TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MAP VE REDUCE İLE MOBİL AĞLARDA UÇTAN UCA İNTERNET HIZ ANALİZİ

YÜKSEK LİSANS TEZİ

Mete UZUN

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Doç. Dr. Osman ABUL

NİSAN 2016

Fen Bilimleri Enstitüsü Onayı

……..
Prof. Dr. Osman EROĞUL
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksininlerini sağladığını onaylarım.

…….
Prof. Dr. Murat ALANYALI
Anabilimdalı Başkanı

TOBB ETÜ, Fen Bilimleri Enstitüsü'nün 131111032 numaralı Yüksek Lisans Öğrencisi **Mete UZUN** 'nun ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"MAP VE REDUCE İLE MOBİL AĞLARDA UÇTAN UCA İNTERNET HIZ ANALİZİ"** başlıklı tezi **11.04.2016** tarihinde aşağıda imzaları olan jüri tarafından kabul edilmiştir.

Tez Danışmanı :	**Doç. Dr. Osman ABUL** TOBB Ekonomive Teknoloji Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Erdoğan DOĞDU (Başkan)** TOBB Ekonomive Teknoloji Üniversitesi	.....
	Yard. Doç. Dr. Ahmet Burak CAN Hacettepe Üniversitesi	.....

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

.

Mete UZUN

ÖZET

Yüksek Lisans Tezi

MAP VE REDUCE İLE MOBİL AĞLARDA UÇTAN UCA İNTERNET HIZ ANALİZİ

Mete UZUN

TOBB Ekonomi ve Teknoloji Üniveritesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Osman ABUL

Tarih: Nisan 2016

Mobil veri iletişimi, mobil cihazların kullanımın ve veri yoğunluklu uygulamaların artmasıyla hızlı bir şekilde artmaktadır. Artan veri kullanımından dolayı veri iletişimi trafiğinin izlenmesi ve analiz edilmesi, mobil servis sağlayıcıları için ağ yönetimi ve optimizasyonu bakımından önemli hale getirmiştir. Paket tabanlı mobil ağlarda kullanıcı deneyimini ölçmek önemlidir fakat şuanki sistemlerde kullanıcı deneyimini yakalamak bir hayli zordur. Çünkü trafik analizi uçtan uca alınarak yapılamamaktadır. Çalışmamız kapsamında geliştirdiğimiz sistem, mobil ağlarda, test sistemleri gerçek kullanım senaryoları ile hız testi koşup, iletişim paketlerini uçtan uca analiz ederek saniye başına indirilen ve yüklenen veri miktarlarını hesaplamaktadır. Büyük veri kümesi halinde alınan test sonuçları paralel ve dağıtık sistemlerde MapReduce programlama modelleri kullanılarak Hadoop ortamında analiz etmektedir. Analiz edilen büyük test sonuçları ile internet hız karakteristik haritası çıkartmaktadır. Sonuç olarak sistem, uçtan uca ağ analizi yaparak, ağ hız kalitesini ölçmekte ayrıca büyük veri analizi ile birlikte kullanıcı deneyimleri ve mobil ağda oluşan problemleri kullanıcı şikâyeti üremeden tespit etmektedir.

Anahtar Kelimeler: Büyük veri, MapReduce, Hadoop, Paket analizi, Mobil performans analizi, Kullanıcı deneyimi.

ABSTRACT

Master of Science

END-TO-END INTERNET SPEED ANALYSIS OF MOBILE NETWORKS

WITH MAPREDUCE

Mete UZUN

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Computer Engineering Science Programme

Supervisor: Associate Professor Dr. Osman ABUL

Date: April 2016

Packet-based mobile networks are increasingly carrying internet data traffic for data intensive applications. Because of increased data usage, data traffic analysis have been a very essential way in network management and optimization for the mobile service providers. It is an important issue to measure user experiences in packet network, but the current systems cannot correctly capture user experiences. This is simply because; data traffic analysis is not end-to-end traffic trace. In the context of this study, the analysis system has been developed for mobile network. The test system runs speed tests with actual usage scenarios, trace communication packets and analysis download and upload. Test results as large data sets are processed and analyzed in parallel and distributed systems on cluster of computers using MapReduce programming model on Hadoop environment. Internet speed characteristics map is rendered with regional analyzed big test results. Additionally, the system reviews end-to-end network analysis and consequently, measure network speed quality too, with big data analysis, detects user experiences and problems without user complaints.

Keywords: Big data, MapReduce, Hadoop, Packet analysis, Mobile performance analysis, User experience.

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Doç. Dr. Osman ABUL'a, kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine ve destekleriyle her zaman yanımda olan aileme ve arkadaşlarıma çok teşekkür ederim.

İÇİNDEKİLER

Sayfa

ÖZET ..... **iv**
ABSTRACT ..... **v**
TEŞEKKÜR ..... **v**
İÇİNDEKİLER ..... **vii**
ŞEKİL LİSTESİ ..... **viii**
ÇİZELGE LİSTESİ ..... **ix**
KOD LİSTESİ ..... **x**
1. GİRİŞ **1**
1.1 Tezin Amacı 1
1.2 Sistemin Getirdiği Yeni Yaklaşımlar 3
1.2.1 Uçtan uca trafik analizi 3
1.2.2 Kullanıcı deneyiminin ölçülmesi 6
1.2.3 Büyük veri analizi 9
1.2.4 Müşteri şikayeti üremeden problem tespiti 11
1.3 Sistemin Uygulama Aşamaları 12
1.3.1 Test sistemlerinin oluşturulması 14
1.3.2 Test sonuçlarının analiz edilmesi 14
1.3.3 Analiz sonuçlarının gösterilemesi 14
1.4 Sistemin Genel Mimarisi 15
1.5 Diğer Sistemlere Göre Avantajları ve Dezavantajları 17
2. TEST SİSTEMİ UYGULAMASI **19**
2.1 Test Sistemi 19
2.2 Test Sistemi için Kullanılan Teknolojiler 19
2.3 Paket Analizine Giriş 21
2.4 Paket Analizi 24
3. BÜYÜK VERİ ANALİZ SİSTEMİ UYGULAMASI **35**
3.1 Büyük Veri Analiz Sistemi 35
3.2 Büyük Veri Analizi için Kullanılan Teknolojiler 36
3.3 Büyük Veri Analizine Giriş 38
3.3.1 Dağıtık dosya sistemi 39
3.3.2 Map ve Reduce algoritmaları 43
3.4 Büyük Veri Analizi 46
3.5 Analiz Sonuçlarının Gösterilmesi 57
4. SONUÇ VE ÖNERİLER **61**
4.1 Uçtan Uca Trafik Analizi 62
4.2 Kullanıcı Deneyiminin Ölçülmesi 63
4.3 Müşteri Şikayeti Üremeden Problem Tespiti 66
4.4 Hızlı Büyük Veri Analizi 67
KAYNAKLAR **69**
ÖZGEÇMİŞ **73**

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1 : Cisco'nun aylık mobil veri trafiği öngörüsü. 1
Şekil 1.2 : Mobil ağ gelişim süreci ……..2
Şekil 1.3 : Uçtan uca servis kalitesini ölçen sistemin yapısı. 3
Şekil 1.4 : Düğüm bazlı servis kalitesini ölçen sistemin yapısı. 4
Şekil 1.5 : Veri optimizasyonlu mobil ağ. 5
Şekil 1.6 : Veri trafiği kullanılabilirliği grafiği. 7
Şekil 1.7 : Kötü TCP oturumu grafiği. 8
Şekil 1.8 : Geliştirilen sistemin adımları. 13
Şekil 1.9 : Sistemin genel mimari yapısı. 16
Şekil 2.1 : Katmanlı yapı 22
Şekil 2.2 : Gidiş geliş gecikmesi 24
Şekil 2.3 : TCP/IP modeli ve OSI modeli 25
Şekil 2.4 : TCP/IP modeli katmanları 26
Şekil 2.5 : Yakalanan paketler 32
Şekil 3.1 : Dağıtık dosya sistemi mimarisi 39
Şekil 3.2 : İstemcinin DDS' den veri okuması 41
Şekil 3.3 : İstemcinin DDS' e veri yazması 41
Şekil 3.4 : Veri yedekleme boru hattı 42
Şekil 3.5 : Örnek veri yığını 45
Şekil 3.6 : Map fonksiyonuna giren veri 45
Şekil 3.7 : Map fonksiyonu çıktısı 45
Şekil 3.8 : Reduce fonksiyonu girdisi 46
Şekil 3.9 : Reduce fonksiyonu çıktısı 46
Şekil 3.10 : Map ve Reduce fonksiyonları veri akışı 46
Şekil 3.11 : Test sistemi veri yığını 50
Şekil 3.12 : Map fonksiyonuna giren test verisi 51
Şekil 3.13 : Map fonksiyonundan çıkan test verisi 51
Şekil 3.14 : Reduce fonksiyonuna giren test verisi 52
Şekil 3.15 : Reduce fonksiyonunda ortalamaların alınması 52
Şekil 3.16 : Reduce fonksiyonundan çıkan test verisi 52
Şekil 3.17 : Saniye başına indirilen veri miktarı analiz sonucu 58
Şekil 3.18 : Saniye başına yüklenen veri miktarı analiz sonucu 59
Şekil 3.19 : İstemci ile sunucu arasında gidip gelme süresi 60
Şekil 4.1 : Uçtan uca servis kalitesi yapısı 62
Şekil 4.2 : Saniye başına indirilen veri miktarı ortalaması 64
Şekil 4.3 : Saniye başına indirilen veri miktarı analizi 65
Şekil 4.4 : Kullanıcıların hız memnuniyeti geribildirimi 65
Şekil 4.5 : İnternet kullanımı 67
Şekil 4.6 : Kayseri için saniye başına indirilen veri miktarı analizi 67

ÇİZELGE LİSTESİ

Sayfa

Çizelge 2.1 : Dumpcap parametreleri..... 30
Çizelge 3.1 : Hadop ve İlişkisel Veritabanı Yönetim Sistemi' nin karşılaştırılması . 38
Çizelge 3.2 : DDS dosya sistemi komutları 40
Çizelge 3.3 : Test sonucu veri yapısı 48

KOD LİSTESİ

Sayfa

Kod 2.1 : Kurulum komutu 28
Kod 2.2 : Github kurulum komutu 28
Kod 2.3 : Kütüphane tanımlamaları 29
Kod 2.4 : Canlı trafiği dinleme 29
Kod 2.5 : SpeedTest kütüphanesi kullanımı 31
Kod 2.6 : Trafik analizi 33
Kod 3.1 : SpeedMeterTest sınıfı 53
Kod 3.2 : SpeedMeterMapper sınıfı 54
Kod 3.3 : SpeedMeterReducer sınıfı 55
Kod 3.4 : Sistemin çalıştırılması 57

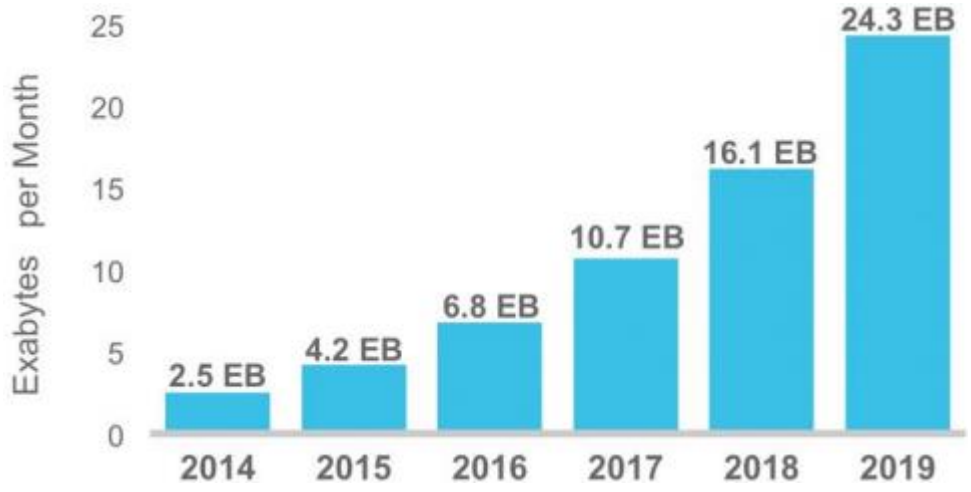
1. GİRİŞ

1.1 Tezin Amacı

1897 yılında başlayan kablosuz iletişim teknolojisi, gün geçtikçe gelişen teknoloji ile birlikte hızlı bir gelişim süreci geçirmektedir. Bu gelişim süreci de günümüz dünyasında insanlar için en önemli unsur olan mobil iletişim teknolojisini olumlu yönde etkilemiştir. Bu etkileşimin yanı sıra mobil kullanıcıların ihtiyaç alanlarının ses trafiğinden veri trafiğine doğru değişim göstermesi de mobil şebeke altyapılarının değişimine zemin hazırlamıştır.

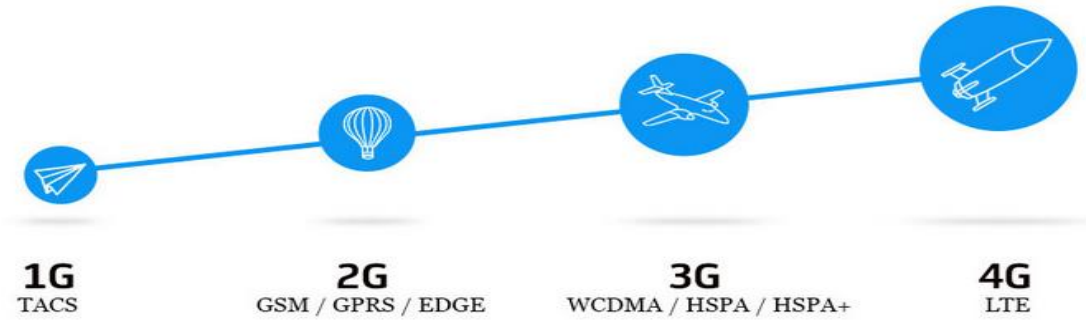
Mobil kullanıcıların haberleşme, oyun, müzik, video, eğlence ve günlük hayattaki internet ihtiyaçlarını karşılamak için mobil platformları kullanması, mobil ağ trafiğindeki veri miktarını artmasına sebep olmuştur. Bu artış mobil dünyadaki ihtiyaçların artmasıyla paralel olarak da gün geçtikçe devam etmektedir. Bu bakımdan mobil servis sağlayıcıları için veri trafiği ses trafiğinden daha önemli hale gelmiştir.

Cisco tarafından yayınlanan "Küresel Mobil İnternet Veri Trafiği Öngörü" [1] raporuna göre 2019 yılında aylık mobil veri trafiği 24,3 exabaytı geçeceği öngörülmüştür.



Şekil 1.1 : Cisco'nun aylık mobil veri trafiği öngörüsü [1].

Mobil iletişim teknolojisinin gelişimi ve mobil kullanıcıların taleplerindeki bu değişim, yeni nesil mobil ağ şebekelerinin alt yapılarını devre anahtarlama yönetimden paket anahtarlama yöntemine dönüştürmüştür. Veri iletişimi için daha uygun olan paket anahtarlama yöntemi tamamıyla 4. ve 5. nesil mobil ağ şebekelerinde, (4G LTE ve 5G ERA) verilerin daha iyi spektral verimlilikte ve daha yüksek hızlarda aktarılması için ağ iletişim yöntemi olarak kullanılmaktadır.



Şekil 1.2 : Mobil ağ gelişim süreci.

Yeni nesil mobil ağ şebekelerinin alt yapılarının değişmesi sonucu servis kalitesinin ölçülmesi için kullanılan anahtar performans göstergeleri de değişmiştir. Servis kalitesinin kullanıcı deneyimine yakın bir şekilde ölçülmesi değişen mobil ağ altyapısı ile birlikte daha zor bir hale gelmiştir ve rekabet açısından stratejik olarak önemli bir konu olmuştur. Bununla birlikte veri iletişimi trafiğinin izlenmesi ve analiz edilmesi, mobil servis sağlayıcıları için ağ yönetimi ve optimizasyonu bakımından önemli hale gelmiştir.

Mobil operatörler internet hizmetini kullanıcılarına daha hızlı sağlamak için kullanıcı deneyimini ölçmek amacıyla ağ trafiğinin izleyerek büyük veri içerisinde yapay zekâ ve veri madenciliği algoritmaları ile analiz etmektedirler. Yalnız bazı durumlarda yapılan bu analizler uçtan uca trafik analizi olmadıkları için ve canlı trafik analizinin sonuçları olan bazı parametreler kullanıcının gerçek hız hissiyatını yansıtmadığı için kullanıcı memnuniyetinin gerçek durumunu yansıtmamakta veya şebekede çok özel durumlardaki problemlerin yakalanmasına olanak sağlamamaktadır.

Bu ihtiyaçlar doğrultusunda, mobil şebekede belirli aralıklarda düğüm temelli hız testinin yapılıp anlık olarak ağ trafiğinin incelenerek saniye başına indirme ve yükleme veri miktarları ve paketlerin istemci ile sunucu arasında gidip gelme sürelerinin hesaplanıp ortak bir alanda tutulması sağlanmıştır. Birçok test düğümünden gelen sonuçların oluşturduğu veri yığının Hadoop platformu ile Map ve

Reduce programlama yöntemi algoritmaları kullanılarak bölgesel olarak analiz edilmiştir. Analizler sonucunda, düğüm temelinde ağ kalitesinin ölçülmesi, verimli bir şekilde kapasite planlamalarının yapılması, kullanıcı deneyimlerinin gerçeğe yakın şekilde ölçülmesi, uçtan uca trafik analizinin yapılması, şebekedeki çok özel bir durumda oluşan problemlerin yakalanması ve otomatik saha testlerinin yapılması hedeflenmiştir.

1.2 Sistemin Getirdiği Yeni Yaklaşımlar

1.2.1 Uçtan uca trafik analizi

Ağ iletişim yönteminin değişmesi ve mobil veri trafiğinin artması ile birlikte mobil servis sağlayıcıları için veri iletişimi trafiğinin izlenmesi ve analiz edilmesi, ağ yönetimi ve optimizasyonu bakımından önemli hale gelmiştir.

Günümüz dünyasında mobil operatörler veri trafiğini canlı olarak düğüm temelli kaydedip çeşitli yapay zekâ ve veri madenciliği algoritmaları ile analiz etmektedirler. Canlı trafikte olağanüstü durum olduğunda da alarm üretmekte ve önceden yazılmış bazı izlenmesi gereken iş adımlarını uygulamaktadır. Yalnız bu uygulanan yöntem bazı durumlarda gerçek durumu yansıtmamaktadır. Çünkü canlı trafik özel bir düğüm temel alınarak izlenmekte ve izlenen düğümler arasında analizi yapılmaktadır. Kısacası uçtan uca bir analiz yapılmadığı için bazı durumlarda kullanıcı deneyimini gerçeğe yakın bir şekilde yansıtmamaktadır.

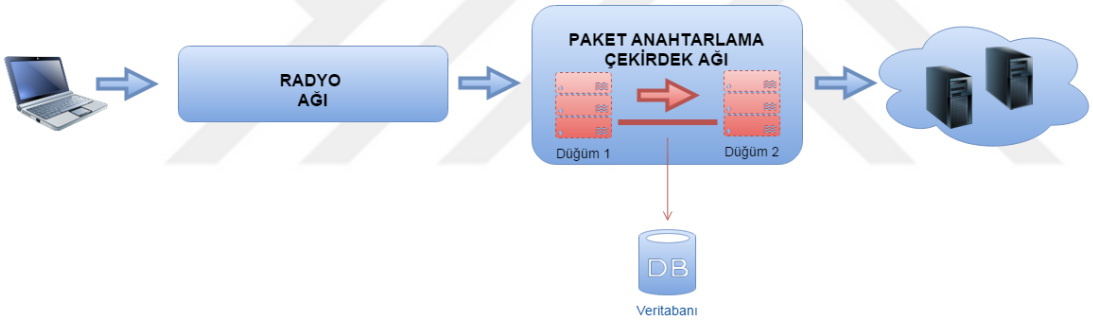
Mobil bir cihaz internetteki bir sunucuya herhangi bir istek gönderdiğinde radyo arayüzünden ve çekirdek ağındaki düğümlerden geçerek iletişim sağlanmaktadır. Bu durumda da radyo ara yüzünü ve çekirdek ağdaki düğümleri içerisine alan uçtan uca bir analiz önemli hale gelmektedir. Varolan araçlara göre sistemimizin en önemli avantajı ise test sistemlerimizin gerçek kullanıma yakın senaryoları test edip, uçtan uca veri trafiğinin analizini yapmasıdır. Şekil 1.3 ve Şekil 1.4' de iki ayrı analiz durumu gösterilmektedir.

Şekil 1.3 : Uçtan uca servis kalitesini ölçen sistemin yapısı.

Şekil 1.3 uçtan uca servis kalitesini ölçen sistemin yapısını göstermektedir. Şekilde de görüldüğü gibi istemcinin internetteki bir sunucuya istek göndermesi sonucu oluşan veri trafiğini uçtan uca analiz edilmesi görülmektedir. İstemci ile sunucu arasındaki trafik radyo arayüzü ve paket anahtarlama çekirdek ağındaki düğümleri de içerisine alarak analiz edilmektedir.

Uçtan uca trafik analizinin gerçekleşebilmesi için veri iletişimini sağlayan uçlar temel alınarak bu uçlar üzerinden trafiğin izlenmesi ile yapılması gerekmektedir. Analiz yönteminin böyle yapıldığı takdirde uçtan uca trafik analizi sağlanmış olmaktadır ve böylelikle kullanıcıya yansıyan hız kalitesini gerçeğe yakın bir şekilde ölçmüş olunmaktadır.

Bu çalışmamızda test sistemlerimiz çeşitli alanlarda testlerini yaparak ağ trafiğini analiz edip çalışmamız sırasında belirlediğimiz internet hız karakteristiğini yansıtan parametreleri ortak bir alana yazmaktadır. Böylelikle veri iletişimindeki trafik, uçtan uca analiz edilmiş olmaktadır.



Şekil 1.4 : Düğüm bazlı servis kalitesini ölçen sistemin yapısı.

Şekil 1.4 ise düğüm temelli servis kalitesini ölçen sistemin yapısını göstermektedir. Günümüzde şekil 1.4' de gösterilen sistemde olduğu gibi analiz yapılmaktadır. Şekilde de görüldüğü gibi analiz paket anahtarlama ağındaki düğümler arasında yapılmaktadır. Yani analiz aşamasına radyo arayüzü girmemektedir. Gerçek duruma baktığımızda genelde veri iletişiminde paketlerin gecikmesi radyo arayüzünden kaynaklanmaktadır ve istemciye bu gecikmenin nasıl yansıdığının ölçülmesi radyo ara yüzünü de analize katmakla mümkün olmaktadır.

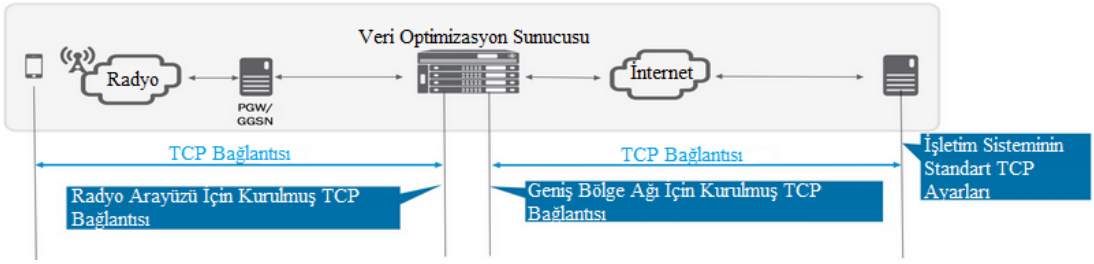
Paket anahtarlama ağındaki düğümler arasındaki trafiğin uçtan uca analiz edilmesi istemcinin servis kalitesi algısının anlaşılmasında önemli bir husus olmaktadır. Şekil 1.4' deki analiz yöntemi de uçtan uca bir analiz olmadığı için kullanıcı memnuniyetini doğrudan yansıtmamaktadır. Bu bağlamda da servis kalitesini düğüm

temelli analiz sonuçlarına göre ölçmek ve değerlendirmek yanlışlığa sebep olmaktadır veya bazı durumlarda oluşan problemlerin algılanmasına olanak sağlamamaktadır.

Günümüz dünyasında paket anahtarlama ağında uçtan uca trafik analizinin yapılmasını zorlaştıran bir durum da mobil operatörlerin mobil ağlarında kullanmış olduğu veri optimizasyon sunucuları çözümüdür. Bu sunucular mobil ağlarda verilerin daha hızlı aktarılması için istemci ile sunucu arasındaki bağlantılarda bazı optimizasyonlar yapmaktadır.

TCP teknolojisi kablolu ağlarda kullanılmak üzere geliştirilmiş bir teknolojidir. TCP çözümü kablosuz ağlarda kullanıldığında bir de bunun yanında mobil özelliği girince de bazı durumlarda istemci ve sunucu arası veri iletişiminde zorluklar yaşanmaktadır. Bundan dolayı mobil servis sağlayıcıları kablolu ve kablosuz ağları birbirinden ayıran ve aradaki arayüzü yönetimini sağlayabilecek olan yabancı düğüm adı verilen veri optimizasyon sunucuları koymaktadır. Bu sunucular istemci ve sunucu arasındaki veri iletişimini ikiye ayırıp istemci ile kendisi ve kendisi ile sunucu arasında yeni bir bağlantı açarak kablolu ve kablosuz ağları birbirinden izole ederler. Böylelikle veri optimizasyonları, TCP bağlantı optimizasyonları veya internet içerik depolama gibi yaklaşımlar getirerek veri iletişimini daha hızlı ve daha güvenilir yapmaktadır.

Bu teknolojik yaklaşımlar her ne kadar yeni nesil mobil ağlar ile birlikte bant genişliği artsa da her zaman önemini koruyacaklardır. Çünkü istemci ve sunucu arasında hangi sorunlarla karşılaşacağımızı bilemediğimiz radyo arayüzü ve mobil olma özelliği her zaman olacağından kablolu ve kablosuz ağı birbirinden ayıran TCP ve veri optimizasyon araçları her zaman mobil ağlarda yerini koruyacaktır. Aşağıda bu ağ yapısını anlatan bir şekil bulunmaktadır.



Şekil 1.5 : Veri optimizasyonlu mobil ağ [2].

Bu tür kurulumların olduğu mobil ağlarda da uçtan uca trafik analizi önem kazanmaktadır. Çünkü PGW-SGW/ GGSN şeklinde adlandırılan paket anahtarlama çekirdek ağındaki düğümlerden canlı trafik izlendiğinde yine daha önce de bahsettiğimiz gibi uçtan uca bir analiz yapmış olunmamaktadır. Bununla birlikte radyo arayüzü için ve geniş bölge ağı için kurulmuş iletişim trafiğinin sonuçlarını bir bütün halinde istemciye nasıl yansımışı ölçmek imkânsız olmaktadır. Çünkü veri iletişimi için kurulan bir bağlantı iki ayrı iletişim kanalı ile sunucuya ulaşmaktadır. Bu bağlantı kanalları radyo arayüzü için kurulmuş TCP bağlantısı ve geniş bölge ağı için kurulmuş TCP bağlantısıdır. Yine trafik analizi için düğüm temelli ortadan alınmış bir izleme, kullanıcıya gerçek anlamda yansıyan hız kalitesini göstermemektedir.

Sonuç olarak bu çalışmamızla birlikte mobil veri ağı, uçtan uca gerçeğe yakın senaryolar izlenerek testler yapılıp analiz edilmektedir. Test sistemlerimiz internetteki herhangi bir sunucuya istekte bulunarak ve bu istek doğrultusunda kurulan bağlantı izleme altına alınarak veri transferi için giden gelen paketlerin analizi yapılmaktadır. Bununla birlikte saniye başına indirilen ve yüklenen veri miktarı ve paketlerin istemci ile sunucu arasında gidip gelme süreleri hesaplanmaktadır.

Veri iletişimi için kurulan aradaki bağlantı, ağ bileşenleri kapsüllenerek ve test cihazı ile sunucu arasındaki bağlantı bir bütün halinde ele alınarak analiz edilmektedir. Böylelikle radyo arayüzü ve paket anahtarlama çekirdek ağı temel alınarak, uçtan uca servis kalitesi ölçülmekte ve kullanıcı memnuniyeti gerçeğe yakın bir şekilde ölçülmesi sağlanmaktadır.

1.2.2 Kullanıcı deneyiminin ölçülmesi

Mobil kullanıcıların ihtiyaçlarının değişmesi ile birlikte mobil veri trafiğinin artmasına neden olmuş ve bununla birlikte mobil servis sağlayıcıları için veri iletişimi trafiğinin izlenmesi ve analiz edilmesi önemli bir konu haline gelmiştir.

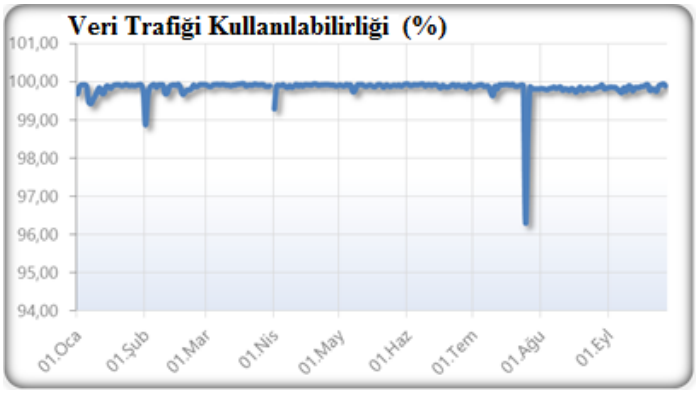
Veri iletişimi trafiği incelenip analiz edilerek kullanıcı deneyimini ölçmek mobil iletişim sektöründe stratejik bir konu olmuş ve rekabet alanında aktif olarak kullanılabilir bir yöntem haline gelmiştir. Yalnız mobil veri iletişiminde kullanıcı deneyimini ölçmek o kadar kolay bir süreç değildir. Çünkü günümüz dünyasında uçtan uca analiz bölümünde de anlattığımız gibi canlı veri trafiği düğüm temel alınıp

veri tabanına kayıt edilmektedir. Böylelikle oluşan veri yığını içerisinde çeşitli yapay zekâ ve veri madenciliği algoritmaları kullanılarak analiz edilmektedir. Yalnız bu analizler yapılırken veri iletişimindeki bazı parametreler incelenerek yapılmaktadır ve bu kullanılan parametrelerin gerçek durumu göstermesi önemli bir konudur.

Günümüz mobil operatörleri paket anahtarlama ağı kalitesini ölçebilmek için düğüm temelli veri trafikleri ışığında bazı anahtar performans göstergeleri belirlemişleridir. Bu anahtar performans göstergeleri ile birlikte kullanıcının hissettiği hizmet kalitesini ölçmeye çalışmaktadırlar.

"Veri Trafiği Kullanılabilirliği Parametresi", kullanıcıların veri iletişimi kesilmeksizin yaptığı hareketleri yüzdesel olarak gösteren bir parametredir. Yalnız bu parametre % 95' lerin üzerinde olabilir ama direk olarak kullanıcının hissettiği servis kalitesini yansıtıyor diyemeyiz. Çünkü kullanıcı veri trafiğini kesintisiz yaşayabilir ama veriye ulaşım hızlı olmayabilir. Ya da veri trafiği kesintisi altyapı sıkıntısından değil de kullanıcının isteği doğrultusunda kesilebilir.

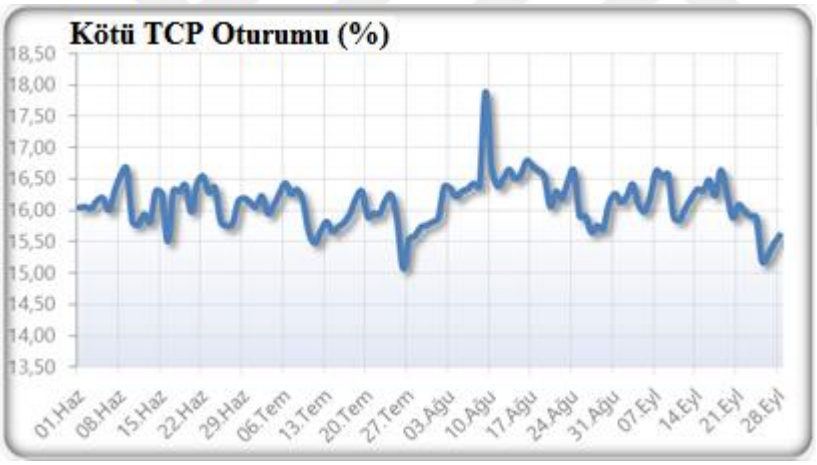
"Veri Trafiği Kullanılabilirliği Parametresi" kullanıcı deneyimini direk yansıtan bir parametredir diyemeyiz. Aşağıdaki grafikte belirli bir zaman aralığında analiz edilmiş "Veri Trafiği Kullanılabilirliği Parametresi" grafiği gösterilmektedir. Grafikte de görüldüğü gibi 27 - 29 Temmuz günleri arasında veri trafiği kullanılabilirliği % 96' lara kadar düşüş yaşamıştır. Müşteri şikâyetleri ve saha testleri çıktılarına baktığımızda o günlere ait ağ kalitesini etkileyecek bir sorun yaşanmadığı görülmüştür ama analiz parametrelerine bakıldığında gerçek durumu yansıtmadığı görülmektedir. Ya da yine "Veri Trafiği Kullanılabilirliği Parametresi" grafiğine bakıldığında % 99 oranlarına sahip durumlara bakıldığında kullanıcıların sorunsuz bir şekilde veri iletişimi yaptığını söyleyemeyiz.



Şekil 1.6 : Veri trafiği kullanılabilirliği grafiği.

Analiz için kullanılan başka bir parametre de “Kötü TCP Oturumu” parametresidir. Bu parametre de paket anahtarlama çekirdek ağındaki düğümler arasında alınmaktadır. İstemci ve sunucu arasında veri iletişimi için kurulan TCP oturumundaki paket kayıplarını, hatalı paket tespitlerini ve tekrar paket gönderimi gibi durumların süresini gösteren bir parametredir.

“Kötü TCP Oturumu” parametresi “Veri Trafiği Kullanılabilirliği Parametresinde” de olduğu gibi kullanıcı deneyimini direk yansıtan bir parametre değildir. Çünkü TCP teknolojisinin bazı özellikleri olan hatalı paket yakalama, hatalı paket düzeltme veya hatalı paketleri tekrar gönderme gibi bazı özelliklerinden dolayı kullanıcıya iletişim sırasında karşılaşılan sorunları hissettirmeyecek bazı yetenekleri vardır. Bundan dolayı “Kötü TCP Oturumu Parametresi” kullanıcının servis kalitesi hissiyatını doğrudan yansıtacak bir parametre olmadığını söyleyebiliriz. Aşağıdaki grafikte belirli bir zaman aralığında analiz edilmiş “Kötü TCP Oturumu Parametresi” grafiği gösterilmektedir.



Şekil 1.7 : Kötü TCP oturumu grafiği.

Sistemimizde kullanıcı deneyimini ölçmek için kullanılacak parametreler olarak saniye başına indirme ve yükleme veri miktarı ve paketlerin istemci ile sunucu arasında gidip gelme süreleri kullanılmaktadır. Bu parametreler veri iletişimi yapacak uçlarda alınan trafik izleme kayıtları analiz edilerek elde edilebilir.

Test sistemlerimiz uçtan uca trafik analizi yaparak saniye başına indirme ve yükleme veri miktarı ve paketlerin istemci ile sunucu arasında gidip gelme sürelerini hesaplamaktadır. Mobil hizmet sağlayıcıları paket anahtarlama çekirdek ağındaki düğüm temelli trafik izleme kayıtlarını analiz ederek bu parametreleri elde edememektedir. Bu parametreler kullanıcı deneyimini gerçeğe yakın bir şeklide

gösteren performans göstergeleri olmaktadır. Çünkü istemci, inetrnette herhangi bir sunucuya bağlandığındaki hız hissiyatını bu parametreler direk olarak yansıtmaktadır.

Örneğin radyo arayüzünde veya paket anahtarlama çekirdek ağında herhangi bir sorun olduğunda istemciye yansıyan saniye başına indirme ve yükleme veri miktarı azalacak ve paketlerin istemci ile sunucu arasında gidip gelme süreleri artacaktır. Bu bakımdan da bu parametrelerdeki değişim karakteristiği bize kullanıcının hızlı bir şekilde mi yoksa yavaş bir şekilde mi servisten hizmet aldığını gösterecektir. Böylelikle yukarıda bahsetmiş olduğumuz "Veri Trafiği Kullanılabilirliği" ve "Kötü TCP Oturumu" anahtar performans gösterge parametrelerinden daha gerçeğe yakın bir sonuç elde edilecektir. Bununla birlikte de kullanıcı deneyiminin ölçülmesi daha sağlıklı analizler ışığında yapılacaktır.

Bu çalışmamızda mobil şebekede belirli aralıklarda düğüm temelli hız testinin yapılıp anlık olarak ağ trafiğinin incelenerek saniye başına indirme ve yükleme miktarları ve paketlerin istemci ile sunucu arasında gidip gelme sürelerinin hesaplanıp ortak bir alanda tutulması sağlanmaktadır. Böylelikle birçok test düğümünden gelen büyük veri şeklindeki test sonuçları analiz edilip test düğümlerinin bulundukları bölgeler temel alınarak hız karakteristiği çıkarılmaktadır.

Böylelikle saniye başına indirme ve yükleme veri miktarı ve paketlerin istemci ile sunucu arasında gidip gelme süreleri parametreleri kullanılarak kullanıcının hissettiği servis kalitesini direk olarak yakalanması sağlanmaktadır.

1.2.3 Büyük veri analizi

Günümüz teknolojisinde bir hayli popüler olan Büyük Veri, bilişim teknolojilerinin her alana girmesiyle birlikte daha geniş bir alana hitap etmeye başladı. Mobil iletişim araçları, akıllı sayaçlar, elektronik ortam platformları, ulaşım araçları, almaçlar ve bunun gibi dijital ortama kayan diğer çeşitli platformlar sürekli olarak veri üretmektedirler. Bu veriler gün geçtikçe büyümektedir ve mevcut bilgi sistemlerinin işleyemeyeceği kadar geniş ve karmaşık veri kümelerine dönüşmektedir.

Bilinen geleneksel veritabanı yönetim sistemleri ve yazılım araçlarının, verileri toplama, saklama, çözümleme yöntemleri ışığında yeteneklerini aşan büyüklükteki oluşan verileri yönetmek ve içerisinden anlamlı işe yarayan bilgi çıkartmak artık zor ve farklı bir konu haline dönüşmüştür. Çünkü oluşan büyük veri yığını içerisinden

anlamlı ve işe yarayan bilgi üretmek tüm hizmet alanları için önemli bir konu haline gelmiştir ve kuruluşların yapısal ve yapısal olmayan verileri yakalamasını, yönetmesini ve analiz etmesini sağlayan veriler artık tüm hizmet alanları için belirleyici bir konuma gelmiştir.

Büyük veri nerde sorusunu sorduğumuzda, büyük veri her yerde cevabını kolaylıkla verebiliriz. Örneğin; elektronik bir banka için müşterilerin hareketlerinin kayıtları incelenmesi ve bu hareketlerin örüntüsünün çıkarılması olabilir. Ya da bir devlet kurumunun veya organizasyonun iç ağ güvenlik denetimini sağlayan sisteminin üretmiş olduğu ağ hareketleri kayıtlarının siber terörizm, casusluk, elektronik sistemlere sızılması, başlıca elektronik dolandırıcılık ve sahtekârlık çıkarımlar için analiz edilmesi örnek verilebilir.

Başka bir örnek de operasyonel analiz için makine verileri veya almaçlar gibi cihazların üretmiş olduğu çıktıların yorumlanması, verilerin farklı formatlara çevrilmesi veya başka veriler ile karşılaştırılmasının sağlanması verilebilir. Örneklerde de bahsettiğimiz gibi, büyük veri içerisinde çok değerli hatta stratejik olarak üstünlük sağlayacak çok önemli bilgiler vardır. Pazarlama ve müşteri talebi analizi, müşteri karakteri analizi, ürün önerileri, sunucu hareketlerini inceleme, sahtecilik analizi, gerçek zamanlı raporlama ve kullanıcı deneyimini geliştirme gibi çok daha fazla sayabileceğimiz bilgileri çok büyük veriler içerisinden elde edebiliriz.

Çalışmamızda birçok alanda konuşlandırdığımız test sistemlerimiz düğüm temelli hız testini yaparak anlık olarak ağ trafiğinin inceleyip saniye başına indirme ve yükleme veri miktarları ve paketlerin istemci ile sunucu arasında gidip gelme sürelerinin hesaplanıp ortak bir alana yazmaktadır. Bu yazılan test sonuçları Hadoop platformunda Map ve Reduce programlama yöntemleri algoritmaları ile analizinin yapılması ve sonuç olarak bölgesel veri hızı trafiğinin karakteristik olarak belirlenmesi sağlanmaktadır. Bu testler her bir test sistemimiz tarafından yapılıp sonuçları sürekli ortak bir alana yazmaktadır.

Daha detaylı bir anlatım ile test sistemlerimiz internetteki herhangi bir sunucuya bir dosyanın indirilmesi ve yüklenmesi senaryolarını yapıp istemci ve sunucu arasındaki veri iletişimi trafiğindeki giden gelen paketler bazında incelenerek analizi yapmaktadır. Bu analizler sonucunda çıktı olarak saniye başına indirilen veri miktarı, saniye başına yüklenen veri miktarı ve istemci ile sonucu arasındaki gidip gelme

süreleri verilmektedir. Bu çıktılar ortak bir alanda tüm test sonuçlarının yazıldığı yere yazılmaktadır. Bunun sonucunda oluşan büyük veri içerisinde bölge bazında hız analizi yapılmaktadır. Analizin detayları ilerideki başlıklarda daha detaylı olarak anlatılacaktır.

1.2.4 Müşteri şikâyeti üremeden problem tespiti

Mobil servis sağlayıcıları için müşteri şikâyetleri, satış sonrası destek ve problemlerin yönetilmesi bakımından önemli bir konudur. Telekomünikasyon sektöründe müşteri algısı o kadar önemlidir ki problem anında müşteriye minimum etki yansıması için stratejik olarak alınması gereken önlemler vardır. Bu önlemler sektörde rekabet açısından da önemli bir yer tutar.

Mobil servis sağlayıcıları müşteri şikâyetleri gelmeden problemlerin tespiti için canlı trafiği izleyip bazı parametreler eşik değerinin altında ya da üstünde ise alarm ürettirip bazı problemleri yakalamaya çalışmaktadırlar. Yalnız bu problem tespiti yaklaşımı müşteriye yansıyan tüm sorunları algılamamaktadır. Bazı durumlar parametrelere etki etmemiş ya da bazen eşik değerinin altına düşen parametreler müşteriye olumsuz olarak yansımamış olmaktadır. Bu durumların üstesinden gelmek pratik hayatta mobil servis sağlayıcıları için zor bir durumu oluşturmaktadır ve müşteri şikâyetleri üremeden problem tespiti ve problemlerin hızlı bir şekilde çözülmesi operasyonel mükemmellik açısından önemli bir durumdur.

Tez kapsamında geliştirdiğimiz sistem ile birlikte test sistemlerimizin yapmış olduğu testlerin sonuçlarını bölgesel olarak analiz edip, saniye başına indirme ve yükleme veri miktarları ve paketlerin istemci ile sunucu arasında gidip gelme sürelerinin belirli bir eşik değerinin üzerinde ya da altında ise alarm üretmek suretiyle birlikte problem tespitini sağlamış olmaktayız. Böylelikle test sistemlerimizin yapmış olduğu test sonuçlarına göre herhangi bir problem anını yakalamış olup operasyonel olarak gerekli adımlar yerine getirilmiş olacaktır.

Örneğin gerçek hayatta çok karşılaştığımız bir problem olan dağıtık hizmet aksatma saldırısı, paket anahtarlama çekirdek ağındaki internete çıkış sağlayan sunucularımızı servis dışı bırakabilmekte ve bununla birlikte mobil istemcilere geç cevap vermektedir. Bu durumların müşteri şikâyeti gelmeden yakalanması ve sunucuların gerekli işlemler yapılarak tekrar servise verilmesinin sağlanması önemli bir operasyonel hareket olmaktadır. Böyle durumların varlığını test sonuçlarının analizi

ile birlikte anlayabiliyor ve gerekli operasyonel adımları yerine getirilmiş olunacaktır.

Bazı durumlarda problem olmaksızın ağ yoğunluğunun getirmiş olduğu müşteri şikâyetleri de gelmektedir. Bu durumlar ağ kapasitesinin kullanım yoğunluğunu kaldıramaması veya kapasite planlamalarının yanlışlığından dolayı olmaktadır.

Ağ kapasite planlamaları ve ağ kalitesinin ölçülmesi de mobil servis sağlayıcıları için önemli bir konudur. Bu gibi kapasite planlamaları daha önce kullanım yoğunluğu temel alınarak yapılmakta ve trafik yoğunluğu incelenip kapasite artırımına gidilmektedir. Bu yaklaşım bazı durumlarda iyi planlamanın yapılmadığı veya kullanım yoğunluğunun olağanüstü sebeplerden dolayı arttığı zaman müşteri şikâyetine sebep olmaktadır.

Geliştirmiş olduğumuz sistem ile birlikte bölgesel olarak test sonuçlarına göre analiz ettiğimizde test edilen bölgenin ağ yükünü de tespit etmiş ve bu tespitler ışığında ağ kalitesinin karakteristiğini çıkartmış olacağız. Bu çıkarımlara göre de gerekli ağ kapasite planlamasının yapılması daha verimli ve daha gerçeğe yakın kullanım oranlarına göre yapılmış olacaktır. Böylelikle ağ yoğunluğunun getirmiş olduğu müşteri şikâyetleri de önlenmiş olacaktır.

Müşteri şikâyeti üremeden problem tespiti, test sonuçlarının bölgesel olarak analizi doğrultusunda sağlanmış olunacaktır. Mobil servis sağlayıcıları için önemli olan müşteri algısı ve bunun sonucunda doğan müşteri şikâyetleri, geliştirmiş olduğumuz sistemin çıktıları ve bu çıktıların analizi sonucunda, daha iyi bir şekilde yönetilmiş olacaktır.

1.3 Sistemin Uygulama Aşamaları

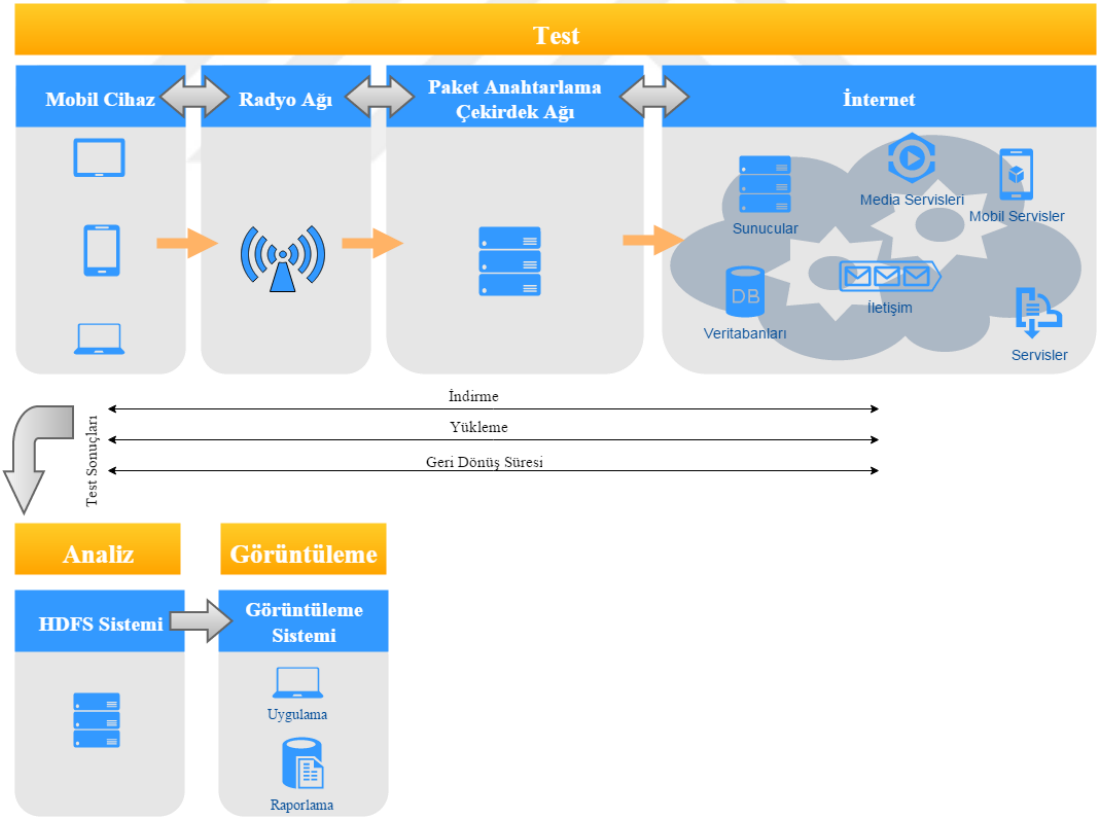
Tez kapsamında üzerinde çalıştığımız sistemin oluşturulması aşamalarını 3 ana başlık halinde inceleyebiliriz. Bu ana başlıklar şöyle sıralanabilir.

- Test Sistemlerinin Oluşturulması (Veri Toplama)
- Test Sonuçlarının Analiz Edilmesi (Veri Analizi)
- Analiz Sonuçlarının Gösterilmesi (Analiz Sonuçlarının Gösterge Panelinde Gösterilmesi)

Şekil 1.8’ de üzerinde çalıştığımız sistemin adımları bulunmaktadır. Şekilde de görüldüğü gibi test sistemleri ile veri toplama işlemlerinin yapılması, analiz sistemi ile çeşitli bölgelerden toplanan büyük verinin analiz edilmesi ve gösterge paneli ile sistem durumun gösterilmesi aşamaları görülmektedir.

Mobil cihaz gibi gösterilen test cihazımız gerçek kullanım senaryolarını izleyerek internetteki herhangi bir sunucuya veri içeriği indirme ve yükleme test işlerini yapmaktadır. Bu veri iletişimi sonucunda oluşan paket trafiği, radyo ağı ve paket anahtarlama çekirdek ağı bir bütün halinde incelenerek saniye başına indirme, yükleme miktarları ve gecikme süresi gibi parametreleri hesaplanıp, daha sonra açıklayacağımız “Hadoop Dağıtık Dosya Sistemi” (Hadoop Distributed File System - HDFS) sunucularına yazmaktadır.

Bu sunucularda bölgesel olarak büyük veri analizi yaparak ve sonuçları tez kapsamında geliştirdiğimiz görüntüleme sistemlerinde, bölgesel internet hız karakteristiği olarak gösterilmektedir.



Şekil 1.8 : Geliştirilen sistemin adımları.

1.3.1 Test sistemlerinin oluşturulması (Veri Toplama)

Test sistemleri, belirli aralıklarda internette herhangi bir sunucu ile iletişim bağlantısı kurarak dosya indirme ve dosya yükleme işlemlerini yapmaktadır. Bu veri alışverişinde oluşan giden gelen paketlerin, aradaki ağ bağlantısı dinlenerek analizi yapılıp ve sonuç olarak saniye başına indirilen ve yüklenen veri miktarları ve istemci ile sonucu arasındaki gidip gelme süreleri hesaplanıp ortak bir alana yazılmaktadır.

Test programı Python programlama dili ile yazılmıştır. Program Ubuntu işletim sistemli bir bilgisayarda servis şeklinde yayımlanıp belirli aralıklarda çalışabilir hale getirilmiştir. Böylelikle test sistemleri belirli bölgelere konuşlandırılıp gerçeğe yakın şekilde kullanıcı kullanımına yakın şekilde test senaryolarını uygulayıp sonuçları ortak bir alana yazılmak suretiyle gönderilmektedir. Test sistemlerinin içinde koşacak program hakkında daha detaylı şekilde ilerleyen bölümlerde bahsedilecektir.

1.3.2 Test sonuçlarının analiz edilmesi (Veri Analizi)

Birçok test sisteminden gelen ve ortak bir alana yazılmış büyük veri şeklinde olan test sonuçlarının Hadoop platformunda Map ve Reduce programlama yöntemi algoritmaları ile analizi yapılmaktadır. Birçok alanda konuşlandırdığımız test sistemlerimizin düğüm temelli hız testini yaparak anlık olarak ağ trafiğinin inceleyip, saniye başına indirme ve yükleme veri miktarları ve paketlerin istemci ile sunucu arasında gidip gelme sürelerinin hesaplanıp ortak bir alana yazılmaktadır.

Test sonuçlarının Map ve Reduce programlama yöntemleri algoritmaları ile analizinin yapılması ve sonuç olarak bölgesel veri hızı trafiğinin karakteristik olarak belirlenmesi sağlanmaktadır.

Bu bölümde kullanılan teknolojiler ise Hadoop platformu, Hadoop Dağıtık Dosya Sistemi (Hadoop Distributed File System - HDFS), Map ve Reduce programlama yöntemi algoritmaları, Cloudera platformu, programlamaların yazılacağı Java programlama dili ve sistemin çalışacağı ortamın sanallaştırılması için kullanılan VMware platformudur. Test sonuçlarının analizi ile alakalı daha detaylı bilgi ilerleyen bölümlerde verilecektir.

1.3.3 Analiz sonuçlarının gösterilmesi

Analiz sonuçlarının gösterilmesi bölümü, test sonuçlarının bölgesel analizinden sonra elde ettiğimiz hız karakteristiğini, gösterge paneli yardımıyla gösterdiğimiz

bölümdür. Bu bölümde Map ve Reduce yardımıyla elde ettiğimiz büyük veri analiz sonuçlarını okuyup arayüzü Türkiye haritası olan gösterge panelinde gösterilmektedir.

Kullandığımız teknolojiler arayüz tasarlamak için kullandığımız WPF (Windows Presentation Foundation) ve programı yazdığımız dil olarak da .NET kütüphanesi ile birlikte C# programlama dilidir.

Analiz sonuçlarının anlık olarak gösterilmesi operasyonel mükemmellik ve büyük resmi tam olarak görme açısından önemli bir aşamadır. Operasyonel işlerden sorumlu ekibin problem anını doğru ve hızlı bir şekilde yakalaması ve gerekli operasyonların verimli bir şekilde yapılması için gösterge panelinde analiz sonuçlarını göstermek, sistemin kullanılabilirliğini arttıracaktır. Müşteriye hata anının minimum şekilde yansıması için operasyonel işlerin hızlı ve verimli şekilde yürümesi önemli bir husus olduğundan, mobil servis sağlayıcıları için sistem durumunu anlık gösteren bu gibi gösterge panelleri önemli bir araç olmaktadır.

Bu bağlamda test sistemlerinin veri toplaması ve analiz sistemlerimizin büyük veriyi bölgesel olarak analiz etmesi bölümlerinin sonuncusu olarak analiz sonuçlarının gösterilmesi ile birlikte sistemi bir bütün olarak tamamlamış olacağız.

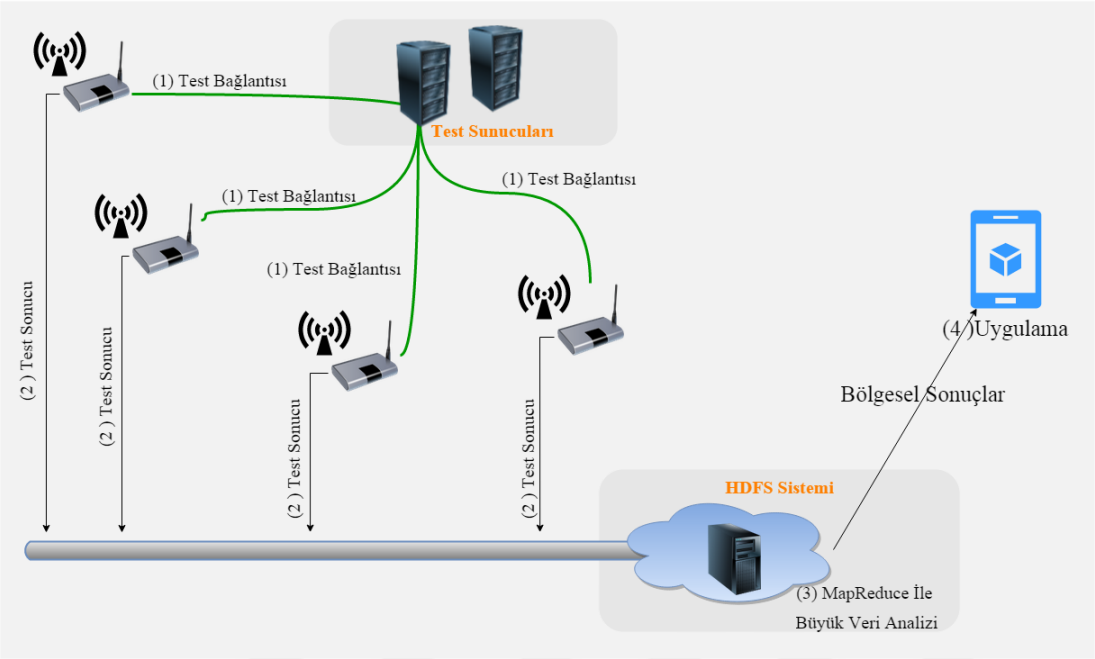
1.4 Sistemin Genel Mimarisi

Mobil veri trafiğinin artması ve mobil ağ altyapısının tamimiyle paket anahtarlama yöntemine geçmesi ile birlikte veri iletişimi trafiğinin izlenmesi ve analiz edilmesi, mobil servis sağlayıcıları için ağ yönetimi ve optimizasyonu bakımından önemli hale getirmiştir. Mobil operatörler internet hizmetini kullanıcılarına daha hızlı sağlamak için kullanıcı deneyimini ölçmek amacıyla ağ trafiğinin uçtan uca izlenmesi ve analiz edilmesi gelişen yeni nesil mobil ağ teknolojileri için önemli olmuştur.

Veri trafiği hızının uçtan uca trafik analizi ile birlikte incelenmesi, müşteri deneyimlerinin gerçeğe yakın şekilde algılanması ve ağ altyapısında oluşabilecek problemlerin müşteri şikâyeti üremeden yakalanması gibi bazı gereksinimler tez kapsamında geliştirdiğimiz sistemin amaçlarındandır.

Yeni nesil mobil ağ teknolojilerinin gelişmesiyle birlikte ağ alt yapısının tamamıyla paket anahtarlama mimarisine dönüşmesi sonucu veri iletişimi trafiğinin incelenmesi gelişim konusu açısından bir hayli önemli bir konu olmuştur. Veri iletişim hızının

kesintisiz ve yüksek olması mobil servis sağlayıcıları açısından müşterilerine sundukları hizmetin temelini oluşturmaktadır. Bundan dolayı sistem kapsamında bu alanda hizmet iyileştirme yapılması ve yeni nesil ağ teknolojilerinin ihtiyaçlarının karşılanması hedeflenmiştir. Şekil 1.9' da sistemin genel mimari yapısı verilmiştir.



Şekil 1.9 : Sistemin genel mimari yapısı.

Sistemin genel mimari yapısına göre incelediğimizde, 4 aşamadan oluşan süreçler şekilde de görüldüğü gibi kademe kademe gösterilmiştir. Sistem genel hatları şöyle açıklanabilir.

Birinci aşamada çeşitli konumlara konuşlandırdığımız test sistemlerimizin belirli aralıklarda kullanıcı hareketlerine yakın şekilde oluşturulmuş test senaryolarını uygulaması yapılmaktadır.

İkinci aşamada bu durumda oluşan veri trafiğinin istemci ve sunucu arasındaki bağlantının dinlenerek giden gelen paketlerinin analizlerinin yapılmaktadır ve bu analizler sonucu elde edilen saniye başına indirilen ve yüklenen veri miktarları ve istemci ile sunucu arasındaki gidip gelme süreleri sonuçlarının bulut ortamına yazılması yapılmaktadır.

Üçüncü aşamada, çeşitli test sistemlerinden gelen test sonuçlarının oluşturmuş olduğu büyük verinin, Hadoop Dağıtık Dosya Sistemi (HDFS) sisteminde Hadoop platformu kullanılarak Map ve Reduce programlama algoritmaları ile birlikte analiz edilmektedir.

Son olarak dördüncü aşamada ise, aşamada ise sistem durumunu göstermek için bu analizlerin gösterge panelinde bölgesel olarak gösterilmesi sağlanmaktadır.

1.5 Diğer Sistemlere Göre Avantajları ve Dezavantajları

Mobil network altyapısını kullanan kullanıcıların deneyimlerini gerçeğe yakın bir şekilde anlaşılması mobil servis sağlayıcıları için önemli bir konudur. Bu konuda geliştirilmiş sistemler, analiz yöntemleri ve operasyon yöntemleri yeni nesil networklerin hayatımıza girmesiyle birlikte önemli bir durum oluşturmuştur [3]. Bu konuda sosyal medya analizleri de müşteri deneyimini ölçmek için yeni bir bakış açısı kazandırmıştır [4].

Şuanki kullanılan sistemlerde müşteri deneyimini ölçmek için canlı ortamdaki trafiği aynalayıp veritabanına kayıt eden ve kayıt edilmiş trafiği çeşitli veri madenciliği ve yapay zekâ algoritmaları ile analiz eden sistemler kullanılmaktadır. Piyasadaki birçok firma bu tür ürünleri mobil ağlara entegre etmek için çeşitli hizmetler vermektedirler. Bu bölümde bu sistemlerin ne olduğundan, avantajlarından, dezavantajlarından bahsedeceğiz ve tez kapsamında geliştirmiş olduğumuz sistemimizle karşılaştıracağız.

Citrix [5] firmasının kullanıcı deneyimini ölçmek için mobil veri ağları için geliştirilmiş "ByteMobile Smart Capacity" [6] isimli ürünü yeni nesil mobil ağlar için yeni bir servis hizmeti sağlamaktadır. Daha önceki bölümlerde de bahsettiğimiz gibi ByteMobile sunucusu düğüm temelli network hareketlerini inceleyip çeşitli analizler sonucunda müşteri deneyimini tahmin etmektedir [7]. Bu sistemin en büyük dezavantajlarından biri uçtan uca servis kalitesini ölçmemektedir. Çünkü herhangi bir internet sayfasına girdiğimizde oluşan veri trafiğini uçtan uca analiz edebilmek için ya istemci tarafından ya da sunucu tarafından trafiği izleyip analiz etmek gerekmektedir. Bu durum da mobil network uçlarında olamayacağı için bu gibi analizler çekirdek networkdeki düğümlerden yapılmaktadır. Bu da bize uçtan uca bir analiz sağlayamadığı için müşteri deneyimini doğruya yakın bir şekilde algılayamamaktadır.

Müşteri deneyimini düğüm temelli ölçen bir diğer sistem olan infovista firması tarafından geliştirilen "End User Experience Monitoring" [8] ürünü de uçtan uca bir analiz çözümü sunmamakla birlikte mobil networkler için akıbeti belli olmayan hava

arayüzünü de analiz edememektedir. Bu ürünün en büyük dezavantajı ByteMobile sisteminde olduğu gibi uçtan uca trafik analizi yapamamasıdır. Bir diğer dezavantajı ise yüklü miktarda oluşan mobil veri trafiğini hızlı bir şekilde analiz edip sonuçları göstermek için birhayli kapasiteli donanım ve enerji ihtiyacının olmasıdır.

Müşteri deneyimini farklı bir yöntemle ölçen bir başka sistem olan Ookla [9] firmasının sistemi "Speed Test" [10] uygulaması diğer bahsettiğimiz sistemlere avantaj olarak uçtan uca network altyapısını [11] uçtan uca ölçen ve müşteri deneyimini gösteren bir sistemdir. Speed Test uygulaması farklı sunuculara gerçek testler uygulanarak veri trafiği hızını ölçmektedir [12]. Bu test sonuçları da kullanıcı memnuniyetini gösteren analiz sonuçlarıdır. Speed Test uygulamasının en büyük avantajı uçtan uca sistemin indirme [13] ve yükleme [14] hız kalitesini ölçmektir. Bu sistemin kendi geliştirdiğimiz sisteme göre dezavantajı ise, kullanıcı bağımlı bir sistem olmasıdır ve networkün dağıldığı tüm alanın hız testinin yapılamammasıdır.

Müşteri deneyimini düğüm temelli ölçen bir diğer yaklaşım olan "Customer Experience Management -CEM" çözümleri de bulunmaktadır [15]. Bu çözümler çeşitli firmaların hizmet kataloglarında bulunmaktadır. Bunlardan bazıları SAS [16], huawei [17] gibi firmalardır. Bu çözümler düğüm temelli network hareketlerini inceleyip çeşitli analizler sonucunda müşteri deneyimini tahmin etmektedir. Bu ürünün en büyük dezavantajı ByteMobile sisteminde olduğu gibi uçtan uca trafik analizi yapamamasıdır. Bir diğer dezavantajı ise yüklü miktarda oluşan mobil veri trafiğini hızlı bir şekilde analiz edip sonuçları göstermek için birhayli kapasiteli donanım ve enerji ihtiyacının olmasıdır.

2. TEST SİSTEMİ UYGULAMASI

2.1 Test Sistemi

Test sistemleri, belirli aralıklarda internette herhangi bir sunucu ile ağ bağlantısı kurarak dosya indirme ve dosya yükleme işlemlerini yapmaktadır. Bu veri alışverişinde oluşan giden gelen paketlerin aradaki ağ bağlantısı dinlenerek analizi yapılıp ve sonuç olarak saniye başına indirilen ve yüklenen veri miktarları ve istemci ile sonucu arasındaki gidip gelme süreleri hesaplanarak ortak bir alana herhangi bir veri yapısına veya teknolojiye bağlı kalmadan yazılmaktadır.

2.2 Test Sistemi İçin Kullanılan Teknolojiler

Test sistemi yazılırken kullandığımız teknolojiler programın koşacağı işletim sistemi Ubuntu işletim sistemi, programın yazımı için kullandığımız dil olan Python programlama dili, TCP bağlantısı analizinde kullandığımız tcptrace, pcap dosyası ve pcap çıktısını analiz ettiğimiz araç olan Wireshark platformudur. Bu teknolojiler hakkında kısaca bilgi verdikten sonra test sistemimizin yazımına başlayabiliriz.

Ubuntu işletim sistemi Linux tabanlı ücretsiz bir işletim sistemidir. Ubuntu işletim sistemini test sistemlerimizde kullanmamız birçok avantajdan dolayı tercih edilmiştir. Tercih sebeplerinden birincisi ücretsiz olması diğeri ise internet sağlayıcı cihazın (terminal), bilgisayara bağlandığı port' un, paket alışverişinde, ağ trafiğinin dinlenmesine olanak sağlamasıdır. Diğer işletim sistemlerinde bu durum bazen zor bazen de imkânsız olmaktadır. Bundan dolayı Ubuntu işletim sistemi test sistemlerimizin işletim sistemi olarak kullanılmıştır. Yazdığımız test programı Ubuntu işletim sistemli bir bilgisayarda servis şeklinde yayımlanıp belirli aralıklarda çalışabilir hale getirilmiştir.

Test sistemimizin yazımında Python programlama dili kullanılmıştır. Python, nesne yönelimli, yorumlamalı, birimsel ve etkileşimli yüksek seviyeli bir programlama dilidir. Test sistemlerimizin yazımı için kullandığımız bu programla dilinin kullanımının kolay, programlamasının hızlı, basit sözdizimi, modüler yapısı, sınıf

dizgesini ve her türlü veri alanı girişini desteklemesi ve hemen hemen her türlü platformda çalışabilmesi özelliklerinin olması sebebiyle tercih edilmiştir. Ağ yazılımlarında kullanımı pratik olan Python programlama dili, ağ trafiğinin dinlenmesi için gerekli fonksiyonlarının yazımı, birçok kütüphanenin yardımıyla birlikte çok daha rahat ve hızlı olmaktadır.

Ağ bağlantısı analizinde kullandığımız "pcap" uzantılı dosya "packet capture" İngilizce kelimelerinin kısaltılmasıyla oluşturulmuştur. "Pcap" uzantılı dosyalar ağ paket verilerini içerir. Bu tür dosyalar veri trafiğindeki istemci ve sunucu arasındaki çeşitli bağlantılar sonucu oluşan giden gelen paketleri belirli bir formatta gösterir ve genel anlamda ağ özelliklerini analiz edilmek üzere kullanılan dosya türüdür.

Test sistemimiz internetteki herhangi bir sunucu ile veri alışverişinde bulunduğu zaman istemci ile sunucu arasındaki tüm giden gelen paketlerden oluşan ağ trafiği "pcap" uzantılı dosyada analizi yapılmak suretiyle kayıt edilmektedir. Linux işletim sisteminde ağ trafiğinin dinlenmesi için gerekli bazı komutlar kullanılarak "pcap" çıktısı elde edilmektedir. Bu komutlardan biri olan "dumpcap" fonksiyonu, internete çıkış yapılan arayüzün dinlenmesi için kullanılmaktadır.

İstemci ve sunucu arasındaki ağ trafiğini kayıt ettiğimiz "pcap" uzantılı dosyayı açıp analiz edebileceğimiz araç olarak Wireshark platformunu kullanılmıştır. Wireshark platformu birçok işletim sisteminde çalışabilen ve bilgisayara bağlı olan her türlü ağ kartlarındaki (Ethernet Kartı veya Modem gibi) tüm TCP/IP mesajlarını analiz edebilen bir programdır. Wireshark ağ trafiğinin dinlenmesinde ve TCP/IP mesajlarının yorumlanıp analiz edilmesine yarayan bir platformdur.

TCP bağlantısında giden gelen paketlerin analizi için kullandığımız "tcptrace" aracı, Ohio Üniversitesinde Shawn Ostermann tarafından yazılmıştır. Her bir bağlantı için çeşitli tip çıktı üreten bu araç gecikme zamanı, gönderilen ve alınan paketlerin dilimlerini ve ne kadar bayt olduğunu, tekrar gönderilen paketlerin bilgisini, paketlerin istemci ve sunucu arasında gidip gelme sürelerini ve saniye başına gönderilen ve alınan paketlerin miktarını göstermektedir.

Bu araç, trafiğin kayıt edildiği "pcap" dosyası verilerek gerekli parametrelerin çıkartılmasında ve bu parametrelerin çeşitli tür uzantılı dosyalara yazılmasında kullanılmıştır. Sistemimizde ağ trafiğinin kayıt edildiği "pcap" dosyası, "tcptrace"

fonksiyonu yardımıyla TCP parametreleri analiz edilerek ve “csv” uzantılı dosyaya yazılmaktadır.

2.3 Paket Analizine Giriş

Test sistemlerimizde, veri alışverişinde oluşan giden gelen paketlerin aradaki ağ bağlantısı dinlenerek analizi yapılarak ve sonuç olarak saniye başına indirilen ve yüklenen veri miktarları ve istemci ile sonucu arasındaki gidip gelme süreleri hesaplanmaktadır.

Bunun için istemci ve sunucun arasındaki trafiğin incelenip gerekli paket analizinin yapılması gerekmektedir. Paket analizi yapılıp bizim için gerekli olan ve kullanıcı deneyimini direk yansıtacak parametreler hesaplanmıştır.

İlk olarak bu kavramlar için kullandığımız ölçü birimlerinden bahsedelim. Bit iletişim ve bilgisayar teknolojilerinde kullanılan en temel veri birimidir. İletişim ve bilgisayar teknolojilerinde 1’ ler ve 0’ lar olmak üzere ikili sayı sistemi kullanılmaktadır. Bilgisayarlarda ve iletişim ağında verilerin aktarılabilmesi için 1’ ler ve 0’ lar kullanılmaktadır. Veri iletişimi sırasında kullanılan 1 bit 1 ya da 0’ a denk gelmektedir. “Kilo”, “mega”, “giga” ve “tera” ön ekleri olmak üzere tüm ölçü birimlerinde olduğu gibi bit ölçü biriminin de katları vardır. Her bir ön ek 10 ve katlarını temsil eder.

Bu ölçüm birimleri istemci ve sunucu arasındaki indirme ve yükleme için veri transfer hızlarını belirtmek için kullanacağız. İndirme ve yükleme hızı olarak saniye başına alınan ve gönderilen veri miktarını gösterirken mbps biriminde ve Megabits Per Second İngilizce kelimelerinin kısaltılmasıyla oluşturulmuş internet servisi sağlayıcıları endüstrisi tarafından kullanılan standarttı kullanacağız. mbps birimi 1 saniyede gönderilen ya da alınan veri miktarını göstermektedir.

Veri miktarını belirtmek için kullanılan bir başka ölçü birimi de bayttır. Bit ile bayt farklı kavramlardır. 8 bitin bir araya gelmesiyle 1 bayt oluşmaktadır. Kısaltmaları ise bayt “B” olarak bit ise “b” olarak yapılmaktadır. Veri miktarı aktarımı hızı hesaplanırken kullanılan ölçü birimi genelde karıştırılır. Saniye başına aktarılan veri miktarını söylerken “B” (büyük B) kullanıldığında baytı “b” (küçük b) kullanıldığında ise bit kastetmiş oluyoruz.

“Kilo”, “mega”, “giga” ve “tera” ön ekleri olmak üzere tüm ölçü birimlerinde olduğu gibi bayt ölçü biriminin de katları vardır. Her bir ön ek 2 ve katları temsil eder.

İletişim ve bilgisayar teknolojilerinde kullanılan en temel kavramlardan olan paket, veri miktarı, bant genişliği, veri trafiği, veri aktarım hızı, indirme hızı, yükleme hızı ve gidiş geliş gecikme parametrelerinden bahsedelim.

Paket kavramı, ağ teknolojilerinde veya internet protokolünde bilginin temel birimidir. Dijital ağ iletişiminde uygun boyuttaki bloklar içerisinde aktarılan verinin küçük parçalara bölünmüş haline paket denir. Paket içerisinde verinin içeriği, verinin gideceği adres ve paket hakkında bir takım bilgiler bulunur.

Dijital iletişimde iki uç arasında giden gelen bu paketler iki ucun haberleşmesine olanak sağlarlar. Paket kavramı aslında ağ katmanındaki protokol veri ünitesine (Protocol Data Unit - PDU) karşılık gelmektedir ama genel anlamda ağ teknolojilerinde istemci ile sunucu arasında giden ve gelen her bir veri parçasına paket adı verilmektedir. Şekil 2.1’ de katmanlı yapı verilmiştir.

Şekil 2.1 : Katmanlı yapı.

Test sistemlerimizde “pcap” uzantılı olarak yakaladığımız paketler şekildeki katman yapısında gösterildiği gibi veri ünite yapısına özgü bilgiler kullanılarak analizi yapılmaktadır. Taşıma katmanında eklenen TCP bölüt alanında bulunan hedef ve kaynak port numaraları, paket seri numaraları, paket tasdik numaraları ve pencere boyutu gibi bazı değerler kullanılmaktadır. Ağ katmanında eklenen IP başlığında hedef ve kaynak ip serileri ve başlık sağlama toplamı gibi değerler kullanılmaktadır. “tcptrace” fonksiyonu yardımı ile yakalanan paketler, katman bazında eklenen başlıklar ışığında analiz edilmektedir.

Veri miktarı, bir verinin bit ya da bayt cinsinden değerine denir. Veri miktarı teknolojinin gelişimiyle ve kullanıcıların bilişim teknolojilerinde gereksinimi artması ile birlikte gün geçtikçe artmaktadır. Günümüzde exabaytlar cinsinden veri miktarlarından söz edilmektedir. Veri istemci ve sunucu arasında iletilen bilgidir. Örneğin bir web sayfasına giriyoruz. Web sitesinin paylaşıldığı sunucudan web sayfasını kendi bilgisayarımıza indiriyoruz. Transferi yapılan her bir doküman veriyi oluşturmaktadır.

Bant genişliği, veri iletişim için kullanılan bir haberleşme kanalının kapasitesini ifade etmek için kullanılır. Bant genişliği, veri iletişim kaynaklarındaki veri miktarının bit/saniye ya da bayt/saniye cinsinden ölçülmesidir. Belli bir süre içinde aktarılabilecek verinin hacmi bant genişliği ile doğru orantılıdır.

Veri trafiği ise, banttaki mevcut veri akışı olarak tanımlanabilir. Veri iletişimi yapan iki ucun arasındaki banttan geçen verilerin yarattığı trafiktir. Test sistemlerimiz sunucuyla bağlantı kurduktan sonra indirme ve yükleme işlemlerini yaparken oluşan veri trafiğini analiz edip indirme hızı, yükleme hızı ve gecikme parametrelerini elde etmektedir.

Veri aktarım hızı, ağ ya da internet ortamında bulunan bir verinin iki nokta arasında iletilmesi sırasında bir saniyede geçen bit ya da bayt sayısıdır. Başka bir deyişle bir saniyede iletilen bit ya da bayt sayısı ne kadar ise veri aktarım hızı o kadardır. Veri aktarım hızı kilobit, megabit, gigabit, terabit ya da kilobayt, megabayt, gigabayt, terabayt ile ölçülür. Daha önce de bahsettiğimiz gibi bir saniyede aktarılan veriyi ifade etmek için mbps (mb/saniye) kısaltması kullanılacaktır.

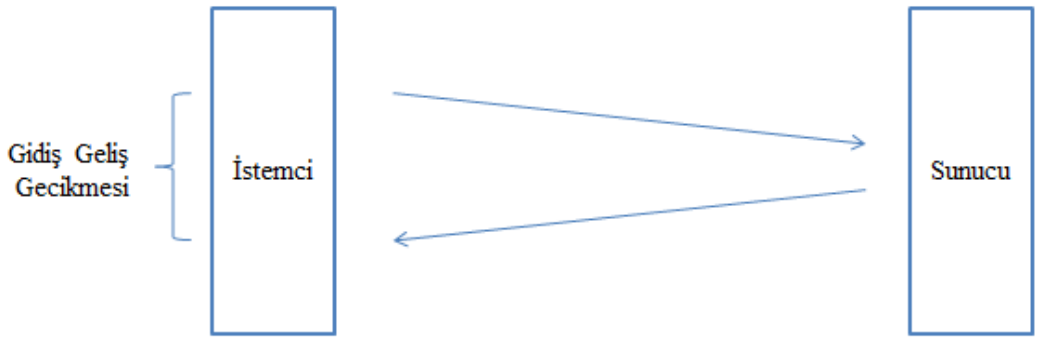
İndirme hızı sunucudan ya da başka bir uçtan indirilen verinin ne kadar hızlı bir şekilde olduğunu gösteren parametredir. Yükleme hızı ise sunucuya ya da başka bir

uca yüklenen verinin ne kadar hızlı bir şekilde olduğunu gösteren parametredir. Test sistemlerimizin canlı trafiği izleyip hesapladığı bir parametre olan saniye başına indirme ve yükleme veri miktarları, bize veri aktarım hızının ne kadar olduğunu belirten bir gösterge olmaktadır.

Gidiş Geliş Gecikmesi, bir bilgi paketinin bir kaynaktan gidip karşı taraftan geri gelmesine kadar geçen süredir. Burada kaynak iletişimi başlatan bir bilgisayar sistemi, varış noktası ise yine bir bilgisayar ya da iletiye cevap veren bir sistemdir. Test sistemlerimiz gidiş geliş gecikmesini milisaniye mertebesinde analiz etmektedir. Gidiş geliş gecikmesi de veri iletim hızını bize gösteren bir parametredir.

Test sistemlerimiz saniye başına indirme ve yükleme veri miktarlarının yanı sıra istemci ile sunucu arasındaki gidip gelme süresini de hesaplayarak test sonucu olarak dönmektedir.

Şekil 2.2' de de görüldüğü gibi iletişimi başlatan düğüm ile diğer düğüm arasındaki mesajların gidip gelme süresinin gösterimi verilmiştir. Bu sürenin miktarı da ağ iletişim kalitesinin nasıl olduğunu bize sayısal olarak göstermektedir.



Şekil 2.2 : Gidiş geliş gecikmesi.

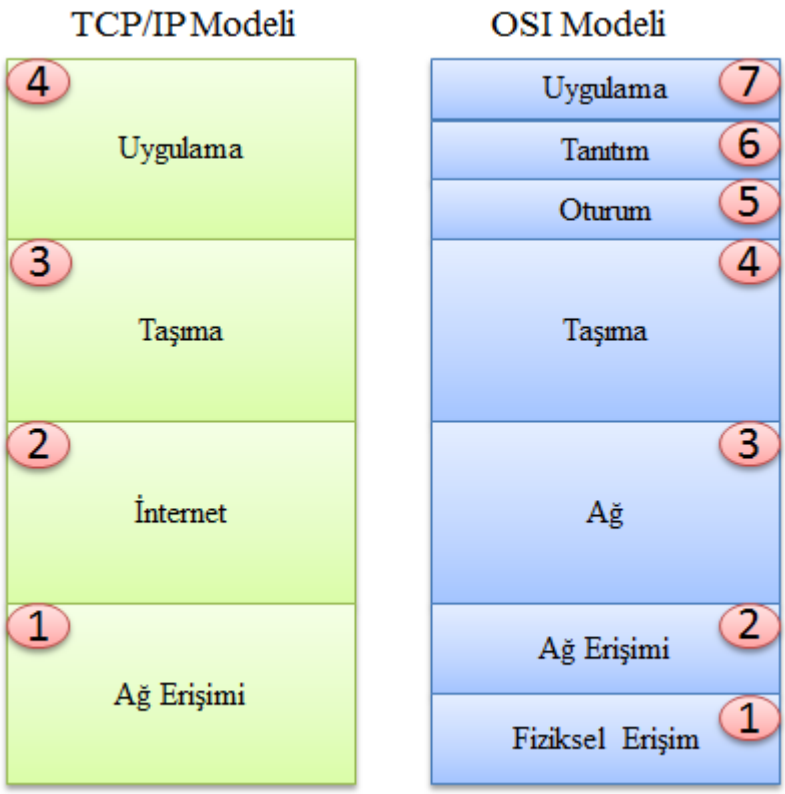
2.4 Paket Analizi

Test sistemlerinde istemci ve sunucu arasında giden gelen paketleri analiz ederek indirme ve yükleme esnasındaki veri transfer hızını ve istemci ile sunucu arasında gidip gelme süresini hesaplanmaktadır. Her bir testin çıktısı da bu hesapladığımız parametreler olmaktadır. İndirme ve yükleme esnasındaki veri transfer hızı ve

istemci ile sunucu arasında gidip gelme süresi ağ katmanı düzeyinden yukarı çıkılarak paketin incelenmesi ile bulunmaktadır.

Ağ iletişimine standart getirilmesi ve ağ iletişim aşamalarının katmanlı olarak gösterilmesi için bazı modellemeler kabul edilmiştir. Bu modellerden biri olan TCP/IP modeli Amerika kaynaklı bir model olup günümüz internet dünyasında kullanılmaktadır. Diğer modellere göre daha önce ortaya çıkması, daha yaygın kullanılması, gerçek dünyada uygulamalarının olması ve ağ teknolojilerinin Amerika kaynaklı olmasından dolayı üstün bir modeldir. TCP/IP modelinin diğer bir model olan OSI modeliyle karşılaştırılması aşağıdaki şekilde verilmiştir.

TCP/IP modeli Şekil 2.3' de de görüldüğü gibi uygulama, taşıma, internet ve ağ erişimi olmak üzere 4 katmandan oluşmaktadır.



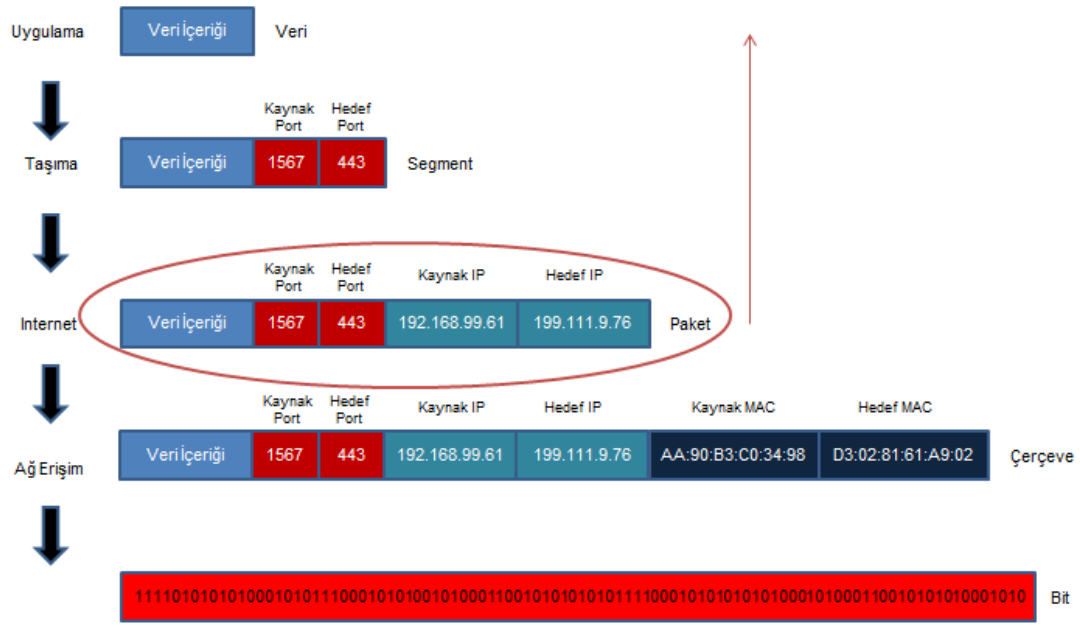
Şekil 2.3 : TCP/IP modeli ve OSI modeli.

Uygulama katmanı, uygulama protokollerini ve sistem programlarının ağı kullanmak için taşıma katmanı hizmetleriyle nasıl bir arabirim oluşturacağını tanımlar. HTTP, Telnet, FTP, TFTP, SNMP, DNS, SMTP gibi bazı protokolleri vardır.

Taşıma katmanı, sistemler arasında iletişim oturumu yönetimi sağlar. Veri taşınırken kullanılan bağlantının hizmet düzeyini ve durumunu tanımlar. TCP, UDP, RTP gibi bazı protokolleri vardır.

İnternet katmanı, verileri IP veri birimleri olarak paketler. Bu paketler, veri birimlerini sistemler ve ağlar arasında iletmek için kullanılan kaynak ve hedef bilgilerini içerir. IP veri birimlerinin yönlendirilmesini gerçekleştirir. IP, ICMP, ARP, RARP gibi bazı protokolleri vardır.

Ağ erişim katmanı, koaksiyel kablo, optik fiber veya çift bükümlü bakır kablo gibi bir ağ ortamıyla doğrudan arabirim oluşturan donanım aygıtları tarafından bitlerin elektriksel olarak nasıl işaret haline getirileceği de kapsam içinde olmak üzere verilerin fiziksel olarak ağ içinden nasıl gönderileceğini belirtir. Ethernet, FDDI, X.25 gibi bazı protokolleri vardır. TCP/IP katmanlı yapısı Şekil 2.4 verilmiştir.



Şekil 2.4 : TCP/IP modeli katmanları.

Yukarıdaki şekilde TCP/IP modelinin katmanlar halinde gösterilmiş örnek yapıları bulunmaktadır. Paket analizini, internet katmanından başlayıp yukarı çıkarak her katmana özgü bilgiler toplanarak yapılmaktadır.

Ağ katmanından istemci ve sunucunun ip adreslerini, taşıma katmanından uygulamanın kullandığı port bilgisi ve uygulama katmanından da verinin içeriği

alınarak analiz süreci başlamış olacaktır. Bu bilgiler test sırasında dinleyeceğimiz ağ trafiği çıktıları kullanılarak yapılmaktadır.

Test sistemi Python programlama dili kullanılarak yazılmıştır ve Ubuntu işletim sisteminde servis şeklinde kurulup belirli aralıklarda çalıştırılması sağlanmıştır. Test programının yazılmasına başlamadan önce ilk olarak algoritmasından bahsedelim. Algoritma olarak şu yol izlenmektedir.

Test sistemi, ilk olarak trafik dinlenmeye başlayacak ve kütüphaneye bağlanılarak test işlemini başlatacaktır. Bağlanılacak kütüphane sonraki adımlarda anlatılacaktır. Kütüphaneden dönen indirme ve yükleme için saniye başına veri transfer miktarları alınmaktadır. Test aşamasında oluşan trafik "pcap" uzantılı dosya halinde alınarak "csv" uzantılı dosyaya çevrilerek analizi yapılmaktadır. İstemci ile sunucu arasında gidip gelme süresi elde edilmektedir. Böylelikle saniye başına veri transfer miktarı ve istemci ile sunucu arasında gidip gelme süresi parametreleri ortak bir alana yazılacaktır. İzlenecek algoritma aşağıdaki şekilde verilmiştir.

Algoritma 1 Speed Meter Algoritması

```
1:
2: while True do
3:    while cagir("dumpcap") do
4:       calistirSpeedTest()
5:    end while
6:    Belirle: sonuclar ⟸ cagir("tcptrace)
7:    if sonuclar =!NULL then
8:       indirme ⟸ hesaplaIndirmeHizi()
9:       yukleme ⟸ hesaplaYuklemeHizi()
10:      gecikme ⟸ hesaplaGecikmeSuresi()
11:   end if
12:   print indirme, yukleme , gecikme
13: end while
```

İndirme ve yükleme esnasındaki saniye başına veri transfer miktarının analizi için Ookla firmasının Speedtest [18] adıyla yayımladığı kütüphane kullanılmaktadır. Speedtest kavramı Türkçe' de "Hız Testi" anlamına gelmektedir. Speedtest dünyanın en çok kullanılan internet hız testi servisidir. İnternet bağlantısının indirme hızını ve yükleme hızını ölçmek için Speedtest uygulaması kullanılmaktadır. Speedtest uygulamasının farklı servislerde kullanabilmemize yarayan bir de uygulama programlama arayüzü vardır. Tez kapsamında geliştirdiğimiz sistemimiz, test

birimlerini yazarken bahsettiğimiz uygulama programlama ara yüzünü kullanmaktadır. Kütüphanenin kullanmamızdaki neden test yaptığımız herhangi bir sunucu dosya yüklemesine izin vermemesi durumu veya bazı güvenlik amaçlı alınmış önlemlerden dolayı testleri daha verimli bir şekilde yapmamızdır. Bir başka neden ise çeşitli alanlarda çeşitli sunuculara testleri gerçekleştirebilme özelliğinin kütüphane kullanımının bize kazandırmasıdır.

Bu nedenlerden dolayı kütüphane kullanmamız testleri daha verimli yapmamıza, farklı bölgelerdeki sunuculara erişebilmemize hem de daha güvenilir saniye başına yüklenen ve indirilen veri miktarlarının ölçülmesine olanak sağlamaktadır.

Kütüphane bileşenlerini ilk olarak yükleyip sonrasında yazdığımız programın içinde çağırarak kullanabiliriz. Kütüphanenin kurulumu aşağıdaki şekilde görülen komut terminale yazılarak yapılabilir.

```
easy_install speedtest-cli
```

Kod 2.1 : Kurulum komutu.

Github üzerinden kurulum yapılabilmesi için de aşağıdaki komutu terminale yazmak yeterli olacaktır.

```
pip install git+https://github.com/sivel/speedtest-cli.git
```

Kod 2.2 : Github kurulum komutu [19].

Kurulumlar yapıldıktan sonra speedtest kütüphanesini, yazacağımız programda kullanabiliriz. Kütüphaneyi kullanabilmek için ilk olarak kod içerisinde tanımlamasının yapılması gerekir. Tanımlama yapıldıktan sonra kütüphene içinde yer alan tüm metodları ve özellikleri yazacağımız program içerisinde kullanabiliriz. Program içinde kullandığımız kütüphaneleri tanımlamak için aşağıdaki kod bloğunda olduğu gibi yazılabilir.

```
import os
import os.path
import sys
import signal
from subprocess import call
import subprocess
import re
import datetime
from multiprocessing import Pool
from time import gmtime, strftime, localtime
```

Kod 2.3 : Kütüphane tanımlamaları.

Trafiği dinlemek için kullanılacak olan trafik dinleme metodunun yazımına başlayabiliriz. Trafik dineleme metodunda ağ trafiğinin dinlenmesi, yakalanan paketlerin kayıt edilmesi, test işlemlerinin başlatılması ve speedtest kütüphanesinin çağrılması gibi başlıca işlemler yapılacaktır. Metodun başlıca prosedülerini gösteren kod bloğu aşağıdaki gibidir.

```
def traceTest(size_str):

    pid = os.fork()

    if pid == 0:
        testWithSpeedTest()
        os._exit(0)

    if pid != 0:
        call(["dumpcap","-i","6","-a",str(size_str),
            "-w","/home/testSystem/Desktop/SpeedMeter/TestOutput.pcap"])
        os.waitpid(pid,0)
```

Kod 2.4 : Canlı trafiği dinleme.

Yukarıdaki kod bloğunu açıklayacak olursak, fork() metodu ile aynı anda birden fazla istemciye hizmet verilebilmesi için aynı özelliklerde açılmış hizmet kanalı oluşturulması sağlanmaktadır. Bu işlemi çok süreçli hizmet kanalı olarak tanımlayabiliriz. Canlı trafiği dinleme metodu paralel olarak iki süreci işlemesi gerekmektedir. Ana süreç canlı trafiğin dinlenmesiyken bu sürece bağlı paralel olarak işleyecek alt süreç ise test işlemlerinin başlatılmasıdır.

Yukarıdaki metodta ana süreç olarak internet bağlantı kanalı dinlenmeye ve pcap çıktısı yazılmaya başlanmış, alt süreç olarak da test işlemlerinin başlatılması için speedtest kütüphanesine bağlanma işlemleri yapılmaktadır.

İnternet bağlantı kanalı, cihazın internet erişimine kablosuz, terminal, ya da başka bir arayüzle mi bağlandığını gösterir. Sistemin hangi arayüzü kullanarak internete çıkış yaptığını görmek için Linux işletim sistemlerinde "dumpcap –D" komutunu terminale yazarak görebiliriz. Hangi arayüz kullanılıyorsa o arayüzü tanımlayan numara, internet bağlantı kanalını dinlemek için çağrılan "dumpcap" fonksiyonuna "-i" olarak parametre şeklinde verilir.

İnternet bağlantı kanalını dinlemek için çağırdığımız dumpcap fonkisyonu ağ trafiğini dinlemek için kullanılan bir yöntemdir. "dumpcap" fonksiyonu canlı ağda oluşan trafikteki paketlerin yazılmasına olanak sağlar ve paketleri "pcap" uzantılı formatta kayıt eder. Metodta kullandığımız dumpcap fonksiyonu "-w" parametresi ile çağrıldığında belirtilen dosya tipi ve uzantıya yazılmasına olanak sağlamaktadır. Fonksiyonu çağırıken kullandığımız "-a" parametresi ise de, yakalama süresinin ya da boyutunu belirten değeri gösterir.

Canlı ağ trafiğini dinlemek ve yakalanan paketlerin kayıt edilmesi için kullanılan "dumpcamp" fonksiyonu program içinde yukarıdaki kod bloğunda görüldüğü gibi çağrılmaktadır. "Dumpcap" fonksiyonu çağırıken kullandığımız parametreler aşağıdaki tabloda gösterilmiştir. Dumpcap, Wireshark aracının bir parçasıdır. Açık kaynaklı kodları ve detaylı bilgileri Wireshark aracının internet sitesinde bulunabilir [20].

Çizelge 2.1 : Dumpcap parametreleri.

Parametre	Kullanım
-i	Dinlemenin yapılacağı ağ arabirim arayüzü
-a	Dinlemenin süresi, paket boyutu veya sayısı
-w	Yakalanan paketlerin formatı ve yeri
-i	Dinlemenin yapılacağı ağ arabirim arayüzü
-a	Dinlemenin süresi, paket boyutu veya sayısı
-w	Yakalanan paketlerin formatı ve yeri

Canlı ağ trafiğinde oluşan paketleri yakalamak için internet bağlantı kanalını dilemeye başladıktan sonra saniye başına indirme ve yükleme veri miktarlarını almak için speedtest kütüphanesine bağlanılması gerekmektedir. Bunun için de test metodunu yazacağız. Test metodunun bazı prosedürleri aşağıdaki kod bloğunda görüldüğü gibidir.

```
def testWithSpeedTest():

    proc = subprocess.Popen(["/usr/local/bin/speedtest-cli"]
                            ,stdout=subprocess.PIPE, shell=True)

    (out, err) = proc.communicate()

    speedTestOutputFileForWriting =
            open('/home/testSystem/Desktop/SpeedMeter/SpeedTestOutput.txt','w')

    speedTestOutputFileForWriting.write(out)

    speedTestOutputFileForWriting.close()
```

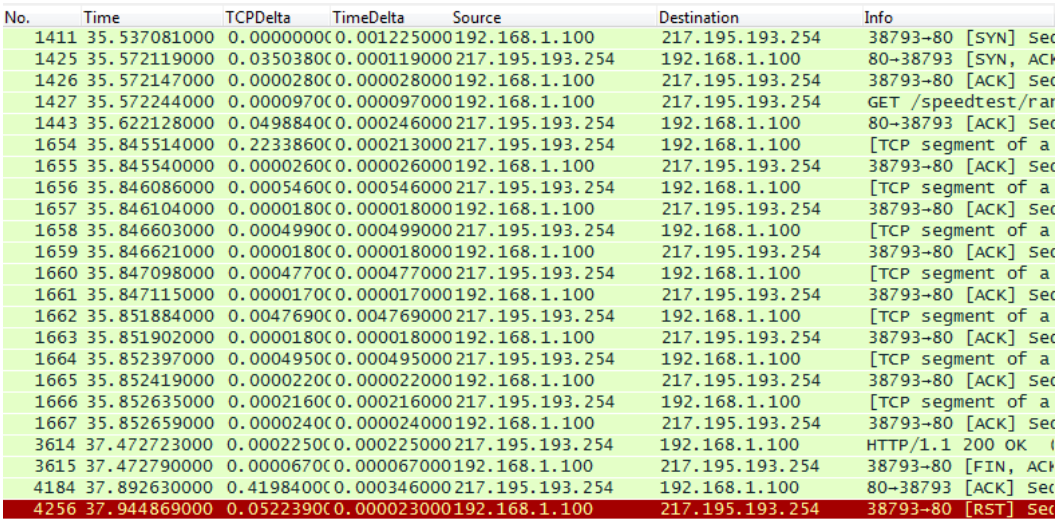
Kod 2.5 : SpeedTest kütüphanesi kullanımı.

Saniye başına indirme ve yükleme veri miktarlarını almak için önceden yüklediğimiz speedtest programlama arayüzünün "Popen()" metodu, kütüphanenin bulunduğu dizin parametresi verilerek kullanılmaktadır. Böylelikle kütühanede yazılmış tüm metodları kod içeresinde istediğimiz gibi kullanabiliriz.

Speedtest kütüphanesi çağrıldıktan sonra "out" parametresiyle saniye başına indirme ve yükleme veri miktarlarına ulaşabiliriz. Bu aşamada speedtest kütüphanesi, anlaşmalı olduğu firmaların sunucuları arasında, test sistemine en yakın olanını seçmektedir ve seçtiği sunucudan çeşitli büyüklükte dosya indirilmesi ve yüklenmesi işlemlerini yapmaktadır. Bu testlerin sonucunda saniye başına indirilen (indirme hızı) ve yüklenen (yükleme hızı) veri miktarlarını hesaplamaktadır.

Yazdığımız test metodu ile birlikte indirme hızını ve yükleme hızını elde etmiş olduk. Bundan sonraki aşamalarda bu testler sonucu oluşan tarfik kayılarını inceleyerek istemci ile sunucu arasında gidip gelme sürelerinin hesaplanması işlemlemlerini yapacağız.

İstemci ve sunucu arasında yapılan test işlemlerinin oluşturduğu trafik "pcap" formatında kayıt edilmektedir. Aşağıdaki şekilde yakalanan paket trafiğinin bir bölümü verilmiştir. Yakalanan trafik "tcp.stream eq 26" filtrelemesi yapılarak aşağıdaki çıktı elde edilmiştir. Bu filitrenin yapılmasının sebebi 26. TCP iletişim akışını görmek içindir. Örnek olarak küçük bir akışı göstermek için böyle bir filitre çalıştırdık. Elde ettiğimiz sonucun filitre uygulanmış halini aşağıdaki şekilde görebiliriz.



No.	Time	TCPDelta	TimeDelta	Source	Destination	Info
1411	35.537081000	0.0000000(	0.001225000	192.168.1.100	217.195.193.254	38793→80 [SYN] Sec
1425	35.572119000	0.0350380(	0.000119000	217.195.193.254	192.168.1.100	80→38793 [SYN, ACK
1426	35.572147000	0.0000280(	0.000028000	192.168.1.100	217.195.193.254	38793→80 [ACK] Sec
1427	35.572244000	0.0000970(	0.000097000	192.168.1.100	217.195.193.254	GET /speedtest/ran
1443	35.622128000	0.0498840(	0.000246000	217.195.193.254	192.168.1.100	80→38793 [ACK] Sec
1654	35.845514000	0.2233860(	0.000213000	217.195.193.254	192.168.1.100	[TCP segment of a
1655	35.845540000	0.0000260(	0.000026000	192.168.1.100	217.195.193.254	38793→80 [ACK] Sec
1656	35.846086000	0.0005460(	0.000546000	217.195.193.254	192.168.1.100	[TCP segment of a
1657	35.846104000	0.0000180(	0.000018000	192.168.1.100	217.195.193.254	38793→80 [ACK] Sec
1658	35.846603000	0.0004990(	0.000499000	217.195.193.254	192.168.1.100	[TCP segment of a
1659	35.846621000	0.0000180(	0.000018000	192.168.1.100	217.195.193.254	38793→80 [ACK] Sec
1660	35.847098000	0.0004770(	0.000477000	217.195.193.254	192.168.1.100	[TCP segment of a
1661	35.847115000	0.0000170(	0.000017000	192.168.1.100	217.195.193.254	38793→80 [ACK] Sec
1662	35.851884000	0.0047690(	0.004769000	217.195.193.254	192.168.1.100	[TCP segment of a
1663	35.851902000	0.0000180(	0.000018000	192.168.1.100	217.195.193.254	38793→80 [ACK] Sec
1664	35.852397000	0.0004950(	0.000495000	217.195.193.254	192.168.1.100	[TCP segment of a
1665	35.852419000	0.0000220(	0.000022000	192.168.1.100	217.195.193.254	38793→80 [ACK] Sec
1666	35.852635000	0.0002160(	0.000216000	217.195.193.254	192.168.1.100	[TCP segment of a
1667	35.852659000	0.0000240(	0.000024000	192.168.1.100	217.195.193.254	38793→80 [ACK] Sec
3614	37.472723000	0.0002250(	0.000225000	217.195.193.254	192.168.1.100	HTTP/1.1 200 OK (
3615	37.472790000	0.0000670(	0.000067000	192.168.1.100	217.195.193.254	38793→80 [FIN, ACK
4184	37.892630000	0.4198400(	0.000346000	217.195.193.254	192.168.1.100	80→38793 [ACK] Sec
4256	37.944869000	0.0522390(	0.000023000	192.168.1.100	217.195.193.254	38793→80 [RST] Sec

Şekil 2.5 : Yakalanan paketler.

Yakalanan paketlerin üzerinden genel kapsamda geçecek olursak,

- 1411 nolu paket, istemciden sunucuya giden istek paketi (SYN),
- 1425 nolu paket, sunucudan istemciye dönen istek cevabı (SYN, ACK),
- 1426 nolu paket, istemciden sunucuya dönen istek cevabını aldığını gösteren paketdir (ACK).

Böylelikle 1411, 1425 ve 1426 nolu paketler istemci ile sunucu arasında üçlü el sıkışmasının yapıldığını ve artık veri alışverişine başlayabileceklerini gösteren paketlerdir.

- 1427 nolu paket, istemciden sunucuya gönderilen istek (GET) mesajıdır.
- 1654 ve 1667 arasındaki paketler, sunucudan istemciye gönderilen verileri ve istemciden sunucuya gönderilen bu verileri başarılı bir şekilde aldığını gösteren paketleri içermektedir.
- 3614 nolu paket, sunucudan istemciye gönderilen istekte bulunulan dosyanın tamamının indirildiğini gösteren mesajdır.
- 3615, 4184 ve 4256 nolu paketler, ise istemci ile sunucu arasındaki bağlantının karşılıklı sonlandırılmasını gösteren mesajlardır.

Yakalanan bu paketler istemci ile sunucu arasında veri alışverişinde gerçekleşen bağlantının kurulmasını, veri transferlerini ve bağlantının sonlandırılmasını göstermektedir. Gecikme süresi bu paketlerin analizi yapılarak, herbir paketin bir

uçtan diğer uca gitmesi ve cevabının gelmesi boyunca geçen sürelerin ortalaması alınarak bulunmaktadır.

Kayıt edilen dosya içinde yakalanan paketlerin analizi yapılarak istemci ile sunucu arasında gidip gelme sürelerinin hesaplanmasını trafik analizi metdoduyla yapılmaktadır. Metodun bazı fonksiyonları aşağıdaki kod bloğunda görüldüğü gibidir.

```
def analyseTestTrace():

    file_awk_output =
        open('/home/testSystem/Desktop/SpeedMeter/PcapAnalyses.txt','w')
    file_awk_output.write("")
    file_awk_output.close()

    call("tcptrace
        "--csv -lnr /home/testSystem/Desktop/SpeedMeter/TestOutput.pcap"
        "> /home/testSystem/Desktop/SpeedMeter/TestOutput.csv",shell=True)

    awkstr = "awk -F\, '{ if(NF>50) {rtta2b+=$96;rttb2a+=$97;cnt+=1;}} "
        "END {printf(\"RTT_avg_a2b=%s RTT_avg_b2a=%s"
        "cnt=%s \\n\",rtta2b/cnt,rttb2a/cnt,cnt)}' "
        "/home/test-heker/Desktop/SpeedMeter/NEW/TestOutput.csv >>"
        "/home/test-heker/Desktop/SpeedMeter/NEW/PcapAnalyses.txt"

    file_awk_output =
        "open('/home/testSystem/Desktop/SpeedMeter/PcapAnalyses.txt','a+')
    file_awk_output.write("\n96:RTT_avg_a2b\t97:RTT_avg_b2a\n")
    file_awk_output.close()
    call(awkstr,shell= True)
```

Kod 2.6 : Trafik analizi.

Trafik analizi metodunda çağırdığımız “tcptrace” fonksiyonu, giden gelen paketlerin analizinin yapılması için kullanılmaktadır. Bu işlemle birlikte her bir bağlantı için çeşitli tip çıktı üreten bu araç gecikme zamanı, gönderilen ve alınan paketlerin dilimlerini ve ne kadar bayt olduğunu, tekrar gönderilen paketlerin bilgisini, paketlerin istemci ve sunucu arasında gidip gelme sürelerini ve saniye başına gönderilen ve alınan paketlerin miktarını göstermektedir. Bu araç, trafiğin kayıt edildiği “pcap” formatındaki dosya verilerek gerekli parametreleri çıkartılmasında ve bu parametrelerin “csv” uzantılı dosyaya yazılmasında kullanılmaktadır.

Yukarıdaki kod bloğunda da görüldüğü gibi “tcptrace” fonksiyonu çağırılırken trafiğin kayıtlı olduğu “pcap” dosyası ve dönüşümü yapılacak “csv” uzantılı dosyanın dizinlerini parametre olarak verilmektedir. Dosyanın dönüşümünden sonra

“csv” uzantılı dosyadaki ilgili hücrelerde istemci ile sunucu arasında giden gelen paketlerin çeşitli özelliklerini görebiliriz.

Trafik analizi metodunda 96 ve 97. hücrelerde yer alan “RTT_avg_a2b” ve “RTT_avg_b2a” değerlerini alacağız. “RTT_avg_a2b” değeri indirme için gidip gelme süresini verirken, “RTT_avg_b2a” değeri ise yükleme için gidip gelme süresini vermektedir.

Böylelikle speedtest kütüphanesi yardımıyla indirme ve yükleme hızları ve paket analizi ile birlikte de istemci ile sunucu arasında gidip gelme süreleri elde edilmiş oldu. Sonuç olarak bu değerler test sisteminin numarası ile ortak bir alana yazılmaktadır.

Test sistemi için yazdığımız program Ubuntu işletim sisteminde servis şeklinde yayımlanıp belirli aralıklarda test sisteminin çalışması sağlanmaktadır. Yazılan programın Linux işletim sistemlerinde servis şeklinde kurulması için “crontab -e” komutu terminal ekranına yazılarak gerekli zaman ayarlamaları yapmak suretiyle kurulmuş olacaktır.

3. BÜYÜK VERİ ANALİZ SİSTEMİ UYGULAMASI

3.1 Büyük Veri Analiz Sistemi

Gün geçtikçe gelişen teknoloji ile birlikte veri her alanda üretilmeye başlanmış ve bunun sonucunda oluşan büyük veri analizi bir hayli önemli bir konu olmuştur. Mobil iletişim araçları, akıllı sayaçlar, elektronik ortam platformları, ulaşım araçları, almaçlar ve bunun gibi dijital ortama kayan diğer çeşitli platformlar sürekli olarak veri üretmektedirler. Oluşan büyük veri yığını içerisinden anlamlı ve işe yarayan bilgi üretmek tüm hizmet alanları için önemli bir konu haline gelmiştir.

Büyük veri içerisinde çok değerli hatta stratejik olarak üstünlük sağlayacak çok önemli bilgiler vardır. Pazarlama ve müşteri talebi analizi, müşteri karakteri analizi, ürün önerileri, sunucu hareketlerini inceleme, sahtecilik analizi, gerçek zamanlı raporlama ve kullanıcı deneyimini geliştirme gibi çok daha fazla sayabileceğimiz bilgileri çok büyük veriler içerisinden analiz etmek suretiyle elde edebiliriz.

Diğer servis alanlarında olduğu gibi mobil servis sağlayıcıları için de büyük veri analizi çok önemli bir konu olmuştur. Mobil servis ağlarında da sürekli olarak büyük veri oluşmakta ve bu veriler servis sağlayıcıları için önemli bir kazanım olmaktadır. Kullanıcı hareketleri örüntüsü, çeşitli sistem düğümlerinin kayıt ettiği veriler veya ağ trafik hareketleri gibi birçok örnek verebiliriz.

Bu sistemlerin oluşturmuş olduğu büyük veriler analiz edilerek çok değerli bilgiler elde edilebilir ve ağ kalitesinin veya kullanıcı deneyiminin ölçülmesi, ağ trafik güvenliğinin sağlanması, problemlerin önceden tespit edilmesi veya stratejik pazarlama adımlarının belirlenmesi gibi değişik alanlarda bu bilgiler kullanılabilir.

Bu çalışmamızda test sistemlerimizin üretmiş olduğu büyük veriyi, analiz ederek uçtan uca trafik analizinin yapılması, kullanıcı deneyimini gerçeğe yakın bir şekilde ölçülmesi, problemlerin müşteri şikâyeti gelmeden tespit edilip önlenmesi ve ağ kalitesinin tespit edilip bu alandaki gelişimlerin devreye alınması gibi bazı kazanımlar elde edilmeye çalışılmıştır.

Çeşitli alanlara konuşlandırılmış test sistemleri belirli aralıklarda hız testleri yapıp sonuçları ortak bir alana yazmaktadırlar. Test sistemlerinin ortak alana yazmış olduğu test sonuçları büyük veri yığınını oluşturmaktadır. Oluşan bu büyük veri, hızlı bir şekilde test alanına göre analiz edilip internet hız karakteristiğini bölgesel olarak çıkarılması gerekmektedir. Bu bağlamda Hadoop platformu kullanılarak Map ve Reduce programlama algoritmaları ile büyük veri analizi yapılmaktadır.

3.2 Büyük Veri Analizi İçin Kullanılan Teknolojiler

Büyük veri analizi için kullandığımız başlıca teknolojiler vardır. Çalışmamızda kullanılan başlıca teknolojiler şu şekildedir.

Büyük verinin analiz edileceği sistem olan Hadoop platformu, büyük veriyi sistematik olarak analizini yaptığımız Map ve Reduce programlama algoritmaları, Hadoop platformunu kurulumunun yaptığımız Cloudera aracı ve Cloudera aracını kurduğumuz sanal makine ortamını sunan VMware uygulaması, kullandığımız başlıca teknolojilerdir. Bu teknolojiler hakkında kısa bilgi verdikten sonra büyük veri analizi uygulamasının oluşturulmasına başlanabilir.

Sanallaştırma, günümüz bilgisayarlarının çok sayıda işletim sistemi ve uygulamalarla çalışmasını mümkün kılarak, altyapınızı daha basit ve daha etkili hale getirmektedir. VMware uygulaması sistemlerin sanallaştırılmasına ve aynı anda çeşitli işletim sistemlerini ve uygulamaları çalıştırmasına olanak sağlayacak altyapı sunan bir çözümdür. Çalışmamızda kullandığımız Cloudera sisteminin kurulumunu yapmak üzere bilgisayarımızı sanallaştırmamız gerekmektedir. Çalıştığımız bilgisayarı sanallaştırmak için VMware uygulaması kullanılmıştır.

Cloudera aracı, veri odaklı işletmelerin yapısal veya yapısal olmayan verilerinin analiz edilerek anlamlı bilgiler ve ilişkiler çıkarılmasını sağlayan Apache Hadoop tabanlı açık kaynak kodlu yazılımın destek ve servis hizmetlerini sağlayan bir çözümdür. Cloudera, Apache Hadoop' un çekirdek ve ekosistem projelerini bir araya getirerek büyük veri analiz ortamını sağlamaktadır.

Hadoop, sıradan sunuculardan oluşan dağıtık mimari üzerinde, büyük verileri işlemek amaçlı uygulamaları çalıştıran, Hadoop Dağıtık Dosya Sistemi (Hadoop Distributed File System -HDFS) olarak adlandırılan ve Map ve Reduce özelliklerini bir araya getiren, Java ile geliştirilmiş açık kaynaklı bir kütüphanedir.

Hadoop, dağıtık dosya sistemi ile Map ve Reduce bileşenlerini bir araya getiren bir yazılımdır. 2005 yılında açık kaynak yazılımı olacak şekilde Apache projesi olarak Yahoo mühendisleri tarafından başlatılmıştır. Verimli şekilde veri arşivi, veri analizi, veri toplama, veri dönüşümleri ve veri örüntüsü karşılaştırma gibi işlerde kullanılan Hadoop sistemin ismi lider mühendis Doug Cutting' in oğlunun oyuncak filinin adından gelmektedir. Hadoop platformunun kökeni aslında açık kaynaklı web arama motoru projesi olan ve Apache Lucene projesinin bir parçası şeklinde geliştirilmiş Nutch projesinden gelmektedir. Sonrasında Google tarafından duyurulan bu sistem aslen 1960'lı yıllarda geliştirilen fonksiyonel programlamadaki map ve reduce fonksiyonlarından esinlenilerek yazılmıştır.

Hadoop platformu ile Facebook, Twitter, Amazon, Google, eBay gibi şirketlerin geleneksel veri tabanları sistemlerini bırakmalarıyla birlikte büyük veri analizini hızlı şekilde yaparak müşterilerine iyi bir hizmet sağlamaktadırlar. Hadoop' un sistemi hakkında daha detaylı teknik bilgi ilerleyen bölümlerde verilecektir.

Map ve Reduce algoritmaları, dağıtık mimari üzerinde çok büyük verilerin kolay bir şekilde analiz edilebilmesini sağlayan bir programlama modelidir. Java, Python, C#, C++ gibi farklı dillerle kullanılabilir. Büyük veri kümelerini analiz edebilmek için paralel süreçler oluşturmaktadır.

Map ve Reduce iki adımdan oluşmaktadır. İlk adımı olan Map fonksiyonu, verinin okunup daha ufak parçalara ayrılmasını ve istenenin çıktı olarak dağıtık sistemlere dağıtılmasını sağlar. Diğer adım olan Reduce fonksiyonu ise, tamamlanan işler işin mantığına göre birleştirilerek sonuç elde eder. Bir başka anlatımla, Map aşamasında analiz edilen veri içerisinden almak istediğimiz veriler çekilir; Reduce aşamasında ise bu çektiğimiz veri üzerinde istediğimiz analiz gerçekleşir.

Map aşaması bağımsız olarak gerçekleşebildiği için paralel olarak çalışabilir. Bu sayede büyük miktardaki veri, düğümler tarafından hızlı bir şekilde okunabilir.

Reduce aşamasında ise aynı anahtara sahip veriler paralel olarak işlenebilir ve sonuç hızlı bir şekilde elde edilir. Map ve Reduce algoritmaları için daha detaylı bilgi ilerleyen bölümlerde verilecektir.

Hadoop platformu ve kullanılan Map ve Reduce algoritmaları yardımıyla, çok büyük bir veriyi işleyebilmek için çok yüksek donanıma sahip sunucular kullanmak yerine sıradan sunuculardan oluşan bir küme üzerinde aynı işlem çok daha etkin bir şekilde

gerçekleştirilir. Bundan dolayı büyük veri çok daha hızlı analiz edilmekte ve büyük veri yönetimi daha kolay olmaktadır.

3.3 Büyük Veri Analizine Giriş

Test sistemlerinin bulut ortamına yazmış olduğu büyük veri yığınını olan test sonuçlarının analiz etmek için Hadoop platformu ile Map ve Reduce algoritmaları kullanılmaktadır.

Hadoop, sıradan sunuculardan oluşan dağıtık mimari üzerinde, büyük verileri işlemek amaçlı uygulamaları çalıştıran, Hadoop Dağıtık Dosya Sistemi (HDFS) olarak adlandırılan ve Map ve Reduce özelliklerini bir araya getiren, Java ile geliştirilmiş açık kaynaklı bir kütüphanedir. Hadoop Sistemin günümüzde kullanılan İlişkisel Veritabanı Yönetim Sistemleri' ne göre farklı özellikleri vardır. Bu özellikler verinin her geçen gün katlanarak çoğaldığını düşünürsek klasik veritabanları sistemlerine göre üstünlük sağlamaktadır.

Aşağıdaki tabloda Hadoop Sistemi ile İlişkisel Veritabanı Yönetim Sistemleri' nin karşılaştırılma tablosu verilmiştir.

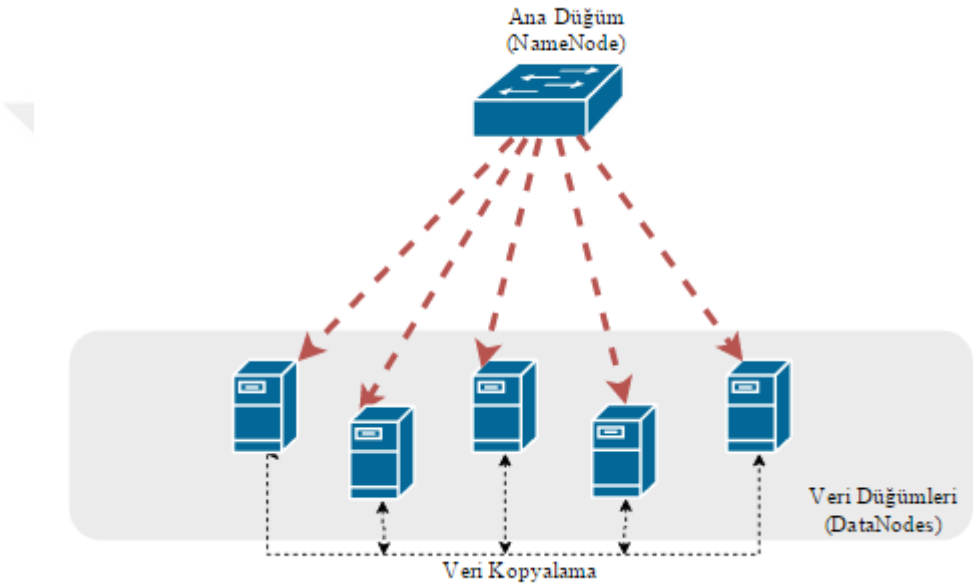
Çizelge 3.1 : Hadop ve İlişkisel Veritabanı Yönetim Sistemi' nin karşılaştırılması.

Hadoop	İlişkisel Veritabanı Yönetim Sistemi
Veri boyutu daha büyüktür. Petabayt boyutlarındadır.	Veri boyutu daha küçüktür. Gigabayt boyutlarındadır.
Yığın işlemleri yapılır.	İnteraktif ve yığın işlemleri için kullanılabilir.
Veri birkere yazılır ve birçok kez okunur.	Okuma ve yazma işlemi birçok kez olabilir.
Değişken veri yapısı vardır.	Statik veri yapısı vardır.
Veriler arasında ilişki yoktur.	Veriler arasında ilişki vardır.
Bütünlük azdır.	Bütünlük çoktur.

Hadoop kütüphanesi Dağıtık Dosya Sistemi ve Map ve Reduce bileşenlerinden oluşmaktadır. Dağıtık Dosya Sistemi, sunucular ve sunucuların dosya sistemlerinin yönetimini sağlamaktadır. Map ve Reduce algoritmaları ise yönetilen bu sunucuların dosya sistemlerinde yer alan dağıtık veri yığınlarını paralel süreçler oluşturarak işlenmesini sağlayan bileşenlerdir. Dağıtık Dosya Sistemleri, Map ve Reduce bileşenlerini daha detaylı olarak ilerleyen aşamalarda göreceğiz.

3.3.1 Dağıtık dosya sistemi (Hadoop Distributed File System - HDFS)

Dağıtık Dosya Sistemi, sıradan sunucuların disklerini bir araya getirerek büyük ve tek bir sanal disk oluşturmaktadırlar. Bu sayede çok büyük boyutta birçok dosya bu sanal disk sisteminde saklanabilir. Bu dosyalar bloklar halinde (varsayılan 64MB) birden fazla ve farklı sunucu üzerine (varsayılan 3 kopya) dağıtılarak yedeklenir. Sistemin bu özellikleriyle birlikte veri kaybı önlenmiş olur ve çok büyük boyutlu dosyalar üzerinde okuma işlemi imkânı sağlamaktadır. Bu özelliklerin yanı sıra rastlantısal erişim özelliği bulunmaz.



Şekil 3.1 : Dağıtık dosya sistemi mimarisi.

Dağıtık Dosya Sistemi "Name Node" adı verilen ana düğüm ve "Data Node" adı verilen veri düğümlerinden oluşmaktadır.

Ana düğüm, blokların sunucular üzerindeki dağılımından, yaratılmasından, silinmesinden, bir blokta sorun meydana geldiğinde yeniden oluşturulmasından ve her türlü dosya erişiminden sorumludur. Tüm dosyalar hakkındaki bilgileri ihtiva eden ana düğüm bu bilgilerin saklanması ve yönetilmesinden sorumludur. Her kümede yalnızca bir adet bulunmaktadır.

Veri düğümünün işlevi ise veri bloklarını saklamak için görevlendirilmiş işçi süreçtir. Her veri düğümü kendi yerel diskindeki veriden sorumludur ve bunun yanı sıra diğer veri düğümlerindeki verilerin yedeklerini de muhafaza eder. Bu düğümler

sistemde birden fazla olabilirler. Dağıtık Dosya Sisteminin genel mimari yapısı yukarıdaki şekilde verilmiştir.

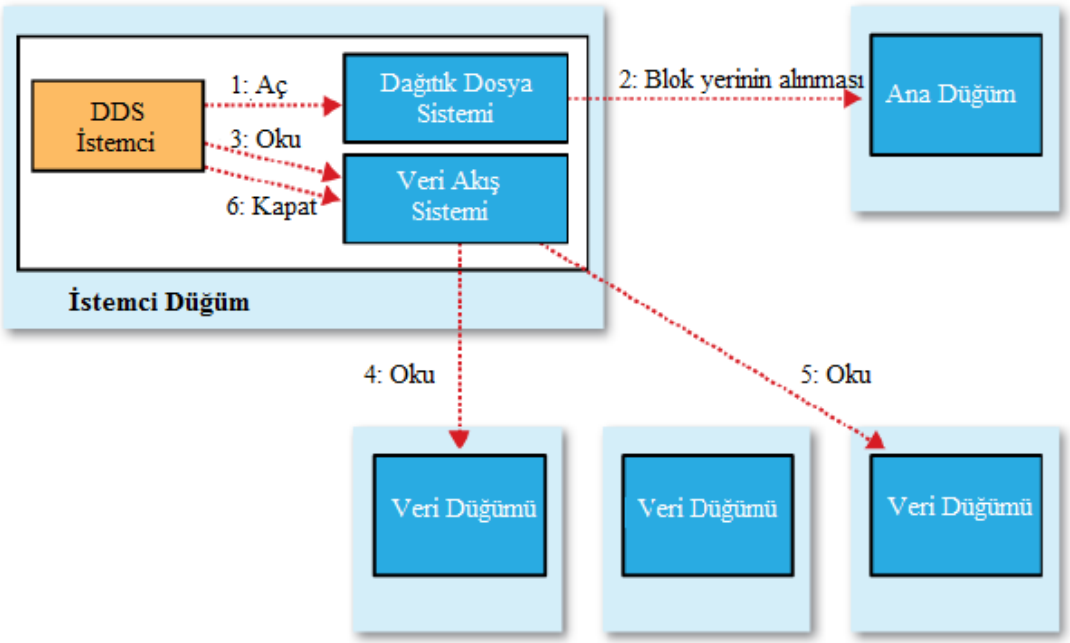
Dağıtık Dosya Sistemi' nin yapısı, sıradan sunucuların diskleri bir araya gelerek büyük ve tek bir sanal disk oluşturmaktadırlar. Dağıtık Dosya Sistemi, Hadoop sisteminin veri depolama ve yönetimi bakımından en önemli bir parçasıdır. Dağıtık Dosya Sistemi' nin özelliklerini aşağıdaki gibi sıralayabiliriz.

- Hadoop DDS' de bloklar kullanılarak veri depolar. 64 MB veya 128 MB kapasiteli bloklar kullanılır. Bir dosya tek bir diskin boyutundan büyük olabilir. Bir dosya ya da yığın blok boyutundan küçük ise sadece gerekli olan alan kullanılır. Örneğin 420 MB' lık bir verimizin olduğunu düşünürsek, 420 MB = 128 MB + 128 MB + 128 MB + 36 MB olacak şekilde bloklara paylaşım yapılır.
- DDS' de temel dosya sistemi operasyonları kullanılmaktadır. Dosya okunabilir, klasör oluşturulabilir, dosyalar taşınabilir, veriler silinebilir, klasörler listelenebilir. "fs -help" komutu girilerek komutlar hakkında detay bilgiler alınabilir. Tanımlı birçok dosya sistemi komutları vardır. Çalışmamız süresince bu komutlardan bazıları kullanılmıştır. Kullandığımız komutların bazıları aşağıdaki tabloda verilmiştir.

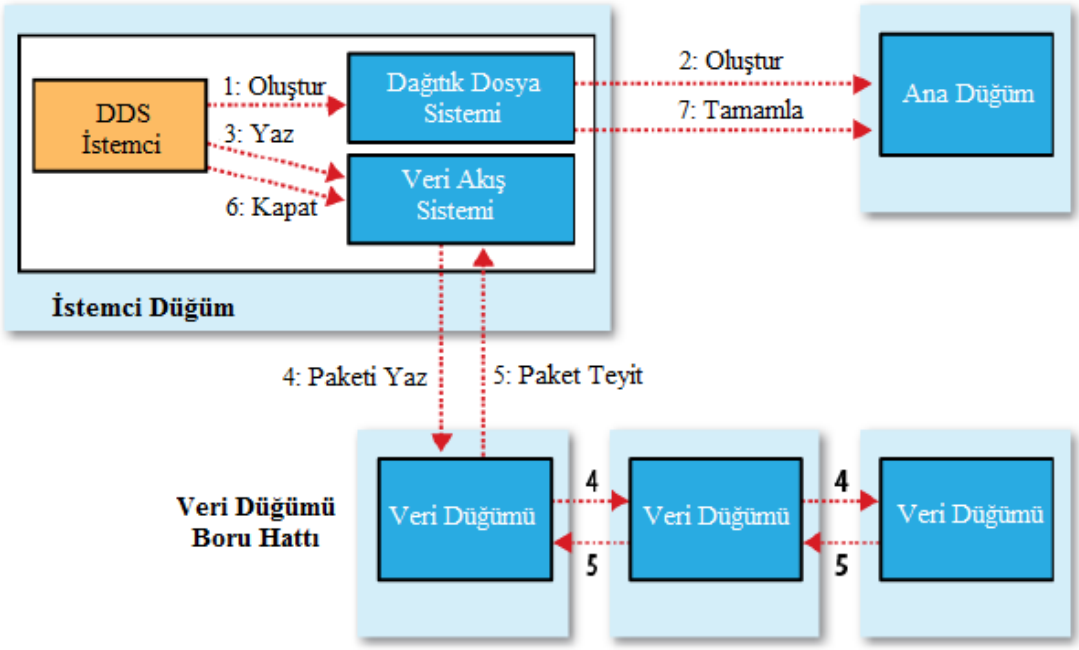
Çizelge 3.2 : DDS dosya sistemi komutları.

Komut	Kullanım
hadoop fs -mkdir [isim]	Klasör oluşturma
hadoop fs -rmr [yol]	Dosya yada klasör silme
hadoop fs –count [yol]	Klasör içindeki dosyaların sayısını alma
hadoop fs -cp [yol] [yol]	Dosya kopyalama
hadoop dfs –df [yol]	Klasör içindeki kullanılabilir alanı gösterme
fs -help	Yardım dosyasını görüntüleme

- DDS veriyi birkere yazıp, birçok kez okumayı sağlar. Verinin DDS sisteminden okunmasını ve yazılmasını gösteren şekiller aşağıda verilmiştir.



Şekil 3.2 : İstemcinin DDS' den veri okuması [21].



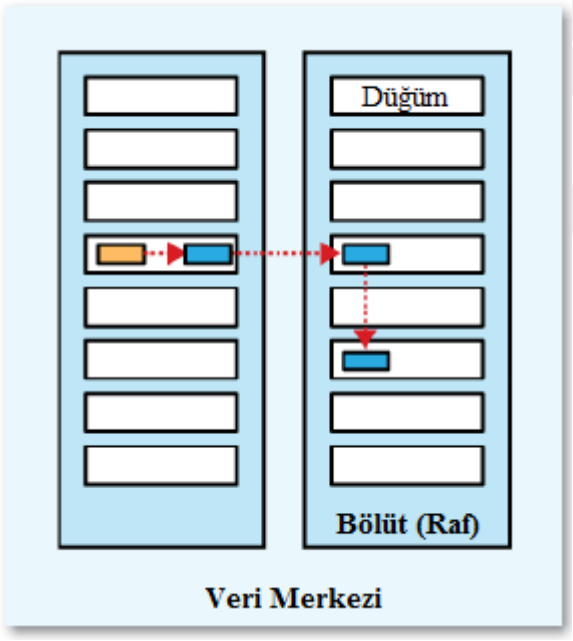
Şekil 3.3 : İstemcinin DDS' e veri yazması [21].

- DDS büyük boyutlu dosyalar için eniyilenmiştir. Bu bakımdan büyük dosyaların depolanması, kopyalanıp diğer disklerde çoğaltılması ve yönetilmesi gibi operasyonel işler verimli bir şekilde yapılmaktadır.
- DDS verinin birden fazla kopyasını tutmaktadır. Verinin kopyalanması düğüm çalışmaz hale geldiğinde veri kaybının önlenmesini sağlamaktadır. Bu özellik ile

birlikte büyük verilerin kaybolması veya ulaşılamaması durumları risklerini azaltmaktadır. Genelde 3 kopya olacak şekilde veriler çoğaltılmaktadır.

Veri düğümlerinde kopyalamanın nasıl yapılacağı önemli bir konudur. Çünkü güveninirlik, okuma ve yazma bant genişliği durumlarını da düşünürsek bu bağlamda kopyaların nereye yazılacağı hassas bir konu olmaktadır. Örneğin tüm kopyaları aynı düğüme yazarsak az bant genişliği kullanımı olur fakat pratikte yedekleme yapmamış oluruz ve bu sebeple düğüm çalışmaz duruma gelince bloğa erişilemeyecek ve veriye ulaşılamaz olacaktır.

Başka bir durum ise kopyalanan veri farklı bir bölüte (raf) yazıldığında fazla bant genişiliği kullanılmış olacak ama bunun yanında güvenilir bir yedekleme yapmış olacağız. Bundan dolayı veri kopyalarının nereye yazılacağı önemli bir stratejidir. Şekil 3.4' de DDS' de veri yedekleme boru hattı verilmiştir.



Şekil 3.4 : Veri yedekleme boru hattı [21].

Hadoop sisteminde standart strateji olarak, ilk kopya aynı düğüme, ikinci kopya rasgele seçilen farklı bölüte (raf) ve üçüncü kopya ise ikinci kopya ile aynı bölüte (raf) ama rasgele seçilen farklı düğüme yazılmaktadır. Böylelikle düğümde ve bölütte sorun olmadığı ama veride sorun olduğu durumda yedek veriye hızlı erişim sağlanmış olacaktır. Düğümde veya bölütte sorun olduğunda farklı bölütte yer alan diğer yedek kopyaya yedekleme güvenilir bir şekilde yapıldığı için

ulaşılmış olacaktır. Kopyalamanın stratejik yaklaşımı aşağıdaki şekilde verilmiştir.

- DDS veri aktarım kapasitesi yüksek olacak şekilde tasarımı yapılmıştır.
- DDS şema tabanlı değildir ve hertürlü veri depolanabilir veya işlenebilir.

3.3.2 Map ve Reduce algoritmaları

Eskiden beri bilgisayar bilimi temel kavramları arasında yer alan bölme ve fethetme algoritmaları, büyük veri sorunlarının üstesinden gelen bir çözümdür. Temel yaklaşım büyük bir sorunu küçük ve bağımsız alt parçalara bölmektir. Böylelikle alt parçalara bölünen bu iş parçacıkları paralel bir şekilde çözülmek üzere birden fazla sisteme verilerek çözülebilir.

Sonuç olarak her bir iş parçacığının çözümü birleştirilerek genel sonuç elde edilir. Bölme ve fethetme algoritmaları genel bir çözüm yelpazesi sunduğundan dolayı çeşitli problemleri farklı uygulama altyapıları kullanılarak çözülebilir ama bazı durumlar için bu çözüm algoritması karmaşıklığa ve kullanım zorluğuna sebep olmaktadır. Örneğin aşağıdaki problemleri sıralamak mümkün olmaktadır [22].

- Büyük problem küçük parçalara nasıl bölünecek?
- Küçük problemler paralel olarak nasıl çözümlenecek?
- Parçalara ayrılan küçük problemler dağıtık bir şekilde nasıl paylaştırılacak?
- Farklı çözümleyiciler araında eşleme nasıl sağlanacak?
- Çözümleyiciler arasında veri alışverişi yönetimi nasıl olacak?
- Yazılım veya donanım sorunları ile karşılaşıldığında nasıl üstesinden gelinecek?

Geleneksel paralel veya dağıtık programlama ortamlarında, geliştirici yukarıda saydığımız tüm problemleri çözerek bir sistem oluşturması gerekiyor. Bu durumda tüm problemlerin üzerinden gelmek büyük problemleri çözmeği bir hayli zorlaştırmaktadır.

Map ve Reduce programlama modelinin en büyük avantajlarından biri bu durumları soyutlamasıdır. Map ve Reduce programlama modeli sistem düzeyindeki ayrıntıları geliştiriciden gizler. Büyük veri işlemek için esnek bir çözüm sunan Map ve Reduce programlama modeli basit olduğu kadar kullanılabilir bir programlama modelidir.

Hadoop sistemi çeşitli dillerde yazılmış Map ve Reduce programlarını çalıştırmaktadır.

Map ve Reduce programları Java, Ruby, Python, C#, C++ gibi dillerle yazılabilmektedir. Map ve Reduce programlama modeli büyük veri kümelerini analiz edebilmek için paralel süreçler oluşturmaktadır. 2 adımdan oluşmaktadır.

İlk adımı olan Map fonksiyonu, verinin okunup daha ufak parçalara ayrılmasını ve istenenin çıktı olarak dağıtık sistemlere dağıtılmasını sağlar. Diğer adım olan Reduce fonksiyonu ise, tamamlanan işler işin mantığına göre birleştirilerek sonuç elde eder. Bir başka anlatımla, Map aşamasında analiz edilen veri içerisinden almak istediğimiz veriler çekilir, Reduce aşamasında ise bu çektiğimiz veri üzerinde istediğimiz analiz gerçekleşir. Bu aşamalar bağımsız olarak gerçekleşebildiği için paralel olarak çalışabilir.

Map ve Reduce modellerinin kullandığı temel veri yapısı [anahtar, değer] şeklindedir. Buradaki anahtar parametresi int, double, string, byte olabileceği gibi karmaşık yapılar olan liste, dizi, üçlü grup yapıları da olabilir.

Örneğin hava sıcaklıklarını gönderen sistemlerin oluşturduğu veri yığınını ele alırsak, hava sıcaklığını gönderen sistemin tekil numarası anahtar parametresini oluştururken hava sıcaklığı da değer parametresini oluşturmaktadır. Ya da başka bir örnek verirsek web sayfalarının içeriklerinden oluşan bir veri yığınını düşünebiliriz. Bu durumda web sayfası adresi tekil olduğu için anahtar parametresini, web sayfası HTML içeriği de değer parametresini oluşturmaktadır.

Verdiğimiz bu örnekler çeşitli sistemlerde oluşan büyük verilere göre çoğaltılabilir. Aşağıda Map ve Reduce modellerinin kullandığı imza yapısı verilmiştir.

$map: (k_1, v_1) \rightarrow [(k_2, v_2)]$

$reduce: (k_2, [v_2]) \rightarrow [(k_3, v_3)]$

Map fonksiyonu anahtar ve değer çiftlerini girdi olarak alır ve her bir çift için tekrar orta anahtar ve değer çiftleri oluşturur. Reduce fonksiyonu da Map fonksiyonundan çıktı olarak gelen orta anahtar ve değer çiftleri ile ilişkili olan bütün değerler için anahtar ve değer çiftlerini çıktı olarak verir. Yani orta anahtar ve değer çiftleri için anahtarlara göre birleştirme operasyonu olduktan sonra Reduce fonksiyonu da istenen amacı bilgide uygular ve tekrar anahtar ve değer çıktısını sonuç olarak verir.

Hava sıcaklıklarını gönderen sistemlerin oluşturduğu veri yığınını örnek olarak vermiştik. Bu örneğin üzerinden veri akışının nasıl ilerleyeceğini aşağıdaki adımlarda olduğu gibi gösterebiliriz. Aşağıdaki gibi veri yığınımızın olduğunu düşünelim. İşlenecek veri herhangi bir formatta olması, düzenli bir yapısının olması veya herhangi bir platforma bağımlı olması gibi zorunlulukları yoktur.

```
0067011990999991110051507004...9999999N9+00025+99999999999...
0043011990999991110051512004...9999999N9+00037+99999999999...
0043011990999991120051518004...9999999N9-00040+99999999999...
0043012650999991120032412004...0500001N9+00048+99999999999...
0043012650999991120032418004...0500001N9+00043+99999999999...
0043012650999991130032418004...0500001N9+00010+99999999999...
```

Şekil 3.5 : Örnek veri yığını.

Map fonksiyonuna Şekil 3.6'daki gibi anahtar ve değer parametresi ile giriş yapacaktır ve fonksiyon, girdi olarak aldığı anahtar ve değer parametrelerine göre işlem yapacaktır. Buradaki anahtar parametresi sistem tarafından verilmektedir. Her bir satır için tekil bir anahtar belirlenmektedir.

```
(0, 0067011990999991110051507004...9999999N9+00025+99999999999...)
(1, 0043011990999991110051512004...9999999N9+00037+99999999999...)
(150, 0043011990999991120051518004...9999999N9-00040+99999999999...)
(227, 0043012650999991120032412004...0500001N9+00048+99999999999...)
(315, 0043012650999991120032418004...0500001N9+00043+99999999999...)
(1789, 0043012650999991130032418004...0500001N9+00010+99999999999...)
```

Şekil 3.6 : Map fonksiyonuna giren veri.

Map fonksiyonu, veri yığını içerisinden istediğimiz bilgiyi alıp verinin özelliğine göre tekrar anahtar ve değer parametreleri ile çıktı olarak Şekil 3.7'deki gibi çıktı üretmektedir. Buradaki anahtar değeri ise geliştirici tarafından belirlenmiş bir değer olmaktadır.

```
(5150, 25)
(5151, 37)
(5151, 40)
(3241, 48)
(3241, 43)
(3241, 10)
```

Şekil 3.7 : Map fonksiyonu çıktısı.

Reduce fonksiyonuna girmeden önce Map fonksiyonundan çıkan çıktılar, grup halinde anahtar ve değerler şeklinde organize edilip girdi olarak Reduce fonksiyonuna verilir. Şekil 3.8' de bu senayonun anlatımı verilmiştir. Bu işlemin amacı ise Map fonksiyonu ile Reduce fonksiyonu arasındaki veri transferini azaltmaktır. Aynı anahtara sahip değerler gruplanıp tek bir yapı içerisinde Reduce fonksiyonuna gönderilmektedir.

```
(5150, [25])
(5151, [37, 40])
(3241, [48, 43, 10])
```

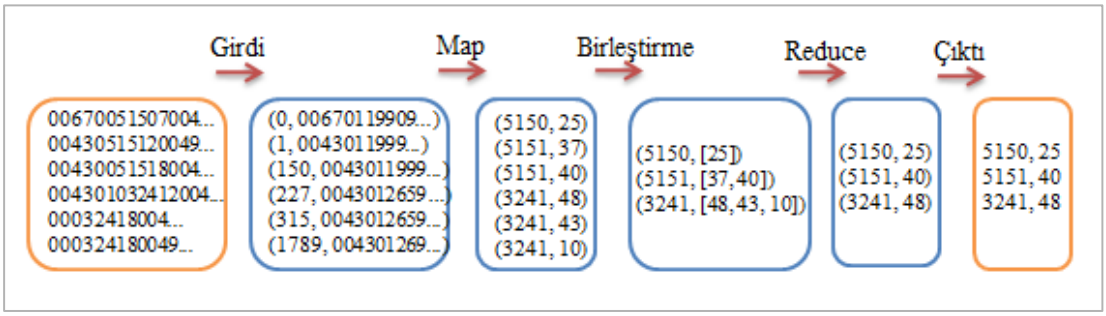
Şekil 3.8 : Reduce fonksiyonu girdisi.

Son olarak Reduce fonksiyonu, elimizdeki bilgilerden istediğimiz alanı çıktı olarak vermektedir. Örneğin verdiğimiz örnekte hava sıcaklıklarını bildiren sistemlerde en büyük hava sıcaklıklarını listeleyelim. Bu durumda reduce fonkisyonun çıktısı Şekil 3.9' daki gibi olmaktadır.

```
(5150, 25)
(5151, 40)
(3241, 48)
```

Şekil 3.9 : Reduce fonksiyonu çıktısı.

Verdiğimiz hava sıcaklıkları örneğinde gerçekleşen Map ve Reduce fonksiyonları arasında veri akışı aşağıdaki gibi olmaktadır.



Şekil 3.10 : Map ve Reduce fonksiyonları veri akışı.

3.4 Büyük Veri Analizi

Test sistemlerimizin üretmiş olduğu büyük veriyi analiz ederek, uçtan uca trafik analizi yapılması, kullanıcı deneyimini gerçeğe yakın bir şekilde ölçülmesi,

problemlerin müşteri şikâyeti gelmeden tespit edilip önlenmesi ve ağ kalitesinin ölçülüp bu alandaki gelişimlerin devreye alınması gibi bazı kazanımlar elde edilmeye çalışılmıştır.

Çeşitli alanlara konuşlandırılmış test sistemleri belirli aralıklarda hız testleri yapıp sonuçları ortak bir alana yazmaktadırlar. Test sistemlerinin ortak alana yazmış olduğu test sonuçları büyük veri yığınını oluşturmaktadır. Oluşan bu büyük veri, hızlı bir şekilde test alanına göre analiz edilip internet hız karakteristiğini bölgesel olarak çıkarılması gerekmektedir. Bu bağlamda Hadoop platformu kullanılarak Map ve Reduce programlama algoritmaları ile büyük veri analiz edilmektedir.

Büyük veri kümelerini analiz edebilmek için kullandığımız Map ve Reduce programlama modeli, bir önceki bölümde görmüş olduğumuz mimari model doğrultusunda çalışmaktadır. Test sonuçlarının analizi için kullandığımız Map ve Reduce programlama modelinin oluşturulmasına başlayabiliriz.

İlk olarak test sistemlerimizin oluşturmuş olduğu test sonuçlarının veri yapısını inceleyip hangi alanların neye karşılık geldiğini görelim. Ankara Söğütözü Bölge, Ankara Başkent Bölge ve Kayseri bölgelerinden aldığımız örnek veri aşağıdaki gibidir. Test sistemlerimiz veriyi aşağıdaki formatta yazmaktadır. Aşağıdaki örnek veri, gerçek test sistemlerinden alınan test verisidir.

```
Söğütözü 7,39 2,11 12,2 0,2
Söğütözü 8,20 1,15 17,1 0,4
Kayseri 9,39 2,21 18,9 0,6
Kayseri 7,99 1,14 16,7 0,5
Kayseri 5,32 2,91 10,3 0,4
Başkent 8,99 1,10 16,2 0,7
Söğütözü 9,36 2,11 12,2 0,9
Söğütözü 7,29 1,11 11,3 0,3
Başkent 7,39 2,11 16,7 0,3
Başkent 8,54 1,12 13,8 0,3
Başkent 8,34 2,19 16,6 0,5
Başkent 7,99 1,11 13,6 0,7
Kayseri 8,73 1,19 16,9 0,9
Kayseri 7,78 1,11 16,2 0,3
Söğütözü 7,39 2,11 12,3 0,1
Başkent 7,59 1,11 15,3 0,1
Kayseri 4,33 1,21 16,5 0,2
Kayseri 7,89 2,18 19,3 0,2
Söğütözü 7,37 2,11 12,2 0,5
Başkent 6,57 2,51 16,8 0,6
Söğütözü 7,99 2,11 12,7 0,4
Başkent 4,10 2,11 13,2 0,3
Kayseri 7,19 1,21 16,2 0,2
```

Yukarıdaki veri canlı ortamdan alınan test sonuçlarının bir kısmıdır. Her bir değerin arasında bir boşluk olacak şekilde yazılmıştır. İlk değer test biriminin bulunduğu alanın ismi, ikinci değer saniye başına indirilen veri miktarı, üçüncü değer saniye başına yüklenen veri miktarı, dördüncü değer istemci ile sunucu arasında gidip gelme süresi ve son olarak beşinci değer sunucu ile istemci arasında gidip gelme süresidir.

Test sistemleri ile alınan bu değerler, belirli aralıklarda kullanıcı senaryolarının uygulanıp ağ trafiği dinlenilerek elde edilen değerlerdir. Çeşitli test alanlarına konuşlandırılmış test sistemleri yukarıda verilen örnekteki gibi çıktılar üretmekte ve ortak bir alana yazmaktadırlar.

Test sonucundaki alanların hangi değere karşılık geldiği, aşağıdaki tabloda verilmiştir. Bu veri yapısı test sistemimizin sonuçlarını yazarken belirlediğimiz bir formattır. Bu veri düzeni istenen yapıda düzenlenebilir ama Map ve Reduce fonksiyonlarını yazarken elimizdeki veri yapısına göre ayrıştırmak önemli bir hususu oluşturmaktadır.

Veriden bilgi çıkartmak için çeşitli ayrıştırma yöntemlerini kullanarak yapılmaktadır. Kullanılan bu ayrıştırma yöntemleri verinin yapısına göre belirlenip Map ve Reduce algoritmalarının da bu yöntemlere göre yazılması gerekmektedir. Örneğin test sistemlerimizin ürettiği veriyi ayrıştırırken boşluklara göre ayırmak suretiyle gerekli parametreler elde edilmektedir.

Çizelge 3.3 : Test sonucu veri yapısı.

Değer	Anlam
1.Alan	Test sisteminin bulunduğu alan
2.Alan	Saniye başına indirilen veri miktarı
3.Alan	Saniye başına yüklenen veri miktarı
4.Alan	İstemci ile sunucu arasında gidip gelme zamanı
5.Alan	Sunucu ile istemci arasında gidip gelme zamanı

Analiz yöntemi olarak izleyeceğimiz yol, ilk olarak veriden bilgileri ayrıştırmak olmaktadır. Her bir test alanı için saniye başına indirilen veri miktarı, saniye başına yüklenen veri miktarı, istemci ile sunucu arasında gidip gelme süresi ve sunucu ile istemci arasında gidip gelme süresi değerleri ayrıştırılmaktadır. Sonrasında ise aynı test alanları için saniye başına indirilen veri miktarı, saniye başına yüklenen veri miktarı, istemci ile sunucu arasında gidip gelme süresi ve sunucu ile istemci arasında gidip süresi değerlerinin ortalamaları bulunmaktadır.

Değerlerin ortalamaları bulunduktan sonra son aşamada parametre değerlerine göre belirlenen eşik değerleri ile kıyaslanarak "iyi", "orta" ve "kötü" olarak etiketlenmektedir. Böylelikle test alanlarına göre belirlenen etiketler ortak bir alana yazılarak ve test bölgelerinin hız karakteristiği çıkartılmış olmaktadır.

Bu işlemler için Map ve Reduce fonksiyonları yazılmıştır. Veriden bilgileri ayrıştırma işlemi Map fonksiyonunda olurken, test alanlarına göre ayrıştırılmış değerlerin ortalamasının alınması Reduce fonksiyonunda olmaktadır ve sonrasında yine Reduce fonksiyonunda değişkenler eşik değerleri ile karşılaştırılarak "iyi", "orta" ve "kötü" şeklinde etiketleme işlemi yapılmaktadır. Bu işlemler için izlenecek çözüm yöntemi adımları aşağıda yazdığımız Map ve Reduce algoritmalarında gösterilmektedir.

Algoritma 2 Map(dokNo a, dok d)

1: **for** $satir : dokd$ **do**
2: $sistemNo \Longleftarrow$ alSistemNo(satir)
3: $indirme \Longleftarrow$ alIndirme(satir)
4: $yukleme \Longleftarrow$ alYukleme(satir)
5: $gecikme \Longleftarrow$ alGecikme(satir)
6: $Gonder(sistemNo_i, indirme)$
7: $Gonder(sistemNo_y, yukleme)$
8: $Gonder(sistemNo_g, gecikme)$
9: **end for**

Map metoduna tekil anahtar ile birlikte gelen metine boşluk karakterine göre ayrıştırma işlemi uygulandıktan sonra test alanı, indirme hızı, yükleme hızı, istemci ile sunucu arasındaki gidip gelme süresi ve sunucu ile istemci arasındaki gidip gelme süresi değerlerini elde etmiş olacağız. Bu değerleri, tekil test alanı anahtarı belirlenerek Reduce fonksiyonuna gönderilmek üzere çıktı olarak verilecektir.

Test alanı anahtarı, indirme hızı, yükleme hızı, istemci ile sunucu arasındaki gidip gelme süresi ve sunucu ile istemci arasındaki gidip gelme süresi değerleri için farklı üretilmektedir. Parametre değerleri ayrı ayrı parametreye özgü test alanı numarası ile birlikte yayımlanmaktadır. Farklı şekilde çalışma mantığı işletilerek değişik mantıkta algoritmalar yazılabilir. Bizim için paralelleştirme önemli bir konu olduğu için ve veri akışının paralel yürüyebilmesi için paralelleştirilebilen sistemler oluşturulması önemlidir. Bundan dolayı parametreleri bir diziye atmak yerine farklı farklı

anahtarlarla Reduce fonksiyonuna gönderebiliriz. Böylelikle farklı parametrelerin de analiz edilmesi paralelleşmiş olmaktadır.

Map metodundan gelen anahtar ve değer ikililerinin Reduce fonksiyonunda ortalamaları alınmaktadır. Buradaki anahtar parametreye özgü test alanını belirtmektedir. Bu işlem gerçekleştikten sonra hız kalitesinin belirlenebilmesi için etiketleme işlemleri yapılacaktır.

Algoritma 3 Reduce(sistemNo s, degerler[d1,d2...] d)

```
1: for deger : d do
2:     toplam ← toplam + deger
3:     sayac + +
4: end for
5: etiket ⟸ etiketle(toplam/sayac)
6: Gonder(s, etiket)
```

Test alanı ve ortalama değer ikilileri, indirme hızı, yükleme hızı, istemci ile sunucu arasındaki gidip gelme süresi ve sunucu ile istemci arasındaki gidip gelme süresi parametreleri için belirlenmiş eşik değerleri ile karşılaştırılıp "iyi", "orta" ve "kötü" şeklinde etiketleme işlemi Reduce fonksiyonunda yapılmaktadır. Böylelikle test alanına özgü internet hız karakteristiği belirlenmiş olmaktadır.

Yukarıda vermiş olduğumuz test sonuçları verisini ilk sekiz satırını kullanarak indirme hızı parametresi temel alınarak Map ve Reduce fonksiyonlarındaki gerçekleşen veri akışı aşağıdaki gibi olmaktadır. Şekil 3.11' de veri akışı için seçilmiş test sonucu verilmiştir.

```
Söğütözü  7,39 2,11 12,2 0,2
Söğütözü  8,20 1,15 17,1 0,4
Kayseri 9,39 2,21 18,9 0,6
Kayseri 7,99 1,14 16,7 0,5
Kayseri 5,32 2,91 10,3 0,4
Başkent 8,99 1,10 16,2 0,7
Söğütözü  9,36 2,11 12,2 0,9
Söğütözü  7,29 1,11 11,3 0,3
```

Şekil 3.11 : Test sistemi veri yığını.

Şekil 3.12' deki veri, satır satır tekil anahtarları ile birlikte Map fonksiyonuna parametre olarak girmektedir. Şekil 3.12' de görüldüğü gibi herbir satır için tekil

anahtar değeri verilmiştir. Anahtar değerini sistem, Map fonksiyonuna giren herbir satır için vermektedir. Buna göre herbir satır Map fonksiyonunda tekil anahtarlarına göre işleme girecektir. Map fonksiyonunda yayımlayacağımız anahtar ve değer ikilisi için anahtar değerini uygulama geliştiricisi verecektir. Yani Map fonksiyonuna giren anahtar ile Map fonksiyonundan çıkan anahtar değerleri birbirinden farklı olacaktır.

```
(0, Söğütözü  7,39 2,11 12,2 0,2)
(1, Söğütözü  8,20 1,15 17,1 0,4)
(2, Kayseri 9,39 2,21 18,9 0,6)
(3, Kayseri  7,99 1,14 16,7 0,5)
(4, Kayseri  5,32 2,91 10,3 0,4)
(5, Başkent  8,99 1,10 16,2 0,7)
(6, Söğütözü  9,36 2,11 12,2 0,9)
(7, Söğütözü  7,29 1,11 11,3 0,3)
```

Şekil 3.12 : Map fonksiyonuna giren test verisi.

Şekil 3.13' deki gibi Map fonksiyonundan çıkan herbir test alanı için ilgilendiğimiz değerler, Reduce fonksiyonuna ortalamaları alınmak ve etiketlenmek üzere gönderilmektedir. Şekil 3.13' deki örnek çıktı için sadece test sonuçlarından indirme süresi alınarak yapılmıştır. Yani boşluklara göre ayıklama yaptığımızda 3 sütun örnek olarak alınmıştır. Diğer parametreler de ayrı ayrı farklı anahtar değerleri ile Map fonksiyonundan Reduce fonksiyonuna girmek üzere çıktı olarak gönderilmektedir.

```
(Söğütözü,  7.39)
(Söğütözü,  8.20)
(Kayseri,  9.39)
(Kayseri,  7.99)
(Kayseri,  5.32)
(Başkent,  8.99)
(Söğütözü,  9.36)
(Söğütözü,  7.29)
```

Şekil 3.13 : Map fonksiyonundan çıkan test verisi.

Reduce fonksiyonuna girmeden önce veri, aşağıdaki gibi anahtar değerlerine göre guruplanır. Aynı anahtara sahip değerler, dizi veri yapısı içinde (anahtar, [değerler]) şeklinde gruplanmıştır. Gruplanan anahtarlar değerleri ile birlikte Şekil 3.14' deki gibi Reduce fonksiyonuna girmektedir. Bu gruplama işlemi Dağıtık Dosya Sistemi

tarafından organize edilerek, map ve reduce fonksiyonları arasında veri alışverişi az olması amacaıyla yapılmaktadır. Aynı anahtara sahip olan değerler bir dizi içerisinde gruplanarak Reduce fonksiyonuna tek seferde gönderilmektedir. Böylelikle Map fonksiyonundan Reduce fonksiyonuna gönderilen veri azalmış olmaktadır.

```
(Söğütözü,  [7.39, 8.20, 9.36, 7.29])
(Kayseri,  [9.39, 7.99, 5.32])
(Başkent,  [8.99])
```

Şekil 3.14 : Reduce fonksiyonuna giren test verisi.

Anahtar değerleri ile birlikte Reduce fonksiyonuna giren değerlerin ortalaması alınır ve anahtar değeri ile birlikte bulunan ortalama, sonuç olarak yazılır. Şekil 3.14' deki gibi Reduce fonksiyonuna giren veri Şekil 3.15' deki gibi ortalamaları alınmıştır.

```
(Söğütözü,  8.06)
(Kayseri,  7.56)
(Başkent,  8.99)
```

Şekil 3.15 : Reduce fonksiyonunda ortalamaların alınması.

Reduce fonksiyonunda parametrelerin ortalaması alındıktan sonra indirme hızı, yükleme hızı, istemci ile sunucu arasındaki gidip gelme süresi ve sunucu ile istemci arasındaki gidip gelme süresi parametreleri için belirlenmiş eşik değerleri ile karşılaştırılıp "iyi", "orta" ve "kötü" şeklinde etiketleme işlemi yapılmaktadır. Şekil 3.16' daki gibi Reduce fonksiyonundan sonuç olarak hız karakteristiği bilgisi elde edilmektedir.

```
(Söğütözü,  İyi)
(Kayseri,  Orta)
(Başkent,  İyi)
```

Şekil 3.16 : Reduce fonksiyonundan çıkan test verisi.

Yazmış olduğumuz Map ve Reduce algoritmalarını Java programlama dili kullanarak yazımına başlayabiliriz. Programı yazarken Cloudera platformu kullanılmıştır. Cloudera, yapısal veya yapısal olmayan verilerinin analiz edilerek anlamlı bilgiler ve ilişkiler çıkarılmasını sağlayan Apache Hadoop tabanlı açık kaynak kodlu yazılımın destek ve servis hizmetlerini sağlayan bir çözümdür. Cloudera platformu içinde yer

alan Eclipse bütünleşmiş geliştirme ortamında Java programlama dili kullanılarak yazılmıştır. Yazılan program Hadoop dağıtık dosya sisteminde gerekli komutlar girilerek çalıştırılmıştır.

Analiz programını yazmak için 3 tane sınıf yazılmıştır. Map fonksiyonunu içeren "SpeedMeterMapper", Reduce fonksiyonunu içeren "SpeedMeterReducer" ve Main fonksiyonunu içeren "SpeedMeterTest" sınıflarının yazımı ve anlatımı aşağıdaki adımlarda verilmiştir.

```
public class SpeedMeterTest {

@SuppressWarnings("deprecation")
public static void main(String[] args)
        throws Exception {

    Configuration conf = new Configuration();
    Job job = new Job(conf, "Speed Meter");

    job.setJarByClass(SpeedMeterTest.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(DoubleWritable.class);

    job.setMapperClass(SpeedMeterMapper.class);
    job.setReducerClass(SpeedMeterReducer.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    boolean success = job.waitForCompletion(true);
    System.exit(success ? 0 : 1);
  }
}
```

Kod 3.1 : SpeedMeterTest sınıfı.

Main fonksiyonu giren parametreler ise girdi verisinin bulunduğu dizin ve çıktıların yazılacağı dizin olarak belirlenmiştir. Main fonksiyonunda veri analizi için çalıştırılacak işin konfigurasyonu belirtilerek "SpeedMeterMapper" ve "SpeedMeterReducer" sınıflarının çalışmasını sağlamaktadır.

```
public class SpeedMeterMapper extends
    Mapper<LongWritable, Text, Text, DoubleWritable>
{
    private String location;

    @Override
    public void map(LongWritable key,
        Text value,
            Context context)
            throws IOException, InterruptedException
    {
        String line = value.toString();

        String[] tokens = line.split("\\s");

        location = tokens[0];
        String locDownload = location + "_download";
        String locUpload = location + "_upload";
        String locRTTab = location + "_rttab";
        String locRTTba = location + "_rttba";
        double download = Double.parseDouble(tokens[1]);
        double upload = Double.parseDouble(tokens[2]);
        double RTTab = Double.parseDouble(tokens[3]);
        double RTTba = Double.parseDouble(tokens[4]);

        context.write(new Text(locDownload),
                    new DoubleWritable(download));
        context.write(new Text(locUpload),
                    new DoubleWritable(upload));
        context.write(new Text(locRTTab),
                    new DoubleWritable(RTTab));
        context.write(new Text(locRTTba),
                    new DoubleWritable(RTTba));
    }
}
```

Kod 3.2 : SpeedMeterMapper sınıfı.

Map fonksiyonu "SpeedMeterMapper" sınıfında bulunmaktadır ve fonksiyon imzasında genel olarak 4 tane parametre vardır. Bu parametreler girdi anahtarı, girdi

değeri, çıktı anahtarı ve çıktı değerleridir. Girdi anahtarı her bir girdi değeri için sistem tarafından verilen tekil anahtardır. Girdi değeri ise okunan metnin bir satırıdır. Çıktı anahtarı ve değeri de okuduğumuz satırdan çıkarttığımız istenen bilgi ve ona bağlı anahtardır. "SpeedMeterMapper" sınıfında bulunan Map fonksiyonu, giren satırı boşluklara göre ayrıştırmaktadır. Buna göre birinci değer test alanı, ikinci değer saniye başına indirilen veri miktarı, üçüncü değer saniye başına yüklenen veri miktarı, dördüncü değer istemci ile sunucu arasında gidip gelme süresi ve beşinci değer sunucu ile istemci arasında gidip gelme süresi değerleri alınmaktadır.

```
public class SpeedMeterReducer extends
    Reducer<Text, DoubleWritable, Text, Text>
{
    private String param;
    private String quality;

    @Override
    public void reduce(Text key,
        Iterable<DoubleWritable> values,
            Context context)
            throws IOException, InterruptedException
    {
        double sum = 0.0;
        int count = 0;
        double average = 0.0;

        for (DoubleWritable val : values)
        {
            sum += val.get();
            count++;
        }
        average = sum / count;
        String line = key.toString();
        String[] tokens = line.split("_");
        param = tokens[1];
        quality = label(param, average);
        context.write(key, new Text(quality));
    }
}
```

Kod 3.3 : SpeedMeterReducer sınıfı.

Reduce fonksiyonu "SpeedMeterReducer" sınıfında yer almaktadır ve fonksiyon imzasında genel olarak 4 tane parametre vardır. Bu parametreler girdi anahtarı, girdi

değerleri, çıktı anahtarı ve çıktı değeridir. Girdi anahtarı her bir değer için verdiğimiz test alanını gösteren anahtar olurken, girdi değerleri de aynı parametreler için bulunmuş değerlerdir.

"SpeedMeterReducer" sınıfı Reducer sınıfından kalıt almaktadır ve Reduce fonksiyonu tekrar yazılmıştır. Reduce fonksiyonunda, test alanlarına göre gelen saniye başına indirilen veri miktarı, saniye başına yüklenen veri miktarı, istemci ile sunucu arasında gidip gelme süresi ve sunucu ile istemci arasında gidip gelme süresi değerlerinin ortalaması alınarak etiketleme işlemleri yapılmaktadır.

Ortalama değerler parametrelere göre belirlenmiş eşik değerleri ile karşılaştırılarak test alanlarına göre "iyi", "orta" ve "kötü" şeklinde etiketleme işlemi yapılmaktadır. "compareQualityDownAndUp" ve "compareQualityRTT" metotları ile eşik değer ve ortalama karşılaştırması yapılmaktadır. Reduce fonksiyonundan çıkan anahtar ve değer ikilisi test alanı numarası ve parametreye özgü hesaplanmış etiket bilgisidir.

Eşik değerlerini karşılaştırken aşağıdaki mantık doğrultusunda karşılaştırılmaktadır ve sonuç olarak internet hızı "iyi", "orta" ve "kötü" şeklinde etiketleme işlemi yapılmaktadır. Aşağıdaki eşitlikte n: test sayısını göstermektedir.

$$\frac{\sum_{i=1}^{n} x_i}{n} > \langle eşikDeğeri_{max} \rangle \xrightarrow{etiket} \text{İYİ}$$

$$\frac{\sum_{i=1}^{n} x_i}{n} < \langle eşikDeğeri_{min} \rangle \xrightarrow{etiket} KÖTÜ$$

$$\langle eşikDeğeri_{min} \rangle > \frac{\sum_{i=1}^{n} x_i}{n} < \langle eşikDeğeri_{max} \rangle \xrightarrow{etiket} ORTA$$

Eclipse bütünleşmiş geliştirme ortamını kullanarak oluşturduğumuz sistemin "jar" olarak çalıştırılabilir dosya formatını elde ettikten sonra programı Hadoop platformu üzerinde çalıştırma aşamasını gerçekleştirebiliriz. Hadoop dağıtık dosya sistemi üzerinde girdi verilerimizi koyacağımız bir klasör ve çıktıların yazılacağı bir klasör aşağıdaki komutlar ile oluşturulmaktadır. Girdi klasörüne girdi verisi taşındıktan sonra oluşturulan "jar" dosyası aşağıdaki komut ile birlikte çalıştırılabilir.

```
//Klasör oluşturma
hadoop fs -mkdir work

//Klasör oluşturma
hadoop fs -mkdir work/input

//Girdi verisinin ilgili klasöre kopyalanması
time hadoop fs -put data/*.txt work2/input

//Programın çalıştırılması
time hadoop jar SpeedMeter.jar SpeedMeterTest work2/input work2/output

//Çıktının terminalde görülmesi
hadoop fs -cat work2/output/*
```

Kod 3.4 : Sistemin çalıştırılması.

3.5 Analiz Sonuçlarının Gösterilmesi

Test sonuçlarının bölgesel analizinden sonra elde ettiğimiz hız karakteristiğini, gösterge paneli yardımıyla gösterilmektedir. Map ve Reduce algoritmaları yardımıyla elde ettiğimiz büyük veri analiz sonuçlarını okuyup, arayüzü Türkiye haritası olan gösterge panelinde gösterilmiştir.

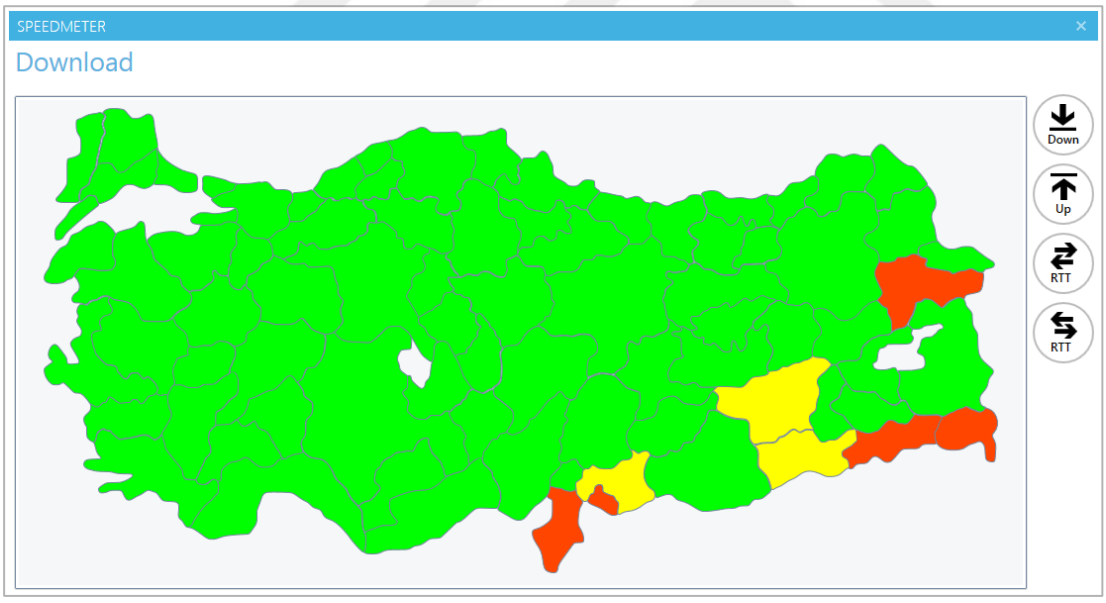
Operasyonel işler için problemin hızlı bir şekilde tespit edilmesi ve gerekli adımların hızlı bir şekilde uygulanması önemli bir süreçtir. Bu bağlamda problem tespitinin hızlı bir şekilde yapılabilmesi için anlık olarak sistemin durumunu gösteren bir uygulamanın olması bir hayli önemlidir. Gösterge paneli yardımıyla canlı sistemin durumunu görmek operasyonel işlerin hızlı bir şekilde yapılmasına çok büyük olanak sağlamaktadır.

Analiz sonuçlarının anlık olarak gösterilmesi operasyonel mükemmellik ve büyük resmi tam olarak görme açısından önemli bir aşamadır. Operasyonel işlerden sorumlu ekibin problem anını doğru ve hızlı bir şekilde yakalaması ve gerekli operasyonların verimli bir şekilde yapılması için gösterge panelinde analiz sonuçlarını göstermek sistemin kullanılabilirliğini arttıracaktır. Müşteriye hata anının minimum şekilde yansıması için operasyonel işlerin hızlı ve verimli şekilde yürümesi önemli bir husus olduğundan, mobil servis sağlayıcıları için sistem durumunu anlık gösteren bu gibi gösterge panelleri önemli bir araç olmaktadır. Bu bağlamda test

sistemlerinin veri toplaması ve analiz sistemlerimizin büyük veriyi bölgesel olarak analiz etmesi bölümlerinden sonra analiz sonuçlarının gösterilmesi ile birlikte sistemi bir bütün olarak tamamlanmaktadır. Çalışmamızın sonunda test alanlarına göre hız analizi sonuçlarını harita yardımıyla kullanıcılara gösteren ve anlık olarak hız karakteristiğini haritaya yansıtan bir uygulama geliştirilmiştir.

Kullandığımız teknolojiler ara yüz tasarlamak için kullandığımız WPF (Windows Presentation Foundation) ve programı yazdığımız dil olarak da .NET kütüphanesi ile birlikte C# programlama dilidir. .NET kütüphanesinin 4,5 olan sürümü kullanılmıştır. Programı yazarken Windows Visual Stüdyo 2012 aracı kullanılmıştır. Bunun yanı sıra uygulama başka dillerde ve diğer platformlarda da geliştirilebilir.

Saniye başına indirilen ve yüklenen veri miktarı, istemci ile sunucu arasında gidip gelme süresi ve sunucu ile istemci arasında gidip gelme süreleri örnek verileri Türkiye geneli olarak analiz edildiğinde çıkan sonucu geliştirmiş olduğumuz uygulamada aşağıdaki Şekil 3.17' deki gibi görmekteyiz.

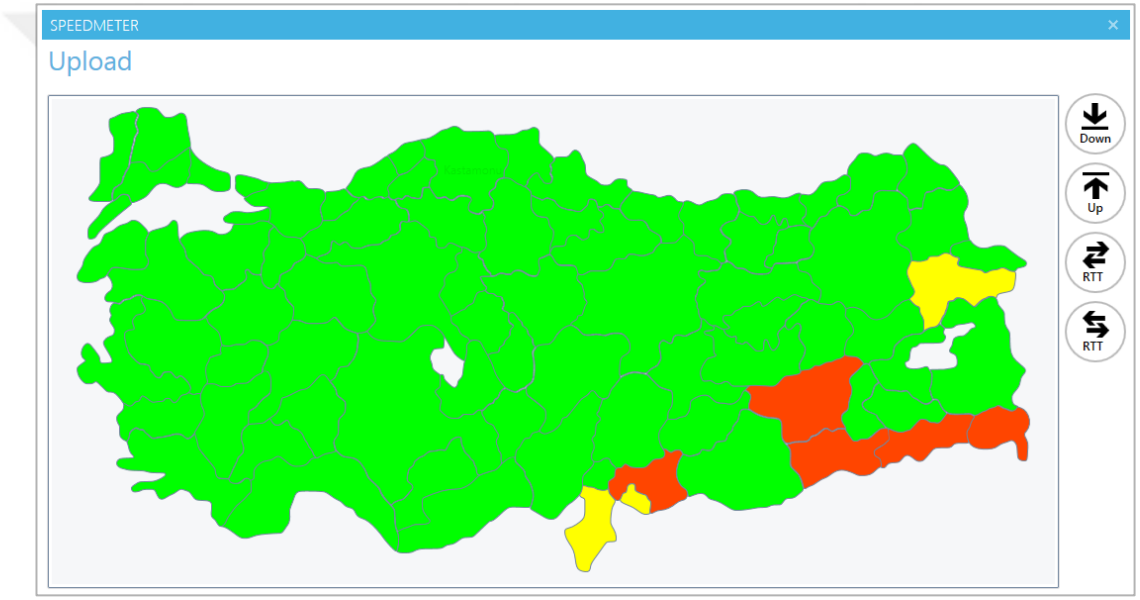


Şekil 3.17 : Saniye başına indirilen veri miktarı analiz sonucu.

Şekil 3.17' de görmüş olduğumuz analiz sonucu saniye başına indirilen veri miktarı göz önüne alınarak bulunmuştur. Hatay, Kilis ve Gaziantep illerinin ortalaması kırmızı ve sarı renkte olmasının sebebi o bölgedeki yoğun Suriye mültecilerinin oluşturmuş olduğu yoğunluk sebebiyle hat yük kapasitesinin kaldıramaması sorunundan kaynaklanmıştır. Diyarbakır, Mardin, Şırnak, Hakkâri ve Ağrı illerindeki

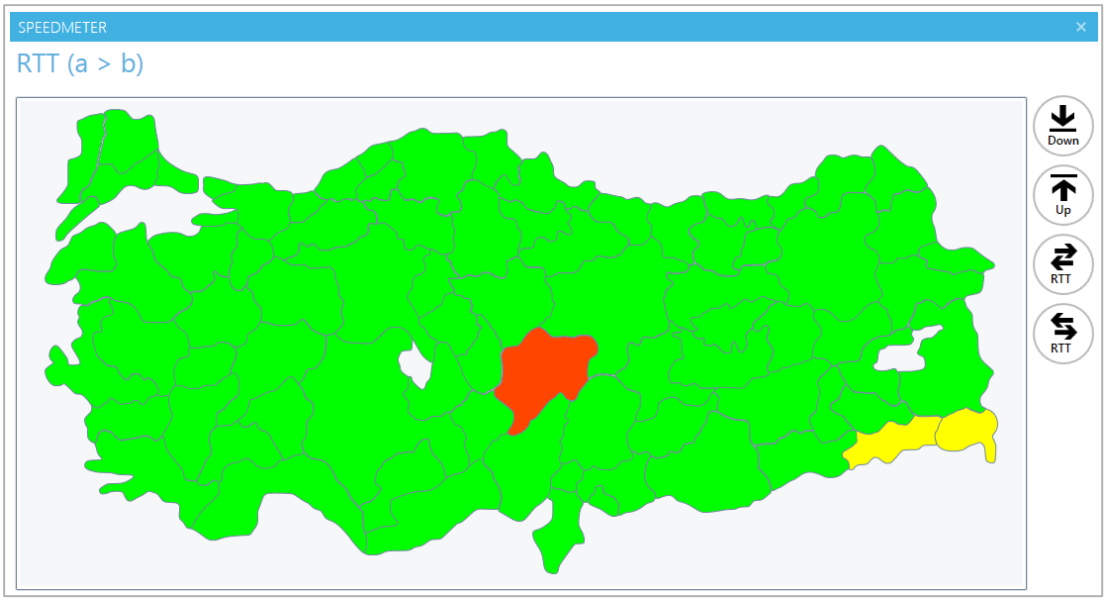
sorun ise 7 Haziran 2015 yılı seçimlerinden sonra artan terör olaylarının etkisinde kalan baz istasyonlarının servis verememesinden kaynaklı oluşmuştur.

Şekil 3.18' de başka bir tarihte alınmış verilerin analizi sonucu saniye başına yüklenen veri miktarı göz önüne alınarak bulunmuştur. Şekil 3.17' te olduğu gibi Hatay, Kilis ve Gaziantep illerinin ortalaması kırmızı ve sarı renkte olmasının sebebi o bölgedeki yoğun Suriye mültecilerinin oluşturmuş olduğu yoğunluk sebebiyle hat yük kapasitesinin kaldıramaması sorunundan kaynaklanmıştır. Yine aynı şekilde Diyarbakır, Mardin, Şırnak, Hakkâri ve Ağrı illerindeki sorun ise 7 Haziran 2015 yılı seçimlerinden sonra artan terör olaylarının etkisinde kalan baz istasyonlarının servis verememesinden kaynaklı oluşmuştur.



Şekil 3.18 : Saniye başına yüklenen veri miktarı analiz sonucu.

Şekil 3.19' da diğer analizlere göre farklı bir tarihte alınmıştır. Kayseri iline hizmet veren ve internete çıkış yapılması için kullanılan sunucuya (SGSN- Serving GPRS Support Node) dağıtık hizmet aksatma saldırısı yapıldığından sorun oluşmuştur. Kayseri genelinde mobil kullanıcılar kısa süreli internete çıkış yapamamışlardır. Test sistemlerinin yapmış oldukları test sonuçlarına göre yapılan analizler sonucu problem anı yukarıdaki gibi tespit edilmiştir.



Şekil 3.19 : İstemci ile sunucu arasında gidip gelme süresi.

Uygulama üzerinden de görüldüğü gibi çeşitli zaman dilimlerinde alınan örnek analiz sonuçlarına göre problemlerin tespiti, anlık olarak sistem üzerinden gösterilebilmektedir. Daha detaylı anlatım Sonuçlar bölümünde tartışılıp bu kapsamdaki kazanımlardan bahsedilecektir.

4. SONUÇ VE ÖNERİLER

Mobil kullanıcıların ihtiyaç alanlarının ses trafiğinden veri trafiğine doğru değişim göstermesi ve mobil şebeke altyapılarının paket anahtarlama teknolojisi yönünde değişmesi ile birlikte veri iletişimi trafiğinin izlenmesi ve analiz edilmesi, mobil servis sağlayıcıları için ağ yönetimi ve optimizasyonu bakımından önemli hale getirmiştir.

Veri iletişimi trafiğinin izlenmesi ve analiz edilmesi ve sonuç olarak kullanıcı deneyiminin bu sonuçlar doğrultusunda gerçeğe yakın bir şekilde ölçülmesi bir hayli zor olmuştur. Veri trafiği analizi yapmak amacıyla düğüm temelli yapılan incelemeler ve elde edilen anahtar gösterge parametreleri uçtan uca analiz edilmediğinden dolayı gerçek durumu yansıtmamaktadır.

Bu doğrultuda uçtan uca trafik analizi yapmakta olan test sistemlerinin, saniye başına indirilen ve yüklenen veri miktarı, istemci ile sunucu arasında gidip gelme süresi ve sunucu ile istemci arasında gidip gelme sürelerini belirli aralıklarda hesaplayarak ortak bir alana yazmaktadırlar. Yazılan test sonuçları da Map ve Reduce programlama algoritmaları ile analiz edilerek bölgesel olarak internet hız karakteristiği çıkarılmaktadır.

Son olarak bölgesel olarak analiz edilen internet hız karakteristiği bir uygulama ile gösterilmesi sağlanmıştır. Çalışma sonucunda elde ettiğimiz kazanımlar aşağıdaki gibi gruplanabilir.

- Uçtan uca trafik analizi,
- Kullanıcı deneyiminin ölçülmesi,
- Müşteri şikâyeti üremeden problem tespiti,
- Hızlı büyük veri analizi

4.1 Uçtan Uca Trafik Analizi

Mobil bir cihaz internetteki bir sunucuya herhangi bir istek gönderdiğinde radyo arayüzünden ve çekirdek ağındaki düğümlerden geçerek iletişim sağlanmaktadır. Bu durumda da radyo ara yüzünü ve çekirdek ağdaki düğümleri içerisine alan uçtan uca bir analiz önemli hale gelmektedir [23].

Var olan araçlara göre sistemimizin en önemli avantajı ise test sistemlerimizin gerçek kullanıma yakın senaryoları test edip, uçtan uca veri trafiğinin analizini yapmasıdır. Aşağıdaki şekilde analiz durumu gösterilmektedir.

Şekil 4.1 Uçtan uca servis kalitesi yapısı.

Şekilde de görüldüğü gibi istemci internetteki bir sunucuya istek göndermesi sonucu oluşan veri trafiğini uçtan uca analiz edilmesi görülmektedir. İstemci ile sunucu arasındaki trafik radyo arayüzü ve paket anahtarlama çekirdek ağındaki düğümleri de içerisine alarak analiz edilmektedir [24].

Bu çalışmamızda test sistemlerimiz, çeşitli alanlarda testlerini yaparak ağ trafiğini analiz edip çalışmamız sırasında belirlediğimiz parametreleri ortak bir alana yazmaktadır. Böylelikle veri iletişimindeki trafik uçtan uca analiz edilmiş olacaktır. Bu durumun bize getirmiş olduğu avantajı ise radyo ağı ve paket anahtarlama çekirdek ağı olmak üzere istemci ile sunucu arasındaki veri iletişimi uçtan uca ölçülmüş olacaktır [25].

Bunun sonucunda kullanıcıya gerçek anlamda yansıyan hizmet kalitesi ölçülmüş olacaktır ve şebekede var olan problemlerin analizi hızlı, kesin ve güvenilir bir şekilde yapılmaktadır. Günümüz teknolojisinde bu durum paket anahtarlama çekirdek ağındaki herhangi bir düğümden gözlemlenerek ölçülmektedir. Bu durumda da radyo ağını analize katmamış olduğumuzdan gerçek duruma yakın bir şekilde kullanıcının hissiyatını yakalayamamaktadır ve mobil ağda oluşmuş bazı problemleri algılayamamaktadır.

Uçtan uca trafik analizi olmadığı için de şebekede var olan problemlerin tespitini kesin ve güvenilir bir şekilde yapamamaktadır [26].

Uçtan uca trafik analizinin bir diğer avantajı ise mobil ağda yapılan bir değişiklik sonrası yapılacak olan saha testlerinin otomatik olarak insan gücünden bağımsız bir şekilde yapılması sağlanmaktadır. Bu şekilde daha güvenilir alan testleri, daha hızlı ve pratik yeni sistem kurulumları yapılmış olmaktadır [27].

Uçtan uca trafik analizinin getirmiş olduğu başka avantajı ise uçtan uca analiz sonucunda internet hız karakteristiğin ölçülmesi sağlanacak ve bu şekilde de ağ kapasitesi yetersizliğinden kaynaklı yavaşlamalar analiz edilerek ağ kapasite planlamaları gerçek duruma uygun bir şekilde planlamaların yapılması sağlanmaktadır. Özel günlerdeki ya da saatlerdeki kapasite ihtiyaçlarının analizi doğru bir şekilde yapılmaktadır [28].

Böylelikle tez kapsamında geliştirmiş olduğumuz sistemimiz sayesinde uçtan uca trafik analizinin yapılması, saha testlerinin otomatik, güvenilir, daha hızlı ve pratik şekilde yapılması, ağ kapasitesi yetersizliğinden kaynaklı yavaşlamalar analiz edilerek ağ kapasite planlamaları gerçek duruma uygun bir şekilde yapılmasını sağlamak gibi bazı kazanımlar elde edilmiş oldu.

4.2 Kullanıcı Deneyiminin Ölçülmesi

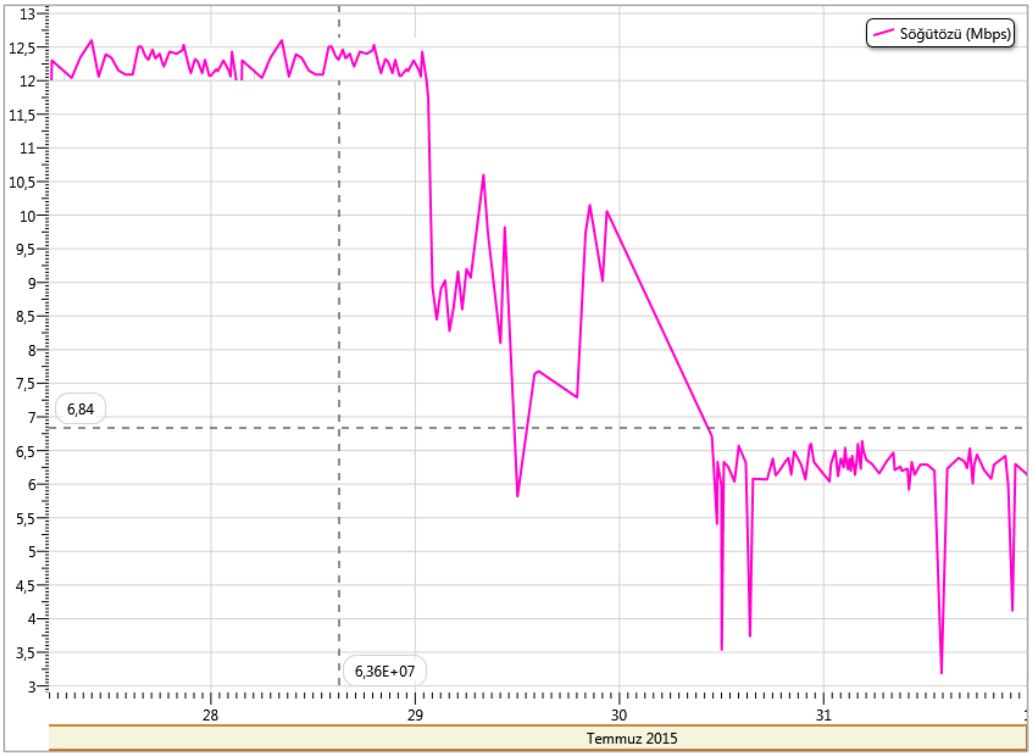
Mobil kullanıcıların ihtiyaçlarının değişmesi ile birlikte mobil veri trafiğinin artmasına neden olmuş ve bununla birlikte mobil servis sağlayıcıları için veri iletişimi trafiğinin izlenmesi ve analiz edilmesi önemli bir konu haline gelmiştir. Veri iletişimi trafiği incelenip analiz edilerek kullanıcı deneyimini ölçmek mobil iletişim sektöründe stratejik bir konu olmuş ve rekabet alanında aktif olarak kullanılabilir bir yöntem haline gelmiştir.

Günümüz dünyasında, paket anahtarlama çekirdek ağındaki herhangi bir düğümünde yapılan analizler ve bu analizler sonucunda belirlenen anahtar performans gösterge parametreleri gerçek durumu yansıtmamaktadır [29]. Böyle durumlarda da kullanıcı deneyimlerinin ölçülmesi ve kullanıcıların internete girerken nasıl bir hizmet kalitesi aldığını gösterilmesi bir hayli zor bir durumdur.

Gerçek anlamda kullanıcıya yansıyan hız kalitesini ölçebilmemiz için uç noktalardan paket trafiğini incelememiz ve giden gelen paketlerin analizini yapmamız gerekmektedir. Bu çalışmamız sonucunda test sistemlerimizin uçtan uca veri

iletişimini analiz etmesi ve sonuç olarak kullanıcıların hissiyatını ne derece iyi ya da kötü olduğunu belirleyebilmesi, çalışmamızın en büyük kazanımlarındandır.

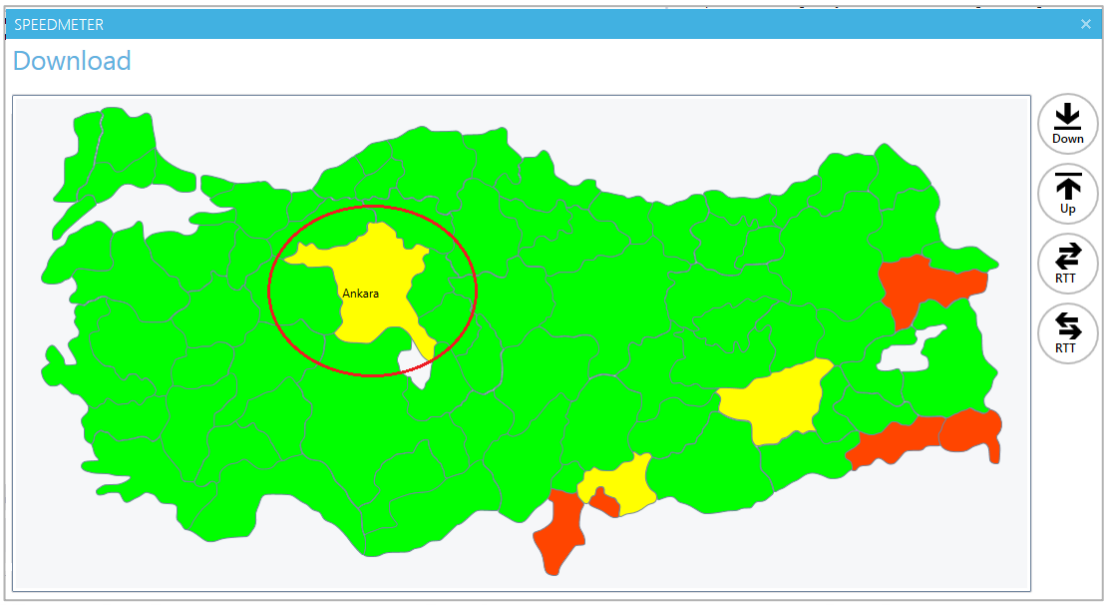
Aşağıdaki Şekil 4.2' deki test sistemlerimizin elde etmiş olduğu test sonuçlarının grafiği ve aynı zaman diliminde gelen müşteri deneyimi geribildirim ilgileşimi verilmiştir.



Şekil 4.2 Saniye başına indirilen veri miktarı ortalaması.

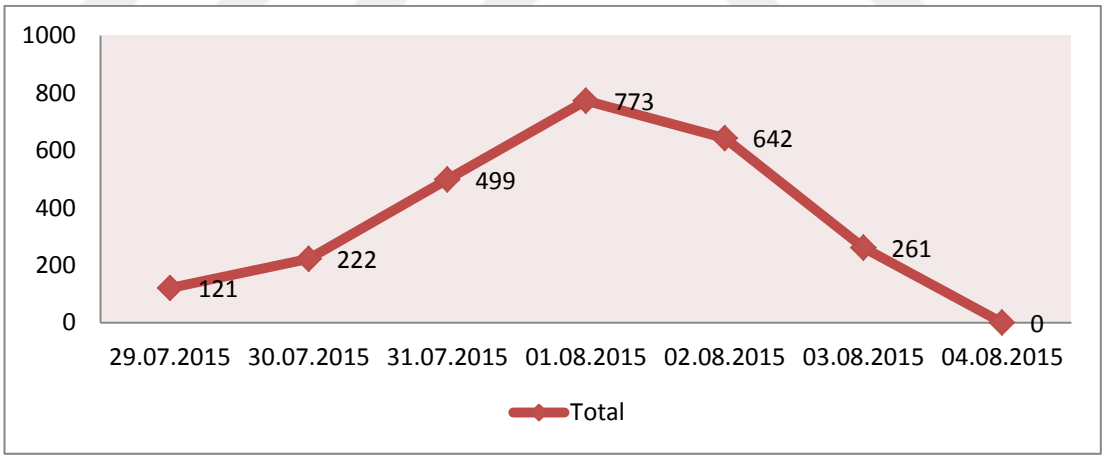
Şekil 4.2' de verilen test sistemimizin saniye başına indirilen veri miktarı ortalaması grafiği verilmiştir. Test yapılan zaman diliminde, baz istasyonlarında var olan bir konfigurasyonun yazılım güncellemesi yapılarak değiştirilmesi gerçekleştirilmiştir. Yalnız yapılan güncellemede bir hata olduğundan kullanıcıların internete girme hızlarında yarı yarıya düşüş yaşanmıştır.

Bu durumu, test sistemlerimiz yapmış olduğu analizler sonucunda yakalamış ve gerekli alarmları üretmiştir. Test sonuçlarının analizi yapılarak aşağıdaki Şekil 4.3' deki hız karakteristik haritası elde edilmiştir.



Şekil 4.3 Saniye başına indirilen veri miktarı analizi.

Aynı zaman dilimindeki kullanıcı hız memnuniyeti geribildirimlerini de analiz ettiğimizde kullanıcı memnuniyetinin belirtilen tarih aralıklarında olumsuz olduğu görülmektedir. Şekil 4.4' de geribildirimler sisteminden aldığımız rapor sonucu gösterilmektedir.



Şekil 4.4 Kullanıcıların hız memnuniyeti geribildirimi.

Telekomünikasyon sistemlerindeki paket anahtarlama mimarisi dönüşümü sonrası müşteri deneyimlerini ölçmek ve gerçek anlamda kullanıcı deneyimlerini yakalamak bir hayli zorlaşmaktadır [30].

Çalışmamızın en büyük avantajı test sistemlerimizin yapmış olduğu anlık testler ve bu testlerin oluşturmuş olduğu veri yığının analiz edilerek bölgesel olarak hız karakteristiğinin çıkarılması ile birlikte kullanıcı deneyimini gerçek anlamıyla

anlaşılmasıdır. Yukarıdaki analizlerden de görüleceği üzere şimdiye kadarki yapılan ölçümlerde kullanıcının hız hissiyatını anlamak bu şekilde kolaylaşmış olmaktadır.

4.3 Müşteri Şikâyeti Üremeden Problem Tespiti

Mobil servis sağlayıcıları için müşteri şikâyetleri, satış sonrası destek ve problemlerin yönetilmesi gibi hususlar önemli konuları oluşturmaktadır. Mobil iletişim sektöründe müşteri algısı o kadar önemlidir ki problem anında müşteriye minimum etki yansıması için stratejik olarak alınması gereken önlemler vardır.

Bu önlemler sektördeki hizmet sağlayıcıları tarafından alınmakta ve bu işlevlerini geliştirmek için bu alanda araştırma ve geliştirme faaliyetlerine önem vermektedirler. Çünkü müşteri algısı hizmet sağlayıcıları için çok önemli bir hizmet çıktısıdır.

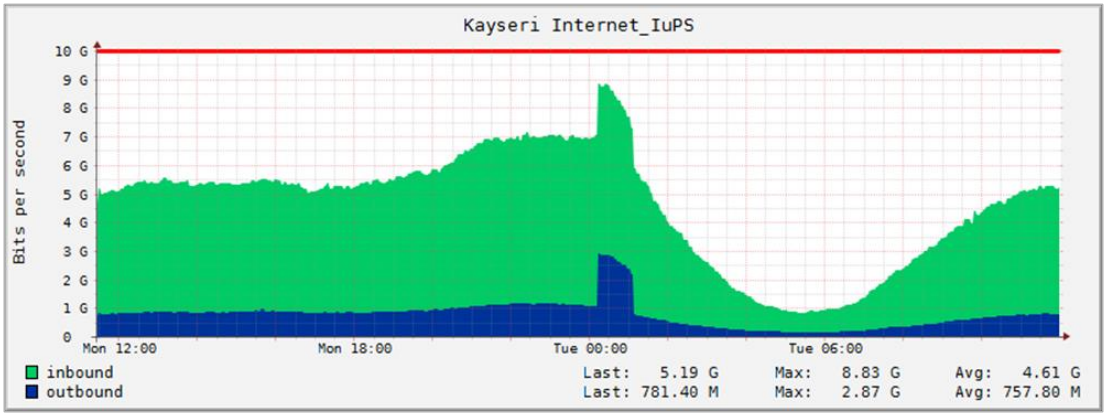
Verilen hizmetin kalitesini ölçmek ya da müşteri tarafındaki kalite hissiyatını algılamak servis sağlayıcıları için rekabet açısından önemli bir konuyu oluşturmaktadır. Bu bağlamda canlı ağ şebekelerinde problem tespitinin hızlı bir şekilde yapmak ve problem anının müşteriye minimum etki etmesini sağlamak servis sağlayıcılarının hedefledikleri konuların başında gelmektedir [31].

Tez kapsamında geliştirdiğimiz sistem ile birlikte test sistemlerimizin yapmış olduğu testlerin sonuçlarını bölgesel olarak analiz edip, saniye başına indirme ve yükleme veri miktarları ve paketlerin istemci ile sunucu arasında gidip gelme sürelerinin belirli bir eşik değerinin üzerinde ya da altında ise alarm üretmek suretiyle birlikte problem tespitini sağlamış olmaktayız. Böylelikle test sistemlerimizin yapmış olduğu test sonuçlarına göre herhangi bir problem anını yakalamış olup operasyonel olarak gerekli adımlar yerine getirilmiş olmaktadır.

Kayseri iline hizmet veren ve internete çıkış yapılması için kullanılan sunucuya (SGSN- Serving GPRS Support Node) dağıtık hizmet aksatma saldırısı yapıldığı sırada alınan test sonuçları analizi ile birlikte sunucunun servis verememesi durumu sistem tarafından algılanmış ve müşteriye etkisi tespit edilerek gerekli aksiyonlar alınmıştır.

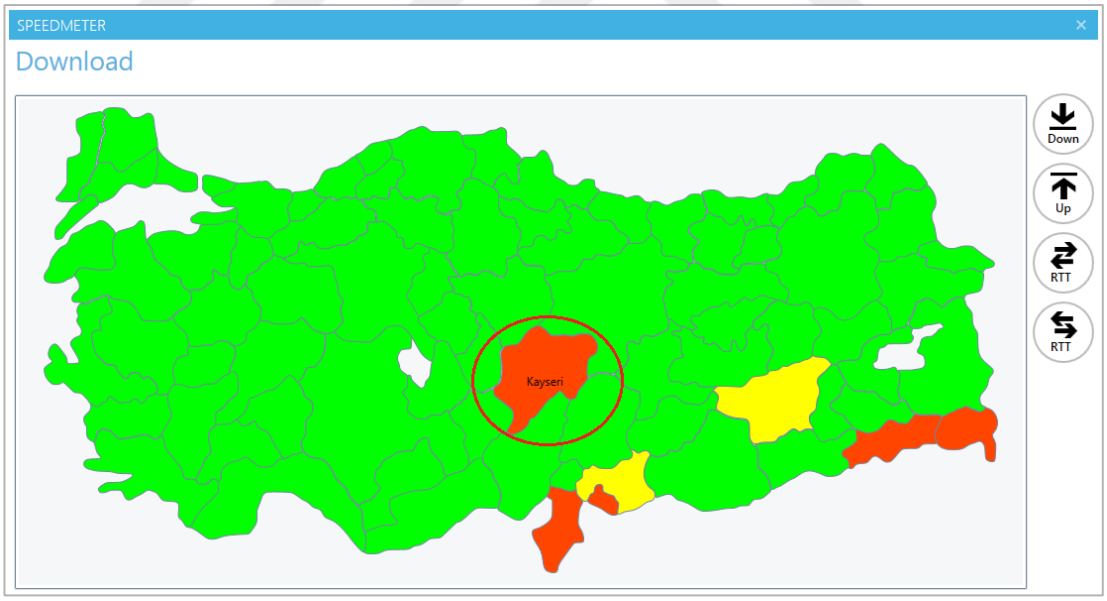
Problem anında saldırı yapılan sunucuyu kullanan Kayseri genelindeki müşteriler internete çıkış yaparken sorunlar yaşamıştır. Sorun anında kullanıcıların internet kullanımı aşağıdaki şekilde verildiği gibidir. Saldırı anında şekilde görüldüğü gibi

saat 00.15 gibi alışılmadık şekilde artmış ve sonrasında da servis kesintisinden dolayı hızlı bir şekilde internet kullanımı azalmıştır.



Şekil 4.5 İnternet kullanımı.

Problem durumunda alınan test sonuçlarının analizi sonrası sisteme yansıyan hız karakteristik haritası aşağıdaki gibi olmuştur. Böylelikle problem hızlı bir şekilde test sistemleri tarafından yakalanmış ve müşteri şikâyeti fazla üremeden sorun çözülmüştür.



Şekil 4.6 Kayseri için saniye başına indirilen veri miktarı analizi.

4.4 Hızlı Büyük Veri Analizi

Günümüz teknolojisinde bir hayli popüler olan "Büyük Veri", bilişim teknolojilerinin her alana girmesiyle birlikte birçok alana hitap etmeye başlamıştır. Kullandığımız bu veriler gün geçtikçe büyümektedir ve mevcut bilgi sistemlerinin

işleyemeyeceği kadar geniş ve karmaşık veri kümelerine dönüşmektedir. Bundan dolayı büyük veriyi toplama, saklama ve çözümleme yeteneklerini aşan büyüklükteki oluşan verileri yönetmek ve içerisinden anlamlı işe yarayan bilgi çıkartmak önemli bir konudur.

Çalışmamızda birçok alanda konuşlandırdığımız test sistemlerimiz düğüm temelli hız testini yaparak anlık olarak ağ trafiğinin inceleyip saniye başına indirme ve yükleme veri miktarları ve paketlerin istemci ile sunucu arasında gidip gelme sürelerinin hesaplanıp ortak bir alana yazmaktadır. Bu yazılan test sonuçları Map ve Reduce programlama yöntemleri algoritmaları ile analizinin yapılması ve sonuç olarak bölgesel veri hızı trafiğinin karakteristik olarak belirlenmesi sağlanmaktadır.

Test sonuçlarının oluşturmuş olduğu büyük veri Hadoop platformu kullanılarak hızlı bir şekilde analiz edilmesi ve bu analizler sonucu mobil veri iletişimi hız karakteristiğinin analizinin yapılması çalışmamız sonucunda ulaştığımız en önemli sonuçlardandır. Bu şekilde büyük veri analizini kullanarak internet hız analizi yapılmış bu analizler ışığında mobil ağ servislerinde daha iyi bir hizmet kalitesi verilmeye başlanmıştır.

KAYNAKLAR

[1] **Cisco,** (2014–2019), Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update.

[2] **Bart Salaets,** (2015), Why TCP optimisation has become more important than content optimization.

[3] **Broadpeak, Philip Hunter, Laurent Tescari,** Content Delivery Networks, A CTOiC White Paper

[4] **Tongqing Qiu, Junlan Feng, Zihui Ge, Jia Wang, Jun (Jim) Xu, Jennifer Yates,** (2010), Listen to Me if You can: Tracking User Experience of Mobile Network on Social Media, IMC'10, November 1–3, 2010, Melbourne, Australia.

[5] **Citrix**, https://www.citrix.com/.

[6] **Citrix**, Bytemobile adaptive traffic management, https://www.citrix.com/products/bytemobile-adaptive-traffic-management/overview.html.

[7] **Citrix**, Quality of experience for mobile data networks, https://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/quality-of-experience-for-mobile-data-networks.pdf.

[8] **Infovista**, End user quality experience QoE monitoring, http://www.infovista.com/solutions/end-user-quality-experience-QoE-monitoring.

[9] **Ookla,** (2016), Ookla, https://www.ookla.com/.

[10] **Ookla Speed Test,** (2016), Ookla SpeedTest, http://www.speedtest.net/.

[11] **Ookla Speed Test,** Speedtest intelligence, https://www.ookla.com/speedtest-intelligence.

[12] **Ookla Speed Test,** How to test your internet speed, http://www.speedtest.net/articles/how-to-test-your-internet-speed/.

[13] **Ookla Speed Test,** How to get faster download speed, http://www.speedtest.net/articles/how-to-get-faster-download-speed/.

[14] **Ookla Speed Test,** How to get faster upload speeds http://www.speedtest.net/articles/how-to-get-faster-upload-speeds/.

[15] **Caroline Chappell, Nokia**, (2014) Managing customer experience of the network: Strategies for success, Heavy Raeding, May 2014, White paper.

[16] **SAS**, (2016), Lessons from the Leading Edge of Customer Experience Management, http://www.sas.com/en_us/offers/sem/2299197-customer-experience-management/register.html?gclid=CjwKEAjwi9K4BRCQzq7d1c6A_XASJABueAO2gKW4uEFmsRtvDEZohl6k2i7MwrBFw485XjAnCwRQsxoCi9fw_wcB.

[17] **Huawei**, (2016), Customer Experience Management http://www1.huawei.com/en/solutions/broader-smarter/hw-197861-customerexperiencemanagement.htm.

[18] **Python,** (2016), SpeedTest kütüphanesinin kurulum komutları, https://pypi.python.org/pypi/speedtest-cli.

[19] **Github** , Github Kurulum Komutu, https://github.com/sivel/speedtest-cli.

[20] **Wireshark,** (2016), https://www.wireshark.org.

[21] **O'Reilly, Tom White**, (2012), Hadoop: the Definitive Guide.

[22] **University of Maryland, College Park, Jimmy Lin and Chris Dyer,** (2010), Data-Intensive Text Processing with MapReduce.

[23] **Utkarsh Goel, Mike P. Wittie, Kimberly C. Claffy, Andrew Le,** (11/2015) **,**Survey of End-to-End Mobile NetworkMeasurement Testbeds, Tools, and Services, IEEE Communications Surveys & Tutorials.

[24] **KwangSik Kim, S. Uno, S. Khadka, Moo Wan Kim,** (January 2010), End-to-End QoS manegement mechanism for mobile network, IEEE Communications Surveys & Tutorials.

[25] **Chandreyee Chowdhury, Sarmistha Neogy,** (January, 2012), Reliability of Mobile Agent System in QoS Mobile Network, Fourth International Conference on Communication Systems and Networks, Bangalore, India.

[26] **Sangheon Pack, Yanghee Choi,** (2001), An end to end QoS provisioning architecture in mobile network, Brain Korea 21 project of Ministry of Education and in part by the National ResearchLaboratory project of Ministry of Science and Technology, 2001, Korea.

[27] **Hye-Lim Koo, Byeong-hee Roh, Yu-Seon Kim, Young-Keun Park,** (2009), Data delivery with end-to-end QoS guarantee in tactical network interconnected with Wireless Ad-hoc Network and heterogeneous networks, Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on, Volume: 01.

[28] **H. Montes, G. Gómez and D. Fernández,** (2002), An end-to-end QOS framework for multimedia streaming services in 3G networks, Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on, Volume: 4.

[29] **Raj Jain,** Quality of Service In Data Quality of Service In Data Networks: Problems, Networks: Problems, Solutions, and Issues Solutions, and Issues, The Ohio State UniversityColumbus, OH 43210Columbus, OH 4321

[30] **Houda Alaoui Soulimani,** (2013), Dynamic management of the end-to-end QoS of a "user-centric" session, Pastel-00834199, 2013

[31] **Sonia Forconi, Alessandro Vizzarri,** (2013), Review of studies on end-to-end QoS in LTE networks, DOI: 10.1109/AEIT.2013.6666818 Conference: AEIT Annual Conference, 2013

ÖZGEÇMİŞ

Ad-Soyad : **Mete UZUN**

Uyruğu : **TC**

Doğum Tarihi ve Yeri : **24-11-1989, Samsun**

E-posta : **meteuzun@etu.edu.tr**

ÖĞRENİM DURUMU:

- **Lisans** : 2013, TOBB Ekonomi ve Teknoloji Üniversitesi, Mühendsilik Fakültesi, Bilgisayar Mühendisliği

MESLEKİ DENEYİM VE ÖDÜLLER:

Yıl	Yer	Görev
2011-2013	**TOBB ETÜ Hastanesi**	**Yazılım Mühendisi**
2013- 2014	**Atos**	**Yazılım Mühendisi**
2014- halen	**Turkcell**	**Network Yazılım Mühendisi**

YABANCI DİL: İngilizce, İspanyolca

TEZDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

- **Uzun M., Abul O**. 2016. End-to-end Internet Speed Analysis of Mobile Networks with MapReduce, The 3rd International Symposium on Networks, Computers and Communications (ISNCC), 11-13 May 2016, Hammamet, Tunisia