultra-Reliable Low Latency Communication (Ultra Güvenilir ve Düşük Gecikmeli İletişim
UPF : User Plane Function (Kullanıcı Düzlemi Fonksiyonu)
VIM : Virtualized Infrastructure Manager (Sanal Altyapı Yöneticisi)
VLAN : Virtual Local Area Network (Sanal Yerel Alan Ağı)
VNF : Virtual Network Function (Sanal Ağ Fonksiyonu)
VNFM : Virtual Network Function Manager (Sanal Ağ Fonksiyonu Yöneticisi)
VR : Virtual Reality (Sanal Gerçeklik)

GELECEK NESİL KABLOSUZ AĞLAR İÇİN SANALLAŞTIRMA VE YAZILIM TANIMLI AĞ İLE DİNAMİK SERVİS KALİTESİ YÖNETİMİ

ÖZET

Beşinci Nesil (5G) ve gelecek nesil mobil haberleşme sistemlerinin yüksek veri hızı, düşük gecikme ve çok sayıda cihaz bağlantısı gibi gereksinimleri karşılaması beklenmektedir. Her bir gereksinim farklı bir kullanım senaryosu olarak karşımıza çıkmaktadır. Her kullanım senaryosu için belirlenen servis kalitesinin (Quality of Service, QoS) sürekliliğinin sağlanması önemlidir. 5G'de ağ kaynaklarının aynı altyapı üzerinde farklı ihtiyaçlara uygun şekilde tahsis edilmesi ağ dilimleme ile mümkün olmaktadır. Bu bağlamda esnek ve dinamik bir ağ yönetimine ihtiyaç duyulmaktadır. Yazılım Tanımlı Ağ (Software Defined Network, SDN) ve Ağ Fonksiyonu Sanallaştırma (Network Function Virtualization, NFV) teknolojilerinin birlikte kullanılmasıyla daha akıllı, ölçeklenebilir, az maliyetli bir altyapının oluşması sağlanmaktadır. Bu çalışmada, SDN ve NFV tabanlı ağlarda, kullanıcıların servis kalitesinin sürekliliğinin sağlanabilmesi için bir SDN uygulaması geliştirilmiştir. SDN çalışmalarında kullanılabilecek açık kaynak kodlu bir simülasyon ortamı oluşturulmuştur. Elde edilen simülasyon sonuçlarına göre SDN ve NFV teknolojilerinin birlikte kullanılmasıyla, değişen ağ koşullarında kullanıcıların servis kalitesinde sürekliliğin sağlanabileceği görülmüştür.

Anahtar Kelimeler: Ağ Dilimleme, NFV-MANO, OpenDaylight, SDN, Yazılım Tanımlı Ağ.

DYNAMIC QUALITY OF SERVICE MANAGEMENT WITH VIRTUALIZATION AND SOFTWARE-DEFINED NETWORKING FOR NEXT-GENERATION WIRELESS NETWORKS

ABSTRACT

5G and next-generation mobile communications systems are expected to meet requirements such as high data speeds, low latency and large number of device connections. Each requirement faces us as a different use case. It is important to ensure the consistency of the determined quality of service for each usage scenario. In 5G, it is possible to allocate network resources to different needs on the same infrastructure through network slicing. In this context, flexible and dynamic network management is needed. The combination of Software Defined Network (SDN) and Network Function Virtualization (NFV) technologies creates a more intelligent, flexible, scalable and cost-effective infrastructure. In this study, an SDN application is developed to demonstrate that the consistency of user service quality can be ensured in SDN and NFV-based networks. An open-source simulation environment has been created, which can be used in SDN studies. The simulation results showed that the combined use of SDN and NFV technologies could ensure QoS consistency for users in changing network conditions.

Keywords: Network Slicing, NFV-MANO, OpenDaylight, SDN, Software Defined Network.

1. GİRİŞ

Günümüzde, Beşinci Nesil (5G) mobil iletişim teknolojisi geniş bant hizmetlerinin yanı sıra Nesnelerin İnterneti (IoT), akıllı şehirler, sanal gerçeklik (VR) gibi pek çok yeni uygulama alanı için temel bir altyapı oluşturmaktadır. Bu yeni kullanım senaryoları, geleneksel ağ yönetim ve kontrol yöntemlerinin sınırlarını zorlamaktadır. 5G ağlarının karmaşıklığı ve çeşitliliği, dinamik ve esnek bir ağ yönetimi yaklaşımını gerektirmektedir.

Yazılım Tanımlı Ağlar (SDN) ve Ağ Fonksiyonu Sanallaştırma (NFV), bu ihtiyaca cevap verebilecek teknolojiler arasında öne çıkmaktadır. SDN, ağ yönetimini merkezi bir denetleyici üzerinden programlanabilir hale getirerek ağın esnekliğini artırırken, NFV ise ağ fonksiyonlarını yazılım tabanlı sanal makineler üzerinde çalıştırarak ağ hizmetlerinin dinamik dağıtımını sağlamaktadır.

Bu tezde, 5G ağlarında SDN ve NFV tabanlı dinamik servis kalitesi yönetimi konusu ele alınacaktır. Tezin amacı, 5G ağlarında SDN ve NFV teknolojilerinin kullanımıyla, farklı servis kalitesi gereksinimlerine uygun olarak ağ kaynaklarının dinamik olarak tahsis edilmesini sağlamaktır. Bu doğrultuda, 5G ağlarının temel bileşenleri ve mimarisi incelenecek, SDN ve NFV'nin bu ağlar üzerindeki rolü tartışılacaktır. Ayrıca, tezin kapsamında önerilen algoritmanın ve bu algoritmanın 5G ağlarında uygulanabilirliği üzerinde durulacaktır.

Bu çalışma, 5G ağlarının performansını ve kullanıcı deneyimini iyileştirmek amacıyla, SDN ve NFV teknolojilerinin etkin bir şekilde nasıl kullanılabileceğini araştırmayı hedeflemektedir. Sonuç olarak, 5G ağlarında SDN ve NFV tabanlı dinamik servis kalitesi yönetiminin, geleceğin ağ yönetimi için önemli bir adım olabileceği öngörülmektedir.

2. LİTERATÜR ÖZETİ

Mobil veri kullanımının gün geçtikçe artması ve gelişmiş uygulamaların kullanılmaya başlaması ile birlikte esnek olmayan mobil ağ mimarilerinin gelişen servis ihtiyaçlarını karşılayamadığı görülmüştür. 5G hücresel ağlarının, Dördüncü Nesil (4G) mobil iletişim teknolojisinde etkili bir şekilde ele alınamayan daha yüksek kapasite, daha yüksek veri hızı, uçtan uca daha düşük gecikme süresi, çok sayıda cihaz bağlantısı, azaltılmış maliyet ve sürekli servis kalitesi gibi zorlukların üstesinden gelmesi gerektiği belirtilmektedir. Bu zorlukların aşılmasını kolaylaştıracak teknolojilerin ve tasarım temellerinin incelenmiş olduğu bir çalışmada (Gupta ve Jha, 2015), kullanıcıların servis kalitesinin sürekliliğini sağlayabilmek için trafik yönetiminin akıllı bir mekanizma ile yapılması gerektiğini ifade etmişlerdir. 5G ağlarında farklı hizmetlerin desteklenmesi için ağ dilimleme konseptinin tanıtıldığı bir çalışmada (Jose Ordonez-Lucena ve diğ., 2017), ağ dilimini, mevcut mimari yaklaşım olan "herkese uyan tek çözüm"ün aksine, belirli bir uygulamanın talep ettiği çeşitli gereksinimleri karşılamak üzere tasarlanmış, izole edilmiş, uçtan uca bir ağ olarak tanımlamışlardır. SDN mimarisinin ağ dilimlemeyi mümkün kılan teknoloji olsa da, ağ dilimlerinin ve onu oluşturan kaynakların yaşam döngüsünü verimli bir şekilde yönetmek için hayati önem taşıyan yeteneklerden yoksun olduğunu belirtmişlerdir. Bu bakımdan altyapı kaynaklarını yöneten, Sanal Ağ Fonksiyonları (VNF) ve ağ hizmetlerini sunmak için gereken kaynakların tahsisini düzenleyen NFV mimarisini (ETSI GS NFV 002, 2014) ideal çözüm olarak sunmuşlardır.

SDN paradigmasının incelendiği bir çalışmada (Kreutz ve diğ., 2015), geleneksel IP ağlarının karmaşıklıklarını ve yönetim zorluklarını ele alarak, SDN'in bu sorunları nasıl çözebileceğini araştırmışlardır. SDN'nin, ağın kontrol düzlemini veri düzleminden ayırarak dikey entegrasyonu kırdığı, ağ kontrolünü merkezi hale getirdiği ve ağın programlanabilirliğini artırdığını vurgulamışlardır.

NFV mimarisinde SDN çözümlerinin nasıl kullanılabileceğine dair bir rehber sunan ve standartlaştırılmış olan çalışmada (ETSI GS NFV-EVE 005, 2015), SDN'nin NFV mimarisindeki rolü ve çeşitli kullanım durumları incelenmiştir.

SDN ve NFV teknolojilerinin incelendiği bir çalışmada (Yousaf ve diğ., 2017), ağların esnek ve programlanabilir bir şekilde çok sayıda kullanıcı ve cihazın ihtiyaçlarını karşılamak için büyük bir evrim geçirdiğini belirtmişlerdir. Bu süreçte, IoT, sanallaştırma, yazılımsallaştırma ve bulut tabanlı yaklaşımlar gibi kavramlar öne çıktığını öne sürmüşlerdir. İncelemelerinde, 5G ağ dilimlerinin yönetimi ve işletimi için NFV ve SDN teknolojilerinin ağ mimarisi, tasarımı, işletimi ve yönetimi açısından kilit rol oynadığını belirtmişlerdir. Avrupa Telekomünikasyon Standartları Enstitüsü (ETSI), Açık Ağ Oluşturma Vakfı (ONF) (URL-1) ve 3. Nesil Ortaklık Projesi (3GPP) gibi çeşitli standartlaşma kuruluşlarının, bu teknolojilerin birleştirilmesi ve işlevsel bir sistem olarak uygulanabilmesi için gerekli olan mimari çerçeveleri ve arayüzleri standartlaştırma çabalarına değiniliyor. Aynı zamanda, OpenStack ve Açık Ağ İşletim Sistemi (ONOS) gibi açık kaynak projelerinin de bu standartlaşma sürecine katkıda bulunduğu ve bazı durumlarda ticari ürünlerin yerini alma potansiyeline sahip olduğu belirtiliyor.

SDN ve NFV kullanılarak 5G ağ dilimleme konusunda kapsamlı inceleme yapılan bir çalışmada (Barakabitze ve diğ., 2020), 5G hizmet kalitesi ve iş gereksinimleri ele alınmış olup, 5G ağ yazılımsallaştırma ve dilimleme paradigmaları temel kavramlar, tarihçe ve farklı kullanım senaryolarıyla açıklanmıştır. SDN, NFV, Mobil Uç Bilişim (MEC), bulut/sis bilişim, ağ hipervizörleri, sanal makineler ve konteynerler gibi 5G ağ dilimleme teknolojisini mümkün kılan teknolojilerle ilgili bilgi verilmektedir. 5G ağ dilimlemenin hayata geçirilmesinde SDN ve NFV teknolojilerinin benimsenmesini teşvik eden çeşitli endüstriyel girişimler ve projeler kapsamlı bir şekilde incelenmiştir.

5G ile birlikte ağ dilimlemenin mobil ağların çeşitli gereksinimlerini karşılamada önemli bir rol oynadığı vurgulanan bir çalışmada (Shu ve Taleb, 2020), ağ operatörlerinin fiziksel bir ağı, farklı servis kalitesi (QoS) gereksinimlerini esnek bir şekilde karşılamak için mantıksal olarak ayrılmış birden fazla ağa bölebileceğini belirtmişlerdir. Farklı gereksinimleri karşılayacak olan standartlaştırılmış ağ dilimleri olan Geliştirilmiş Mobil Geniş Bant (eMBB), Yoğun Makine Tipi İletişim (mMTC) ve Ultra Güvenilir ve Düşük Gecikmeli İletişim (URLLC) için SDN ve NFV tabanlı yeni bir QoS çerçevesi önermektedirler. Önerilen QoS çerçevesinde, 5G ağı Radyo Erişim Ağı (RAN), İletim Ağı (TN) ve Çekirdek Şebeke (CN) olarak üç kısma ayrılarak, farklı ağ kaynak tahsis algoritmalarına sahip üç tür ağ dilimi oluşturulmaktadır. Önerilen QoS çerçevesinin farklı

akışları sanal anahtarın (OVS) farklı kuyruklarına yönlendirebildiği, çeşitli ağ dilimi türleri için ağ kaynaklarını planlayabildiği ve kullanıcılara güvenilir uçtan uca QoS sağlayabildiği gösterilmektedir. Önerilen çözüm ile mevcut çözümlerin karşılaştırmalı analizi yapılmıştır.

Servis kalitesi odaklı bir çalışmada (Roddy, 2019), SDN ve NFV özellikli farklı 5G yönetim alanları içinde ağ kaynaklarının paylaşımının etkili bir şekilde yapılması için bilişsel teknikleri kullanılmıştır.

SDN ve NFV özellikli 5G ağ dilimleme teknolojisinin incelendiği bir çalışmada (Matencio-Escolar ve diğ., 2020), çoklu kiracıların (tenant) aynı fiziksel altyapıyı paylaştığı durumlarda garanti edilmiş QoS sağlama yeteneklerini ele almaktadır. Makalede, 5G çok kiracılı (tenant) ağlarda QoS garantileri sağlayan yenilikçi bir yazılım veri yolu mimarisi tasarlanmış ve uygulanmıştır. Geleneksel paket sınıflandırıcıları geliştirerek 5G ağlar için QoS garantileri sağlanması ve ağ dilimi yaşam döngüsünün gerçek zamanlı olarak yönetilmesi incelenmiştir.

5G ağlarında ağ dilimi tahsis algoritmalarının incelendiği bir çalışmada (Gupta ve Misra, 2019), NFV ve SDN teknolojilerinin bir araya gelerek oluşturduğu ağ dilimleme konseptini ele alınmaktadır. Makalede Ağ Kontrol ve Orkestrasyon katmanı, istenildiğinde kullanıcı talebini alıp, uygun bir dilim tahsis ederek bu talebi karşılamaktadır. Dilimde kullanılan kaynaklar, doğrudan VNF'ler ile sanallaştırılmıştır. Orkestrasyon, bir kullanıcının talebini aldığında, ağ dilimi tahsis mekanizması uygun bir dilimi tahsis etmektedir. Makaleye göre büyük sorunlardan biri, herhangi bir hizmet için mükemmel dilimi tahsis etmektir. En uygun ağ dilimi tahsisi yapılabilmesi için makine öğrenmesi tekniğini kullanmışladır.

3. MATERYAL VE METOT

3.1. 5G Ağları ve Temel Bileşenleri

3.1.1. Genel 5G Mimarisi

5G ağları, önceki nesil mobil iletişim teknolojilerine kıyasla çok daha gelişmiş ve karmaşık bir mimariye sahiptir. Bu yeni nesil ağlar, sadece daha yüksek hızlar sunmakla kalmaz, aynı zamanda daha düşük gecikme süreleri, daha yüksek cihaz yoğunluğu ve daha geniş kapsam alanları gibi bir dizi yeni özellik ve avantaj sağlar. 5G mimarisi, temel olarak üç ana bileşenden oluşur: Kullanıcı Ekipmanı (UE), RAN ve CN. Şekil 3.1'de, genel 5G mimarisi gösterilmiştir. Kullanıcı ekipmanları, 5G ağına bağlanan ve iletişim hizmetlerini kullanan cihazlardır. Radyo Erişim Ağı, kullanıcı cihazları ile çekirdek ağ arasında veri iletişimini sağlayan ağ bileşenlerinden oluşur. 5G çekirdek ağı, kullanıcı verilerini yönlendiren, kontrol eden ve yönetim fonksiyonlarını sağlayan ana ağ bileşenidir. 5G CN, esnek ve modüler bir yapıya sahiptir ve ağ dilimleme, sanallaştırma ve bulut teknolojileri gibi ileri düzey özellikleri destekler. 5G çekirdek ağının temel bileşenleri şunlardır:

Erişim ve Hareketlilik Yönetim Fonksiyonu (AMF): Şebekeye erişim ve mobilite yönetimini sağlar.

Oturum Yönetim Fonksiyonu (SMF): Oturum yönetimini gerçekleştirir.

Kullanıcı Düzlemi Fonksiyonu (UPF): Kullanıcı düzlemi veri yollarını yönetir.

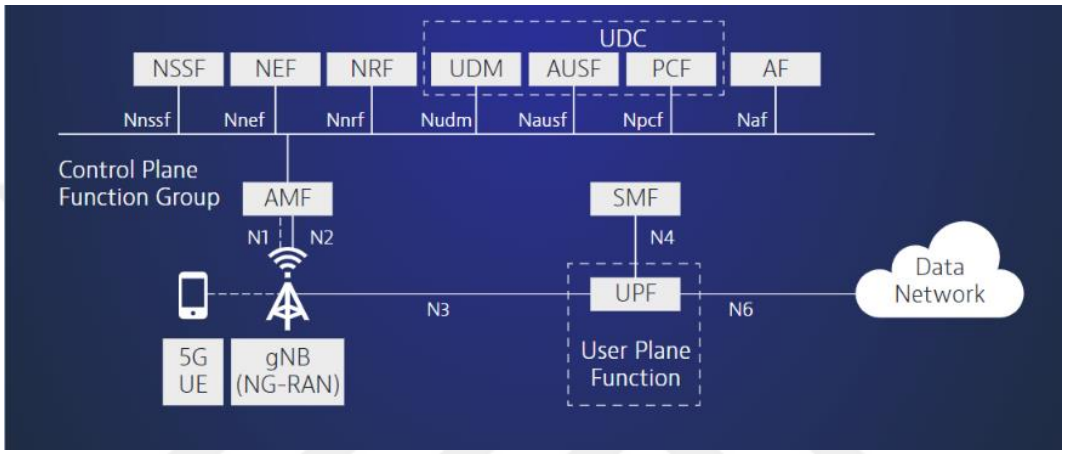
Politika Kontrol Fonksiyonu (PCF): QoS kontrolünü sağlar.

Ağ Dilimi Seçme Fonksiyonu (NSSF): Kullanıcıların uygun ağ diliminden hizmet almasını sağlar.

Kimlik Doğrulama Sunucu Fonksiyonu (AUSF): 5G ağlarında güvenli ve etkili bir kimlik doğrulama sürecinin gerçekleşmesini sağlayan bileşendir.

Birleşik Veri Yönetimi (UDM): Kullanıcıların abonelik bilgilerini merkezi olarak yönetir. Kimlik doğrulama süreçlerinde kullanılmak üzere gerekli bilgileri sağlar. Bu, kullanıcıların ağa erişim yetkilerini doğrulamak için kullanılır.

Ağ Depolama Fonksiyonu (NRF): 5G ağının merkezinde yer alan ve ağ fonksiyonlarının etkin bir şekilde çalışmasını sağlayan bir bileşendir. 5G ağının esneklik ve ölçeklenebilirlik gereksinimlerini karşılamasına yardımcı olur.



Şekil 3.1. Genel 5G mimarisi

N1 arayüzü, UE ile AMF arasında Erişim Dışı Katman (NAS) protokolünü kullanarak sinyalizasyon mesajlarının iletilmesini sağlar. Bu arayüz, UE'nin ağa bağlanmasından, oturum yönetimine ve mobiliteye kadar geniş bir yelpazede iletişim gereksinimlerini karşılamaktadır.

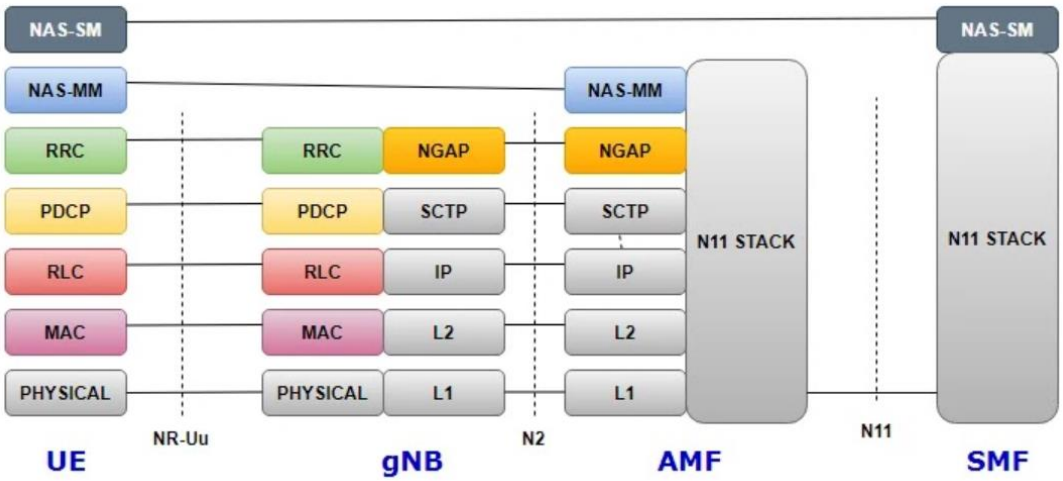
N2 arayüzü, 5G RAN ve AMF arasında kullanılan bir arayüzdür. Bu arayüz, Gelecek Nesil Uygulama Protokolü (NGAP) radyo erişim ağı ile çekirdek ağ arasında kontrol düzlemi mesajlarının taşınması için kullanılmaktadır.

N1 ve N2 arayüzünde taşıma katmanında Akış Kontrollü İletim Protokolü (SCTP) protokolü kullanılmaktadır.

N3 arayüzü, 5G RAN ve UPF arasında kullanılan bir veri düzlemi arayüzüdür. Bu arayüz, kullanıcı verilerinin (kullanıcıların internet trafiği, video akışı, sesli aramalar gibi) 5G RAN 'den çekirdek ağa ve oradan da dış ağlara iletilmesini sağlar. N3 arayüzü üzerinden

veri iletimi GPRS Tünelleme Protokolü - Kullanıcı Düzlemi (GTP-U) kullanılarak yapılır. GTP-U, veri paketlerini 5G RAN ile UPF arasında taşımak için kullanılır.

N4 arayüzü, 5G ağlarında UPF ve SMF arasındaki veri iletim ve oturum yönetimi işlemlerini kontrol eden kritik bir arayüzdür. Paket Yönlendirme Kontrol Protokolü (PFCP) protokolü üzerinden çalışan bu arayüz, ağdaki veri akışlarının ve QoS parametrelerinin etkili bir şekilde yönetilmesini sağlar. Şekil 3.2'de 5G kontrol düzlemi protokol yapısı gösterilmiştir.



Şekil 3.2. 5G kontrol düzlemi protokol yapısı

CN modülleri kendi aralarında servis tabanlı mimari üzerinde Hiper Metin Transfer Protokolü Sürüm 2 (HTTP/2) protokolü aracılığıyla haberleşmektedir. Her modüldeki servislerin Uygulama Programlama Arayüzü (API) ve data modelleri 3GPP Stage 3 dökümanlarında tanımlanmaktadır.

Bir UE içindeki Abone Kimlik Modülü (SIM) kartı, kullanıcının mobil ağa erişimini sağlamak için gerekli olan çeşitli bilgileri içerir. SIM kart içerisindeki önemli bilgiler:

Uluslararası Mobil Abone Kimliği (IMSI): kullanıcıyı global olarak tanımlayan benzersiz bir kimlik numarasıdır. Mobil ağ operatörleri, IMSI'yi kullanarak aboneyi tanımlar ve hizmet sağlar. IMSI genellikle 15 hanelidir ve Mobil Ülke Kodu (MCC), Mobil Şebeke Kodu (MNC) ve abone numarasını içerir.

Operatör Kodu (Opc) : Operatör tarafından belirlenen ve SIM kartta depolanan bir şifreleme anahtarıdır. Kimlik doğrulama ve şifreleme işlemlerinde kullanılır.

Şifreleme Anahtarı (K): Kullanıcı verilerinin şifrelenmesi için kullanılan bir anahtardır. Bu anahtar, SIM kart ile çekirdek ağ arasında güvenli bir iletişim sağlamak amacıyla kullanılır.

Kısa Mesaj Servis Merkezi (SMSC): SMS mesajlarının iletilmesi için kullanılan merkezi bir sistemin adresini içerir. SIM kart, SMS mesajlarını göndermek ve almak için bu bilgiyi kullanır.

Kamu Alanı Mobil Ağı (PLMN) Listesi: PLMN listesi, SIM kartın bağlanabileceği ağ operatörlerini belirler. Bu liste, öncelikli olarak bağlanılacak ağları içerir.

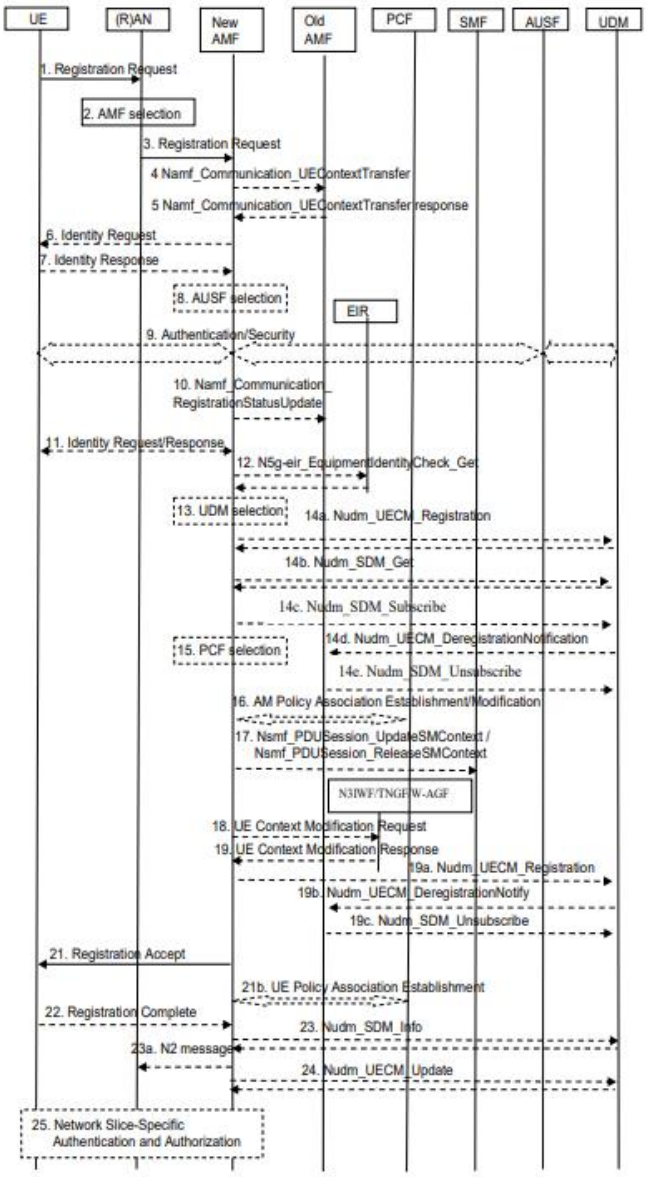
Bir UE'nin 5G ağına kayıtlanması için UE, RAN ve CN arasında 3GPP standartlarında belirtilen sinyalleşme akışı tamamlanmalıdır. Şekil 3.3'te 3GPP 23.502'de yer alan 5G ağına kayıtlanma akışı gösterilmiştir.

UE ağa ilk kez kayıtlanıyorsa mobil kimliği SUCI olacak şekilde, daha önce ağa kayıtlanmış ise mobil kimliği CN tarafından atanmış Geçici Kullanıcı Tanımlayıcı (GUTI) olacak şekilde Kayıt İsteği (Registration Request) mesajı gönderir. Bu mesaj, AMF'ye yönlendirilir. SUCI, 5G ağlarında hava arayüzünde kullanıcı kimliğini gizlemek için kullanılan bir tanımlayıcıdır. SUCI, abonenin gerçek kimliğini (IMSI) gizleyerek kullanıcıların gizliliğini koruma amacıyla tasarlanmıştır. 4G ağlarında UE ağa ilk kayıtlandığında IMSI hava arayüzünde açık bir şekilde gönderilmektedir. Geçici Kullanıcı Tanımlayıcı (GUTI), 5G ağlarında kullanıcı kimliğini geçici olarak tanımlamak

ve gizlemek için kullanılan bir tanımlayıcıdır. UE, ağa ilk bağlandığında ağ tarafından GUTI atanır. Bu geçici kimlik, kullanıcıyı ağ üzerinde tanımlamak ve işlemleri yürütmek için kullanılır.

AMF, UE'nin kimliğini doğrulamak için Authentication (Kimlik Doğrulama) prosedürünü başlatır. Bu prosedürde AUSF (Authentication Server Function), UDM (Unified Data Management) ve UDR (Unified Data Repository) modülleri görev almaktadır. Bu işlemde, UE ve ağ arasında karşılıklı doğrulama yapılır. Bu aşamada

UE'de bulunan SIM kart içindeki IMSI, OPC, K ve PLMN bilgileri UDR'da bulunan "AuthenticationSubscriptionData" bilgileri ile uyumlu olmalıdır. Aksi halde kimlik doğrulaması başarısız olduğu için UE ağa erişemeyecektir.

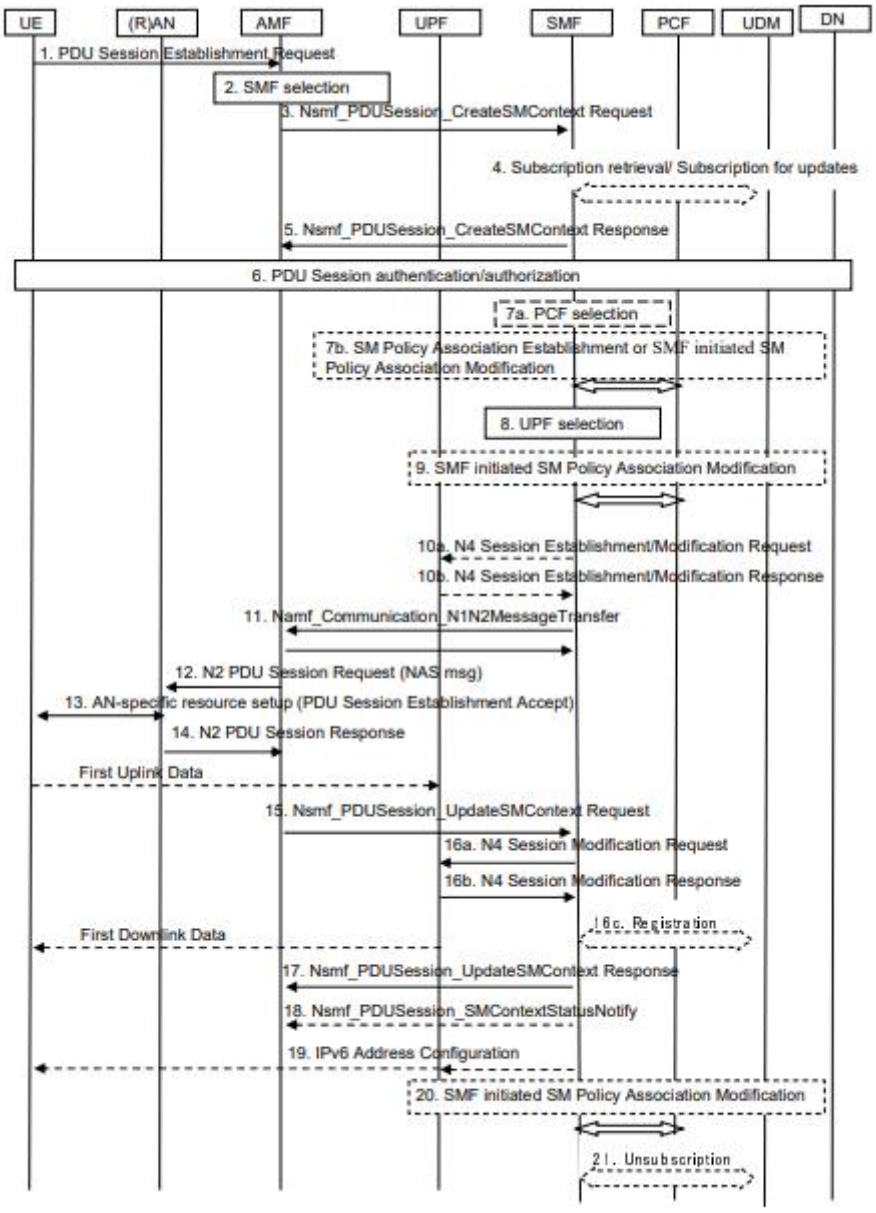


Şekil 3.3. 5G Ağına Kayıtlanma Akışı

Kimlik doğrulama başarılı olduktan sonra, AMF UE'den kayitlanma istegi icinde gonderilen ağ dilimi bilgisini ve UDM aracılığıyla temin ettiği üyelik verisindeki ağ dilimi bilgisini NSSF'e göndermektedir. NSSF bu bilgileri ve MANO'dan aldığı ağ dilimi bilgilerini dikkate alarak kullanıcıyı uygun ağ dilimine yönlendirmesi icin AMF'e yanit vermektedir. Daha sonra ilk kayitlanma istegini alan AMF yada yönlendirilen ağ dilimi

üzerindeki AMF aracılığıyla UE ağa kayıtlanır ve UE'ye dogru Kayıt Kabul (Registration Accept) mesajı gönderilir.

Ağa kayıtlanma işlemi tamamlandıktan sonra, UE'nin veri trafiği gönderebilmesi ve alabilmesi için Protokol Veri Birimi (PDU) Oturumu kurulması gerekir. Şekil 3.4'te 3GPP 23.502'de yer alan PDU Oturumu akışı gösterilmiştir.

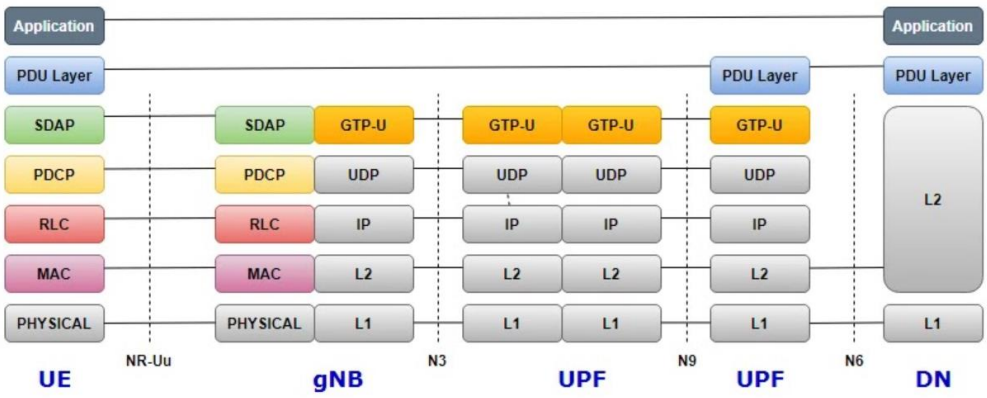


Şekil 3.4. 5G PDU Otırumu kurulum akışı

UE, bir PDU Oturumu Kurma İsteği (PDU Session Establishment Request) mesajı gönderir. Bu mesaj AMF aracılığıyla, SMF'ye yönlendirilir. Ağda birden fazla SMF olması durumunda; örneğin, altyapıda birden fazla ağ dilimi olduğunda ve bu her ağ dilimine tahsis edilmiş SMF var ise, AMF'in uygun ağ dilimindeki SMF ile pdu oturumu kurması gereklidir. Bunun için CN modüllerinden NRF kullanılmaktadır. NRF'nin servis keşfi sağlayan arayüzü aracılığıyla AMF uygun ağ dilimindeki SMF'nin ip adresi ve port numarasına yada Tam Nitelikli Alan Adı (FQDN) bilgisine erişebilmektedir. Çünkü CN modülleri devreye alındıktan sonra NRF'ye ağ fonksiyonu profilleri ile birlikte kayıtlanmaktadırlar ve durumlarını kalp atışı mesajlarıyla güncel tutmaktadırlar. CN'deki ağ fonksiyonu tüketicileri NRF aracılığıyla ağ fonksiyonu sunucularına erişebilmektedir.

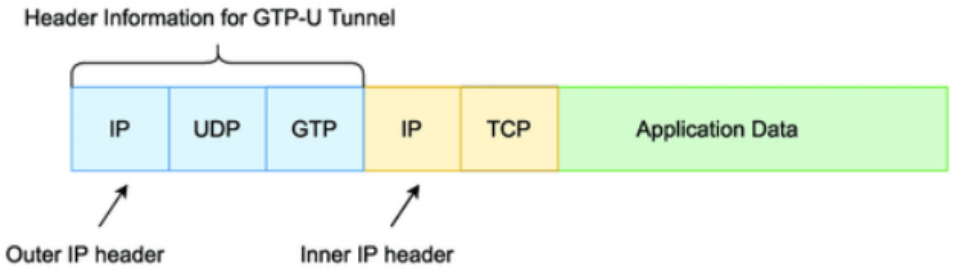
PDU Oturumu kurulumu için uygun SMF seçildikten sonra, SMF, UE'den gelen parametrelere uygun bir UPF seçimi yapmak için NRF'yi kullanmaktadır. UE'den alınan ağ dilimi ve Veri Ağı Adı (DNN) bilgisine göre uygun UPF seçilip, N4 arayüzü aracılığıyla veri akış kuralları ve QoS parametreleri gönderilmektedir. SMF PDU Oturumu bazlı, QoS akışı bazlı veya servis akışı bazlı izin verilen QoS parametrelerini PCF aracılığıyla elde etmektedir.

PDU Oturumu için gerekli sinyalleşme mesajları tamamlandıktan sonra 5G RAN ve UPF arasında GTP-U tüneli kurulmuş olup ve kullanıcı veri trafiği bu tünel aracılığıyla taşınmaktadır. Şekil 3.5'te 5G kullanıcı düzlemi protokol yapısı gösterilmiştir.



Şekil 3.5. 5G kullanıcı düzlemi protokol yapısı

3GPP 29.244 standardında belirtildiği gibi, UPF veri aktarımı sırasında GTP enkapsülasyon ve dekapsülasyon işlemlerini yapmaktadır. Enkapsülasyon, bir veri paketinin başka bir protokol çerçevesi içinde taşınması için hazırlık işlemidir. UPF, kullanıcı verilerini alır ve bu verileri, GTP tünelleme protokolü kullanarak kapsüller. UPF, UE'ye gönderilmesi beklenen veri paketini alır. Bu veri genellikle IP paketleri şeklindedir. UPF, alınan veri paketine bir GTP başlığı ekler. Bu başlık, veri paketinin hangi tünelden geçeceğini, hangi QoS akışına ait olduğunu ve diğer yönetim bilgilerini içerir. UPF'nin aşağı yönlü (downlink) paketleri uygun QoS akışına haritalalandırması gerekmektedir. yukarı yönlü (uplink) yönündeki RAN'dan veri paketleri için ise dekapsülasyon işlemi yapmaktadır. Dekapsülasyon, enkapsüle edilmiş bir veri paketinin orijinal haline getirilmesi işlemidir. UPF, GTP tünelinden gelen veri paketlerini alır ve orijinal IP paketlerini çıkartır. GTP başlığı çıkartıldıktan sonra, orijinal IP paketi elde edilir. Bu paket N6 arayüzü aracılığıyla internet gibi harici ağlara yönlendirilmek üzere işlenir. Şekil 3.6' da UE veri trafiğinin 5G ağ üzerinde nasıl taşındığı gösterilmektedir.



Şekil 3.6. UE veri trafiğinin GTP-U başlığı ile taşınması

3.1.2. 5G Ağ Dilimleme

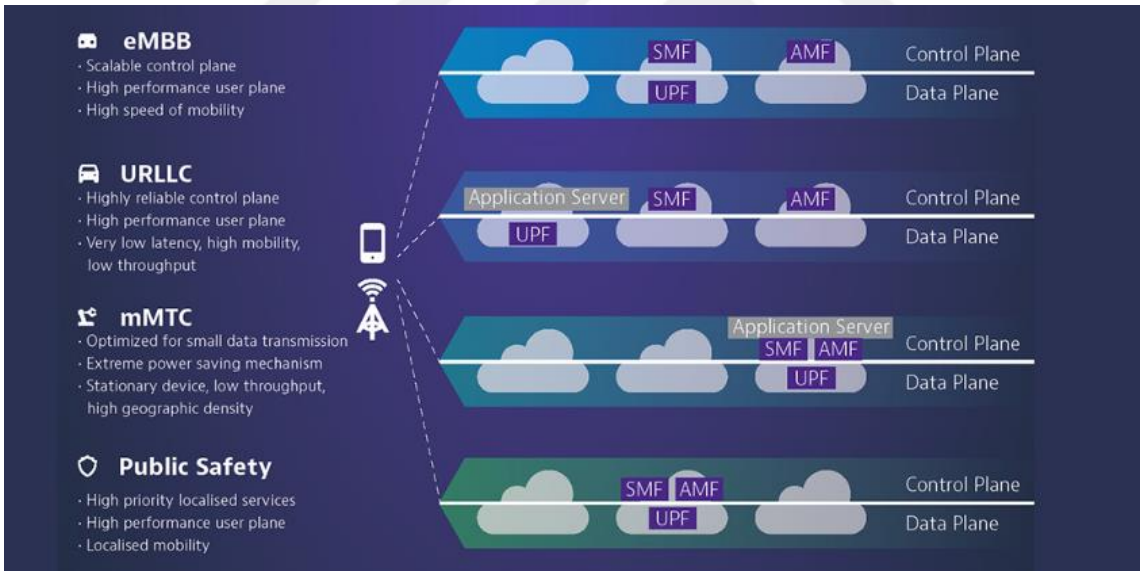
Ağ dilimleme, 5G ağının en yenilikçi özelliklerinden biridir. Bu teknoloji, tek bir fiziksel ağ altyapısının, birbirinden bağımsız ve farklı hizmet gereksinimlerine sahip sanal ağ dilimlerine bölünmesini sağlar. Her dilim, belirli bir hizmetin ihtiyaçlarına göre optimize edilir, böylece farklı kullanım senaryoları için (örneğin, IoT, ultra hızlı mobil geniş bant, düşük gecikmeli uygulamalar) özel ağ kaynakları tahsis edilebilir.

Bu farklı hizmet türlerinden biri eMBB olarak adlandırılır. eMBB, yüksek veri hızları ve geniş bant genişliği sunarak yüksek kaliteli medya akışı, canlı yayın ve interaktif oyun

gibi uygulamalar için idealdir. Örneğin, 4K veya 8K video akışları gibi yüksek çözünürlüklü içerikler, eMBB dilimleri üzerinden hızlı ve kesintisiz bir şekilde iletilir.

URLLC ise, kritik iletişim uygulamaları için tasarlanmıştır. Bu uygulamalar, çok düşük gecikme süreleri ve yüksek güvenilirlik gerektirir. Örneğin, otonom araçlar, akıllı fabrikalar ve tıbbi cerrahi gibi alanlarda kullanılan uygulamalar, URLLC dilimleri üzerinden iletilir ve bu sayede anlık tepki gerektiren durumlarda güvenilir bir iletişim sağlanır.

mMTC de 5G'nin önemli bir bileşenidir. mMTC, büyük miktarda cihazın ve nesnenin birbirleriyle iletişim kurmasını sağlar. IoT uygulamaları, akıllı şehirler, akıllı tarım ve endüstriyel otomasyon gibi alanlarda mMTC büyük önem taşır. Bu uygulamaların yaygınlaşmasıyla birlikte, 5G'nin mMTC desteği, büyük ölçekli cihaz iletişimi için gereken düşük güç tüketimi ve düşük veri aktarımı maliyetleri gibi özellikler sağlar. Şekil 3.7'de ağ dilimleme mimarisi gösterilmiştir.



Şekil 3.7. Ağ dilimleme mimarisi

5G şebekesinde kullanıcının uygun ağ dilimini seçmesi ve o ağ diliminden hizmet alması gerekmektedir. Bu noktada çekirdek şebeke fonksiyonlarından NSSF devreye girer. NSSF kuzey arayüzü aracılığıyla Yönetim ve Orkestrasyon (MANO) biriminden ağ dilimi bilgilerini alır. MANO biriminden alınan ağ dilimi bilgileri, erişilmek istenen ağ

dilimi bilgisi ve abonelik bilgilerindeki ağ dilimi bilgilerine bakılarak, kullanıcının varsayılan ağ dilimine ya da yetkilendirilmiş ağ dilimine erişebilmesi sağlanır.

3.1.3. 5G Ağlarında Servis Kalitesi

QoS, ağ hizmetlerinin performansını belirleyen ve kullanıcı deneyimini doğrudan etkileyen bir dizi parametreyi ifade eder. Başlıca QoS parametreleri şunlardır:

Gecikme (Latency): Bir veri paketinin kaynağından hedefe ulaşması için geçen süre. 5G, ultra düşük gecikme süresi (1 ms altında) sunarak gerçek zamanlı uygulamalar için idealdir.

Bant Genişliği (Bandwidth): Belirli bir zaman diliminde iletilebilen veri miktarı. 5G, yüksek bant genişliği ile yoğun veri gerektiren uygulamaları destekler.

Titreşim (Jitter): Veri paketlerinin iletiminde yaşanan gecikme değişiklikleri. Düşük jitter, özellikle ses ve video gibi gerçek zamanlı hizmetlerde kritik öneme sahiptir.

Paket Kaybı (Packet Loss): İletim sırasında kaybolan veri paketlerinin oranı. Düşük paket kaybı, ağ performansı ve kullanıcı deneyimi için önemlidir.

Güvenilirlik (Reliability): Ağ bağlantılarının sürekliliği ve istikrarı. 5G, yüksek güvenilirlik sunarak kritik görev uygulamaları için güvenli bir iletişim sağlar.

3GPP TS 23.501 standardı, 5G ağlarında QoS akışlarının oluşturulması, yönetilmesi ve izlenmesi için gerekli mekanizmaları ve prosedürleri tanımlar. Başlıca 5G QoS parametreleri şunlardır:

Beşinci Nesil Servis Kalitesi Tanımlayıcısı (5QI) : QoS akışlarını tanımlamak için kullanılan ve her QoS akışına belirli bir öncelik ve parametre kümesi atayan benzersiz bir göstergedir.

Tahsis ve Tutma Önceliği (ARP): Ağ kaynaklarının tahsisi ve korunması için öncelik düzeyini belirler.

Garanti Edilmiş Bit Hızı (GBR): Belirli bir QoS akışı için garanti edilen minimum veri hızı.

Maksimum Bit Hızı (MBR): Belirli bir QoS akışı için izin verilen maksimum veri hızı.

Servis Kalitesi Akış Tanımlayıcısı (QFI): QoS akışlarını benzersiz bir şekilde tanımlayan göstergedir.

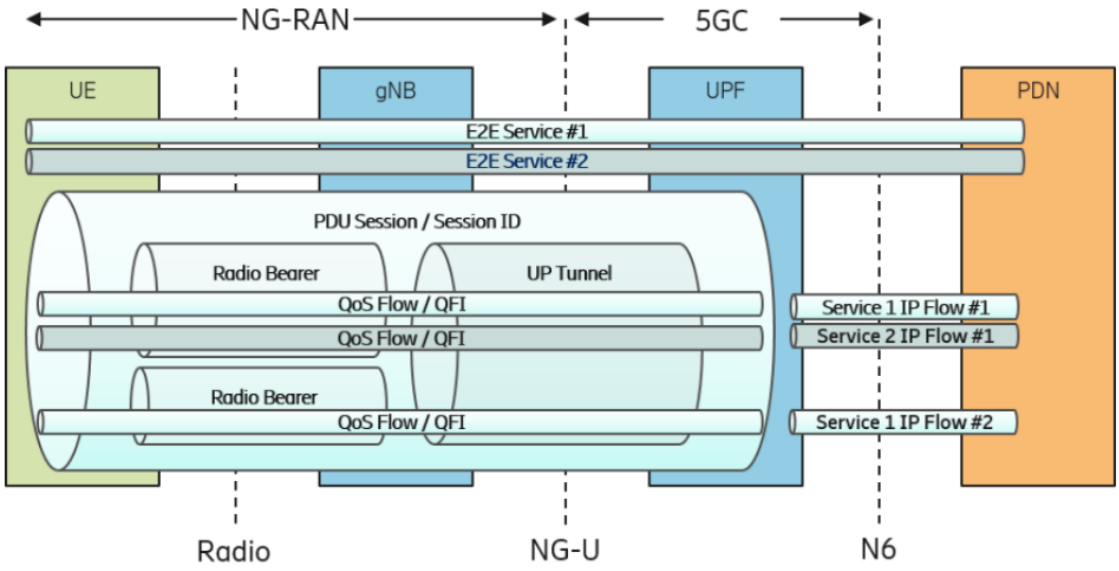
Paket Gecikme Bütçesi (PDB): Bir veri paketinin ağ boyunca taşınması için izin verilen maksimum gecikme süresi.

Paket Hata Oranı (PER): Kabul edilebilir paket hata oranı.

5G ağlarında QoS, QoS akışları ve PDU oturumları aracılığıyla yönetilir. QoS akışları, belirli bir hizmetin QoS gereksinimlerini karşılamak için tanımlanan veri akışlarıdır. Şekil 3.8'de 5G QoS bileşenleri gösterilmiştir.

QoS Akışı: QoS parametreleri ile tanımlanan ve belirli bir hizmetin gereksinimlerini karşılamak üzere yapılandırılan veri akışıdır. Her QoS akışı, bir QFI ile tanımlanır.

PDU Oturumu: UE ile veri ağı (DN) arasındaki bağlantıyı temsil eder. Bir PDU Oturumu, bir veya daha fazla QoS akışını içerebilir.



Şekil 3.8. 5G QoS bileşenleri

5G çekirdek ağında QoS yönetiminde önemli rolü olan fonksiyonlar ve görevleri aşağıda belirtilmiştir:

PCF: QoS kurallarını yönetir. PCF, QoS akışlarının oluşturulması ve yönetilmesi için gerekli politikaları belirler.

SMF: PDU oturumlarını yönetir ve QoS akışlarının kurulmasını sağlar. SMF, Kullanıcı ekipmanından gelen talepler doğrultusunda QoS akışlarını oluşturur ve yönetir.

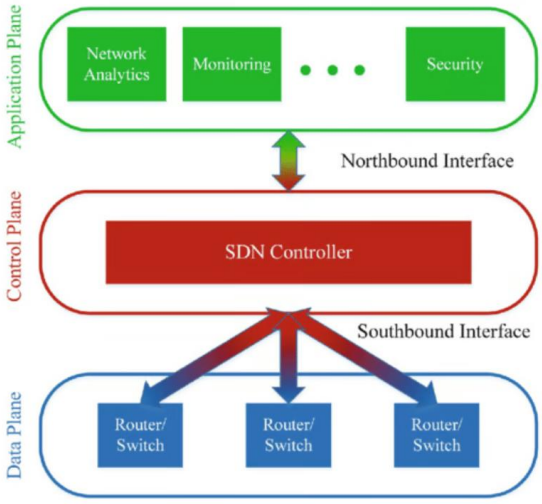
UPF: Kullanıcı verisinin yönlendirilmesini ve QoS parametrelerine göre işlenmesini sağlar. UPF, QoS akışlarının belirlenen parametrelere uygun şekilde iletilmesini sağlar.

3.2. Yazılım Tanımlı Ağlar

Yazılım Tanımlı Ağlar (SDN), ağ yönetimi ve kontrolünü geleneksel ağ cihazlarından (yönlendiriciler ve anahtarlar) ayıran ve bu işlevleri merkezi bir yazılım tabanlı kontrol düzlemine devreden bir ağ mimarisi yaklaşımıdır. SDN, ağ kaynaklarının dinamik, programlanabilir ve merkezi olarak yönetilmesini sağlar.

3.2.1. SDN Mimarisi

SDN teknolojisi ile geleneksel ağ mimarisinde birleşik olan veri ve kontrol düzleminin birbirinden ayrılması sağlanarak, ağ kontrolünün merkezileşmesi amaçlanır. Kontrol düzlemi ile veri düzleminin ayrılması, ağ anahtarları ve SDN Denetleyici arasındaki iyi tanımlanmış bir programlama arayüzü aracılığıyla gerçekleştirilir. SDN Denetleyici, bu uygulama programlama arabirimi (Application Programming Interface, API) aracılığıyla veri düzlemi cihazlarının veri iletim faaliyetlerini doğrudan kontrol edebilir. En çok bilinen API, OpenFlow protokolüdür. OpenFlow anahtarında bir veya daha fazla paket işleme kuralı tablosu bulunur. Trafikle eşleşen her kural trafik üzerinde belirli eylemleri (bırakma, iletme, değiştirme vb.) gerçekleştirir. SDN Denetleyici uygulama tarafından yüklenen kurallara bağlı olarak, OpenFlow anahtarına denetleyici tarafından bir yönlendirici, güvenlik duvarı veya yük dengeleyici gibi davranması talimatı verilebilir. SDN mimarisi, Uygulama Düzlemi, Kontrol Düzlemi ve Veri Düzlemi olmak üzere üç ana bileşenden oluşur. Şekil 3.9'da genel SDN mimarisi gösterilmiştir.



Şekil 3.9. Genel SDN mimarisi

3.2.1.1. Uygulama Düzlemi

SDN uygulama düzlemi, ağ yönetimi ve kontrolü için gerekli olan uygulamaları içerir. Bu düzlem, ağın üst katmanında yer alır ve ağın işleyişini optimize etmek için çeşitli işlevleri yerine getirir. Uygulama düzlemi, SDN Denetleyicisi tarafından sağlanan programlanabilirlik sayesinde ağın dinamik ve esnek bir şekilde yönetilmesini sağlar. Bu düzlemde bulunan uygulamalar, trafik mühendisliği, ağ analizi ve izleme gibi işlevleri gerçekleştirir. Trafik mühendisliği uygulamaları, ağ trafiğinin verimli bir şekilde yönlendirilmesini ve optimize edilmesini sağlar. Bu şekilde, SDN uygulama düzlemi, ağın performansını artırmak için kritik bir rol oynar.

3.2.1.2. Kontrol Düzlemi

SDN kontrol düzlemi, ağın merkezi kontrolünü sağlayan ve ağ kaynaklarının yönetimini ve yönlendirmesini gerçekleştiren önemli bir bileşendir. Bu düzlem, SDN Denetleyicisi olarak adlandırılan merkezi bir yazılım tarafından yönetilir. SDN Denetleyicisi, ağın durumunu izler, ağ topolojisini keşfeder ve ağdaki tüm anahtarları ve yönlendiricileri yönetir. Kontrol düzlemi, ağın işlevselliğini sağlayan ve ağdaki trafiği yönlendiren akıllı algoritmalara dayanır. SDN Denetleyicisi, ağ politikalarını uygular, trafiği optimize eder ve ağdaki kaynakları etkin bir şekilde kullanır. Kontrol düzlemi, ağın veri düzlemi ile uygulama düzlemi arasında bir arayüz sağlar. Bu sayede, ağın dinamik ihtiyaçlara hızlı bir şekilde uyum sağlaması ve gerektiğinde otomatik olarak yeniden yapılandırılması

mümkün olur. SDN kontrol düzlemi, ağ yöneticilerine merkezi bir noktadan ağın durumunu izleme ve yönetme imkanı sağlar. Bu da ağın daha verimli, esnek ve güvenli bir şekilde yönetilmesini mümkün kılar.

3.2.1.3. Veri Düzlemi

Veri düzlemi, ağdaki veri iletim işlevlerini gerçekleştiren fiziksel ve sanal ağ cihazlarını içerir. Bu düzlem, ağ cihazlarının veri paketlerini alması, işlemesi ve yönlendirmesiyle ilgilenir. Ağdaki anahtarlar ve yönlendiriciler gibi cihazlar, veri paketlerini alır ve belirlenen hedeflere iletmek için gerekli işlemleri gerçekleştirir. Veri düzlemi, SDN mimarisinde merkezi bir kontrol noktası olmaksızın, geleneksel ağ yönetimi ve kontrol yöntemlerine dayanır. Ancak, SDN'nin merkezi denetleyici mimarisi, veri düzlemine hâlâ geniş bir şekilde uygulanabilir. Örneğin, SDN Denetleyicisi tarafından sağlanan yönlendirmeleri uygulamak için veri düzlemi kullanılabilir. Bu durumda, veri düzlemi, SDN Denetleyicisinden gelen talimatları alır ve belirlenen ağ politikalarını uygular. Veri düzlemi, ağdaki veri iletim işlevlerinin etkin bir şekilde yerine getirilmesini sağlar. Bu sayede, ağ trafiği akıcı bir şekilde iletilir ve belirlenen hizmet kalitesine uygun olarak yönetilir. Veri düzlemi, ağın gerçek zamanlı gereksinimlerini karşılamak üzere optimize edilmiştir. Bu düzlem, ağdaki veri akışlarını güvenli bir şekilde yönlendirir ve ağ kaynaklarını etkin bir şekilde kullanır. Veri düzlemi, SDN mimarisinin önemli bir bileşenidir ve ağın etkin bir şekilde çalışmasını sağlar.

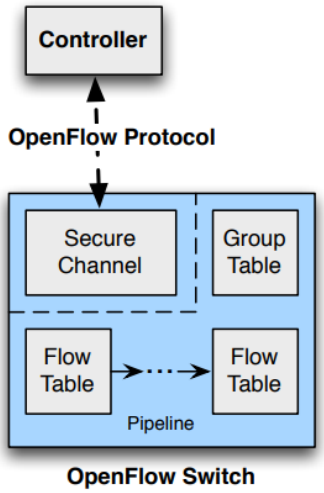
3.2.2. Openflow

OpenFlow, Yazılım Tanımlı Ağların (SDN) temel protokollerinden biridir ve ağ cihazları arasındaki iletişimi sağlamak için kullanılır. OpenFlow protokolü, ağdaki anahtarlar ve denetleyiciler arasında bir iletişim standardıdır. Bu protokol, ağ cihazlarının kontrolünü merkezi bir denetleyiciye devreder, böylece ağın davranışı yazılım tabanlı olarak programlanabilir hale gelir. OpenFlow, ağdaki veri iletim işlevlerini fiziksel cihazlardan (örneğin, anahtarlar ve yönlendiriciler) ayırarak, ağ yönetimini daha esnek ve verimli hale getirir.

Bir OpenFlow anahtarı, bir veya daha fazla akış tablosu ve bir grup tablosundan oluşur. Bu tablolar, paketleri arar ve yönlendirir. Ayrıca, anahtar ile harici bir kontrolör arasında

iletişim kurmak için OpenFlow kanalı kullanılır. Bu bileşenler, OpenFlow protokolü aracılığıyla kontrol edilir ve yönetilir.

OpenFlow anahtarları, fiziksel, mantıksal ve rezerve edilmiş portlar dahil olmak üzere çeşitli port türlerini destekler. Fiziksel portlar, gerçek donanım portlarıdır. Mantıksal portlar, Sanal Yerel Alan Ağı (VLAN) veya tünel portları gibi yazılım tarafından tanımlanan portlardır. Rezerve edilmiş portlar, CONTROLLER veya ALL gibi özel amaçlar için kullanılır. Şekil 3.10'da OpenFlow anahtarın ana bileşenleri gösterilmiştir.



Şekil 3.10. OpenFlow anahtarının ana bileşenleri

OpenFlow, üç temel tablo türü kullanır:

Akış Tablosu: Paketlerin hangi işlemlere tabi tutulacağını belirler. Her akış girdisi, eşleşme alanları, sayaçlar ve talimatlardan oluşur. Eşleşme alanları, gelen paketlerle karşılaştırılır ve uygun talimatlar uygulanır.

Grup Tablosu: Paketlerin birden fazla çıkış portuna yönlendirilmesini sağlar. Bu, yük dengeleme veya yedekli iletim gibi işlemler için kullanılır.

Metre Tablosu: Trafik hızını izler ve belirli hız sınırlarının aşılmasını önler.

Paketler, OpenFlow anahtarına ulaştığında, öncelikle akış tablolarında tanımlı olan akış girdileri ile eşleşir. Bu eşleşme işlemi, öncelik sırasına göre gerçekleştirilir ve her tablo içinde ilk eşleşen akış girdisi kullanılır. Eğer bir paket, bir akış girdisiyle eşleşirse, bu

girdiye bağlı talimatlar uygulanır. Talimatlar, paket yönlendirme, paket modifikasyonu veya grup tablosu işlemleri gibi çeşitli eylemleri içerir. Eğer bir paket, herhangi bir akış girdisiyle eşleşmezse, bu durumda tabloya özel "table-miss" girdisinin yapılandırmasına bağlı olarak işlem görür. "Table-miss" girdisi, paketin kontrolöre gönderilmesi, düşürülmesi veya bir sonraki akış tablosuna yönlendirilmesi gibi sonuçlar doğurabilir. Bu süreç, paketlerin ağda nasıl yönlendirileceğini ve işleneceğini dinamik ve esnek bir şekilde belirler.

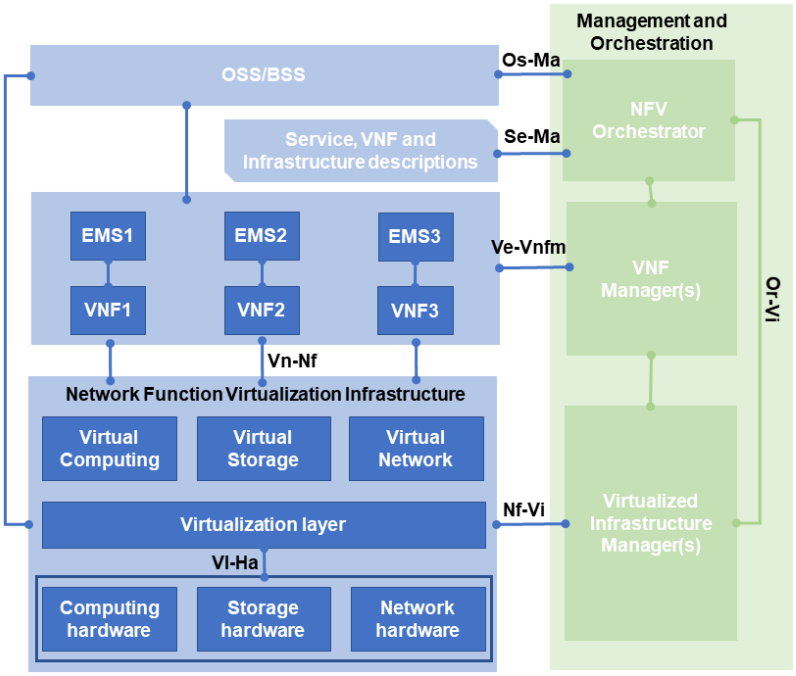
3.3. Ağ Fonksiyonu Sanallaştırma

Ağ Fonksiyonu Sanallaştırma (NFV), geleneksel ağ altyapısını yeniden şekillendiren ve ağ hizmetlerini sanal makineler veya konteynerlar gibi yazılım tabanlı ortamlarda çalıştıran bir teknoloji ve mimari yaklaşımıdır. NFV'nin temel amacı, ağ hizmetlerini sunan donanım tabanlı cihazların karmaşıklığını azaltmak ve ağ hizmetlerini esnek, dinamik ve daha az maliyetli bir şekilde sağlamaktır. Geleneksel olarak, yeni bir ağ hizmeti sunmak veya mevcut bir hizmeti genişletmek için yeni donanım cihazları satın almak ve bunları ağa entegre etmek gerekiyordu. Bu durum hem yatırım maliyetlerini artırıyor hem de hizmet sunum sürelerini uzatıyordu.

NFV, ağ hizmetlerinin yazılım tabanlı sanal makineler veya konteynerlar üzerinde çalıştırılması sayesinde bu sorunları ortadan kaldırır. Bu yaklaşım, ağ hizmetlerinin daha esnek ve hızlı bir şekilde sağlanmasını sağlar. Ayrıca, fiziksel donanım bağımlılığını azaltarak, ağ hizmetlerinin daha iyi ölçeklenebilir ve yönetilebilir olmasını sağlar.

3.3.1. NFV Mimarisi

NFV faaliyetleri Avrupa Telekomünikasyon Standartları Enstitüsü (European Telecommunications Standards Institute, ETSI) tarafından yürütülmektedir. ETSI NFV mimarisi, ağ hizmetlerini sanal makineler veya konteynerlar üzerinde çalıştırarak geleneksel donanım tabanlı ağ cihazlarından ayrıştırmayı ve esnek bir şekilde yönetmeyi hedefler. Bu mimari, NFV'nin genel yapısal ve işlevsel gereksinimlerini tanımlar ve NFV ortamında çalışacak bileşenlerin rollerini ve etkileşimlerini belirler. Şekil 3.11’de ETSI NFV mimarisi gösterilmiştir.



Şekil 3.11. ETSI NFV mimarisi

3.3.2. Ağ Fonksiyonu Sanallaştırma Altyapısı

Ağ Fonksiyonu Sanallaştırma Altyapısı (NFVI), sanallaştırılmış VNF'lerin çalıştırıldığı fiziksel ve sanal altyapı kaynaklarını içerir. Bu kaynaklar, ağ hizmetlerinin sanallaştırılması için gerekli olan hesaplama, bellek, depolama ve ağ kaynaklarını sağlar.

3.3.3. Sanallaştırılmış Altyapı Yöneticisi

Sanallaştırılmış Altyapı Yöneticisi (VIM), NFVI'nın hesaplama, depolama ve ağ kaynaklarının kontrolünden ve yönetilmesinden sorumludur.

3.3.4. Sanallaştırılmış Ağ Fonksiyonu

Sanallaştırılmış Ağ Fonksiyonları (VNF), ağ hizmetlerinin yazılım tabanlı sanal makineler veya konteynerlar üzerinde çalışan uygulamalardır. VNF'ler, ağ hizmetlerinin gereksinimlerine göre dinamik olarak yapılandırılabilir ve ölçeklendirilebilirler. Ayrıca, farklı sanallaştırma platformları ve bulut ortamları arasında taşınabilirlik sağlarlar. Bu, VNF'lerin farklı altyapılar üzerinde çalışabilmesini sağlar.

3.3.5. Sanallaştırılmış Ağ Fonksiyonu Yöneticisi

Sanallaştırılmış Ağ Fonksiyonu Yöneticisi (NFV) ortamında çalışan VNF'lerin yönetimini sağlar. VNF'lerin dağıtılması, konfigüre edilmesi, güncellenmesi ve izlenmesini sağlar. VNF'lerin yaşam döngüsünü yönetir. VNF'lerin oluşturulması, devreye alınması, etkinleştirilmesi, durdurulması, güncellenmesini sağlar. VNF'lerin otomatik ölçeklendirilmesi ve yedeklenmesini sağlar. VNF'lerin performansını ve durumunu izler. VNF'lerden gelen alarmları ve olayları yönetir. Bu, sistemdeki anormallikleri tespit eder, sorunları rapor eder ve gerekli aksiyonların alınmasını sağlar.

3.3.6. NFV Orkestratör

NFV Orkestratör (NFVO) NFV ortamlarında ağ hizmetlerinin yönetimi ve dağıtımını sağlayan bir yazılım veya sistemdir. NFV Orkestratörünün iki ana sorumluluğu vardır. Bunlardan ilki, ağdaki sanal kaynakların doğru bir şekilde tahsis edilmesini ve kullanılmasını sağlamak, ikincisi ise ağ servislerinin yaşam döngüsünün yönetimidir.

3.4. SDN ve NFV İlişkisi

SDN ve NFV arasındaki ilişki, modern ağ altyapılarının dönüşümünde kritik bir rol oynar. SDN, ağ yönetimi ve kontrolünü merkezi bir yazılım tabanlı denetleyici üzerinden sağlayarak, ağın esnekliğini ve programlanabilirliğini artırır. Bu, ağ trafiğinin dinamik olarak yönlendirilmesini, ağ kaynaklarının daha etkin bir şekilde kullanılmasını ve hizmet kalitesinin iyileştirilmesini sağlar. Öte yandan, NFV, geleneksel donanım tabanlı ağ cihazlarının yerini yazılım tabanlı sanal makineler veya konteynerlar üzerinde çalışan VNF'ler ile değiştirerek, ağ hizmetlerinin esnekliğini ve ölçeklenebilirliğini artırır. NFV, ağ hizmetlerinin daha hızlı bir şekilde dağıtılmasını, güncellenmesini ve yönetilmesini sağlar. Bu iki teknolojinin birlikte kullanılmasıyla, ağ altyapısının yönetimi ve hizmetlerin sağlanması daha verimli hale gelir. SDN, ağın dinamik olarak yönetilmesini sağlarken, NFV ise ağ hizmetlerinin esnek ve hızlı bir şekilde sağlanmasını sağlar. Örneğin, SDN denetleyicileri, NFV altyapısının kaynak tahsisi ve ağ trafiğinin yönetimi için kullanılabilir. Böylece, ağ yönetimi ve hizmetlerin sağlanması daha koordineli ve otomatik hale gelir. Sonuç olarak, SDN ve NFV'nin birlikte kullanılması, ağ altyapısının esnekliğini, ölçeklenebilirliğini ve yönetilebilirliğini artırarak, modern ağ ortamlarının

gereksinimlerini daha iyi karşılamasını sağlar. Tablo 3.1'de SDN ve NFV teknolojilerinin birlikte kullanılmasının faydaları tablo halinde gösterilmiştir.

Tablo 3.1. SDN ve NFV teknolojilerinin birlikte kullanılmasının faydaları

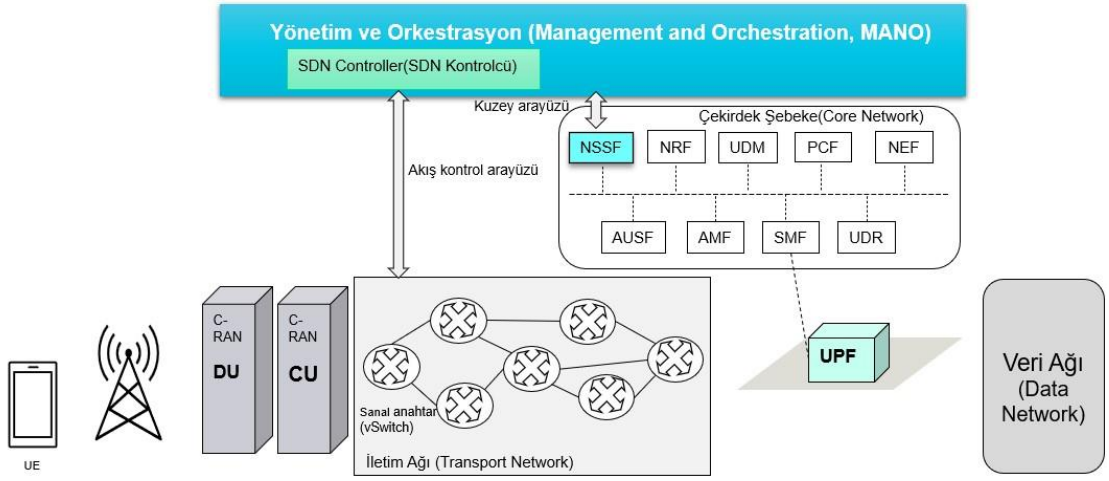
Faydalar	**Açıklama**
Esneklik ve Ölçeklenebilirlik	SDN, ağ yönetimini merkezileştirerek, ağ kaynaklarının esnek ve dinamik bir şekilde yönetilmesini sağlar. NFV ise, ağ fonksiyonlarını yazılım tabanlı sanal makinelerde çalıştırarak, ağ hizmetlerinin esnek ve ölçeklenebilir olmasını sağlar. Bu, ağın gereksinimlere göre daha hızlı ölçeklenmesine ve uyum sağlamasına olanak tanır.
Hızlı Hizmet Dağıtımı	NFV, ağ fonksiyonlarının yazılım tabanlı olarak çalıştırılmasını sağlar, bu da yeni hizmetlerin daha hızlı bir şekilde dağıtılmasını mümkün kılar. SDN ise, hızlı ve merkezi bir kontrol sağlayarak, ağ hizmetlerinin daha hızlı bir şekilde dağıtılmasına yardımcı olur.
Maliyet ve Verimlilik	NFV, donanım cihazlarına olan bağımlılığı azaltarak, ağ maliyetlerini düşürebilir. SDN ise, ağ kaynaklarının daha etkin bir şekilde kullanılmasını sağlayarak, verimliliği artırır. Bu da toplamda maliyetlerin azaltılmasına yardımcı olur.
Dinamik Hizmet Yönetimi	SDN ve NFV bir arada kullanıldığında, ağ hizmetleri daha dinamik bir şekilde yönetilebilir. Yeni gereksinimler doğduğunda veya trafik paternleri değiştiğinde, ağ hizmetleri hızlı bir şekilde yeniden yapılandırılabilir veya ölçeklenebilir.
İyileştirilmiş Güvenlik	SDN ve NFV, ağdaki güvenlik önlemlerini artırmak için birlikte kullanılabilir. Dinamik olarak uygulanabilen politikalar ve sanal güvenlik cihazları gibi özellikler, ağdaki güvenlik seviyelerini artırabilir ve tehditlere karşı daha hızlı bir şekilde yanıt verilebilir.

3.5. SDN ve NFV Tabanlı Dinamik QoS Yönetimi

SDN ve NFV tabanlı dinamik QoS ölçeklendirmesi, ağ trafiğinin uygulama gereksinimlerine göre esnek bir şekilde yönetilmesini sağlar. Örneğin, artan veri trafiğine bağlı olarak, ağdaki mevcut kaynaklar sınırlı olduğunda, kullanıcılar daha düşük bir veri hızı deneyimi yaşayabilir. Örneğin, bir kullanıcı video akışı izlerken, yoğun saatlerde ağ trafiği arttığında, video akışının kalitesi düşebilir veya izleme deneyimi kesintiye uğrayabilir. SDN ve NFV tabanlı bir yapı, bu gibi problemlerin aşılması için ağ kaynaklarının dinamik olarak tahsis edilmesini ve QoS'nin optimize edilmesini mümkün kılar.

Bu çalışmada, NFV-MANO entegrasyonu yapılmış kabul edilen bir ağ üzerinde trafik yoğunluğunun artması durumunda, kullanıcıların servis kalitesinde sürekliliği sağlamak için, ağ kaynaklarının dinamik olarak ölçeklenmesini sağlayan bir SDN algoritması önerilmektedir.

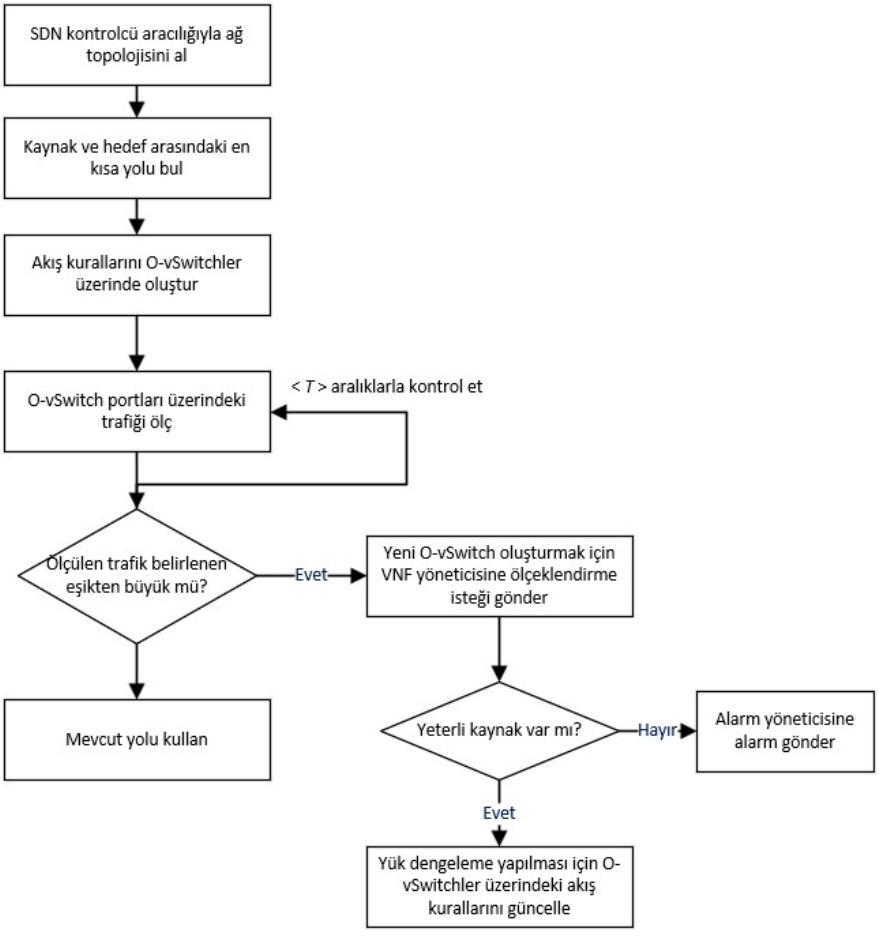
SDN-NFV ağında, sanal anahtarlar NFV mimarisindeki VNF olarak düşünülmektedir. Bu çalışmada, Şekil 3.12'deki 5G iletim ağında tıkanıklık olması durumunda, SDN-NFV tabanlı dinamik ağ kaynağı ölçeklendirme yöntemi önerilmektedir.



Şekil 3.12. Uçtan uca SDN ve NFV tabanlı 5G ağ mimarisi

Geliştirilen algoritmada ağdaki tıkanıklığın tespiti SDN Denetleyici aracılığıyla yapılmaktadır. İlk olarak SDN Denetleyici kuzey arayüzüne Hiper Metin Transfer

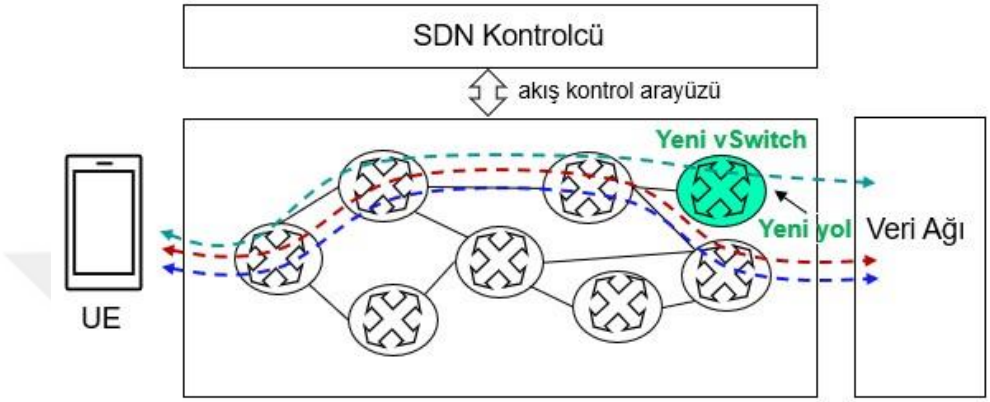
Protokolü (HTTP) GET isteği gönderilerek ağ topoloji bilgisi alınır. Elde edilen topoloji bilgisi ile Dijkstra algoritması kullanılarak kaynak ve hedef arasındaki en kısa yol bulunur. Daha sonra SDN Denetleyici'ye HTTP POST istekleri gönderilerek sanal anahtarlar üzerinde akış kuralları oluşturulur. Böylece trafiğin en kısa yol üzerinden akması sağlanır. Daha sonra T=2 saniye aralıklarla SDN Denetleyici aracılığıyla sanal anahtarların portları üzerindeki bant genişliği değeri ölçülür. Ölçülen bant genişliği değeri, belirlenen eşik değerine ulaştığında tıkanıklığın olduğu sanal anahtar tespit edilmiş olur. Tıkanıklığın olmadığı durumda ise trafik mevcut yol üzerinden akmaya devam eder. Geliştirilen SDN-NFV algoritmasının akış diyagramı Şekil 3.13'te gösterilmiştir.



Şekil 3.13. Geliştirilen SDN-NFV algoritma akış diyagramı

Ağda oluşan tıkanıklık problemi tespit edildikten sonra, tıkanıklığın yaşandığı anahtardaki yükü azaltmak için VNF Yöneticisine yeni bir sanal anahtar oluşturulması

için istek gönderilir. Sistemde yeterli kaynak varsa, VNF Yöneticisi yeni bir sanal anahtar oluşturur. Yeterli kaynak olmaması durumunda, alarm yöneticisine alarm gönderilir. Topolojiye yeni bir sanal anahtar eklenmiş ise, yeni trafik yolu oluşması için gerekli olan akış kuralları sanal anahtarlar üzerinde oluşturulur. Şekil 3.14'te ölçeklendirme yapıldıktan sonra oluşan ağ topolojisi gösterilmiştir.



Şekil 3.14. Ölçeklendirme Sonrası SDN-NFV Ağ Topolojisi

4. TEST ORTAMI

Bu çalışmada SDN Denetleyici olarak Opendaylight kullanılmıştır. Opendaylight, Linux Foundation yöneticiliğinde açık kaynak kodlu bir SDN Denetleyicidür. Veri düzlemini oluşturmak için de Mininet aracı kullanılmıştır. Trafik oluşturmak için iperf aracı kullanılmıştır. Trafiğin topoloji üzerindeki hangi sanal anahtarlar üzerinden aktığını kontrol etmek için SFlow-RT (URL-5) aracı kullanılmıştır. Testler aşağıdaki özelliklere sahip bilgisayar üzerinde yapılmıştır:

İşlemci: 1th Gen Intel® Core™ i5-1135G7 @ 2.40GHz × 8
Hafıza: 7.5 GiB
Depolama: 512.1 GB
Grafik Kart: Mesa Intel® Xe Graphics (TGL GT2)
İşletim Sistemi: Ubuntu 20.04.2 LTS

4.1. Mininet

Mininet (URL-3), ağ ve iletişim protokollerinin simülasyonunu gerçekleştirmek için kullanılan bir açık kaynaklı araçtır. Tek bir komutla basit ağ topolojileri oluşturulabildiği gibi Python programlama dili kullanılarak özelleştirilmiş ağ topolojileri de oluşturulabilmektedir. Bu ağlar, farklı ağ cihazları arasında iletişimi simüle edebilir. Üzerinde ağ anahtarı, link, host oluşturulabilir.

4.2. Opendaylight

OpenDaylight (URL-2), SDN teknolojisinin geliştirilmesi ve uygulanması için oluşturulmuş açık kaynaklı bir platformdur. Linux Foundation tarafından yönetilen OpenDaylight, ağ kontrol ve yönetim işlevlerini merkezi bir denetleyici üzerinden sağlamak amacıyla tasarlanmıştır. Bu platform, SDN Denetleyicisi olarak, ağ yöneticilerinin ağ trafiğini programlanabilir ve dinamik bir şekilde yönetmelerine olanak tanır. Modüler ve genişletilebilir bir yapıya sahiptir.

4.3. Iperf

Iperf (URL-6), ağ performansını ölçmek için kullanılan açık kaynaklı bir araçtır. Ağın bant genişliği, gecikme ve paket kaybı gibi kritik performans metriklerinin

ölçülebilmesini sağlar. İletim Kontrol Protokolü (TCP), Kullanıcı Datagram Protokolü (UDP) gibi farklı protokolleri destekler ve çift yönlü veri akışlarının test edilebilmesini sağlar. Hem istemci hem de sunucu modlarında çalışabilir ve iki nokta arasında veri akışını simüle ederek ağın performansının ölçülebilmesini sağlar.

5. DENEYSEL SONUÇLAR

5.1. Test Senaryosu

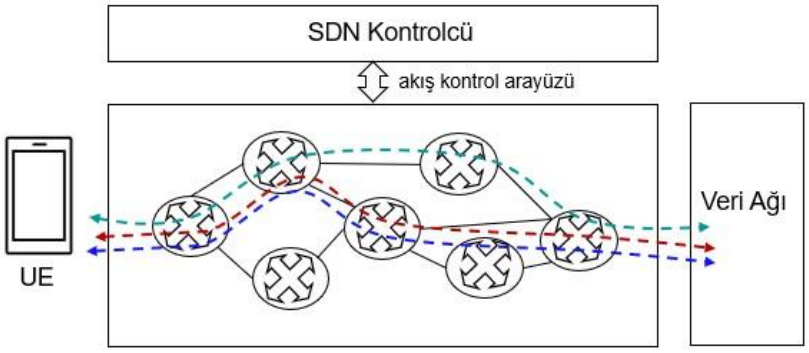
Doğrulama için oluşturulan test senaryosundaki trafik karakteristikleri Tablo 5.1'de gösterilmektedir. Veri ağı tarafında konumlandırılan test sunucularında beklenen veri hızları sırasıyla Akış-1'deki oyun trafiği için 50Mbps, Akış-2'deki video trafiği için 10Mbps ve Akış-3'teki HD video trafiği için 50Mbps'dir.

Tablo 5.1. SDN Ağ Topolojisindeki Trafik Karakteristikleri

Akış No	Trafik Tipi	Beklenen Veri Hızı [Mbps]
Akış-1	Oyun	50
Akış-2	Video	10
Akış-3	HD Video	50

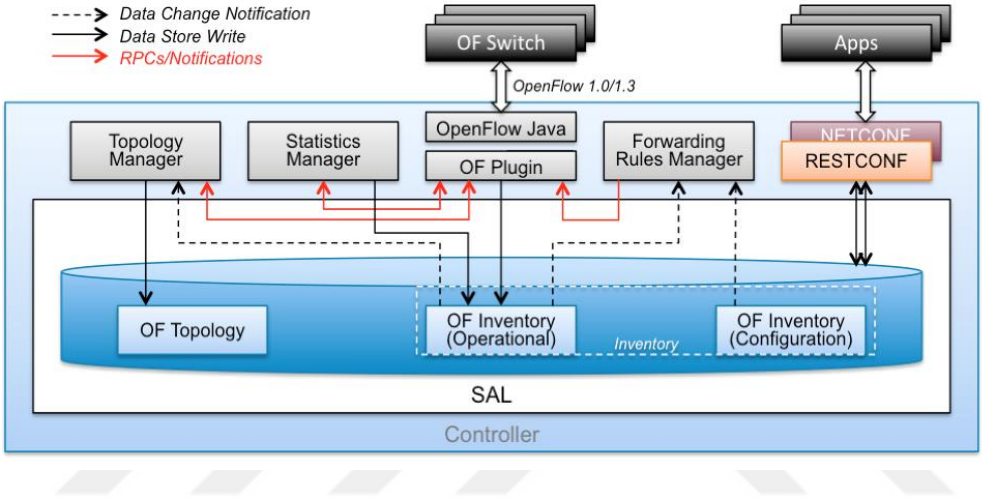
5.2. Simülasyon Sonuçları

Şekil 5.1'de Mininet üzerinde Açık Sanal Anahtarlar (OVS) (URL-4) kullanılarak oluşturulmuş SDN ağ topolojisi gösterilmektedir. Topolojideki anahtarlar arası bant genişliği 100 Mbps'dir. UE'den başlatıldığı kabul edilen Akış-1,2,3 Şekil-15'teki 5G mimarisindeki iletim ağından geçerek veri ağına gönderilen veri akışlarını temsil etmektedir. İletim ağındaki sanal anahtarlar akış kontrol arayüzü aracılığıyla SDN Denetleyici'ye bağlanmaktadır.



Şekil 5.1. SDN ağ topolojisi

Mininet ağındaki topoloji bilgisinin alınması için Opendaylight SDN Denetleyicisinin OpenFlow eklentisi kullanılmıştır. Eklenti, Model Driven Service Abstraction Layer (MD-SAL) mimarisi üzerine kuruludur. Bu mimari, ağ hizmetlerinin soyutlanmasını ve modellenmesini sağlayarak, ağ yöneticilerinin ve uygulama geliştiricilerinin daha kolay bir şekilde ağ hizmetlerini yönetmelerine olanak tanır. Bu sayede, OpenFlow eklentisi, MD-SAL mimarisini kullanarak esnek, ölçeklenebilir bir OpenFlow uygulaması geliştirmeyi hedefler. Şekil 5.2'de OpenFlow eklentisinin mimarisi gösterilmiştir.



Şekil 5.2. Opendaylight – Openflow eklentisinin mimarisi

OpenFlow Eklentisi aşağıdaki RESTCONF API'lerini sağlar:

OpenFlow Topolojisi

OpenFlow İstatistikleri

OpenFlow Programlama

Ağ topoloji bilgileri (düğümler ve bağlantılar) denetleyiciye RESTCONF isteği gönderilerek alınır. Alınan topoloji bilgisi üzerinde Dijkstra algoritması kullanılarak en kısa yollar bulunur. Trafiğin bulunan en kısa yol üzerinden akması için OpenFlow eklentisi kullanılarak OpenFlow anahtarlar üzerinde akış kuralları oluşturulur. Akış kurallarını oluşturmak için denetleyiciye RESTCONF istekleri gönderilir. Portlar üzerindeki trafik miktarının ölçülmesi için OpenFlow eklentisinin istatistik özelliği kullanılır.

Bu bölümde, ağda tıkanıklık olması durumunda kullanıcıların servis kalitesinin düşmemesi için kullanılan 2 yöntemin karşılaştırması yapılmıştır. İlk yöntemde tıkanıklık durumunda trafik farklı en kısa yollar üzerinden yönlendirilmiştir. İkinci olarak bu çalışmada önerilen SDN-NFV tabanlı dinamik ağ ölçeklendirme yöntemi uygulanmıştır. Her 2 yöntem simülasyon ortamında gerçeklenerek, performansları karşılaştırılmıştır. Performans belirteci olarak QoS parametrelerinden biri olan ortalama veri hızları dikkate alınmıştı Simülasyon ortamında Tablo 1'deki trafik karakteristiklerine uygun akışlar eş zamanlı olarak başlatılmıştır. Bir süre sonra veri aktarım hızlarında düşüşlerin olduğu ve ağda tıkanıklık meydana geldiği gözlemlenmiştir.

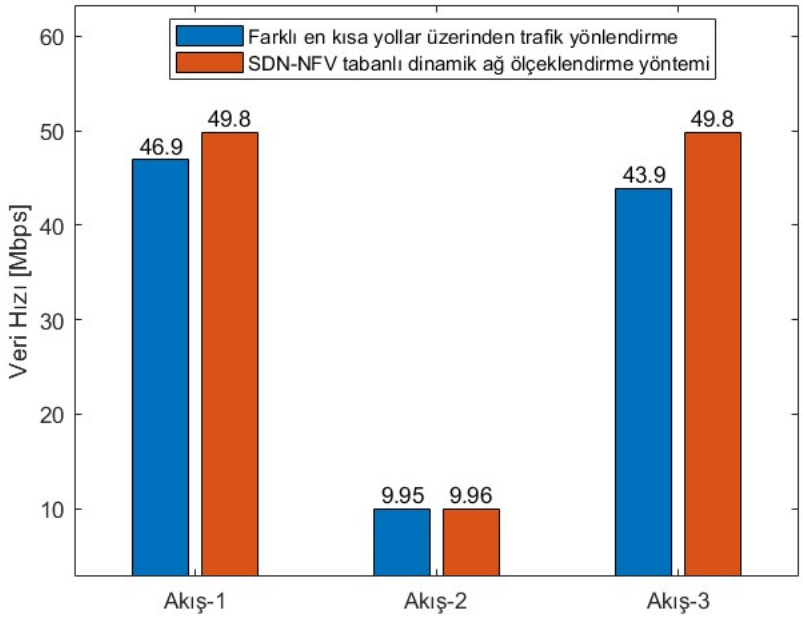
İlk olarak ağda tıkanıklık olduğu durumda, ölçeklendirme yapılmadan, SDN Denetleyici aracılığıyla Dijkstra algoritması kullanılarak farklı en kısa yollar üzerinden yönlendirme yapılmıştır. Bu yöntem uygulandığında alıcıdaki ortalama veri hızları akış başına sırasıyla Akış-1 için 46,9 Mbps, Akış-2 için 9,95 Mbps, Akış-3 için 43,9 Mbps olarak ölçülmüştür. Bu yöntem başarım oranı açısından değerlendirildiğinde alıcı tarafta Akış-1 için %93,8 oranında, Akış-2 için %99,5 oranında, Akış-3 için %87,8 oranında beklenen hız değerlerine ulaşıldığı görülmüştür.

Bu çalışmada önerilen SDN-NFV tabanlı dinamik ağ ölçeklendirme yöntemi uygulandığında, alıcıdaki ortalama veri hızları akış başına sırasıyla Akış-1 için 49,8 Mbps, Akış-2 için 9,96 Mbps, Akış-3 için 49,8 Mbps olarak ölçülmüştür. Başarım oranı olarak değerlendirildiğinde bütün akışlar için beklenen veri hızlarına %99,6 oranında ulaşıldığı görülmüştür. Tablo 5.2'de her iki yöntem için akış başına elde edilen ortalama veri hızları gösterilmektedir.

Tablo 5.2. Farklı En Kısa Yollar Üzerinden Trafik Yönlendirme ve SDN-NFV Tabanlı Dinamik Ağ Ölçeklendirme Simülasyon Sonuçları

Akış No	**Farklı en kısa yollar üzerinden trafik yönlendirme**	**SDN-NFV tabanlı dinamik ağ ölçeklendirme yöntemi**
Akış-1	46,9Mbps	49,8Mbps
Akış-2	9,95Mbps	9,96Mbps
Akış-3	43,9Mbps	49,8Mbps

Şekil 5.3’te farklı en kısa yollar üzerinden trafik yönlendirme ve önerilen SDN-NFV tabanlı dinamik ağ ölçeklendirme yönteminin simülasyon sonuçları grafiksel olarak gösterilmektedir.



Şekil 5.3. Ortalama veri hızına göre sistem performans karşılaştırılması

6. SONUÇLAR VE ÖNERİLER

Bu çalışmada, 5G ağlarında SDN ve NFV teknolojilerinin entegrasyonu ile kullanıcıların servis kalitesinin dinamik olarak nasıl yönetilebileceği incelenmiştir. 5G ve ötesi haberleşme sistemlerinin artan veri taleplerine yanıt verebilmek için ağların esnek ve ölçeklenebilir olması gerekmektedir. Bu gereksinimleri karşılamak amacıyla, Opendaylight SDN Denetleyicisi kullanılarak iletim ağındaki tıkanıklıkların tespiti sağlanmış ve NFV yöneticisine ağ ölçeklendirme için talepte bulunan özel bir uygulama geliştirilmiştir.

Opendaylight SDN Denetleyicisi, ağ trafiğini merkezi bir noktadan yöneterek tıkanıklık noktalarını etkin bir şekilde belirler. Bu tıkanıklık tespiti sonucunda, ağ performansını optimize etmek için NFV yöneticisi, sanal ağ fonksiyonlarını dinamik olarak ölçeklendirme talimatı alır. Böylece, tıkanıklığın olduğu bölgelerde ek kaynaklar devreye alınarak ağın genel performansı iyileştirilir.

Ağda tıkanıklık meydana geldiğinde, geleneksel yöntemler kullanılarak farklı en kısa yollar üzerinden trafik yönlendirme yöntemi uygulandığında, alıcı tarafta beklenen veri hızlarına %93,7 oranında ulaşıldığı gözlemlenmiştir. Bu yöntem, mevcut ağ kaynaklarının yeniden dağıtılması ve trafiğin alternatif yollara yönlendirilmesi ile gerçekleştirilmiştir. Ancak, bu yaklaşım her zaman yeterli olmamakta ve bazı durumlarda beklenen performansa ulaşmakta zorlanılmaktadır.

Bu çalışmada önerilen SDN-NFV tabanlı dinamik ağ ölçeklendirme yönteminin kullanılması durumunda ise beklenen veri hızlarına %99,6 oranında ulaşıldığı görülmüştür. Önerilen yöntem, SDN Denetleyicisinin tıkanıklık tespiti ve NFV yöneticisinin dinamik kaynak tahsisi ile birleştirilmesi sayesinde, ağ kaynaklarının daha etkin kullanılmasını ve kullanıcı deneyiminin iyileştirilmesini sağlamaktadır. Bu yüksek başarı oranı, ağ performansını optimize etmek için SDN ve NFV teknolojilerinin birlikte kullanılmasının etkinliğini göstermektedir.

Sonuç olarak, bu çalışmada geliştirilen SDN-NFV tabanlı dinamik ağ ölçeklendirme yönteminin, 5G ve ötesi haberleşme sistemlerinde servis kalitesinde sürekliliğin sağlanması için etkili bir çözüm sunduğu ortaya konulmuştur. Bu yöntem, ağların değişen

taleplere hızlı bir şekilde yanıt vermesini ve yüksek veri hızlarını sürekli olarak sağlamasını mümkün kılmaktadır. 5G teknolojilerinin yaygınlaşması ile birlikte, bu tür yenilikçi yaklaşımların önemi daha da artacak ve ağ yönetiminde standart bir uygulama haline gelecektir.

KAYNAKLAR

3GPP. (Ocak 2024). *Procedures for the 5G System. Technical Specification (TS) 23.502*, Rel. 17.

3GPP. (Nisan 2024). *LTE; 5G; Interface between the Control Plane and the User Plane nodes. Technical Specification (TS) 29.244*, Rel. 17.

Barakabitze, A., Ahmad, A., Mijumbi, R. ve Hines, A., (2020). 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges. *Computer Networks, 167*(3), 106984.

ETSI. (Aralık 2014). *GS NFV 002 Network Functions Virtualization (NFV); Architectural Framework*, v.1.2.1.

ETSI. (Aralık 2015). *GS NFV-EVE 005 Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework*, v.1.1.1.

Gupta, A. ve Jha, R. K., (2015). A Survey of 5G Network: Architecture and Emerging Technologies. *IEEE Access, 3*, 1206-1232.

Gupta, R. K.ve Misra, R., (2019). Machine Learning-based Slice allocation Algorithms in 5G Networks. *International Conference on Advances in Computing, Communication and Control (ICAC3)*, Mumbai, India, 2019.

Kreutz, D., Ramos, F. M. V., Veríssimo, P. E., Rothenberg, C. E., Azodolmolky, S ve Uhlig, S., (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE, 103*(1), 14-76.

Matencio-Escolar, A., Wang, Q. ve Calero, J. M. Alcaraz, (2020). SliceNetVSwitch: Definition, Design and Implementation of 5G Multi-Tenant Network Slicing in Software Data Paths. *IEEE Transactions on Network and Service Management, 17*(4), 2212-2225.

Ordonez-Lucena, J., Ameigeiras, P., Lopez, D., Ramos-Munoz, J. J., Lorca, J. ve Folgueira,J., (2017). Network slicing for 5G with SDN/NFV: Concepts architectures and challenges. *IEEE Commun. Mag., 55*(5), 80-87.

Roddy, M., (2019). 5G Network Slicing for Mission-critical use cases. *IEEE 2nd 5G World Forum (5GWF)*, Dresden, Germany, 2019.

Shu, Z. ve Taleb, T., (2020). A Novel QoS Framework for Network Slicing in 5G and Beyond Networks Based on SDN and NFV. *IEEE Network, 34*(3), 256-263.

URL-1: https://www.opennetworking.org, (Ziyaret tarihi: 10 Şubat 2023).

URL-2: https://docs.opendaylight.org/en/stable-potassium/index.html, (Ziyaret tarihi: 10 Şubat 2023).

URL-3: https://github.com/mininet/mininet/wiki/Documentation, (Ziyaret tarihi: 11 Şubat 2023).

URL-4: https://www.openvswitch.org/, (Ziyaret tarihi: 11 Şubat 2023).

URL-5: https://inmon.com/products/sFlow-RT.php, (Ziyaret tarihi: 12 Şubat 2023).

URL-6: https://iperf.fr/, (Ziyaret tarihi: 15 Şubat 2023).

Yousaf, F. Z., Bredel, M., Schaller, S. ve Schneider, F., (2017). NFV and SDN—Key Technology Enablers for 5G Networks. *IEEE Journal on Selected Areas in Communications, 35*(11), 2468-2478.

EKLER

Ek-A Mininet Topology

```
#!/usr/bin/python
from mininet.topo import Topo
from mininet.cli import CLI
from mininet.net import Mininet
from mininet.log import setLogLevel
from mininet.node import RemoteController
from mininet.link import TCLink
REMOTE_CONTROLLER_IP = "127.0.0.1"
class SingleLoopTopo(Topo):
   # Single switch connected to n hosts
   def __init__(self, **opts):
      # Initialize topology and default options
      Topo.__init__(self, **opts)
      switches = []
      hosts = []
      # create switches
      for s in range(7):
         switches.append(self.addSwitch('s%s' % (s + 1), protocols='OpenFlow13'))
      # create hosts
      for h in range(5):
         hosts.append(self.addHost('h%s' % (h + 1)))
      self.addLink(hosts[0], switches[0],cls=TCLink,bw=100)
      self.addLink(hosts[1], switches[0],cls=TCLink,bw=100)
      self.addLink(hosts[2], switches[0],cls=TCLink,bw=100)
      self.addLink(hosts[3], switches[6],cls=TCLink,bw=100)
      self.addLink(hosts[4], switches[6],cls=TCLink,bw=100)
      self.addLink(switches[0], switches[2],cls=TCLink,bw=100)
      self.addLink(switches[0], switches[1],cls=TCLink,bw=100)
      self.addLink(switches[1], switches[4],cls=TCLink,bw=100)
      self.addLink(switches[3], switches[2],cls=TCLink,bw=100)
      self.addLink(switches[3], switches[1],cls=TCLink,bw=100)
      self.addLink(switches[4], switches[6],cls=TCLink,bw=100)
      self.addLink(switches[3], switches[6],cls=TCLink,bw=100)
      self.addLink(switches[3], switches[5],cls=TCLink,bw=100)
      self.addLink(switches[5], switches[6],cls=TCLink,bw=100)
def main():
   setLogLevel('info')
   topo = SingleLoopTopo()
   net = Mininet(topo=topo,
             controller=None,
             autoStaticArp=True)
   net.addController("c0",
                controller=RemoteController,
                ip=REMOTE_CONTROLLER_IP,
                port=6633, protocols='OpenFlow13')
   net.start()
```

```
    CLI(net)

if __name__ == '__main__':
    main()
topos = { 'mytopo': ( lambda: main() ) }
```

Ek-B SDN Uygulama Fonksiyonları

```
import requests
import json
import networkx as nx
import argparse
# ODL controller details
ODL_CONTROLLER_IP = '127.0.0.1'
ODL_CONTROLLER_PORT = '8181'
ODL_USERNAME = 'admin'
ODL_PASSWORD = 'admin'
DEFAULT_PRIORITY = 32768
def get_topology():
    # OpenDaylight RESTCONF API endpoint for topology information
    topology_endpoint =
f"http://{ODL_CONTROLLER_IP}:{ODL_CONTROLLER_PORT}/restconf/operatio
nal/network-topology:network-topology"
    # Get topology information from OpenDaylight
    response = requests.get(topology_endpoint, auth=(ODL_USERNAME,
ODL_PASSWORD))
    print(response.json())
    if response.status_code != 200:
        print(f"Failed to retrieve topology information. Status code:
{response.status_code}")
        return
    # Parse topology data
    topology_data = response.json()
    return topology_data
def out_link_finder(topology_in,source_node_in, destination_node_in):
    if topology_in:
      links = topology_in.get("network-topology", {}).get("topology", [])[0].get('link', [])
      ports_dict = dict()
      if links:
          print("Links in the network:")
          for link in links:
              source_node = link.get('source', {}).get('source-node', '')
              source_port = link.get('source', {}).get('source-tp', '')
              destination_node = link.get('destination', {}).get('dest-node', '')
              destination_port = link.get('destination', {}).get('dest-tp', '')
              print(f"Link from {source_node}:{source_port} to
{destination_node}:{destination_port}")
```

```
            if source_node == source_node_in and
destination_node==destination_node_in:
                 # Split the string using space as the delimiter
                source_port_split = source_port.split(":")
                out_port = source_port_split[-1]
                destination_port_split = destination_port.split(":")
                in_port = destination_port_split[-1]
                ports_dict['out_port']=out_port
                ports_dict['in_port']=in_port
                break
        print(ports_dict)
        return ports_dict
    else:
        print("No links found in the topology.")
  else:
     print("Unable to retrieve links information.")
def add_flow_to_odl(node, in_port, out_port,dst_ip,priority_in):
  x = dst_ip.split("/")[0]
  url =
f'http://{ODL_CONTROLLER_IP}:{ODL_CONTROLLER_PORT}/restconf/config/o
pendaylight-inventory:nodes/node/{node}/table/0/flow/{in_port}{x}'
  headers = {
    'Content-Type': 'application/json',
    'Accept': 'application/json',
  }
  auth = (ODL_USERNAME, ODL_PASSWORD)
  print(f"NODE= {node} IN_PORT={in_port} OUT_PORT={out_port}")
  flow_data = {
    "flow-node-inventory:flow": [
      {
        "id": str(in_port)+x,
        "table_id": 0,
        "match": {
          "in-port": str(in_port),
                "ipv4-destination": dst_ip,
          "ethernet-match": {
            "ethernet-type": {
            "type": 2048
            }
    }
        },
        "instructions": {
          "instruction": [
            {
              "order": 0,
              "apply-actions": {
                "action": [
                  {
```

```
                        "order": 0,
                        "output-action": {
                            "output-node-connector": out_port
                        }
                    }
                ]
            }
        }
    ]
  },
  "priority": priority_in
 }
 ]
}
response = requests.put(url, headers=headers, auth=auth,
data=json.dumps(flow_data))

if response.status_code == 200 or response.status_code == 201:
    print(f"Flow added successfully on node {node}.")
else:
    print(f"Failed to add flow on node {node}. Status code: {response.status_code}")
    print(response.text)
def get_port_statistics(topology_in):
 # OpenDaylight RESTCONF API endpoint for port statistics information
   nodes = topology_in.get("network-topology", {}).get("topology", [])[0].get('node',
[])
   all_port_stats = []
   if nodes:
      for node in nodes:
         node_id = node.get('node-id', '')
         node_ports = node.get('termination-point', [])
         for each_node_ports in node_ports:
            port_stats = dict()
            port = each_node_ports.get('tp-id', '')
            port_stat_endpoint =
f"http://{ODL_CONTROLLER_IP}:{ODL_CONTROLLER_PORT}/restconf/operatio
nal/opendaylight-inventory:nodes/node/{node_id}/node-connector/{port}/opendaylight-
port-statistics:flow-capable-node-connector-statistics/bytes"
            response = requests.get(port_stat_endpoint, auth=(ODL_USERNAME,
ODL_PASSWORD))
            odl_port_stat = response.json().get('opendaylight-port-statistics:bytes',{})
            odl_port_transmitted_mbyte = "{:.2f}".format(odl_port_stat['transmitted']/
(1024.0 ** 2))
            odl_port_received_mbyte= "{:.2f}".format(odl_port_stat['received']/
(1024.0 ** 2))
            print(f"Switch: {node_id} port: {port} received
MB:{odl_port_received_mbyte} transmitted MB:{odl_port_transmitted_mbyte}")
            port_stats["node"] = node_id
```

```
                port_stats["interface"] = port
                port_stats["received"] = odl_port_received_mbyte
                port_stats["transmitted"] = odl_port_transmitted_mbyte
                all_port_stats.append(port_stats)
        return all_port_stats
def delete_all_flows(node_id):
    url =
f'http://{ODL_CONTROLLER_IP}:{ODL_CONTROLLER_PORT}/restconf/config/o
pendaylight-inventory:nodes/node/{node_id}/table/0'
    headers = {
        'Content-Type': 'application/xml',
        'Accept': 'application/xml',
    }
    auth = (ODL_USERNAME, ODL_PASSWORD)
    # Send a DELETE request to delete all flows in the table
    response = requests.delete(url, headers=headers, auth=auth)

    if response.status_code == 204:
        print(f"All flows deleted successfully for OVS node {node_id}.")
    else:
        print(f"Failed to delete flows for OVS node {node_id}. Status code:
{response.status_code}")

def delete_one_flow(node_id,in_port):
    url =
f'http://{ODL_CONTROLLER_IP}:{ODL_CONTROLLER_PORT}/restconf/config/o
pendaylight-inventory:nodes/node/{node_id}/flow-node-
inventory:table/0/flow/{in_port}'
    print(url)
    headers = {
        'Content-Type': 'application/xml',
        'Accept': 'application/xml',
    }
    auth = (ODL_USERNAME, ODL_PASSWORD)
    # Send a DELETE request to delete all flows in the table
    response = requests.delete(url, headers=headers, auth=auth)
    if response.status_code == 204:
        print(f"All flows deleted successfully for OVS node {node_id}.")
    else:
 print(f"Failed to delete flows for OVS node {node_id}. Status code:
{response.status_code}")
def get_flow(node_id)  :
    url = f'http://127.0.0.1:8181/restconf/operational/opendaylight-
inventory:nodes/node/{node_id}/flow-node-inventory:table/0'
    print(url)
    headers = {
        'Content-Type': 'application/xml',
        'Accept': 'application/xml',
```

```
    }
    auth = (ODL_USERNAME, ODL_PASSWORD)
    # Send a GET request to delete all flows in the table
    response = requests.get(url, headers=headers, auth=auth)
    if response.status_code == 200:
        print(f"All flows get successfully for OVS node {node_id}.")
        o = xmltodict.parse(response.text)
        return json.loads(json.dumps(o))
    else:
        print(f"Failed to get flows for OVS node {node_id}. Status code:
{response.status_code}")
def
add_flow_manager(topology_in,shortest_path_in,in_port,out_port,dest_ip,priority_in,o
ut) :
        path_info_list = []
        path_info_all = dict()
        for i in range(len(shortest_path_in) - 1):
            path_info = dict()
            ports_dict =
out_link_finder(topology_in,shortest_path_in[i],shortest_path_in[i+1])
            next_in_port = ports_dict['in_port']
            if ports_dict['out_port'] and next_in_port:
              path_info["node"] = shortest_path_in[i]
              path_info["in"] = in_port
              path_info["out"] = ports_dict['out_port']
              path_info_list.append(path_info)
add_flow_to_odl(shortest_path_in[i],in_port,ports_dict['out_port'],dest_ip,priority_in)
              in_port = next_in_port
        path_info = dict()
        path_info["node"] = shortest_path_in[len(shortest_path_in) - 1]
        path_info["in"] = in_port
        path_info["out"] = out_port
        path_info_list.append(path_info)
        path_info_all["shortest_path"] = path_info_list
        if dest_ip == "10.0.0.4/32":
            out_port=out
            add_flow_to_odl(shortest_path_in[len(shortest_path_in) -
1],in_port,out_port,dest_ip,priority_in)
        elif dest_ip == "10.0.0.5/32":
            out_port=out
            add_flow_to_odl(shortest_path_in[len(shortest_path_in) -
1],in_port,out_port,dest_ip,priority_in)
        return path_info_all
def convert_shortest_path_2_ofsw(shortest_path_in):
    of_shortest_path = []
    for each_shortest_path in shortest_path_in:
            of_shortest_path.append("openflow:"+str(each_shortest_path))
    return of_shortest_path
```

```
def candidate_shortest_path_finder(initial_shortest_path_in,all_shortest_paths_list_in):
    max_diff = 0
    candidate_sp = initial_shortest_path_in
    for each_all_shortest_path in all_shortest_paths_list_in:
            print(each_all_shortest_path)
            node_disjoint_sp = [x for x in each_all_shortest_path if x not in
initial_shortest_path_in]
            print("node_disjoint_sp:", node_disjoint_sp)
            if len(node_disjoint_sp)>max_diff:
                max_diff = len(node_disjoint_sp)
                candidate_sp = each_all_shortest_path
    return candidate_sp
def get_flow_direction(flow_info_in,dest_ip_in,exceed_port_in,direction_in):
     for each_flow in flow_info_in["table"]["flow"]:
        if each_flow["match"]  and "ipv4-destination" in each_flow["match"]:
         if each_flow["match"]["ipv4-destination"] == dest_ip_in:
          if direction_in == "recevied" and "in-port" in each_flow["match"]:
               if each_flow["match"]["in-port"] == exceed_port_in:
                  return True
          else:
           if "instructions" in each_flow:
              instruction = each_flow["instructions"]["instruction"]
              if "apply-actions" in instruction:
                 action_list = instruction["apply-actions"]["action"]
              else:
                 action_list = instruction["action"]
              if isinstance(action_list, list):
                 for x in action_list:
                    print(x["output-action"]["output-node-connector"])
                    if x["output-action"]["output-node-connector"] == exceed_port_in:
                        return True
              else:
                 if action_list["output-action"]["output-node-connector"] ==
exceed_port_in:
                    return
```

KİŞİSEL YAYIN VE ESERLER

Çubukçu, A., **Çubukçu, Ö**., Kavak, A. ve Küçük, K. (2024). Evaluation of Cloud Based Dynamic Network Scaling and Slicing for Next-Generation Wireless Networks. *MDPI Engineering Proceedings*, *70*(1), 45. 12 Ağustos 2024. https://doi.org/10.3390/engproc2024070045.

Çubukçu, A., **Çubukçu, Ö.**, Kavak, A. ve Küçük, K. (2024). Performance Evaluation of 5G RAN Slicing in terms of Physical Resource Usage. *SIU 2024*, 16-18 Mayıs 2024.

Çubukçu, A., **Çubukçu, Ö.**, Sabucu Sandal, Y., Kavak, A. ve Küçük, K. (2023). A Study on Slice-Aware Industrial Edge Applications for Next-Generation Private Networks. *IISEC 2024*, 21-22 Aralık 2023.

Çubukçu, Ö., Çubukçu, A., Kavak, A. ve Küçük, K. (2024). SDN and NFV Based Dynamic QoS Management for 5G and Beyond Networks. *SIU 2024*, 16-18 Mayıs 2024.

ÖZGEÇMİŞ

2015 yılında Kocaeli Üniversitesi Elektronik ve Haberleşme Mühendisliği bölümünden mezun olmuştur. 2015-2018 yılları arasında Huawei' de Kablosuz Haberleşme Mühendisi olarak çalışmıştır. 2018 yılından itibaren Ulak Haberleşme'de 5G Sistem ve Test Mühendisi olarak çalışmaktadır. İlgi alanları Düşük Gecikmeli Haberleşme, Servis Kalitesi, Ağ Dilimleme, Uç Bilişim, Çekirdek Şebekedir.