Bilgisayar Bilimleri ve Mühendisliği Dergisi

Cilt-17
Sayı-2
2024

Hava Trafik Yönetim Sistemleri Çekirdek Ağlarında Servis Kalitesi Temelli Ağ Trafik Yönetimi (Air-SDN)

Quality-of-Service Based Network Traffic Management in Air Traffic Management System's Core Network (Air-SDN)

Sertan PEKEL
Ege Üniversitesi
Uluslararası Bilgisayar Enstitüsü
İzmir - Türkiye
sertanpekel@gmail.com
ORCID: 0000-0003-0994-3307

Müge SAYIT
Ege Üniversitesi
Uluslararası Bilgisayar Enstitüsü
İzmir - Türkiye
muge.sayit@ege.edu.tr
ORCID: 0000-0002-7990-1463

Öz

Hava Trafik Yönetim (ATM) sistemleri, ticari hava taşımacılığının yanı sıra hava sahasında gerçekleşen tüm uçuşların yönetimini sağlar. Bu sistemin gerçek zamanlı, sürekli ve kritik veri barındırması nedeniyle, sistemin çekirdek ağında gerçekleşen ağ trafiğinin bu parametrelere cevap verebilecek şekilde yönetilmesi, başka bir ifade ile bu ağ yapısında belirli bir servis kalitesi (QoS) yaklaşımının uygulanması gerekmektedir. Bu çalışmada önerilen Air-SDN yöntemi, ATM sistemlerindeki çekirdek ağlarda Yazılım Tanımlı Ağ (SDN) konsepti çerçevesi içerisinde belirli bir servis kalitesi (QoS) yöntemi uygulayarak ağ trafiğinin hızlı ve güvenli yönetilmesine olanak tanımaktadır. Air-SDN yöntemi bir test ortamında benzetilmiş ve önerilen yöntemin, TCP taşıma protokolü ile %153, UDP ve Burst-UDP taşıma protokolleri için de %63 oranında veri çıkış hacmi parametresinde performans artışı sağladığı gözlenmiştir.

Anahtar sözcükler: Hava Trafik Yönetimi, ATM Sistemlerinde Çekirdek Ağ Trafik Yönetimi, Yazılım Tanımlı Ağlar (SDN), Servis Kalitesi (QoS)

Abstract

Air Traffic Management (ATM) Systems provide managing all flights in the airspace including commercial air transporting. Because of this system contains real-time, consistent and critical data, the network traffic which occurs in the core network of this system should be managed to satisfy these parameters; in other terms, a quality of service (QoS) approach shou

ld be applied to this network structure. Air-SDN method, which is proposed in this paper makes enable to manage network traffic fast and reliable within ATM System's core network using a specified quality of service (QoS) manner in Software Defined Networking (SDN) concept. Air-SDN method is emulated in a test-bed and it has been observed that this method provides performance gain as 153% for TCP transport protocol, 63% for UDP and Burst-UDP transport protocols.

Keywords: Air Traffic Management, Traffic Management in ATM System's Core Network, Software Defined Networks (SDN), Quality of Service (QoS)

1. Giriş

Günümüzde milyonlarca yolcunun zaman tasarrufu ve güvenlik sebepleri ile havayolu ulaşımını tercih etmesi, ülkelerin toprakları üzerinde yer alan hava sahalarının da aynı kara ve deniz yolları gibi yoğun bir şekilde kullanılmasına yol açmaktadır [1]. Bununla birlikte kara ve deniz ulaşımında kullanılan taşıtların görece düşük hızları, bu taşıtları idare eden kişilerin (şoför, kaptan vb.) diğer taşıtlar ile aralarındaki güvenli mesafeyi kendilerinin ayarlayabileceği seviyededir. Ancak hava sahasını kullanan hava araçlarına ait hızların diğer ulaşım araçlarına nazaran daha yüksek oluşu, hava sahasının spesifik gözetleme araçları ile gözetlenerek trafiğinin aktif olarak yönetilmesi gerekliliğini ortaya koymaktadır.

Hava sahasında gerçekleşen trafik akışı, sivil yolcu taşımacılığı ile birlikte askeri ve sivil amaçlı eğitim, taktik, savunma,

Gönderme, düzeltme ve kabul tarihi: 18.06.2023 - 21.11.2023 – 20.03.2024
Makale türü: Araştırma

önleme, istihbarat, müdahale, intikal, iş, sportif, zirai, tıbbi, güvenlik ve turizm amaçlı yapılan tüm uçuşları da içine alır. Bu uçuşların güvenli, düzenli ve acil bir durumda müdahale edilebilinecek şekilde gerçekleşmesi, Hava Trafik Yönetimi (*Air Traffic Management* – ATM) ile mümkün olur. ATM, devletler bazında, kamu hizmeti olarak yürütülür. Ülkemizde bu sorumluluk, ulusal hava ulaşım hizmet sağlayıcısı (*Air Navigation Service Provider* – ANSP) misyonu ile Ulaştırma ve Altyapı Bakanlığı'nın ilgili kuruluşu olan Devlet Hava Meydanları İşletmesi (DHMİ) Genel Müdürlüğü tarafından üstlenilmiştir.

ATM kullanıcıları (hava sahasının yönetiminden sorumlu hava trafik kontrolörleri, uçuş planlamalarından sorumlu havacılık bilgi yönetimi uzmanları, kullanılan teknik altyapıdan sorumlu kişiler ve uçuş düzenleyen şahıs, kurum ve kuruluşlar vb.), coğrafi anlamda farklı konumlarda olmaları sebebiyle fiziksel ve mantıksal olarak birbirine bağlı bir bilgisayar ağ sistemi (ATM sistemi) ile iletişim kurarlar. Günümüzde pek çok ATM sistemi geleneksel ağ yönlendirme yapısı baz alınarak tasarlanmıştır. Bu sistemlerde ağ yönlendirme yapısını ve ağ mimarisini etkileyen köklü değişikliklere gitmek, taşınan verinin gerçek zamanlı, sürekli ve kritik yapısı nedeniyle ATM ağını kuran ve işleten devletler açısından çok tercih edilmemektedir. Böyle bir değişikliğin sistemleri tamamen devre dışı bırakarak çekirdek ağdan uç birimlere, duyargalara ve antenlere doğru yapılması gerekmektedir. ATM sisteminin -anlık da olsa- devre dışı kalması hava sahasının ve uçuşların güvenliğini doğrudan etkilediğinden, sistemin devre dışı kaldığı zaman aralığında yedek bir ATM sisteminin kullanılması elzemdir. Bu sebeple dünya üzerinde kullanılan hemen hemen tüm ATM sistemleri geleneksel ağ yapısı üzerine tasarlanmıştır ve bu sistemlerde ağ yönlendirme yapısı ya da ağ mimarisinde (bu çalışmada önerilen yöntem gibi) köklü bir değişiklik yapmak, aktif olarak çalışacak yedek bir ATM sistemi gerektirmektedir.

Yazılım Tanımlı Ağlar (*Software Defined Networks* – SDN), bilgisayar ağlarındaki kontrol ve veri düzlemlerini birbirinden ayırarak, kontrol düzleminin esnek ve ihtiyaca özgü tasarlanabilmesini sağlayan bir teknolojidir. Sahip olduğu avantajlar sebebiyle, ATM sistemlerinde kullanımı bu sistemlerin ihtiyacı olan gerçek zamanlı, kritik ve sürekli iletişimi mümkün kılabilir.

Mevcut ATM sistemlerinde servis kalitesi (*Qualiy of service* – QoS), genellikle veriyi taşıyan kuruluşa (çoğunlukla ulusal iletişim kuruluşu – *national carrier*) bırakılmış, bu kuruluşlar ile imzalanan hizmet seviyesi anlaşması (*Service Level Agreement* – SLA) ile belirli bir seviyede sağlanmaya çalışılmıştır. Böylelikle ATM sistemlerinde QoS sağlama genellikle uç ağda hava ulaşımı hizmet sağlayıcısının kontrolünde olmakta, çekirdek ağda ise veriyi taşıyan kuruluşa bağımlılık sebebi ile genellikle bir QoS sağlama çabası güdülmemektedir. Bu sebeple bu çalışmanın motivasyonu, SDN konseptinin 'programlanabilir' yetisini ve OpenFlow protokolünün güçlü akış kontrolü kabiliyetini çeşitli parametreler üzerinden kullanarak bir ATM sistemi çekirdek ağında QoS sağlama olarak açıklanabilir.

ATM Sistemlerinde SDN ve NFV (*Network Functions Virtulalization* – Ağ Fonksiyonları Sanallaştırma) kavramları, ilk kez 2019 yılında Brezilya'nın hava ulaşım hizmet sağlayıcı (ANSP) kuruluşu olan (*Department of Airspace Control* – DECEA) tarafından ATM sistemlerine entegre edilmiş ve daha önce dengesiz seviyedeki servis kalitesinin arttığı ve stabil hale geldiği kuruluş tarafından açıklanmıştır [2].

Bu çalışmada bir ATM sistemi çekirdek ağında, geleneksel ağ yönlendirme yapısı yerine yazılım tanımlı ağ (SDN) konsepti uygulanarak çekirdek ağdaki veri trafiğinin uyarlamalı bir servis kalitesi temelli yönetilmesi ele alınacaktır. Önerilen yöntem, sanal bir ağ ortamında benzetimlenecektir. Bildiğimiz kadarıyla bu çalışma, ATM sistemleri için SDN teknolojisini kullanarak QoS sağlama amaçlı ilk akademik çalışmadır. Çalışmanın katkıları şu şekilde listelenebilir:

- SDN ve NFV teknolojilerini kullanarak ATM ağlarının karakteristiklerine uygun bir QoS çözümü geliştirilmiş ve yapılan deneyler ile bu çözümün uygulanabilirliği gösterilmiştir.
- ATM ağlarında QoS sağlamak amacıyla geliştirilen yaklaşım, her bir akış için dinamik kuyruk yönetimini sağlamaktadır. Bu özelliği ile literatürden farklılaşarak, mevcut çalışmalarda bulunan sınıf bazlı kuyruk atama yaklaşımından daha detaylı ve gerçek zamanlı bir yaklaşım sunmaktadır.
- Gerçekleştirilen deneylerde, önerilen yaklaşımın performansı TCP, UDP ve Burst-UDP trafiği ile ölçümlenerek, farklı iletim katmanlarına ait performansı irdelenmiştir.

Çalışmanın ilerleyen bölümlerinde teorik arka plan ve ilgili çalışmaların tanıtılması, Air-SDN yönteminin açıklanması, simülasyon ve performans detaylarının belirtilmesi ile sonuçların değerlendirilmesi ve tartışma kısımları yer almaktadır.

2. Teorik Arka Plan ve İlgili Çalışmalar

2.1 Hava Trafik Yönetimi (ATM) Sistemleri

Hava Trafik Yönetimi (ATM), hava sahası ve kullanıcılarının tamamına etkin hizmet verecek şekilde yapılandırılmış bir ATM sistemi ile donatılmış olmalıdır. Bir ATM sistemi, hizmet verdiği hava sahasındaki iletişim, ulaşım ve gözetleme (*Communication, Navigation, Surveillance* – CNS) hizmetlerini ve bu hizmetlerin arasındaki ilişkileri kapsar ve yönetir.

ATM sistemlerinin birden fazla hizmet ve altyapıyı kapsaması ve yönetebilmesi ise üretilen gerçek zamanlı verinin vakit geçirilmeden ATM çekirdek ağındaki sunucularda işlenmesi, kaydedilmesi ve işlenen verinin gecikme olmaksızın muhataplarına aktarılabilmesi ile gerçekleşir. Bunun için de sağlam, sürekli, hızlı ve bütüncül veri akışı sağlayabilecek bir bilgisayar ağının, hava sahası kullanıcıları arasında tesis edilmiş olması gerekmektedir. Hava trafiğinin emniyetli ve hızlı yönetilebilmesi, uçuşların başladığı ve bittiği havaalanları (*aerodromes*), hava sahasının yönetildiği hava ulaşım üniteleri (kule, yaklaşma, saha kontrol birimleri ve uçuş bilgi merkezleri [*flight information center* – *FIC*]) ile uçuşların gerçekleştiği

hava yolu ve tanımlanmış hava sahalarındaki (*designated airspace*) CNS hizmetlerinin aksamadan sağlanması ve bu birimlerin birbirleri ile anlık ve doğru koordine kurabilmeleri ile mümkün olmaktadır. Bu da tüm uç birimlerden (havalimanları, bağımsız oluşturulan hava ulaşım ve komşu ATM üniteleri vb.), anten ve duyargalardan (radar ve meteorolojik gözetim aygıtları) ve hava araçları ile iki yönlü telsiz iletişimi için kullanılan hava-yer frekansı antenlerinden alınan uçuş bilgi (*Flight Information Data* – FID), uçuş gözetleme (*Flight Surveillance Data* – FSD), hava sahası yönetim mesajı (*Airspace Management Message* – ASM), frekans ve telefon konuşmaları (*Voice-over-IP* – VoIP) ile hava ulaşım merkezlerinde kullanılan gözetleme ekran görüntüsü verisinin gerçek zamanlı olarak çekirdek ağa iletilmesi ve çekirdek ağda işlenen ve kaydedilen verinin yine gerçek zamanlı olarak bu veriyi kullanacak uç birimlere, komşu ATM ünitelerine ve hava araçlarına gönderilmesi ile mümkün olmaktadır.

Bir ATM sisteminin çekirdek ağı, kendisine ulaşan her tür veriyi işleyebilecek, kaydedebilecek ve gerektiğinde işlenen veriyi vakit geçirmeden muhataplarına gönderebilecek komponentleri barındırmalıdır. Frekans verisi (hava araçları ve hava ulaşım ünitelerinin iki yönlü telsiz iletişimi) ve hava ulaşım ünitelerinin kendi aralarındaki koordineler için kullandıkları telefon haberleşmesi verisini işleyen ve kaydeden bir VoIP sistemi, gözetleme duyargalarından alınan ham veriyi işleyen ve hava aracının konumunun dört boyutlu düzlemde izlenmesini sağlayan bir gözetleme verisi işleme (*Surveillance Data Processing* – SDP) sistemi, uçuşla ilgili her türlü bilginin kayıt ve kontrol edilebilmesini sağlayan bir uçuş bilgi işleme (*Flight Data Processing* – FDP) sistemi, hava sahası ile ilgili tüm mesaj ve parametrelerin kayıt ve kontrol edilebilmesini sağlayan hava sahası yönetim (*Airspace Management* – ASM) sistemi, tüm hava ulaşım ünitelerinde takip edilen ve girdi yapılan anlık hava resminin akış videosu halindeki medyasını kaydedip arşivleyecek bir kayıt ve çalma (*Recording and Playback* – RNP) sistemi ile bu sistemleri birbirlerine ve uç birimlere bağlayacak ağ yönlendirme elemanları bu bileşenlere örnek verilebilir.

2.2 SDN ve QoS

SDN, klasik ağ sistemlerinde ağ trafiğini yönlendiren cihazlarda birlikte bulunan yönlendirme karar mekanizmasını yöneten kontrol düzlemi ve yönlendirme işlemini gerçekleştiren veri düzleminin birbirinden ayrılması ve kontrol düzleminin tek elden yazılımsal olarak yönetilebilmesini sağlayan yeni bir ağ konseptidir. Bu sayede klasik ağ yapısında üretici bağımlı, düşük seviyede ve esnek olmayan bir şekilde yönetilebilen yönlendirme cihazları basit birer yönlendirme cihazı haline gelmekte, yönlendirme karar mekanizması ise yazılımsal fonksiyon yürütebilme yetisi kazanmış şekilde tek cihazda (genellikle bir SDN denetleyicisi (*SDN Controller*)) toplanabilmektedir.

Kontrol düzleminin veri düzleminden ayrılması ile birlikte ağ zekası ve ağ durumu mantıksal olarak merkezi hale gelmekte, ayrıca ağ kullanıcıları ve uygulamalar ağ donanımından soyutlanmaktadır [3]. Dolayısıyla SDN, ağ yöneticilerinin paket yönlendirmesini çeşitli uygulamaların ihtiyaçlarına göre donanımdan ve cihaz üreticisinden bağımsız olarak programlayabilmesi kolaylığı getirmekle birlikte, klasik ağ yönetiminde karşılaşılan zorlukları da maliyetsiz bir şekilde aşabilen bir paradigma olarak karşımıza çıkmaktadır.

SDN konseptinde kontrol düzlemi ve veri düzlemi, SDN teknolojisinin 'yazılımsal' ilkesini gerçekleştirmek için ortak bir protokolde buluşmak zorundadır. Bu sebeple bu düzlemlerin birbirleri ile iletişim kurduğu güney köprü arayüzü (*southbound interface* – SBI ya da *control-data plane interface* – CDPI) için ölçünlü protokoller bulunmaktadır. Bu protokollerin en bilinen ve en yaygın kullanılanı OpenFlow'dur [4].

İnternet başta olmak üzere pek çok klasik ağ sistemi, en-iyi-çaba modelini temel alarak çalışmaktadır. En iyi çaba modeli, bir ağın kullanıcılarına sunabileceği ağ servisleri (bant genişliği, paket teslim garantisi, sıralı paket iletimi, zamanlı iletim vb. için herhangi bir garanti vermeden, o anki durumuna göre yapabileceğinin en iyisini yapmayı taahhüt etmesi olarak tanımlanabilir. Ancak, kullanıcıların ağdan beklediği performans düzeyi bu şekilde karşılanamayabilir. Bu sebeple bir ağın sunabileceği genel servisler için servis kalitesi (QoS) yaklaşımı benimsenmiştir. QoS, ağ kullanıcıların tükettikleri veri bazında, ilgili ağdan performans beklenti düzeyleri olarak tanımlanabilir.

Bilgisayar ağları için QoS, ağ yöneticisinin bir ağda taşınan verinin türüne, sınıfına, büyüklüğüne, kaynak ve varış noktalarına ya da kullanıcı isterlerine göre belirlediği veri iletim önceliklendirmesi olarak tanımlanabilir. Klasik ağ sistemlerinde QoS, yönlendirme cihazlarının yönetiminin esnek olmaması sebebi ile kısıtlı düzeyde uygulanabilmiştir. SDN konsepti ile birlikte yönlendirme cihazlarının ve dolayısı ile trafik akışının yazılımsal düzeyde esnek olarak yönetilebilmesi, SDN yetisine haiz olan ağlarda QoS uygulanabilmesini oldukça kolay ve derin keşiflere açık bir hale getirmiştir.

Klasik ağlarda ve özellikle İnternet'te zaman içerisinde QoS yaklaşımı sağlayan algoritmalar geliştirilmiş ve benimsenmiştir. Bunlardan en önemlileri IntServ (*Integrated Service*) ve DiffServ (*Differentiated Service*)'dir [5,6]. Ancak IntServ, çok karmaşık ve ölçeklenemeyen yapısı nedeniyle, DiffServ ise kaynak rezervasyonu yönetiminde yetersiz kalmasından dolayı stabil bir QoS yaklaşımı sunamadığından günümüz talepleri açısından yetersiz kalmaktadır [7,8].

2.3 SDN Teknolojisi Kullanılarak Gerçekleştirilen QoS Çalışmaları

SDN teknolojisinin getirdiği esneklik ve merkezilik imkanları ile birlikte QoS sağlanması konusunda kaynaklarda pek çok önemli çalışma sunulmuştur. Bu çalışmalarda, QoS sağlamak için kimi zaman uçtan uca yolların belirlenmesi, kimi zaman kuyruk yönetimi, kimi zaman da hem yol belirleme hem de kuyruk yönetimi yaklaşımları öne sürülmüştür.

[9] çalışmasında veri merkezi ağları için yük dengeleme ve servis kalitesini (QoS) gerçeklemeyi amaçlayan algoritma sunulmuştur ve algoritma, talep edilen bant genişliğini göz önüne alarak 'en iyi' iletim yolunu hesaplamaktadır. Gerçek zamanlı ve uçtan uca servisler için SDN ağında bir sistem

prototipi dizayn edilen [10] çalışmasında yazarlar, en iyi bir çevrimiçi yönlendirme algoritması önermişlerdir. [11] çalışmasında SDN ağlarında QoS gerçeklemesini sağlamak için, şayet talep karşılanacak ise ağ yolu talebi, talep eden uygulamanın türüne göre belirlenmektedir. [12] çalışmasında ise SDN konseptini temel alan bir Nesnelerin İnterneti (IoT) mimarisi üzerinde gri delik atakları ve Dağıtık Servis Reddi (DDoS) saldırılarını saptayan bir yöntem önerilmiştir. Bu çalışmaların tamamında SDN ile QoS sağlamak için yol belirleme yaklaşımlarından yararlanılmış, kuyruklar üzerinde bir çalışma yapılmamıştır.

Vanitchasatit ve Sanguankotchakorn tarafından yapılan çalışmada, SDN kavramını temel alan bir kablosuz yerel alan ağındaki (WLAN) trafiği VoIP, video akışı ve dosya transferi (*file transfer*) olarak sınıflandırmıştır [13]. Daha sonra sınıflanan her bir veri türü için hedeflenen QoS parametreleri belirlenerek her bir veri sınıfı türünde maksimum akış trafiğine izin verecek şekilde tam sayılı doğrusal programlama (*Integer Linear Programming* – ILP) optimizasyon tekniği uygulanmıştır. Elde edilen sonuçlara dayalı olarak her bir trafik türü için gerekli olan minimum bant genişliği belirlenmiştir.

Al-Haddad ve Ark., ağırlıklandırılmış adil kuyruklama (*Weighted Fair Queue* – WFQ) tekniğini kullanarak SDN tabanlı veri kesitlerini taşıyan ağlar için yeni bir trafik şekillendirme algoritması önermişlerdir [14]. Algoritma, kaynak cihazlardan gelen veriyi DiffServ protokolü ile sınıflandırarak ilk katman anahtarlayıcılarına göndermekte ve bu anahtarlayıcılarda bir paket ayrıştırma ve yönlendirme algoritması uygulayarak veri kesitinin türüne göre veriyi ikinci katman anahtarlayıcılarına göndermektedir. İkinci katman anahtarlayıcılardan üçüncü katman anahtarlayıcılarına gönderilen veri, üçüncü katman anahtarlayıcılarıda WFQ mantığını temel alan bir kuyruklandırmaya sokulmakta ve ağırlık oranına göre belirlenmiş bant genişliği atanarak alıcı cihazlara gönderilmektedir.

Li ve Ark., çalışmalarında ağdaki QoS yönetimini sağlayabilmek için SDN denetleyicisine C4.5 karar ağacı (*C4.5 karar ağacı*) uygulamışlardır [15]. Bu karar ağacının budanması ile uygulama tipleri belirlenerek her bir uygulama tipine ait QoS seviyesi hızlandırılmış yönlendirme, teslim-garantili yönlendirme ve en iyi çaba atanmaktadır. Daha sonra paket akışları ilgili anahtarlayıcı iskelesi için kuyruklandırılarak bu kuyruklara QoS seviyesine göre öncelik atanmaktadır.

Baek ve Ark., SDN ağlarında bant genişliği atanan uygulamaların güvenilir bir servis kalitesini yakalayabileceğini gösteren bir çalışma sunmuşlardır [16]. Çalışmada Mininet, Ryu Controller ve Open vSwitch yapısında oluşturulan bir topolojide belirlenmiş dört öncelik seviyesindeki uygulamalara hiyerarşik sınıflandırılmış jeton bazlı (*Hierarchical Token Based* – HTB) kuyruklama tekniği uygulayarak, uygulamaların atanmış bant genişliğini gerçekten alıp almadıklarını ölçmüşlerdir.

Krishna ve Ark., çalışmalarında QoS ve *best-effort* olarak iki sınıfa ayırdıkları trafik akışları için atanan bant genişliğini aşan bir veri çıkış hacmi gerçekleştiğinde servis kalitesinin (QoS) nasıl etkilendiğini araştırmışlardır [7]. Bunun için yaptıkları ilk denemede OpenFlow protokolünün sayaç işlevini kullanarak rezerve edilen bant genişliğini aşan hacimdeki akış paketlerini düşürme yöntemi ile servis kalitesinin (QoS) korunmasını sağlamışlardır. Yazarlar ikinci denemelerinde ise bant genişliği ödünç alma yöntemini uygulayarak daha önce atanan bant genişliğini aşan akışlara ait paket kayıp oranını ölçmüşlerdir. Yazarlar ayrıca test ortamında kullandıkları sanal anahtarlayıcıların temel aldığı kuyruklama yöntemi olan Hiyerarşik Sınıflandırılmış Jeton Kovası'nın (*Hierarchical Token Bucket* – HTB) da QoS paketleri için paket kayıp oranını %0 bularak önerdikleri yöntemin üretim ağlarında kullanılabileceğini belirtmişlerdir.

Thazin ve Ark., 3 farklı servis sınıfı oluşturarak akışların bağlı olduğu sınıfa göre kuyruklar atamış, bant genişliğinin yeterli gelmemesi durumunda yolun değiştirilmesini önermişlerdir [3].

Akella ve Xiong, bulut bilişim kullanıcıları için farklı düzeydeki servis kalitesi (QoS) ihtiyacına, uçtan uca bant genişliği garanti edecek bir çalışma gerçekleştirmişlerdir [17]. Bu çalışmada da QoS sağlamak için yol seçimi ön plana çıkmaktadır. Kuyruklama için Open vSwitch tarafından desteklenen HTB ve Sınıflandırılmış Adil Sıralı Eğri (*Hierarchical Fair Sequence Curve* – HFSC) kuyruklama teknikleri kullanılmış olup, akış başına kuyruk belirleme üzerinde bir çalışma gerçekleştirilmemiştir.

[3], [8], [9], [11] çalışmaları en iyi yolu bulma üzerine yoğunlaşmıştır. Bu çalışmada temel olarak ele alınan ağ sistemlerinin (ATM ağları) kapalı sistemler olması ve ağ yollarının statik olması nedeniyle, en iyi yolu bulma metodunun bu çalışmanın problemi için bir çözüm getirmediği saptanmıştır.

[8] çalışması statik kuyruk yapısı üzerine bir yöntem önermiştir. Air-SDN yönteminde statik kuyruk yapısı değil, veri tipleri için adaptif kuyruk yapısı uygulanmıştır. [12] çalışması ise ağ güvenliği üzerine yoğunlaştığı ve Air-SDN yönteminin uygulandığı ağ ortamının kapalı bir ağ olmasından dolayı bu çalışmanın çözümüne uymamaktadır.

Literatürde SDN ve QoS ile ilgili burada bahsedilmeyen daha pek çok çalışma bulunmaktadır. Bu çalışmaların güzel bir özeti [18] çalışmasında sunulmuştur.

Buna rağmen ATM ağlarında QoS sağlanması ile ilgili literatürde kayda değer nadir çalışmalar bulunmaktadır. Bunlardan birisi, Ongkasae ve Ark. uçtan uca QoS seviyesini arttırma adına ağ sanallaştırma tekniği kullanılan SWIM (*System-Wide Information Management* - Sistem Seviyesinde Bilgi Yönetimi) modeli önerdikleri bir çalışmadır [19]. Bu çalışmada SDN ya da NFV teknolojileri kullanılmamış olup, geleneksel ağ altyapısının getirdiği kısıtlar sebebiyle akış değil sınıf bazlı bir QoS modeli önermişlerdir.

Diğer bir çalışma ise Izquierdo-Zaragoza ve Ark. bir çalışmasıdır [20]. Bu çalışmada ATM ağlarının geniş ağ yapısının (WAN - *Wide Area Network*) yazılım tanımlı ve hiyerarşik olarak nasıl tasarlanabileceği ve farklı QoS profillerinin bunlara nasıl bir temel sağlayabileceği tartışılmıştır. Çalışma daha çok tasarım üzerine olduğundan ve

trafik yönetimi veri bazında ele alınmadığından bu çalışma da önerilen yöntemin problemi için bir çözüm getirmemektedir

Sonuç olarak, mevcut literatürde QoS sağlamak için önerilen yol seçim yaklaşımları ATM ağları için anlamlı olmamaktadır. Akışlara kuyruk atama yaklaşımlarında ise akış bazlı değil, sınıf bazlı atamalar yapılmış, akışlar için dinamik olarak kuyruk değişimi göz önünde bulundurulmamıştır. Bu çalışmada önerilen yaklaşımın literatürdeki çalışmalardan farkı, akış bazında kuyruk atamanın yapılması ve önceliğe göre dinamik olarak bu kuyrukların değiştirilmesinin sağlanmasıdır.

3. Air-SDN

3.1 ATM Sistemindeki Veri Tipleri

Bir önceki başlıkta da belirtildiği üzere ATM sistemlerinde üretilen ve işlenen her türlü veri 'en iyi çaba' yaklaşımı ile taşınamayacak kadar kritik, gerçek zamanlı ve sürekli karaktere sahiptir. Bu sebeple belirli bir servis kalitesi (QoS) sağlama adına önerilen yöntem, bu verinin iletilmesi esnasında yaşanabilecek paket kayıplarına ve gecikmelere karşı duyarlı, ayrıca iletilen verinin türüne göre belirli bir veri çıkış hacmi garanti edebilen bir yöntem olmalıdır. SDN konsepti ve OpenFlow protokolü çerçevesinde uyarlamalı kuyruklama ve yük dengeleme paradigmalarını ilke alan ve bu çalışmada önerilen yönteme 'Air-SDN' adı verilmiştir.

Air-SDN yönteminde öncelikle bir ATM çekirdek ağında taşınan ve işlenen veri tipleri tespit edilmiştir. Daha önce açıklandığı üzere, bir ATM ağında genel olarak beş çeşit veri tipi bulunabilir: VoIP verisi, uçuş gözetim verisi (FSD), uçuş bilgi verisi (FID), hava sahası yönetimi verisi (ASM) ve video akış verisi (*Video Stream* - VS). Bu veri tipleri, simülasyonlar için tespit edilen ortalama bant genişliği ve minimum bant genişliği ihtiyacı ve trafik yönetimindeki öncelikleri Çizelge-1'de verilmiştir.

Çizelge-1: Veri Tipleri ve Önceliklendirilmeleri

Sıra	Veri Tipi			
	Tanımı	Ort. BW	Min. BW	Önceliği
1	VoIP	25 Mbit/s	10 Mbit/s	En Yüksek
2	FSD	20 Mbit/s	10 Mbit/s	Yüksek
3	FID	15 Mbit/s	5 Mbit/s	Orta
4	ASM	5 Mbit/s	2 Mbit/s	Düşük
5	VS	40 Mbit/s	10 Mbit/s	Yok

Tespit edilen veri tiplerinde en öncelikli veri tipi frekans ve telefon haberleşmesi (VoIP) olarak belirlenmiştir. Bunun sebebi hava trafik yönetiminin (ATM) en önemli parçası olan hava araçlarına verilen kontrol hizmetinin (*Air Traffic Control Service* – ATCS) hava-yer frekansları üzerinden gerçekleşmesi ve bunlarla ilgili tüm kritik koordinelerin hava ulaşım merkezleri arasında telefon üzerinden yapılmasıdır. Bu önceliğin verilmesinde hava trafik hizmetinin herhangi bir ATM sistemi olmasa dahi sadece hava-yer muhaberesi ile sunulabileceği kuralı da göz önüne alınmıştır. İkinci öncelikli veri tipi ise uçuş gözetim verisidir (FSD). Radar antenleri ve sensörlerden alınan ham gözetim verisi, hava sahasının ve hava araçlarının dört boyutlu takibi için azami düzeyde önem arz etmektedir. Üçüncü ve dördüncü öneme haiz veri tipleri uçuş bilgi (FID) ve hava sahası yönetim mesajlarıdır (ASM). Hava ulaşım ünitelerinde takip edilen ve üzerinde çeşitli girdiler yapılabilen anlık hava resminin çekirdek ağa aktarımı için kullanılan video akış (VS) verisine ise öncelik verilmemiştir. Bu veri tipine öncelik verilmemesinin sebebi bu verinin çekirdek ağda sadece kaydedilmesi, aynı verinin eş zamanlı olarak hizmeti sunan hava ulaşım ünitesinde de saklanmasıdır.

3.2 Yöntem Detayları

Air-SDN yönteminde kullanılan uyarlamalı kuyruklama ve yük dengeleme ilkesi, ilgili veri tipine ait akış ağ üzerinde bulunduğunda, ihtiyaç duyacağı bant genişliğine ve orta anahtarlayıcıların yük durumlarına göre belirlenmiş kuyrukları adaptif olarak kullanma üzerine kurulmuştur. Sistemde video akış verisi (VS) dışında kalan veri tipleri, bu çalışmada 'öncelikli veri tipleri' olarak adlandırılmıştır. Ağda bulunan akışlar, önerilen yöntemin uygulanması esnasında Çizelge-1'deki öncelik sırasına göre öncelik almaktadır.

Önerilen yönteme göre, başlangıçta, teorik olarak en yüksek bant genişliğine ihtiyaç duyacağı öngörülen video akış (VS) veri tipi en yüksek bant genişliğine sahip kuyruklara atanmaktadır. Ancak, dinamik kuyruk yapısı ile, yüksek bant genişliği sağlayabilen kuyruklar, ihtiyaç olması halinde öncelikli veri tiplerine tahsis edilebilmektedir. Bu ihtiyaç ortadan kalktığında video akış verisi (VS) yine en yüksek bant genişliği sunan kuyruklara atanır.

Önerilen Air-SDN yönteminin motivasyonu, paket iletiminde veri tiplerinin önceliklendirme sırasına göre ele alınarak ortalama bant genişliği ihtiyacına uygun bir kuyruğa atanması ve orta anahtarlayıcılarda yükün dengelenmesidir. Bunun için sunucularda oluşturulan veriye ait trafik akış istatistiklerinin, kullanılan SDN denetleyicisi tarafından okunarak ilgili veri tipi için gerekli bant genişliğinin belirlenmesi, bu değerin o veri tipi için belirlenmiş olan minimum bant genişliği değerinin altına düşmeyecek şekilde ilgili trafik akış yönlendirmesinin orta anahtarlayıcılar ile ana fonksiyon anahtarlayıcı (*core function switch*) arasındaki linke bağlı çıkış portuna tanımlanmış kuyruklara yönlendirilmesinin sağlanması, bu işlem yapılırken aynı zamanda yük dengelemesi yapılabilmesi için aynı veri tipine ait akışların farklı orta anahtarlayıcılara yönlendirilmesi hedeflenmiştir.

3.3 Air-SDN Algoritması

Air-SDN mimarisinde kullanılan algoritma Algoritma-1'de verilmiştir. Algoritma, SDN denetleyicisi tarafından periyodik olarak çalıştırılmaktadır. Algoritmanın çalıştırılması için bir başka yöntem, herhangi bir akış başladığında çalıştırılması olabilir. Ancak bu yöntem, akış başlangıcının tespiti için, gelen paketin SDN denetleyicisine gönderilmesini (*packet-in*) gerektirmektedir. Bu da gerçek zamanlılığı ve arzu edilen performansı olumsuz yönde etkileyen ve önerilen yöntem için gerek duyulmayan bir işlem olduğundan, algoritmanın herhangi bir akış başladığında değil, belirli periyotlar ile çalıştırılması tercih edilmiştir.

Algoritma-1: Air-SDN Algoritması

```
1:  DT ← Veri tipi listesi
2:  F ← Akış (flow) listesi
3:  FN ← Akış (flow) listesi (F'ten türetilmiş, kuyruk seçilmemiş, akış önceliğine göre sıralanmış)
4:  Q ← Kuyruk (queue) listesi (orta switch'ler için)
5:  QN ← Kuyruk (queue) listesi (Q'dan türetilmiş, herhangi bir akış atanmamış, kuyruk max_rate'e göre sıralanmış)
6:  ES ← Uç switch'lerin listesi
7:  PS ← VS veri tipi haricindeki (öncelikli) veri tiplerine ait uç switch'lerin listesi
8:  atanmış_queue_id ← Akışların kuyruklarına ait kuyruk_id'leri
9:  veri_tipi_dpid[] ← İlgili veri tipi akışlarına atanmış switch_id'leri
10: //veri akış istatistiklerinin okunması
11: for switch s in ES do
12:     f_f.demand ← FLOW_STATS(f_f), ∀ f ∈ F
13: end for
14: for data d in DT do
15:     for flow f in F do
16:         for switch s in PS do
17:             for queue q in Q do
18:                 //kuyruğa bir akış atanmışsa ve kuyruk switch_id'si, veri tipi akışlarının atandığı switch'ler listesinde ise döngüden çık
19:                 if f için atanmış_queue_id değeri varsa ve s_q.switch_id ∈ veri_tipi_dpid[] then
20:                     break
21:                 end if
22:                 //mevcut kuyruğun en yüksek bant genişliği parametresi akışa ait ihtiyacın üzerinde ya da eşitse ve bu değer akış için belirlenmiş en düşük bant genişliğinden yüksek ya da eşitse akışı kuyruğa ata
23:                 if s_q.max_rate ≥ f.demand ve s_q.max_rate ≥ f.min_rate then
24:                     atanmış_queue_id ← s_q.id
25:                     f.queue_id ← s_q.id
26:                     //yük dengeleme için ilgili veri tipine ait diğer akışların atandığı switch_id'lerinin saklanması
27:                     veri_tipi_dpid[] ← s_q.id
28:                 end if
29:             end for
30:         end for
31:     end for
32: end for
33: //kuyruk seçilmemiş akışlar için döngüyü başlat
34: for flow f in FN do
35:     for queue q in QN do
36:         f.queue_id ← q.id
37:     end for
38: end for
39: wait (t saniye)
```

Algoritmanın genel akışı verilerin tanımlanması, veri akış istatistiklerinin uç anahtarlayıcı okunması, iç içe döngüler vasıtası ile her bir veri tipi, her bir akış, her bir uç anahtarlayıcı ve her bir kuyruk için değerlerin kontrol edilerek ilgili akışların ilgili kuyruklara atanması, herhangi bir akış atanmamış kuyruklara ait liste üzerinden akışlara kuyruk atanması ve belirlenmiş süre kadar (t) beklenerek algoritmanın tekrar çalıştırılması şeklindedir.

Bu sebeple, algoritmada öncelikle algoritmanın kullanacağı değer ve listeleri saklayacak diziler tanımlanmıştır. Algoritmanın 11. ve 12. satırlarında tüm uç anahtarlayıcılardan akış istatistikleri talep edilmekte ve akışlara ait parametre olarak kaydedilmektedir. 19. ve 20. satırlarda ilgili veri tipi için daha önce kuyruklara atanmış akışların anahtarlayıcı kimlikleri kontrol edilerek, şayet ilgili veri tipine ait uç anahtarlayıcılardan herhangi birine ait akışlara daha önce kuyruk ataması yapılmışsa son döngüden çıkılmakta ve o veri tipine ait akışı farklı bir orta anahtarlayıcılardaki (kuyrukları barındıran ve asıl yönlendirmenin yapıldığı anahtarlayıcılar) kuyruğa atanması sağlanmaktadır. Böylece algoritma aynı zamanda yük dengeleme fonksiyonunu da yerine getirmektedir. 23.-27. arasındaki satırlarda ise asıl akış atama işlemi yapılmakta ve algoritmanın kullandığı diğer değerlere ilişkin atamalar gerçekleştirilmektedir. Algoritmanın son kısmındaki 34.-36. satır aralarında ise daha önce herhangi bir kuyruğa atanmayan akışlar (öncelikli olmayan veri tiplerine ilişkin akışlar ile bir önceki kısımda kuyruğa atanmayan akışlar) kuyruğa atanmaktadır. En son olarak 39. satırda bu algoritma ile yapılan kuyruklama ve yük dengeleme konfigürasyonu ile *t* saniye boyunca devam edilebilmesi için (bir sonraki çerçeveye kadar) *t* saniyelik bekleme yapılmaktadır.

Algoritmanın zaman karmaşıklığı $O(d.s.q.f)$ şeklinde gösterilebilir. Burada *d* (veri türü), *s* (orta anahtarlayıcıların sayısı) *ve q* (kuyrukların sayısı) belirli sayıda sabit olup, bu çarpıma *c* dersek, karmaşıklık $O(c.f)$ olarak hesaplanabilir. *d*'nin sabit değeri 5; bir anahtarlayıcı için tanımlanabilecek kuyruk sayısı da sabit olduğundan, *c* değeri burada *s* değerine bağımlı olarak artacaktır. Bu sebeple algoritma karmaşıklığı *f* (akış sayısı) ve orta anahtarlayıcıların sayısı *s* ile sınırlıdır. Bu sebeple algoritmanın zaman karmaşıklığı doğrusaldır.

Algoritmanın bu yapısı ile yalnızca test ortamında değil, çok daha büyük ölçekteki ATM sistemi çekirdek ağlarında rahatlıkla kullanılabilecek, klasik ağ sistemlerindeki yönlendirme problemlerine çözüm olabilecek bir algoritma olduğu 'Sonuçların Değerlendirilmesi ve Tartışma' bölümünde ortaya konmuştur.

4. Benzetim ve Başarım

4.1 Benzetim Sınama Test Ortamı

4.1.1 Kullanılan Topoloji

Air-SDN yönteminin benzetilebileceği sınama ortamı için gerçek bir ATM sistemi çekirdek ağ topolojisi model alınarak bir topoloji oluşturulmuştur. Bu topolojide öncelikle ağa giren verinin üretildiği aygıtlar (*host*) ve onlara bağlı anahtarlayıcılar tanımlanmıştır. Her bir veri tipi için bir sunucu çifti

belirlenmiştir. Bu sunucular aynı zamanda havalimanları, hava ulaşım merkezleri, duyargalar ya da antenler olarak da düşünülebilir. Daha sonra asıl yönlendirme işlevini gerçekleştirecek orta anahtarlayıcılar tanımlanarak tüm uç anahtarlayıcılara *mesh* topoloji yöntemi ile bağlanmıştır. Ham verilerin işlenmesini ve kaydedilmesini sağlayan fonksiyonel sunucuların (VoIP server, SDP server, FDP server, ASM server ve RNP server) orta anahtarlayıcılar ile bağlantısını sağlayacak bir ana fonksiyon anahtarlayıcısı topolojiye eklenmiştir. Topolojide tüm linkler 100 MBit/s veri taşıyacak şekilde belirlenmiştir. İlgili topoloji Şekil-1'de verilmiştir.

Bu topolojide öncelikli veri tiplerini orta anahtarlayıcıları taşıyacak olan anahtarlayıcıları sayısı 8, orta anahtarlayıcıların sayısı da 3 olarak belirlenmiştir. Üretim ortamında bu sayılar ihtiyaca göre kullanıcılar tarafından tekrar düzenlenebilir. Algoritmanın bu konuda herhangi bir kısıtı bulunmamaktadır.

4.1.2 Belirlenmiş Kuyruklar ve Parametreleri

OpenFlow protokolünün kuyruk yönetimi işlevi olmadığından, bu işlem simülasyon ortamı için tercih edilen sanal anahtarlayıcılar vasıtası ile yapılmıştır. Bu amaçla her bir orta anahtarlayıcıda çıkış iskelesine tanımlanan kuyruklar ve parametreleri Çizelge-1'de belirlenen veri tipleri ve parametrelerine uygun olarak Çizelge-2'de verilmiştir.

Çizelge-2: Kuyruklar ve Atanmış Veri Tipleri

Sıra	Kuyruk ID	Min-Rate	Max-Rate	Atanan Veri Tipi
1	q0	10 Mbit/s	40 Mbit/s	VS
2	q1	10 Mbit/s	25 Mbit/s	VoIP
3	q2	10 Mbit/s	20 Mbit/s	FSD
4	q3	5 Mbit/s	15 Mbit/s	FID
5	q4	2 Mbit/s	5 Mbit/s	ASM

4.1.3 Tercih Edilen Yazılım Bileşenleri

Simülasyon ortamında kullanılacak sanal anahtarlayıcılar için Open vSwitch (OVS) tercih edilmiştir. Bu tercihin sebebi, Open vSwitch'lerin üretim ortamı düzeyinde, açık kaynak kodlu ve çok katmanlı yapıda olması ve en önemlisi her bir OpenFlow protokolününe tam destek vermesidir [21].

Çizelge-3: Simülasyon Ortamı Yazılım Bileşenleri

Sıra	Yazılım Bileşeni	Marka	Versiyon
1	İşletim Sistemi	Ubuntu	20.04.6 LTS (x64)
2	SDN Denetleyicisi	Ryu	4.34
3	Sanal Switch'ler	Open vSwitch	2.13.8
4	Sanal Ağ Emülatörü	Mininet	2.3.0.dev6
5	SDN SBI[1]	OpenFlow	1.3
6	Programlama Dili	Python	3.8.10
7	Trafik Oluşturucusu	iPerf	2.0.13

1: SBI: Güney Köprü Arayüzü

Open vSwitch'ler (OVS), OpenFlow protokolüne bağlı kuyruk yönetimi için Linux işletim sistemi çekirdeğinde bulunan HTB algoritmasını kullanmaktadır. HTB algoritmasında iletilecek paketler için, belirlenmiş oran kadar jeton bulunması halinde paketin iletilmesi -şayet kuyruk için bir maksimum oran belirlenmiş ise bu limit dahilinde iletilmesi-, değilse kuyrukta beklemeye devam etmesi esas alınır [7].

SDN denetleyicisi için Ryu Controller tercih edilmiştir [22]. Bu tercihin sebebi, Ryu'nun komponent bazlı yapısı, tamamen Python programlama dili ortamında çalıştırılabilmesi ve OpenFlow protokolünün tüm versiyonlarını desteklemesidir [23]. Ryu Controller ayrıca Open vSwitch (OVS) kuyruk yapısını yönetebilen bir uygulama programlama arayüzü (API) barındırmaktadır. Kullanılan algoritmaya bağlı olarak Ryu Controller'ın çalıştırabileceği bir Python kodu yazılarak önerilen Air-SDN yöntemi simülasyon ortamında test edilmiştir.

Önerilen Air-SDN yöntemini sanal bir ağ ortamında test edebilmek için Mininet Network Emulator kullanılmıştır [24]. Mininet, SDN konsepti ve OpenFlow protokolünün test ortamı olarak yaygın bir şekilde kullanılmaktadır. Simülasyonlar için Mininet'in tercih edilmesinin sebebi yüksek seviyede topoloji kullanımına izin vermesi, tüm fonksiyonlarının Python programlama dili ile çalıştırılabilmesi ve işletim sistemi düzeyinde (*OS-level*) sanallaştırma uygulayabilmesidir.

Mininet ortamında gerekli topoloji, sanal anahtarlayıcılar ve SDN denetleyicisi çalıştırıldıktan sonra, benzetilecek ağ trafiğini oluşturmak adına trafik oluşturucu programı olan iPerf kullanılmıştır [25]. Trafik oluşturucusu olarak iPerf'in tercih edilme sebebi, TCP ve UDP kullanan istemci-sunucu bağlantıları için güçlü bir ölçüm aracı olması, çeşitli modülasyon parametreleri ile çalışabilmesi, UDP paketleri için belirlenmiş bant genişliğinde trafik gönderebilmesi ve UDP paketleri için paket kaybı ve seğirme ölçümleri yapabilmesidir.

Simülasyon ortamı için Asus marka R507U model (Intel® Core™ i7-7500U CPU @ 2,70GHz x 4, 8 GB RAM, 1 TB M.2 NVMe SSD, NV118 / Mesa Intel® HD Graphics 620 [KBL GT2]) bir taşınabilir bilgisayar üzerine kurulan Ubuntu işletim sistemi kullanılmıştır. Simülasyon ortamında kullanılan yazılımsal bileşenler Çizelge-3'te verilmiştir.

4.1.4 Sınama Parametreleri

Air-SDN yönteminin sınanması için farklı parametreler belirlenmiş ve yöntem, bu parametrelere göre sınanmıştır. Bu parametreler şu şekilde sıralanmıştır:

1- Air-SDN Yönteminin Kullanılmaması/Kullanılması:
   a) No Air-SDN: Air-SDN yönteminde SDN denetleyicisi kullanmadan olmadan, orta switch'lerde veri tipleri için Çizelge-2'de verilmiş sabit kuyruklar kullanılarak),
   b) Air-SDN: Önerilen Air-SDN yöntemini baz alan SDN denetleyicisi (*Ryu Controller*) kullanılarak.
2- Tekli/Çoklu Kullanıcı:
   a) Tekli Kullanıcılı: İlgili veri tipine ait tekli kullanıcıya (*host*) ait verinin ağda test edildiği,

b) Çoklu Kullanıcılı: İlgili veri tipine ait çoklu kullanıcıya (*host pair*) ait verinin ağda test edildiği.

3- Veri Tipi Yükü/Tam Yük:
   a) Veri Tipi Yükü: Çizelge-1'de verilmiş veri tipleri için yalnızca bir veri tipine ait verinin ağda test edildiği,
   b) Tam Yük: Çizelge-1'de verilmiş tüm veri tiplerine ait verinin aynı anda ağda test edildiği.

4- Taşıma Protokolü:
   a) TCP: TCP taşıma protokolü kullanılarak, 30 s süreli,
   b) UDP: UDP taşıma protokolü kullanılarak, her bir veri tipi için Çizelge-1'de verilen ortalama bant genişliği kadar trafik üretilerek, 10 s süreli,
   c) Burst-UDP: UDP taşıma protokolü kullanılarak, her bir veri tipi için 50 MBit/s kadar trafik üretilerek, 10 s süreli.

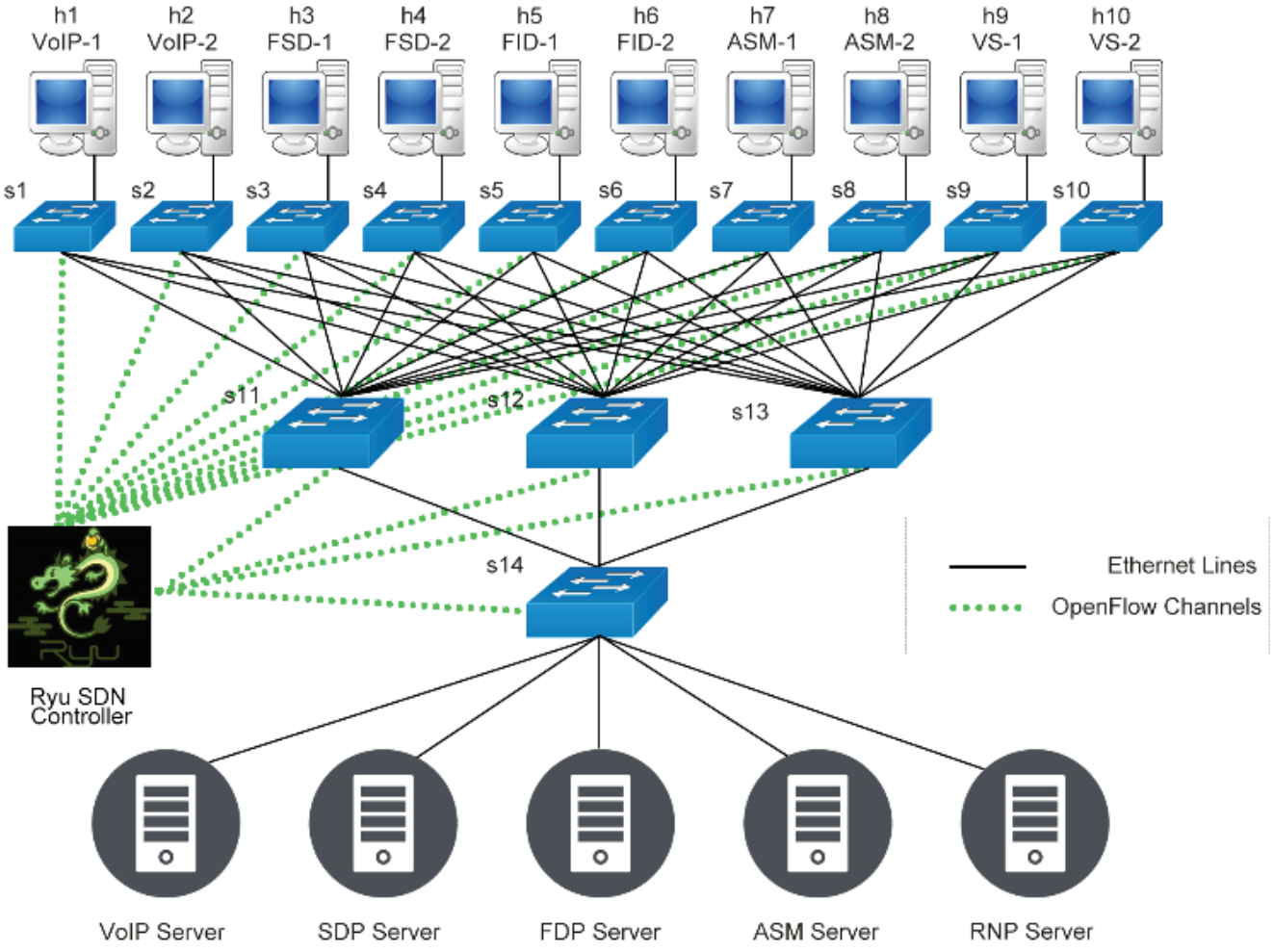
4.2 Başarım Ölçümleri

Önerilen Air-SDN yöntemi ve karşılaştırıldığı No Air-SDN yöntemi tutarlı sonuçlar elde edilene kadar bir önceki başlıkta belirtilen sınama ortamında ve parametrelerde benzetilmiş ve benzetime ilişkin ölçümler kaydedilmiştir. Bu benzetimlerde TCP taşıma protokolü için veri çıkış hacmi ve gecikme ölçümleri, UDP ve Burst-UDP taşıma protokolleri için de veri çıkış hacmi, seğirme ve paket kaybı ölçümleri yapılmıştır. Bu ölçümler parametreler bazında şu şekilde sıralanmıştır:

1- Tekli/Çoklu Kullanıcılı, Veri Tipi Yükünde, TCP Veri Çıkış Hacmi ve Gecikme Ölçümü: Bu parametrelere ait simülasyonlar sonucunda Air-SDN yönteminin, No Air-SDN yöntemine göre öncelikli veri tiplerine daha fazla veri çıkış hacmi ve daha az gecikme sunduğu görülmektedir. Bu sonuçlar Şekil-2 ve Şekil-4'te verilmiştir.

2- Tekli/Çoklu Kullanıcılı, Veri Tipi Yükünde, UDP Veri Çıkış Hacmi, Seğirme ve Paket Kaybı Ölçümü: Bu parametrelere ait simülasyonlar sonucunda ölçülen veri çıkış hacminin, No Air-SDN yöntemine göre Air-SDN yönteminde tüm veri tipleri için çok küçük farklarla aynı kaldığı, seğirmenin öncelikli veri tipleri için daha az olduğu ve paket kaybının VoIP ve FSD veri tipleri için arttığı, diğer veri tiplerinde ise paket kaybı olmadığı görülmektedir. Bu artışın sebebi 'Sonuçların Değerlendirilmesi ve Tartışma' kısmında açıklanmıştır. Bu sonuçlar Şekil-2, Şekil-5 ve Şekil-7'de verilmiştir.

3- Tekli/Çoklu Kullanıcılı, Veri Tipi Yükünde, Burst-UDP Veri Çıkış Hacmi, Seğirme ve Paket Kaybı Ölçümü: Bu parametrelere ait simülasyonlar sonucunda Air-SDN yönteminin, No Air-SDN yöntemine göre tüm veri



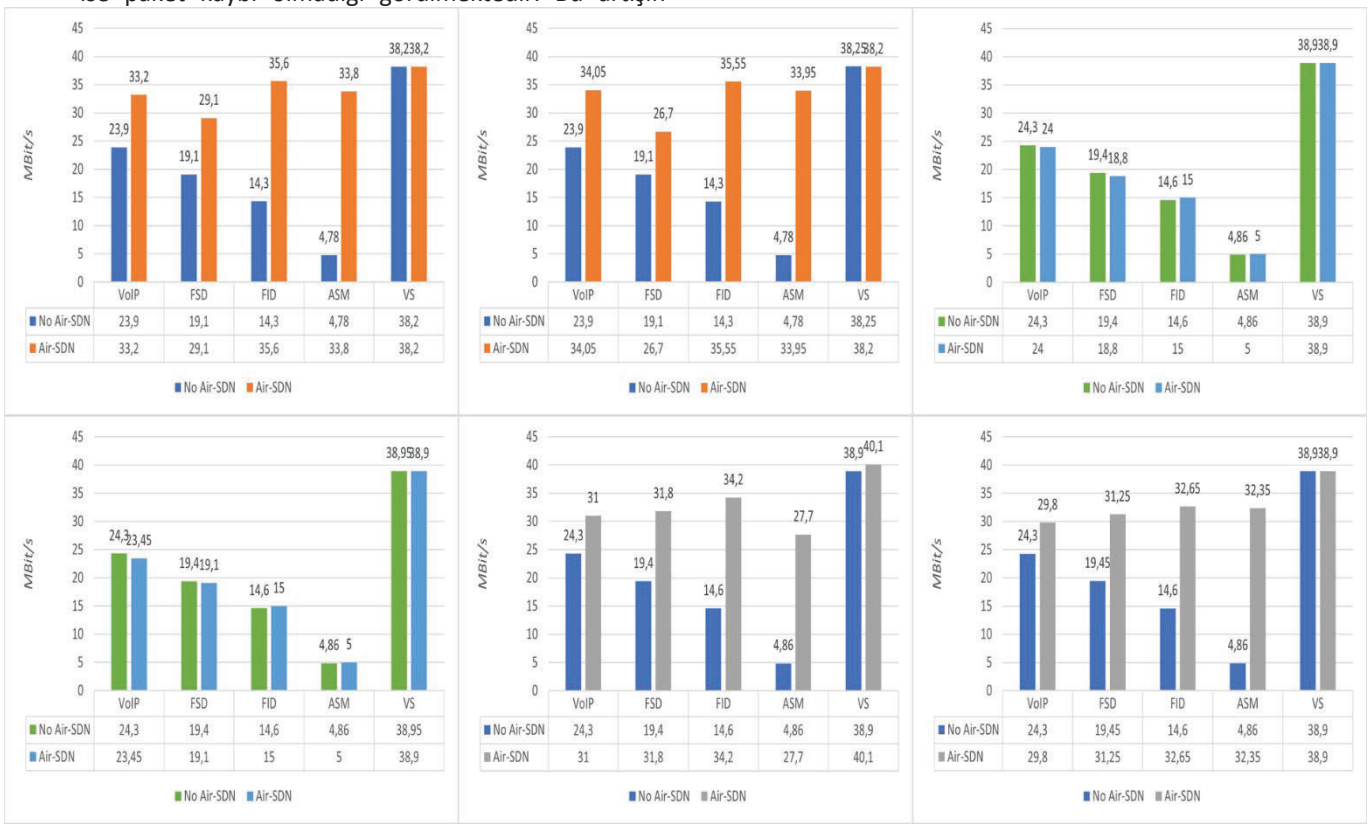
Şekil-1: Air-SDN Sınama ortamı topolojisi

tiplerine daha fazla veri çıkış hacmi sunduğu, seğirmenin VoIP veri tipi için artış gösterdiği, diğer veri tipleri için de azaldığı ve paket kaybının öncelikli veri tipleri için azaldığı görülmektedir. Bu sonuçlar Şekil-2, Şekil-5 ve Şekil-7'de verilmiştir.

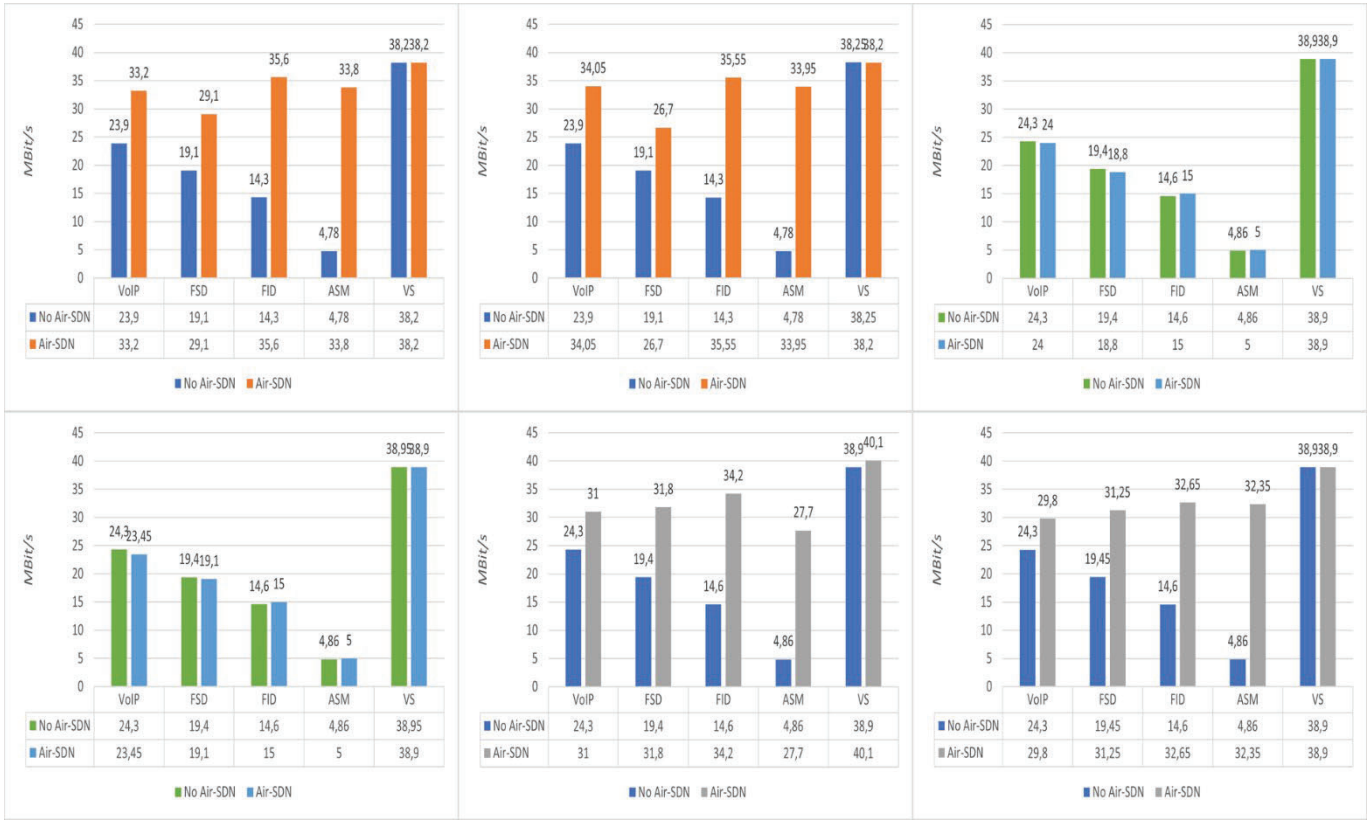
4- Tekli/Çoklu Kullanıcılı, Tam Yükte, TCP Veri Çıkış Hacmi ve Gecikme Ölçümü: Bu parametrelere ait simülasyonlar sonucunda Air-SDN yönteminin, No Air-SDN yöntemine göre öncelikli veri tiplerine daha fazla veri çıkış hacmi ve daha az gecikme sunduğu görülmektedir. Bu sonuçlar Şekil-3 ve Şekil-4'te verilmiştir.

5- Tekli/Çoklu Kullanıcılı, Tam Yükte, UDP Veri Çıkış Hacmi, Seğirme ve Paket Kaybı Ölçümü: Bu parametrelere ait simülasyonlar sonucunda ölçülen veri çıkış hacminin, No Air-SDN yöntemine göre Air-SDN yönteminde öncelikli veri tipleri için çok küçük farklarla aynı kaldığı, seğirmenin tekli kullanıcıda tüm veri tipleri için daha az olduğu ve çoklu kullanıcıda VoIP veri tipi hariç diğer öncelikli veri tipleri için daha az olduğu, paket kaybının ise tekli kullanıcıda VoIP ve FSD veri tipleri için, çoklu kullanıcıda ASM veri tipi haricindeki veri tipleri için arttığı, diğer veri tiplerinde ise paket kaybı olmadığı görülmektedir. Bu artışın sebebi 'Sonuçların Değerlendirilmesi ve Tartışma' kısmında açıklanmıştır. Bu sonuçlar Şekil-3, Şekil-6 ve Şekil-8'de verilmiştir.

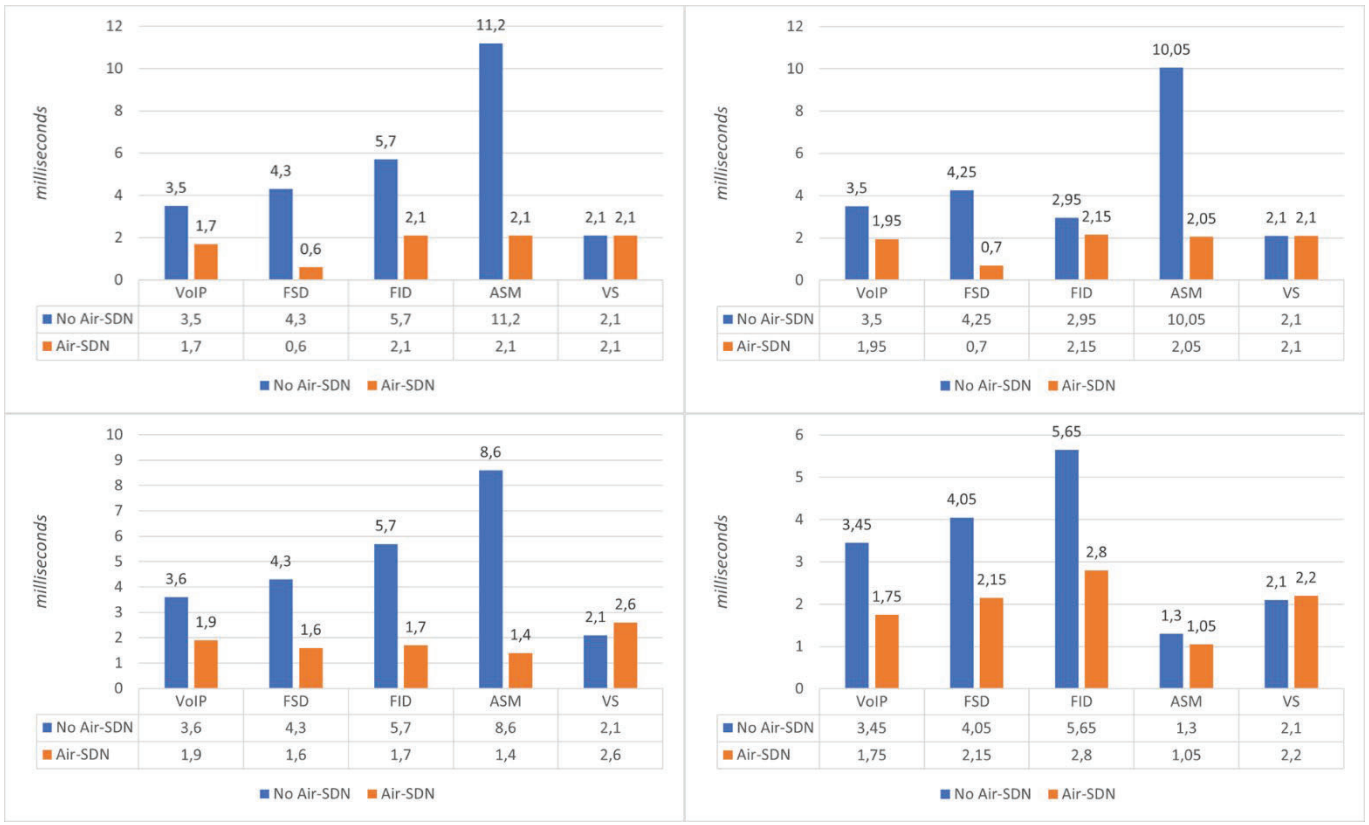
6- Tekli/Çoklu Kullanıcılı, Tam Yükte, Burst-UDP Veri Çıkış Hacmi, Seğirme ve Paket Kaybı Ölçümü: Bu parametrelere ait simülasyonlar sonucunda Air-SDN yönteminin, No Air-SDN yöntemine göre öncelikli veri tiplerine daha fazla veri çıkış hacmi sunduğu, seğirmenin tekli kullanıcıda ASM veri tipi için, çoklu kullanıcıda VoIP ve FID veri tipleri için artış gösterdiği ve diğer öncelikli veri tipleri için azaldığı, paket kaybının ise tekli kullanıcıda öncelikli veri tipleri için, çoklu kullanıcıda ise FSD veri tipi haricinde öncelikli veri tipleri için azaldığı görülmektedir. Bu artışın sebebi 'Sonuçların Değerlendirilmesi ve Tartışma' kısmında açıklanmıştır. Bu sonuçlar Şekil-3, Şekil-6 ve Şekil-8'de verilmiştir.



Şekil-2: Tekli/Çoklu Kullanıcılı, Veri Tipi Yükünde, TCP, UDP ve Burst-UDP Veri Çıkış Hacmi

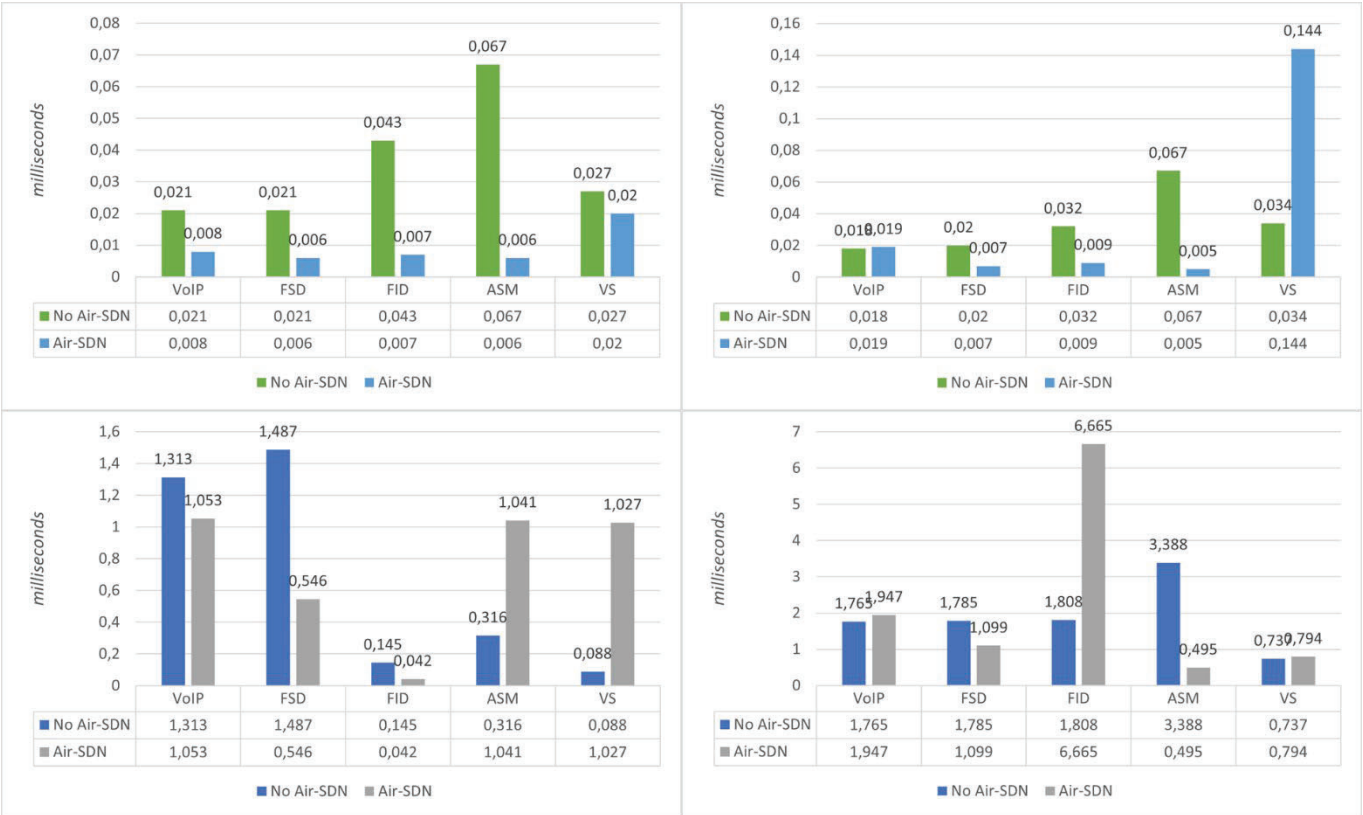


Şekil-3: Tekli/Çoklu Kullanıcılı, Tam Yükte, TCP, UDP ve Burst-UDP Veri Çıkış Hacmi

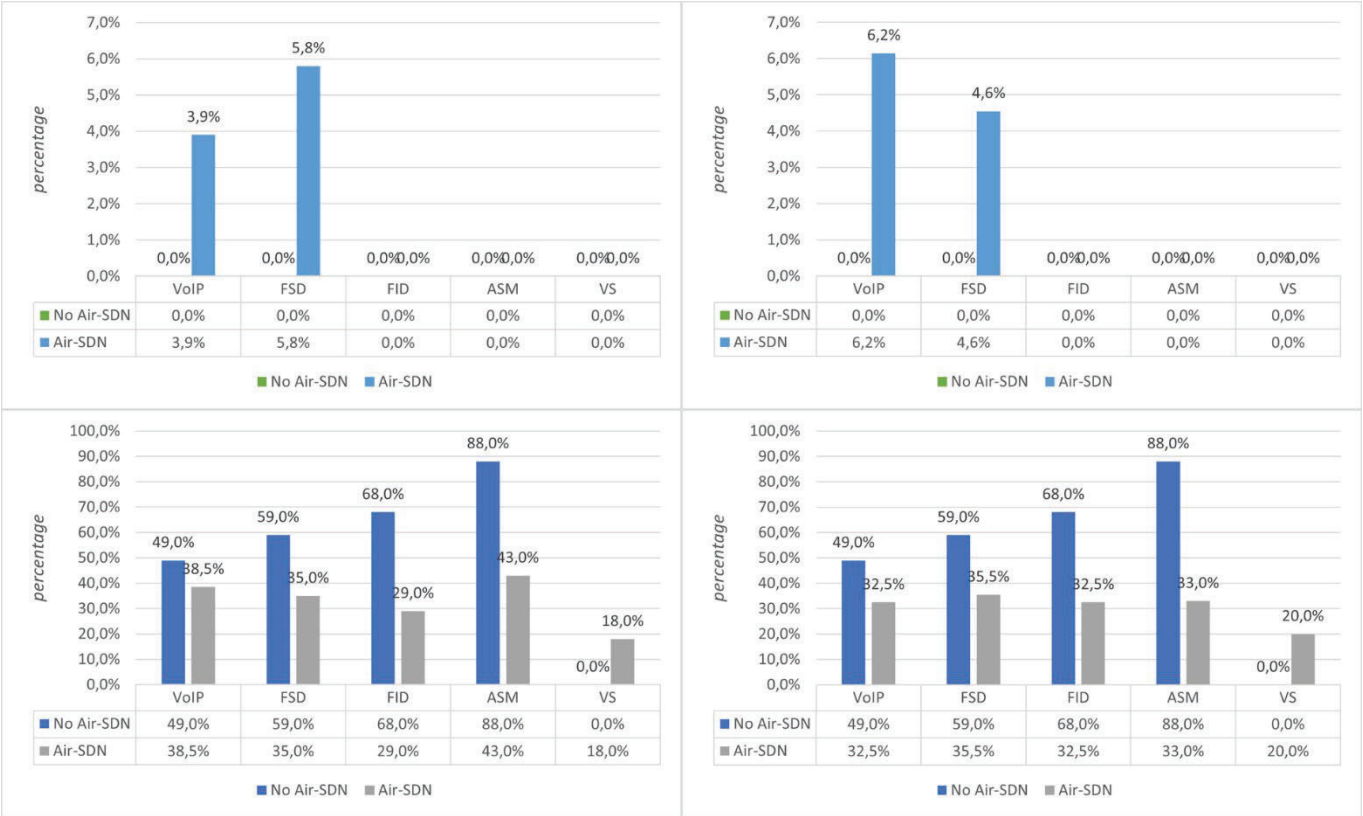


Şekil-4: Tekli/Çoklu Kullanıcılı, Veri Tipi/Tam Yükte, TCP Gecikme

Şekil-5: Tekli/Çoklu Kullanıcılı, Veri Tipi Yükünde, UDP ve Burst-UDP Seğirme



Şekil-6: Tekli/Çoklu Kullanıcılı, Tam Yükte, UDP ve Burst-UDP Seğirme



Şekil-7: Tekli/Çoklu Kullanıcılı, Tam Yükte, UDP ve Burst-UDP Paket Kaybı

Şekil-8: Tekli/Çoklu Kullanıcılı, Veri Tipi Yükünde, UDP ve Burst-UDP Paket Kaybı

5. Sonuçların Değerlendirilmesi ve Tartışma

Bir önceki başlıkta verilen simülasyon sonuçları incelendiğinde, Air-SDN yönteminin TCP taşıma protokolünde tüm parametrelerde beklenen sonuçlara ulaştığı görülmektedir. Air-SDN yönteminin TCP taşıma protokolü ile, belirlenmiş kuyrukları uyarlamalı olarak kullandığı, No Air-SDN yöntemine göre öncelikli veri tiplerine daha fazla veri çıkış hacmi ve daha az gecikme sağladığı anlaşılmaktadır.

Air-SDN yöntemi UDP taşıma protokolü kullanıldığında No Air-SDN yöntemine göre veri çıkış hacmi ve seğirme ölçümlerinde gayet başarılı olsa da paket kaybı ölçümlerinde ulaşılan bazı sonuçlar dikkat çekicidir. Örneğin tekli kullanıcılı, tam yükte VoIP ve FSD veri tipleri için paket kaybı %7'ler civarında olup, diğer veri tipleri için paket kaybı olmadığı saptanmıştır. Bu artışların nedeni şu şekilde açıklanabilir: Air-SDN algoritmasında tüm uç anahtarlayıcılardan akış istatistikleri belirli bir *çerçeve* içerisinde talep edilmekte, tüm anahtarlayıcıların cevap verebilmesi için bir saniyelik bekleme yapılmaktadır. Daha sonra değerlendirme fonksiyonu çalışarak akışlar için uyarlamalı kuyruklama ve yük dengeleme işlemi gerçekleştirilmekte ve kalan *çerçeve* süresi boyunca trafik akışına belirlenmiş konfigürasyonlar ile devam edilmektedir. Talep cevabı ile elde edilen değer, bir önceki *çerçeve* esnasında ölçülen bant genişliği ihtiyacı olup, bir önceki *çerçeve* içerisinde herhangi bir zamanda akışa dahil olan paketler için, daha önceden belirlenmiş kuyruğa ait bant genişliği, başlatılan akış bant genişliğinden daha az ise bu durum ilgili akış için paket kaybına yol açmaktadır. İkinci *çerçeve* başlangıcında akışın ihtiyaç duyduğu bant genişliğine göre kuyruk seçimi yapılmakta ve öncelikli veri tipleri için paket kaybı ortadan kalkmaktadır. Bu duruma örnek olarak VoIP trafiğine çoklu kullanıcılı, veri tipi yükünde Burst-UDP taşıma protokolü parametresine ait akış-kuyruk atamaları, Çizelge-4'te zaman bazında (*t* zamanı 3 alınarak) verilmiştir.

Çizelge-4: Çoklu Kullanıcılı, Veri Tipi Yükünde Uyarlamalı Kuyruk Ataması (Burst-UDP)

Host	Switch	0-3 s	3-6 s	6-9 s	9-12 s	12-15 s
h1	s11	q2	q0	q0	q0	q2
h2	s12	q2	q0	q0	q0	q2

Görüldüğü üzere başlangıçta q2'ye atanan ve 50 MBit/s bant genişliğinde verilen *h1* sunucusuna ait akış 0-3 saniye arasında bir noktada başlamış, 3. saniyede hesaplanan bant genişliği ihtiyacı göz ününde bulundurularak ikinci *çerçeveden* itibaren bu akışa q0 atanmıştır. İlk *çerçeve* esnasındaki akış paketleri daha dar bir bant genişliği ile iletildiğinden paket kaybı ortaya çıkmıştır.

Burst-UDP taşıma protokolü ile yapılan simülasyonlarda No Air-SDN yöntemine göre Air-SDN yönteminin öncelikli veri tiplerine daha fazla veri çıkış hacmi sağladığı görülmüştür. Bu taşıma protokolü ile yapılan simülasyonlarda bazı veri tipleri için paket kaybı ölçümlerinin Air-SDN yönteminde No Air-SDN yöntemine nazaran yüksek çıkmasının sebebi önceki paragraflarda açıklanmıştır. Başka bir dikkat çeken ölçüm tekli/çoklu kullanıcılı veri tipi yükünde Burst-UDP taşıma protokolü ile gözlenen seğirme değerleridir. Bu değerlerde VoIP veri tipi için No Air-SDN yöntemi değerlerine nazaran bir artış gözlemlense de bu artışın yöntemden bağımsız olduğu, her iki yöntemde de tüm veri tipleri için ölçülen seğirmenin, veri tiplerin Çizelge-1'de verilen öncelik oranına göre arttığı ve kendi içerisinde tutarlı olduğu görülmüştür. Bir diğer dikkat edilmesi gereken ölçüm ise tekli/çoklu kullanıcılı tam yükte ölçülen seğirme değerleridir. Bu parametrelerde yapılan tüm simülasyonlarda Air-SDN yöntemi ile bir ya da iki veri tipine ait seğirme değerlerinin sürekli No Air-SDN yöntemi değerlerine göre yüksek çıktığı, bazen ise doruk değerler elde edildiği görülmüştür. Parametrelerin sabit tutulduğu her simülasyonda bu durum için farklı veri tiplerinde yeni ve tutarsız değerler ölçülmüştür. Bu durumun trafik oluşturma programı iPerf'ün seğirme ölçme fonksiyonu kaynaklı rastgele bir davranışı olduğu başta düşünülse de Mininet emülatörünün kullanıldığı seğirme ölçümlerinde rastgele ve tutarsız sonuçlar verdiği Wang'ın çalışmasında ortaya konmuştur [26].

Her ne kadar simülasyonlarda karşılaşılan bu yönteme ait paket kayıpları önceki paragraflarda belirtildiği şekilde açıklanabilir olsa da, paket kayıpları bir ATM sistemi için tolere edilebilir bir durum olmadığından, bu yöntemin UDP ve Burst-UDP taşıma protokolü ile kullanılması durumunda, belirli kuyruk parametrelerinde arzu edilen sonuçlara göre değişikliğe gidilmesi gerektiği saptanmıştır.

Sonuç olarak; önerilen Air-SDN yönteminin, SDN konsepti uygulanmış Hava Trafik Yönetimi (ATM) çekirdek ağlarında TCP taşıma protokolü ile başarılı bir şekilde kullanılabilir olduğu, UDP ve Burst-UDP taşıma protokolü parametrelerinde ise kuyruk parametrelerinin (*min-rate* ve *max-rate*), veri tipine ait parametrelerinin (*average/minimum bandwidth*), veri tipi bazında ağa verilen bant genişliği oranı ve *frame* süresi değerlerinin, üretim ortamındaki gerçek parametreler ile değiştirilerek rahatlıkla kullanılabileceği görülmektedir. Test için kullanılan ağ topolojisinin de üretim ortamına uyumlu bir şekilde ölçeklenebilir olduğu, kapalı bir sistem olduğu için herhangi bir spesifik ağ güvenliği çözümü gerektirmediği de ulaşılan bulgular arasındadır.

Yapılan çalışma ve performans değerlendirmesi sonucunda, Hava Trafik Yönetimi (ATM) ağlarının gerçek zamanlı ve kritik veri yapısını taşıyabilecek kadar güvenilir ve sıkışıklık kontrolü sağlayabilen, sürekli veri yapısına da hızı ile cevap verebilecek ek protokollere ihtiyaç duyulduğu gözlenmiştir. TCP'nin paket iletim garantisi ve sıkışıklık kontrolü gibi özelliklerini taşıyan ancak UDP kadar hızlı olabilecek RTP (*Real Time Protocol*), MTCP (*Multipath TCP*) ve QUIC (*Quick UDP Internet Connections*) gibi bir protokolünün, ATM ağlarının veri yapısını, Air-SDN yöntemi ile sağlanan veri çıkış hacmi, seğirme ve paket kaybı ölçümlerindeki başarıyı ileri taşıyabilecek ve üstteki paragraflarda açıklanan tutarsız sonuçları da ortadan kaldırabileceği düşünülmektedir.

Kaynakça

[1] https://www.eurocontrol.int/Economics/DailyTrafficVariation-States.html (21.02.2023 tarihinde erişildi.)

[2] https://www.frequentis.com/en/pr/air-traffic-managements-first-software-defined-network-sdn-proves-operational-success-brazil (17.03.2023 tarihinde erişildi.)

[3] Thazin N., Nwe K. M., Ishibashi Y., *End-to-End Dynamic Bandwidth Resource Allocation Based on QoS Demand in SDN,* 2019, 25th Asia-Pasific Conference on Communications (APPC), Ho Chi Minh City, Vietnam, pp. 244-249.

[4] Open Networking Foundation, *Software-defined networking: the new form for networks*, 2012, white paper.

[5] Braden R., Clark D., Shenker S. *Integrated services in the internet architecture: an overview*, 1994, IETF RFC 1633.

[6] Baker F., Black D., Blake S., Nichols K. *Definition of the differentiated services field (DS field) in the IPv3 and IPv6 headers*, 1998, RFC 2474.

[7] Krishna H., van Adrichem L.M., Kuipers F.A. *Providing bandwidth guarantees with OpenFlow*, 2016, IEEE Symposium on Communications and Vehicular Technology in the Benelux, SCVT.

[8] Xu C., Chen B., Qian H. *Quality of service guaranteed resource management dynamically in software defined network*, 2015, Journal of Communications, Vol.10 No.11.

[9] Kadim U.N., Mohammed I.J. *SDN-RA: An optimized reschedule algorithm of SDN load balancer for data center networks based on QoS*, 2020, 2nd International Scientific Conference of Al-Ayen University (ISCAU-2020).

[10] Gu Z., Wang Y., Lin X. *Achieving real-time quality of service in software defined networks*, 2018, 44th Annual Conference of the IEEE Industrial Electronics Society.

[11] Al-Jawad A., Shah P., Gemikonaklı O., Trestian R. *Policy-based QoS management framework for software-defined networks*, 2018, International Symposium on Networks, Computers and Communications (ISNCC).

[12] Guo X., Lin H., Li Z., Peng M., *Deep-Reinforcement-Learning-Based QoS-Aware Secure Routing for SDN-IoT,* 2020, IEEE Internet of Things Journal, Vol.7, No.7.

[13] Vanitchasatit P, Sanguankotchakorn T. *A class-based adaptive QoS control scheme adopting optimization technique over WLAN SDN architecture*, 2022, International Journal of Computer Networks & Communications (IJCNC), vol. 14, no.3, pp.55-72

[14 Al-Haddad R., Velazquez E.S., Fatima A., Winckles A. *A novel traffic shaping algorithm for SDN-slices networks using a new WFQ technique*, 2021, International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 12, No.1.

[15] Li F., Cao J., Wang X., Sun Y., Sahni Y. *Enabling software defined networking with QoS guarantee for cloud applications,*", 2017, IEEE 10th International Conference on Cloud Computing.

[16] Baek S.U., Park C.H., Kim E., Shin D. *Implementation and verification of QoS priority over software defined networking*, 2016, International Conference of Internet Computing and Internet of Things (ICOMP).

[17] Akella A.V., Xiong K. *Quality of service (QoS) guaranteed network resource allocation via software defined networking (SDN)*, 2014, IEEE 12th International Conference on Dependable, Autonomic and Secure Computing.

[18] Karakus M., Durresi A. *Quality of Service (QoS) in Software Defined Networking (SDN): A Survey,* 2017, Journal of Network and Computer Applications, Vol.80, pp. 200-218.

[19] Ongkasae K., Nakazato H., Koga T., Lu X. *QoS implementation in system wide information management (SWIM) network model,* 2015, IEEE Twelfth International Syposium on Autonomous Decentralized Systems, Taichung, Taiwan, pp. 181-184.

[20] Izquierdo-Zaragoza J. -L., Lins W., Leydold P., Eier D. *Hierarchical software-defined networks for wide-area air traffic management networks,* 2019, Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, USA, pp. 1-11.

[21] Open vSwitch *What is Open vSwitch?*, https://openvswitch.org (19.04.2023 tarihinde erişildi.)

[22] Ryu *What's Ryu?*, https://ryu-sdn.org (20.04.2023 tarihinde erişildi.)

[23] Pyhton *Get Started*, https://www.python.org/ (20.04.2023 tarihinde erişildi.)

[24] Mininet *An Instant Virtual Network on your Laptop (or other PC)*, https://mininet.org (20.04.2023 tarihinde erişildi)

[25] iPerf *The ultimate speed test tool for TCP, UDP and SCTP*, https://iperf.fr (20.04.2023 tarihinde erişildi.)

[26] Wang S. *Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet*, 2014, IEEE Symposium on Computers and Communications (ISCC), Funchal, Portugal.