T.C.

EGE ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü

SDN İLE KURULMUŞ MPEG-DASH SAĞLAYICISINI KULLANARAK ANDROID İŞLETİM SİSTEMİ ÜZERİNDE VİDEO OYNATICI UYGULAMASI GELİŞTİRME VE SİSTEMDE ALGISAL KALİTEYİ ETKİLEYEN FAKTÖRLERİN İNCELENMESİ

Yüksek Lisans Tezi

Emrecan ÇETİN

Elektrik Elektronik Mühendisliği

İzmir

2020

T.C.

EGE ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü

SDN İLE KURULMUŞ MPEG-DASH SAĞLAYICISINI KULLANARAK ANDROID İŞLETİM SİSTEMİ ÜZERİNDE VİDEO OYNATICI UYGULAMASI GELİŞTİRME VE SİSTEMDE ALGISAL KALİTEYİ ETKİLEYEN FAKTÖRLERİN İNCELENMESİ

Emrecan ÇETİN

Danışman(lar) : Dr. Öğr. Üyesi Nükhet ÖZBEK

Elektrik Elektronik Mühendisliği Anabilim Dalı
Elektronik Mühendisliği Yüksek Lisans Programı

İzmir
2020

Emrecan ÇETİN tarafından Yüksek Lisans tezi olarak sunulan "SDN ile Kurulmuş MPEG-DASH Sağlayıcısını Kullanarak Android İşletim Sistemi Üzerinde Video Oynatıcı Uygulaması Geliştirme ve Sistemde Algısal Kaliteyi Etkileyen Faktörlerin İncelenmesi" başlıklı bu çalışma EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliği ile EÜ Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 17.01.2020 tarihinde yapılan tez savunma sınavında aday oybirliği/~~oyçokluğu~~ ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	**: Dr. Öğr. Üyesi Nükhet Özbek**	.....
Raportör Üye	**: Doç. Dr. Radosveta Sokullu**	.....
Üye	**: Doç. Dr. Tuncay Ercan**	.....

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi olarak sunduğum "SDN ile Kurulmuş MPEG-DASH Sağlayıcısını Kullanarak Android İşletim Sistemi Üzerinde Video Oynatıcı Uygulaması Geliştirme ve Sistemde Algısal Kaliteyi Etkileyen Faktörlerin İncelenmesi" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

17 / 01 / 2020

Emrecan ÇETİN

ÖZET

SDN İLE KURULMUŞ MPEG-DASH SAĞLAYICISINI KULLANARAK ANDROID İŞLETİM SİSTEMİ ÜZERİNDE VİDEO OYNATICI UYGULAMASI GELİŞTİRME VE SİSTEMDE ALGISAL KALİTEYİ ETKİLEYEN FAKTÖRLERİN İNCELENMESİ

ÇETİN, Emrecan

Yüksek Lisans Tezi, Elektrik-Elektronik Mühendisliği Anabilim Dalı

Tez Danışmanı: Dr. Öğr. Üyesi Nükhet ÖZBEK

Ocak 2020, 62 sayfa

Bu tezde video iletiminde kullanılan popüler yöntemlerin üzerindeki algısal kaliteyi etkileyen faktörler incelenmiştir. Ayrıca tüm unsurlarıyla bir sistem tasarımı önerilmiştir.

Çalışmada önerilen sistemde son zamanlarda popüler olan MPEG-DASH standardı video iletim standardı olarak kullanılmıştır. Ağ bazlı kalite yönetimi için sunucu tarafı ile istemci arasındaki ağda SDN (yazılımsal tanımlı ağ) kullanılmıştır. Bu ağın kontrolcülerinin içerisinde QoS (servis kalitesi) düzeyleri tanımlanmış ve bunların yönetimi yapılmıştır. Tanımlanan düzeylerin testleri için farklı ağ trafik senaryoları belirlenmiştir.

İstemci olarak Android işletim sistemi üzerinde bir video oynatıcı uygulaması geliştirilmiştir. Kendi sunucumuza bağlanan bu uygulama önceden belirlenmiş ağ trafik senaryoları ile çalıştırılmış ve bireylere testler yaptırılmıştır. Uygulamadan toplanan verilerle algısal kalite tabloları oluşturulmuş ve bu tablolardaki veriler SSCQE (Tek Uyaranlı Devamlı Kalite Değerlendirmesi) yöntemiyle değerlendirilmiştir. Yapılan değerlendirmeler sonucunda QoE'i etkileyen en önemli faktörler ortaya çıkarılıp sistemin bu bağlamda nasıl optimize edileceğine dair tavsiyeler verilmiştir.

Anahtar sözcükler: MPEG-DASH, SDN, QoE, QoS, SSCQE, Android, Exoplayer, yazılımsal tanımlı ağ, tecrübe kalitesi, video iletişimi.

ABSTRACT

DEVELOPING ANDROID BASED VIDEO CLIENT APPLICATION WITH USING SDN MPEG-DASH CONTENT PROVIDER AND EXAMINING FACTORS AFFECTING PERCEPTUAL QUALITY IN THE SYSTEM

ÇETİN, Emrecan

MSc in Electrical and Electronics Eng.

Supervisor: Assist. Prof. Dr. Nükhet ÖZBEK

January 2020, 62 pages

In this thesis, factors that affect perceptual quality in popular methods of video communication have been studied. Also, a full system has been proposed in this study.

In the proposed system, MPEG-DASH method wich is popular nowadays has been used as a video communictation method between server and client sides. In server side, SDN (Software Defined Networks) has been used for quality management of the network. QoS (Quality of Service) levels have been defined and managed in this SDN's controllers. Different network trafiic scenarios have been defined to test defined QoS levels.

Android video player application has been developed as a client. This application which is connected to our server has been run and made some tests in different network traffic scenarios by people. Data tables were created with the data that is collected from this application and they had been analyzed by SSCQE (Single Stimulus Continuous Quality Evaluation) method. As a result of analyze, the facts have been determined which are affecting QoE mostly and some suggestions have been given to improve optimization for QoE in the system.

Keywords: MPEG-DASH, SDN, QoE, QoS, SSCQE, Android, Exoplayer, software defined networks, quality of experience, video communication.

ÖNSÖZ

Bu tezde video akış iletimiyle ilgili güncel konular ele alınmıştır. Tezin amacı, bir video akış sistemini tüm unsurlarıyla oluşturmak ve bu sistemin üzerindeki algısal kaliteyi etkileyen faktörleri incelemektir. Bu bağlamda bir sunucu ve bir istemci tasarlanmıştır. Günceli yansıtmak için MPEG-DASH standardı kullanılmıştır. Performans geliştirmesi için sunucu tarafında SDN araçlarından faydalanılmıştır. İstemci uygulaması, son zamanlarda popüler olan Android işletim sistemi üzerinde uygulama olarak geliştirilmiş ve bu uygulama tüm sistemi SSCQE değerlendirme yöntemiyle değerlendirebilecek ve sistemi monitör edebilecek özelliklerle donatılmıştır. Sistemin çalışabilmesi için SDN parametreleri belirlenmiş ve kodlanmıştır. Performans testleri için belirli ağ senaryoları tasarlanmıştır.

Son olarak ise kurulan sistemin performansı hem nesnel hem de öznel olarak ölçülmüş ve yorumlanmıştır.

İZMİR

17/01/2020

Emrecan ÇETİN

İÇİNDEKİLER

Sayfa

İÇ KAPAK iii

KABUL ONAY SAYFASI v

ETİK KURALLARA UYGUNLUK BEYANI vii

ÖZET ix

ABSTRACT xi

ÖNSÖZ xiii

İÇİNDEKİLER DİZİNİ xv

ŞEKİLLER DİZİNİ xix

TABLOLAR DİZİNİ xxiii

SİMGELER VE KISALTMALAR DİZİNİ xxv

1. GİRİŞ 1

2. GENEL BİLGİLER 3

2.1 Literatür Özeti 3

2.2 Geleneksel Akış 5

2.2.1 Gerçek zamanlı aktarım protokolü (rtp) 5

2.2.2 Gerçek zamanlı akış protokolü (rtsp) 6

İÇİNDEKİLER (devam)

2.3 Aşamalı İndirme 6

2.4 Uyarlamalı/Uyarlanabilir Akış 6

2.4.1 Kod dönüştürme 7

2.4.2 Ölçeklenebilir kodlama 8

2.4.3 Akış değiştirme 8

2.5 Http Tabanlı Uyarlamalı Akış 9

2.5.1 Neden http? 11

2.5.2 Apple’s http live streaming (hls) 11

2.5.3 Microsoft’un canlı düzgün akışı (lss) 12

2.5.4 Adobe’nin http dinamik akışı 12

2.5.5 Mpeg http üzerinden dinamik uyarlamalı akış (mpeg-dash) 12

2.6 Video Codec'leri 16

2.6.1 Video çerçeveleri 16

2.6.2 Kod çözme ve sunum zaman damgaları 17

2.6.3 H.264/mpeg-4 avc 17

2.7 Konteyner Formatları 18

2.8 Video Kalite Seviyeleri 19

2.9 Talep Üzerine Video (Vod) Ve Canlı Yayın 20

İÇİNDEKİLER (devam)

2.10 Google'ın Android İşletim Sistemi 21

2.10.1 Android'de desteklenen medya biçimleri 22

2.10.2 Android'de desteklenen http üzerinden uyarlanabilir protokoller 22

2.10.3 Exoplayer medya oynatıcısı 22

2.11 Farklı Http Tabanlı Uyarlamalı Çözümler Arasında Karşılaştırma 22

2.12 Yazılım Tanımlı Ağ Teknolojisi (Sdn) 24

2.13 Mininet 25

2.14 Ryu 26

3. GEREÇ VE YÖNTEM 28

3.1 Sunucu Tarafı 28

3.1.1 Http sunucusu 29

3.1.2 Mininet sdn ağı 29

3.2 İstemci Uygulaması 33

3.2.1 Uygulamanın özellikleri 34

3.2.2 Uygulama yapısı ve çekirdek modülü 34

3.2.3 Segment indirici ile adaptasyon mekanizması 35

3.2.4 Adaptasyon algoritması 38

İÇİNDEKİLER (devam)

3.2.5 Uygulama arayüzü ve akışı 39

3.3 Testler 41

3.3.1 Performans testleri 43

3.3.2 Öznel testler 47

4. TARTIŞMA 55

5. SONUÇ 56

KAYNAKLAR DİZİNİ 57

TEŞEKKÜR 61

ÖZGEÇMİŞ 62

ŞEKİLLER DİZİNİ

Şekil Sayfa

2.1. Kod Dönüştürme ile Adaptif Akış 8

2.2. Ölçeklenebilir Kodlama ile Adaptif Akış 8

2.3. Akış Değiştirme ile Adaptif Akış 9

2.4. Zaman-Akış Değişimi Grafiği 9

2.5. İstemci ve Sunucu Arasındaki İletişim 10

2.6. MPEG-DASH Standardizasyonu 13

2.7. MPD Dosyasının Yapısı 14

2.8. MPEG-DASH İstemci ve Sunucu Arasındaki Akış 15

2.9. Çerçeveler arasındaki ilişki 16

2.10. Android Resmi Logosu 21

2.11. Geleneksel Ağ ile SDN Ağları Arasındaki Fark 24

2.12. SDN Katmanları 25

2.13. Mininet Aracının Yapısı 26

3.1. Sistemin Genel Yapısı 28

ŞEKİLLER DİZİNİ (devam)

3.2. Topoloji Görsel Şeması 30

3.3. Alıcı Tarafındaki Mekanizma 34

3.4. Uygulamanın Katmanları 35

3.5. Segment İndirici ve Adaptasyon Mantığı 35

3.6. Uygulamanın Arayüz Akışı 39

3.7. Uygulama Ekran Görüntüsü-1 İlk Açılış ve MPD URL'i Giriş Ekranı 40

3.8. Uygulama Ekran Görüntüsü-2 Video Oynatıcı 40

3.9. Uygulama Ekran Görüntüsü-3 Puanlama 40

3.10. 320x240 42

3.11. 1280x720 42

3.12. EDGE Hızı Testi, Bitrate-Saniye Grafiği 43

3.13. 25Mbit Internet Bağlantısı Hızı Testi, Bitrate-Saniye Grafiği 44

3.14. Network senaryosu 44

3.15. Network Senaryosu Sonucu 45

3.16. Özel kullanıcı τ(t) throughput ve b(t) video bitrate grafiği 46

3.17. Normal kullanıcı τ(t) throughput ve b(t) video bitrate grafiği 46

3.18. Sistemin Bant Genişliği Grafiği 48

3.19. Uygulamadaki Öznel Değerlendirme Mekanizması 49

ŞEKİLLER DİZİNİ (devam)

3.20. Test A. 81. Port Testi, 81. Portta İzleyici (Trafik) Var 50

3.21. Test B. 80. Port Testi, 81. Portta İzleyici (Trafik) Var 50

3.22. Test C. 82. Port Testi 81. Portta İzleyici (Trafik) Var 50

3.23. Test D. 82. Port Testi, 80. Portta İzleyici (Trafik) Var 51

3.24. Test E. 80. Port Testi, 80. Portta İzleyici (Trafik) Var 51

3.25. Test F. 81. Port Testi, 80. Portta İzleyici (Trafik) Var 51

3.26. Test G. 80. Port Testi, Trafiksiz 52

3.27. Test H. 82. Port Testi, Trafiksiz 52

3.28. Test I. 81. Port Testi, Trafiksiz 52

3.29. Test J. 82. Port Testi, 82. Portta İzleyici (Trafik) Var 53

3.30. Test K. 80. Port Testi, 82. Portta İzleyici (Trafik) Var 53

3.31. Test L. 81. Port Testi, 82. Portta İzleyici (Trafik) Var 53

3.32. Öznel Puanlama Sonuçları 54

3.33. Öznel Puanlama Ortalamaları 55

TABLOLAR DİZİNİ

Tablo Sayfa

2.1. Örnek MPD Dosyası 14

2.2. CBP, BP ve MP Profilleri Arasındaki Temel Farklar 18

2.3. LSS, HLS ve MPEG-DASH Arasındaki Farklar 23

3.1. Kullanılan modülün yüklenmesi için şu komut kullanılmıştır 29

3.2. Topolojideki elemanları yaratan kod parçası 31

3.3. Ağı başlatan kod parçası 31

3.4. Kontrolörün programlanması ve switch'e bağlanması 32

3.5. QoS parametrelerinin kontrolörde tanımlanması 32

3.6. QoS parametrelerinin kontrolörde belirli bir kurala atanması 33

3.7. Host üzerinde HTTP yayının başlatılması ile ilgili kod parçası 33

3.8. Bağlantıyı Başlatan Fonksiyon 36

3.9. Veri Okuma Fonkisyonu 36

3.10. Veri Okumayı Bitiren Fonkisyon 36

3.11. Akış Başladığında Çağırılan Fonksiyon 37

3.12. Transfer Edilen Toplam Byte'ları Hesaplayan Fonksiyon 37

3.13. Akış Bittiğinde Çağırılan Fonksiyon 38

3.14. Adaptasyon Fonkisyonu 38

TABLOLAR DİZİNİ (devam)

3.15. Segment-Kalite Tablosu 41

3.16. Ölçüm Metrikleri 42

3.17. Ağ Senaryosunun Baş. Gecikmesi ve Bant Gen. Kullanım Yüzdesi 45

3.18. SDN Kontrolcü Kuralları 45

3.19. Test Senaryoları 47

3.20. Test Ortamı Unsurları 48

3.21. Test Senaryolarının Yapılış Sırasına Göre Harflendirilmiş Tablosu 49

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler	Açıklama
T	Periyot (saniye)
τ	Throughput (Çıktı)
$\tau(t)$	Anlık Throughput (Çıktı)
$b(t)$	Bitrate
$u(t)$	Ağ Kullanım Oranı
bw_avail	Ağın Gerçek Bant Genişliği

Kısaltmalar

MPEG	Moving Pictures Experts Group
DASH	Dynamic Adaptive Streaming over HTTP
HTTP	Hyper Text Transfer Protocol
ML	Machine Learning
ITU	International Telecommunication Union
QoS	Quality of Service
QoE	Quality of Experience
LSS	Microsoft Live Smooth Streaming
SSCQE	Single Stimulus Continuous Quality Evaluation

1. GİRİŞ

Günümüzde internetteki verinin çoğunluğunu video içerikleri oluşturmaktadır. En çok kullanılan uygulamaları video içeriği sağlayan platformlar oluşturmaya başlamıştır. Bunlardan bazıları Youtube, Netflix, Instagram, Facebook gibi uygulamalardır. Haliyle veri olarak oldukça büyük olan video içeriklerini en iyi şekilde kullanıcılara ulaştırma gerekliliği doğmuştur. Video'lar en iyi şekilde sıkıştırılıp paketlense dahi internet ağında oluşan pek çok etmen bu paketlerin iletimini etkileyebilmektedir. Kullanıcılar ise video içeriklerini kesintisiz, pürüzsüz ve kaliteli bir şekilde izlemek istediği için bu durum kullanıcı deneyimini doğrudan etkiler.

İnternet ortamı üzerinden yapılan video yayınlarının son kullanıcıya akıcı ve kaliteli bir şekilde iletilmesi uzun zamandır üzerinde çalışılan bir konudur. Şimdiye kadar pek çok farklı sağlayıcı farklı yöntemler geliştirmiştir. Bir MPEG (Moving Pictures Experts Group) standardı olan DASH (Dynamic Adaptive Streaming over HTTP), bu yöntemler içerisinde günümüzde yaygın olarak kullanılan modellerden biridir. MPEG-DASH, HTTP sunucusundaki farklı bit hızlarında işlenmiş video parçalarının istemci tarafından indirilerek video gösterimi sağlayan bir ISO standartıdır. Bu protokol istemcinin sahip olduğu bant genişliğini adaptasyon algoritmalarından geçirerek istemci için en doğru bit oranına sahip video dosyalarını indirir. Ayrıca HTTP protokolünü kullandığı için implementasyonu ve uygulanması oldukça kolaydır.

Son kullanıcıların videoları kesintisiz, akıcı ve kaliteli bir şekilde izleyebilmesi için MPEG-DASH adaptasyon algoritmaları üzerine pek çok makale yazılmıştır. Bazı örnek çalışmalar sonraki bölümde verilecektir.

MPEG-DASH adaptasyon yöntemleri içerik sağlayıcısında tutulan farklı kalitedeki video dosyaları arasında geçiş yapmayı sağlar. Bu geçişleri anlık ölçülen ağ durumuna göre karar vererek yapar. Böylece mevcut ağ olanaklarını maksimum düzeyde kullanarak kullanıcıya en kaliteli ve en akıcı video deneyimini sunmak amaçlanır. Bu düzeydeki optimizasyon, video istemcisi tarafında hesaplanır ve uygulanır. Fakat istemci tarafında yapılan bu hesaplamalar her zaman yeterli olmaz. Ağdaki trafik durumu, sunucu tarafındaki yoğunluk gibi etmenler bu deneyimi olumsuz yönde etkileyebilir. Bu durumda istemcinin kabiliyeti sınırlı kalır.

Tüm sistemin kullanıcı deneyimi söz konusu olduğunda, MPEG-DASH sağlayıcısı tarafında SDN (Software Defined Network) tabanlı bir ağ yapısı kullanarak ağ kaynaklı sorunların üstesinden gelmek amaçlanmıştır. SDN'de ağ trafiği düzenlenerek tüm kullanıcılara eşit ve yüksek kaliteli video akışı sağlamak amaçlanır. Bunun yanı sıra bazı ayrıcalıklı kullanıcıların izleme kalitesini maksimum düzeye getirmek gibi özelleştirilebilir kabiliyetler de yaratılabilir. Örneğin, Youtube gibi platformlarda ücretli Premium abone olan insanlara daha yüksek bant genişliği ve öncelik tanınabilir. Kontrolcünün nasıl programlandığına bağlı olarak sistem akıllı kararlar verebilir. Bu sayede farklı katmanlarda farklı olanaklar sunularak her katmanda en iyi optimizasyon elde edilebilir.

İçerik sağlayıcılar kullanıcı deneyimini ölçmek için farklı yöntemler geliştirmiştir. Bu ölçümlerin değerlendirmeleri sayesinde en iyi kullanıcı deneyimini elde etmek amaçlanmıştır. Kullanıcıların en sık karşılaştığı sorunlardan bazıları şunlardır;

- Video iletimi sırasında yaşanan donmalar
- Video'nun geç yüklenmesi
- Ağ koşulları uygun olmasına rağmen görüntü kalitesinin düşük olması
- Video'nun kalite geçişlerinin fazla agresif ya da yavaş olması

Bu sorunların sonucu olarak algısal kalite faktörünün düşük olması beklenir. Bu faktörü ölçümlemek için video iletimi sırasında puanlama yöntemleri kullanılır. Alınan puanlar ise belli başlı algoritmalarla sentezlenir.

Video akışı kalite değerlendirmesinde, en yaygın kullanılan ITU-R 500-11 SSCQE standardı kullanılmıştır.

Bu çalışmada Android işletim sistemi olan cihazlarda çalışabilecek bir MPEG-DASH video istemcisi uygulaması yazılmıştır. Android işletim sisteminin seçilmiş olmasındaki etken bu işletim sisteminin günümüzde açık kaynak kodlu olarak en yaygın kullanılan işletim sistemlerinden birisi olmasıdır. Android işletim sistemi telefon, tablet, tv ve bazı laptoplar olmak üzere her türlü ortamda kullanılmaktadır. Bunun sonucu olarak pek çok video sağlayıcı platformunun bu işletim sistemi için uygulama çözümleri vardır. Kullanılan uygulamaya video oynatımı sırasında videonun algısal kalitesini ölçebilmek için bir değerlendirme sistemi eklenmiştir. Aynı zamanda çalışma kapsamında sunucu tarafında Ubuntu üzerinde çalışan bir Mininet sistemi kullanılmıştır. Mininet'te yaratılan yazılımsal switch'leri kontrol etmek için RYU[1] kontrolcüsü kullanılmıştır. Bu kontrolcü REST-API aracılığıyla programlanarak belirli QoS (Quality of Service) düzeyleri yaratılmıştır. Kontrolcüye girilen farklı QoS parametreleri sayesinde farklı QoE (Quality of Experience) odaklarımız olmuştur. Tüm bu odaklar ayrı ayrı senaryolarla sübjektif puanlamaya alınıp bu sonuçlar tablolarla sergilenmiştir.

[1] RYU, Mininet'teki switch kontrolcülreinin REST API ile programlanabilmesini sağlayan bir araç

2. GENEL BİLGİLER

Bu bölümde ilk olarak örnek çalışmalar ve ardından çalışmada kullanılan kavramlarla ilgili bilgiler verilmektedir.

Video aktarımı için üç ana yöntem vardır:

Geleneksel akış.

Aşamalı indirme.

Uyarlamalı/uyumsal akış.

Uyarlamalı akış bu tez konusu için uygun görülmüştür, dağıtım protokolü olarak HTTP, ağ yönetimi olarak SDN, QoE çözümleri ile birlikte sonuçların değerlendirilmesinde SSCQE kullanılmıştır. HTTP ile video akışı yapılan tekniğin bazı uygulamaları olan Apple'ın HTTP Canlı Yayını, Microsoft'un Live Smooth Streaming'i, MPEG Dinamik Uyarlamalı HTTP Üzerinden Akış standartlarına değinilmiştir. İsteğe bağlı video veya canlı yayın olarak iki farklı hizmet türü sağlanabilir.

Kullanılan video ve ses CODEC'leri açıklanmaktadır. Konteyner formatları verilmiştir. Android işletim sistemi yeteneklerine değinilmiştir. Sonrasında, farklı akış yaklaşımlarının kısa bir karşılaştırması yapılmıştır. En sonda, SDN araçları ve Android işletim sistemi ele alınmıştır.

2.1 Literatür Özeti

MPEG-DASH standartı kullanılarak gerçeklenmiş bazı video istemci örnekleri incelenmiştir. İsveç'teki KTH teknoloji enstitüsünde yazılmış bir tezde bazı adaptasyon algoritmalarının performanslarının ele alındığı bir Android uygulaması örneği çalışması bulunmuştur (Luciano Rubio Romero, 2011).

Bu çalışmada MPEG-DASH ve diğer DASH benzeri HTTP tabanlı bazı standartlar ele alınmış, Android ortamında uygulamanın nasıl yazılacağı anlatılmış ve 3 farklı adaptasyon algoritması incelenmiştir (Luciano Rubio Romero, 2011). Bu algoritmalar agresif adaptasyon mekanizması, muhafazakar adaptasyon algoritması ve ortalama adaptasyon mekanizmasıdır (Luciano Rubio Romero, 2011). Çalışmada Android uygulaması yazılmış ve adaptasyon algoritmaları karşılaştırılıp bir tanesi seçilmiştir. Sonuç olarak bu uygulamanın bant genişliği kullanım verimleri incelenmiştir (Luciano Rubio Romero, 2011).

Önerilen adaptasyon mekanizmaları şöyledir;

- Agresif Adaptasyon Algoritması (Luciano Rubio Romero, 2011). Bu algoritma ölçülen bant genişliğini hiçbir işleme sokmadan kullanır. Bu değerin altında kalan bitrate'e segmentleri indirir.
- Ölçülü Adaptasyon Algoritması (Luciano Rubio Romero, 2011). Bu algoritma ölçülen bant genişliğini belirli bir duyarlılık katsayısıyla çarpar. Bu çarpımdan çıkan sonucun altında kalan bitrate'e sahip segmentleri indirir.
- Ortalama Adaptasyon Algoritması (Luciano Rubio Romero, 2011). Bu algoritma ölçülen son 3 bant genişliğininin ortalamasını alır ve bu değeri belirli bir duyarlılık katsayısıyla çarpar. Bu çarpımdan çıkan sonucun altında kalan bitrate'e sahip segmentleri inidirir.

Android uygulamasında gerçeklenen bu algoritmalar belirli ağ senaryolarıyla karşılaştırılmış ve sonuçlar ayrıntılı bir şekilde verilmiştir. Agresif adaptasyonun değişimlere çabuk tepki verdiği ancak çok fazla akış değişimi yaptığı, ölçülü adaptasyonun daha yavaş cevap verdiği ve ortalama adaptasyon algoritmasının da daha geç cevap verdiği ve ağın bant genişliğini tam kullanamadığı gözlenmiştir.

Bir makalede HTTP tabanlı adaptif video akışındaki (HAS) deneyim kalitesini SDN kullanılarak nasıl optimize edileceği ele alınmıştır (Sangeeta Ramakrishnan, vs., 2015).

Ana amaç olarak tüm kullanıcıların ortalama video kalitesi ölçümlerini adaletli ve ağın el verdiğince yüksek seviyede tutmak olmuştur. Ele alınan sistemde Cisco switchlerini kontrol eden OpenDaylight kontrolcüsü kullanılmıştır. Video kalite parametreleri olarak PSNR (Peak Signal to Noise Ratio) ve SSIM (Structural Similarity Index) kullanılmıştır. Çalışmada eş-oranlı optimizasyon şemasına karşılık ortaklaşa optimizasyon yöntemi önerilmiştir. Bu 2 yöntemin ve baz durumun video kalite karşılaştırmaları yapılmış, önerilen ortaklaşa optimizasyon yönteminin %75 daha iyi sonuç verdiği gözlenmiştir (Sangeeta Ramakrishnan vd., 2015).

Farklı bir çalışmada ise DASH'in deneyim kalitesini yükseltmek için SDN kullanımı ele alınmıştır (Ali C. Begen vd., 2016).

İncelenen bu çalışma kapsamında DASH sağlayıcısı kurulmuş ve bu sağlayıcının ağ katmanına SDN kontrolcüsü yerleştirilmiştir. Kontrolcü olarak RYU kullanılmıştır. Bu kontrolcü Mininet tarafından yaratılan sanal switchleri kontrol ederek kullanıcı deneyimini yükseltmek için ağ trafiğinin yönetimini yapmaktadır. Bu bağlamda bazı QoE parametreleri belirlenmiş ve bunlar kural olarak kontrolcüye girilmiştir. Kontrolcünün yanında SDN tabanlı bir network uygulaması da bu sistemde yer almıştır. Bu uygulama kontrolcüye QoS kuralı mesajları göndererek onu güncelleyebilmektedir. Bunun yanı sıra kontrolcü de bu uygulamaya ağ durumu ile ilgili bazı mesajlar atmakta ve uygulama kontrolcüye atayacağı QoS mesajlarına bu şekilde kara vermektedir (Ali C. Begen vd., 2016). QoE parametreleri olarak ise yığın kalitesi, başlangıç gecikmesi, duraklama süreleri, ortalama video kalitesi, kalite değişimleri alınmıştır (Ali C. Begen vd., 2016). Çalışma sonunda ise bu parametreler

ölçülerek geliştirilen kontrolcü güncellemeli uygulamanın performans metrikleri çıkarılmıştır. Standart DASH çözümlerine göre kazanım olduğu gözlenmiştir.

TÜBİTAK'ta yayınlanan bir makalede makine öğrenimi ile destekleniş SDN sistemi sayesinde video akış servislerindeki deneyim kalitesini artırmaya yönelik bir çalışma ele alınmıştır (Asma BEN LETAIFA, 2018).

Ele alınan sistemde sübjektif olarak kullanıcılardan alınan puanların ve objektif kalite parametrelerinin (PSNR, VQM, SSIM gibi) harmanlanmasıyla bu verileri girdi olarak alan ML algoritmaları çalıştırılıp makine öğrenmesi yapılmıştır. Öğrenme aşamasından sonra bu sistem çalışırken objektif kalite parametrelerini ölçerek daha önce öğrendiği bilgilerle sübjektif puanları tahmin etmiştir. Bunun sonucunda deneyim kalitesine duyarlı bir izleyici yaratılmıştır. Böylece bu sistem sübjektif kaliteyi yükseltecek olan SDN kontrolcüsü parametrelerini bulup onlar üzerinde yoğunlaşabilecek ve hatta onları düzeltecek seviyeye getirilmiştir (Asma BEN LETAIFA, 2018).

2.2 Geleneksel Akış

Geleneksel akış, oturum kuran, durum bilgisi olan bir protokol gerektirir. Servis sağlayıcı ile istemci arasında medya bir dizi paket olarak gönderilir. Gerçek Zamanlı Akış Protokolü (RTSP) ile birlikte Gerçek Zamanlı Aktarım Protokolü (RTP) bu tür bir servisi uygulamak için sıklıkla kullanılır.

2.2.1 Gerçek zamanlı Aktarım protokolü (RTP)

Unicast ve Multicast servisleri üzerinden gerçek zamanlı iletimi yönetmek için kullanılan İnternet Protokolüdür (agciyiz.net, 2019). Ses ve görüntünün internet üzerinden iletilmesi için kullanılan standart paket formatını ifade eder. (3cx, 2019).

RTP spesifikasyonu, veri aktarım protokolü (RTP) ve RTP Kontrol Protokolü (RTCP) olan iki alt protokolü açıklar:

1. RTP, zamanla birlikte farklı CODEC'ler kullanarak multimedya verilerini aktarmak için kullanılır. Bu protokolde zaman damgaları ve sıra numaraları bulunmaktadır. Bu zaman damgaları ve sıra numaraları paket kaybını algılamak ve gerektiğinde yeniden sıralamayı gerçekleştirmek ve senkronize etmek için kullanılırlar.
2. RTCP, senkronizasyon ve servis kalitesi için kontrol bilgilerini belirtir. Bu paketlerde gönderilebilecek parametreler belirtilmektedir. Bu protokol, toplam akışın bant genişliğinin en fazla %5'ini kullanmalıdır.

RTP gerçek zamanlı uçtan uca çoklu ortam uygulama transferleri için dizayn edilmiştir. UDP üzerinden çalışır, çok noktaya yayın dağıtımına uygundur. RTP bir alt protokolü olan RTCP (Real-Time Transfer Control Protocol) ile birlikte çalışmaktadır (agciyiz.net, 2019). Verinin kendisi RTP tarafından iletilir ve RTCP ise bu hizmetin

kalitesiyle ilgili geribildirim yapmak için kullanılır (3cx, 2019). RTCP sayesinde veri alıcısı herhangi bir paket kaybının olup olmadığını tespit eder, ya da jitter gecikmelerini gidermeye çalışır.

RTP, zamanında teslimatı sağlayacak bir hizmet sağlamaz ve hizmet kalitesini garanti etmez. Ek olarak, protokolün kendisi tarafından sağlanan bir akış kontrolü yoktur. Akış kontrolü ve tıkanıklıktan kaçınma, uygulanacak uygulamaya bağlıdır.

2.2.2 Gerçek zamanlı akış protokolü (RTSP)

The Real Time Streaming Protocol (RTSP) eğlence ve iletişim sistemlerinde medya sunucularındaki verilerin akışını kontrol etmek için tasarlanan bir ağ denetim protokolüdür. Bu protokol bitiş noktaları arasındaki medya bağlantılarının kurulması ve kontrol edilmesinde kullanılır (opax, 2019). Medya sunucusu üzerindeki bir medyayı uzaktan durdurmak veya oynatmak gibi bir takım temel eylemler bu protokol ile yapılır (sozluk.cozumpark, 2019).

RTSP RTP tabanlı dağıtım mekanizmalarına dayanır. RTSP durumsaldır, hem istemci hem de sunucuya istek gönderebilir. Bu istekler üç farklı yolla olabilir: (1) birkaç istek / cevap için kullanılan kalıcı bağlantılar, işlemler, (2) istek/yanıt işlemi başına bir bağlantı veya (3) bağlantı yok.

2.3 Aşamalı indirme

Aşamalı indirme, bir web sunucusundan bir dizüstü bilgisayarda veya cep telefonunda video oynatıcı gibi bir istemciye medya akışı sağlamak için kullanılan bir tekniktir. Yani dijital medya dosyalarının sunucudan istemciye, genellikle bilgisayardan başlatıldığında HTTP protokolünü kullanarak aktarılmasıdır (nginx, 2019).

Aşamalı indirme ile oynatma başlamadan önce video dosyasının yalnızca küçük bir kısmının indirilmesi gerekir (Aşamalı indirme işleminde bile, düşük bant genişliği indirme işlemini oynatma hızının gerisinde bırakabilir, bu durumda izlemeye devam etmek için yeterli malzeme indirilene kadar duraklatılır).

Aşamalı indirme işleminin bazı dezavantajları vardır: (1) Kullanıcı karar verirse çok fazla bant genişliği kaybı video içeriğini izlemeyi durdurmak için oynanabilir. (2) Her istemcinin bant genişliği eşit kabul edildiğinden dolayı, bit hızı uyarlaması yoktur. (3) Canlı medya kaynaklarını desteklemez.

2.4 Uyarlamalı/Uyarlanabilir Akış

Uyarlamalı akış uyarlamalı bit hızı akışı olarak da bilinir. Adaptif bit hızı akışı, bağlantı, yazılım veya cihaz ne olursa olsun mümkün olan en iyi video kalitesini ve görüntüleyici deneyimini sağlar. Kısaca ABR (Adaptive Bit Rate) olarak adlandırılan bu akışların çoğu, MPEG DASH ve Apple'ın HLS gibi HTTP tabanlı teknolojiler aracılığıyla

sağlanır (wowza, adaptive-bitrate-streaming, 2019). Uyarlanabilir akış, kullanıcıya sunulan videonun kalitesini ayarlamak ve mevcut durumda bu kullanıcıya verilebilecek en iyi kaliteyi sunmak için, kullanıcının mevcut bant genişliğini ve CPU kapasitesini tespit eden bir tekniktir. Ek olarak, çoklu canlı ve isteğe bağlı yayınlarını tüm değişen koşullarla uyumlu hale getirir.

Bağlantı iyi olduğunda, monitör yüksek kaliteli bir akış alır, ancak bağlantı hızı azalırsa, sunucu bağlantıyı sürdürmek için düşük kalitede olsa bile daha düşük bir veri hızı dosyası gönderir. (medianova, icerik-dagitim-ve-optimizasyon, adaptive-streaming, 2019) Videoyu kullanıcıya her mümkün kullanıcı için mümkün olan en verimli şekilde ve en yüksek kullanılabilir kalitede sunmak için tasarlanmış bir teknolojidir. Video sağlayıcısının, hedeflemek istediği ekran boyutlarının (veya cihazların) her biri için farklı bir video oluşturmasını sağlar. İzleyicinin her zaman iyi görünecek bir video almasını sağlamak için belirli ekran boyutlarına uyması için belirli bir video dosyası yayınlanabilir.

Tamponlama veya akış kesintisi yoktur. Tamponlama işleminde uyarlamalı akış şu şekilde çözüm sunar; arabelleğe alma, bir kullanıcı videoyu oynatmaya devam edecek kadar hızlı bir şekilde bir video dosyasını indiremediğinde gerçekleşir. Çoğu video saniyede 24 kare oynar, bu nedenle arabelleğe alma işlemini önlemek için internet bağlantısının her saniye en az 24 kare indirmesi gerekir. Uyarlamalı akış, bu durumu kullanıcının internet bağlantısının hızına “adapte ederek” çözebilir. Basit bir ifadeyle, küçük bir videonun büyük bir videodan daha hızlı bir şekilde indirilebileceğini açıklar. Böylece bir kullanıcı yavaş bir internet bağlantısına sahipse, uyarlanabilir video akışı videonun oynatılmasını sağlamak için daha küçük bir video dosya boyutuna geçecektir. Kullanıcının önceliğinin kaliteyi korumak yerine tamponlamadan kaçınmak olduğuna dikkat edilmektedir (Bitmovin, 2019).

Video kaynağının bit hızını değişken bant genişliğine uyarlama teknikleri üç kategoriye ayrılabilir: kod dönüştürme (bölüm 2.4.1), ölçeklenebilir kodlama (bölüm 2.4.2) ve akış değiştirme (bölüm 2.4.3).

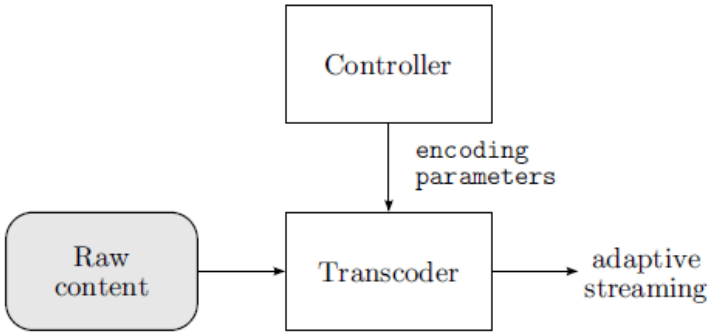
2.4.1 Kod Dönüştürme

Kod dönüştürme, bir medya dosyasının oynatılabileceği uyumlu hedef cihazların sayısını artırmak için bir ses veya video dosyasını bir kodlama biçiminden diğerine dönüştürme işlemidir (stackpath, transcoding, 2019). Bu, uyumsuz verilerin daha iyi desteklenen, daha modern bir veri biçimine dönüştürülmesini sağlar. Hedef aygıt formatı desteklemiyorsa veya yalnızca sınırlı depolama kapasitesine sahipse, kod dönüştürme genellikle yapılır.

Kod dönüştürmeyle ham video içeriğini anında sunucu tarafında dönüştürmek mümkündür. Belirli bir bit hızını eşleştirmek için bir kodlamadan diğerine kodlar dönüştürülür (techopedia, transcoding, 2019).

Genellikle video formatlarını gizlemek için kullanılır. Grafiklerin ve HTML dosyalarının mobil cihazlarda ve küçük ekranlı diğer Web özellikli ürünlerde kullanılmasına

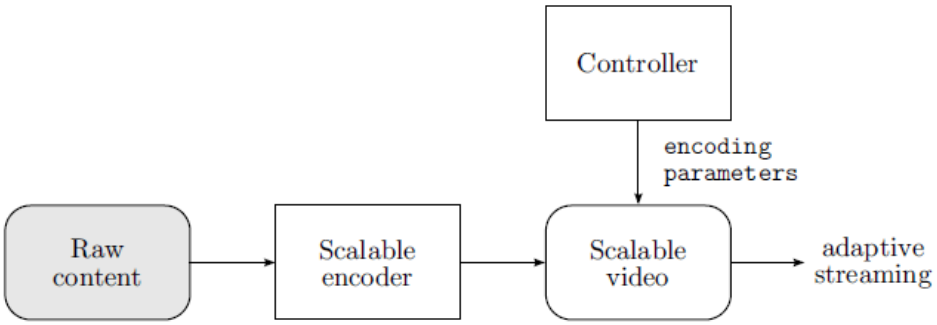
olanak sağlar, bant genişliği düşer ve nispeten daha az bellek kullanır. Kod dönüştürme, bir dosyayı alan ve istemciye göre değiştirmek için belirli bir format kullanan bir proxy sunucusu kullanılarak uygulanır.



Şekil 2.1: Kod Dönüştürme ile Adaptif Akış (Luciano Rubio Romero, 2011)

2.4.2 Ölçeklenebilir Kodlama

İki tür video nesnesi katmanı vardır; MPEG-4 işlevselliği sağlayan video nesnesi katmanı ve azaltılmış işlevsellik video nesnesi katmanı (sciencedirect, scalable-coding, 2019). İkincisi, H.263 ile bit akışı uyumluluğu sağlar. H.264 / MPEG-4 AVC gibi ölçeklenebilir bir CODEC standardı kullanılarak (bölüm 2.5.4'te ayrıntılı olarak açıklanmıştır), resim çözünürlüğü ve kare hızı, ham video içeriğini yeniden kodlamak zorunda kalmadan uyarlanabilir. Bu yaklaşım işlem yükünü azaltma eğilimindedir, ancak açıkça bir dizi ölçeklenebilir CODEC formatıyla sınırlıdır.

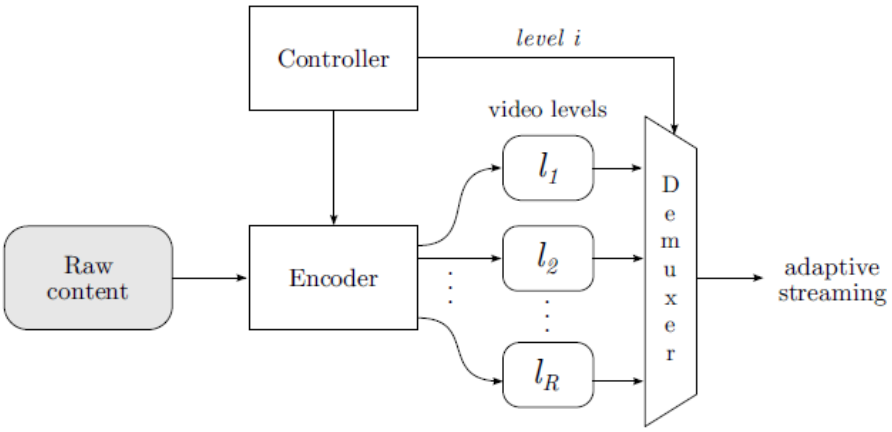


Şekil 2.2: Ölçeklenebilir Kodlama ile Adaptif Akış (Luciano Rubio Romero, 2011)

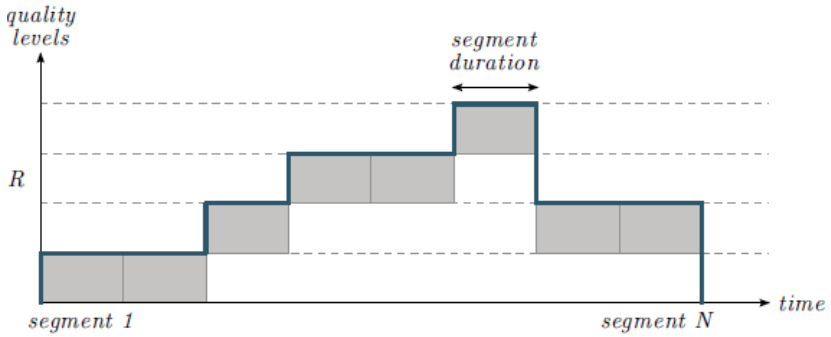
2.4.3 Akış Değiştirme

Akış değiştirme yaklaşımı, ham video içeriğini, video içeriği olarak bilinen aynı içeriğin R versiyonlarını üreten birkaç farklı artan bit hızında kodlar. Şekil 2.3'te gösterildiği gibi, bir algoritma, kullanıcının mevcut bant genişliğine uygun video seviyesini dinamik olarak seçmelidir. Mevcut bant genişliğinde değişiklikler meydana geldiğinde, algoritma sürekli oynatımı sağlamak için basitçe farklı seviyelere geçer.

Bu yöntemin temel amacı, işlem maliyetlerini en aza indirmektir, çünkü tüm video seviyeleri üretildikten sonra başka bir işleme gerek yoktur. Ek olarak, bu yaklaşım uygulanacak belirli bir CODEC formatı gerektirmez, yani tamamen CODEC agnostiğidir. Buna karşılık, aynı video içeriği R kere (ancak farklı bit hızlarında) kodlandığından depolama ve iletim gereksinimleri dikkate alınmalıdır. Kalite seviyelerinin artımlı olmadığını unutmayın, bu nedenle sadece bir alt talep istenmelidir. Bu yaklaşımın tek dezavantajı, yalnızca belirli bir seviye kümesi olduğu için kaba taneciklilikliktir. Ek olarak, belirli bir kalite düzeyi için istemci yoksa bu seviyeyi oluşturmaya gerek yoktur. Bununla birlikte, yalnızca sunuculardaki depolama alanına mal olur ve tüm sunucuların bir akışın tüm düzeylerini depolaması gerekmez.



Şekil 2.3: Akış Değiştirme ile Adaptif Akış (Luciano Rubio Romero, 2011)



Şekil 2.4: Zaman-Akış Değişimi Grafiği (Luciano Rubio Romero, 2011)

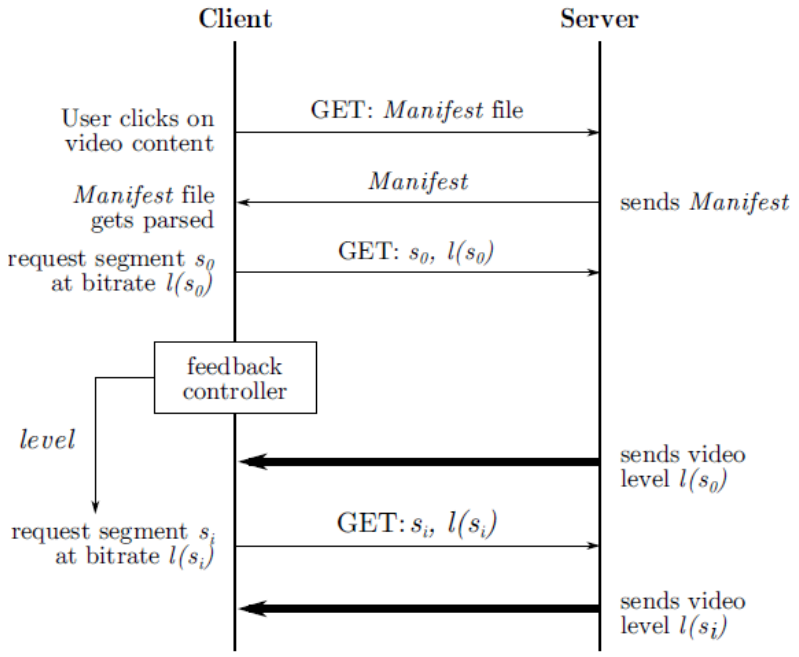
2.5 HTTP Tabanlı Uyarlamalı Akış

HTTP (Hiper-Metin Transfer Protokolü), İnternet'te ağ üzerinde (www-world web wide) kullanılan dağıtım protokolü olarak yaygın şekilde kullanılmaktadır. Bilginin internet üzerinden sunucudan kullanıcıya ne şekilde ve nasıl aktarılacağını gösteren bir nevi yoldur (Mediaclick, 2019). HTTP protokolü istemci (PC) ile sunucu (Server) arasındaki alışveriş kurallarını belirler. Port olarak ise 80 portunu kullanır. İstemci sunucuya bir istek gönderir. Bu istek Internet Explorer, Google Chrome veya Mozilla Firefox gibi web browser'lar

aracılığıyla iletilir. Sunucu bu isteği alır ve Apache veya IIS gibi web sunucu programları aracılığıyla cevap verir (Hosting, 2019). HTTP sürümlerinde bazı kurallar farklıdır. Örneğin HTTP 1.0 sürümünde istekler ve cevaplar tek tek bağlantı oturumu kurulup tek tek gönderilip/alınırken, HTTP 1.1 sürümünde istekler ve cevaplar sıralı şekilde gönderilir/alınır ve tüm alış veriş bittikten sonra bağlantı oturumu kapatılır (Networkkampus, 2019).

HTTP tabanlı uyarlamalı akış, yeni bir protokol tanımlamak yerine HTTP'yi dağıtım protokolü olarak kullanan karma bir yöntemdir. Video ve ses kaynakları aynı uzunlukta kısa bölümlere kesilir (genellikle birkaç saniye). İsteğe bağlı olarak bölümler bir video grubu boyunca kesilebilir (bölüm 2.5.1'de açıklanmıştır), böylece her bölüm kendi aralarında geçmiş/gelecek bağımlılıklarının olmadığı anlamına gelen anahtar bir çerçeveyle başlar. Son olarak, tüm bölümler istenen formatta kodlanır ve bir HTTP sunucusunda barındırılır.

İstemciler sırayla segmentleri talep eder ve bunları HTTP aşamalı indirmeyi kullanarak indirir. Segmentler sırayla oynatılır ve bitişik olduklarından sonuçta ortaya çıkan genel oynatım düzgündür. Tüm adaptasyon mantığı istemci tarafından kontrol edilir. Bu, istemcinin bit hızını değiştirmek veya düşürmek için her bölümün getirme süresini hesapladığı anlamına gelir. Geri bildirim denetleyicisinin, istemci tarafında uygulanan anahtarlama mantığını temsil ettiği Şekil 2.5'te temel bir örnek gösterilmektedir. Daha kalın oklar, gerçek bir veri bölümünün iletilmesine karşılık gelir.



Şekil 2.5: İstemci ve Sunucu Arasındaki İletişim (Luciano Rubio Romero, 2011)

2.5.1 Neden HTTP?

HTTP, İnternet'te dağıtım protokolü olarak yaygın şekilde kullanılmaktadır. HTTP çok yaygın bir şekilde kullanıldığından, HTTP tabanlı hizmetler NAT ve güvenlik duvarı sorunlarından kaçınır. Çünkü istemci güvenlik duvarının arkasından veya Ağ Adresi

Çevirisinden (NAT) TCP bağlantısını başlatır veya HTTP için delikler güvenlik duvarı veya NAT servisi aracılığıyla bilerek açılmıştır. NAT veya güvenlik duvarı, HTTP sunucusundan gelen paketlerin istemciye bir TCP bağlantısı veya SCTP ilişkisi üzerinden teslim edilmesine izin verecektir (bu tezin geri kalanı için TCP'nin HTTP için aktarım protokolü olarak kullanıldığını varsayacağız). Ek olarak, HTTP TCP kullandığından, güvenilir bayt akışı teslimi için otomatik olarak alır ve TCP yoğun tıkanıklık önleme mekanizmaları sağlar. HTTP tabanlı servisler mevcut HTTP sunucularını ve CDN altyapılarını kullanabilir.

Son olarak, akış oturumu tamamen müşteri tarafından kontrol edilir, böylece istemciler yalnızca TCP bağlantılarını açıp başlangıç içerik bit hızı seçtikleri için HTTP sunucusuyla anlaşmaya gerek yoktur. Daha sonra müşteriler mevcut bant genişliklerine bağlı olarak sunulan akışlar arasında geçiş yapar.

2.5.2 Apple’s HTTP Live Streaming (HLS)

HTTP Canlı Akışı (HLS olarak da bilinir), Apple tarafından uygulanan HTTP tabanlı uyarlamalı bit hızı akış iletişim protokolüdür (Wikipedia, 2019). Ayrıca HLS görsel ve sesli medyayı internet üzerinden izleyicilere sunmak için bir medya yayın protokolüdür (Dacast, hls-streaming-protocol, 2019). HLS uyarlamalı bitrate videoda standart olarak ortaya çıkmıştır. Uyarlanabilir bit hızı video dağıtımı, bir müşterinin bant genişliği kapasitesini tespit eden ve video akışının kalitesini birden fazla bit hızı ve/veya çözünürlük arasında ayarlayan bir sunucu ve istemci yazılımı birleşimidir. Uyarlanabilir bit hızı video deneyimi, tek bir bit hızında statik bir video dosyası sunmaktan daha iyidir, çünkü video akışı, istemcinin kullanılabilir ağ hızı kadar iyi ya da kötü olmak üzere orta akışta kaydırılabilir (oynatmada arabelleğe alma ya da kesintiye karşın istemcinin ağ hızı videonun kalitesini destekleyemediğinde olur). Standart olarak HLS akış formatı ve Apple'ın iOS ve Google’ın Android işletim sistemi gibi önde gelen oyuncularla Encoding.com, geliştiricilere standardın evrimi, mimarisi ve cihaz uyumluluğu konusunda eksiksiz bir rehber sağlamıştır. Birden fazla cihaza video dağıtımını çevreleyen sorunları hepimiz iyi biliyoruz. Buna karşın HLS içeriği mobil/tablet, OTT ve masaüstü için uyarlanabilir bit hızı teknolojisi için hızla standart hale geldi. Bu kaynakla, sektörde HLS akışının kökeni, teknik özellikleri, cihaz desteği ve diğer birçok değişkenle ilgili mevcut özellikleri ortaya koyulmuştur (Encoding, http-live-streaming-hls, 2019). Mayıs 2009'da Apple, sınırlı ve sınırsız multimedya veri akışlarını iletmek için HTTP tabanlı bir akış ortamı iletişim protokolü (AppleHLS) yayınladı. AppleHLS, 1998 yılında piyasaya sürülen Emblaze Network Media Streaming teknolojisine dayanmaktadır. HLS akış protokolü, MP4 video içeriğini 10 saniyelik kısa parçalara böler (Luciano Rubio Romero, 2011). HTTP daha sonra bu kısa klipleri izleyicilere sunar. Bu teknoloji, HLS'yi çok çeşitli cihaz ve güvenlik duvarlarıyla uyumlu hale getirir. HLS canlı akışları için gecikme (veya gecikme süresi), şartnameye uygun olarak 15-30 saniye aralığında olma eğilimindedir. Bu kesinlikle akılda tutulması gereken önemli bir faktördür. (Dacast, hls-streaming-protocol, 2019)

Web'e güç sağlayan aynı protokolü kullanan HLS, sıradan web sunucularını ve içerik dağıtım ağlarını kullanarak içerik dağıtmanıza izin verir. HLS güvenilirlik için tasarlanmıştır ve mevcut kablolu ve kablosuz bağlantı hızı için oynatmayı optimize ederek ağ koşullarına dinamik olarak adapte olur(Apple Developer, 2019).

Her bir medya dosyası bir MPEG-2 taşıma akışı veya bir MPEG-2 ses temel akışı olarak biçimlendirilmelidir.

2.5.3 Microsoft'un Canlı Düzgün Akışı (LSS)

2009 yılında Microsoft Corporation, HTTP üzerinden uyarlamalı akış için yaklaşımını yayınladı (Microsoft Corporation, 2009). Microsoft'un Live Smooth Streaming (LSS2) format özelliği ISO Base Media Dosya Formatını temel alır ve Korumalı Birlikte Çalışabilir Dosya Formatı (PIFF) olarak standartlaştırılır, ancak manifest dosyası Genişletilebilir İşaretleme Dilini (XML) temel alır. (liste 2.3'te basitleştirilmiş bir örnek gösterilmiştir). Microsoft, Silverlight eklentisinin kurulmasını gerektiren bir Smooth Streaming demo3 sağlar. Bu çevrimiçi uygulamada, mevcut bant genişliği çok basit bir kullanıcı arayüzü içerisinde kolayca ayarlanabilir. Bir ağ kullanım grafiği, uyarlanmış video çıkışının yanı sıra dinamik olarak görüntülenir.

2.5.4 Adobe'nin HTTP Dinamik Akışı

HTTP Dinamik Akış Adobe'nin akış sunucularında uyarlamalı akış desteğidir. Flash Media Server'da (FMS) tanıtılan HTTP Dinamik Akış (HDS), aynı zamanda Adobe Media Server 5'in bir parçasıdır (Leadtools, 2019). Medya dosyaları (standart MPEG-4 Part 12'ye dayanan Flash Video veya F4V) ve bildirimler (Flash Media Manifest veya F4M) için farklı format özellikleri kullanır. Adobe'nin çözümünü uygulamak için özel ve ticari bir ürün olan bir Flash Media Streaming Server kurmak gereklidir. Ayrıca, kullanıcıların Adobe'nin Flash Player'ını yüklemeleri gerekir (Luciano Rubio Romero, 2011).

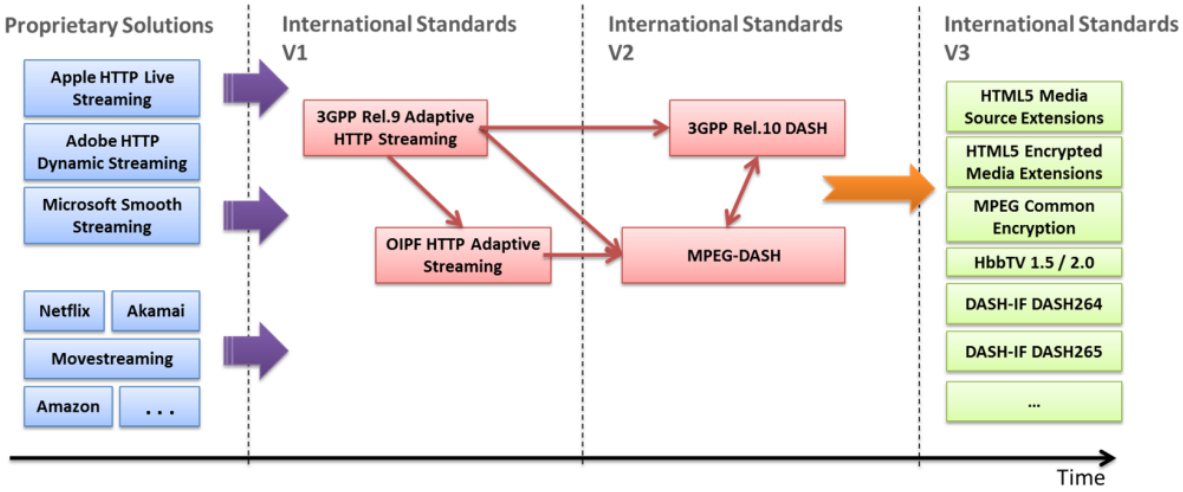
Adobe HTTP Dinamik Akış (HDS), isteğe bağlı ve yüksek kaliteli içeriğin canlı yayınlanması için tescilli bir çözümdür. RTP gibi UDP tabanlı protokollerin aksine, Adobe HDS, normal HTTP bağlantıları üzerinden standart MP4 ortamları sunmak için uyarlanabilir bir bit hızı tekniği kullanır. Uyarlanabilir olduğundan, akış ayarları mevcut ağ koşullarına ve istemci donanımına göre değiştirilebilir. HDS, dosya şifreleme desteğine sahiptir ve 6 Mbps'ye kadar bit hızlarıyla HD kalitesinde video (1080p'ye kadar) sunmak için kullanılabilir. HDS, H.264 veya VP6 video ve AAC veya MP3 ses kullanabilir (Leadtools, 2019).

2.5.5 MPEG HTTP Üzerinden Dinamik Uyarlamalı Akış (MPEG-DASH)

MPEG DASH (HTTP üzerinden Dinamik Uyarlamalı Akış), geleneksel HTTP web sunucularından gönderilen Internet üzerinden yüksek kalitede ses/video içeriği akışını sağlayan MPEG tarafından sunulan bir bit hızı akış tekniğidir. Çeşitli sunucular ve cihazlar arasında birlikte çalışabilirlik sağlar. Web üzerinden video akışına olan talep gün geçtikçe artmaktadır. Ancak Apple aygıtları, Android aygıtları ve diğer tüm aygıtlarda tüm formatları ve oynatımları destekleyen ortak bir video akışı yoktur. MPEG-DASH, çeşitli sunucular ve aygıtlar arasında birlikte çalışabilirliği ele almayı amaçlayan bir standarttır (Synopi, 2019). Cep telefonları, tabletler, PC'ler, TV'ler, dizüstü bilgisayarlar, set üstü kutuları, oyun konsolları gibi çok çeşitli cihazları destekleyen ilk uluslararası standardize edilmiş HTTP

tabanlı uyarlamalı bit hızı yayın protokolüdür (Synopi, 2019). Bu protokol yakın zamanda bir ISO standardı olarak kabul edilmiştir. MPEG-DASH, isteğe bağlı, canlı ve zaman kaydırma 4'ü görüntülemeyi destekleyen uyarlamalı akış için Microsoft-LSS'ye benzer bir yapı tanımlar ancak XML tabanlı bir bildirim dosyası tanımlayarak dosya biçimlerinde değişiklikler önerir. MPEG-DASH medya sunumu kavramını tanıtmıştır (Luciano Rubio Romero, 2011).

Apple HLS, Microsoft Smooth Streaming, Adobe HDS vb. gibi önceki uyarlanabilir akış teknolojileri, şirketten bağımsız akış sunucularının ve oynatma istemcilerinin sınırlı desteği olan satıcılar tarafından piyasaya sürüldü. Böyle bir satıcıya bağlı durum istenmediğinden, standardizasyon kuruluşları 2012 yılında MPEG-DASH'in onaylanmasıyla sonuçlanan bir uyumlaştırma süreci başlattı.(Bitmovin, dynamic-adaptive-streaming-http-mpeg-dash, 2019)



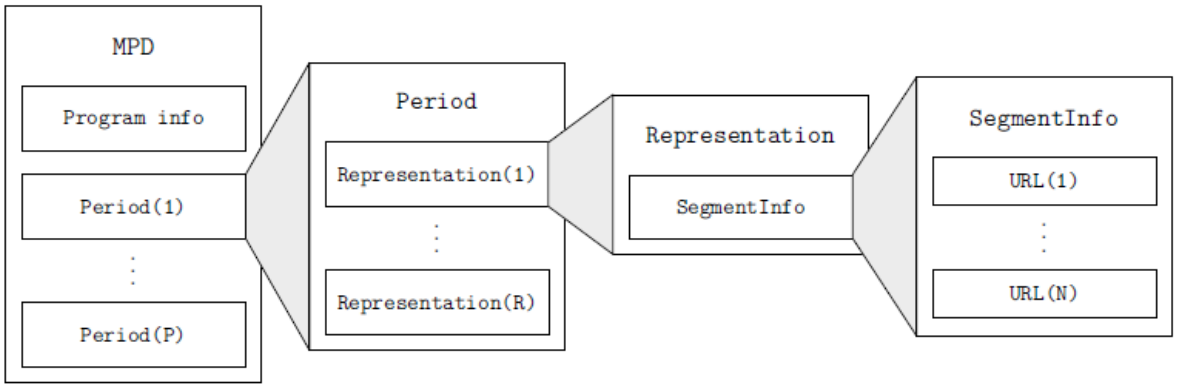
Şekil 2.6: MPEG-DASH Standardizasyonu (Bitmovin)

MPEG-DASH medya sunumu kavramını sunmaktadır. Bir medya sunumu, yapılandırılmış video/ses içeriği koleksiyonudur (Luciano Rubio Romero, 2011):

- Bir medya sunumu, ardışık olan ve çakışmayan bir veya daha fazla periyot dizisinden oluşur.
- Her dönem aynı medya içeriğinin bir veya daha fazla temsilinden oluşur. Dönemler, medya sunumunun başlangıcına göre belirlenmiş bir başlama zamanına sahiptir.
- Her gösterim, bant genişliği, kodlama ve çözünürlük gibi çeşitli parametrelerden oluşan bir video kalitesi profilini belirtir. Temsiller, URL'ler tarafından temsil edilen bir veya daha fazla segment içerir.
- Segmentler, gerçek video içeriğinin fragmanlarını içerir.

MPEG-DASH video sunucusu biraz önce bahsedilen MPD dosyasını barındırır. Bu dosya XML formatındadır. XML içerisinde video'nun segmentlerinin tümü belirtilmektedir. Segmentleri ifade eden girdiler bant genişliği (kalite), çözünürlük, periyot, url ve codec gibi bilgileri içermektedir. İstemci bu dosyayı ayıklayarak segment listesini oluşturur ve ağın

bant genişliğine göre adaptasyon yaparak mevcut durum için en uygun segmentleri listeden seçer ve indirir. Bu indirilen segmentler oynatıcı kuyruğuna eklenir ve izleyiciye sırayla gösterilir. Ağın durumuna göre kaliteler arasında geçişler yapılabilir.



Şekil 2.7: MPD Dosyasının Yapısı (Luciano Rubio Romero, 2011)

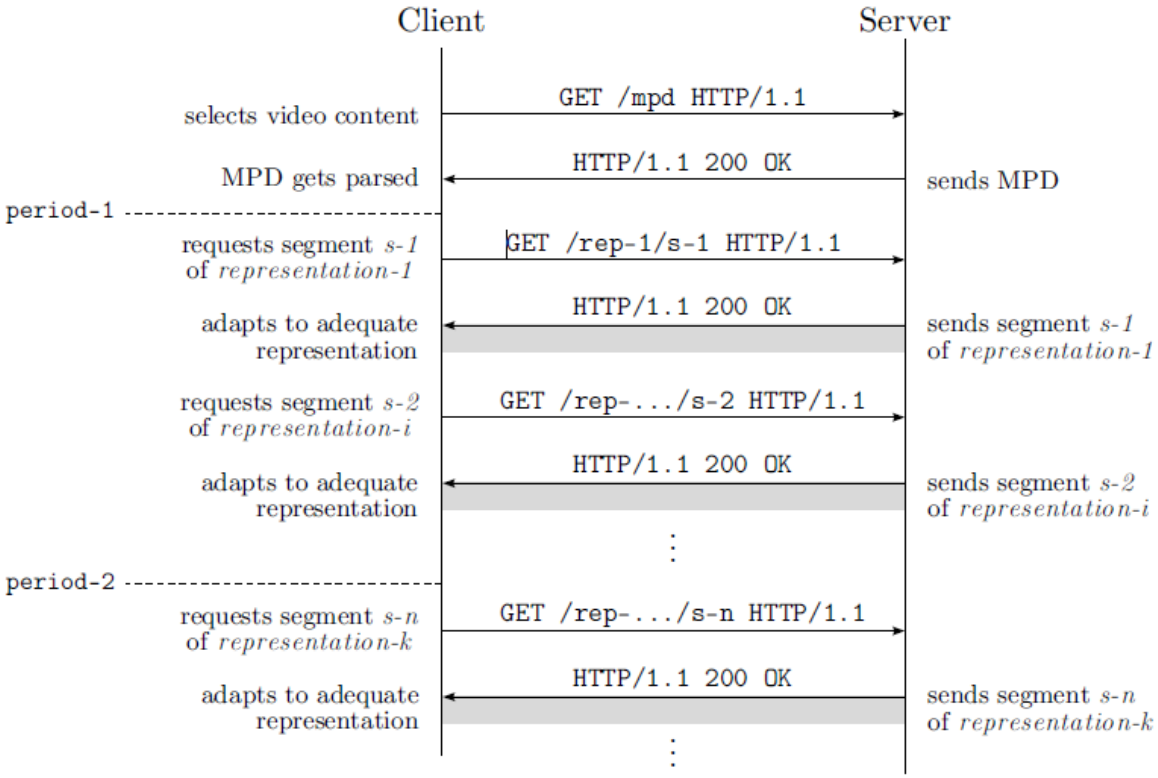
Tablo 2.1: Örnek MPD Dosyası (Luciano Rubio Romero, 2011)

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD minBufferTime="PT10S">
<Period start="PT0S">
<Representation mimeType="video/3gpp; codecs=263, samr" bandwidth="256000" id="256">
<SegmentInfo duration="PT10S" baseURL="rep1/">
<InitialisationSegmentURL sourceURL="seg-init.3gp"/>
<Url sourceURL="seg-1.3gp"/>
<Url sourceURL="seg-2.3gp"/>
<Url sourceURL="seg-3.3gp"/>
</SegmentInfo>
</Representation>
<Representation mimeType="video/3gpp; codecs=mp4v.20.9" bandwidth="128000" id="128">
<SegmentInfo duration="PT10S" baseURL="rep2/">
<InitialisationSegmentURL sourceURL="seg-init.3gp"/>
<Url sourceURL="seg-1.3gp"/>
<Url sourceURL="seg-2.3gp"/>
<Url sourceURL="seg-3.3gp"/>
</SegmentInfo>
</Representation>
</Period>
</MPD>
```

MPEG-DASH protokolü, MPD'nin (Media Presentation Description) sözdizimini ve semantiğini, bölümlerin formatını ve dağıtım protokolünü (HTTP) belirtir. Farklı yapılandırma hizmetlerini uygulamak için esnek yapılandırmalara izin verir. Aşağıdaki parametreler esnek bir şekilde seçilebilir: (1) bölümlerin büyüklüğü ve süresi (bunlar her temsil için ayrı ayrı seçilebilir), (2) temsil sayısı ve (3) her gösterimin profili (bit hızı, CODEC'ler, konteyner formatı, vb.) (Luciano Rubio Romero, 2011).

İstemcinin davranışına ilişkin olarak esnek bir şekilde: (1) segmentlerin ne zaman ve nasıl indirileceğine karar verir, (2) uygun gösterimi seçer, (3) anahtar gösterimleri değiştirir

ve (4) sadece HTTP yoluyla değil, başka yollarla da alınabilen MPD dosyasının aktarımını seçer (Luciano Rubio Romero, 2011).



Şekil 2.8: MPEG-DASH İstemci ve Sunucu Arasındaki Akış (Luciano Rubio Romero, 2011)

MPEG-DASH'ın Temel Hedefleri ve Yararları (Bitmovin, dynamic-adaptive-streaming-http-mpeg-dash, 2019):

- Video sırasında başlangıç gecikmelerinin ve tamponlamanın/duraklamanın azaltılması
- İstemcinin bant genişliği adaptasyonunu sağlaması
- En yüksek ölçeklenebilirlik ve esneklik sağlayan istemci tabanlı akış mantığı
- Mevcut ve uygun maliyetli HTTP tabanlı CDN'lerin, proxy'lerin, önbelleklerin kullanımı
- HTTP kullanımıyla NAT'lerin ve güvenlik duvarlarının atlanması
- Ortak şifreleme, aynı dosyadan birden fazla eşzamanlı DRM şemasının sinyalizasyonu, teslimatı ve kullanımı
- Basit birleştirme ve (hedefli) reklam ekleme

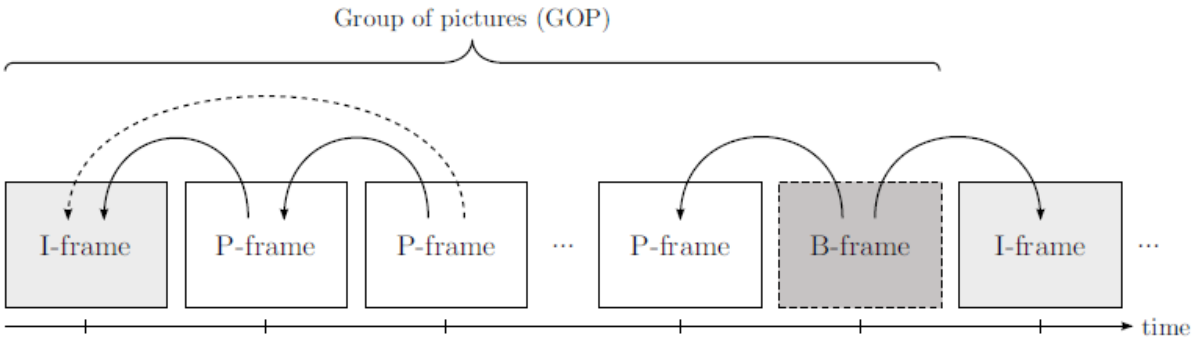
2.6 Video CODEC'leri

Bir video codec bileşeni, dijital videoyu sıkıştıran veya açan bir elektronik devre veya yazılımdır. Ham (sıkıştırılmamış) dijital videoyu sıkıştırılmış bir biçime veya tam tersi olarak dönüştürür. Video sıkıştırma bağlamında, "codec", "kodlayıcı" ve "dekoder" 'in

birleşimidir; yalnızca sıkıştıran cihaz tipik olarak kodlayıcı olarak adlandırılır ve sadece dekompresyon yapan cihaz bir kod çözücüdür. Eski kodeklerin algoritma temelinde Huffman algoritması yatar. MPEG-4, MPEG-2 ve wavelet gibi. Ancak yeni nesil H.264 kodeğinde Huffman algoritmasıyla beraber yeni Arithmetic teknolojisi de kullanılır (Turksan, 2019). Bu bölümde, video kodlayıcıların ve kod çözücünün bu tezin okuyucusuyla ilgili bir dizi yönü açıklanmaktadır.

2.6.1 Video çerçeveleri

Sıkıştırılmış video standartları yalnızca, ana kareler, içi kareler veya basitçe I-kareler olarak bilinen belirli kareler için tam kare verilerini kodlar. Bir anahtar kareyi takip eden kareler, öngörülen kareler veya P kareler, sadece önceki kareyle olan farklılıklar dikkate alınarak kodlanır. Bu sonraki kareleri kodlamak için daha az veriye ihtiyaç duyulur. Kare bilgisi hızla değişen videolar, yavaşça değişen görsel sahneden daha fazla ana kare gerektirir. Birkaç kare arasındaki ilişkinin bir örneği şekil 2.9'da gösterilmiştir.



Şekil 2.9: Çerçeveler arasındaki ilişki

İki yönlü kodlama, Bi-tahminli çerçeveler (B-çerçeveleri) aracılığıyla da mümkündür. B-çerçeveleri daha iyi sıkıştırma elde etmek için hem önceki hem de sonraki çerçeve farklılıklarını göz önünde bulundurur. Bir Resim Grubu (GOP), bir I-karesinin ardından birkaç P-kareden ve isteğe bağlı olarak B-karelerden oluşur. GOP boyutunu düşürmek (anahtar kare aralığı) fayda sağlayabilir: daha sık kullanılan anahtar karelerin kullanılması kayıplı bir ortamda akış yaparken bozulmayı azaltmaya yardımcı olur. Ancak düşük GOP boyutu, medya çerçeveleri boyutunu artırır, çünkü anahtar kareler öngörücü karelerden daha fazla bit içerir.

2.6.2 Kod çözme ve sunum zaman damgaları

H.263, diğer birçok uygulamada yaygın olarak kullanılmasına rağmen, video konferans görüşmeleri için düşük bit hızında sıkıştırılmış formatta tasarlanmış bir video kodek standardıdır. Mobil ağlar üzerinden video akışı için standart olarak yaygın bir şekilde benimsenmiştir. 1996 yılında ITU-T Video Kodlama Uzmanları Grubu (VCEG) tarafından geliştirilmiştir. H.263, Flash video uygulamalarında desteklenmiştir ve YouTube veya Vimeo gibi internet üzerinden talep üzerine servisler tarafından yaygın olarak

kullanılmaktadır. H.263 bit hızları 24 kb/s ile 64 kb/s arasında değişmektedir. Video ücretsiz LGPL lisanslı libavcodec kütüphanesi (FFmpeg projesinin bir parçası) ile bu formata kodlanabilir ve kodu çözülebilir. H.263, ISO H.261 standardının bir ilerlemesidir, temel olarak MPEG'in geliştirilmesi için bir başlangıç noktası olarak kullanılmıştır (daha yüksek veri hızları için optimize edilmiştir) (H263l.com, 2019)

2.6.3 H.264/MPEG-4 AVC

H.264/MPEG-4-Part10 veya Gelişmiş Video Kodlaması (AVC), hem düşük hem de yüksek kaliteli videoları başarıyla kodlayabilir. H.264 standardı, MPEG-4-Part10 olarak da bilinir ve MPEG-2 ve MPEG-4 gibi daha önceki standartların halefidir. H.264, dört kata kadar daha büyük bir çerçeveyle MPEG-4 kalitesini sunar. Ayrıca orijinal bant genişliklerinin üçte biri kadar küçük bir bant genişliğinde MPEG-2 kalitesini sağlayabilir (H264encoder, 2019).

H.264, Blu-ray diskleri için desteklenen CODEC'lerden biridir. H.264, 2003 yılında ISO/IEC MPEG ile birlikte ITU-Video Kodlama Uzman Grubu tarafından geliştirilmiştir. Adobe'nin Flash Player ve Microsoft'un Silverlight'ında desteklenmektedir. Bu nedenle, Vimeo, YouTube ve Apple iTunes Store gibi çoklu akışlı İnternet kaynakları H.264 standardını izler. H.264, çoklu uygulama türlerine yönelik on yedi profili belirtir. Kısıtlanmış Taban Çizgisi Profili (CBP) en temel olanıdır ve öncelikle düşük maliyetli uygulamalar için bu profil en çok video konferans ve mobil uygulamalarda kullanılır; bunu, artan karmaşıklık düzeninde Temel Çizgisi Profili (BP) ve Ana Profil (MP) takip eder. MP, DVB standardında tanımlandığı şekilde MPEG-4 formatını kullanan standart tanımlı dijital TV yayınları için kullanılır. BP, Sınırlı Temel Profil'de desteklenen tüm özellikleri ve ayrıca kayıp sağlamlığı için (veya düşük gecikmeli çok noktalı video akışı birleştirme gibi diğer amaçlar için) kullanılabilecek üç ek özelliği içerir (Techex, h264, 2019). Ek olarak, CBP, BP VE MP Android'in yerel medya çerçevesi tarafından desteklenen H.264 profilleridir. Tablo 2.2, bu üç profil arasındaki temel farklılıkları özetlemektedir. H.264 standardına eklenen en yeni özelliklerden biri Ölçeklenebilir Video Kodlamasıdır (SVC). SVC, bir video akışının birden fazla çözünürlük, kalite ve kare hızı katmanlarına bölünmesini sağlayan bir tekniktir. SVC, H.264/MPEG-4 AVC video sıkıştırma standardının Annex G uzantısıdır (Techex, h264-svc-scalable-video-coding, 2019). SVC kodekleri, resmin ayrılmasını önleyen bir resmin kare hızını, çözünürlüğünü veya bant genişliği tüketimini azaltmak için bu bit akış alt gruplarını veya paketlerini düşürerek alt ağ bağlantılarına uyum sağlar (Searchunifiedcommunications, scalable-video-coding-SVC, 2019). Ek olarak, Çoklu Görüntülü Video Kodlama (MVC 3D olarak da bilinir), video sıkıştırma için tek bir video akışında aynı anda birden fazla kamera açısından aynı anda yakalanan video dizilerinin etkin şekilde kodlanmasını sağlayan stereoskopik bir video kodlama standardıdır. (Wikipedia, 2019)

Tablo 2.2: CBP, BP ve MP Profilleri Arasındaki Temel Farklar

Feature	CBP	BP	MP
Android support	Yes	Yes	No
Flexible macro-block ordering (FMO)	No	Yes	No
Arbitrary slice ordering (ASO)	No	Yes	No
Redundant slices (RS)	No	Yes	No
B-frames	No	No	Yes
CABAC entropy coding	No	No	Yes

H.264 avantajları (Turksan, h264, 2019):

- MPEG4 part 2 den iki kat yüksek verimlilik
- MPEG2 den 3 kat daha düşük dosya boyutu
- Hızlı indirme süresi
- Yüksek kaliteli video
- Yüksek hareket kompanzasyonu. MPEG4 te olduğu gibi yüksek harekette bulanıklaşma olmaz
- Mobil ağda hata eğilimini sezme

2.7 Konteyner Formatları

Bir kapsayıcı veya sarıcı formatı, teknik özellikleri farklı veri ve meta veri öğelerinin bir bilgisayar dosyasında nasıl bir arada bulunduğunu açıklayan bir meta dosya biçimidir (Wikipedia, 2019). Ayrıca, konteyner formatları, konteyner formatının hangi kodeki kullanacağını belirlemediklerinden, genellikle video, ses ve diğer verilerin konteynerde nasıl depolandığını tanımlar (Webopedia, Container Format, 2019). Konteynerler, farklı video türlerini, örneğin video akışlarını, altyazıları ve hatta meta-veri bilgilerini serpiştirmek için kullanılır. Çok çeşitli konteyner formatları geliştirilmiştir, farklı özellikler ve sınırlamalar sunar. En önemli multimedya konteynerleri aşağıda kısaca açıklanmıştır.

- **MP4:** Genel olarak MPEG-4 Bölüm 14 veya MPEG-4 AVC (Gelişmiş Video Kodlama) olarak da bilinen MP4 dosya formatı, ses ve video dosyalarını dijital olarak saklamak için kullanılan bir multimedya dosya formatıdır (Bytescout, 2019). 2003'te sonlandırılmış Apple QuickTime MOV formatına dayanır. Hemen hemen her türlü medya verisini destekler. Tipik olarak bir MP4 kabı, sırasıyla H.264 ve AAC ile kodlanmış video ve ses akışlarını içerir. Bu format ayrıca altyazıların ve hareketsiz görüntülerin saklanmasına izin verir (Igi-global, 2019). MP4 dosya formatı, İnternet üzerinden video akışı için de kullanılır. MP4 dosya formatı temel olarak dijital olarak

kodlanmış ses ve video dosyalarını tutan bir kapsayıcıdır. MP4 dosya formatı yalnızca bir resmi dosya uzantısına sahiptir.

- **3GP:** Bir 3GP dosyası, 3. Nesil Ortaklık Projesi (3GPP) tarafından geliştirilen bir ses ve video kabı formatında kaydedilmiş bir multimedya dosyasıdır (Fileinfo, 2019). 3GP video konteyner formatı, disk alanından, bant genişliğinden ve veri kullanımından tasarruf etmek amacıyla geliştirildi. Bu yüzden sık sık mobil cihazlardan yaratıldıkları ve bunlar arasında aktarıldıklarını gördüler. 3GP, Multimedya Mesajlaşma Servisi (MMS) ve Multimedya Yayın Multicast Servisleri (MBMS) kullanılarak gönderilen ortam dosyaları için gerekli standart formattır. 3G cep telefonlarında kullanılır ancak bazı 2G ve 4G telefonlarda da çalınabilir. 3GP, MPEG-4 Kısım 12'nin bir uzantısı olarak tanımlanır. 3GP normal olarak, MPEG-4 Kısım 2, H.263 veya H.264 ile kodlanmış video akışlarını ve AAC (LC profili) veya HE-AAC ile ses akışlarını depolar.

- **MPEG Aktarım Akışı:** MPEG aktarım akışı (aktarım akışı, MPEG-TS, MTS veya TS), ses, video ve Program ve Sistem Bilgisi Protokolü (PSIP) verilerinin iletimi ve depolanması için standart bir dijital konteyner formatıdır. DVB ve IPTV gibi yayın sistemlerinde kullanılır (Medium, Mpeg Transform, 2019). MPEG TS, veri bağlantısı katmanını ve ayrıca hizmet bilgi verisi katmanlarını (bir OSI model bakış açısında,MPEG TS'in veri bağlantısından taşıma katmanları veri bağlantısını kapsar) tanımlar (Thinkincredible, Mpeg Transport Stream Introduction, 2019).

- **Ogg:** Ogg (.ogg), Xiph.Org Foundation tarafından geliştirilen herkesin kullanması için ücretsiz ve açık bir konteyner formatıdır. Ogg genellikle Theora ve Vorbis CODEC'lerini birleştirir. Ayrıca, Ogg bir multimedya konteyner biçimi ve Xiph.org multimedya kodekleri için yerel dosya ve akış biçimidir (Xiph, ogg, 2019).

- **WebM:** Web için tasarlanmış, Matroska konteynerine dayanarak yakın zamanda yayımlanan açık, telifsiz bir konteyner formatıdır. WebM, Web içeriği için en uygun biçimlerden biri olarak kabul edilir. Patentsiz ve açık olması nedeniyle, web sayfalarının baskın biçimlendirme dili olan HTML 5, CODEC agnostik olarak tanımlanır. WebM, dosya konteyner yapısını, video ve ses formatlarını tanımlar. WebM dosyaları, VP8 veya VP9 video kodekleriyle sıkıştırılmış video akışlarından ve Vorbis veya Opus ses kodekleriyle sıkıştırılmış ses akışlarından oluşur (Webmproject, 2019). Mozilla Firefox, Opera ve Google Chrome web tarayıcıları, WebM dosyalarının video oynatılmasını destekler, ancak Internet Explorer 9 ve Safari, WebM ortamının oynatılması için genellikle üçüncü taraf yardımına (QuickTime gibi) ihtiyaç duyar. YouTube, WebM dosyalarını HTML5 oynatıcı denemelerinin bir parçası olarak kullanır (Online-convert, Webm, 2019).

2.8 Video Kalite Seviyeleri

Video ve ses içeriği, farklı son kullanıcı tiplerine yetecek şekilde birden fazla gösterimde (kalite seviyeleri) sunulabilir. Son kullanıcıların diğer sınırlamaların yanı sıra, ağ yetenekleri, ekran çözünürlükleri ve desteklenen medya formatları bakımından çok çeşitli kısıtlamalardan etkilendiği bilinen bir gerçektir. Sunucunun tarafında ne kadar çok temsil

edildiyse, medya içeriği o kadar çok çeşitli alternatif sürümlere sunulduğu için daha iyi ayrıntı düzeyi sistemi karakterize eder. Bununla birlikte, birden fazla kalite seviyesinin oluşturulması, işlem süresi, depolama gereksinimleri ve CPU tüketimi açısından daha yüksek bir maliyete neden olur. Aşağıdaki kodlama parametreleri özellikle bir temsil seviyesi tanımlarken geçerlidir:

- Video bit hızı (kb/s): Bit hızı, saniye başına bit sayısıdır. Sembolü bit/s'dir. Genellikle video ve ses dosyalarının boyutunu ve kalitesini belirler: bit hızı ne kadar yüksek olursa kalite o kadar iyi olur ve dosya boyutu da artar (Filmora, What is video bitrate, 2019).
- Kare hızı (fps): Bir süre aralığında görüntüyü oluşturan kare sayısıdır. Genel olarak saniye bazında ölçülür. Saniyedeki kare sayısı (FPS), ekran cihazının performansını ölçen bir birimdir. Her saniye gerçekleşen ekranın tam tarama sayısından oluşur. Bu, ekrandaki görüntünün her saniye yenilenme sayısı veya bir görüntüleme cihazının çerçeveler adı verilen benzersiz sıralı görüntüler üretme hızıdır (Techopedia, Frames-per-second-fps, 2019).
- Ses bit hızı (kb/s): Ses akışındaki bilgi hızı.
- Ses kanalları: stereo (2) veya mono (1).
- Örnekleme oranı (Hz): Örnekleme hızı veya örnekleme frekansı, ayrık veya dijital bir sinyal oluşturmak için sürekli bir sinyalden alınan saniye başına (veya diğer birim başına) örnek sayısını tanımlar.
- GOP boyutu: Anahtar kareyi izleyen kare sayısıdır.
- Çözünürlük (piksel): Video görüntüsünün boyutunu belirtir (çerçeve). Çerçeveyi oluşturan piksel sayılarıyla ölçülür. Çözünürlük ne kadar yüksek olursa, görüntü kalitesi de o kadar yüksek olur ve görüntüde daha fazla ayrıntı bulunur.

GOP boyutu dışında, bu parametrelerden herhangi birinin arttırılması daha yüksek kalitede bir ses veya video çıktısına yol açar, sonuç olarak daha büyük bir dosya boyutu oluşturur (daha fazla bit iletilmesi gerekir).

2.9 Talep Üzerine Video (VOD) ve Canlı Yayın

Akış tekniklerini kullanmanın iki farklı yolu vardır. İlki, istek üzerine video (VOD), kullanıcılar daha önce kaydedilmiş ve sıkıştırılmış ve bir sunucuda depolanan medya dosyalarını ister. VOD, temel olarak, canlı yayının tam tersidir ve kullanıcılara videoları uygun bir zamanda ve İnternet'e bağlı herhangi bir cihazdan izleme fırsatı verir. Avantajları arasında; bağlanabilirlik; VOD içeriğine her zaman erişilebilir, ne zaman ve nerede olursa olsun içerik izlenebilir, çeşitlilik; videoları kaydeder, düzenler ve önizleme imkanı sunar, ayrıca videolar olabildiğince geliştirebilir ve farklı efektler eklenebilir, uygunluk; uygulaması daha kolaydır ve kullanıcılar için daha iyi video kalitesi sunar. Bugün bu teknik çok popüler hale geldi ve YouTube isteğe bağlı akış sunan en popüler web sitesi oldu. İkincisi, ortamın üretildiği, sıkıştırıldığı ve anında iletildiği sınırsız bir aktarımı mümkün kılan canlı yayındır. Canlı yayın durumunda, eşzamanlı bir kayıt olabilir veya olmayabilir (daha sonra talep üzerine iletilebilir). Canlı yayının bazı avantajları: gerçek zamanlı

deneyim; örnek olarak markanızın canlı yayınladığı bir etkinliği yaptığınızda, izleyicilerinizle etkileşime girersiniz, etkileşim; markalarınızın canlı şekilde yayınlanması müşterilerde heyecan duygusu yaratır ve özel hissettirir, maliyet azaltma; bazı şirketler, yeni çalışanların işe alımlarında, eğitim programlarının bir parçası olarak canlı akışı kullanıyor. Herkes aynı eğitim ve ürün bilgilerini aynı anda alır, böylece şirketin bütçesinden tasarruf edilir.

Canlı yayın ve VOD (İstek Üzerine Video), farklı marka tanıma avantajları, daha fazla müşteri havuzu ve takipçilerin katılımını sağlayabilir. Her iki akış tekniği, kullanıcıya duraklatma, durdurma ve geri sarma gibi temel video kontrol işlevleri sunabilir. Ek olarak, isteğe bağlı akış için hızlı ileri bir komut verme olasılığı olabilir. Hızlı ileri sarmanın yalnızca medya dosyaları saklandığında mümkün olduğunu, dolayısıyla gelecekteki içeriğin bilindiğini unutmayın. Elbette, can yayın sırasında kullanıcı oynatmayı duraklattıysa, sistemin hızlı ileri alma komutunu gerçekleştirmesi de mümkündür, ancak bu, içeriğin yakın zamanda oluşturulan bölümüne ilerlemekle sınırlı olacaktır.

2.10 Google'ın Android işletim sistemi

Şekil 2.10: Android Resmi Logosu

Android, mobil cihazlar için özel olarak tasarlanmış Linux çekirdeğinin ve diğer sürümlerinin değiştirilmiş bir sürümüne dayanan bir işletim sistemidir. Open Handset Alliance (OHA) geliştirilmesinde ve yayınlanmasında işbirliği yapmış olmasına rağmen, esas olarak Google Inc. tarafından geliştirilmekte ve desteklenmektedir. Android, kullanıcılara akıllı, doğal telefon kullanımı için "doğrudan manipülasyon" arayüzü sunar. Android'in açık kaynak kodu ve izin verilen lisansı, cihaz üreticilerinin, kablosuz taşıyıcıların ve geliştiricilerin platformu serbestçe ayarlamalarını, özelleştirmelerini ve dağıtmalarını sağlar (WorsStream, Android OS, Ekim 2019).

Android, Linux çekirdeğinin değiştirilmiş bir sürümüne dayanmaktadır ve uygulamaları normal olarak Java programlama dilinde geliştirilmektedir. Ancak, Android resmi Java Sanal Makinesi'ni (JVM) kabul etmedi, bu da Java Byte kodunun doğrudan çalıştırılamayacağı anlamına geliyor. Bunun yerine, uygulamalar Android için özel olarak tasarlanmış JVM tabanlı bir sanal makine olan Dalvik Sanal Makine (DVM) üzerinde çalışır. DVM, genellikle CPU performansı ve bellek sınırlamaları olan mobil cihazlar için optimize edilmiştir. Ayrıca, DVM pili daha verimli kullanır. Google DVM'den sonra doğrudan doğal manpülasyonu da destekleyen daha özelliştirilmiş bir VM olan ART'yi (Android Run Time) çıkarmıştır. Uygulamalar genellikle Google'ın resmi çevrimiçi mağazası olan Android Market aracılığıyla yayınlanır.

2.10.1 Android'de desteklenen medya biçimleri

Android, H.263 ve H.264 dahil olmak üzere birçok multimedya formatını ve CODEC'lerini destekler. Medya oynatımı için, sadece kod çözme yetenekleri önemlidir (kodlama genellikle kayıt amacıyla kullanılır).

2.10.2 Android'de desteklenen HTTP üzerinden uyarlanabilir protokoller

Android'in medya çerçevesi, RTP ve RTSP'den akışı doğal olarak destekliyor. Ancak maalesef Android sürümlerinin çoğu daha önce belirtilen HTTP üzerinden uyarlanan protokollerin hiçbirini desteklememektedir. Yalnızca Honeycomb'dan sonraki versiyonlar yerel olarak Apple-HLS'ye sahiptir. Bu yüksek lisans projesindeki geliştirmeler aşamasında son MPEG-DASH standardını destekleyen Google'ın kendi bünyesinde geliştirdiği Exoplayer adlı bir player incelenmiştir. Bu bölüm AppleHLS, Microsoft-LSS, Adobe-HDS ve MPEG-DASH ile ilgili mevcut uyumlulukları incelemektedir.

2.10.3 Exoplayer[2] Medya Oynatıcısı

Android için MPEG-DASH desteği de dahil pek çok farklı protokolün desteğini sağlayan bir multimedya player olan ExoPlayer kullanılmıştır. Exoplayer, Android için bir uygulama seviyesi medya oynatıcısıdır. Hem yerel hem de internet üzerinden ses ve video oynatmak için Android'in MediaPlayer API'sine bir alternatif sunar. ExoPlayer, bölüm 2.5'te verilen HLS, DASH ve Smooth Streaming uyarlamalı oynatımları dahil olmak üzere şu anda Android'in MediaPlayer API'sı tarafından desteklenmeyen özellikleri desteklemektedir. MediaPlayer API'sinden farklı olarak ExoPlayer'ın özelleştirilmesi ve genişletilmesi kolaydır ve Play Store uygulama güncellemeleriyle güncellenebilir (Google, Kasım 2019).

2.11 Farklı HTTP tabanlı uyarlamalı çözümler arasında karşılaştırma

Tablo 2.3, en alakalı HTTP tabanlı uyarlamalı akış çözümlerinin ana özelliklerini özetlemektedir: Microsoft-LSS, Apple-HLS ve MPEG-DASH.

[2] Exoplayer'ın kaynak kodları Github'ta Google tarafından paylaşılmıştır: https://github.com/google/ExoPlayer

Tablo 2.3: LSS, HLS ve MPEG-DASH Arasındaki Farklar (Luciano Rubio Romero, 2011)

Feature	Microsoft-LSS	Apple-HLS	MPEG-DASH
Specification	Proprietary	Proprietary	Standard
Video on demand	Yes	Yes	Yes
Live	Yes	Yes	Yes
Delivery protocol	HTTP	HTTP	HTTP
Origin server	MS IIS	HTTP	HTTP
Media container	MP4	MPEG-TS	3GP or MP4
Supported video CODECs	Agnostic	H.264	Agnostic
Recommended segment duration (s)	2	10	flexible
End-to-end latency (s) (variable, depending on the size of segments)	> 1.5	30	> 2
File type on server	Contiguous	Fragmented	Both

Android için bir canlı yayın hizmetini yerine getirmek için, işletim sisteminin tüm sınırlamalarının yanı sıra uyumlu bir sunucuyu dağıtma olasılığının da göz önünde bulundurulması gerekir. Açıkça aşağıdakileri göz önünde bulundurduk:

• Adobe'nin ve Microsoft'un çözümleri özeldir ve her ikisi de özel sunucular gerektirir. Bu tür yaklaşımlar maliyeti arttırır ve ortaya çıkan hizmetin açıklığını azaltır.

• Apple'ın HTTP Canlı Yayın (HLS) protokolü, kullanıcılara video akışı sağlamak için tercih edilen yöntem haline geldi. HLS, video ve ses verilerini medya sunucularından izleyicilerin ekranlarına taşımak için kullanılan, uyarlanabilir bir HTTP tabanlı protokoldür (Wowza, Ekim 2019). Apple-HLS bir IETF standardı olmayı amaçlamaktadır, ancak spesifikasyonu (CODEC ve segmentlerin konteyner formatı ile ilgili) Android için basit bir uygulama düşünecek kadar katıdır. Her ne kadar H.264 Android'de tam olarak desteklenen bir CODEC olsa da, bir konteyner formatı olarak MPEG-TS yalnızca Android sürüm 3.0 (kod adı Honeycomb) ve sonrasında bulunur. Bu nedenle, Apple-HLS'yi desteklemek için bir dosya dönüştürme gerekir.

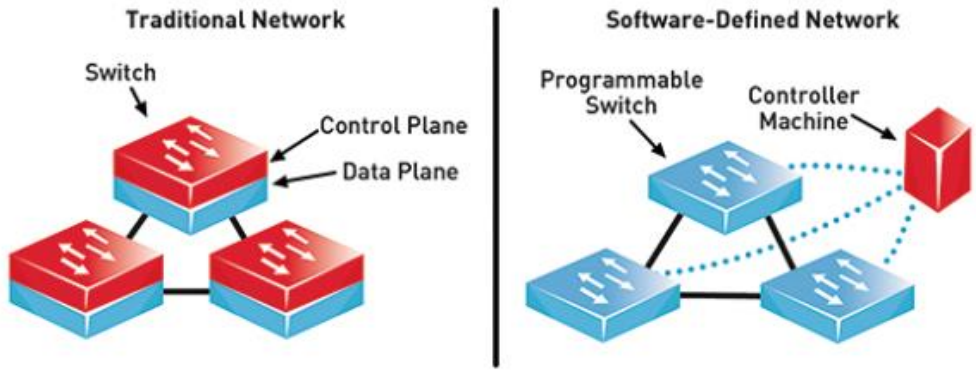
• MPEG-DASH, Endüstride yerleşik bir cihazla birlikte uygulanabilecek yeterince mevcut olan yeni bir adaptif HTTP yayını standardıdır. Ayrıca, HDS ve HLS'ye benzer şekilde, MPEG DASH bir video dağıtım teknolojisi standardıdır; en düşük kayıp ve en az olası tamponlama ile en iyi kalitede içerik sunmak amacıyla gerçek zamanlı olarak internet üzerinden canlı yayın yayınlama yöntemi (Dacast, MPEG-DASH, Ekim 2019).

HTTP kullanılarak yayınlanan videolar teknik olarak "akışlar" değildir. Aksine, düzenli web sunucuları aracılığıyla gönderilen aşamalı indirmelerdir. Endüstri bunlardan yana değişti çünkü HTTP tabanlı protokoller, bağlantı, yazılım veya cihaz ne olursa olsun, mümkün olan en iyi video kalitesini ve görüntüleyici deneyimini sunar. MPEG-DASH ve

Apple'ın HLS'leri en yaygın kullanılan HTTP tabanlı iki protokoldür (Wowza, HLS, Kasım 2019).

2.12 Yazılım Tanımlı Ağ Teknolojisi (SDN)

Şebeke kontrol düzleminin yönlendirme düzleminden fiziksel olarak ayrılması ve bir kontrol düzleminin birkaç cihazı kontrol ettiği yer. Yazılım Tanımlı Ağ İletişimi (SDN), dinamik, yönetilebilir, uygun maliyetli ve uyarlanabilir olan ve günümüz uygulamalarının yüksek bant genişliğine sahip dinamik doğası için ideal olan gelişmekte olan bir mimaridir (OpenNetworking Kasım 2019). Yazılım tanımlı ağ teknolojisi (SDN) , ağ performansını artırmak ve geleneksel ağ yönetimine göre daha fazla bulut bilişim gibi izlemesini izlemek için dinamik, programatik olarak etkin bir ağ yapılandırması sağlayan ağ yönetimine bir yaklaşımdır. Bu, operatörlerin temel ağ teknolojisine bakılmaksızın tüm ağı tutarlı ve bütünsel olarak yönetmelerine yardımcı olur (Ciena, SDN, Kasım 2019).

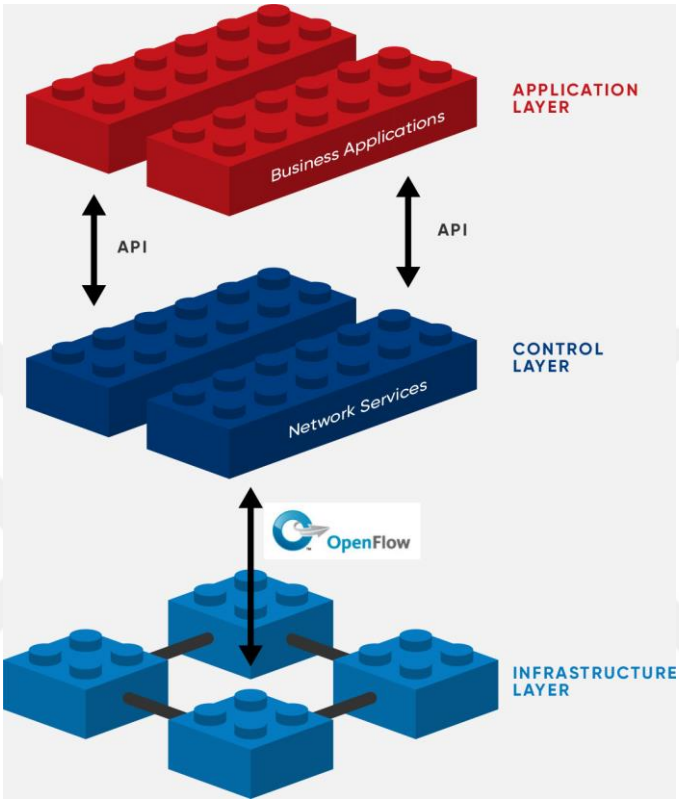


Şekil 2.11: Geleneksel Ağ ile SDN Ağları Arasındaki Fark (Commsbusiness, SDN, Kasım 2019)

SDN teknolojisinin bir organizasyon için fark yaratabileceği dört kritik alan vardır;

1. Ağ Programlama: SDN ağdaki fiziksel cihazların bağlantıları üzerindeki değişikliklerle ağın davranışını yazılımla kontrol edebilme özelliği sunar.
2. Mantıksal olarak merkezileştirme ve kontrol: SDN, ağ kaynaklarının akıllıca kontrol edilmesini ve yönetilmesini sağlayan mantıksal olarak merkezi ağ topolojileri üzerine kuruludur. Geleneksel ağ kontrol yöntemleri dağıtılmaktadır. Aygıtlar, ağın durumunun sınırlı farkındalığı ile özerk bir şekilde çalışır. Merkezi bir kontrol türü SDN tabanlı bir ağ sağlar, bant genişliği yönetimi, restorasyon, güvenlik ve politikalar oldukça akıllıdır ve optimize edilebilir. Tüm bunların toplamı ağın bütünsel bir yönetimini sağlar.
3. Ağın soyutlanması: SDN teknolojisinde çalışan hizmetler ve uygulamalar, ağ kontrolünden fiziksel bağlantı sağlayan temel teknolojilerden ve donanımdan soyutlanır. Uygulamalar, donanıma sıkıca bağlanmış yönetim arabirimleri yerine, API'ler aracılığıyla ağla etkileşime girer.
4. Açıklık: SDN mimarileri yeni bir açıklık çağına giriyor; kapalı kaynak kodlu özel satıcıların ürünleri ile birlikte çalışabilirliğini sağlamanın yanı sıra, tarafsız bir

ekosistemi de geliştiriyor. Açıklık SDN yaklaşımının temelinden gelir. Açık API'ler, bulut düzenlemesi, OSS/BSS, SaaS ve iş açısından kritik ağ bağlantılı uygulamalar dahil olmak üzere çok çeşitli uygulamaları destekler. Ek olarak, akıllı yazılımlar, OpenFlow gibi açık programlı arayüzlere sahip birden çok satıcının donanımını kontrol edebilir. Son olarak, SDN içinden akıllı şebeke servisleri ve uygulamaları ortak bir yazılım ortamında çalışabilir.



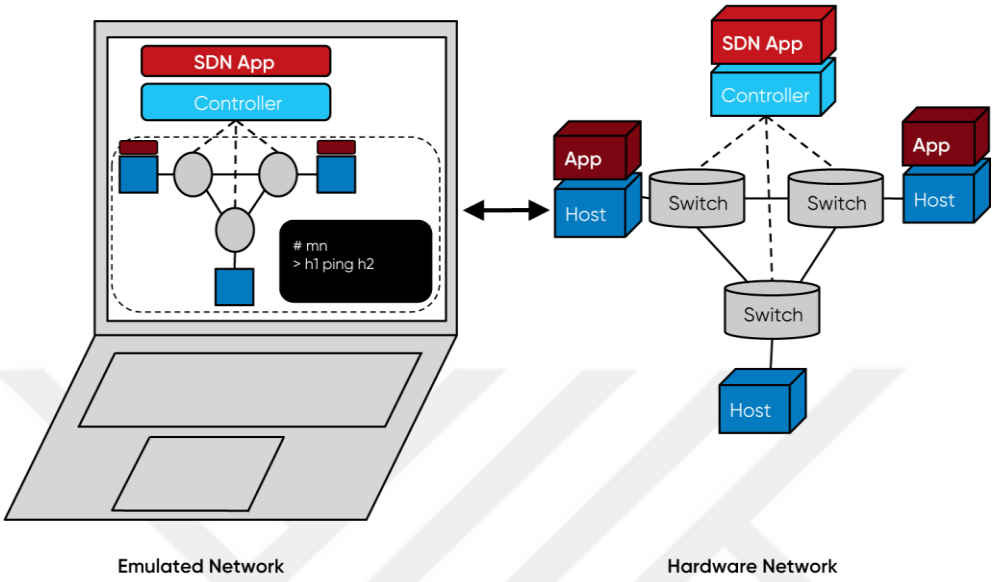
Şekil 2.12: SDN Katmanları (Opennetworking, Kasım 2019)

SDN teknolojisinin önemli bir avantajı, şebeke operatörlerinin SDN API'leri kullanan programlar yazma ve uygulamaların ağ davranışı üzerinde kontrol etmelerini sağlama yeteneğidir. SDN, kullanıcıların ağ tanıyan uygulamalar geliştirmelerine, ağ koşullarını akıllıca izlemelerine ve ağ yapılandırmasını gerektiğinde otomatik olarak uyarlamalarına olanak tanır (Sdx Central, Kasım 2019).

2.13 Mininet

Mininet, tek bir Linux çekirdeğinde tek bir komutla anahtarları, ana bilgisayarları ve SDN denetleyicilerini içeren sanal bir ağ başlatılmasını sağlayan ağ öykünme yazılımıdır (ResearchGate, Kasım 2019). Ayrıca, büyük ağları tek bir bilgisayara veya sınırlı kaynaklarla sağlanan sanal bir makineye dağıtmak için kullanılabilir. OpenFlow cihazı ve SDN kontrol cihazlarını kullanan ücretsiz bir açık kaynaklı yazılımdır. Mininet yazılım tanımlı ağlar (SDN) oluşturmanıza, test etmenize, gerçeklemenize olanak sağlayan açık kaynak kodlu bir projedir. Mininet ile hem interaktif REPL terminal üzerinden hem de Python API'si üzerinden SDN'ler oluşturmaya olanak sağlamaktadır. Switch ve uygulama

kodunu OS kernel üzerinde vm, cloud yada native pc üzerinde saniyeler içerisinde oluşturmaktadır. Mininet aktif olarak geliştirilir, desteklenir ve izin verilen BSD Açık Kaynak lisansı altında yayınlanır. Mininet, araştırma, geliştirme, öğrenme, prototip oluşturma, test etme, hata ayıklama ve bir dizüstü bilgisayarda veya başka bir bilgisayarda tam bir deneysel ağa sahip olmanın faydası olabilecek diğer işleri destekler.



Şekil 2.13: Mininet Aracının Yapısı (OpenNetworking, Kasım 2019)

Mininet;

• OpenFlow uygulamalarını geliştirmek için basit ve ucuz bir test ağı sağlar.

• Birden fazla eşzamanlı geliştiricinin aynı topoloji üzerinde bağımsız olarak çalışmasını sağlar.

• Tekrarlanabilir ve kolayca paketlenen sistem düzeyinde regresyon testlerini destekler.

• Fiziksel bir ağ bağlamaya gerek kalmadan karmaşık topoloji testi sağlar.

• Ağ çapında sınamaları hata ayıklamak veya çalıştırmak için topoloji ve OpenFlow uyumlu bir CLI içerir.

• İsteğe bağlı özel topolojileri destekler ve temel bir parametreli topolojiler kümesi içerir, programlama olmadan kutudan kullanılabilir, aynı zamanda ağ oluşturma ve deneme için basit ve genişletilebilir bir Python API sağlar.

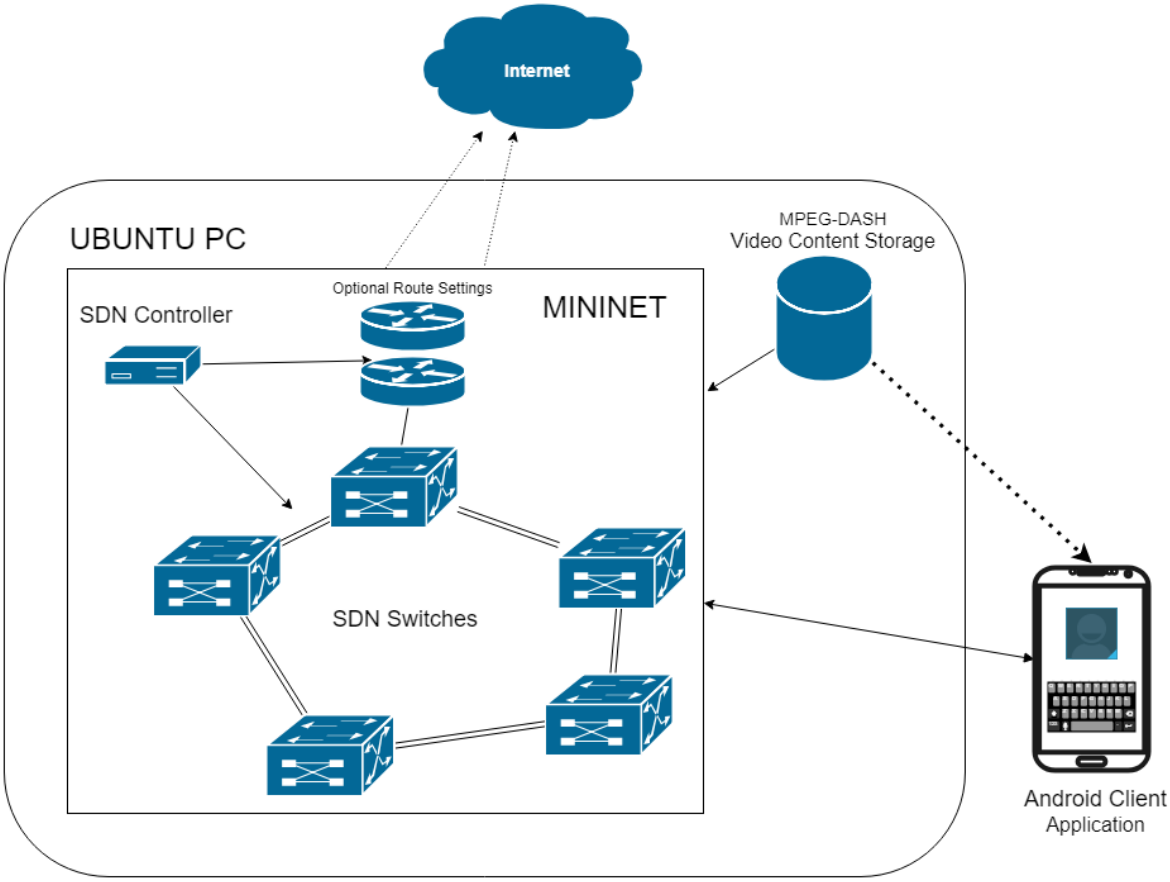
2.14 RYU

Ryu, bileşen tabanlı bir yazılım tanımlı ağ aracıdır. Ryu denetleyici, trafiğin nasıl yönetildiğini ve uyarlanmasını kolaylaştırarak ağın çevikliğini artırmak için tasarlanmış açık, yazılım tanımlı bir ağ (SDN) denetleyicisidir. Ryu, geliştiricilerin yeni ağ yönetimi ve kontrol uygulamaları oluşturmasını kolaylaştıran, REST API içeren yazılım bileşenleri

sağlar. Genel olarak, SDN denetleyici, SDN ortamının beyinleridir, bilgileri güneye giden API'lar ile anahtarlara ve yönlendiricilere ve kuzeydeki API'lerle uygulamalara ve iş mantığına iletir. Ryu denetleyicisi NTT tarafından desteklenir ve NTT bulut veri merkezlerinde de dağıtılır. Ryu denetleyicisi, geliştiricilerin yeni ağ yönetimi ve kontrol uygulamaları oluşturmasını kolaylaştıran iyi tanımlanmış uygulama programı arabirimleri (API) ile yazılım bileşenleri sunar. Bu bileşen yaklaşımı, kuruluşların kendi gereksinimlerini karşılamak için dağıtımları özelleştirmelerine yardımcı olur; geliştiriciler mevcut bileşenleri hızla ve kolayca değiştirebilir veya temel ağın uygulamalarının değişen taleplerini karşılamasını sağlamak için kendi bileşenlerini uygulayabilir (SDXCentral, Ryu, Kasım 2019). Ryu, OpenFlow, Netconf, OF-config, vb. gibi ağ cihazlarını yönetmek için çeşitli protokolleri destekler. Kodun tümü Apache 2.0 lisansı altında serbestçe kullanılabilir (Osrg, Kasım 2019).

3. GEREÇ VE YÖNTEM

Bu bölümde çalışmada kullanılan her bir elemanla ilgili bilgiler verilecektir. Sistemin temel parçaları olan sunucu ve istemci tarafında kullanılan yöntemler ve teknolojilerle ilgili konular ele alınacaktır. İki taraf arasındaki iletişim protokolü olan HTTP ile ilgili bilgiler geçmiş bölümde verilmiştir.



Şekil 3.1: Sistemin Genel Yapısı

Şekil 3.1'de görüldüğü gibi sistemin genel yapısında Android tabanlı bir uygulama, Ubuntu işletim sistemi olan bir bilgisayar bulunmaktadır. Bu bilgisayara aynı zamanda Mininet kurulmuştur. Mininet programlanarak SDN switchleri ve kontrolcüsü oluşturulmuştur. MPEG-DASH video dosyaları bu bilgisayara yüklenmiş ve bu dosyalar HTTP sunucusu yardımıyla yayına açılmıştır. Bu yayıncı içeriden Mininet ağına dahil edilerek SDN switchlerinden geçmesi sağlanmıştır. Böylece bu sunucunun ağ özellikleri SDN kontrolcüsü tarafından kontrol edilebilir hale getirilmiştir.

3.1 Sunucu Tarafı

Sunucu tarafında Şekil 3.1'de görüldüğü üzere bir Ubuntu PC kullanılmıştır. Bu PC'ye Mininet ortamı kurulmuş ve RYU switch özellikleri indirilmiştir. Ardından BigBuck Bunny

adlı video'nun MPD ve segment dosyalarının bulunduğu bir dizin açılıp bu dizin http protokolü ile yayınlanmıştır.

3.1.1 HTTP Sunucusu

HTTP sunucusu kurulumu için Python'un HTTP Server modülleri kullanılmıştır. Python'daki HTTP Server modülüyle istediğimiz bir port üzerinden HTTP yayını yapabilen bir server kurabiliriz. Python SimpleHTTPServer yalnızca iki HTTP yöntemini destekler - GET ve HEAD. Bu yüzden dosyaları ağ üzerinden paylaşmak için iyi bir araçtır. SimpleHTTPServer kullanarak, dosyalar aynı ağda bulunan istemcilerle kolaylıkla paylaşılabilmektedir.

Bu modüller Python'un yükleme aracı pip ile kolayca kurulabilmektedir. Bu çalışmada yer alan HTTP sunucusu için SimpleHTTPServer modülünün çoklu iş desteği olan modeli, ComplexHTTPServer modülü kullanılmıştır. Bu sayede sunucumuza bağlanan birden fazla istemciye aynı anda yanıt verebilmek amaçlanmıştır.

Tablo 3.1: Kullanılan modülün yüklenmesi için şu komut kullanılmıştır

```
pip install ComplexHTTPServer
```

Bu modülün yanı sıra basic authentication yeteneği olan bir sunucuya da ihtiyaç olmuştur. Bu sunucu ayrıcalıklı portlar için çalıştırılacak ve sadece özel kullanıcılar için yayın verme özelliği olacaktır. Özel portlardan ayrıcalıklı yayın almak isteyen kullanıcılar bu portun anahtarı ile istek göndermek zorunda olacaklardır. Bunu sağlamak için python'daki http server modülleri kullanılarak yeni bir server modeli[3] yazılmıştır.

Daha sonrasında bu sunucular Mininet ağındaki sanal bir makine içerisinde çalıştırılacaktır.

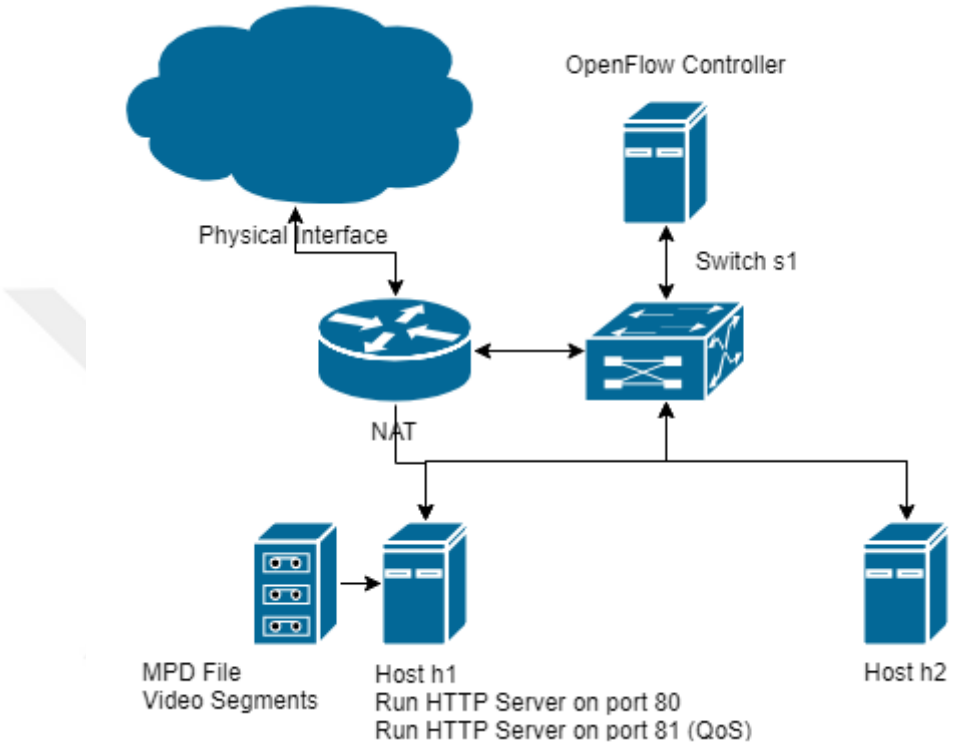
3.1.2 Mininet SDN Ağı

Sunucunun bulunduğu ağ üzerindeki SDN switchlerini ve bunun kontrolcülerini yaratmak için Mininet aracının API'si ile Python kodları yazılmıştır. Kontrolcü olarak OVS kullanılmıştır. Yaratılan OpenFlow kontrolcüsü RYU API'si ile konfigüre edilmiştir. Bu RYU API'si kontrolcüleri programlamak için REST API arayüzü sağlamaktadır. Önceden belirlenmiş QoS parametreleri REST requestleri olarak yazılıp, Mininet koduna gömülmüştür.

[3] Geliştirilen sunucu modelinin kaynak kodu Github'a eklenmiştir: https://github.com/emrecancetin/ComplexAuthHTTPServer

Yaratılan ağ toplojisinde 2 adet host, 1 adet switch, 1 adet OpenFlow kontrolcü ve 1 adet de NAT çıkışı bulunmaktadır. Bu NAT çıkışı dış ağ arayüzüne erişim için kullanılmıştır.

Topoloji'ye ek olarak fiziksel interface'e gelen istekleri NAT üzerinden HTTP server çalışan hostumuza (h1) yönlendirmek için iptables'a yönlendirme eklenmiştir (s1 ve kontrolcü yönetiminde).



Şekil 3.2: Topoloji Görsel Şeması

Tablo 3.2: Topolojideki elemanları yaratan kod parçası

```
"Create a network with nat and external controller."
net = Mininet(controller=RemoteController, switch=OVSSwitch)

info("*** Creating (reference) controllers\n")
c0 = net.addController(name='c0',
controller=RemoteController,
ip='127.0.0.1',
port=6633)

info("*** Creating switches\n")
s1 = net.addSwitch('s1')

info("*** Creating hosts\n")
hosts1 = [net.addHost('h%d' % n) for n in (1, 2)]

info("*** Creating links\n")
for h in hosts1:
    net.addLink(s1, h)

info("*** NAT activating\n")
nat0 = net.addNAT()
nat0.configDefault()
net.addLink(s1, nat0)
```

Bu kod parçasında Mininet apisinin OVSSwitch switchleri ve uzak kontrolör ile birlikte açılıp, sırasıyla kontrolör'ün girilmesi, switchin ve hostun eklenmesi ve ağın NAT'a bağlanması gerçeklenmiştir. Yaratılan hostların her biri s1 switch'ine bağlanmıştır. Bu kodların sonunda yaratılan bu ağ başlatılmıştır.

Tablo 3.3: Ağı başlatan kod parçası

```
info("*** Starting network\n")
net.build()
c0.start()
s1.start([c0])

info("*** Adding routes\n")
for h in hosts1:
    h.cmdPrint('route add default gw %s' % nat0.IP())
```

Bu kısımda ise önceki kodlarda yaratılan ağ başlatılmış, hostların hepsine nat0 noktası üzerinden gateway verilmiştir.

Tablo 3.4: Kontrolörün programlanması ve switch'e bağlanması ile ilgili olan kod parçası

```
    info("*** Configuring controller\n")

    info("||")
    s1.cmd('ovs-vsctl set Bridge s1 protocols=OpenFlow13')
    s1.cmd('ovs-vsctl set-manager ptcp:6632')

    c0.cmd(
        'ryu-manager
         ryu.app.rest_qos
         ryu.app.qos_simple_switch_13
         ryu.app.rest_conf_switch > /dev/null 2>&1 &'
        )
    time.sleep(2)
    info("||||")
    c0.cmd(
        'curl -X PUT -d
         \'\"tcp:127.0.0.1:6632\"
\' http://localhost:8080/v1.0/conf/switches/0000000000000001/ovsdb_addr'
        )
    time.sleep(1)
```

Buradaki kodlarda ise s1 switch'i üzerinde Shell komutları kullanılarak kontrolörün protokolü ve portu girilmiştir. Bu sayede switch, yarattığımız c0 kontrolörüne bağlanmıştır. Ardından c0 kontrolöründe de aynı ayarlamalar shell komutları sayesinde yapılmıştır. C0 kontrolöründe çalıştırılan ilk komutla RYU kontrolör yöneticisi çalıştırılmıştır. RYU yöneticisi çalıştıktan sonraki tüm komutlar bu yöneticinin REST API'sine gönderilmiştir. Bu komutlar cURL sayesinde HTTP request'i olarak çağırılmıştır. Burada s1 switch'imiz kontrolöre 0000000000000001 no'lu switch olarak tanıtılmıştır.

Tablo 3.5: QoS parametrelerinin kontrolörde tanımlanması ile ilgili kod parçası

```
c0.cmd('curl -X POST -d
       \'{\"port_name\":
       \"s1-eth1\", \"type\": \"linux-htb\",
       \"max_rate\": \"4000000\",
       \"queues\": [{\"max_rate\": \"1000000\"},
       {\"min_rate\": \"3000000\"}]}\'
       http://localhost:8080/qos/queue/0000000000000001')
```

Bu bölümde s1 switch'inin interface'lerine (örnekte s1-eth1 interface'i görülüyor) QoS kuralları eklenmiştir. Bu örnekte görüldüğü üzere queue'nun ilk elemanı max_rate= 1000000 (1Mbps) olarak tanımlı iken ikinci elemanı min_rate= 3000000 (3Mbps) olarak tanımlıdır. Bir sonraki aşamada burada tanımlı olan elemanlar QoS parametresi olarak atanacaktır.

Tablo 3.6: QoS parametrelerinin kontrolörde belirli bir kurala atanması ile ilgili kod parçası

```
1   c0.cmdPrint('curl -X POST -d
2 \'{\"match\":
3  {\"nw_src\": \"10.0.0.1\",
4   \"nw_proto\": \"TCP\", \"tp_src\": \"80\"},
5   \"actions\":{\"queue\": \"0\"}}\'
6 http://localhost:8080/qos/rules/0000000000000001')
```

Bu kod parçasında daha önce tanımladığımız QoS parametrelerinden birinin atanması yapılmıştır. Yukarıdaki örnekte şu kural dizisi uygulanmıştır;

- 10.0.0.1 IP'sine (h1 hostumuzun IP'si),
- TCP protokolü ile,
- 80 portundan

Gerçekleştirilen bağlantılar için QoS queue'sinde tanımlı 0 numaralı kural atanmıştır. Bu kural bir önceki kod parçasında da görüleceği üzere max_rate= 1Mbps olarak sınırlı bir ağ kuralıdır. Bu örnekte kullanılan port odaklı kuralın yanı sıra özel kullanıcı portuna temel yetkilendirme yöntemi (basic authentication) eklenmiştir. Bu sayede sadece özel kullanıcıların (ücretli üyelik gibi) bu porta bağlanabilmeleri sağlanmıştır. Pratiğe uygunluk açısından bu yöntem tercih edilmiştir. Özel kullanıcı portu olarak 81 numaralı port kullanılacaktır. Bu porta sadece kullanıcı adı ve şifre bilgileri bulunan özel kullanıcılar bağlanabilecektir.

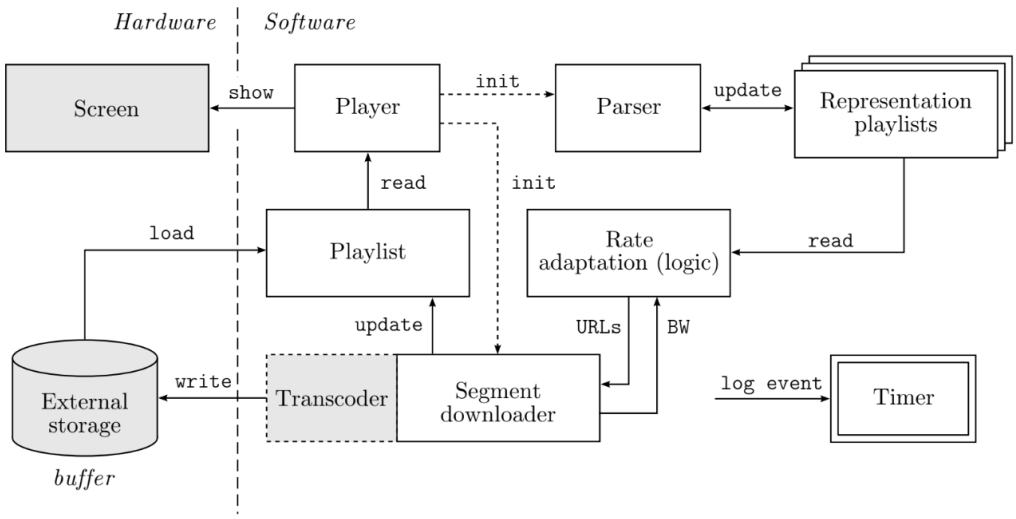
Tablo 3.7: Host üzerinde HTTP yayının başlatılması ile ilgili kod parçası

```
1    hosts1[0].cmdPrint('cd /home/emrecan/BigBuckBunny')
2    hosts1[0].cmdPrint('python -m ComplexHTTPServer 80 &')
```

Son aşamada h1 hostunda bazı komutlar çalıştırılmıştır. Bu komutlarla MPEG-DASH dosyalarının bulunduğu dizin ComplexHTTPServer modülü ile yayınlanmaya başlanmıştır.

3.2 İstemci Uygulaması

İstemci uygulaması Android uygulaması olarak yazılmıştır. Bu uygulama Android Studio ortamında geliştirilmiştir. Android'in standart media oynatıcı API'sinde DASH standartı desteklenmemektedir. Bu yüzden Exoplayer isimli Google'ın açık kaynak kodlu özel bir media oynatıcının kodları kullanılmıştır. Bu oynatıcı Github'ta tüm kaynak kodlarıyla birlikte açık bir şekilde paylaşılmıştır. Bu kodların içerisinde bir de demo uygulaması vardır. Geliştirilen uygulama bu demo uygulaması üzerinde değişiklikler yapılarak geliştirilmiştir.



Şekil 3.3: Alıcı Tarafındaki Mekanizma (Luciano Rubio Romero, 2011)

3.2.1 Uygulamanın Özellikleri

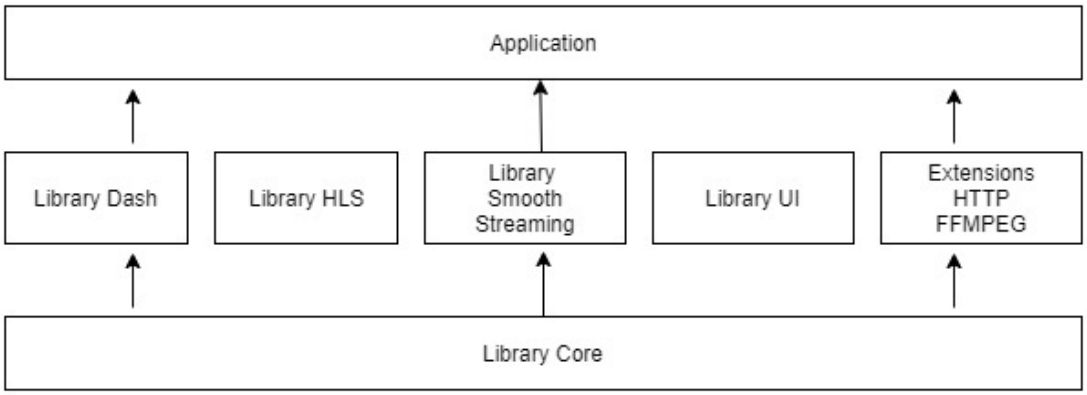
Uygulamanın sahip olduğu yetenekler şöyledir;

- MPEG-DASH ile yayın yapan url'lerden video oynatabilme özelliği
- MPD URL'i girişi
- Bant genişliğini ve oynatılan video'nun bit rate özelliklerini anlık olarak kayıt edebilme
- Belirli periyotlarla video'yu durdurup kullanıcıdan puan alabilme ve bunları dosyaya kaydedebilme

3.2.2 Uygulama Yapısı ve Çekirdek Modülü

Exoplayer'ın çekirdek modülünde segmentlerin indirilmesi, bant genişliği ölçümü, oynatıcının durumları ve video dosyalarını çözen sınıflar bulunmaktadır. Bu modül Android'in standart Media paketinin üzerine inşa edilmiştir. Bu modül Android'in standart Media paketini kullanarak ona DASH, HLS ve SmoothStreaming gibi ekstra özellikler katmak için temel özellikleri barındırmaktadır.

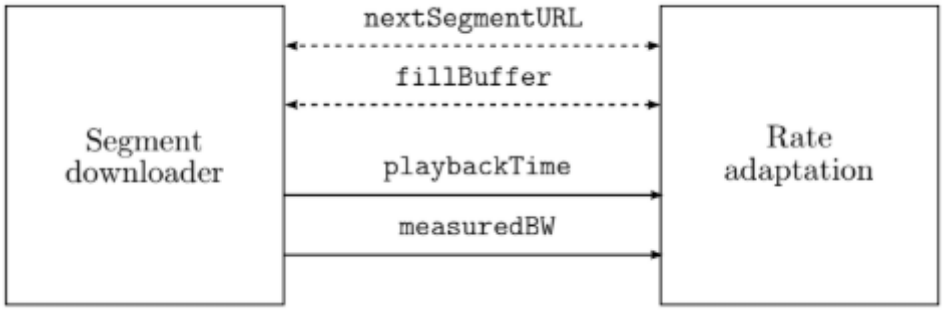
Çekirdek modülünün üzerinde Dash, HLS, SmoothStreaming ve UI katmanları bulunmaktadır. Bu modüller ana çekirdek modülündeki sınıfları ve methodları kullanarak video alanı oluşturur ve bunu yönetir. Geliştirilen uygulamada bu modüllerden DASH desteği sunan modül kullanılmıştır. Segmentleri okuyan, adaptasyon algoritmalarını çalıştıran ve uygun segmentleri indirme sınıfına verip bunları kuyruğa ekleyen kodlar bu modülün içerisinde bulunmaktadır.



Şekil 3.4 Uygulamanın Katmanları

3.2.3 Segment İndirici ile Adaptasyon Mekanizması

Adaptasyon algoritmasında görülen bant genişliği verisini segment indirici modül ölçmektedir. Bu modül, segmentleri indirirken bunların indirilme süresinden ağın bant genişliğini ölçümler ve bu değeri sürekli günceller.



Şekil 3.5 Segment İndirici ve Adaptasyon Mantığı

$$\tau = \frac{Si}{Ti} \tag{3.1}$$

Bant genişliği bu formül ile ölçülür. Buradaki τ bant genişliği çıktısı, Si segmentin boyutu ve Ti ise bu segmentin indirilme süresini ifade eder.

Segment indirici library-core içerisindeki DefaultHttpDataSource class'ından türetilir. HTTP bağlantısını açarak segment url'inden gelen dosyanın içeriğini bufferlayarak okur. Aynı zamanda BandwidthMeter isimli yardımcı class'ların callback'lerini çağırarak mevcut bant genişliğinin hesaplanmasını sağlar.

Bağlantı kurulup dosya indirimine başlandığında şu kod parçacığı ile BandwithMeter bilgilendirilir.

Tablo 3.8: Bağlantıyı Başlatan Fonksiyon

```
1  public long open(DataSpec dataSpec) throws HttpDataSourceException {
2    .
3    .
4    .
5    if (listener != null) {
6      listener.onTransferStart(this, dataSpec);
7    }
8    .
9    .
10   .
11 }
```

Buffer'lar okunurken okunan byte boyutları onBytesTransferred callback'ine düşürülür. Böylece BandwithMeter kaç byte'lık veri okunduğunu öğrenecektir.

Tablo 3.9: Veri Okuma Fonkisyonu

```
1   private int readInternal(byte[] buffer, int offset, int readLength)
2   throws IOException {
3   .
4   .
5   .
6   bytesRead += read;
7   if (listener != null) {
8     listener.onBytesTransferred(this, read);
9   }
10  .
11  .
12  .
13 }
```

Tüm bufferlar okunup, bağlantı sonlandırıldığında ise şu kod parçacığı çalışır;

Tablo 3.10: Veri Okumayı Bitiren Fonkisyon

```
1  public void close() throws HttpDataSourceException {
2    .
3    .
4    .
5    if (listener != null) {
6      listener.onTransferEnd(this);
7    }
8    .
9    .
10   .
11 }
```

BandwithMeter class'ı onTransferStart bildirimini aldığı zaman transferin başlangıç anını saklar.

Tablo 3.11: Akış Başladığında Çağırılan Fonksiyon

```
1 public synchronized void onTransferStart(Object source, DataSpec
2 dataSpec) {
3   if (streamCount == 0) {
4     sampleStartTimeMs = SystemClock.elapsedRealtime();
5   }
6   streamCount++;
7 }
```

BandwithMeter okunan byte'ların uzunluğunu segment indiricinin verdiği bilgilerle hafızalı olarak biraz önce bahsi geçen şu fonksiyonla tutar;

Tablo 3.12: Transfer Edilen Toplam Byte'ları Hesaplayan Fonksiyon

```
1 public synchronized void onBytesTransferred(Object source, int bytes) {
2   sampleBytesTransferred += bytes;
3 }
```

Akış bittiğinde çağırılan fonksiyon aşağıda verilmiştir. Bu fonksiyonun içerisinde önceden alınmış dosya akışının başladığı an, indirilmiş dosya boyutu ve dosya akışın bittiği an bilgileri kullanılarak 3.1 no'lu fonksiyon yardımıyla mevcut ağın bant genişliği hesaplanır.

Tablo 3.13: Akış Bittiğinde Çağırılan Fonksiyon

```
1 public synchronized void onTransferEnd(Object source) {
2   Assertions.checkState(streamCount > 0);
3   long nowMs = SystemClock.elapsedRealtime();
4   int sampleElapsedTimeMs = (int) (nowMs - sampleStartTimeMs);
5   totalElapsedTimeMs += sampleElapsedTimeMs;
6   totalBytesTransferred += sampleBytesTransferred;
7   if (sampleElapsedTimeMs > 0) {
8     float bitsPerSecond = (sampleBytesTransferred * 8000) /
9     sampleElapsedTimeMs;
10    slidingPercentile.addSample((int)
11    Math.sqrt(sampleBytesTransferred), bitsPerSecond);
12    if (totalElapsedTimeMs >= ELAPSED_MILLIS_FOR_ESTIMATE
13        || totalBytesTransferred >= BYTES_TRANSFERRED_FOR_ESTIMATE) {
14      float bitrateEstimateFloat =
15      slidingPercentile.getPercentile(0.5f);
16      bitrateEstimate = Float.isNaN(bitrateEstimateFloat) ?
17      NO_ESTIMATE : (long) bitrateEstimateFloat;
19    }
20  }
21  notifyBandwidthSample(sampleElapsedTimeMs, sampleBytesTransferred,
22  bitrateEstimate);
23  if (--streamCount > 0) {
24    sampleStartTimeMs = nowMs;
25  }
26  sampleBytesTransferred = 0;
27 }
```

3.2.4 Adaptasyon Algoritması

3.2.3'te gördüğümüz mekanizmadan elde edilen tahmin edilmiş bant genişliği bilgisi bu algoritma içerisinde kullanılır. Exoplayer'ın Dash adaptasyon algoritması, segment kalitesini ölçülen bant genişliğinin bir basamak altında tutacak şekilde kurulmuştur. Adaptasyon fonksiyonu şu şekildedir;

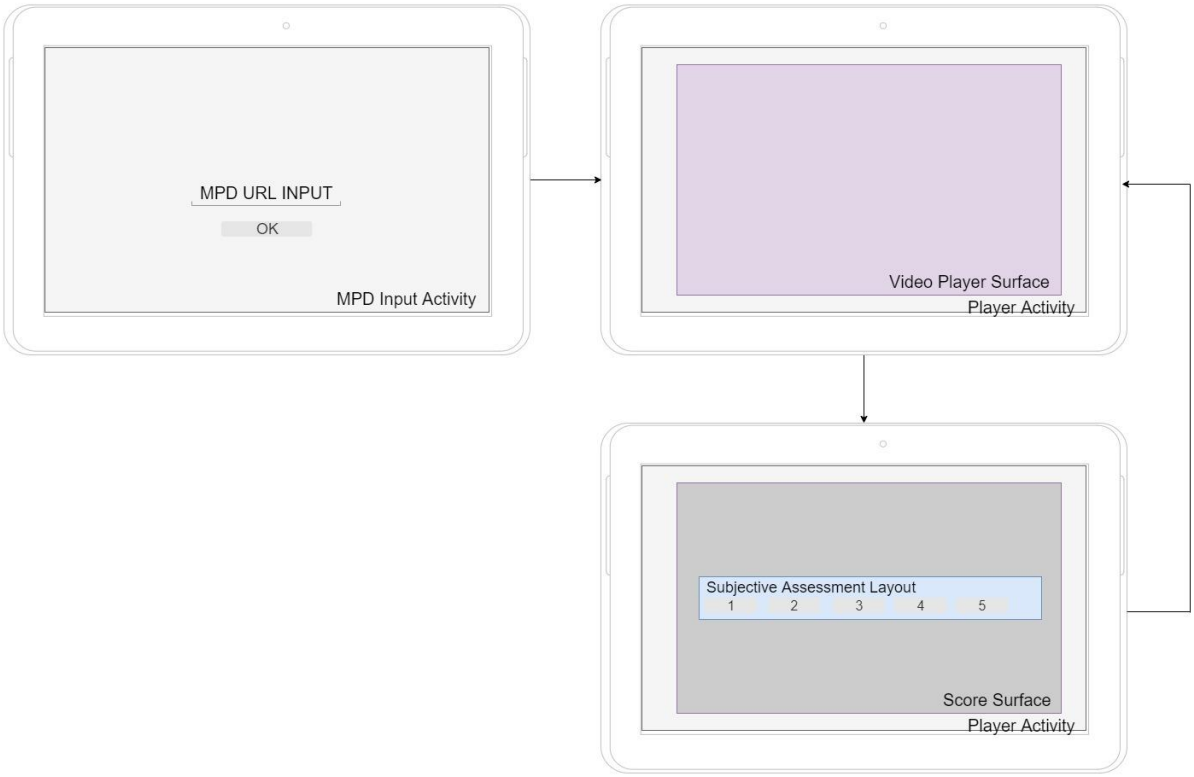
Tablo 3.14: Adaptasyon Fonkisyonu

```
1 private int determineIdealSelectedIndex(long nowMs) {
2   long bitrateEstimate = bandwidthMeter.getBitrateEstimate();
3   long effectiveBitrate = bitrateEstimate ==
BandwidthMeter.NO_ESTIMATE
4       ? maxInitialBitrate : (long) (bitrateEstimate *
5 bandwidthFraction);
6   int lowestBitrateNonBlacklistedIndex = 0;
7   for (int i = 0; i < length; i++) {
8     if (nowMs == Long.MIN_VALUE || !isBlacklisted(i, nowMs)) {
9       Format format = getFormat(i);
10      if (format.bitrate <= effectiveBitrate) {
11        return i;
12      } else {
13        lowestBitrateNonBlacklistedIndex = i;
14      }
15    }
16 }
17   return lowestBitrateNonBlacklistedIndex;
18 }
```

3.2.5 Uygulama Arayüzü ve Akışı

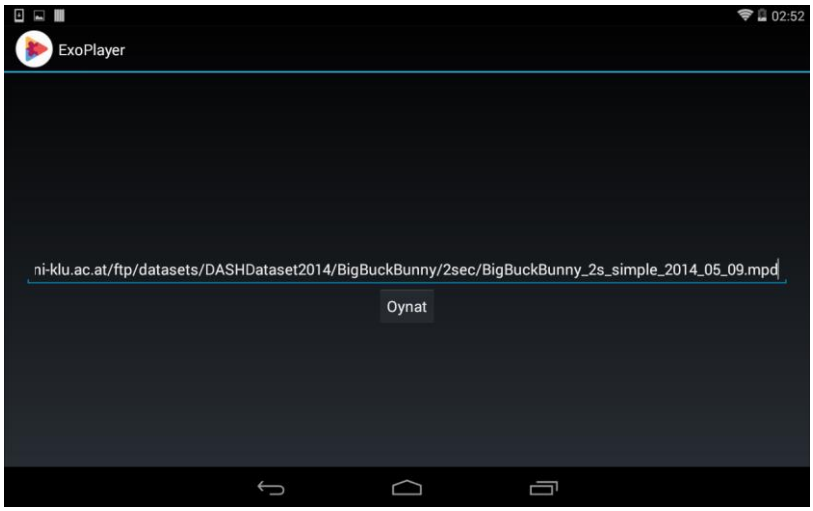
En üst katmanda Android uygulama katmanı bulunmaktadır. Bu uygulamada alt katmanlardaki tüm fonksiyonlar kullanılarak bir arayüz oluşturulur. Çekirdek kütüphanesi ve diğer dash, hls vs. gibi eklentilerdeki fonksiyonlar bu uygulama tarafından çağırılır ve kullanılır.

Uygulamamızda 2 adet Activity bulunmaktadır. Bunlardan ilki MPD url'ini girebileceğimiz bir Activity, diğeri ise video oynatıcı ve puanlama elemanlarının bulunduğu Activity'dir.

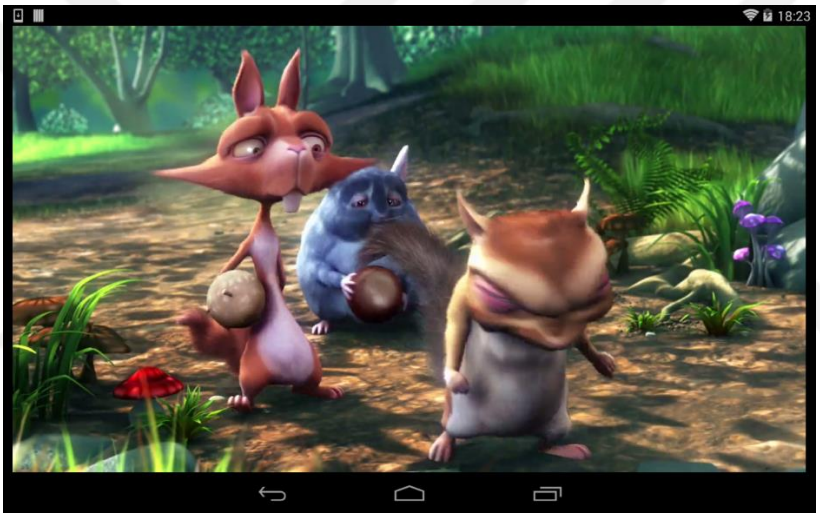


Şekil 3.6: Uygulamanın Arayüz Akışı

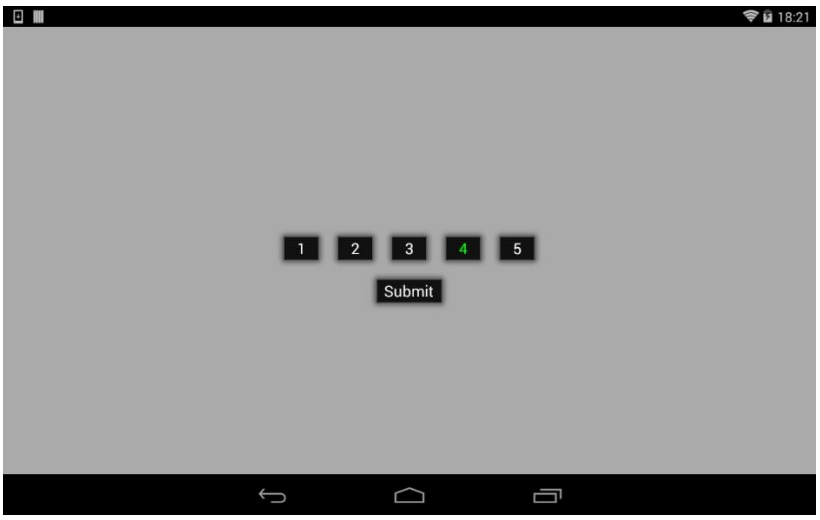
Uygulama MPD URL girişi ekranı ile açılır. Burada kullanıcıdan izlemek istediği video'nun MPD url'inin girilmesi istenir. Bu URL girilip onay verildiğinde video oynatıcının bulunduğu ekran açılır. Bu ekranda video oynatıcı yüzeyi ve değerlendirme bölümleri bulunur. Girilen URL dash kütüphanesine verilerek video oynatıcı yüzeyinde bu video'nun segmentlerinin indirilip oynatılması sağlanır.



Şekil 3.7: Uygulama Ekran Görüntüsü-1 İlk Açılış ve MPD URL'i Giriş Ekranı



Şekil 3.8: Uygulama Ekran Görüntüsü-2 Video Oynatıcı



Şekil 3.9: Uygulama Ekran Görüntüsü-3 Puanlama

Log tutma işlemi, video oynatma işlemine geçildiğinde başlatılır ve her saniye elde edilen video segmentlerinin bitrate'leri /sdcard/dashTest klasörünün altına zaman damgası adı ile .txt formatındaki bir dosyaya yazılır. Buffer'ın boş kaldığı durumlarda ise bitrate 0 olarak kaydedilir. Bu bitrateler dosyanın içerisinde her satır 1 saniyeyi gösterecek şekilde alt alta bulunur. Bu dosyadaki veriler ile bit-saniye grafikleri herhangi bir program ile çizdirilecektir.

Kalite loglarının yanı sıra, Şekil 3.9'da görüldüğü gibi gri yüzey üzerinde gelen bir puan tablosu bulunmaktadır. 30'ar saniyelik periyotlarla video durdurulup kullanıcıdan puan vermesi istenir. Kullanıcı puan girişini yaptıktan sonra video devam eder. Bu süreç içerisinde indirme ve oynatma işlemlerinin tümü durdurulur. Puanlamanın tümü ayrı bir dosyaya kaydedilir. Puanlar 1 ve 5 arasında verilmiştir. 5 çok iyi, 1 ise çok kötü olarak değerlendirilir.

3.3 Testler

Gerçeklenen sistemin performansı video kalite değerlendirme yöntemleriyle ölçülmüştür. Testlerde BigBuckBunny isimli video kullanılmıştır. Bu video'nun DASH'e uygun şekilde bölünmüş ve segmentize versiyonu ITEC'nin sunduğu veri setlerinden alınmıştır. Video'nun segmentlerinin uzunluğu T=2 saniye olarak seçilmiştir. Segmentler 45652bps'ten 4219897bps'e kadar 20 farklı kalite düzeyinde parçalanmıştır.

Tablo 3.15 Segment-Kalite Tablosu

Çözünürlük	Tip	Kodek	Kare/s	SAR	Bant Genişliği
320x240	video/mp4	avc1.42c00d	24	1:1	46.0kbps
320x240	video/mp4	avc1.42c00d	24	1:1	89.0kbps
320x240	video/mp4	avc1.42c00d	24	1:1	131.0kbps
480x360	video/mp4	avc1.42c015	24	1:1	178.0kbps
480x360	video/mp4	avc1.42c015	24	1:1	222.0kbps
480x360	video/mp4	avc1.42c015	24	1:1	263.0kbps
480x360	video/mp4	avc1.42c015	24	1:1	334.0kbps
480x360	video/mp4	avc1.42c015	24	1:1	396.0kbps
854x480	video/mp4	avc1.42c01e	24	1:1	522.0kbps
854x480	video/mp4	avc1.42c01e	24	1:1	595.0kbps
1280x720	video/mp4	avc1.42c01f	24	1:1	791.0kbps
1280x720	video/mp4	avc1.42c01f	24	1:1	1.0Mbps
1280x720	video/mp4	avc1.42c01f	24	1:1	1.2Mbps
1280x720	video/mp4	avc1.42c01f	24	1:1	1.5Mbps
1920x1080	video/mp4	avc1.42c032	24	1:1	2.1Mbps
1920x1080	video/mp4	avc1.42c032	24	1:1	2.5Mbps
1920x1080	video/mp4	avc1.42c032	24	1:1	3.1Mbps
1920x1080	video/mp4	avc1.42c032	24	1:1	3.5Mbps
1920x1080	video/mp4	avc1.42c032	24	1:1	3.8Mbps
1920x1080	video/mp4	avc1.42c032	24	1:1	4.2Mbps

Şekil 3.10: 320x240

Şekil 3.11: 1280x720

İlk aşamada bu çalışma kapsamında geliştirilmiş olan istemci uygulamasının temel fonksiyonları test edilmiştir. Bu testlerin ilki gerçek internet ortamı üzerinden farklı bağlantı türleriyle gerçekleştirilmiştir. EDGE hızındaki bir mobil bağlantı ve 25Mbit hızında bir fiber bağlantı olmak üzere 2 farklı bağlantı türü kullanılmıştır.

Gerçek ağ testleri sonrasında SDN ortamı ile bir performans testi gerçekleştirilmiştir. Bu test sırasında 2 farklı QoS düzeyi belirlenmiştir. 1. düzey, standart kullanıcı olarak tanımlı ve bu kullanıcıların bağlanabildiği ağa toplamda maksimum 1Mbit bant genişliği sunulmuştur. 2. düzey ise özel kullanıcı olarak tanımlı ve bu kullanıcılara toplamda ağın el verdiğince minimum 4Mbit bant genişliği sunulmaktadır. Toplam bant genişliği 2 kullanıcının çalışması sırasında oluşacak farkların net bir şekilde görülebilmesi için 5Mbit olarak belirlenmiştir. Kullanıcı türleri portlarla ayırt edilmiştir. Standart kullanıcının bağlanabildiği port 80 iken özel kullanıcılar 81. porttan video akışını sağlamaktadır.

Bu iki testin test metrikleri ağın ölçülen bant genişliği formül 3.1'den hesaplanan $\tau(t)$ değeri ve b(t) video segmentlerinin bit rate'idir. Ayrıca ağ kullanım oranı u(t) formül 3.2'den hesaplanmıştır.

Tablo 3.16 Ölçüm Metrikleri

Sembol	Birim	Açıklama
b(t)	Bps	Video kalitesi
u(t)	%	Ağ kullanım oranı
$\tau(t)$	Bps	Ağın ölçülen bant genişliği
bw_avail	Bps	Ağın atanan bant genişliği
$E_{startup}$	s	Videonun başlangıç gecikmesi

Performans testleri sonrasında sübjektif testler de yapılmıştır. Bu testler sadece SDN kurulu sistem üzerinde gerçekleştirilmiştir. Bu testin senaryo ve ağ ayarlamaları daha ayrıntılı bir şekilde verilecektir.

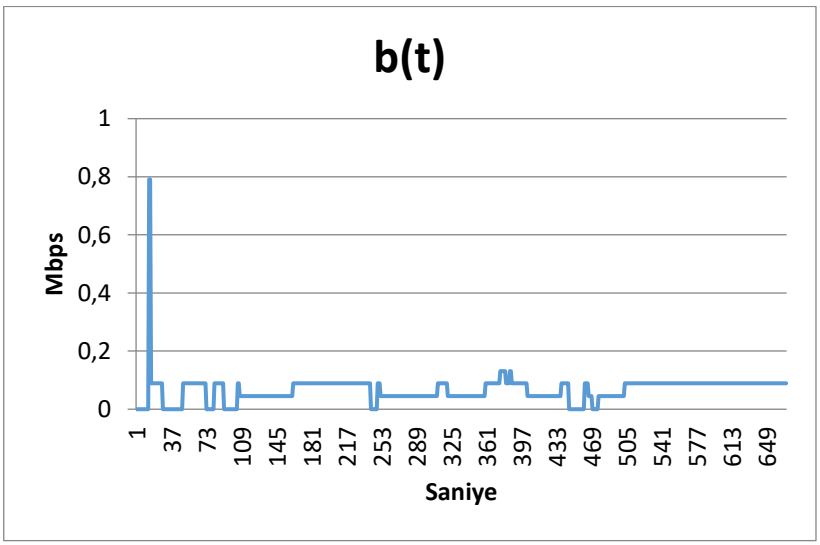
$$u(t) = \frac{b(t)}{bw\_avail(t)} \tag{3.2}$$

3.3.1 Performans Testleri

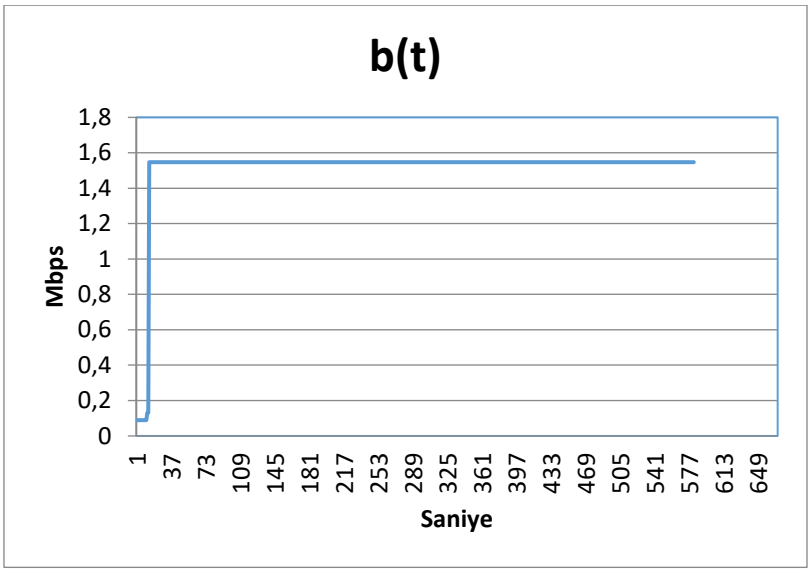
3.3.1.1 İnternet Ortamı Testleri

İnternet ortamında yapılan testlerin sonuçlarında alınan veriler aşağıdaki grafiklerde çizilmiştir. Bu grafiklerde sadece izlenen segmentlerin bitrate'leri çizdirilmiştir. Buffer'ın boş kaldığı ve donma olduğu anlarda bitrate değeri 0 olarak kayıt edilmeye devam edilmiştir. Yani grafiklerde 0 olarak görülen yerlerde video donmuştur. Örneğin edge hızındaki testte videonun donduğu pek çok yer görülebilmektedir. Bu yüzden videonun tamamının izlenmesi daha uzun sürmüştür. Ayrıca bu testte aradaki farkın açıkça gözükmesi için iki farklı uçta bağlantı özellikleri kullanılmıştır.

Video'nun mevcut tablet cihazından izlenebileceği maksimum çözünürlük 1280x720 olduğu için maksimum video bitrate'i 1.5Mbit olarak sınırlıdır.



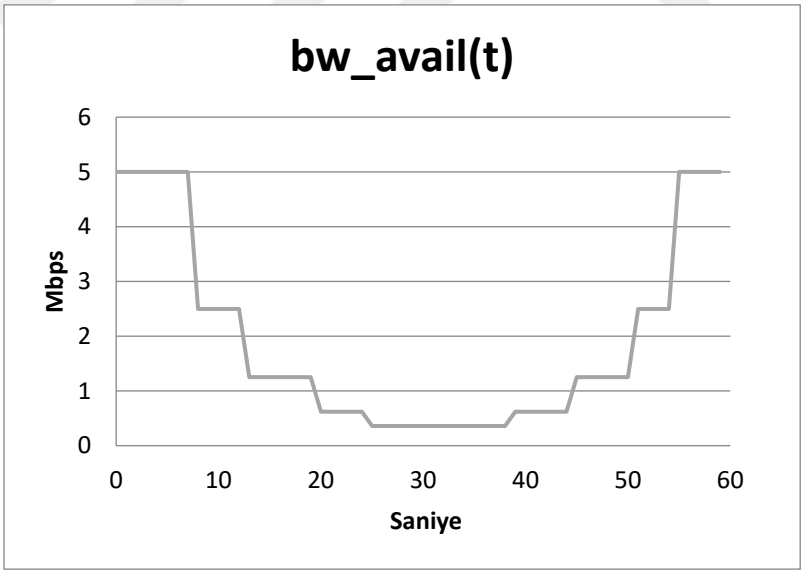
Şekil 3.12: EDGE Hızı Testi, Bitrate-Saniye Grafiği



Şekil 3.13: 25Mbit Internet Bağlantısı Hızı Testi, Bitrate-Saniye Grafiği

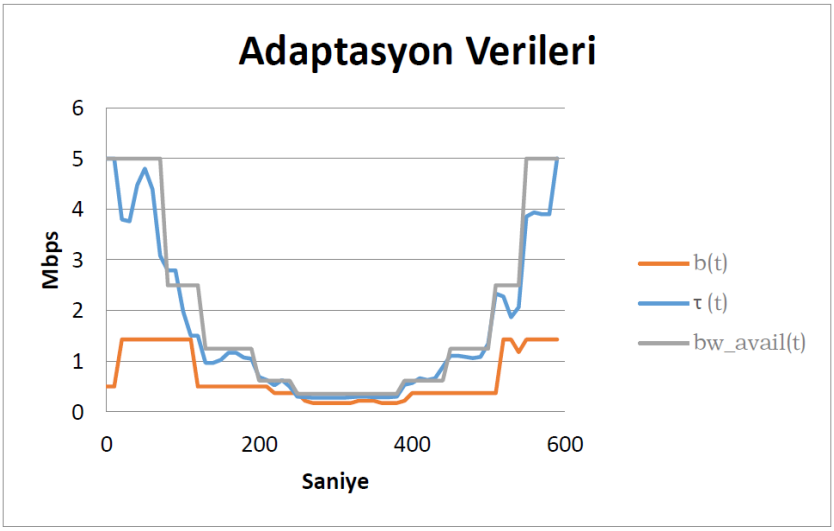
3.3.1.2 SDN'siz Network Testleri

İstemci uygulamasının adaptasyon yeteneklerini incelemek için yerel ağda basit bir http sunucusu kurulup bu sunucunun bant genişliği aşağıdaki grafikte görüldüğü gibi zamanla değişen şekilde sınırlandırılmıştır. Bu işlem için HFS Http File Server yazılımı kullanılmıştır. Düzeyler sırasıyla 5Mbit, 2.5Mbit, 1.25Mbit, 0.62Mbit ve 0.36Mbit'tir.



Şekil 3.14: Zamanla değişen bant genişliği grafiği

Uygulama bu ağ senaryosunda çalıştırılarak video'nun tamamı izlenmiştir. Bu izleme süresi boyunca toplanan b(t), τ(t) verileri ağ senaryosu (bw_avail(t)) ile birlikte çizdirilmiştir. Yine cihazımızda izleyebileceğimiz maksimum çözünürlük 1280x720 olduğu için maksimum video bitrate değeri (b(t)) 1.5Mbit ile sınırlı kalmıştır.



Şekil 3.15: Zamanla değişen bant genişliğine uygulamanın verdiği tepkilerin grafiği

Tablodan da görüldüğü üzere uygulama τ(t) değerini gerçek değere en yakın şekilde ölçümlemiş ve adaptasyon yaparak b(t) değerini ağı maksimum şekilde kullanabilmek için düzenlemiştir.

Tablo 3.17: Ağ Senaryosunun Başlangıç Gecikmesi ve Bant Genişliği Kullanım Yüzdesi

$E_{startup}$	**1.932s**
u(t)	%44.62

Tabloda senaryonun performans metrikleri görülmektedir. Video play tuşuna basıldıktaın 1.932 saniye sonra oynamaya başlamıştır. Bu değer uygulamanın tuttuğu performans logu dosyasından alınmıştır. Video bitrate'inin ağ bant genişliğine oranı ise %44.62 çıkmıştır. Bu oranın bu düzeyde çıkmasının temel sebebi maksimum video bitrate'inin 1.5Mbit olmasından dolayıdır.

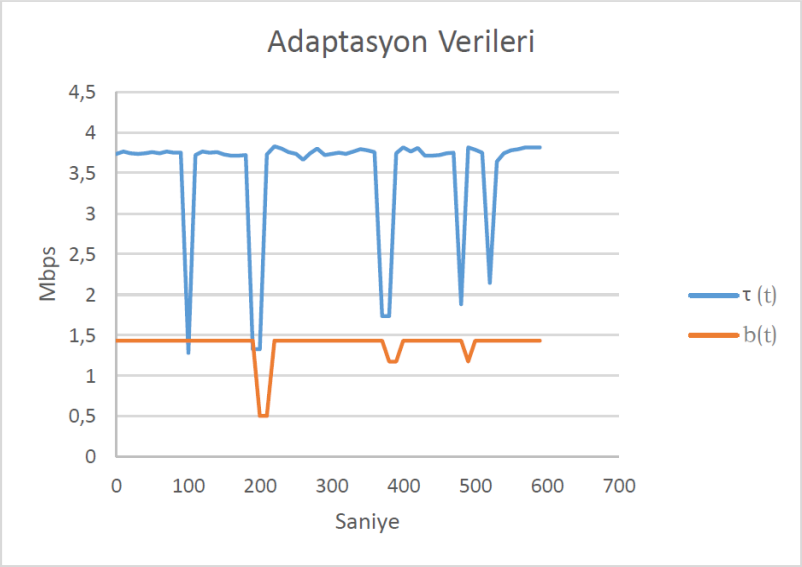
3.3.1.3 SDN'li Network Testleri

İstemci uygulaması kurulan SDN sistemiyle test edilmiştir. SDN Topolojisinde normal (best effort) ve QoE kullanıcıları olarak 2 farklı QoS düzeyi belirlenmiştir. Bu kullanıcılar portlar ile birbirinden ayrılmıştır. Bu testler sırasında farklı video alıcıları da çalıştırılmış ve trafik oluşturulmuştur.

Tablo 3.18 SDN Kontrolcü Kuralları

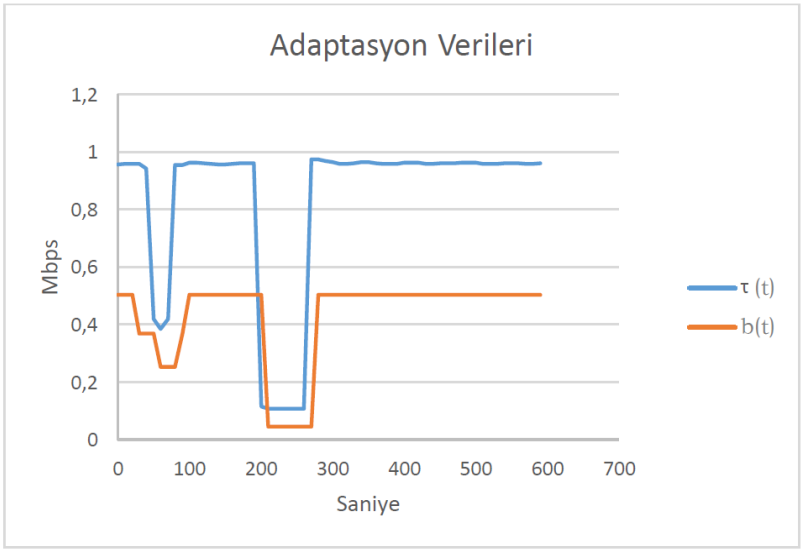
Port Numarası	Minimum Bant Genişliği	Maksimum Bant Genişliği
80	-	1Mbit
81	4Mbit	-

QoE kullanıcının (Minimum 4Mbit bant genişliği) video izlemesi esnasında alınan τ(t) ve b(t) değerlerinin grafiği Şekil 3.16'da verilmiştir. Bu test sırasında aynı anda 1 adet normal (best effort) kullanıcısı da video'yu izlemeye çalışıyor.



Şekil 3.16: Özel kullanıcı τ(t) throughput ve b(t) video bitrate grafiği

Normal (best effort) kullanıcının (Max 1Mbit bant genişliği) video izlemesi esnasında alınan τ(t) ve b(t) değerlerinin grafiği Şekil 3.17'de verilmiştir. 200. saniyede QoE kullanıcısı izlemeye başlamış ve 300. saniyede durdurulmuştur.



Şekil 3.17: Normal kullanıcı τ(t) throughput ve b(t) video bitrate grafiği

3.3.2 Öznel Testler

Gerçeklenen sistemdeki algısal kaliteyi ölçmek için bazı senaryolar belirlenmiştir. Bu senaryolar Tablo 3.19'da verilmiştir.

Tablo 3.19: Test Senaryoları

Kullanıcı Türü	Trafik Türü
Standart	Ağda trafik yok
Standart	Ağda sadece standart izleyici trafiği var
Standart	Ağda sadece 81. Port özel izleyici trafiği var
Standart	Ağda sadece 82. Port özel izleyici trafiği var
Özel (81. Port)	Ağda trafik yok
Özel (81. Port)	Ağda sadece standart izleyici trafiği var
Özel (81. Port)	Ağda sadece 81. Port özel izleyici trafiği var
Özel (81. Port)	Ağda sadece 82. Port özel izleyici trafiği var
Özel (82. Port)	Ağda trafik yok
Özel (82. Port)	Ağda sadece standart izleyici trafiği var
Özel (82. Port)	Ağda sadece 81. Port özel izleyici trafiği var
Özel (82. Port)	Ağda sadece 82. Port özel izleyici trafiği var

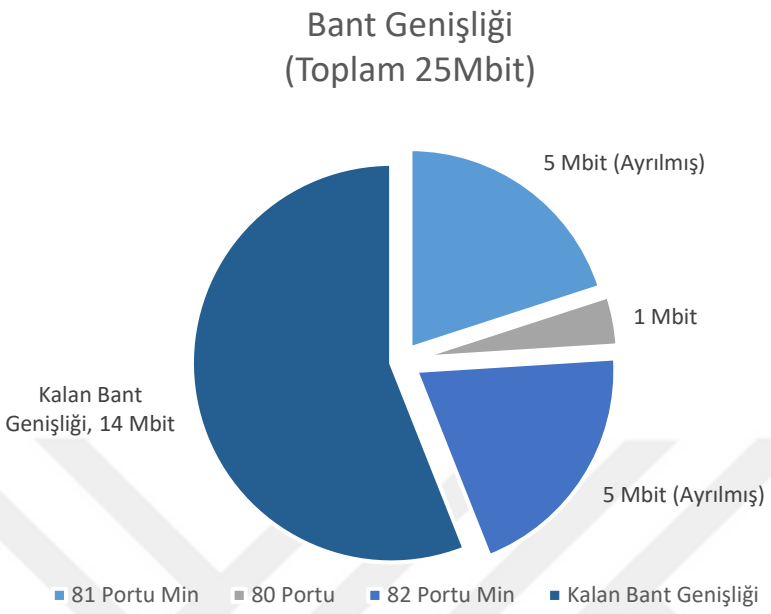
Bu senaryoların amacı, standart ve özel kullanıcı arasındaki farkı ortaya koymak ve SDN sisteminin sınanmasıdır. Ağda sadece standart kullanıcı trafiği var iken standart izleyicilerin ağı aralarında adil paylaşması beklenmektedir. Ağda özel izleyici trafiği var ise öncelik bu izleyiciye verilir ve bu türden başka izleyiciler var ise bu izleyicilerin ağı kendi aralarında adil paylaşması beklenir. Ağda özel izleyici trafiği var iken standart izleyicinin sadece bu özel izleyiciden geriye kalan bant genişliği ile video akışını sürdürmesi beklenmektedir.

Ağ trafiğini yaratmak için ise sunucu dışında bulunan dosyaları sürekli olarak indiren bir dosya indirici kullanılmıştır. Bu ağ trafiği yaratıcısı bash scripti olarak yazılıp server'ın içerisinde çalıştırılmıştır.

3.3.2.1 SDN Ağ Kuralları

Sistemin ağ trafiğini en iyi şekilde yönetmek için belirli bant genişliği atamaları yapılmıştır. 80 portunda bulunan standart kullanıcılar için toplam ağ trafiği bant genişliği 1Mbit olarak atanmıştır. Özel kullanıcılar için atanan 81 ve 82 portlarının ise bant genişlikleri minimum 5Mbit olarak ayarlanmıştır. Bunun yanı sıra 82 portunun kural tablosuna önceliği 1 olarak girilmiştir. 81 portunun ise öncelik sırası 2'dir. Toplam ağ bant genişliği ise 25Mbit olarak atanmıştır. Bu sayede şekil 3.18'de görüldüğü gibi özel kullanıcı trafiği ile standart kullanıcı trafiğinin birbirine karışmaması sağlanmıştır. Yoğun trafik durumlarında özel kullanıcılar kendi aralarında bant genişliğini paylaştıktan sonra eğer yeterli gelmezse 80 numaralı portun ağ genişliğini de eriteceklerdir. 80 numaralı porta atanan

1Mbit'lik bant genişliği bir nevi genişleme bant genişliği gibi düşünülebilir. Bu değerlerin tümü sistemin çalışması sırasında dinamik olarak değiştirilebilir.



Şekil 3.18 Sistemin Bant Genişliği Grafiği

3.3.2.2 Test Ortamı

Test ortamı Şekil 3.2'deki diyagrama göre oluşturulmuştur. Server ve SDN topolojisinin çalıştığı bir Ubuntu sanal makinesi yaratılmıştır. İstemci uygulaması ise Nexus 7 (2012) modelli tablette çalıştırılmıştır.

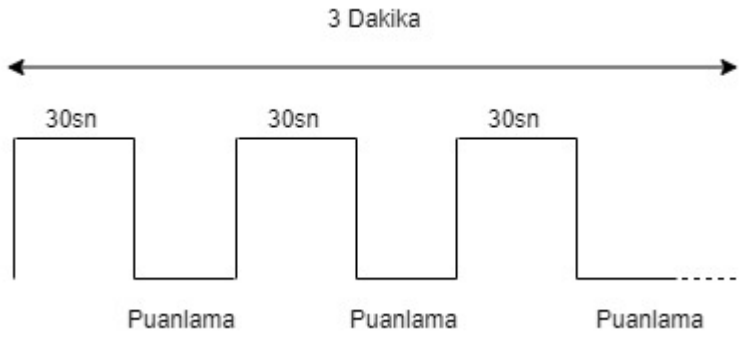
Testler sırasındaki senaryolarda kullanılan trafik ise yine farklı bir sanal makinede bulunan bir script sayesinde yaratılmıştır. Bu script segmentleri sürekli indirerek sistemde farklı bir dash izleyici varmış gibi davranmıştır.

Tablo 3.20: Test Ortamı Unsurları

	Sunucu	**İstemci**
Tür	Sanal Makine	Nexus 7 (2012) Tablet
İşletim Sistemi	Ubuntu	Android 4.4.4
Ekran Çözünürlüğü	-	1280x720
Kullanılan Uygulamalar	HTTP Server, SDN, Mininet, RYU	Geliştirilen İstemci Uygulaması (Exoplayer)

3.3.2.3 Test

Testler tablo 3.19'da verilen senaryolarla yapılmıştır. Bu senaryolar rastgele sırayla gerçeklenmiştir. Değerlendirme sistemi olarak istemci uygulamasındaki SSCQE standardıyla gerçeklenmiş ve puan düzeyleri 1-5 arası olarak belirlenmiştir. Bu testler sırasında Big Buck Bunny adlı videonun hareket düzeyi yüksek olan 3 dakikalık bir kesiti kullanılmıştır (4:30 ve 7:30 dakikaları sarası). Test aşamasında deneklerden 30'ar saniyelik periyotlarla gri ekrandaki değerlendirme girişini yapması istenmiştir (Şekil 3.19).

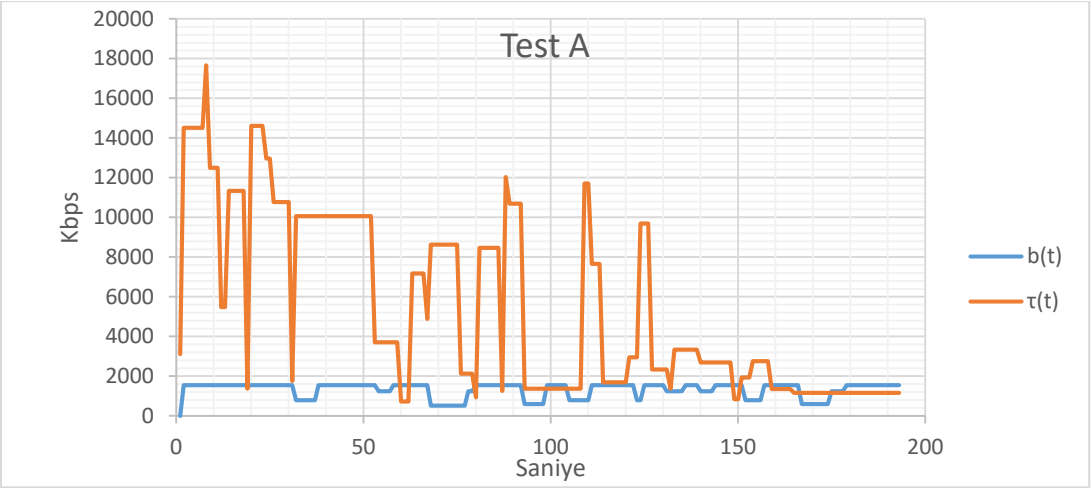


Şekil 3.19: Uygulamadaki Öznel Değerlendirme Mekanizması

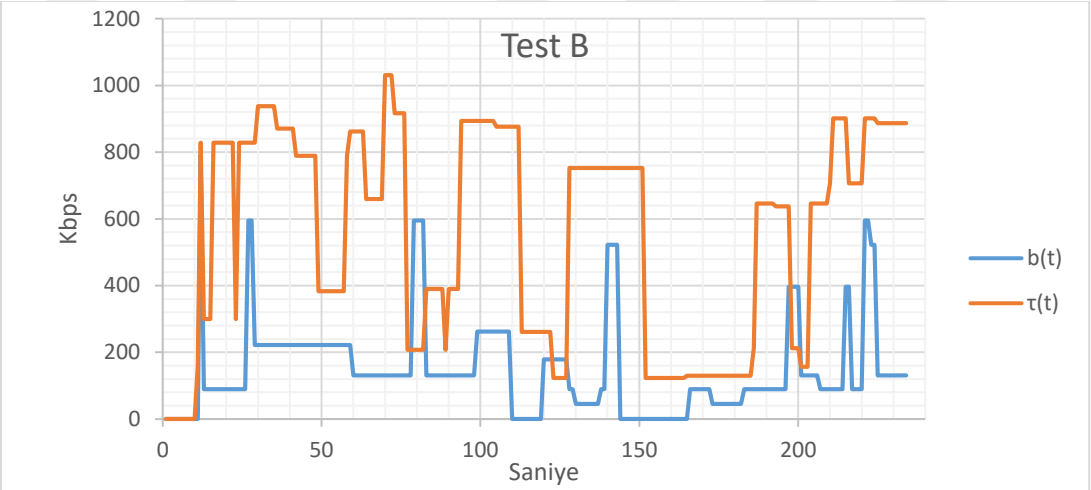
Aşağıdaki bu testler sırasında toplanan bazı metrikler yapılış sırasıyla harflendirilerek verilmiştir.

Tablo 3.21: Test Senaryolarının Yapılış Sırasına Göre Harflendirilmiş Tablosu

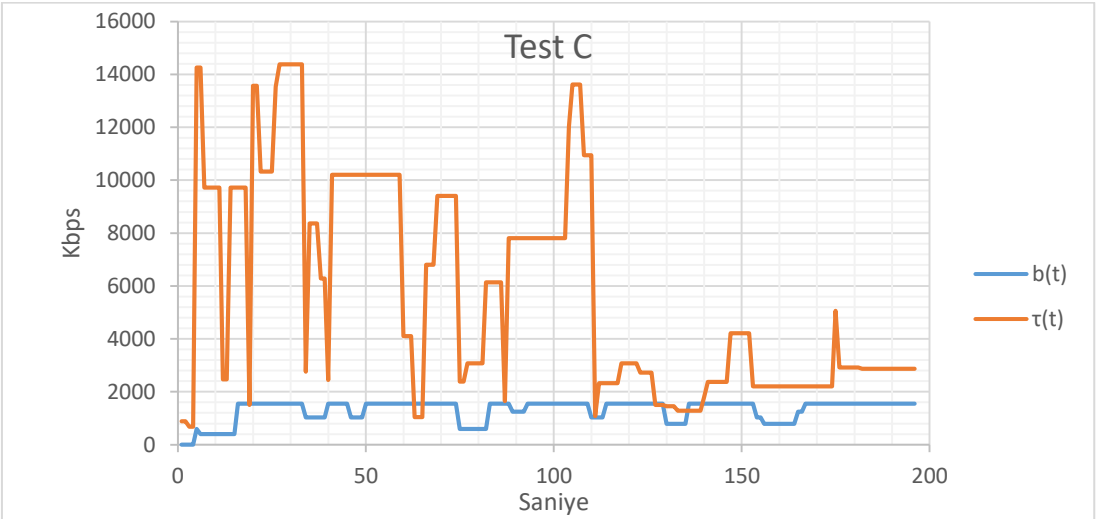
Kullanıcı Türü	Trafik Türü
(G) Standart	Ağda trafik yok
(E) Standart	Ağda sadece standart izleyici trafiği var
(B) Standart	Ağda sadece 81. Port özel izleyici trafiği var
(K) Standart	Ağda sadece 82. Port özel izleyici trafiği var
(I) Özel (81. Port)	Ağda trafik yok
(F) Özel (81. Port)	Ağda sadece standart izleyici trafiği var
(A) Özel (81. Port)	Ağda sadece 81. Port özel izleyici trafiği var
(L) Özel (81. Port)	Ağda sadece 82. Port özel izleyici trafiği var
(H) Özel (82. Port)	Ağda trafik yok
(D) Özel (82. Port)	Ağda sadece standart izleyici trafiği var
(C) Özel (82. Port)	Ağda sadece 81. Port özel izleyici trafiği var
(J) Özel (82. Port)	Ağda sadece 82. Port özel izleyici trafiği var



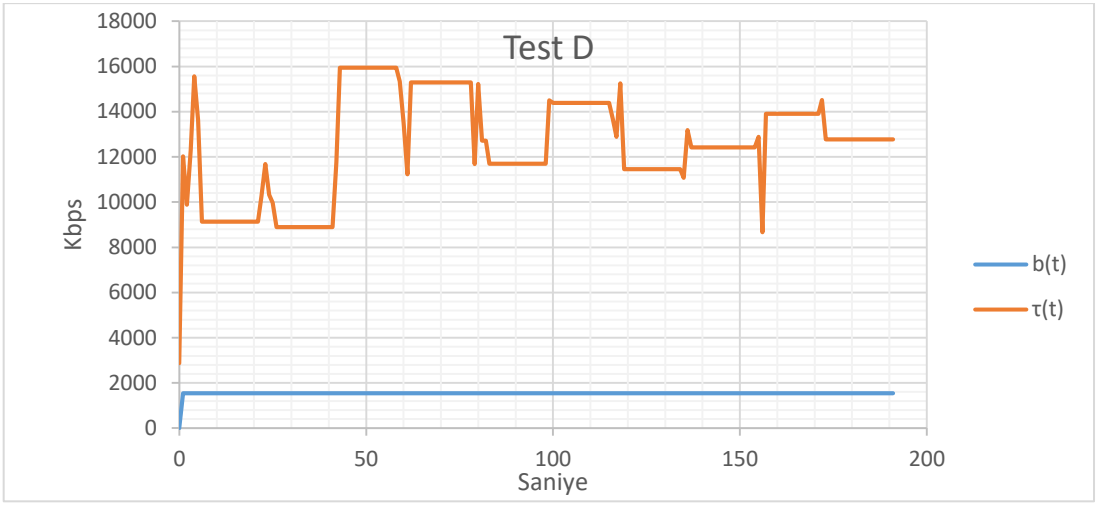
Şekil 3.20: Test A. 81. Port Testi, 81. Portta İzleyici (Trafik) Var



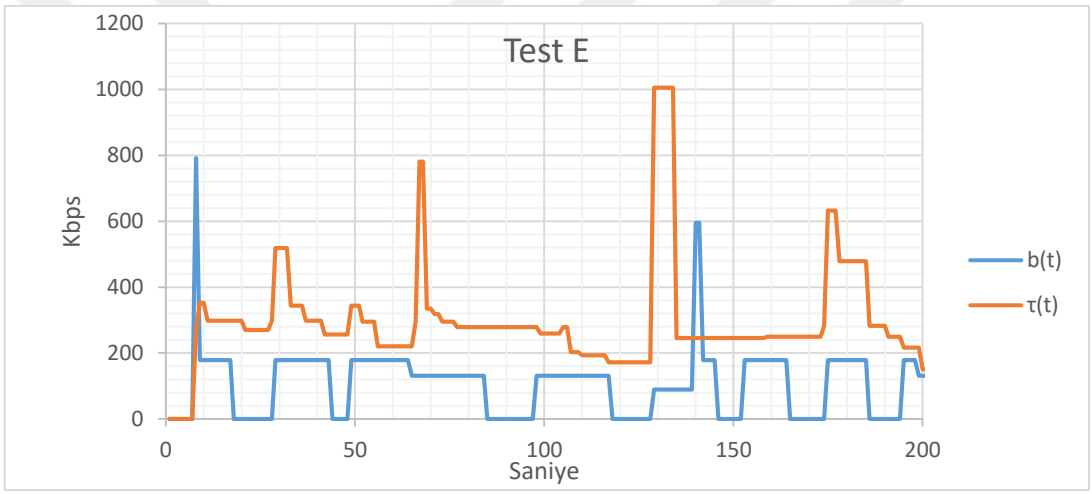
Şekil 3.21: Test B. 80. Port Testi, 81. Portta İzleyici (Trafik) Var



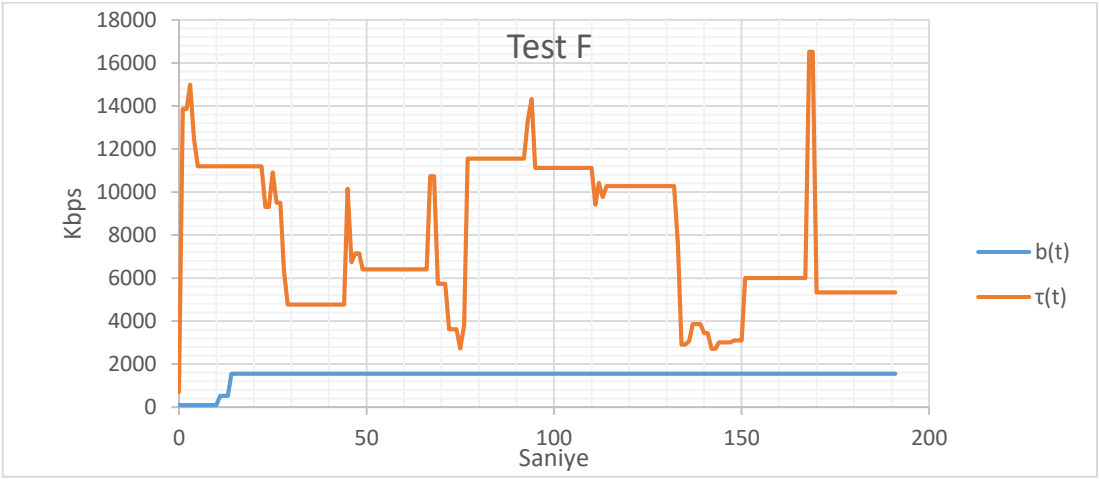
Şekil 3.22: Test C. 82. Port Testi 81. Portta İizleyici (Trafik) Var



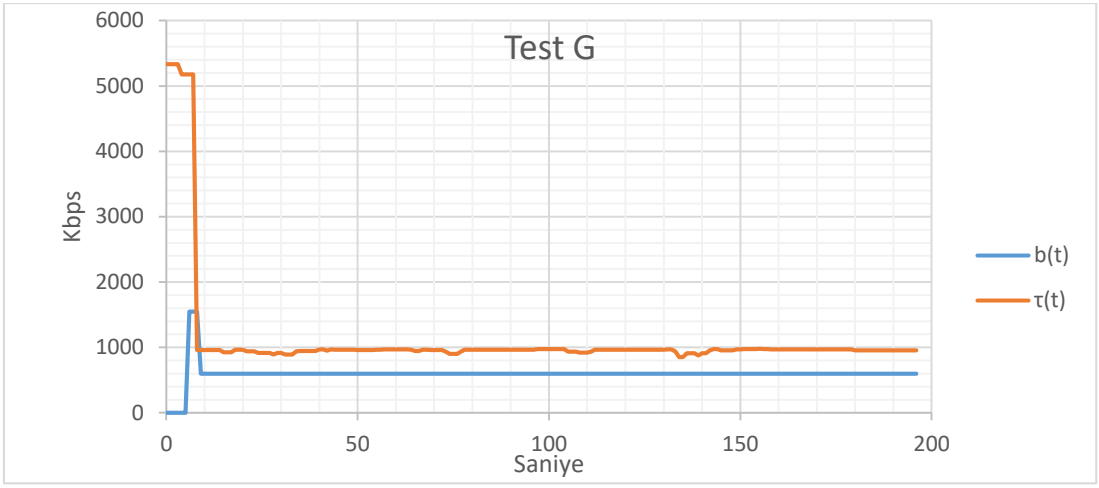
Şekil 3.23: Test D. 82. Port Testi, 80. Portta İzleyici (Trafik) Var



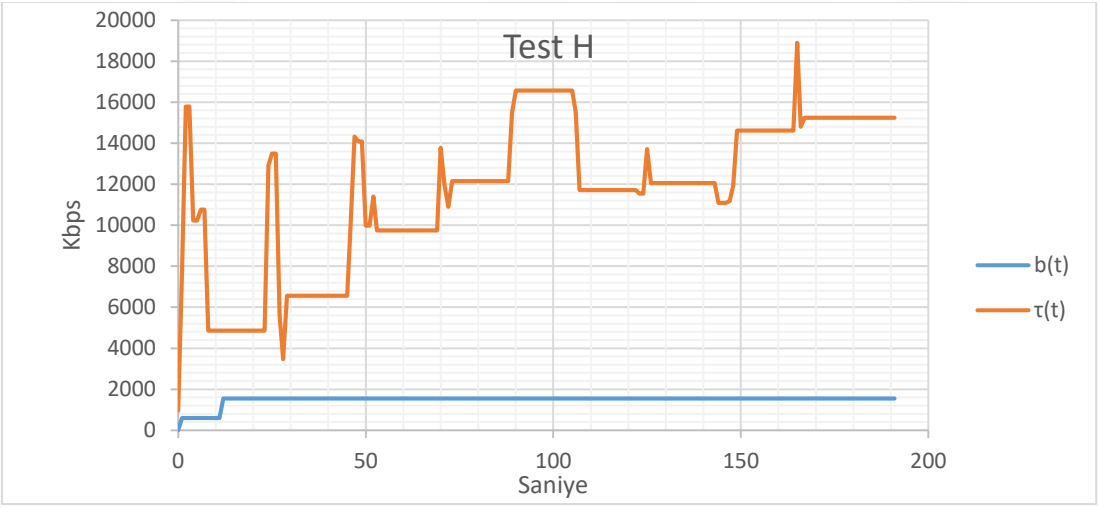
Şekil 3.24: Test E. 80. Port Testi, 80. Portta İzleyici (Trafik) Var



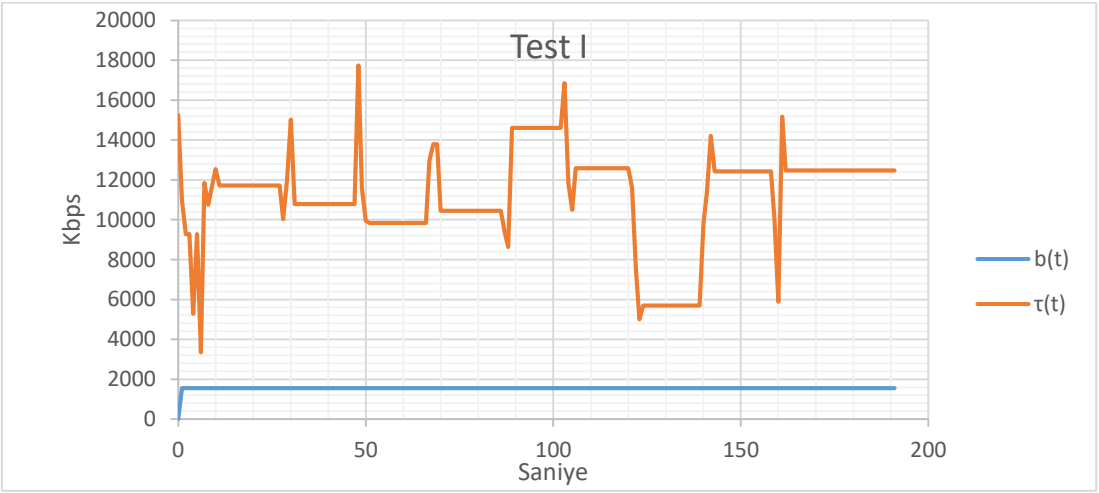
Şekil 3.25: Test F. 81. Port Testi, 80. Portta İzleyici (Trafik) Var



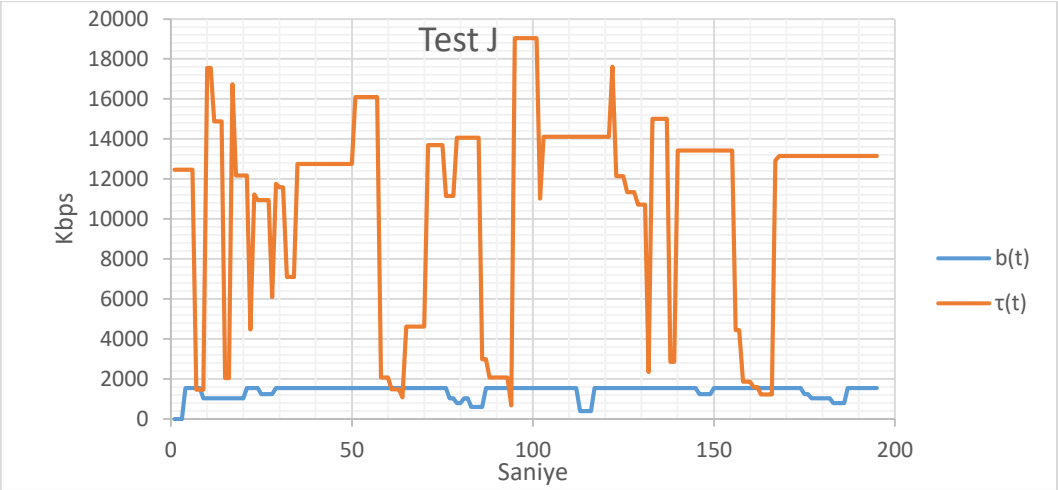
Şekil 3.26: Test G. 80. Port Testi, Trafiksiz



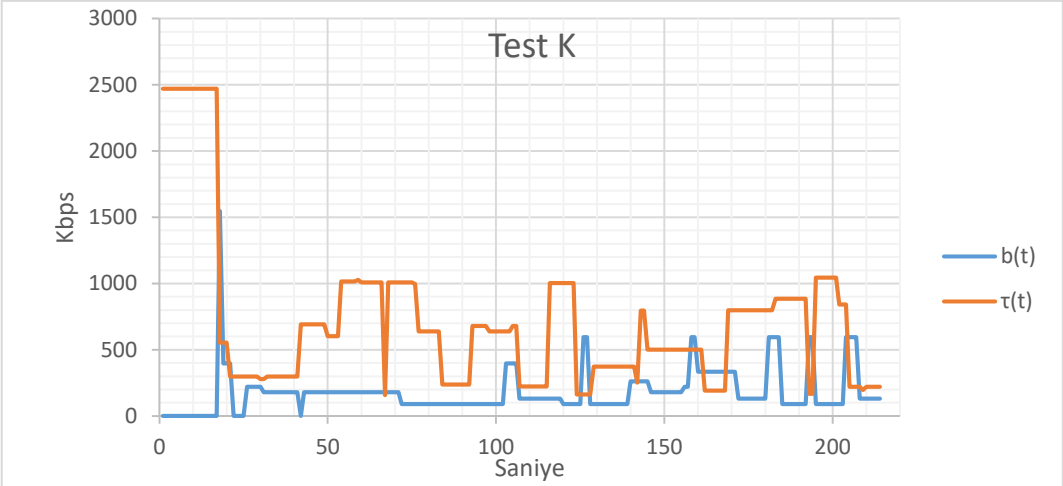
Şekil 3.27: Test H. 82. Port Testi, Trafiksiz



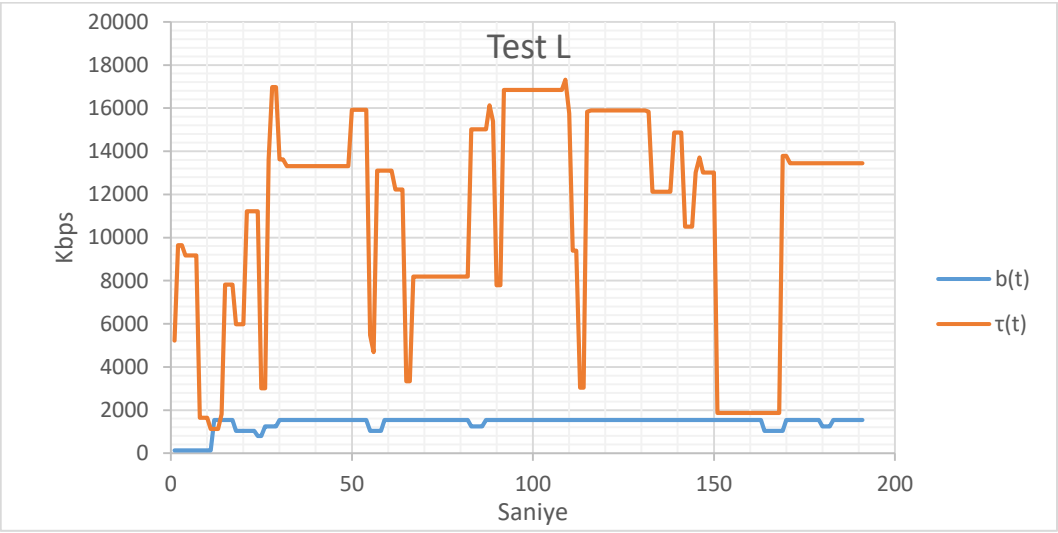
Şekil 3.28: Test I. 81. Port Testi, Trafiksiz



Şekil 3.29: Test J. 82. Port Testi, 82. Portta İzleyici (Trafik) Var



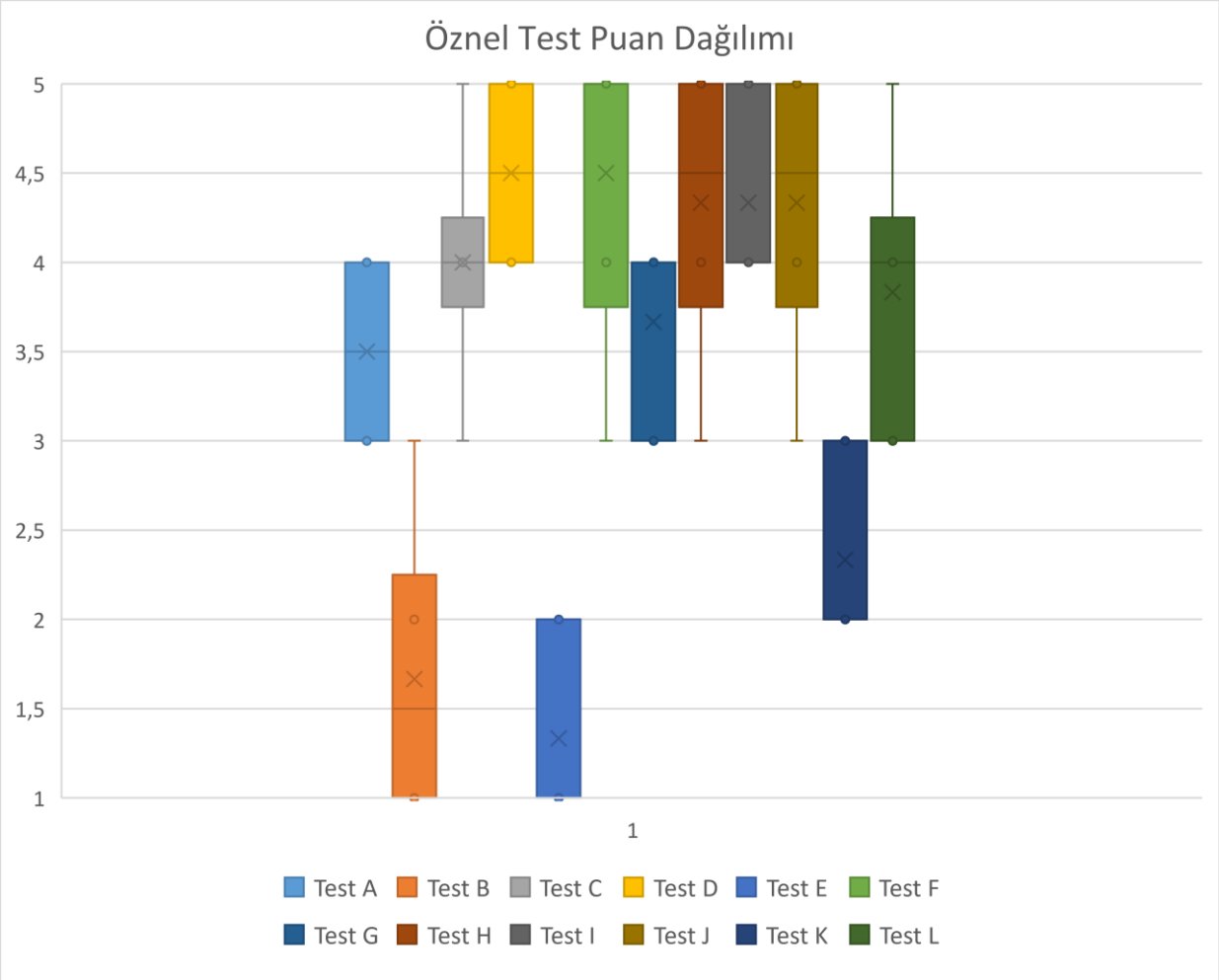
Şekil 3.30: Test K. 80. Port Testi, 82. Portta İzleyici (Trafik) Var



Şekil 3.31: Test L. 81. Port Testi, 82. Portta İzleyici (Trafik) Var

Yukarıdaki şekillerde görüldüğü üzere 12 farklı senaryonun yapılması sırasında toplanan video bitrate b(t) ve bant genişliği τ(t) grafikleri verilmiştir. Bu grafiklerde öznel sonuçları düşünmeden şunlar söylenebilir; 80. Porttan video akışı alan standart kullanıcılar trafikten çok fazla etkilenmektedir. 81 ve 82. Porttaki özel kullanıcılar ise 80. Porttaki standart kullanıcı trafiğinden etkilenmemekte, kendileriyle aynı türdeki trafikten ise az etkilenmektedirler.

Bu bölümün devamında ise öznel puanlamadan toplanan puanlar verilmiştir:

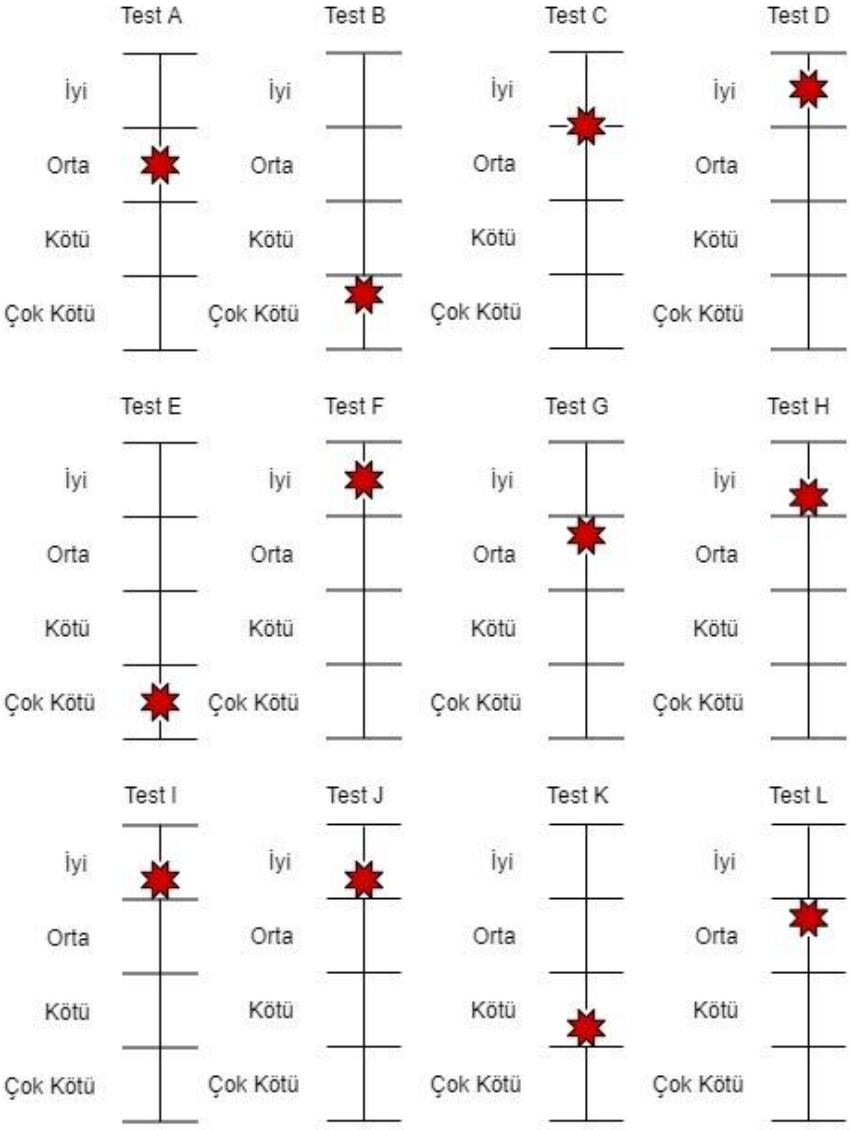


Şekil 3.32: Öznel Puanlama Sonuçları

Öznel puanlama sonuç grafiğinde yapılan değerlendirmenin puan dağılımları görülmektedir. Örneğin Test B'yi ele alacak olursak, test boyunca verilen en düşük puanın 1, en yüksek puanın ise 3 olduğu görülebilir. Ayrıca bu test için puanların çoğunun 1 ve 2 etrafında yoğunlaşmış olduğu ve Test B'nin ortalama puanının 1,66 olduğu okunmaktadır.

Değerlendirme sonuçlarına bakıldığında bu grafikler ile gösterilmiş sonuçların performans metrikleriyle örtüştüğü gözlenebilmektedir. Standart kullanıcıların trafik olan koşullarda gerçekleştirilmiş olan Test B, E ve K için değerlendirme sonucu okunduğunda kötü bir deneyim sunduğu, diğer özel kullanıcı ile yapılmış testlerin ise ortalama puan olarak

yüksek kaldığı okunabilmektedir. Şekil 3.33'te bu testlerin ortalama puanlarıyla birlikte hangi düzeyde deneyim sunduğunu gösteren bir grafik verilmiştir.



Şekil 3.33: Öznel Puanlama Ortalamaları

4. TARTIŞMA

Bu çalışma kapsamında ağ yöneticisi, sunucusu ve istemcisiyle birlikte tam bir sistem video akış sistemi önerilmiştir. Ayrıca önerilen sistemin performans testleri ile ilkel testleri yapılmıştır. Gerçek internet testlerinde internet hızının elverdiği en iyi seviyede video bit genişliği aktarımı yapıldığı görülmüştür. İstemci uygulaması bunu adaptasyon algoritmalarıyla ağ koşullarını analiz ederek sağlamıştır. Bu sayede her türlü internet bağlantısı koşulu ile hedeflenen video seyredilebilmiştir. Bu deneyimler, uygulamanın içerisindeki veri tutma sistemi yardımıyla kaydedilip grafiklerle görselleştirilmiştir. Gerçek internet testlerinin yanı sıra önceden belirlenmiş ağ bant genişliği düzeyleri kullanılarak bir test yapılmıştır. Bu testin de metrikleri grafiklerle görselleştirilmiştir. Testin grafiklerinden

adaptasyon mekanizmasının davranışları ve ayrıntıları gözlenebilmektedir. Objektif testler bölümünün sonunda SDN tabanlı ağa sahip sunucu kurulmuş ve aynı testler bu sunucuyla da yapılmıştır. SDN tabanlı sunucuya ait objektif testin sonucundan SDN'in sağladığı farklı kullanıcı tipleri arasındaki deneyim farkı gözlenmiş ve bu deneyimlerin metrikleri kaydedilip grafiklerle görselleştirilmiştir. Ancak şu göz önünde bulundurulmalıdır ki istemci uygulamasının çalıştırıldığı tabletin ekran çözünürlüğü 1280x720 olduğu için bu çözünürlüğün üzerindeki segmentler kullanılmamıştır. Bu da maksimum 1.5Mbit bit hızına sahip video aktarımı olduğu anlamına gelmektedir.

Öznel testler 12 farklı senaryo kullanılarak tamamlanmış olup, bütün testlerin hem ölçülen metrikleri hem de öznel puanlama sonuçları grafiklerle görselleştirilerek paylaşılmıştır. Öznel testler bölümünde paylaşılan b(t), τ(t) metrikleri ve puanlama sonuçları karşılaştırılarak incelendiğinde kullanıcı tipleriyle deneyim kalitesinin doğrudan ilişkili olduğu sonucu elde edilir. 80 portuna bağlanan standart kullanıcılar 1Mbit maksimum bant genişliğini paylaştığı için bariz bir şekilde düşük bit hızında video iletimi yapabilmektedirler. Diğer kullanıcı tipleri 81 ve 82 numaralı portlar üzerinden 25Mbit bant genişliğinin tümünü kullanabilmektedirler. Bunun yanı sıra SDN tarafından bu portlara 5Mbit minimum bant genişliği atanmıştır. Bu sayede cihazın elverdiği maksimum video bit aktarım hızı olan 1.5Mbit düzeyinde video kalitesi elde edilmiştir. Testler süresince trafikten kaynaklanan bant genişliği dalgalanmaları gözlenmiş, bunun sonucunun da puanlara düşükte olsa etkileri görülmüştür. Öncelik değeri daha yüksek olan 82 numaralı portun 81'e göre fark edilebilir bir fark yaratmadığı gözlenmiştir.

5. SONUÇ

Sonuç olarak bu çalışmada sunucu ve alıcı taraflarıyla birlikte bir video iletim sistemi kurulmuştur. Bu sistemde MPEG-DASH standardı kullanılmıştır. Sunucu tarafında SDN metotlarının faydalarından yararlanılmıştır. Mininet aracıyla kurulan SDN sistemi için bir topoloji önerilmiştir. Tüm bu sunucu tarafı geliştirmeleri tek tek açıklanmıştır. Bunun yanı sıra sisteme bağlanıp videoları izleten, performans logları tutabilen ve algısal değerlendirme özelliği barındıran bir Android alıcı uygulaması da yazılmıştır. Uygulamanın yazılması aşamasında Android işletim sisteminin medya yetenekleri araştırılmış, MPEG-DASH desteği bulunan açık kaynak kodlu Exoplayer adında üçüncü parti bir medya oynatıcı kütüphanesi bulunmuştur. Kütüphanenin özellikleri bu çalışmaya özel olarak değiştirilerek istenilen gereksinimleri sağlaması amaçlanmıştır. Bu uygulama sayesinde kullanılan metotların performans metrikleri ölçülmüş ve öznel değerlendirme testlerine sokulmuştur. Konu olan testler pek çok farklı senaryolarla yapılmıştır. Bunlar internet ortamı testleri, yerel ağda standart ağ testleri ve SDN ortamında belirlenmiş senaryolarla yapılan testler gibi testlerdir. Son olarak önerilen SDN topolojisini test etmek için farklı trafik senaryoları belirlenerek öznel değerlendirmeler yapılmıştır. Bu öznel değerlendirmelerden elde edilen veriler üzerinde incelemeler yapılmış, bu veriler performans metrikleriyle karşılaştırılmıştır. Çalışmadan elde edilen tecrübelerle kurulan sistemin gelişme yetenekleri ile ilgili fikirler edinilmiştir.

KAYNAKLAR DİZİNİ

Iraj Sodagar, The MPEG-DASH Standard for Multimedia Streaming Over the Internet IEEE MultiMedia Year: 2011, Volume: 18, Issue: 4 Pages: 62 – 67

Wei Pu, Zixuan Zou, Chang Wen Chen, Video adaptation proxy for wireless Dynamic Adaptive Streaming over HTTP Publication Year: 2012, Page(s): 65 – 70

Sunghee Lee, Hyunsuk Roh, Namkyung Lee, Enhanced quality adaptation scheme for improving QoE of MPEG DASH 2017 International Conference on Information and Communication Technology Convergence (ICTC) Year: 2017 Pages: 357 – 362

Sergio García, Julián Cabrera, Narciso García, Quality-optimization algorithm based on stochastic dynamic programming for MPEG DASH video streaming 2014 IEEE International Conference on Consumer Electronics (ICCE) Year: 2014 Pages: 574 – 575

Bruno Astuto A. Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, Thierry Turletti, A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks IEEE Communications Surveys & Tutorials Year: 2014, Volume: 16, Issue: 3 Pages: 1617 – 1634

Mu Mu; Matthew Broadbent; Arsham Farshad; Nicholas Hart; David Hutchison; Qiang Ni; Nicholas Race, A Scalable User Fairness Model for Adaptive Video Streaming Over SDN-Assisted Future Networks IEEE Journal on Selected Areas in Communications Year: 2016, Volume: 34, Issue: 8 Pages: 2168 – 2184

Kalpana Seshadrinathan, Rajiv Soundararajan, Alan Conrad Bovik, Lawrence K. Cormack, Study of Subjective and Objective Quality Assessment of Video IEEE Transactions on Image Processing Year: 2010, Volume: 19, Issue: 6 Pages: 1427 - 1441

Luciano Rubio Romero, "A Dynamic Adaptive HTTP Streaming Video Service for Google Android" Master Thesis - Royal Institute of Technology, Stockholm, Sweden (2011).

Manu Sood, K.K. Sharma "Mininet as a Container Based Emulator for Software Defined Networks", Ijarcsse (2014).

ITU-R RECOMMENDATION BT.500-11, Methodology for the subjective assessment of the quality of television pictures (2002)

KAYNAKLAR DİZİNİ (devam)

Sangeeta Ramakrishnan, Xiaoqing Zhu, Frank Chan, and Kashyap Kambhatla, "SDN Based QoE Optimization for HTTP-Based Adaptive Video Streaming", Cisco Systems, Inc., San Jose, CA 95134, USA

Abdelhak Bentaleb, Ali C. Begen, Roger Zimmermann, "SDNDASH: Improving QoE of HTTP Adaptive Streaming Using Software Defined Networking"

Asma BEN LETAIFA, An adaptive machine learning-based QoE approach in SDN context for video-streaming services, TÜBİTAK

ITEC, "DASH Datasets", https://dash.itec.aau.at/dash-dataset/ (Erişim tarihi: Mayıs 2017)

Agciyiz, "RTP Nedir", http://agciyiz.net/rtp-nedir/ (Erişim tarihi: Kasım 2019)

3CX, "RTP", https://www.3cx.com.tr/voip-sip/rtp/ (Erişim tarihi: Kasım 2019)

Opax, "RTSP Portu Üzerinden NVR Görüntüleme", https://www.opax.com/musteri-hizmetleri/blog/10066/rtsp-portu-uzerinden-nvr-goruntuleme (Erişim tarihi: Kasım 2019)

Sözlük Çözümpark, "Real Time Streaming Protocol RTSP", http://sozluk.cozumpark.com/goster.aspx?id=1318&kelime=Real-Time-Streaming-Protoco-RTSP (Erişim tarihi: Kasım 2019)

Nginx, "Progressive Download", https://www.nginx.com/resources/glossary/progressive-download/ (Erişim tarihi: Kasım 2019)

Wowza, "Adaptive Bitrate Streaming", https://www.wowza.com/blog/adaptive-bitrate-streaming (Erişim tarihi: Kasım 2019)

Medianova, "Adaptive Streaming", https://www.medianova.com/tr/icerik-dagitim-ve-optimizasyon/adaptive-streaming/ (Erişim tarihi: Kasım 2019)

Bitmovin, "Adaptive Streaming", https://bitmovin.com/adaptive-streaming/ (Erişim tarihi: Kasım 2019)

Stackpath, "Transcoding", https://blog.stackpath.com/transcoding/ (Erişim tarihi: Kasım 2019)

KAYNAKLAR DİZİNİ (devam)

Techopedia, "Transcoding", https://www.techopedia.com/definition/973/transcoding (Erişim tarihi: Kasım 2019)

Sciencedirect, "Scalable Coding", https://www.sciencedirect.com/topics/engineering/scalable-coding (Erişim tarihi: Kasım 2019)

Hosting, "Http Nedir?", https://www.hosting.com.tr/bilgi-bankasi/http-nedir/ (Erişim tarihi: Kasım 2019)

Network Kampus, "Http Nedir, Ne İşe Yarar?", https://networkkampus.com/http-nedir-ne-ise-yarar/ (Erişim tarihi: Kasım 2019)

Dacast, "HLS Streaming Protocol", https://www.dacast.com/blog/hls-streaming-protocol/ (Erişim tarihi: Kasım 2019)

Encoding, "HTTP Live Streaming HLS", https://www.encoding.com/http-live-streaming-hls/ (Erişim tarihi: Kasım 2019)

Apple, "Streming", https://developer.apple.com/streaming/ (Erişim tarihi: Kasım 2019)

Leadtools, "Adobe HDS", https://www.leadtools.com/sdk/multimedia/adobe-hds (Erişim tarihi: Kasım 2019)

Synopi, "MPEG-DASH", https://www.synopi.com/mpeg-dash/ (Erişim tarihi: Kasım 2019)

Bitmovin, "Dynamic Adaptive Streming HTTP MPEG DASH", https://bitmovin.com/dynamic-adaptive-streaming-http-mpeg-dash/ (Erişim tarihi: Kasım 2019)

Turksan, "Kodek", http://www.turksan.com/kodek.html (Erişim tarihi: Kasım 2019)

H2631, "H263", http://www.h263l.com/ (Erişim tarihi: Kasım 2019)

H264 Encoder, "H264", http://www.h264encoder.com/ (Erişim tarihi: Kasım 2019)

Webopedia, "Container Format", https://www.webopedia.com/TERM/C/container_format.html (Erişim tarihi: Kasım 2019)

KAYNAKLAR DİZİNİ (devam)

Inplayer, "Live Streaming vs VOD", https://inplayer.com/live-streaming-vs-vod/ (Erişim tarihi: Kasım 2019)

Open Networking, "SDN Definition", https://www.opennetworking.org/sdn-definition/?nab=1 (Erişim tarihi: Kasım 2019)

SDX Central, "What is Ryu?", https://www.sdxcentral.com/networking/sdn/definitions/what-is-ryu-controller/ (Erişim tarihi: Kasım 2019)

Ciena, "What is SDN?", https://www.ciena.com/insights/what-is/What-Is-SDN.html (Erişim tarihi: Kasım 2019)

Mininet, "Mininet", http://mininet.org/ (Erişim tarihi: Kasım 2019)

TEŞEKKÜR

Bu çalışma süresince bana destek olan ve testlere katkıda bulunan eşim Seçil Korkmaz Çetin'e, tezin biçimlenmesinde değerli katkılarını aldığım danışmanım Dr. Öğr. Üyesi Nükhet Özbek'e teşekkürü bir borç bilirim.

17 / 01 / 2020

Emrecan ÇETİN

ÖZGEÇMİŞ

İSİM:	Emrecan ÇETİN
DOĞUM TARİHİ:	14.05.1992
DOĞUM YERİ:	İZMİR
E-POSTA:	emrecan.cetin@outlook.com
ADRES:	6417/1 Yalı Mah. No: 10 Daire: 1 Karşıyaka/Izmir
LİSANS:	Ege Üniversitesi, İzmir, 2015 Elektrik Elektronik Mühendisliği
ÇALIŞMA ALANI:	Gömülü Yazılım Mühendisi