IPv6 YAPILANDIRMASI VE SOKET TABANLI IPv6 DESTEKLİ SUNUCU YAZILIMI GELİŞTİRİLMESİ

İbrahim AKŞİT

YÜKSEK LİSANS TEZİ

YÖNETİM BİLİŞİM SİSTEMLERİ

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

TEMMUZ 2011

ANKARA

İbrahim AKŞİT tarafından hazırlanan IPv6 YAPILANDIRMASI VE SOKET TABANLI IPv6 DESTEKLİ SUNUCU YAZILIMI GELİŞTİRİLMESİ adlı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Prof. Dr. Şeref SAĞIROĞLU

Tez Yöneticisi

Bu çalışma, jürimiz tarafından oy birliğiyle Yönetim Bilişim Sistemleri Anabilim Dalında Yüksek Lisans Tezi olarak kabul edilmiştir.

Başkan : Prof. Dr. İlhami ÇOLAK

Üye : Prof. Dr. Cevriye GENCER

Üye : Prof. Dr. Şeref SAĞIROĞLU (Danışman)

Tarih : 07/07/2011

Bu tez, Gazi Üniversitesi Bilişim Enstitüsü tez yazım kurallarına uygundur.

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

İbrahim AKŞİT

IPv6 YAPILANDIRMASI
VE
SOKET TABANLI IPv6 DESTEKLİ
SUNUCU YAZILIMI GELİŞTİRİLMESİ

(Yüksek Lisans Tezi)

İbrahim AKŞİT

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

TEMMUZ 2011

ÖZET

IPv6 ağına bağlanacak mevcut ve yeni teknolojik ürünler üzerinden hizmet veren yazılımların da IPv6 desteği sunması kaçınılmaz hale gelmektedir. Mevcut yazılımların IPv6 adreslerinden gelen isteklere cevap vermesi büyük önem arz etmektedir. IPv6 desteği verebilen az sayıda yazılımın olması gelişen teknolojiye bakıldığında sorun teşkil etmektedir.

İnternetin, yaygınlaşan hareketli (mobil) veya hareketsiz elektronik ve kablosuz cihazların geleceğinin IPv6 protokolüne dayandığı ve internete bağlı cihazların donanım ve yazılımlarının buna göre şekilleneceği görülmektedir. Kullanmakta olduğumuz bütün cihazların ve hizmetlerin IPv6 desteklememesinden dolayı IPv6 kullanımına geçiş sürecinde IETF (Internet Engineering Task Force) tarafından önerilen ve uygulanması gereken çeşitli yöntemler vardır. Bu geçiş

mekanizmaları ikili yığın, tünelleme ve çeviriciler olmak üzere üç türde sınıflandırılmaktadır. Türkiye'de IPv6 geçişini sağlamak, hizmetini sunmak ve farkındalığı oluşturmak amacıyla TÜBİTAK - ULAKBİM'in yönetici, Gazi Üniversitesi ve Çanakkale 18 Mart Üniversitesi'nin yürütücü, Bilgi Teknolojileri ve İletişim Kurumu'nun müşteri olarak katıldığı "Ulusal IPv6 Protokol Altyapısı Tasarımı ve Geçişi Projesi" yapılmıştır.

IPv6 trafiğini uygulamalarda kullanabilmek için öncelikle soket tabanlı yazılımlarda IPv6 yapılandırması gerekir. Sunucu ve istemci tabanlı yazılımlarda IPv6 desteğini sağlayarak uygulamalarımızda IPv6 paketleri üzerinden haberleşme sağlanabilir.

Bu tez çalışmasında, mevcut sunucu tabanlı bir POP3 uygulamasına IPv6 desteği verilmesi için gerekli olan altyapı yazılım desteği ve değişiklikler yapılmış ve bunun için uygulama geliştirilmiştir. Geliştirilen bu yazılım ile gelen e-postaların IPv6 üzerinden iletilmesini sağlamak amacıyla IPv6'nın yapılandırılması yapılmış, geliştirilen uygulama Microsoft Office Outlook programı kullanılarak test edilmiş, IPv6 destekli soket tabanlı yazılımların geliştirilmesi noktasında çözüm önerileri sunulmuştur.

Bilim Kodu : 902.1.013
Anahtar Kelimeler : IPv4, IPv6, IPv6 adresleme, IPv6 yapılandırma, IPv6 geçiş yöntemleri, IPv6 destekli yazılım ve hizmetler
Sayfa Adedi : 145
Tez Yöneticisi : Prof. Dr. Şeref SAĞIROĞLU

IPv6 CONFIGURATON
AND
DEVELOPING SOCKET BASED
IPv6 SUPPORTED SERVER SOFTWARE

(M.Sc. Thesis)

İbrahim AKŞİT

GAZİ UNIVERSITY

INSTITUTE OF INFORMATICS

JULY 2011

ABSTRACT

New and existing technological products those will connect to IPv6 network providing IPv6 support for software servicing over those products are becoming inevitable. The current software responding to requests from IPv6 addresses present very important aspect. Having a small number of software being able to provide IPv6 support causes problems.

The future of spreading the movable (mobile) or immovable electronic and wireless devices based on IPv6 protocol and the devices connected to the Internet, which use hardware and software, are seen to be shaped accordingly. Since all devices and services those are used currently do not support IPv6, there are several methods proposed by IETF (Internet Engineering Task Force) needed to be applied to implement IPv6 support. Those transition mechanisms

are classified into three types named dual stack, tunneling and translators. "Design of National IPv6 Infrastructure and Transition to IPv6 Protocol" project coordinated by TÜBİTAK - ULAKBİM with participation of Gazi University, Çanakkale 18 Mart University and Turkish Information Technologies and Communication Authority has been carried out to implement IPv6 transition, supply IPv6 service and create awareness of IPv6 in Turkey

Socket-based software should be configured with IPv6 in order to use IPv6 traffic in applications. The communication can be achieved through IPv6 packets by implementing IPv6 support in the server and client-based software.

In this thesis study, the infrastructure software support and necessary changes have been made and an application has been developed in order to provide IPv6 support for an existing server based on POP3 application. IPv6 configuration has been carried out in order to ensure received e-mails transmission over IPv6 with this developed software, the software has been tested using Microsoft Office Outlook. Moreover, the solution proposals has been presented in the point of development of IPv6 supported socket-based software.

Science Code : **902.1.013**
Key Words : **IPv4, IPv6, IPv6 addressing, IPv6 configuration, IPv6 transition methods, IPv6 supported software and services**
Page Number : **145**
Advisor : **Prof. Dr. Şeref SAĞIROĞLU**

TEŞEKKÜR

Çalışmalarım boyunca yardım ve katkılarıyla beni yönlendiren tez danışmanım değerli hocam Prof. Dr. Şeref SAĞIROĞLU'na, çalışmamın test edilmesi konusunda bana yardımcı olan Emre YÜCE'ye ve Onur BEKTAŞ'a, maddi manevi destekleriyle beni hiçbir zaman yalnız bırakmayan eşime, biricik kızım Zeynep Sare'ye ve aileme sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklama
AFRINIC	Afrika Ağı Bilgi Merkezi (African Network Information Centre)
APNIC	Asya Pasifik Ağı Bilgi Merkezi (Asia Pacific Network Information Centre)
ARIN	Amerika İnternet Numaraları Kayıt Merkezi (American Registry For Internet Numbers
BIA	Uygulama Programlama Ara yüzünde Sarsıntı (Bump-In-The-API)
BIS	Yığında Sarsıntı (Bump-In-The-Stack)
CIDR	Sınıfsız Alanlar Arası Yönlendirme (Classless InterDomain Routing)
DHCP	Dinamik Makine Yapılandırma Protokolü (Dynamic Host Configuration Protocol)
DNS	Alan Adı Sistemi (Domain Name System)
DSTM	İkili Yığın Geçiş Mekanizması (Dual Stack Transition Mechanism)
EGP	Dış Ağ Geçidi Protokolü (Exterior Gateway Protocol)
EIGRP	Geliştirilmiş İç Ağ Geçidi Yönlendirme Protokolü (Enhanced Interior Gateway Routing Protocol)
FTP	Dosya Aktarım Protokolü (File Transfer Protocol)
GRE	Genel Yönlendirme Kapsüllenmesi (General Routing Encapsulation)
HTTP	Hiper Metin Aktarım Protokolü (Hyper-Text Transfer Protocol)
IANA	İnternet Numara Tahsis Otoritesi (İnternet Assigned Numbers Authority)
ICMP	İnternet Kontrol Mesaj İletişim Protokolü (Internet Control Message Protocol)

Kısaltmalar	Açıklama
IDRP	Alanlararası Yönlendirme Protokolü (Interdomain Routing Protocol)
IETF	İnternet Mühendisliği Görev Gücü (Internet Engineering Task Force)
IGMP	İnternet Grup Yönetim Protokolü (Internet Group Management Protocol)
IP	İnternet Protokolü (Internet Protocol)
IPv4	İnternet Protokolü Sürüm 4 (Internet Protocol Versiyon 4)
IPv6	İnternet Protokolü Sürüm 6 (Internet Protocol Versiyon 6)
IPSec	İnternet Protokol Güvenliği (Internet Protocol Security)
ISATAP	İç Site Otomatik Tünelleme Adres Protokolü (Intra-Site Automatic Tunnel Addressing Protokol)
L2TP	İkinci Katman Tünelleme Protokolü (Layer 2 Tunneling Protocol)
LACNIC	Latin Amerika ve Karayipler Ağı Bilgi Merkezi (Latin America and Caribbean Network Information Centre)
MAC	Ortam Erişim Kontrolü (Media Acces Control)
MTU	Maksimum İletim Birimi (Maximum Transfer Unit)
NAT	Ağ Adres Dönüşümü (Network Adress Translation)
NAT-PT	Ağ Adres Dönüşümü ve Protokol Dönüşümü (Network Adress Translation - Protocol Translation)
NIC	Ağ Arayüz Kartı (Network Interface Card)
OSPF	İlk Açık Yöne Öncelik (Open Shortest Path First)
RFC	Yorumlar İçin Talep (Request for Comments)
RIPE NCC	Avrupa İnternet Protokolü Ağ Koordinasyon Merkezi (Réseaux IP Européens Network Coordination Centre)

Kısaltmalar	Açıklama
RSVP	Kaynak Ayırma Protokolü (Resource Reservation Protocol)
SCTP	Akış Kontrol İletim Protokolü (Stream Control Transmission Protocol)
SNMP	Temel Ağ Yönetim Protokolü (Simple Network Management Protocol)
SIIT	Durumsuz IP/ICMP Çeviri Algoritması (Stateless IP/ICMP Translation Algorithm)
STD	İnternet Standardı (Internet Standart)
TCP	Aktarım Kontrol Protokolü (Transfer Control Protocol)
TCP/IP	Aktarım Kontrol Protokolü/İnternet Protokolü (Transfer Control Protocol/Internet Protocol)
TRT	Ulaşım Röle Çevirici (Transport Relay Translator)
TTL	Yaşam Süresi (Time To Live)
UDP	Kullanıcı Veribloğu Protokolü (User Datagram Protocol)

1. GİRİŞ

Günümüzde yaygın olarak kullandığımız IPv4 1982 yılında uluslararası bir organizasyon olan IETF (Internet Engineering Task Force) tarafından kabul edilmiştir [1]. IPv4 32- bit'lik bir yapıya ve yaklaşık 2^{32} = ~ 4 294 967 296 (yaklaşık 4,29 milyar) adrese sahiptir. Dünya nüfusu 1982 yılında yaklaşık 4,5 milyar iken öngörülen IPv4 o zamanki bilgisayar sayısına bakılarak yeterli görülmüştür. ABD Nüfus Sayımı Bürosunun verilerine göre dünya nüfusu 6 923 062 685'dır [2]. Sadece dünya nüfusu göz önünde bulundurulduğunda kişi başına bir IPv4 adresi bile düşmemektedir. Dünya nüfusuna bağlı olarak internetin, teknolojik alt yapıların hızla büyümesi neticesinde kullanılan IPv4 (Internet Protocol version 4) temelli ağların yetersiz kalması, IPv4 adreslerinin tükeneceği ve neredeyse elde IP'nin kalmaması büyük bir sorun teşkil etmektedir. 3 Şubat 2011 tarihinde ABD'nin Miami eyaletinde gerçekleştirilen basın konferansında yapılan açıklamaya göre APNIC (Asia Pacific Network Information Centre) ve IANA (Internet Assigned Numbers Authority) verilerine göre en son kalan beş (5) IPv4 adres bloğu ARIN (American Registry For Internet Numbers), AFRINIC (African Network Information Centre), APNIC (Asia Pacific Network Information Centre), LACNIC (Latin America and Caribbean Network Information Centre), ve RIPE NCC (Réseaux IP Européens Network Coordination Centre) arasında eşit olarak dağıtılmıştır [3,4].

Ağ protokollerinin geçmişine bakıldığında en başarılı ağ protokolünün IP olduğu ortaya çıkmaktadır. İnternet protokolünün diğer protokollerden daha başarılı olması, IPv4'ün artık ihtiyaçlara cevap verememesi, internet kullanıcısının hızla artması yeni nesil internet protokolünün geliştirilmesine yön vermiştir. Yeni nesil tasarım için 1992 yılı Temmuz ayında IETF IPv6 için öneride bulunmuştur. IANA tarafından yeni nesil IP (IPng) adına version 6 (v6) verilmiştir. 1995 yılı Ocak ayında IETF IPv6 yapısı için önerilerini tamamlamış ve RFC 1752 olarak yayınlamıştır [5,6].

IPv6 için yapılan önerilerden sonra bu protokole geçişler için çalışmalar başlamıştır. Dünya ülkeleri başta Çin, Japonya olmak üzere hızla IPv6'ya geçiş süreçleri ve çalışmaları başlatmıştır. IPv6 protokolünü ilk kez pratiğe geçiren ülke 1998 yılında

Çin olmuştur. Japonya Eylül 2000'de, Güney Kore Şubat 2001'de, Avrupa Komisyonu/Birliği Nisan 2001'de IPv6'yı benimsediklerini duyurmuşlardır [7]. Türkiye ise 15 Şubat 2009 tarihinde TÜBİTAK tarafından desteklenen "Ulusal IPv6 Protokol Altyapısı Tasarımı ve Geçişi Projesi" kapsamında IPv6'ya geçiş için çalışmalar başlamıştır [8]. Türkiye proje kapsamında 12-13 Ocak 2011 tarihlerinde Ankara'da Ulusal IPv6 Konferansı gerçekleştirmiştir. Konferansta IPv6 her yönüyle ele alınmış ve IPv6 hakkında yapılan çalışmalar sunulmuştur [8]. Yeni nesil IP'ye geçiş süreci mevcut altyapının IPv6 için eksiklikleri, engelleyici bazı faktörlerden dolayı kolay olmayacağı açıktır. İnternetin ve kablosuz bağlantıların (cihazların) geleceğinin IPv6 protokolüne dayandığı ve internete bağlı cihazların donanım ve yazılımlarının buna göre şekilleneceği görülmektedir.

Kullanmakta olduğumuz bütün cihazların ve hizmetlerin IPv6 desteklememesinden dolayı IPv6 kullanımına geçiş sürecinde uygulanması gereken çeşitli yöntemler vardır. IPv6'ya geçiş süreçlerinin belirlenmesi ve uygun altyapının oluşturulması gereklidir. IPv6 geçiş süreçlerinden bazıları hem IPv4 hem de IPv6'yı aynı anda kullanma olanağı sağlamaktadır. Bu geçiş süreçleri ikili yığın (Dual Stack), tünelleme (Tunelling) ve çeviri (Translation) metotları şeklindedir [9,10]. İkili yığın yaklaşımı mevcut IPv4 alt yapısını kullanarak IPv6'yı destekler hale getirmeyi hedefler. Hem IPv4 hem de IPv6 paketleri aynı ağ üzerinden iletişim sağlar. İkili yığın normalde tek başına IPv6'ya geçişi sağlayabilir fakat uygulanması zor ve maliyetli bir geçiş mekanizmasıdır [9,10]. Bu sebeple ikili yığına ek olarak tünelleme ve çeviri kullanılarak geçiş için çözüm önerileri sunulmuştur. Tünelleme türleri ise elle ayarlanmış tünelleme (Manual Configured Tunelling), ISATAP (Intra-Site Automatic Tunnel Addressing Protocol), Teredo (Tunneling IPv6 over UDP through NATs), 6 üzerinden 4 (6over4) ve 6'dan 4'e otomatik tünelleme (auto 6to4) mekanizmalarıdır [10-13]. Çeviri metotları IPv4 ve IPv6 tabanlı ağların TCP ve IP başlıklarını yeniden yazarak uygulama katmanı ağ geçitlerini - ALG (Application Layer Gateways) kullanarak birbirleriyle iletişimini sağlar. Çeviri metotları uygulamaları farklı türdeki ağları birbirine bağlayarak çalışmasını sağlarlar. Farklı ağların bir araya getirilerek iletişimleri sağlandığından yönetimi zor geniş ağlar meydana gelmektedir [14,15].

Bu çalışmada yeni nesil IP (Internet Protocol) olarak bilinen IPv6 (Internet Protocol version 6) hakkında kapsamlı bilgiler verilmiştir. IPv4'ten IPv6'ya geçişte çeşitli platformlarda - sistemlerde IPv6 adres yapılandırması, yönetimi hakkında bilgiler verilmiştir. IPv6 (Internet Protocol version 6) kullanacak cihazlar, bilgisayar ve bilgisayar ağlarının ve yazılımların IPv6 desteklemesi için gerekli süreçler, aşamalar ve yapılandırma ve hakkında bilgiler verilmiştir. Ayrıca var olan POP3 programı için IPv6 desteğini sağlayan kodlamalar yapılmıştır. IPv6 protokolünü kullanan sistemlerin dünyadaki diğer IPv6 ya da IPv4 protokolünü kullanan sistemlerle etkileşiminin, bilgi iletişiminin sağlanabilmesi için yapılması gereken yapılandırmaların ve yönetim tekniklerinin ne olması nasıl olması gerektiği araştırılmış; çözüm önerileri belirlenmiştir.

Bu çalışmanın ikinci bölümünde mevcut olarak kullandığımız internet protokolleri IPv4 ve IPv6 hakkında detaylı bilgiler verilip araştırma, çalışma ve karşılaştırmalar yapılmıştır.

Üçüncü bölümde, yayınlanan kitap, makale ve dergiler ışığında IPv6 için geçiş yöntemleri ve süreçleri ile ilgili bilgiler verilmiştir.

Dördüncü bölümde, geçiş yöntemleri ve süreçleri ışığında IPv6 desteği sağlanan ve en çok kullanılan donanım ve yazılımlar açıklamalarıyla birlikte verilmiştir.

Beşinci bölümde, C programlama dili kullanılarak mevcut sunucu tabanlı POP3 yazılımına IPv6 destek vermek için gerekli geliştirmeler yapılmıştır.

Altıncı bölümde, bu çalışmadan elde edilen sonuçlar verilmiş ve sonuçlar doğrultusunda gelecekte yapılabilecek çalışmalarla ilgili olarak öneriler sunulmuştur.

2. İNTERNET PROTOKOLLERİ

2.1. IPv4 (Internet Protocol version 4)

İnternet protokollerinin gelişimi sürecinde IP dördüncü kez gözden geçirilmiş ve geniş çapta kullanılan ilk sürümüdür. Günümüzde kullanılan internet IPv4 üzerine kurulmuştur. Şu ana kadar en yaygın kullanılan internet katman protokolüdür. Bu protokol, IP tabanlı sistemlerin ve cihazların uç uca adresleme yaparak birbirleriyle iletişim kurmasını sağlar. IPv4 IETF tarafından ilk tanımı Ocak 1980, son tanımı Eylül 1981 tarihinde yapılmıştır [1,16].

IP'nin bizlere sağladığı olanaklar şu şekilde sıralanabilir [16];

- Her yerde kullanılabilen (evrensel) adresleme yapısı,
- Gelen istekleri sınıflandırma,
- Gelen / Giden paketlerin uygun bir biçimde iletmek için parçalara ayırma,
- Hedef alıcıya gidecek paketleri tekrar birleştirme,
- TCP - UDP ile birlikte çalışabilme
- Ağlar arasında paketleri yönlendirme

TCP (Transmission Control Protocol), UDP (User Datagram Protocol) ve IP birbirleriyle çalışabilmektedirler. TCP hedef bilgisi belirlenen veriyi IP'ye iletir. IP ise bu veriyi alır ve yönlendirileceği bilgisayar veya sisteme yönlendirir. IP kendisine gönderilen verileri TCP ve UDP'den alır. Gönderilen verinin kontrolü TCP tarafından yapılır. Gönderilen veri, ses ve görüntü aktarımı gibi gerçek zamanlı veri aktarımı ise UDP tarafından kullanılır. TCP, IP başlığında bir bozulma olup olmadığını kontrol eder. Gelen IP paketinde bir bozulma var ise TCP kendi akış kontrol ve paket sıralama mekanizmasını kullanarak gönderilen bozuk paketin tekrar gönderilmesini sağlar [16].

TCP / IP kullanılma nedenleri [16];

- Temel bir standart olma
- Üreticilerden bağımsız tüm ürünleri ve sistemleri birbiriyle görüştürme

- Bütün bilgisayar türlerini birbirleriyle görüştürme (sunucu, dizüstü, mini bilgisayarlar v.b.)
- Unix, Linux ve Microsoft'un işletim sistemleriyle tam uyumluluk
- Birçok OSI (Open System Interconnection) 1. ve 2. katman protokollerini desteklemesi (Ethernet, Token-Ring, FDDI v.b.)

şeklinde sıralanabilir.

2.1.1. IPv4 paket yapısı

IPv4'de temel paket yapısı Çizelge 2.1'de gösterilmiştir. Datagram ifadesi ağ katmanında bulunan paket yapısıdır. Burada IP paket yapısında bulunan alanların açıklamaları yapılmaktadır. Aşağıda IPv4 paketinde bulunan alanların açıklamaları soldan sağa ve yukarıdan aşağıya sırası şeklinde verilmiştir. IPv4 paket yapısında bulunan alanlar sırasıyla şunlardır;

Sürüm (Version)

İnternet protokolünün sürümünü belirtir. 4 bit'lik alana sahiptir. Yönlendiriciler bu alana bakarak IP paketinin kalan alanlarını nasıl çevireceğini belirler. Varsayılan değeri 4'tür [16].

Çizelge 2.1. IPv4 paket yapısı [16]

<table>
<tr><th colspan="5">32-bit</th></tr>
<tr><th>4-bit</th><th>4-bit</th><th>8-bit</th><th>3-bit</th><th>13-bit</th></tr>
<tr><th>Sürüm</th><th>Başlık Uzunluğu</th><th>Servis (Hizmet) Türü</th><th colspan="2">Paket Boyutu</th></tr>
<tr><th colspan="3">16-bit Tanımlayıcı</th><th>Bayraklar</th><th>13-bit Bölümlendirme Katsayısı</th></tr>
<tr><th colspan="2">Yaşam Süresi</th><th>Üst Katman Protokolü</th><th colspan="2">Başlık Kontrolü</th></tr>
<tr><th colspan="5">32-bit Kaynak IP Adresi</th></tr>
<tr><th colspan="5">32-bit Hedef IP Adresi</th></tr>
<tr><th colspan="5">Seçenekler (eğer varsa)</th></tr>
<tr><th colspan="5">Veri</th></tr>
</table>

Başlık uzunluğu (Header length)

IPv4 paket yapısı birden fazla seçenek içerebildiğinden 4 bit'lik olan başlık IP paketinin nereden başlaması gerektiğini belirler. IP paketlerinin çoğu seçenek içermezler bu yüzden tipik IP paketi başlığı 20 byte'a sahiptir [16].

Servis-hizmet türü (Type of service - TOS)

IPv4 başlığında bulunan servis türü farklı IP paketlerinin birbirinden ayırt edilmesine olanak sağlar. Örneğin, gerçek zamanlı paketleri (IP telefon uygulaması gibi) gerçek zamanlı olmayan paket trafiğinden (FTP uygulaması gibi) ayırmak için kullanılabilir. 8 bit'lik bir yapıya sahiptir [16].

Paket uzunluğu-boyutu (Datagram length)

IPv4 paketinin (başlık ve veri) toplam uzunluğudur. 16 bit'lik yapıya sahip olduğundan teorik olarak IP paketinin maksimum boyutu olan 65 535 byte (64 KB) olabilir. Genelde paketlerin boyutu 1 500 byte'dan büyük olur. Fakat fiziksel bağlantılardaki sınırlamalar sebebiyle paket uzunluğu 64KB değerine ulaşamaz [16].

Tanımlayıcı (Identifier), Bayrak (Flag), Bölümlendirme katsayısı (Fragment offset)

Bu üç alan IP bölümlendirmesi olarak adlandırılır. IP veri katmanında bulunan eternet üzerinden çalıştığından paket veri boyutu maksimum 1500 Byte olabilir. Bu paketin veri boyutu 1 500 Byte üzerine çıkarsa eternet üzerinden bu verinin gönderilmesi zorlaşır ve paket bölümlendirilir [16].

Yaşam süresi (Time to live - TTL)

8 bit'lik uzunluğa sahip olan TTL, bir gönderici paketinin hedefine ulaşıncaya kadar azalarak geçen süreye denir. Normal şartlarda TTL değeri hedef noktasına ulaşıncaya kadar her geçtiği bilgisayarda bir azalarak devam eder. Eğer TTL değeri paket hedefine ulaşmadan önce sıfır (0) olursa, gönderilen paket göz ardı edilir ve bir zaman aşımı paket hatası ICMP (Internet Control Message Protocol) aracılığıyla gönderici pakete gönderilir [16,17].

Üst katman protokolü (Upper layer protocol)

Veri paketinin hangi üst katman protokolünden alındığını veya hangi üst katman protokolüne gönderileceğini belirleyen 8 bit'lik IP başlık alanıdır. OSI modelinin 3.katmanı olan ağ (Network) katmanında bulunan IP, OSI modelinin 4.katmanı olan taşıma (Transport) katmanından veriyi TCP, UDP, ICMP veya diğer taşıma protokollerinden alır. TCP için 6, UDP için 17 ve ICMP için ise 1 değerini alır. IP paket başlığında üst katman protokolü alanı kullanmanın amacı alt katman veya üst katman alıcı ve gönderici paketlerinin geçiş zorunluluğu bulunan protokolden geçmesini sağlamaktır [16].

Başlık kontrolü (Header checksum)

IPv4 başlığının 16 bit'lik kısmıdır. Bu kısım veri paketinin geçişinde oluşabilecek kopmalara karşı basit bir koruma sağlar. Paketin geçeceği yol boyunca yönlendiricilere paket başlığının bozulup bozulmadığının kontrolünü yapmasını sağlar. Yönlendiriciler hata tespit edilen paketlerinin geçişine izin vermezler. Hata kontrolünün yapılmasının amacı taşıma katmanı protokollerinin her zaman aynı veri paketlerinin taşınmasını engellemektir [16].

Kaynak ve hedef IP adresleri (Source and destination IP addresses)

Kaynak ve Hedef adresleri 32 bit'lik yapıda olan IPv4 adresleridir. Kaynak adres tarafından bir paket oluşturduğunda kaynak adres bu paketin kaynak IP adres alanına kendi adresini, hedef adres alanına ise paketi göndereceği hedef adresi yerleştirir [16].

Seçenekler (Options)

İsteğe bağlı olan seçenekler alanı bir IP başlığının genişletilebilmesine olanak sağlar. IPv4 başlığında yer alan bu kısım nadiren kullanılır. Bu yüzden çoğunlukla seçenekler alanı boştur. Boyutu 32 bit (4 Byte)'tir. Seçenekler alanı boş yani sıfır byte olursa IPv4 başlığı 160 bit = 20 Byte olur. Ayrıca, 160 bit = 20 byte TCP başlığı vardır. Bir IPv4 paketi toplam 320 bit = 40 byte başlık içerir [16].

Veri (Data)

IPv4 paketinin en son ve önemli alanıdır. Çoğu durumda veri alanı hedef adrese ulaşılması gereken taşıma katmanı bölümlerini (TCP veya UDP) içerir. Veri alanı sadece taşıma katmanı bölümlerini içermez. Diğer veri türlerini, örneğin, ICMP iletilerini, de barındırabilir [16].

2.1.2. IPv4 adresleme yapısı

IPv4 için her TCP/IP bağlantısı olan bilgisayar / ağ aygıtları mantıksal bir IP adresi tarafından tanımlanır. IP adresi bir ağ katmanı adresidir. Veri katmanı adresleri, örneğin MAC (Media Access Control) adresi, üzerinde bir bağlayıcı yönü yoktur. TCP/IP kullanarak iletişim sağlayan her bilgisayar ve ağ bileşeni için mutlaka bir IP adresi gereklidir. Bu adres statik veya dinamik olarak atanabilir. Her IP adresi üzerinde;

- Ağ kimliği (Network ID)
- Ağ aygıtı kimliği (Host ID)

barındırır.

Ağ kimliği (ağ adresi olarak da bilinir), yönlendiriciler tarafından sınırlandırılan aynı fiziksel ağ üzerinde bulunan sistemleri belirler. Aynı fiziksel ağ üzerinde bulunan bütün sistemlerin aynı ağ kimliğine sahip olma zorunluluğu vardır. Ağ kimliği belirtilen ağa özgü ve benzersiz olmalıdır.

Ağ aygıtı kimliği (ağ aygıtı adresi olarak da bilinir) bir ağda bulunan sunucu, yönlendirici ve diğer TCP/IP aygıtlarını tanımlar. Ağ aygıtı kimliği bulunduğu ağa özgü olmak zorundadır.

IP adresi ağları birbirinden ayırmak için hiyerarşik bir yapıya sahiptir. IP adresinin uzunluğu 32-bit yani 4 byte'dır. IPv4 adres sayısı 2^{32} = ~ 4 294 967 296 (yaklaşık 4 milyar)'dır. Bu adreslerden bazıları özel ağlar (~ 18 milyon adres) ve bazıları da çoklu gönderim adresleri (~ 270 milyon adres) gibi özel amaçlar için ayrılmıştır.

IPv4 küresel (global) bir adresleme tekniğine sahiptir. Bu adresleme sayesinde binlerce ağ, on binlerce ağ aygıtı ve milyonlarca bilgisayara adres vermek olanaklı hale gelmiştir [16]. Mevcut IPv4 adreslerini 32 bit uzunluğunda ikilik sisteme göre yazmak ve akılda tutmak çok zor olduğu için 32 bit'lik adresi 4 adet 8 bitlik (oktet) alanlar şeklinde yazılır. Bu oktetlerin her biri onluk sisteme çevrilir. Elde edilen değer 0-255 arasında bir sayıdır. Bu sayılar nokta (.) ile birbirinden ayrılarak yazılır. Bu yazım şekline noktalı onluk gösterimi denir. Çizelge 2.2'de IPv4 Yazım Formatı gösterilmiştir.

Çizelge 2.2. IPv4 yazım biçimi [16]

İkili Gösterimi (Binary Format)	**Noktalı Onluk Gösterimi (Dotted Decimal Notation)**
11000000 10101000 00000011 00011000	192.168.3.24

Çizelge 2.2'de gösterilen IPv4 adresinin ikilik sistemdeki gösterimi 11000000101010000000001100011000 şeklindedir. Bu adresin akılda tutulması zordur. Bu yüzden bahsedilen IPv4 adresinin gösterimi;

- Adres 8-bit'lik bloklar şeklinde yazılır. (**11000000 10101000 00000011 00011000** gibi)
- Her blok onluk sisteme çevrilir. (**192 168 3 24** gibi)
- Daha sonra bloklar (oktetler) arasına nokta (.) konulur. (**192.168.3.24** gibi)

aşamalarından oluşur.

Bir IPv4 adresinin bir kısmı ağ kimliğini, diğer kısmı ise ağda bulunan bilgisayarı veya ağ cihazını belirtir. Bu belirlemeyi sağlayan yapıya alt ağ (subnet) adı verilir. Alt ağ birbirine bağlı ağ cihazlarının mantıksal olarak gruplanmasıdır. Alt ağ, ağ yapılandırmasına göre ayrılmış aygıtların ağ trafik akışına izin verir. Bu da ağ güvenliği ve performansı sağlar. Alt ağ yapısının en fazla üzerinde durulan kavramı alt ağ maskesi (subnet mask)'dir. IPv4 adresleri gibi bir alt ağ maskesi de 4 byte (32 bit) içerir ve IPv4 adresi gibi yazılır. Bir subnet adresine 255.255.255.0 örnek olarak gösterilebilir.

2.1.3. IPv4 adres türleri ve sınıfları

IPv4 32 bit adres yapısını kullanır. IPv4 adres sayısı 2^{32} = ~ 4 294 967 296 (yaklaşık 4 milyar)'dır. Bu adreslerden bazıları özel ağlar (~ 18 milyon adres) ve bazıları da çoklu gönderim adresleri (~ 270 milyon adres) gibi özel amaçlar için ayrılmıştır. Bu sayılar göz önüne alındığında yaklaşık 288 milyon IPv4 adresi önceden harcanmıştır. Bunun için ağ adresleme mimarisinin sınıflı ağ (Classful Network) tasarımı aracılığıyla yeniden düzenlenmiştir [17-21]. Bu ağ sınıfları aşağıda detaylı bir şekilde ele alınmıştır.

İnternet standartları (STD) tarafından IPv4 adres türleri [18,20];

- Tekli Gönderim - Tek Alıcılı (**Unicast**) Adresleri
- Çoklu Gönderim - Çok Alıcılı (**Multicast**) Adresleri
- Yayın (**Broadcast**) Adresleri

şeklinde tanımlanıyor.

Tekli gönderim adresleri

Özel bir alt ağ (subnet) üzerinde bulunan bir ağ ara yüzü için tanımlanır. Bire bir iletişimler için kullanılır.

Çoklu gönderim adresleri

Çeşitli alt ağlar üzerinde bulunan bir veya birden fazla ağ ara yüzleri için tanımlanır. Bir ara yüz ile birçok ara yüz arasında iletişim sağlamak için kullanılır.

Yayın adresleri

Bir alt ağ üzerinde bulunan bütün ağ ara yüzleri için tanımlanır. Bir ara yüz ile bir alt ağ üzerinde bulunan tüm ara yüzlerle iletişim sağlamak için kullanılır [18,20].

IPv4 beş adres sınıf sistemine sahiptir. Bu sınıflar A, B, C, D ve E olarak isimlendirilmiştir. Sadece A, B ve C adres sınıfları internete erişim için gerçek IPv4

adreslemeyi sağlar. D adres sınıfı çoklu gönderim (multicast) için kullanılır. E adres sınıfı ise deneysel çalışmalar için ayrılmıştır [17-21]. Çizelge 2.3'te bu sınıfların adres yapıları verilmiştir.

Çizelge 2.3. IPv4 adres ve sınıf yapısı [20]

A Sınıfı	**Ağ Kimliği (Network ID)**	Ağ Aygıt Kimliği (Host ID)		
Oktet	**1**	2	3	4
B Sınıfı	**Ağ Kimliği (Network ID)**		Ağ Aygıt Kimliği (Host ID)	
Oktet	**1**	**2**	3	4
C Sınıfı	**Ağ Kimliği (Network ID)**			Ağ Aygıt Kimliği (Host ID)
Oktet	**1**	**2**	**3**	4
D Sınıfı	Ağ Aygıt Kimliği (Host ID)			
Oktet	1	2	3	4

A sınıfı adresler

İlk oktet dediğimiz 8 bitlik kısımda bulunan bitlerden ilk bit 0 (sıfır) ise bu adres sınıfı A sınıfıdır. Bu kısım ağ kimliğini belirtir. Geriye kalan 24 bit ağ aygıtlarının (bilgisayar, yönlendirici, anahtar v.b.) adreslerini belirler. İlk 8 bitlik kısımda yer alan en küçük sayı ikilik sistemde 00000000, onluk sistemde ise 0'dır. En büyük sayı ikilik sistemde 01111111, onluk sistemde ise 127'dir. İlk okteti 0 ve 127 arasında bir sayı başlayan IPv4 adresleri A sınıfı adreslerdir. Fakat 0 ve 127 değerleri farklı amaçlar için ayrılmıştır. Bu sayılar ağ adresi olarak kullanılamazlar. Her ağ mutlaka bir ağ numarası veya kimliğine sahip olmalıdır. A sınıfında olası 128 ağ mevcuttur. Bu ağlardan 2 tanesi özel amaçlar için ayrılmıştır. Geriye kalan 128 – 2 = 126 olası

ağ ile adresleme yapılabilir. A sınıfında bulunan 0.0.0.0 ve 127.0.0.1 adresleri özel amaçlar için kullanılmıştır. 0.0.0.0 adresi bir ağda bütün amaçlara hizmet verecek varsayılan yol adresi olarak kullanılır. 127.0.0.1 adresi ise bir ağda bulunan bütün ağ aygıtlarının kendi üzerlerinden ağ trafiğini geçirmek için kullanılır. İlk 8 bitlik (1 Byte) kısımda yer alan değerler ağ numarası / kimliği için ayrıldığından geriye kalan 24 bit (3 Byte) yani 2^{24} = 16 777 216 adres ağ aygıtları (bilgisayar, yönlendirici) için kullanılır. Fakat bu adreslerden 2 tanesi özel amaçlar (ağ numarası / kimliği ve yayın) için kullanıldığından olası 126 adet ağın her birisi için 16 777 216 – 2 = 16 777 214 adres kullanılabilir durumdadır. A sınıfında toplam 16 777 214 adet bilgisayar veya ağ aygıtı adreslenebilir. A sınıfı adresler çok fazla IPv4 adresine sahip olduklarından çok büyük ağlarda kullanılmaktadırlar [20].

B sınıfı adresler

İlk oktet denilen 8 bitlik kısmın ilk 2 bitlik alanı ikilik sistemde 10 ile başlayan adresler B sınıfı adreslerdir. B sınıfı adreslerin en küçük değeri ikilik sistemde 10000000 onluk sistemde 128, en büyük değeri ise ikilik sistemde 10111111 onluk sistemde 191'dir. İlk okteti 128 – 191 değerleri arasında bir sayı ile başlayan her hangi bir IPv4 adresi B sınıfı adresidir. Orta büyüklükteki ağlar için kullanılır. İlk iki oktet alanı yani bir IPv4'ün 16 bitlik (2 Byte) kısmı B sınıfı adresler için ağ numarası/kimliği olarak kullanılır. Geriye kalan 16 bitlik (2 Byte) kısım ise ağ aygıtlarını adreslemek için kullanılır. B sınıfının ilk 2 bit her zaman ikilik sistemde 10 olduğundan ağ numarası / kimliği bitlerinden geriye kalan 14 bit ile 2^{14} = 16,384 B sınıfı ağ elde edilir. Bu B sınıfı ağlardan her biri 2^{16} = 65 536 tane adrese sahiptir. Bu adreslerden 2 tanesi özel amaçlar (ağ numarası / kimliği ve yayın) için kullanıldığından 16,384 ağlarının her biri için toplam 65,536 – 2 = 65 534 tane IPv4 adresi kullanılabilir durumdadır [20].

C sınıfı adresler

İlk oktet denilen 8 bitlik (1 Byte) kısmın ilk 3 bitlik alanı ikilik sistemde 110 ile başlayan adresler C sınıfı adreslerdir. C sınıfı adreslerin en küçük değeri ikilik sistemde 11000000 onluk sistemde 192, en büyük değeri ise ikilik sistemde

11011111 onluk sistemde 223'dür. İlk okteti 192 – 223 değerleri arasında bir sayı ile başlayan her hangi bir IPv4 adresi C sınıfı adresidir. Küçük ölçekli ağlar için kullanılır. İlk üç oktet alanı yani bir IPv4'ün 24 bitlik (3 Byte) kısmı C sınıfı adresler için ağ numarası / kimliği olarak kullanılır. Geriye kalan 8 bitlik (1 Byte) kısım ise ağ aygıtlarını adreslemek için kullanılır. C sınıfının ilk 3 bit her zaman ikilik sistemde 110 olduğundan ağ numarası / kimliği bitlerinden geriye kalan 21 bit ile 2^{21} = 2 097 152 adet C sınıfı ağ elde edilir. Bu C sınıfı ağlardan her biri 2^8 = 256 tane adrese sahiptir. Bu adreslerden 2 tanesi özel amaçlar (ağ numarası / kimliği ve yayın) için kullanıldığından 2 097 152 ağlarının her biri için toplam 256 – 2 = 254 tane IPv4 adresi kullanılabilir durumdadır [20]. Çizelge 2.4'te A,B ve C sınıflarının ağ sayıları ve ağ aygıt sayıları bilgileri verilmiştir.

Çizelge 2.4. IPv4 sınıfları için ağ ve ağ aygıtları sayısı [16,20]

Adres Sınıfı	Ağ Kimliği Değer Aralığı	Muhtemel Ağ Sayısı	Her Ağ için Ağ Aygıt Sayısı
A Sınıfı	1 – 126	128 (2^7) (2 adet ayrılmış) 126	16 777 214
B Sınıfı	128.1 – 191.254	16 384	65 534
C Sınıfı	192.0.1 – 223.255.254	2 097 152	254

D sınıfı adresler

İlk oktet denilen 8 bitlik (1 Byte) kısmın ilk 4 bitlik alanı ikilik sistemde 1110 ile başlayan adresler D sınıfı adreslerdir. D sınıfı adreslerin en küçük değeri ikilik sistemde 11100000 onluk sistemde 224, en büyük değeri ise ikilik sistemde 11101111 onluk sistemde 239'dur. İlk okteti 224 – 239 değerleri arasında bir sayı ile başlayan her hangi bir IPv4 adresi D sınıfı adresidir. D sınıfı adresler kişisel ağ aygıtlarını adreslemek için kullanılmaz. Bunun yerine sadece ağ aygıt gruplarını veya çoklu gönderim gruplarını temsil etmek amacıyla kullanılır. Örneğin, EIGRP (Enhanced Interior Gateway Routing Protocol) protokolünü kullanan bir yönlendirici aynı protokolü (EIGRP) kullanan düğümler içeren bir grupla birbirine bağlanır. Bu grubun üyeleri kendilerine özgü A, B ve C sınıfı adreslerinden IPv4 adreslerine sahip olmakla birlikte D sınıfı adresi olan 224.0.0.10 adresine gelen mesajları da dinlerler.

Böylece, tek bir yönlendirme güncelleme mesajını 224.0.0.10 adresi ve bütün EIGRP yönlendiricileri alırlar. Çoklu gönderim adreslerinden seçtiklerimize de bu mesaj yollanır. Bu yüzden D sınıfı adresler çoklu gönderim adresleri olarak da adlandırılır. Çoklu gönderim (multicast) kavramı yayın (broadcast) kavramından farklıdır. Mantıksal bir ağ üzerinde bulunan her cihaz bir yayın yapmak zorundadır. Fakat sadece bir D sınıfı adresini dinlemek için ayarlanmış cihazlar / aygıtlar birçoklu gönderim mesajı alırlar [20].

E sınıfı adresler

İlk oktet denilen 8 bitlik (1 Byte) kısmın ilk 4 bitlik alanı ikilik sistemde 1111 ile başlayan adresler E sınıfı adreslerdir. E sınıfı adreslerin en küçük değeri ikilik sistemde 11110000 onluk sistemde 240, en büyük değeri ise ikilik sistemde 11111111 onluk sistemde 255'tir. E sınıfı adresler deneysel amaçlar için kullanılır. E sınıfı adresler kişisel ağ aygıtlarını veya çoklu gönderim gruplarını adreslemek için kullanılmaz [20].

Özel IPv4 adresleri

IPv4 adres sınıflarından farklı olarak özel olarak tanımlanmış adresler vardır. Bu adresler şunlardır;

Ağ numarası adresleri

Ağ ortamını tanımlamak ve yönlendiriciler tarafından yol belirleme tablolarını oluşturmak için kullanılan ve ağ aygıtları için ayrılan kısımların 0 (sıfır) olduğu adres türleridir. Örneğin, A sınıfı adreslerinden olan 10.0.0.0, B sınıfı adreslerinden olan 128.128.0.0 ve C sınıfı adreslerinden olan 192.168.1.0 ağ numarası olarak kullanılır [20,21].

Yayın adresleri

Tanımlı bir ağda ağ aygıtlarını (bilgisayar, yönlendirici v.b.) belirten kısımların 255 olduğu adres türleridir. Tanımlanan ağdaki her bilgisayarı bir seferde belirlemek ve

gönderilmesi gereken paketleri bütün kullanıcılara iletmek amacıyla kullanılır. Örneğin, A sınıfı adreslerinden olan 46.255.255.255, B sınıfı adreslerinden olan 172.16.255.255 ve C sınıfı adreslerinden olan 192.168.2.255 adresleri yayın adresleri (broadcast) olarak kullanılır [20,21].

Varsayılan yol adresi

Bir ağda bütün amaçlar için kullanılan adres türüdür. Bütün byte değerleri sıfırdır. 0.0.0.0 adresi olarak bilinir [20,21].

Geri dönüş devre (loopback) adresi

Bir ağda bulunan bütün ağ aygıtlarının ağ trafiğini kendi üzerlerinden geçirmelerini sağlayan adres türüdür. Bir aygıt üzerinden çalışan TCP/IP uygulamaları ve hizmetleri bu adres türünü kullanarak kısa yoldan birbirleriyle iletişim sağlarlar. Ayrıca bu adres türüne PING (Packet INternet Gropper) göndererek yerel aygıt üzerindeki TCP/IP yapılandırmasını test edebiliriz. Her ağ aygıtının kendisini belirten adrestir. Bu adrese gönderilen paketler ağa çıkartılmaz. Bu adres 127.0.0.1 adresidir [20,21].

Evrensel bütünsel-yayın adresi

Bir IPv4 adresinin evrensel bütünsel-yayın olarak kullanılan adresidir. Bütün ağlar ve ağ aygıtlarını tek seferde adreslemek için kullanılır [20,21].

Çizelge 2.5'de özel adresler ve bu adreslere ait adres aralıkları verilmiştir.

Çizelge 2.5. IPv4 özel adresler ve adres aralıkları [20,21]

Adres Türleri	**Kullanım Amacı**	**Adres**
Varsayılan Yol	IPv4 ağ aygıtlarını tanımlamak için kullanılır.	0.0.0.0
Geri Dönüş Devre Adresi	Her ağ aygıtının kendisini belirten adresler olduklarından sadece yerel testler için kullanılır.	127.0.0.1
Evrensel Bütünsel -Yayın Adresi	Tüm ağ ve ağ aygıtlarını bir seferde adreslemek için kullanılır.	255.255.255.255
Çoklu Gönderim Adresleri	Bir yerel ağ üzerinde çoklu gönderim grupları için kullanılır.	224.0.0.0 – 239.255.255.255
Deneysel veya Araştırma Adresleri	Deney veya araştırma adresleridir. Ağ aygıtlarını adreslemek için kullanılmaz.	240.0.0.0 – 255.255.255.254

2.1.4. IPv4 alt ağları

IETF tarafından 1993 yılında mevcut Sınıflı Ağ mimarisinin yerini alacak Sınıfsız Alanlar Arası Yönlendirme (Classless Inter-Domain Routing (CIDR)) ağ adresleme mimarisi duyurulmuştur [22]. Amaçları, internet üzerinden yönlendiricilerde barınan yönlendirme tablolarının büyümesini engellemek ve IPv4 adreslerinin hızla tükenen oranını düşürmektir [23]. CIDR, IPv4'ün ölçeklenebilirlik ve verimliliğini artırmak için;

- Sınıflı adreslemeyi daha esnek ve daha az kayıplı sınıfsız adres şemasıyla değiştirmek,
- Gelişmiş yol/rota barındırmak (özetleme olarak da bilinir) v
- Alt ağ maskesi tarafından tanımlanan yeni bir adresin bitişik ağ adresleriyle birleştirilmesi,

işlevlerini sağlar [20,23].

Sınıflı ağlarda bulunan adreslerin dağılımı yeteri oranda esnek değildir. Aynı ağda bulunan tüm ağ aygıtları aynı ağ adresine sahip olmak zorundadır. Ağlar büyüdükçe ağ yöneticilerinin bu adresleri yönetmeleri sorun teşkil etmeye başlamıştır. Alt ağ tanımlamaları yapmak gerektiği savunulmuştur. Alt ağlar daha az IPv4 adresinin

tüketimi ve daha iyi mantıksal ağ yapılandırma olanaklarını sağlar. Her ağ aygıtına IPv4 adresinin yanında alt ağ maskesi adresinin de tanımlanması gerekir. Böylece aynı ağ adresine sahip bütün aygıtlar birbirleriyle iletişim kurabilir. Sınıflı ağlarda varsayılan ağ maskeleri vardır [22,23]. Çizelge 2.6'da bu değerler verilmiştir.

Çizelge 2.6. IPv4 sınıfları ağ adresleri ve maskeleri [20]

Adres Sınıfı	**Ağ Adresi**	**Varsayılan Ağ Maskesi**
A	**N**.0.0.0	255.0.0.0
B	**N.N**.0.0	255.255.0.0
C	**N.N.N**.0	255.255.255.0

N: Ağ numarası/adresini ifade eder.

Bir ağda mevcut ağ, alt ağ ve ağ aygıtı adres alanları / kısımları ağ maskesine bakılarak belirlenebilir. Bu alanlar ağ maskesinin boyutunu belirler. Bir ağ maskesi IPv4 adresi gibi 32-bit'lik yapıya sahiptir. Ağ maskesinin her bir bit değeri IPv4 adresindeki değer ile mantıksal VE işlemine tabi tutularak bir IPv4 adresi elde edilir. Elde edilen bu IPv4 adresinin değeri 1 olan her bit ağ adresini belirtir. 0 olan değerler ise ağ aygıtını ifade eder. Örneğin, IPv4 adresi 172.24.100.45 olan bir adres ile alt ağ maskesi 255.255.255.0 olan bir adresin bulunduğu ağ adresi/numarası adımları;

1. IPv4 Adresi: 172.24.100.45 → **10101100 00011000 01100100 00101101**
2. Ağ Maskesi: 255.255.255.0 → **11111111 11111111 11111111 00000000**
3. VE(**AND**) İşlemi: 172.24.100.0 → **10101100 00011000 01100100 00000000**

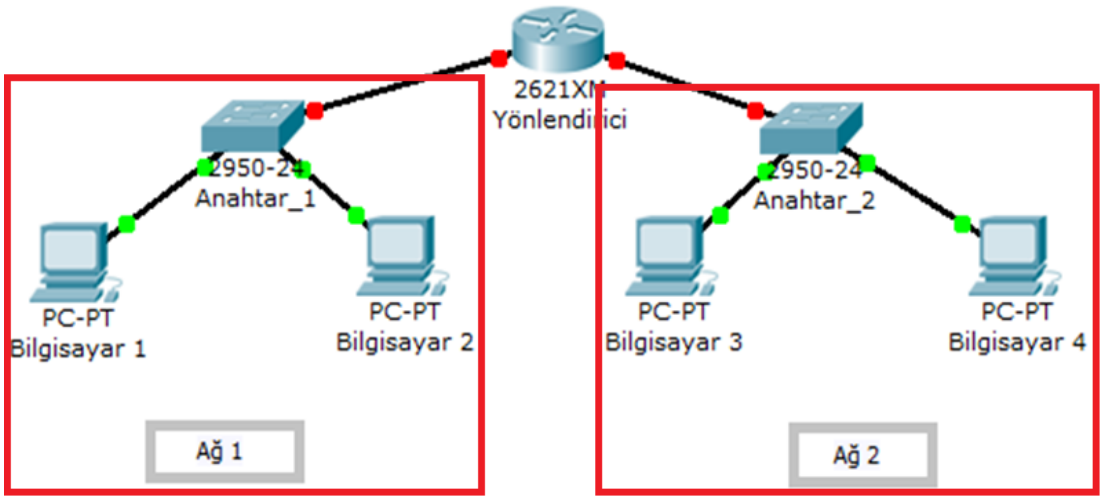
şeklindedir.

3 nolu adımda elde edilen sonuç verilen adresin ait olduğu ağ adresini/numarasını belirtir. Bu ağın adresi *172.24.100.0*'dır.

2.1.5. IPv4 yönlendirme ve protokolleri

Şekil 2.1'de verildiği gibi yönlendirme, birbirine bağlı ağlar arasında paketlerin haberleşmesidir. Başka bir ifadeyle birbirine bağlı ağlardaki bilgisayar veya ağ

aygıtlarının yönlendiriciler aracılığıyla iletişim kurma işlemine IP yönlendirme adı verilir. Şekil 2.1'de Ağ-1 üzerinde bulunan Bilgisayar 1 ve/veya Bilgisayar 2, Ağ-2 üzerinde bulunan Bilgisayar 3 ve/veya Bilgisayar 4'e erişmek istediğinde ARP (Address Resolution Protocol) aracılığıyla Bilgisayar 3 ve/veya Bilgisayar 4'ün IP adresini bulmaya çalışır. Eğer Bilgisayar 3/4 ile Bilgisayar 1/2 aynı ağ üzerinde ise iletişim doğrudan sağlanır. Fakat farklı ağlarda iseler iletişimin sağlanabilmesi için bir yönlendiriciye ihtiyaç vardır. Şekil 2.1'de 26221XM modeline sahip yönlendirici verilmiştir. Bu yönlendirici Ağ-1 ve Ağ-2 üzerinde bulunan bütün bilgisayar ve ağ aygıtlarının adreslerini yönlendirme tablosunda bulundurur ve tümünün IP adreslerini bilir. İnternet gibi büyük ağ yapıları segment denilen küçük parçalara ayrılarak daha etkin yönetilmesi sağlanır. Bu ağları yönlendirecek mekanizmalara ihtiyaç vardır. Bu mekanizmalara yönlendirici (router) adı verilir. Yönlendirici, farklı ağları birbirine bağlayan ve özel olarak yönlendirme görevini yerine getirmek için hazırlanan cihaz veya bilgisayarlara denir. Yönlendiriciler sadece ağlardan sorumludur ve en iyi yol hesaplamaları yaparak IP paketlerini hedef ağlara ulaştırırlar.



Şekil 2.1. IPv4 yönlendirme

Yönlendiriciler IP paketlerini bir ağdan başka bir ağa geçirirken yönlendirme işlemini iki şekilde yaparlar. Bu yönlendirme türleri statik ve dinamik yönlendirme olarak adlandırılır. Bu iki yönlendirme türü aşağıdaki başlıklarda açıklanmıştır.

Statik yönlendirme

Bu yönlendirme türünde yönlendiricilerin yönlendirme tablosundaki bilgiler elle verilir. Ağ yöneticisi tarafından hedef ağ bilgileri girilir ve ağdaki gerekli bütün güncellemeler elle yapılır. Statik yönlendirmenin bazı avantajları ve dezavantajları vardır [20,23]. Bunların

Avantajları

- Yönlendirici işlemcisinin kullanılmasına gerek kalmaz.
- Veriler elle girildiğinden diğer yönlendiricilerle ilgilenmez. Hedef yönlendirici ile doğrudan iletişim kurulur.
- Ağ yöneticisi tarafından bilgiler elle verildiğinden daha güvenlidir.

Dezavantajları

- Sisteme yeni bir ağ eklenmesi elle yapılmalıdır. Bu durum büyük ağlar için düşünüldüğünde içinden çıkılmaz bir hale gelir.
- Elle girilen yönlendirme bilgileri için yedekleme yapılmaz.

Dinamik yönlendirme

Statik yönlendirmedeki sorun elle verilen bilgiler doğrultusunda ağın büyümesidir. Ağın büyümesi belli bir zaman sonra büyük sıkıntılar oluşturur. Büyük ağlarda yönetim daha zor olduğundan ağ trafiğinin yönlendirilmesinin daha sağlıklı ve etkin bir şekilde gerçekleşmesi gerekir. Dinamik yönlendirme protokolleri, yönlendirme tablosundaki verileri otomatik olarak oluşturur. Böylece dinamik yönlendirme protokolü tarafından ağdaki bilgisayar veya ağ aygıtlarının adresleri otomatik olarak işlenir. Böylece ağ yöneticisinin elle müdahalesi gerekmeksizin en uygun yol bulunur. Bir çıkış noktasında meydana gelen bir sorunda tüm trafik otomatik olarak diğer çıkış noktalarına yönlendirilir.

Dinamik Yönlendirme Protokolleri üç başlık altında incelenir. Bu yönlendirme protokolleri;

- Uzaklık vektör yönlendirme protokolleri (Distance vector routing protocols)
- Bağlantı durum yönlendirme protokolleri (Link state routing protocols)
- Dengeli melez yönlendirme protokolleri (Balanced hybrid routing protocols)

şeklinde tanımlanır [20-23]. Bu protokoller aşağıdaki alt başlıklarda kısaca açıklanmıştır.

Uzaklık vektör yönlendirme protokolleri

Yönlendirme tablolarında belirli zaman aralıklarında ağ bilgilerini komşu yönlendiricilere göndererek yönlendirme tablolarını komşu yönlendiriciden gelen bilgiyle güncellerler. Böylece bu sorguların sonucunda her yönlendirici sistemdeki tüm ağ adreslerini öğrenmiş ve en uygun yol seçimini yapmış olur. Bu yönlendirme protokollerine örnek olarak RIP (Routing Information Protocol) ve IGRP (Interior Gateway Routing Protocol) verilebilir [20-23].

Bağlantı durum yönlendirme protokolleri

Sürekli bir güncelleme yapmak yerine yönlendiricilerin çalışır durumda olup olmadığını anlamak için her 10 saniyede çok küçük “Hello” paketlerini gönderirler. Böylece var olan sisteme yeni bir yönlendirici eklendiğinde veya sistemde herhangi bir yönlendirici çalışmaz duruma gelirse sadece bu yönlendiriciye ait bilgi yönlendirme tablosunda güncellenir. Bu yönlendirme protokollerine örnek olarak OSPF (Open Shortest Path First) verilebilir [20-23].

Dengeli melez yönlendirme protokolleri

Hem uzaklık vektör yönlendirme hem de bağlantı durum yönlendirme protokollerinin özelliklerini taşırlar. Bu yönlendirme protokollerine örnek olarak CISCO tarafından geliştirilen ve sadece CISCO yönlendiricilerde çalışan EIGRP (Enhanced Interior Gateway Routing Protocol) verilebilir [20-23].

2.2. IPv6 (Internet Protocol version 6)

Çalışmanın giriş bölümünde de bahsedildiği gibi ağ protokollerinin geçmişine bakıldığında en başarılı ağ protokolünün IP olduğu ortaya çıkmaktadır. İnternet protokolünün diğer protokollerden daha başarılı olması, IPv4'ün artık ihtiyaçlara cevap verememesi, internet kullanıcısının hızla artması yeni nesil internet protokolünün geliştirilmesine yön vermiştir. Yeni nesil tasarım için 1992 yılı Temmuz ayında IETF IPv6 için öneride bulunmuştur. IANA tarafından yeni nesil IP (IPng) adına version 6 (v6) verilmiştir. 1995 yılı Ocak ayında IETF IPv6 yapısı için önerilerini tamamlamış ve RFC 1752 olarak yayınlamıştır IPv6 hakkında bilgiler yapılan çalışmalar ışığında aşağıdaki başlıklarda verilmiştir.

2.2.1. Tarihçesi

1970'li yıllarda geliştirilen ve halen kullanılan IPv4 protokolünün yerini alması için 1990'lı yılların başından itibaren IETF (Internet Engineering Task Force) tarafından yeni nesil internet protokolü (IPng) geliştirmek için ilk çalışmalar başlatılmıştır. IP'nin kullanışlı bir protokol olması ve mevcut adreslerin hızla tükenmesi nedeniyle IETF Temmuz 1992'de yeni nesil IP için önerilerde bulunmuştur. 17 Kasım 1994'te IETF Toronto toplantısında RFC 1752 belgesinde belirtilen IPv6 taslağı oluşturulmuştur. IPv4'ün yerini alması öngörülen IPv6 "Internet Protocol Version 6" kelimesinin baş harflerinin kısaltılmış halidir. IETF tarafından Aralık 1998'de RFC 2460 internet standardı belgesi adında yayımlanmıştır [5,6,24,26].

IPv6'nın en belirgin özelliği daha fazla adres sunmasıdır. Günümüzde dünya ülkeleri başta Çin, Japonya olmak üzere hızla IPv6'ya geçiş süreçleri ve çalışmaları başlatmıştır. IPv6 protokolünü ilk kez pratiğe geçiren ülke 1998 yılında Çin olmuştur. Çin 8-24 Ağustos 2008 tarihlerinde 4G'li IPv6 teknolojisini kullanmıştır. Japonya Eylül 2000'de, Güney Kore Şubat 2001'de, Avrupa Komisyonu Nisan 2001'de IPv6'yı benimsediklerini duyurdular. Amerikan hükümeti Haziran 2008 itibariyle tüm devlet ağlarının IPv6 olması yönünde talimat vermiştir. IPv6'ya geçişin 2025 yılında tamamlanması gerektiği belirtilmiştir [7]. Türkiye ise 15 Şubat 2009 tarihinde TÜBİTAK tarafından desteklenen "Ulusal IPv6 Protokol Altyapısı

Tasarımı ve Geçişi Projesi" kapsamında IPv6'ya geçiş için çalışmalar başlatmıştır [8].

2.2.2. IPv6'nın gerekçeleri

IPv4 adres 32 bittir. 2^{32} = 4 294 967 296 adet adreslemeye imkân tanır. Fakat verimsiz adres atama mekanizmalarından dolayı etkin adres sayısı hiçbir zaman bu sayıya ulaşamaz. Çalışmanın giriş bölümünde bahsedildiği sadece dünya nüfusu göz önünde alındığında kişi başına bir IPv4 adresi bile düşmemektedir. Dünya nüfusuna bağlı olarak internetin, teknolojik alt yapıların hızla büyümesi neticesinde kullanılan IPv4 temelli ağların yetersiz kalması, IPv4 adreslerinin tükeneceği ve neredeyse elde IP'nin kalmaması büyük bir sorun teşkil etmektedir. Uluslararası organizasyon IETF'nin çalışmaları 2008 ile 2018 yılları arasında mevcut IPv4 adreslerin tamamen tükeneceği yönünde veriler sunmuştur [24,25].

İnternetin yaygınlaşması, mobil hizmetler, kablosuz yayınlar, kablolu TV, e-ticaret, gibi yeni uygulamaların hızla artması IP ihtiyacını arttırmıştır. Bir bilgisayara bir IP verilirken gelişen teknoloji ile birlikte birden çok IP verilebilir hale gelmiştir. Ayrıca, gelişen teknoloji IPv4 adreslerinin hızla azalmasına hatta bu altyapıya cevap verememesine yol açmıştır. İnternet hızlı bir şekilde yaygınlaşırken adreslerin hızla tükenmesi büyük bir engel teşkil etmektedir. Yakın gelecekte internet ağları, mobil cihazlar, ağ aygıtları ve bilgisayarlar mevcut IPv4 yapısının kaldıramayacağı bir hale gelecektir. Bu durum, daha yeni ve daha geniş adresleme olanağı sağlayan bir yapıyı gerektirecektir [4,7,25].

IPv4 adreslerinin tükenmesi, gelişen teknoloji ve uygulamalar IPv4'ün eksikliklerini ve ihtiyaçlarını ortaya koymuştur. Bahsedilen bu eksiklikler ve yeni ihtiyaçların başlıcaları aşağıda kısaca açıklanan maddelerden oluşmaktadır.

<u>Veri doğrulama</u>

Kaynak adresten gelen IP paketinin gerçekten bu adresten geldiğinden emin olunmalıdır. İnternet protokolü ile kimlik gizleme (spoofing) olarak bilinen saldırıya

imkân vermemelidir. Bunun çözümü SA (Security Associations) dediğimiz güvenli iletişim desteğidir.

Veri bütünlüğü

Kaynak adresten gelen verinin hedef adrese ulaşıncaya kadar değişmediğinden/değiştirilmediğinden emin olunmalıdır. Başka bir deyişle bu verinin açılıp değiştirilmediğinden emin olunmalıdır. Bunun çözümü olarak başlık doğrulama (Authentication Header) yapılabilinmesidir.

Veri şifrelemesi

Veri bütünlüğünde bahsedilen paketin hedefine ulaşıncaya kadar açılıp okunmadığından emin olunmalıdır. Çözüm olarak kapsüllenmiş güvenlik veri yükü (Encapsulated Security Payload) yapılmasıdır.

Hiyerarşik adresleme

Var olan IPv4, internete bağlı ağların trafiğini sınıflandırmak için kendisine özgü adres hiyerarşisini kullanmaktadır. İnternet teknoloji ve uygulamalarının hızla gelişmesi bu adres hiyerarşisinin omurga yönlendiricileri ile IP adresi eklerini kullanarak ağ trafiğinin geçişini yönlendirir. Fakat bu hiyerarşi tek tür olmadığından ve IPv4 adreslerinin verimsiz dağılımından internet adresleme ve yönlendirmesini zorlaştırmaktadır. Ayrıca, var olan IPv4 web sitelerinin yeniden adreslenmesi zor ve maliyeti fazla olan bir işlemdir. Gelişen teknolojinin hızla büyümesi IPv4 adres yetersizliğine sebep olmaktadır. Çözüm ancak yeni nesil bir internet protokolü ile sağlanabilir. Yeni nesil internet protokolü IPv6 olarak adlandırılmaktadır.

Bütün bu ihtiyaç ve eksiklikler IPv6 ile çözülmektedir. Yukarıda bahsedilen ihtiyaç ve eksikliklere ek olarak yeni uygulamalar için daha karmaşık adresleme ve yönlendirme ihtiyaçları doğmuştur. Görüntülü konuşma ve telefon konferansı uygulamalarında / sistemlerinde bir paketi aynı anda internet üzerinden tüm gruba iletmek gerekir. IPv6 bu özellikleri de içinde barındıran yönlendirme ve adresleme tekniğine sahip bir internet protokolüdür [7,8,16,25].

2.2.3. IPv6'nın sunduğu yenilikler

IPv6, teknoloji ve ağ tarihinin en fazla iyileştirilmiş yapılarından biridir. IPv6'nın mevcut IPv4 altyapısına ve gelecek ağ altyapısına uyarlanması devam etmektedir. IPv4 ile çalışabilecek durumdadır. IPv6'da gerçekleştirilen temel değişiklikler mevcuttur. Bahsedilmesi gereken bu temel değişiklikler [26];

Genişletilmiş adresleme

IPv6'nın IPv4'ten temel farkı 128 bit olması ve 2^{128} = 340 282 366 920 938 463 463 374 607 431 768 211 456 adet adresi adreslemeye olanak sağlamasıdır. Bu sayede 1 m²'lik alana 1 mol = 6 024 x 10^{23} adet IPv6 adresi düşmektedir. Başka bir deyişle gezegenimiz üzerinde her bir kum tanesine bir IP adresi tanımlanabilecektir. IPv6'da tekli ve çoklu gönderim adreslerine ek olarak herhangi birine gönderim adresleme türü oluşturulmuştur. Herhangi birine gönderim adresleme ile gönderilen paketler bu adres türünü kullanan birçok düğümden veya birden fazla bilgisayar içinden herhangi birine gider. Örneğin, istediğimiz bir belgeyi barındıran internet sitesine HTTP GET isteği gönderildiğinde bu internet sitesine yakın ve uygun durumda olan başka sitelere de bağlanarak bilgi alınır [25,26]. IPv6 yüksek performanslı (Gigabit Ethernet, ATM v.b.) ağlarda alıcı ve verici arasında yüksek kalitede yol oluşturup paketlerin bu yol üzerinden daha iyi iletilmesini sağlar. Böylece daha kaliteli ve performanslı iletim isteyen ses ve görüntü iletimi için zemin hazırlar. Ayrıca, IPv6 yakın gelecekte gerekli olacak genişletilmiş adres, daha fazla güvenlik, servis kalitesi (Quality of Service) gibi yeni internet işlevselliği için bir alt yapı sağlamıştır [26].

Otomatik yapılandırma

IPv6'nın en ilgi çekici ve yeni özelliği durumsuz otomatik yapılandırma mekanizmadır. IPv6 dünyasında yer alan bir önyükleme aygıtı çalışır hale gelip ağ ön ekini sorguladığında, IPv6 yönlendiricisi bir veya daha fazla ağ ön ekini bağlı olduğu ağ üzerinden alabilir. Bu ön ek bilgilerini kullanarak, bir veya birden fazla evrensel IP adresini oluşturmak için ya kendi MAC tanımlayıcısını ya da özel rastgele bir sayı kullanarak otomatik yapılandırabilir. IPv4 dünyasında, her cihaz için benzersiz bir IP

adresini ya el ile yapılandırma ya da DHCP kullanılarak atamamız gerekir. Durumsuz otomatik yapılandırma ağ yöneticilerinin işlerini daha kolay hale getirir ve IP tabanlı ağları yapılandırmada önemli maliyet tasarrufu sağlar. Ayrıca, gelecekte evimizde veya işyerimizde IP adresi gerektiren cihaz sayısını düşünürsek, durumsuz otomatik yapılandırma özelliği vazgeçilmez olur. Evimize yeni bir televizyon satın aldığımızda DHCP sunucumuzu yeniden yapılandırmadan hemen IP adresi aldığını düşündüğümüzde otomatik yapılandırmanın bize sunduğu eşsiz kolaylığı fark edebiliriz. Durumsuz otomatik yapılandırma ayrıca yabancı bir ağa geçildiğinde örneğin, cep telefonu, avuç içi, iPad gibi mobil cihazlar için kolay bağlantı sağlar [25,26].

Basitleştirilmiş başlık biçimi

IPv6 başlığı IPv4 başlığına göre daha basit bir yapıya sahiptir. Başlık boyutu 40 byte'lık sabit uzunluğa sahiptir. Böylece daha kısa sürede daha fazla işlem olanağı doğmuştur. Basit olarak IPv4'e göre birçok değişiklik yapılmış bazı alanların da yerleri değiştirilmiştir. Temelde kaynak ve hedef adres için iki sefer 16 byte (128-bit) ve genel başlık bilgileri için ise sadece 8 byte (64-bit) alan tahsis edilmiştir. 64 bitlik işlemcilere göre tasarlanmış paket başlığı, paketlerin ayrıştırılmasının sadece uç noktalarda yapılması yönlendiricilerin farklı davranması hedeflenerek gerçek zamanlı veri trafiği ile anlık iletim istemeyen veri trafiğini daha hızlı bir biçimde işleyerek servis kalitesi arttırılmıştır [26].

Seçenekler ve uzantılar için geliştirilmiş destek

IPv4, seçenekler kısmını temel başlıkta barındırırken IPv6 seçenekler kısmını temel başlıktan çıkararak sadece gerektiğinde kullanılan uzatılmış başlıklara taşımıştır. Böylece daha esnek bir yapı sağlanmış, bu paketlerin gerektiğinde kullanılabilmesi amaçlanmış ve daha hızlı işlenmesine olanak tanınmıştır. Temel başlık özelliklerinde ek başlıklar tanımlanmıştır. Mobil IPv6, yönlendirme başlıkları, servis kalitesi, güvenlik bu ek başlıklara örnek gösterilebilir. Gelecekte ihtiyaç duyulduğunda ek başlık sayısı arttırılabilir şekilde tasarlanmıştır [26].

Akış etiketleme ve öncelik

IPv6'da başlık kısmına eklenen akış etiketleme alanı RFC 1752 ve RFC 2460 belgelerinde açıklanmıştır [5,6]. Akış etiketleme, gönderici isteklerine özel taşıma hizmeti vermek için özel akışlara ait paketlerin etiketlenmesi olarak tanımlanır. Bir örnek vermek gerekirse, ses ve görüntü iletimi bir akış olarak ele alınabilir veya değerlendirilebilir. Öte yandan, daha geleneksel uygulamalar, dosya aktarımı ve elektronik posta gibi, akış olarak değerlendirilmeyebilir. Daha iyi hizmet almak için daha fazla ödeme yapan yüksek öncelikli bir kullanıcının kullandığı ağ trafiği akış olarak değerlendirilebilir. Açık olan şudur ki, IPv6 tasarımcıları bir akışın anlamı tam olarak belirlenmemiş olsa bile akışların nihai ihtiyaçları arasında ayrım yapabilmeyi öngörmüşlerdir. IPv6 başlığı 4-bit öncelik alanına da sahiptir. Bu alan IPv4'deki TOS (Type of Service) - hizmet türü alanı gibi bir akışta belirli paketlere veya diğer belirli uygulamaların datagramlarına (ICMP paketleri) öncelik vermek için kullanılabilir [16,26].

Kimlik doğrulama ve gizlilik

IPv6 ile gelen en önemli özelliklerinden biri de kimlik doğrulama, veri bütünlüğü ve isteğe bağlı veri gizliliğini destekleyen uzantıların olmasıdır. Böylece veri güvenliği de ön plana çıkmıştır [26].

2.2.4. IPv6'nın başlık yapısı

IPv6 temel başlık yapısı Çizelge 2.7'de gösterilmiştir. Burada IPv6 paket yapısında bulunan alanların açıklamaları yapılmaktadır. Alanların açıklama sırası soldan sağa ve yukarıdan aşağıya şeklindedir. IPv6 başlık yapısında bulunan alanlar açıklamalarıyla birlikte sırasıyla şu şekildedir;

Sürüm (Version)

İnternet protokolünün sürümünü belirtir. 4 bit'lik boyuta sahiptir. Yönlendiriciler bu alana bakarak IP paketinin kalan kısımlarını nasıl çevireceğini belirler. IPv6'daki değeri 6'dır. IP başlığındaki dizilimi $(0110)_2$ şeklindedir [26].

Çizelge 2.7. IPv6 başlık biçimi [26]

<table>
<tr><td>Sürüm
(4-bit)</td><td>Trafik Sınıfı (8-bit)</td><td colspan="2">Akış Etiketi
(20-bit)</td></tr>
<tr><td colspan="2">Yük Uzunluğu-Boyutu
(16-bit)</td><td>Sonraki Başlık
(8-bit)</td><td>Atlama Sınırı
(8-bit)</td></tr>
<tr><td colspan="4">Kaynak Adresi (128-bit)</td></tr>
<tr><td colspan="4">Hedef Adresi (128-bit)</td></tr>
</table>

Trafik sınıfı (Traffic class)

Bu alan IPv4 başlığında bulunan servis türünün yerini almıştır. 8-bit'lik boyuta sahiptir. Gerçek zamanlı ve özel işlem gerektiren veriyi işleme kolaylığı sağlar. Ses ve görüntüyü diğer dosya aktarımlarından ayırt etme ve farklı şekilde iletme işlevini yapar. Ayrıca, gönderim düğümleri ve aktarım yönlendiricileri bu alanı IPv6 paketlerinin önceliklerini veya farklı sınıflarını belirlemek ve aralarındaki farkı bulmak için kullanabilir [16,26].

Akış etiketi (Flow label)

20 bit'lik boyuta sahip olan bu alan gerçek zamanlı trafiği işleme kolaylığını sağlamak için aynı işleyişi gerektiren paketleri ayırt eder. Gönderici, paketlerin sıralamasını bir dizi seçenek ile etiketleyebilir. Böylece, yönlendiriciler akışların izlerini sürebilirler ve aynı akışa ait paketleri daha verimli bir şekilde işleyebilirler çünkü her paketin başlığını tekrar tekrar işlemek zorunda kalmazlar. Burada dikkat edilmesi gereken husus, aynı akışa ait bütün paketler aynı kaynak ve hedef IP adreslerine sahip olmak zorundadırlar [16,26].

Yük uzunluğu-boyutu (Payload length)

16 bit'lik yapıya sahiptir. IP başlığından sonra taşınan verinin uzunluğunun sahip olduğu yükü belirler. IPv6'daki yük hesaplaması IPv4'deki hesaplamadan farklıdır. IPv4'deki uzunluk alanı IPv4 başlık uzunluğunu içermesine rağmen IPv6'daki yük uzunluğu sadece IPv6 başlığını izleyen veriyi içerir. Hesaplamalara ek başlıklar da katılır çünkü ek başlıklar alanı da yükün bir parçası olarak düşünülür. Aslında yük

uzunluğu alanının 16 bit yani 2 byte sınırı vardır. Paket yükü en fazla 64 KB olabilir. Fakat IPv6'nın gerektiğinde daha büyük paket boyutunu destekleyen ek başlık alanı sayesinde sadece IPv6 düğümlerine bağlı maksimum aktarım birimi (MTU) değeri 64 KB'dan büyük paketlere veya bağlantılara olanak sağlar [16,26].

Sonraki başlık (Next header)

Bu alan IPv4 başlığında protokol türü olarak adlandırılır. IPv6 başlığında ise IP paketlerinin yeni yapılarını yansıtmak için yeniden düzenlenerek "Sonraki Başlık" olarak adlandırılmıştır. Sonraki başlık, UDP veya TCP ise bu alan IPv4'de tanımlanan protokol numaralarının aynısını içerir. Örneğin, protokol numarası TCP için 6 veya UDP için 17'dir. Fakat IPv6 başlık yapısında yer alan ek başlıklar kullanılırsa, bu alan sonraki ek başlık türünü içerir. Ek başlıklar, IP başlığı ve TCP ya da UDP başlıkları arasında yer alır. Bu alanı bir ek başlık takip ediyorsa buraya takip eden ek başlığın adı yazılır. Eğer TCP ya da UDP izliyorsa bu kısma TCP veya UDP yazılır. Çizelge 2.8'de sonraki başlık alanına ait muhtemel değerler verilmiştir [16,26].

Çizelge 2.8. Sonraki başlık alanı değerleri [26]

Değer	**Açıklama - Tanım**
0	*IPv4* başlığında ayrılmış ve kullanılmamıştır. *IPv6* başlığında atlamalar arası geçiş seçenekleri başlığını belirtmek için kullanılır.
1	*IPv4*'de internet kontrol mesaj protokolü (ICMPv4) desteği için kullanılır.
2	*IPv4'*de internet grup yönetim protokolü (IGMPv4) desteği için kullanılır.
4	IPv4 için kullanılır.
6	TCP için kullanılır.
8	Dış ağ geçidi protokolü (EGP) için kullanılır.
9	Cisco tarafından IGRP için kullanılır.
17	UDP için kullanılır.
41	IPv6 için kullanılır.
43	Yönlendirme başlığı için kullanılır.
44	Bölümlendirme başlığı için kullanılır.
45	Alanlar arası yönlendirme protokolü (IDRP) için kullanılır.
46	Kaynak ayırma protokolü (RSVP) için kullanılır.
47	Genel yönlendirme kapsüllenemsi (GRE) için kullanılır.
50	Şifreli güvenlik yükü başlığı için kullanılır.
51	Kimlik doğrulama başlığı için kullanılır.
58	*IPv6*'da internet kontrol mesaj protokolü (ICMPv6) için kullanılır.
59	IPv6'da sonraki başlığın olmadığını bilgilendirmek için kullanılır.
60	Hedef seçenekleri başlığı için kullanılır.
88	Geliştirilmiş iç ağ geçidi yönlendirme protokolü (EIGRP) için kullanılır.
89	İlk açık yöne öncelik (OSPF) için kullanılır.
108	IP yükü sıkıştırma protokolü (IP Payload Compression Protocol) için kullanılır.
115	İkinci katman tünelleme protokolü (L2TP) için kullanılır.
132	Akış kontrol iletim protokolü (SCTP) için kullanılır.
135	Taşınabilirlik başlığı (Mobile IPv6) için kullanılır.
136-254	Atanmamış (Unassigned)
255	Ayrılmış (Reserved)

Atlama sınırı (Hop limit)

Bu alan IPv4 başlığında tanımlanan TTL (Time To Live) alanına benzerdir. TTL alanı bir paketin yok edilmeden önce ağ ortamında ne kadar süreyle kalacağı bilgisini içerir. IPv4'de birçok yönlendirici basitçe bu değeri her bilgisayar veya ağ aygıtını geçince bir azaltır. Bu değer sıfıra ulaşınca paket yok edilir. IPv6 başlık yapısında bu alanın adı "atlama sınırı" olarak değiştirilmiştir. Bu alanın değeri saniye sayısı yerine atlama sayısı olarak belirtilmiştir. Her aktarım düğümü bu sayıyı bir azaltır ve sayı sıfıra ulaşırsa paket yok edilir. Bir paketin en çok kaç atlama (bilgisayar, ağ aygıtı vb.) yapabileceği bu değer ile belirlenir. Örneğin, herhangi bir yönlendirici atlama sınırı bir olan paketi alırsa atlama sınırını sıfır yapar ve paketi yok eder. Göndericiye de "Aktarımda atlama sınırı aşıldı" ICMPv6 mesajını gönderir. Buradaki amaç, ağ ortamında gideceği yeri bulamayan paketlerin sonsuz bir döngüye girmesini engellemektir [16,26].

Kaynak ve hedef adresleri (Source and destination addresses)

IPv6 paketinin kaynağı ve varılacak noktayı belirleyen RFC 2373 ve RFC 4291'de özellikleri belirlenmiş 128 bit'lik IPv6 adresleridir [29,30]. Kaynak adres tarafından bir paket oluşturduğunda kaynak IP adres alanına kendi adresini, hedef adres alanına ise paketi göndereceği hedef adresi yerleştirir. Başka bir deyişle, kaynak adresin bilgisi hedef adresin içerisinde, hedef adresin bilgisi ise kaynak adresin içerisinde taşınır [16,26].

2.2.5. IPv6'nın ek başlıkları

IPv4 başlığı, seçenekleri (güvenlik seçenekleri, kaynak yönlendirme, zaman damgası v.b.) belirtmek için en az 20 byte ve en fazla 60 byte değerlerini alabilir. Bu kapasite nadiren kullanılır çünkü performans düşüşüne neden olur. Bir başlık ne kadar basit olursa işleme hızı da o kadar hızlı olur. IPv6'nın en esnek ve ilgi çekici özelliklerinden biri de ek başlıklardır. Ek başlıklar her zaman kullanılmayan sadece seçeneklere ihtiyaç duyulduğunda bir pakete yerleştirilir ve yeni ek başlıklar tanımlanabilir. Temel IP başlığında yer alan sonraki başlık kısmı sayesinde eklenmek

istenen ek başlıklar temel başlığa dâhil edilir. Ek başlıkların bazıları sabit bazıları da değişken boyuttadır. Ek başlık kullanmanın temel nedenlerinden biri ekonomi diğeri genişletilebilmedir. Böylece gereksiz başlıkların datagramdan çıkarılması ile zaman ve bant genişliğinden tasarruf edilmiştir. Performans artışı sağlanmış olacaktır. Bunlar ek başlıkların ne kadar ekonomik olabileceğini ve performans artışı sağlayabileceğini gösterir.

Buna ek olarak, IPv4'de sabit bir başlık kullanılmıştır ve herhangi bir değişiklik yapılmak istendiğinde tüm IPv4 başlığı değişir. IPv6'da ise her bir değişiklik yeni bir ek başlık olarak algılanır. Sonraki başlık (Next Header) bölmesi bu işlevi sağlar. Böylece, veri paketine yeni bir özellik eklenmesi durumunda tüm paketi değiştirmek yerine yeni bir ek başlık tanımlamak yeterli olur. Örneğin, datagramın (paketin) tamamı şifrelenmek istenirse bir adet şifre ek başlığı ile bu işlem yapılır. IPv6'nın bu esnek yapısı sayesinde gerekli/yararlı görülen başlıkların uygunluğu görüldüğünde kullanılabilir. Böylece ek başlıkların esnekliği, genişletilebilme ve geliştirilebilme özellikleri sayesinde gerekli zaman ve mekânda kullanılabilir.

Mevcut IPv6 özelliklerinin anlatıldığı RFC 2460 belgesinde altı ek başlık belirtilmiştir [26]. Bu ek başlıklar sırasıyla;

- *Atlamalar Arası Geçiş Seçenekleri Başlığı (Hop-by-Hop Options Header)*
- *Yönlendirme Başlığı (Routing Header)*
- *Bölümlendirme Başlığı (Fragment Header)*
- *Son Hedef Seçenekleri Başlığı (Destination Options Header)*
- *Kimlik Doğrulama Başlığı (Authentication Header)*
- *Şifreli Güvenlik Yükü Başlığı (Encrypted Security Payload Header)*

şeklinde kullanılmaktadır.

Bir IPv6 paketinde sıfır, bir veya birden fazla ek başlık olabilir. Ek başlıklar, IPv6 başlığı ve üst katman protokolü başlığı arasına yerleştirilir. Her bir başlık, önceki başlık içerisinde bulunan sonraki başlık alanı tarafından belirlenir. Ek başlıklar sadece IPv6 başlığında bulunan hedef/varılacak adres içerisinde belirtilen düğüm

tarafından denetlenir ve işlenir. Bu kuralda bir istisna vardır. Eğer ek başlık atlamalar arası geçiş seçenekleri başlığı (Hop-by-Hop Options Header) ise, taşınan veri paketin geçeceği her düğümün belirlediği yol boyunca incelenmek ve işlenmek zorundadır. IPv6 başlığında atlamalar arası geçiş seçenekleri başlığı mevcut ise mutlaka IPv6 başlığından hemen sonra gelmek zorundadır. Atlamalar Arası Geçiş Seçenekleri Başlığı, Çizelge 2.8'de gösterildiği gibi IPv6'nın başlığında yer alan sonraki başlık alanı değeri 0 (sıfır) olmalıdır. Eğer hedef adres birçoklu gönderim adresi ise ek başlıklar çoklu-gönderim adres grubuna bağlı bütün düğümler tarafından denetlenir ve işlenir. Ek başlıkların kullanımı RFC 2460 belgesinde belirtilmiştir. Şekil 2.2'de ek başlıkların kullanım biçimi verilmiştir.

Şekil 2.2. Ek başlıkların kullanımı [6,26]

Her ek başlığın uzunluğu sonraki başlıklarla uyumlu olsun diye 8 Byte ve katları şeklindedir. Her ek başlık 8 Byte'lık parçalar halinde kullanılmalıdır. Bu değere tamamlanamıyorsa gerekli yerlere boşluk doldurma işlemi yapılır. Bir düğümün sonraki başlığı işlemesi gerekiyorsa fakat sonraki başlık alanındaki değeri bilemiyorsa, paketi yok etmesi ve bir ICMPv6 parametre problem hatasını paketin kaynağına geri göndermesi gerekir. Tek bir pakette birden fazla ek başlık kullanılırsa, aşağıdaki başlık sıralaması kullanılmalıdır [6,26].

1. IPv6 Başlığı (IPv6 Header)
2. Atlamalar Arası Geçiş Seçenekleri Başlığı (Hop-by-Hop Options Header)
3. Son Hedef Seçenekleri Başlığı (Destination Options Header) (seçeneklerin IPv6 hedef adres alanında görülen ilk hedef tarafından işlenebilmesi için)

4. Yönlendirme Başlığı (Routing Header)
5. Bölümlendirme Başlığı (Fragment Header)
6. Kimlik Doğrulama Başlığı (Authentication Header)
7. Şifreli Güvenlik Yükü Başlığı (Encrypted Security Payload Header)
8. Son Hedef Seçenekleri Başlığı (Destination Options Header) (seçeneklerin sadece paketin son hedef noktası tarafından işlenebilmesi için)
9. Üst Katman Başlığı (Upper Layer Header)

Son hedef seçenekleri başlığı dışındaki her ek başlık en fazla bir kez kullanılmalıdır. Son hedef seçenekleri başlığı en fazla iki kez kullanılır. Bunlardan ilki yönlendirme başlığından önce diğeri ise üst katman başlığından önce kullanılmalıdır. Üst katman başlığı IPv6 üzerinden tünelleme ve kapsülleme durumunda kendi ek başlığını takip eder ve bu sıraya girer. Ek başlıklar ve temel özellikleri kısaca aşağıda özetlenmiştir.

Atlamalar Arası Geçiş Seçenekleri Başlığı (Hop-by-Hop Options Header)

Atlamalar arası geçiş seçenekleri ek başlığı ağ aygıtları (bilgisayar, anahtar, yönlendirici v.b.) arası geçişlerde her ara düğümlerde paketin aktarım yolu boyunca kullanacağı kontrol bilgilerini ve isteğe bağlı bilgiyi taşımak için kullanılır. Boyutu değişkendir. Atlamalar arası geçiş seçenekleri ek başlığı IPv6 başlığında yer alan ve değeri 0 (sıfır) olan "sonraki başlık" tarafından belirlenir [6,26]. Şekil 2.3'te atlamalar arası geçiş seçenekleri ek başlığının biçimi verilmiştir.

Şekil 2.3. Atlamalar arası geçiş seçenekleri ek başlık biçimi [6,26]

Bu başlık içerisinde sırasıyla şu alanlar vardır;

Sonraki Başlık

8 bit uzunluğundadır. Kendisinden hemen sonra gelecek ek başlığın türünü belirler. IPv4 protokol alanında yer alan değerlerin aynısını kullanır.

Başlık Genişleme Uzunluğu

8 bit uzunluğundadır. Atlamalar arası geçiş seçenekleri başlığının uzunluğunun ilk 8-sekizli (octets) birim hariç kaç adet 8-sekizli (octets) birimden oluştuğunu belirtir.

Seçenekler

Bir veya birden fazla seçenek olabilir. Uzunluğu değişkendir ve başlık genişleme uzunluğu alanı tarafından belirlenir.

Bu alan içinde yer alan *seçenek türü* 4 değer alabilir. Seçenekler alanlarının ilk 8-bitlik (1 byte) kısmı olan *seçenek türü*, işlem düğümünün seçeneği tanımaması halinde bu seçeneğin nasıl incelenmesi gerektiği hakkında bilgi içerir. Seçenek türü alanının ilk 2-bitlik kısmının yapması gereken işlemler;

- **00:** Geç ve işleme devam eder.
- **01:** Paketi yok eder.
- **10:** Paketi yok eder ve paketin kaynağına seçenek anlaşılmadı ICMP parametre hata mesajını gönderir.
- **11**: Paketi yok eder ve eğer paketin alıcı yani hedef adresi birçoklu yayın adresi değilse paketin kaynağına seçenek anlaşılmadı ICMP parametre hata mesajı gönderilir.

şeklinde tanımlanır [6,26].

Yönlendirme Başlığı (Routing Header)

Yönlendirme başlığı paketin hedefine ulaşması için izlemesi gereken bir veya birden fazla ara düğümün adres bilgilerinin listesini içerir. Yönlendirme başlığı değeri 43 olan sonraki başlık tarafından belirlenir. Şekil 2.4'de yönlendirme başlık biçimi gösterilmiştir.

Şekil 2.4. Yönlendirme ek başlık biçimi [6,26]

Yönlendirme başlığının sahip olduğu alanlar sırasıyla şunlardır;

Sonraki Başlık

8 bit uzunluğundadır. Yönlendirme başlığından hemen sonra gelecek başlığın türünü belirler. IPv4 protokol alanında yer alan değerlerin aynısını kullanır [6,26].

Başlık Genişleme Uzunluğu

8 bit uzunluğundadır. Yönlendirme başlığının uzunluğunun ilk 8-sekizli (octets) birim hariç kaç adet 8-sekizli (octets) birimden oluştuğunu belirtir. Yönlendirme türü değeri 0 (sıfır) olması durumunda başlık genişleme uzunluğu değeri başlık içerisinde listelenen adres sayısının iki katına eşit olur [6,26].

Yönlendirme Türü

8 bit uzunluğundadır. Belirli yönlendirme başlık biçimlerini ifade eder. Varsayılan değer 0'dır [6,26].

Kalan Segment

8 bit uzunluğundadır. Paketin son alıcı düğümüne varmadan önce geçmesi gereken ara düğüm sayısını gösterir [6,26].

Türe Özel Veri

Değişken uzunluktadır. Yönlendirme türü tarafından belirlenir. Kalan segment alanından sonraki tüm kısmı kapsar [6,26].

Paket bir düğümde işlem görürken yönlendirme türü alanında tanımlanmayan bilgi varsa düğümün nasıl davranması gerektiği kalan segment alanının değerine bağlı olur. Kalan segment alanının değerine bağlı olarak aşağıdaki işlemlerden biri yapılır;

- Kalan segment değeri 0 (sıfır) ise düğüm yönlendirme başlığını göz ardı etmek ve yönlendirme başlığında yer alan sonraki başlık tarafından türü belirlenen paket içindeki sonraki başlığı işleme sürecine devam etmek zorundadır.
- Kalan segment değeri 0 (sıfır) değilse düğüm paketi göz ardı etmek ve paketin kaynak adresinde yer alan adrese ICMP parametre hatası, kod 0 (sıfır), tanınmayan yönlendirme türü hatasını işaret eden bir mesaj gönderir.

Alınan paketin yönlendirme başlığının işlenmesinden sonra bir ara düğüm paketi MTU boyutunun paket boyutundan çok az olmasından dolayı işleyemezse ara düğüm paketi göz ardı eder ve paketin kaynağına ICMP "paket çok büyük" mesajını gönderir [6,26]. RFC 2460 [6] belgesinde tanımlanan tek yönlendirme türü 0 (sıfır) türünde yönlendirme başlığıdır. Bu türün çalışma ve işlenmesi şöyledir;

IPv6 başlığında yer alan sonraki başlık alanı yönlendirme başlığı için 43 değerini gösterir. Kaynak ve hedef adresler daha sonraki bölümlerde bahsedilecek olan 6to4 geçiş mekanizması sitelerine tahsis edilmiş 2002 ön ekine sahipler. Yönlendirme başlığı için sonraki başlık değeri 58 olan ICMPv6 olur. Yukarıda bahsedilen yönlendirme başlık alanlarından biri olan "kalan segment" 1 değerini alır. Çünkü seçenekler alanında sadece bir adet adres girdisi bulunur. En sonunda seçenekler alanı dolaşılması gereken adresleri listeler. Bu durumda sadece bir adres girdisi

mevcuttur. Eğer burada birçok ağ aygıtı adresi listelenmiş olsa IPv6 başlığında yer alan her hedef adres bu listeden kalan segment değerini 1 (bir) azaltarak sonraki adres değerini alır ve paketi yönlendirir. Bu süreç listede yer alan en son adrese varıncaya kadar devam eder. Örneğin, bir kaynak düğüm K yönlendirme başlığını kullanarak ara düğümler A1, A2 ve A3 aracılığıyla hedef düğüm H'ye bir paket gönderdiğinde Çizelge 2.9'da gösterilen değişmeler gözlenir.

Çizelge 2.9. Yönlendirme ek başlığının işlenmesi [6,26]

Paket Gönderim Durumu	**IPv6 Başlığı**	**Yönlendirme Başlığı**
K → A1	Kaynak Adres **K** Hedef Adres **A1**	Kalan Segment **3** Adres (1) = **A2** Adres (2) = **A3** Adres (3) = **H**
A1 → A2	Kaynak Adres **K** Hedef Adres **A2**	Kalan Segment **2** Adres (1) = **A1** Adres (2) = **A3** Adres (3) = **H**
A2 → A3	Kaynak Adres **K** Hedef Adres **A3**	Kalan Segment **1** Adres (1) = **A1** Adres (2) = **A2** Adres (3) = **H**
A3 → H	Kaynak Adres **K** Hedef Adres **H**	Kalan Segment **0** Adres (1) = **A1** Adres (2) = **A2** Adres (3) = **A3**

K:Kaynak Adresi **H**: Hedef Adresi, **A1:** Ara Düğüm 1 **A2:** Ara Düğüm 2 **A3:** Ara Düğüm 3'ü belirtir.

Bölümlendirme Başlığı (Fragment Header)

Bölümlendirme başlığı IPv6 kaynağı tarafından boyutu MTU (Maximum Transmission Unit) değerinden büyük bir paketin hedef adrese gönderilirken kullandığı ek başlık türüdür. Bölümlendirme özelliği IPv4'te de bulunur ve paketin geçtiği yol üzerinde bulunan yönlendiricilerde bölümlendirme işlemi yapılır. Fakat IPv6'da yapılan bölümlendirme işlemi sadece paketin gönderildiği kaynak tarafından yapılabilmesidir. Paketin birleştirilmesi işlemi hedef adres tarafından yapılır. Bir bölümlendirme başlığı önceki başlıkta yer alan ve değeri 44 olan "sonraki başlık" tarafından belirlenir. Şekil 2.5'te bölümlendirme ek başlık biçimi verilmiştir.

Şekil 2.5. Bölümlendirme ek başlığının biçimi [6,26]

Bölümlendirme başlık yapısında yer alan alanlar sırasıyla şunlardır;

Sonraki Başlık

8 bit uzunluğundadır. Bölümlendirme başlığından sonra gelecek başlığın türünü belirler. IPv4 protokol alanında yer alan değerlerin aynısını kullanır. Çizelge 2.8'de alabileceği değerler verilmiştir [6,26].

Saklı Tutulan Alan

8 bit uzunluğundadır. İletim için önceden tanımlı değeri 0 (sıfır)'dır. Paket iletiminde algılanması göz ardı edilir. Bu alan kullanılmaz [6,26].

Konum Bilgisi - Bölümlendirme Katsayısı

13 bit uzunluğundadır. Bu pakette yer alan 8-sekizlik birimler halindeki kayık/katsayı orijinal paketteki verinin başlangıcına bağlıdır. İlk parçanın değeri 0 (sıfır)'dır [6,26].

Saklı Tutulan Alan

2 bit uzunluğundadır. İletim için önceden tanımlı değeri 0 (sıfır)'dır. Paket iletiminde algılanması göz ardı edilir. Bu alan kullanılmaz [6,26].

M - Bayrağı

Bu alanın değerinin 1 (bir) olması durumunda daha bölümlendirilmiş parçanın olduğu anlamına gelir. 0 (sıfır) olması durumda ise son bölümlendirme parçası olduğu anlamına gelir [6,26].

Kimlik (Belirleme)

32 bit uzunluğundadır. Paketin kaynağı tarafından bölümlendirilmiş her bölüme bir kimlik verilir. Bölümlendirilecek her paket için kaynak düğüm bir kimlik değeri üretir. Bu verilen kimlik değeri/bilgisi bu alanda saklanır. Kimlik bilgisi, önceden gönderilen aynı kaynak ve hedef adreslerine sahip bölümlendirilmiş paketlerin her birisi için farklı olmak zorundadır. Kimlik bilgisi her bölümlendirme kısmı için farklıdır ve o kısma özgüdür [6,26].

Bir paketin ilk ve bölümlendirilmemiş büyük haline orijinal paket adı verilir. Orijinal paket iki kısımdan oluşur. *Bölümlendirilemeyen* ve *bölümlendirilebilen* kısım. *Bölümlendirilemeyen kısım* IPv6 başlığı ve paket hedefe varıncaya kadar ara düğümlerde işlenmesi gereken herhangi bir ek başlıktan (atlamalar arası geçiş seçenekleri, yönlendirme, kimlik doğrulama ek başlık v.b.) oluşur. *Bölümlendirilebilen kısım* ise aktarımı sağlanacak veriyi ve hedef düğümde işlenmesi gereken ek başlıkları içerir. Bu kısım 8 ve katları uzunluğundadır [6,26]. Şekil 2.6'de orijinal bölümlendirme paketi gösterilmiştir.

Şekil 2.6. Orijinal bölümlendirme paket biçimi [6,26]

Her bölümlendirme paketi, orijinal paketin bir kısmı olan *bölümlendirilemeyen kısım,* sonraki başlık, bölümlendirme katsayısı, M-bayrağı, kimlik alanlarını içeren *bölümlendirme başlığı* ve *bölümlendirmenin kendisinden* oluşur. Bölümlendirmelerin uzunlukları (boyutları) paketlerin hedef adreslerine gidecek yolun MTU büyüklüğüne sığmak zorundadır. Bölümlendirme işlemi Şekil 2.7'de gösterilmiştir.

Şekil 2.7. IPv6'da bölümlendirme işlemi [6,26]

Orijinal paketin bir alanı olan ve her bölümlendirmede görünen *bölümlendirilemeyen kısmı*, sırasıyla *bölümlendirme başlığı* ve *bölümlendirilebilir veri* izlemektedir. Orijinal pakette bulunan IPv6 başlığında biraz değişiklik yapılmalıdır. Uzunluk alanı sadece bölümlendirme (IPv6 başlığı hariç) uzunluğunu yansıtır. Bu uzunluk orijinal paketin uzunluğu değildir.

Hedef düğüm bütün bölümlendirmeleri toplar ve onları tekrar birleştirir. Bölümlendirmelerin tekrar birleştirilmesi için aynı kaynak - hedef adreslerine ve aynı kimlik değerlerine sahip olmalıdır. İlk kısımdan (bölümlendirme) sonra 60 saniye içerisinde bütün kısımlar (bölümlendirmeler) hedef adrese ulaşmazsa hedef bütün paketleri göz ardı eder yani işleme almaz. Eğer hedef katsayı değeri 0 olan ilk kısmı (bölümlendirme) alırsa kaynak adrese bir "bölümlendirme-parça birleştirme zamanı aşıldı" ICMPv6 mesajını gönderir. Hedef düğümde bölümlendirme paketleri Şekil 2.8'de gösterilen orijinal, bölümlendirilmemiş, paket haline çevrilir.

Şekil 2.8. Birleştirilmiş orijinal bölümlendirme paket biçimi [6,26]

Birleştirme işlemi aşağıdaki kurallar çerçevesinde yapılır [6,26];

- Birleştirilecek bölümlendirmelerin kaynak ve hedef adresleri ile kimlik değerleri aynı olmak zorundadır.
- Birleştirilen paketin bölümlendirilebilen kısmı bölümlendirme başlığından sonra gelen bütün bölümlendirmelerden (parçalardan) oluşturulur.
- Her bölümlendirmenin uzunluğu IPv6 başlığı ve bölümlendirmenin kendisi ile veri yükü değerinin çıkarılmasıyla yeniden hesaplanır.

Hedef/Alıcı Seçenekleri Başlığı (Destination Options Header)

Hedef seçenekleri ek başlığı aktarım yolu boyunca sadece paketin hedef düğümü tarafından incelenmesi gereken isteğe bağlı bilgiyi taşımak için kullanılır. Hedef seçenekleri ek başlığı IPv6 başlığında yer alan ve değeri 60 olan Sonraki Başlık tarafından belirlenir. Daha önceden de bahsedildiği gibi bir IPv6 paketinde iki kez hedef seçenekleri başlığı görünebilir. Yönlendirme başlığından önce kullanılırsa yönlendirme başlığında listelenen yönlendiriciler aracılığıyla işlenecek bilgileri içerir. Üst katman protokol başlıklarından önce kullanılırsa paketin son hedef bilgilerini içerir [6,26]. Şekil 2.9'da hedef - alıcı seçenekleri ek başlığının biçimi verilmiştir.

Şekil 2.9. Hedef/alıcı seçenekleri ek başlık biçimi [6]

Bu başlık içerisinde sırasıyla şu alanlar vardır;

Sonraki Başlık

8 bit uzunluğundadır. Hedef - alıcı seçenekler başlığından sonra gelecek ek başlığın türünü belirler. Çizelge 2.8'de yer alan değerleri kullanır.

Başlık Genişleme Uzunluğu

8 bit uzunluğundadır. Hedef - alıcı seçenekleri başlığının uzunluğunun ilk 8-sekizli (octets) birim hariç kaç adet 8-sekizli (octets) birimden oluştuğunu belirtir.

Seçenekler

Bir veya birden fazla seçenek olabilir. Uzunluğu değişkendir ve başlık genişleme uzunluğu alanı tarafından belirlenir.

Seçenekler alanı ilk 8-bitlik (1 byte) kısmı olan *seçenek türü*, işlem düğümünün seçeneği tanımaması halinde bu seçeneğin nasıl incelenmesi gerektiği hakkında bilgi içerir. Seçenek türü alanının ilk 2-bitlik kısmının yapması gereken işlemler;

- **00**: Geç ve işleme devam eder.
- **01**: Paketi yok eder.
- **10**: Paketi yok eder ve paketin kaynağına seçenek anlaşılmadı ICMP parametre hata mesajını gönderir.
- **11**: Paketi yok eder ve eğer paketin alıcı yani hedef adresi birçoklu yayın adresi değilse paketin kaynağına seçenek anlaşılmadı ICMP parametre hata mesajı gönderilir.

şeklinde tanımlanır [6,26].

Mobile IPv6, hedef-alıcı seçenekler ek başlığının kullanımına örnek olarak gösterilebilir [26].

Sonraki Başlık Yok (No Next Header)

IPv6 başlığında veya herhangi bir ek başlıkta yer alan sonraki başlık değeri 59 ise hiç bir şeyin bu başlığı takip etmeyeceğini belirtir. Eğer IPv6 başlığının yük uzunluğu sonraki başlık değeri 59 olan bir başlığın oktetlerini işaret ederse, bu oktetler göz ardı edilmeli ve hiçbir değişiklik yapılmadığına dair belirtilen yere yönlendirilmelidir [6,26].

Kimlik Doğrulama Başlığı (Authentication Header)

Kimlik doğrulama başlığı taşınan paketin güvenliğine yönelik olarak üç farklı unsuru destekler.

1. Veri Doğruluğu,
2. Veri Bütünlüğü,
3. Tekrar Engeli

olarak belirlenen bu unsurların kısaca işlevleri şöyledir;

Veri Doğruluğu

Bir paketin kaynak ve adresinin doğru olup olmadığının kontrol edilmesidir.

Veri Bütünlüğü

Verinin aktarım sırasında değiştirilip değiştirilmediği ve veriye müdahalede bulunulup bulunulmadığının kontrol edilmesidir.

Tekrar Engeli

Veriyi içeren paketin bir düğüm tarafından kopyalanıp içeriğinin değiştirilip yeniden gönderilmesinin ve aktarılmasının engelleme desteğidir.

Şekil 2.10. Kimlik doğrulama başlık biçimi [26]

Kimlik doğrulama başlığı yalnız kullanılmakla birlikte şifreli güvenlik yükü başlığı (Encrypted Security Payload Header) ile de kullanılabilir. IPSec (Internet Protocol Security) protokol takımından biridir. Sonraki başlık değeri 51 olan ek başlık türüdür. Şekil 2.10'da kimlik doğrulama başlık biçimi gösterilmiştir. Bu başlık içinde yer alan alanlar sırasıyla şöyledir;

Sonraki Başlık

8 bit uzunluğundadır. Kimlik doğrulama başlığından sonra gelen veri yükünün türünü belirler [26,27].

Veri Yükü Uzunluğu

8 bit uzunluğundadır. Kimlik doğrulama başlığının uzunluğunun 32 bit katları türünden karşılığının 2 eksiğidir. Bu başlığın ilk 8-sekizli (octets) birimi hesaplamaya dâhil edilmez. Alabileceği en düşük değeri 1'dir [26,27].

Saklı Tutulan Alan

16 bit uzunluğundadır. Gelecekte kullanım için ayrılmıştır. İletim için değeri mutlaka 0 (sıfır) olmalıdır [26,27].

Güvenlik Parametreleri Dizini

32 bit uzunluğundadır. Gönderenin kimliğinin doğrulanması için kullanılır. IPSec güvenlik birliği (security association) desteği de bu alanda verilmektedir. Bu alan için 0 değerinin kullanılması önerilmiştir. Değerin 0 olması yerel (local) kullanımda hata tespiti için kullanılabilir. Gelecekte kullanılması için IANA tarafından alabileceği değerler 1–255 arası olarak tanımlanmıştır [26,27].

Ardışıklık Numarası

32 bit uzunluğundadır. Monoton olarak artan düzende sayaç değeri içerir. Güvenlik birliği (security association) kurulduğunda alıcı ve gönderici düğümlerde bu sayaç sıfırlanır. Buradaki değer hiçbir zaman bir döngü içine alınmaz. Bu işlem ile *tekrar engelleme* desteği sağlanmış olur [26,27].

Kimlik Doğrulama Verisi

Değişken boyutta olup 32 bit katları şeklinde artar. Bu alanda *bütünlük kontrol değeri (integrity check value)* tutulur. Gerekli görülmesi halinde doldurma işlemi yapılmalıdır. Bu alanda gerçekleştirilen işlemler sayesinde veri bütünlüğü kontrolü sağlanmaya çalışılır [26,27].

Şifreli Güvenlik Yükü Başlığı (Encrypted Security Payload Header)

Bu başlık türü IPv4'de ve IPv6'da IPSec (Internet Protocol Security) protokol güvenlik hizmetine destek amaçlı tasarlanmıştır. Bu başlık dört farklı unsura bünyesinde yarar sağlamaktadır [28]. Bu unsurlar şunlardır;

Veri Gizliliği

Verinin başkasının eline geçmesi durumunda içeriğinin anlaşılmaması için farklı yöntemlerle (şifreleme v.b.) saklanmasıdır.

Veri Doğruluğu

Bir paketin kaynak ve adresinin doğru olup olmadığının kontrol edilmesidir.

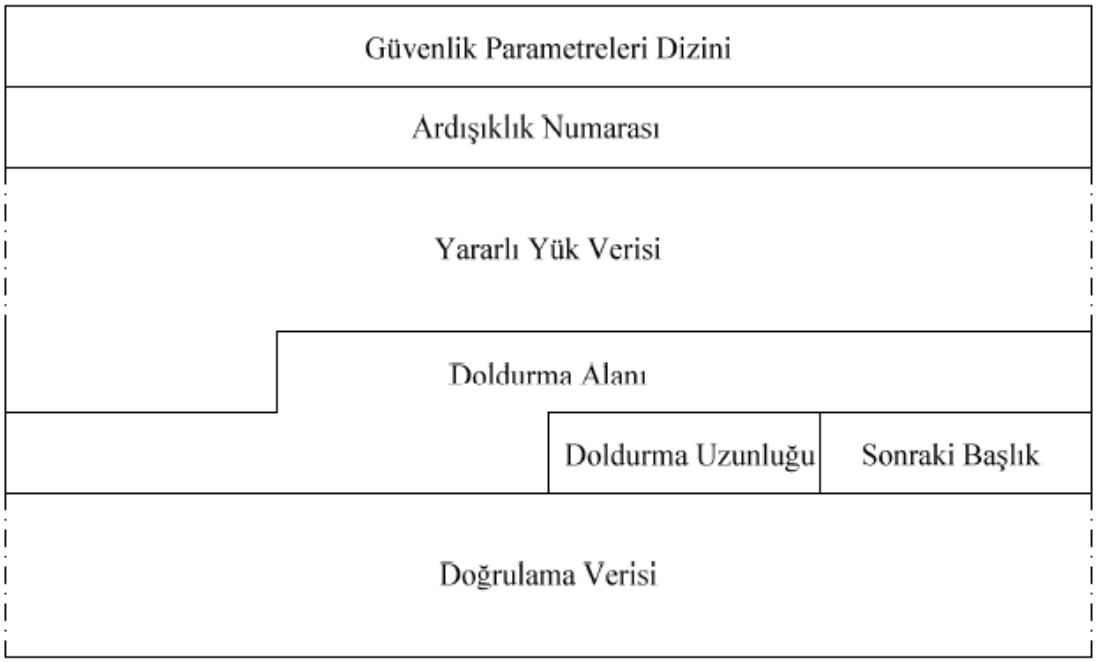
Tekrar Engeli

Veriyi içeren paketin bir düğüm tarafından kopyalanıp içeriğinin değiştirilip yeniden gönderilmesinin ve aktarılmasının engelleme desteğidir.

Sınırlı Trafik Akış Gizliliği

Bu unsurun kullanılması tünelleme yoluyla ile iletişimin sağlanması gerekir. Böylece daha etkili bir ağ geçit yolu güvenliği sağlanmış olur.

Bu başlık türü tek başına kullanılabileceği gibi daha önceden bahsedilen kimlik doğrulama başlığı ile birlikte veya iç içe (tünelleme yoluyla iletişim gibi) kullanılabilir. IPv6 başlığında yer alan sonraki başlık ve IPv4'te yer alan protokol alanın değeri 50 olan ek başlık türüdür. Bu başlığın kullanılması durumunda veri yapısında bulunan bazı alanların kullanılması gerekir [26]. Şekil 2.11'de şifreli güvenlik yükü başlık biçimi gösterilmiştir.



Şekil 2.11. Şifreli güvenlik yükü başlık biçimi [26]

Bu başlık içinde yer alan alanlar sırasıyla şöyledir;

Güvenlik Parametreleri Dizini

32 bit uzunluğundadır. Alıcı tarafından gönderici kimliğinin doğruluğunu belirlemek için kullanılır. Bu alanın kullanılması zorunludur. IPSec güvenlik birliği (security association) desteği bu alanda verilmektedir. Standartta yerel kullanım için 0 değerinin kullanılması önerilmiştir. Bununla hata tespiti amaçlanmıştır. IANA tarafından 1-255 arası değerler gelecekteki kullanımlar için saklı tutulmuştur [26,28].

Ardışıklık Numarası

32 bit uzunluğundadır. Bu alanda monoton düzende artan sayaç değeri tutulur. Gönderici tarafından ayarlanmalıdır. Güvenlik birliği kurulduğunda alıcı ve gönderici düğümlerde bu sayaç sıfırlanır. Buradaki değer hiçbir zaman bir döngüye girmez. Yeni bir güvenlik birliği kurulduğunda bu sayaç sıfırlanır. Bu işlem tekli gönderim güvenlik birliğinde tekrarı engelleme desteği sağlar. Bu alanın kullanılması zorunludur ve alıcı belirli bir güvenlik birliği için tekrar engelleme hizmetini aktif hale getirmezse bile mutlaka bu alanın olması gerekir [26,28].

Yararlı Yük Verisi

Değişken uzunlukta orijinal IP paketinde sonraki başlık alanında türü tanımlanmış veriyi içerir. Bu alanın kullanılması zorunludur. Şifreleme mekanizması tarafından ihtiyaç duyulduğunda şifrelenmiş veriye ek olarak şifreleme başlangıç vektörünü içerir. Kullanılacak şifreleme algoritması kriptografik eşleme verisi gerektiriyorsa bu veri bu alanda taşınabilir [26,28].

Doldurma (şifreleme için)

Paketi 4 byte katları şeklinde hizalamak için kullanılan 0–255 Byte uzunluğuna sahip olabilen alandır. Bir seferlik bile olsa şifreleme mekanizması ihtiyaç duyduğunda en az bir paket boyutuna ulaşması için kullanılır. Şifreleme algoritması tarafından ihtiyaç duyulan boyuta ulaşmaması durumunda boş kalan yerlere düz metin

doldurmak için de kullanılır. Bir ek-başlık içerisinde kullanılması zorunlu değildir fakat bütün uygulamaların bu kısmı desteklemesi zorunlu kılınmıştır. Kullanılabileceği bazı durumlar için bilgiler ilgili RFC belgesinde açıklanmıştır [26,28].

Doldurma Uzunluğu

8 bit uzunluğundadır. Kendisinden hemen bir önceki alanında, doldurma alanında, kaç Byte'lık doldurma yapıldığının değeri tutulur. Geçerli olan değerleri 0 ile 255 arasıdır. 0 değeri hiçbir byte değrinin olmadığını belirtir. Bu alanın kullanılması zorunludur [26,28].

Sonraki Başlık

Kullanılması zorunlu ve 8 bit uzunluğunda alandır. Yararlı yük verisi alanında yer alan verinin türünü belirler. Bu alandaki değerler IANA tarafından belirlenen ve Çizelge 2.8'de verilmiş olan numaralar arasından seçilir. Örneğin, 4 değeri IPv4, 41 değeri IPv6 ve 6 değeri TCP değerlerini ifade eder [26,28].

Doğrulama Verisi

Bütünlük kontrol değeri olarak da bilinen bu alan değişken uzunluktadır. Kullanılması zorunlu değildir sadece bütünlük hizmeti seçildiğinde kullanılır. Uzunluğu seçilen bütünlük algoritması ve güvenlik birliği tarafından belirlenir [26,28].

2.3. IPv6'nın Adresleme Yapısı

2.3.1. IPv6 adresler

IPv4'de olduğu gibi IPv6 da her bir ağ aygıtına bir fiziksel bağlantı adresi verilir. Fakat IPv6 adresleme yapısı IPv4'ten farklıdır. Bir IPv4 adresi 32-bit'e, IPv6 adresi ise 128-bit'e sahiptir. IPv4 2^{32} adrese olanak sağlarken IPv6 ise 2^{128} adet adrese olanak sağlar. IPv4'te adres sayısı yaklaşık 4,29 milyar iken IPv6'da bu sayı 340 282 366 920 938 463 463 374 607 431 768 211 456 veya 6 65 x 10^{23} demektir. IPv6

adreslerinde çok basamaklı hiyerarşik bir yapı vardır ve yayın türü (broadcast) adresleme yoktur. Bunun dışında herhangi bir alıcılı (anycast) adres türü geliştirilmiştir. Çizelge 2.10, RFC 2373 ile belirlenmiş IPv6 adres dağılımını göstermektedir. RFC 2373 belgesinin güncel şekli RFC 4261 belgesinde yer almıştır [29]. Bu dağılımda en önemli adres küresel tekil adreslerdir. Bu adresler internete eklenecek her yeni bilgisayar için tekil bir biçimde eklenecektir. 128 bit'in ilk üç biti "001" ikilik öneki ile başlar ve bu ön ekin en küçük değeri 0010'dır. Bu değer on altılık olarak 2'ye karşılık gelir. En büyük değeri ise 0011'dir. Bu değer on altılık olarak 3'e karşılık gelir. Bu durumda küresel tekil adresler "**2000:0:0:0:0:0:0:0**" adresinden başlar ve "**3FFF: FFFF: FFFF: FFFF: FFFF: FFFF: FFFF: FFFF**" adresine kadar devam eder [26,29,30].

Çizelge 2.10. IPv6 adreslerinin dağılımı [29]

Dağılım	**İkilik Öneki**	**Adres Aralığı Oranı**
Ayrılmış	0000 0000	1/256
Atanmamış	0000 0001	1/256
NSAP Dağılım için Ayrılmış	0000 001	1/128
IPX Dağılım için Ayrılmış	0000 010	1/128
Atanmamış	0000 011	1/128
Atanmamış	0000 1	1/32
Atanmamış	0001	1/16
Küresel Tekli Adresler	001	1/8
Atanmamış	010	1/8
Atanmamış	011	1/8
Atanmamış	100	1/8
Atanmamış	101	1/8
Atanmamış	110	1/8
Atanmamış	1110	1/16
Atanmamış	1111 0	1/32
Atanmamış	1111 10	1/64
Atanmamış	1111 110	1/128
Atanmamış	1111 1110 0	1/512
Yerel Hat Tekli Adres	1111 1110 10	1/1024
Yerel Site Tekli Adres	1111 1110 11	1/1024
Çoklu Yayın Adresleri	1111 1111	1/256

Bir IPv6 adresinin türü adresin en yüksek öncelikli bit'ler tarafından belirlenir. Çizelge 2.11'de RFC 4291 ile belirlenmiş adres türü belirleme bilgileri verilmiştir [29].

Çizelge 2.11. IPv6 adres türleri ve gösterimi [29]

Adres Türü	**İkilik Öneki**	**IPv6 Gösterimi**
Belirtilmemiş	00...0 (128 bit)	::/128
Geri Dönüş Devresi (Loopback)	00...1 (128 bit)	::1/128
Çoklu Alıcı	11111111	FF00::/8
Yerel-Hat (Link-Local)	1111111010	FE80::/10
Küresel Tek Alıcılı	(geriye kalanlar)	

Yeni nesil internet protokolü olan IPv6'da adresler üç kategoriye-türe ayrılır;

Tek alıcılı (unicast) adresler

Herhangi bir düğümün bir tek ara yüzünü belirtir. Aynı ara yüz bir den fazla adrese sahip olabilir. Bir paket bu adrese yollanırsa en kısa yoldan bu adres tarafından belirlenen hedefe iletilir. Örneğin bir abonenin iki farklı internet erişim sunucusuna bağlı iken her iki erişim sunucusunda görünen IP adresleri birbirinden farklıdır [26,29].

Çoklu alıcılı (multicast) adresler

Birden fazla ara yüzü belirten adreslerdir. Bu tür adreslere gönderilen bir paket bu adrese sahip bütün ara yüzler tarafından alınır [26,29].

Herhangi bir alıcılı (anycast) adresler

Farklı düğümlere ilişkin ara yüzler ve birden çok ara yüzü belirten bir adres türüdür. Bu adrese gönderilen bir paket en kısa yoldan bu ara yüzlerde bulunan bilgisayar veya ağ aygıtlarından sadece birine gider. Başka bir ifadeyle bir paket bu adrese sahip bir ara yüzlerden en yakın ve uygun olanına iletilir. Bu adreslere sahip ve birbirinin yedeğini barındıran ara yüzlerden bir tanesinin arızalanması durumunda bile kullanıcılar bunu fark edemeyeceklerdir [26,29].

Yukarıda söz edilen adresler dışında ağ aygıtlarımızın alabileceği (tek alıcılı adresler içerisinde gösterilebilen) IPv6 ile birlikte kullanılan adresler de vardır [26,29]. Aşağıdaki başlıklarda kısaca açıklanan bu adres sınıfları şunlardır:

Küresel adresler (Global addresses)

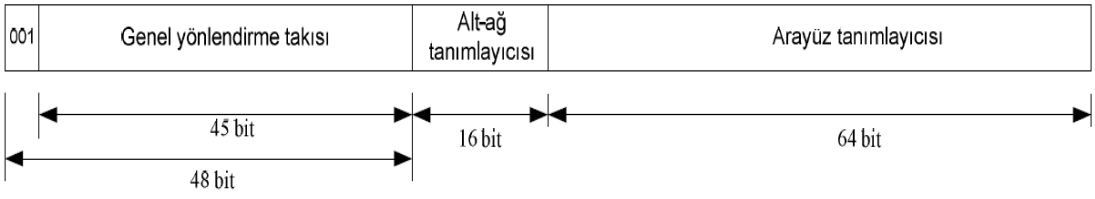
Her yerde kullanılabilen adreslerdir. Bu adresler internet ortamında yönlendirilebilir ve ulaşılabilir durumdadır. IANA tarafından ayrılmış olan küresel tek alıcılı adreslerin ilk üç bitlik kısmı “001” olmaktadır. Bundan sonraki ilk 45 bit küresel yönlendirme takısı için kullanılmıştır. Bundan sonraki 16 bit alt ağ tanımlayıcısı ve son kalan 64 bit ara yüz tanımlayıcısı için ayrılmıştır. Şekil 2.12’de küresel tek alıcılı IPv6 adres yapısı verilmiştir [26,29].

Yerel-bağlantı-hat adresler (Link-local addresses)

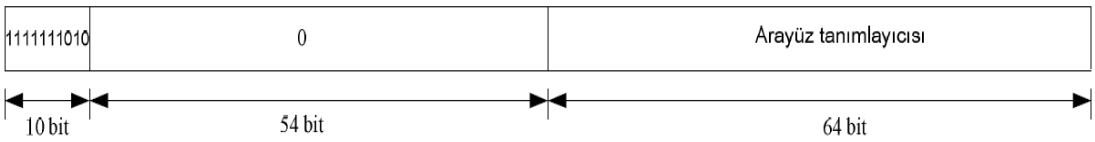
Aynı bağlantıya sahip ve aynı bağlantı üzerinde yer alan düğümler arasında iletişimi sağlamak için kullanılır. Yönlendiriciler bu adreslere ait trafiği hat (link) dışına çıkarmazlar. Bu adres türünde en anlamlı 10 bit "1111 1110 10" şeklindedir. Sonraki 54 bit "0" dır. Geriye kalan 64 bit ise ara yüz tanımlayıcısı bilgisini içerir. Bu adres türleri her zaman FE80 ile başlarlar. Şekil 2.13’te yerel-hat tek alıcılı IPv6 adres yapısı verilmiştir [26,29].

Yerel-mekan adresler (Site-local addresses)

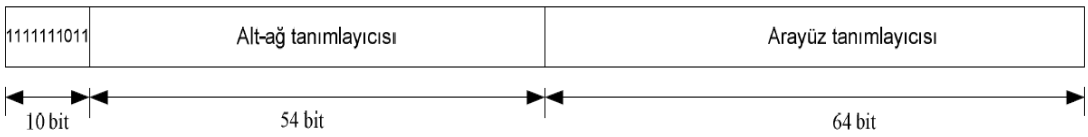
Aynı mekân veya mevki içerisinde kullanılacak adres türüdür. Günümüzde ev, iş veya ofis ağlarımızda kullandığımız adreslere benzetilebilir. Bu adres türünde trafik yönlendiriciler tarafından mekân içerisinde kalacak şekilde yönlendirilir. Tanımlanan mekân dışına çıkmak isteyen paketler yönlendiriciler tarafından geçirilmezler. Bu adres türünde en anlamlı 10 bit "1111 1110 11" şeklindedir. Sonraki 54 bit alt ağ tanımlayıcısını belirtir. Geriye kalan son 64 bit ise ara yüz tanımlayıcısını belirtir. Bu adreslerin elle atanması gerekir. Şekil 2.14’de yerel-mekân tek alıcılı IPv6 adres yapısı verilmiştir [26,29,30].



Şekil 2.12. Küresel tek alıcılı IPv6 adres biçimi [29,30]



Şekil 2.13. Yerel-hat (link-local) tek alıcılı IPv6 adres biçimi [29]



Şekil 2.14. Yerel-mekan (site-local) tek alıcılı IPv6 adres biçimi [29]

2.3.2. IPv6 özel adresler

IPv6'da bulunan özel adres türleri aşağıdaki alt başlıklarda kısaca açıklanmıştır. Bu özel adres türleri şunlardır;

Belirtilmemiş adres (The unspecified address)

IPv4 adres yapısında bulunan ve "**0.0.0.0**" adresine karşılık gelen IPv6 adresidir. Belirtilmemiş adres (unspecified address) değeri `0:0:0:0:0:0:0:0`'dır. Bütün değerlerinin sıfır olmasından dolayı ayrıca "*hepsi sıfır adresi*" olarak da adlandırılmaktadır. Bu adres türü :: olarak kısaltılabilir. Genellikle soket bağlamada veya yönlendirme tablolarında kullanılır. Yönlendirme tablolarında bütün muhtemel ağ ve ağ aygıtlarını belirlemek için kullanılır. Bu adres hiçbir zaman dinamik veya statik olarak herhangi bir ara yüze verilmemelidir. Ayrıca bir hedef IP adresinde veya bir IPv6 yönlendirme başlığında görünmemelidir [26,29].

Geri dönüş arabirim-devre adresi (The loopback address)

IPv4 adres yapısında bulunan ve "**127.0.0.1**" geri dönüş arabirim adresine karşılık gelen IPv6 adresidir. IPv6 da yazım şekli şöyledir;

```
0000:0000:0000:0000:0000:0000:0000:0001
```

Geri dönüş arabirim-devre adresi **::1** olarak kısaltılabilir. Kaynak veya hedef adresinde bu adres bulunan paketler alt ağa erişmezler ve gönderici adresi terk etmezler. IP yığınlarını sınamak veya hata bulmak için kullanılırlar. Bu adres hiçbir zaman dinamik veya statik olarak herhangi bir ara yüze verilmemelidir [26,29].

Gömülü olarak IPv4 adresleri barındıran IPv6 adresleri (IPv6 addresses with embedded IPv4 addresses)

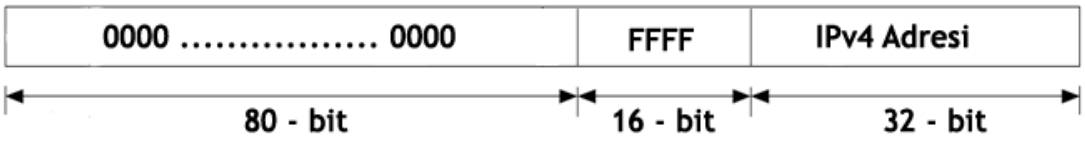
IPv4 adreslerini içeren iki çeşit IPv6 adresi vardır [26,29].

- IPv4 Uyumlu IPv6 Adresleri
- IPv4 Haritalı IPv6 Adresleri

IPv4 uyumlu IPv6 adresleri, IPv4 yönlendirme mimarisi üzerinden IPv6 paketlerini dinamik olarak tünellemek için kullanılır. IPv6 adresini en düşük öncelikli son 32-bit'lik kısmına IPv4 adresi eklenir. Bu adres 96 bit uzunluğunda ön ek ile tanımlanır. X.Y.Z.W IPv4 adresi olarak belirtildiği örneğimiz şöyledir; **0:0:0:0:0:X.Y.Z.W /96** veya kısaca **::X.Y.Z.W** olarak gösterilebilir. Bu adres türü çok az kullanıldı ve RFC 4291 belgesinde bu adresin kaldırıldığı bildirilmiştir [29]. Şekil 2.15'te IPv4 uyumlu IPv6 adres yapısı verilmiştir [26,29].

Şekil 2.15. IPv4 uyumlu IPv6 adres biçimi [29]

IPv4 haritalı IPv6 adresleri, IPv4 yönlendirme mimarisi üzerinden IPv6 paketlerini dinamik olarak tünellemek için kullanılır. IPv6 adresini en düşük öncelikli son 32-bit'lik kısmına IPv4 adresi eklenir. Bu adres 96 bit uzunluğunda ön ek ile tanımlanır. X.Y.Z.W IPv4 adresi olarak belirtildiği örneğimiz şöyledir; **0:0:0:0:FFFF:X.Y.Z.W /96** veya kısaca **::FFFF:X.Y.Z.W** olarak gösterilebilir. Şekil 2.16'da IPv4 haritalı IPv6 adres yapısı verilmiştir [26,29].



Şekil 2.16. IPv4 haritalı IPv6 adres biçimi [29]

Bu iki adres türü birbirine çok benzerdir. Aralarındaki tek fark Şekil 2.17 ve Şekil 2.18'de gösterilen 16-bit'lik kısımların aldığı değerlerin farklı olmasıdır. Bu kısmın değerinin 0 (sıfır) olması IPv4 Uyumlu IPv6 adresi, 1 (bir) olması durumunda ise IPv4 Haritalı IPv6 adresi olduğunu belirtir [26,29].

2.3.3. IPv6 adres gösterimi

Bir IPv6 adresi 128 bit veya 16 byte'dır. IPv6 adresleri 16'lık tabanda (hexadecimal) gösterilmektedir. Bir IPv6 adresi 8 adet 16-bit (8 x 16 = 128 bit) on altılık taban bloklarının "**:**" (iki nokta üst üste) ile ayrılmasıyla gösterilir. Büyük küçük harf duyarlı değildir. Bir bloğun veya kolonun en büyük on altılık tabandaki değeri "**FFFF**" değeridir. Örnek,

2001:0DB8:0000:0000:0202:B3FF:FE1E:FFFF

Örnekte de görülebileceği gibi her dört karakterden oluşan bloklar-kolonlar birbirinden "**:**" karakteriyle ayrılmıştır ve adresin en son bloğu olan **FFFF** değeri bu IPv6 adresinin en büyük kolonudur. IPv6 adreslerinin daha kolay anlaşılması için bazı kısaltmalar yapılabilir. Sıfırların çok olduğu durumlar için kısaltma yapılarak tek sıfırla gösterilir. Yukarıda gösterilen örnek;

2001:DB8:0:0:202:B3FF:FE1E:FFFF

şeklinde gösterilebilir. Diğer bir kısaltma şekli ise birden fazla kolonda yer alan değerleri 0 olan bu kolonların arasına iki tane "**:**" (**::**) karakteri yerleştirilir. Bir IPv6 adresinde bu sadece bir kez gösterilir. Örneğimize bahsettiğimiz bu kuralı uygularsak;

2001:DB8::202:B3FF:FE1E:FFFF

şeklinde gösterilecektir [26,29].

Başka bir IPv6 adresini "**2001:DB8:0000:0056:0000:ABCD:EF12:1234**" örnek olarak ele alalım. Bahsettiğimiz kurallar çerçevesinde adresimizi sırasıyla şöyle gösterilebiliriz;

2001:DB8:0000:0056:0000:ABCD:EF12:1234

2001:DB8:**0:56:0**:ABCD:EF12:1234

2001:DB8:**:56:0**:ABCD:EF12:1234

2001:DB8:**0:56:**:ABCD:EF12:1234

Ön ek (Prefix)

Alt ağları veya özel bir adres türünü tanımlamak için kullanılan bir adresin en yüksek öncelikli bit'lerinden oluşan kısmına Ön Ek denir. Daha önceden biçim ön eki (format prefix) olarak adlandırılmıştır. Şimdi ise küresel yönlendirme ön eki olarak adlandırılmaktadır. Ön ek kavramı IPv4 adreslerinde yer alan CIDR (Classless InterDomain Routing) yazım şekli ile çok benzerdir. Bir IPv6 adresinde yer alan bitlerin sayısının "/" karakteriyle ayrılarak yazılmasına ön ek uzunluğu denmektedir. Gösterim şekli şöyledir;

IPv6 Adresi / Ön Ek uzunluğu

Ön ek uzunluğu ön eki tanımlayan yüksek öncelikli bit'lerin kaç tanesinin kaldığını belirtir. IPv4 adreslemede kullanılan alt ağ maskesi (subnet mask) kullanımına benzer. Örneğin bir IPv6 adresinin ön ek uzunluğu ile birlikte gösterimi şöyledir;

2E78:DA53:1200::/40

Bu adresi anlamak için Çizelge 2.12'de gösterilen adresin onaltılık (16'lık) ve ikilik (2'lik) gösterimlerine bakalım.

Çizelge 2.12. Ön ek gösterimi [26]

Onaltılık Tabanda Gösterim	**İkilik Tabanda Gösterim**	**Bit Sayısı**
2E 78	0010 1110 0111 1000	16
DA 53	1101 1010 0101 0011	16
12 00	0001 0010 0000 0000	8
00 00	0000 0000 0000 0000	0
00 00	0000 0000 0000 0000	0
		Toplam: 40 bit

Örneğimizde gösterilen adresin açık hali Çizelge 2.12'de verilmiştir.

2.3.4. IPv6 adres yönetimi

IPv6 adresleri de IPv4 adresleri gibi IANA tarafından uluslararası adres dağıtımı çeşitli bölgesel kayıt hizmetlerine verilmiştir. Amerika için ARIN (American Registry For Internet Numbers), Afrika için AFRINIC (African Network Information Centre), Asya – pasifik bölgesi için APNIC (Asia Pacific Network Information Centre), Latin Amerika ve Karayipler bölgesi için LACNIC (Latin America and Caribbean Network Information Centre), ve Avrupa bölgesi için RIPE NCC (Réseaux IP Européens Network Coordination Centre) tarafından adreslerin dağıtımı yapılmaktadır. Bu bölgesel kayıt hizmetlerini veren her kuruluşun kendi internet sitelerinde adres dağıtım-tahsis konuları, mevcut uygulamaları ve izlenmesi gereken yollar hakkında bilgi vardır. Çizelge 2.10'da IPv6 adreslerinin dağılımı gösterilmiştir. Bu dağılımlara ek olarak RFC belgelerinde belirtilen mevcut dağılımlar bulunmaktadır. Çizelge 2.13'te bu dağılımlar gösterilmiştir.

Çizelge 2.13. IPv6 mevcut dağıtımları [26]

Ön Ek	**Dağılım**	**RFC Belgesi**
2000::/3	Atanabilen Küresel Tek Alıcılı Adres Uzayı	RFC 3513
2001:0000::/32	Teredo	RFC 4380
2001:DB8::/32	Sadece Belgelendirme Amaçlı, Yönlendirilemez Adresler	RFC 3849
2002::/16	6to4	RFC 3056
3FFE::/16	6Bone Test	RFC 2471

2000::/3 adresleri küresel tek alıcılı adres olarak ağ aygıtlarına atanabilen adreslerdir. Bu adresler RFC 3513 belgesinde detaylı olarak verilmiştir.

2001:0000::/32 adresleri IPv6 geçiş mekanizmalarından olan Teredo için kullanılan adreslerdir. RFC 4380 belgesinde detayları verilmiştir.

2001:DB8::/32 adresleri sadece belgelendirme amaçlı kullanılan ve ağ içerisinde yönlendirilmeyen adreslerdir.

2002::/16 adresleri IPv6 ağları üzerinden IPv4 ağlarını kullanabilmek için yapılandırılan 6to4 tünelleme geçiş mekanizması için kullanılmaktadır.

3FFE:: /16 ise IPv6 omurga test adresleri olarak kullanılırlar.

Adres yönetiminde bir IPv6 adresi IPv4 adreslerinde olduğu gibi iki parçadan oluşur;

1. *Ağ tanımlayıcısı* Belirli ve yüksek öncelikli bit'lerin tanımladığı ağ için kullanılır.
2. *Ağ aygıtı / bilgisayar tanımlayıcısı* Ağ tanımlayıcısının dışında kalan bit'lerin tanımladığı bilgisayar – ağ aygıtları için kullanılır.

Genellikle IPv6 adreslerinin ilk 48-bit'lik kısmı ağ tanımlayıcısını, sonraki 16-bit alt ağları ve en son kalan 64-bit ise ağ aygıtlarını – bilgisayarları belirtir. IPv4'ten IPv6'ya geçişte adresleme sisteminde bazı çalışmalar yapılmıştır. IEEE (Institute of

Electrical and Electronics Engineers) tarafından geliştirilen EUI64 adresleme bu çalışmalardan biridir.

EUI64 Adresleme

Bu adresleme mekanizmasında bir aygıta verilecek olan IPv6 adresi o aygıtın ağ ara yüz kartının (NIC – Network Interface Card) sahip olduğu MAC (Media Access Control) adresinden elde edilmesi amaçlanmıştır. MAC adresi her aygıta özgü ve 48-bitlik yapıdadır. Durumsuz otomatik yapılandırma bu mekanizmayı ağa bağlı olan arabirimin MAC adresini alarak bir IPv6 adresini türetir. Bu adresleme mekanizmasına bir örnek verecek olursak;

MAC adresi **00:10:A4:E3:95:66** olan bir arabirimin IPv6 adresi şöyle belirlenir. Öncelikle MAC adresimiz 48-bit olduğundan bunu 64-bit'e tamamlamamız gerekir. EUI64 standardına göre MAC adresimizin ilk 24-bit ve son 24-bit'lik kısımlarının arasına her zaman **FF:FE** değeri eklenir. Böylece 64-bitlik bir değer elde etmiş oluruz. Bu değer **00:10:A4:**FF:FE**:E3:95:66** olacaktır. Bundan sonra 64-bitlik ağ tanımlayıcısı numarası bu değerin başına getirilir. Ağ tanımlayıcımızın yerel bağlantı adreslerimizden **FE80::/64** olduğunu varsayalım. MAC adresimizden EUI64 mekanizmasını kullanarak elde ettiğimiz **00:10:A4:**FF:FE**:E3:95:66** değerini dörderli gruplar şeklinde birleştirerek **0010:A4**FF:FE**E3:9566** değerini elde ederiz. En son olarak ağ tanımlayıcımızı ve elde ettiğimiz son değeri birleştirerek **FE80::0010:A4**FF:FE**E3:9566** IPv6 adresini elde etmiş oluruz.

2.4. IPv4 ve IPv6 Karşılaştırması

İnternet protokolü bilgisayar veya ağ aygıtlarının bir ağ üzerinden nasıl iletişim kuracağını belirleyen teknik kuralların dizisidir. Karışık yapıdaki ağların birbirleri ile haberleşmesini sağlaması, hızlı değişimlere cevap vermesi ve interneti mümkün kılması IP (internet protokolü)'nin gayet başarılı bir protokol olduğunu gösterir. Birbirinden farklı yapıdaki ağlar arasında iletişimi sağlamak amacıyla IP, tek tip paket formatı ve iletim mekanizması sunmuştur. Şu anda aktif olarak kullanılmakta olan iki çeşit IP sürümü vardır; *IPv4* ve *IPv6*.

Bir uygulama herhangi bir bilgisayardan diğerine internet veya başka ağ üzerinden veri aktarırken IP paketi kullanır. IP, OSI katmanlarından 3. katman olan ağ katmanında yer aldığından fiziksel olarak hattın ne olduğuna bakmadan ve donanım olarak ağ kartının türüne bağımlı olmadan bir üst katmanda (taşıma katmanı) farklı yapıdaki bilgisayarların haberleşmesini sağlar. Böylece IP fiziksel veya donanımsal değişimlerinden etkilenmemiştir. Bu yüzden internet milyonlarca bilgisayara ve kullanıcıya ulaşmıştır. Mevcut internet teknolojisi IPv4 ile IPv6'yı aynı ortamda kullanmayı sağlamaktadır. IPv6 desteği olan ağların çoğu hem IPv4 hem de IPv6 adreslerini kullanabilmektedir (ikili yığın mekanizması gibi) [26,32].

IPv6'nın getirmiş olduğu yenilikler sayesinde IPv4'de yaşanılan sorunlar giderilmiş durumdadır. IPv4 ile IPv6 arasında bazı farklılıklar vardır. IPv4'ün eksiklikleri ve adres sayısının tükenmesi, IPv6'nın getirdiği yenilikler ve daha geniş bir adres uzayına sahip olması ile giderilmiş durumdadır. Her ne kadar IPv4 ile IPv6 aynı ortamda kullanılabilse de zamanla IPv6'ya tamamen geçilecektir. Aşağıdaki alt başlıklarda IPv4 ve IPv6 arasındaki farklılıklar verilmiştir. Çizelge 2.14'te IPv4 ve IPv6 karşılaştırmaları verilip kısaca açıklanmıştır.

Çizelge 2.14. IPv4 ve IPv6 karşılaştırması [26,33]

Özellik	**IPv4**	**IPv6**
Yayımlanma tarihi	1981	1999
Adres boyutu	32-bit (4 byte)	128-bit (16 byte)
Adres gösterimi	Noktalı Onluk Gösterim 192.146.253.78	Onaltılık Gösterim 3FFE:F200:0234:AB00:0123:7654 :9801:ABCD
Önek gösterimi	192.146.0.0/24	3FFE:F200:0234::/48
Adres sayısı	2^{32} =~4 294 967 296	2^{128} = ~ 340 282 366 920 938 463 463 374 607 431 768 211 456
Paket başlığı	Değişken uzunluktadır. İşlenmesi zaman almaktadır.	Sabit boyutta (40-bit) olduğundan daha verimlidir.
Paket boyutu	Maksimum 65536 bit (64KB) olabilir.	Normal paketi 65536 bit (64KB)'dir. Fakat, IPv6'da bulunan ve 4 milyar bit'e kadar ulaşabilen "**jumbogram**" sayesinde yerel ağlarda (LANs) yüksek performansta veri işleme özelliği sağlar.
Adres tahsisi	A, B, C gibi sınıflara göre ayrılmıştır. CIDR kullanılmıştır. Yerel kullanımı sadece bağlantı-hat için kullanılabilir.	IPv4 ile uyumluluğu vardır. Bölgesel adres kayıt sağlayıcılar, aboneler, coğrafi bölgeler ve alt ağlar tarafından hiyerarşik bir şekilde tahsis edilebilir durumdadır. Bağlantı-hat ve mekan tarafından yerel olarak kullanılabilir. Gelecekte kullanım için adreslerin %70'lik bir kısmı ayrılmıştır.
Bölümlendirme	Yönlendiriciler tarafından bir çok adımda bölümlendirme işlemi yapılarak yönlendirmelerde performans düşüşü yaşanmaktadır.	Bölümlendirme işlemi en fazla bir kez ve bilgisayar tarafından MTU değeri belirlendikten sonra yapılır. Böylece yönlendiricinin performansında artış sağlanmaktadır.
Hizmet-Servis kalitesi (QoS)	Hizmet kalitesi IPv4 paketinde tanımlanmış fakat genellikle kullanılmaz.	Akış etiketleme, öncelik, gerçek zamanlı veri ve multimedya yayın desteği ve ek başlıklar ile daha yüksek düzeyde hizmet kalitesi sağlamaktadır.

Çizelge 2.14. IPv4 ve IPv6 karşılaştırması (Devamı) [26,33]

Güvenlik	IP seviyesinde kimlik doğrulama veya şifreleme yapılamamaktadır. IP katmanı güvenlik protokolü "**IPSec**" seçime bağlıdır. Böylece daha üst katmanlarda hizmet inkarı (denial-of-service) ve adres aldatması (spoofing) gibi güvenlik zaafiyetlerine sebep olabilir.	Paket içerisinde kimlik doğrulama, şifreleme ve güvenlik birliği sağlanmaktadır. IP katmanı güvenlik protokolü "**IPSec**" zorunludur.
Yapılandırma yönetimi	Adresleme, elle veya DHCP ile yapılandırma yapılabilir.	Adresleme, elle, otomatik veya DHCP ile yapılabilir. Yerel bağlantı adreslerinde otomatik yapılandırmayı, temel ağlar için durumsuz otomatik yapılandırma sağlamaktadır.
Çoklu konumluluk (multihoming)	Bu özellik mevcut değildir.	IPv6 ağ arayüzlerine birden çok adres atanabilmektedir. Adresler ihtiyaçlara göre güvenlik, güvenilirlik, yük dengeleme ve hizmet kalitesi amacıyla kulanılabilir [33].
Adres türleri	Paketler tekli gönderim (unicast), çoklu gönderim (multicast) ve yayın (broadcast) olarak gönderilebilir.	Paketler tekli gönderim (unicast), çoklu gönderim (multicast) ve herhangi bir alıcılı gönderim (anycast) olarak gönderilebilir. Anycast ile daha az paket trafiği sağlanmıştır.

2.4.1. IPv6 ve IPv4'te yapılan temel değişiklikler

Daha fazla adres

IPv6, 128 bit veya 16 byte adresleme yapma özelliğine sahiptir. 2^{128} adet adresi adresleyebilir. IPv4 adresi 32-bit'e sahiptir ve 2^{32} adet adresi adresleyebilir. IPv6, IPv4'e göre 2^{96} kat daha fazla adres imkânı sağlamıştır. IPv4'te adres sayısı yaklaşık 4,29 milyar iken IPv6'da bu sayı 340 282 366 920 938 463 463 374 607 431 768 211 456 veya 6,65 x 10^{23} demektir. IPv6 da IPv4 gibi her bir ağ aygıtına bir fiziksel bağlantı adresi verilir. IPv6 adresleme ile yer yüzeyinde metre kareye $6{,}02x10^{23}$ adet tekil IP düşmesi sağlanmıştır. Yani 1 $m^{2'}$lik alana yaklaşık bir mol ($6{,}02x10^{23}$) adres düşmektedir. En verimsiz kullanımla bile bu kadar adres uzun bir sürede bitirilemez.

Ek başlıklar ve güvenlik

IPv6'nın en esnek ve ilgi çekici özelliklerinden biri de ek başlıklardır. Ek başlıklar her zaman kullanılmayan sadece seçeneklere ihtiyaç duyulduğunda bir pakete yerleştirilir ve yeni ek başlıklar tanımlanabilir. Temel IP başlığında yer alan sonraki başlık kısmı sayesinde eklenmek istenen ek başlıklar temel başlığa dâhil edilir. Ek başlıkların bazıları sabit bazıları da değişken boyuttadır. Ek başlık kullanmanın temel nedenlerinden biri ekonomi diğeri genişletilebilmedir. IPv4 temel başlığına ek başlık yerleştirilirse bazen ihtiyaç olmadığı halde bu kısımlar datagramda olacaktır. Böylece, fazladan gereksiz veri iletimi ve bant genişliği kullanma söz konusu olacaktır. Örneğin, IPv4 bölmelendirme istenmeyen durumda dahi bölmelendirme bilgisi taşırdı, ama IPv6 gerekirse bunun için ilgili ek başlığa başvurur, gerekmezse kullanmaz. Böylece gereksiz başlıkların datagramdan çıkarılması ile zaman ve bant genişliğinden tasarruf edilmiştir. Performans artışı sağlanmış olacaktır. Bunlar ek başlıkların ne kadar ekonomik olabileceğini ve performans artışı sağlayabileceğini gösterir. IPv4'de sabit bir başlık kullanılmıştır ve herhangi bir değişiklik yapılmak istendiğinde tüm IPv4 başlığı değişir. IPv6'da ise her bir değişiklik yeni bir ek başlık olarak algılanır. Sonraki başlık (Next Header) bölmesi bu işlevi sağlar. Böylece, veri paketine yeni bir özellik eklenmesi durumunda tüm paketi değiştirmek yerine yeni bir ek başlık tanımlamak yeterli olur. Örneğin, datagramın (paketin) tamamı

şifrelenmek istenirse bir adet şifre ek başlığı ile bu işlem yapılır. IPv6'nın bu esnek yapısı sayesinde gerekli görülen başlıkların uygunluğu belirlendiğinde kullanılabilir. Böylece ek başlıkların esnekliği, genişletilebilme ve geliştirilebilme özellikleri sayesinde gerekli zaman ve mekânda kullanılabilir.

Otomatik adres yapılandırma ve gelişmiş adres türleri

IPv6 otomatik yapılandırma yoluyla dinamik adreslemeye imkân tanımaktadır. IPv6 adreslerinde çok basamaklı hiyerarşik bir yapı vardır ve IPv4 adres yapısında bulunan yayın türü (broadcast) adresleme IPv6 da kaldırılmıştır. Bunun yerine ilk kez herhangi bir alıcılı (anycast) adres türü geliştirilmiştir. Bu IP adresine sahip bir grup bilgisayardan gönderenin durumuna göre en uygun bilgisayara yollanır. Çoklu alıcılı yönlendirme ise tarama alanı eklenerek geliştirilir. IPv6 paket yapısında yer alan akış etiketleme sayesinde göndericiye özel veya trafik türüne özel taşıma imkânı sağlanır. Böylece hangi paketlerin, hizmetlerin öncelikli olacağı tanımlanabilir. Örneğin, canlı video yayını, görüntülü konuşma, mobil hizmetler, öncelikli taşıma gibi. IPv4'te yer alan internet kontrol mesaj iletişim kuralı olan ICMP, IPv6'ya uygun olacak şekilde gözden geçirilip ICMPv6 şeklinde tasarlanmıştır [26,32].

2.4.2. IPv4 başlığında yer alan ve IPv6'da kaldırılan alanlar

IPv4 temel başlık yapısında bulunan fakat IPv6 da silinen beş alan vardır [26]. Bu alanlar açıklamalarıyla birlikte aşağıda verilmiştir.

- Başlık uzunluğu (Header length)
- Tanımlayıcı (Identifier)
- Bayraklar (Flags)
- Bölüm katsayısı (Fragment offset)
- Başlık kontrolü (Header checksum)

Başlık uzunluğu (Header length) alanı bir başlıkta sabit bir uzunluğa ihtiyaç duyulmadığından silinmiştir. IPv4'te başlık uzunluğunun minimum değeri 20 byte'dır. Fakat seçenekler kısmı eklenmişse, 4'er byte artarak 60 byte'a kadar

genişleyebilir. Bu yüzden IPv4'te toplam başlık uzunluğu hakkındaki bilgi önemlidir. IPv6'da seçenekler kısmı Ek Başlıklar kısmında tanımlanmıştır [6,26].

Tanımlayıcı (Identifier), bayraklar (flags) ve bölüm katsayısı (fragment offset) alanları IPv4 başlığında bir paketin bölümlendirme (Fragmentation) işlevini yerine getirirler. Bölümlendirme işlemi sadece daha küçük boyuttaki paketleri destekleyen ağ üzerinde büyük bir paket gönderildiğinde başlar. IPv6'da herhangi bir bilgisayar veya ağ aygıtı bir paketi bölümlendirmek isterse, IPv6 başlık yapısında bulunan Ek Başlıklar kısmını kullanarak bu işlemi yapar. Tanımlayıcı, Bayraklar ve Bölüm katsayısı alanlarının IPv6 başlık yapısından silinmesinin sebebi gerektiğinde ek başlıklar kısmına eklenebilmesidir [6,26].

Başlık kontrolü (Header checksum) alanı işlem hızını arttırmak için IPv6 başlık yapısından çıkarılmıştır. Transfer katmanında yer alan UDP ve TCP protokollerinin zaten kontrol alanına sahip olmaları Başlık Kontrolü alanının çıkarılmasına sebep olarak gösterilmektedir. IPv4'te UDP kontrol alanı isteğe bağlıydı. Fakat IPv6 da UDP kontrol alanı zorunlu hale getirilmiştir. IP en iyi ölçülerde aktarım sağlayan bir protokoldür. Bütünlüğü sağlamak üst katman protokollerini sorumluluğundadır [6,26].

Ayrıca, IPv6'da protokol türü ve yaşam süresi alanlarında bazı değişiklikler yapılarak yeniden adlandırıldı. Bu alanların yerine *akış etiketi* alanı eklenmiştir.

Veri boyutu sabitlendiğinden bölümlendirme ihtiyacı ortadan kalkmıştır. İhtiyaç durumunda IPv6'da yer alan ek başlıkla bu sağlanabilmektedir.

TCP ve eternet zaten hata kontrolü yapmaktadır bu nedenle IP'den hata kontrolü özelliği çıkarılarak paketin hızlanması hedeflenmiştir.

IPv6'da başlık boyutu 40 byte ile sabitlenip gerekli eklemeler ek başlıklara bırakılmış ve temel başlıktan seçenekler çıkarılarak gerektiğinde kullanılabilecek yapıya kavuşturulmuştur [6,26].

3. IPv4'TEN IPv6'YA GEÇİŞ AŞAMASI VE YÖNTEMLERİ

IPv6'ya geçiş aşaması devam ediyor ve edecektir. Mevcut IPv4 adreslerin en son beş bloğunun da dağıtılmasıyla elde hiç IPv4 adresi bloğu kalmamıştır [3,4]. DHCP ve NAT gibi çözümler bir nebze de olsa IPv4 sayısının hızla azalmasını engellemiştir. Fakat bunların da artık yeterli olmadığı görülmüştür. İnternetin temelinde büyüme ve internet protokolünün işlevsel olmasının yanında IPv6'nın geliştirilmesi internetin hızla büyümesine katkıda bulunacaktır. Ayrıca, internet üzerinden yeni uygulamaların yayılmasını ve teknolojik gelişmelere geniş bir alan açmasını sağlayacaktır. Bazı büyük şirketler (Microsoft, Nokia gibi) IPv6 gelişimini hızlandırmak için çeşitli makaleler yayınladılar. Çoğu yeni uygulamalar ve işletim sistemleri, Windows XP, Linux Kernel 2.1.8 ve üzeri, IPv6 işlevlerini, özelliklerini kullanmaya ve desteklemeye başladılar. Fakat etkili, akıcı ve kolay bir IPv4'ten IPv6'ya geçişin sağlanabilmesi için bazı zorlukların olduğunu görmek ve bu zorluklara çözümler sunmak gerekir. IP'nin bulunduğu katman önemli bir yapıya sahiptir. Bu yüzden uygulama katmanında yapılan değişiklikler hemen hayata geçirilebildiği halde IP'nın bulunduğu OSI ağ katmanındaki değişiklikler hemen hayata geçirilemez.

Bu kadar hızlı gelişen bir teknolojik altyapıya paralel olarak teknolojiyi kullanan insan sayısı da gün geçtikçe artmaktadır. İnternetin evrensel bir şekilde kullanılabilmesi ve daha fazla kullanıcıya hizmet verebilmesi için IPv6 tek çözüm olarak gösterilmektedir.

3.1. IPv6'ya Geçiş Yöntemleri

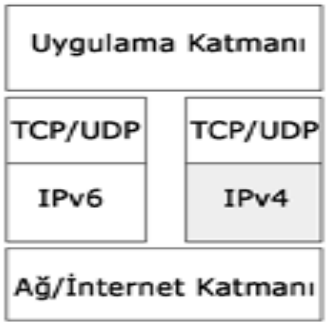
IPv6 geçiş yöntemleri veya mekanizmaları, IPv4 altyapısından yeni nesil adresleme sistemi olan IPv6'ya geçişi kolaylaştıran teknolojilerdir. Özellikle insanlar şimdi IPv4 ağları üzerinden nasıl IPv6 ağlarını kullanabileceklerinin yollarını arıyorlar. Aslında IPv4'ten IPv6'ya geçişten kasıt IPv4/IPv6 ikilisini aynı anda kullanabilmektir. IPv4 adres sayısı bitmektedir fakat elimizde bulunan IPv4 adresleri uzun süre kullanılacaktır çünkü mevcut sistemler bu adres üzerine kurulmuştur. Bu bölümde IPv4'ten IPv6'ya geçiş için gerekli olan geçişler ve aşamalar

anlatılmaktadır. IPv4 ve IPv6 ağlarını birlikte kullanabilmek için geniş bir aralığa sahip yöntemler-mekanizmalar vardır. Ayrıca IPv4'ten IPv6'ya geçiş yöntemi olarak kabul edilmeyen yalın IP (IPv6) yöntemi de bulunmaktadır. [9,10,11,26]. Bu geçiş yöntemlerini veya mekanizmalarını üç ana kategoriye ayırabiliriz.

- İkili yığın yöntemleri/mekanizmaları (Dual stack mechanisms)
- Tünelleme yöntemleri/mekanizmaları (Tunneling mechanisms)
- Çeviri yöntemleri/mekanizmaları (Translation mechanisms)

3.1.1. İkili yığın yöntemleri/mekanizmaları (Dual stack mechanisms)

İkili yığın mekanizması hem IPv4 hem de IPv6 protokollerini aynı anda desteklerler. Böylece bir ağ aygıtı hem IPv4 hem de IPv6 paketini gönderme ve alma özelliklerine sahip olacaktır. Başka bir ifadeyle, bu mekanizma aynı cihaz üzerinde hem IPv4 hem IPv6 adresini destekleme özelliğine sahiptir. Bu sebeple ikili yığın mekanizmasını kullanan cihazların yalın IPv4 veya yalın IPv6 kullanan cihazlara göre daha düşük performans göstereceği belirtilmiştir [9,34]. Şekil 3.1'de ikili yığın mekanizmasının altyapısı gösterilmiştir. IPv6 destekleyen uygulamaların çoğu bağlanmak istedikleri noktanın ilk olarak IPv6 adresi olup olmadığını kontrol ederler. IPv6 adresi var ise IPv6 üzerinden bağlanır IPv6 adresi olmaması durumunda ise IPv4 adresi üzerinden bağlantı sağlarlar. IPv4 ve IPv6 protokol yapıları genellikle birbirinden bağımsızdırlar. Mantıksal ara yüzleri ayrı numaralara sahip ve ayrı hizmet verebilirler. IPv4 ve IPv6 adreslerine sahip bir aygıt, alıcı aygıtın hangi tür IP adresine sahip olduğunu ya IPv6 bağlantısının varlığı ya da DNS (Domain Name System) kayıtları sayesinde anlayarak ona göre paket yollayacaktır [9,11,34].



Şekil 3.1. İkili yığın mekanizması

İkili yığın yönteminin en çok söz edilen iki dezavantajı vardır.

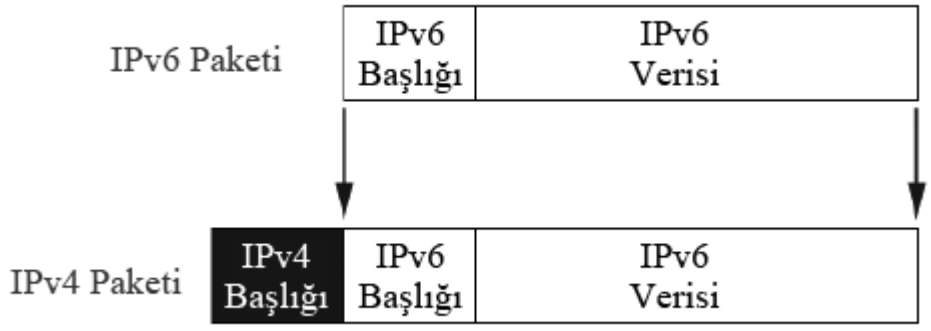
Birincisi, her bilgisayara veya ağ aygıtına verecek IPv4 adresine ihtiyaç olmasıdır. Buna çözüm sunmak için DSTM (Dual Stack Transition Mechanism) metodu geliştirilmiştir. DSTM, IPv4 adres havuzunda bulunan adreslerin ihtiyaç duyulduğunda bilgisayar veya ağ aygıtlarına geçici olarak verilmesi mekanizmasıdır. Böylece daha fazla ikili yığına sahip yapılandırılmış ara yüzler ve daha az IPv4 adres kullanılmış olacaktır.

İkincisi, IPv4 ve IPv6 ikilisi ile iki adet yönlendirme tablosu ve işlemi gerekecektir. Bu da performans, hız ve zaman kaybına neden olabilir.

İkili yığın mekanizmasını destekleyen bilgisayar ve yönlendiriciler IPv6 trafiğini IPv4 ağları üzerinden yönlendirmek için aşağıda bahsedilecek tünelleme mekanizmasını da kullanabilirler.

3.1.2. Tünelleme yöntemleri/mekanizmaları (Tunneling mechanisms)

Tünelleme mekanizması, IPv6 trafiğini mevcut yönlendirme alt yapısını kullanarak IPv4 üzerinden taşıma olanağı sağlar. Çünkü IPv6 mevcut IPv4 alt yapısı üzerinden geliştirilmeye başlanmıştır. IPv6 paketlerinin IPv4 alt yapısı üzerinden tünellenmesi IPv6 paketlerinin IPv4 paketleri içerisine kapsüllenmesi ile yapılır [10,34,35]. Şekil 3.2'de IPv6 paketlerinin IPv4 içerisine kapsüllenmesi gösterilmiştir.

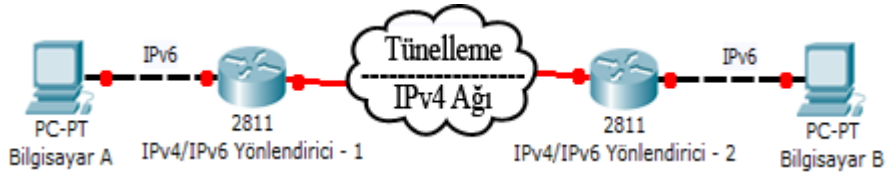


Şekil 3.2. IPv6 paketinin IPv4 içerisine kapsüllenmesi [34]

IPv6 paketlerinin IPv4 içerisine kapsüllenmesi dört farklı yolla yapılabilir [34];

Yönlendiriciden-yönlendiriciye (Router-to-router)

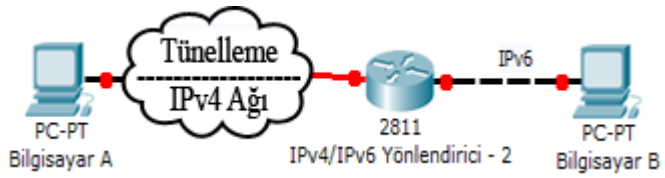
IPv4 ağ alt yapısı, Şekil 3.3'te gösterilen IPv6 trafiğini birbirine bağlı ikili yığın mekanizmasına sahip yönlendiriciler arasına tünelleme yapacaktır [34].



Şekil 3.3. Yönlendiriciden yönlendiriciye kapsüllenme

Bilgisayardan yönlendiriciye (Host-to-router)

İkili yığın mekanizmasına sahip (IPv6/IPv4) bilgisayarlar, Şekil 3.4'te gösterilen IPv6 trafiğini ikili yığın mekanizmasına sahip yönlendiricilerden birine IPv4 ağ alt yapısı aracılığıyla tünelleme yapacaktır [34].



Şekil 3.4. Bilgisayardan yönlendiriciye kapsüllenme

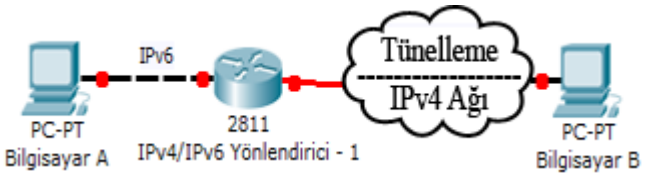
Bilgisayardan bilgisayara (Host-to-host)

IPv4 ağ alt yapısı, Şekil 3.5'te gösterilen birbirine bağlı ikili yığın mekanizmasına sahip bilgisayarlar arasında IPv6 tünelleme işlevini görecektir [34].

Şekil 3.5. Bilgisayardan bilgisayara kapsüllenme

Yönlendiriciden bilgisayara (Router-to-host)

İkili yığın mekanizmasına sahip (IPv6/IPv4) yönlendiricilerden biri tünellemeyi kullanarak, Şekil 3.6'da gösterilen ikili yığın mekanizmasına sahip bilgisayarlara ulaşacaktır [34].



Şekil 3.6. Yönlendiriciden bilgisayara kapsüllenme

Tünelleme, uyumsuz ağların birbirleri ile iletişim kurmasını sağlar. Tünellemenin iki temel amacı vardır [34];

1. Uç düğümlerde bulunan ikili yığın yönlendiriciler aracılığıyla ağlar arası bağlantı sağlamak,
2. Ağ düğüm aygıtlarını farklı ağlarla iletişim kurabilecek hale getirmek.

Tünelleme mekanizması genel anlamda aşağıdaki türlerden oluşur [34];

- Elle yapılandırılmış tünelleme
- Otomatik tünelleme

Elle yapılandırılmış tünelleme

IPv6 paketlerinin IPv4 içerisine kapsüllenmesi yollarından yönlendiriciden yönlendiriciye (router-to-router) ve bilgisayardan yönlendiriciye (host-to-router) şekillerinden de görülebileceği gibi IPv6 trafiğinin, ikili yığın mekanizmasına (IPv6/IPv4) sahip IPv6 paketlerinin ayrıştırılması ve son hedef adreslerine ulaştırılmasından sorumlu yönlendiricilerden birinin son noktaya tünellenir [34]. Yönlendiricinin IP adresinin son alıcının IP adresi ile aynı olmamasından dolayı tünellemenin elle yapılandırılması zorunludur. Yapılan bu yapılandırmaya elle

yapılandırma denir. İki ucun bitişi de elle ayarlandığı için sıkıntı vericidir, fakat kolaydırlar. Tünellenmiş paketlerin ağ üzerindeki yolları tahmin edilebilirler.

Otomatik tünelleme

IPv6 paketlerinin IPv4 içerisine kapsüllenmesi yollarından bilgisayardan bilgisayara (host-to-host) ve yönlendiriciden bilgisayara (router-to-host) şekillerinden de görülebileceği gibi IPv6 trafiğinin, ikili yığın mekanizmasına (IPv6/IPv4) sahip IPv6 paketi doğrudan son hedef noktasına/bilgisayara yönlendirilir. Otomatik tünelleme daha önceden bir elle yapılandırma olmadan IPv4 adresinden IPv6 adresinin (::IPv4-adresi) otomatik olarak türetilmesi işlemine denilmektedir [26,34,35]. Elle yapılandırılmış tünellemeden farklı olarak otomatik tünelleme trafiğin yer değiştirmesi gereken ağlar arasında kullanılır. Otomatik tünelleme mekanizması otomatik yapılandırma sağladığı için kolaydır. Fakat her bilgisayar için bir IPv4 adresi gerektirdiğinden uygulanması zordur.

Güncel olarak beş farklı otomatik tünelleme mekanizması tanımlanmıştır.

1. *Otomatik Tünelleme (Automatic Tunnelling):* IPv4 uyumlu adreslerin kullanımı (RFC 2893) [10].
2. *6to4* (Connection of IPv6 Domains via IPv4 Clouds)*:* IPv4 bulutları yoluyla IPv6 alanları bağlantısı olarak da bilinmektedir (RFC 3056) [36].
3. *ISATAP* (Intra-Site Automatic Tunnel Addressing Protocol)(RFC 5214-4124) [13].
4. *6over4* (Transmission of IPv6 over IPv4 Domains without Explicit Tunnels)*:* IPv6'nın IPv4 alanları üzerinden açık tüneller kullanmadan iletimi (RFC 2529) [12].
5. *Teredo* (Tunneling IPv6 over UDP through Network Address Translations (NATs)) (RFC 4380) [14].

Otomatik tünelleme (Automatic tunnelling)

İlk olarak otomatik tünelleme mekanizması "automatic tunelling" diye adlandırıldı. Otomatik tünelleme 6to4 mekanizmasına çok benzerdir, farklı olarak tekli IPv4

adresleme haritaları bir tek IPv6 adresi ile sınırlandırıldığından otomatik tünelleme kadar verimli değildir. Bu IPv6 adresleri "IPv4 Uyumlu Adresler" olarak da bilinirler. IPv6 ön eki /96 olarak gösterilir. Şekil 2.15'te IPv4 uyumlu IPv6 adres yapısı verilmiştir. IETF standartları tarafından resmi olarak onaylanmamıştır.2006 tarihinde kaldırılmıştır. Bu geçiş mekanizmasının artık kullanılmayacağı düşünülmektedir [26,34,35].

6to4 (Connection of IPv6 domains via IPv4 clouds)

6to4 site içerisinde otomatik tünellemeye izin vermesi hariç ISATAP'a benzerdir. Geçerli ve yönlendirilebilir IPv4 adresi otomatik olarak 6to4'ün başına 16 bit ve değeri 2002 olan adresler oluşturulur. Çizelge 2.13'te bu tünelleme mekanizmasının alacağı değerler verilmiştir. Herhangi bir 6to4 uyumlu sistem diğer bir 6to4 uyumlu sisteme bir paket göndermek istediği zaman, IPv4 paketinin içerisine IPv6 paketini kapsüller ve IPv4 adresi ile kodlanmış paket ile bu paketi 6to4 hedef adresine yönlendirir. Paket karşılandığında hedef IPv4 bilgisayarı IPv4 başlığını kaldırır ve işlem IPv6 paketinden devam eder. 6to4 ve normal IPv6 internet arasındaki iletişim aktarmalarla – yayınlarla kolaylaştırılır. 6to4 herhangi bir alıcılı sabit bir adres ile çalıştırmak mümkündür [26,34,35]. RFC 3056 belgesinde detaylı bilgileri verilmiştir.

ISATAP (Intra-site automatic tunnel addressing protocol)

ISATAP olarak yapılandırılmış düğümleri birbirine bağlamak için kullanılır. Benzersiz yerel-hat belirleyicilerini oluşturmak için özel bir IPv4 adresi ile birlikte çalışır. Bu tünelleme mekanizmasının en önemli işlevi otomatik tünelleme ile IPv4 üzerinden IPv6 paketlerini iletmektir. ISATAP adreslerinin son 32 bit'lik kısmı IPv4 adreslerini barındırır. Adres gösterimi şu şekildedir;

<ISATAP değişken önek/64>:0:5efe:<IPv4 adresi>

Değişken ISATAP öneki 64 bit, 0000:5EFE değeri 32 bit ve IPv4 adresi 32 bit olarak kullanılır.

Yönlendirme kararları ara yüz tanımlayıcısı ve kullanılan öneke bağlıdır. Bu mekanizmanın sağladığı kolaylıklar vardır. Kolay yayılması ve IPv4 özel adreslerini kullanma olanağı sağlayan platformlar tarafından desteklenmesi avantajları arasında yer alır [26,34,35].

6over4 (Transmission of IPv6 over IPv4 domains without explicit tunnels)

IPv4 uyumlu IPv6 adresleri olarak da bilinen 6over4, var olan IPv4 adreslerinden IPv6 adreslerini oluşturmak için kullanılır. Bu yaklaşımda IPv4 çoklu gönderim grupları kullanılarak IPv4 adresleri bir sanal bağlantı katmanı adresi olur. Komşu keşfi (Neighbour Discovery) IPv6 çoklu alıcılı adreslerini IPv4 çoklu alıcılı adreslerine haritalandırarak başlar. IPv4 çoklu alıcılı yönlendirmesinin olabilmesi için yönlendiricinin 6over4 olarak yapılandırılması zorunludur [35]. Bu çalışmanın ikinci bölümünde IPv6 özel adresler kısmında IPv4 uyumlu IPv6 adresleri ve yapısı açıklanmıştır. Detaylı bilgi için buraya bakılabilir (Şekil 2.15). RFC 2529 belgesinde detaylı bilgileri verilmiştir [12].

Teredo (Tunneling IPv6 over UDP through network address translations (NATs))

Teredo, ağ aygıtlarına bir veya birden çok NAT katmanı üzerinden IPv6'yı kullanabilme desteği vermek için UDP üzerinden paketlerin tünellenmesi için tasarlanmıştır. Teredo'nun arkasındaki fikir, IPv6'dan IPv4'e ağ adres çeviricilerinin ardındaki ağ aygıtına (host) izin vermektir. Bazı Teredo uygulamaları geçerli olmalarına rağmen, protocol bir şekilde akış halindedir ve gerekli sunucu aktarma altyapısı daha gerçekleşmemiştir [34,35]. Çizelge 2.13'te gösterildiği gibi Teredo adresleri için `2001:0000::/32` IPv6 dağılımı verilmiştir. RFC 4380 belgesinde detayları verilmiştir [14].

3.1.3. Çeviri yöntemleri/mekanizmaları (Translation mechanisms)

IPv4 / IPv6 geçişlerinde kullanılan çeviri mekanizmaların temel amacı IP paketlerini çevirmektir. TCP/IP referans ve OSI modelinde çeşitli katmanlarda yer alan çeviri mekanizmaları vardır. Bunlar bulundukları katmanlara göre şu şekildedir [34];

- Ağ katmanı (Network layer)
  - *Stateless IP/ICMP Translation Algorithm (SIIT)* - RFC 2765
  - *Network Address Translation - Protocol Translation (NAT-PT)* - RFC 2766
  - *Bump In the Stack (BIS)* - RFC 2767
- Taşıma-Ulaşım katmanı (Transport layer)
  - *Transport Relay Translator (TRT)* - RFC 3142
- Uygulama katmanı (Application layer)
  - *Bump In the API (BIA)* - RFC 3338

SIIT (Stateless IP/ICMP translation algorithm)

SIIT algoritması RFC 2765 belgesinde tanımlanmıştır. ICMPv4 ve ICMPv6 başlıkları arası ile IPv4 ve IPv6 paket başlıkları arasında çift yönlü çeviri yapan algoritmadır. Sadece IPv6 destekli düğümler ile sadece IPv4 destekli düğümler arasında çeviri yapma özelliğine sahiptir. Bu algoritma ile TCP ve UDP başlıklarının çeviri işleminden etkilenmemesi hedeflenerek tasarlanmıştır. SITT algoritması üzerine kurulan BIS (Bump In the Stack) ve NAT-PT (Network Address Translation-Protocol Translation) çeviri metodları vardır. OSI ve TCP/IP referans modelinde ağ katmanında yer aldıkları için ağların birbiriyle çalısmasını destekeleyebilir. Bu çeviri metodunun geniş alanlarda yönetimi zordur [26,34].

NAT-PT (Network address translation-protocol translation)

Bu çeviri mekanizması RFC 2765 belgesinde tanımlanmıştır. NAT-PT, IPv4 adreslerine karşılık gelen IPv6 adresi bilgilerini bir sunucu aracığıyla saklar. Ağ trafiğinin tamamı bu sunucu üzerinde gönderilerek IPv4 adreslerinin doğru IPv6 adreslerini almaları sağlanarak uygun bir şekilde haberleşme sağlanır. SIIT algoritması kullanılarak paketlerin tespiti yapılır [26,34].

BIS (Bump In the Stack)

BIS çeviri mekanizması RFC 2767 belgesinde tanımlanmıştır. IPv6 ağları üzerinden IPv4 paketlerinin haberleşmesini sağlar. Ağa gönderilen paketlerin hangi IP türünde olduğu belirlenir. IPv6 adresleri IPv6 modüllerinde çeviri işleminden geçirilerek işlenir. IPv4 adresleri ise doğrudan TCP/IP protokolü aracılığıyla işlenir [26,34].

TRT (Transport Relay Translator)

TRT çeviri mekanizması RFC 3142 belgesinde tanımlanmıştır. IPv6 ağları tarafından kullanılan TCPv6 / UDPv6 oturumları ile IPv4 ağları tarafından kullanılan TCP / UDP oturumlarının haberleşmesini sağlamak için OSI taşıma-ulaşım katmanında çeviri işlemini yapar. IPv6 ve IPv4 bağlantılarını sağlayan bir ikili yığın yönlendirici üzerinde TRT sistemi kurulabilir. Ipv6 bağlantısından gelen IPv6 adresi yönlendirici üzerinde bulunan TRT tarafından IPv4 adresine çevrilerek iletişim sağlanır [26,34].

BIA (Bump In the API)

BIA çeviri mekanizması RFC 3338 belgesinde tanımlanmıştır. BIA, BIS metodunda olduğu gibi IPv6 ağları üzerinden IPv4 bilgisayarlarının haberleşmesini sağlamaktadır. BIS'ten farkı OSI modelinin katmanında bulundukları konumdur. BIS bulunduğu ağ katmanı konumu itibariyle IP paketlerinin çevrilmesini doğrudan sağlarken, BIA uygulama katmanında bulunduğundan IPv4 API ve IPv6 API'leri arasında çeviri işlemini yapar [26,34].

4. IPv6 DESTEKLİ DONANIM VE YAZILIMLAR

Mevcut sistemlerde IPv6 desteği veren yazılım ve donanımlar hakkında bilgiler aşağıdaki başlıklar altında çizelgeler içerisinde verilmiştir. Sıralama programlama dilleri, işletim sistemleri, yazılımlar ve donanımlar şeklinde belirlenmiştir.

4.1. Programlama Dillerinde IPv6 Desteği Durumu

Günümüzde çeşitli yazılımları geliştirmek için kullandığımız programlama dilleri artık IPv4'ün yanında IPv6'ya destek vermektedirler. En çok kullanılan programlama dillerinin IPv6 desteği hakkında bilgiler Çizelge 4.1'de verilmiştir.

Çizelge 4.1. Programlama dilleri IPv6 desteği durumu [37]

Programlama dili	**Açıklama**	**IPv6 destek durumu**
C /C ++	getaddrinfo() Single unix specification version 3 tarafından bir standart haline getirildi. Bu standart POSIX 1003.1g-2000 ve RFC 2353 belgelerinde yayımlandı. 1999 tarihinden itibaren Unix tarafından kullanılmaktadır. Ayrıca, gethostbyname(), gethostbyname2(), gethostbyaddr(), getipnodebyaddr(), ve getipnodebyname() standartları da kullanılmaktadır.	Destekliyor
Perl	Perl programlama diline ait socket6 veya IO::Socket::INET6 modüllerinin kullanılması gerekir. Böylece, getaddrinfo(3) ve getnameinfo(3) kütüphane fonksiyonlarına soket tabanlı bağımsız uygulamaların geliştirilmesi olanağı sağlanıyor. Perl programlarının IPv6 desteğinin sağlanması zor olduğundan bu modüllere ihtiyaç duyulmuştur.	Destekliyor
Python	Python 2.6 ve üzeri sürümlerde IPv6 desteğini socket.create_connection() metoduyla sağlamıştır.	Destekliyor
Java	Java 1.4 sürümünden beri IPv6 desteği vermektedir. İkili yığın mekanizmasını kullanarak bu desteği sağlamıştır.	Destekliyor
Microsoft Platform Software Development Kit (SDK)	Ocak 200 ve sonrası sürümlerde IPv6 destekli Winsock programlarını geliştirme olanağı sağlamıştır.	Destekliyor
Microsoft Visual C++	Microsoft Visual C++ sürüm 6 ve üzeri ile IPv6 destekli yazılımlar geliştirilebiliniyor.	Destekliyor

4.2. İşletim Sistemlerinde IPv6 Desteği Durumu

Günümüzde en çok kullanılan işletim sistemleri artık IPv4'ün yanında IPv6'ya destek vermektedirler. Günümüzde en çok kullanılan işletim sistemlerinin IPv6 desteği hakkında bilgiler Çizelge 4.2'de verilmiştir.

Çizelge 4.2. İşletim sistemleri IPv6 desteği durumu [37]

İşletim sistemi	**Açıklama**	**IPv6 destek durumu**
Linux (Redhat, Ubuntu, Centos, Debian v.b.)	Linux kernel 2.2 ve sonraki sürümlerde	Destekliyor
Sun Solaris	Solaris 8.9 ve sonraki sürümlerde	Destekliyor
Mac OS X	Mac OS X 10.4.8 ve sonraki sürümlerde	Destekliyor
FreeBSD	FreeBSD 6.1 ve sonraki sürümlerde	Destekliyor
HP-UX 11i	HP-UX 11i ve ayrıca HP tarafından desteklenen diğer işletim sistemlerinin tümü 21 Mart 2011 itibariyle	Destekliyor
NetBSD	NetBSD 1999 yılından itibaren IPv6 destekliyor. NetBSD 1.4.1 ve sonraki bütün sürümlerinde IPv6 desteği vermektedir.	Destekliyor
OpenBSD	NetBSD 1999 yılından itibaren IPv6 destekliyor. NetBSD 1.4.1 ve sonraki bütün sürümlerinde IPv6 desteği vermektedir.	Destekliyor
IBM AIX	IBM AIX 5 ve sonraki sürümler	Destekliyor
Windows	Microsoft Windows XP SP2 ve sonrası Windows Vista RC1 ve sonrası Windows 2000 ve sonras Windows Server 2003 ve sonrası Windows 7	Destekliyor

4.3. Yazılımlarda IPv6 Desteği Durumu

Günümüzde en çok kullanılan yazılımlar artık IPv4'ün yanında IPv6'ya destek vermektedirler. Bahsedilen yazılımların türlerine göre IPv6 desteği hakkında bilgiler Çizelge 4.3'te verilmiştir.

Çizelge 4.3. Yazılımların IPv6 desteği durumu [37]

Yazılımlar	**Türü**	**Açıklama**	**IPv6 durumu**
Internet Explorer 6 Safari Firefox 2 Opera 7.20 Google Chrome Kongueror 3.1.2 Netscape Navigator 7.1	Web Tarayıcı	Belirtilen ve sonraki sürümlerde IPv6 desteği sağlanmıştır.	Destekliyor
Apache 2.x IIS 6 Lighttpd 1.4.27 Webfs 1.19	Web Sunucu	Belirtilen ve sonraki sürümlerde IPv6 desteği sağlanmıştır.	Destekliyor
Solidpop3d 0.15 Courier-pop3d 0.42.2 Courier-imapd 0.42.2 Dovecot 0.99.10.6	Posta Sunucu	Belirtilen ve sonraki sürümlerde IPv6 desteği sağlanmıştır.	Destekliyor
Thunderbird 2 Windows Mail (XP ve Vista üzerinde) Windows Live Mail (XP ve Vista üzerinde) Apple Mail App Microsoft Outlook 2007 Pine 4.62 KMail 3.1.2	Posta İstemcileri	Belirtilen ve sonraki sürümlerde IPv6 desteği sağlanmıştır.	Destekliyor
Pure-ftpd 1.0.14 ProFTPD 1.2.9rc2 Vsftpd 2.0.0 Wzdftpd 0.3.3	FTP Sunucu	Belirtilen ve sonraki sürümlerde IPv6 desteği sağlanmıştır.	Destekliyor
Ftp 0.17-51.fc12 lftp 2.6.5 fget 0.4.1 kongueror 3.1.2 FileZilla 3.3.5.1	FTP İstemci	Belirtilen ve sonraki sürümlerde IPv6 desteği sağlanmıştır.	Destekliyor
Bind 8 PowerDNS 3.3.1	DNS Sunucu	Belirtilen ve sonraki sürümlerde IPv6 desteği sağlanmıştır.	Destekliyor
Google Arama Facebook CNN	Web Sitesi	http://ipv6.google.com/ http://www.v6.facebook.com/ http://ipv6.cnn.com/	Destekliyor [42,43]

Burada Google, Facebook ve CNN gibi kuruluşların artık web hizmetlerini IPv6 destekli hale getirmeleri büyük bir önem arz etmektedir. Özellikle Google gibi çeşitli hizmetler sunan büyük bir kuruluş IPv6 üzerinden arama hizmeti sunuyorsa IPv6'nın geleceğimizi ne kadar etkileyeceği açıktır.

4.4. Donanımlarda IPv6 Desteği Durumu

Günümüzde en çok kullanılan donanımlar artık IPv4'ün yanında IPv6'ya destek vermektedirler. Bahsedilen donanımlar ve IPv6 desteği hakkında bilgileri Çizelge 4.4'te verilmiştir.

Cisco dünya pazarında ağ aygıtları, hizmetleri ve çözümleri konusunda en büyük paya sahiptir. Bu yüzden Cisco tarafından geliştirilen cihazlar ve yazılımlar artık IPv6 destekli olarak piyasaya sürülmektedir. Sadece donanımsal anlamda baktığımızda yeni üretilecek her aygıtta IPv6 desteği olacağı açıktır.

Türkiye, "Kamu Kurum ve Kuruluşları için IPv6'ya Geçiş Planı" konulu Başbakanlık Genelgesi 8 Aralık 2010 tarihli ve 27779 sayılı Resmi Gazete'de yayımlamıştır. Bu genelgede yeni alınacak donanımların artık IPv6 destekli olması gerektiğinden açıkça bahsedilmiştir [38].

Çizelge 4.4. Donanımların IPv6 desteği durumu [37]

Donanım	**Türü**	**Açıklama**	**IPv6 durumu**
Cisco	Yönlendirici (Router)	Cisco CRS-1 Cisco 12000 Series Cisco 10720 Series Cisco 10000 Series Cisco ASR 1000Series Cisco 10000 Series Cisco 7600 Series Cisco 7500 Series Cisco 7304 Cisco 7301 Cisco 7200 Series Cisco AS5850 Cisco AS5400 Cisco AS5350 Cisco 4000 Series Cisco 3800 Series Cisco 3700 Series Cisco 3600 Series Cisco 3200 Series Cisco 2800 Series Cisco 2600 Series (2691 HARİÇ) Cisco 2500 Series Cisco 1800 Series Cisco 1700 Series Cisco 870 series Cisco 860 series Cisco Catalyst 6500Series Cisco Catalyst 4500 Series Cisco Catalyst 3750 ve 3750-E Series Cisco Catalyst 3560 ve 3560-E Series Cisco UBR 7200 Cisco Gateway GPRS Support Node Release 7.0 (KABLOSUZ AĞ GEÇİDİ)	Destekliyor
Cisco	Anahtar (Switch)	Cisco Catalyst Express 500 Series Switch Cisco Catalyst 2900XL Series Switch Cisco Catalyst 2960 Series Cisco Catalyst 3500XL Series Switch Cisco Catalyst 3560, 3560-E, 3750 ve 3750-E Series Switch Cisco Catalyst 4500 Series Switch Cisco Catalyst 4500-E Series Switch Cisco Catalyst 5000 Series Switch Cisco Catalyst 6500 Series Switch Cisco Nexus 7000	Destekliyor

Çizelge 4.4. Donanımların IPv6 desteği durumu (Devamı) [37]

Cisco	Güvenlik duvarı (Firewall)	Cisco IOS Software IPv6 firewall Cisco PIX Firewall Cisco ASA 5500 Series Adaptive Security Appliances Cisco Catalyst 6500/7600 Series Firewall Services Module (FWSM)	Destekliyor
Alcatel	Yönlendirici (Router)	Alcatel-Lucent 7710 SR Alcatel-Lucent 7750 SR Alcatel-Lucent 7670 Routing Switch Platform	Destekliyor
Alcatel	Anahtar (Switch)	Alcatel-Lucent OmniSwitch 6400 Alcatel-Lucent OmniSwitch 6850 Alcatel-Lucent OmniSwitch 9000	Destekliyor
Juniper	Yönlendirici (Router)	T Series Core Routers: T320, T640, T1600, TX MATRIX, ve TX MATRIX Plus E Series Broadband Services Routers: ERX310, ERX700 SERİSİ, ERX1410, ERX1440, E120, E320 CTP series circuit to Packet Platforms: CTP 1002, CTP 1004, CTP 1012, CTP 2008, CTP 2024, CTP 2056 J Series Services Routers: J2320, J2350, J4350, J6350, JCS1200 Control System	Destekliyor
3Com	Yönlendirici (Router)	3Com® MSR 50 Series Multi-Service Routers 3Com® MSR 30 Series Multi-Service Routers 3Com® MSR 20 Series Multi-Service Routers	Destekliyor
3Com	Anahtar (Switch)	3Com® Switch 8800 Ailesi 3Com® Switch S7900E	Destekliyor
D-Link	Anahtar (Switch)	DGS-3627G DGS-3650 DGS-3627 DGS-3612G DES-6500	Destekliyor

4.5. IPv6'da Karşılaşılan/Karşılaşılacak Problemler

IPv6 kullanımının yaygın hale gelmesiyle güvenlik, geçiş aşaması, ağ ve sistem yöneticilerinin bilgi eksiklikleri, mevcut yazılımların IPv6 desteğine sahip olmaması gibi problemler ortaya çıkacaktır.

IPv6 ağlarda kullanılmak istenen saldırı tespit ve önleme sistemleri, güvenlik duvarları ve ağ yönetim gibi yazılımların IPv6 desteği olmak zorundadır. Aksi taktirde bu yazılımlar IPv6 ağlarında hizmet veremeyecektir. Bu da güvenlik açısından IPv6 ağlarda sorun teşkil etmektedir.

IPv6'ya geçiş aşamasında özellikle ikili yığın geçiş mekanizmasında aynı anda hem IPv4 hem de IPv6 bulunacağından güvenlik duvarları yazılımlarında tanımlanan kurallar birbirinden farklı olmalıdır. Başka bir ifadeyle IPv4 ve IPv6 güvenlik duvarı kuralları kesinlikle birbirinden farklı olmalıdır. Böyle bir durumda ağ trafiği hızında yavaşlamalar olacaktır. Bu problemin çözümü mevcut güvenlik duvarı uygulamasını IPv6 destekli duruma getirmek yerine IPv6 destekli ayrı bir güvenlik duvarı uygulamasını mevcut ağa yerleştirmektir [26,37,41].

IPv6 gelişen teknoloji için yenilikler sunmaktadır. Fakat, sistem ve ağ yöneticilerinin bilgi eksiklikleri bu yenilikleri kullanma noktasında sıkıntılar oluşturacaktır. Sistem ve ağ yöneticilerinin IPv6 konusunda bilgilendirilmeleri gerekmektedir Ağ yönetiminin en önemli standardı olan SNMP için fazla sayıda geliştirmeler yapılmamıştır. SNMP ile IPv6 ağlarını IPv4 üzerinden gözlemleyebilirsiniz fakat sadece IPv6 destekli ağları gözlemleyemezsiniz [26].

Mevcut uygulamaların bazılarının doğrudan internet katmanı ile bağlantısı var iken bazılarının yoktur. İnternet-ağ katmanı ile doğrudan ilişkisi olmayan uygulamaların IPv4 ve IPv6 ortamlarında hiçbir değişiklik yapmadan çalışırlar. Fakat, internet-ağ katmanı ile doğrudan ilişkili uygulamalarda değişiklikler yapılmalıdır. Protokol bağımsız uygulamalar haline getirilmeldir. Böylece, hem IPv4 hem de IPv6 paketlerini işleyebilir duruma gelecektir ve uygulamaların hizmet verme noktasında sıkıntıları olmayacaktır. İnternet-ağ katmanına bağlı olarak hizmet veren diğer bir ifadeyle soket tabanlı uygulamaların sadece IPv6 destekli ağlarda daha fazla problemleri bulunmaktadır [26,37,41].

Bu çalışmada soket tabanlı bir POP3 uygulamasının IPv6 destekli olarak hizmet vermesi için gerekli düzenlemeler ve geliştirmeler yapılmıştır. Çalışmanın beşinci bölümünde uygulamaya ait bilgiler verilmiştir.

5. IPv6 DESTEKLİ UYGULAMA GELİŞTİRİLMESİ

Mevcut uygulamaların çoğunda IPv6 desteği bulunmamaktadır. Bu uygulamaların IPv6 paketlerinden gelen veriyi işleyebilmeleri için mutlaka IPv6 desteğine sahip olmaları gerekir. Aşağıdaki başlıklarda C programlama dilinde hazırlanan mevcut uygulamaların yeni nesil internet protokolü-IPv6 ile haberleşmeyi sağlayacak geliştirme yöntemleri ve açıklamaları verilmiştir.

5.1. Mevcut Uygulamalara IPv6 Desteği Verilmesi

Günümüzdeki uygulamaların çoğu henüz IPv6 destekler durumda değildir. Bu uygulamalara IPv6 desteği sağlayabilmek için mevcut kodlamalara yamalar yapılmalı veya yeniden hem IPv4 hem IPv6 desteği sağlayan kodlamalar yapılmalıdır. Böylece uygulamalarımız hem IPv4 hem de IPv6 ile çalışabilecek duruma gelirler. IPv4 adresleri uygulamalar tarafından tamsayı (integer) olarak saklanmaktadır. IPv4 adreslerinin 32 bit olmasından dolayı uygulamalarda herhangi bir sıkıntı oluşmamıştır. Fakat yeni nesil internet protokolü - IPv6, 128 bit uzunluğundadır. Uygulamalar tarafından mevcut IPv4 adreslerinde kullanılan tamsayı alanı bu protokol için kullanılamaz. Bu yüzden adreslerin tutulacağı alan IPv4/IPv6 ikilisi için değiştirilmesi gerekir. C programlama dili sistem dosyalarından olan **<sys/socket.h>** dosyasında bu değişiklikler yapılabilir.

Uygulama olarak ikili yığın mekanizmasını desteklemek için öncelikle IPv4 ve IPv6 adreslerini ele almamız gerekir. Uygulamalarımızda hem IPv4 hem de IPv6 adreslerini kullanabilmek için *sockaddrs* yapılarını, örneğin, *sockaddr_in* veya *sockaddr_in6* gibi, kullanmamalıyız. Böylece, adres sınıfı bilgilerini içeren veriler tanımlı adres türlerine göre veriyi taşıyabilirler.

IPv4 soketlerini belirlemek için soket tabanlı uygulamaların programlama ara yüzleri **AF_INET** sabitini kullanarak tanımlarlar. IPv6 soketlerini belirlemek için ise **AF_INET6** sabiti kullanılır. C programlama dili yapısında yer alan IPv4 akranlarını belirlemek için *sockaddr_in* yapısı kullanılır. IPv6 akranlarını belirlemek için ise *sockaddr_in6* yapısı kullanılır.

IPv4'de kullanımı şöyledir; **sDegiskenIPv4 = socket (AP_INET, SOCK_STREAM, IPPROTO_TCP)**

IPv6'da kullanım şekli şöyledir; **sDegiskenIPv6 = socket (AP_INET6, SOCK_STREAM, IPPROTO_TCP)**

Aşağıdaki kodlar *sockaddr_in* ve *sockaddr_in6* yapılarının tanımlamalarını göstermektedir.

IPv4 sockaddr_in tanımlama:

```
struct sockaddr_in {
    u_int8_t sin_len; /* sockaddr uzunluğu */
    u_int8_t sin_family; /* Adres sınıfı */
    u_int16_t sin_port; /* TCP/UDP port numarası */
    struct in_addr sin_addr; /* IPv4 adresi */
    int8_t sin_zero[8]; /* iç dolgu değeri - padding */
};
```

IPv6 sockaddr_in6 tanımlama:

```
struct sockaddr_in6 {
    u_int8_t sin6_len; /* bu yapının uzunluğu (socklen_t) */
    u_int8_t sin6_family; /* AF_INET6 (sa_family_t) */
    u_int16_t sin6_port; /* Taşıma katmanı portu */
    u_int32_t sin6_flowinfo; /* IPv6 akış bilgisi */
    struct in6_addr sin6_addr; /* IPv6 adresi*/
    u_int32_t sin6_scope_id;/*indeks değeri link-site yerel adresler*/
};
```

Yukarıda verilen kodlarda *sockaddr_in6*, *sockaddr_in* ile karşılaştırıldığında *sockaddr_in6* iki farklı alana sahiptir. *sin6_flowinfo* ve *sin6_scope_id*.

Soket tabanlı API (Application Programming Interface) ile IPv6 destekli uygulamalar geliştirilebilir. Bunun için IP adres sınıfını **getaddrinfo** ve **getnameinfo** kullanarak adres türünden bağımsız bir hale getirebiliriz. Uygulamalarımızda adreslerimizi temsil etmek için **sockaddr** yapısını kullanacağımızı belirttik. Bu yüzden Çizelge 5.1'de artık kullanılmayan ve protokollerden bağımsız yeni uygulama programlama ara yüzleri gösterilmiştir [39,40,41]. Geleneksel olarak, sadece IPv4 uyumlu programlar IPv4 adreslerini

in_addr yapısını kullanarak çalıştırırlar. **`in_addr`** veya **`in6_addr`** yapısını kullanan soket API'lerinin kullanılmaması gerekir. Aksi takdirde uygulamalarımız hangi adres türünden veri alacağını anlayamaz ve çalışamaz duruma gelir.

Çizelge 5.1. Artık kullanılmayan ve protokolden bağımsız yeni API'ler [39,40,41]

Kullanılmayan API'ler	**Protokolden bağımsız yeni API'ler**
struct sockaddr_in (adresleri saklamak için)	struct sockaddr_storage
gethostbyname	getaddrinfo
gethostbyname2	
getservbyname (port numarasını yerleştirirken kullanılır)	
gethostbyaddr	getnameinfo
getservbyport (soket adresini alırken kullanılır)	
inet_addr	getaddrinfo (AI_NUMERICHOST) parametresi ile kullanılır
inet_aton	
inet_nsap_addr	
inet_ntoa	getnameinfo (NI_NUMERICHOST) parametresi ile kullanılır
inet_nsap_ntoa	
inet_makeaddr	Kesinlikle bu uygulama programlama ara yüzlerinden kaçınılmalıdır. Artık kullanılmamaktadır.
inet_netof	
inet_network	
inet_neta	
inet_net_ntop	
inet_net_pton	
rcmd	rcmd_af
rexec	rexec_af
rresvport	rresvport_af

5.2. Mevcut Uygulama Kodlarının Düzenlenmesi

Bir programın mevcut kodlarını IPv6 desteği verecek şekilde tekrar yazmak için IPv4 bağımlı veri türlerinin yanında fonksiyonları da bulmamız gerekir. C programlama dili ile genellikle soket tabanlı IPv4 uygulamaları yazılmıştır. Geliştirilecek uygulamada C programlama dili ile yazılacağından bu çalışmada C programlama diline uygun kodlamalardan bahsedilecektir.

Öncelikle *.c ve *.h uzantılı C programlama dilinin kullandığı dosyalarda IPv4 tarafından kullanılan soket tabanlı uygulamaların fonksiyonlarını bulmak gerekir. Örneğin, UNIX tabanlı işletim sistemlerinde (LINUX gibi) IPv6 desteğini vermemiz gereken IPv4 soket dosyalarını şu şekilde arayabiliriz;

- grep gethostby *.c *.h
- grep inet_aton *.c *.h
- grep sockaddr_in *.c *.h
- grep in_addr *.c *.h

Microsoft tarafından geliştirilen ve Microsoft Visual Studio kurulu işletim sistemlerinde *C:\Program Files\Microsoft SDKs\Windows\v6.0A\Bin* klasörü altında yer alan *checkv4.exe*. programı kullanılarak mevcut C programlama dili ile yazılmış IPv4 yazılımlarının IPv6 desteği sağlaması için gerekli kodlamalar hakkında daha detaylı bilgiler alınabilir.

Soket API dosyaları sadece bir *.c (.c uzantılı) dosyadan oluşuyorsa, düzeltilmesi çok kolay olacaktır. Aksi takdirde IPv4 bağımlı verileri kullanan bütün dosyaları bulup protokol bağımsız olacak şekilde düzeltmemiz gerekir. Böyle bir durumda bütün kodları (yapı tanımlarını, fonksiyon parametreleri) bağımsız adres sınıfı yapısına uygun olarak değiştirmemiz gerekecektir. Düzeltmiş olduğumuz uygulamaların/programların UNIX tabanlı sistemlerde çalışabilmesi için bazı RFC belgelerinde açıklamalar yapılmıştır [39,40,41].

UNIX işletim sistemi ve türevlerinde (LINUX gibi) sunucu uygulamalarının çalışabilmesi için iki yol izlenir. Bu yollar;

- İnternet süper sunucu olarak da bilinen ***inetd*** modudur. *Inetd* ağdan gelen talepler doğrultusunda en az kaynak kullanımı ile birçok ağ programını yükler. Inetd(8) olarak da gösterilebilir.
- Tek başına (Stand-alone) çalışan program modudur.

Sunucu tabanlı uygulamaların hem IPv4 hem de IPv6 kullanıcılarının kullanabileceği (yararlanabileceği) bir hizmet sağlamaları için iki tane dinleme soketini açmamız gerekir. Bunlar **`AF_INET`** ve **`AF_INET6`** dinleme soketleridir. Bu soketlerin açılması bir kaç yol ile yapılabilir.

1. Uygulamayı IPv6 destekler hale getirmek. *Inetd(8)*, süper sunucusunu, hem AF_INET hem de AF_INET6 bağlantılarına cevap verebilecek şekilde ayarlanması. Genel olarak Linux ve Unix işletim sistemlerinde bulunan **`/etc/inetd.conf`** dosyasından bu yapılandırma yapılabilir.
2. Çoklu dinleme soketlerini algılayan bir uygulama çalıştırmak. Bunu **`select(2)`** veya **`poll(2)`** fonksiyonlarını-metodlarını kullanarak yapılabilir.
3. Uygulamanın iki durumunu (AF_INET ve AF_INET6) da çalıştırmak. Bu durumlardan birisi IPv4 için AF_INET, diğeri ise IPv6 durumu için AF_INET6'dır.

Bahsi geçen durumlardan 1. ve 2. durumlar RFC belgelerinde önerilmektedir. 3. durumda uygulamada soketler arasında geçiş yapabilmek için komut satırından komutlar verilmelidir. Bu nedenle, uygulamamız adres sınıfından bağımsız olmayacaktır [39,40,41].

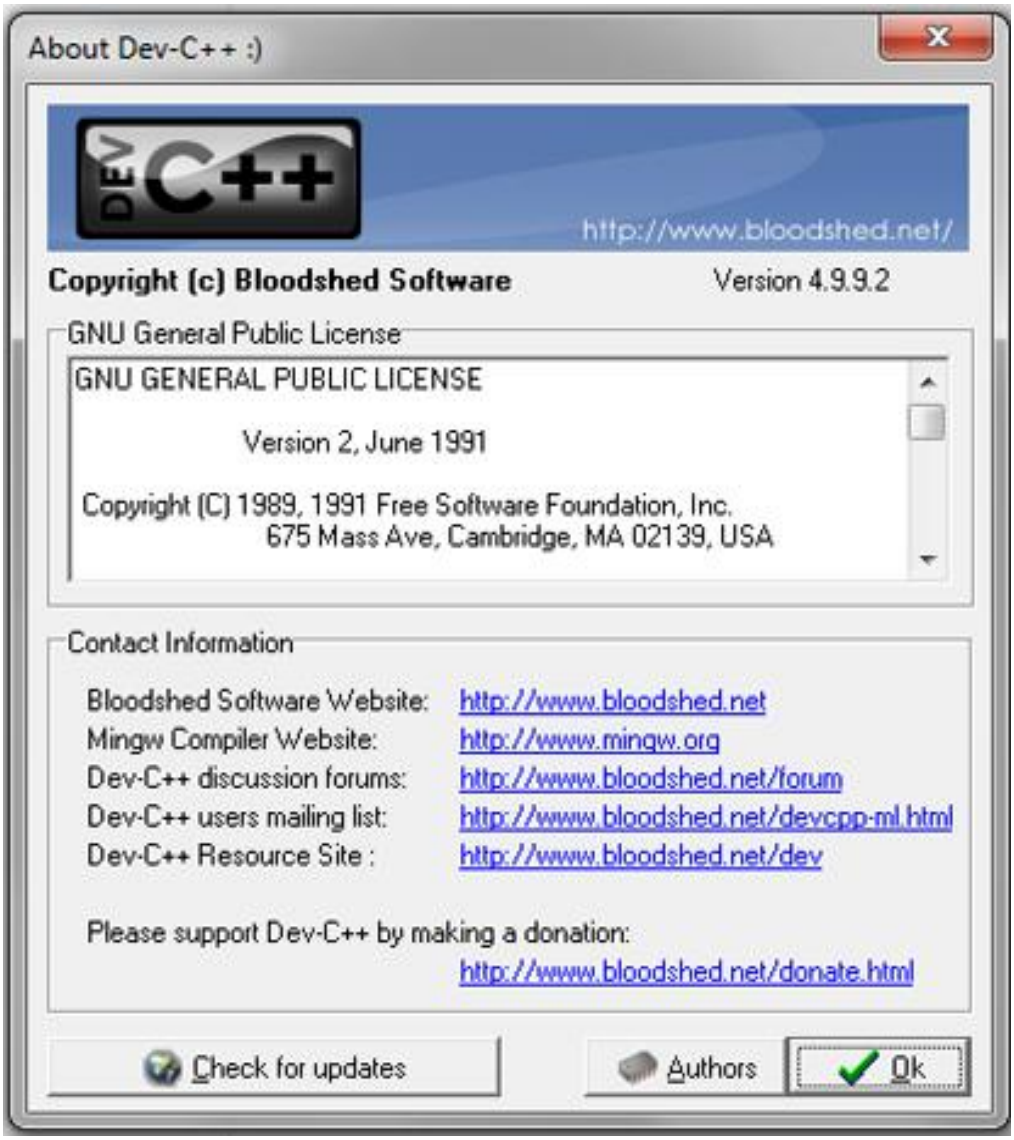
5.3. IPv6 Destekli Sunucu Uygulamasının Geliştirilmesi

Yukarıda verilen bilgiler doğrultusunda POP3 (Post Office Protokol version 3) sunucusunun güvenli, hızlı ve verimli bir şekilde iletişim sağlamasını amaçlayan ve çeşitli platformlarda (FreeBSD, NETBSD, OpenBSD, Debian, Red Hat v.b.) çalışabilen bir sunucu uygulamasının, *popa3d,* IPv6 üzerinden de çalışabilmesi için gerekli düzenlemeler yapılmıştır. *popa3d* özgür yazılım özelliğine sahip, dağıtılabilir bir POP3 sunucu yazılımıdır. Yazılım resmi web sitesinden (http://www.openwall.com/popa3d/) temin edilebilir.

Bu yazılım *inetd (süper sunucu),* ve *tek başına (stand-alone)* gelen çağrılara cevap verebilmektedir. Ek-1'de verilen dosya C programlama dili ile yazılmış ve süper sunucu (inetd)'dan gelen sadece IPv4 ve IPv4/IPv6 ikilisinin çağrılarına cevap verecek kodlamaları görebilirsiniz. Aynı şekilde Ek-2'de C programlama dili ile yazılmış tek başına çalışan (stand-alone) uygulamadan gelen sadece IPv4 ve IPv4/IPv6 ikilisinin dinleme soket çağrılarına cevap verecek kodlamaları görebilirsiniz.

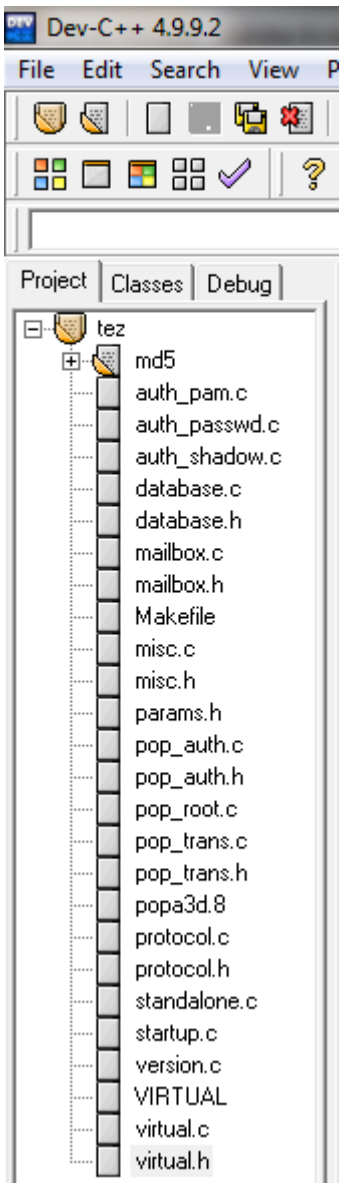
5.3.1. Uygulama geliştirme ortamı

Popa3d uygulamasını resmi web sitesinden (http://www.openwall.com/popa3d/) bilgisayarımıza indirdik. İndirdiğimiz dosya *popa3d-1.0.2.tar.gz* adında sıkıştırılmış halde gelmektedir. Dosyayı WinZIP, WinRAR, 7-Zip gibi bir yardımcı program aracılığıyla açtıktan sonra elimizde programa ait bütün dosyaların yer aldığı bir popa3d-1.0.2 klasörü olacaktır. Artık editörümüz aracılığıyla uygulamayı geliştirmeye başlayabiliriz. Bu uygulama Dev C++ editörü kullanılarak geliştirilmiştir. Şekil 5.1'de kullanılan editör bilgileri verilmiştir. Popa3d sunucu program dosyalarının yeni oluşturduğum tez adlı proje içerisine dâhil edilmesinin ardından editör içerisindeki görünümü Şekil 5.2'de verilmiştir.



Şekil 5.1. Uygulama geliştirme editörü

Öncelikle tez projemizde *.c ve *.h uzantılı dosyalarda IPv4 tarafından kullanılan soket tabanlı uygulamaların fonksiyonlarını bulmak gerekir. Bunun için aşağıdaki komutlar bir Linux türü olan Ubuntu 10.10 maverick sunucusu üzerinden çalıştırılmıştır.



Şekil 5.2. Editörde tez projesinde popa3d dosyalarının görünümü

IPv6 desteğini vermemiz gereken IPv4 soket dosyalarının hangileri olduğunu bulmak için;

- `% grep in_addr *.[ch]`

```
standalone.c: if ((sock = socket(AF_INET, SOCK_STREAM,
IPPROTO_TCPT)) < 0)
standalone.c: addr.sin_family = AF_INET;
virtual.c: if (length != sizeof(sin) || sin.sin_family !=
AF_INET
virtual.c = ) return NULL;
```

- `% grep in_addr *.[ch]`

```
standalone.c: addr.sin_addr.s_addr) struct in_addr addr)
/* Source IP address */
standalone.c: addr.sin_addr.s_addr = inet_addr(DAEMON_ADDR);
standalone.c: addr.sin_addr.s_addr)
standalone.c: inet_ntoa(addr.sin_addr
standalone.c:));
standalone.c: inet_ntoa(addr.sin_addr));
standalone.c: inet_ntoa(addr.sin_addr));
standalone.c: inet_ntoa(addr.sin_addr));
standalone.c: sessions[j].addr = addr.sin_addr;
virtual.c: return inet_ntoa(sin.sin_addr);
```

- `% grep sockaddr_in *.[ch]`

```
standalone.c: struct sockaddr_in addr;
virtual.c: struct sockaddr_in sin;
```

komutlarını yazdıktan sonra kalın harflerle gösterilmiş iki tane dosyanın olduğunu görüyoruz. Bunlar *virtual.c* ve *standalone.c* dosyalarıdır. Uygulamamızın IPv6 desteği verebilmesi için bu iki dosyanın içinde gerekli düzenlemelerin yapılması gerekir. Ayrıca, Microsoft tarafından geliştirilen *checkv4.exe* programını kullanarak uygulamamızda yer alan bütün dosyalar kontrol edilmiştir. UNIX ortamında kontrollerden sonra bulunan *virtual.c* ve *standalone.c* dosyalarının yanında Microsoft tarafından geliştirilen, Microsoft Visual Studio kurulu işletim sistemlerinde *C:\Program Files\Microsoft SDKs\Windows\v6.0A\Bin* klasörü altında yer alan ve Windows ortamında çalışan *checkv4.exe* aracılığıyla bütün uygulama dosyaları kontrol edilmiştir [37-41]. Bulunan *virtual.c* ve *standalone.c* dosyalarına ek olarak değişiklik yapılması gereken diğer dosyalar *params.h* ve *startup.c* dosyalarıdır. Bu dosyaların içeriği kontrol edilip gerekli düzeltmeler yapılmıştır. Aşağıda yapılan değişiklikler hakkında açıklamalar verilmiştir.

"virtual.c" dosyasının düzeltilmesi ve açıklamaları

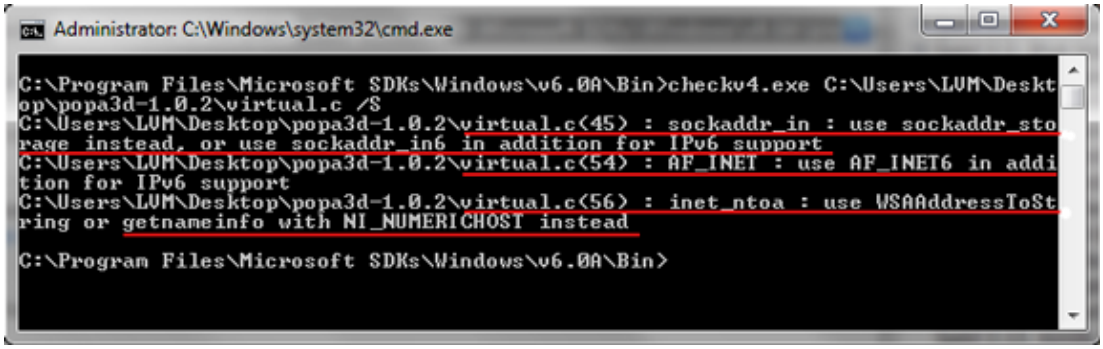
Bu dosya kullanıcı tarafından POP3 sunucunun IP adresine bağlı olarak birbirine bağlı dosyalarını çoklu dosyalara ayırmak için sanal ana dizin oluşturma işlevini sağlar. Dosyanın içerisinde yer alan IPv4 bağımlı olan kod **`lookup()`** fonksiyonu içerisinde yer alır. Şekil 5.3'te bu kodlar gösterilmiştir.

```
static char *lookup(void)
{
    struct sockaddr_in sin;
    socklen_t length;

    length = sizeof(sin);
    if (getsockname(0, (struct sockaddr *)&sin, &length)) {
        if (errno == ENOTSOCK) return "";
        log_error("getsockname");
        return NULL;
    }
    if (length != sizeof(sin) || sin.sin_family != AF_INET) return NULL;

    return inet_ntoa(sin.sin_addr);
}
```

Şekil 5.3. virtual.c dosyasında IPv4 bağımlı kısım

Bu kodların adresten bağımsız olarak çalışması için düzeltilmesi gereken alanlar vardır. Bu alanlar Çizelge 5.1'de gösterilmiştir. Linux (Ubuntu 10.10) ortamında popa3d yazılımının hangi dosyalarında IPv6 desteği verilmesi gerektiğini bulduktan sonra Windows ortamında çalışabilen editörümüz aracılığıyla bu dosyalarımızı düzenleyeceğiz. IPv6 desteği verilmesi gereken dosyalarımızı Microsoft tarafından geliştirilen küçük bir uygulama olan checkv4.exe'yi çalıştırarak daha net bir şekilde anlayabiliriz. Şekil 5.4'te checkv4.exe programı *virtual.c* dosyasında yapılması gereken değişiklikleri listeliyor. Bu değişiklikleri şu şekilde sıralayabiliriz;

- sockaddr_in yerine ***sockaddr_storage*** veya sadece IPv6 desteği için ***sockaddr_in6*** kullanılmalıdır.
- AF_INET yerine **AF_INET6** kullanılmalıdır.
- inet_ntoa yerine ***getnameinfo()*** kullanılmıştır. Çizelge 5.1'de belirtildiği gibi ***getnameinfo (NI_NUMERICHOST)*** şeklinde kullanılmalıdır.



Şekil 5.4. virtual.c dosyasının checkv4.exe ile kontrolü

Bu değişikliklerden sonra *virtual.c* dosyamız Şekil 5.5'te gösterildiği gibi IPv6 desteğine sahip bir dosya olacaktır.

```
static char *lookup(void)
{
    //struct sockaddr_in sin; pasif

    // IBRAHIM AKSIT -> sockaddr_storage -> IPv6 için
    /* struct sockaddr_in sin değeri struct sockaddr_storage sin olarak değiştirildi.*/
    struct sockaddr_storage sin;
    socklen_t length;

   // IBRAHIM AKSIT -> hata ve IPv6 için değişken tanımlandı.
   /* IPv6 adreslerini de almak için sabit bir değişken tanımlandı.*/
    static char hbuf[NI_MAXHOST];
    int hata;

    length = sizeof(sin);
    if (getsockname(0, (struct sockaddr *)&sin, &length)) {
        if (errno == ENOTSOCK) return "";
        log_error("getsockname");
        return NULL;
    }

    // IBRAHIM AKSIT -> getnameinfo -> IPv6 için
    /* Yukarıda yorum olarak işaretlenen kısımların yerine aşağıdaki düzeltmeler yapıldı. */
    hata = getnameinfo((struct sockaddr *)&sin, length, hbuf, sizeof(hbuf),
        NULL, 0, NI_NUMERICHOST);
    if (hata) {
            /* hata var ise değer boş dönecektir. */
            return NULL;
            }
    return hbuf;
}
```

Şekil 5.5. virtual.c dosyasının IPv6 destekli kısmının görünümü

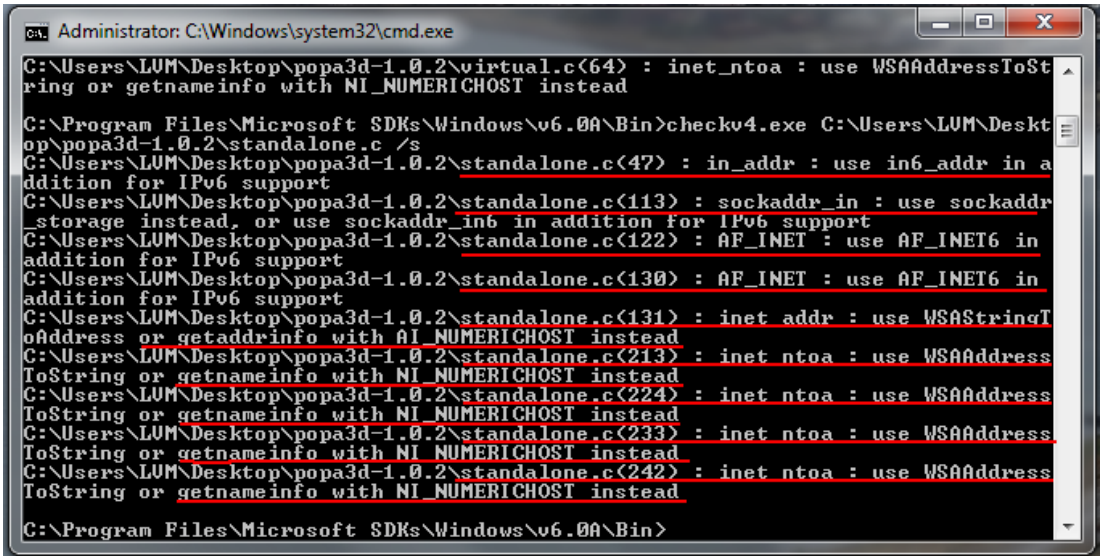
Bu dosya üzerinde yaptığımız değişiklikler ve açıklamaları şu şekildedir.

- Öncellikle **`struct sockaddr_in sin;`** olan satırı yorum yapıp bu satırın yerine **`struct sockaddr_storage sin;`** satırını yazdık. Böylece hafızada sadece IPv4 değil IPv6 adreslerini de alabilmemiz için yer ayırmış olacağız.
- IPv6 adreslerini de alabilmek için adresleri numara/sayılardan oluşacak şekilde değil de bir dizi olarak kaydetmek için bir sabit değişken olarak tanımlandı. **`static char hbuf[NI_MAXHOST];`**
- Uygulamamızın deneme aşamasında IPv4/IPv6 adres bilgilerini alamaması durumunda oluşacak bir hatayı belirtmek için ***int hata*** diye bir değişken tanımlandı.
- **`getnameinfo()`** girdi olarak soket adresi ve almak istediğimiz IPv4/IPv6 adreslerini gönderip çıktı olarak IP adreslerini bir dizi içerisinde aldık. Bu çıktıyı bir Boolean değer alan hata değişkenine atadık. Eğer almak istediğimiz IP adresinde bir sorun var ise boş değer dönecektir.

"standalone.c" dosyasının düzeltilmesi ve açıklamaları

Popa3d uygulamasının süper sunucu (Inetd) olarak mı yoksa tek başına (standalone) çalışıp çalışmayacağını *params.h* adlı başlık dosyasında yer alan POP_STANDALONE değeri belirler. POP_STANDALONE değeri 0 (sıfır) ise süper sunucu tarafından tetikleneceğini belirtir. POP_STANDALONE değeri 1 (bir) ise ya süper sunucu tarafından ya da tek başına tetikleneceğini belirtir. POP_STANDALONE değeri 1 olarak değiştirilmiştir.

Bu dosya basit olarak tek başına çalışma olarak tetiklendiği durumda popa3d uygulamasının hizmet vermesini sağlamaktır. Ek-3'te *standalone.c* dosyasının sadece IPv4 destekli değiştirilmemiş önceki hali verilmiştir. Aynı şekilde Ek-4'te *standalone.c* dosyasının IPv4/IPv6 ikilisinin destekli değiştirilmiş son hali verilmiştir. Checkv4.exe programı tarafından kontrol edilen ve Şekil 5.6'da *standalone.c* dosyasında değiştirilmesi gereken kodlar verilmiştir.



Şekil 5.6. standalone.c dosyasının checkv4.exe ile kontrolü

standalone.c dosyasında yaptığımız değişiklikler (Ek-4) ve açıklamaları şu şekilde sıralanabilir;

- Öncellikle **`standalone.c`** dosyamıza ağ veritabanı işlemlerinin tanımlamalarından sorumlu **`netdb.h`** başlık dosyası **`(#include <netdb.h>)`**, giriş çıkış arabirimlerinden sorumlu **`ioctl.h`** sistem başlık dosyası **`(#include <sys/ioctl.h>)`** ve ağ kartından sorumlu **`if.h`** ağ başlık dosyası **`(#include <net/if.h>)`** dahil edilmiştir.
- IP adresini alacağımız aygıt tanımlanmıştır. **`(#define DEVICE "eth0")`**
- Kaynak IPv4 adresinin tanımlandığı **`struct in_addr addr`** yapısı kaynak IPv6 adresini de alabilsin diye aşağıdaki kod satırı eklenmiştir.

```
#if HAVE_INET6
     struct in6_addr addr6; /* Kaynak IPv6 adresi */
#endif
```

- Belirtilen adres IPv4 veya IPv6 adresi olması durumunda **`int main (void)`** fonksiyonunda aşağıda gösterilen gerekli tanımlamalar yapılmıştır.

```
#if !HAVE_INET6
                struct sockaddr_in addr;
                int addrlen;
#else
                struct sockaddr_in6 addr6;
```

```
                struct ifreq ifr;
                char temp[24];
                int addrlen6;
#endif
```

- IPv6 / IPv4 soketlerinin TCP protokolünden dinlenmesi için **`int main (void)`** fonksiyonunda gerekli kodlamalar yapılmıştır.

```
#if HAVE_INET6
if((sock=socket(AF_INET6,SOCK_STREAM, IPPROTO_TCP)) <0)
                return log_error("socket");
#else
if((sock=socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
                return log_error("socket");
#endif
```

- IPv6 / IPv4 adres sınıfı türü, portu, soketi bilgilerini almak ve bunları adlandırmak için şu değişiklikler yapılmıştır;

```
/*Adres sınıf türü, port, soket, IPv4/IPv6 adresleri
tanımlamaları */
#if !HAVE_INET6
      memset(&addr, 0, sizeof(addr));
      addr.sin_family = AF_INET;
      addr.sin_addr.s_addr=inet_addr(DAEMON_ADDR);
      addr.sin_port = htons(DAEMON_PORT);
#else
      memset(&addr6, 0, sizeof(addr6));
      addr6.sin6_family = AF_INET6;
      addr6.sin6_port = htons(DAEMON_PORT);
if(inet_pton(AF_INET6,DAEMON_ADDR6,&addr6.sin6_addr)<0)
      return log_error("inet_pton");
      memset(&ifr, 0, sizeof(ifr));
strncpy(ifr.ifr_name, DEVICE, sizeof(ifr.ifr_name));
      if (ioctl(sock, SIOCGIFINDEX, &ifr) < 0)
      return log_error("ioctl");

#endif

#if HAVE_INET6
if(bind(sock,(struct sockaddr *)&addr6, sizeof(addr6))){
                return log_error("bind");
}
#else
if(bind(sock,(struct sockaddr *)&addr, sizeof(addr))){
                return log_error("bind");
}
#endif
/*tanımlamalar-kodlamalar*/
```

- IPv6 / IPv4 adres türüne göre gelecek olan paketlerin kabul edilmesi için aşağıdaki kodlamalar yapılmıştır.

```
/* Eğer IPv6 değil ise yani IPv4 ise */
#if !HAVE_INET6
   addrlen = sizeof(addr);
   new=accept(sock, (struct sockaddr *)&addr, &addrlen);

/* Eğer IPv6 ise */
#else
   addrlen6 = sizeof(addr6);
   new=accept(sock,(struct       sockaddr       *)&addr6,
&addrlen6);
#endif /* IPv6 bitis*/
```

- IPv6 / IPv4 adres türüne göre aynı adres üzerinden birden fazla oturum açılmaması için gerekli *for* döngüsü içerisine aşağıdaki kod eklenmiştir.

```
/* kaynak adres için en fazla oturum sayısı */
/* Eğer IPv6 değil ise yani IPv4 ise */
#if !HAVE_INET6
   if (sessions[i].addr.s_addr == addr.sin_addr.s_addr)
   if (++n >= MAX_SESSIONS_PER_SOURCE) break;

/* Eğer IPv6 ise */
#else
   if(sessions[i].addr6.s6_addr==addr6.sin6_addr.s6_addr
)
   if (++n >= MAX_SESSIONS_PER_SOURCE) break;
#endif

#if !HAVE_INET6
   syslog(SYSLOG_PRI_HI,"%s:  sessions  limit  reached",
   inet_ntoa(addr.sin_addr));

#else
   syslog(SYSLOG_PRI_HI,  "%s:  sessions  limit  reached",
   inet_ntop(AF_INET6,&addr6.sin6_addr,temp,sizeof(temp)
));
#endif
```

- Oturum bilgilerinin hangi adres ve adres türünden geldiğini kayıt altına almak için aşağıdaki kodlamalar yapılmıştır.

```
/* IPv4 adresine göre olan sistem logları */
#if !HAVE_INET6
```

```
   syslog(SYSLOG_PRI_LO, "Session from %s",
   inet_ntoa(addr.sin_addr));

/* IPv6 adresine göre olan sistem logları */
#else
   syslog(SYSLOG_PRI_LO, "Session from %s",
   inet_ntop(AF_INET6, &addr6.sin6_addr, temp,
   sizeof(temp)));
#endif
```

- IPv6 / IPv4 adreslerinden gelen oturum bilgilerinin doğruluğunu onaylamak için aşağıdaki kod satırı eklenmiştir.

```
/* IPv4 adresi ise */
#if !HAVE_INET6
   sessions[j].addr = addr.sin_addr;

/* IPv6 adresi ise */
#else
   sessions[j].addr6 = addr6.sin6_addr;
#endif
```

Ek-4'te kalın olarak yazılan kodlamalar IPv6 desteğini sağlamak için geliştirilmiş kısımlardır. Bu değişikliklerden sonra popa3d uygulaması artık komut satırından -4 ve -6 parametrelerinden herhangi birini girerek hangi IP türüne göre işlem yapacağını belirtebiliriz.

"params.h" dosyasının düzeltilmesi ve açıklamaları

Popa3d uygulamasının süper sunucu (Inetd) olarak mı yoksa tek başına (standalone) çalışıp çalışmayacağını *params.h* adlı başlık dosyasında yer alan POP_STANDALONE değeri belirler. POP_STANDALONE değeri 0 (sıfır) ise süper sunucu tarafından tetikleneceğini belirtilir. POP_STANDALONE değeri 1 (bir) ise ya süper sunucu tarafından ya da tek başına tetikleneceğini belirtir.

Bu dosya basit olarak tek başına çalışma olarak tetiklendiği durumda popa3d uygulamasının hizmet vermesini sağlamaktır. Ek-5'te *params.h* dosyasının sadece IPv4 destekli değiştirilmemiş önceki hali verilmiştir. Aynı şekilde Ek-6'da *params.h* dosyasının IPv4 / IPv6 ikilisinin destekli değiştirilmiş son hali verilmiştir.

params.h dosyasında yaptığımız değişiklikler (Ek-6) ve açıklamaları şu şekilde sıralanabilir;

- IPv6 paketlerini de uygulamamızda kullanabilmek için aşağıdaki kod satırı eklendi.

```
//IBRAHIM AKSIT -> IPv6 Desteği sağlamak için tanımlandı
#define HAVE_INET6                1
```

- Sadece IPv4 adresleri için bütün arayüzleri dinleyecek olan 0.0.0.0 adres tanımlamasına (**`#define DAEMON_ADDR "0.0.0.0" /* INADDR_ANY */`**) ek olarak IPv6 adreslerini dinleyecek olan "`::`" adresi için aşağıdaki tanımlama yapılmıştır.

```
//IBRAHIM AKSIT -> IPv6 dinlenecek adres için tanımlandı
#define DAEMON_ADDR6              "[::]"
/* IPv6 için dinlenecek adres - bütün arayüzleri dinler
*/
```

Ek-6'da kalın olarak yazılan kodlamalar IPv6 desteğini sağlamak için yapılmış/eklenmiş kodlamalardır. Bu değişikliklerden sonra popa3d uygulaması artık IPv4 adreslerinin yanında IPv6 adreslerini de dinleyecektir.

"startup.c" dosyasının düzeltilmesi ve açıklamaları

Popa3d uygulamasının başlangıç dosyasıdır. Komut satırından gelen parametrelerle işlem yapar. Ek-7'de *startup.c* dosyasının sadece IPv4 destekli değiştirilmeden önceki hali verilmiştir. Aynı şekilde Ek-8'de *startup.c* dosyasının IPv4/IPv6 ikilisinin destekli değiştirilmiş son hali verilmiştir.

startup.c dosyasında yaptığımız değişiklikler (Ek-8) ve açıklamaları şu şekilde sıralanabilir;

- Temel anlamda soket başlıkları sistemden çağrılarak uygulamamıza dahil edilmiştir.

```
/* socket başlık dosyasını uygulamaya dahil ediyoruz */
#include <sys/socket.h>
```

- IPv6 adres sınıfını da dinleyebilmek için aşağıdaki kod satırı eklenmiştir.

```
/* Adres sınıfına IPv6 ekledim */
int af = AF_INET6;
```

- Uygulamamızın kullanım fonksiyonuna komut satırından IPv4 ve IPv6 adreslerini parametre olarak belirtebileceğimiz kodlar eklenmiştir.

```
/* Komut satırı IPv6 desteği için ekledim.*/
fprintf(stderr, "Usage: %s [-D] [-V] [-4] [-6] \n",
progname);
```

- Yukarıdaki maddede bahsedilen komut satırı desteğini yapan kodlar eklenmiştir.

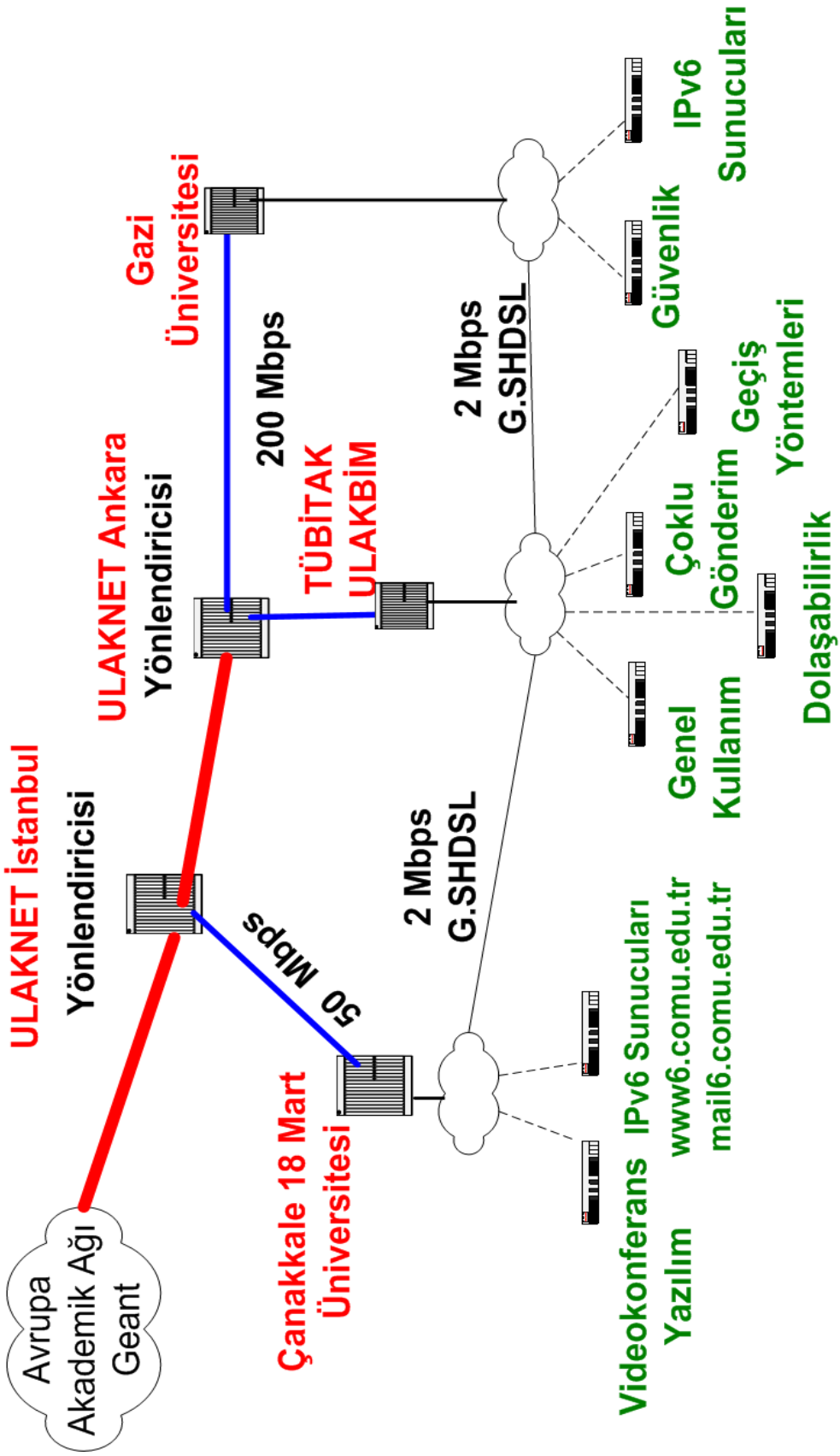
```
//DV46 programın Date = D, Version = V, IPv4 = 4, IPv6 =
6 için tanımlandı.
while ((c = getopt(argc, argv, "DV46")) != -1) {
                ......
                // IPv4 ise
                case '4':
                af = AF_INET;
                break;

                // IPv6 ise
                case '6':
                af = AF_INET6;
                break;
                ......
}
```

Ek-8'de kalın olarak yazılan kodlamalar IPv6 desteğini sağlamak için geliştirilmiştir. Bu değişikliklerden sonra popa3d uygulaması artık IPv4 adreslerinin yanında IPv6 adreslerini de işleyecek ve komut satırından -6 parametresini de kabul edecektir.

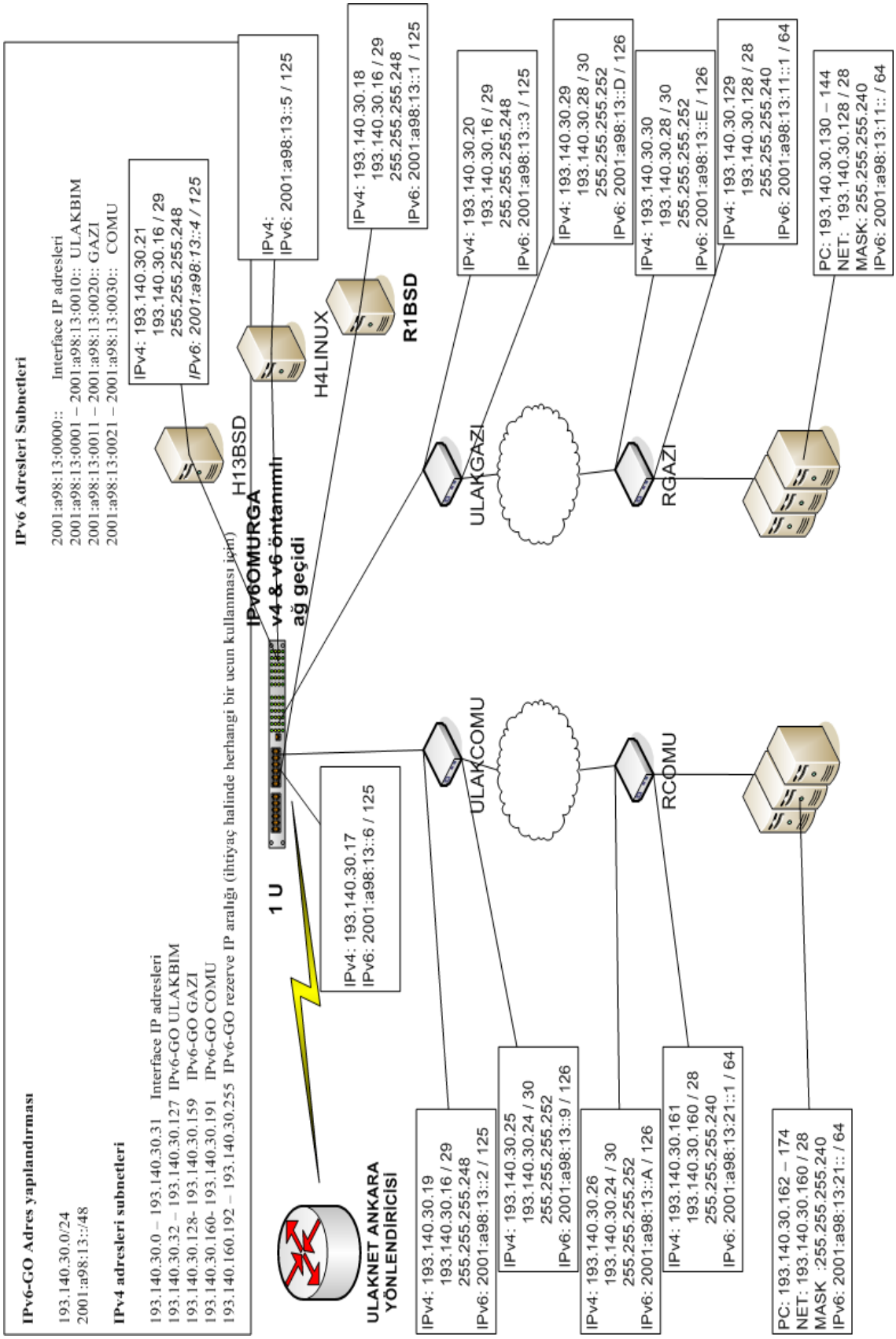
5.3.2. Uygulama test ortamı

Geliştirilen IPv6 destekli popa3d uygulaması IPv6 erişiminin ULAKBİM tarafından sağlandığı "Ulusal IPv6 Protokol Altyapısı Tasarımı ve Geçişi Projesi" kapsamında geliştirilen Şekil 5.7 ve Şekil 5.8'de genel topolojisi gösterilen IPv6 Test Yatağı ve Geliştirme Ortamı (IPv6-GO)'nda test edilmesi amaçlanmıştı. Fakat, IPv6 destekli popa3d uygulaması proje süresinde tamamlanamamıştır. Ayrıca, "Ulusal IPv6 Protokol Altyapısı Tasarımı ve Geçişi Projesi" tamamlandıktan sonra Gazi Üniversitesi Bilgisayar Mühendisliği Bölümünde kurulan güvenlik laboratuvarındaki

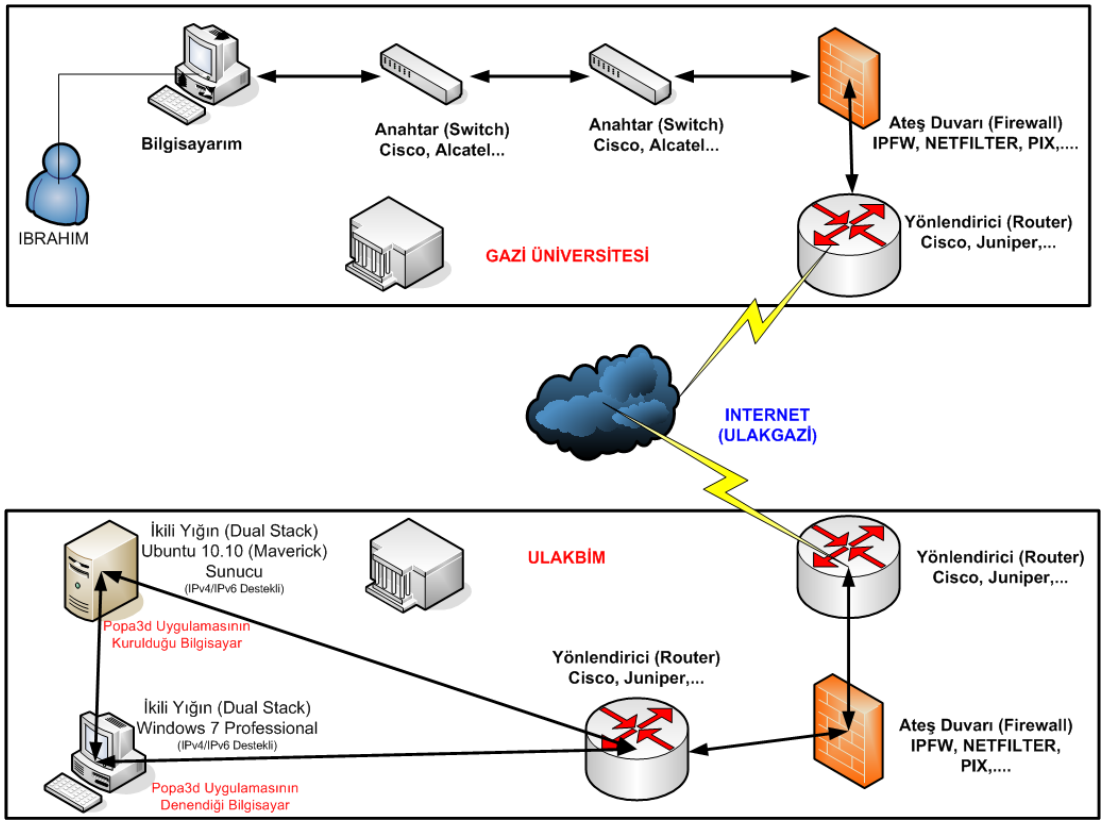
IPv6 bağlantısı maddi sebeplerden dolayı kesilmek zorunda kalmıştır. Bu yüzden uygulama bu test ortamında denenememiştir. Bunun yerine geliştirilen uygulama, ULAKBİM tarafından sağlanan test ortamı ile bir bağlantı ortamı oluşturulmuş ve oluşturulan bu IPv6 destekli ağ ortamında denenmiştir. ULAKBİM tarafından sağlanan ve uygulamaya ait test ortamı ağ topolojisi Şekil 5.9'da gösterilmiştir.



Şekil 5.7. IPv6-GO ağ topolojisi [7]



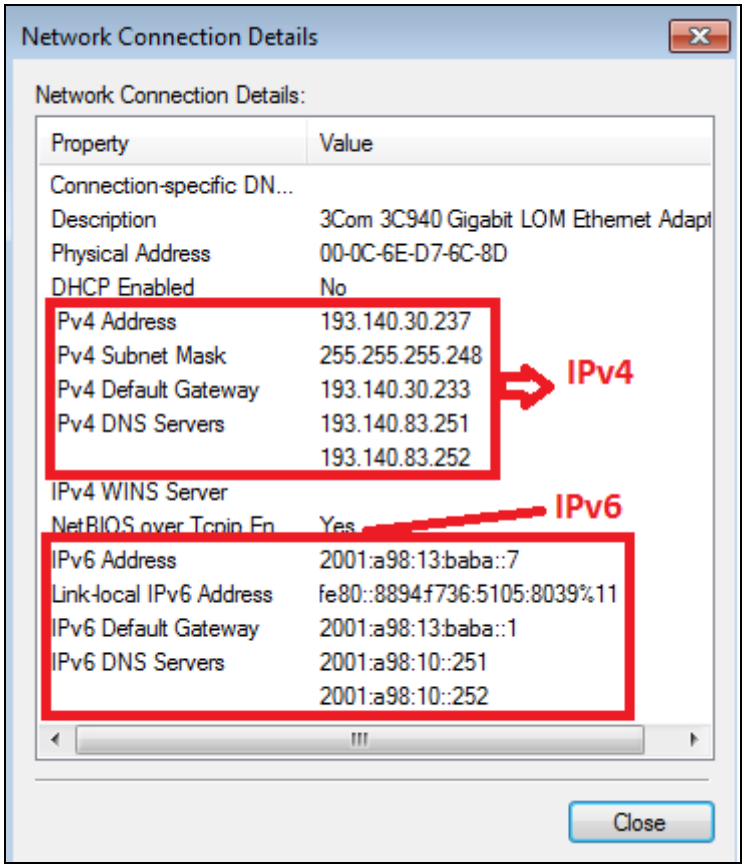
Şekil 5.8. IPv6-GO genel topolojisi [7]



Şekil 5.9. Uygulama test ortamı genel ağ topolojisi

ULAKBİM tarafından IPv6 destekli test ortamı sağlandıktan sonra Gazi Üniversitesi Bilgi İşlem Dairesi Başkanlığı'nda bulunan sadece IPv4 adresine sahip bilgisayarım üzerinden ULAKBİM'de bulunan ikili yığın mekanizmasına sahip (IPv4/IPv6) Linux Ubuntu 10.10 maverick yüklü sunucuya uzaktan erişim sağlanmıştır. Uzaktan erişim sağlanan sunucu üzerine geliştirmiş olduğum uygulamanın dosyaları aktarılmıştır.

Windows 7 Professional yüklü istemci bilgisayara ait IP bilgileri Şekil 5.10'da verilmiştir.



Şekil 5.10. Windows 7 professional IP bilgileri

Linux Ubuntu 10.10 (maverick meerkat) yüklü sunucuya ait IP bilgileri Şekil 5.11'de verilmiştir.

```
root@H16LINUX:/home/test# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0c:6e:d7:6a:8d
IPv4      inet addr:193.140.30.22  Bcast:193.140.30.23  Mask:255.255.255.248
          inet6 addr: fe80::20c:6eff:fed7:6a8d/64 Scope:Link
IPv6      inet6 addr: 2001:a98:13::5/125 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15271 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11277 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1884747 (1.8 MB)  TX bytes:2234455 (2.2 MB)
          Interrupt:22
```

Şekil 5.11. Linux Ubuntu 10.10 (maverick meerkat) sunucu IP bilgileri

Test ortamında bulunan istemci ve sunucu IP adres bilgilerini verdikten sonra uygulamamızın kurulumuna ait aşamalara geçebiliriz. Bu aşamalar şu şekildedir;

- Sunucuda /home/test/iaksit/tez/ klasörüne aktarmış olduğumuz uygulamamız **make** komutu (root@H16LINUX:/home/test/iaksit/tez# **make**) ile derlenip Şekil 5.12'de derleme sonucu verilmiştir.
- Derleme işlemi hatasız yapıldıktan sonra **make install** komutu (root@H16LINUX:/home/test/iaksit/tez# **make install**) ile uygulamamız sunucuda çalışır duruma getirilmiştir. Şekil 5.13'te uygulamanın yükleme durumu ve sonucu verilmiştir.
- Uygulamamız yüklendikten sonra **popa3d -D** komutunu (root@H16LINUX:/home/test/iaksit/tez# **popa3d –D**) yazarak çalıştırılır. Popa3d uygulamamızın çalışıp çalışmadığını **netstat -tapn** komutu ile görülebilir. Şekil 5.14'te uygulamanın hizmet verme durumuna ait bilgiler verilmiştir.
- Windows 7 Professional yüklü istemcimiz üzerinde bulunan Putty programı aracılığıyla telnet bağlantısı 110 port numarası ile sunucu üzerinde bulunan ve IPv6 destekli hizmet vermeye hazır popa3d uygulamamıza sağlanmıştır. Şekil 5.15'te Windows 7 professional istemci üzerinden 110 port numaralı POP3 bağlantı durumu verilmiştir. Şekil 5.16'da ise istemci tarafından gönderilen TCP 110 bağlantısına ait sunucu üzerinde bulunan popa3d uygulamamızın hizmet verme durumu verilmiştir.
- Windows 7 Professional yüklü istemcimizde yüklü bulunan Microsoft Office Outlook programı aracılığıyla geliştirilmiş olan Popa3d uygulamasının gelen e-postaları alıp almadığı test edilmiştir. Ayrıca e-posta gönderme yazılımı olarak da sunucu üzerine yüklenen Postfix uygulaması kullanılmıştır. Şekil 5.17'de Microsoft Office Outlook programında e-posta göndermek ve almak için yapılan yapılandırmalar verilmiştir.
- Microsoft Office Outlook programı ile POP3 hizmeti yapılandırmasından sonra geliştirilen uygulamayı test etmek amacıyla dosya menüsünden yeni ileti tıklanarak Şekil 5.18'de gösterilen bilgiler girilmiştir.
- Microsoft Office Outlook programı aracılığıyla gönderilen e-posta iletisinin ulaşıp ulaşmadığını test etmek için araç çubuğunda bulunan gönder/al bağlantısına tıklanmıştır. Geliştirilen uygulamanın Microsoft Office Outlook

aracılığı ile gelen e-postaları sorunsuz bir şekilde aldığı Şekil 5.19'da verilmiştir.

```
root@H16LINUX:/home/test/iaksit/tez# make
gcc -Wall -O2 -fomit-frame-pointer -c version.c
gcc -Wall -O2 -fomit-frame-pointer -c startup.c
gcc -Wall -O2 -fomit-frame-pointer -c standalone.c
standalone.c: In function âdo_standaloneâ:
standalone.c:248: warning: pointer targets in passing argument 3 of âacceptâ dif
fer in signedness
/usr/include/sys/socket.h:214: note: expected âsocklen_t * __restrict__â but arg
ument is of type âint *â
standalone.c:204: warning: ignoring return value of âchdirâ, declared with attri
bute warn_unused_result
gcc -Wall -O2 -fomit-frame-pointer -c virtual.c
gcc -Wall -O2 -fomit-frame-pointer -c auth_passwd.c
gcc -Wall -O2 -fomit-frame-pointer -c auth_shadow.c
auth_shadow.c: In function âauth_userpassâ:
auth_shadow.c:58: warning: ignoring return value of âwriteâ, declared with attri
bute warn_unused_result
gcc -Wall -O2 -fomit-frame-pointer -c auth_pam.c
gcc -Wall -O2 -fomit-frame-pointer -c pop_root.c
gcc -Wall -O2 -fomit-frame-pointer -c pop_auth.c
gcc -Wall -O2 -fomit-frame-pointer -c pop_trans.c
gcc -Wall -O2 -fomit-frame-pointer -c protocol.c
gcc -Wall -O2 -fomit-frame-pointer -c database.c
gcc -Wall -O2 -fomit-frame-pointer -c mailbox.c
gcc -Wall -O2 -fomit-frame-pointer -c misc.c
gcc -Wall -O2 -fomit-frame-pointer -c md5/md5.c -o md5/md5.o
gcc -s version.o startup.o standalone.o virtual.o auth_passwd.o auth_shadow.o au
th_pam.o pop_root.o pop_auth.o pop_trans.o protocol.o database.o mailbox.o misc.
o md5/md5.o  -lcrypt -o popa3d
root@H16LINUX:/home/test/iaksit/tez#
```

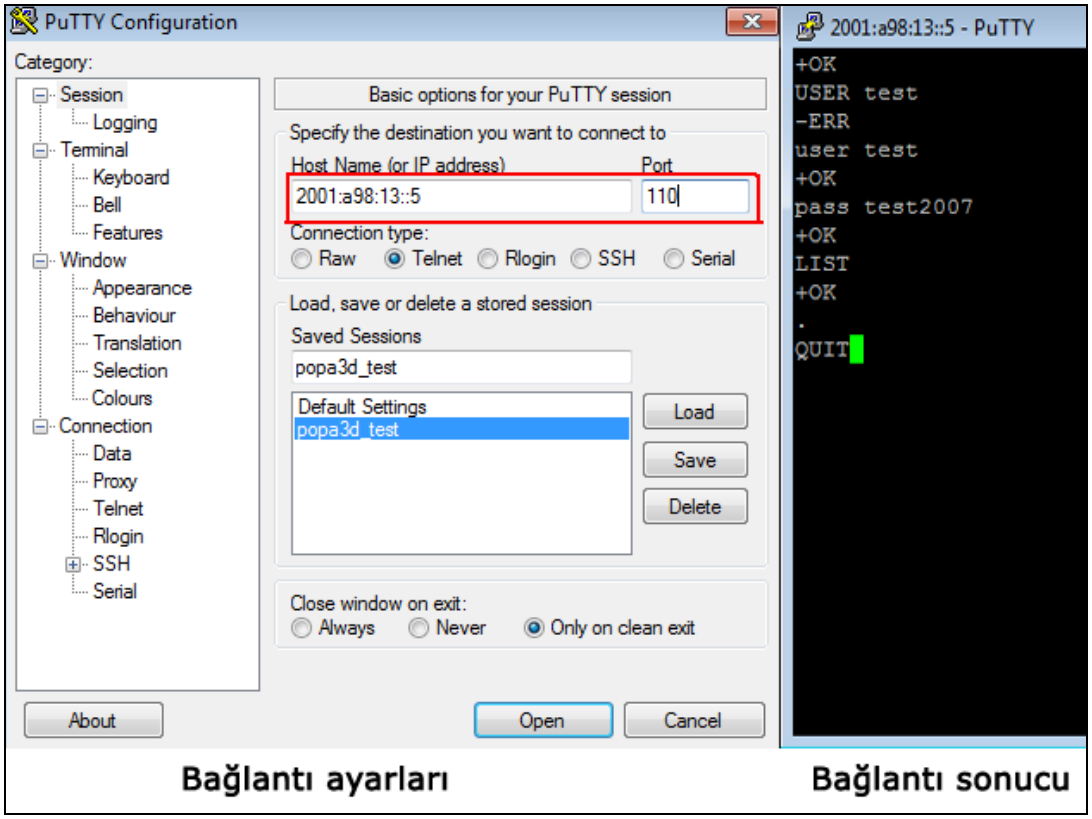
Şekil 5.12. Uygulama derleme ve sonucu

```
root@H16LINUX:/home/test/iaksit/tez# make install
mkdir -p -m 755 /usr/local/sbin /usr/local/man/man8
install -c -m 700 popa3d /usr/local/sbin/
install -c -m 644 popa3d.8 /usr/local/man/man8/
root@H16LINUX:/home/test/iaksit/tez#
```

Şekil 5.13. Uygulama yükleme durumu ve sonucu

```
root@H16LINUX:/home/test/iaksit/tez# netstat -tapn
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      565/sshd
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      751/cupsd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1226/exim4
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      941/mysqld
tcp        0    240 193.140.30.22:22        194.27.15.253:50091     ESTABLISHED 4008/sshd: test [pr
tcp6       0      0 :::80                   :::*                    LISTEN      1371/apache2
tcp6       0      0 :::22                   :::*                    LISTEN      565/sshd
tcp6       0      0 ::1:631                 :::*                    LISTEN      751/cupsd
tcp6       0      0 ::1:25                  :::*                    LISTEN      1226/exim4
tcp6       0      0 :::110                  :::*                    LISTEN      4492/popa3d
root@H16LINUX:/home/test/iaksit/tez#
```

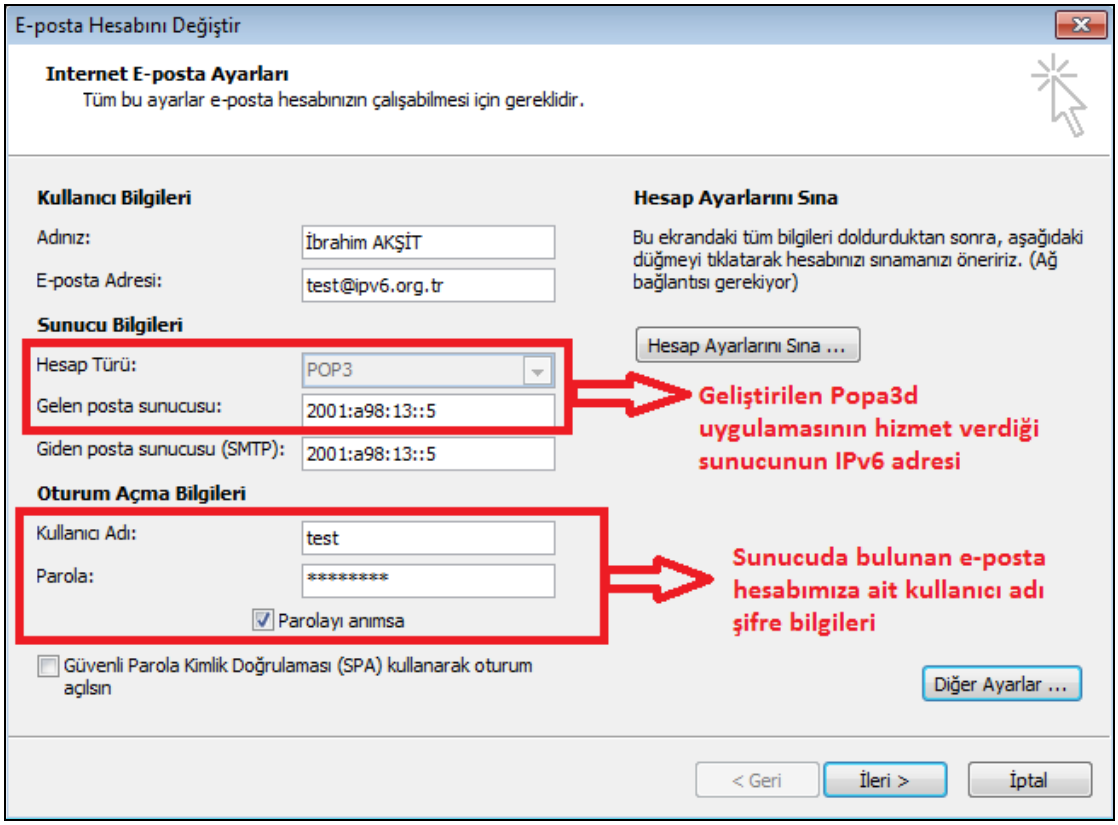
Şekil 5.14. Uygulama hizmet verme durumu ve sonucu



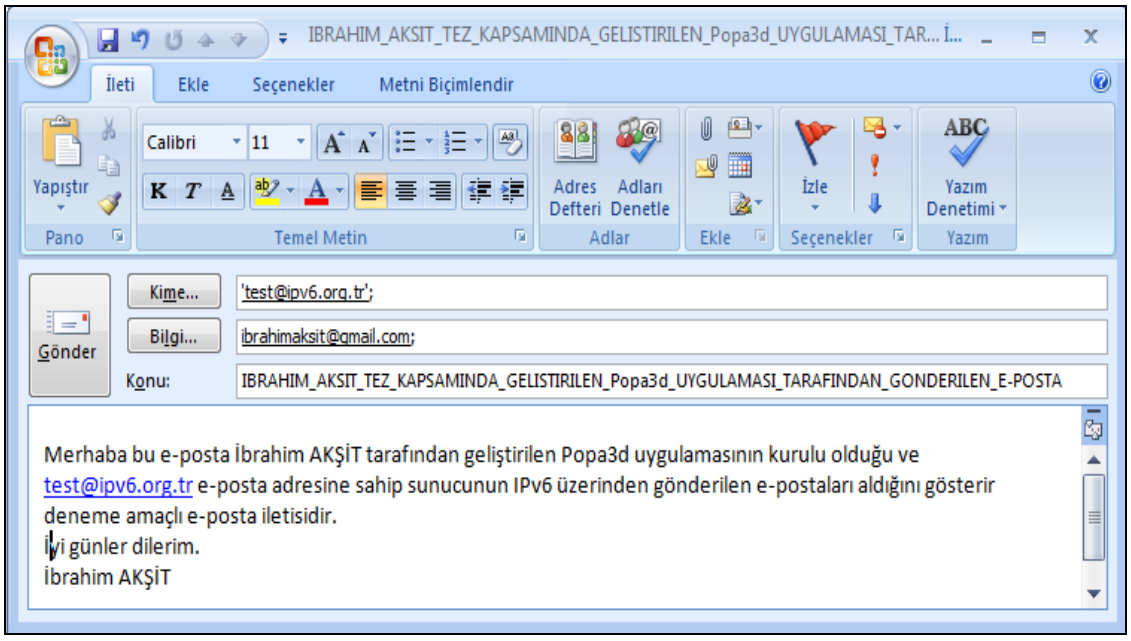
Şekil 5.15. Windows 7 üzerinden 110 portuna telnet bağlantısı

```
root@H16LINUX:/home/test/iaksit/tez# netstat -tapn
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      565/sshd
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      751/cupsd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1226/exim4
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      941/mysqld
tcp        0     48 193.140.30.22:22        194.27.15.253:50091     ESTABLISHED 4008/sshd: test [pr
tcp6       0      0 :::80                   :::*                    LISTEN      1371/apache2
tcp6       0      0 :::22                   :::*                    LISTEN      565/sshd
tcp6       0      0 ::1:631                 :::*                    LISTEN      751/cupsd
tcp6       0      0 ::1:25                  :::*                    LISTEN      1226/exim4
tcp6       0      0 :::110                  :::*                    LISTEN      4492/popa3d
tcp6       0      0 2001:a98:13::5:110      2001:a98:13:baba::49303 ESTABLISHED 4515/popa3d
root@H16LINUX:/home/test/iaksit/tez#
```

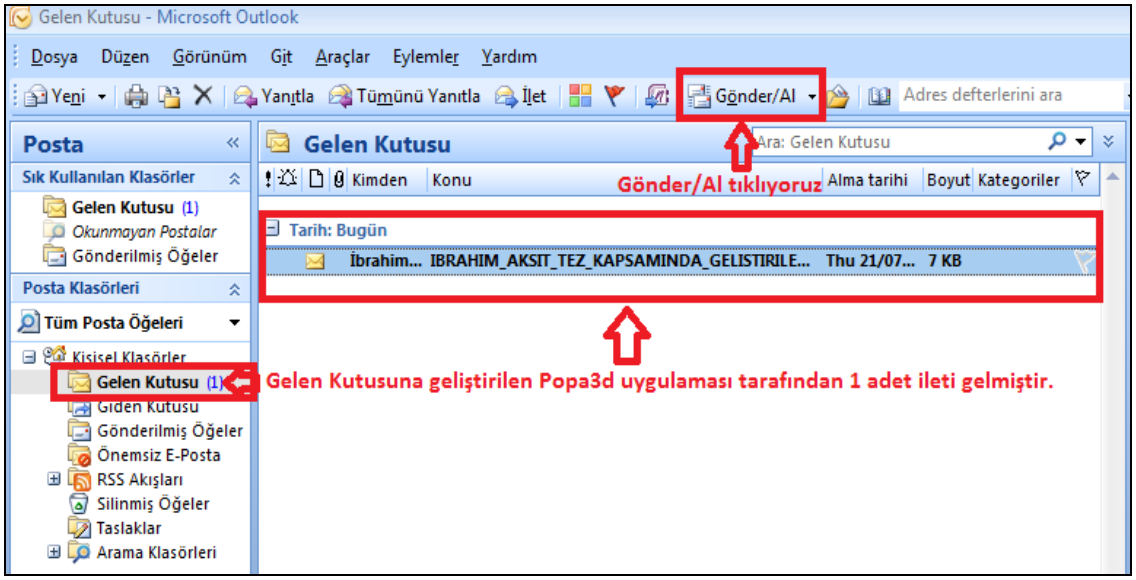
Şekil 5.16. Uygulamanın 110 portuna hizmet vermesi



Şekil 5.17. Uygulamanın Microsoft Office Outlook ile POP3 yapılandırması



Şekil 5.18. Microsoft Office Outlook ile yeni ileti gönderimi



Şekil 5.19. Microsoft Office Outlook ile gelen iletinin alınması

Bu çalışma kapsamında geliştirilen POP3 uygulaması ULAKBİM tarafından sağlanan IPv6 destekli ağ ortamında test edilmiştir. Geliştirilen uygulamanın, IPv6 paketlerine hizmet verme durumu ve sonucu Şekil 5.16'da ve Microsoft Office Outlook programı ile yapılandırılması Şekil 5.17'de, gelen e-postaların başarıyla alınması Şekil 5.19'da gösterilmiştir. Yapılan testler sonucunda uygulamanın başarıyla çalıştığı görülmüştür.

6. SONUÇ VE ÖNERİLER

Günümüzde en yaygın olarak kullanılan ve internet teknolojilerinin alt yapısını oluşturan internet protokolü (IP)'nün ilk zamanlarda sınırlı sayıda kullanıcıya hizmet vermesi düşünülmüştür. IPv4 olarak adlandırılan eski internet protokolü artık ihtiyaçlara cevap vermemekte ve en önemlisi adres sayısı hızla tükenmiştir. Bilgisayar kullanım oranının hızla artması, mobil haberleşme cihazlarının internete erişim sağlaması, IP tabanlı teknoloijilerin (IP telefon, IP televizyon, IP tabanlı görüntülü konferans, VoIP v.b.) kullanımının artması ve güvenlik, hız, performans ihtiyaçlarının önem arz etmesi yeni nesil bir internet protokolü ihtiyacını doğurmuştur. IPv6 olarak adlandırılan yeni nesil IP, milyarlarca adres, yüksek performans, hız, güvenlik, hizmet kalitesi, mobil kullanım, dolaşılabilirlik, geliştirilebilme ve eski-yeni protokoller ile bir arada kullanılabilme imkanlarını sağlamaktadır. IPv6 sadece sunduğu hizmet ve imkanlar göz önüne alındığında yaşantımızda ne kadar büyük bir öneme sahip olduğu anlaşılacaktır.

Mevcut uygulamaların IPv6 ile kullanılabilmesi teknolojik yaşamımızın yanında günlük yaşamımız açısından da önemlidir. IPv6'ya geçişin hızla devam ettiği bu günlerde ulusal, bölgesel ve küresel anlamda hizmet verilen veya alınan uygulamaların IPv6 desteğine sahip olması büyük önem arz etmeye başlamıştır. Uygulamaların IPv6 hizmeti verebilecek duruma getirilebilmesi akademik çalışmaların yapılması, IPv6'ya gerek uygulama gerekse teknoloji olarak hazır olunması ülkemiz için önem arz etmektedir. Uygun alt yapının sağlanmasından sonra uygulamaların da IPv6 desteği sağlaması gerçeği ortaya çıkmaktadır.

Bu çalışmada IPv6 desteğini sağlamak için mevcut özgür yazılım lisansına sahip Popa3d olarak adlandırılan ve POP3 hizmeti sunan sunucu tabanlı bir uygulama geliştirilmiştir. Geliştirilen uygulama çeşitli Unix platformlarında güvenliği, dayanıklılığı, RFC uyumluluğu ve performansı temel amaç edinen bir uygulamadır. Bu uygulamada IPv6 paketleri üzerinden gelen e-postaları güvenli bir şekilde almak için gerekli düzenlemeler yapılmış, Microsoft Office Outlook programı kullanılarak e-postaların başarıyla alındığı beşinci bölümde şekillerle gösterilmiştir.

Mevcut uygulamaların IPv6 paketlerini de işleyebilmesi, IPv6 dünyasında dolaşarak gerekli yerlere ulaştırılması ve IPv6 destekli uygulamaların geliştirilmiş olması çalışmanın sağladığı katkılar arasındadır.

Bu tez çalışmasında, uygulama yazılımının geliştirilmesi esnasında denenmesi donanımsal ve yazılımsal IPv6 desteği veren henüz bir ortam olmadığı için pek çok zorluklarla karşılaşılmıştır.

Bu çalışmadan sonra en sık kullanılan uygulamaların IPv6 desteği durumu detaylı bir şekilde araştırılmalı, gerekli düzenlemeler ve çalışmalar yapılmalıdır. Günümüzde yaygın olarak kullanılan fakat henüz IPv6 desteğine sahip olmayan istemci ve sunucu tabanlı uygulamalara IPv6 desteği vermek amacıyla gerekli geliştirilmeler yapılması için bu tür uygulamaların sayısının arttırılması faydalı olacaktır.

KAYNAKLAR

1. İnternet: Pastel, J., "Internet Protocol Darpa Internet Program Protocol Specification", ***Internet Engineering Task Force (IETF) Magazine,*** **RFC 791**, http://www.ietf.org/rfc/rfc791.txt, 11-31 (1981). (10.09.2010)

2. İnternet: U.S Census Bureau, "World POPClock Projection", ***International Data Base (IDB)***, http://www.census.gov/IPc/www/popclockworld.html, (25.05.2011).

3. İnternet: APNIC, "APNIC Triggers Last IANA IPv4 Allocations", http://www.apnic.net/publications/press/releases/2011/APNIC_Final-Five.pdf, (25.04.2011).

4. İnternet: RIPE NCC, "IPv4 Exhaustion", http://www.IPv6actnow.org/info/what-is-IPv4/, (25.04.2011).

5. İnternet: Bradner, S. and Mankin, A., "The Recommendation for the IP Next Generation Protocol", ***Internet Engineering Task Force (IETF) Magazine,*** **RFC 1752**, http://www.ietf.org/rfc/rfc1752.txt, 11-33 (1995). (18.09.2010)

6. İnternet: Deering, S. and Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification", ***Internet Engineering Task Force (IETF) Magazine,*** **RFC 2460**, http://www.ietf.org/rfc/rfc2460.txt, 1-35 (1998). (12.10.2010)

7. Bektaş, O., Soysal, M. ve Orcan, S., "Türkiye için IPv6 geçişi zaman/aşama planı önerisi", ***Ulusal IPv6 Konferansı***, Ankara, 12-14 (2011).

8. İnternet: TÜBİTAK-ULAKBİM, Gazi Üniversitesi ve Çanakkale 18 Mart Üniversitesi, "Ulusal IPv6 Protokol Altyapısı Tasarımı ve Geçişi Projesi", http://www.ipv6.net.tr/, 2009. (15.10.2010)

9. Waddington, D.G. and Chang, F., "Realizing the Transition to IPv6", **IEEE Communications Magazine,** 138-148 (2002).

10. İnternet: Nordmark, E. and Gilligan, R., "Transition Mechanisms for IPv6 Hosts and Routers", ***Internet Engineering Task Force (IETF) Magazine***, **RFC 2893,** http://www.ietf.org/rfc/rfc2893.txt, 6-24 (2000). (17.10.2010)

11. İnternet: Nordmark, E. and Gilligan, R., "Basic Transition Mechanisms for IPv6 Hosts and Routers", ***Internet Engineering Task Force (IETF) Magazine***, **RFC 4213**, http://www.ietf.org/rfc/rfc4213.txt, 2-19 (2005). (18.10.2010)

12. İnternet: Carpenter, B. and Jung, C., "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", ***Internet Engineering Task Force (IETF) Magazine***, **RFC 2529,** http://www.ietf.org/rfc/rfc2529.txt, 2-6 (1999). (20.10.2010)

13. İnternet: Templin, F., Gleeson, T., Talwar, M. and Thaler, D., "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", ***Internet Engineering Task***

Force (IETF) Magazine, **RFC 5214**, http://www.ietf.org/rfc/rfc5214.txt, 2-10 (2008). (20.10.2010)

14. İnternet: Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", Microsoft, ***Internet Engineering Task Force (IETF) Magazine***, **RFC 4380,** http://www.ietf.org/rfc/rfc4380.txt, 4-20 (2006). (24.11.2010)

15. İnternet: Almquist, P., "Type of Services in the Internet Protocol", ***Internet Engineering Task Force (IETF) Magazine***, **RFC 1349,** http://www.ietf.org/rfc/rfc1349.txt, 4-25 (1992). (25.11.2010)

16. Kurose, J. F. and Ross, K. W., "Computer Networking A Top - Down Approach Featuring The Internet 5th Edition", ***Addison Wesley***, Pearson, USA, 341-421 (2010).

17. İnternet : Wikipedia, "Time to Live", http://en.wikIPedia.org/wiki/Time_to_live. (28.11.2010).

18. İnternet: Reynolds, J. and Postel, J., "Assigned Numbers", ***Internet Engineering Task Force (IETF) Magazine***, **RFC 1700**, http://www.ietf.org/rfc/rfc1700.txt, (1994). (26.01.2011)

19. İnternet: Microsoft Technet, "IPv4 Addressing", http://technet.microsoft.com/en-us/library/dd469716%28WS.10%29.aspx, (28.04.2011).

20. İnternet: Cisco Systems Inc., "CCNP1:Advanced IP Addressing Management", ***Cisco Press***, http://www.ciscopress.com/articles/article.asp?p=330807&seqNum=2, (2004). (21.04.2011)

21. İnternet: IANA, "Special–Use IPv4 Addressess", ***Internet Engineering Task Force (IETF) Magazine***, **RFC 3330**, http://www.ietf.org/rfc/rfc3330.txt, (2002). (22.04.2011)

22. İnternet: Hinden, R., "Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR)", ***Internet Engineering Task Force (IETF) Magazine***, **RFC 1517**, http://www.ietf.org/rfc/rfc1517.txt, (1993). (24.02.2011)

23. İnternet: Fuller, V. and Li, T., "Classless Inter-Domain Routing (CIDR) : The Internet Address Assignment and Aggregation Plan", ***Internet Engineering Task Force (IETF) Magazine***, **RFC 4632**, http://www.ietf.org/rfc/rfc4632.txt, 4-25 (2006). (27.02.2011)

24. Bieringer, P., "Status of IPv6 (Information & Workshop)", **IEEE CNF, Applications and the Internet Workshops**, 54-57 (2005).

25. Stallings, W., "Computer Networking with Internet Protocols And Technology International Edition", ***Prentice Hall***, **Pearson**, USA, 290-300 (2004).

26. Hagen, S., "IPv6 Essentials 2nd Edition", ***O'Reilly,*** **O'Reilly Media**, (2006).

27. İnternet: Kent, S. and Atkinson, R., "IP Authentication Header", ***Internet Engineering Task Force (IETF),*** **RFC 2402,** http://www.ietf.org/rfc/rfc2402.txt, 2-16 (1998). (08.03.2011)

28. İnternet: Kent, S., "IP Encapsulating Security Payload (ESP)", ***Internet Engineering Task Force (IETF)***, **RFC 4303**, http://www.ietf.org/rfc/rfc4303.txt, 5-32 (2005). (08.03.2011)

29. İnternet: Hinden, R. and Deering, S., "IP Version 6 Addressing Architecture", ***Internet Engineering Task Force (IETF),*** **RFC 4291,** http://www.ietf.org/rfc/rfc4291.txt, 2-18 (2006). (10.03.2011)

30. İnternet: Hinden, R., Deering, S. and Nordmark, E., "IPv6 Global Unicast Address Format", ***Internet Engineering Task Force (IETF),*** **RFC 3587,** http://www.ietf.org/rfc/rfc3587.txt, 1-5 (2003). (10.03.2011)

31. İnternet: Conta, A. and Deering, S., "Generic Packet Tunneling in IPv6 Specification", ***Internet Engineering Task Force (IETF)***, **RFC 2473**, http://www.ietf.org/rfc/rfc2473.txt, 2-30 (1998). (12.03.2011)

32. İnternet: Vaughan-Nichols, J. S., "IPv6 and IPv4 Co-existence", http://content.dell.com/us/en/enterprise/d/large-business/IPv6-IPv4-co-existence.aspx, (2010). (20.04.2011)

33. Aktaş, M. and Sağıroğlu, Ş., "IPv6: Uluslararası çalışmalar ve Türkiye'de durum", ***Ulusal IPv6 Konferansı***, Ankara, 5-7 (2011).

34. Ahrouch, A. A. and Ezzine, S., "IPv4 to IPv6 Migration", Analytical Network Project, Master Program System and Network Engineering, ***University of Amsterdam,*** 5-17 (2004).

35. Beijnum, V. I., "Running IPv6", **Apress Press**, New York, USA, 1-131 (2006).

36. İnternet: Carpenter, B. and Moore, T., "Connection of IPv6 Domains via IPv4 Clouds", ***Internet Engineering Task Force (IETF),*** **RFC 3056,** http://www.ietf.org/rfc/rfc3056.txt, 4-19 (2001). (23.04.2011)

37. İnternet: Morr, D., "IPv6 Programming", IPv6 Deployment, PennState University, https://wikispaces.psu.edu/display/ipv6/Home, 2009 (28.04.2011).

38. İnternet: Başbakanlık Genelgesi, "Kamu Kurum ve Kuruluşları için IPv6'ya Geçiş Planı", http://www.resmigazete.gov.tr/eskiler/2010/12/20101208-7.htm, 2010. (25.02.2011)

39. İnternet: Stevens, W., Thomas, M., Nordmark, E. and Jinmei T., "Advanced Sockets Application Program Interface (API) for IPv6", ***Internet Engineering Task Force (IETF),*** **RFC 3542,** http://www.ietf.org/rfc/rfc3542.txt, 5-52 (2003). (16.04.2011)

40. İnternet: Gilligan, R., Thomson, S., Bound, J. and Stevens, W., "Basic Socket Interface Extensions for IPv6", ***Internet Engineering Task Force (IETF),*** **RFC 2553,** http://www.ietf.org/rfc/rfc2553.txt, 3-35 (1999). (18.04.2011)

41. Hagino, J. I., "Address-family independent socket programming for IPv6", ***KAME/WIDE Project***, Research Laboratory, Internet Initiative Japan, (2004)

42. İnternet: Google, "Access Google Services over IPv6", ***Google Inc.***, http://www.google.com/intl/en/ipv6/, 2008. (29.04.2011).

43. İnternet: Lee, D.,"World IPv6 Day: Solving the IP Address Chicken-and-Egg Challenge", ***Facebook Inc.,*** http://www.facebook.com/notes/facebook-engineering/world-ipv6-day-solving-the-ip-address-chicken-and-egg-challenge/484445583919, 2011. (29.04.2011).

EKLER

EK-1 . Inetd tarafından gelen TCP çağrılarına cevap veren C programı

```
/** Sadece IPv4 tarafından gelen TCP çağrılarının Inetd tarafından
cevaplanması**/

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>

int main __P((int, char **));

int
main(argc, argv)
          int argc;
          char **argv;
{
     struct sockaddr_in from;
     socklen_t fromlen;
     char hbuf[INET_ADDRSTRLEN];
     fromlen = sizeof(from); /* IPv4 adreslerinin alınması */

     if (getpeername(0, (struct sockaddr *)&from, &fromlen) < 0) {

          exit(1); /* Ulaşılmadı */
     }

     if (from.sin_family != AF_INET ||
     fromlen != sizeof(struct sockaddr_in)) {

          exit(1); /* Ulaşılmadı */
     }

     if (inet_ntop(AF_INET, &from.sin_addr, hbuf, sizeof(hbuf)) ==
     NULL) {

          exit(1); /* Ulaşılmadı */
     }

write(0, "Merhaba ", 6);
write(0, hbuf, strlen(hbuf));
write(0, "\n", 1);
exit(0);

}
```

EK-1 (Devamı). Inetd tarafından gelen TCP çağrılarına cevap veren C programı

```
/** IPv6 ve IPv4 (çoklu protokol desteği) tarafından gelen TCP
çağrılarının Inetd tarafından cevaplanması. **/
/** NOT: Kalın yazı olarak gösterilen satırlar IPv6 desteğinin
sağlanması için değiştirlen kısımlardır. **/
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>
#include <arpa/inet.h>

int main __P((int, char **));

int
main(argc, argv)
          int argc;
          char **argv;
{
     struct sockaddr_storage from;
     /* IPv6/IPv4 adres. kaplayacağı alanın boyutu arttırıldı. */

     socklen_t fromlen;
     char hbuf[NI_MAXHOST];

     /* IPv6 adresi akranlarının alınması */
     fromlen = sizeof(from);

     if (getpeername(0, (struct sockaddr *)&from, &fromlen) < 0) {

          exit(1); /* Ulaşılmadı */
     }

     if (from.sin_family != AF_INET ||
     fromlen != sizeof(struct sockaddr_in)) {

          exit(1); /* Ulaşılmadı */
     }

     /* IPv6 adreslerinin daha okunaklı bir hale getirmek için
getnameinfo(3) kullanıldı. */
     if (getnameinfo((struct sockaddr *)&from, fromlen,
     hbuf, sizeof(hbuf), NULL, 0, NI_NUMERICHOST) != 0) {
          exit(1); /* Ulaşılmadı */
     }

write(0, "Merhaba ", 6);
write(0, hbuf, strlen(hbuf));
write(0, "\n", 1);
exit(0);

}
```

EK-2 . Stand-alone tarafından gelen TCP çağrılarına cevap veren C programları

```
/** Sadece IPv4 tarafından gelen TCP (tek dinleme soketi)
çağrılarının Inetd tarafından cevaplanması**/

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <arpa/inet.h>

int main __P((int, char **));

int
main(argc, argv)
        int argc;
        char **argv;
{
        struct servent *sp;
        unsigned long lport;
        u_int16_t port; char *ep;
        struct sockaddr_in serv;
        int servlen; struct sockaddr_in from;
        socklen_t fromlen;
        int s;
        int ls;
        char hbuf[INET_ADDRSTRLEN];

        if (argc != 2) {

                fprintf(stderr, "kullanım: deneme portu\n");
                exit(1); /* Ulaşılmadı */

        }

        sp = getservbyname(argv[1], "tcp");
        if (sp)
                port = sp->s_port & 0xffff;
        else {
                ep = NULL; errno = 0;
                lport = strtoul(argv[1], &ep, 10);
                if (!*argv[1] || errno || !ep || *ep) {
                fprintf(stderr, "%s: böyle bir hizmet yok\n",
argv[1]);
                exit(1); /* Ulaşılmadı */
                }
                if (lport & ~0xffff) {
                fprintf(stderr, "%s: aralıkları dışındadır\n",
argv[1]);
                exit(1); /* Ulaşılmadı */
```

EK-2 (Devamı). Stand-alone tarafından gelen TCP çağrılarına cevap veren C programları

```
                }
                port = htons(lport & 0xffff);
        }
        endservent();
        memset(&serv, 0, sizeof(serv));
        serv.sin_family = AF_INET;
        /* Linux/Solaris işletim sistemleri aşağıdaki kod satırına
ihtiyaç duymamaktadırlar*/
        serv.sin_len = sizeof(struct sockaddr_in);
        serv.sin_port = port;
        servlen = sizeof(struct sockaddr_in);

        s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
        if (s < 0) {
                perror("socket");
                exit(1); /* Ulaşılmadı */
        }

        if (bind(s, (struct sockaddr *)&serv, servlen) < 0) {
                perror("bind");
                exit(1); /* Ulaşılmadı */
        }
        if (listen(s, 5) < 0) {
                perror("listen");
                exit(1); /* Ulaşılmadı */
        }

        while (1) {
                fromlen = sizeof(from);
                ls = accept(s, (struct sockaddr *)&from, &fromlen);

                if (ls < 0)
                        continue;

                if (from.sin_family != AF_INET ||
                        fromlen != sizeof(struct sockaddr_in)) {

                        exit(1); /* Ulaşılmadı */
                }

                if (inet_ntop(AF_INET, &from.sin_addr, hbuf,
                sizeof(hbuf)) == NULL) {

                        exit(1); /* Ulaşılmadı */
                }

                write(ls, "hello ", 6);
                write(ls, hbuf, strlen(hbuf));
                write(ls, "\n", 1);
                close(ls);
        }
        /* Ulaşılmadı */
}
```

EK-2 (Devamı). Stand-alone tarafından gelen TCP çağrılarına cevap veren C programları

```
/** getaddrinfo bağlı olarak IPv6 ve IPv4 (çoklu protokol desteği)
dinleme soketleri ile çağrılarının Stand-alone tarafından
cevaplanması.**/

/** NOT: Kalın yazı olarak gösterilen satırlar IPv6 desteğinin
sağlanması için değiştirlen kısımlardır. **/


#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <arpa/inet.h>

#define MAXSOCK 20

int main __P((int, char **));

int
main(argc, argv)
       int argc;
       char **argv;
{

       struct addrinfo hints, *res, *res0;
       int error;
       struct sockaddr_storage from;

       socklen_t fromlen;
       int ls;
       int s[MAXSOCK];
       int smax;
       int sockmax;
       fd_set rfd, rfd0;
       int n;
       int i;

       char hbuf[NI_MAXHOST], sbuf[NI_MAXSERV];
       #ifdef IPV6_V6ONLY
              const int on = 1;
       #endif

       if (argc != 2) {

                fprintf(stderr, "kullanım: deneme portu\n");
                exit(1); /* Ulaşılmadı */
```

EK-2 (Devamı). Stand-alone tarafından gelen TCP çağrılarına cevap veren C programları

```
}


        memset(&hints, 0, sizeof(hints));

        /* bind(2) ile kullanılabilecek adreslerin listesinin
getaddrinfo(3) tarafından alınması*/

        hints.ai_socktype = SOCK_STREAM;
        hints.ai_flags = AI_PASSIVE;
        error = getaddrinfo(NULL, argv[1],&hints, &res0);

        if (error) {
          fprintf(stderr, "%s:%s\n", argv[1], gai_strerror(error));
          exit(1); /* Ulaşılmadı */
        }

        if (bind(s, (struct sockaddr *)&serv, servlen) < 0) {
                perror("bind");
                exit(1); /* Ulaşılmadı */
        }

        smax = 0;
        sockmax = -1;
        for (res = res0; res && smax rMAXSOCK; res = res->ai_next) {
                s[smax] = socket(res-ai_family, res-ai_socktype,
                res->ai_protocol);

                if (s[smax] < 0)
                     continue;
                /* FD_SET tekrar çalışmasını engeller.*/
                if (s[smax] = FD_SETSIZE) {
                     close(s[smax]);
                     s[smax] = -1;
                     continue;
                }

        #ifdef IPV6_V6ONLY
                if (res->ai_family == AF_INET6 &&
                setsockopt(s[smax], IPPROTO_IPV6, IPV6_V6ONLY, &on,
                sizeof(on)) < 0) {
                     perror("bind");
                     s[smax] = -1;
                     continue;
                }
        #endif

                if (bind(s[smax], res-ai_addr, res-ai_addrlen) 0) {
                     close(s[smax]);
                     s[smax] = -1;
                     continue;
                }
```

EK-2 (Devamı). Stand-alone tarafından gelen TCP çağrılarına cevap veren C programları

```
                if (listen(s[smax], 5) 0) {
                     close(s[smax]); s[smax] = -1;
                     continue;
                }

                error = getnameinfo(res-ai_addr, res-ai_addrlen,
hbuf, sizeof(hbuf), sbuf, sizeof(sbuf), NI_NUMERICHOST |
NI_NUMERICSERV);
                if (error) {
                fprintf(stderr, "deneme: %s\n",
gai_strerror(error));
                     exit(1); /* Ulaşılmadı */
                }
                fprintf(stderr, "dinliyor %s %s\n", hbuf, sbuf);

                if (s[smax] > sockmax)
                           sockmax = s[smax];
                smax++;
           }

           if (smax == 0) {
                fprintf(stderr, "Deneme: dinlenecek soket yok\n");
                exit(1); /* Ulaşılmadı */
           }

           FD_ZERO(&rfd0);
           for (i = 0; i < smax; i++)
                      FD_SET(s[i], &rfd0);

           while (1) {
                  rfd = rfd0;
                  n = select(sockmax + 1, &rfd, NULL, NULL, NULL);
                  if (n < 0) {
                         perror("select");
                         exit(1); /* Ulaşılmadı */
                  }

                  for (i = 0; i < smax; i++) {
                            if (FD_ISSET(s[i], &rfd)) {
                                   fromlen = sizeof(from);
                                   ls = accept(s[i], (struct
                                   sockaddr *)&from &fromlen);
                            if (ls < 0)
                                   continue;
                            write(ls, "Merhaba\n", 6);
                            close(ls);
                  }
           }
      }
      /* Ulaşılmadı */
}
```

EK-3 . standalone.c dosyasının değiştirilmeden önceki hali

```
/* Standalone POP server: accepts connections, checks the anti-flood
limits, logs and starts the actual POP sessions. */
#include "params.h"

#if POP_STANDALONE
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <syslog.h>
#include <time.h>
#include <errno.h>
#include <sys/times.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#if DAEMON_LIBWRAP
#include <tcpd.h>
int allow_severity = SYSLOG_PRI_LO;
int deny_severity = SYSLOG_PRI_HI;
#endif

/*
 * These are defined in pop_root.c.
 */
extern int log_error(char *s);
extern int do_pop_startup(void);
extern int do_pop_session(void);

typedef volatile sig_atomic_t va_int;

/*
 * Active POP sessions.  Those that were started within the last
MIN_DELAY
 * seconds are also considered active (regardless of their actual
state),
 * to allow for limiting the logging rate without throwing away
critical
 * information about sessions that we could have allowed to proceed.
 */
static struct {
                    struct in_addr addr;  /* Source IP address */
                    volatile int pid;     /* PID of the server, or
0 for none */
                    clock_t start;        /* When the server was
started */
                    clock_t log;          /* When we've last logged
a failure */
} sessions[MAX_SESSIONS];
```

EK-3 (Devamı) . standalone.c dosyasının değiştirilmeden önceki hali

```
static va_int child_blocked;               /* We use blocking to
avoid races */
static va_int child_pending;               /* Are any dead children
waiting? */

/* SIGCHLD handler.*/
static void handle_child(int signum)
{
                      int saved_errno;
                      int pid;
                      int i;
                      saved_errno = errno;
                      if (child_blocked)
                      child_pending = 1;
                      else {
                      child_pending = 0;
                      while ((pid = waitpid(0, NULL, WNOHANG)) > 0)
                      for (i = 0; i < MAX_SESSIONS; i++)
                      if (sessions[i].pid == pid) {
                      sessions[i].pid = 0;
                      break;
                      }
                      }
                      signal(SIGCHLD, handle_child);
                      errno = saved_errno;
}

#if DAEMON_LIBWRAP
static void check_access(int sock)
{
                      struct request_info request;
                      request_init(&request,
                      RQ_DAEMON, DAEMON_LIBWRAP_IDENT,RQ_FILE,
sock,0);
                      fromhost(&request);
                      if (!hosts_access(&request)) {
/* refuse() shouldn't return... */
                      refuse(&request);
/* ...but just in case */
                      exit(1);
                      }
}
#endif

#if POP_OPTIONS
int do_standalone(void)
#else
int main(void)
#endif
{
                      int true = 1;
                      int sock, new;
                      struct sockaddr_in addr;
                      socklen_t addrlen;
```

EK-3 (Devamı). standalone.c dosyasının değiştirilmeden önceki hali

```
                    int pid;
                    struct tms buf;
                    clock_t min_delay, now, log;
                    int i, j, n;
                    if (do_pop_startup()) return 1;
                    if ((sock = socket(AF_INET, SOCK_STREAM,
IPPROTO_TCP)) < 0)
                    return log_error("socket");
                    if (setsockopt(sock, SOL_SOCKET, SO_REUSEADDR,
                        (void *)&true, sizeof(true)))
                    return log_error("setsockopt");
                    memset(&addr, 0, sizeof(addr));
                    addr.sin_family = AF_INET;
                    addr.sin_addr.s_addr = inet_addr(DAEMON_ADDR);
                    addr.sin_port = htons(DAEMON_PORT);
                    if (bind(sock, (struct sockaddr *)&addr,
sizeof(addr)))
                    return log_error("bind");
                    if (listen(sock, MAX_BACKLOG))
                    return log_error("listen");
                    chdir("/");
                    setsid();
                    switch (fork()) {
                    case -1:
                    return log_error("fork");
                    case 0:
                    break;
                    default:
                    return 0;
}
                    setsid();
#if defined(_SC_CLK_TCK) || !defined(CLK_TCK)
                    min_delay = MIN_DELAY * sysconf(_SC_CLK_TCK);
#else
                    min_delay = MIN_DELAY * CLK_TCK;
#endif

                    child_blocked = 1;
                    child_pending = 0;
                    signal(SIGCHLD, handle_child);
                    memset((void *)sessions, 0, sizeof(sessions));
                    log = 0;
                    new = 0;
                    while (1) {
                    child_blocked = 0;
                    if (child_pending) raise(SIGCHLD);
                    if (new > 0)
                    if (close(new)) return log_error("close");
                    addrlen = sizeof(addr);
                    new = accept(sock, (struct sockaddr *)&addr,
&addrlen);
```

EK-3 (Devamı). standalone.c dosyasının değiştirilmeden önceki hali

```
if (new < 0) continue;
now = times(&buf);
if (!now) now = 1;
child_blocked = 1;
j = -1; n = 0;
for (i = 0; i < MAX_SESSIONS; i++) {
if (sessions[i].start > now)
sessions[i].start = 0;
if (sessions[i].pid ||(sessions[i].start && now-sessions[i].start <
min_delay)) {
if (sessions[i].addr.s_addr == addr.sin_addr.s_addr)
if (++n >= MAX_SESSIONS_PER_SOURCE) break;
} else
if (j < 0) j = i;
}
if (n >= MAX_SESSIONS_PER_SOURCE) {
if (!sessions[i].log || now< sessions[i].log || now-sessions[i].log
>= min_delay) {
                    syslog(SYSLOG_PRI_HI,"%s: per source limit
reached", inet_ntoa(addr.sin_addr));
                    sessions[i].log = now;
}                   continue;
}                   if (j < 0) {
                    if (!log || now<log || now-log >= min_delay) {
                    syslog(SYSLOG_PRI_HI,"%s: sessions limit
reached",inet_ntoa(addr.sin_addr));
                    log = now;
}                   continue;
}                   switch ((pid = fork())) {
                    case -1:
                    syslog(SYSLOG_PRI_ERROR, "%s: fork:
                    %m",inet_ntoa(addr.sin_addr));
                    break;
                    case 0:
                    if (close(sock)) return log_error("close");
#if DAEMON_LIBWRAP
                    check_access(new);
#endif
                    syslog(SYSLOG_PRI_LO, "Session from
                    %s",inet_ntoa(addr.sin_addr));
                    if (dup2(new, 0) < 0) return log_error("dup2");
                    if (dup2(new, 1) < 0) return log_error("dup2");
                    if (dup2(new, 2) < 0) return log_error("dup2");
                    if (close(new)) return log_error("close");
                    return do_pop_session();
                    default:
                    sessions[j].addr = addr.sin_addr;
                    sessions[j].pid = pid;
                    sessions[j].start = now;
                    sessions[j].log = 0;
}
}
}
#endif
```

EK-4 . standalone.c dosyasının değiştirilmiş son hali

```
/*
 * Standalone POP server: accepts connections, checks the anti-flood
limits,
 * logs and starts the actual POP sessions.
 */

#include "params.h"

#if POP_STANDALONE

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <syslog.h>
#include <time.h>
#include <errno.h>
#include <netdb.h>
#include <sys/ioctl.h>
#include <sys/times.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <net/if.h>

#if DAEMON_LIBWRAP
#include <tcpd.h>
int allow_severity = SYSLOG_PRI_LO;
int deny_severity = SYSLOG_PRI_HI;
#endif

//IP adresini dinleceğimiz aygıtı belirtiyoruz
#define DEVICE "eth0"
/*
 * These are defined in pop_root.c.
 */
extern int log_error(char *s);
extern int do_pop_startup(void);
extern int do_pop_session(void);

typedef volatile sig_atomic_t va_int;

static struct {
                    struct in_addr addr; /* Source IP address */
#if HAVE_INET6
                    struct in6_addr addr6; /* Kaynak IPv6 adresi */
#endif

/* IPv6 adresini de alabilsin diye değiştirildi.*/
```

EK-4 (Devamı) . standalone.c dosyasının değiştirilmiş son hali

```
volatile int pid;     /* PID of the server, or 0 for none */
clock_t start;        /* When the server was started */
clock_t log;          /* When we've last logged a failure */
}
sessions[MAX_SESSIONS];

static va_int child_blocked; /* We use blocking to avoid races */
static va_int child_pending; /* Are any dead children waiting? */
static void handle_child(int signum)
{
                      int saved_errno;
                      int pid;
                      int i;
                      saved_errno = errno;
                      if (child_blocked)
                                            child_pending = 1;
                      else {
                                            child_pending = 0;
                      while ((pid = waitpid(0, NULL, WNOHANG)) > 0)
                      for (i = 0; i < MAX_SESSIONS; i++)
                      if (sessions[i].pid == pid) {
                                            sessions[i].pid = 0;
                                            break;
                                            }
                      }
                      signal(SIGCHLD, handle_child);
                      errno = saved_errno;
}

#if DAEMON_LIBWRAP
static void check_access(int sock)
{
                      struct request_info request;
                      request_init(&request,RQ_DAEMON,
                      DAEMON_LIBWRAP_IDENT,RQ_FILE, sock,0);
                      fromhost(&request);
                      if (!hosts_access(&request)) {
                      refuse(&request);
                      exit(1);
                      }
}
#endif

#if POP_OPTIONS
                      int do_standalone(void)
#else
                      int main(void)
#endif

{
                      int true = 1;
                      int sock, new;
```

EK-4 (Devamı) . standalone.c dosyasının değiştirilmiş son hali

```
#if !HAVE_INET6
                        struct sockaddr_in addr;
                        int addrlen;
#else
                        struct sockaddr_in6 addr6;
                        struct ifreq ifr;
                        char temp[24];
                        int addrlen6;
#endif

/* IPv6 adresinin de alınacağı dosya adresi belirtirtildi*/
                        int pid;
                        struct tms buf;
                        clock_t min_delay, now, log;
                        int i, j, n;

                        if (do_pop_startup()) return 1;
//Eğer IPv6 adres var ise
#if HAVE_INET6
                        if ((sock = socket(AF_INET6, SOCK_STREAM,
                        IPPROTO_TCP)) < 0)
                        return log_error("socket");
#else
                        if ((sock = socket(AF_INET, SOCK_STREAM,
                        IPPROTO_TCP)) < 0)
                        return log_error("socket");
#endif

          if (setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, (void
          *)&true, sizeof(true)))
                    return log_error("setsockopt");

/*Adres sınıf türü, port, soket, IPv4/IPv6 adresleri için*/
#if !HAVE_INET6
                        memset(&addr, 0, sizeof(addr));
                        addr.sin_family = AF_INET;
                        addr.sin_addr.s_addr = inet_addr(DAEMON_ADDR);
                        addr.sin_port = htons(DAEMON_PORT);
#else
                        memset(&addr6, 0, sizeof(addr6));
                        addr6.sin6_family = AF_INET6;
                        addr6.sin6_port = htons(DAEMON_PORT);

                        if (inet_pton(AF_INET6, DAEMON_ADDR6,
&addr6.sin6_addr) < 0)
                        return log_error("inet_pton");

                        memset(&ifr, 0, sizeof(ifr));
                        strncpy(ifr.ifr_name, DEVICE,
                        sizeof(ifr.ifr_name));
                        if (ioctl(sock, SIOCGIFINDEX, &ifr) < 0)
                        return log_error("ioctl");
#endif
```

EK-4 (Devamı) . standalone.c dosyasının değiştirilmiş son hali

```
#if HAVE_INET6
if (bind(sock, (struct sockaddr *)&addr6, sizeof(addr6))) {
                    return log_error("bind");
}
#else
if (bind(sock, (struct sockaddr *)&addr, sizeof(addr))) {
                    return log_error("bind");
}
#endif

       if (listen(sock, MAX_BACKLOG))
                    return log_error("listen");

       chdir("/");
       setsid();

       switch (fork()) {
         case -1:
                    return log_error("fork");

         case 0:
                    break;

         default:
                    return 0;
       }

       setsid();

#if defined(_SC_CLK_TCK) || !defined(CLK_TCK)
         min_delay = MIN_DELAY * sysconf(_SC_CLK_TCK);
#else
         min_delay = MIN_DELAY * CLK_TCK;
#endif

         child_blocked = 1;
         child_pending = 0;
         signal(SIGCHLD, handle_child);

         memset((void *)sessions, 0, sizeof(sessions));
         log = 0;
         new = 0;

         while (1) {

              child_blocked = 0;
              if (child_pending) raise(SIGCHLD);

              if (new > 0)
              if (close(new)) return log_error("close");
```

EK-4 (Devamı) . standalone.c dosyasının değiştirilmiş son hali

```
#if !HAVE_INET6
                    addrlen = sizeof(addr);
                    new = accept(sock, (struct sockaddr *)&addr,
&addrlen);
#else
                    addrlen6 = sizeof(addr6);
                    new = accept(sock, (struct sockaddr *)&addr6,
&addrlen6);
#endif

          if (new < 0) continue;

          now = times(&buf);
          if (!now) now = 1;

          child_blocked = 1;

          j = -1; n = 0;
          for (i = 0; i < MAX_SESSIONS; i++) {
               if (sessions[i].start > now)
               sessions[i].start = 0;

               if (sessions[i].pid || (sessions[i].start &&
                       now - sessions[i].start < min_delay)) {
#if !HAVE_INET6
          if (sessions[i].addr.s_addr == addr.sin_addr.s_addr)
          if (++n >= MAX_SESSIONS_PER_SOURCE) break;
#else
          if (sessions[i].addr6.s6_addr == addr6.sin6_addr.s6_addr)
          if (++n >= MAX_SESSIONS_PER_SOURCE) break;
#endif

               } else if (j < 0) j = i;
          }

          if (n >= MAX_SESSIONS_PER_SOURCE) {
               if (!sessions[i].log || now < sessions[i].log ||
                       now - sessions[i].log >= min_delay) {

#if !HAVE_INET6
          syslog(SYSLOG_PRI_HI, "%s: per source limit reached",
          inet_ntoa(addr.sin_addr));
#else
          syslog(SYSLOG_PRI_HI, "%s: per source limit
          reached", inet_ntop(AF_INET6, &addr6.sin6_addr,
          temp, sizeof(temp)));
#endif

                    sessions[i].log = now;
               }
               continue;
          }
if (j < 0) {
               if (!log || now < log || now - log >= min_delay) {
```

EK-4 (Devamı) . standalone.c dosyasının değiştirilmiş son hali

```
#if !HAVE_INET6
                      syslog(SYSLOG_PRI_HI, "%s: sessions limit
                      reached", inet_ntoa(addr.sin_addr));
#else
                      syslog(SYSLOG_PRI_HI, "%s: sessions limit
                      reached", inet_ntop(AF_INET6, &addr6.sin6_addr,
                      temp, sizeof(temp)));
#endif

                    log = now;

          }

          continue;

        }

        switch ((pid = fork())) {
                    case -1:
                      #if !HAVE_INET6

                      syslog(SYSLOG_PRI_ERROR, "%s: fork: %m",
                      inet_ntoa(addr.sin_addr));

                      #else

                      syslog(SYSLOG_PRI_ERROR, "%s: fork: %m",
                      inet_ntop(AF_INET6, &addr6.sin6_addr,
                      temp, sizeof(temp)));

                      #endif

                      break;

                    case 0:
                      if (close(sock)) return log_error("close");
#if DAEMON_LIBWRAP
                      check_access(new);
#endif

#if !HAVE_INET6
                      syslog(SYSLOG_PRI_LO, "Session from %s",
                      inet_ntoa(addr.sin_addr));
#else
                      syslog(SYSLOG_PRI_LO, "Session from %s",
                      inet_ntop(AF_INET6, &addr6.sin6_addr, temp,
                      sizeof(temp)));
#endif

          if ((dup2(new, 0) < 0) || (dup2(new, 1) < 0) ||
          (dup2(new, 2) < 0)) return log_error("dup2");
                      if (close(new)) return log_error("close");
                      return do_pop_session();
                    default:
```

EK-4 (Devamı) . standalone.c dosyasının değiştirilmiş son hali

```
#if !HAVE_INET6
                        sessions[j].addr = addr.sin_addr;
#else
                        sessions[j].addr6 = addr6.sin6_addr;
#endif
                        sessions[j].pid = pid;
                        sessions[j].start = now;
                        sessions[j].log = 0;
                    }
            }
}
#endif
```

EK-5 . params.h dosyasının değiştirilmeden önceki hali

```
/** Global POP daemon parameters. */
#ifndef _POP_PARAMS_H
#define _POP_PARAMS_H
/** Our name to use when talking to various interfaces. */
#define POP_SERVER   "popa3d"
/* Are we going to be a standalone server or start via an inetd
clone? */
#define POP_STANDALONE                      0
#if POP_STANDALONE
/* Should the command line options be supported? If enabled, popa3d
will default to inetd mode and will require a -D to actually enable
the standalone mode. */
#define POP_OPTIONS                         1
/** The address and port to listen on. */
#define DAEMON_ADDR                         "0.0.0.0"/* INADDR_ANY */
#define DAEMON_PORT                         110
/* Should libwrap be used? This may make things slower and also adds
to code running as root,so it is recommended that you use a packet
filter instead.  This option is provided primarily as a way to meet
conventions of certain systems where all services obey libwrap
access controls.*/
#define DAEMON_LIBWRAP                      0
#if DAEMON_LIBWRAP
/* * How do we talk to libwrap? */
#define DAEMON_LIBWRAP_IDENT                POP_SERVER
#endif
/* Limit the number of POP sessions we can handle at a time to
reduce the impact of connection flood DoS attacks. The defaults are
rather large.  It is recommended that you decrease MAX_SESSIONS and
MAX_SESSIONS_PER_SOURCE to 100 and 10, respectively,if that would be
sufficient for your users.*/
#define MAX_SESSIONS                        500
#define MAX_SESSIONS_PER_SOURCE             50
#define MAX_BACKLOG                         5
#define MIN_DELAY                           10
#endif
/* Do we want to support virtual domains? */
#define POP_VIRTUAL                         0
#if POP_VIRTUAL
/* VIRTUAL_HOME_PATH is where the virtual domain root directories
live. */
#define VIRTUAL_HOME_PATH                   "/vhome"
/* Subdirectories within each virtual domain root for the
authentication information and mailboxes, respectively. These
defaults correspond to full pathnames of the form
"/vhome/IP/{auth,mail}/username". */
#define VIRTUAL_AUTH_PATH                   "auth"
#define VIRTUAL_SPOOL_PATH                  "mail"
/* Do we want to support virtual domains only?  Normally, if the
connected IP address doesn't correspond to a directory in
VIRTUAL_HOME_PATH, the authentication will be done globally. */
#define VIRTUAL_ONLY                        0
#else
```

EK-5 (Devamı) . params.h dosyasının değiştirilmeden önceki hali

```
/* We don't support virtual domains (!POP_VIRTUAL), so we're
definitely not virtual-only.  Don't edit this.*/
#define VIRTUAL_ONLY                        0
#endif
/* A pseudo-user to run as before authentication.  The user and its
UID must not be used for any other purpose.*/
#define POP_USER                            POP_SERVER
/* An empty directory to chroot to before authentication.  The
directory and its parent directories must not be writable by anyone
but root. */
#define POP_CHROOT                          "/var/empty"
/* Sessions will be closed if idle for longer than POP_TIMEOUT
seconds. RFC 1939 says that "such a timer MUST be of at least 10
minutes' duration", so I've made 10 minutes the default.  In
practice, you may want to reduce this to, say, 2 minutes. */
#define POP_TIMEOUT                         (10 * 60)
/** Do we want to support the obsolete LAST command, as defined in
RFC 1460?  It has been removed from the protocol in 1994 by RFC
1725, and isn't even mentioned in RFC 1939.  Still, some software
doesn't work without it. */
#define POP_SUPPORT_LAST                    1
/* Introduce some sane limits on the mailbox size in order to
prevent a single huge mailbox from stopping the entire POP service.
The defaults are rather large (2 GB filled with messages as small as
1 KB each). It is recommended that you decrease
MAX_MAILBOX_MESSAGES,MAX_MAILBOX_OPEN_BYTES, and
MAX_MAILBOX_WORK_BYTES to, say, 100000,100000000 (100 MB), and
150000000 (150 MB), respectively, if that would be sufficient for
your users.*/
#define MAX_MAILBOX_MESSAGES                2097152
#define MAX_MAILBOX_OPEN_BYTES              2147483647
#define MAX_MAILBOX_WORK_BYTES              2147483647
#if !VIRTUAL_ONLY
/* Choose the password authentication method your system uses:
AUTH_PASSWD Use getpwnam(3) only, for *BSD or readable passwd;
AUTH_SHADOW Use shadow passwords directly (not via PAM); AUTH_PAM
Use PAM in the old-fashioned way; AUTH_PAM_USERPASS Talk to
pam_userpass via Linux-PAM binary prompts USE_LIBPAM_USERPASS ...and
use libpam_userpass.Note that there's no built-in password aging
support. */
#define AUTH_PASSWD                         0
#define AUTH_SHADOW                         1
#define AUTH_PAM                            0
#define AUTH_PAM_USERPASS                   0
#define USE_LIBPAM_USERPASS                 0

#if AUTH_PAM || AUTH_PAM_USERPASS
#define AUTH_PAM_SERVICE                    POP_SERVER
#endif
#endif
#if POP_VIRTUAL || AUTH_PASSWD || AUTH_SHADOW
/*A salt used to waste some CPU time on dummy crypt(3) calls and
make it harder (but still far from impossible, on most systems) to
check for valid usernames.  Adjust it for your crypt(3). */
```

EK-5 (Devamı) . params.h dosyasının değiştirilmeden önceki hali

```
#define AUTH_DUMMY_SALT                     "xx"
#endif
/* Message to return to the client when authentication fails.  You
can #undef this for no message.*/
#define AUTH_FAILED_MESSAGE "Authentication failed (bad password?)"
#if !VIRTUAL_ONLY
/* Your mail spool directory.  Note: only local (non-NFS) mode 775
mail spools are currently supported. #undef this for qmail-style
$HOME/Mailbox mailboxes.*/
#define MAIL_SPOOL_PATH                     "/var/mail"
#ifndef MAIL_SPOOL_PATH
/* The mailbox file name relative to the user's home directory.*/
#define HOME_MAILBOX_NAME                   "Mailbox"
#endif
#endif
/* Locking method your system uses for user mailboxes.  It is
important that you set this correctly.BSDs use flock(2), others
typically use fcntl(2). */
#define LOCK_FCNTL                          1
#define LOCK_FLOCK                          0
/* How do we talk to syslogd?  These should be fine for most
systems. */
#define SYSLOG_IDENT                        POP_SERVER
#define SYSLOG_OPTIONS                      LOG_PID
#define SYSLOG_FACILITY                     LOG_DAEMON
#define SYSLOG_PRI_LO                       LOG_INFO
#define SYSLOG_PRI_HI                       LOG_NOTICE
#define SYSLOG_PRI_ERROR                    LOG_CRIT
/*There's probably no reason to touch anything below this comment.
According to RFC 1939: "Keywords and arguments are each separated by
a single SPACE character.  Keywords are three or four characters
long. Each argument may be up to 40 characters long."  We're only
processing up to two arguments, so it is safe to truncate after this
length.*/
#define POP_BUFFER_SIZE                     0x80
/* There's no reason to change this one either.  Making this larger
would waste memory, and smaller values could make the authentication
fail.*/
#define AUTH_BUFFER_SIZE                    (2 * POP_BUFFER_SIZE)
#if POP_VIRTUAL
/*Buffer size for reading entire per-user authentication files.*/
#define VIRTUAL_AUTH_SIZE                   0x100
#endif
/*File buffer sizes to use while parsing the mailbox and retrieving
a message, respectively.  Can be changed.*/
#define FILE_BUFFER_SIZE                    0x10000
#define RETR_BUFFER_SIZE                    0x8000
/* The mailbox parsing code isn't allowed to truncate lines earlier
than this length.  Keep this at least as large as the longest header
field name we need to check for, but not too large for performance
reasons.*/
#define LINE_BUFFER_SIZE                    0x20
#endif
```

EK-6 . params.h dosyasının değiştirilmiş son hali

```
#ifndef _POP_PARAMS_H
#define _POP_PARAMS_H
#define POP_SERVER                          "popa3d"
//Tek başına çalışabilmesi için değeri 1 yapıldı.
#define POP_STANDALONE                      1
#if POP_STANDALONE
#define POP_OPTIONS                         1
//IBRAHIM AKSIT -> IPv6 Desteği sağlamak için tanımlandı
#define HAVE_INET6                          1
#define DAEMON_ADDR                         "0.0.0.0"/* INADDR_ANY */
//IBRAHIM AKSIT -> IPv6 dinlenecek adres için tanımlandı
#define DAEMON_ADDR6                        "[::]" /* IPv6 için
dinlenecek adres-bütün arayüzleri dinler */
#define DAEMON_PORT                         110
#define DAEMON_LIBWRAP                      0

#if DAEMON_LIBWRAP
#define DAEMON_LIBWRAP_IDENT                POP_SERVER
#endif

#define MAX_SESSIONS                        500
#define MAX_SESSIONS_PER_SOURCE             50
#define MAX_BACKLOG                         5
#define MIN_DELAY                           10
#endif

#define POP_VIRTUAL                         0
#if POP_VIRTUAL
#define VIRTUAL_HOME_PATH                   "/vhome"
#define VIRTUAL_AUTH_PATH                   "auth"
#define VIRTUAL_SPOOL_PATH                  "mail"
#define VIRTUAL_ONLY                        0
#else
#define VIRTUAL_ONLY                        0
#endif
#define POP_USER                            POP_SERVER
#define POP_CHROOT                          "/var/empty"
#define POP_TIMEOUT                         (10 * 60)
#define POP_SUPPORT_LAST                    1
#define MAX_MAILBOX_MESSAGES                2097152
#define MAX_MAILBOX_OPEN_BYTES              2147483647
#define MAX_MAILBOX_WORK_BYTES              2147483647
#if !VIRTUAL_ONLY
#define AUTH_PASSWD                         0
#define AUTH_SHADOW                         1
#define AUTH_PAM                            0
#define AUTH_PAM_USERPASS                   0
#define USE_LIBPAM_USERPASS                 0
#if AUTH_PAM || AUTH_PAM_USERPASS
#define AUTH_PAM_SERVICE                    POP_SERVER
#endif
#endif
```

EK-6 (Devamı) . params.h dosyasının değiştirilmiş son hali

```
#if POP_VIRTUAL || AUTH_PASSWD || AUTH_SHADOW
#define AUTH_DUMMY_SALT                    "xx"
#endif
#define AUTH_FAILED_MESSAGE                "Authentication failed
(bad password?)"
#if !VIRTUAL_ONLY
#define MAIL_SPOOL_PATH                    "/var/mail"
#ifndef MAIL_SPOOL_PATH
#define HOME_MAILBOX_NAME                  "Mailbox"
#endif

#endif

#define LOCK_FCNTL                         1
#define LOCK_FLOCK                         0
#define SYSLOG_IDENT                       POP_SERVER
#define SYSLOG_OPTIONS                     LOG_PID
#define SYSLOG_FACILITY                    LOG_DAEMON
#define SYSLOG_PRI_LO                      LOG_INFO
#define SYSLOG_PRI_HI                      LOG_NOTICE
#define SYSLOG_PRI_ERROR                   LOG_CRIT
#define POP_BUFFER_SIZE                    0x80
#define AUTH_BUFFER_SIZE                   (2 * POP_BUFFER_SIZE)
#if POP_VIRTUAL
#define VIRTUAL_AUTH_SIZE                  0x100
#endif

#define FILE_BUFFER_SIZE                   0x10000
#define RETR_BUFFER_SIZE                   0x8000
#define LINE_BUFFER_SIZE                   0x20

#endif
```

EK-7 . startup.c dosyasının değiştirilmeden önceki hali

```
/*Command line option parsing.*/
#include "params.h"
#if POP_OPTIONS
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
/* version.c */
extern char popa3d_version[];
extern char popa3d_date[];
/* standalone.c */
extern int do_standalone(void);
/* pop_root.c */
extern int do_pop_startup(void);
extern int do_pop_session(void);
#ifdef HAVE_PROGNAME
extern char *__progname;
#define progname __progname
#else
static char *progname;
#endif
static void usage(void)
{
              fprintf(stderr, "Usage: %s [-D] [-V]\n", progname);
              exit(1);
}

static void version(void)
{
                     printf("popa3d version %s (%s)\n",
popa3d_version, popa3d_date);
                     exit(0);
}

int main(int argc, char **argv)
{
                     int c;
                     int standalone = 0;
                     #ifndef HAVE_PROGNAME
                     if (!(progname = argv[0]))
                            progname = POP_SERVER;
                     #endif
                     while ((c = getopt(argc, argv, "DV")) != -1) {
                                          switch (c) {
                                          case 'D':
                                          standalone++;
                                          break;
                                          case 'V':
                                          version();
                                          default:
                                          usage();
                     }
}
```

EK-7 (Devamı) . startup.c dosyasının değiştirilmeden önceki hali

```
                    if (optind != argc)
                                        usage();
                    if (standalone)
                                        return do_standalone();
                    if (do_pop_startup()) return 1;
                    return do_pop_session();
}
#endif
```

EK-8 . startup.c dosyasının değiştirilmiş son hali

```
/* * Command line option parsing.*/
#include "params.h"
#if POP_OPTIONS

/* IBRAHIM AKSIT ekledi.> socket başlık dosyasını uygulamaya
katıyoruz */
#include <sys/socket.h>
/* IBRAHIM AKSIT ekledi. */

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
/* version.c */
extern char popa3d_version[];
extern char popa3d_date[];
/* standalone.c */
extern int do_standalone(void);
/* pop_root.c */
extern int do_pop_startup(void);
extern int do_pop_session(void);
#ifdef HAVE_PROGNAME
extern char *__progname;
#define progname __progname
#else
static char *progname;
#endif
/* IBRAHIM AKSIT ekledi. * Adres sınıfına IPv6 ekledim/
int af = AF_INET6;
static void usage(void)
{
          /* Kullanım şeklinde IPv6 desteği için ekledim.*/
          fprintf(stderr, "Kullanımı: %s [-D] [-V] [-4] [-6] \n",
          progname);
          /* IBRAHIM AKSIT ekledi. */
          exit(1);
}

static void version(void)
{
          printf("popa3d version %s (%s)\n",
          popa3d_version, popa3d_date);
          exit(0);
}

int main(int argc, char **argv)
{
          int c;
          int standalone = 0;
#ifndef HAVE_PROGNAME
            if (!(progname = argv[0]))
                      progname = POP_SERVER;
#endif
```

EK-8 (Devamı) . startup.c dosyasının değiştirilmiş son hali

```
/* IBRAHIM AKSIT ekledi. */
//DV46 programın Daemon = D, Version = V, IPv4 = 4, IPv6 = 6 için
tanımlandı.
                while ((c = getopt(argc, argv, "DV46")) != -1) {
                  switch (c) {
                    case 'D':
                      standalone++;
                      break;

                    case 'V':
                      version();

                    // IPv4 ise
                    case '4':
                      af = AF_INET;
                    break;

                    // IPv6 ise
                    case '6':
                      af = AF_INET6;
                    break;

                    default:
                      usage();
                  }
            }

            if (optind != argc)
                  usage();

            if (standalone)
                  return do_standalone();

            if (do_pop_startup()) return 1;
            return do_pop_session();
}

#endif
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Soyadı, Ad	:	AKŞİT, İbrahim
Uyruk	:	T.C.
Doğum Tarihi	:	15.02.1983 Merkez/Bitlis
Medeni Durum	:	Evli
Telefon	:	0 (312) 495 81 85
Cep Telefon	:	0 (505) 684 36 08 - 0 (537) 451 83 12
E- Posta	:	ibrahimaksit@gmail.com

EĞİTİM BİLGİLERİ

Derece	Eğitim Birimi	Mezuniyet Tarihi
Yüksek Lisans	GÜ/Bilişim Enstitüsü Yönetim Bil. Sist.	2011
Lisans	ODTÜ/ Bilgisayar ve Öğretim Tekn.Eğitimi	2008
Lise	Tatvan Çok Programlı ve Endüstri Meslek Lisesi	2001

YABANCI DİL

İngilizce
Arapça

İŞ DENEYİMİ

Yıl	Yer	Görev
2009 -	GÜ Bilgi İşlem Dairesi Başkanlığı	Uzman
2009 - 2011	GÜ Bil. Müh. TÜBİTAK-KAMAG	Yazılım Müh.
2008 - 2008	ODTÜ Koleji 3 ay Staj	Bilgisayar Öğrt.
2007 - 2008	ARI Koleji 3 ay staj	Bilgisayar Öğrt.
2004 - 2008	ODTÜ Bilgi İşlem Merkezi K.D.Birimi	Bilgisayar Oper.
2004 - 2004	ODTÜ Koleji 3 ay Staj	Bilgisayar Öğrt.
2001 - 2001	Tatvan Feribot İşletmesi 3 ay Tah. Serv.	Stajer Öğrenci
2000 - 2000	Tatvan AKTEL Bilgisayarda 3 ay	Teknik Ser. Desteği

YURT DIŞI DENEYİMİ

2008 - 2008	George Mason Üniv. Yük. Lisans Eğit. Fairfax,Virginia,USA
2006 - 2008	Kültürel Değişim Programı, Virginia ve Maryland, USA

SERTİFİKALAR

2011	VMware Vsphere Inst. Conf. Man. v4.1, Bilge Adam, Ankara
2007	Cisco Certified Net. Associate(CCNA1),ODTÜ, ANKARA

HOBİLER

Yüzme, Futbol, Fitness, Dağcılık, Doğa yürüyüşü, Bilgisayar oyunları, Web programlama, Bilgi, bilgisayar ve ağ güvenliği