KOCAELİ ÜNİVERSİTESİ*FEN BİLİMLERİ ENSTİTÜSÜ

İNTERNET PROTOKOLÜ ÜZERİNDEN SES İLETİMİ VE BİR YAZILIM UYGULAMASI

YÜKSEK LİSANS TEZİ

Elektronik Mühendisi Nurten ERKAN

Anabilim Dalı: Bilgisayar Mühendisliği

Danışman: Doç. Dr. Yaşar BECERİKLİ

KOCAELİ, 2007

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

İNTERNET PROTOKOLÜ ÜZERİNDEN SES İLETİMİ VE BİR YAZILIM UYGULAMASI

YÜKSEK LİSANS TEZİ

Elektronik Müh. Nurten ERKAN

Tezin Enstitüye Verildiği Tarih: 04 Haziran 2007
Tezin Savunulduğu Tarih: 04 Temmuz 2007

Tez Danışması	Üye	Üye
Doç.Dr. Yaşar BECERİKLİ	Prof.Dr. Hasan DİNÇER	Prof.Dr. A.Coşkun SÖNMEZ
(.....)	(.....)	(.....)

KOCAELİ, 2007

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışmasında internet protokolü üzerinden ses iletimi (VoIP) hakkında kavramsal açıklama yapılarak, Microsoft Visual C++ geliştirme ortamı kullanılarak bir VoIP uygulamasının nasıl yapıldığı açıklanmaktadır. Gerçekleştirilen çalışmanın bu konuda çalışmak isteyenlere faydalı olmasını temenni ederim.
Zaman ve emek isteyen bu tür bir çalışmanın değişik insanların yardımları olmadan gerçeklenmesi mümkün olmayacaktır. Bu tezin hazırlanmasında özendirici ve titiz katkılarını esirgemeyen, yüksek bilgilerinden yararlanma olanağı sağlayan ve beni herzaman destekleyip yönlendiren çalıştığım danışman hocam Doç. Dr. Yaşar Becerikli'ye, çeşitli yöntemlerle cesaret ve destek veren genç mühendis arkadaşlarım Cüneyt Aksakallı, Ergün Uyar, Eda özüntürk, Murat Kuyluk ve Zafer Ozulu'ya , yüksek lisans süreci boyunca defalarca yol gösteren Türk Telekom A.Ş 'den mesai arkadaşlarım Abdullah Yıldırım, Hacer Atar Yıldız ve Süleyman Ergin'e, değerli dostlarım Ceylan Çoban Ayhan ve Selda Çolak'a teşekkür ederim. Ayrıca beni destekleyen Aileme ve Başkan ailesine sonsuz minnet duygularımı sunarım.

İÇİNDEKİLER

ÖNSÖZ VE TEŞEKKÜR i
İÇİNDEKİLER..... ii
ŞEKİLLER DİZİNİ v
TABLOLAR DİZİNİ..... ...vi
KISALTMALAR ii
ÖZET ix
İNGİLİZCE ÖZET x
1. GİRİŞ..... 1
2. GENEL BİLGİLER..... 5
2.1 .OSI Referans Modeli 5
2.1.1.Osi Katmanları..... 6
2.2.TCP/IP Referans Modeli..... 8
2.2.1.TCP/IP internet Protokolü..... 8
2.2.2.TCP /IP katmanları 9
2.3.IP Paket Formatı ve Datagram İletim Modu 12
2.3.1.IP paket formatı 12
2.3.2.Datagram iletim modu..... ..14
2.4.Şebeke Teknolojileri 15
2.4.1.Devre bağlaşmalı sistemler 15
2.4.2.Paket bağlaşmalı sistemler 16
3. İNTERNET PROTOKOLÜ ÜZERİNDEN SES İLETİMİ(VOIP)..... ..18
3.1.VOIP ‘in Tanımı ve Avantajları..... ..18
3.2.VOIP Sisteminin İşleyişi..... ..19
3.3.Genel Sıkıştırma Teknikleri 20
3.3.1.Lempel-Ziv kodlaması 21
3.3.2.Huffman kodlaması..... ..21
3.3.3.Dalgaşekli kodlaması 21
3.3.3.1.Diferansiyel kodlama 22
3.3.3.2.Diferansiyel PCM (DPCM) kodlama..... ..22
3.3.3.3.Adaptif DPCM (ADPCM) kodlama 22
3.3.3.4.Delta modülasyonu 23
3.3.3.5.Vektör kuantalama..... ..23
3.3.3.6.Dönüşüm kodlaması..... ..23
3.3.4.Ses kodlaması 24
3.3.5.Hibrit kodlama..... ..24
3.3.5.1.RELP (Residual Excited Linear Prediction)..... ..24
3.3.5.2.CELP (Codebook Excited Linear Prediction) 25
3.3.5.3.MPE ve RPE (Multipulse and Regular Pulse Excited) 25
3.4 .Ses Sıkıştırma Standartları 26
3.4.1.G.711 sıkıştırması 26
3.4.2.G.723 sıkıştırması 26

3.4.3.G.726 sıkıştırması ..26
3.4.4.G.728 sıkıştırması ..26
3.4.5.G.729 sıkıştırması: ..27
3.5.Serviz Kalitesi (Quality of Service -QoS) ..27
3.5.1.Servis kalitesini etkileyen unsurlar ..27
3.5.1.1.Gecikme (Delay) ..28
3.5.1.2.Jitter ..29
3.5.1.3.Yankı(Echo): ..29
3.5.1.4.Net çıkış hızı ..29
3.5.1.5.Paket kayıp oranı ..29
3.5.1.6.Gürültü ..30
3.5.2.Servis kalitesi (Quality Of Service-QoS) için kullanılan yöntemler ..30
3.5.2.1.Veri bağı katmanı servis kalitesi (Data Link Layer-QoS) ..30
3.5.2.2.Servis tipi (Type of Service-TOS) ..30
3.5.2.3.RSVP (Resource Reservation Protocol) servis kalitesi ..31
3.5.2.4.Farklılaştırılmış servisler ..31
3.6.VoIP' de Güvenlik ..31
4. VOIP PROTOKOLLERİ ..34
4.1.VoIP İşaretleşme Protokolleri ..34
4.1.1.Oturum başlatma protokolü (SIP) ..34
4.1.1.1.SIP bileşenleri ..36
4.1.1.2.SIP mesaj tipleri ..38
4.1.1.3.SIP adreslemesi ..39
4.1.1.4 SIP mesaj yapısı ..40
4.1.1.5. SIP oturum kurulum örnekleri ..41
4.1.2.H.323 protokolü ..44
4.1.2.1.H.323 protokol yığını ..45
4.1.2.2.H.323 bileşenleri ..46
4.1.2.3.H323 mesajları ..50
4.1.2.4.H.323 oturum kurulum örnekleri ..51
4.1.2.5.H.323. versiyonları ..55
4.1.2.5.1.H.323 Versiyon 2 (v.2) ..55
4.1.2.5.2.H.323 Versiyon 3 (v.3) ..56
4.1.2.5.3.H.323 Versiyon 4 (V.4) ..57
4.1.3.H.323 ve SIP protokollerinin karşılaştırılması ..57
4.1.3.1.Karmaşıklık ..58
4.1.3.2.Boyut ve performans ..60
4.1.3.3.Özellik karşılaştırması ..60
4.2.VoIP Veri Aktarım Protokolleri ..62
4.2.1.Kaynak Ayırma Protokolü (RSVP) ..63
4.2.1.1.RSVP'nin Çalışması ..63
4.2.1.2.RSVP mesajları ..64
4.2.1.3.RSVP başlık formatı ..64
4.2.1.4.RSVP nesne formatı ..65
4.2.1.5.RSVP nesne türleri ..66
4.2.2.Gerçek zaman taşıma protokolü (RTP) ..67
4.2.2.1.RTP mesaj formatı ..68
4.2.2.2.RTP başlık sıkıştırması (cRTP) ..69
4.2.3.Gerçek zaman kontrol protokolü (RTCP) ..70

4.2.3.1.RTCP paket türleri 71
4.2.3.2.RTCP mesaj formatı..... ..72
5.VOIP UYGULAMASI 74
5.1.Genel Bilgiler 74
5.1.1.Visual C++ geliştirme ortamı 74
5.1.2.TCP /IP ve soket mimarisi..... ..74
5.1.3.Microsoft temel sınıfları (MFC) 75
5.2.Sistem analizi ve uygulama aşamaları 76
5.2.1.Giriş..... ..76
5.2.2.Arayüzler ve oluşturulan formatlar..... ..76
5.2.2.1.Kullanıcı (Client) arayüzü 77
5.2.2.2.Sunucu (Server) arayüzü 78
5.2.2.3.Uygulamada kullanılan mesaj formatları 78
5.2.3.Uygulamanın çalışma akışı 79
5.2.3.1.Sunucu uygulaması 79
5.2.3.2.Kullanıcı (Client) uygulaması..... ..85
6.SONUÇLAR VE ÖNERİLER 96
KAYNAKLAR..... ..98
KİŞİSEL YAYINLAR VE ESERLER..... 103
EKLER..... 104
ÖZGEÇMİŞ 159

ŞEKİLLER DİZİNİ

Şekil 2.1:OSI referans modeli 6
Şekil 4.1:İki SIP uç birimi arasında oturum kurulumu 41
Şekil 4.2:Yönlendirme sunucusu kullanılarak oturum kurulumu 42
Şekil 4.3:Proxy Sunucusu kullanılarak oturum ı kurulumu 43
Şekil 4.4:H.323 bileşenleri 46
Şekil 4.5:İki H.323 uç arasında geçiş denetim sistemi kullanılmadan oturum kurulumu 51
Şekil 4.6:Geçiş denetim sistemi kullanılarak iki H.323 nokta arasında oturum kurulumu 53
Şekil 4.7:SIP ve H.323 protokolleri arasındaki bağlantının sağlanması 62
Şekil 4.8:RTP başlık sıkıştırması (cRTP) 70
Şekil 5.1:Kullanıcı (Client) arayüz formu (1) 77
Şekil 5.2:Kullanıcı (Client) arayüz formu (2) 77
Şekil 5.3:Sunucu (Server) arayüz formu 78
Şekil 5.4:Sunucu (Server) akış diyagramı 79
Şekil 5.5:Sunucu (Server) bağlantı şeması 85
Şekil 5.6:Kullanıcı (Client) akış diyagramı (1) 86
Şekil 5.7:Kullanıcı (Client) bağlantı arayüzü (1) 87
Şekil 5.8:Kullanıcı (Client) bağlantı arayüzü(2 91
Şekil 5.9:Kullanıcı (Client) akış diyagramı 2 91

TABLOLAR DİZİNİ

Tablo 2.1:OSİ katmanları, katmanların işlevleri ve kullandıkları protokoller 8
Tablo 2.2:TCP/IP katmanları ve katmanların kullandığı protokoller..... 12
Tablo 2.3:IP başlık formatı 13
Tablo 3.1:Kodlama standartları 27
Tablo 3.2:VoIP güvenlik tehditleri ve tavsiye edilen çözüm metotları..... 33
Tablo 4.1:SIP talep mesaj tipleri ve açıklamaları 38
Tablo 4.2:SIP cevap mesaj kodları ve açıklamaları 39
Tablo 4.3:H.323 protokol yığını..... 45
Tablo 4.5:H.323 ve SIP protokollerinin özellik karşılaştırılması..... 61
Tablo 4.6:Ortak RSVP Başlığı..... 64
Tablo 4.7:RSVP nesne formatı..... 65
Tablo 4.8:RTP mesaj formatı..... 68
Tablo 4.9:RTCP paket türleri..... 71
Tablo 4.10:RTCP Mesaj Formatı..... 72

KISALTMALAR

A.D.P.C.M:	Adaptive Differential Pulse Code Modulation
ADC:	Analog-to-Digital Converter
A.R.P:	Adress Resolution Protocol
A.T.M:	Asynchronous Transfer Mode
C.E.L.P:	Code Excited Linear Prediction
C.S.N:	Circuit Switched Network
C S R C:	Contributing Source Identifieri
DAC:	Digital –to-Analog Converter
D H C P:	Dynamic Host Configuration Protocol
D M:	Delta Modulation
DoS:	Denial Of Service
D.P.C.M:	Differential Pulse Code Modulation
D.S.L:	Digital Subscriber Line
D.S.P:	Digital Signal Processing
F.F.T:	Fast Fourier Transform
F.R:	Frame Relay
F.T.P:	File Transfer Protocol
GCP:	Gateway Control Protocol
GK:	Gatekeeper
GSM:	Global System For Mobile Communication
GW:	Gateway
HTML:	Hyper Text Markup Language
HTTP:	Hyper Text Transfer Protocol
ICMP:	Internet Control Messaging Protocol
IETF:	Internet Engineering Task Force
IGMP:	Internet Group Management Protocol
IN:	Intelligent Network
IP:	Internet Protocol
IPv4:	Internet Protocol Versiyon Four
ISDN:	Integrated Services Digital Network
ISO:	International Standarts Organization
ITU:	International Telecommunications Union
ITU-T:	ITU Telecommunication Standarts Department
JPEG:	Joint Picture Experts Group
LAN:	Local Area Network
LEC:	Local Exchange Carrier
MAC:	Media Access Control
MCU:	Multi Point Control Unit
MEO:	Middle Earth Orbit
MEGAO:	Media Gateway Control Protocol
MG:	Media Gateway
MGC:	Media Gateway Controller
MGCP:	Media Gateway Control Protocol

MPEG:	Mtion Pictures Experts Group
MPLS:	Multiprotocol Label Switching
NAT:	Network Address Translation
NMC:	Network Management Center
NTP:	Network Time Protocol
OLC:	Open Logical Channel
OPWA:	One Pass With Advertising
OSI:	Open System Interface
PBX:	Private Branch Exchange
PC:	Personal Computer
PCM:	Pulse Code Modulation
POP:	Post Office Protocol
PPP:	Point To Point Protocol
PSTN:	Public Switched Telephony Network
QoS:	Quality of Service
RADIUS:	Remote Authentication Dial In User Service
RAS:	Remote Access Server
RCF:	Request for Comments
RELP:	Residual Excited Linear Prediciton
RRJ:	Request Reject
RRQ:	Request to Register
RTP:	Real Time Transport Protocol
RSVP:	Resource Reservation Protocol
RTCP:	Real Time Control Protocol
SAP:	Session Announcement Protocol
SDH:	Synchronous Digital Hierarchy
SDP:	Session Description Protocol
SG:	Signalling Gateway
SIP:	Session Initiation Protocol
SMS:	Short Message Service
SMTP:	Simple Mail Transfer Protocol
SNMP:	Simple Network Management Protocol
SONET:	Synchourous Optical Network
SSRC:	Synchourszation Source Identifier
SS7:	Signalling System 7
TCP:	Transmission Kontrol Protocol
TCS:	Terminal Capability Set
UA:	User Agent
UAC:	User Agent Client Side
UAS:	User Agent Server Side
UDP:	User Datagram Protocol
URL:	Uniform Resource Locator
VoIP:	Voice Over Internet Protocol
WAN:	Wide Area Network
WWW:	World Wide Web

İNTERNET PROTOKOLÜ ÜZERİNDEN SES İLETİMİ VE BİR YAZILIM UYGULAMASI

Nurten ERKAN

Ahantar Kelimeler: VoIP, TCP, UDP, IP, SIP, PSTN, H.323 protokol, Soket programlama, MFC programlama.

Özet:İnternet protokolü üzerinden ses iletimi (VoIP-Voice over IP), günümüz devre anahtarlamalı telefon santralleri yerine sesin IP paketlerine dünüştürülerek tamamen IP temelli şebekeler üzerinden iletilmesidir. VoIP özellikle ekonomik açıdan sahip olduğu avantajlar nedeniyle şirketler için çok cazip bir ortam oluşturmaktadır. Bu tez kapsamında geliştirilen yazılım uygulaması günümüzde internet ve intranet kullanıcıların yoğun olarak kullandığı SKYPE,MSN gibi gelişmiş ses görüşmesi yapmayı sağlayan programların temel özelliklerine sahiptir. Uygulama Visual C++ geliştirme ortamında nesne tabanlı programlama teknikleri kullanılarak geliştirilmiştir. Sunucu (Server) –Kullanıcı (Client) tabanlı bir uygulamadır. Bilgisayardan –bilgisayara (PC to PC) ses görüşmesi yapmayı sağlar. Kullanıcılar sunucuya programın kullanıcı ara yüzünden bir isim yazarak bağlanırlar. Sunucuda tüm bağlanan kullanıcıların isim listesi tutulur. Sunucu yeni bir bağlantı olması durumunda tüm kullanıcılara güncellenmiş listeyi gönderir. Kullanıcılar gelen listeden istediği kişiyi seçerek ses görüşmesi yapar veya kendisini seçmiş olan bir kullanıcıdan gelen sesi alır. Uygulama kullanıcı hesabı sistemine dayanmadığı için tek oturumlutur. Kullanıcılara ait bilgiler sadece sistemde kaldıkları süre boyunca sunucuda tutulur. Sistemden çıktıklarında ise veritabanından silinirler. Uygulama denenmiş ve ses görüşmesi başarılı bir şekilde yapılmıştır. Bu uygulama gelişmiş ses görüşmesi yapmayı sağlayan programlara alternatif olarak kullanılabilir. Kullanıcılara ekonomik ve sınırsız görüşme imkanı sağlar. Uygulamamanın bağlantı kısmında TCP/IP haberleşmesi, iletim kısmında ise TCP protokolü kullanılmıştır. Tez kapsamında uygulama ve bu uygulamanın altyapısında kullanılan IP temelli ses iletimi hakkında açıklamalar, VoIP uygulamarında kullanılan protokoller çeşitli başlıklar altında detaylı bir şekilde incelenmiştir.

VOICE OVER INTERNET PROTOCOL AND
A SOFTWARE APPLICATION

Nurten ERKAN

Keywords: VoIP, TCP, UDP, IP, SIP, PSTN, H.323 Protocol, Socket programming, MFC programming.

Abstract: Voice Over Internet Protocol (VoIP) is the transmission of voice over completely IP based Networks, instead of today's circuit switching phone networks. VoIP especially forms a very attractive environment for companies, because of it's advantages. The software which was developed around the scope of this thesis, has the basic properties of programs like SKYPE, MSN, which provides having improved voice meetings and which used intensively by internet users. Application is improved in Visual C++ development environment by using object-oriented programming technics. It is a server-client based application. It provides voice meetings to be made from one pc to another. Users are connected to the server by entering a user name to the client interface. Name list of all connected users is stored on the server. Server sends the updated name list to all users when a new connection is established. Users choose the desired person from the list which has come, or receives the voice coming from the user who has choosen herself/himself. Application is single-sessioned since it is not based on user accounting system. Information of users is stored on server only when they are active on system. When they log out, they are deleted from the database. Application is tested and voice meeting has been succesfully made. This program may be used as an alternative to programs which provide improved voice meeting. It provides users economic and unlimited meeting opportunity to users. At the connection side of the application TCP/IP communication, and on the transmission side TCP protocol, is used. Explanations about the IP based voice transmission which is used in the infrastructure of the application, the application and the protocols which are used in VoIP applications are examined in detail under various headlines around the scope of this thesis.

1. GİRİŞ

İnternet üzerinden ses iletimi (VoIP), gerçek zamanlı sesin hem kaliteli hem de verimli bir şekilde iletimini sağlayan bir uygulamadır. Günümüz teknolojisinde geliştirilen bu tür yeni uygulamaların, eski uygulamalarla da uyumlu çalışması istenir. Bu bağlamda bir VoIP uygulaması, hem PSTN hatlar üzerinden hem de direk olarak başka bir VoIP uygulaması ile başarılı bir şekilde haberleşebilir [1].

İnternet üzerinden ses iletimini sağlayan ilk uygulamalar yeterince başarılı olamamıştır. IP üzerinden ses paketleri iletimi yapılıyorsa, sorunsuz bir iletişim için paketlerin zamanında ve doğru bir şekilde alıcı tarafa iletilmesi gerekir. Paket iletimi sırasında gecikme oluşursa iletişimde güçlük yaşanır. IP üzerinden ses iletiminde gecikme ve gecikmenin varyansı, konuşmanın kalitesini belirler. İnsan kulağı 150-200 ms'e kadar gecikmeleri hissetmeyebilir. Fakat bundan büyük değerdeki gecikmeler iletişim güçlüklerinin ortaya çıkmasına neden olur. Sorunsuz bir görüşme için bantgenişliğinin istenen düzeyde olması gerekir. VoIP ile ilgili ilk uygulamalarda bantgenişliği hesaplamaları yeterince uygun yapılamadığından, iletişimde bazı güçlüklerle karşılaşılmıştır. Günümüzde hem sıkıştırma yapan modüllerin gelişmesi hemde daha fazzla bantgenişliği sunan haberleşme sistemlerinin ortaya çıkmasıyla VoIP uygulamaları kurumların ihtiyaçlarını karşılayabilecek duruma gelmiştir.

VoIP teknolojinin gelişmesiyle, haberleşme sektöründeki firmalar rekabette geri kalmamak için VoIP çözümlerini sunmuşlardır. Ses iletimini devre –bağlaşmalı sistemler üzerinden yapan firmalar, VoIP çözümlerini eski sistemleriyle birleştirme yoluna gitmişlerdir. Gelecekte PSTN şebekelerinin yerini alacağına kesin gözüyle bakılan bu yeni teknoloji bir çok avantaja sahiptir. Bu avantajlar; İletişim yatırım maliyetlerinin çok büyük oranda düşmesi, bakım onarım giderlerinin düşmesi, mobiliteyi arttırması ve dünyanın her yerinden erişilebilirlik, kurulum maliyetlerinin düşmesi, operasyon maliyetlerinin düşmesi, yüksek güvenilirlik, gelişme ve

yeniliklere açıklık, varolan internet altyapısı üzerine kurulabilmesi, ses ve veri hizmetlerinin bir arada verilebilmesidir [2]. Günümüzde VoIP uygulamaları bir çok alanda kullanılmaktadır. [3]' de VoIP'in askeri amaçlı kullanımı anlatılmaktadır. Askeriye ülke genelinde birçok noktada konuşlanmış ve noktalar arasında kiralık hatlar ile birbirine bağlı intraneti olan gelişmiş bir iletişim ağına sahiptir. VoIP daha az ekipman gerektirmesi ve minimim PSTN kiralık hatta ihtiyaç duyması nedeniyle daha ekonomiktir. Bu nedenlerden dolayı VoIP askeri amaçlı olarak kullanılmaktadır. VoIP uygulamarı için temel ağ (network) bilgileri çok büyük önem taşımaktadır. [2,4]' de bu kavramlar ayrıntılı olarak açıklanmaktadır.

VoIP uygulamalarında kaliteli bir ses görüşmesi yapmak için bant genişliği tasarrufu önem arz etmektedir.Bu nedenle ses bilgisi sıkıştırma işlemine tabi tutulur. Sıkıştırma algoritmaları kullanılarak bant genişliğin kullanımı daha verimli hale getirililir. [5]'de sıkıştırma algoritmarı ayrıntılı şekilde anlatılmaktadır. VoIP uygulamaları işaretleşme ve iletim olmak üzere iki fazda gerçekleştirilir. VoIP'de işaretleşme fazında , ITU-T' nin geliştirmiş olduğu H.323 ile IETF' nin geliştirdiği SIP protokolleri kullanılmaktadır. SIP, H.323'e göre daha yeni bir protokol olduğu için, H.323 'ün cevap veremediği ölçeklenebilir ve karmaşıklık problemlerini ortadan kaldırmıştır.[6] 'de SIP protokolü hakkında ayrıntılı açıklamalar yapılarak, SIP işaretleşmesinin nasıl yapıldığı anlatılmaktadır. [7,8]'de ise daha karmaşık olduğu nitelendirilen H.323 protokolü hakkında geniş açıklamalar yapılarak H.323 protokolünü kullanan bir VoIP uygulamasında sinyalleşmenin nasıl yapıldığı anlatılmakdır. Günümüzde firmalar her iki protokolü birlikte kullanabilmektedirler. SIP ve H323 protolleri arasında geçiş (interworking) özelliği sağlanmıştır. [9]' da bu geçişin nasıl yapıldığı ayrıntılı açıklanmaktadır. Bu özellik sayesinde H.323 prokolülü ile kurulmuş ağ üzerinden görüşme yapmak isteyen VoIP kullanıcısı, SIP protokolü ile kurulmuş ağ üzerindeki bir kullanıcıyı arayabilir. VoIP sistemlerinde kullanılan bu iki protokolün arasındaki farklar [10,11]'de ayrıntılı olarak anlatılmaktadır. VoIP uygulamalarında kullanılan iletim protokolleri RTP, RSVP ve RTCP' dir. Kaliteli ses görüşmesi yapılmasında büyük önem taşıyan bu protokoller hakkında kapsamlı açıklamalar [12,13]' de bulunmaktadır.

VoIP uygumaları üç farklı yapıda gerçeklenmektedir. Bu yapılar; analog telefondan diğer bir analog telefonu arama (PSTN-to-PSTN), analog telefondan uzak mesafede

bulunan bir bilgisayar ile ses görüşmesi yapma (PSTN-to-PC) ve iki farklı bilgisayar arasında [PC-to-PC] ses görüşmesi yapmaktır. [14] 'da iki farklı bilgisayar arasında ses görüşmesi yapmayı sağlayan bir VoIP uygulaması anlatılmaktadır.

VoIP uygulamarı için servis kalitesi (quality of servis] çok önemlidir.VoIP'de amaç ses görüşmesi için ekonomik fayda sağlamak olsada kalite ön planda tutulmaktadır.[15]' da servis kalitesinin VoIP için önemi anlatılmakta ve kaliteli ses iletimi için yapılması gerekenler anlatılmaktadır. VoIP sistemleri için diğer önemli bir kavram işse güvenliktir. Güvenli ses iletimi için ağ sürekli kontrol altında tutulmalıdır. Bunun için mevcut ağ güvenlik çözümleri kullanılmalıdır. [16]'daVoIP sistemleri için oluşturulan tehditler ve bunların çözümleri ayrıntılı anlatılmaktadır.

Günümüzde internet ve intranet kullanıcıların yoğun olarak kullandığı GİZMO – ASTERİSK gibi gelişmiş ses görüşmesi yapmayı sağlayan yazılımlar bulunmaktadır. GİZMO, ASTERİSK ve türevi yazılımların çok farklı kullanım alanları var. Karşı tarafta da aynı yazılım kuruluyken internet üzerinden ücretsiz görüşme yapılır ve yerel telefonlar da (yurtiçi veya yurtdışı) normalden daha ucuza arababilir.Bu tür VoIP yazılımları geniş band internet hizmeti alan ve sık sık görüşme yapan kullanıcılar çok önemlidir.

GİZMO yazılımının temel özellikleri aşağıda belirtilmiştir.

1-Gizmo yazılımında internet'teki iki ucun herhangi bir yapısal bilgiyi birbirleri arasında eş zamanlı aktarmalarına olanak sağlayan açık bir XML protokol ve tekonoloji bütünü olan JABBER kullanılır.

2-Gizmo ile PC`den PC`ye ücretsiz olarak telefon görüşmesi yapılabilir.

3-Gizmo ile normal telefonlar (Sabit, GSM, Uydu) aranabilir.

4- Gizmo 4 dili desteklemektedir. (İngilizce, İspanyolca, Fransızca, İtalyanca)

5-Konferans görüşmeyi destekler. Bu sayede birden çok Gizmo kullanıcısı ve normal telefon kullanıcıları ile aynı anda görüşme yapılabilir.

6- Arama, konuşma ve yazılı mesajların kayıtları tutulur.

7-Standart mesaj ve ses görüşmesi yapan yazılımlarının özelliklerinin hepsini içerir. (Görüntü resmi ekleme, ifade kullanma gibi) [17].

ASTERİSK yazılımının temel özellikleri aşağıda belirtilmiştir.

1-Asterisk sesin işlenmesi ve taşınması için genel VoIP protokollerini destekler. (H.323, SIP, MGCP, SCCP)

2-Asterisk yazılımı geleneksel telefon sistemleri ve VoIP ile uyumlu çalışır.

3-Asterisk yazılımı bir çok işletim sistemi ile uyumludur.(Linux, Sun Solaris)

4-Asterisk yazılımı için lisans serbest ve kurulumu ücretsizdir.

5-Asterisk yazılımı VoIP için mevcut sistemlere ilave bir donanım gerektirmez.

6- Konferans görüşmeyi destekler. Bu sayede birden çok Asterisk kullanıcısı ve normal telefon kullanıcıları ile aynı anda görüşme yapılabilir [18].

Bu tezde internet protokolü üzerinden ses iletimi (VoIP) kavramı incelenmiştir. "Genel Bilgiler'' adlı ikinci bölümde OSI (Open System Interconnection) referans modeli hakkında bilgi verilmekte, daha sonra TCP/IP referans modeli ,IP protokolünün çalışma prensibi ve şebeke teknolojilerine değinilmektedir. Üçüncü bölümde ise, VoIP 'nin avantaj ve dezavantajları, sistemin işleyişi, ses sıkıştırma teknikleri, servis kalitesini etkileyen unsurlar ve VoIP'de güvenlik konularına değinilmiştir. Dördüncü bölümde, sesin iletimi esnasında işaretleşme ve veri aktarım fazlarının bulunduğu anlatılmakta ve bu fazlara ilişkin protokol tanımlamaları ve protokollerin karşılaştırılmaları yapılmaktadır. Beşinci bölümde ise, Microsoft Visual C++ geliştirme ortamı kullanılarak bir VoIP uygulamasının nasıl gerçekleştirildiği anlatılmaktadır. Altıncı ve son bölümde ise, VoIP sistemi ve yapılan uygulama çalışması kısaca özetlenmektedir.

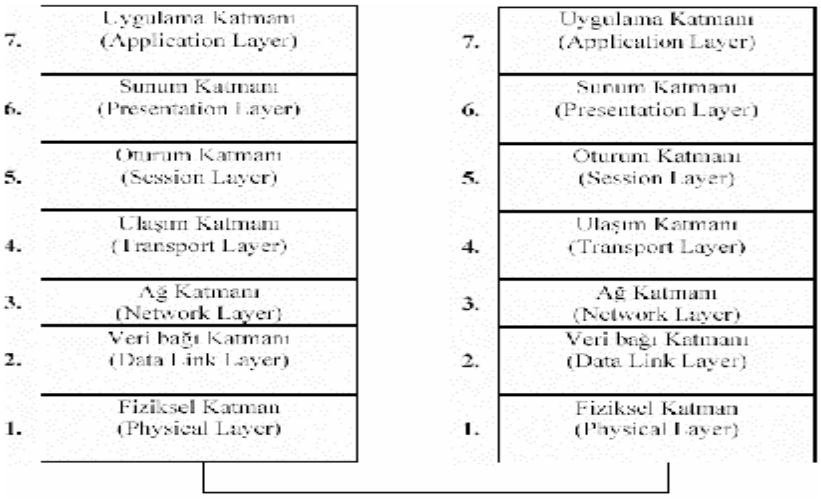
2. GENEL BİLGİLER

Bu bölümde İnternet üzerinden ses iletimi (VoIP) konusunun daha iyi anlaşılması için gerekli genel bilgiler verilecektir. Bu bilgiler OSI referans modeli, TCP/IP referans modeli, IP protokolünün çalışma prensibi ve şebeke teknolojileridir.

2.1 OSI Referans Modeli

Bilgisayarlar arası iletişimin başladığı günden itibaren farklı bilgisayar sistemlerinin birbirleri arasındaki haberleşme daima en büyük problemlerden birisi olmuş ve bu sorunun üstesinden gelebilmek için uzun yıllar boyunca çeşitli çalışmalar yapılmıştır. 1980'li yılların başında Uluslararası Standartlar Organizasyonu (International Standarts Organization-ISO) bilgisayar sistemlerinin birbirleri ile olan iletişiminde ortak bir yapıya ulaşmak yönünde çabaları sonuca bağlamak için bir çalışma başlatmıştır. Bu çalışmalar sonucunda 1984 yılında Acık Sistem Bağlantıları (Open Systems Interconnection-OSI) referans modeli ortaya çıkarılmıştır. Bu model sayesinde değişik bilgisayar firmalarının ürettikleri bilgisayarlar arasındaki iletişimi bir standarda oturtmak ve farklı standartlar arası uyumsuzluk sebebi ile ortaya çıkan iletişim sorununu ortadan kaldırmak hedeflenmiştir. OSI referans modelinde, iki bilgisayar sistemi arasında yapılacak olan iletişim problemini çözmek için 7 katmanlı bir ağ sistemi önerilmiştir [4]. Bu temel problem 7 adet küçük probleme parçalanmış ve her bir problem için ayrı ayrı bir çözüm yaratılmaya çalışılmıştır. Bu 7 katmanın en altında yer alan iki katman yazılım ve donanım, üstteki beş katman ise genelde yazılım yolu ile çözülmüştür. OSI modeli, bir bilgisayarda çalışan uygulama programının, iletişim ortamı üzerinden başka bir bilgisayarda çalışan diğer bir uygulama programı ile olan iletişiminin tüm adımlarını tanımlar. En üst katmanda görüntü ya da yazı seklinde yola çıkan bilgi, alt katmanlara indikçe makine diline dönüşür ve sonuç olarak 1 ve 0 lar'dan ibaret elektrik sinyalleri halini alır. Şekil 2.1

’de OSI referans modeli katmanları ve bir yerel ağ üzerindeki durumu gösterilmektedir.



Şekil 2.1: OSI referans modeli

2.1.1 Osi katmanları

Fiziksel katman: Bu katman ağın elektriksel ve mekanik karakteristiklerini belirler. Modülasyon teknikleri çalışma voltajı, çalışma frekansı bu katmanın temel özelliklerindendir. Verinin fiziksel bir haberleşme kanalında iletilmesini sağlar. Bitler, eğer kablolar üzerinden aktarılıyorsa elektriksel büyüklüklere; özellikle belirli gerilim değerlerine dönüştürülür. Eğer bilgi, hava veya boşluk gibi bir ortamdan iletilecekse, bu durumda bitler genellikle yüksek frekanslı taşıyıcı dalgaya genlik, frekans veya faz değişimi şeklinde bindirilerek iletilir [2,4].

Veri iletim katmanı: Bu katman fiziksel katmana verilerin doğru aktarılmasından sorumludur. Kendi içerisinde iki bölüme ayrılır. MAC (Media Access Control-Ortama Erişim Adresi) ve LLC (Mantıksal Yol Kontrolü). Fiziksel katmana aktaracağı veriye kendisine ve alıcıya ait MAC numarasını ekler. Değişken uzunluktaki veriye çeşitli uzunlukta 1’ler ve 0’lar ekleyerek çerçeve büyüklüğünü 64 bitte sabitler. Bu katmanda hem hata denetimi hem de hata düzeltmesi yapılır.

Ağ katmanı: Bu katman veri paketinin düzgün yönlendirilmesinden sorumludur. Routerlara gelen verilerin yönlendirme işlemi bu katmada yapılır. Yönlendirme protokollleri bu katmanda çalışır. Ağ içi veri akışı için routerda belirlenen tablolar vardır. Bu tablolar sabit veya değişken olabilir. Ağ içi trafiğin çok yoğun olduğu yerlerde dinamik bir yapı kullanılarak, verinin en kısa zamanda hedefine ulaşması sağlanır. Normal ağ koşullarında bu tablolar sabittir ve veri paketlerinin gideceği yol önceden belirlenmiştir. Ağ içerisindeki veri trafiği bu katman tarafından düzenlenir. Olası veri çakışmalarına karşı önlem alır. IP protokolünün çalışma alanıdır.

Taşıma katmanı: Bu katman, bilginin uçtan uca (end to end) doğru bir şekilde iletilmesinden sorumludur. Burada akış kontrol mekanizmaları devreye girer. Veri paketleri doğru şekilde yerine ulaşmazsa tampon bellekte (buffer) tutulan veri paketleri tekrardan gönderilir. Ayrıca bu katman kendisine gelen veri eğer doğru alınmışsa karşıdaki bilgisayara ya da ağ aygıtına bir onay mesajı da göndermekle yükümlüdür. TCP protokolü bu katmanda çalışır. Ağ içerinde bağlantıların kurulumu ve sona erdirilmeside bu katmanda yapılır.

Oturum katmanı: Bu katman birden fazla bilgisayarın aynı anda sorun yaşamadan anlaşmasını sağlar. Soket protokolleri bu katmanda çalışır. Ayrıca oturum katmanı iletişimin kurulması için kendisine verinin ulaşmasının ardından işaretleşme işlemini başlatır.

Sunum katmanı: Bu katman verinin formatına karar verir. Bilginin karakter set çevrimi veya değiştirilmesi, şifrelenmesi bu katmanda yapılır. Burada sentaks ve şematik kontrol mekanizmaları çalışır. Protokollerin birbirleriyle haberleşebilmeleri için birbirlerinin dilinden anlamaları gerekmektedir. Bunun için protokoller birbirlerinin diline bu katmanda çevrilirler. Ayrıca çeşitli sıkıştırma algoritmaları da (Huffman gibi) bu katmanda kullanılır.

Uygulama katmanı: Konum olarak en yukarıda bulunan katmandır. Kullanıcıların çalıştırdıkları programlar ve bu programların çalıştırdıkları protokoller bu katmanda bulunurlar. (Browser ftp bağlantı isteği, server uygulamaları). Bu katman, kullanıcı programlarının, üzerinde çalıştığı makineden ve alttaki katmanlardan bağımsız

olmasına izin verir**.** Telnet, http, ftp , smtp protokoleri bu katmanda çalışır. Tablo 2.1 ’de osikatmanları, katmanların işlevleri ve kullandıkları protokoller gösterilmektedir.

Tablo 2.1: OSİ katmanları, katmanların işlevleri ve kullandıkları protokoller

Katman	İşlev	Protokoller
7.Uygulama	Kullanıcıların çalıştırdıkları programlar bu katmanda bulunur.	Telnet,http,www,ftp
6.Sunum	Verinin nasıl sunulacağına karar verilir. Şifreleme gibi işlemler yapılır.	JPEG, ASCII, EBCDIC MPEG,
5.Oturum	Veriyi farklı uygulamardan ayri tutar.	RPC, SQL, Apple Talk, ASP
4.İletim	Güvenilir ya da güvensiz iletim yapılır. Çoğullama (multiplexing) yapılır.	TCP, UDP, SPX
3.Ag	Router’ ların yol seçiminde kullandıkları mantıksal adresleme yapılır.	IP, IPX, ICMP
2.Veri bağlantı	Bitler byte, byte lar çerçeve halinegetirilir. MAC adresleri kullanılır.	HDLC, Frame Relay, PPP FDDI, ATM
1.Fiziksel	Bitlerin cihazlar arası iletilmesi. Voltaj, iletim hızı ve kablo pinlerinin kullanımı	Ethernet,V35

2.2 TCP/IP Referans Modeli

2.2.1 TCP/IP internet protokolü

TCP/IP’nin ilk ortaya çıkışı 1960’ların sonunda ve 1970’lerin başında Amerikan Savunma Bakanlığına bağlı İleri Araştırma Projeleri Ajansının (Advanced Research Projects Agency, ARPA) yürüttüğü paket anahtarlamalı ağ deneylerine kadar uzanır. TCP/IP’nin yaratılmasını sağlayan proje, ABD’deki bilgisayarların bir felaket anında da ayakta kalabilmesini, birbiriyle iletişiminin devam etmesini amaçlıyordu.

TCP/IP genelde bir bilgisayar ortamında iki program arasında iletişim kurulması için kullanılan bir protokoldür. Burada programlar Sunucu (Server) ve Kullanıcı (Client) olarak iki ayrı konumda çalışır. Programlardan biri gelen bilgiyi işleyip diğer programa göndermeyi gerçekleştirirken, bilgiyi alan program, işlenmiş bilgiyi görüntüler ya da kullanır .

İletim Kontrol Protokolü/Internet Protokolü (Transmission Control Protocol/Internet Protocol TCP/IP) bir protokoller kümesidir. Her biri değişik işler yapan bir yığın protokolden oluşur [1].

TCP/IP ile kurulan bir bilgisayar ağında bir bilgisayar üç parametre ile tanımlanır. Bu parametreler: Bilgisayarın ismi, IP adresi ve MAC adresi (Media Access Control-Ortama Erişim Adresi)'dir. TCP/IP protokoller kümesi bu üç parametreyi kullanarak bilgisayarları birbirine bağlar.

Bilgisayarın ismi kullanıcı tarafından işletim sistemine yüklenirken bilgisayara verilen addır. IP adresi ise 172.43.126.99 örnek adresinde olduğu gibi dört bölmeden oluşan bir adrestir. Nokta ile birbirinden ayrılan bu bölmelerin her biri 0 (sıfır) ile 255 arasında bir değer alabilir. MAC adresi, bilgisayarların ağ kartının ya da benzer cihazlarının içine değiştirilmez bir şekilde yerleştirilmiş bulunan bir adrestir. 0023BCC-9F881 örneğinde olduğu gibi onaltılık düzende (Hexadecimal) rakamlardan oluşur [4].

Ağ üzerinde iletişim aslında yalnızca MAC adresleri ile gerçekleşir. Çünkü IP adresleri TCP/IP protokolüne özeldir. Başka bir protokolde, örneğin, Novell'in kullandığı IPX/SPX protokolünde IP adresi diye bir şey yoktur. Bütün protokollerde değişmeden kalan tek şey MAC adresidir. Bir bilgisayar bir başka bilgisayarın IP adresine sahipse ama MAC adresine sahip değilse Adres Çözümleme Protokolü (Address Resolution Protocol, ARP) adı verilen bir protokol kullanarak IP adresini MAC adresine çevirir. ARP protokolü TCP/IP protokol kümesinin bir üyesidir [1,19].

2.2.2 TCP /IP katmanları

TCP/IP Protokolü 4 farklı katmandan oluşmaktadır.

Uygulama katmanı: Ağ üzerinden iş yapacak uygulamaların bulunduğu katmandır. Bu katmanda HTTP (HyperText Transfer Protocol), SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol), SNMP (Simple Network Management Protokol) ve Telnet gibi protokoller çalışır.

HTTP: İnternet'te web sayfaların içeriğini görüntülemek için kullanılır.
FTP: İnternet'te hızlı dosya transferi yapılması için geliştirilmiştir.
SNMP : Belirli bir şebeke üzerindeki cihazların birbirleriyle haberleşmelerini ve ortak bir dil kullanmalarını sağlamak için geliştirilmiştir.
SMTP : İnternet üzerinden e-mail göndermeye yarayan bir protokoldür.
Telnet: İnternet ağı üzerindeki çok kullanıcılı bir makineye uzaktaki başka bir makineden bağlanmak için geliştirilen bir protokoldür..

Taşıma katmanı: İki bilgisayar (host) arasındaki tüm iletişimin kurulması ve oturumun açılmasını sağlayan katmandır. Bu katmanda iki adet protokol çalışır. Bunlar TCP (Transmission Control Protocol) ve UDP (User Datagram Protocol) olarak adlandırılan protokollerdir [20].

TCP: Bağlantılı ve güvenilir bir iletişim sağlar.Bir üst katmandan gelen verileri bölümlere (segment) ayırır.Her bölüme bir numara verirlir ve gönderirlir. TCP bölümlerinin karşı makinaya ulaşıp ulaşmadığını kontrol eder.Eğer bölüm(segment) ulaşmadıysa yeniden aynı segment gönderilir.

UDP: Bu protokol TCP'nin yaptığı denetimleri yapmaz. Dolayısıyla veri alışverişi çok daha hızlı gerçekleşir. Garantili veri hizmeti sunmaz. Bağlantı kurma ve çözme süreci olmadığı için bu süreçlerden kaynaklanan gecikmeler de oluşmaz. Dolayısıyla gerçek zaman uygulamalarında tercih edilir [1].

İnternet katmanı: İletim katmanından gelen verilerin IP paketleri haline dönüştürülüp yönlendirilmesinden sorumludur. Ayrıca bilgisayarlar arasındaki iletişimin nasıl gerçekleşeceğinden de sorumludur. Bu katmanda çalışan başlıca protokoller dört tanedir. Bunlar IP(Internet Protocol), ARP(Adres Resolution Protocol), ICMP

(Internet Control Message Protocol) ve IGMP (Internet Group Management Protocol)'dir [21].

IP: Ağ üzerindeki herhangi bir cihaza ulaşacağı zaman sanal bir adres üretilir. Paket bağlaşmalı şebeke sistemlerinin birbirlerine bağlanabilmesi için tasarlanmıştır. IP paketlerine datagram denir. Bu protokolde paketlerin bölünmesi ve tekrardan birleştirilmesi de tanımlıdır [4].

ARP: Bir ağdaki makinelerin sabit MAC adreslerinin belirlenmesinde ve bu makinelerin birbirleriyle konuşurken MAC adreslerinin kullanımlarını sağlayan kurulları içeren bir protokoldür [22].

ICMP: Verilerin yerine ulaşıp ulaşmamasıyla ilgili mesajların yollanmasından sorumlu olan protokoldür. İletim esnasında oluşacak sorunlarda ICMP protokolü devreye girerek gelen verinin bozuk olduğunu belirtir. Böylece verinin tekrar gönderilmesi için gerekli olan mekanizma çalıştırılmış olur.
IGMP: Bu protokol bir veri paketinin birden çok bilgisayara gönderilmesi işleminden (multicasting) sorumludur.

Fiziksel katman : Bu katman bilgisayarlarda bulunan ağ kartını, kablolar v.b gibi cihazları göstermektedir. Verilerin elektriksel işaretler yoluyla çeşitli iletim ortamlarından fiziksel olarak aktarılmasını sağlar. ATM ya da Ethernet gibi protokoller bu katmanda çalışırlar. Ayrıca MAC adreslerinin nasıl kullanılacağı da bu katmanda tanımlanmıştır [20].

ATM (Asynchronous Transfer Mode): Ses, veri, video gibi değişik türde bilgilerin aynı ortamdan aktarılmasını sağlayan bir anahtarlama teknolojisidir. Veri aktarımında hücre olarak adlandırılan küçük boyutlu ve sabit uzunluklu veri paketleri kullanılmasıdır. ATM bağlantıya yönelik bir aktarım protokolüdür. İki nokta arasında akratım yapılabilmesi için öncelikle bu noktalar arası bir bağlantı kurulur ve veri paketleri bu yol üzerinden gönderilir. Kullanılan hücreler 53 byte uzunluğundadır. Hücrelerin sabit uzunlukta olması hızlı ve karmaşık olmayan ATM anahtarlama cihazlarının tasarlanmasına imkan vermiştir [23].

Ethernet: Lokal alan şebekelerinde çok sık kullanılan bir teknolojidir. Ethernet standardı10 Mbit/sin'lik hıza izin verirken, Fast Ethernet standardı 100Mbit/sn' lik hıza ve Metro Ethernet standartdı 1 Gbit/sn ' lik hıza izin verir [23].

Tablo 2.2 ' de TCP/IP katmanları ve katmanların kullandığı protokoller gösterilmektedir.

Tablo 2.2: TCP/IP katmanları ve katmanların kullandığı protokoller

<table>
<tr><td>Katmanlar</td><td colspan="6">Protokoller</td></tr>
<tr><td>Uygulama</td><td>Smtp</td><td>Snmp</td><td>ftp</td><td>http</td><td>Tftp</td><td>Telnet</td></tr>
<tr><td>Taşıma</td><td colspan="2">TCP</td><td colspan="4">UDP</td></tr>
<tr><td>İnternet</td><td colspan="3">IP</td><td colspan="3">ICMP</td></tr>
<tr><td>Fiziksel</td><td colspan="6">Ethernet, ATM ,X25, Token-Ring, Dial-Up</td></tr>
</table>

2.3 IP Paket Formatı ve Datagram İletim Modu

2.3.1 IP paket formatı

IP protokolü ile gönderilen bir paket, başlık bilgisi ve onu takip eden veri bilgisi içerir. Tablo 2.3 'de IP başlık formatı gösterilmiştir. Burada, en anlamlı bir sıfır numaralı soldaki bit, en anlamsız bit ise otuz bir numaralı sağdaki bittir.

Tablo 2.3 : IP başlık formatı

Versiyon	IBU	Servis Tipi	Toplam Uzunluk	
Kimlik			Bayrak	Bölümleme Ofseti
Yaşam Süresi	Protokol		Başlık Kontrolü	
Kaynak IP Adresi				
Hedef IP Adresi				
Seçime Bağlı (Zorunlu Değil)				

Versiyon alanı : İnternet protokolünün versiyonunu belirtir. Günümüzde kullanılan IP versiyonu, versiyon dörttür. Bu alan sayesinde farklı versiyonlar bir arada bulunabilir ve yeni bir versiyona geçmek kolaylaşır [3,24].

IBU (İnternet başlık uzunluğu) alanı: İnternet başlık bilgisini içerir. Bu alan başlık bilgisini 32 bitlik kelimeler şeklinde belirtir. 4 bitlik bir değer olduğundan, başlık bilgisinin maksimum uzunluğu 16 oktet (kısım) olacaktır. Ayrıca başlık bilgisinin zorunlu doldurulması gereken alanı 5 oktet olduğundan, alabileceği en küçük değer 5 oktet olur.

Servis tipi alanı: Servis kalitesini (QoS) sağlamak amacıyla kullanılır. Fakat pratikte çok sık kullanılmaz. Ancak ses verisinin gerçek-zamanlı iletilmesi söz konusu olduğunda uçtan-uca gecikmenin önemsenmesi gerekebilir.Bu durumlarda kullanılır.

Toplam uzunluk alanı: IP datagramın büyüklüğünü gösterir. 16 bitlik bir alan olduğundan alabileceği en büyük değer 65535'tir. Ağlar genellikle maksimum büyüklükteki datagramları işleme yeteneğinde değildir. Bu nedenle gerçek uygulamalarda datagram büyüklüğü genellikle bu değerden oldukça düşüktür. Ancak tüm bilgisayarların en az 576 oktet'lik datagramları alabilmeleri ve gönderebilmeleri gerekir.

Kimlik alanı: Büyük olduğu için parçalanan datagram parçalarının tekrar birleştirilmesi için kullanılır. Bölünmüş her datagram parçası aynı değeri alır. IP datagramları gönderildiği zaman, bu değer bir artırılır.

Bayrak alanı:3 bitlik bir alandır. Birinci bit rezerve edilmiştir ve değeri sıfır olmalıdır. İkinci bit "bölümleme yapma" ve son bit "bir anda parçala" şeklindedir. Eğer bir datagram büyük olduğu için bir ağdan iletilemiyorsa DF (don't fragment) biti bir ise hata oluşur ve hata ilgili tarafa iletilir.
Bölümleme Ofseti alanı, İnternet katmanının bölünmüş datagramları tekrar birleştirmesini sağlar. Bu 13 bitlik değer orijinal datagramın bölümleme ofsetini gösterir.

Yaşam süresi(TTL) alanı:Bir datagramın gerektiğinde ağı daha fazla meşgul etmemesi için yok edilmesini belirtir.

Protokol alanı:Datagram içindeki verinin hangi protokolü kullandığını belirtir. Bu taşıma katmanı protokolü olabileceği gibi, internet katmanının kontrol protokollerinden biri de olabilir.

Başlık kontrolü alanı: Datagramın doğru bir şekilde ulaşıp ulaşmadığının kontrolü için kullanılır.

Kaynak IP adres alanı: Gönderen tarafın IP adresini gösterir.

Hedef IP adres alanı:Alıcı tarafın IP numarasını gösterir.

Seçime bağlı alanlar: Zorunlu değildir. Seçime bağlı olarak kullanılabilir.

2.3.2 Datagram iletim modu

Temel olarak bir IP datagramının iletilebileceği üç iletim modu vardır. Bunlardan ilkinde sadece bir kaynak ve bir hedef vardır. Kaynak hedefe IP datagramlarını yollar (Unicast).

İkinci olarak, bir kaynak belirli bir ağdaki tüm kullanıcılara bir veri göndermek istediğinde kullandığı yöntemdir (Broadcast).

Diğer iletim modu ise bir veri bir ağdaki bazı kullanıcılara gönderilmek istendiğinde (ağdaki herkese değil) uygulanan yöntemdir (Multicasting). Bir IP datagramı bir kullanıcı grubuna gönderilmek istendiğinde bir yöntem olarak, kopyalanıp tüm hedeflere gönderilebilir. Ancak bu yöntem çok fazla kaynağın gereksiz yere kullanılmasına neden olur. Bunun yerine bir datagram bir yoldan gönderilirken sadece yol üzerinde farklı bir yöne saptığı zaman kopyalanıp ilgili hedeflere gönderilmesidir. Günümüzde kaynak kullanımın önemi artığından IP ağları bu ikinci yöntemi kullanmaktadır. Bant genişliğinden tasarruf edilmiş olunur [21].

2.4 Şebeke Teknolojileri

Günümüzde ses haberleşmesi devre bağlaşmalı ve paket bağlaşmalı sistemler üzerinden yapılabilmektedir. Aşağıda bu bağlaşma tekniklerinin özellikleri kısaca açıklanacaktır [25].

.

2.4.1 Devre bağlaşmalı sistemler

1876'da telefonun icadı, konuşmanın analog elektriksel işaretlerle temsil edilmesi yoluyla uzaktan haberleşmeyi sağlamıştır. Telefon haberleşmesinin çok hızlı yaygınlaşma eğilimi, telefon santrali denilen bağlaşma merkezlerinin kurulmasını gerektirmiştir.

Devre bağlaşma tekniğinde, bir veya bir çok seçici üzerinden geçerek, kullanıcılar arasında uzayda, zamanda veya frekans döneminde uçtan-uca bir bağlantı kurulur. Haberleşme, taraflar arasında yol kurulunca başlar ve kurulan yol görüşme süresince tutulur. Oturumun sonunda bağlantılar çözülür ve imkanlar başka kullanıcılara tahsis edilir.

Devre bağlaşmasında haberin depolanması söz konusu değildir. Gerçek zamanda haberleşilmekte ve iletimdeki gecikme sadece elektriksel işaretin yayılma hızının sınırlı olmasından doğmaktadır.

Devre bağlaşmasında bağlaşma organlarının sayısal olması veya başka bir deyişle sayısal devre bağlaşması ancak 1960'lardan sonra pratik hale gelmiştir. Bugün sayısal devre bağlaşma sistemlerinin hemen hemen tamamını sayısal telefon santralleri oluşturmaktadır. Yeni kurulan telefon santrallerinin hemen hemen hepsi sayısal santrallerdir.

Devre bağlaşmasının klasik uygulama alanı, telefon haberleşmesidir. Çünkü devre bağlaşmasında kurulan yol oturum boyunca tutulduğundan, yol kurma ve çözme süresinin oturum süresinden çok küçük olduğu durumlar için bu teknik, elverişli olmaktadır. Telefon haberleşmesinde yukarıda bahsedilen koşul sağlanır [2].

Sayısal devre bağlaşma tekniğinin yaygın olarak uygulanabilmesi için en büyük haberleşme hacmini oluşturan telefon haberleşmesinde konuşma işaretlerinin sayısallaştırılması söz konusu olmuştur. Sayısallaştırma tekniğinin esasları 1930'lardan beri bilindiği halde, bu teknik ancak oldukça hızlı çalışan sayısal devrelerin gerçekleştirilebilmesinden sonra yaygın olarak uygulanabilmiş ve sayısal santraller 1970 ortaklarından itibaren telefon şebekesi içerinde yer almaya başlamıştır.

2.4.2 Paket bağlaşmalı sistemler

1964 yılında, bilgisayarlarda ve terminallerde üretilen verinin amaçlanan uç noktasına gönderilmesinde o güne kadar kullanılan bağlaşma tekniklerinin elverişsiz olması nedeni ile yeni bir bağlaşma tekniği ortaya atılmıştır. Bu yeni teknik veri trafiğinin patlamalı (bursty) tipten olması halinde, iletim yolunu gerek duyulduğu anda bu trafiğe tahsis etmesi sebebiyle ekonomik bir çözüm getirmiştir. VoIP uygulamalarının temel çıkma nedeni ekonomik olmasıdır. Paketlenerek IP ağları üzerinden yapılan ses görüşmeleri kullanıcılar tarafında için her zaman tercih edilmektedir. Uzun sakin aralıkları izleyen, bazen birkaç saniye yada daha kısa

süren patlamalı tipten veri trafiğinin bağlaştırılmasında, bağlantı kurma-çözme süresi, oturum süresinden kısa olmadığından devre bağlaşma tekniği uygun olmamaktadır. Bu durumda paket bağlaşma tekniği uygun olmaktadır.Bilgisayar ve ağ teknolojilerinin gelişmesiyle, paket bağlaşma tekniği oldukça değer kazanmıştır [6].

3. İNTERNET PROTOKOLÜ ÜZERİNDEN SES İLETİMİ (VoIP)

3.1 VoIP 'in Tanımı ve Avantajları

VoIP, normal telefon sistemindeki sesin IP tabanlı internet üzerinden aynı ses kalitesinde, güveninirliğinde ve özelliklerinde taşınmasıdır. VoIP'de, DSP (Digital Signal Processing – Dijital sinyal prosesi) segmantleri ile ses sinyali frame'lere çevrilir ve belli bir gurup oluşturduklarında ses paketlerine çevrilir ve IP tabanlı bir şebeke üzerinden gönderilir [26].

VoIP uygulamaları üç türlü gerçekleştirilmektedir.
1.Geleneksel uluslararası ve uzun mesafe telefon hizmetlerinde (Telefon-PC)
2.Geleneksel telefon şebekesi ile bilgisayar ve İnternet tabanlı telefon hizmetlerinde (Telefon-PC)
3.Tamamen bilgisayar ve internet tabanlı telefon uygulamalarında (PC-PC)

PSTN olarak isimlendirilen klasik bildiğimiz telefon şebekesinde analog ses sinyalleri ve işaretleşme olarak ise CCSS (Common Channel Signalling System) kullanılmaktadır. Internet tarafı ise IP tabanlı bir network olup IP datagram olarak adlandırılan sayısal veri protokolü kullanılmaktadır. Kullanılan verilerin ve protokollerin farklı olması dolayısıyla PSTN ile Internet network'ü arasında Ağgeçidi (Gateway) kullanılmaktadır. Ağgeçidi PSTN network' ünden aldığı ses ve CCSS sinyalleşme bilgilerini IP protokolüne dönüştürmektedir. Aynı şekilde Internet network'ünden aldığı IP paketlerini ise ses ve CCSS işaretleşme bilgilerine dönüştürmekte ve PSTN şebekesine göndermektedir.

Gelecekte PSTN şebekelerinin yerini alacağına kesin gözüyle bakılan VoIP teknolojisi bir çok avantaja sahiptir. Aşağıda bu avantajlar maddeler halinde açıklanmıştır [2].

1.Düşük destek maliyeti , sadece standart temelli bir network maliyeti gerektirir.

2.Potansiyel telefon görüşmelerinden tasarruf, herkese açık telefon şebekesi imkanı sağlar.

3.Daha büyük esneklik - insanlar, aldıkları servisler kesilmeksizin ofisler arasında ses görüşmesi yapabilir ve çok düşük miktarda kesinti ile karşılaşırlar.

4.Birleşik mesajlaşma ve ilişki merkezleri gibi yeni üretken servislere erişim sağlar.

5.Daha yüksek seviyelerde ölçeklenebilirlik ,yeni kullanıcılar hızla ve kolaylıkla eklenebilme imkanına sahip olurlar.

6.Daha ucuz ve daha az bant genişliği kullanılarak yapılır.

7.Veri servisleriyle gerçek zamanlı ses iletim servislerinin birleştirilebilmesi imkanı sağlar [27].

3.2 VoIP Sisteminin İşleyişi

VoIP sisteminin işleyişinin başlangıcında, ses paketlerinin oluşturulabilmesi için öncelikle analog ses işaretinin sayısal forma dönüştürülmesi gereklidir. Bu olay sayısal analog çeviriciler sayesinde yapılabilmektedir ve bunun için kullanılabilecek en basit donanım ise bilgisayarlarda kullanılan ses kartlarıdır. Bu sayısallaştırma işlemi PCM (Darbe Kod Modulasyonu) adıyla da anılır. Bu işlem üç aşamadan oluşur. Birinci aşama örnekleme aşamasıdır ve sürekli ses işaretinden belirli aralıklarla örnekler alınır. İkinci aşama, alınan bu darbe şeklindeki ses örneklerini belirli değerlere yuvarlama aşamasıdır ki bu aşamaya kuantalama da denir. Üçüncü ve son aşama da, daha önceden belirlenmiş genlik değerlerine ötelenmiş sayısal ses

işaretinin her genlik seviyesi için ikili kodlarla (binary) olarak kodlanmasıdır. Böylece PCM işaret elde edilmiş olur. VoIP sistemlerinde çoğu zaman 4 bitlik kod kelimeleri kullanılır.PCM işaretler daha sonra ITU-T'nin G.764 tavsiyesine uygun olarak paketlenirler ve karşı bilgisayara iletilirler. Alıcı tarafta ise tersi işlemler yapılarak ses bilgisi alınır. Band genişliğinden kazanmak için ABD ve Japonya'da μ-Kuralı, Avrupa'da ise A-kuralının kuantalama şemaları kullanılır. Bu şemalar ITU-T'nin G.711 tavsiyesine uygundur. Alıcı tarafta bu sıkıştırılmış ses işaret çözülür. G.711 haricinde Huffmann sıkıştırması, G.723 ve G.729 gibi çeşitli sıkıştırma teknikleri de band genişliğini verimli kullanmak amacıyla tercih edilir [25,28]. VoIP uygulamalarında ses bilgisi analog sayısal dönüştürücü yardımı ile sayısala dönüştürülür. Sıkıştırma algoritmaları kullanılarak sıkıştırılan ses bilgisi VoIP protokolleri yardımıyla internetten gönderilir. Daha sonra alıcı tarafta protokol ayıklanması yapılır. Ses açma algoritmaları kullanılarak sıkıştırılmış ses bilgisi açılır. Sayısal analog dönüştürücü yardımı ile ses bilgisi analog hale dönüşür. Bu işlemlerden ses bilgisi alınmış olur. Aşağıda VoIP sisteminin işleyi gösterilmektedir:

3.3 Genel Sıkıştırma Teknikleri

Bir yönlü ses iletişimi için 64kb/s' lik bir bant genişliği gereklidir. Bir yerel alan ağında (LAN) bu çok fazla problem yaratmaz. Fakat dial-up bağlantılar göz önüne alındığında bu bit oranının sağlanması mümkün değildir. Bir yerel alan ağında bile, kullanıcı sayısı fazla ise ağ aşırı yüklenebilir. Bir geniş alan ağı (WAN) 'da bit oranının her zaman sağlanamaz.. Yavaş olan bir bağlantı, tüm sistemin iletişimini bozar [5].

Bu nedenle VoIP uygulamalarında sıkıştırma bir gerekliliktir ve ses bilgisi yüksek sıkıştırma oranlarını destekler. Bu bölümde bazı sıkıştırma tekniklerine değinilecek ve VoIP için önemi vurgulanacaktır.

Bu kısımda Lempel-Ziv ve Huffman kodlama gibi oldukça geniş bir şekilde kullanılan sıkıştırma teknikleri açıklanacaktır. Bunlar kayıpsız sıkıştırma

teknikleridir. Sıkıştırılıp gönderilen bilgi, alıcı tarafta çözüldüğünde, orijinal veri tekrar aynı şekilde elde edilir. Bu nedenle bu sıkıştırma teknikleri hem veri hem ses için kullanılabilir. Bu yöntemler doğrudan uygulandığından fazla bir sıkıştırma oranı sağlanamaz. Bu nedenle sıkıştırma genelde bu yolla yapılmaz. Diğer bazı sıkıştırma teknikleri uygulandıktan sonra, bu sıkıştırma teknikleri kullanılır [29].

3.3.1 Lempel-Ziv kodlaması

Birkaç farklı Lempel-Ziv (LZ) sıkıştırma algoritması vardır. Fakat genellikle aynı prensibe göre çalışırlar. LZ ailesi algoritmalar bir karakter serisinin yerine sabit uzunlukta kod yerleştirir. Bu ailenin iki önemli üyesi LZ77 ve LZ78 algoritmalarıdır.
LZ78 algoritması bir "sözlük" kullanılır. Bu karakter dizilerinin sayısını ve uzunluklarını tutan bir veri yapısıdır. Algoritmanın başlangıcında sözlük boştur. Sözlük hem kodlama yapılırken hem de kod çözülürken inşa edilir.

3.3.2 Huffman kodlaması

Huffman kodlaması belli bir değer kümesi ve her bir değerin oluşma frekansı için, her bir değeri ikili olarak kodlama için bir yol belirler. Daha da önemlisi, Huffman kodlama optimum bir kodlama sağlar.
Huffman kodlaması, kodlanacak her bir değer için, ikili bir dizi üretir. Bu dizilerin uzunluğu keyfi bir değerdir. Uygulamanın doğru bir şekilde işleyebilmesi için, hiçbir ikili dizinin başka bir ikili dizinin önceki olmaması gerekir.

3.3.3 Dalgaşekli kodlaması

Dalgaşekli kodlaması, dalgaşeklinin kendisini etkin bir şekilde kodlamaya çalışır. Dalgaşekli kodlama teknikleri, konuşma işaretlerinin kodlanması için kullanıldığı kadar ses işaretlerinin kodlanması içinde kullanılır. Dalgaşekli kodlamanın en basit biçimi PCM kodlama işaretidir. Fakat işaretin daha az yer işgal etmesini sağlamak için işlenmesi gerekir. Bu teknikler kayıplı sıkıştırma sağlar ve kodu çözülmüş veri

orijinal veriden daha farklı olabilir. Dalgaşekli kodlama teknikleri genellikle oldukça iyi ses kalitesi sunar ve 16kb/s veya daha fazla bant genişliğine gereksinim duyar [27].

3.3.3.1 Diferansiyel kodlama

Diferansiyel kodlamanın temel prensibi, ses işaretlerinde bir örnek değerinin önceki örnek değerlerinden bir şekilde kestirilebilmesidir. Belirli bir sayıda örnek verilirse, algoritma bir sonraki örnek değerini kestirebilir. Kestirilmiş değer ile gerçek değer arasındaki fark kaydedilir. Fark, genelde gerçek değerinden daha az sayıda bit ile ifade edilir. Bu da sıkıştırma sağlamış olur.

3.3.3.2 Diferansiyel PCM (DPCM) kodlama

Diferansiyel PCM, bir PCM işaretinin kestirilmiş ve gerçek değerleri arasındaki farkı hesaplar ve bu değeri tutmak için sabit sayıda bit kullanır. Bu farkı tutmak için kullanılan bit sayısı, hatalar göz ardı edildiğinde, işaretin sahip olduğu maksimum eğimi gösterir. Eğer bu eğim aşılırsa, bir örneğin değerine ancak belli bir hatayla yaklaşılabilir. Genelde iyi bir konuşma kalitesi için farkı beş bit ile kodlamak ile yeterli olmaktadır. Fakat dört bit ile kodlandığında da, kalite sağlanmış oluyor [18].

3.3.3.3 Adaptif DPCM (ADPCM) kodlama

DPCM’ in bir çeşidi de adaptif DPCM’dir. Bu kodlama yönteminde, farkı ifade etmek için sabit sayıda bit kullanılır. DPCM yönteminde tüm bitler farkı kodlamak için kullanılır. Buna karşılık ADPCM yönteminde bazı bitler kuantalama değerini kodlamak için kullanılır. Bu yolla farkın kararlılığı ayarlanabilir [24].

3.3.3.4 Delta modülasyonu

Delta modülasyonu DPCM' in basit bir şekli olarak görünebilir. Bu yöntemde fark değerini kodlamak için sadece bir bit kullanılır. Bu bitin alacağı değere göre kestirilmiş değer belirli bir miktarda arttırılır veya azaltılır. Bu yöntemin değişik bir şekli de adaptif delta modülasyonu (ADM) olarak adlandırılır. Bu yöntemde, kestirilmiş değeri arttırmak veya azaltmak için kullanılan adım uzunluğu değiştirilebilir. Bu yolla, orijinal işarete daha yakın bir sonuç elde edilebilir [6].

3.3.3.5 Vektör kuantalama

Vektör kuantalamada, giriş, vektör diye adlandırılan eşit uzunlukta parçalara ayrılır. Bu kodlama yönteminde sözlük (codebook) diye adlandırılan vektör dizileri mevcuttur. Girişteki her bir vektör için, sözlükte bu değere en yakın vektör alınır. Sözlükte buna karşılık gelen değerin indeksi, giriş vektörünü kodlamak için kullanılır. Bu yöntem PCM verisi dışındaki diğer veri tiplerine de uygulanabilir. Örnek olarak, diğer sıkıştırma tekniklerinin hata terimini tutmak için, vektör kuantalama kullanılabilir [5].

3.3.3.6 Dönüşüm kodlaması

PCM verisi göz önüne alındığında, işaretin zaman domenindeki yapısına bakılır. Dönüşüm kodlamasında, işarete öncelikle orijinal şeklinden daha iyi bir sıkıştırma oranının sağlanabileceği başka bir domene dönüştürülür. İşaret sıkıştırıldıktan sonra, orijinal işarete dönüştürmek için ters dönüşüm işlemi uygulanır [6]. İşaretin dönüştürülebileceği domenlerden biri frekans domenidir. Bir sıkıştırma algoritması, insan sesi ve işitme sistemi hakkındaki bilgiyi kullanarak, hangi frekans bileşenlerinin daha önemli olduğuna karar verebilir. Bu bileşenler, diğerlerine göre daha hassas bir şekilde kodlanır. Bu amaçla kullanılan dönüştürme yöntemlerinden en önemlileri Ayrık Fourier Dönüşümü (DFT-Discrete Fourier Transform) ve Ayrık Cosinus Dönüşümü (DCT-Discrete Cosine Transform) dür.

3.3.4 Ses Kodlaması

Dalgaşekli kodlama yöntemleri, dalgaşeklini gerçeğine olabildiğince yakın bir şekilde modellemeye çalışır. Fakat sadece konuşma işaretlerinin kodlanması gerekiyorsa, konuşma işaretlerinin ayrıntılı bir analizi yapılarak daha az sayıda bit ile kodlama yapılabilir. Ses kodlama teknikleri, konuşma işaretlerinin kendisini kodlamaktan çok, konuşma işaretinin insan ses sisteminde nasıl oluşturulduğuyla ilgili bilgiyi kodlar. Bu teknikle çok düşük bit hızlarında bile (4.8 kb/s' nin altında) anlaşılabilir şekilde iletişim sağlanabilir. Ancak tekrar üretilen işaret genellikle oldukça suni bir işarettir ve konuşanların birbirini tanıması oldukça güçtür.

3.3.5 Hibrit kodlama

Dalgaşekli kodlayıcılar genelde 16kb/s'nin altındaki veri oranlarında iletim sağlayamazlar. Ses kodlayıcılar ise oldukça düşük hızlarda iletime izin verirler. Fakat sesin kalitesi düşüktür. Ayrıca algoritmanın genelde ortam gürültüsü ile ilgili zayıf olduğu taraflar vardır. Hibrit kodlayıcılar her iki tekniğin de avantajlı olduğu yönleri kullanır. Bunun sonucunda da konuşmanın anlaşılabilir ve konuşmacının tanınabilir olduğu, aynı zamanda da düşük hızlarda iletim sağlanır. Tipik bant genişliliği gereksinimi 4.8 ile 16kb/s arasındadır. RELP, CELP, MPE ve RPE gibi hibrit kodlayıcılar aşağıda açıkanacaktır [6].

3.3.5.1 RELP (Residual Excited Linear Prediction)

RELP (Residual Excited Linear Prediction) kodlayıcının çalışma prensibi hemen hemen LPC kodlayıcının çalışma prensibiyle aynıdır. İşaretin analizi için ses bölgesi filtresinin parametreleri belirlenir ve elde edilen filtrenin tersi işaret uygulanır. Bu rezidüel işareti sağlar. LPC kodlayıcısı daha sonra işaretin sesli veya sessiz olduğunun tayini için kontrol edilir. Bunun sonucunda da uyarma işareti modellenir. Fakat RELP kodlayıcıda ise, rezidüel kısım daha fazla analiz edilmez. Doğrudan konuşma sentezi için uyarma olarak kullanılır. Rezidü, dalgaşekli kodlayıcılar kullanılarak sıkıştırılır ve daha az bant genişliği kullanılması sağlanır. RELP

kodlayıcılar oldukça iyi bir ses kalitesine izin verir. RELP kodlayıcılarda 9.6kb/s civarında hızlarda iletişim sağlanır [6].

3.3.5.2 CELP (Codebook Excited Linear Prediction)

CELP (Codebook Excited Linear Prediction) kodlayıcılar, ses kodlayıcıların doğal olmayan ses kalitesini iyileştirmek için kullanılır. Bunun için geniş bir uyarma işareti kümesi kullanılır. Uyarma işaretleri CELP sözlüğünde (Codebook) tutulur. Hangi uyarma işaretinin kullanılacağının belirlenmesi için, kodlayıcı oldukça ayrıntılı bir arama yapar. Sözlükteki her bir değer için konuşma işareti sentezlenir ve en küçük hatayı oluşturan değer seçilir. Daha sonra uyarma işareti bu değere karşılık düşen indeks ile kodlanır. Görüldüğü gibi kodlayıcı temel olarak, uyarma işaretini kodlamak için vektör kuantalama yöntemini kullanır. Bu teknik sentez ile analiz(analysis-by-synthesis) tekniği olarak adlandırılır. Çünkü işareti analiz etmek için bazı olasılıkları sentezlemesi gerekir ve en az hata oluşturanı seçmesi gerekir. CELP yöntemi 4.8kb/s hızlarında iletime izin verir [30].

3.3.5.3 MPE ve RPE (Mutipulse and Regular Pulse Excited)

MPE ve RPE (Mutipulse and Regular Pulse Excited coding) teknikleri de uyarma işaretinin gösterilimini daha iyi bir şekilde ifade ederek konuşma kalitesini iyileştirmeye çalışır. MPE yönteminde uyarma işareti seri darbeler şeklinde modellenir. Her bir darbe, uyarma işaretinin genliğini gösterir. Darbeleri konumları ve genlikleri, sentezle analiz yöntemiyle belirlenir. MPE yöntemi 9.6kb/s civarında yüksek kalitede konuşma üretir. RPE tekniği de benzer şeklinde çalışır, yalnızca bu yöntemde farklı olarak darbeler düzenli bir şekilde yerleştirilir. MPE ve RPE yöntemi 13kb/s hızlarında iletime izin verir [31].

3.4 Ses Sıkıştırma Standartları

Farklı uygulamaların birbirleriyle çalışabilmesi için, ses sıkıştırma standartlarının oluşturulması gerekir. VoIP uygulamalarında en çok kullanılan standartlar, ITU-T (International Telecommunication Union –Telecommunication) standartlarıdır [21].

3.4.1 G.711 sıkıştırması

Örnekleme frekansı 8 kHz'dir. Örnek başına 8 bit kullanılır. Toplam bit hızı gereksinimi Ortalama görüş tayini (Mean Opinion Score- MOS) değeri 4.3 'dür.

3.4.2 G.723 sıkıştırması

Genel olarak düşük bit hızlarında iletim için kullanılır kalite olarak G.711' den kötüdür. Ama band genişliği gereksinimi de G711'in yaklaşık onda biri kadardır. Cebirsel CELP(ACELP) (6,3 kb/sn) yada MP-PLQ (5,3 kb/sn) algoritmaları kullanılır. MOS değeri 4.1 'dir [32].

3.4.3 G.726 sıkıştırması

16,24,32,40 kb/sn hızlarında adaptif diferansiyel PCM kodlaması kullanılır. MOS değeri 2 ile 4.3 arasında değişir.

3.4.4 G.728 sıkıştırması

Düşük gecikmeli CELP kullanılır. Bit hızı 16kb/sn dir. MOS değeri 4.1 dir.

3.4.5 G.729 sıkıştırması

CS-ACELP algoritması kullanılır. Bit hızı 8 kb/sn dir. MOS değeri 2 ile 4.3 arasında değişir [14].

Tablo 3.1 ' de kodlama standartları gösterilmektedir.

Tablo 3.1 : Kodlama standartları

Standart	Tanımlama	Bit Oranı (kb/s)	MOS
G.711	Örnek başına sekiz bit kullanılan darbe kod modülasyonu (PCM). Örnekleme 8000 Hz'de yapılır	64	4-4.3
G.723	Düşük bit oranları için tasarlanmış konuşma işaretleri kodlayıcısı. Aşağıdaki kodlama tekniklerinden birini kullanır. -MP-PLQ (Multiple Maximum liklihood Quantis) -ACELP (Algeebratic CELP)	6.3 5.3	4.1- 4.1
G.726	Kodlayıcı ADPCM sıkıştırma tekniğini kullanır. G:721 ve G.723 standartlarını içerir.	16, 24, 32 ve 40	2-4.3
G.728	LD-CELP (Düşük gecikmeli CELP)	16	4.1
G.729	CS-ACELP (Conjugate Structure ACELP)	8,6.4,11.8	4.1 3.7

3.5 Servis Kalitesi (Quality Of Service-QoS)

VoIP konusu göz önüne alındığında servis kalitesi (Quality of service), önceden tanımlanmış uçtan uca hizmet gereksinimlerinin IP şebekesi tarafından gerçekleştirilebilme yeteneği olarak tanımlanabilir [33].

3.5.1 Servis kalitesini etkileyen unsurlar

Servis kalitesi bir performans kriteridir. Servis kalitesini etkileyen her etken, sistem performansını doğrudan etkileyecek ve hatta belirli bir QoS değerinin

tutturulabilmesi için de sisteme ek maliyet getirecek yatırımlar yapmak gerekecektir. VoIP uygulamaları gerçek zamanlı veri akışına intiyaç duyar. Bu nedenle ses iletişimin kaliteli yapılması gerekmektedir. Bu nedenle VoIP uygulmasına destek veren ağlarda ekin çözümler üretilerek belli QoS değeri yakalanmalıdır. Çoklu lortam (multi media) servislerine olan talep, farklı trafik akışları için tek bir servis sınıfından farklı servis sınıflarına kaymayı gerekli kılmıştır. Paketlenmiş sesi bu farklı haberleşme bulutları üzerinden verimli olarak taşımak için ortak bir QoS modeli gerekmektedir. Servis kalitesini etkileyen unsurlar aşağıda verilmiştir [33,34].

3.5.1.1 Gecikme (Delay)

Ses paketinin kaynaktan yollanması ile alıcı tarafından alınması arasında geçen süre olarak tanımlanır. Gecikmeler beş adettir ve aşağıda kısaca açıklanmaktadırlar.

Örnekleme gecikmesi: Darbe kodlama modülasyonu yapılmadan önce ses işareti örneklenir. Örnekleme esnasında belirli bir süre geçer. Bu geçen süreye örnekleme gecikmesi denir. Bu değer işaretin örnekleme aralığı ile doğrudur orantılıdır. Çoğu zaman 125-150 ms kadardır.

Sıkıştırma gecikmesi: Sesin belirli bir kodlayıcı ile kodlanması sırasında geçen süredir. Bu süre kodlama algoritmasının karışıklığına ve kodlama yapan işlemcinin işlem yapma hızına bağlıdır.

Kod çözme gecikmesi: Belirli bir kodlayıcı ile sıkıştırılmış sesin açılması (kodçözme) sırasında geçen süredir. Bu süre ; kod çözücü, işlemcinin hızına ve sıkıştırma algoritmasının karmaşıklığına bağlı olarak değişir.

Transmisyon gecikmesi: Ses paketin iletimi esnasında iletim yollarında geçen süredir. Bu süreyi azaltmanın en iyi yolu, iletim hattının enformasyon hızını arttırmaktır. Örnek olarak , 16kb'lik bir veri paketi 4kb/s hızındaki bir hattan iletildiğinde iletim 4 saniye alırken, 48kb/s'lik bir hattan (ISDN-D kanalı) yollandığında iletim 0,33 saniye alır.

Bellekleme gecikmesi: Tüm paket gecikmelerinin aynı yapılması amacıyla verilerin belirli bir süre belleklerde (buffer) tutulmasından kaynaklanan gecikmedir..

3.5.1.2 Jitter

Bütün paketlerin iletilmesinde yaşanan toplam gecikme miktarına denir. Bir kaynaktan çıkan paketlerin her birinin farklı gecikme değerleriyle alıcıya ulaşması sonucu oluşur. Jitter'e gecikme varyasyonu da denir. İletim süresinin uzaması ile oluşan değişimdir [9].

3.5.1.3 Yankı (Echo):

İletim hatlarında ki ek noktalarında veya hattın uygun empedans ile sonlandırılmaması nedeniyle oluşur. Empedans uyuşmazlığı olan noktalarda işaretlerin bir bölümü hatta yoluna devam ederken elektriksel işaretin diğer bir bölümü ise yansır. Benzer etki PSTN' deki iletim hatlarında kullanılan 2 tel-4 tel dönüşümü yapan hibrit trafolarda da ortaya çıkar [35].

3.5.1.4 Net çıkış hızı

Alıcı tarafından alınan verilerin ortalama hızıdır. Bu ortalama hesaplanırken tıkanma durumu da göz önünde tutulmalıdır. Tıkanma durumu değerlendirilerek ortalama hız belirlenir.

3.5.1.5 Paket kayıp oranı

Hedefe ulaşamayan veri paketlerinin ve hedefe yollanan veri paketlerine oranıdır.

3.5.1.6 Gürültü

Hattın enformasyon hızını dolayısıyla servis kalitesini etkileyen bir başka unsur da gürültüdür. Hatta bulunan gürültü arttıkça kanal kapatisesi düşecek; dolayısıyla iletim süresi artacaktır servis kalitesi olumsuz etkilenecektir.

3.5.2 Servis kalitesi (Quality Of Service-QoS) için kullanılan yöntemler

Ses sinyallerindeki küçük değişiklikler VoIP sistemlerinin sağlıklı çalışmasını etkiler. Bu durumda Servis kalitesini sağlamak çok önemlidir. IP ağlarında ses paketlerinin yanında bir çok veri paketi de iletilmektedir. Ses paketleri ile beraber büyük bir veri paketi de iletilmek istendiğinde ses paketinin veri paketi iletilene kadar beklemesi gerekebilir. Bu gecikmenin artmasına sebep olmaktadır. IP üzerinden ses taşımada servis kalitesinin sağlanması için kullanılan yöntemler aşağıda belirtilmiştir [15].

3.5.2.1 Veri bağı katmanı servis kalitesi (Data link layer-QoS)

ATM teknolojisi veri bağı katmanında servis kalitesi ile ilgili parametreleri barındırmaktadır. Etnernet tekonolojisinde ise veri bağı katmanında servis kalitesi ile ilgili bir tanımlama yapılmamıştır.Ethernet servis kalitesinin sağlanması için IEEE 802 1.q VLAN başlığının kullanılması gerekmektedir. VLAN başlığında her çerçeve için sekiz seviyeli önceliklendirme seçeneği sunulmaktadır. Bu önceliklendirme değerleri iletişim yapan uçlar arasında yol boyunca kontrol edilerek ses paketlerindeki gecikme minimum düzeye indirilmektedir.

3.5.2.2 Servis tipi (Type of service-TOS)

IP başlığında 8 bitlik bir TOS alanı bulunur. Bu alanın 3 biti önceliklendirme için ayrılmıştır. Geriye kalan 4 bit ise servis tipini belirtmektedir. TOS kısmını kullanarak servis kalitesi sağlamak için iki özelliğe ihtiyaç duyulmaktadır. Öncelikli olarak

yönlendiricilerin (Router) TOS kısmını fark edebilir özellikte olması gerekmektedir. Bu alandaki bit değerlerine bakarak farklı servis sınıfları oluşturup buna önceliklendirme yapabilmelidir.İkinci olarak TOS alanının doldurulması gereklidir. TOS değerleri uçbirim tarafından doldurulabileceği gibi trafik tipine göre bir yönlendirici tarafından da atanabilir.

3.5.2.3 RSVP (Resource reservation protocol) servis kalitesi

RSVP ses oturumu başlatıldığı zaman oturumun kurulması için gerekli kaynakların rezerve edilmesini sağlar. Eğer uygun kaynaklar sağlanamıyorsa bağlantı isteği reddedilir. Bu özellik çok güçlü bir servis kalitesi sağlamaktadır. Fakat ölçeklenebilir bir servis değildir. Bunun sebebi bu kaynakların iletişim kurmak isteyen uçlar arasındaki bütün yönlendiricilerde (router) rezerve edilmesi gerekmektedir. LAN (Local Area Network) uygulamarında kullanışlı olmasına rağmen geniş IP ağları uygun bir çzöüm değildir.

3.5.2.4 Farklılaştırılmış servisler

Farklılaştırılmış servisler ağ trafiğini sınıflandırarak kontrol etmek ve ses trafiği gibi bazı trafik türlerinin öncelikli olarak iletilmesi için kullanılır. Farklılaştırılmış servisler TOS baytında bulunan 6 bitlik alanı önceliklendirme için kullanır. Farklılaştırılmış servisler ile iletilen bir paket birbirinden farklı 64 olası iletim sınıfından biri ile gönderilebilir. Farklılaştırılmış servisler RSVP ’ den daha basit bir yapıya sahiptir ve uygulamarın ihtiyaçlarına göre paketler sınıflara ayrılabilir. VoIP iletiminde için servis kalitesini sağlamak için IP ağlarında kullanışlı bir yöntemdir.

3.6 VoIP’de Güvenlik

VoIp güvenliği; ses trafiğin başladığı noktadan bittiği noktaya kadar güvenli bir şekilde iletilmesi demektir. VoIP güvenliği için bütün İnternet tehditlerine karşı önlem alınmalıdır. En çok karşılaşılan güvenlik tehdidi, VoIP ses akışına kelimeler

sokmaktır. Bu veri ağlarında ortadaki adam (man-in-the-middle) olarak tanınır. Bu saldırıda saldırgan konuşmakta olan 2 kişinin akmakta olan VoIP trafiğine (UDP den giden ses akışı) kelimeler ekleyebilir. Bunu taraflardan birisinin duymasına imkan yoktur. VoIP'de güvenlik ses paketlerinin güvenliği ve IP güvenliği adı altında iki kısımda incelenebilir. Ses paketlerinin güvenliği uygulamaya yönelik güvenlik problemleri olarak ele alınmalıdr. IP güvenliği ise ağ ya da iletim katmanı protokolünün güvenliğinin sağlanması olarak ele alınmaktadır [35].

İnternet ortamı birçok tehdidin gelebileceği açık bir ortamdır. Bu saldırıların birçoğu takip edilemez. İnternet ortamında tam anlamıyla bir güvenlik hiçbir zaman mümkün olamaz. VoIP uygulamaları ağ güvenliği tehditlerine ve saldırılarına karşı açık kapılar barındırmaktadır. IP üzerinden ses iletiminin olduğu ağlarda iki tür tehditten söz edilebilir. Dış kaynaklı ve iç kaynaklı tehditler. Dış kaynaklı tehditler VoIP oturumuna eşlik etmeyen yani dışarıdan bir kullanıcının oluşturduğu saldırıdır. Dış kaynaklı saldırılar ses ve sinyalleşme paketleri güvenli olmayan üçüncü bir ağ üzerinden kullanıcılar üzerinden başlatılır. Bu tip saldırılar daha tehlikelidir. VoIP oturumunda herhangi bir kullanıcı iç kaynaklı bir saldırı başlattığında güven ilişkisi ortadan kalkmış olur. olur. Genelde bir kurum içerisindeki uçbirimler güvenlik duvarı yazılımları veya donanımları tarafından dış tehditlere karşı korunsa da iç kaynaklı tehditlere karşı çok büyük bir koruma sağlamamaktadır [15,36]. VoIP güvenlik tehditleri ve tavsiye edilen çözüm metotları Tablo 3.2 de gösterilmektedir.

Tablo 3.2: VoIP güvenlik tehditleri ve tavsiye edilen çözüm metotları

Güvenlik Tehditleri	Çözümler
Dos (Denial of Service) saldırıları : SIP Proxy sunucusuna yada ağgeçidi cihazlarının ağ servislerini vermesini engellemek için paket bombardımanına tutulmasıdır.	Cihazlar bu tip saldırıları önleyecek şekilde konfigüre edilmelidir.
Gizlice Dinleme: Yetkisi olmayan kullanıcıların geçen RTP ortam akışının önünü kesmesi ve mesajları çözerek dinlemesi	Gönderilen verinin Güvenli RTP(S RTP) gibi yöntemlerle şifrelenmesi.
Paket taklit edilmesi(packet spoofing) : Veri gönderen yetkili bir kullanıcının yerine geçmek.	Çağrıya katılan kullanıcıların adres (IP adresi gibi) bilgilerini kaynak kodlaması için birbirine göndermesi.
Ses paketlerinin tekrar gönderilmesi: Gerçek mesajın yeniden gönderilmesi bu sayede hedef kullanıcının mesajı yeniden işlemesi.	Gönderilen mesajın şifrelenmesi ve her mesaja uygulama katmanında bir sıra numarası atanması gerekir.
Mesaj İçeriği: Alınan mesaj ile gönderilen mesajın içeriğinin aynı olduğunun kontrol edilmesi.	Çeşitli yöntemlerle (http) mesajın gerçekliğinin doğrulanması.

4. VoIP PROTOKOLLERİ

İnternet üzerinden ses taşınırken iki önemli faz vardır. Bunlar işaretleşme ve veri aktarım fazıdır. İşaretleşme esnasında güncel olarak kullanılan iki önemli protokol vardır. Bunlar ITU'nun bir standardı olan H.323 ve IETF'nin bir standardı olan Oturum başlatma protokolü (Session Initiation Protocol-SIP)'dir [37].

Veri aktarım fazı ise; işaretleşmeyle anlaşmaya varan ve senkronize olan iki uç birim cihazının birbirleriyle gerçek zamanlı haberleşmeye başlamasıyla gerçekleşir. Veri aktarım protokolleri Gerçek zaman protokolü (Real Time Protocol-RTP), Gerçek zaman kontrol protokolü (Real Time Control Protocol-RTCP) ve Kaynak ayırma protokolü (Resource Reservation Protocol-RSVP) dür.

4.1 VoIP İşaretleşme Protokolleri

4.1.1 Oturum başlatma protokolü (SIP)

SIP, IETF' nin Multiparty Multimedia Session Control (MMUSIC) grubu tarafından geliştirilen multimedia uygulamaları için bir protokol grubudur. Protokol ilk defa 1999 yılında IETF RFC 2543 isminde yayınlanmıştır. Daha sonra geliştirilerek bir takım değişiklikler ilave edilerek RFC 3261 olarak yeniden yayınlanmıştır [38].

MMUSIC gurbu, H.323' ün aksine küçük bir çekirdek protokol ile başlayıp bu protokolü ihtiyaçlara göre geliştirmeyi amaçlamıştır. Çok basit bir yapıya sahiptir ve HTML bazlıdır. HTML'de kullanılan kodlar ufak değişikliklerle SIP'de kullanılabilir. Genişleme yeteneğine sahip bir protokoldür. Zamanla yeni özellikler bu protokole kazandırılabilir. Modüllerden oluşan bir yapıya sahiptir .En büyük özelliği, kullanıcıyı oturuma davet eden protokolden ayırabilmesidir. Büyük tarafik hacimlerini karşılayabilir. Web ile entegre olma yeteneğine sahiptir. Böylece e-posta,

akan medya uygulamaları ve diğer protokollerle kolayca çalışabilir. TCP ve UDP ' nin ikisinide destekler. Uygulama katmanı protokolüdür. VoIP sistemlerinde sinyalleşme için kullanıldığı gibi , görüntü konferansı veya etkileşimli oyunlar için oturum açar. IP ağlarında arama aktarma özelliği getirerek, VoIP hizmeti veren servis sağlayıcıların web, elektronik posta veya chat servisleri ile bu özelliği birleştirmesini sağlar. Kullanıcı tanımlama, yönlendirme ve kayıt olma servisleri sağlar. SIP geleneksel telefon özellikleri (kişisel dolaşım, günün saatine göre arama yönlendirme, coğrafi duruma göre arama yönlendirme)' nide desteklemektedir [39].

SIP ile internet telefonu, multimedya (ses, görüntü ve data) iletimi, multimedya konferansları gibi oturumlar yönetilebilir. SIP esnek bir protokoldür. İstendiğinde devam eden çoklu gönderim bir konferansa yeni kullanıcılar dahil edilebilir ya da çıkartılabilir. İsim eşlemesi (name mapping) ve yönlendirme servisini destekler. SIP'de multimedya bağlantılarını kurmak ve sonlandırmak için 5 adet yüzey tanımlanmıştır.

Kullanıcı konumu: İletişim kurulması istenen uç noktanın yerinin belirlenmesi için kullanılır.

Kullanıcı uygunluğu: Çağrılan tarafın haberleşmeye katılma isteğinin belirlenmesi için kullanılır.

Kullanıcı yetenekleri: Ortam parametrelerinin belirlenmesi için kullanılır.

Oturum kurulumu: "Çalma işlemi", çağrı gönderen taraf ve aranan tarafın oturum parametrelernin nasıl kurulacağını tanımlar.

Oturum yönetimi**:** Oturumda veri transferleri, oturumun sonlandırılması, oturum parametrelerinin değiştirilmesi ve çağırma servislerini tanımlar [3,40] .

SIP tam anlamıyla tümleşik bir haberleşme sistemi değildir. SIP'in ses, görüntü ve multimedya servislerini verebilmesi için birtakım başka protokollerle işbirliği içerisinde çalışması gerekir. Bu protokoller arasında gerçek zamanlı veri iletimi için

RTCP, PSTN şebekesine bağlantıda kullanılan ağ geçitlerinin kontrolü için MEGACO (Ortam Geçidi Denetim Protokolü), çoklu ortam oturumlarını tanımlamak için SDP (Oturum Tanımlama Protokolü) protokolleri sayılabilir. Buna rağmen temel fonksiyonları ve kullanımı açısından SIP bu protokollere bağımlı değildir.SIP kullanıcılara birtakım güvenlik servisleri sağlar. Bunlar arasında DoS (Denial of Service) saldırıları önlemesi, kullanıcı doğrulanması (hem kullanıcılar arası, hem de kullanıcı ile Proxy arası) içerik koruması, şifreleme sayılabilir. SIP IPv4'ün yanı sıra IPv6 ile beraber çalışabilen bir protokoldür [60]. SIP, SMTP, ve HTTP gibi metin bazlı bir protokol olduğundan UTF-8 karakter setini kullanır. Bazı farklılıklar dışında SIP'in mesaj ve başlık alanının biçimi http ile aynı sayılır.SIP ağı dört mantıksal bileşenden oluşur. Her bir SIP bileşeninin özel bir fonksiyonu vardır. Bir fiziksel cihaz bu mantıksal bileşenlerden biri ya da birkaçının görevini üstlenebilir. Örneğin Proxy sunucusu olarak çalışan bir sunucu aynı zamanda yönlendirme sunucusu ya da kayıt sunucusu görevini de üstlenebilir [41].

4.1.1.1 SIP bileşenleri

SIP protokolünün tanımladığı 4 tip SIP mantıksal bileşeni vardır.

1) Kullanıcı arabirimi (User Agent-UA) :

Uç noktadaki SIP bileşenleridir. Kullanıcı arabiriminin görevi çağrıyı başlatmak ve oturum tamamlandığında çağrıyı sonlandırmaktır. RFC 2543'de UA uç noktalarda çalışan bir uygulama olarak tanımlanmaktadır. Kullanıcı arabirimi İstemci ve Sunucu olarak iki alt bileşenden oluşmaktadır. Bir UA her iki durumda da bulunabilir [3,42].

İstemci kullanıcı arabirimi (User agent client-UAC): İstemci Kullanıcı Birimi SIP oturumunu çağrı gönderme isteği ile başlatır.

Sunucu kullanıcı birimi (User agent server-UAS): Sunucu Kullanıcı Birimi çağrının gönderildiği hedef kullanıcıyı tanımlar. UAS'ın görevi gelen çağrıyı cevaplamaktır.

Kullanıcı Arabirimi olarak çalışabilen cihazlara yazılımsal ve donanımsal tabanlı IP telefonlar, ağ geçitleri, otomatik cevap verme servisleri örnek verilebilir.

2) Proxy sunucusu (Proxy Server):

SIP sunucusu kullanıcı arabiriminden gelen SIP oturum taleplerini alır ve oturum talebini hedef kullanıcı arabirimi ya da hedef kullanıcının bulunduğu taraftaki başka bir sunucuya gönderir. Proxy Sunucusu hem kullanıcı arabirimi adına oturum talebinde bulunabilen, hem de bu taleplere gelen cevapları kullanıcı arabirimine gönderme fonksiyonunu üstlenen bir ara bileşendir. Proxy sunucusu hem istemci hem de sunucu olarak davranmaktadır. Bu sunucunun başka bir özelliği ise SIP kullanıcısı Proxy sunucusu vasıtasıyla H.323 protokolünü destekleyen bir kullanıcı ile konuşabilir.

3) Yönlendirme sunucusu (Redirect Server):

Kullanıcı Arabirimin'den gelen SIP taleplerini alır, hedef kullanıcı ile bağlantı kurulabilmesi için çağrı gönderen kullanıcıya hedef kullanıcının adresini gönderir kullanıcı adına hedef kullanıcıya doğru herhangi bir oturum talebinde bulunmaz.

4) Kayıt sunucusu(Registrar Server):

Kayıt sunucusu kullanıcılarının konum bilgilerini girmelerini sağlayan sunucudur. İstek gönderen kullanıcıların konum bilgilerini veritabanında günceller [3,43] .
SIP sunucuları SIP uçbirimlerinin mesaj gönderebilmesi, konum bilgisi kaydı yaptırabilmesi, ağlar arası iletişim kurabilmesi ve ağlar arası bağlantı kurabilmesi için gerekli ağ elemanlarıdır. SIP sunucularının desteklediği özellikler sayesinde istenilen yönlendirme ve güvenlik politikaları yüklenebilir, kullanıcıların kimlik doğrulamaları sağlanabilir. SIP sunucuları genelde çok sayıda bağlantıyı destekleyen, düşük gecikmeli, yüksek gerçek zaman performansı gösteren ölçeklenebilir sunucular olmalıdır [41,44].

4.1.1.2 SIP mesaj Tipleri

SIP haberleşmesinde iki tip SIP mesajı tanımlanmıştır. Bu mesajlar sunucuya gönderdiği talep mesajları (Request messages) ve sunucunun kullanıcıya gönderdiği cevap mesajlarıdır (Resposes messages).

1) Talep Mesajları:

SIP haberleşmesi altı tip talep mesajdan oluşmaktadır. Talep mesaj tipleri ve tanımlamaları Tablo 4.1 ’de gösterilmektedir.

Tablo 4.1 :SIP talep mesaj tipleri ve açıklamaları

Yöntem	Tanım
INVITE	Çağrıyı başlatmak için kullanılan mesaj tipidir.
ACK	Çağrıyı başlatma talebinin sonucunu bildirir.
OPTIONS	Kullanıcı arabirim yeteneklerinin ve sunucu yeteneklerinin sorgulanması için kullanılır.
BYE	Oturuma devam eden kullanıcıların çağrıyı sonlandırma için kullandıkları mesaj tipidir.
CANCEL	Devam eden çağrı talebinin iptal edilmesi için kullanılır.
REGISTER	Kullanıcıların konum bilgilerinin kayıt sunucusuna bildirilmesinde kullanılır.

2) Cevap mesajları:

SIP cevap mesajları talep mesajlarına cevap olarak gönderilen mesaj tipleridir. Mesajlar HTTP kodlarına benzer sayısal kodlar içermektedir. Bu mesaj tipleri geçici mesajlar ve sonuç mesajları olarak iki kısımda incelenebilir. Geçici mesajlar talep mesajlarının sonuçlanması sürecine kadar gönderilen bilgilendirme mesajlarıdır. Sonuç mesajları ise çağrı isteğinin sonucunu bildirir. Cevap mesaj kodları ve açıklamaları Tablo 4.2 ’de gösterilmektedir.

Tablo 4.2: SIP cevap mesaj kodları ve açıklamaları

Cevap Sınıfı	Durum Kodu	Açıklaması
Bilgilendirme	100	Deneniyor
	180	Çalıyor
	181	Çağrı yönlendirildi
	182	Kuyruğa alındı
Başarı	200	Tamam
	300	Çoklu seçim
	301	Sürekli olarak taşındı
	302	Geçici olarak taşındı
	3xx	
İstemci Hatası	400	Hatalı talep
	401	Yetkisiz
	402	Ücretlendirme gerekli
	403	Yasak
	404	Bulunamadı
	405	Yöntem kabul edilmedi
	4xx	
Sunucu Hatası	500	Dahili sunuuc hatası
	502	Hatalı ağgeçidi
	5xx	
Global Başarısızlık	600	Meşgul
	603	Ret
	604	Mevcut değil
	606	Kabul edilemez

4.1.1.3 SIP adreslemesi

SIP adresleri e-posta adreslerine benzer bir yapıda kullanıcı@etki_alani yapısındadır. Kullanıcı kısmında kullanıcı adı ya da telefon numarası bulunabilir. Etki_alani kısmında etki alanı ya da ağ adresi bulunabilir. E-posta adresleri ile aynı formatta olması bir kişinin SIP adresinin bulunmasını kolaylaştırmaktadır. Örnek SIP Adresleri

deneme@kou.edu.tr
902629876543@160.75.1.1

4.1.1.4 SIP mesaj yapısı

SIP karakter tabanlı bir protokoldür ve UTF-8 karakter setini kullanır. SIP mesajları istemciden sunucuya giden bir talep mesajı formatında ya da sunucudan istemciye giden cevap mesajı yapısındadır. Bütün SIP mesajları bir başlangıç satırı, bir veya daha fazla başlık alanı ve isteğe bağlı mesaj gövdesinden oluşur [3,45].

1. Başlangıç satırı:

Başlangıç satırı SIP mesajının başlık kısmını tanımlar. Bu alan mesaj türüne göre değişmektedir.

Talep mesajları için bu alanda talep mesajı türü (invite, ack, bye, vs.), çağrılan kişinin SIP adresi ve SIP protokol sürümü belirlenir. SIP adresi Proxy sunucusu tarafından değiştirilebilmektedir.

Cevap mesajları için bu alanda cevap mesajının durum kodu (200,300 vs.) ve durum kodunun metinsel ifadesi bulunur.

2. Başlık kısmı:

SIP başlık alanları mesaj niteliklerini barındırarak mesajın anlamını tanımlar. Birkaç ufak tefek fark dışında SIP mesajları ve başlık kısmının biçimi HTTP ile aynıdır.

2. Mesaj gövdesi:

Mesaj gövdesi kurulacak oturumun detaylandırılmasını sağlamak için kullanılır. Örnek olarak kurulacak çoklu ortam oturumunda kullanılan ses kodekleri, örnekleme oranları gibi parametreler bu kısımda tanımlanmıştır. Bu alanda ayrıca oturum ile ilgili ikili ya da metin formatında veri bilgileri taşınır. Bu kısım hem talep mesajlarında hem de cevap mesajlarında bulunabilir. SIP'de gövde kısmında bulunan

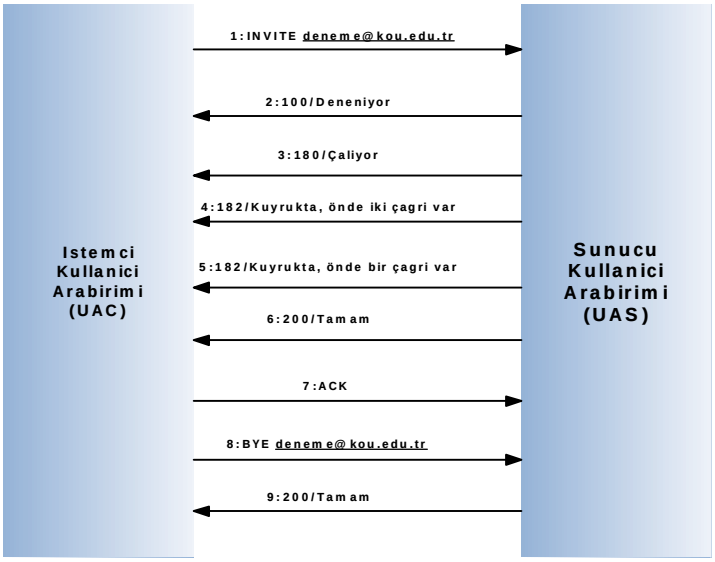
bilgiler ile başlangıç satırı ve başlık kısmında tanımlanan sinyalleşme bilgisi arasında herhangi bir ilişki bulunmamaktadır. Gövde türleri aşağıdaki yapılarda olabilir:
Oturum Tanımlama Protokolü (SDP)
Çok Amaçlı Posta Uzantıları (MIME)

4.1.1.5 SIP oturum kurulum örnekleri

SIP bileşenlerinin oturum kurulum örnekleri aşağıda incelenmiştir [3].

1) İki SIP uç birimi arasında SIP oturum kurulumu:

Şekil 4.1' de iki SIP uç birimi arasında oturum kurulumu aşamaları incelenmiştir.



Şekil 4.1: İki SIP uç birimi arasında oturum kurulumu

Çağrı akışı:

1) İstemci Kullanıcı Arabirimi deneme@kou.edu.tr SIP adresini yazarak INVITE mesajı ile çağrı talebinde bulunur. Bu mesajın içerisinde UAC'in yeteneklerini tanımlayan SDP paketi bulunmaktadır.

2**)** UAS çağrı talebini alır ve hemen “100” cevap kodlu “deneniyor” mesajını gönderir.

3) UAS uçbirimi çaldırırken aynı anda UAC’ı bilgilendirme amaçlı “180” cevap kodlu “çalıyor” mesajını gönderir.

4) UAS “182” cevap kodlu mesaj ile çağrın mevcut iki çağrının arkasında kuyrukta olduğunu bildirir.

5) UAS “182” cevap ko dlu mesaj ile çağrın mevcut bir çağrının arkasında kuyrukta olduğunu bildirir.

6) UAS çağrıyı cevaplar ve UAC’a “200” cevap kodlu çağrının cevaplandığını bildiren “tamam” mesajı gönderilir. Bu mesajla UAS’in yeteneklerini tanımlayan SDP paketide bulunmaktadır.

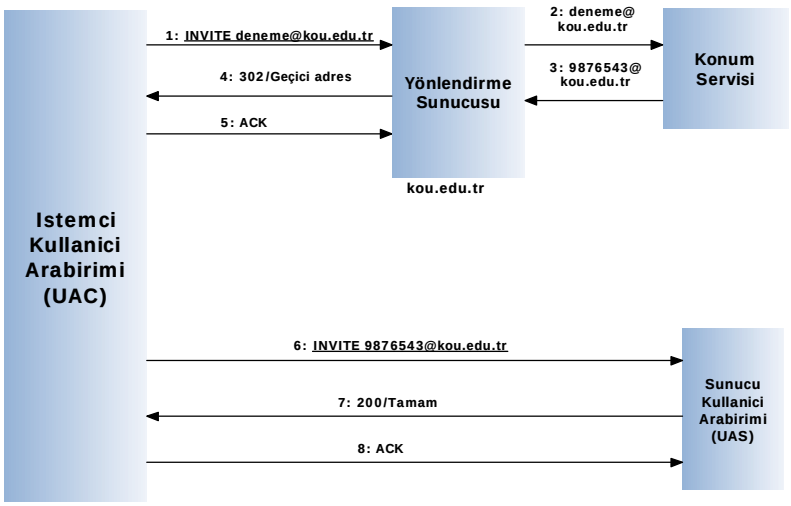
7) UAC ACK cevap kodlu mesaj ile cevabın alındığını bilgisini iletir. Bu mesaj ile birlikte ortam akışı başlamış olur.

8) UAC çağrıyı sonlandırmak istediğinde UAS’a “BYE” talebini gönderir.

9) UAS bu talebi “200” cevap kodlu mesaj ile onaylar ve çağrı sonlandırılır.

2) Yönlendirme sunucusu kullanılarak oturum kurulumu :

Şekil 4.2’ de yönlendirme sunucusu kullanılarak oturum kurulumu aşamaları incelenmiştir.



Şekil 4.2: Yönlendirme sunucusu kullanılarak oturum kurulumu

Çağrı akışı:

1) İstemci Kullanıcı Arabirimi deneme@kou.edu.tr SIP adresini yazarak INVITE mesajı ile çağrı talebinde bulunur.

2) Yönlendirme sunucusu hedef SIP adresini çözümleyemiyorsa adres çözümleme işlemi için Konum Servisi sunucusuna istekte bulunur.

3) Konum Servisi UAS'ın o anki konumunu 9876543@kou.edu.tr olarak döndürür.

4) Yönlendirme sunucusu yeni konum bilgisini UAC'ye "302" cevap kodlu "geçici taşıma" cevabı ile iletir.

5) UAC yönlendirme sunucusuna ACK mesajı yollayarak cevabı onaylar.

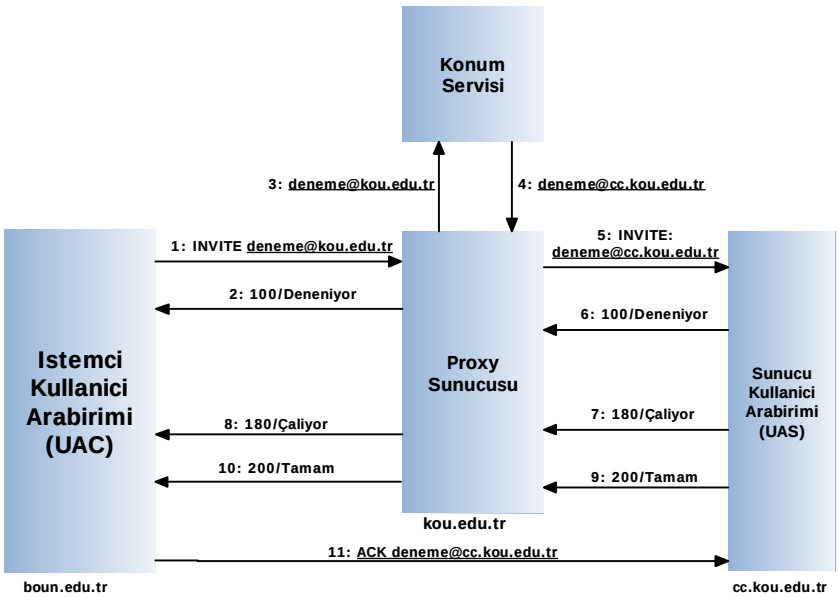
6) UAC 9876543@kou.edu.tr SIP adresini yazarak yeni bir INVITE mesajı ile çağrı talebinde bulunur.

7) UAS çağrıya cevap verir ve bu mesaj "200" cevap kodu ile UAC'ye bildirilir.

8) UAC ACK mesajı ile çağrı kurulumunu onaylar.

3) Proxy sunucusu kullanılarak oturum kurulumu:

Şekil 4.3' Proxy sunucusu kullanılarakı oturum kurulumu aşamaları incelenmiştir



Şekil 4.3: Proxy Sunucusu kullanılarak otuurm kurulumu

Çağrı akışı

1) İstemci Kullanıcı Arabirimi deneme@kou.edu.tr SIP adresini yazarak INVITE mesajı ile çağrı talebinde bulunur.
2) Proxy sunucusu "100" cevap kodlu "deneniyor" mesajını gönderir.
3) Yönlendirme sunucusu hedef SIP adresini çözümleyemiyorsa adres çözümleme işlemi için Konum Servisi sunucusuna istekte bulunur.
4) Konum Servisi UAS'ın o anki konumunu deneme@cc.kou.edu.tr olarak döndürür.
5) Proxy sunucusu UAC adına UAS'ın yeni SIP adresini deneme@cc.kou.edu.tr yazarak INVITE mesajı ile çağrı talebinde bulunur.
6) UAS "100" cevap kodlu çağrı talebinde bulunur.
7) UAS arabirimi çaldırırken aynı anda UAC'ı bilgilendirme amaçlı "180" cevap kodlu "çalıyor" mesajını gönderir.
8) Proxy Sunucusu "180" cevap kodlu mesajı UAC'a iletir.
9) UAS çağrıya '200' kodlu mesaj ile cevap verir.
10) Bu mesaj "200" cevap kodunu UAC'a bildirir. Artık Proxy sunucusu devreden çıkarılmış olur.
11) UAC ACK mesajı ile çağrı kurulumunu onaylar.

4.1.2 H.323 protokolü

H.323 ITU-T tarafından iki yada daha fazla taraf arasında IP benzeri QoS desteği olmayan bir ağ üzerinde ses yada görüntü trafiğini taşımak için geliştirilen bir protokol grubudur. Önceleri yerel ağlar üzerinde multimedia konferansı (iki yada daha fazla kullanıcının sesli ve görüntülü haberleşmesi) için geliştirilmiş, fakat sonradan IP üzerinden ses uygulamasını kapsayacak şekilde genişletilmiştir. Bu standardın tanımlanmasında Microsoft, IBM, Intel, telefon operatörleri ve ISP'lerden oluşan bir çok kurum ve firmanın geniş katılımı ve desteği sağlanmıştır. Internet telefonu amacıyla kullanılan en geniş ve en etkin standartlardan birisidir. Ses ile beraber tüm multimedia (data, ses, video, resim gibi) uygulamalarını desteklemektedir. H.323 standardı bir şemsiye standart olup birçok standardı kapsamaktadır. Bu standartlar ses kodlama, video kodlama, sistem kontrol, çoklama, multimedia, yayın senkronizasyonu ve yapısını içermektedir. Bu standartlar aynı

zamanda PSTN, Mobil, ATM, F/R, LAN, WAN, IP tabanlı Internet gibi şebekeleri içermektedir [45,46].

H.323 protokolünde yönetim (managment) ve muhasebe(accounting) desteği de sağlar. Bu yolla örnek olarak bir H.323 çağırmasının en fazla ne kadar bantgenişliği kullanacağını belirtmek mümkün olur. Arayan kişilerin faturalama ile ilgili bilgilerinin desteklenmesi için muhasebe işlevleri sağlanmıştır.H.323, paket bağlaşmalı ağların fazla güvenli olmamasından dolayı, güvenlik ile ilgili mekanizmalar da sunmaktadır.

4.1.2.1 H.323 protokol yığını

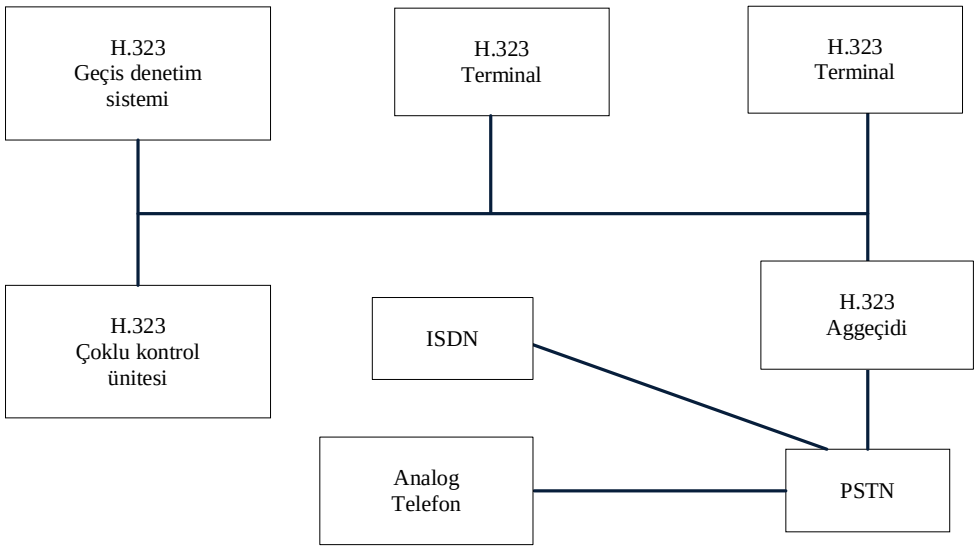
H323 protokol yığını; işaretleşme ve kontrol (H.245,H.225,RTCP), Ses kodekleri (G.7xx), Görüntü kodekleri (H.26x), Çoklu ortam haberleşmesi (T.12x), Taşıma (RTP) gibi temel bileşenlerden oluşur. Tablo 4.3 ’ de H323’ de hangi protokollerin hangi katmanlarda kullanıldığı görülmektedir [47].

Tablo 4.3 : H.323 protokol yığını

<table>
<tr><td rowspan="2">Veri</td><td colspan="2" rowspan="2">Kontrol ve işaretleşme</td><td rowspan="2">Sistem Kontrolü</td><td>Ses Çözücüler</td><td>Görüntü Çözücüler</td><td>Ses ve Görüntüsel Araçlar</td></tr>
<tr><td>G.7xx</td><td>H.26x</td><td rowspan="2">RTCP</td></tr>
<tr><td>T.12</td><td>H.245</td><td>H.225.0</td><td>H.225.0 RAS</td><td colspan="2">RTP</td></tr>
<tr><td colspan="3">UDP/TCP</td><td colspan="4">UDP</td></tr>
<tr><td colspan="7">IP</td></tr>
<tr><td colspan="7">Değişken Katman 2 Protokolleri</td></tr>
<tr><td colspan="7">Değişken Katman 1 Protokolleri</td></tr>
</table>

4.1.2.2 H.323 bileşenleri

H.323 protokolü; Terminal, Geçis Denetim Sistemi (Gatekeeper) Ağgeçidi (Gateway) ve Çok Noktalı Kontrol Ünitesi (Multipoint Control Unit-MCU) olarak dört temel bileşenden oluşmaktadır. Şekil 4.4 'de H.323 bileşenleri gösterilmektedir [7,48].



Şekil 4.4 : H.323 bileşenleri

Ağgeçidi (Gateway-GW)

Ses ağı (PSTN) ve IP (Internet Protocol) ağının birbirleri ile uyum içinde çalışması için "gateway" adı verilen ekipman kullanılması gerekir. Gateway, ses içeren IP paketlerini ses ağının anlayabileceği biçime sokarak, iki kullanıcı arasındaki bağlantıyı kurar. Voip Gateway PSTN ile veri ağı arasındaki geçiş noktasını göstermektedir. Gateway, yerel telefon santralıyla IP network arasında bir köprü görevi yapmaktadır. Gateway, yerel telefon santralından çevrilen bir telefon numarasını veri ağındaki bir adrese (IP adresi) dönüştürmekte ve aranan telefon santralına bağlı en yakın H.323 gateway' ine iletmektedir. IP network üzerinde kullanılan uygun protokollerin de yardımıyla bu telefon çağrısını algılayan hedef H.323 gateway' i aranan telefon numarasına yine yerel telefon santralı üzerinden

bağlantıyı kurmaktadır.Gateway, PSTN ağları ile IP ağları arasındaki interface yada geçiş elemanları olarak çalışan, dönüşüm (interworking) fonksiyonlarını yerine getiren modüldür. Bir gateway, paket anahtarlamalı bir ağ üzerindeki H.323 uyumlu terminaller ile devre anahtarlamalı bir ağdaki diğer H.323 terminalleri veya diğer bir gateway arasında gerçek zamanlı çift yönlü trafik sağlayan bir ağda "end point" olarak çalışır. Diğer ITU terminalleri H.310 (B-ISDN), H.320 (ISDN) , H.321 (ATM), H.322 (GQoS-LAN), H.324 (PSTN), H.324 (Mobile) yada POTS terminaller olabilir. Gateway iletim formatları (örneğin H.323 uyumlu bir uçtaki H.225.0 bir terminalle H.320 bir uçtaki H.221 bir terminal arasındaki dönüşüm) ve işaretleşme benzeri iletişim prosedürleri (H.323 bir uçtaki H.245 ile H.320 arasındaki bir H.242 arasındaki dönüşüm gibi) arasında gerekli dönüşümleri yapar. Bu dönüşümlerin nasıl olacağı H.246'da tanımlanmıştır. IP ağ ile PSTN ağ arasındaki çağrı kurulum ve kaldırma (call setup and clearing) işlemlerini de gatewayler üstlenir. Video, ses ve data formatları dönüşümünüde sağlarlar. Gateway üç fonksiyonel birimden oluşur.

1) Sinyalleşme Ağgecidi (Signalling Gateway -SG): SG, IP temelli ağ ile SCN (Switched Circuit Network) arasında işaretleşme bilgilerinin aktarımından sorumludur.

2) Media Ağgecidi (Media Gateway-MG): IP ağındaki taşınan veri ile (örneğin RTP/UDP/IP üzerinden taşınan veri) SCN ağda kullanılan veri (örneğin PCM kodlanmış ses yada GSM vb.) arasındaki eşleme ve birbirine dönüşüm işlemlerinden sorumludur.

3) Media Ağgecidi Kontrolbirimi (Media Gataway Controller-MGC): MGC, MG, SG ve Gatekeeper arasındaki iletişimleri düzenler. Gateway için gerekli çağrı işleme (call processing) işlemlerini sağlar.

Geçiş denetim sistemi (Gatekeeper)

Terminallerin ve gatewaylerin kayıt, kabul ve statü (Registration, Admission and Status-RAS) takibinden sorumlu olan ağ modülüdür. Gatekeeper'lar çağrı işleme/işaretleşme işlevlerini yerine getirirler. Aranan telefon numaralarının

iletileceği gatewaylerin IP adresleri gatekeeper cihazlarında yada yazılımlar kullanılarak tutulur. Gatekeeper'lar aşağıdaki fonksiyonları yerine getirirler .

Adres dönüşümleri: Ağdaki uçbirimleri takma (alias) isimlerinin gerçek transport isimlerine dönüştürmeyi sağlar.. Gatekeeper'lar bu işlevi yerine getirirken kendisine bağlanan uçbirimlerden aldığı kayıt (registration) mesajları ile sürekli olarak güncellediği tablolardan yararlanır. Bu tablolar kayıt mesajları dışındaki (dizin hizmetleri gibi) yöntemlerle de güncellenebilir [8].

Yetki denetimleri: Admission Request, Confirm ve Reject mesajları (ARQ/ ARC/ ARJ) ile uçbirimlerin LAN erişim taleplerini onaylar yada reddeder. LAN erişim istekleri değerlendirilirken çağrı izinleri (call authorization) bant genişliği sınırlamaları ya da benzeri diğer kriterler kullanılabilir. Bu fonksiyon NULL olarak gerçekleştirilerek gelen bütün taleplerin LAN' a erişimleri de sağlanabilir.

Bant genişliği yönetimi: Bandwidth Request, Confirm ve Reject mesajları ile uçbirimlerin LAN bant genişliği taleplerini onaylar yada reddeder.

Bölge yönetimi (Zone management): Tek bir gatekeeper tarafından yönetilen terminallerin, gatewaylerin ve MCU'ların toplamı zone olarak adlandırılır. Gatekeeper yukarıda anlatılan bütün fonksiyonları kendi yönetimindeki zone için sağlar [48].

Gatekeeper'ların kullanılma amacı, çağrıları yaparken makine adresleri yerine makinelere verilecek takma isimleri kullanabilme, ağdaki bant genişliği kullanımının yönetilmesi, Gateway ve MCU gibi ağ kaynaklarının yönetilebilmesidir. Gatekeeper gerçek H.323 tanımında video konferansları sırasında ağa erişimi kontrol eden bir birim olarak tasarlanmıştı. Zamanla adres dönüşümü benzeri fonksiyonlarını da kazandı. Bant genişliği denetimi ise ücretlendirme ihtiyaçları sonucunda ortaya çıktı. Gatekeeper'ların sağlayabileceği bir diğer serviste çeşitli doğrulama (authantication) yöntemlerini kullanarak bir çağrıya güvenlikle ilgili opsiyonların eklenmesidir. İşaretleşmede kullanılan Q.931 yada ve H.245 mesajları gatekeeper tarafından yönlendirilebilir ve çağrılar hakkında istatistiksel bilgilerin toplanması sağlanabilir. Çağrı gönerdeme (Call forwarding) yada Çağrı aktarımı (call transfering) gibi telefon hizmetleri de Gatekeeper'lar aracılığı ile verilebilmektedir [49].

Çok noktalı kontrol ünitesi (Multi-point control unit -MCU)

MCU ağda ikiden fazla terminalin yada Gateway'in çoklu bir konferansa katılımlarını sağlamaya yarayan cihazlardır. Sonradan çoklu bir konferansa dönüşebilecek ikili görüşmeler de MCU'lar aracılığı ile sağlanabilir. MCU iki kısımdan oluşur:

1) Çoklu kontroller (Multipoint controller- MC): Bu kısım çok noktalı kontrol ünitesinde mutlaka bulunmak zorundadır. MC, çağrı süreçlerine, konferansa katılacak bütün terminallerin ortak iletişim seviyelerinde bulunmalarını sağlamak için iletişim parametreleri üzerindeki uzlaşmaları (negotiation) sağlar.

2) Çoklu işlemci (Multipoint processor -MP) : Bu kısım çok noktalı kontrol ünitesinde mutlaka bulunmak zorunda değildir. MP, MC'nin denetiminde medya stream'lerinin işlenmesi (mixing, switching vb.) görevlerini yürütür. MP, yürütülen konferansın tipine göre tek bir media stream'ini yada daha çok sayıda media stream'ini işleyebilir [50]

Terminaller

Terminaller uç noktalarda gerçek zamanlı iki yönlü haberleşme sağlar. Tüm H.323 Terminalleri H.245, Q.931, Registration Admission Status (RAS) ve Real Time Transport Protocol (RTP) protokollerini desteklemelidir. H.245, kanal kullanım izni için, Q.931 çağrı kurulması ve sinyalleşme için, RTP gerçek zamanlı olarak ses paketlerinin taşınması için, RAS ise Gatekeeper ile haberleşme için kullanılan protokollerdir. Terminaller, uç noktalardaki başka bir H.323 terminali, ağgecidi veya MCU ile gerçek zamanlı, iki yönlü ses, görüntü ve veri haberleşmesi yapabilirler.Terminaller arasındaki iletişim sadece ses ieltimi olabileceği gibi, ses ve veri, ses ve görüntü, veri ve görüntü biçiminde de olabilir. Terminaller PC üzerinde yazılımsal olarak çalışan bir uygulama yada donanımsal bir telefon olabilir [51].

4.1.2.3 H323 mesajları

H.323'de konuşma yolunun kurulması esnasında çeşitli mesajlar kullanılır.Bu mesajlar aşağıda açıklanmaktadır.

RRQ (Request to Register): Uç birim noktalarının GK'ye kaydolma isteklerini gösterir.

RCF (Request Confirm): GK'ye kayıt işleminin gerçekleştiğini bildiren mesajdır.

RRJ (Request Reject): GK 'nin kayıt işlemini reddettiğini bildiren mesajdır.

ARQ (Admission Request): GW 'nin GK ile kontak kurup uç birim ile bağlantının nasıl yapılacağını sormasını sağlayan mesajdır.

ACF (Admission Confirm): GK 'nin hedef uç birimin bağlı bulunduğu GW'nin adresini kaynak taraftaki GW'e vermesi ve bağlantının kurulmasına izin vermesini sağlar.

RAS ARQ: Hedef GW'nin hedef uç birime konuşmaya müsait olup olmadığını anlamak için yolladığı mesajdır.

Çağrı ilerleme durumunda (Call Proceeding): Hedef GW , kendisine bağlı olan uç birime RAS ARQ gönderirken, kaynak GW' e Çağrı İlerleme Durumunda mesajı yollar. Böylece konuşma yolu kurulurken herhangi bir zaman aşımının gerçekleşmesinin önüne geçilmiş olur.

Bağlantı kuruldu (Connect): Hedef GW,kaynak GW 'e çağrının kurulduğunu bildirmek için bu mesajı yollar .

TCS (Terminal Capability Set): Bu mesaj GW'lerin yeteneklerini birbirlerine söylemelerine yarar.

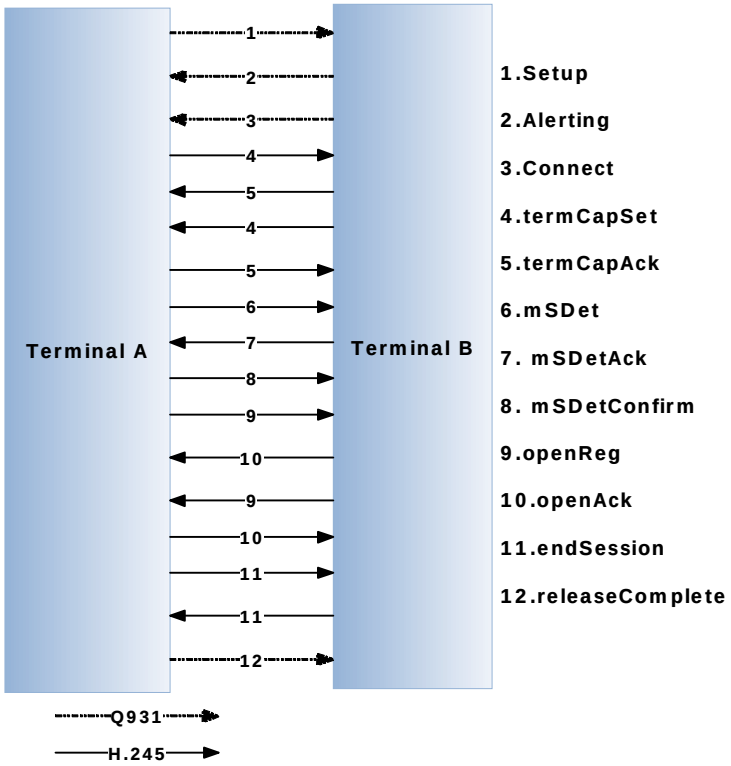
ACK (Acknowledge): TCS mesajını alan GW 'in bu durum hakkında karşı tarafa bildirmek için yolladığı mesajdır.

OLC (Open logical channel): Kaynak GW'in hedef GW ile mantıksal kanal açılması amacıyla yolladığı mesaj türüdür [6].

4.1.2.4 H.323 oturum kurulum örnekleri

1) Geçiş denetim sistemi (Gatekeeper) kullanmadan oturum kurulumu:

Şekil 4.5' de iki H.323 uç arasında geçiş denetim sistemi kullanılmadan oturum kurulumunun nasıl yapıldığı gösterilmektedir.



Şekil 4.5 : İki H.323 uç arasında geçiş denetim sistemi kullanılmadan oturum kurulumu

Çağrı akışı:

1)İletişim A terminalinden B terminaline hedef adresi içeren bir Setup (kurulum talep) mesajı göndermesi ile başlar.

2)Terminal B ,terminal A' ya bir Q.931 Alerting (uyarı) mesajı gönderir.

3)Eğer çağrı kabul edilirse terminal B, terminal A' ya Connect (bağlantı) mesajı göndererek cevap verir. Bu noktada çağrı kurulması işlemi tamamlanmış olur ve H.245 uzlaşma (negotiation) işlemi başlar.

4)Her iki terminalde terminal yeteneklerini (terminal capabilities) terminal CapabilitySet mesajları göndererek karşı tarafa bildirir. Terminal yeteneklerine örnek olarak media tipleri, kodlama yöntemleri verilebilir.

5)Terminaller bu mesajlara termCapabilitySetAck mesajları ile cevap verirler. Oturum sırasında herhangi bir anda terminal yetenekleri yeniden gönderilebilir.

6)İki terminal arasında Master/ Slave belirleme aşamasına geçilir. H.245 Master/Slave belirleme prosedürlerinin her ikisi de bir konferansa MC olarak servis verebilecek uç noktalar yada her ikisi de iki-yönlü iletişim kanalı açmaya çalışan uç noktalar arasında ortaya çıkabilecek anlaşmazlıkları gidermek için kullanılır. Terminal A, terminal B' ye mSDet (master/Slave Determination) mesajını gönderir.

7)Terminal B ,terminal A' ya mSDetAck mesajını gönderir

8)Terminal A ,terminal B' ye mSDetConfirm mesajını gönderir

9)Master/ Slave belirleme işleminden sonra, iki terminal de mantıksal kanal açmak için mesajlaşmaya başlarlar. Her iki terminal birbirlerine OpenReg mesajını gönderir.

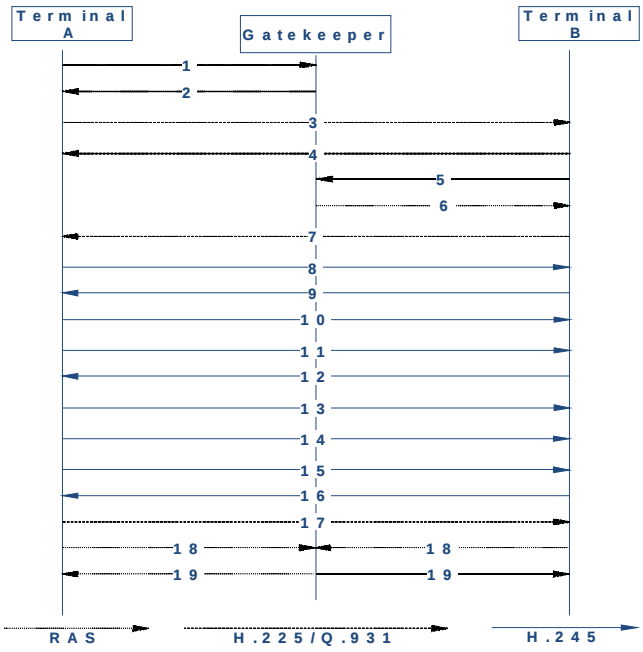
10)Her iki terminal birbirlerine OpenAck mesajını gönderir.

11)Oturumun (ya da iletişimin) kapatılmasına taraflardan birinin göndereceği endSession mesajı ile başlanır. EndSession mesajını alan taraf aynı mesajla cevap verir. Her iki terminal birbirlerine endSession mesajını gönderir

12)Terminal B, terminal A' ya ReleaseComplete mesajını gönderir ve oturum sonlandırılır [52, 53].

2)Geçiş denetim sistemi (gatekeeper) kullanılarak iki H.323 nokta arasında oturum kurulumu:

Şekil 4.6 ' de geçiş denetim sistemi kullanılarak otutum kurulumu gösterilmektedir.



Şekil 4.6 : Geçiş denetim sistemi kullanılarak iki H.323 nokta arasında oturum kurulumu

Çağrı akışı:

1)Kabul İsteği (Admission Reauest-ARQ): Terminal A'nın veri şebekeye katılma isteği.

2)Kabul Onayı (Admission Confirm-ACF): Terminal A'nın veri şebekeye katılma onayı.

3)Kurulum (Setup):Terminal A'ın kabul etme durumu.

4)Çağrı İşleme (Call Proceeding): Kurulum başlangıcı kurulumu.

5)Kabul İsteği (Admission Reauest-ARQ): Terminal B'nın veri şebekeye katılma isteği.

6)Kabul Onayı (Admission Reauest-ARQ): Terminal B'nın veri şebekeye katılma onayı.

7)Bağlantı Uyarısı (Alert): Terminal B uyarılmıştır."telefon çalıyor"!

8)Terminal Niteliklerinin Değişimi (Terminal Capability Set): Terminal A niteliklerinin tanımlanması

9)Terminal Niteliklerinin Değişimi + Onay (Terminal Capability Set+ Acknowledgement): Terminal B niteliklerinin tanımlanması ve Terminal A niteliklerinin onayı (Terminal B tarafından).

10)Terminal Niteliklerinin Değişimi Onayı (Terminal Capability Set Acknowledgement): Terminal B niteliklerinin onayı (Terminal A tarafından).

11)Mantıksal Kanalın Açılımı (Open Logical Channel): Ses/Görüntü/Veri bilgilerinin iletimi için mantıksal kanalın açılması.

12)Mantıksal Kanalın Açılımı + Onay (Open Logical Channel + Acknowledgement): Ses/Görüntü/Veri bilgilerinin iletimi için mantıksal kanalın açılması onayı (Terminal B tarafından)

13)Mantıksal Kanalın Açılımı + Onay (Open Logical Channel + Acknowledgement): Ses/Görüntü/Veri bilgilerinin iletimi için mantıksal kanalın açılması onayı (Terminal A tarafından)

14)Bilgi Dizisi Akışı (Media Stream): RTP protokolü kullanılarak bilgi dizisinin iletimi.

15)Mantıksal Kanalın Kapanışı (Close Logical Channel): Terminal A tarafından mantıksal kanalın kapanışı.

16)Mantıksal Kanalın Kapanışı + Onay (Close Logical Channel + Acknowlegement): Terminal tarafından mantıksal kanalın kapanışı onayı.

17)Oturumun Tamamlanması (Relase Complete): Çağrının tamamlanması.

18)Sonlanma İsteği (Disengage Request-DRQ): Terminal çağrının sonlanması.

19)Sonlanma Onayı (Disengage Confirm): Terminal çağrının sonlanması onayı

4.1.2.5 H.323 versiyonları

Mevcut H.323 protokolüne zamanla değişik özellikler katılmıştır.Aşağıda H.323 'ün versiyonları ve eklenen yeni özellikler açıklanmaktadır.

4.1.2.5.1 H.323 versiyon 2 (v.2)

Versiyon 2 ile H.323 protokolüne eklenen aşağıda verilmiştir [20].

Hızlı bağlanma (Fast connect)

Çağrının kurulma hızını artırır. Böylece çağrı kurulum süresi kısalır.

Ek hizmetler (Supplementary services)

H.450 serisi standartlarda da sisteme entegre olmuştur. Bu standartlardan H.450.1 bu ek servislerin kontrolünü ve işaretleşmesini sağlar. H.450.2 (çağrı transferi) bi A kullanıcısının B kullanıcısıyla olan görüşmesini B ile C kullanıcısı arasındaki görüşmeye dönüştüren ve bu transferin A kullanıcısı tarafından seçilmesine izin veren standarttır. H.450.3 ise; gelen çağrının başka bir hedefe yönlendirilmesini sağlar.

H.235 güvenliğini destekleyen H serisi çoklu ortam terminalleri:

Bu özellik doğrulama, güvenilirlik, gizlilik ve reddedilmeme gibi işlemleri tanımlar. Doğrulama sadece yetkili kullanıcıların haberleşebilmesini sağlayan mekanizmadır. Güvenilirlik, gelen paketle kaynaktan yollanan paketin şifreleme yoluyla sadece hedeflenen alıcı tarafından alımasını sağlar. Reddedilmeme ise kullanıcının bir konferans görüşmeye katılmak istadğinide onun o görüşmeye katılma isteğinin reddedilmesini garantiler.

Üst üste bindirilmiş gönderim (Overlapped sending):

GK'nin hafızasında bir rota belirlenemezse kullanıcıdan ekstra adres enformasyonu ve yönlendirme enformasyonu istenir. Böylece bağlantı daha hızlı kurulmuş olur. Diğer bir değişle, arayan tarafın direkt olarak GK ile temas kurması sağlanır.

Alternatif Uç birim noktası (Alternate endpoint):

Yedekleme amacaıyla bir uç birim noktasına kendisine alternatif bir bitim noktası adresi veya ikincil şebeke arayüzü tanımlaması olanağı verir.

Şebeke sorumlusu fazlalığı (GK redundancy):

Birincil GK hizmet dışı kaldığında kullanıcının ikincil ve üçüncül GK'lar tanımlamasına olanak verir. Böylece verilen hizmetin sürekliliği artar.

4.1.2.5.2 H.323 versiyon 3 (v.3)

Bu sürüm daha önceki sürümlere ek olarak hızlı bağlantı kurulması için TCP yerine UDP üzerinden işaretleşmenin yapılmasını sağlar. Binlerce çağrının üzerinden geçtiği şebekelerde UDP uzun çağrı kurulum süreleri oluşmasına neden olur. H.323 versiyon3 sürümü ile PSTN şebekelerine de entegre olmuştur ve üç adet yeni ek servise destek vermeye başlamıştır. Bunlar H.450.4 ile o anda etkin olan çağrıyı belirli bir süre bekletilip; daha sonra önceden belirlenmiş farklı telefonlar tarafından karşılanabilmesini sağlar. H.450.6 ile aranan tarafın meşgul olduğu durumlarda gelen

çağrının bekletilmesi sağlanır. H.450.7 kullanıcıya kendisini bekleyen mesajların olduğu bildirilir [1].

4.1.2.5.3 H.323 versiyon 4 (V.4)

H.323 Versiyon 4 (V.4) çoklu akış iletimine izin verir. Bu iletim ses ile görüntü verilerinin birleştirilmiş bir şekilde aynı anda akmasını sağlar. Bu özelliğin sağladığı en büyük fayda ise; ses ve görüntünün önceki sürümlere göre çok daha kolay bir şekilde senkronize olabilmesidir. Bu versiyon ile eklenen yeni özellikler aşağıda anlatılmıştır.

- H.450.8:Kullanıcı bilgilerinin karşı taraftaki uç birim noktasına gönderilmesini sağlar.
- H.450.9: aranan taraf meşgul olduğunda veya cevap yoksa çağrının otomatik olarak kesilmesini sağlar.
- H.450.10: Arayan tarafın isteğine bağlı olarak aranan taraf meşgulse, arayanın hatta tutulması ve meşgul durumun sona ermesiyle birlikte çağrının kurulmasını sağlar.
- H.450.11: Arayan A kullanıcısının, aranan B kullanıcısı C kullanıcısıyla görüşme halindeyse araya girmesini sağlar [20,54].

4.1.3 H.323 ve SIP protokollerinin karşılaştırılması

H.323 ve SIP VoIP uygulamalarında birbirleri ile yarışan iki protokol konumundadır. H.323 çok daha önce geliştirilen bir protokol olmanın avantajı ile yaygın olarak birçok ağda kullanılmaktadır. Diğer taraftan SIP'teki gelişmeler yakın gelecekte SIP'in, H.323'ün yerini alacağını işaret etmektedir.H.323 bir örtü tanımlamasıdır, tek başına bir protokol değildir. Yeni bir servis yaratmak için diğer protokollerin nasıl kullanılacağını bildirmektedir. Yapısında H.225 RAS sinyalleşmesi, Q.931 çağrı sinyalleşmesi (call signalling), H.245 kontrol sinyalleşmesi RTP gerçek zaman protokolü, ses ve görüntü sayısallaşması ve sıkıştırılması işlemlerini yerine getiren birçok standartı barındırmaktadır [31,55].

Daha sonra protokol daha fazla arama seçeneklerini ve aynı yerel alan ağında (LAN) bulunmayan kullanıcıları da kapsayacak şekilde genişletilmiştir. Bu, protokolün ve altında yatan uygulamaların daha da karışmasına sebep olmuştur. H.323 kodlama ve veri alanı için geniş bir hafızaya ihtiyaç duyan çok geniş bir protokol kümesidir. Mesajlaşma biçimi olarak Abstract Syntax Notation'ı kullanmaktadır (ASN.1). ASN.1 ITU tarafından yaygın olarak kullanılmakla beraber, kullanıcıların bu protokolü doğrudan okuması ve anlaması zordur. ASN.1'in anlamlı olabilmesi için ASN.1 mesajlarını çözen daha kolay ağ testi ve hata ayıklama (Debugging) özelliklerini destekleyen ve insanın anlayabileceği yapıya çeviren özel yazılımlara ihtiyaç duyulmaktadır . H.323'ün karmaşıklığı basit bir çağrı kurulumunda bile çok sayıda mesajın gönderilmesine neden olur. Ağ farklı gatekeeperlar tarafından yönetilen segmentlere ayrılabilir. Bir çağrı kurulumunda birden çok gatekeeper işin içine girdiğinden mesajlaşma çok daha karmaşık bir durum almaktadır. H.323 alternatif olarak Internet Engineering Task Force (IETF) çağrı kurulumu ve VoIP uygulamalarının yönetimine destek sağlayacak çok daha basit ama daha güçlü bir protokol oluşturmak üzere çalışmalara başlamıştır. Oluşturulan çalışma grubu IETF tarafından 1999'da RFC 2543'de yayınlanan SIP protokolünü geliştirmiş. IETF ve ITU protokol özelliklerini geliştirirken çok farklı yaklaşımlarda bulunmuşlardır. ITU'nun protokolü geliştirirken ki yaklaşımı herhangi birini ihtiyaç duyacağı bütün servisleri ele alıp bunlara çözüm bulacak şekilde tanımlamalarda bulunması şeklinde olmuştur. Uzun bir süreç sonunda geliştirilen, standartlar çok genişlemiş ve fazla detay içermiştir. Birçok VoIP uygulaması için anlaşılması güç elemanlar yerleştirilmiş ve detaylı tanımlamalar yapılmıştır. IETF, çok daha az mesajlaşma adımı kullanmıştır. SIP RFC'sinde VoIP dışındaki diğer uygulamalar için detaylı bir tanımlama yapılmamıştır. SIP, H.323'e göre esnek yapısıyla çok daha hızlı geliştirilmiş ve çok daha az kod içeren bir protokol ortaya çıkmıştır [49,20].
Aşağıda SIP ve H.323 protokolleri birtakım kriterlere göre karşılaştırılmaktadır.

4.1.3.1 Karmaşıklık

H.323 çok karmaşık bir protokol grubudur. H.323 ile ilgili yapılan en büyük eleştiri bu protokolün network üzerinde gereksiz yere bir sürü mesaj göndermesi

yönündedir. H.323 geniş bir sistem için ölçeklenebilir bir yapı sunmaktadır. Çok yoğun oturumların kurulduğu bir sistemde çok sayıda mesajın gidip gelmesi sistem performansını olumsuz etkilemektedir. Bu da ses kalitesinin düşmesine sebep olmaktadır.Bir çağrı kurulması için 16 adet mesajın LAN üzerinde değiş tokuş edilmesi gerekmektedir. Bu sayı, terminallerin farklı geçiş denetim sistemleri tarafından yönetilen networklerde olması durumunda belirgin bir biçimde artmaktadır. Aynı çağrı kurulumu için SIP kullanıldığında gerekli olan mesaj sayısı 4 adettir [10,56].

SIP kullanıldığında H.323'e göre gerekli olan kullanıcı uygulaması çok daha basit olmaktadır. H.323'de terminaller konfigürasyon bilgisini karşılıklı olarak gönderdikleri mesajlarla oluşturmaktadır. Bu sayede karşılıklı uçlar kullanılacak parametreler konusunda anlaşmaktadır. SIP'de bu mesajlaşma farklı mesajlara ihtiyaç duymadan sadece Talep ve Cevap olarak (200:OK) mesajları ile tamamlanmaktadır. İki protokolde de ses paketlerinin ilerlemesi için uçlar arasında mantıksal bağlar kurulmaktadır. SIP'de bütün bu bilgiler aynı talep ve cevap mesajının içerisinde yer almaktadır.SIP mesajları H.323'e göre daha fazla yer kaplamaktadır. Ama bir çok küçük çerçeve göndermektense birkaç büyük çerçeve göndermek her zaman daha performanslı bir yöntemdir [57].

H.323' de basit çağrıların daha hızlı kurmak için bir takım yöntemler geliştirilmiştir. Bu yöntemlerden bir tanesi "Hızlı Başlangıç" yapısında iki tarafın da mesajlarında detaylı bilgi göndermesi ve haberleşme için daha önceden belirlenmiş pottların kullanılması ile gönderilecek mesaj sayısı azaltılmıştır. Ama bu da aynı anda birçok noktaya "Hızlı Başlangıç" ile çağrı kurabilmesi için diğer terminallerin adreslerini bilmesi ve bu uçlarla ilgili gerekli konfigürasyon bilgisini barındırması gerekmektedir. Diğer taraftan bütün terminaller "Hızlı Başlangıç" özelliğini desteklememektedir. Bu da protokolün faydalarını sınırlamaktadır. "Hızlı Başlangıç" daha çok kullanıcıların birbirini tanıdığı ofis ortamları gibi kapalı sistemler için geliştirilmiştir. Hızlı Başlangıç sistemi ile ilgili diğer bir problem de faturalandırılabilir durum tespitinin yapılamamasıdır. Bu sistemin çağrının gerçekten kurulup kurulmadığını tespit edemeyeceği ve ücretli çağrıları desteklemediği belirtilmektedir. SIP'de cevap mesajlarında iletilen 200:OK mesajı kaynağın karşı

uçtaki çağrıyı kabul ettiğini belirtir. H:323 Hızlı Başlangıç'ta ise eğer CONNECT mesajı kaybolursa çağrı kurulmuş olsa bile karşı tarafın bağlantı kurulduğunu anlamaması ve çağrının bir süreliğine faturalandırılmaması söz konusu olabilir [29].

4.1.3.2 Boyut ve performans

SIP daha basit bir protokol olduğundan H:323 kurulumuna göre daha az kod gerektirir. Bu sayede gerekli kod satırı sayısının yanı sıra hem protokol kümesi için hem de kullanıcı uygulaması için daha düşük dinamik belleğe ihtiyaç duyar. Çağrı kurulması için işlemcinin daha az sayıda mesajı ele alması SIP'in H.323'e göre daha hızlı bir protokol olmasını sağlar. SIP'in saniyede ele alabileceği çağrı sayısı H.323'e göre daha fazladır ve SIP aynı sayıda çağrıyı ele almak için daha düşük kapasiteli işlemciye ihtiyaç duymaktadır. SIP'in sistem kurulum maliyeti daha düşüktür [58].

4.1.3.3 Özellik karşılaştırması

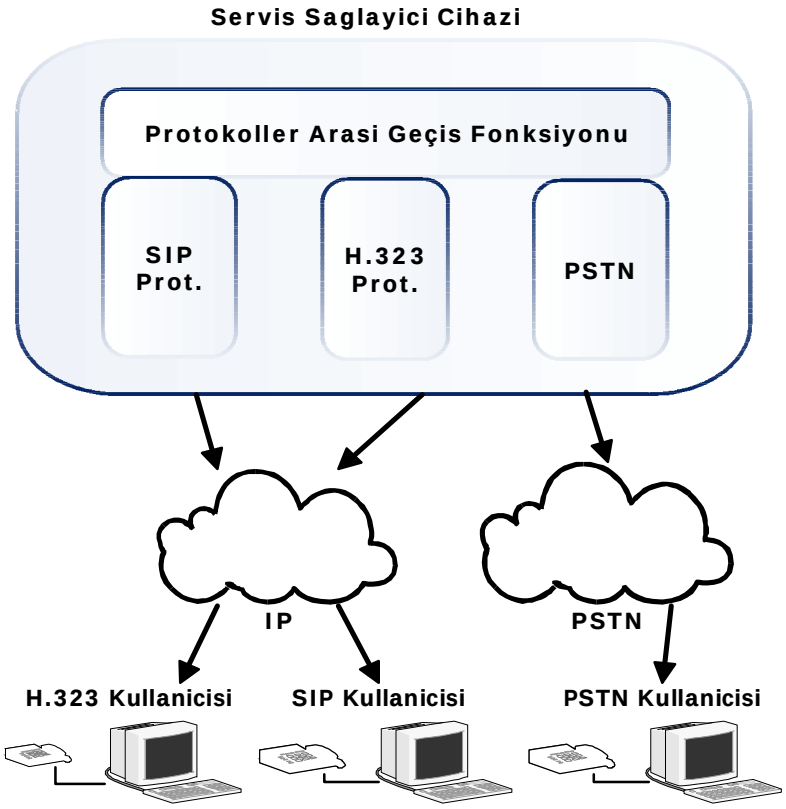
Mesaj biçimi olarak ASN.1 kullanılması her protokol değişikliğinde H.323 hata ayıklama (Debugging) araçlarında güncellemelere ihtiyaç duymaktadır. SIP mesajları metin tabanlı olduğu için hata ayıklamada güncelleme ihtiyacı duymazlar. İki protokolde de kullanıcılar kendi eklentilerini ekleyebilirler ama H.323'de bu eklentiler ASN.1 formatında olmalıdır. SIP'de ise eklentiler geliştirilmesi çok daha kolay olan metin mesajları biçiminde eklenebilir. H.323 elemenlarının hepsi durum (state) bilgisini tutmak zorundadır. SIP'de ise sadece UA'nin bunu yapması zorunludur. Bazı durumlarda Proxy sunucusu da durum bilgilerini tutabilir. Bu H.323'lere ek yük getirmekle beraber H.323 kodunun karmaşıklığını arttırmaktadır İki protokolde güçlü telefon özelliklerini destekler. Bir protokolün diğeri üzerinde belirgin bir avantajı yoktur [11].
Tablo 4.5 'de H.323 ve SIP protokollerinin özellik karşılaştırılması gösterilmektedir.

Tablo 4.5: H.323 ve SIP protokollerinin özellik karşılaştırılması

Ölçüt	H.323	SIP
Karmaşıklık	Çok karmaşık	Basit
Mesaj kümesi	Birçok mesaj iletimi	Az sayıda mesaj iletimi
Hata ayıklama	Protokol genişlediğinde araçlarda değiştirilmeli	Basit araçlar sürekli kullanılabilir
Genişleyebilme	Genişleyebilir	Çok genişleyebilir
Durum bilgilerini tutmak zorunda olan elemanlar	Terminaller, Geçiş Denetim Sistemleri, MCU, Ağgeçitleri	UA ve bazı Proxy Sunucular
İşlemci kullanımı	İşlemciye çok fazla yük bindirir	İşlemciye az yük bindirir
Telefon özellikleri	Güçlü	Güçlü
Kullanıcı uygulamaları	Karmaşık	Basit
Kod büyüklüğü	Uzun	Kısa
Dinamik hafıza kullanımı	Geniş	Orta

H.323 çok yaygın bir kullanım alanına sahiptir. Kullanıcılar VoIP ürünleri üretirken cihazlar ve servisler ile uyumlu bir yapı kurmak istemektedirler. SIP ürünlerine H.323 özelliklerini de eklemek SIP temelli ürün maliyeti yanında çok büyük yatırım maliyetlerinide beraberinde getirmektedir. (H.323 lisanslaması, kullanıcı uygulamalarının geliştirilmesi, test edilmesi vs.) SIP ve H.323'ün uyumlu hale getirilmesi uzun mühendislik çalışmalarına gereksinim duyar. SIP daha iyi bir protokol olsa da H.323 ağlarla konuşabilmek için H.323 gereklidir. Günümüzde kullanılan H.323 ürünler aynı zamanda SIP'i de desteklemektedir ve SIP/H.323 entegrasyonu sağlanmıştır. Yani bir protokolü kullanan bir network diğer protokolün ürünleri ile iletişim kurabilmelidir [59,60].

Şekil 4.7' de SIP protokolü ile H.323 protokolü arasındaki bağlantının nasıl sağlandığı gösterilmektedir.



Şekil 4.7: SIP ve H.323 protokolleri arasındaki bağlantının sağlanması

4.2 VoIP Veri Aktarım Protokoleri

Veri aktarım fazında başlıca üç protokol kullanılır. Bunlar; kaynak ayırmak için kullanılan RSVP (Kaynak Ayırma Protokolü), gerçek zamanlı veri akışı için kullanılan RTP (Gerçek Zaman Protokolü) ve bu protokolün kontrolünü sağlayan RTCP (Gerçek Zaman Kontrol Protokolü) olarak adlandırılır.

Sistemde veri aktırılmaya başlanmadan önce SIP veya H.323 ile işaretleşme yapılır. Daha sonra RSVP ile sistem kaynaklarının bir bölümü VoIP görüşmesi için ayrılır. Sonra uç birimler RTP ve RCTP kullanılması için hangi UDP portlarının kullanılacağından haberdar olurlar [54] .

4.2.1 Kaynak ayırma protokolü (RSVP)

RSVP İnternet üzerinden oturum açmak için gerekli olan kaynakların ayrılması için kullanılır. IP bağlantısız bir protokol olduğu için yol kurulması gibi bir durum oluşmaz. Dolayısıyla bu yollar için belirli bir band genişliği ayrılmaz. Kurulan yollardaki trafik akışı için gerekli olan band genişliğini sağlamak için RSVP tasarlanmıştır. Bütünleşik servisler araşebekesi için tasarlanmış bir kaynak rezervasyon protokolüdür. RSVP yönlendirme faaliyetlerinde bulunmasa da ICMP ve IGMP'de olduğu gibi taşıma mekanizması olarak IP'nın çeşitli sürümlerini kullanır. IP'de olduğu gibi kendi mesajları için lokal yönlendirme tablolarına bakarak yol çizer. RSVP bir multicast gruba ilk katılımda kullanmak için IGMP'yi kullanır ve multicast grubu için kaynak ayırma protokollerini çalıştırır. QoS rezervasyonlarının kaynakların yeniden atanmasıyla desteklenebildiği ortamlar için kullanışlıdır. Bir host , veri kaynağından gelen belli bir veri akışı için şebekeden özel bir QoS istemek amacıyla RSVP kulllanır.Alıcı ile gönderici arasındaki tüm host' lar , router'lar ve diğer şebeke elemanları RSVP desteklemek zorundadır. RSVP'nin dezavantajı paketlerin bir öncelik sırasında analizi ve işlenmesi amacıyla router' lar için bilgi işleme kaynaklarının gerekmesidir. Bunu kolaylaştırmak için çoklu protokol etiket etiket anahtarlama (MPLS) gibi yaklaşımlar geliştirilmiştir..Bir başka araştırma alanı , alternatif ve sabit yollar sağlayan yönlendirme servislerini kullanmak için RSVP' nin genişletilmesidir. Orijinal RSVP, geniş şebekeler için fazla uygun değildir. Çeşitli trafik akışları için kaynakların rezerve edilmesi, tüm WAN router' ları ve diğer Katman 3 linkleri boyunca her akış için "durum'' bilgisinin (oturum ayarları ve band genişliği bilgisi gibi) korunması gerekir. Bu durum , kaynaklara ek trafik için kapasiteyi azaltacak olan büyük bir yük oluşturabilir [16].

4.2.1.1 RSVP'nin çalışması

RSVP'nin iki çalışma şekli vardır. Bunlar Yol kurma ve Yer ayırmadır.

Yol kurma : Bu çalışma şeklinde , RSVP tek (unicast) ve çoklu (multicast) çalışma prosedürlerinden birini işletir.IGMP multicast gruba ilk katılım anında kullanılır. Daha sonra kaynak ayırma prosedürleri çalıştırılır. RSVP trafiği alan tarafların akış

için servis kalitesi isteklerine ihtiyaç duyar. Alıcı tarafa çalışan uygulama hangi QoS profilinin RSVP'ye geçirileceğine karar verir. İstek mesajının alınmasından sonra RSVP veri akışının gerçekleşeceği tüm düğümlere istek (request) mesajları yollar. RSVP ayrıca yönlendiriciler tarafından dağıtılan QoS istek mesajlarının düğümlere ulaşmasını ve her düğümde bu istek mesajları için gerekli kaynağın ayrılması için kullanılır [20,61].

Yer ayırma : Bu çalışma şeklinde ise alıcı taraf yollayıcı tarafa ve ara elemanlara (yönlendiriciler gibi) kendi QoS gereksinimlerini haber verir. Rezervasyon çalışma şekli olarak da geçer.

4.2.1.2 RSVP mesajları

RSVP mesajlarının başlık kısmı ortaktır. Mesajın gövdesi farklılık gösterebilir. Bu farklılığın nedeni de gövdede kullanılan RSVP nesneleridir.

4.2.1.3 RSVP başlık formatı

Tablo 4.6' da ortak RSVP başlık formatı verilmiştir. Başlık standart olarak 32 bit uzunluğundadır.

Tablo 4.6 : Ortak RSVP Başlığı

<table>
<tr><td>0</td><td>1-2</td><td>3</td><td>4</td><td>5-6</td><td>7</td><td>8</td><td>9-14</td><td>15</td><td>16</td><td>17-30</td><td>31</td></tr>
<tr><td colspan="3">Sürüm</td><td colspan="3">Bayrak</td><td colspan="2">Mesaj Türü</td><td colspan="4">Kontrol Alanı</td></tr>
<tr><td colspan="6">Yaşam Süresi</td><td colspan="2">Rezerve</td><td colspan="4">RSVP Uzunluğu</td></tr>
</table>

Sürüm: Kullanılan protokolün sürüm numarasıdır.

Bayraklar: Şu ana kadar bir bayrak tanımlanmamıştır.

Mesaj türü: 7 farklı mesaj türü vardır. Bunlar aşağıda verilmiştir.

Yol Kurma ,Yol Ayırma, Yol Kurma Hatası, Yol Ayırma Hatası, Kurulan Yolu Çözme, Ayrılma Yolu Çözme , Yol Ayırma Onayı'dır.

RSVP kontrol alanı: Mesajın birli tümleyenlerinin toplamının tümleyenidir.

RSVP uzunluğu: Toplam RSVP mesaj uzunluğunun (başlık+gövde) byte cinsinden ifadesidir.

Yaşam süresi: Yollanan mesajın şebekede ya da düğümler arasında yok edilmeden ne kadar süre kalabileceğini belirler [9,20].

4.2.1.4 RSVP nesne formatı

RSVP nesneleri 32 bit ve katları büyüklüğündeki kelimelerden oluşurlar. Bu nesneler RSVP gövdesini değiştirerek yollanan mesajın farklı görevleri yapmasını sağlarlar. Tablo 4.7 'de RSVP nesne formatı verilmiştir.

Tablo 4.7 : RSVP nesne formatı

<table>
<tr><td>0</td><td>1-2</td><td>3</td><td>4</td><td>5-6</td><td>7</td><td>8</td><td>9-14</td><td>15</td><td>16</td><td>17-30</td><td>31</td></tr>
<tr><td colspan="3">Sürüm</td><td colspan="3">Bayrak</td><td colspan="2">Mesaj Türü</td><td colspan="4">Kontrol Alanı</td></tr>
<tr><td colspan="6">Yaşam Süresi</td><td colspan="2">Rezerve</td><td colspan="4">RSVP Uzunluğu</td></tr>
</table>

Uzunluk: Nesne alanının genişliği yazılır.

Sınıf numarası: Nesnenin hangi sınıfa ait olduğunu bildirir**.**

Sınıf türü: Aynı sınıftaki nesnelerin özelliklerini belirler.

4.2.1.5 RSVP nesne türleri

RSVP mesajlarındaki enformasyon nesneler şeklinde kodlanmıştır ve bu enformasyon sunucular, kullanıcılar ve düğümler arasında değiş-tokuş edilirler. RSVP aşağıdaki nesne sınıflarını destekler.

NULL: Bu nesne alıcıda gözlemlendiğinde içeriği ihmal edilir.

SESSION: Bu nesne IP adresini, IP sürüm numarasını ve hedef makinede kullanılan port numarasını içerir.

RSVP_HOP: Mesajı yollayan düğümün IP adresini ve lojik arabağdaşım kontrolünü (LIC) tutar.

TIME_VALUES: Mesajın üreticisi tarafından tazelenme periyodunu tutar.

STYLE: Rezervasyon stilini ve bu stile özgü enformasyonu tutar.

FLOWSPEC: Bir yol ayırma mesajında istenen servis kalitesini alması gereken veri paketlerinin alt kümesini tanımlar.

SENDER_TEMPLATE: Yollayan tarafın IP adresini içerir. Bazı e bilgiler de bulunabilir.

ADSPEC: Yol kurma mesajındaki OPWA verisini taşır. OPWA ile RSVP kontrol paketleri yollayan düğümler kendisinden daha alt akıştaki (downstream) düğümlere ulaşırken kullanılan yolların ve ulaşılan düğümlerin gerekli QoS değerlerini sağlayıp sağlamadığını tahmin etmeye yarar [12,20].

ERROR_SPEC:Yol kurma hatası, yol ayırma hatası ve yol ayırma onayı mesajları içindeki hataları tanımlar.

POLICY_DATA: Sistem yöneticisi tarafından izin verilip verilmediğine karar veren lokal poliçe enformasyonunu tutar. Bu poliçe verisinde kotalar, hesap numaraları, limitler ve kullanıcı bilgileri yer alır.

INTEGRITY: RSVP mesajının içeriğini onaylamak için mesajı üreten tarafı yetkilendirmek için şifrelenmiş veriyi taşır. Özellikle RSVP komşularının gizli bir enformasyonu paylaşmaları amacıyla kullanılır.

SCOPE: RSVP mesajının yollanacağı kullanıcıların listesini tutar. Yol ayırma, yol ayırma hatası ve yol çözme mesajlarında kullanılabilir.

RESV_CONFIRM: Bir onay mesajı isteyen tarafın IP adresini taşır.

4.2.2 Gerçek zaman taşıma protokolü (RTP)

RTP (Gerçek Zaman Taşıma Protokolü) uçtan uca gerçek zamanlı ses ve görüntü verisinin taşınmasını sağlayan iletim protokolüdür. İlk olarak çok katılımcı çoklu ortam konferanslarındaki ihtiyaçları karşılamak için ortaya atılmıştır. Veri paketlerinin iletimini garanti eden bir protokol değildir.. RTP veri iletimini gözlemlemek için yardımcı bir protokol olan RTCP ‘den faydalanır. RTP veri akışının senkronizasyonunu kontrol eder. RTP, UDP üzerinde çalışır. UDP’nin çoğullama ve başlık kontrol mekanizmalarını kullanır. Buna rağmen RTP başka alt seviye protokolleriyle de çalışabilir. RTP’nin bir diğer önemli özelliği ise çoklu (multicast) ortamlarda birden çok kullanıcının veri transfer işlemini gerçekleştirebilmesidir. Bu şekilde seli ve görüntülü konferans uygulamaları gerçeklenebilir. RTP sahip olduğu dizi numaraları sayesinde veriyi alan tarafa ses veya görüntüyü tekrar birleştirme olanağı sağlar. RTP paket kayıplarını telafi etmek gibi bir yapıya sahip değildir. Kayıp paketlerin tekrar iletilmesi için H.261 RTP başlığı kullanılabilir. Kayıp paketlerin telafisi için diğer bir yöntem de düşük bit oranı ile beraber aynı ses paketlerinin yedekleme amaçlı birden çok kez iletilmesidir [13,62].

4.2.2.1 RTP mesaj formatı

Tüm RTP mesajları aynı formatta oluşturulurlar. Bu mesajlar yollanırken standart olarak 5004 numaralı UDP portunu kullanır. Mesajlar farklı türlerdeki yükleri taşımak için tasarlanmışlardır. Bu yükler arasında G.729 gibi ses kodek yapıları olabildiği gibi JPEG görüntü standardı da bulunabilir. RTP Protokol Veri Birimi (PDU) IP ve UDP üzerinden taşınır. Bu protokollerin başlık enformasyonu PDU'nun veri alanında kendilerine yer bulurlar. Tablo 4.8 'de RTP mesaj formatı gösterilmiştir.

Tablo 4.8 : RTP mesaj formatı

0	1	2	3	4-7	8	9	10-14	15	16	17-30	31
V=2		P	E	CC	M	PT			Dizi Numarası		
Zaman Damgası											
Senkronizasyon Kaynak Tanımlayıcısı(SSRC)											
El Kaynak Tanımlayıcısı (CSRC)(Değişken)											
Veri (Değişken)											

V (Versiyon): RTP'nin sürüm numarasını belirtir.

P (Padding): Bu bayarak 1 yapıldığında çeşitli bir ekleme oktetlerinin bulunduğunu gösterir.

E (Extension): RTP başlığından sonra başka bir başlığın bulunduğunu gösterir.

CC (Contibuter conut): Mesajdaki ek kaynak tanımlayıcılarının sayısıdır.

M (Marker): Veri akımının sınırlarını belirlemek için kullanılır.

PT (Payload type): Hangi tür yükün (G.729 ses veya JPEG gibi) taşınacağını belirtir.

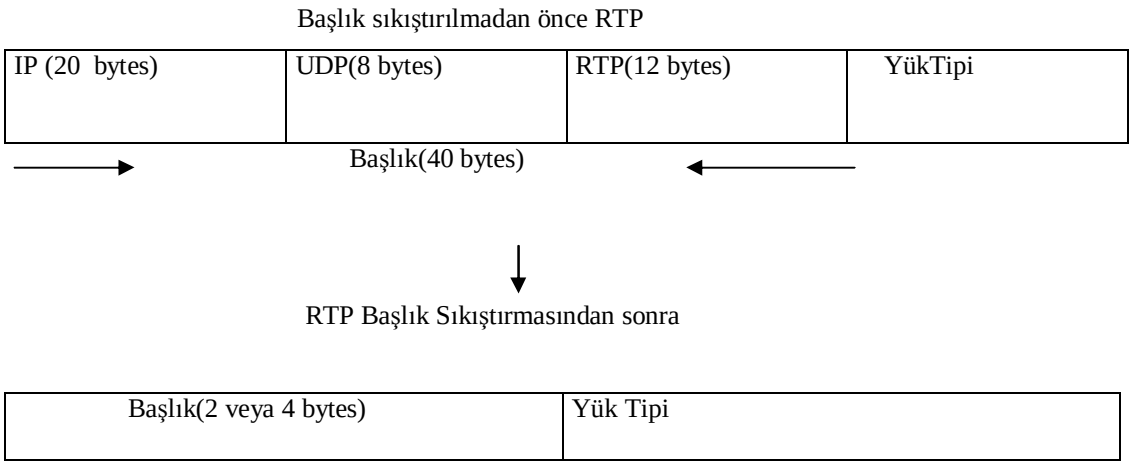
Dizi numarası: Her RTP paketi yollandığında değeri 1 artan bir numaradır.

Zaman damgası: İlk oktetin örnekleme anını RTP ve paketine yansıtır. Örnekleme anı monoton olarak artan bir saatten alınabilir. Böylece senkronizasyon ve jitter hesaplamaları yapılabilir. Eğer RTP paketleri periyodik olarak üretiliyorsa sistem saati yerine örnekleme saati de nominal örnekleme anının tespiti için kullanılabilir.

SSRC: Senkronizasyon kaynağını belirtir. Bu belirteç rastgele olarak seçilir. Aynı oturumdaki senkronize olacak her kaynağın SSRC numarası farklı olur.
CSRC: CSRC listesi paket yüküne katkıda bulunan kaynakları tanımlar. Bu kaynakların sayısı CC alanında verilir. Eğer 15'den fazla ek katkı kaynağı varsa o duruma sadece 15 katkı kaynağına yer verilir [20,60].

4.2.2.2 RTP başlık sıkıştırması (cRTP)

Bütün ses paketleri iki bileşenden oluşmaktadır.Bu bileşenler Ses paketleri ve IP/UDP/RTP başlıklarıdır. Ses örnekleri DSP tarafından sıkıştırılmakta ve bu örnekler kullanılan kodek türüne göre boyut olarak değişse de IP/UDP/RTP başlıkları sabit olarak 40 byte uzunluğundadır. Bu başlık alanı varsayılan 20 byte uzunluğunda bir G.729 çağrısı ile karşılaştırıldığında belirgin bir yük getirmektedir. Noktadan noktaya iletişimlerde cRTP (compressed RTP) kullanılarak 40 byte'lık bu alan 2 yada 4 byte 'a kadar sıkıştırılabilir. Bu sıkıştırma belirgin derecede bant genişliği kazancı sağlamaktadır. Bir G.729 çağrısı RTP ile iletildiğinde 24 Kb harcarken cRTP kullanıldığında 12 Kb ile iletilebilmektedir. CRTP VoIP çağrılarını noktadan noktaya sıkıştırdığı için IP bağlantısında iki uçta da cRTP konfigüre edilmelidir.CRTP konfigürasyonu ile paket boyutları belirgin bir şekilde azalmaktadır. CRTP kurulumu için ağ üzerindeki tüm yönlendiricilerin başlık sıkıştırma yeteneğine sahip olması ve sıkıştırılmış başlığı anlayabilmesi gerekir [12].
Şekil 4.8 'da RTP protokolünü kullanılarak iletilen ses paketinin başlık sıkıştırılması yapıldıktan sonraki durumu gösterilmektedir.



Şekil 4.8: RTP başlık sıkıştırması (cRTP)

4.2.3 Gerçek zaman kontrol protokolü (RTCP)

Gerçek zaman kontrol protokolü (RTCP) veri iletimi kalitesi konusunda geribildirim sağlamak için kullanılır. Gönderici ve alıcı olarak kontrol paketlerinin iletilmesine dayanan bir protokoldür. Gönderici ve alıcı raporları tüm kullanıcılara periyodik olarak iletilir. Bu raporlar paket sayısı, paket kayıp oranları, paket iletim gecikmeleri ve gecikme süresi farkını içerir. RTCP ağın mevcut durumu hakkında önemli bilgiler sağlar. Bazı uygulamalar bu bilgileri iletim oranları ve yığılmayı azaltmak için kullanmaktadır. RTCP 'nin dört temel işlevi vardır [10].

1) RTCP'nin birinci işlevi iletimin kalitesiyle ilgili geribildirimde bulunmaktır. Uygulama bilgiyi kullanarak, akış ve yığılma kontrolü yapar. Ayrıca bu bilgiyi çıkan sorunların giderilmesinde de (Diagnostic) kullanır [63].

2) RTCP bir tanımlayıcı sayesinde farklı ortam verilerinin gruplandırılabilmesini sağlar. RTP' nin böyle bir işlevi olmadığından, oldukça gerekli bir işlevdir.

3) RTCP paketleri periyodik olarak gönderildiğinden , her katılımcı o an oturumdaki tarafların sayısını gözlemler. RTP verisi bu amaçla kullanılmaz. Çünkü bazı taraflar geri göndermez sadece alır.

4) RTCP' nin diğer bir işlevi de bir oturumdaki taraflar için bilgi sağlamasıdır. Bu bilgi kullanıcı arayüzleri için kullanılır.

RSVP ile yer ayırma işleminin ardından, veri paketleri RTP yardımıyla uç birimler arasında akmaya başlar. Daha sonra RTCP devreye girer ve uç birimlerin sağlayabilecekleri ve alabilecekleri hizmet kalitesi seviyesinden haberdar olmalarını sağlar. Aslında bir sunucu da bağlı olduğu düğümlerden aldığı geri bildirim yoluyla servis kalitesi ayarlamaları yapabilir. Ama bu hareket biçimi RTCP tarafından tanımlanmamıştır.

4.2.3.1 RTCP paket türleri

RTCP ile taşınan raporlar sistemdeki uç birimler hakkında bilgi ve istatistikler verir. Bu raporların tipini değiştiren ise RTCP paket numarasıdır. RTCP paket türleri Tablo 4.9 'de gösterilmiştir [20].

Tablo 4.9: RTCP paket türleri

Numara	Paket Türü	Kullanım Amacı
200	Gönderici Raporu	Aktif paket göndericilerinin gönderme ve alma istatistiklerini gösterir.
201	Alıcı Raporu	Alan taraflar için sadece alma istatistiklerini gösterir.
202	Kaynak Açıklaması	Kaynak açıklama bölümlerini içerir.
203	Sonlandırma Mesajı	Katılımın sonunu belirtir.
204	Uygulama Özel	---

4.2.3.2 RTCP mesaj formatı

RTCP mesajları da RTP mesajları gibi 32 bitlik kelimelerdir. Tablo 4.10' da 200 numaralı gönderici raporu için geçerli olan RTCP mesaj formatı gösterilmiştir

Tablo 4.10: RTCP Mesaj Formatı

0	1	2	3	4-7	8	9-14	15	16-23	24-31	
V=2			P	RC	PT=SR=200			Uzunluk		Başlık
Gönderen Tarafın SSRC Numarası										
NTP Zaman Damgası (En yüksek önemli kelime)										Gönderici Enformasyonu
NTP Zaman Damgası (En düşük önemli kelime)										
RTP Zaman Damgası										
Gönderen Tarafın Paket Sayısı										
Gönderen Tarafın										
										1 numaralı Rapor Enformasyonu
Kayıp Oranı					Toplam Kayıp Paket Sayısı					
Alınan En Büyük Dizi Numarası										
Aravarış Jitteri										
Son Gönderici Raporunun Zamanı										
Gelen Son Gönderici Raporu İle Arasındaki Gecikme										
SSRC_2(İkinci Kaynağın SSRC Numarası)										
…										
Uygulamaya Özel Genişlemeler										

RC (Receiver block count): Kaç tane alıcı bloğunun mesaj içinde yer aldığını gösterir.

PT (Packet type): Gönderici rapor mesajı için 200'e ayarlıdır.

NTP Zaman damgası: Sistem saatine ilişkin ifadedir**.** Lokal saatten bağımsız olarak NTP (Network Time Protocol) kullanır. Senkronizasyon amacıyla kullanılır.

Gönderici paket sayısı ve oktet büyüklüğü: Alıcı tarafı kaç PDU'nun (paketin) ve kaç oktetlik enformasyonun yollandığı hakkında bilgilendirir.
Kayıp oranı: Kayıp paketlerin gelmesi düşünülen paketlere oranıdır.

Toplam kayıp paket sayısı: Algılamanın SSRC_n adlı kaynaktan çıkan toplam kayıp paket sayısı veren ifadedir.

Alınan en büyük dizi numarası: SSRC_n nolu kaynaktan alınan RTP paketlerin en büyük dizi numarasıdır.

Arageliş jitteri: Paketlerin gelişleri arasındaki zaman varyasyonlarının zaman damgası birimleri cinsinden ifadesidir [20,24].

4.3 Genel Bilgiler

Bu bölümde Visual C++ geliştirme ortamı, TCP/IP ve Soket mimarisi hakkında ve Microsoft temel sınıfları (MFC) hakkında bilgi verilecektir.

4.3.1 Visual C++ geliştirme ortamı

Visual C++, C++ ile program yazmak üzere donatılmış, görsel yönelimli bir yazılım geliştirme ortamıdır. C++ ya da C dilinde yazılması olası olan her türlü program Visual C++ ile yazılabilir. C++, C nin üst kümesidir. C dilinin geliştirilmesiyle oluşturulmuştur. C++, C nin kapsamakta olduklarına ek olarak nesneye dayalı programlamayı (OOP – Object Oriented Programming) da destekler [64].

4.3.2 TCP /IP ve soket mimarisi

TCP soket'lerinde bağlantı türleri soketin özelliğine göre değişir. TCP iletişiminde iki ana kavram vardır. Biri bağlantılı, diğeri bağlantısız iletişimdir. Bağlantılı iletişimde soket' ten mesaj bekleyen (listen komutu ile soketi dinleyen) programa bağlantı, connect komutu ile yapılır. Bağlantı sağlandıktan sonra mesaj gönderme ve alma işlemi oku (read) ve yaz (write) komutları ile gerçekleşir. Bağlantılı iletişimde kullanılan soket türü SOCK_STREAM olmalıdır. TCP/IP üzerinden iletişimde, hattın yoğunluğu ve kalitesine bağımlı olarak parçalanan TCP paketleri bu soket türü kullanıldığında alıcı soket altında sıralanır ve birleştirilerek TCP katmanına ulaştırılır. Bu durumda uygulama programı, bir read komutu ile tüm mesajı bir seferde okuyabilir. Kısa mesajlar SOCK_DGRAM soket üzerinden iletilebilir. Bu tür mesajların parçalanması söz konusu olmadığından yukarıdaki sorun yaşanmaz. Bu tür mesajlarda mesaj boyu 256 byte ya da daha kısa olmalıdır. Ve mesajlar hep aynı

uzunlukta olmalıdır. Bağlantısız iletişim ortamında kullanılır. Yani soket dinleyen program accept komutu kullanmaz, sokete bilgi gönderen de connect komutunu kullanmaz. Burada yalnız listen komutu vardır. Programlar karşılıklı "read" ve "write" komutu ile haberleşir. İletişim açısından çok güvenli bir soket haberleşmesi değildir. Daha güvenli ve bağlantılı iletişimde (connected) DATA_ DGRAM mesajları, SOCK_ SEQPACKET tipi soket' ler üzerinde iletilmelidir. Bir bilgisayar içindeki programlar arası iletişim SOCK_RAW tipi soket ile kurulabilir. Bu tür soket' lerde Data Gram tipi mesajlar iletilir.

Host cihazındaki bir TCP üst-katman kullanıcısı bir port numarası ile tanımlanır. Port değeri IP internet adresi ile birleşerek bir soket oluşturur. Bu değer internet boyunca tek olmalıdır. Bir soket çifti her bir uç-nokta bağlantısını tek olarak tanımlar. örneğin,

Gönderici Soket = Kaynak IP Adresi + Kaynak Port Numarası numarası
Alıcı soket = Hedef IP Adresi + Hedef Port Numarası

Bir bilgisayarda birden çok soket bulunabilir. Örneğin aynı anda hem telnet soketi hem de ftp soketi açık olabilmektedir. Soketleri birbirinden ayırmak ve istemci-sunucu ikilisini birbiri ile buluşturmak için her soket programın Port numarası vardır. Örneğin ftp'nin port numarası 21'dir. Bir ftp istemci, ftp sunucunun 21. portta çalıştığını bildiğinden direk onunla temasa geçer. Telnet 23.'u portta çalıştığından, telnet sunucu ile ftp sunucu karışmaz. 1-1024 arasındaki portlar standarttır ve yalnız root tarafından kullanılabilir. Normal kullanıcılar soket programları için 1023'ten büyük bir port numarası seçmeleri gerekir [65,66].

4.3.3 Microsoft temel sınıfları (MFC)

MFC microsoft windows ortamında işletim sisteminin Uygulama Programlama Arayüzü (Application Programming Interface – API)' ne ulaşıp işlem yapmamızı sağlayar. MFC sınıf sistemi içerisinde yüzden fazla sınıf ve binlerce üye fonksiyon vardır. Sınıf sisteminin en tepesinde Cobject denilen bir sınıf bulunur. Bu Object sınıfı uygulamalar için diğer sınıflara taban sınıfı işlevini yerine getirmektedir. MFC sınıflarının büyük çoğunluğu Cobject sınıfından türetilmiştir. MFC sınıf isimleri C harfi ile başlayarak isimlendirilir. MFC sınıfları kullanım alanlarına göre çeşitli

gruplara ayrılabilir. Bazı MFC sınıfları windows programlamayla ilgisi olmayan sınıflardır, örneğin Cstring ve Cfile gibi sınıfların doğrudan Windows programlamayla bir ilgisi yoktur. MFC STL sınıflarına gereksinim duymadan bir çalışmayı sağlamak amacıyla düşünülmüştür, bu yüzden kütüphane sistemi içerisinde çeşitli nesne tutan sınıflar (Container Class) vardır. Ayrıca MFC de az miktarda küçük template sınıflarda bulunmaktadır. MFC çoklu türetmenin tercih edilmediği bir sınıf sistemidir. Çünkü çoklu türetmenin hızı azaltacağı ve karmaşıklığı artıracağı düşünülmüştür. MFC de sanal fonksiyon mekanizması orta düzeyde kullanılmıştır. MFC sınıflarına ilişkin fonksiyon kodları hem statik hem de dinamik kütüphanelere yerleştirilmiştir. Programcı isteğine göre statik ya da dinamik link işlemi yapabilir [67].

4.4 Sistem Analizi ve Uygulama Aşamaları

4.4.1 Giriş

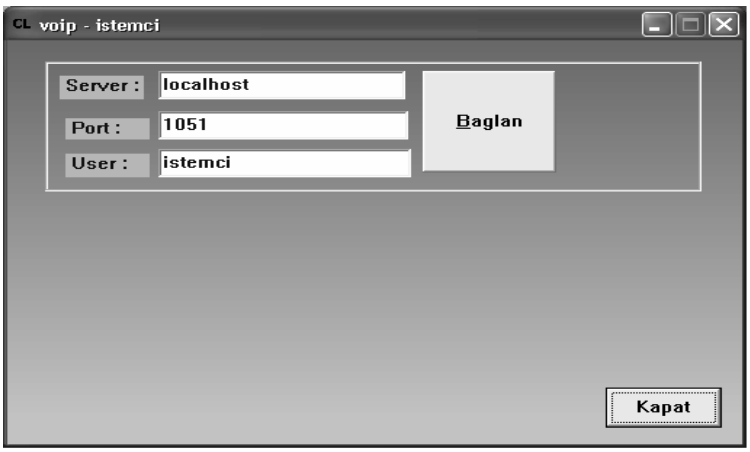
Bu uygulama internet üzerinden ses iletiminin gerçekleştirilmesi için tasarlanmıştır. Internet ortamında varolan kullanıcıların birbirlerine bağlanarak sesli görüşme işlemini yapabilmeleri düşünülmüştür. Günlük hayatta kullandığımız telefon görüşmesinin internet ortamına taşınması şeklinde düşünülebilecek olan bu uygulama ses verisinin bir kullanıcıdan diğer bir kullanıcıya iletimi esnasında oluşan olayları ve bu işlemi yerine getirecek olan işlevler hakkında çeşitli bilgiler içermektedir [14,68].

4.4.2 Arayüzler ve oluşturulan formatlar

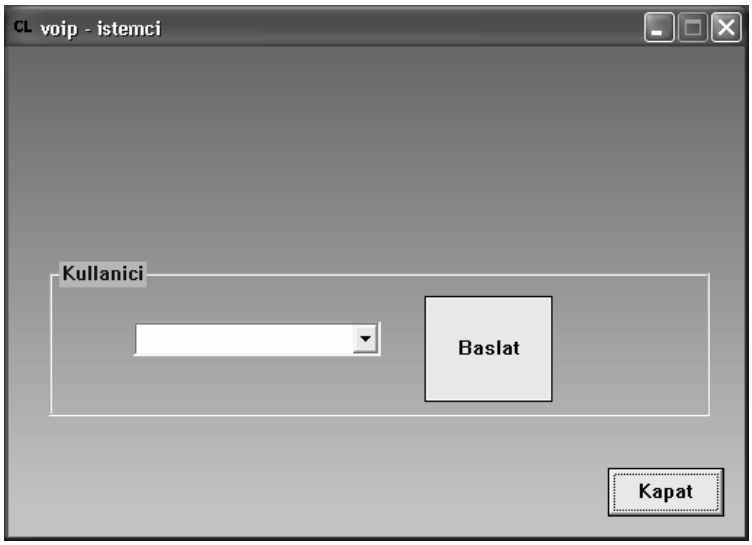
Sunucu (Server) –Kullanıcı (Client) şeklinde çalışacak uygulama için; iki tane arayüz ve bir mesaj formatı oluşturuldu. [69,70].

4.4.2.1 Kullanıcı (Client) arayüzü

Uygulamanın kullanıcı arayüzü sunucuya bağlanma, sunucu bağlantısını kesme, diğer kullanıcılar ile konuşma işlevlerini yerine getirir. Kullanıcı arayüzü iki ayrı formdan oluşur.



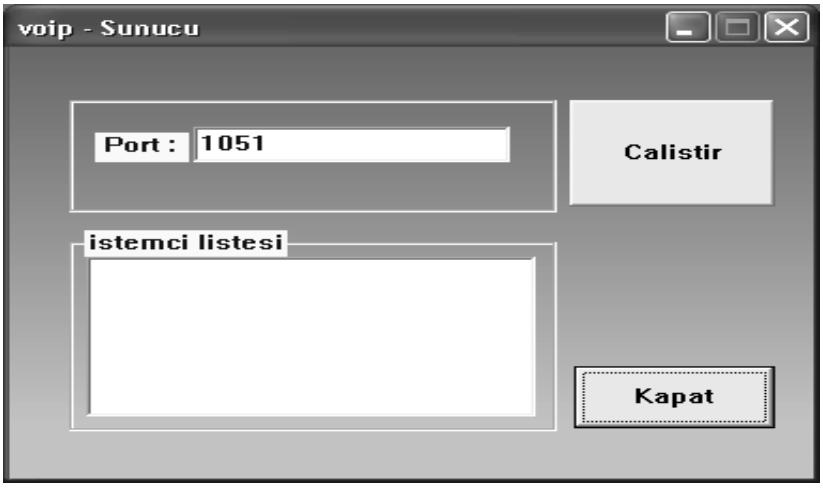
Şekil 5.1:Kullanıcı (client) arayüz formu (1)



Şekil 5.2: Kullanıcı (Client) arayüz formu (2)

4.4.2.2 Sunucu (Server) Arayüzü

Uygulamanın sunucu arayüzü tüm kullanıcı bilgilerini tutma, güncelleme ve diğer kullanıcılara sunma işlevlerini yerine getirir. Tüm kullanıcılar sunucu ile devamlı olarak iletişim içerisinde bulunmak zorundadırlar. Sunucu üzerinde sisteme bağlı kullanıcıların bilgileri tutulur .



Şekil 5.3 :Sunucu (Server) arayüz formu

4.4.2.3 Uygulamada kullanılan mesaj formatları

Uygulamada iki çeşit mesaj formatı belirlenmiştir. Birincisi kullanıcı bilgisi, ikincisi ise ses bilgisini göndermek içindir. Birinci format, (header:mesg) şeklinde belirlenmiş; aşağıdaki gibidir [71,72]

USER:msj(tunc,deniz),istemci listesi(Mesaj Yönü:server-client)
NEW:msj(tunc),istemci adı (Mesaj Yönü:client-server)

DOWN:msj(NULL),sunucu kapanma (Mesaj Yönü:server-client)

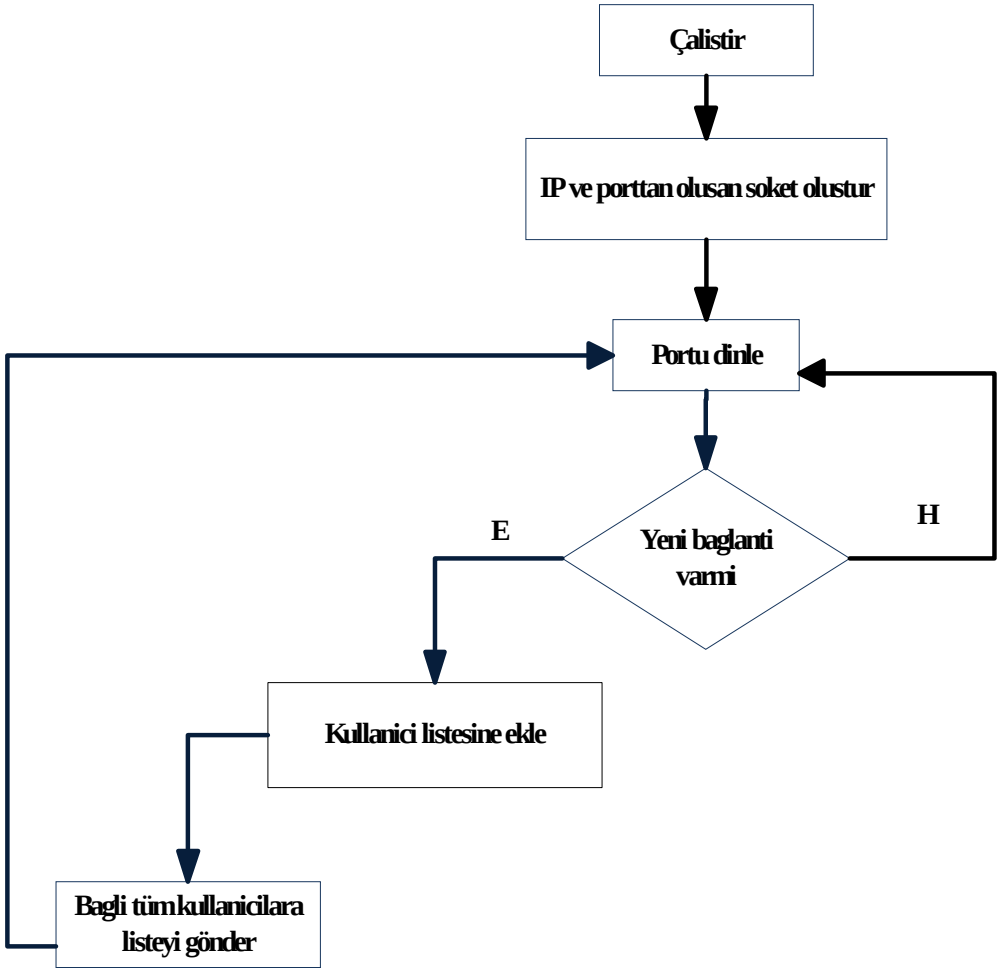
İkinci format, (header:length:mesg) şeklinde belirlenmiştir. Header kısmı ses gönderen kullanıcının ip ve port numarasını içermektedir. Length:sesbilgisinin

boyutunu, mesg ise ses bilgisini oluşturmaktadır. Ses bilgisinin örnekleme frekansı 22.5 kHz alındı.

4.4.3 Uygulamanın çalışma akışı

4.4.3.1 Sunucu (Server) uygulaması

Sunucunun çalışması için gerekli bazı adımlar vardır, bu adımlar aşağıda ayrıntılı açıklanmıştır.



Şekil 5.4: Sunucu (Server) akış diyagramı

Yukarıdaki algoritma akış diyagramına göre gerekli fonksiyonlar;

Çalıştır; çalıştır komutu bir kullanıcı arayüzü ile kullanıcıdan bu komutun girilmesi beklenir, bu komut girildikten sonraki işlemler,

void Serdlg::OnStart() fonksiyonu çağrılır, bu fonksiyonun kodları ise şu şekildedir. Çalıştır butonuna tıklandığında kullanıcıdan port bilgisi alınır, portun uygunluğu kontrol edilir, eğer port no uygunsa Sunucu Soketi oluşturulur.

```
void Serdlg::OnStart()

{

int port=GetDlgItemInt(IDC_EDIT1);

CButton *start=(CButton*)this->GetDlgItem(IDC_BUTTON1);

if(port<1024)

{

AfxMessageBox("1023 den buyuk bir prot nuamrasi giriniz");

return;

}

if(sersock.Create(port)) // server socket oluşturmak için gerekli komut

{

sersock.Listen(); //eğer server socketi oluşturulduysa, gelen verileri alabilmek için server portu dinlemeye başlar.

sersock.setparent(this);

}

if(start!=NULL)      //çalıştır butonunu disable ediyor.

start->EnableWindow(FALSE);

}
```

Yeni bağlantı oluşturma aşaması; Sunucu ilk aşamadan sonra çalışmaya başlar ve kullanıcının girmiş olduğu portu dinlemeye alır. Bu aşamada ise, herhangi bir kullanıcıdan bağlantı isteği geldiğinde, bu bağlantı isteği değerlendirilir ve uygun bir bağlantı ise, istemci bilgileri, IP ve kullanıcı adı, alınarak bağlantı oluşturulur. Burada bağlantı oluşturulurken kurulmuş olan soket yapısı üzerinden mysocket isimli

sınıf içerisinden OnAccept fonksiyonu çağırılır, bu fonksiyonun kodları aşağıda gösterilmektedir.

```
void mysocket::OnAccept(int errcode)
{
if(errcode==0)
((Serdlg*)dlg)->Accept();
CSocket::OnAccept(errcode);
}
```

bu fonksiyon Serdlg sınıfı içerisinden Accept() isimli fonksiyonu çağırır. Serdlg sınıfındaki Accept() fonksilyonun kodları aşağıdaki gibidir;

```
void Serdlg::Accept()
{
mysocket *client=new mysocket();
if(sersock.Accept(*client))
{
clientlist.AddTail(client);
client->setparent(this);
}
}
```

Bağlantı kuruldu ve bu bağlantı kurulan kullanıcı'ın bilgilerini bir liste yapısında tutabilmek için varolan kullanıcı listesine yeni bağlantı kurulmuş olan kullanıcının bilgileri eklenir, bu işlemleri yapabilmek için mysoket sınıfından OnReceive() fonksiyonnu çağrılır bu fonksiyonun kodları aşağıdadır.

```
void mysocket::OnReceive(int errcode)
{
if(errcode==0)
((Serdlg*)dlg)->Receive(this);
```

```
CSocket::OnAccept(errcode);
}
```

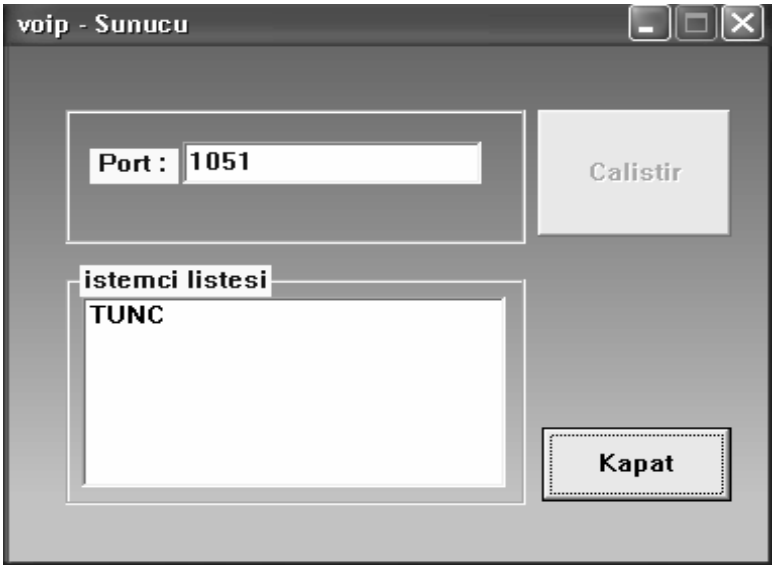
bu fonksiyon Serdlg sınıfı içerisinden Receive() fonksiyonunu çağırır. Bu fonksiyon gelen mesajın içeriğini çözer ve kullanıcı bilgilerini alarak kullanıcı listesini günceller. Bu fonksiyonun kodları aşağıdaki gibidir.

```
void Serdlg::Receive(mysocket *client)
{
char buff[2021];
char str[100];
int size=2020,count,index,i;
CString mesg,header,test,bmesg;
//msj alinir ve buff da saklanir
count=client->Receive(buff,size);
if(count==SOCKET_ERROR)
{
return;
}
if(count>2020 || count<0)
{
return;
buff[2020]=NULL;
}
else
{
buff[count]=NULL;
}
mesg=buff;
```

```
index=mesg.Find(':');
if(index==-1)
{
return;
}
header=mesg.Left(index);
mesg=mesg.Right(mesg.GetLength()-index-1);
if(header.CompareNoCase("NEW")==0)  //yeni istemcinin kullanici adi
{
test=mesg;
test.MakeUpper();
displist->AddString(test);
userlist+=test+";";
client->name=test;
test="USER:"+userlist;
SendToAll(client->name,(char*)(LPCTSTR)test,test.GetLength(),0);
return;
}
if(broadcast==1 && buser.CompareNoCase(client->name)!=0)
return;
if( broadcast==1 && header.CompareNoCase("OVER")==0)
{
broadcast=0;
buser.Empty();
bmesg="RESUME:";
SendToAll(client->name,(char*)(LPCTSTR)bmesg,bmesg.GetLength(),1);
return;
```

```
}
for(i=0;i<client->name.GetLength();i++)
buff[i]=client->name[i];
buff[i]=':';
if(header.CompareNoCase("ALL")==0)
{
if(broadcast==0)
{
buser=client->name;
bmesg="WAIT:"+buser;
broadcast=1;
SendToAll(client->name,(char*)(LPCTSTR)bmesg,bmesg.GetLength(),1);
Sleep(200);
SendToAll(client->name,buff,count,1);
}
else
SendToClient(header,buff,count);
reccount++;
}
```

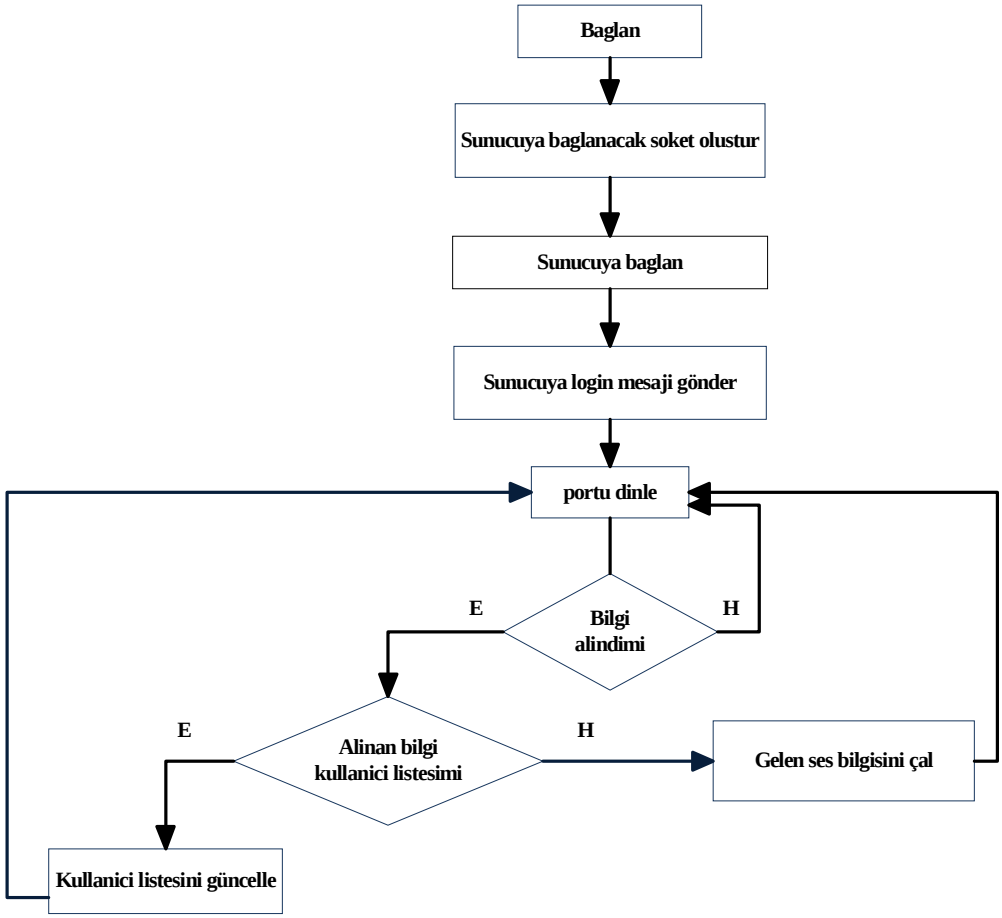
Bir önceki aşamada yapılan değişiklikler yani kullanıcı listesinin güncel hali tüm kullanıcılara gönderilmesi işlemi yapılmalıdır. Bu işlem ise var olan liste, sendtoAll() fonksiyonu ile yapılmaktadır.



Şekil 5.5: Sunucu (Server) bağlantı şeması

4.4.3.2 Kullanıcı (Client) uygulaması

Kullanıcı uygulaması, ses verisinin karşılıklı aktarılacağı konuşmacılar olarak düşünülebilir. Bu yapı sayesinde kullanıcılar sunucudan aldıkları listeler vasıtası ile, seçmiş oldukları başka bir kullanıcı ile online bir şekilde konuşabilmektedirler .Bu uygulamanın gerekli kodları ve açıklmaları aşağıda belirtilmiştir. Şekil 5.6'da kullanıcı (Client) akış diyagramı (1) gösterilmektedir [73].

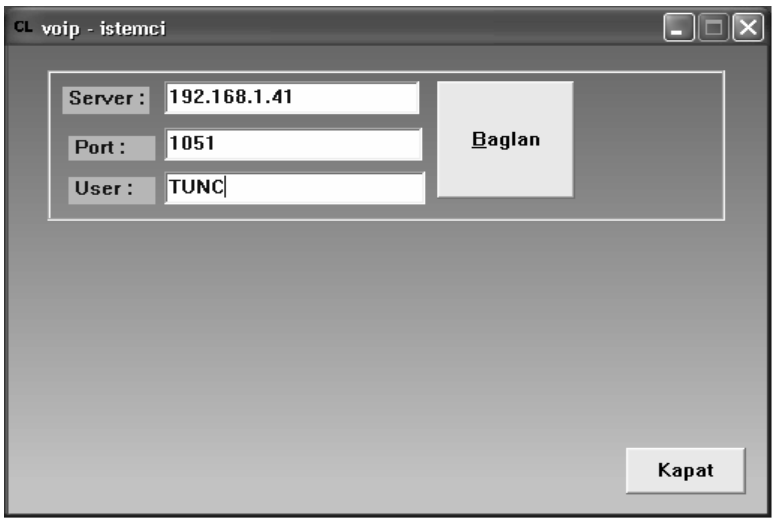


Şekil 5.6: Kullanıcı (Client) akış diyagramı (1)

.

Kullanıcı uygulamasının başlaması için bir soket oluşturmaya gerek vardır. Bu soket sayesinde kullanıcı sunucudan güncel kullanıcı listesini alabilmektedir hem de diğer kullanıcılara ses mesajı gönderebilmektedir. Bu soket yapısının oluşturulması aşaması şu şekildedir.

Socket.Create(); komutu ile yeni bir soket oluşturulur. Bu soket uygulama içerisinde sunucuya ve kullanıcılara bağlanabilmek için kullanılacaktır.



Şekil 5.7:Kullanıcı (Client) bağlantı arayüzü (1)

Şekil 5.7'deki arayüz vasıtası ile kullanıcı ismi, kullanıcının bağlanmak istediği sunucu IP' si ve sunucu port numarası alınmaktadır. Bu alınan bilgiler yardımı ile kullanıcı sunucuya bağlanmak isteyecektir. Eğer girilen IP ve Port bilgileri doğru ise kullanıcı sunucu bağlantısı sağlanacaktır. Bu işlemi yapan ilgili fonksiyonların kod bölümü aşağıda verilmiştir;

```
void Display::Onconnect()
{
int port;
CString sername,username,mesg;
GetDlgItemText(IDC_EDIT1,sername);
GetDlgItemText(IDC_EDIT3,username);
port=GetDlgItemInt(IDC_EDIT2);
if(username=="" || sername=="" )
{
AfxMessageBox("Lutfen bilgileri dogru bicimde giriniz");
return;
}
//clienti servera baglfcrea
```

```
username.MakeUpper();
sockclt.name=username;
SetDlgItemText(IDC_BUTTON1,"Baglaniyor");
if(sockclt.Connect(sername,port))
{
 // setDlg()
}
else
{
SetDlgItemText(IDC_BUTTON1,"&Baglan");
MessageBox("Sunucuya baglanamiyor\nPort ve sunucu bilgilerini kontrol edin");
return;
}
```

sockclt.Connect(sername,port) komutu yardımı ile bağlanma isteğinde bulunur, bu bağlantı isteği sonucunda bağlantı kabul edilmişse true bağlantı kabul edilmemişse false değeri elde edilir. Bağlantı sağlanamazsa kullanıcıya bilgilerini kontrol etmesi için uyarı verilir.

Bu aşamada kullanıcı bilgilerini sağlanan soket bağlantısı sayesinde sunucuya iletir. Kullanıcı bilgilerini bir mesaj formatına dönüştürmesi gerekmektedir. Bu mesaj formatı 5.2.2.3 de belirtilmiştir. Kullanıcı bilgilerini sunucuya gönderen işlemi yapan kod aşağıdaki gibidir.

```
//sunucuya login msj gonder
mesg="NEW:"+username;
sockclt.Send(mesg,mesg.GetLength());
```

bu kod bloğundan sonra kullanıcı sunucuya dahil olmuş olur. Sunucu-kullanıcı arası mesaj alış-veriş işlemi başlatılabilir.

Bu aşamada kullanıcı sunucya bağlı olan kullanıcı listesini elde edecektir ve bu liste vasıtası ile, herhangi bir kullanıcıyı seçip o kullanıcıya ses verisini gönderebilir. Kullanıcının sunucudan kullanıcı listesini elde etme aşaması;

```
buff[rcount]=NULL;
mesg=buff;
index=mesg.Find(':');
//hatali msj formati
if(index==-1)
return;
header=mesg.Left(index); // mesajın içeriği çözülüyor
//sunucu kullanici listesi mi gondermis
if(header=="USER")
{
mesg=mesg.Right(mesg.GetLength()-index-1);
updateList(mesg); // kullanıcı listesini güncelleme
return;
}
```

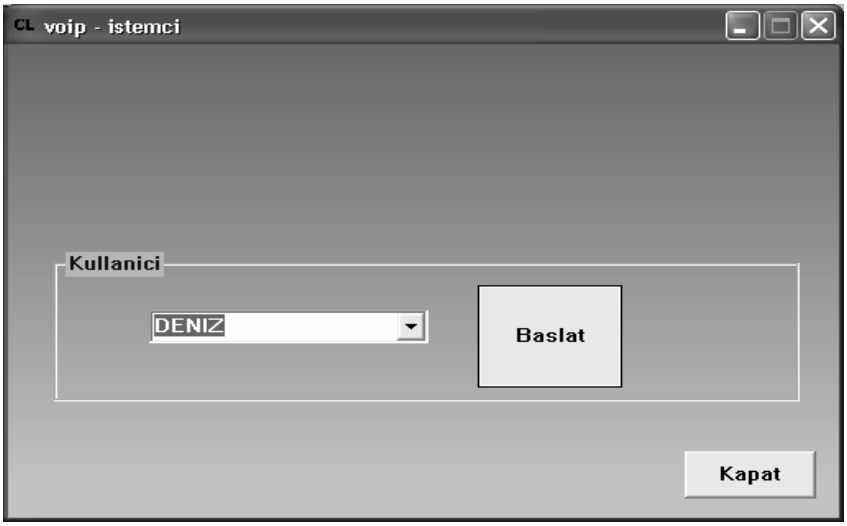
Sunucudandan gelen mesaj anlaşılması durumu oluşmakta bu anlaşılması kısmı için gelen mesajın içeriği çözülür, elde edilen bu içerik kontrol edilir. Eğer kullanıcı listesi bu mesajda var ise, gelen mesaj sunucudan gelmiştir ve kullanıcı da bulunan kullanıcı listesi güncellenir.

Kullanıcı görüşme yapmak istediği diğer kullanıcıyı seçer ve o kullanıcıyla sesli görüşme yapabilir. Ses mesajının alınması ile ilgili kod kısmı aşağıda satır satır açıklanmıştır;

Gelen mesajın ses olduğu mesaj içeriği çözülerek anlaşılır; mesajda kullanıcı listesi yoksa mesaj ses mesajı olduğunun farkına varılır. Kullanıcıya mesaj geldiği uyarısı verilir ve ses mesajı hoparlörden çalınır [74].

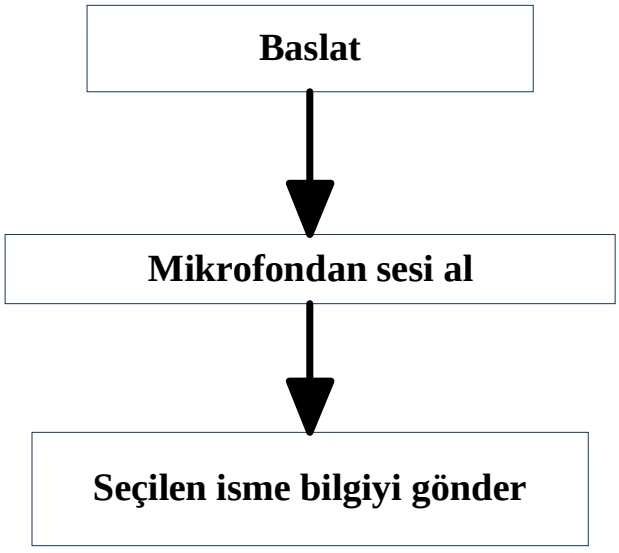
```
length=0;

sscanf(&buff[15],"%d",&length);

if(length<1 || length>PLAYBUFFER)

 return    ;

if(play->Playing==FALSE)

play->PostThreadMessage(WM_PLAYSOUND_STARTPLAYING,0,0);

LPWAVEHDR lpHdr=playhead[curhead];

curhead=(curhead+1)%MAXBUFFER;

//           playmesg=new char[2020];

for(i=20,j=0;j<length;i++,j++)

lpHdr->lpData[j]=buff[i];

 lpHdr->dwBufferLength=length;

lpHdr->dwFlags=0;

play->PostThreadMessage(WM_PLAYSOUND_PLAYBLOCK,0,(LPARAM)lpHdr);

if(isSave==TRUE && writeuser.CompareNoCase(header)==0) {

write->PostThreadMessage(WM_WRITESOUND_WRITEDATA,0,(LPARAM)lpHdr);

}

disp="  "+header+" konusuyor...";

SetDlgItemText(6051,(char*)(LPCTSTR)disp);
```

Seçilen kullanıcıya ses verisinin gönderilmesi**;**



Şekil 5.8: Kullanıcı (Client) bağlantı arayüzü (2)

Şekil 5.8'de bir kullanıcı seçilmiştir ve bu kullanıcıya ses mesajı gönderilme işlemi yapılmaktadır, bu işlem için programı kullanan kullanıcı görüşmek istediği kişiyi seçer ve ona Şekil 5.9'daki akış diyagramında var olan adımları izleyerek ses mesajını gönderir.



Şekil 5.9 : Kullanıcı (Client) akış diyagramı 2

Başlat butonuna basıldığında kullacı seçilip seçilmediği kontrolü yapılır, bu kontrolden sonra eğer kullanıcı seçilmiş ise, mikrofondan ses verisi alınır ve bu ses verisi paket haline getirilerek seçilmiş olan kullanıcıya yollanır. Bu işlemlerin kodları aşağıda verilmiştir.

```
void Display::sendMessage(char *mesg,int length)
{
char buflen[15];
char str[PLAYBUFFER+50];
int i,j;
//hicbir kullanici secilmemisse
if(selectflag==0 )
return;
if(mesg==NULL)
{
// buffer bos ses alinmamis
return;
}
if(length<1 || length>PLAYBUFFER)
{
// cok uzun buffer
return;
}
mesg=mesg-20;
//ilk 15 bits hedef adresi iceriyor
//sonraki   5 bits msj uzunlugunu verir
//kalan kisim ses bilgisi
```

```
if(curuser.IsEmpty())
{
return;
}
for(i=0;i < curuser.GetLength();i++)
mesg[i]=curuser[i];
mesg[i]=':';
sprintf(buflen,"%d",length);
for(j=0,i=15;j<5;j++,i++)
mesg[i]=buflen[j];
sockclt.Send(mesg,length+20);
}
```

Ses verisinin okunması ve çalınması;

Kullanıcıların gelen ses verisini alması aşamasında, ses veya liste kontrolünden sonra, eğer gelen veri ses ise aşağıdaki işlemler yapılmaktadır. Bu işlemleri kısaca açıklayacak olursak;

Öncelikle, Display::Receive fonksiyonu içerisinde ses verisi alınıyor bu veri rcount=sockclt.Receive(buff,size); ile tampon (buffer) içerisine alınıyor, tampon içerisindeki verinin büyüklüğü kontrol ediliyor eğer tampondan büyük değilse işleme alınıyor eğer tampondan büyük bir veri gelmişse işlem sonlandırılıyor. Tampona veri alındıktan sonra hoparlörden çalınması aşamasına geçiliyor, bu aşamada ses verisi bir iş parçacığı (thread) vasıtası ile ses cihazına (sound device) gönderiliyor ve bundan sonraki aşamada ses verisi çalınıyor. Burada ses verisinin donanıma gönderimi işlemi yapılıyor ve çalma işlemi işletim sistemi ve ses donanımı vasıtası ile değerlendirilip, çalma işlemi gerçekleştiriliyor. Bu işlemleri yapan kod bloğu ve fonksiyonlar aşağıdaki gibidir. PlaySound1.cpp isimli sınıf içerisinde, bulunan çeşitli fonksiyonlar var bu fonksiyonlar sesin çalınması aşamasında gerekli olan, donanım kontrolü, donanım hata durumu, donanım bilgilerinin alınması ve ses verisinin çalmaya başlaması aşamasındaki gerekli olan komutları ve kod bloğunu

içermektedir. Bu sınıf yapısı MFC programlamanın içerisinde hazır bir durumda olup, uygulamaya dahil edilmiştir. Kodlar aşağıdaki gibidir;

Ses Çalma fonksiyonu;

```
LRESULT PlaySound1::OnStartPlaying(WPARAM wParam, LPARAM lParam)
{
MMRESULT mmReturn = 0;
if(Playing==TRUE)
return FALSE;
log.WriteString("\n Starting playing");
// open wavein device
mmReturn = ::waveOutOpen( &m_hPlay, WAVE_MAPPER,
&m_WaveFormatEx, ::GetCurrentThreadId(), 0, CALLBACK_THREAD);
if(mmReturn )
displayError(mmReturn,"PlayStart");
else
{
Playing = TRUE;
DWORD volume=0xffffffff;
char str[100];
if(!waveOutSetVolume(m_hPlay,volume))
{
volume=0;
if(!waveOutGetVolume(m_hPlay,&volume))
{
wsprintf(str,"\n Volume is  %lx",volume);
log.WriteString(str);
}
```

```
}

}

return TRUE;

}
```

Bu fonksiyona ek olarak değinilmesi gereken diğer bir nokta ise, ses verisinin nasıl saklandığı normalde ses dosyaları fiziksel olarak sabit diskte saklanması gerekli ve bu fiziksel dosyanın içeriği ses çalma işlemini gerçekleyen donanıma gönderilmeli. Bu aşamada biz ses verisini dosya halinde sabit diskte tutmayıp, gelen verileri sürekli kullanıcıya dinletmeye çalışıyoruz. Gelen bilgi doğrudan LPWAVEHDR nesnesine kopyalanıyor, bu işlem Display::Receive fonksiyonu içerisinde, aşağıdaki kod bloğu yapmaktadır;

```
for(i=20,j=0;j<length;i++,j++)
lpHdr->lpData[j]=buff[i];
```

oluşturulan ve ses verisi kopyalanan bu nesne, PlaySound1.cpp isimli sınıf içerindeki OnStartPlaying fonksiyonu ile ses donanımında çalınıyor.

5. SONUÇLAR VE ÖNERİLER

Bu çalışmada İnternet üzerinden sesin iletilmesi konusu incelenmiş ve Microsoft Visual C++ geliştirme ortamı ve nesne tabanlı programlama teknikleri kullanılarak bir VoIP uygulamasının nasıl gerçekleştirildiği anlatılmıştır.

VoIP 'nin ilk aşamasında, analog ses işaretinin örneklenmesi, belirli değerlere ötelenmesi (kuantalanması) ve bu ötelenmiş değerlere belirli kodlar atanması işlemi yapılır. PCM kodlaması da denen bu teknik kullanıldıktan sonra band genişliğinden kazanmak amacıyla sayısala çevrilmiş 64kbit/sn 'lik ses işaretleri G.723 (6.4 kbit/sn) yada G.729 (8kbit/sn) kodlayıcı yardımıyla kodlanır. Daha sonra işaretler hatta verilemden önce kapsüllenme işlemine tabi tutulurlar. Başlık bitlerinin eklenmesiyle 8kbit/sn 'lik işaretin istediği kanal kapasitesi 27 kbit/sn olur. Ses işaretine sessizlik bastırımı uygulanır ve 27kbit/sn 'lik işaret 11 kbit/sn değere kadar düşürülmüş olur. Böylece kodlanmış ve paketlenmiş olan sayısal işaret iletim hattına verilir. Alıcı tarafta tersi işlem uygulanarak orijinal işaret elde edilir.

Günümüzde VoIP uygulamalarının ana problemi ,QoS (Servis Kalitesi) garantisi verememeleridir. RSVP (Kaynak Ayırma Protokolü) gibi QoS garantisi veren protokoller geniş bir şekilde kullanıldığı zaman insanlar istediği kalitede iletişim yapabileceği için ,VoIP uygulamaları daha popüler olacaktır. Yerel alan ağlarında (LAN) , genelde bantgenişliği problemi olmadığından VoIP uygulamaları sorunsuz bir şekilde yapılmaktadır. Ancak geniş alan ağlarında (WAN) ve internette VoIP uygulamalarının sorunsuz bir şekilde gerçekleştirilebilmesi için QoS garantisi sağlayan protokollerin kullanımı gereklidir. VoIP'in önünde aşması gereken bazı engeller olsa da; PSTN'e karşı sağladığı ekonomik avantajlar (düşük işletme ve kurulum giderleri), bu sistemin yakın bir gelecekte standart devre bağlaşmalı telefon şebekelerinin yerini alacağını göstermektedir.

Sunucu (Server)-Kullanıcı (Client) şeklinde çalışan VoIP uygulaması gelişime açıktır. Uygulama bir sunucu ve bu sunucuya bağlanan kullanıcılardan oluşmaktadır. Uygulamanın ilk aşamasında ses görüşmesi yapmak isteyen kullanıcılar sunucuya bağlanır. Sunucu kendisine bağlı tüm kullanıcılara online olan durumda olan tüm kullanıcı bilgilerinin tutulduğu listeyi gönderir.Kullanıcı uygulamasında listeden her hangi bir kullanıcı seçilir. Mikrofondan ses alınır. (Örnekleme frekansı 8 kHz ve bit derinliği 8 bit seçilir). Mikrofondan alınan ses sinyali bu özelliklerde sayısal değerlere çevrilerek 2000 bitlik bir tampona (buffer) yazılıyor. Daha sonra uygun fonksiyonlar yardımıyla TCP protokolü kullanılarak ses iletimi yapılıyor. Uygulamaya yeni yazılımlar eklenerek (örneğin multimedia konferans yazılımı, filtreleme yazılımı gibi) daha üst kapsamlı bir uygulama haline getirebilir.

KAYNAKLAR

[1] Jiang, W., Lennox, J., Narayanan, S., Schulzrinne, H., Singh, K., and Wu, X., "Integrating internet telephony services'', ***IEEE Internet Computing,pages,*** 6, 64-72, (2002).

[2] Riordan, J., " Telephone traffic time avarages" , ***Bell Syst.Tech.J***., vol.39, pp. 8, 1129-1144, (1951).

[3] Bük , O., "İnternet Protokolü Üzerinden Ses İletimi ve Askeri Amaçlı Kullanımı '', Yüksek Lisans Tezi, ***İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, (2004).

[4] www.enderunix.org/docs/tcpip/tcp/tcp.html ***(Ziyaret Tarihi: 22 Mayıs 2007).***

[5] Sundstrom, K., Rueda, A., and McLeod, R.D., "Internet Telephony compression algorithms'', ***WESCANEX'97,winnepeg, MB,pp***.13-18, (1998).

[6] Fincan, E., " İnternet Protokolü Üzerinden Ses Haberleşmesi", Yüksek Lisans Tezi, ***İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü,*** İstanbul, 45-48, (2002).

[7] Karim,A. , " H.323 and Associated Protocols'', ***Published on Prof.Jain's Web site*** (2005).

[8] Handley, M.,Handley, V., Jacobson, V., " H.323 and SIP protocols'', ***RFC :2327*** April , (1998).

[9] http://www.cisco.com/web/TR/learn_events/index.html ***(Ziyaret Tarihi: 22 Mayıs 2007).***

[10] Protocol Directory, http://www.protocols.com/pbook/VoIP.htm#H323 **(*Ziyaret Tarihi: 22 Mayıs 2007).***

[11] Sriram, K., and Whitt, W., "Characterizing superposition arrival processes in packet multiplexers for voice and data'', ***IEEE J.Select Areas Commun.,vol.SAC-4,pp***. 833-846,Sept., (1986).

[12] " Real Time Control Protocol attribute in Session Description Protocol " ***IETF Recommendation RFC 3605***, (2003)**.**

[13] Schulzrinne, H., ” RTP :A Transport Protocol for Real-Time Applications ’’, ***IETF RFC*** 1889 Jan., (1996).

[14] http://www.csharpnedir.com/makale.asp?cat=cpp ***(Ziyaret Tarihi: 22 Mayıs 2007).***

[15] ”One way transmission time’’, ***ITU-T Recommendation*** G.114, ,Feb., (1999).

[16] Abazi, A., ” İnternet Protokolü Üzerinden Ses İletimi ” , Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul, 34-37 , (2000).

[17] https://www.gizmocall.com/about ***(Ziyaret Tarihi: 23 Mayıs 2007).***

[18] https://www.asterisk.org/about ***(Ziyaret Tarihi: 23 Mayıs 2007).***

[19] Thompson, K., Miller, G., and Wilder, J., ” Wide area internet tarffic pattems and characteristics’’ ***, IEEE Network, vol.11,no.6,pp***. 10-23, Now.-Dec., (1997).

[20] Deva , M., ” İnternet Protokolü Üzerinden Ses Aktarımı ’’ , Yüksek Lisans Tezi, ***İstanbul Teknik Üniversitesi*** , İstanbul, (2004).

[21] Zhang, G., Reuther, B., ,and Müller, P., ” User oriented ip accounting in multi-user systems’’, ***In Proceedings of the 8 th IFIP/IEEE International Symposium on Integrated Network Management*** (Colorade Springs,USA),3 , (2003).

[22] Bettag,U., ” Informatik Spektrum, pages ” , ***Web-services,*** 302-304 , (2001).

[23] Leland, W. , Taqqu, M., Willinger, W., and Wilson, D., ”On theself-similar nature of ethernet taraffic’’***, IEEE/ACM Trans .Netw.,vol.2,no.1,pp*** 1-15 ,Feb, (1994).

[24] Hoshi, T.,Tanigawa, K., and Sukada, K., ” Proposal of a method of voice stream multiplexing for IP Telephony systems ’’ , ***Proc.IWS ’99,*** Feb., pp.182-188., (1999).

[25] Black, U., “Data Networks”, ***Concepts, Theory and Practice. Englewood Cliffs, Prentice Hall***, New Jersey, (1989).

[26] Durusoy, G., ” Voıp Sistemleri Ders Notları, Kısım 1: Temel Prensipler ”, ***İ.T.Ü. Elektrik - Elektronik Fakülte*****si**, İstanbul, (2003).

[27] Fundamentals of Voice over IP, http://www.fcc.gov/voip/voip ***(Ziyaret Tarihi : 22 Mayıs 2007).***

[28] http://www.pusula.net.tr/voipurun.htm ***(Ziyaret Tarihi: 22 Mayıs 2007).***

[29] Hassan, M., Nayandoro, A., and Atiquzzaman, M., ” Internet telephony: Services ,technical challenges,and products’’***, IEEE Commun.Mag.,pp.*** 96-103,Apr. (2000).

[30] Arango, M., Dugan, A., Elliott, I., Huitema, C., ,and.Pickett, S., ” Media Gateway Control Protocol (VoIP and MGCP) Version 1.0 ”, ***IETF RFC*** 2705, (1999).

[31] http://www1.alcatel-lucent.com/solutions/ ***(Ziyaret Tarihi: 22 Mayıs 2007).***

[32]http://www.newport-networks.com/pages/voip-bnadwidth-calculator.html ***(Ziyaret Tarihi:22 Mayıs 2007).***

[33] Markopoulou, A.P., Tobagi, F.A., and Karam, M.J., ”Assesing the Quality of Voice Communications over Internet Backbones***’’ ,IEEE/ACM Trans.Networking ,vol.11,no.5,pp***.747-760, (2003).

[34] Vaudreuil, G., Parsons, G., ” Toll Quality Voice – 32 kbit/s ADPCM MIME Sub-type Registration ”, ***IETF Recommendation RFC 2422***, (1998).

[35] Hood, C., and Ji, C. ” Proactive network fault detection ’’, ***in Proc.IEEE INFOCOM ,vol . 3,Kobe Japan,Apr.,pp.*** 1147-1155., (2005).

[36] ”Security Considerations for Voice IP Systems’’, ***Information Technology Laboratory National Institute of Standards and Technology(NIST),*** (Draft),October) , (2003).

[37] Deng, S., ” Traffic characteristics of packet voice’’, in ***Proc.ICC’ 95, vol.3 Seattle,pp***.1369-1374, (2005).

[38] Handley, H., ” SIP ,Session İnitiation Protocol ’’, ***IETF RFC*** , Mar., (1999).

[39] http://www.datagrup.com.tr/teknoloji05.asp?InMenuId=4#a ht ***(Ziyaret Tarihi: 22 Mayıs 2007).***

[40]Agrawell, H., ” SIP-H.323 Interworking Requirements’’, ***IETF draft ,work-in progress***, July , (2000).

[41] Sinnreich, H., and Johnston,A.B., ” Internet Communications Using SIP” John Wiley & Sons, (2001).

[42] Wang, S., Xuan, D., Bettati, R., and Zhao, W., ” Differentiated Services with Statistical Real –Time Guaranteess in Static-Priority Scheduling Networks ” , ***Proc.IEEE Real –Time Systems Symp***., Dec. , (2001).

[43] Schulzrinne, H., and Rosenberg, J., ” Signaling for internet telephony”, ***In Proceedings of 6 th IEEE International Conference on Network Protocols ICNP*** (Austin,Texas), (1998).

[44] Waldron, G., J., Welch, R., ” Voice over IP: The Future Communications”, ***Internet Policy Initiative***, Washington, (2002).

[45] Abazi, A., ” İnternet Protokolü Üzerinden Ses İletimi ” , Yüksek Lisans Tezi, ***İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü,*** İstanbul, 34-37 , (2000).

[46] ” Call Signalling Protocols and Media Stream Packetization for Packet –Based Multimedia Communication Systems ’’, ***ITU-T Rec.H.225.0 ,*** (1998).

[47] Schulzrinne, H., and Rosenberg, J., ” A comparison of sip and h.323 for internet telephony .In Proceedings of Network and Operating System Support for Digital Audio and Video ’’, ***NOSSDAV*** (Cambridge,England), (1998).

[48] ” Registration, Admission and Status Signaling (Recommendation H.225 /RAS / ’’, ,***Int ‘l Telecomm.Union (ITU)***, (1998).

[49]Huitema, C., ” An Architecture for Internet Telephony Service for Residential Customers ’’, ***IEEE Network***, May/June, (1999).

[50] ” IP Telephony Design Guide”, ***Alcatel Telecommunications Review***, California, (2003).

[51] http://www.openh323.org/standards.html ***(Ziyaret Tarihi: 28 Mayıs 2007)***

[52] ” Packed-Based Multimedia Communications Systems,” ***ITU-T Rec.H.323*** , (1999).

[53] Kumar, V., Korpi, M., and Sengodan, S., ” IP Telephony with H.323: Architectures for Unified Networks and Integrated Services ”, ***John Wiley&Sons*** , (2001).

[54] Liu, H., Mouchtaris, P., ” Voice over IP Signalling: H.323 and Beyond ”, ***IEEE Communications Magazine***, 142. , (2001).

[55]http://www.iec.org/online/tutorials/vfoip/topic01.html ***(Ziyaret Tarihi:22 Mayıs 2007).***

[56]Thomsen, G., and Jani, Y., ” Internet telephony:Going like crazy ”, ***IEEE Spectrum,pp.***52-58, May, (2000).

[57] Minoli D., Minoli E., ” Delivering Voice over IP Networks ’’, ***Wiley Computer Publishing*** , (1998).

[58] http://www.skype.com/intl/tr/download/features/ **(*Ziyaret Tarihi: 22 Mayıs 2007)***

[59] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A ., Peterson, J., Sparks, R., Handley, M., and E.Schooler,"SIP:Sesion initiation protocol '', ***IETF RFC 3261***, June, (2002).

[60] http://computing-dictionary.thefreedictionary.com/VoIP ***(Ziyaret Tarihi: 22 Mayıs 2007)***

[61] http://www.internetbilgisi.com/7_IP_Telefon.htm#1 ***(Ziyaret Tarihi: 28 Mayıs 2007).***

[62] Davidsone J., " Deploying Cisco Voice over IP Solutions '' ***,CiscoPress***, (2002).

[63]http://www.iptelephony.org/cgibin/DataMaker/DataMaker.pl?Selection=PublicViewData&dbIdNum=1&dfIdNum=1 ***(Ziyaret Tarihi: 22 Mayıs 2007).***

[64]Benes,V. " The covariance function of a simple trunk group ,with applications to traffic measurement ***'', Bell System.Tech.J.pp***.117-148, (1961).

[65] Halman, G., Halaç, A., MCSE (Microsoft Certified System Engineer), 2. Basım, ***Sistem Yayıncılık***, İstanbul, (2002).

[66]http://www.byte.com.tr/makaleler/default.asp/Gorev/MakaleGoster/Makale/74/Sayfa/270 ***(Ziyaret Tarihi: 28 Mayıs 2007).***

[67] http://www.vbfrance.com/codes/ASYNC-TCP-SOCKET_40792.aspx ***(Ziyaret Tarihi: 22 Mayıs 2007).***

[68] Güngören, B**.,** " C++ ile Nesne Tabanlı Programlama'', 1.Basım, ***Sistem Yayıncılık***, Ankara , (2001).

[69] http://msdn2.microsoft.com/en-us/default.aspx ***(Ziyaret Tarihi :22 Mayıs 2007).***

[70] Çölkesen ,R. , " C Proglama Dili ", ***Papatya Yayıncılık***, (2001).

[71] Pappas, C.H., Murray, W.H. " C/C ++ Programcının Rehberi ", ***Sistem Yayıncılık*** , (2002)

[72] http://www.somurgen.com/pagenum1.html ***(Ziyaret Tarihi: 22 Mayıs 2007).***

[73] özkan, Y., " ***Nesneye Yönelik Programlama:C++ ile Programlama'', Alfa Yayıncılık,*** (2003).

[74]http://http://www.programlama.com/sys/c2html/viewCategory.php?DocCategory=3&DocType=1 ***(Ziyaret Tarihi : 22 Mayıs 2007).***

KİŞİSEL YAYINLAR VE ESERLER

Erkan, N., Becerikli, Y., ve Aksakallı, C., "İnternet Protokolü Üzerinden Ses İletimi Ve Bir Yazılım Uygulaması Gerçeklenmesi", 12. Elektrik Elektronik Bilgisayar Biyomedikal Mühendisliği Ulusal Kongresi ve Fuarı, 14-18 Kasım, Eskişehir, (2007).

EK-1

VoIP uygulaması kapsamında sunucu (server) uygulamasında bulunan kodlar aşağıdaki gibidir.

```
// mysocket.cpp: mysocket sınıfının uygulaması
#include<afxwin.h>
#include<afxsock.h>
#include "mysocket.h"
#include "Serdlg.h"
mysocket::mysocket()
{
name="";
}
mysocket::~mysocket()
{
}
void mysocket::setparent(CDialog *dialog)
{
dlg=dialog;
}
// Bir client sunucuya bağlandığında bu foksiyon çağrılır.
void mysocket::OnAccept(int errcode)
{
```

```
if(errcode==0)
((Serdlg*)dlg)->Accept();
CSocket::OnAccept(errcode);
}
void mysocket::OnReceive(int errcode)
{
if(errcode==0)
((Serdlg*)dlg)->Receive(this);
CSocket::OnAccept(errcode);
}
void mysocket::OnClose(int errcode)
{
if(errcode==0)
((Serdlg*)dlg)->OnCloseClient(this);
}
```

myapp.cpp: myapp sınıfının uygulaması

```
#include<afxwin.h>
#include "myapp.h"
#include "Serdlg.h"
#include "resource.h"
BOOL myapp::InitInstance()
{
Serdlg dlg(IDD_DIALOG1);
m_pMainWnd=&dlg;
dlg.DoModal();
```

```
return FALSE;
}
myapp a;
// Serdlg.cpp: serdlg sınıfının uygulaması
#include <afxwin.h>
#include "Serdlg.h"
#include "about.h"
#include "mysocket.h"
#include "resource.h"
BEGIN_MESSAGE_MAP(Serdlg,CDialog)
ON_COMMAND(IDC_BUTTON1,OnStart)
ON_COMMAND(IDCANCEL,OnCloseDlg)
ON_WM_PAINT()
ON_WM_CTLCOLOR()
ON_WM_SYSCOMMAND()
END_MESSAGE_MAP()
Serdlg::Serdlg(int n):CDialog(n)
{
reccount=0;
buser.Empty();
broadcast=0;
}
Serdlg::~Serdlg()
{
}
```

```
int Serdlg::OnInitDialog()
{
SetDlgItemInt(IDC_EDIT1,1051);
displist=(CListBox *)GetDlgItem(IDC_LIST2);
CDialog::OnInitDialog();
HICON m_hIcon = AfxGetApp()->LoadIcon(IDI_ICON2);
SetIcon(m_hIcon,TRUE);
if(!AfxSocketInit())
{
AfxMessageBox("Unable to initialize socket");
CDialog::OnCancel();
}
userlist.Empty();
broadcast=0;
return TRUE;
}
void Serdlg::OnPaint()
{
CRect r;
CPen p[64];
int i,factor;
CBrush mybrush;
CPaintDC pdc(this);
GetClientRect(&r);
factor=r.bottom/63;
```

```
for(i=0;i<64;i++)
p[i].CreatePen(PS_SOLID,1,RGB(100,90+i*2,250));

for(i=0;i<r.bottom;i++)
{
pdc.SelectObject(&p[i/factor]);
pdc.MoveTo(0,i);
pdc.LineTo(r.right,i);
}
}
HBRUSH Serdlg::OnCtlColor(CDC *pdc,CWnd *pwnd,UINT ctrl)
{
int id=pwnd->GetDlgCtrlID();
pdc->SetTextColor(RGB(0,0,255));

switch(id)
{
case 1093:case 1094:
pdc->SetBkMode(TRANSPARENT);
HBRUSH hbr=CreateSolidBrush(RGB(255,250,220));
return hbr;
}
return NULL;
}
/* Start butonuna basıldığında çağrılan foksiyon
void Serdlg::OnStart()
```

```
{
int port=GetDlgItemInt(IDC_EDIT1);
CButton *start=(CButton*)this->GetDlgItem(IDC_BUTTON1);
if(port<1024)
{
AfxMessageBox("1023 den büyük bir prot numarası giriniz");
return;
}
if(sersock.Create(port))
{
sersock.Listen();
sersock.setparent(this);
}
if(start!=NULL)      //start butonu kapat
start->EnableWindow(FALSE);
}
/* Kapat butonuna basıldığında çağrılan fonksiyon
/* istemcilere sunucunun kapatıldığını  haber verir

void Serdlg::OnCloseDlg()
{
sersock.Close();
CDialog::OnCancel();
}
/* istemci bağlantıları kabul eder ve listeye yerlestirir
```

```
/* "mysocket OnAccept fonksiyonundan cağrılıyor.

void Serdlg::Accept()

{

mysocket *client=new mysocket();

if(sersock.Accept(*client))
{

clientlist.AddTail(client);

client->setparent(this);

}

}

/* istemciden msj alır ve diğer istemcilere yönlendirir.

/*  "mysocket OnReceive fonk dan cağrılıyor.

void Serdlg::Receive(mysocket *client)

{

char buff[2021];

char str[100];

int size=2020,count,index,i;

CString mesg,header,test,bmesg;

//msj alinir ve buff da saklanir

count=client->Receive(buff,size);


if(count==SOCKET_ERROR)

{

return;

}

if(count>2020 || count<0)
```

```
return;
buff[2020]=NULL;
}
else
{
buff[count]=NULL;
}
mesg=buff;
index=mesg.Find(':');
if(index==-1)
{
return;
}
header=mesg.Left(index);
mesg=mesg.Right(mesg.GetLength()-index-1);
if(header.CompareNoCase("NEW")==0)  //yeni istemcinin kullanıcı adı
{
test=mesg;
test.MakeUpper();
displist->AddString(test);
userlist+=test+";";
client->name=test;
test="USER:"+userlist;
SendToAll(client->name,(char*)(LPCTSTR)test,test.GetLength(),0);
```

```
return;
}
if(broadcast==1 && buser.CompareNoCase(client->name)!=0)
return;
if( broadcast==1 && header.CompareNoCase("OVER")==0)
{
broadcast=0;
buser.Empty();
bmesg="RESUME:";
SendToAll(client->name,(char*)(LPCTSTR)bmesg,bmesg.GetLength(),1);
return;
}
for(i=0;i<client->name.GetLength();i++)
buff[i]=client->name[i];
buff[i]=':';
if(header.CompareNoCase("ALL")==0)
{
if(broadcast==0)
{
buser=client->name;
bmesg="WAIT:"+buser;
broadcast=1;
SendToAll(client->name,(char*)(LPCTSTR)bmesg,bmesg.GetLength(),1);
Sleep(200);
}
```

```
SendToAll(client->name,buff,count,1);
}
else
SendToClient(header,buff,count);
reccount++;
}
/* tüm bagli istemcilere msj gonderir
void Serdlg::SendToAll(CString name,char *data,int size,int flag)
{
mysocket *temp;
POSITION pos=clientlist.GetHeadPosition();
for(int i=0;i<clientlist.GetCount();i++)
{
temp=(mysocket *)clientlist.GetNext(pos);
if(flag && (name.CompareNoCase(temp->name)==0))
continue;
temp->Send(data,size);
}
}
/* belirli kullanıcıya gönderme  fonksiyonu
void Serdlg::SendToClient(CString user,char *data,int size)
{
mysocket *temp;
POSITION pos=clientlist.GetHeadPosition();
```

```
for(int i=0;i<clientlist.GetCount();i++)
{
temp=(mysocket *)clientlist.GetNext(pos);
if(user.CompareNoCase(temp->name)==0)
{
temp->Send(data,size);
break;
}
}
}
/* bir kullanıcı icin acılmış socket i kapatır
/*  mysocket onClose fonk   dan cağrılıyor.
void Serdlg::OnCloseClient(mysocket *client)
{
mysocket *temp;
POSITION delpos,pos=clientlist.GetHeadPosition();
for(int i=0;i<clientlist.GetCount();i++)
{
delpospos;
temp=(mysocket *)clientlist.GetNext(pos);
if(temp->name==client->name)
{
clientlist.RemoveAt(delpos);
break;
}
```

```
}
int index=userlist.Find(client->name);
if(index!=-1)
userlist.Delete(index,client->name.GetLength()+1);
CString test="USER:"+userlist;
//Yeni kullanıcı listesini tüm bağlı istemcilere gönder
SendToAll(client->name,(char*)(LPCTSTR)test,test.GetLength(),0);
UpdateList(client->name);
client->Close();
}
/* diyalogdaki listboxdan istemcinin ismini cıkarır.
/* OnCloseClient dan cağrılıyor.
void Serdlg::UpdateList(CString name)
{
int index=displist->FindString(0,name);
if(index!=LB_ERR)
displist->DeleteString(index
```

EK-2

VoIP uygulaması kapsamında kullanıcı (client) uygulamasında bulunan kodlar aşağıdaki gibidir.

```
// display sınıfının uygulaması
#include<afxwin.h>
#include<afxcmn.h>
#include<afxdlgs.h>
#include "about.h"
#include "Display.h"
#include "PlaySound1.h"
#include "resource.h"
#include "Volume.h"
#include "Save.h"
#include "WriteSound.h"
BEGIN_MESSAGE_MAP(Display,CDialog)
ON_COMMAND(IDC_BUTTON1,Onconnect)
ON_COMMAND(IDC_BUTTON5,OnStart)
ON_COMMAND(IDC_BUTTON6,OnStop)
ON_WM_DESTROY()
ON_WM_PAINT()
//ON_WM_ERASEBKGND()
ON_WM_CTLCOLOR()
```

```
END_MESSAGE_MAP()
Display::Display(int n):CDialog(n)
{
isconnected=FALSE;
sendcount=0;
reccount=0;
PreCreateHeader();
}
Display::~Display()
{
//ayrılan bellegi geri verir.
for(int i=0;i<MAXBUFFER;i++)
{
if(playhead[i]->lpData)
delete playhead[i]->lpData;
if(playhead[i])
delete playhead[i];
}
}
void Display::PreCreateHeader()
{
curhead=0;
isstart=1;
for(int i=0;i<MAXBUFFER;i++)
{
```

```
playhead[i] = new WAVEHDR;
ZeroMemory(playhead[i], sizeof(WAVEHDR));  //baslangic 0
char *temp=new char[PLAYBUFFER+50];
playhead[i]->lpData =temp;
playhead[i]->dwBufferLength = PLAYBUFFER;
playhead[i]->dwFlags=0;
}
}
int Display::OnEraseBkgnd(CDC *pdc)
{
return CDialog::OnEraseBkgnd(pdc);
}
int Display::OnInitDialog()
{
CFont myfont;
isSave=FALSE;
myfont.CreateFont(20,12,0,0,0,0,0,0,0,0,0,0,0,"System");
SetDlgItemText(IDC_EDIT1,"localhost");
SetDlgItemInt(IDC_EDIT2,1051);
SetDlgItemText(IDC_EDIT3,"istemci");
cbox=(CComboBox*)GetDlgItem(IDC_COMBO1);
curuser.Empty();
HICON m_hicon=AfxGetApp()->LoadIcon(IDI_ICON1);
SetIcon(m_hicon,TRUE);
SetIcon(m_hicon,FALSE);
```

```
//socket olustur
if(!::AfxSocketInit())
{
return 0;
}
//client socket nesnesi olustur
sockclt.setparent(this);
sockclt.Create();
record=new RecordSound(this);
record->CreateThread();
play=new PlaySound1(this);
play->CreateThread();
write=new WriteSound();
write->CreateThread();
selectflag=0;
start=stop=NULL;
return CDialog::OnInitDialog();
}
void Display::OnPaint()
{
CRect r;
CPen p[64];
int i,factor;
CPaintDC pdc(this);
GetClientRect(&r);
```

```
factor=r.bottom/63;
for(i=0;i<64;i++)
p[i].CreatePen(PS_SOLID,1,RGB(100 ,90 + i*2,250));
for(i=0;i<r.bottom;i++)
{
pdc.SelectObject(&p[i/factor]);
pdc.MoveTo(0,i);
pdc.LineTo(r.right,i);
}
}
HBRUSH Display::OnCtlColor(CDC *pdc,CWnd *pwnd,UINT ctrl)
{
int id=pwnd->GetDlgCtrlID();
pdc->SetTextColor(RGB(0,0,255));
switch(id)
{
case 1098:
case 1099:
case 1086:
case 1071:
case 1075:
pdc->SetBkMode(TRANSPARENT);
HBRUSH hbr=CreateSolidBrush(RGB(100,200,250));
return hbr;
}
```

```
return NULL;
}
/* kullanici  bağlan (connect ) butonuna bastığında çağrılan foksiyon,
void Display::Onconnect()
{
int port;
CString sername,username,mesg;
GetDlgItemText(IDC_EDIT1,sername);
GetDlgItemText(IDC_EDIT3,username);
port=GetDlgItemInt(IDC_EDIT2);
if(username=="" || sername=="" )
{
AfxMessageBox("Lutfen bilgileri dogru bicimde giriniz");
return;
}
//clienti servera bağlar
username.MakeUpper();
sockclt.name=username;
SetDlgItemText(IDC_BUTTON1,"Baglanıyor");
if(sockclt.Connect(sername,port))
{
// setDlg()
}
else
{
```

```
SetDlgItemText(IDC_BUTTON1,"&Baglan");
MessageBox("Sunucuya baglanamıyor\nPort ve sunucu bilgilerini kontrol edin");
return;
}
isconnected=TRUE;
//sunucuya login msj gönder
mesg="NEW:"+username;
sockclt.Send(mesg,mesg.GetLength());
updateState(FALSE,TRUE);
}
/* sunucuya başarılı biçimde bağlanıldığında diyaloğun durumunu değiştiriyor
void Display::updateState(BOOL pstate,BOOL nstate)
{
CEdit *eser,*eport,*euser;
CStatic *sser,*sport,*suser,*sgroup1,*sgroup2;
CButton *con;
con=(CButton*)GetDlgItem(IDC_BUTTON1);
con->ShowWindow(pstate);
eser=(CEdit*)GetDlgItem(IDC_EDIT1);
eser->ShowWindow(pstate);
eport=(CEdit*)GetDlgItem(IDC_EDIT2);
eport->ShowWindow(pstate);
euser=(CEdit*)GetDlgItem(IDC_EDIT3);
euser->ShowWindow(pstate);
sser=(CStatic*)GetDlgItem(1098);
```

```
sser->ShowWindow(pstate);
suser=(CStatic*)GetDlgItem(1086);
suser->ShowWindow(pstate)
sport=(CStatic*)GetDlgItem(1099);
sport->ShowWindow(pstate);
sgroup1=(CStatic*)GetDlgItem(1097);
sgroup1->ShowWindow(pstate);
sgroup2=(CStatic*)GetDlgItem(1075);
sgroup2->ShowWindow(nstate);
start=(CButton*)GetDlgItem(IDC_BUTTON5);
stop=(CButton*)GetDlgItem(IDC_BUTTON6);
start->ShowWindow(nstate);
cbox->ShowWindow(nstate);
}
/*  client sunucudan msj alındığında cağrılan fonksiyon
/*("mysocket" OnReceive fonk.dan cağrılıyor  )
void Display::Receive()
{
char buff[PLAYBUFFER+25],str[PLAYBUFFER+60];
int size=PLAYBUFFER+20,rcount,index,i,j,length;
CString mesg,header,disp;
rcount=sockclt.Receive(buff,size);
if(rcount==SOCKET_ERROR)
{
return;
```

```
}
if(rcount>(PLAYBUFFER+20))
{
return;
}
buff[rcount]=NULL;
mesg=buff;
index=mesg.Find(':');
//hatali msj formatı
if(index==-1)
return;
header=mesg.Left(index);
//sunucu kullanıcı listesi mi göndermiş
if(header=="USER")
{
mesg=mesg.Right(mesg.GetLength()-index-1);
updateList(mesg);
return;
}
if(header=="WAIT")
{
 if(isstart==0)  // programı durdur
OnStop();
start->EnableWindow(FALSE);
mesg=mesg.Right(mesg.GetLength()-index-1);
```

```
showFlash();
return;
}
if(header=="RESUME")
{
start->EnableWindow(TRUE);
showFlash();
return;
}
else          // kayıtlı ses çalmak için
{
length=0;
sscanf(&buff[15],"%d",&length);
if(length<1 || length>PLAYBUFFER)
return  ;
if(play->Playing==FALSE)
play->PostThreadMessage(WM_PLAYSOUND_STARTPLAYING,0,0);
LPWAVEHDR lpHdr=playhead[curhead];
curhead=(curhead+1)%MAXBUFFER;
//playmesg=new char[2020];
for(i=20,j=0;j<length;i++,j++)
lpHdr->lpData[j]=buff[i];
lpHdr->dwBufferLength=length;
lpHdr->dwFlags=0;
play-
```

```
>PostThreadMessage(WM_PLAYSOUND_PLAYBLOCK,0,(LPARAM)lpHdr)

if(isSave==TRUE && writeuser.CompareNoCase(header)==0)
{
write
>PostThreadMessage(WM_WRITESOUND_WRITEDATA,0,(LPARAM)lpHdr
}
disp="  "+header+" konusuyor...";
SetDlgItemText(6051,(char*)(LPCTSTR)disp);
}
}
void Display::showFlash()
{
FlashWindow(TRUE);
Sleep(200);
FlashWindow(TRUE)
}
void Display::OnStartWrite(char *name)
{
isSave=TRUE;
write
>PostThreadMessage(WM_WRITESOUND_CREATEFILE,0,(LPARAM)name);
}
void Display::OnStopWrite()
{
```

```
isSave=FALSE;
write->PostThreadMessage(WM_WRITESOUND_CLOSEFILE,0,0);
}
/*  client sunucudan yeni listeyi alınca
/*kendi listesini güncellemek icin kull. fonk.
void Display::updateList(CString mesg)
{
int index,num,prevcount;
CString name;
static int first=0;
CString disp;
prevcount=cbox->GetCount();
cbox->ResetContent();
do
{
index=mesg.Find(';');
if(index==-1)
break;
name=mesg.Left(index);
if(name.CompareNoCase(sockclt.name)!=0)
cbox->AddString(name);
mesg=mesg.Right(mesg.GetLength()-index-1);
}
while(TRUE)
if(cbox->GetCount()==1)
```

```
{
cbox->SetCurSel(0);
cbox->GetLBText(0,curuser);
selectflag=1;
if(prevcount==0 && first==1)
{
showFlash();
disp="yeni kullanıcı " + curuser + " baglandı";
SetDlgItemText(6051,(char*)(LPCTSTR)disp);
}
}
first=1;
if(cbox->GetCount()>1)
{
if(curuser.IsEmpty()==FALSE)
{
if((num=cbox->FindStringExact(-1,curuser))>=0)
{
selectflag=1;
cbox->SetCurSel(num);
return;
}
}
selectflag=0;
OnStop();
```

```
cbox->SetCurSel(0);
if(curuser.IsEmpty()==FALSE && curuser!="ALL" )
{
showFlash();
//SetDlgItemText(6051,(char*)(LPCTSTR)disp);
}
else if(curuser!="ALL")
{
showFlash();
//SetDlgItemText(6051,(char*)(LPCTSTR)disp);
}
}
if(cbox->GetCount()==0)
{
OnStop();
if(curuser.IsEmpty()==FALSE && curuser!="ALL")
{
showFlash();
disp=curuser+" logged out";
SetDlgItemText(6051,(char*)(LPCTSTR)disp);
}
//SetDlgItemText(6051,"");
//from->ResetContent();
curuser.Empty();
}
```

```
}
void Display::OnStart()
{
if(cbox->GetCount()<1)
{
MessageBox("Kullanıcı  secilmedi");
return;
}
if(!isstart)
return;
cbox->GetLBText(cbox->GetCurSel(),curuser);
selectflag=1;
startRecording();
startPlaying();
isstart=0;
start->ShowWindow(FALSE);
stop->ShowWindow(TRUE);
}
void Display::OnStop()
{
if(isstart)
return;
stopRecording();
stopPlaying();
selectflag=0
```

```
isstart=1;
start->ShowWindow(TRUE);
stop->ShowWindow(FALSE);
}
void Display::startRecording()
{
if(record->recording==FALSE)
{
//if(sockclt.name.CompareNoCase("logname")==0)
record->PostThreadMessage(WM_RECORDSOUND_STARTRECORDING,0,0);
}
}
void Display::stopRecording()
{
if(record->recording==TRUE)
record->PostThreadMessage(WM_RECORDSOUND_STOPRECORDING,0,0);
}
void Display::startPlaying()
{
if(play->Playing==FALSE)
play->PostThreadMessage(WM_PLAYSOUND_STARTPLAYING,0,0);
}
void Display::stopPlaying()
{
if(play->Playing==TRUE)
```

```
play->PostThreadMessage(WM_PLAYSOUND_STOPPLAYING,0,0);
}

/*  sunucuya ses mesajı  göndermek için kullnılan fonk.
void Display::sendMessage(char *mesg,int length)
{
char buflen[15];
char str[PLAYBUFFER+50];
int i,j;
//hicbir kullanıcı seçilmemişse
if(selectflag==0 )
return;
if(mesg==NULL)
{
// buffer boş ses alınmamış
return;
}
if(length<1 || length>PLAYBUFFER)
{
// cok uzun buffer
return;
}
mesg=mesg-20;
//ilk 15 bits hedef adresi iceriyor
//sonraki  5 bits msj uzunlugunu verir
```

```
//kalan kısım ses bilgisi
if(curuser.IsEmpty())
{
return;
}
for(i=0;i < curuser.GetLength();i++)
mesg[i]=curuser[i];
mesg[i]=':';
sprintf(buflen,"%d",length);
for(j=0,i=15;j<5;j++,i++)
mesg[i]=buflen[j];
sockclt.Send(mesg,length+20);
}
/*  pencereyi, tum thread ve socketleri kapat
void Display::OnCancel()
{
//calma ve kaydetme threadlerinin tumunu kapat
if(record->recording==TRUE)
record->PostThreadMessage(WM_RECORDSOUND_STOPRECORDING,0,0);
record->PostThreadMessage(WM_RECORDSOUND_ENDTHREAD,0,0);
if(play->Playing==TRUE)
play->PostThreadMessage(WM_PLAYSOUND_STOPPLAYING,0,0);
play->PostThreadMessage(WM_PLAYSOUND_ENDTHREAD,0,0);
write->PostThreadMessage(WM_WRITESOUND_ENDTHREAD,0,0);
if(sockclt.closeflag==1)
```

```
{
showFlash();
MessageBox("Server has shutdown");
}
if(isconnected==TRUE)
sockclt.Close();
CDialog::OnCancel();
}
```

MicPhone sınıfının uygulaması:

```
#include<afxwin.h>
#include <mmsystem.h>
#include"MicPhone.h"
void MicPhone::SetMicrophone(
log.Open("mic.txt",CFile::modeCreate | CFile::modeWrite);
if(openMixer())
{
if(getMicControl())
{
selectMic(1);
}
else
log.WriteString("\nGet mic control failed");
closeMixer();
}
else
```

```
log.WriteString("\nOpen mixer failed");
}
BOOL MicPhone::openMixer()
{
int n;
n=mixerGetNumDevs();
m_mixer=NULL;
::ZeroMemory(&mixcap,sizeof(MIXERCAPS));
if(n==0)
return FALSE;
if(mixerOpen(&m_mixer,0,NULL,NULL,MIXER_OBJECTF_MIXER)!=MMSYSE
RR_NOERROR)
return FALSE;
if(
::mixerGetDevCaps((UINT)m_mixer,&mixcap,
sizeof(MIXERCAPS) )!=MMSYSERR_NOERROR)
return FALSE;
 return TRUE;
}
void MicPhone::closeMixer()
{
if(m_mixer!=NULL)
::mixerClose(m_mixer);
}
BOOL MicPhone::getMicControl()
```

```
{
int mmret;
MIXERLINE mxln;
MIXERCONTROL mxcon;
MIXERLINECONTROLS mxlncon;
mxln.cbStruct=(sizeof(MIXERLINE));
mxln.dwComponentType=MIXERLINE_COMPONENTTYPE_DST_WAVEIN
mmret=mixerGetLineInfo(
(HMIXEROBJ)m_mixer,&mxln,MIXER_OBJECTF_HMIXER |

MIXER_GETLINEINFOF_COMPONENTTYPE );
if(mmret!=MMSYSERR_NOERROR)
{
return FALSE;
}
log.WriteString("\n Got the line info");
mcontype=MIXERCONTROL_CONTROLTYPE_MIXER;
mxlncon.cbStruct=sizeof(MIXERLINECONTROLS);
mxlncon.dwControlType=mcontype;
mxlncon.cControls=1;
mxlncon.dwLineID=mxln.dwLineID;
mxlncon.cbmxctrl=sizeof(MIXERCONTROL);
mxlncon.pamxctrl=&mxcon;
mmret=mixerGetLineControls((HMIXEROBJ)m_mixer,&mxlncon,MIXER_GETLI
NECONTROLSF_ONEBYTYPE | MIXER_OBJECTF_HMIXER );
```

```
log.WriteString("\n Got the line control info");
if(mmret!=MMSYSERR_NOERROR)
{
log.WriteString("\n Once again find.. line control");
mcontype = MIXERCONTROL_CONTROLTYPE_MUX;
mxlncon.cbStruct = sizeof(MIXERLINECONTROLS);
mxlncon.dwLineID = mxln.dwLineID;
mxlncon.dwControlType = mcontype;
mxlncon.cControls = 1;
mxlncon.cbmxctrl = sizeof(MIXERCONTROL);
mxlncon.pamxctrl = &mxcon;
if (::mixerGetLineControls(reinterpret_cast<HMIXEROBJ>(m_mixer),
 &mxlncon,
MIXER_OBJECTF_HMIXER |
MIXER_GETLINECONTROLSF_ONEBYTYPE)
!= MMSYSERR_NOERROR)
{
return FALSE;
}
}
mitems=mxcon.cMultipleItems;
conname=mxcon.szName;
mcontrolid=mxcon.dwControlID;
if(mitems==0)   //SHOULD BE GREATER THAN 2
{
```

```
return FALSE;
}
log.WriteString("\n Finding src for microphone");
MIXERCONTROLDETAILS_LISTTEXT *pmxcdSelectText =
new MIXERCONTROLDETAILS_LISTTEXT[mitems];
if (pmxcdSelectText != NULL)
{
MIXERCONTROLDETAILS mxcd;
mxcd.cbStruct = sizeof(MIXERCONTROLDETAILS);
mxcd.dwControlID = mxcon.dwControlID;
mxcd.cChannels = 1;
mxcd.cMultipleItems = mitems;
mxcd.cbDetails = sizeof(MIXERCONTROLDETAILS_LISTTEXT);
mxcd.paDetails = pmxcdSelectText;
if (::mixerGetControlDetails(reinterpret_cast<HMIXEROBJ>(m_mixer),
&mxcd,
MIXER_OBJECTF_HMIXER |

MIXER_GETCONTROLDETAILSF_LISTTEXT)
== MMSYSERR_NOERROR)
{
for (DWORD dwi = 0; (signed)dwi < mitems; dwi++)
{
MIXERLINE mxl;
mxl.cbStruct = sizeof(MIXERLINE);
mxl.dwLineID = pmxcdSelectText[dwi].dwParam1;
```

```
if (::mixerGetLineInfo(reinterpret_cast<HMIXEROBJ>(m_mixer),
&mxl,
MIXER_OBJECTF_HMIXER |
MIXER_GETLINEINFOF_LINEID)
== MMSYSERR_NOERROR &&
mxl.dwComponentType==
MIXERLINE_COMPONENTTYPE_SRC_MICROPHONE)
{
mindex = dwi;
micname = pmxcdSelectText[dwi].szName;
break;
}
}
if ((signed)dwi >= (signed)mitems)
{
for (dwi = 0; (signed)dwi < (signed)mitems; dwi++)
{
if (::lstrcmp(pmxcdSelectText[dwi].szName,
_T("Microphone")) == 0)
{
log.WriteString("\n Got the Source");
mindex = dwi;
micname = pmxcdSelectText[dwi].szName;
break;
```

```
}
}
}
}
delete []pmxcdSelectText;
}
return((unsigned) mindex <(unsigned) mitems);
}
BOOL MicPhone::selectMic(int val)
{
if (m_mixer == NULL ||
mitems == 0 ||
mindex >= mitems)
{
return FALSE;
}
BOOL bRetVal = FALSE;
 log.WriteString("\n Setting value");
MIXERCONTROLDETAILS_BOOLEAN *pmxcdSelectValue =
new MIXERCONTROLDETAILS_BOOLEAN[mitems];
if (pmxcdSelectValue != NULL)
{
MIXERCONTROLDETAILS mxcd;
mxcd.cbStruct = sizeof(MIXERCONTROLDETAILS);
mxcd.dwControlID = mcontrolid;
```

```
mxcd.cChannels = 1;
mxcd.cMultipleItems = mitems;
mxcd.cbDetails = sizeof(MIXERCONTROLDETAILS_BOOLEAN);
mxcd.paDetails = pmxcdSelectValue;
if (::mixerGetControlDetails(reinterpret_cast<HMIXEROBJ>(m_mixer),
 &mxcd,
MIXER_OBJECTF_HMIXER |
MIXER_GETCONTROLDETAILSF_VALUE)
== MMSYSERR_NOERROR)
{
if (val != 0 && mcontype == MIXERCONTROL_CONTROLTYPE_MUX)
{
::ZeroMemory(pmxcdSelectValue,
 mitems * sizeof(MIXERCONTROLDETAILS_BOOLEAN));
}
pmxcdSelectValue[mindex].fValue = val;
mxcd.cbStruct = sizeof(MIXERCONTROLDETAILS);
mxcd.dwControlID = mcontrolid;
mxcd.cChannels = 1;
mxcd.cMultipleItems = mitems;
mxcd.cbDetails = sizeof(MIXERCONTROLDETAILS_BOOLEAN);
mxcd.paDetails = pmxcdSelectValue;
if (::mixerSetControlDetails(reinterpret_cast<HMIXEROBJ>(m_mixer),
 &mxcd,
```

```
MIXER_OBJECTF_HMIXER |

MIXER_SETCONTROLDETAILSF_VALUE)
== MMSYSERR_NOERROR)
{
bRetVal = TRUE;
}
}
delete []pmxcdSelectValue}
return bRetVal;
}
```

mysocket sınıfının uygulaması

```
include<afxwin.h>
#include<afxsock.h>
#include "mysocket.h"
#include"Display.h"
mysocket::mysocket()
{
closeflag=0;
name="";
mysocket::setparent(CDialog *dialog)
{
dlg=dialog;
closeflag=0;
}
```

```
void mysocket::OnReceive(int errcode)
{
if(errcode==0)
((Display*)dlg)->Receive();
CSocket::OnReceive(errcode);
}
//sunucu kapatıldığında çağrılır.
void mysocket::OnClose(int errcode)
{
closeflag=1
((Display*)dlg)->OnCancel();
}
```

PlaySound1 sınıfı için yazılan uygulama:

```
#include<afxwin.h>
#include<mmsystem.h>
#include<mmreg.h>
#include "PlaySound1.h"
#include "Display.h"
IMPLEMENT_DYNCREATE(PlaySound1, CWinThread)
BEGIN_MESSAGE_MAP(PlaySound1, CWinThread)
ON_THREAD_MESSAGE(WM_PLAYSOUND_STARTPLAYING,
OnStartPlaying)
ON_THREAD_MESSAGE(WM_PLAYSOUND_STOPPLAYING, OnStopPlaying)
ON_THREAD_MESSAGE(WM_PLAYSOUND_PLAYBLOCK,
OnWriteSoundData)
```

```
ON_THREAD_MESSAGE(MM_WOM_DONE, OnEndPlaySound1Data)
ON_THREAD_MESSAGE(WM_PLAYSOUND_ENDTHREAD,OnEndThread)
END_MESSAGE_MAP(
PlaySound1::PlaySound1()
{
}
PlaySound1::PlaySound1(CDialog *dialog)
{
log.Open("playfile.txt",CFile::modeCreate | CFile::modeWrite);
dlg=dialog;
GetDevProperty();
memset(&m_WaveFormatEx,0x00,sizeof(m_WaveFormatEx));
m_WaveFormatEx.wFormatTag = WAVE_FORMAT_PCM;
m_WaveFormatEx.nChannels = 1;
m_WaveFormatEx.wBitsPerSample = 8;
m_WaveFormatEx.cbSize = 0;
m_WaveFormatEx.nSamplesPerSec = SAMPLEPSEC;
m_WaveFormatEx.nAvgBytesPerSec = SAMPLEPSEC ;
m_WaveFormatEx.nBlockAlign = 1;
Playing = FALSE;
log.WriteString("\n In the constructor of Play sound");
}
PlaySound1::~PlaySound1()
{
log.Close();
```

```
}
void PlaySound1::GetDevProperty()
{
CString format;
WAVEOUTCAPS wavecap;
int propno[]= {
WAVECAPS_LRVOLUME ,
WAVECAPS_PITCH ,
WAVECAPS_PLAYBACKRATE,
WAVECAPS_SYNC ,
WAVECAPS_VOLUME,
WAVECAPS_SAMPLEACCURATE ,
};
CString propstr[]={
"WAVECAPS_LRVOLUME ",
"WAVECAPS_PITCH ",
"WAVECAPS_PLAYBACKRATE",
"WAVECAPS_SYNC" ,
"WAVECAPS_VOLUME",
"WAVECAPS_SAMPLEACCURATE" ,
};
format.Empty();
format="\nSpecial properties... \n";
for(int j=0;j<6;j++)
{
```

```
if( (wavecap.dwSupport & (unsigned)propno[j]) ==(unsigned) propno[j])
{
format+=propstr[j]+"\n";
}
}
log.WriteString(format);
}
BOOL PlaySound1::InitInstance()
{
return TRUE;
}
int PlaySound1::ExitInstance()
{
return CWinThread::ExitInstance();
}
LRESULT PlaySound1::OnStartPlaying(WPARAM wParam, LPARAM lParam)
{
MMRESULT mmReturn = 0;
if(Playing==TRUE)
return FALSE;
log.WriteString("\n Starting playing");
mmReturn = ::waveOutOpen( &m_hPlay, WAVE_MAPPER,
&m_WaveFormatEx, ::GetCurrentThreadId(), 0, CALLBACK_THREAD);
if(mmReturn )
displayError(mmReturn,"PlayStart");
```

```
else
{
Playing = TRUE;
DWORD volume=0xffffffff;
char str[100];
if(!waveOutSetVolume(m_hPlay,volume))
{
volume=0;
if(!waveOutGetVolume(m_hPlay,&volume))
{
wsprintf(str,"\n Volume is  %lx",volume);
log.WriteString(str);
}
}
}
return TRUE;
}
void PlaySound1::displayError(int code,char mesg[])
{
char errorbuffer[MAX_PATH];
char errorbuffer1[MAX_PATH];
waveOutGetErrorText( code,errorbuffer,MAX_PATH);
sprintf(errorbuffer1,"PLAY : %s :%x:%s",mesg,code,errorbuffer);
AfxMessageBox(errorbuffer1);
}
```

```
LRESULT PlaySound1::OnStopPlaying(WPARAM wParam, LPARAM lParam)

{

MMRESULT mmReturn = 0;

if(Playing==FALSE)

return FALSE;

log.WriteString("\n Stop playing");

mmReturn = ::waveOutReset(m_hPlay);

if(!mmReturn)

{

Playing = FALSE;

Sleep(500);

mmReturn = ::waveOutClose(m_hPlay);

}

return mmReturn;

}

LRESULT PlaySound1::OnEndPlaySound1Data(WPARAM wParam, LPARAM
lParam)

{

LPWAVEHDR lpHdr = (LPWAVEHDR) lParam;

if(lpHdr)

{

::waveOutUnprepareHeader(m_hPlay, lpHdr, sizeof(WAVEHDR));

}

return ERROR_SUCCESS;

}
```

```
LRESULT PlaySound1::OnWriteSoundData(WPARAM wParam, LPARAM lParam)
{
MMRESULT mmResult = 0;
LPWAVEHDR lpHdr=(LPWAVEHDR)lParam;
if(lpHdr==NULL)
return ERROR_SUCCESS;
if(Playing)
{
mmResult = ::waveOutPrepareHeader(m_hPlay, lpHdr, sizeof(WAVEHDR));
if(mmResult)
{
log.WriteString("\nError while preparing header");
return ERROR_SUCCESS;
}
mmResult = ::waveOutWrite(m_hPlay, lpHdr, sizeof(WAVEHDR));
if(mmResult)
{
log.WriteString("\nError while writing to device");
return ERROR_SUCCESS;
}
}
return ERROR_SUCCESS;
}
LRESULT PlaySound1::OnEndThread(WPARAM wParam, LPARAM lParam)
{
```

```
if(Playing==TRUE)

OnStopPlaying(0,0);

log.WriteString("\nEnding the play device");

// RecordSound sınıfı için uygulama

#include<afxwin.h>

#include<mmsystem.h>

#include<mmreg.h>

#include "RecordSound.h"

#include "Display.h"

#include "MicPhone.h"

#include "MicMute.h"

IMPLEMENT_DYNCREATE(RecordSound, CWinThread)

BEGIN_MESSAGE_MAP(RecordSound,CWinThread)

ON_THREAD_MESSAGE(MM_WIM_DATA, OnSoundData)

ON_THREAD_MESSAGE(WM_RECORDSOUND_STARTRECORDING,OnStart
Recording)

ON_THREAD_MESSAGE(WM_RECORDSOUND_STOPRECORDING,OnStopR
ecording)

ON_THREAD_MESSAGE(WM_RECORDSOUND_ENDTHREAD,OnEndThread)

END_MESSAGE_MAP()

RecordSound::RecordSound()

{

}

RecordSound::RecordSound(CDialog *dialog)

{

dlg=dialog;
```

```
log.Open("recfile.txt",CFile::modeCreate | CFile::modeWrite);
log.WriteString("In the Recordsound Constructor\n");
MicPhone *mic=new MicPhone;
mic->SetMicrophone();
MicMute *mute=new MicMute;
mute->SetMicrophone();
recording=FALSE;
isallocated=0;
GetDevProperty();
PreCreateHeader();
memset(&m_WaveFormatEx,0x00,sizeof(m_WaveFormatEx));
m_WaveFormatEx.wFormatTag=WAVE_FORMAT_PCM;
m_WaveFormatEx.nChannels=1;
m_WaveFormatEx.wBitsPerSample=8;
m_WaveFormatEx.cbSize=0;
m_WaveFormatEx.nSamplesPerSec=SAMPLERSEC;  //22.05 KHz
m_WaveFormatEx.nBlockAlign=1;
m_WaveFormatEx.nAvgBytesPerSec=SAMPLERSEC ;
}
RecordSound::~RecordSound()
{
log.Close();
if(!isallocated)
return;
for(int i=0;i<MAXRECBUFFER;i++)
```

```
{
if(rechead[i])
delete rechead[i];
}
}
BOOL RecordSound::InitInstance()
{
return TRUE;
}
int RecordSound::ExitInstance()
{
return CWinThread::ExitInstance();
}
void RecordSound::PreCreateHeader()
{
for(int i=0;i<MAXRECBUFFER;i++)
rechead[i]=CreateWaveHeader();
isallocated=1;
}
void RecordSound::GetDevProperty()
{
WAVEINCAPS wavecap;
int i,j,n=waveInGetNumDevs();
char str[100];
CString format;
```

```
int form[]={WAVE_FORMAT_1M08,
WAVE_FORMAT_1M16,
WAVE_FORMAT_1S08,
WAVE_FORMAT_1S16,
WAVE_FORMAT_2M08,
WAVE_FORMAT_2M16,
WAVE_FORMAT_2S08,
WAVE_FORMAT_2S16,
WAVE_FORMAT_4M08,
WAVE_FORMAT_4M16,
WAVE_FORMAT_4S08,
WAVE_FORMAT_4S16,
};
CString fstr[]={
 "WAVE_FORMAT_1M08",
"WAVE_FORMAT_1M16",
"WAVE_FORMAT_1S08",
"WAVE_FORMAT_1S16",
"WAVE_FORMAT_2M08",
"WAVE_FORMAT_2M16",
"WAVE_FORMAT_2S08",
"WAVE_FORMAT_2S16",
"WAVE_FORMAT_4M08",
"WAVE_FORMAT_4M16",
"WAVE_FORMAT_4S08",
```

```
"WAVE_FORMAT_4S16",
};
sprintf(str,"\n Total no of devices = %d ",n);
log.WriteString(str);
for(i=0;i<n;i++)
{
waveInGetDevCaps(i,&wavecap,sizeof(wavecap));
sprintf(str,"\n Product Name = %s ",wavecap.szPname);
log.WriteString(str);
sprintf(str,"\n No of channels %d ",wavecap.wChannels);
format.Empty();
format="\nIt supports  \n";
for(j=0;j<12;j++)
{
if( (wavecap.dwFormats & (unsigned)form[j]) ==(unsigned) form[j])
{
format+=fstr[j]+"\n";
}
}
log.WriteString(format);
}
}
LRESULT RecordSound::OnStartRecording(WPARAM wp,LPARAM lp)
{
MMRESULT mmReturn = ::waveInOpen( &m_hRecord, WAVE_MAPPER,
```

```
&m_WaveFormatEx, ::GetCurrentThreadId(), 0, CALLBACK_THREAD);
log.WriteString("In OnStartrecording\n");
//Error has occured while opening device
if(mmReturn!=MMSYSERR_NOERROR )
{
displayError(mmReturn,"Open");
return FALSE;
}
if(mmReturn==MMSYSERR_NOERROR )
{
for(int i=0; i < MAXRECBUFFER ; i++)
{
mmReturn = ::waveInPrepareHeader(m_hRecord,rechead[i], sizeof(WAVEHDR));
mmReturn = ::waveInAddBuffer(m_hRecord, rechead[i], sizeof(WAVEHDR));
}
mmReturn = ::waveInStart(m_hRecord);
if(mmReturn!=MMSYSERR_NOERROR )
displayError(mmReturn,"Start");
else
recording=TRUE;
}
return TRUE;
}
void RecordSound::displayError(int mmReturn,char errmsg[])
{
```

```
char errorbuffer[MAX_PATH];
char errorbuffer1[MAX_PATH];
waveInGetErrorText( mmReturn,errorbuffer,MAX_PATH);
sprintf(errorbuffer1,"RECORD: %s : %x : %s",errmsg,mmReturn,errorbuffer);
AfxMessageBox(errorbuffer1);
}
LRESULT RecordSound::OnStopRecording(WPARAM wp,LPARAM lp)
{
MMRESULT mmReturn = 0;
log.WriteString("\nIn the onstop recording");
if(!recording)
return FALSE;
mmReturn = ::waveInStop(m_hRecord);
if(!mmReturn)
{
recording = FALSE;
mmReturn = ::waveInReset(m_hRecord);
}
if(!mmReturn)
recording = FALSE;
Sleep(500);
if(!mmReturn)
mmReturn = ::waveInClose(m_hRecord);
return mmReturn;
}
```

```
LRESULT RecordSound::OnSoundData(WPARAM wParam, LPARAM lParam)
{
log.WriteString("\nIn the onsound data");
LPWAVEHDR lpHdr = (LPWAVEHDR) lParam;
if(lpHdr->dwBytesRecorded==0 || lpHdr==NULL)
return ERROR_SUCCESS;
::waveInUnprepareHeader(m_hRecord, lpHdr, sizeof(WAVEHDR));
if(lpHdr->lpData!=NULL )
((Display*)dlg)->sendMessage(lpHdr->lpData,lpHdr->dwBytesRecorded);
if(recording)
{
::waveInPrepareHeader(m_hRecord,lpHdr, sizeof(WAVEHDR));
::waveInAddBuffer(m_hRecord, lpHdr, sizeof(WAVEHDR));
}
return ERROR_SUCCESS;
}
LRESULT RecordSound::OnEndThread(WPARAM wp,LPARAM lp)
{
log.WriteString("\nIN the onendthread");
if(recording)
OnStopRecording(0,0);
::PostQuitMessage(0);
return TRUE;
}
LPWAVEHDR  RecordSound::CreateWaveHeader()
```

```
{
LPWAVEHDR lpHdr = new WAVEHDR;
if(lpHdr==NULL)
{
log.WriteString("\n Unable to allocate the memory");
return NULL;
}
ZeroMemory(lpHdr, sizeof(WAVEHDR));
char* lpByte = new
char[RECBUFFER+20];//m_WaveFormatEx.nBlockAlign*SOUNDSAMPLES)];
if(lpByte==NULL)
{
log.WriteString("\n Unable to allocate the memory");
return NULL;
}
lpHdr->lpData =  &lpByte[20];
lpHdr->dwBufferLength =RECBUFFER;
return lpHdr;
}
```

ÖZGEÇMİŞ

1975 yılında Trabzon'da doğdu. İlk, orta ve lise öğrenimini Trabzon'da tamamladı. 1993 yılında girdiği Karadeniz Teknik Üniversitesi Mühendislik Fakültesi Elektrik ve Elektronik Mühendisliği Bölümü'nden 1997 yılında Elektronik Mühendisi olarak mezun oldu. 1997-1999 yılları arasında Teknotel İletişim A.Ş'de Telefon Şebekeleri ve Kablo TV bölümünde Planlama Mühendisi olarak çalıştı. 1999 yılında Türk Telekom A.Ş Kocaeli İl Müdürlüğünde göreve başladı. 1999-2001 yılları arasında Erişim Şebekeleri Bölümünde Kontrol Mühendisi olarak çalıştı. 2001 yılından beri aynı şirketin Data Direktörlüğü (TDM, ATM, ADSL ve IP/MPLS) bölümünde Network Mühendisi olarak çalışmaktadır. 2004 yılından beri Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı'nda Yüksek Lisans'a devam etmektedir.