YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İNTERNET PROTOKOLÜ ÜZERİNDEN SES İLETİMİNDE HİZMET KALİTESİNİN ANALİZİ

Elektronik ve Haberleşme Mühendisi Erdem Halit HAKİ

FBE Elektronik ve Haberleşme Mühendisliği Anabilim Dalı Haberleşme Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Yrd. Doç. Dr. N. Özlem ÜNVERDİ

İSTANBUL, 2007

YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İNTERNET PROTOKOLÜ ÜZERİNDEN SES İLETİMİNDE HİZMET KALİTESİNİN ANALİZİ

Elektronik ve Haberleşme Mühendisi Erdem Halit HAKİ

FBE Elektronik ve Haberleşme Mühendisliği Anabilim Dalı Haberleşme Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Yrd. Doç. Dr. N. Özlem ÜNVERDİ

İSTANBUL, 2007

İÇİNDEKİLER

KISALTMA LİSTESİ

ADPCM	Adaptive Differential PCM
ASCII	American Standard Code for Information Interchange
CAC	Call Admission Control
CELP	Code Excited Linear Prediction
cRTP	Compressed RTP
DARPA	Defense Advanced Research Project Agency
DSP	Digital Signal Processor
DSCP	Differentiated Services Code Point
ENUM	T*E*lephone *NU*mber *M*apping
FIFO	First In First Out
FTP	File Transfer Protocol
FXS	Foreign Exchange Station
GSM	Global System for Mobile
HTTP	Hypertext Transfer Protocol
IAD	Integrated Access Device
ICMP	Internet Control Message Protocol
IEFT	International Engineering Task Force
IP	Internet Protocol
ITU	International Telecommunication Union
LFI	Link Fragmentation and Interleaving
PCM	Pulse Code Modulation
PSTN	Public Switched Telephone Network
RAS	Registration, Admission and Status
RED	Random Early Detection
RFC	Request For Comments
RSVP	Resource Reservation Protocol
RTCP	Real Time Control Protocol
RTP	Real Time Protocol
SMTP	Simple Mail transfer Protocol
ISDN	Integrated Services Digital Network
LAN	Local Area Network
MCU	Multi-Point Control Unit
MOS	Mean Opinion Score
OSI	Open Systems Interconnection
QoS	Quality of Service
SIP	Session Initiation Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VoIP	Voice Over IP
VoFR	Voice Over FR
VoATM	Voice Over ATM
WAN	Wide Area Network
UAC	User Agent Client
UAS	User Agent Server

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

ÖNSÖZ

Bu çalışmada bana her zaman yol gösterici olan ve iş hayatımdaki yoğunluk nedeniyle zaman zaman aksayan çalışmalarım için sabır gösteren hocam Sayın Yrd. Doç. Dr. N. Özlem ÜNVERDİ'ye teşekkürü bir borç bilirim. Ayrıca üyesi olduğum Tellcom İletişim Hizmetleri IP Çözümleri Takımı'na da tez çalışmalarım süresince gösterdikleri anlayış ve bana vermiş oldukları destekten dolayı teşekkür ederim.

Son olarak, her zaman yanımda olan aileme ve bu tezin bitmesinde verdiği manevi destek ile tartışmasız en büyük pay sahibi olan Deniz YASPEK'e şükranlarımı sunarım.

Erdem Halit HAKİ

Ekim 2007

ÖZET

IP altyapısı kullanılarak ses hizmetlerinin verilmesinin maliyeti, geleneksel PSTN sisteminin maliyetinden çok daha düşüktür. Bu durum, IP teknolojisini kullanarak ses hizmetleri satan yeni tip operatörlerin ortaya çıkmaya başlamasını sağlamış, bu da son kullanıcılar için daha geniş bir operatör seçme yelpazesi sunmuştur. Bununla birlikte kullanıcılar, PSTN sisteminin sağladığı ses kalitesine yakın kalitede ses hizmeti almaya devam etmek istemişlerdir. Günümüzde IP kullanılarak verilen ses hizmetleri düşük maliyetle yüksek hizmet kalitesi sunmaya odaklanmıştır.

Ses kalitesi, IP altyapısı kullanıldığında özellikle yoğun saatlerde değişkenlik gösterebilmektedir. Bu nedenle, hedeflenen kalitede ses iletimini sağlayabilmek için hali hazırda kullanılan veri ağında bazı değişikliklerin yapılması gerekebilir. Sabit kalite garantisi sunmak yıllardır araştırmacıların uğraştığı bir konu iken şimdi sektörün aşması gereken bir konu olarak önümüze çıkmaktadır. Çok kullanıcılı paket anahtarlamalı ağlar kullanılırken ses kalitesinin düzensizlik göstermesi temel bir problemdir. Kısa süreli beklenmedik yüklenmeler, ağ performansının garanti edilememesi, uç sistemlerin kontrolünün kaybedilmesi ve ses kalitesinin sağlanamaması VoIP'i başarıyla gerçekleştirilebilmesi zorlu bir uygulama yapmaktadır.

Bu tez, internet protokolü üzerinden ses iletişiminde hizmet kalitesinin geliştirilmesi için kullanılan yöntemlerin irdelenmesini amaçlamıştır. Tezin birinci bölümünde VoIP sistemi ve bu sistemin karşı karşıya olduğu problemler kısaca açıklanarak diğer VoX teknolojileri ile karşılaştırması yapılmıştır. İkinci bölüm, VoIP'in tarihsel gelişimini ve bu sistemin günümüzde hangi protokoller üzerinde çalıştığını detaylarıyla ele almıştır. Bu bölümde aynı zamanda en popüler iki sinyalleşme protokolü olan SIP ve H.323 de incelenmiştir. Tezin üçüncü bölümünde hizmet kalitesi kavramı açıklanarak, bant genişliği, gecikme, gecikme farklılığı (jitter) ve paket kaybı konularında kullanılabilecek hizmet kalitesi teknikleri sıralanmıştır. Ayrıca hizmet kalitesinin donanım veya yazılım ile sağlanmasının arasındaki farklara değinilmiştir. Son bölümde iki farklı uygulama bulunmaktadır. İlk uygulamada, çeşitli parametrelere göre hattın hizmet kalitesinin VoIP için uygun olup olmadığına karar veren basit bir Java uygulaması yapılmıştır. İkinci uygulamada, gerçek zamanlı ses, ftp, http ve icmp trafiği yaratmak için Cisco IAD 2431 8FXS gateway'lerin, dizüstü bilgisayarların ve analog telefonların kullanıldığı gerçek ortam koşullarına sahip bir test ortamı yaratılmıştır. Bu ortamda sıkıştırma, kuyruklama, çağrı izin kontrolü ve rasgele erken tespit teknikleri kullanılmış ve bu tekniklerin ses kalitesine etkileri incelenmiştir. Uygulamalarda TOS byte'ının DSCP bitlerinin dikkate alındığı ayrık sevisler (DiffServ) yaklaşımı kullanılmıştır.

Anahtar Kelimeler: IP üzerinden ses iletimi, VoIP, hizmet kalitesi, QoS

ABSTRACT

The cost of providing and running voice services using an IP infrastructure is considerably less than a traditional public exchange system. Therefore, new types of operators selling voice services are emerging that use IP technology, allowing a broader choice of operators for end users. Users however, would like to receive voice quality akin to that provided by the traditional telephony network. Voice services using IP in service today, focus on lower cost and high service quality.

The voice quality can be variable when using the IP infrastructure, especially in peak hours. Therefore, to achieve our goal of good quality audio communication, some changes might be needed to the ubiquitous Network. Providing strict quality guarantees has plagued researchers for many years and now industry is facing the same challenges. The degradation of voice quality which can occur when using a multi-user packet switched network is the fundamental problem. Unpredictable short term loads, lack of guarantees on network performance, lack of control over the end systems and stringent requirements on the voice quality make VoIP a challenging application to realize successfully.

This thesis proposes and investigates methods to improve quality of voice communication over the Internet Protocol. In the first part of thesis, VoIP system and it's problems are introduced briefly and compared with VoX technologies. In the second part, historical progress of VoIP and it's protocols are taken up. SIP and H.323, which are most popular signaling protocols in IP telephony, are also covered on this section. In the third part, Quality of Services concept is explained and various QoS techniques used for bandwidth, delay, jitter and packet loss are given. Performance differences between hardware based QoS and software based QoS are also discussed. In the last part of thesis, a simple Java application developed to determine line quality, whether good or bad, for VoIP. Besides, a test environment is designed. Cisco IAD 2431 8FXS gateways, laptop PCs and analog telephones are used to generate and monitor network traffic including voice, ftp, http and icmp in real time. In this environment, compression, queuing, call admission control and random early drop techniques are used. The implementation is based on a Differentiated Service approach using the DSCP bits of the TOS byte.

Keywords: Voice over IP, VoIP, Quality of Service ,QoS

1. GİRİŞ

İnternet protokolü üzerinden sesin taşınması son yıllarda çok popüler bir konu haline gelmiştir. Özellikle ekonomik açıdan çeşitli avantajlara sahip olan bu sistem, varolan internet altyapısını kullanması, katma değerli servisler verilebilmesi ve ek maliyet getirmemesi açısından şirketler için çok cazip bir ortam oluşturmaktadır.

VoIP, gerçek zamanlı paket iletimini gerektiren bir sistemdir. Bu sistemde örneksel ses sinyalleri sayısal hale dönüştürülüp IP paketleri haline getirilerek sayısal veri ağı üzerinden gönderilmektedir. IP ağları, her bir paket için yönlendirme protokollerinin de yardımıyla kaynaktan hedefe doğru en uygun yolun bulunmasını sağlamaktadır. Bununla birlikte, belirli bir noktadan yola çıkan ses paketleri farklı yollar ve yönlendiriciler üzerinden geçtiğinden hedef noktaya ya gecikmeli ulaşmakta ya da yol üzerindeki hat problemlerinden dolayı hiç ulaşamamaktadırlar. Kayıp ses paketlerinin alıcı tarafından tekrar istenmesi söz konusu olmadığından hedef noktaya farklı zamanlarda ulaşan ses paketlerinin sıraya sokulması, birleştirilmesi ve tekrar analog ses sinyaline dönüştürülmesi gerekmektedir.

VoIP sistemi kullanılırken iyi bir hizmet kalitesi sağlamak için veri ağı üzerinden düzensiz olarak gelen bu ses paketlerinin düzenli hale getirilmesi, gürültü ve gecikmelerden dolayı oluşan yankılardan arındırılması ve temiz ses sinyaline dönüştürülmesi garanti edilmelidir. Bu konudaki en büyük problem, VoIP cihazı üreten firmaların ürünleriyle VoIP uygulamaları arasında tam bir uyum sağlanamamasıdır. Her ne kadar VoIP teknolojisine ilişkin bazı standartlar oluşturulmuşsa da, bu standartlar tüm gereksinimleri kapsamadığından her üreticinin kendine özgü protokolleri ve farklı algoritmaları bulunmaktadır. Bu farklılıklar dinamik bant genişliği dağıtımında, paket kayıplarının etkisinin azaltılmasında, uyarlamalı yankı önleme ve ses kalitesini yükseltmek amacıyla kullanılan konuşma işleme algoritmalarında kendini göstermektedir.

Veri ağları üzerinden ses iletimi denince her ne kadar ilk akla gelen VoIP kavramı olsa da, biraz daha derine inildiğinde uygulamanın sadece IP üzerinden yapılmadığı görülür. Bu noktada karşımıza VoIP'in yanında VoFR ve VoATM gibi seçenekler çıkacaktır. Bir seçim yapmadan önce hangi tür teknolojinin daha uygun olduğuna karar vermek gerekir. OSI katmanlarından üçüncü seviyede çalışan VoIP ya da ikinci seviyede çalışan VoFR veya VoATM'in kullanılmasına karar verirken ses iletiminin yanında diğer hangi hizmetlerin kullanılacağını da belirlemek gerekir. VoFR ve VoATM'in geniş alan ağlarında bant genişliğini VoIP'e göre daha etkin kullanması nedeniyle birçok kullanıcı grubu tarafından

tercih edilebilmektedir. Fakat ikinci seviyede çalışan bu servisler yerel alan ağlarına, ya da masaüstü uygulamalarına doğrudan dahil edilemediklerinden VoIP günümüzde kullanılan VoX teknolojileri içinde en fazla tercih edilenidir. VoIP, aynı zamanda kurulu bulunan internet ve intranet alt yapısını yönlendirme ve noktadan noktaya arama özellikleri açısından olduğu gibi kullanabildiğinden ses iletiminde tercih edilecek tek adaydır.

2. İNTERNET PROTOKOLÜ ÜZERİNDEN SES İLETİMİ

Bu bölümde, internet protokolü üzerinden ses iletimi konusunun rahatlıkla anlaşılabilmesi ve uygulama bölümündeki kavramlara temel oluşturması için gerekli olan genel bilgiler verilmiştir.

2.1 IP Üzerinden Ses İletimi (VoIP)

İnternet protokolü üzerinden ses iletimi olarak çevirebileceğimiz VoIP, analog ses sinyallerinin sayısal hale dönüştürülüp IP paketleri haline getirilerek sayısal veri ağları üzerinden iletilmesidir. Bu sebeple VoIP, internet ve intranet gibi IP kullanan herhangi bir veri ağı üzerinde başarıyla gerçekleştirilebilir. Her nekadar teknolojinin ismi "ses" ile bağdaşsa da, genelde "ses ve çoklu-ortam" olarak genişletilmekte ve sesi olduğu kadar faks, konferans gibi çoklu ortam uygulamalarını da gerçek zamanlı olarak karşılamaktadır.

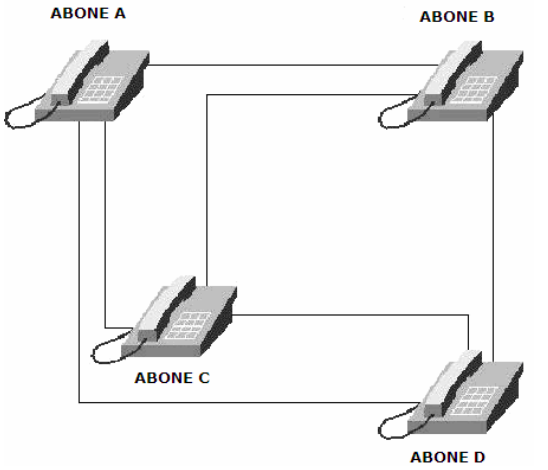
2.2 VoIP'in Tarihçesi

"Watson buraya gelebilir misin? Yardımına ihtiyacım var." 10 Mart 1876'da, ses iletiminin başladığını belirten bu tarihi sözler telefonun mucidi Alexander Graham Bell tarafından asistanı Thomas Watson'a söylenmiştir. Karşıdaki kullanıcıya ulaşmak için numara çevrilmesini gerektirmeyen bu uygulama doğrudan konuşmaya başlanmasını sağlamıştır. Bell, daha sonra bu uygulamayı geliştirerek patentini almıştır. Zamanla, bu basit tasarım aynı anda sadece bir kişinin konuşabildiği tek yönlü ses iletimi sağlayan bir düzenek olmaktan çıkartılmış, her iki tarafın da aynı anda konuşabildiği çift yönlü ses iletimi sağlayacak şekilde geliştirilmiştir.

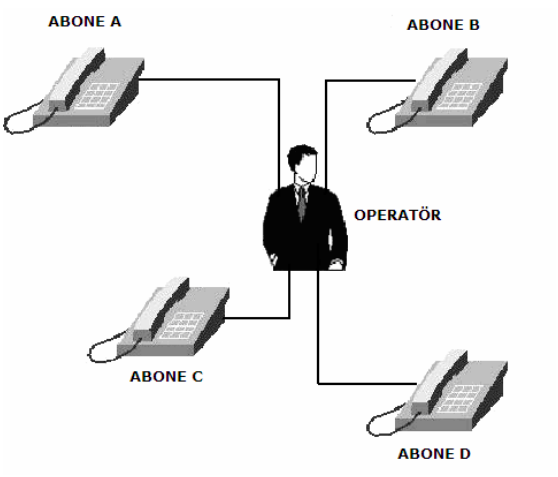
Bu gelişmeler telefon'a olan ilgiyi arttırmış, bunu abone sayısının hızlı bir şekilde yükselmesi izlemiştir. Ancak bu yükseliş Şekil 2.1'de görüldüğü gibi kullanıcının aramak istediği her yere fiziksel kablo çekilmesi gerekliliğini gündeme getirdiğinden, telefon sisteminin artan abone sayısı karşısında içinden çıkılmaz bir hal almaya başlamasına neden olmuştur. Maliyetin fazla olması ve dünyada telefon kullanmak isteyenler arasında fiziksel kablo döşemenin mümkün olmayacağı göz önüne alınarak, bir telefonu başka bir telefona eşleştirebilen yeni bir mekanizma geliştirilmiştir. Telefon kullanıcısının "santral" adı verilen bu aygıta bağlanması için tek bir kablo yeterli olmuştur.

İlk telefon santralleri manuel olarak çalışmıştır. Yani, bağlantılar elle yapılmıştır. Telefonla birisine ulaşmak isteyen insanlar önce bir operatörü aramış, görüşmek istedikleri kişinin

telefon numarasını söylemiştir. Sonrasında Şekil 2.2'de görülgüğü gibi, operatör arayan ile aranan kişilerin hatlarını elle birbirine bağlamıştır.



Şekil 2.1 Tüm abonelerin birbirine bağlı olduğu telefon sistemi



Şekil 2.2 Tüm abonelerin operatöre bağlı olduğu telefon sistemi

Bununla birlikte, farklı santrallerde bulunan aboneler birbirlerini aramak istediklerinde konuşmaları birçok operatör tarafından zincirleme olarak birbirine bağlanmış, bu da bağlanma süresini uzatmıştır. Bu problemlerin etkisiyle, 1891 yılında Strowger ve Keith tarafından operatöre gerek duymayan otomatik santraller geliştirilmiş ve abonelerin birbirine direk bağlanması sağlanmıştır. Bunu, 1948 yılından sonra transistorun keşfiyle birlikte sayısal santrallerin geliştirilmesi izlemiştir (*). Transistorun keşfi ve tüm devre teknolojisinin geliştirilmesi aynı zamanda bilgisayar çağının da kapılarını aralamıştır.

Internet'in ortaya çıkışı, Amerikan Federal Hükümeti Savunma Bakanlığı'nın araştırma ve geliştirme kolu olan "Savunma İleri Düzey Araştırma Projeleri Kurumu"na (DARPA) dayanmaktadır. 1969'da çeşitli bilgisayar bilimleri ve askeri araştırma projelerini desteklemek için Savunma Bakanlığı ARPANET adında paket anahtarlamalı ağ'ı oluşturmaya başlamış, bu ağ ABD'deki üniversite ve araştırma kuruluşlarının değişik tipteki bilgisayarlarını da içine alarak büyümüştür. 1973 yılında, ağ için bir protokol seti geliştirmek amacıyla Stanford Üniversitesi'nde bir "internetworking" projesi başlatılmıştır.

*Türkiye'de ilk manuel telefon santrali 3 Mayıs 1909'da İstanbul Büyük Postane binasına 50 hatlık olarak tesis edilmiştir. Sonrasında ilk otomatik telefon santrali 11 Eylül 1926'da, ilk sayısal telefon santrali ise 18 Aralık 1984'te Ankara'da hizmete girmiştir.

1978'e kadar iletim kontrol protokolü'nün (TCP) dört uyarlaması geliştirilmiş ve denenmiştir. 1980'de bu küme sabitleşmiş ve ARPANET'e bağlı bilgisayarlar arasındaki iletişimi kolaylaştırmıştır. 1983'te tüm ARPANET kullanıcıları İletim Kontrol Protokolü/İnternet Protokolü (TCP/IP) olarak bilinen yeni protokole geçiş yapmış ve o yıl TCP/IP, ARPANET'i de içeren Savunma Bakanlığı internetinde kullanılmak üzere standartlaştırılmıştır. ARPANET 1990 Haziran'ında kullanımdan kaldırılmasına rağmen ABD, Avrupa, Japonya ve Pasifik ülkelerinde ticari ve hükümet işletimindeki omurgalar onun yerini almıştır. ARPANET'in kaldırılmasına rağmen, TCP/IP protokolü kullanılmaya devam etmiş ve geliştirilmiştir (*).

1995'lerde modemlerin 14,4 kbps hızına erişmesi, aynı zamanda GSM için geliştirilen 8 kbps'lik düşük hızlı codec'lerin kullanılabilir hale gelmesiyle IP ağları üzerinden ses iletimi teknik olarak mümkün hale gelmiştir. Ses iletiminin halen kullanılan sabit telefon standardı yerine internet üzerinden yapılabileceği fikri, ilk defa Şubat 1995 tarihinde VocalTec adlı bir şirketin geliştirdiği yazılım sayesinde gerçeğe dönüşmüştür. Söz konusu yazılım, 486/33 MHz veya daha yüksek hıza sahip, üzerinde ses kartı, hoparlör ve mikrofon ile modemi bulunan bilgisayara yüklenerek, ses sinyallerinin sıkıştırılması ve IP paketlerine dönüştürülerek internet üzerinden gönderilmesi ile sağlanmıştır. Bu uygulama yaygın olarak kullanılmamakla birlikte standartlaştırma çalışmaları da aynı yıl başlamıştır. ITU 1996 yılında G.723 ve G.729 kodlama standartlarını oluşturmuştur. VOIP uygulamasında kullanılan gerçek zaman protokolü'de (RTP) aynı yıl IETF tarafından standartlaştırılmıştır. 1997'de IETF tarafından standartlaştırılan kaynak ayırma protokolü (RSVP), ilerleyen yıllarda IP üzerinden ses iletiminde ihtiyaç duyulan bant genişliğini garanti etmek için kullanılmış ancak yaygın olarak kabul görmemiştir. Paket anahtarlamalı ağlar üzerinden ses iletiminde sinyalleşme için ITU tarafından 1999 yılında H.323 standardı önerilmiştir. 2000'li yıllarda oturum başlatma protokolü (SIP) H.323'e alternatif olarak IETF tarafından duyurulmuştur. Bu standartlar ses iletiminde ihtiyaç duyulan arama, kapatma, yol kurulumu, raporlama gibi ihtiyaçları yerine getirmektedir. 2002 yılında SIP versiyon 2 (SIPv2) ile yeniden düzenlenmiştir, önümüzdeki günlerde SIP versiyon 3 (SIPv3)'ün duyurulması beklenmektedir.

Son yıllardaki çalışmalar daha çok paketlerin gecikmesi, jitter, paket kaybının azaltılması ve servis kalitesinin arttırılması konularına odaklanmaktadır.

---

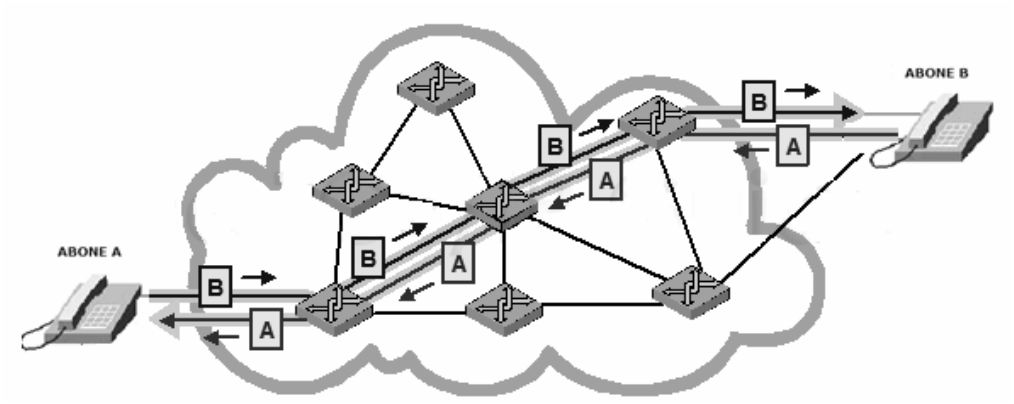
*Türkiye internete 12 Nisan 1993'te bağlanmıştır. 2007 yılı itibarıyla 17 milyon kullanıcıya ulaşılmıştır.

2.3 Anahtarlamalı Ağlar

Haberleşmede amaç verinin bir noktadan haberleşmek istenen diğer noktaya aktarılmasıdır. Ne yazık ki, bu iki nokta arasında doğrudan bir hat kurulması pratik olarak çok masraflı ya da imkânsızdır. Bu nedenle veri, anahtar adı verilen bağlantı noktalarının yardımıyla bir hattan diğerine aktarılarak uçtan uca iletilir. Anahtarlar, ilk telefon hatlarından günümüze kadar haberleşme sistemlerinin vazgeçilmez elemanları olmuşlardır. Genel olarak ses haberleşmesinde iki çeşit anahtarlamalı ağ kullanılmaktadır. Bunlardan ilki, devre anahtarlamalı bir sistem olan PSTN, diğeri ise paket anahtarlamalı bir sistem olan IP ağıdır. Kullanılan veri ve protokollerin farklı olması dolayısı ile PSTN ve IP ağları, farklı ağlar arasında ses iletimini gerçekleştirebilen gateway'ler yardımıyla birbirleriyle haberleşebilmektedir.

2.3.1 Devre Anahtarlamalı Ağ

Daha önce de bahsettiğimiz gibi devre anahtarlamalı ağlarda iletişim için tek bir bağlantı kurulur. Yani iki nokta çift yönlü olarak bağlanır ve bu bağlantı "devre" olarak adlandırılır. Bağlantı boyunca kurulan devrenin değiştirilmesi mümkün değildir. Her bir aramada, önce sinyalleşme protokolü kullanılarak 64 kbps'lik sabit bir devre kurulur. Devre bir kere kurulduktan sonra iletişim başlar. İletişim tamamlandıktan sonra devre ve kullanılan tüm kaynaklar serbest bırakılır.



Şekil 2.3 Devre anahtarlamalı ağ

Devre anahtarlamalı bir ağ üzerinden 10 dakika konuşulduğunu düşünelim. Bu süre boyunca iki telefon arasında kurulmuş olan devre sürekli açık kalır. PSTN üzerinden gerçekleştirilen

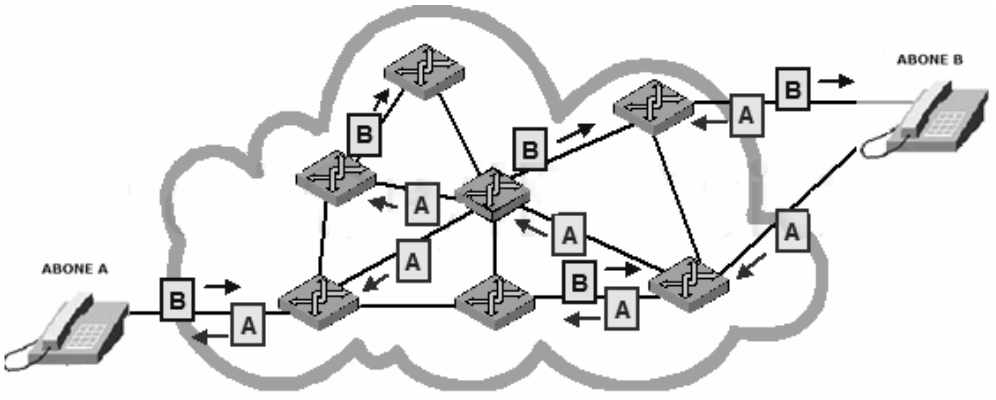
geleneksel telefon görüşmeleri, her iki yönde 64 kbps (toplamda 128 kbps) gibi sabit bir veri iletimi gerektirir. 1 kilobyte veri 8 kilobit'e eşit olduğundan, her bir saniyede toplam 16 kilobyte verinin iletildiği görülür ki, bu da her bir dakikada 960 kilobyte'a karşı gelir. Bu durumda 10 dakika boyunca yaklaşık 9,4 megabyte veri iletilmiş olur.

Böyle bir sistemde iletim için ayrılan bant genişliğinin büyük bir kısmı israf olmaktadır. Çünkü siz konuşurken karşı taraf sizi dinlediği için konuşma anında bağlantının yarısı kullanımdadır, yani bağlantının 4.7 megabyte'lık kısmı boşa harcanır. Bunun yanı sıra konuşmanın önemli bir kısmında da ne siz konuşursunuz ne de karşı taraf konuşur. Bu da bant genişliğinin boşa harcandığı başka bir durumdur. Devre anahtarlamalı ağlardaki en büyük problem, kurulan her çağrıda karşılıklı olarak 64 kbps'lik bant genişliği sabit olarak ayrıldığından kullanıcının daha az ya da daha fazla bant genişliği talep edememesidir. Sessiz periyotlarda bile kaynaklar tamamıyla kullanılır durumda kalmaktadır. Yani, devre anahtarlama yöntemi kullanılmayan bu kapasiteyi esnek bir trafikle doldurma yetisine sahip değildir.

Öte yandan devre anahtarlama yönteminin en önemli avantajı, tüm konuşma süresince ayrılmış olan bant genişliğine, dolayısıyla yollayabileceğimiz bilgi miktarına bağlı olarak aramanın kalitesinin önceden bilinebilmesidir.

2.3.2 Paket Anahtarlamalı Ağ

Paket anahtarlamalı ağlarda, devre anahtarlamalı ağlarda olduğu gibi sürekli bir bağlantının kurulması gerekmezken hiçbir kaynak tahsisi de söz konusu değildir.
Bu ağlarda iletim paketler ile yapılır. Bir anahtarlama noktasına gelen paketin içerisinde nereden geldiği ve nereye gideceği bilgisi bulunur. İki anahtarlama noktası arasındaki yollardan hangisi o an için en uygunsa paket hedefe o yolu tercih ederek ulaşır.



Şekil 2.4 Paket anahtarlamalı ağ

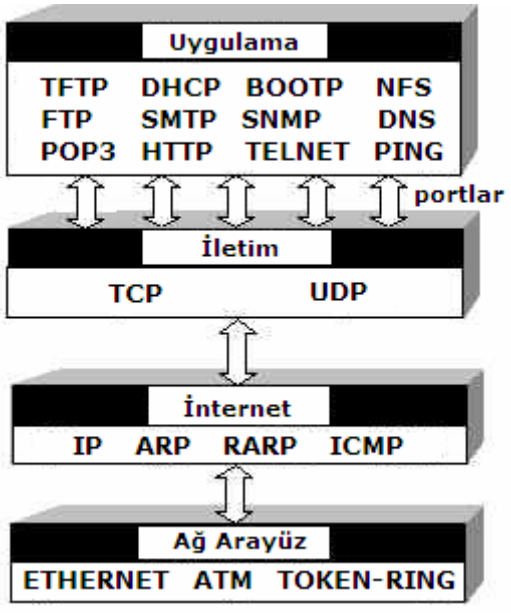
Herhangi bir paket anahtarlama noktasında, bir bağlantı üzerinden iletilmesi gereken birden fazla paket var ise (bu paketlerin alıcı ve gönderici adresleri farklı olabilir) bu paketler sırayla ve hattın tüm kapasitesi kullanılarak iletilir. Bir paketin anahtarlama noktasındaki bekleme süresi o noktadaki iletilecek paket yoğunluğuna bağlı olarak değişmektedir. Paketlerin iletim sırasında yüz yüze kaldığı sıra bekleme gecikmesi, gecikme değişkenliği (jitter) ve düzensiz olarak hedefe ulaşma gibi sebepler sesin paketlenerek iletilmesinde kaliteyi düşürebilmekte veya imkânsız hale getirebilmektedir. Bu nedenle tekrar sıralama, paket anahtarlamalı ağların oluşturduğu gecikmeyi dengeleme (dejitering) gibi teknikler kullanılmalı ve paket kayıplarının önlenmesine çalışılmalıdır.

2.4 TCP/IP Referans Modeli

Şu anda internet üzerinde çalışan birçok protokol ve uygulama, RFC adı verilen ve günümüzde sayıları iki bin civarında olan bir dizi teknik rapor ile belgelenir. RFC kitaplığının bakımı ve jüriliğini yapma görevi, Menio Park, Kaliforniya 'da bulunan *Network Information Center* tarafından yürütülmektedir.

Protokol, bir iletişim sürecinde bağlantıyı sağlayan noktalar arasında gidip gelen mesajlaşmayı düzenleyen kurallar dizisidir. Bu protokoller birbirleriyle iletişim içinde bulunan gerek donanım gerekse yazılımlar arasında çalışır. İletişimin gerçekleşmesi için her öğenin bu protokolü kabul etmiş ve uyguluyor olması gerekir.

İletim Kontrol Protokolü/İnternet Protokolü (TCP/IP) katmanlardan oluşan ve yüzden fazla bilgi iletişim protokolün toplandığı bir protokoller kümesidir. Her katman değişik görevlere sahip olup altındaki ve üstündeki katmanlar ile gerekli bilgi alışverişini sağlamakla yükümlüdür. Dört katmandan oluşan TCP/IP referans modelinde en üstte uygulama katmanı vardır. Uygulama katmanın altında sırasıyla iletim, internet ve ağ arayüzü katmanları bulunur. İletim katmanında TCP ve UDP protokolleri, internet katmanında IP ve ICMP protokolleri tanımlıdır. Her katmanda birçok protokol olmakla birlikte uygulama programları tarafından istenen bir iş yerine getirilirken, her katmandaki protokollerden yalnızca biri kullanılır. Şekil 2.5'de TCP/IP katmanlarını ve bu katmanlarda çalışan protokoller ile servislerden en çok kullanılanları gösterilmektedir. VoIP uygulanmalarında bizim için en önemli protokoller IP ve UDP'dir.



Şekil 2.5 TCP/IP katmanları

2.4.1 Uygulama Katmanı

Kullanıcının etkileşimde bulunduğu veya bilgisayar kaynaklarını başka kullanıcılara açmasını sağlayan uygulama programları doğrudan bu katmanla iletişim içindedir. Uygulama programlarının ağa erişimi için gerekli işlevleri kapsar ve bu programlara hizmet eden SMTP, TELNET gibi çokça ihtiyaç duyulan birçok protokolü içerir.

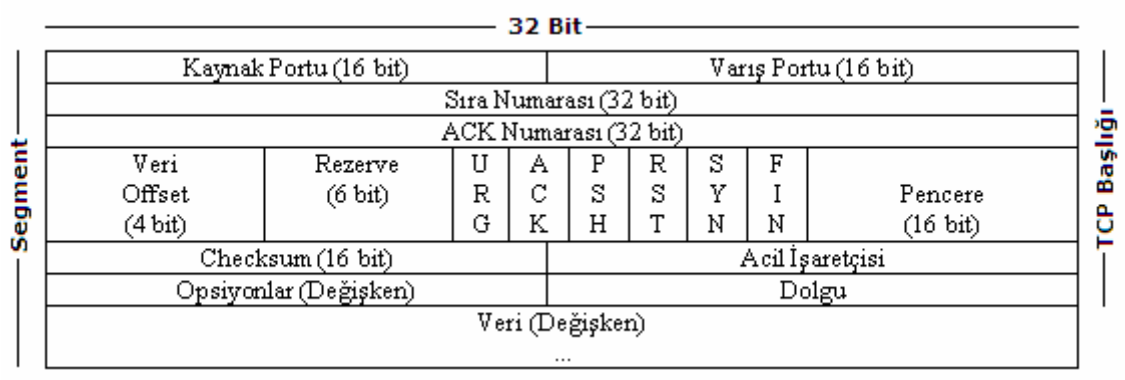
2.4.2 İletim Katmanı

Bu katmanın temel işlevi, iki uç arasında iletişim sağlamak ve üst katmandan gelen veriyi ihtiyaç duyulduğunda küçük bileşenlere ayırıp internet katmanına geçirerek, bu parçaların diğer uca iletilmesine aracılık etmektir. İletim katmanının oluşturduğu bilgi bloklarına "segment" denir. Bu blokların alıcı uca sırası bozulmuş olarak gelmesi duruma önlem olarak yeniden sıralamada kullanılmak üzere bölümler numaralanır. Bu katmanda iki adet protokolden birisi çalışır. Bunlar TCP ve UDP olarak adlandırılan protokollerdir.

2.4.2.1 TCP

TCP'nin temel işlevi, uygulama katmanından gelen bilginin segmentler haline dönüştürülmesi, iletişim ortamında kaybolan bilginin tekrar yollanması ve paketlerin alıcı tarafında doğru şekilde sıralanmasının sağlanmasıdır. Paketler için teslim garantisi isteyen uygulamalar tarafından kullanılır. TCP kendisine atanmış olan bu görevleri yapabilmek için

iletim katmanında veri parçalarının önüne kendi başlık bilgisini ekler. Başlık bilgisi ve veri parçası TCP segmenti olarak anılır. Şekil 2.6'da TCP segment yapısı görülmektedir.



Şekil 2.6 TCP segment yapısı

Kaynak Portu: Bir üst katmanda TCP hizmeti isteyen uygulamanın kimliği durumundadır. Mesaj geldiğinde bir üst katmana iletmek için, o protokolün adı değil de port numarası kullanılır.

Varış Portu: Gönderilen veri paketinin alıcı tarafta hangi porta/uygulamaya ait olduğunu belirtir.

Sıra Numarası: Gönderilen paketin sıra numarasını gösterir. Küçük parçalara ayrılan verinin, alıcı kısımda yeniden aynı sırada elde edilmesi için kullanılır.

ACK Numarası: Onay numarası, göndericiye verinin en son hangi sekizlisinin alındığını iletmek için kullanılır. Örneğin "n" sayısı gönderilirse, n'ye kadar bütün sekizlilerin alındığı belirtilir.

Veri Offset: TCP başlığını oluşturan, 32-bit sıralı kelimelerin sayısını belirtir. Bu alan, veri alanının nerede başladığının tespitinde kullanılır.

Rezerve: 0`a set edilmesi gereken 6 bitten oluşur. Bu bitler gelecekte kullanılmak için saklanmaktadır

URG: Bu bayrak, acil işaretçisi alanının etkin olup olmadığını belirtir.

ACK: Bu bayrak, onay alanının etkin olup olmadığını belirtir.

PSH: Bu bayrak, modülün push fonksiyonunu işletip işletmeyeceğini belirtir.

RST: Bu bayrak, bağlantının resetlenmesi gerektiğini bildirir.

SYN: Bu bayrak, sıra numaralarının eşzamanlamasının oluşturulmaya çalışıldığını bildirir. SYN bayrağı, elsıkışma işlemlerinin oluştuğunu belirtmek için kullanılır.

FIN: Bu bayrak göndericinin gönderecek başka verisi kalmadığını belirtir.

Pencere: Pencere değeri, alıcının kaç tane sekizli almayı beklediğini gösterir. Bu değer atanırken ACK alanındaki değere dayanılır. Pencere alanındaki değer, ACK alanındaki değere eklenir ve göndericinin iletmek istediği veri miktarı hesaplanır.

Checksum: Hata sınama bitleri, başlık ve metin de dahil olmak üzere segmentteki tüm 16-bit kelimelerin 1`e tümlenmiş toplamını içerir. Hata sınama hesabının yapılmasındaki amaç segmentin vericiden bozulmadan gelip gelmediğine karar vermektir.

Acil İşaretçisi: Bu alan yalnızca URG bayrağı aktif edildiğinde kullanılır. Acil işaretçisinin amacı acil verinin yerleştiği veri baytını belirtmektir. Acil veriye bant-dışı veri de denir. TCP acil veri için ne yapılacağını dikte etmez, bu uygulamaya-özeldir. TCP yalnızca acil verinin nereye yerleştirildiğini belirtir.

Opsiyonlar: TCP`ye gelecekte yapılacak eklemeler düşünülerek tasarlanmıştır.

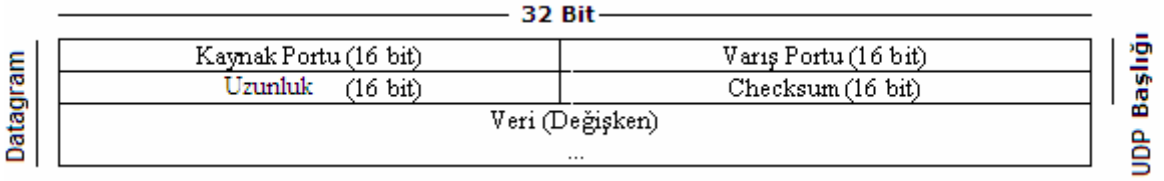
Veri: Aktarılacak veri parçasıdır.

2.4.2.2 UDP

UDP, datagramların belirli bir sıraya koyulmasının gerekli olmadığı uygulamalarda kullanılmak üzere dizayn edilmiştir. İletim katmanında UDP'nin oluşturduğu veri bütününe "datagram", TCP'nin oluşturduğu veri bütününe "segment" adı verilir. İkisi arasındaki temel fark, segmenti oluşturan veri grubunun başında sıra numarası bulunmasıdır. UDP'nin TCP'den ayrıldığı bir diğer nokta da, UDP'nin küçük boyutlu verinin aktarılması için kullanılmasıdır. Veri küçük boyutlu olduğu için parçalamaya gerek duyulmaz.

UDP protokolü ağ üzerinden paket gönderirken paketlerin kaydını tutmaz ve gidip gitmediğini takip etmez. Basit bir protokol olduğu için hızlı iletişim kurulması gereken yerlerde kullanılmaktadır. Buradaki basitlikten kasıt TCP protokolü gibi kontrol mekanizmaları içermemesidir.

TCP'de olduğu gibi UDP'de de bir başlık vardır, ancak UDP daha az bilgi içerdiği için başlığı TCP'nin başlığından daha kısadır ve ağ üzerinde fazla bant genişliği kaplamaz. UDP başlığı, kaynak ve varış port numaraları ile kontrol amaçlı verilerden oluşmaktadır. Ağ yazılımı Şekil 2.7'deki gibi bir UDP başlığını iletilecek bilginin başına koyar.



Şekil 2.7 UDP datagram yapısı

TCP bir veriyi karşıya başlık bilgisi ve veri yükünden oluşan bir paket olarak iletmektedir. Başlık bilgisi, her sırası 32 bitten oluşan 6 katmandan meydana gelmektedir. Yani her veri fazladan 192 bit başlık bilgisi taşımaktadır. Oysa UDP paketleri her sırası 32 bit'ten oluşan 2 katmandan meydana gelir ve toplamda 64 bitlik başlık bilgisine sahiptir. UDP kullanmanın en önemli nedeni protokol yükünün az olmasıdır. Ses gibi gerçek zamanlı veri akışı gerektiren bir uygulama için TCP fazla yük getirir ve ses gerçek zamanlı duyulamayabilir. Ayrıca UDP trafiğinde meydana gelecek az bir veri kaybı sesi veya görüntüyü bozmaz. Bu nedenle sıkı paket kontrolüne gerek yoktur. İyi bir fiziksel bağlantıda hata oranı düşük olacak ve bu nedenle TCP'nin yaptığı paket kontrol işlemleri fazladan yük olacaktır.

SERVİS	TCP	UDP
Bağlantı kurulması	İletime başlamadan önce bağlantının kurulması zaman alır. İki uç arasında oturum açılır.	İletime başlamadan önce bağlantı kurulmasına gerek yoktur. İki uç arasında oturum açılmaz.
Teslim garantisi	TCP paketlerinin alıcıya ulaştığını onaylanır.	Alıcı, paketin alındığına dair sinyal göndermez. Kaybolan paketler tekrar iletilmez.
Paket sıralaması	Paketler ardışık olarak numaralandırılır ve sıralı teslimini garanti altına alır	Paketler numaralandırılmaz. Paketlerin sürekli ulaştığı veya kaybolduğu düşünülür.
Akış kontrolü	Alıcı göndericiye yavaşlaması için sinyal gönderebilir.	Akış kontrolü uygulanmaz
Tıkanıklık kontolü	Network cihazları TCP onayları sayesinde göndericilerin davranışlarını kontrol edebilir.	Tıkanıklık kontrolü uygulanmaz.
Hız	TCP daha yavaştır, ek yükü fazladır ve yalnızca noktadan noktaya iletişimi destekler.	UDP hızlıdır, ek yükü azdır, noktadan noktaya ve noktan birden çok noktaya iletişimi destekleyebilir.

Tablo 2.1 TCP ve UDP'nin karşılaştırması

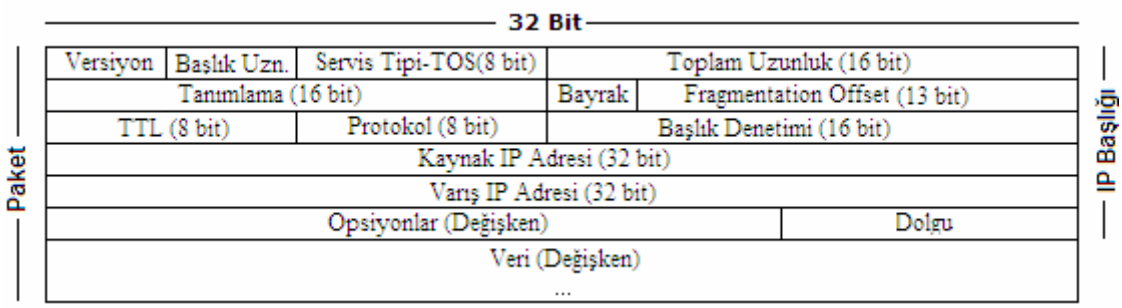
2.4.3 İnternet Katmanı

İnternet katmanı, iletim katmanından gelen verilerin IP paketleri haline dönüştürülüp yönlendirilmesinden sorumludur. Ayrıca bilgisayarlar arasındaki iletişimin nasıl

gerçekleşeceği de bu katman tarafından belirlenir. Bu katmanda çalışan en başlıca protokol IP'dir.

2.4.3.1 IP

TCP ya da UDP bir paket ürettiği zaman, iletim katmanı bu paketin ağ üzerinde iletilmesi için internet katmanına başvurur. Tüm veri iletimi IP üzerinden gerçekleştirilir. IP, TCP ya da UDP' den aldığı pakete ilgili adresleri içeren başlık bilgilerini ekler. IP veri transferi yaparken oturum açmadığı için "bağlantısız" bir protokol olarak adlandırılır, paketlerin iletimi konusunda bir garanti vermez. Veri hedefe ulaşsa da ulaşmasa da gönderici yada alıcı bilgilendirilmez. Bilgilendirme işi TCP gibi üst tabaka katmanların görevidir.



Şekil 2.8 IPv4 paket yapısı

Versiyon: Kullanılmakta olan internet protokolünün versiyonunu gösterir. Şu anda kullanılan IP'nin versiyonu 4' tür. Yakın zamanda versiyon 6'ya geçilmesi hedeflenmektedir. İpv4 ve ipv6 bir arada çalışabilirler.

Başlık Uzunluğu: IP başlığının uzunluğunu gösterir, bunun için 4 bit kullanılır. **Servis Tipi - TOS:** Servis Kalitesini (QoS) sağlamak amacıyla kullanılır. Ses verisinin gerçek-zamanlı iletilmesi söz konusu olduğunda uçtan-uca gecikmenin önemsenmesi gerekir.

Toplam Uzunluk: Başlık bilgisi ve üst katmandan gelen veri dahil olmak üzere IP paketinin toplam uzunluğunu gösterir.

Tanımlama: Bu alan için 16 bit kullanılır. Eğer paketler parçalanırsa (fragmentation), parçalanan tüm mesajlara yeniden birleştirme amacıyla aynı kimlik bilgileri verilir. **Bayraklar:** Bayraklar 1'er bitlik olmak üzere 3 tanedir. Birinci bit rezerve edilmiştir ve değeri sıfır olmalıdır. Bayrak bitinin amacı mesajın parçalanıp parçalanmadığını karşı tarafa iletmektir.

Fragmentation Offset: İnternet katmanının bölünmüş datagramlan tekrar birleştirmesini sağlar.

Yaşama Süresi - TTL: IP paketinin ağ üzerindeki dolaşma süresini gösterir. Paketin ağ üzerinde geçtiği her düğüm için TTL değeri 1 azaltılır. Eğer bu değer 0 olursa datagramın ağı daha fazla meşgul etmemesi için yok edilmesi gerektiğini belirtir.

Protokol: Datagramın içindeki verinin hangi protokolü kullandığını belirtir.

Başlık Denetimi: Başlık bilgisinin hatasız iletildiğini kontrol etmek amacıyla kullanılır. **Kaynak IP Adresi:** Gönderen tarafın IP adresini gösterir.

Hedef IP Adresi: Alıcı tarafın IP numarasını gösterir.

2.4.4 Ağ Arayüz Katmanı

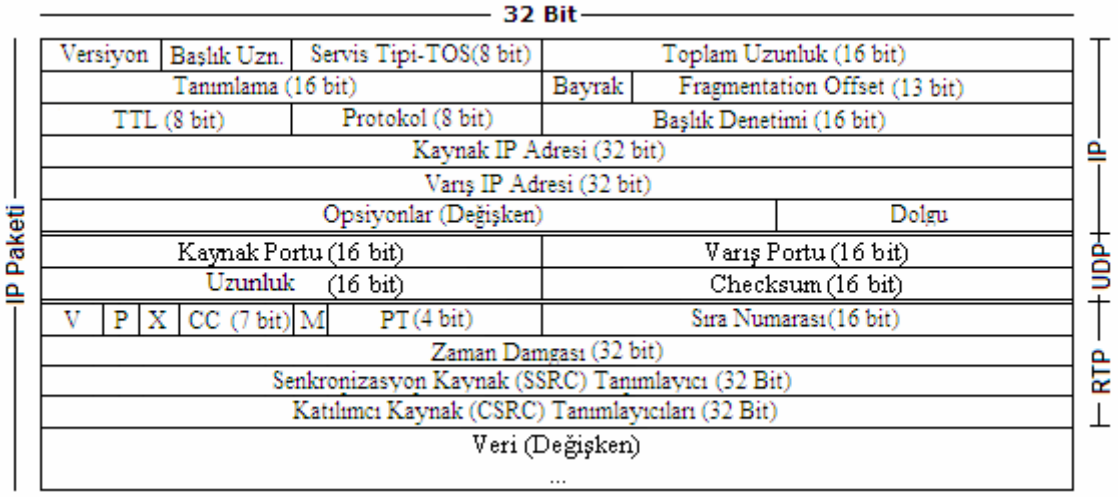
Verilerin elektriksel işaretler yoluyla duyuların algılama sınırlarının ötesinde çeşitli iletim ortamlarından fiziksel olarak aktarılmasını sağlar. ATM ya da Ethernet gibi protokoller bu katmanda çalışırlar.

2.5 RTP, cRTP, RTCP

RTP, gerçek zamanlı veri iletimi için IETF tarafından geliştirilmiş ve RFC 3550 ile tanımlanmış uçtan uca bir işletim protokolüdür, UDP üzerinde çalışır. Ses uygulamaları gerçek zamanlı çalışmalıdır. Gerçek zamandan kasıt sesin en az gecikme ile ve bozulmadan alıcı tarafından alınabilmesidir. Böylece alıcı, alınan işarete karşı düşen sesten istenen anlamı çıkarabilmeli ve etkileşimi bozmadan yanıt verebilmelidir. Geleneksel PSTN'deki devre anahtarlamalı teknoloji ile uçlar arasında atanmış bir iletim yolu kurulduğu için bu zamansal ilişki bozulmadan sağlanabilmektedir. Paket anahtarlamalı ortamda ise ses işaretlerine ilişkin paketlerin alıcı tarafta zamanında alınması bir yana bu paketlerin teslim edilme garantisi bile yoktur. Bu da alınan paketler arasındaki gecikme değişkenliğinin (jitter) verici taraftaki orijinal zamansal ilişkiden çok farklı olmasına yol açmaktadır. Bu nedenle RTP, alınan paketleri önce bir tampon bellekte depolayarak paket numaraları ve zaman damgalarına göre işleme alınmasını sağlar. Ancak paketler arasında ses için kritik olan 150-200 ms'den daha fazla zaman farkı oluşursa geç gelen paketler atılır. Aksi takdirde konuşmanın etkileşimlilik özelliği kaybolmakta, taraflar ya aynı anda konuşmaya başlamaktalar ya da iki taraf birden birbirinin konuşmaya başlamasını beklemektedirler.

Paket kaybı olursa, toplam paketlerin %5'inin veya daha fazlasının kaybedilmesi ile 'anlaşılabilirlik' bozulmaktadır.

IP ağında uçtan uca oluşan paket gecikmelerinin veya gecikme değişkenliğinin düzeltilebilmesi için tanımlanan RTP protokolü aslında TCP veya UDP gibi bir taşıma protokolü değildir. Daha ziyade, gerçek zamanlı uygulamaların kullandığı çerçeveleme protokolüdür. Ses paketleri RTP başlığı ile sarıldıktan sonra UDP ve IP paketleri içine konulur.



Şekil 2.9 RTP, UDP ve IP başlıklarının ilişkisi

RTP'nin 12-byte'lık başlığı paketin ağ içindeki yolculuğu esnasında başına gelebilecek istenmeyen olayların etkilerini belli bir düzeye kadar alıcı tarafta düzeltme olanağını veren bazı alanları içerir. (Şekil 2.9)

V-Version: Versiyon bilgisini içeren 2 bitlik alandır.

P-Padding: Dolgu biti değeri 1 olarak atanmışsa paketin sonunda bir ya da daha fazla, yükün parçası olmayan dolgu baytları mevcut demektir. Dolgu bitleri sabit boyutlu başlıklarda bazı kriptolama algoritmaları için kullanılabilmektedir.

X-Extension: Bu bit, başlığın ek bir başlık içerip içermediğini belirtir.

CC-CSRC Count: (Katılımcı Kaynağı Tanımlayıcısı – Conributing Source Identifier Count), sabit başlığı takip eden CSRC tanımlayıcı sayısını gösterir.

M-Marker: Ses paketleri için, bir konuşma bloğu başlangıcını gösterir. Konuşma bloklarının başlangıcı, şebekedeki gecikme değişkenliklerinin yanı sıra, alıcı ile verici saat oranları arasındaki farklılıkları telafi etmek için, alıcıda çalma gecikmesini ayarlamak amacıyla kullanılır.

PT-Payload Type: PT bitleri paketin içerdiği verinin tipini gösterir.

Sıra Numarası: Uygulama tarafından, alınan paketleri doğru bir şekilde sıralamak için kullanılır.

Zaman Damgası: Paket akışı için senkronizasyon bilgisini içerir.

Senkronizasyon Kaynak Tanımlayıcı (SSRC): Paketi gönderenin tanımlayıcı numarasıdır. Bir uygulamanın aynı anda hem ses hem video verisi iletmesi söz konusu olduğunda kullanılır. Bu yolla uygulama, alınan veriyi gruplama imkânına sahip olur.

Katılımcı Kaynak Tanımlayıcıları (CSRC): Farklı ses paketlerinin karıştırılması söz konusu olduğunda kullanılır.

RTP başlığı veri uzunluk alanı içermez. Çünkü RTP, TCP/IP mimarisinde UDP protokolünün üstünde çalışır. Dolayısıyla UDP veri işaretinin uzunluğunu tuttuğundan, RTP başlığı ilave olarak bu bilgiye yer ayırmaz.

Normalde IP/UDP/RTP başlıkları sırasıyla 20, 8, ve 12 byte'tır. 128 kbps'den düşük hızlı hatlarda, 40 baytlık RTP/UDP/IP toplam başlığı 20 baytlık ses paketinin (G.729 codec ile) taşınmasında çok verimsizdir. Herhangi bir başlık sıkıştırma işlemi (cRTP) olmadan bu yöntem taşıdığı yük için gerekli bant genişliğinden çok fazlasını başlığın taşınması için kullanacaktır. Bu nedenle RTP Başlık Sıkıştırması (cRTP) kullanarak bu geniş başlığı 2 veya 4 byte'a kadar sıkıştırabiliriz.

RTCP, RTP ile bağlantılı olarak çalışan kontrol protokolüdür. Bir RTP oturumundaki her katılımcı, tüm diğer katılımcılara periyodik olarak RTCP kontrol paketleri gönderir. Uygulamada bilgi geri beslemesi, performans kontrolü ve diagnostik amaçları için kullanılabilir.

RTCP aşağıdaki dört fonksiyonu gerçekler:

a. Uygulamaya Bilgi Sağlamak: Ana fonksiyon, bir uygulamaya veri dağıtımının kalitesi ile ilgili bilgi sağlamaktır. Her RTCP paketi, uygulama için kullanışlı istatistikler olan gönderici ve/veya alıcı raporları içerir. Bu istatistikler; gönderilen paket sayısı, kaybedilen paket sayısı, paketler arası jitter ve benzeri bilgiler içerir. Alıcının kalite hakkındaki bu geri beslemesi etkileşen birimler açısından kullanışlı olacaktır. Örneğin, gönderici geri beslemeyi temel alarak iletim hızını değiştirebilir veya problem oluştuğunda alıcılar problemin lokal, bölgesel veya genel olup olmadığını belirleyebilir. Bununla birlikte ağ yöneticileri RTCP paketlerindeki bilgiyi şebekelerinin performansını değerlendirmek amacıyla kullanabilirler.

b. RTP Kaynağını Tanımlamak: RTCP, bir RTP kaynağı için kanonik isim (müsaade edilmiş isim- CNAME) adı verilen bir iletim seviyesi tanımlayıcısı taşır. Bu CNAME, bir RTP oturumunun katılımcılarının kaydını tutmak için kullanılır. Alıcılar CNAME'i ilgili RTP oturumlarının bir grubundaki belli bir katılımcıdan gelen çoklu veri trafiğini ilişkilendirmek için, yani ses ve video senkronizasyonunu sağlamak için kullanabilir.

c. RTCP İletim Aralığının Kontrolü: Büyük ağlarda aşırı miktarda kontrol mesajının gelmesini önlemek ve RTP'nin çok sayıda oturum katılımcısına izin vermesi için, kontrol trafiği genel oturum trafiğinin en fazla yüzde 5'i ile sınırlandırılır. Her katılımcının diğerlerine kontrol paketleri göndermesinden dolayı, her katılımcı toplam sayıyı tutabilir ve bu rakamı RTCP paketlerinin gönderileceği oranı hesaplamada kullanabilir.

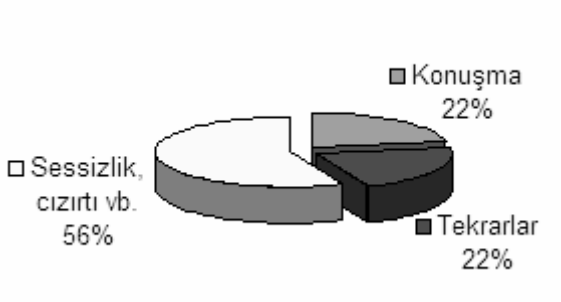
d. Asgari Oturum Kontrol Bilgisinin İletilmesi: RTCP tüm oturum katılımcılarına asgari bir oranda bilgi iletmek için uygun bir yöntem olarak kullanılabilir. Örneğin RTCP, kullanıcının ekranında katılımcının tanınabilmesi için kişisel bir isim taşıyabilir.

2.6 Ses Kodlama ve Sıkıştırma

Kodlama ve sıkıştırma, iletilecek veri miktarını azaltmak için kullanılan yöntemlerdir.

2.6.1 Konuşmanın Bileşenleri

İncelenen bir ses örneğinin analizi, tipik bir diyaloğun sadece yüzde 22'sinin tam bir ses anlaşılırlığı için iletilmesinin gerektiğini göstermiştir. Kalan yüzde 78'lik kısım, yüzde 22 oranında tekrarlanan örneklerden ve yüzde 56 oranında sessizlikten, gereksiz bilgiden oluşmaktadır (Şekil 2.10). Tekrarlı sesler, insan sesinin ayrılmaz parçasıdır ve ses tellerinin titreşmesiyle oluşurlar. Bu tekrarlar (Türkçe'de 'sinek' kelimesindeki "s" veya 'anne' kelimesindeki uzun "n" gibi) kolayca sıkıştırılabilir. Bu eş seslerin iletimi gerekli değildir ve bunların çıkarılması, bant genişliğinin verimliliğini arttırır. Ayrıca, konuşan bir kişi kesintisiz bir bilgi akışı sağlamadığından kelimeler ve cümleler arasındaki kesintiler de çıkarılabilir, bu sessizlik sıkıştırması olarak bilinir. Kesintiler sıkıştırılmış formda sunulup, karşı tarafta konuşma haberleşmesinin doğal kalitesini korumak için yeniden yaratılabilir. Kesintilerin söylenenin anlaşılabilmesi için büyük önemi vardır.



Şekil 2.10 Konuşmanın Bileşenleri

2.6.2 Kodlama Teknikleri

Ses analizinde iki önemli teknik kullanılır, dalga formu kodlama (waveform coding) ve ses kodlama (voice coding - vocoding).

PCM, ADPCM gibi dalga formu kodlama metotları, doğrudan analog konuşma tarafından üretilen dalga formunu, sinyalin örneklenmesi ve her örneklemenin genliğinin en yakın değerine dönüştürülmesiyle kodlar. Bu işleme "kuantalama", gerçek genlik ile örneklenen değer arasındaki farklılığa da "kuantalama hatası" denir.

Bir vocoder, orijinal konuşmaya benzer bir sinyal üretmek için girişi kullanır. Bunu yaparken, vocoder her konuşma örneğini uygun parametre setini tanımlamak amacıyla analiz eder. Bu parametreler konuşmayı daha sonra bir sentezleme işlemi üzerinden yeniden oluşturacak olan alıcıya gönderilir. Vocoder'lar daha az bant genişliği gerektirir ve *Linear Predictive Coding*, *Code-Excited Linear Prediction - Multi Pulse*, *Multi-Level Quantisation* tekniklerini kullanır.

2.6.3 Ses Kodlama

Ses kodlama, örneksel ses sinyalinin sayısal hale dönüştürülmesi anlamına gelmektedir. PCM teknolojisi pek çok ses iletim ağında örneksel ses sinyalinin 64-kbps hızında sayısal bilgi olarak iletimi için standart olarak kullanılmaktadır. Ses kodlamasıyla, standart olarak 64-kbps hızında kodlanmış olan sayısal bilginin daha az miktarda veriyle gösterilmesi mümkün olmaktadır. Teknolojideki gelişmeler sıkıştırılmış sesin kalitesini oldukça artırmış ve ITU standartlarında sıkıştırma algoritmaları belirlenmiştir. Sesin sıkıştırılmasından bahsederken sıkıştırılmış sesten elde edilen kalite ve kullanılan bant genişliği hakkında bazı kararların verilmesi gerekir.

Yaygın kullanılan ses kodlayıcıları ve kod çözücüleri şu şekilde verilebilir.

G.711 Sıkıştırması: Örnekleme frekansı 8 kHz'dir. Örnek başına 8 bit kullanılır. Toplam bit hızı gereksinimi 64 Kbps'dir. Standart PCM sıkıştırmasıdır. MOS değeri 4,3'tür.

G.723 Sıkıştırması: Genel olarak düşük bit hızlarında iletim için kullanılır ve kalite olarak G.711'den kötüdür. Bununla birlikte bant genişliği gereksinimi G711'in yaklaşık onda biri kadardır. ACELP (6,3 Kbps) ya da MP-PLQ (5,3 Kbps) algoritmaları kullanılır. MOS değeri 4,1'dir.

G.726 Sıkıştırması: 16, 24, 32, 40 Kbps bit hızlarında ADPCM kodlaması kullanır. MOS değeri 2 ila 4,3 arasında değişir.

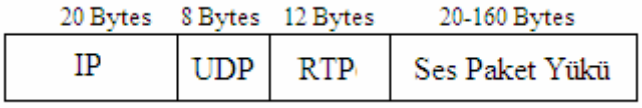
G.728 Sıkıştırması: Düşük gecikmeli CELP kullanır. Bit hızı 16 Kbps ve MOS değeri 4,1'dir.

G.729 Sıkıştırması: CS-ACELP algoritması kullanılır. Bit hızı 8 Kbps'dır. MOS değeri 4,1'dir.

Tablo 2.2'de yaygın olarak kullanılan sıkıştırma algoritmaları ve bunlara ilişkin başarım değerleri verilmiştir.

Codec	Ses Bantgenişliği	Ses Paketi Yükü	Ses Paketi Boyutu	IP/UDP/RTP Başlığı	cRTP	L2 Başlığı (Ethernet)	Toplam Bantgenişliği
g.711	64 Kbps	160 Bytes	20 ms	40 Bytes		14 Bytes	85,6 Kbps
g.711	64 Kbps	160 Bytes	20 ms		2 Bytes	14 Bytes	70,4 Kbps
g.729	8 Kbps	20 Bytes	20 ms	40 Bytes		14 Bytes	29,6 Kbps
g.729	8 Kbps	20 Bytes	20 ms		2 Bytes	14 Bytes	14,4 Kbps
g.723.1	6.3 Kbps	20 Bytes	30 ms	40 Bytes		14 Bytes	23,31 Kbps
g.723.1	6.3 Kbps	20 Bytes	30 ms		2 Bytes	14 Bytes	11,34 Kbps

Tablo 2.2 Yaygın kullanılan codec'lerin başarım oranları



Şekil 2.11 Ses paketinin yapısı

Bant genişliğinin hesaplanmasında

$$\text{Toplam Paket Boyutu} = \text{Ses Paket Yükü} + \text{IP/UDP/RTP(veya cRTP)} + \text{L2 Başlığı} \quad (2.1)$$

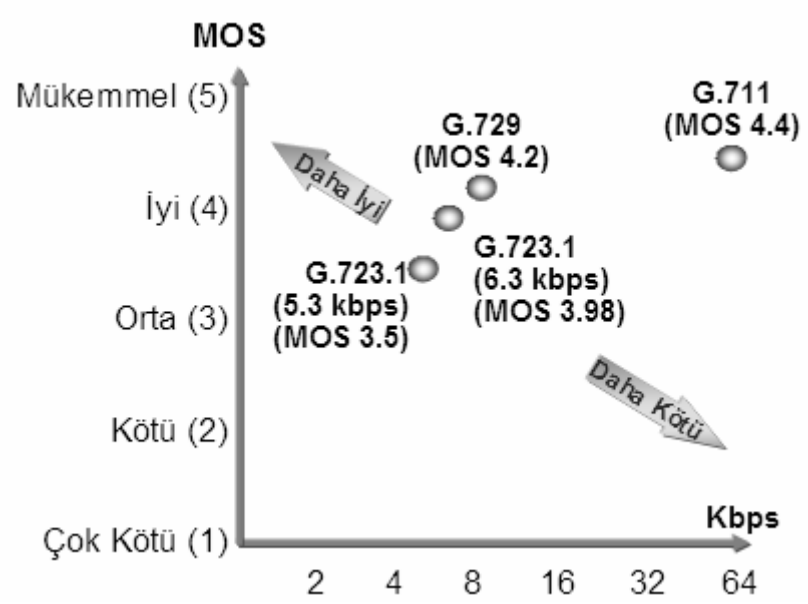
$$\text{Toplam Bant Genişliği} = \frac{\text{Ses Bant Genişliği x Toplam Paket Boyutu}}{\text{Ses Paket Yükü}} \quad (2.2)$$

eşitliklerinden yararlanılır.

Tablo 2.1'den de görülebileceği gibi, VoIP yapılmak istenen ağ WAN ise G.729 sıkıştırma tekniği tercih edilebilir. G.729, G.723 ile kıyaslandığında çok daha iyi bir ses kalitesi sağlar ancak yüzde yirmibeş daha fazla bant genişliğine ihtiyaç duyar. Ayrıca, G.723 gibi yüksek sıkıştırma yapan codec'ler kullanılırken codec gecikmesinin de sistem performansını olumsuz etkileyeceği göz ardı edilmemelidir. Bu durumda bant genişliğinden kazanılsa bile işlemci gücünden kaybedilmiş olacaktır. LAN'larda durum biraz daha farklıdır. Veri hızları 10-1000 Mbps aralığında olduğundan bant genişliği konusunda pek sıkıntı yaşanmaz, bu nedenle G.711 codec'i kullanılabilir.

2.6.4 Ortalama Algı Değeri - MOS

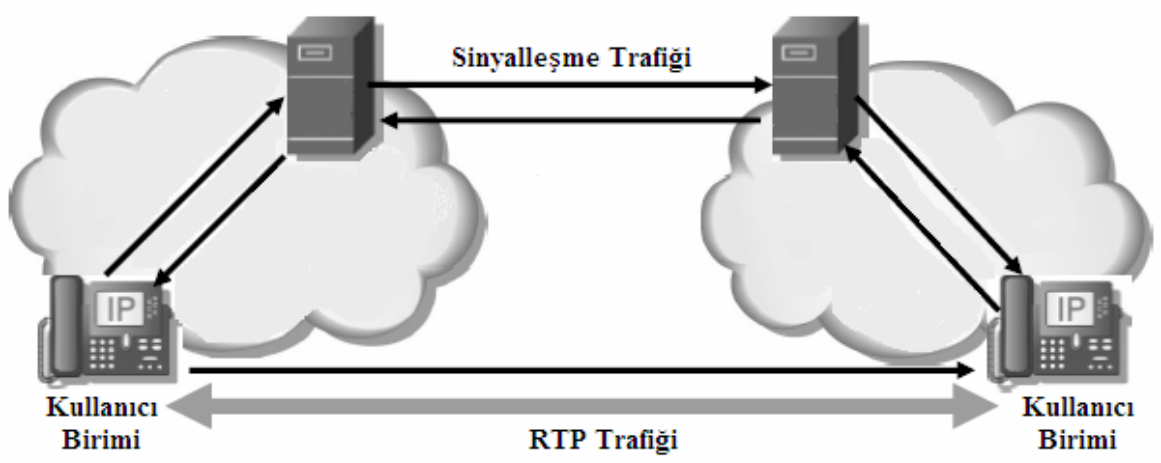
Ortalama algı değeri (MOS) codec'lerin performansını rakamlarla belirtmek için kullanılan subjektif bir karşılaştırmadır. Ses kalitesi ve genel olarak ses, dinleyicilere karşı subjektif olduğu için MOS testleri bir grup dinleyiciye uygulanır. MOS testi yürütülürken çok sayıda dinleyici ve örnek materyal kullanmak önemlidir. Dinleyiciler konuşma materyallerinin her örneğine 1'den (kötü) 5'e (mükemmel) kadar oy verirler. MOS'u elde etmek için sonuçların ortalaması alınır. 4-5 arasındaki değerler kabul edilebilir telefon sesi kalitesi, 3-4 arasındakiler iletişim kalitesi, 3'ün altındaki değerler ise sentetik ses kalitesini ifade eder. Şekil 2.12'de yaygın kullanılan codec'ler için MOS değerleri gösterilmektedir.



Şekil 2.12 Yaygın kullanılan codec'lerin MOS değerleri

2.7 Ses İletiminde Kullanılan Sinyalleşme Protokolleri

VoIP dünyasında trafiğin (ses veya video) kontrolü ve taşınması birbirinden ayrı olarak gerçekleştirilir. Görüntü ve ses verisi RTP ile taşınırken, oturumun yönetimi H.323 veya SIP gibi protokoller ile yönetilir. Bu protokollerin yönetim işlemine sinyalleşme de denir. Bir VoIP görüşmesinde biri gidiş diğeri geliş olmak üzere 2 RTP oturumu ile 1 RTCP oturumu kurulur. (Şekil 2.13)



Şekil 2.13 Sinyalleşme ve RTP trafikleri

2.7.1 SIP

Oturum başlatma protokolü (SIP), IETF tarafından geliştirilen ve IP üzerinden iki veya daha fazla uç nokta arasında çağrı kurma, tutma ve sonlandırma işlemleri için kullanılabilecek ASCII tabanlı, metin içerikli ve şifrelenmemiş bir protokoldür. Protokol 1999 yılında RFC 2543'e uygun olarak yayınlanmıştır. Tıpkı diğer VoIP protokolleri gibi paket anahtarlamalı ağlarda sinyalleşme ve oturum yönetimi işlevlerini yerine getirir. Aranılan uç noktanın konumunu, durumunu, ortam yeteneklerini sorgulayarak arayan ve aranılan noktalar arasında çağrı kurulması, çağrıların aktarılması ve sonlandırılması işlemlerini gerçekleştirebilir.

2.7.1.1 SIP Mesajları

SIP'te haberleşme, uç noktaların birbirlerine gönderdikleri istek ve cevap mesajları olmak üzere 2 tip mesajla sağlanır. Bu mesajlar sistematik biçimde gruplanmıştır.

2.7.1.1.1 İstek Mesajları

İstek mesajları 6 tanedir ve aşağıdaki sözdizimi ile başlar.

İSTEK MESAJI sip:x@y:port sip_versiyonu

Burada "x" ulaşılmak istenen numara/kullanıcı_adı, "y" ise ya ulaşılmak istenen kullanıcının ip_adresi/alan_adı, ya da Proxy ip_adresi/ alan_adı' dır. Aşağıda örnek bir sözdizimi verilmiştir.

INVITE sip:02123340800@85.84.83.82:5060 SIP/2.0

SIP'te kullanılan ve RFC 3261 ile belirlenen 6 istek mesajı INVITE, ACK, OPTIONS, BYE, CANCEL ve REGISTER'dır (Tablo 2.3). Bu mesajlara diğer RFC'ler ile birlikte INFO, PRACK, SUBSCRIBE, NOTIFY, UPDATE, MESSAGE,REFER, PUBLISH mesajları da eklenmiştir

MESAJ	Açıklama
INVITE	Çağrıyı başlatmak için kullanılır.
ACK	Çağrı başlatma talebinin sonucunu bildirir.
OPTIONS	Kullanıcı ve sunucu yeteneklerinin sorgulanması için kullanılır.
BYE	Devam eden çağrıyı sonlandırmak için kullanılır
CANCEL	Devam eden çağrı talebinin iptal edilmesi için kullanılır.
REGISTER	Konum bilgilerinin kayıt sunucusuna bildirilmesinde kullanılır.

Tablo 2.3 SIP istek mesajları

2.7.1.1.2 Cevap Mesajları

Cevap mesajları altı sınıfa ayrılmış durum kod gruplarından oluşur ve aşağıdaki sözdizimi ile başlar.

SIP_VERSİYONU durum_kodu açıklama_kelimesi

Aşağıda örnek bir sözdizimi verilmiştir.

SIP/2.0 100 Trying

SIP cevap mesajları, istek mesajlarına cevap olarak gönderilen mesaj tipleridir.(Tablo 2.4)

DURUM KODU	Durum Açıklaması
1xx	Bilgilendirme
2xx	Başarı
3xx	Yönlendirme
4xx	İstemci hatası
5xx	Sunucu hatası
6xx	Genel hata

Tablo 2.4 SIP Cevap Mesajları

2.7.1.2 SIP Bileşenleri

SIP noktadan noktaya bir protokoldür ve her bir uç "kullanıcı birimi (User Agent-UA) " olarak isimlendirilir. Bir SIP kullanıcı birimi hem istemci uygulaması hem de sunucu uygulaması içerir ve oturum sırasında her iki konumda da çalışabilir. Bu yüzden istekte bulunurken "istemci kullanıcı birimi (UAC)" gibi, isteklere cevap üretirken "sunucu kullanıcı birimi (UAS)" gibi davranır.

UA'lar SIP ağında aşağıdaki görevlerden birini yerine getirirler:

- **İstemci Kullanıcı Birimi (UAC):** SIP isteği gönderen istemci uygulamadır. IP telefonları, yazılımsal SIP telefonları ve PSTN dönüşümünü sağlayan gateway'ler UAC olarak davranırlar.
- **Sunucu Kullanıcı Birimi (UAS):** İstemci isteklerini karşılayan ve cevap veren sunucu uygulamadır. SIP proxy sunucusu, SIP yönlendirme sunucusu ve kayıt sunucusu UAS olarak davranır

UAC gönderdiği isteklerde kendi yeteneklerini ve fonksiyonlarını bildirmelidir. Bu UAS'nin yetenek sorgusu yapmaksızın UAC'nin bilgilerine sahip olmasını sağlar. Aşağıda tek bir INVITE mesajı altında sağlanan bilgiler görülebilir. Bu mesaja geri dönen cevap mesajı da yine aşağıda verilmiştir.

UAC'nin istek mesajı:

INVITE sip:03124440532@85.84.83.81:5060 SIP/2.0
Via: SIP/2.0/UDP 85.83.3.82:5060;branch=z9hG4bK18B1252294
From: "anonymous" <sip:02123340800@85.29.1.24>;tag=940A4FA4-109F
To: <sip: 03124440532@85.84.83.81>
Date: Thu, 21 Jun 2007 06:08:07 GMT
Call-ID: 9D95F64A-1EF411DC-A85CB6EE-2E54AE98@85.83.3.82
Supported: 100rel,timer,replaces
Min-SE: 1800
Cisco-Guid: 2643810850-519311836-3119382552-419390738
User-Agent: Cisco-SIPGateway/IOS-12.x
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, COMET, REFER, SUBSCRIBE, NOTIFY, INFO, UPDATE, REGISTER
CSeq: 101 INVITE
Max-Forwards: 70
Remote-Party-ID: <sip:02123340800@85.83.3.82>; party=calling;screen=no;privacy=full
Timestamp: 1182406087
Contact: <sip: 02123340800@85.83.3.82:5060>
Expires: 180
Allow-Events: telephone-event
Content-Type: application/sdp
Content-Length: 301

v=0
o=CiscoSystemsSIP-GW-UserAgent 1776 4832 IN IP4 85.29.3.13
s=SIP Call
c=IN IP4 85.83.3.82
t=0 0
m=audio 17618 RTP/AVP 18 0 8 101
c=IN IP4 85.83.3.82
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=yes
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16

UAS'nin ilk cevap mesajı:

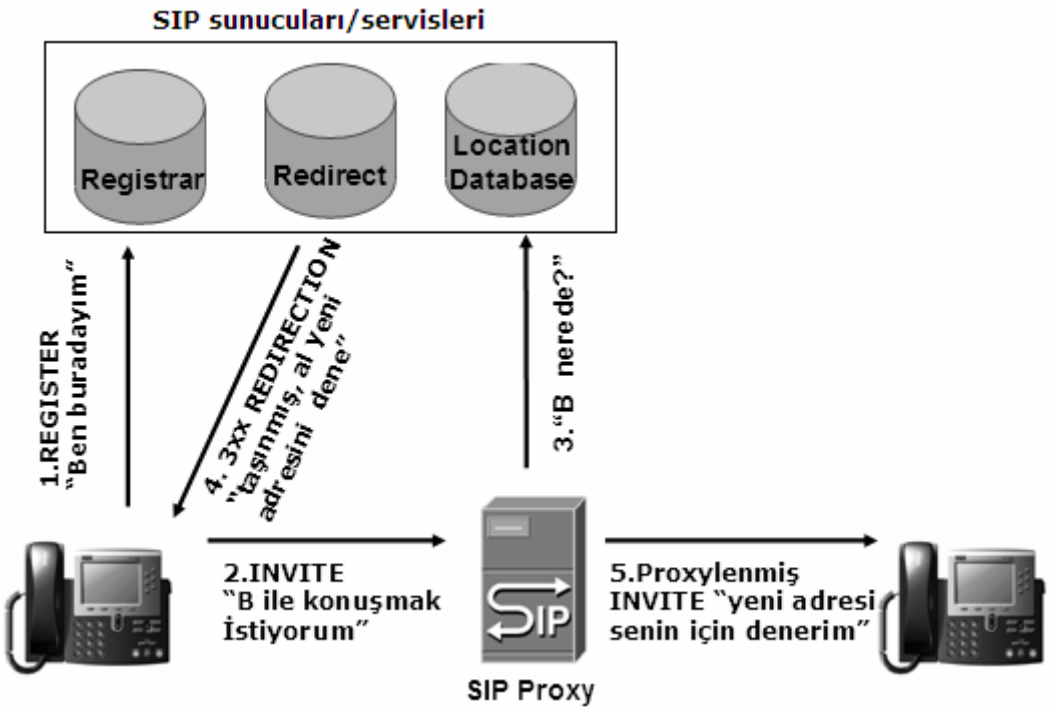
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 85.83.3.82:5060;branch=z9hG4bK18B1252294
From: "anonymous" <sip: 02123340800@85.84.83.81>;tag=940A4FA4-109F
To: <sip: 03124440532@85.84.83.81>
Call-ID: 9D95F64A-1EF411DC-A85CB6EE-2E54AE98@85.29.3.13
CSeq: 101 INVITE
Content-Length: 0

SIP sunucuları SIP isteklerini kabul eden ve onları cevaplayan uygulamalardır ve kullanıcı birimi sunucusu (UAS) ile karıştırılmamalıdır. SIP sunucuları, kullanıcı birimlerine hizmet ve özellik sağlar. (Şekil 2.14)

Proxy Sunucusu (Proxy Server): SIP Proxy sunucusu, kullanıcı biriminden ya da diğer Proxy'den SIP isteğini alır ve isteğin iletilmesi ya da cevaplanması için kullanıcı birimi adına hareket eder. Proxy sunucusunun isteğin işlenmesine yardımcı olan veritabanına veya konum hizmetine giriş izni vardır. Veritabanları SIP kayıtlarını, varlık bilgilerini ve kullanıcının konumunu gösteren diğer bilgileri içerirler.

Yeniden Yönlendirme Sunucusu (Redirect Server): İstekleri iletmeyen sadece cevaplandıran SIP sunucusudur. Proxy sunucusunda olduğu gibi kullanıcının konumunu belirlemek için veritabanı ya da konum hizmeti kullanır.

Kayıt Sunucusu (Registrar): Kayıt sunucusu kullanıcılarının konum bilgilerini girmelerini sağlayan sunucudur. İstek gönderen kullanıcıların konum bilgilerini veritabanında günceller.



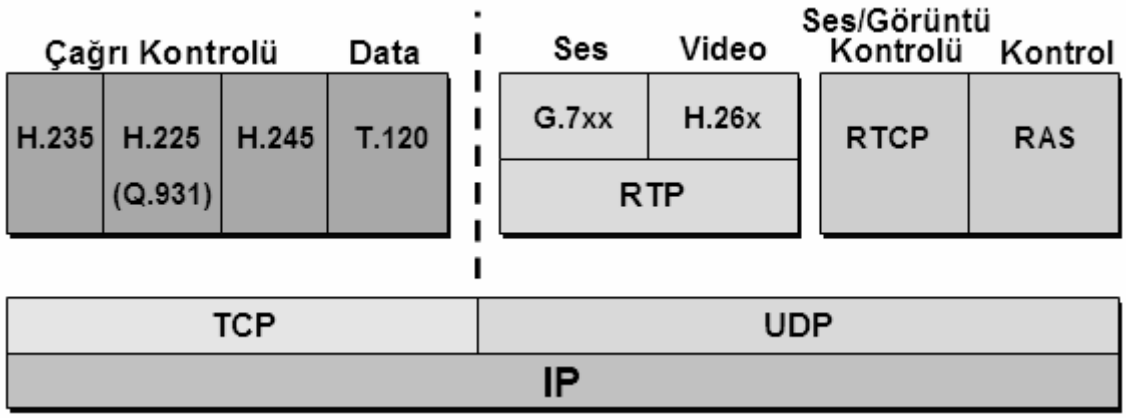
Şekil 2.14 SIP Sinyalleşmesi

2.7.2 H.323

ITU-T tarafından paket anahtarlamalı ağlarda iki ya da daha fazla taraf arasında çoklu ortam trafiğini taşımak için geliştirilen H.323 standardı, ISDN (Q.931) mantığıyla çalışan ve alt protokoller kullanan bir protokol grubudur (Şekil 2.15). İlk sürümü 1996'da çıkan protokolün son sürümü (H.323 v6) 2006 yılında yayınlanmıştır.

2.7.2.1 H.323 Protokol Grubu

Çağrının kurulması, yönetilmesi ve sonlandırılmasında H.323 protokol grubu içindeki birçok protokol görev almakta ve çeşitli işlevleri yerine getirmektedir.



Şekil 2.15 H.323 protokol grubu

H.323'e özgü protokoller aşağıdaki gibidir;

H.235 : Güvenlik servislerinin sağlanması, şifreleme gibi görevleri yerine getirir.

H.225 : H.323 uç noktaları veya bir uç nokta ile bir Gatekeeper arasında çağrı kurma ve RAS (Registration, Admission and Status) mesajlarının taşınması için kullanılan protokoldür. TCP 1720 portundan güvenli bir çağrı kontrol kanalı yaratıldıktan sonra, bu port üzerinden çağrıların kurulması/sonlandırılması için Q.931 çağrı kontrolü sinyalleşme mesajları başlatılır

H.245 : H.323 birimleri arasındaki uçtan uca kontrol mesajlarını tutar. Ses, görüntü, veri ve kontrol kanal bilgisinin iletilmesi için mantıksal kanal kurar. Bir kullanıcı her çağrı için bir H.245 kanalı kurar. Çağrı işaretleşme mesajındaki dinamik TCP portu kullanılarak IP üzerinden güvenli kontrol kanalı oluşturulur. Bu kontrol kanalı üzerinden yeteneklerin takası, mantıksal kanalların açılıp-kapanması, mesaj kontrolü gibi işlemler yapılır.

T.120 : Çoklu ortam veri iletim standardıdır. Verilerin gerçek zamanlı ve güvenilir bir şekilde nasıl iletileceğini tanımlar.

RAS (Registration, Admission and Status) : RAS uç noktalar ve gatekeeperlar çağrı öncesi kontrolü sağlayan bir protokoldür. Diğer tüm kanallar kurulmadan önce RAS kanalı açılır. RAS mesajları UDP üzerinden taşınır ve kayıt, giriş izni, bant genişliği değişikliği, durum, bağlantı kesme işlemlerini içerir.

2.7.2.2 H.323 Sisteminin Bileşenleri

H.323 sistemi; Terminaller, Gateway'ler, Gatekeeper'lar ve Çok Noktalı Kontrol Birimleri'nden (MCU) oluşur.

2.7.2.2.1 Terminal

Terminaller bir diğer H.323 terminali, gateway, gatekeeper veya MCU ile gerçek zamanlı iki yönlü haberleşme sağlayan istemcilerdir. Yazılım veya donanım olabilirler.

2.7.2.2.2 Gateway

Gateway'ler, farklı ağ teknolojilerini (PSTN, IP gibi) ve protokolleri (SIP, H.323 gibi) destekleyen ara yüzleri olan, bu sayede H.323 terminalleri ile bu ara yüzler ardında bulunan aboneleri konuşturabilen cihazlardır.

2.7.2.2.3 Gatekeeper

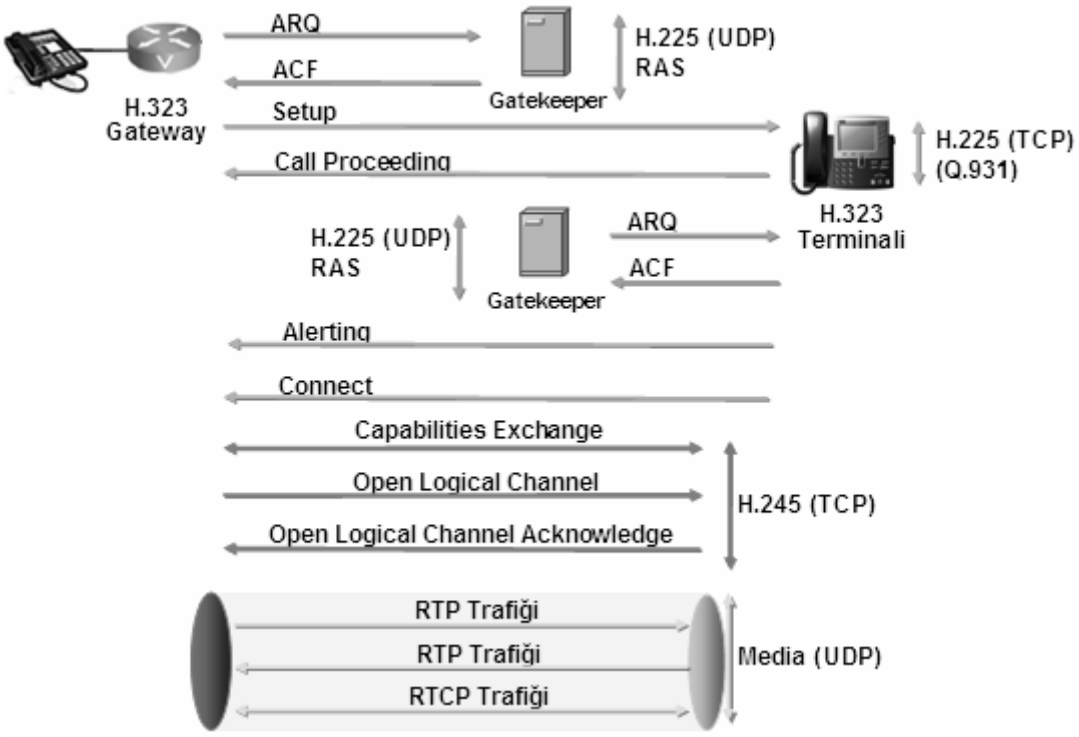
Gatekeeper, terminallerin ve gateway'lerin kayıt, giriş izni ve durum (Registration, Admission and Status - RAS) takibinden sorumlu olan ağ modülüdür. Gatekeeperlar alan yönetimi işlevlerini de yerine getirirler. Uç noktaların hangi gatekeeper'a kayıt olacağı manuel veya otomatik olarak belirlenebilir

2.7.2.2.4 MCU (Multi-point Control Unit)

MCU'lar ikiden fazla terminalin yada gateway'in konferansa katılımlarını sağlamaya yarayan cihazlardır. Konferansta tüm katılımcılar doğrudan MCU cihazına bağlanırlar. MCU'lar, MC (Multipoint Controller) ve bulunması zorunlu olmayan MP (Multipoint processor) olmak üzere iki kısımdan meydana gelirler. MC konferansa katılanlar arasında çağrı sinyalleşmesi ve konferans kontrolünü gerçekleştirirken, MP sesin işlenmesini ve gerektiğinde farklı ses kodlama tekniklerini kullanan kullanıcılar arasında codec dönüşümü yapar.

2.7.2.3 H.323 Çağrı Kurulumu

Şekil 2.16'da gatekeeper kullanılarak iki uç nokta arasında oturum başlatılması gösterilmiştir. Konferans başlamadan önce hem gateway hem de terminalin gatekeeper'a önceden kayıt olduğu varsayılmıştır.



Şekil 2.16 H.323 çağrı kurulumu

Uç noktalardan biri çağrı başlatmak istediğinde gatekeeper'dan "AdmissioncRequest (ARQ)" mesajı ile onay ister. Gatekeeper çağrıya "Admission Confirm (ACF") mesajı ile onay verir veya "Admission Reject (ARJ)" mesajı ile reddeder. Gatekeeper ile yapılan bu sinyalleşmeler için UDP kullanılır. Gerekli sorgulamalardan sonra çağrı yapma isteği kabul edilirse çağrıyı başlatan uç nokta diğer tarafa TCP üzerinden Q.931 Setup mesajı gönderir. Setup mesajını alan taraf da bağlı olduğu gatekeeper'dan ARQ mesajı ile çağrıyı kabul etmek için onay ister. Çağrı kabul edildikten sonra Q.931 işaret akışı H.245 uzlaşma mesajları ile tamamlanır. Bu aşamadan sonra ses paketlerinin iletildiği RTP trafiği iki yönlü olarak başlar.

ARQ mesajları konferans boyunca taraflara gerekecek bant genişliği taleplerini de içerir. Eğer H.245 uzlaşma mesajları sırasında uç taraflardan biri ARQ mesajında belirtilenden daha fazla bant genişliğine ihtiyaç duyarsa, gatekeeper'a "Bandwidth Request (BRQ)" mesajı göndererek bant genişliği talebinde bulunur. Gatekeeper "Bandwidth Confirm (BCF)" mesajı ile talebi kabul ettiğini yada "Bandwidth Reject (BRJ)" mesajı ile talebi kabul etmediğini terminale bildirir.

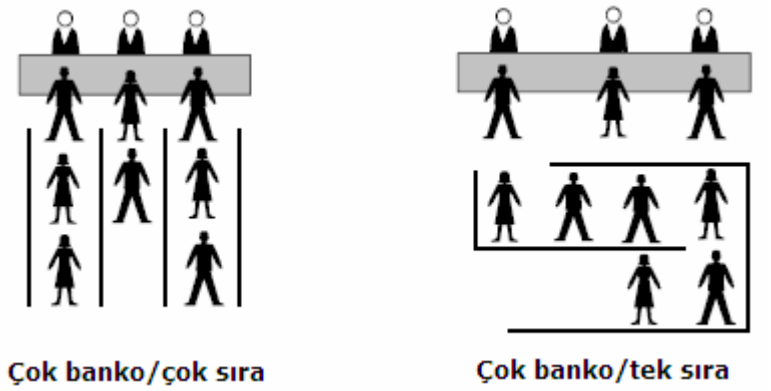
3. IP ÜZERİNDEN SES İLETİMİNDE HİZMET KALİTESİ

Ağ kavramının ortaya çıkması ve kaynakların paylaşılmaya başlanmasıyla hizmet kalitesi konusu da ortaya çıkmıştır. Günümüz kullanıcılarının ses, video gibi çoklu ortam uygulamalarını yoğun olarak kullanmaya başlaması bu konuyu küçük ağlarda bile önemli hale getirmiştir. Bir ağ'da uçtan-uca QoS sağlanabilmesi için ağ bileşenlerinin, üzerlerinden geçen veri akışına sürekli müdahalede bulunması ve kurallara uyulması konusunda zorlayıcı olması gereklidir.

3.1 Hizmet Kalitesi (QoS)

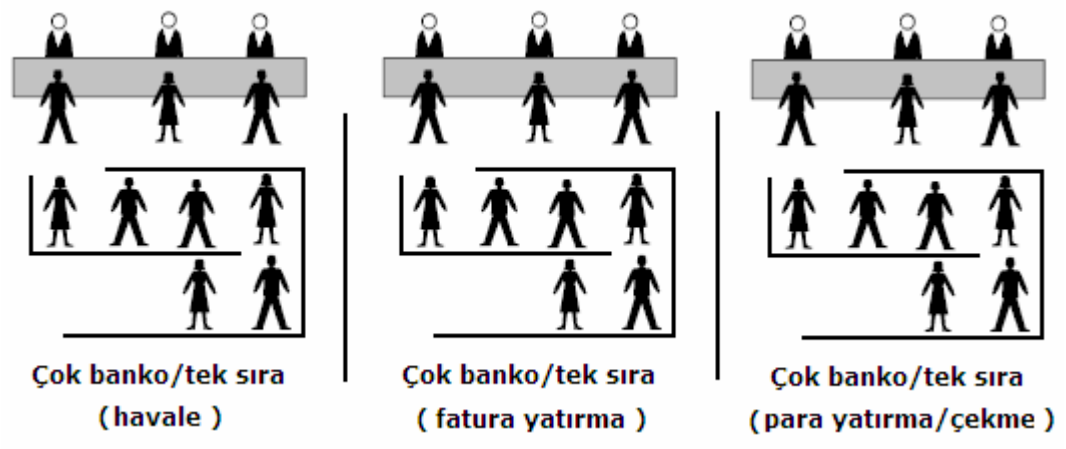
Hizmet kalitesi kavramını Şekil 3.1'de benzetimi verilen sıkça kullandığımız bankacılık sistemi ile açıklamaya çalışalım. Bundan on yıl öncesine kadar, bankaya gittiğinizde birkaç banko görevlisi ve bu görevlilerin önlerinde işlemlerini yaptırmak için sıra bekleyen insanlar görürdük. Eğer önünüzdeki müşterinin işi uzunsa, onun işi bitene kadar sizden sonra bankaya gelen kişiler bile hızlı ilerleyen sıralarda işlerini çabucak bitirip giderlerdi.

Sonraları, bankaya gelen kişilere numara dağıtılmaya başlandı. Bu uygulamada da yine birkaç banko görevlisi vardı, ancak bu sefer her müşteri tek bir sıra için numara almaya başlamıştı. Sizden önceki müşterinin işi uzun olsa dahi diğer bankolar boşaldığında önünüzdekini beklemeye gerek kalmadan işleminizi yaptırabiliyordunuz.



Şekil 3.1 Çok banko/çok sıra ile çok banko/tek sıra karşılaştırması

En sonunda, her iki yöntem birleştirilerek numara dağıtım mekanizması yaptıracağınız işleme göre farklı sıraların numaralarını vermeye başladı.



Şekil 3.2 Çok banko/çok sıra ile çok banko/tek sıra entegrasyonu

Böylece Şekil 3.2'de verilen çok bankolu/çok sıralı bir sistemden çok bankolu/tek sıralı sisteme geçilerek ortalama bekleme süresi kısaltıldı. Ancak, bu seferde hızlı ilerleyen sırada bekleyen müşteriler için işlemini çok daha hızlı bitirip gitme şansı kalmadı. Sonuçta birçok müşteri için hizmet kalitesi artarken bazıları içinse azalmış oldu.

3.2 Hizmet Kalitesini Etkileyen Unsurlar

Bir ağ'da, farklı trafik çeşitleri için farklı performans karakteristiklerine ihtiyaç duyulur. QoS teknikleri bazı trafiklerin performansını arttırırken diğerlerininkini azalttığından uygulamaların ihtiyaçları mutlaka göz önünde bulundurulmalıdır. Bir dosya transfer uygulaması için paketlerin gecikme süreleri ve iletim hızı önemli olmayabilirken, etkileşimli uygulamalar sabit bir gecikme süresi ve iletim hızına ihtiyaç duyabilir.

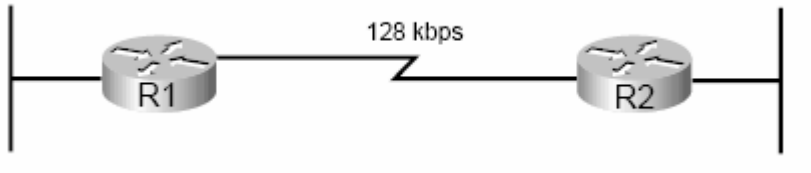
Ağ performansını arttırmak için uygulanan QoS teknikleri, bant genişliği, gecikme, jitter ve paket kayıpları üzerinde etkili olabilir. QoS uygulanmamış bir ağ'da çoklu ortam trafikleri Tablo 3.3'de verilen bazı problemlerle karşılaşabilir.

Trafik	QoS Uygulanmadığında Karşılaşılan Problemler
Ses	Sesin anlaşılmasında zorluklar
	Seste kesintiler ve patlamalar
	Paket gecikmesi nedeniyle, karşıdakinin konuşmasının ne zaman bittiğinin anlaşılamaması
	Çağrıların kesilmesi
Video	Aniden duran ve aniden hareket eden görüntüler
	Görüntü ile sesin eşzamanlı olmaması
	Görüntülerin yavaşlaması
Veri	Verilerin kullanılabilirliği bittikten sonra alınması
	Veriye ulaşmak için uzun süre beklenmesi

Tablo 3.1 QoS yapılmamış bir hattaki trafiklerin davranışı

3.2.1 Bant Genişliği

Bant genişliği, iletişim kanalından bir saniyede geçmesi beklenen bit sayısı olarak tanımlanabilir. Noktadan noktaya bağlı ağlarda bant genişliği, fiziksel hat hızına veya o portun saat hızına (clock rate) eşittir. Yani, hat hızı 128 Kbps ise bu hat üzerinden 128 Kbps miktarında trafik gönderilmesini ve hattın diğer ucunda da aynı miktarda verinin alınmasını bekleyebiliriz.



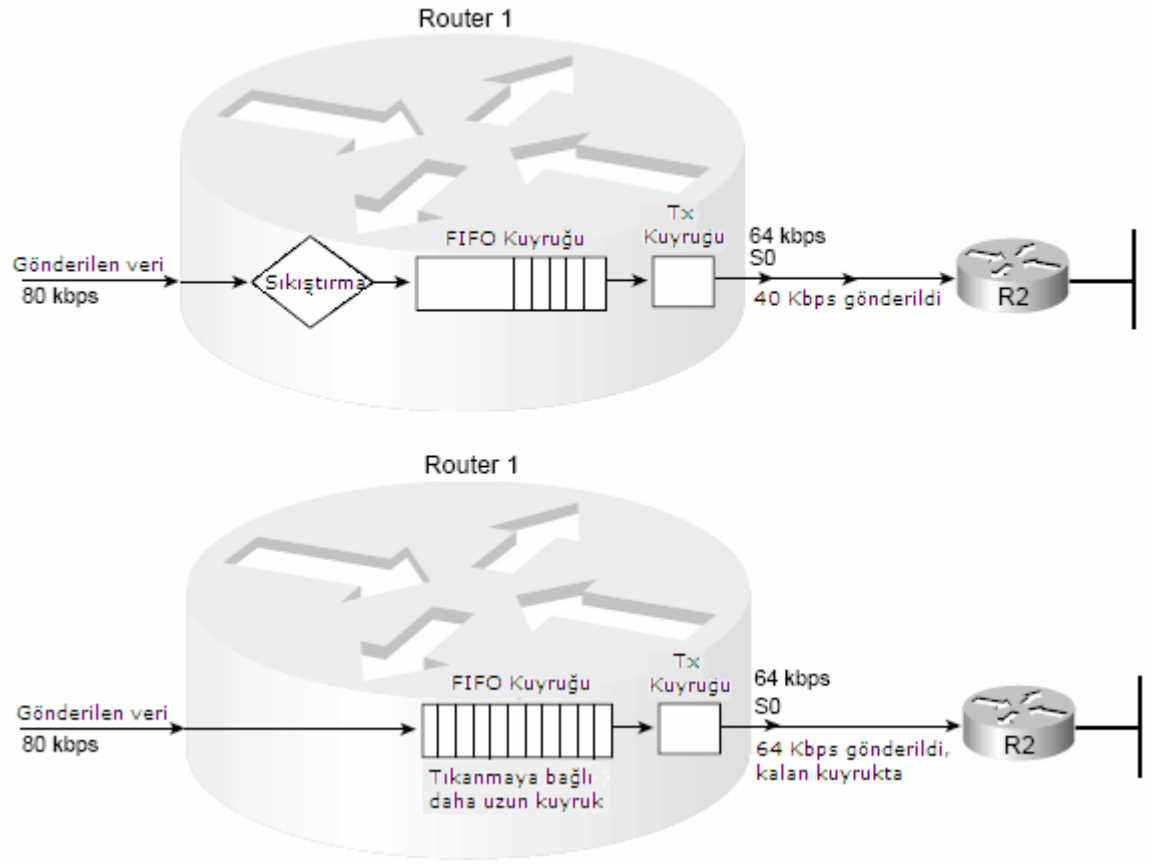
Şekil 3.3 Noktadan noktaya bağlı bir ağ

3.2.1.1 Bant Genişliğine Etki Eden QoS Teknikleri

Bant genişliği konusunda bize yardımcı olacak birkaç QoS tekniği vardır. Bu konuda en iyi QoS tekniği daha çok bant genişliğidir! Bununla birlikte, daha fazla bant genişliği her zaman sorunları çözmeye yetmez, ayrıca çokta masraflıdır.

3.2.1.1.1 Sıkıştırma

Bant genişliği konusunda verimli QoS tekniklerinden biri, iletilecek ver miktarını düşürmektir. Şekil 3.4 noktadan noktaya 64 Kbps hat üzerinden 80 Kbps'lik verinin iletiminde, 2:1 oranını kullanarak sıkıştırmanın etkisini göstermektedir. Doğal olarak sıkıştırma yapılmadığında kuyruk oluşacak ve kuyruğun sonundaki veriler kuyruk boyutuna bağlı olarak atılacaktır.



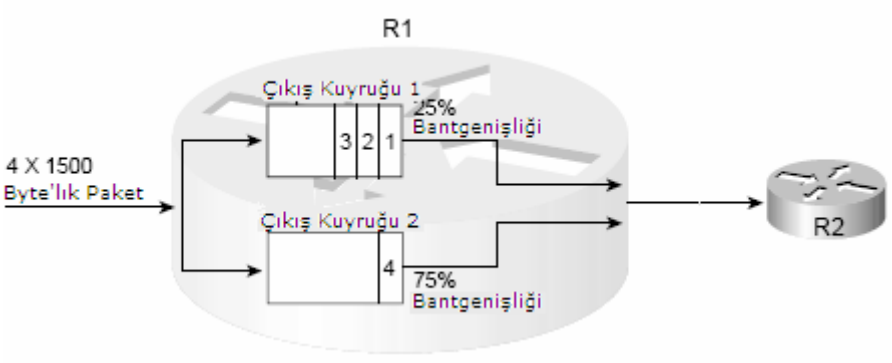
Şekil 3.4 Sıkıştırma varken ve yokken paketlerin durumu

3.2.1.1.2 Çağrı İzin Kontrolü (CAC)

Bant genişliği konusunda bir diğer QoS tekniği çağrı izin kontrolü'dür (CAC). CAC, ağın yeni bir ses veya video çağrısına izin verip vermeyeceğine karar verir. Örneğin, ağ'da eş zamanlı üç G.729 VoIP çağrısının yapılmasına izin verilebilir. Bu durumda, gelecek dördüncü çağrı ya reddedilecek ya da PSTN bağlantısı üzerine yönlendirilecektir.

3.2.1.1.3 Kuyruklama

Kuyruklama tekniği, hangi veri tipinin ne kadar bant genişliği kullanacağı konusunda etkilidir. Bu teknikte birden çok kuyruk yaratılarak gelen paketlerin belirli bir algoritma dahilinde farklı kuyruklara alınması sağlanır. Algoritmalar, belirli bir kuyruğa sabit bir bant genişliği garanti edebilir. Şekil 3.5'de her iki kuyrukta da paketler beklediğinde bant genişliğinin yüzde yirmibeşinin ve yüzde yetmişbeşinin ayrı ayrı tahsis edildiği görülmektedir. Eğer bu kuyruklardan bir tanesi boşalırsa, diğer kuyruk kendine ayrılmış olan bant genişliğinden daha fazlasını kullanabilir.



Şekil 3.5 Kuyruklama tekniği

QoS tekniklerinin bant genişliğine etkileri Tablo 3.2'de verilmiştir.

QoS Tekniği	Bant genişliğine Etkisi
Sıkıştırma	İletilecek veri miktarını düşürmek için başlıkları ve paket yükünü sıkıştırır
CAC	Belirlenen sayıdan fazla çağrı kurulmasını engelleyerek çağrılar için kullanılacak bant genişliğini sınırlar
Kuyruklama	Belli tipteki paketler için istenilen minimum bantgenişliğini garanti eder

Tablo 3.2 QoS tekniklerinin Bant genişliğine etkisi

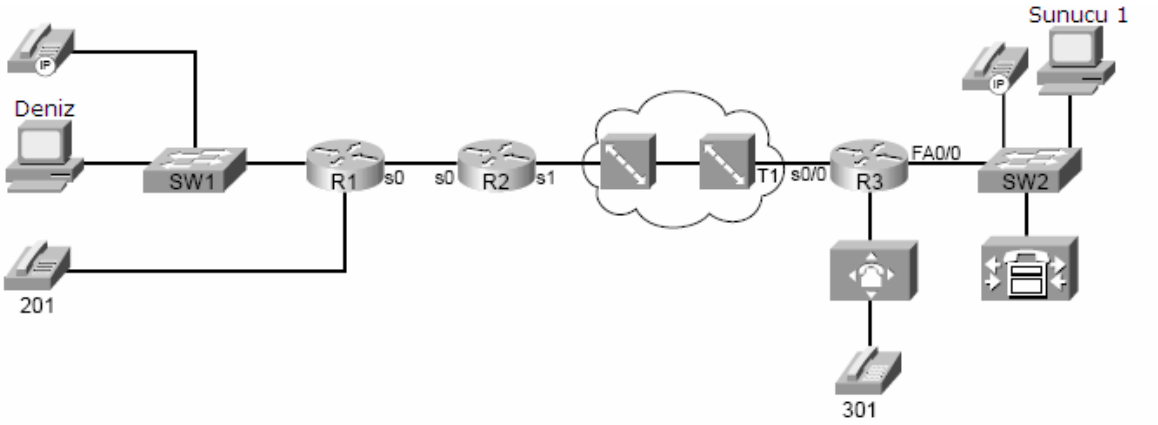
3.2.2 Gecikme

Ağ'daki tüm paketler gönderildikleri yerden gidecekleri yere ulaşıncaya kadar farklı gecikmelere maruz kalırlar. Aslında QoS mekanizmalarının arkasındaki pek çok kavram da bir şekilde gecikme ile ilgilidir.

Ses iletiminde ne kadarlık bir gecikmenin kabul edilebileceği Tablo 3.3'de verilmiştir. Bu değerler ITU tarafından G.114'te belirtilmiştir. Bazı durumlarda yüksek kalitede ses istenirken, özellikle mali tasarruf yapılabilecek durumlarda daha düşük kalitede ses iletimi de kabul edilebilmektedir. Hatta konuşmanın ücretsiz olması durumunda çok kötü kalitede görüşme yapmak bile göze alınabilir durumdadır.

Tek Yönlü Gecikme (ms)	Açıklama
0 -150	ITU G.114'ün ses gecikmesi için önerdiği kabul edilebilir aralık
0 - 200	Cisco'nun ses gecikmesi için önerdiği kabul edilebilir aralık
150 - 400	ITU G.114'ün düşük kaliteli ses iletimi için önerdiği kabul edilebilir aralık
400+	ITU G.114'ün hiçbir durum için önermediği kabul edilemez aralık

Tablo 3.3 İzin verilen tek yönlü gecikme değerleri



Şekil 3.6 Gecikme kavramında kullanılacak örnek Ağ

Şekil 3.6'da gecikme kavramını tartışırken kullanacağımız örnek bir ağ yapısı mevcuttur. Burada aklımıza hemen şu soru gelmektedir. "Gecikme bu ağ'ın hangi noktasında oluyor?" Aslında tüm noktalarda gecikme meydana gelmektedir. Bazı noktalardaki gecikme pratikte ihmal edilebilecek kadar küçükken, bazılarında oldukça önemli miktardadır.

3.2.2.1 Dizilim Gecikmesi

Bir tren istasyonunda durduğunuzu düşünün. Bir tren hiç durmadan önünüzden geçsin. Trenin ilk vagonunun istasyona gelmesi ile son vagonunun istasyondan ayrılması arasında belirli bir süre geçecektir. Tren çok uzunsa, trenin tamamen önünüzden geçip gitmesi daha uzun bir zaman alacaktır. Eğer tren yavaş ilerliyorsa tüm vagonların önünüzden geçip gitmesi çok daha uzun sürecektir. Dizilim gecikmesi de trenin ilk vagonu ile son vagonunun istasyondan geçiş süreleri arasındaki gecikmeye benzemektedir.

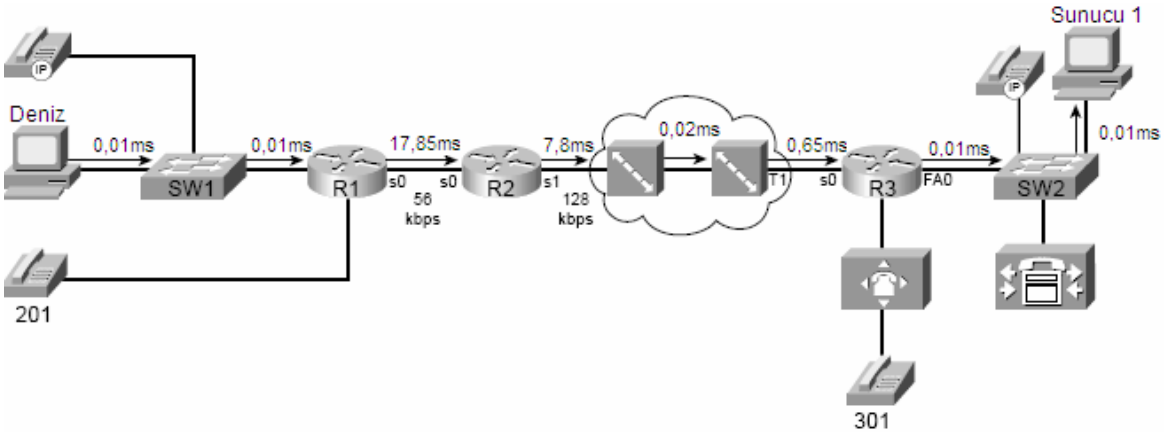
Dizilim gecikmesi, bir paketteki bitlerin kodlanarak fiziksel hatta verilmesi için gereken süre olarak tanımlanır. Eğer hat hızlıysa, bitler hatta çok daha hızlı verilebilir. Eğer hat yavaşsa, bitlerin hatta verilmesi daha uzun sürecektir. Benzer şekilde paket kısaysa uzun paketlerden daha kısa sürede hatta verilebilir. Bir paket için dizilim gecikmesi

$$\text{Dizilim Gecikmesi} = \frac{\text{Gönderilen Bit Sayısı}}{\text{Hat Hızı}} \qquad (3.1)$$

eşitliği ile belirlenir.

Şekil 3.7'de verilen senaryoda, Deniz'in bilgisayarının Sunucu 1'e 125-byte'lık paket gönderdiğini düşünelim. Deniz, paketi önce Fast Ethernet üzerinden Switch'e gönderecektir. 125-byte'ın 1000 bit'e eşit olduğunu düşünürsek Fast Ethernet hızında bu verinin hatta

verilme süresi 1.000bit/100.000.000bps veya 0.01 ms olarak bulunabilir. Bir diğer 0.01 ms'de switch, çerçeveyi R1'e göndermek için hatta verdiğinde geçecektir. R1'in aynı paketleri R2'ye göndermek üzere 56Kbps hızındaki hatta vermesi sırasında 1.000bit/56.000bps veya 17,85 ms'lik dizilim gecikmesi yaşanacaktır. Buradan da anlaşılacağı gibi, yüksek hızlı hatlardaki dizilim gecikmesi önemsiz olurken düşük hızlı hatlarda önemli olmaktadır. Şekil 3.7'de Deniz'in Sunucu1'e gönderdiği paketin maruz kaldığı dizilim gecikmeleri verilmiştir.



Şekil 3.7 Dizilim gecikmeleri

3.2.2.2 Yayılım Gecikmesi

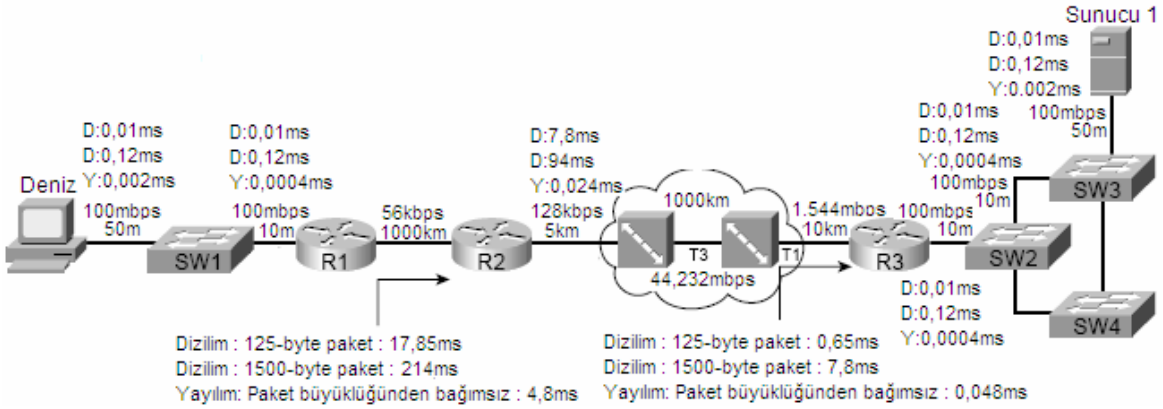
Yine bir treni izlediğinizi düşünün, ancak bu sefer trenlerin üzerindeki bir helikopterden. Trenin bir istasyondan ayrıldığını ve diğer istasyona vardığını görüyorsunuz. Öndeki vagonun ilk istasyondan ayrılıp diğer istasyona varması belirli bir zaman alacaktır. Yayılım gecikmesi de trenin lokomotifinin ilk istasyondan ayrılıp diğer istasyona varması arasındaki gecikmeye benzemektedir.

Yayılım gecikmesi, bir bitin hattın bir ucundan diğer ucuna gönderilmesi sırasında geçen süre olarak tanımlanır. Elektriksel veya optik işaretler hatta verildiği zaman diğer uca aynı anda ulaşamaz, belirli bir gecikme olur. Elektriksel ve optik hatlardaki enerjinin hızı ışık hızına yaklaşmaktadır, enerjinin elektriksel kablolar üzerinden ışık hızının yüzde yetmişi oranında bir hızla ($2.1x10^8$ metre/saniye) ilerlediği kabul edilmiştir. Yayılım gecikmesini yalnızca hattın uzunluğu etkileyebilir ve aşağıdaki formül ile hesaplanır.

$$\text{Yayılım Gecikmesi} = \frac{\text{Hattın Uzunluğu (metre)}}{2,1x10^8 \text{ metre/saniye}} \qquad (3.2)$$

eşitliği ile hesaplanır.

Şekil 3.8'de verilen senaryoda R1 ve R2 arasındaki noktadan noktaya hattın uzunluğunun 1000 km olduğunu düşünelim. Bu durumda yayılım gecikmesi 1.000.000m/2,1x10^{8}m/s veya 4,8ms olacaktır. Şekilden de görüleceği gibi paket boyutunun artmasıyla birlikte dizilim gecikmesi artarken, yayılım gecikmesi sabit kalmaktadır. Burada, yayılım gecikmesine hat hızının veya saat hızının etki ettiği şeklindeki inanışın da doğru olmadığı görülmektedir.

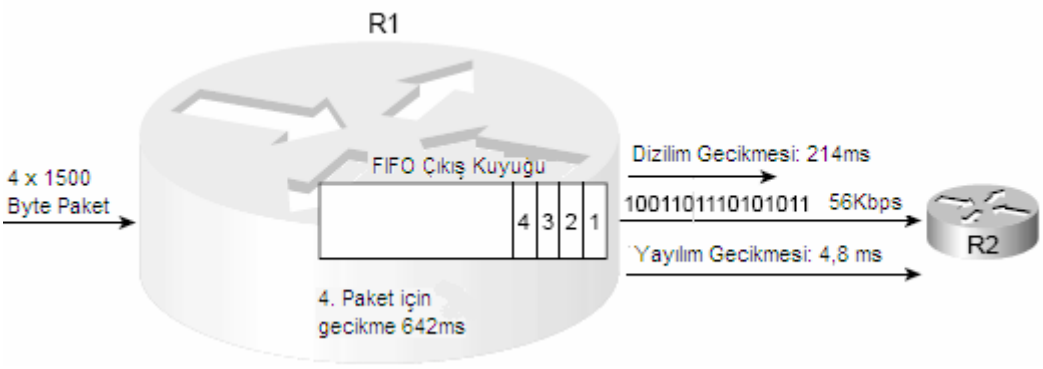


Şekil 3.8 Yayılım gecikmeleri

3.2.2.3 Kuyruklama Gecikmesi

Paketler, önlerinde beklemeleri gereken başka paketler olduğunda kuyruklama gecikmesine maruz kalırlar. Kuyruklama gecikmesi, cihaz içerisindeki kuyruklarda harcanan zamanı belirtir. Bu zaman yüzlerce milisaniye ya da daha fazla olabilir.

Deniz'in, Şekil 3.9'da gösterildiği gibi Server1'e 1500-byte'lık dört paket gönderdiğini düşünelim. Her bir paketin dizilim gecikmesi (1500*8/56.000) 214ms olacağından diğer paketlerin ya hafızada bekletilmesi ya da atılması gereklidir. Bu nedenle router kuyruktaki paketleri tutabilmek için hafıza birimi kullanır. Kuyruklamanın en basit şekli, ilk giren ilk çıkar (FIFO) mantığına dayalı tek bir kuyruk kullanmaktır



Şekil 3.9 Kuyruklama gecikmesi

Şekil 3.9'da verilen senaryoda paketlerin tamamı seri hat üzerinden 856ms sonra gönderilmiş olacaktır. Burada dikkat çekici olan, hattın meşgul olmaması durumunda bile paketlerin gecikmeye maruz kalmasıdır. Gönderilen ilk paket hiç beklemeden hatta verilmeye başlanmasına rağmen, ikinci paket birinciyi 214ms, üçüncü paket ilk iki paketi 428ms ve son pakette önündeki üç paketi 642ms beklemek zorunda kalmıştır.

Bu senaryoyu bir de 100 tane 1500-byte'lık paket gönderildiğini düşünerek ele alalım. R1, 100 paketi de kuyruklayabilecek kadar hafızaya sahip olsun. Bu durumda yüzüncü paket 99*214ms ya da kabaca 21 saniye beklemek zorunda kalacaktır. Eğer Deniz TCP kullanıyorsa, TCP muhtemelen zaman-aşımı hatası alacak ve paketi tekrar göndererek daha fazla tıkanıklığa ve gecikmeye neden olacaktır. Bu gecikmelerin yalnızca bir router'ın çıkışında meydana geldiği dikkate alındığında, kuyruklamanın paketlerin atılmasını engellerken, uzun kuyrukların aşırı gecikmeye sebep olduğu görülebilmektedir.

3.2.2.4 Yönlendirme Gecikmesi

Yönlendirme gecikmesi, paketin router'ın girişinde incelenmesi ve istenilen çıkış kuyruğuna anahtarlanması arasında geçen zamanı belirtmektedir. Çoğunlukla dikkate alınmayacak kadar küçüktür.

3.2.2.5 Trafik Uyarlama Gecikmesi

Trafik uyarlama, kuyrukları yavaşlatarak ek gecikmelere neden olur. Peki bir router mecbur olmadığı halde neden paketleri daha yavaş göndermeye başlar? Aslında trafik uyarlama, paket gönderim hızını servis sağlayıcının belirttiği hıza getirerek paketlerin servis sağlayıcının ağında imha edilmesine karşı önlem alır. Bu nedenle paketlerin hızlı ama kayıplı iletilmesindense, yavaş ama tam olarak iletilmesi tercih edilir. Trafik uyarlama seçimli bir özelliktir.

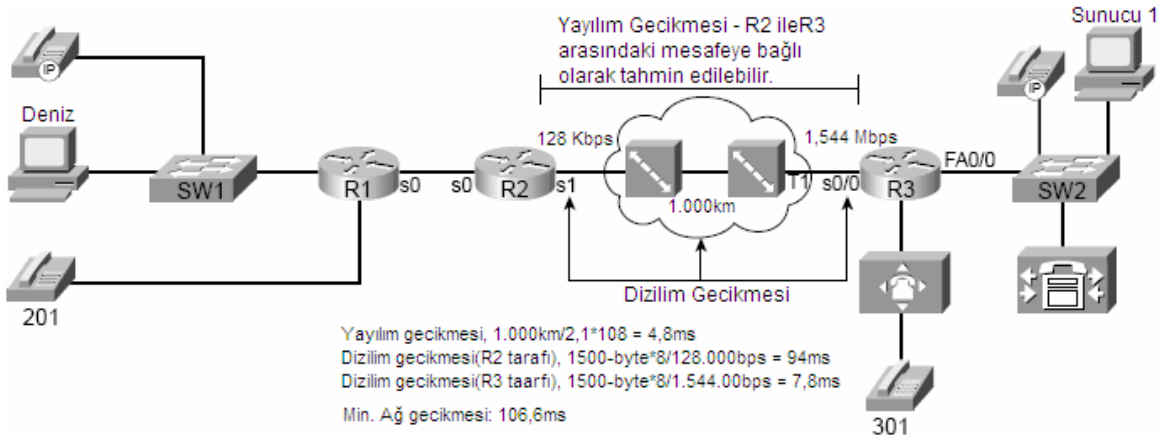
3.2.2.6 Ağ Gecikmesi

Pek çok insan, detaylarını bilmediğinden internet için büyük bir bulut çizmektedir. Aslında buluntun dışında meydana gelen tüm gecikmeler bulutun içinde de geçerlidir. Bu nedenle, bulutun dışında router ve switch'lere sahip olan bir mühendis bulutun içindeki yapıyı bilmediğinden tam olarak QoS kontrolü yapamayabilir.

Ağ gecikmesi, servis sağlayıcının ağ'ında yaşanan gecikme olarak tanımlanır. Peki bulutun içinde ne kadarlık bir gecikme yaşanır? Aslında bu gecikme oluşan kuyruklara, tıkanmalara,

hatların durumuna bağlı değişebilen ve tahin edilmesi oldukça zor bir değerdir. Bununla birlikte servis sağlayıcı size maksimum gecikme süresi ile ilgili bir değer verip, bunu sağlamayı garanti edebilir.

Yayılım gecikmesi ve dizilim gecikmesi, gerçek değerine oldukça yakın bir doğrulukla tahmin edilebilir. Yayılım gecikmesi için R2 ve R3 arasında kaç tane router veya switch olduğu önemli değildir, çünkü R2 ve R3 arasındaki toplam yayılım gecikme değeri en az noktadan noktaya bağlantıdaki kadar olacaktır. Minimum dizilim gecikmesi ise router'lara bağlı olan hatların hızı yardımıyla hesaplanabilir. Sonuçta minimum ağ gecikmesi, yayılım gecikmesi ve router'lara bağlı iki hattaki dizilim gecikmelerinin toplamı olacaktır. Şekil 3.10'da minimum ağ gecikmesi 106,6ms olarak bulunmuştur.



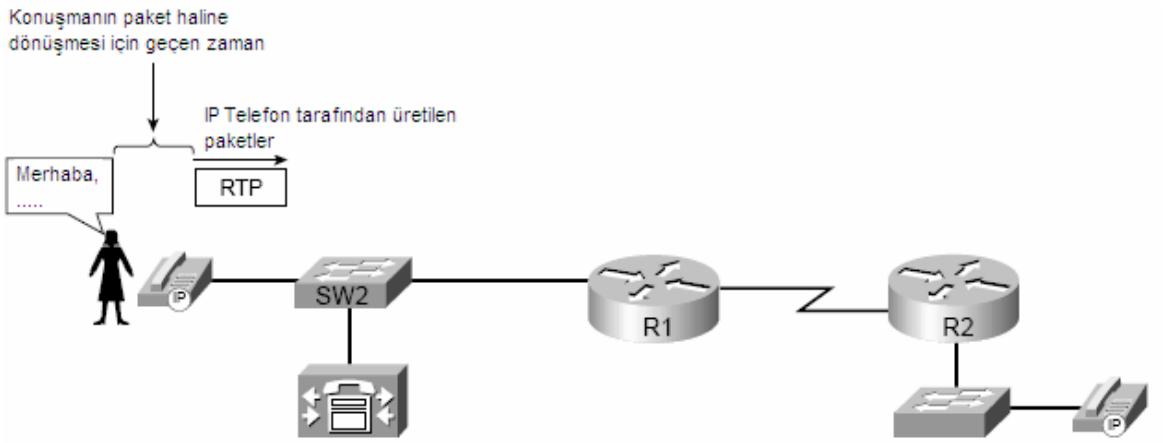
Şekil 3.10 Ağ gecikmesi

3.2.2.7 Codec ve Paketleme Gecikmesi

IP telefon'da paketler gönderilmeden önce hangi işlemlerin yapılması gerektiğine bir bakalım. İlk olarak kullanıcı dijitleri çevirecek ve çağrı kurulacaktır. Çağrı kurulduktan sonra IP telefon içlerinde 20ms'lik (varsayılan) ses yükleri bulunduran RTP paketlerini göndermeye başlar. Peki, kullanıcının konuşmaya başlamasından ses yükünü taşıyan ilk paketin gönderilmesine kadar geçen süre nedir?

Bir an için ses dalgalarını düşünelim. İki kişi bir odada oturup konuşmaya başladığında karşıdaki dinleyicinin konuşulanı duymaya başlaması için geçen süre çok küçüktür. Çünkü konuşulanlar saatte yaklaşık 1000km'lik ses hızıyla hareket edecektir.

IP üzerinden görüşmede cihaz, sesi önce analog sonra sayısal işaretlere çevirecek sonrasında da sayısal hale getirilmiş ses yükünü paketlerin içine koyacaktır. Dolayısıyla konuşulanların paketlere çevrilerek gönderilmesi belirli bir zaman alacak ve gecikmeye neden olacaktır. Konuşmanın paketler haline dönüşmesi sırasında ses Şekil 3.11'de gösterildiği gibi, paketleme ve codec gecikmelerine maruz kalacaktır.



Şekil 3.11 Codec ve paketleme gecikmesi

IP telefon ya da ses gateway'i 20ms'lik ses yükünü paketin içine koymadan önce 20ms'lik ses toplamalıdır, yani 20ms'lik ses yükünü taşıyan paketin oluşturulması için öncelikle bu 20ms'lik sesin elde edilmesi gerekir. Buna paketleme gecikmesi denir.

IP telefon ya da ses gateway'i kullanımda olan codec'e göre gelen analog sinyali kodlayarak onu sayısal eşdeğerine dönüştürmelidir. Örneğin G.729 codec'i bir seferde 10ms'lik analog sesi işleyebilir.

Codec algoritmaları "look-ahead" denen bir özelliği de kullanabilir. Ancak "look-ahead" yalnızca codec öngörülü ise, bir başka deyişle insan sesinin anlık olarak çok farklı seslere geçemeyeceği gerçeğini kullanan bir yapıda ise gerçekleşir. Algoritma, konuşmayı inceleyerek ve sesin birkaç ms sonra önemli miktarda değişmeyeceğini bilerek daha az bit ile kodlama yapabilir. Bununla birlikte, öngörülü algoritmalar tipik olarak codec'in işleyeceği ses sinyalinin yanı sıra, bu sesten sonra gelecek birkaç ms'lik ses parçalarına da ihtiyaç duyarlar. Tablo 3.4'te gösterildiği gibi, G.729 codec'i işleyeceği 10ms'lik ses örneğine ek olarak bu sesten sonra gelen 5ms'lik ses parçalarını da kullanır. Codec ve Paketleme işlemleri birbiri ile çakışan işlemler olduğundan süreç toplamı 20ms'lik codec gecikmesi ile 20ms'lik paketleme gecikmesinin toplamı olarak hesaplanmaz. Tablo 3.4'te gösterildiği gibi bu işlemler için toplam süreç 30ms olur.

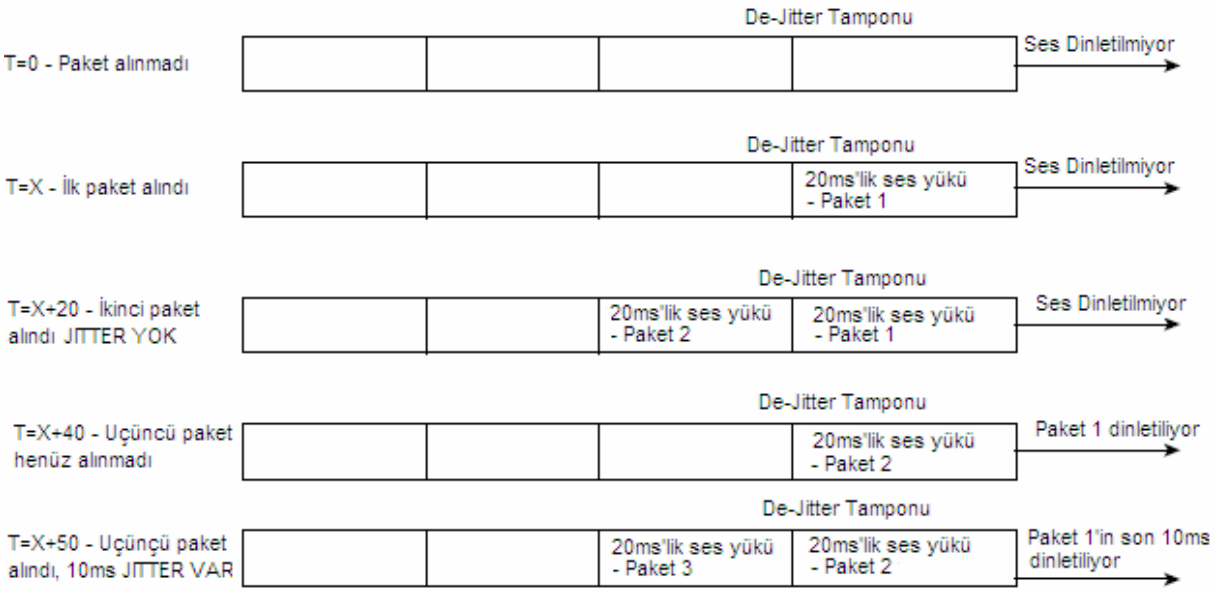
Zaman	Süreç	Codec Gecikmesi	Paketleme Gecikmesi
t=0	A/D dönüşüm ve kodlama için ses örnekleri toplanmaya başlar		Paketleme gecikmesi başlar
t=10	G.729 ile kodlanacak 10ms'lik örneğin toplanması tamamlanır	10ms sonunda Codec gecikmesi başlar	Paketleme gecikmesi devam eder
t=15	ikinci 10ms'lik örneğin ilk 5ms'si toplanır	G.729, algoritmanın ilk 10ms'lik örneği kodlamak için ihtiyaç duyduğu 5ms'lik look-ahead'a sahip olur	Paketleme gecikmesi devam eder
t=20	ikinci 10ms'lik örneğin toplanması tamamlanır	ilk 10ms'lik örnek için codec gecikmesi bitmiştir. İkinci 10ms'lik örnek hafızadadır. İkinci örnek için codec gecikmesi başlar	20ms sonunda paketleme gecikmesi biter, çünkü 20ms'lik ses toplanmıştır.
t=25	üçüncü 10ms'lik örneğin ilk 5ms'si toplanır	G.729, algoritmanın ikinci 10ms'lik örneği kodlamak için ihtiyaç duyduğu 5ms'lik look-ahead'a sahip olur	
t=30	üçüncü 10ms'lik örneğin toplanması tamamlanır	20ms'lik toplam codec gecikmesi; RTP ve ses yükü gönderilmeye hazır.	
ilk paket için toplam gecikme		20ms	20ms

Tablo 3.4 Codec ve Paketleme gecikmesi zaman çizelgesi

3.2.2.8 De-Jitter Tampon Gecikmesi

Veri ağlarında jitter her zaman meydana gelir. Bunu kontrol edebilir ve jitter'a duyarlı trafikler için minimize edilebilir, ancak tamamen ortadan kaldırılamaz.

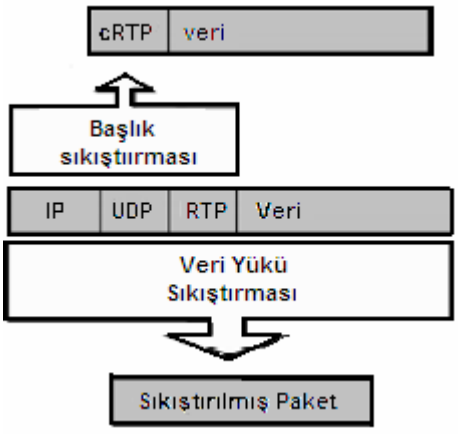
De-jitter tamponu, Şekil 3.12'de verildiği gibi alıcı tarafa ses paketlerini dinletmeye başlamadan önce belli bir süre ses paketlerini tamponlayarak gecikmeye sebep olur. Bu süre genellikle 40ms'dir. Bunu yapmaktaki amaç, alıcının sesi dinlemeye başlamasından sonra, paketlerin geç gelmesi durumunda bile sesi sabit hızda dinletebilmektir. Bu sistem arabalardaki CD çalarlarda da kullanılmaktadır. Araba kasise girdiğinde CD çalıcı bir süreliğine okuma işlemini gerçekleştiremez, bu süre içinde tamponladığı veriyi dinleterek şarkının kesilmesini önler.



Şekil 3.12 De-Jitter Tampon gecikmesi

3.2.2.9 Sıkıştırma Gecikmesi

Sıkıştırma, orijinal paketleri matematiksel algoritmalar çerçevesinde daha düşük boyutlu hale getirmek için kullanılan kodlama tekniğidir ve hesaplama yükü getirdiğinden paketlerin sıkıştırılması belirli bir zaman alır. Bu süre, hesaplamayı yapacak olan işlemcinin hızıyla doğrudan ilişkilidir. Genellikle veri yükü sıkıştırması ve başlık sıkıştırması olmak üzere ikiye ayrılır. Veri yükü sıkıştırması başlıkları ve kullanıcı verisini sıkıştırırken, başlık sıkıştırması sadece verilerin önündeki başlıkları sıkıştırır.



Şekil 3.13 Başlık ve Veri yükü sıkıştırması

Tablo 3.5'de gecikme tipleri ile gecikme süreleri arasındaki ilişki verilmiştir. Bir ağ'da gecikmeyi düşürebilmek için değişken süreli gecikme yaratan mekanizmaları ele almak gerekir.

Gecikme Tipi	Gecikme Süresi
Dizilim (Serialization)	Sabit
Yayılım (Propagation)	Sabit
Kuyruklama (Queuing)	Değişken
Yönlendirme/İşleme (Fowarding/Processing)	Değişken
Trafik Uyarlama (Traffic Shaping)	Değişken
Ağ (Network)	Değişken
Kodek (Codec)	Sabit
Sıkıştırma (Compression)	Değişken

Tablo 3.5 Gecikme tipleri ve gecikme süreleri

3.2.2.10 Gecikmeye Etki Eden QoS Teknikleri

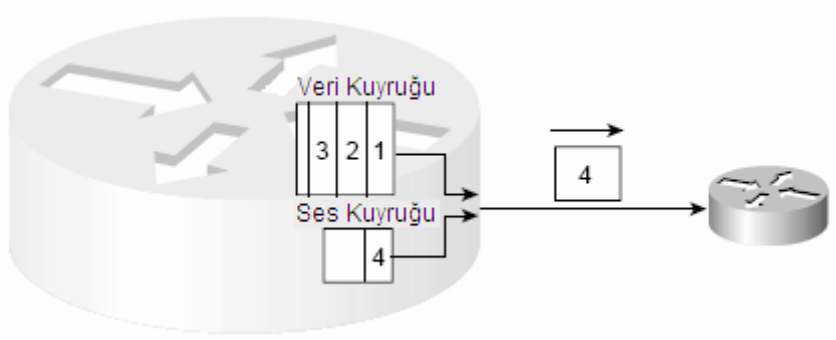
Gecikme konusunda bize yardımcı olacak birkaç QoS tekniği vardır. Bu konuda da en iyi QoS tekniği her zamanki gibi daha çok bant genişliğidir! Daha çok bant genişliği, yetersiz bant genişliği ve gecikme nedeniyle oluşan problemlerin çözümünde yardımcı olur, dizilim gecikmelerini azaltır. Paketler hatta daha hızlı verildiğinden kuyruklama gecikmesi de düşer.

Ancak ne yazık ki, daha fazla bant genişliği her zaman sorunları çözmeye yetmez. Hatta birleştirilmiş ağlarda (ses, görüntü ve veriden oluşan ağlar) QoS teknikleriyle çözülebilecek pek çok problemi maskeleyebilir.

3.2.2.10.1 Kuyruklama

Gecikmeyi azaltmak için ilk giren ilk çıkar (FIFO) mantığıyla çalışan tek bir kuyruk yerine, birçok farklı kuyruk yaratarak paketleri bu farklı kuyruklara yerleştirebiliriz. Bunun sonucunda bazı paketler router'ı çok daha çabuk terk ederken bazıları da daha uzun süre beklemek zorunda kalacaktır. Kuyruklama tüm paketler için gecikme konusunda iyileştirme sağlamazken, zamana duyarlı paketlerin önceliklendirilmesini sağlayacaktır.

Her kuyruklama yöntemi belli sayıda kuyruk tanımlayarak, hangi kuyruğa hangi paketin gireceğini ve kuyrukların öncelik sıralarını belirlemektedir. Şekil 3.14'de zamana duyarlı ses paketinin veri paketlerine karşı öncelikli olduğu görülebilmektedir.

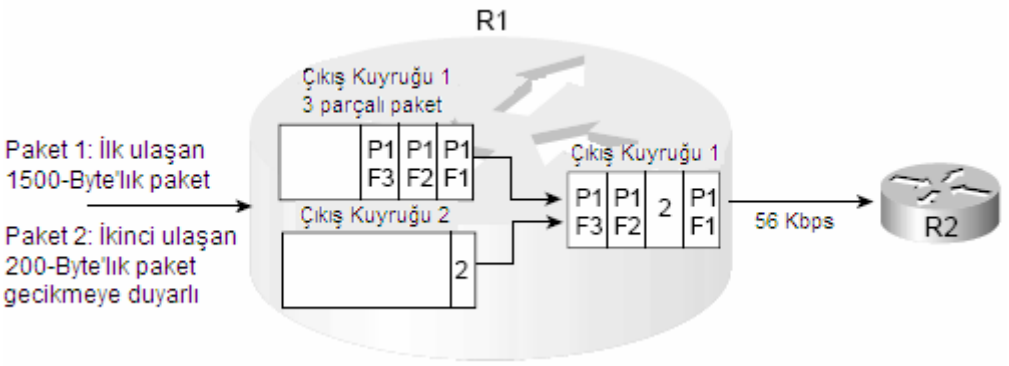


Şekil 3.14 Kuyruklama'nın gecikmeye etkisi

3.2.2.10.2 Parçalama ve Yerleştirme

Bir paketin hatta verilmesi için gerekli olan süre hat hızının ve paket boyutunun bir fonksiyonudur. Router bir paketin ilk bitini göndermeye başladıktan sonra, paketin tüm bitleri gönderilinceye kadar bu işlemi sürdürür. Bu nedenle, büyük bir paket gönderilmeye başlandıktan sonra gecikmeye duyarlı bir paket gelirse, bu paket uzun süre beklemek zorunda kalacaktır.

Şekil 3.15'de görüldüğü gibi R1'e 1500-byte'lık normal veri ve 200-byte'lık gecikmeye duyarlı veri olmak üzere iki paket gelsin. İkinci paket birinciden önce geldiğinden 56 Kbps hatta ilk bu paket verilmeye başlanacak ve dolayısıyla gecikmeye duyarlı ikinci paket birinci paketin 214ms'lik dizilim gecikmesini beklemek zorunda kalacaktır.



Şekil 3.15 Parçalama ve Yerleştirme

Bunu önlemek için kullanılan Hat Parçalama ve Yerleştirme (LFI) sayesinde, birinci paket parçalara bölünecek ve parça parça iletilecektir. İlk parçanın hatta verilmesinden sonra gecikmeye duyarlı ikinci paket araya sokulacak ve daha kısa süreli gecikmeye maruz kalarak iletilmesi sağlanacaktır. Bu sayede ikinci paket birinci paketin tamamının iletilmesini beklemek zorunda kalmadan, sadece birinci paketin ilk parçasını bekleyecektir.

3.2.2.10.3 Sıkıştırma

Sıkıştırma sayesinde paket ya da paket başlıkları daha az bitle temsil edilen veriler haline dönüştürülür. 1500-Byte'lık bir paket yarı yarıya sıkıştırıldığında yani 750-Byte ile ifade edildiğinde, bu verinin dizilim gecikmesi de yarıya düşer.

Sıkıştırma, dizilim gecikmesini azaltmakla birlikte paketlerin sıkıştırılması ve açılması için belirli bir işlem yükü gerektirdiğinden, toplam gecikme artabilir. Bu nedenle gecikme konusunda sıkıştırma tekniği kullanılırken iyi hesap yapılmalıdır. QoS tekniklerinin gecikmeye etkileri Tablo 3.6'da verilmiştir.

QoS Tekniği	Gecikmeye ve Jitter'a Etkisi
Kuyruklama (Queuing)	Paketleri tasnif etmeyi, böylece gecikmeye duyarlı paketlerin diğer paketlerden önce router'ı terk etmesini mümkün kılar
Parçalama ve Yerleştirme (LFI)	Routerlar bir paketi göndermeye başladıktan sonra bu süreci yarıda kesmediklerinden, LFI paketlerin gönderim işlemi başlamadan onları daha küçük parçalara böler. Böylece gecikmeye duyarlı paketler parçaların arasına girerek paketin tamamı gönderilinceye kadar kuyrukta beklemezler.
Sıkıştırma (Compression)	Veri yükü veya başlıkların sıkıştırılmasıyla birlikte iletilecek bit miktarı azalır. Bu sayede dizilim gecikmesini de düşer. Bant genişliği ihtiyacının azalmasıyla kuyruklar küçülür ve kuyruk gecikmesi azalır. Bununla birlikte işlem yükü artar ve sıkıştırma gecikmesi ortaya çıkar.

Tablo 3.6 QoS tekniklerinin gecikmeye ve Jitter'a etkisi

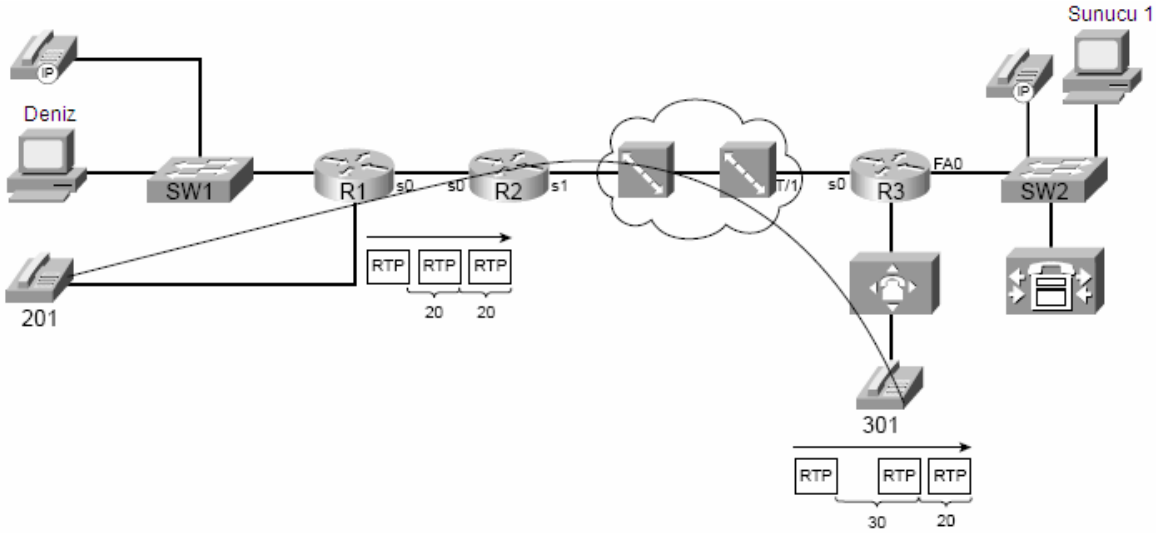
3.2.3 Jitter

Jitter, sabit zaman aralıklarıyla ardışık olarak gönderilen paketlerin alıcı tarafta değişken zaman aralıklarıyla elde edilmesidir. Paket anahtarlamalı ağlarda jitter her zaman oluşur. Genellikle uygulamalar bu bilginin ışığında geliştirilir ve belirli miktarda jitter'dan etkilenmezler. Ancak, ses gibi sabit zaman aralıklarıyla iletilmesi gereken veriler jitter'a çok duyarlıdır. Şekil 3.16'te tipik bir jitter senaryosu verilmiştir.

3.2.3.1 Jitter'e Etki Eden QoS Teknikleri

Ağ'daki paketleri etkileyen diğer problemlerin giderilmesinde ilk başvurulan yöntem olan daha çok bant genişliği sağlama, Jitter içinde bir noktaya kadar kullanılabilir. Jitter, aynı zamanda gecikmenin değişintisi olarak ele alınabileceğinden, gecikmenin azaltılması için kullanılan tüm teknikler aslında Jitter içinde kullanılabilmektedir. Örneğin ortalama 100ms ile 200ms arasında gecikme yaşanan bir ağ'da Jitter en fazla 100ms olabilir. Eğer gecikme 50ms

ile 100ms arasına düşürülebilirse bu sefer Jitter en fazla 50ms olacaktır. Kısacası gecikmenin azaltılması Jitter miktarının da düşürülmesini sağlamaktadır. Gecikme ve dolayısıyla Jitter'e etki eden QoS teknikleri Tablo 3.6'da verilmiştir.

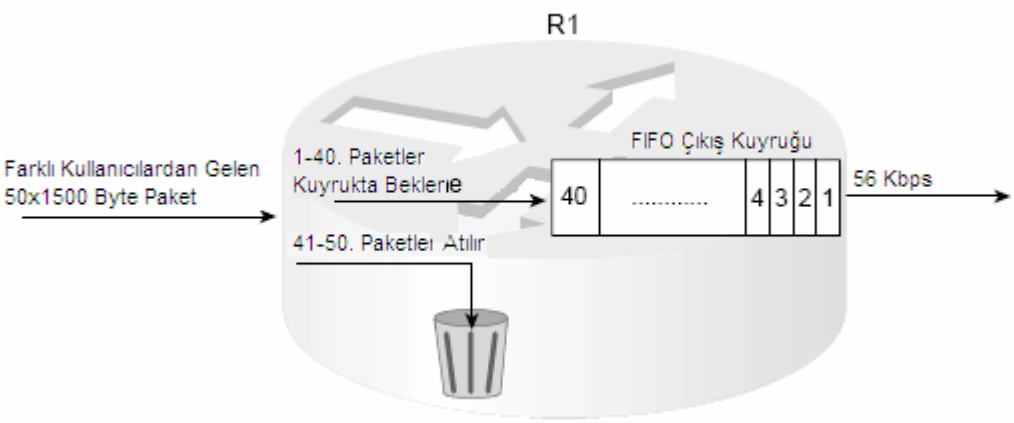


Şekil 3.16 Jitter

3.2.4 Paket Kaybı

Günümüzde bit hataları nedeniyle yaşanan paket kayıpları oldukça azalmış ve oranı miyarda bir'den daha düşük seviyelere inmiştir. Yaşanan paket kayıplarının çok büyük bir kısmı tamponların ve kuyrukların dolması nedeniyle meydana gelmektedir.

Şekil 3.17'de R1'in 40 paketlik kuyruğuna 1500-byte'tan oluşan 50 ardışık paket gönderildiği görülmektedir. Bu senaryoda, kuyruk dolunca kalan 10 paket atılacak ve paket kayıpları meydana gelecektir. Aslında pratikte router'lar 50 paketin hepsini almadan da gönderim yapmaya başlayabilir, ancak bu durumda bile bir miktar paket kaybı yaşanması kaçınılmazdır.



Şekil 3.17 Paket kaybı

Bazı trafikler, diğerlerine göre paket kayıplarından daha az etkilenebilirler. Mesela, insan kulağı 10ms'lik ses kaybını dahi fark edebilmesine rağmen, böyle bir kayıp yaşandığında dinleyici genellikle bu konuşmayı anlayabilir. Cisco router'larda kullanılan DSP'ler G.729 codec kullanılırken 30ms'ye kadar kaybolan ses paketlerinin içeriğini tahmin edebilir. Varsayılan değer olarak her ses paketi 20ms'lik ses taşıdığından, ardışık iki ses paketinin kaybolması durumunda DSP sesi tekrar oluşturamayacak ve alıcı taraf bu süre boyunca sessizlik duyacaktır.

3.2.4.1 Paket Kaybına Etki Eden QoS Teknikleri

Bant genişliğinin arttırılması, paketlerin daha hızlı iletilmesini ve kuyrukların daha zor dolmasını sağlayarak paket kayıplarını azaltmada etkili olur. Ancak bu etki ses, görüntü ve verilerin birlikte bulunduğu ağlarda yine de bazı kalite problemlerinin oluşmasına engel olamaz.

3.2.4.1.1 Rasgele Erken Tespit (RED)

TCP, alıcı taraftan onay gelmeden önce ne kadar verinin gönderilebileceğini tanımlayan pencereleme protokolünü kullanır. Her TCP bağlantısı için kullanılan ve bağlantıya özgü olan TCP pencereleri pek çok unsura bağlı olarak artıp azalabilir. RED'in çalışma mantığı, kuyruk dolmadan önce TCP bağlantılarının pencere boyutunun küçültülmesi ve iletilecek toplam veri miktarının azaltılarak kuyruğun taşmasının engellenmesidir. RED, kuyruk çok dolu olmadığı sürece devreye girmez ve iletilen toplam veri miktarının azaltılması yönünde herhangi bir etkide bulunmaz. RED, kuyruğun sonunu kontrol eden bir mekanizma olarak düşünülebilir.

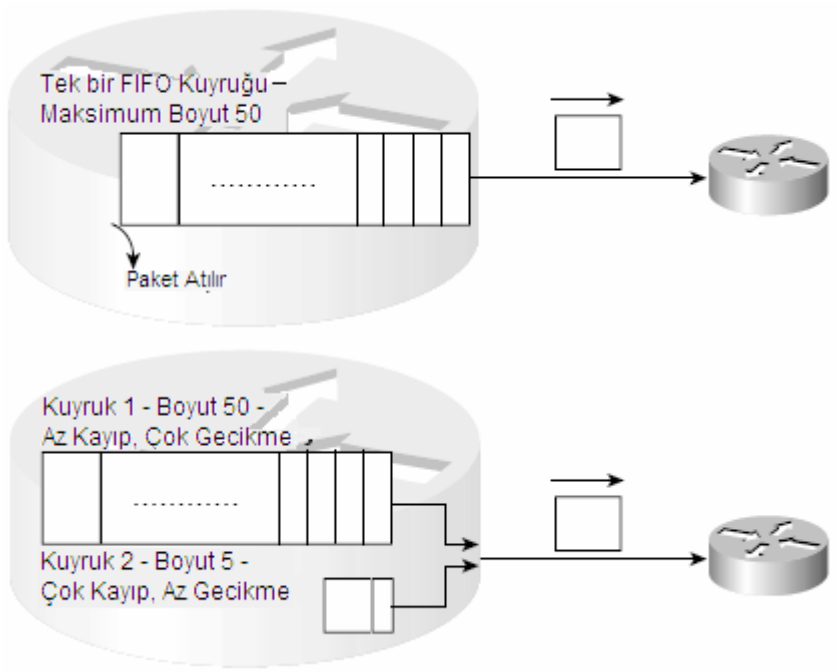
Trafiğin büyük kısmının TCP tabanlı olması ve TCP'nin gönderilen bir önceki paket kaybolduğunda iletim hızını azaltması dolayısıyla, RED kuyruklar dolmadan bazı TCP paketlerini rasgele atarak kuyruğa giren toplam paket sayısını azaltır. Paketler kaybolduğunda, TCP pencere boyutu önceki pencere boyutunun yüzde ellisine kadar düşer. Böylece RED, sadece birkaç TCP bağlantısını yavaşlatarak kuyruğun dolmasını ve tüm bağlantıların paket kaybı yaşanmasını engellemiş olur. Kuyruktaki birikme azaldığında RED devreden çıkarak yavaşlayan TCP bağlantılarının tekrar hızlanmasına izin verecektir.

Şekil 3.17'daki senaryoda, router'a 50 paket gönderilmiş, son 10 paket kuyruğun dolması nedeniyle atılmış ve kuyruk taşmaya başlamıştı. Şimdi, 50 paketin 10 farklı TCP bağlantısının parçası olduğunu ve RED'in kuyruk dolmadan önce her bağlantıdan 1 paket attığını

varsayalım. Burada da ilk etapta 10 TCP paketi kaybolacak, ancak en azından bir sonraki gönderimde toplam TCP paket sayısı 25'e düşerek kuyruğun taşması engellenebilecektir.

3.2.4.1.2 Kuyruklama

Kuyruklama, paket kayıplarının önlenmesi konusu da dahil olmak üzere pek çok sorunun çözümünde kullanılmaktadır. Paket kayıpları kuyruklar dolduğunda oluştuğundan ve kuyruklama yöntemleri genelde sadece kuyruk boyutunun ayarlanmasına izin verdiğinden, yapılabilecek tek şey kuyruk boyutlarıyla oynamaktır. Ayarlama sırasında büyük boyutlu kuyrukların paket kayıplarını azaltırken kuyruklama gecikmesini arttırdığı unutulmamalıdır.



Şekil 3.18 Kuyruklama ve Paket kaybı arasındaki ilişki

Uzun kuyrukların kullanılması, gecikmeye duyarlı trafikler için sorun yaratırken paket kaybına duyarlı trafikler için iyi çalışabilir. Bu nedenle, gecikmeye duyarlı trafikleri Şekil 3.18'deki gibi az gecikme sağlayan ancak kayıp riski yüksek olan kısa kuyruklara, paket kaybına duyarlı trafikleri de az kayıp sağlayan ancak çok gecikme yaratabilecek uzun kuyruklara sokmak mantıklı olabilir.

QoS tekniklerinin paket kaybına etkileri Tablo 3.7'de verilmiştir.

QoS Tekniği	Paket Kaybına Etkisi
Kuyruklama (Queuing)	Büyük boyutlu kuyruklar oluşturmak gecikmeyi arttırırken paket kaybını azaltır.
RED	RED, kuyruklar dolmadan bazı TCP paketlerini rasgele atarak kuyruğa giren toplam paket sayısını azaltır. Paketler kaybolduğunda, TCP pencere boyutu önceki pencere boyutunun yüzde ellisine kadar düşer. Böylece, sadece birkaç TCP bağlantısını yavaşlatarak kuyruğun dolmasını ve tüm bağlantıların paket kaybı yaşanmasını engellemiş olur.

Tablo 3.7 QoS tekniklerinin paket kaybına etkisi

3.3 Donanıma Karşı Yazılım

Hizmet kalitesinin donanım veya yazılım ile gerçekleştirilmesi arasında büyük fark vardır. Birçok satıcı, ürünlerine QoS özelliği eklediğini söyleyerek bu durumla övünmektedir. Onlara göre yapılması gereken daha az masraf yaparak bu donanımlara yazılımlar yardımıyla katkı sağlanmasıdır. Her ne kadar bu fikir çok cazip ve ilgi çekici gibi görünse de pratik olarak uygulanabilirliği çok azdır. Bir QoS mekanizması tamamen yazılım ile gerçekleştiriliyorsa ve donanım ile ilgisi yoksa, bu mekanizma yalnızca ağ trafiğinin çok az olduğu zamanlarda iyi çalışabilir. Ağ trafiği arttığında aynı şeyi söylemek mümkün olmayacaktır.

Tamamen yazılım tabanlı ortamlarda hizmet kalitesi çözümleri etkisiz kalacak, yazılım bazlı sınıflandırıcılar yavaş çalışacağı için gecikmelere neden olacaktır. Bunun sebebi yazılım tabanlı QoS mekanizmalarının her bir paketi yazılım seviyesinde ele almak ve değerlendirmek zorunda oluşudur. Bu doğrultuda, paketlere verilen öncelikler de yazılım seviyesinde tespit edilmek zorunda kalınacaktır.

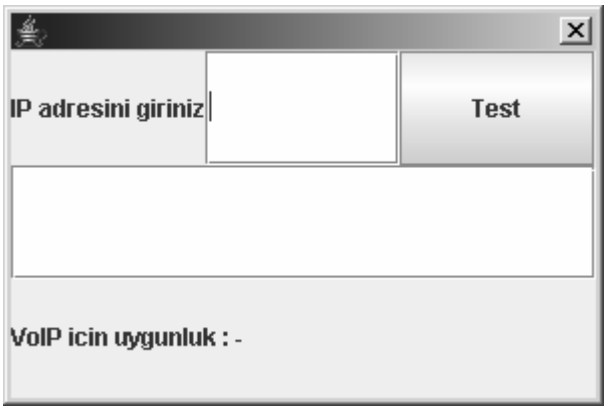
Yazılım bazında çalışan hiçbir QoS mekanizması asla bir switch veya router'in hızına ulaşamayacaktır. Bu durum otomatik olarak ağ performansının düşmesine, trafikte gecikmelere ve hatta paket kayıplarına sebep olabilecektir.

Gerçekçi olarak bakıldığında yazılım ve donanımın en uygun seviyelerde birlikte kullanılması en iyi seçimdir. Yazılım, esneklik ve ileriye dönük büyüme olanaklarını artırırken, donanım da gerçek seviyede hız ve performans vaat etmektedir.

4. UYGULAMALAR

4.1 Hattın VoIP için Uygunluğunu İnceleyen Basit Bir Java Uygulaması

Program, Eclipse 3.1 üzerinde Java programlama dili kullanılarak yazılmıştır. Amaç, ADSL üzerinden çağrı taşıyan CallShop'lar veya bireysel VoDSL kullanıcıları için hattın çağrı yapmak için uygun olup olmadığını belirlemektedir. Program, ICMP'yi kullanan "Ping" komutu tarafından sağlanan bilgiler yardımıyla yorumlama yapmaktadır. Uygulama arayüzü Şekil 4.1'de verilmiştir.



Şekil 4.1 Java uygulamasının arayüzü

4.1.1 ICMP ve PING

ICMP, TCP/IP'nin işlemesine yardımcı olan kontrol mesajları protokolüdür. Her uç birimde mutlaka ICMP protokolü çalışır ve hata durumunda geri bilgilendirmeyi sağlar. ICMP mesajı, IP paketinin veri bölümünde taşınır. Bu yüzden ICMP paketlerinin dağıtım güvenilirliği, IP paketlerinin dağıtım güvenilirliği ile sınırlı kalmaktadır.

Ping, ağ üzerinde bulunan cihazların ulaşılabilirliğini test etmek için ICMP protokolünü kullanan bir uygulamadır. İki cihaz arası ulaşılabilirlik şu şekilde tanımlanabilir. Eğer, A cihazından gönderilen paketler B cihazı tarafından alınıp işlenebiliyor ise; A cihazından B cihazına ulaşılabiliyor demektir. Ping uygulaması, ICMP mesajını içeren IP paketlerini ağ üzerinden gönderir. Bu uygulamanın kullandığı iki tür ICMP paketi vardır. Bunlar,

- *ICMP echo request* paketi
- *ICMP echo reply* paketidir.

Ping uygulaması test işlemi için ***ICMP echo request*** paketini içeren bir IP paketini gönderir. Bu paket aşağıdaki bilgileri içeren ***ICMP echo reply*** paketi ile cevaplanır

- TTL süresi dolduğu zaman paketin sahibine bildirim yapmak

- Herhangi bir durumda yok edilen paket hakkında geribildirim sağlamak

- Parçalanmasın komutu verilmiş paket parçalandığında geribildirim sağlamak

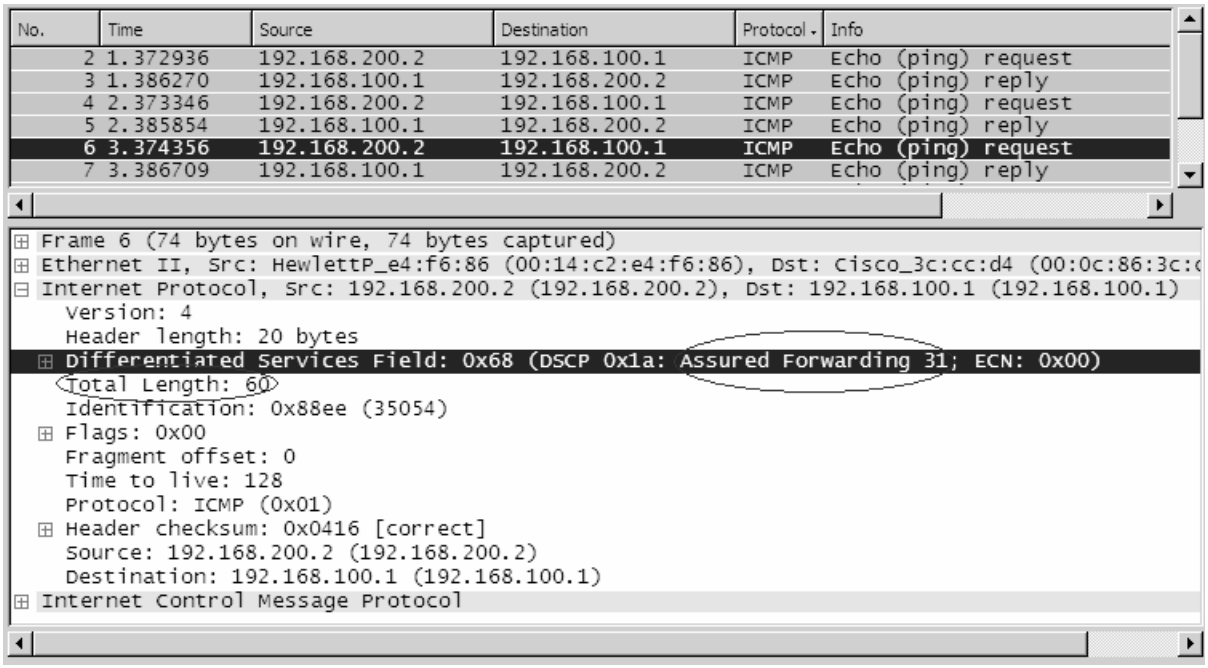
- Hata oluşumlarında geribildirim sağlamak

- Paketin iletim süresi hakkında geribildirim sağlamak.

4.1.2 Java Uygulamasının Çalışması

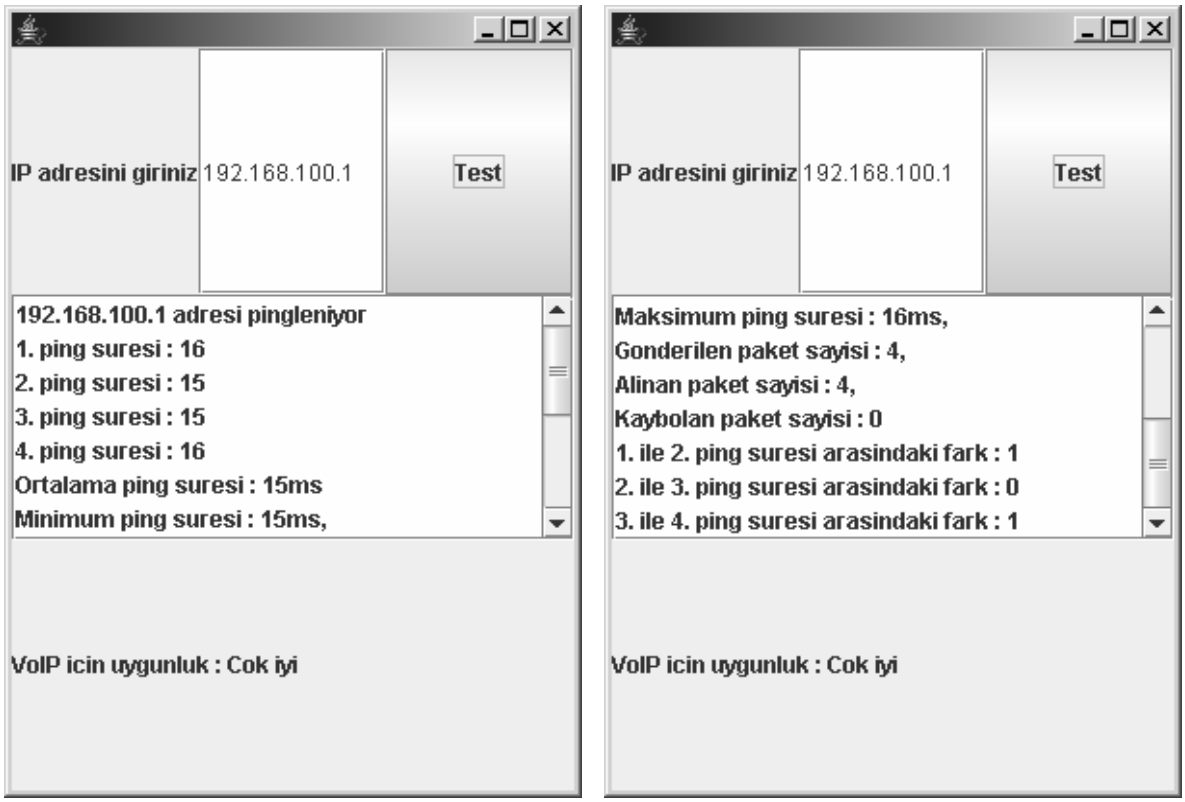
Uygulama, sorgulama yaparken ping konutunun varsayılan paket boyutu olan 32 Byte yerine, genelde ses paketlerinin gönderim boyutu olan 60 Byte'lık (40 Byte'lık IP/UDP/RTP başlıkları ve 20 Byte'lık ses yükü) veri kullanmaktadır. Ayrıca veri, ses sinyalleşme paketlerinin etiketlenmesinde kullanılan AF31 (ikilik sistemde 01101000 veya onluk sistemde 104) ile etiketlenerek, bu etiketi dikkate alan sistemlerde ses paketlerine uygulanan önceliklerden faydalanması amaçlanmıştır. Bu etiket, IP başlığında bulunan Servis Tipi (ToS) Byte'ında ifade edilmektedir. Aşağıda, bu özellikleri sağlayan ping komutu ve sinyalleşmeleri izlemek için kullanılan ethereal programının çıktısı (Şekil 4.2) verilmiştir.

ping –v 104 –l 60 <ip adresi>



Şekil 4.2 Uygulamanın Ethereal çıktısı

Program birbirinin peşi sıra dört ICMP paketi göndermekte ve gelen bilgiler ışığında paket kayıp oranını, paketler arasındaki gecikme ve jitter değerlerini bulmaktadır. Hattın durumu hakkında yorumlama yaparken toplam gidiş dönüş süresinin en fazla 300ms olması gerektiği göz önünde bulundurulmaktadır. Bu sürenin aşılması kaliteyi kötü yönde etkilemektedir. Ayrıca, paketler arası gecikmenin 30ms'den fazla olması durumunda kayıp paketlerin alıcı tarafta yeniden oluşturulması mümkün olmadığından ve bu sürede sessizlik duyulacağından, karar verirken iki paket arasındaki gecikmenin 30ms'den az olması gerekliliği de dikkate alınmıştır. Şekil 4.2'de uygulamanın işletilmesi sonrasındaki durumu görülmektedir. Programın kaynak kodları EK-1'de verilmiştir.

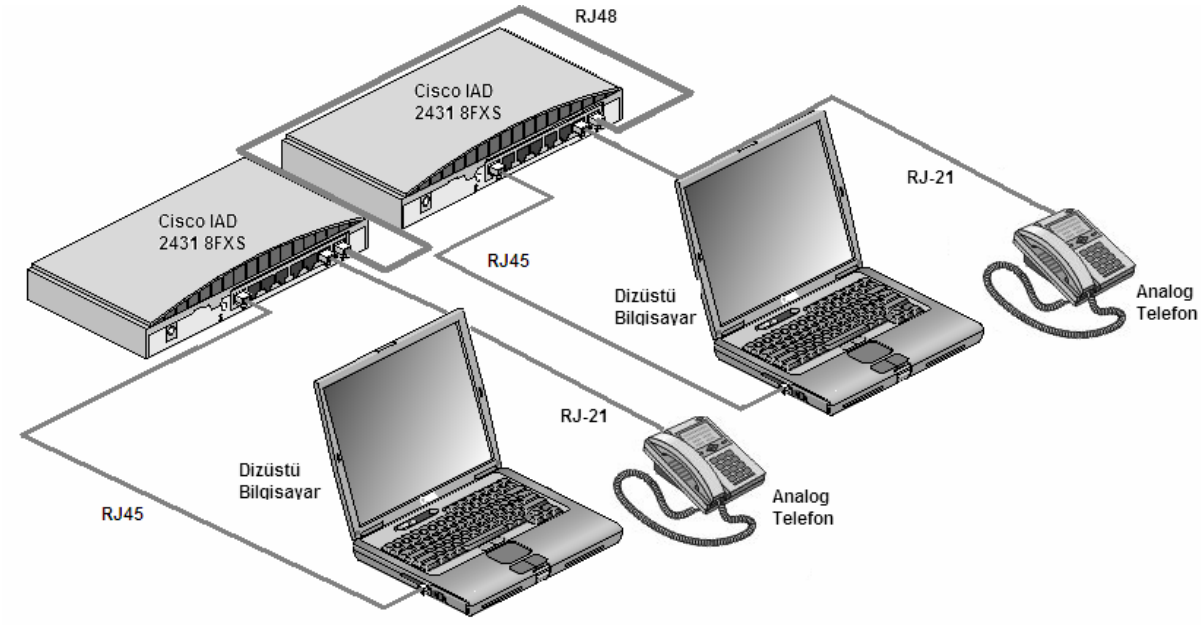


Şekil 4.3 Java uygulamasının çıktıları

4.2 IP Üzerinden Ses İletiminde Hizmet Kalitesi ile İlgili Uygulamalar

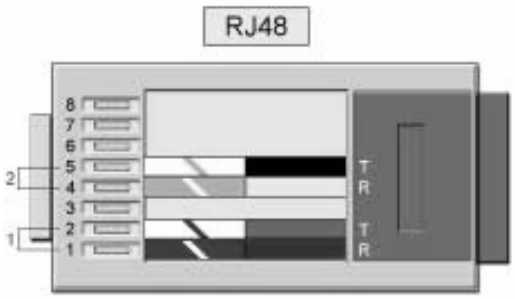
Bu kısımda, IP üzerinden ses iletiminde hizmet kalitesini sağlamak için kullanılan ve "Hizmet Kalitesi" bölümünde anlatılan tekniklerin uygulamaları yapılacaktır. Uygulamaların gerçekleştirileceği test ortamı; ses iletimi için özel olarak imal edilmiş iki adet Cisco IAD 2431 8FXS Gateway, üzerinde FTP ve HTTP sunucu çalışan iki adet dizüstü bilgisayar, iki

adet analog telefon ve bu cihazların birbiriyle haberleşmesi için gerekli olan kablolamalardan oluşmaktadır. Test ortamının temsili gösterimi Şekil 4.4'te verilmektedir.



Şekil 4.4 Uygulama ortamı

Gateway'ler arası haberleşme, cihazların E1 portları arasında RJ-48 bağlantısı ile sağlanmaktadır. Bu bağlantıyı sağlamak için gereken kablo, yarım metre uzunluğundaki CAT5 kablonun ortadan kesilmesi ve Şekil 4.5'te gösterildiği gibi 1-2 ve 4-5 çiftlerinin karşılıklı olarak çapraz bağlanmasıyla elde edilmiştir.



Şekil 4.5 RJ-48 bağlantı şekli

Cihazlar üzerinde bulunan 2048 Kbps hızındaki E1 portları, her biri 64 Kbps hıza sahip otuziki time-slot'a bölünebilmekte, ancak bu time-slot'lardan bir tanesi sinyalleşme için kullanıldığından trafik akıtmak için toplamda otuzbir time-slot kullanılabilmektedir. Orta ölçekli firmaların dahi bu bant genişliğini doldurabilmesi zor olduğundan, test ortamında uygulamalar gerçekleştirilirken bu bant genişliği 128 Kbps olarak ayarlanmıştır.

4.2.1 Hizmet Kalitesine Etki Eden QoS Teknikleri

Hizmet kalitesine etki eden sıkıştırma, çağrı izin kontrolü, kuyruklama ve rasgele erken tesbit QoS teknikleri gerçek ortam koşullarını sağlayan uygulama ortamında test edilmiştir. Uygulaması yapılacak olan bu tekniklerin teori kısımları tezin ikinci bölümde açıklanmaktadır. Gateway'lerin konfigurasyonları EK-2'de verilmiştir.

4.2.1.1 Sıkıştırma

Sıkıştırma iletilecek veri miktarını azaltmak için kullanılan bir yöntemdir. Bu uygulamada 128 Kbps hızındaki hattan G.711 codec kullanılarak bir ses trafiği geçirilmiştir. Sonrasında RTP başlık sıkıştırması kullanılarak gönderilen veri miktarının ne kadar azaldığı incelenmiştir.

Öncelikle ANKARA ve ISTANBUL adını verilen gateway'lerde bir ve ikinci time slotlar kullanılarak hattın hızı 128 Kbps olarak ayarlanmıştır. Bunun için her iki gateway'de de aşağıdaki komutları kullanılmıştır.

```
ANKARA#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ANKARA(config)#card type e1 1
ANKARA(config)#controller e1 1/0
ANKARA(config-controller)#channel-group 0 timeslots 1-2
```

Daha sonra ANKARA'nın seri arayüzü için 192.168.100.2, ISTANBUL'un seri arayüzü için 198.100.1 IP adreslerini verilmiştir.

```
ANKARA#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ANKARA(config)#interface serial1/0:0
ANKARA(config-if)#ip address 192.168.100.2 255.255.255.252
```

Cihazların birbirlerini görüp görmediklerini kontrol etmek için PING komutu kullanılmıştır.

```
ANKARA#ping 192.168.100.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.100.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/18/20 ms
```

Sonrasında, gateway'lere taktığımız telefonlara numara vermek ve birbirlerini arayabilmelerini sağlamak için arama kuralları yazılmıştır. VoIP üzerinden yapılacak çağrının G.711 codec'ini ve SIP protokolünü kullanması için gerekli düzenleme burada yapılmıştır. Bu işlemlerden sonra ANKARA 312 numarasıyla çağrı alıp, 212 numarasıyla ISTANBUL'u arayabilecek duruma gelmiştir.

```
dial-peer voice 6 pots
 destination-pattern 312
 port 2/5
 forward-digits all
!
dial-peer voice 9 voip
 destination-pattern 212
 no modem passthrough
 session protocol sipv2
 session target ipv4:192.168.100.1
 dtmf-relay rtp-nte
 codec g711ulaw
```

RTP başlık sıkıştırmasından önce ve sonra iletilmesi gereken veri miktarı teorik olarak hesaplanarak pratikte elde edilen sonuçla karşılaştırılmıştır. Aşağıdaki formül ve Tablo 4.1'deki veriler kullanılarak hesap yapıldığında, RTP başlığının sıkıştırılmasından önce iletilmesi gereken veri miktarı 85,6 Kbps iken sıkıştırmadan sonra iletilmesi gereken veri miktarının 70,4 Kbps olduğu görülmüştür

$$\text{Toplam Paket Boyutu} = \text{Ses Paket Yükü} + \text{IP/UDP/RTP(veya cRTP)} + \text{L2 Başlığı} \qquad (4.1)$$

$$\text{Toplam Bant Genişliği} = \frac{\text{Ses Bant Genişliği} + \text{Toplam Paket Boyutu}}{\text{Ses Paket Yükü}} \qquad (4.2)$$

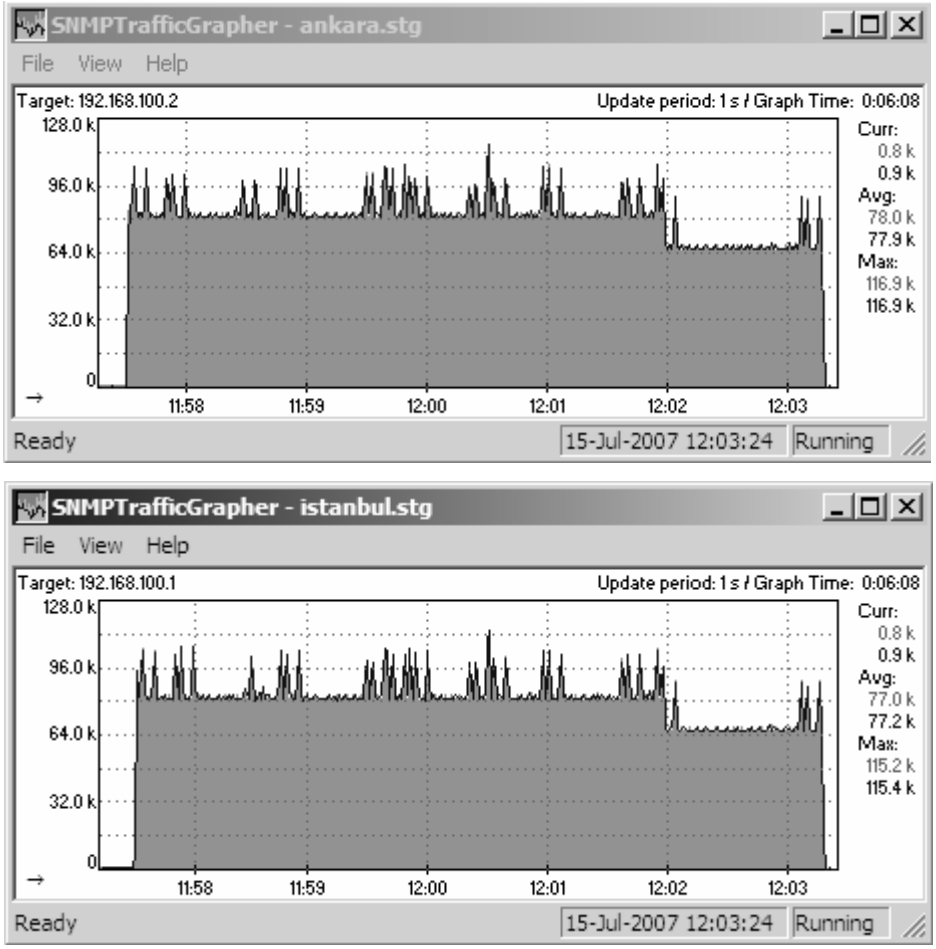
Codec	Ses Bantgenişliği	Ses Paketi Yükü	Ses Paketi Boyutu	IP/UDP/RTP Başlığı	cRTP	L2 Başlığı (Ethernet)	Toplam Bantgenişliği
g.711	64 Kbps	160 Bytes	20 ms	40 Bytes		14 Bytes	85,6 Kbps
g.711	64 Kbps	160 Bytes	20 ms		2 Bytes	14 Bytes	70,4 Kbps

Tablo 4.1 G.711 codec kullanıldığında sıkıştırmanın teorik etkisi

ANKARA gateway'ine bağlı 312 numalı telefon kaldırılarak 212 tuşlandığında ISTANBUL gateway'ine bağlı 212 numaralı telefon çalacaktır. Telefon açıldığında aradaki hat üzerinden RTP trafiği akmaya başlayacaktır. ANKARA gateway'inden RTP trafiğinin detaylarına bakıldığında çağrının 192.168.100.2 ve 192.168.100.1 IP adresleri arasında 17678 ve 18750 RTP portlarını kullanarak gidip gelmekte olduğu görülmektedir.

```
ANKARA#sh voip rtp connections detail
VoIP RTP active connections :
No. CallId  dstCallId  LocalRTP RmtRTP LocalIP         RemoteIP
1   8       7          17678    18750  192.168.100.2   192.168.100.1
 callId 8 (dir=2): called=212 calling=312 redirect=
     dest callId 7: called= calling=312 redirect=
   1 context 653F346C xmitFunc 61A6DFC0
Found 1 active RTP connections
```

Aradaki hattın doluluk oranına hem ANKARA hem de ISTANBUL tarafından STG programı yardımıyla bakıldığında oluşan grafik Şekil 4.6'da verilmiştir. Şekil 4.6'da verilen grafik, 11:57'de G.711 codec'i kullan bir çağrı başladığını göstermektedir. Bu çağrıda, RTP başlık sıkıştırılmasının yapılmadığı durumda iletilmesi gereken veri miktarı ortalama 83 Kbps olarak görülmektedir, 12:02'de hem ANKARA hem de ISTANBUL gateway'inde karşılıklı olarak yapılan RTP başlık sıkıştırmasından sonra iletilmesi gereken veri miktarı ortalama 68 Kbps olmuştur.



Şekil 4.6 RTP başlık sıkıştırmasının etkisi

4.2.1.2 Çağrı İzin Kontrolü (CAC)

CAC, ağın yeni bir ses veya video çağrısına izin verip vermeyeceğine karar verir. Uygulamada, eş zamanlı tek bir G.711 VoIP çağrısının yapılmasına izin verilmiştir. Bunun nedeni, RTP başlık sıkıştırması yapılmadan iki G.711 çağrısının iletmesi gereken veri miktarının yaklaşık 166 Kbps olması gerekirken bizim yalnızca 128 Kbps'lik bir hattımızın olmasıdır.

Çağrı izin kontrolünü kullanmak için, cihazın IP dünyasına açılan bacağında ayarlama yapılması gerekmektedir. Aşağıdaki komut yardımıyla CAC ayarlanarak gelecek ikinci çağrının reddedilmesi sağlanabilmektedir. IP üzerinden gitmesi reddedilen çağrı ya yapılamayacak ya da gerekli bağlantılar yapılmışsa PSTN'e gönderilecektir.

```
ANKARA# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ANKARA(config)#dial-peer voice 9 voip
ANKARA(config-dial-peer)# max-conn 1
```

ANKARA gateway'inden ISTANBUL gatway'i aranarak bir çağrı kurulduğunda CAC için ayarlanan sınır değerine ulaşılmış olacaktır.

```
ANKARA#sh call active voice compact
 <callID>  A/O FAX T<sec> Codec       type  Peer Address IP R<ip>:<udp>
Total call-legs: 2
       21 ANS     T8     g711ulaw    TELE  P312
       22 ORG     T8     g711ulaw    VOIP  P212    192.168.100.1:17586
```

Bu sınır değerini aşan çağrı talepleri aşağıdaki gibi bir uyarı ile reddedilecektir. Bu uygulamada cihazlar'ın PSTN bağlantısı olmadığından çağrılar bu şebekeye aktarılamamıştır.

```
ANKARA#
*Mar  1 02:34:21.667: %CALL_CONTROL-6-MAX_CONNECTIONS:
Maximum number of connections reached for dial-peer 9
```

4.2.1.3 Kuyruklama ve RED

Kuyruklama tekniği, hangi veri tipinin ne kadar bant genişliği kullanacağı konusunda etkilidir. Bu uygulamada, ses paketlerine öncelik verilecek ve ayrı bir kuyruktan hatta girmeleri sağlanacaktır. Öncelik verilecek paketler tamamen ağ yöneticisinin vereceği kararlar doğrultusunda belirlenmektedir. Çıkışta birden fazla kuyruğa öncelik verilerek, örneğin sesin yanında http trafiğinin de hatta diğer paketlerden hızlı girmeleri sağlanabilir. Kullanılan algoritmalar, belirli bir kuyruğa sabit bir bant genişliği garanti edebilir. Uygulamada, sesin taşındığı RTP trafiğine 58 Kbps ve sinyalleşmesine de 8 Kbps ayrılacaktır. Geri kalan 62 Kbps bantgenişliği diğer veri trafikleri tarafından kullanılacaktır.

Kuyruklama'da öncelikli adım verilerin işaretlenmesi ve tasnifidir. Ayrıca, parçalama ve yerleştirme tekniğinin uygulanabilmesi için multilink konfigurasyonu da yapılmalıdır.

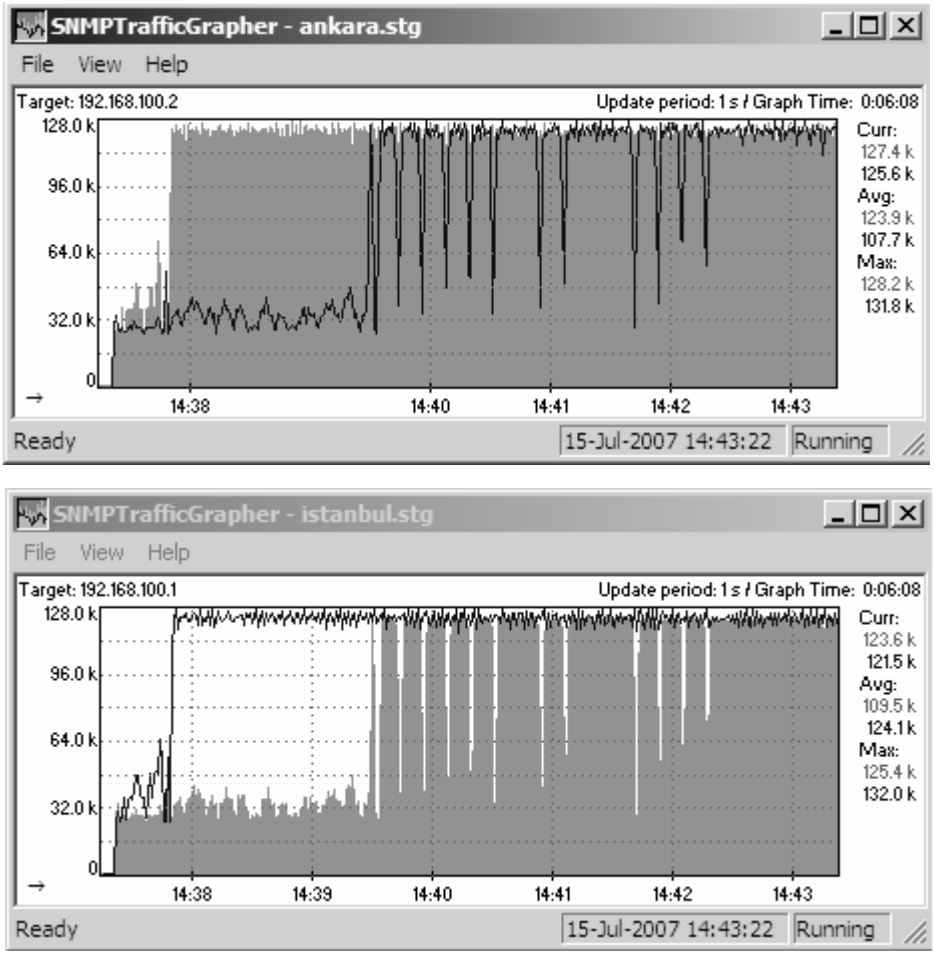
Bu uygulamada, dizüstü bilgisayarlar gateway'lerin Fast Ethernet portlarına bağlanarak ANKARA'daki bilgisayara 192.168.200.2, ISTANBUL'daki bilgisayara 192.168.212.2 IP adresleri verilmiştir.

Hattı doldurmak ve kuyruklamanın oluşmasını sağlamak için farklı trafikler kullanılmıştır. Öncelikle ANKARA gateway'ine bağlı telefondan ISTANBUL gateway'ine bağlı telefon aranmış ve SIP protokolü üzerinden G.711 codec kullanılarak bir çağrı kurulmuştur. Sonrasında ISTANBUL tarafındaki bilgisayardan WANKiller programı yardımıyla ANKARA gateway'ine doğru 1500 byte'lık paketler gönderilmeye başlanmıştır. Ayrıca bilgisayarlarda FTP sunucusu çalıştırılarak karşılıklı olarak dosya transferi başlatılmıştır. Son olarak ANKARA tarafındaki bilgisayar ile üzerinde http sunucusu çalışan ISTANBUL tarafındaki bilgisayara bağlanılarak avi dosyası oynatılmıştır.

Bu sırada daha önceden belirlenen kriterlere göre paketlerin yakalanıp doğru kuyruklara verilip verilmediğine bakılmıştır. Aşağıdaki çıktıda, yaptığımız görüşmede iletilen paketlerin "voice-traffic" ve "voice-signaling" kuralları tarafından yakalandığı, ses içermeyen trafiklerin ise "class-default" kuralına takıldığı görülmektedir. Bu uygulamada ses trafiği için 58 Kbps, ses sinyalleşme trafiği için 8 Kbps ve geri kalan trafikler için de 62 Kbps bant genişliği ayrılmıştır.

```
ISTANBUL#sh policy-map interface multilink 1
 Multilink1

  Service-policy output: VOICE-POLICY

    Class-map: voice-traffic (match-all)
     5949 packets, 368838 bytes
      5 minute offered rate 24000 bps, drop rate 0 bps
      Match: access-group 102
      Queueing
        Strict Priority
        Output Queue: Conversation 40
        Bandwidth 58 (kbps) Burst 1450 (Bytes)
        (pkts matched/bytes matched) 5949/368838
       (total drops/bytes drops) 0/0

    Class-map: voice-signaling (match-all)
     27 packets, 5943 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group 103
      Queueing
        Output Queue: Conversation 41
        Bandwidth 8 (kbps)Max Threshold 64 (packets)
        (pkts matched/bytes matched) 27/5943
       (depth/total drops/no-buffer drops) 0/0/0

    Class-map: class-default (match-any)
     3574 packets, 1914190 bytes
      5 minute offered rate 59000 bps, drop rate 20000 bps
      Match: any
      Queueing
        Flow Based Fair Queueing
        Maximum Number of Hashed Queues 32
       (total queued/total drops/no-buffer drops) 65/4407/0
```

Hattın kullanım durumuna STG programı ile bakıldığında, hatta tıkanmaların yaşandığı görülebilmektedir. Şekil 4.7'deki grafikte çentik şeklindeki çizimler aslında RED mekanizmasının çalıştığını ve kuyruklar dolmaya başladığında bazı TCP paketlerinin atılarak bu TCP bağlantılarının yavaşlamasının sağlandığını göstermektedir.



Şekil 4.7 Tam kapasite kullanıldığından kuyruklamaya neden olan hat ve RED etkisi

Kuyruk sayısı hattın doluluk oranına ve hattan geçen trafik çeşidine göre değişmektedir. Aşağıdaki çıktıda beş ayrı kuyruk görülmektedir. Bu kuyruklardan ilki ToS 184 (DSCP EF) ile işaretlenmiş olan ses trafiği tarafından kullanılmaktadır. Diğer kuyruklarda paket kayıpları yaşanırken bu kuyrukta hiç paket kaybı olmadığı görülmektedir. Bunun nedeni EF ile işaretlenmiş ses paketlerinin önceliklendirilmesi ve hatta en hızlı şekilde verilmesinin sağlanmasıdır.

```
ISTANBUL#sh queue multilink 1
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 1547
  Queueing strategy: Class-based queueing
  Output queue: 67/1000/64/1547/15656 (size/max total/threshold/drops/interleaves)
     Conversations  5/7/32 (active/max active/max total)
     Reserved Conversations 1/1 (allocated/max allocated)
     Available Bandwidth 30 kilobits/sec

  (depth/weight/total drops/no-buffer drops/interleaves) 1/0/0/0/0
  Conversation 40, linktype: ip, length: 62
  source: 192.168.100.1, destination: 192.168.100.2, id: 0x00D9, ttl: 254,
  TOS: 184 prot: 17, source port 17188, destination port 17474

  (depth/weight/total drops/no-buffer drops/interleaves) 1/4626/0/0/428
  Conversation 19, linktype: ip, length: 44
  source: 192.168.100.1, destination: 192.168.200.2, id: 0xF455, ttl: 255,
  TOS: 192 prot: 6, source port 23, destination port 2159

  (depth/weight/total drops/no-buffer drops/interleaves) 1/32384/0/0/0
  Conversation 10, linktype: ip, length: 42
  source: 192.168.212.2, destination: 192.168.200.2, id: 0x4631, ttl: 127,
  TOS: 0 prot: 6, source port 1070, destination port 2170

  (depth/weight/total drops/no-buffer drops/interleaves) 61/32384/1549/0/13347
  Conversation 20, linktype: ip, length: 1458
  source: 192.168.212.2, destination: 192.168.100.2, id: 0x4491, ttl: 127,
  TOS: 0 prot: 17, source port 1068, destination port 7

  (depth/weight/total drops/no-buffer drops/interleaves) 3/32384/0/0/1811
  Conversation 29, linktype: ip, length: 1502
  source: 192.168.212.2, destination: 192.168.200.2, id: 0x4619, ttl: 127,
  TOS: 0 prot: 6, source port 80, destination port 2191
```

5. SONUÇLAR VE TARTIŞMA

VoIP, sesin IP paketlerine dönüştürülerek IP tabanlı şebekeler üzerinden iletilmesi hizmetidir. VoIP hizmetinin sunumu esnasında aboneye hizmetin kalitesi ile ilgili herhangi bir garanti verilmesi teknik anlamda mümkün değildir. Gerçekten de, sesin PSTN şebekeleri üzerinden iletilmesi sırasında aboneler arasında baştan sona kadar devre tahsisi yapılıp, eş zamanlı ve kalitesi garanti edilmiş bir hizmet verilebilmekteyken, sesin IP ortamında nakli sırasında paket kayıpları ve istenmeyen gecikmelerin ortaya çıkması kaçınılmazdır.

Hizmet kalitesinin sağlanması için kullanılan teknikler bir veya birkaç veri tipi için olumlu yönde etki yaparken diğerleri için olumsuz etki yapmaktadır. Aslında temelde yapılan işlem varolan kaynakların bilinçli bir şekilde paylaştırılmasını sağlamaktadır. Bu tezde yapılan uygulamalardan da anlaşıldığı gibi her tekniğin olumlu ve olumsuz yönleri bulunmaktadır. Sıkıştırma tekniği, iletilecek veri miktarını azaltırken, paketlerin sıkıştırılması ve açılması için belirli bir işlem yükü gerektirdiğinden toplam gecikmeyi arttırabilmektedir. Aynı durum kuyruklama için de geçerlidir. Birden çok kuyruk yaratarak verileri bu kuyruklara dağıtmak, öncelikli kuyruklardaki paketlerin hızlı iletilmesini sağlarken diğerlerinin gecikmeye maruz kalmasına neden olmaktadır. Rasgele erken tespit yöntemi de trafik tipine göre bazı trafiklerin paket kaybına maruz kalmaması için diğerlerinin paketlerini bilinçli olarak kuyruktan atmaktadır. Bu sonuçlar göstermektedir ki, önceliklendirilecek veri tiplerinin belirlenmesi ve bu veri tiplerine uygun QoS tekniklerinin seçilmesi sırasında uygulamanın ağdaki diğer veri trafiklerine etkisi de incelenmelidir.

VoIP konusunda dünya üzerindeki uygulamaları incelediğimizde ABD başta olmak üzere pek çok gelişmiş ülkede VoIP'in herhangi bir düzenlemeye tabi tutulmaksızın serbest olduğu, Avrupa Birliği ülkelerinde sesin gerçek zamanlı olması halinde düzenlemeye tabi tutulduğu, gelişmekte olan ve az gelişmiş ülkelerde ise VoIP'nin sınırlandırılmış veya yasak olduğu göze çarpmaktadır. VoIP'nin gelişiminde ülkelerin hukuki, kurumsal, teknik ve ekonomik durumlarının önem taşıdığı, VoIP'nin gelişimi ile birlikte bu ülkelerde hukuki, kurumsal, teknik, ekonomik ve sosyal boyutlarda gelişmeler yaşandığı gözlenmektedir.

Ciro büyüklüğü, teknolojik altyapısı, halen devam eden tekel hakları ve hâkim konumu nedeniyle rekabet edilmesi mümkün olmayan ve 2006 yılında %55'i özelleştirilen Türk Telekom A.Ş. karşısında serbest sektör firmalarının varlıklarını sürdürebilmesi için Ulaştırma Bakanlığı, Telekomünikasyon Kurumu ve Rekabet Kurumu'na büyük görevler düşmektedir. Henüz emekleme safhasını yaşayan serbest sektör işletmecilerinin, Türk Telekom A.Ş.'nin

yok edici yaklaşımları karşısında, yasa ve yönetmelikleri doğru ve ülke yararına yorumlayan Bakanlık ve Kurul kararlarına ihtiyaç duyduğu aşikârdır. 1 Ocak 2004 yılında çıkarılan yasayla alternatif telekom operatörlerinin önünü açan bir dizi gelişme olsa da, Türkiye Avrupa'da şehir içi sabit ses trafiğinin rakabete açık olmadığı tek ülke konumunda bulunmaktadır. Telefon görüşmelerinin %80'den fazla kısmını oluşturan şehir içi görüşmelerin 2007 yılı itibarıyla hala rekabete açık olmaması Türkiye'deki VoIP pazarının gelişmesini de engellemektedir. Pazar ve veriler değerlendirildiğinde ülkemizde iletişim pazarında halen monopol yapının devam ettiği görülebilir. Serbestleşen pazarlarda telekom hizmetlerinin ülke ekonomilerinde büyümeyi tetikleyici bir rol oynadığı, başta bu sektöre yazılım sunan şirketler olmak üzere, gerek teknik gerekse hizmet ve yan sanayilerin de hızla geliştiği, tekelin kalkması ile tüketicilere katma değerlerli ekonomik servisler sunulabileceği açıktır.

VoIP'in önündeki engellerden bir diğeri de uluslar arası bir adresleme mekanizmasının oluşturulmamış olmasıdır. Internet'ten PSTN kaynaklarına ve internet'ten internet kaynaklarına E.164 numaraları ile ulaşmak için kullanılan bir sitem olan ENUM (T*E*lephone *NU*mber *M*apping) yardımıyla PSTN şebekesi ile VoIP entegrasyonu sağlanmalıdır. Ancak ENUM'un tam anlamıyla kullanılabilmesi için tüm ülkelerin bu sisteme nasıl uyum sağlanacağı ile ilgili ortak bir çalışma içerisine girmesi gerekmektedir. Bu konudaki çalışmalar hala devam etmektedir.

KAYNAKLAR

Çetin, Erhan. Ege Üniversitesi (2005), Bilgisayar Ağlarında Servis Kalitesi Üzerine Bir İnceleme

Çölkesen, Rıfat. (2000), Ağ TCP/IP Unix, Papatya Yayınları

Haki, Erdem Halit. (2005), YTÜ seminer dersi raporu

Kirtiş, Veli Tan. (2007), Tellcom İletişim Hizmetleri A.Ş. Basın Toplantısı notları

Netaş Nortel Networks. (2000), Voice over IP

Oktuğ, Sema. (2004), İTÜ Bilgisayar Mühendisliği Bölümü, BLG433-Bilgisayar Haberleşmesi ders notları

Telkoder VoIP Komisyonu Raporu. (2003), No:1, Ankara

Wendell Odom, Michael J. Cavanaugh. (2004), Cisco DQOS Exam Certification Guide

VoIP over PPP Links with Quality of Service (LLQ / IP RTP Priority, LFI, cRTP), Document ID: 7111.

İNTERNET KAYNAKLARI

[1] Arku, Refik ve Arkut İbrahim. Internette çogulortam: sorunlar ve çözümler, http://inet-tr.org.tr/inetconf10/bildiri/29.pdf

[2] http://bid.ankara.edu.tr/start/hii/bolum1.html

[3] H.323 standartları, http://www.packetizer.com/voip/h323/standards.html

[4] http://www.javvin.com/protocolH225.html

[5] International Telecommunication Union, http://www.itu.int/net/home/index.aspx

[6] Salahlı, Vügar. Çanakkale Ondokuz Mart Üniversitesi, 2003, Transmission Control Protocol, www.enderunix.org/docs/tcpip/tcp/tcp.html

[7] The internet engineering task force, http://www.ietf.org/

[8] Türk Telekom tarihçesi, http://www.turktelekom.com.tr/webtech/default.asp?sayfa_id=73

[9] Türker, İpek. Çanakkale Ondokuz Mart Üniversitesi, 2003. User Datagram Protocol,

http://www.enderunix.org/docs/tcpip/udp/udp.htm

[10] Türkiye'de internet, http://www.internethaftasi.org.tr/hafta07/basin_aciklamasi.php

[11] Ünsal, Aziz. Çanakkale Ondokuz Mart Üniversitesi, 2003, Internet Control Message Protocol, http://www.enderunix.org/docs/tcpip/icmp/icmp.htm

EKLER

EK-1

```
package com.voip;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class Ping {
	BufferedReader in = null;

	String degerler[][] = new String[9][12];

	String address;

	String maxSure;

	String minSure;

	String ortSure;

	int sure1;

	int sure2;

	int sure3;

	int sure4;

	String gonderilenPaket;

	String gelenPaket;

	String kayipPaket;

	int uySur;

	int fark1;

	int fark2;

	int fark3;

	String uygunluk;

	String hata = null;
```

```
boolean dil = false; // ingilizce

public Ping() {
	address = "192.168.1.1";
	for (int i = 0; i < 12; i++)
		for (int j = 0; j < 9; j++)
			degerler[j][i] = null;
	uySur = 30;
}

public Ping(String address) {
	this.address = address;
	for (int i = 0; i < 12; i++)
		for (int j = 0; j < 9; j++)
			degerler[j][i] = null;
	uySur = 30;
}

// getter methods

public String getAddress() {
	return address;
}

public String getMaxSure() {
	return maxSure;
}

public String getMinSure() {
	return minSure;
}

public String getOrtSure() {
	return ortSure;
}

public int getSure1() {
	return sure1;
}

public int getSure2() {
	return sure2;
}

public int getSure3() {
	return sure3;
}

public int getSure4() {
	return sure4;
```

```
}

public String getGonderilenPaket() {
        return gonderilenPaket;
}

public String getGelenPaket() {
        return gelenPaket;
}

public String getKayipPaket() {
        return kayipPaket;
}

public int getUySur() {
        return uySur;
}

public int getFark1() {
        return fark1;
}

public int getFark2() {
        return fark2;
}

public int getFark3() {
        return fark3;
}

public String getUygunluk() {
        return uygunluk;
}

public String getDil() {
        return (dil) ? "Turkce" : "Ingilizce";
}

public String getHata() {
        return hata;
}

// setter methods
public void setUygunlukSuresi(int uySur) {
        this.uySur = uySur;
}


public void setAddress(String address) {
        this.address = address;
}
```

```
	public void olustur() {
		try {
			Runtime r = Runtime.getRuntime();
			Process p = r.exec("ping -v 104 -l 60 " + address);
			if (p == null) {
				hata = "Baglanti kurulamadi";
			}
			in = new BufferedReader(new
InputStreamReader(p.getInputStream()));
			String line;
			StringTokenizer st;
			int j = -1;
			while ((line = in.readLine()) != null) {
				st = new StringTokenizer(line, " ");
				int i = -1;
				if (st.hasMoreTokens())
					j++;
				else
					continue;
				System.out.print(j + ":: ");
				while (st.hasMoreTokens()) {
					i++;
					String satir = st.nextToken().toString();
					System.out.print(" " + i + "-> " + satir);
					if (j == 0 && i == 0 &&
Character.isDigit(satir.charAt(0)))
						dil = true;
					degerler[j][i] = satir;
				}
				System.out.println();
			}
			in.close();
		} catch (IOException io) {
			System.err.println(io.toString());
		}
		if (dil){
			if (Character.isDigit(degerler[1][0].charAt(0)) == false) {
				hata = "Baglanti kurulamadi";
			} else {
				getDeger(dil);
			}
		}else {
			if (Character.isDigit(degerler[1][2].charAt(0)) == false) {
				hata = "Baglanti kurulamadi";
			} else {
				getDeger(dil);
			}
		}

	}
```

```
public void getDeger(boolean dil) {
        for (int i = 0; i < 12; i++) {
                for (int j = 0; j < 9; j++) {
                        if (dil) { // turkce
                                switch (j) {
                                case 0:
                                        break;
                                case 1:
                                        sure1 = getSayi(degerler[j][3]);
                                        break;
                                case 2:
                                        sure2 = getSayi(degerler[j][3]);
                                        break;
                                case 3:
                                        sure3 = getSayi(degerler[j][3]);
                                        break;
                                case 4:
                                        sure4 = getSayi(degerler[j][3]);
                                        break;
                                case 5:
                                        break;
                                case 6:
                                        gonderilenPaket = degerler[j][3];
                                        gelenPaket = degerler[j][6];
                                        kayipPaket = degerler[j][9];
                                        break;
                                case 7:
                                        break;
                                case 8:
                                        minSure = degerler[j][3];
                                        maxSure = degerler[j][7];
                                        ortSure = degerler[j][10];
                                        break;
                                }
                                ;
                        } else { // ing
                                switch (j) {
                                case 1:
                                        sure1 = getSayi(degerler[j][4]);
                                        break;
                                case 2:
                                        sure2 = getSayi(degerler[j][4]);
                                        break;
                                case 3:
                                        sure3 = getSayi(degerler[j][4]);
                                        break;
                                case 4:
                                        sure4 = getSayi(degerler[j][4]);
                                        break;
                                case 5:
```

```
                                        break;
                                case 6:
                                        gonderilenPaket = degerler[j][3];
                                        gelenPaket = degerler[j][6];
                                        kayipPaket = degerler[j][9];
                                        break;
                                case 7:
                                        break;
                                case 8:
                                        minSure = degerler[j][2];
                                        maxSure = degerler[j][5];
                                        ortSure = degerler[j][8];
                                        break;
                                }
                                ;
                        }
                } // for j
        } // for i
        fark1 = Math.abs(sure1 - sure2);
        fark2 = Math.abs(sure2 - sure3);
        fark3 = Math.abs(sure3 - sure4);


        if (fark1 < uySur && fark2 < uySur && fark3 < uySur ) {
                uygunluk = "Cok iyi";
        } else if (fark1 > uySur && fark2 > uySur && fark3 > uySur) {
                uygunluk = "Cok kotu";
        } else if (fark1 < uySur && fark2 > uySur && fark3 > uySur) {
                uygunluk = "Kotu";
        } else if (fark1 > uySur && fark2 < uySur && fark3 > uySur) {
                uygunluk = "Kotu";
        } else if (fark1 > uySur && fark2 > uySur && fark3 < uySur) {
                uygunluk = "Kotu";
        } else if (fark1 > uySur && fark2 < uySur && fark3 < uySur ) {
                uygunluk = "Iyi";
        } else if (fark1 < uySur && fark2 > uySur && fark3 < uySur ) {
                uygunluk = "Iyi";
        } else if (fark1 < uySur && fark2 < uySur && fark3 > uySur ) {
                uygunluk = "Iyi";
        }
}

public int getSayi(String yazi) {
        String yanit = "";
        for (int i = 0; i < yazi.length() - 1; i++) {
                yanit += Character.isDigit(yazi.charAt(i)) ? String.valueOf(yazi
                                .charAt(i)) : "";
        }
        return Integer.parseInt(yanit);
}
```

```
package com.voip;

import javax.swing.DefaultListModel;
import javax.swing.JLabel;

class Voip {
	public static void main(String[] args) {
		try {
			Ping ping = new Ping("192.168.2.1");
			//Ping ping = new Ping("127.0.0.1");
			ping.olustur();
			if (ping.getHata() == null) {
				System.out.println(ping.getAddress() + " adresi pingleniyor");
				System.out.println("1. ping suresi : " + ping.getSure1());
				System.out.println("2. ping suresi : " + ping.getSure2());
				System.out.println("3. ping suresi : " + ping.getSure3());
				System.out.println("4. ping suresi : " + ping.getSure4());
				System.out.println("Ortalama ping suresi : " +
ping.getOrtSure());
				System.out.println("Minimum ping suresi : " +
ping.getMinSure());
				System.out.println("Maksimum ping suresi : " +
ping.getMaxSure());
				System.out.println("Gonderilen paket sayisi : " +
ping.getGonderilenPaket());
				System.out.println("Alinan paket sayisi : " +
ping.getGelenPaket());
				System.out.println("Kaybolan paket sayisi : " +
ping.getKayipPaket());
				System.out.println("1. ile 2. ping suresi arasindaki fark : " +
ping.getFark1());
				System.out.println("2. ile 3. ping suresi arasindaki fark : " +
ping.getFark2());
				System.out.println("3. ile 4. ping suresi arasindaki fark : " +
ping.getFark3());
				System.out.println("\nVoIP icin uygunluk : " +
ping.getUygunluk());
				System.out.println();
			} else {
				System.out.println(ping.hata);
			}
		} catch (Exception ex) {
			System.err.println("Hata Olustu!");
			System.out.println(ex.getMessage());
			System.out.println(ex.getStackTrace());
		}
	}

	public static void check(String IP) {
```

```
		check(IP, null, null);
	}

	public static void check(String IP, DefaultListModel liste, JLabel sonuc) {
		try {
			if (liste != null) liste.clear();
			if (sonuc != null) sonuc.setText("VoIP icin uygunluk : -");
			Ping ping = new Ping(IP);
			ping.olustur();
			if (ping.getHata() == null) {
				System.out.println(ping.getAddress() + " adresi pingleniyor");
				if (liste != null) liste.addElement(ping.getAddress() + " adresi
pingleniyor");
				System.out.println("1. ping suresi : " + ping.getSure1());
				if (liste != null) liste.addElement("1. ping suresi : " +
ping.getSure1());
				System.out.println("2. ping suresi : " + ping.getSure2());
				if (liste != null) liste.addElement("2. ping suresi : " +
ping.getSure2());
				System.out.println("3. ping suresi : " + ping.getSure3());
				if (liste != null) liste.addElement("3. ping suresi : " +
ping.getSure3());
				System.out.println("4. ping suresi : " + ping.getSure4());
				if (liste != null) liste.addElement("4. ping suresi : " +
ping.getSure4());
				System.out.println("Ortalama ping suresi : " +
ping.getOrtSure());
				if (liste != null) liste.addElement("Ortalama ping suresi : " +
ping.getOrtSure());
				System.out.println("Minimum ping suresi : " +
ping.getMinSure());
				if (liste != null) liste.addElement("Minimum ping suresi : " +
ping.getMinSure());
				System.out.println("Maksimum ping suresi : " +
ping.getMaxSure());
				if (liste != null) liste.addElement("Maksimum ping suresi : " +
ping.getMaxSure());
				System.out.println("Gonderilen paket sayisi : " +
ping.getGonderilenPaket());
				if (liste != null) liste.addElement("Gonderilen paket sayisi : " +
ping.getGonderilenPaket());
				System.out.println("Alinan paket sayisi : " +
ping.getGelenPaket());
				if (liste != null) liste.addElement("Alinan paket sayisi : " +
ping.getGelenPaket());
				System.out.println("Kaybolan paket sayisi : " +
ping.getKayipPaket());
				if (liste != null) liste.addElement("Kaybolan paket sayisi : " +
ping.getKayipPaket());
				System.out.println("1. ile 2. ping suresi arasindaki fark : " +
ping.getFark1());
```

```
                                if (liste != null) liste.addElement("1. ile 2. ping suresi arasindaki
fark : " + ping.getFark1());
                                System.out.println("2. ile 3. ping suresi arasindaki fark : " +
ping.getFark2());
                                if (liste != null) liste.addElement("2. ile 3. ping suresi arasindaki
fark : " + ping.getFark2());
                                System.out.println("3. ile 4. ping suresi arasindaki fark : " +
ping.getFark3());
                                if (liste != null) liste.addElement("3. ile 4. ping suresi arasindaki
fark : " + ping.getFark3());
                                System.out.println("\nVoIP icin uygunluk : " +
ping.getUygunluk());
                                if (sonuc != null) sonuc.setText("VoIP icin uygunluk : " +
ping.getUygunluk());
                                System.out.println();
                        } else {
                                System.out.println(ping.hata);
                                if (liste != null) liste.addElement(ping.hata);
                        }
                } catch (Exception ex) {
                        System.err.println("Hata Olustu!");
                        if (liste != null) liste.addElement("Hata Olustu!");
                        System.out.println(ex.getMessage());
                        if (liste != null) liste.addElement(ex.getMessage());
                        System.out.println(ex.getStackTrace());
                        if (liste != null) liste.addElement(ex.getStackTrace());
                }
        }
}


package com.voip;


import java.awt.Dimension;
import java.awt.GridLayout;


import javax.swing.DefaultListModel;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JList;

public class Pencere extends JFrame {

        public static void main(String arg[]) {
                Pencere pencere = new Pencere();
```

```
            pencere.show();
        }
        /**:
         *
         */
        private static final long serialVersionUID = 2058292485936350556L;
        private JPanel jContentPane = null;
        private JLabel etiket = null;  //  @jve:decl-index=0:visual-constraint="370,85"
        private JTextField jTextField = null;  //  @jve:decl-index=0:visual-
constraint="339,86"
        /**
         * This is the default constructor
         */
        public Pencere() {
            super();
            initialize();
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }

        /**
         * This method initializes this
         *
         * @return void
         *
         */
        private void initialize() {
            this.setSize(300, 200);
            this.setContentPane(getJContentPane());
        }

        /**
         * This method initializes jContentPane
         *
         * @return javax.swing.JPanel
         */
        private JPanel contentNorth = new JPanel(new GridLayout(1,3));
        private JButton jButton = null;  //  @jve:decl-index=0:visual-constraint="394,89"
        private JLabel jLabel = null;  //  @jve:decl-index=0:visual-constraint="614,128"
        private JList jList = null;  //  @jve:decl-index=0:visual-constraint="385,52"

        private JPanel getJContentPane() {
            if (jContentPane == null) {
                    etiket = new JLabel();
                    etiket.setText("IP adresini giriniz");
                    jContentPane = new JPanel();
                    jContentPane.setLayout(new GridLayout(3,1));
                    contentNorth.add(etiket, 0);
                    contentNorth.add(getJTextField(), 1);
                    contentNorth.add(getJButton(), 2);
                    jContentPane.add(contentNorth, 0);
                    JScrollPane jsp = new JScrollPane(getJList());
```

```
                jsp.setPreferredSize(new Dimension(200, 100));
                jContentPane.add(jsp, 1);
                jContentPane.add(getJLabel(), 2);
            }
            return jContentPane;
        }


        /**
         * This method initializes jTextField
         *
         * @return javax.swing.JTextField
         */
        private JTextField getJTextField() {
            if (jTextField == null) {
                jTextField = new JTextField();
            }
            return jTextField;
        }

        /**
         * This method initializes jButton
         *
         * @return javax.swing.JButton
         */
        private JButton getJButton() {
            if (jButton == null) {
                jButton = new JButton("Test");
                jButton.setSize(100, 18);
                jButton.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent e) {
                        Voip.check(getJTextField().getText(), listModel,
getJLabel());
                    }
                });
            }
            return jButton;
        }

        /**
         * This method initializes jLabel
         *
         * @return javax.swing.JLabel
         */
        private JLabel getJLabel() {
            if (jLabel == null) {
                jLabel = new JLabel();
               jLabel.setPreferredSize(new Dimension(100, 18));
                jLabel.setText("VoIP icin uygunluk : -");
            }
            return jLabel;
```

```
        }

        /**
         * This method initializes jList
         *
         * @return javax.swing.JList
         */
        private DefaultListModel listModel = new DefaultListModel();

        private JList getJList() {
                if (jList == null) {
                        jList = new JList(listModel);
                }
                return jList;
        }

}
```

EK-2

ANKARA Gateway Konfigurasyonu:

```
ANKARA#sh run
Building configuration...

Current configuration : 2509 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname ANKARA
!
boot-start-marker
boot-end-marker
!
card type e1 1
logging buffered 16000 debugging
enable password 1234
!
no aaa new-model
!
resource policy
!
network-clock-participate E1 1/0
!
no ip domain lookup
!
voice-card 0
!
!voice service voip
 fax protocol t38 nse force ls-redundancy 0 hs-redundancy 0 fallback cisco
 sip
!
controller E1 1/0
 channel-group 0 timeslots 1-2
!
class-map match-all voice-signaling
 match access-group 103
class-map match-all voice-traffic
 match access-group 102
!
policy-map VOICE-POLICY
 class voice-traffic
  priority 58
 class voice-signaling
  bandwidth 8
```

```
 class class-default
  fair-queue
!
interface Multilink1
 description 128kbps Istanbul Hatti
 bandwidth 128
 ip address 192.168.100.2 255.255.255.252
 ppp multilink
 ppp multilink interleave
 ppp multilink group 1
 service-policy output VOICE-POLICY
!
interface FastEthernet0/0
 ip address 192.168.200.1 255.255.255.252
 duplex auto
 speed auto
!
interface Serial0/0
 no ip address
 shutdown
 clock rate 2000000
 no fair-queue
!
interface Serial1/0:0
 bandwidth 128
 encapsulation ppp
 ppp multilink
 ppp multilink group 1
 max-reserved-bandwidth 80
!
ip http server
no ip http secure-server
!
ip route 0.0.0.0 0.0.0.0 192.168.100.1
!
access-list 102 permit udp any any dscp ef
access-list 102 permit udp any any precedence critical
access-list 103 permit ip any any dscp af31
access-list 103 permit ip any any precedence flash
snmp-server community bil678&%tel RW
!
control-plane
!
voice-port 2/0
 timeouts interdigit 3
!
voice-port 2/1
 timeouts interdigit 3
!
voice-port 2/2
 timeouts interdigit 3
```

```
!
voice-port 2/3
 timeouts interdigit 3
!
voice-port 2/4
 timeouts interdigit 3
!
voice-port 2/5
 timeouts interdigit 3
!
voice-port 2/6
 timeouts interdigit 3
!
voice-port 2/7
 timeouts interdigit 3
!
dial-peer voice 6 pots
 destination-pattern 312
 port 2/5
 forward-digits all
!
dial-peer voice 9 voip
 max-conn 1
 destination-pattern 212
 no modem passthrough
 session protocol sipv2
 session target ipv4:192.168.100.1
 dtmf-relay rtp-nte
 fax protocol t38 nse force ls-redundancy 0 hs-redundancy 0 fallback cisco
 no vad
!
dial-peer voice 5 pots
 destination-pattern 312
 port 2/4
 forward-digits all
!
line con 0
line aux 0
line vty 0 4
 password 1234
 login
!
end
```

ISTANBUL Gateway Konfigurasyonu:

```
ISTANBUL#sh run
Building configuration...

Current configuration : 2226 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname ISTANBUL
!
boot-start-marker
boot system flash flash:c2430-ik9o3s-mz.124-7.bin
boot system flash flash:c2430-is-mz.123-8.T11.bin
boot-end-marker
!
card type e1 1
enable secret 5 $1$KaGx$5bARWo/SUZoyrWqz2Z3Mn0
!
no aaa new-model
!
resource policy
!
network-clock-participate E1 1/0
!
no ip domain lookup
!
voice-card 0
!
controller E1 1/0
 channel-group 0 timeslots 1-2
!
class-map match-all voice-signaling
 match access-group 103
class-map match-all voice-traffic
 match access-group 102
!
policy-map VOICE-POLICY
 class voice-traffic
  priority 58
 class voice-signaling
  bandwidth 8
 class class-default
  fair-queue
!
interface Multilink1
 description 128kbps Ankara Hatti
 bandwidth 128
```

```
 ip address 192.168.100.1 255.255.255.252
 ppp multilink
 ppp multilink interleave
 ppp multilink group 1
 service-policy output VOICE-POLICY
!
interface FastEthernet0/0
 ip address 192.168.212.1 255.255.255.252
 duplex auto
 speed auto
!
interface Serial0/0
 no ip address
 shutdown
 clock rate 2000000
 no fair-queue
!
interface Serial1/0:0
 bandwidth 128
 no ip address
 encapsulation ppp
 ppp multilink
 ppp multilink group 1
 max-reserved-bandwidth 90
!
ip http server
no ip http secure-server
!
ip route 0.0.0.0 0.0.0.0 192.168.100.2
!
access-list 102 permit udp any any dscp ef
access-list 102 permit udp any any precedence critical
access-list 103 permit ip any any dscp af31
access-list 103 permit ip any any precedence flash
snmp-server community bil678&%tel RW
snmp-server enable traps tty
!
control-plane
!
voice-port 2/0
!
voice-port 2/1
!
voice-port 2/2
!
voice-port 2/3
!
voice-port 2/4
!
voice-port 2/5
!
```

```
voice-port 2/6
!
voice-port 2/7
!
dial-peer voice 6 pots
 destination-pattern 212
 port 2/5
 forward-digits all
!
dial-peer voice 9 voip
 destination-pattern 312
 no modem passthrough
 session protocol sipv2
 session target ipv4:192.168.100.2
 dtmf-relay rtp-nte
 fax protocol t38 nse force ls-redundancy 0 hs-redundancy 0 fallback cisco
 no vad
!
line con 0
line aux 0
line vty 0 4
 password 1234
 login
!
End
```

ÖZGEÇMİŞ

Doğum tarihi	08.03.1981	
Doğum yeri	Ankara	
Lise	1995-1998	Ankara Kurtuluş Lisesi
Lisans	1999-2004	Kocaeli Üniversitesi Mühendislik Fakültesi Elektronik ve Haberleşme Mühendisliği Bölümü
Yüksek Lisans	2004-2007	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Müh. Anabilim Dalı Haberleşme Programı

Çalıştığı kurum(lar)

2005-2006	Tesaş Telekomünikasyon A.Ş.
2006-Devam ediyor	Tellcom İletişim Hizmetleri A.Ş.