YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TCP/IP PROTOKOLÜNDE AKTİF BAĞLANTILARIN FARKLI FİZİKSEL KATMANLAR ARASINDA GEÇİŞ SIRASINDA KORUNMASI VE SÜRDÜRÜLMESİNE YÖNELİK BİR SİSTEMİN TASARLANMASI VE GERÇEKLENMESİ

Bilgisayar Mühendisi Erdem YILMAZ

FBE Bilgisayar Mühendisliği Anabilim Dalında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Yrd. Doç. Dr. A. Gökhan YAVUZ (YTÜ)

Prof. Dr. Sebnem Baydere

Doç. Dr. Selim Akyokuş

İSTANBUL, 2005

İÇİNDEKİLER

Sayfa

KISALTMA LİSTESİ

CN	İletişim kurulan bilgisayar (Correspondent Node)
FA	Yabancı Acente (Foreign Acent)
HA	Yerel Acente (Home Agent)
IPv4	Internet Protokolü versiyon 4
IPv6	Internet Protokolü versiyon 6
MIPv4	Mobil IP versiyon 4
MN	Mobil Cihaz (Mobile Node)
OSI	Open Systems Interconnection
TCP/IP	Transmission Control Protocol / Internet

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖNSÖZ

Yapılan çalışma süresince, her türlü konuda yakın ilgi ve desteğini gördüğüm, çalışma şevki ve bilgisinden yararlandığım danışman hocam Sayın Yrd. Doç. Dr. A. Gökhan YAVUZ'a, Prof. M. Yahya KARSLIGİL'e ve bölüm başkanımız Prof. Oya KALIPSIZ'a teşekkürlerimi sunar ve bunu bir borç bilirim. Tüm hayatım boyunca yanımda ve her konuda destek olan aileme sonsuz teşekkürlerimi sunarım.

ÖZET

Telsiz ve mobil iletişim, hareketli cihazlarda IP datagramlarının saydam bir şekilde yönlendirilmesini sağlamaktadır. Mobil cihazların yaygınlaşması ile farklı arabirimler ile bağlantının bu cihazlar üzerinden kurulabilmesi bu ihtiyacı belirginleştirmiştir. Farklı fiziksel katmanlardaki bağlantıların korunması ve sürdürülmesi, makro düzeyde bir mobilite problemi olarak karşımıza çıkmaktadır. Bu problemin çözümünde temel TCP/IP protokolünün özellikleri ve TCP/IP paket yapısı üzerinden Mobil IP yapısına duyulan ihtiyaç ortaya çıkmaktadır. Mobil cihazlarda sürekli olarak Internet bağlantı noktasının arabirimler arasında değişimi ile oluşan problemler tanımlanmış ve bunun için çözümler geliştirilmiştir. Mobil IP protokolünde birden çok arabirim kullanılması ile arabirimler arasındaki geçişin (vertical handover) gerçekleştirilmesi kullanıcı tarafından tanımlanan cihaz önceliklerine göre yapılmıştır.

Test platformunda yabancı acente (Foreign Agent) kullanılmamıştır. Mobil cihazın doğrudan yerel acenteye (Home Agent) bağlanması sağlanarak iletişimini yerel acente üzerinden tünellenen paketler ile sürdürmesi sağlanmıştır. Farklı arabirimler arasında geçiş işlemi, arabirimlerden alınan detaylı durum bilgisi ile kullanıcının belirlediği öncelik bilgisi ile birlikte kullanılarak gerçekleştirilmiştir. Bu şekilde sistemin kullanıcı tercihlerine bağlı olarak otomatik çalışması sağlanmıştır.

TCP/IP protokolünde aktif bağlantıların farklı fiziksel katmanlar arasında geçiş sırasında korunması ve sürdürülmesi sağlanmıştır. Bu sayede kullanıcının belirleyeceği önceliğe sahip arabirimler arasında geçişin otomatik olması sağlanmıştır.

Anahtar kelimeler: Mobil IP, Arabirim Durum Kontrol, Kullanıcı Öncelikleri, Mobilite Yönetimi

ABSTRACT

Mobile communication makes IP datagrams transparently routable in mobile systems. Today, demand of users for being reachable everywhere and any time has increased. This introduces high requirements for the future Internet technology to support user mobility.

The purpose of this study is to design and implement a system that protects and keeps active sessions between transitions of different physical layers in TCP/IP protocol. With this study, it is possible to switch transparently between wireless LAN, ethernet and dial-up connections according user-configured policies.

To provide connection transitions on different physical layers, we used Mobile IP protocol stack. Four different open-source Mobile IP implementations are compared and HUT (Helsinki University of Technology) Dynamics Mobile IP implementation was choosen. With the modifications to mobile node module implementation, transparent switching between interfaces are achieved. We preferred using a co-located care-of address in our testbed. In this mode, a mobile host has the maximum flexibility for acting as a host virtually attached to its home network without the need of a foreign agent.

It is assumed that mobile node is going to tunnel its outgoing and incoming packets to HA. Interface priorities and detailed device status information are combined in order to make a handover decision between different interfaces. This achieves automatic handovers with user designed policies.

Keywords: Mobile IP, Interface Detection, User Policy, User device priorities, Mobility Management

1 GİRİŞ

Bu çalışmanın amacı, TCP/IP protokolünde aktif bağlantıların farklı fiziksel katmanlar arasında geçiş sırasında korunması ve sürdürülmesine yönelik bir sistemin tasarlanması ve gerçeklenmesidir.

Günümüzde Internet, bilgi erişiminde vazgeçilmez bir unsur haline gelmektedir. Gelişen teknoloji ile kullanıcılar dizüstü bilgisayarlar, cep telefonları ve PDA (Personal Digital Assistant) 'ler gibi çeşitli cihazlar ile Internet erişimi sağlayabilmektedirler Bu cihazların üzerinde bulunan modem, ağ erişim (ethernet) kartı ve kablosuz erişim kartı ile Internet bağlantısı kurulabilmektedir. Kullanıcılar çok çeşitli bağlantı yollarından bir veya birkaçını kullanarak Internet bağlantısı kurabilmektedirler.

Bundan yaklaşık 30 yıl önce geliştirilen Internet protokolü (IP), ağların ve bu ağlara bağlı cihazların sabit olduğu prensibine göre düzenlenmiştir. Mobil cihaz ulaşılabilir olmak istediği sürece sabit bir ağa bağlanmalı ve sabit bir adres kullanmalıdır. Mobil cihaz bağlandığı ağı değiştirir ve hala iletişim kurabilmek isterse, yeni bağlandığı ağa uygun bir adres kullanmalıdır. Bu prensibe dayanılarak OSPF, RIP, BGP gibi yönlendirme protokolleri geliştirilmiştir.[1]

Şu ana kadar bu yapı ihtiyaçları karşılamış olmasına rağmen, hareketli ve telsiz cihazların ve ihtiyaçların artmasıyla yeni problemler doğmaya başlamıştır. İletişimin kesintisiz sağlanabilmesi için, mobil cihazın ağ değiştirmesine rağmen sabit ve bilinen adresini koruması gerekmektedir. Bu adresin sabitliği, TCP gibi bağlantı temelli (connection oriented) protokoller için gereklidir. TCP bağlantısının herhangi bir ucundaki adres değişirse iletişim halindeki cihazlar arasındaki bağlantı kopar. Bunun nedeni bir bağlantıyı tanımlayan unsurlardan birinin uç noktaların IP adresi olmasıdır.[1]

Bu yönlendirme probleminin çözümü Mobil IP protokolünün kullanılmasıyla mümkündür. Mobil IP, ağ seviyesi (network layer) bir çözüm olup, IP protokolünün üzerinde çalıştığı her türlü bağlantı teknolojileri (Ethernet, ATM, 802.11, 3G, vb.) üzerinde çalışır. Internet çapında uygulanabilecek, bir uçtan bir uca hareketliliği idare

edebilecek bir protokol olması sebebiyle de tercih edilen çözüm olmuştur. Değişik bağlantı teknolojilerinin kendilerine has hareketlilik protokolleri de vardır. Ancak bunların, teknolojiler arası geçişe elverişli olmadıkları ve dar alanlarda geçerli oldukları için uygulama alanları kısıtlıdır.[1]

Mobil IP yönlendirme probleminin makro düzeyde bir çözümü olarak düşünülebilir. Mobil IP protokolü ile mobil cihaz, Internet bağlantı noktasının değişmesi durumunda IP adresi değişmeyerek diğer mobil veya mobil olmayan cihazlarla iletişim kurabilecektir. Mobil cihaz kendi yerel ağında olduğu zaman mobil servisleri kullanmayacaktır. Yabancı bir ağ üzerinde olduğu zaman yabancı acente (foreign acent) aracılığı ile veya doğrudan yerel ağında bulunan yerel acente (home agent) ile bağlantı kurup yer bilgisini iletecektir. Bu durumda yerel ağında bulunan yerel acente mobil cihaza ait paketleri mobil cihazın yeni adresine yönlendirecektir.

Mobil IP protokolü yönlendirme problemine bir çözüm getirirken cihazın farklı fiziksel katmanlar arasındaki geçişi sırasındaki bağlantının sürdürülebilmesine olanak sağlamaktadır. Bu sayede farklı cihazlar arasında geçişin gerçekleştirilmesi için Mobil IP protokolü kullanılmıştır.

Farklı fiziksel katmanlar arasındaki geçiş, kullanıcının öncelikli olarak tercih edeceği cihazın aktif bağlantısı olması durumunda gerçekleşecektir. Bu sayede kullanıcı bağlantı ücreti, pil tüketimi gibi kriterlere göre her fiziksel cihaza bir öncelik verebilecektir. Aynı anda birden fazla cihaz aktif ise öncelikli olan cihaz ile yerel acenteye bağlanılacak ve diğer cihazın bağlantısı sonlandırılacaktır.

Mobil IP protokolü olarak "Helsinki University of Technology" tarafından geliştirilen "Dynamics-HUT Mobile IP" açık kaynak kodlu yazılım kullanılmıştır. Bu yazılımın modüler olması ve genişletilebilir bir uygulama arabirimüne sahip olması nedeni ile tercih edilmiştir.

Dynamic Mobil IP yazılımı üç farklı bağımsız modülden meydana gelmektedir. Bunlar yerel acente modülü, yabancı acente modülü ve mobil cihaz modülleridir. Bu

modüllerin ayrıca kullanıcı tarafından yönetimini sağlayan arabirimleri mevcuttur. Yapılan çalışma ilk olarak tüm modüllerin konfigüre edilerek birbirleri ile çalıştığının tespit edilmesidir. Ardından mobil cihaz modülü üzerinde gerçekleştirilen değişiklik ile sistemde bulunan tüm cihazların öncelikleri bir dosyadan alınarak modül tarafından kullanılması sağlanmıştır.

Cihazların bağlantı durumlarının değişimi durumunda cihazların önceliklerine göre en yüksek öncelikli cihaz üzerinden bağlantı kurulmaya çalışılmaktadır. Bağlantı kurulamaması durumunda cihaz bağlantısı sonlandırılarak en yüksek önceliğe sahip diğer cihaz ile bağlantı kurulmaya çalışılmaktadır.

Bu sayede kullanıcı tarafından önceliği belirlenmiş cihazın kullanımı sağlanmış olmaktadır. Ayrıca kullanıcı ihtiyaçları doğrultusunda öncelikleri belirleyebilmektedir.

2 LİNUX İŞLETİM SİSTEMİ

Linux, serbestçe dağıtılabilen, çok görevli, çok kullanıcılı UNIX işletim sistemi türevidir. Linux, Internet üzerinde ilgili ve meraklı birçok kişi tarafından ortak olarak geliştirilmekte olan ve başta IBM-PC uyumlu kişisel bilgisayarlar olmak üzere birçok platformda çalışabilen ve herhangi bir maliyeti olmayan bir işletim sistemidir. [2]

UNIX 70'li yılların ortalarında büyük bilgisayarlar üzerinde çok kullanıcılı bir işletim sistemi olarak geliştirilmiştir. Zaman içerisinde yayılmış ve birçok türevi ortaya çıkmıştır. UNIX ismi UNIX Research Laboratories INC şirketinin tescilli markası olduğundan dolayı birçok şirket, aynı temele dayanan işletim sistemleri için değişik isimler kullanmışlardır. Örnek olarak

- Hewlett-Packard - HP-UX,
- IBM - AIX,
- Sun Microsystems - SunOS,

isimlerini kullanmaktadır. Bugün kişisel bilgisayarlardan süper bilgisayarlara kadar biçok bilgisayar için yazılmış bulunan UNIX türevleri mevcuttur. Ne var ki bu türevlerin çoğu gelişimi belirli bir noktada durmuş ve yüksek fiyatla satılan ticari yazılımlardır.[2]

Linux, temel olarak Finlandiya Üniversitesinde öğrenci olan Linus Torvalds'ın ve Internet üzerinde meraklı bir çok yazılımcının katkıları ile geliştirilmiştir. Linux gelişimi açık bir şekilde yapılmaktadır. Bunun anlamı, işletim sisteminin her aşaması açık olarak Internet üzerinde yayınlanmakta, dünyanın dört bir yanında kullanıcılar tarafından test edilmekte, hataları ve eksiklikleri tesbit edilerek düzeltilmekte ve geliştirilmektedir. Zaman zaman bu deneme aşamaları belirli bir noktada durdurulur ve güvenilir bir işletim sistemi sunulup, geliştirme için ayrı bir seriye devam edilir. Geliştirmede yer alan bu açıklık Linux'un en büyük avantajlarından biridir. Gelişimi evrimseldir, hatalar anında kullanıcılar tarafından tesbit edilip rapor edilmekte ve birçok kişinin katkısıyla düzeltilmektedir. Bazı işletim sistemi sürümleri saatler içerisinde güncellenebilmektedir. [2]

Linux, Andy Tannenbaum tarafından geliştirilmiş olan Minix işletim sistemine dayanmaktadır. Linus Torvalds boş zamanlarında Minix'ten daha iyi bir Minix işletim sistemi yaratmak düşüncesiyle 1991 Ağustos sonlarında ilk çalışan Linux çekirdeğini oluşturmuştur. 5 Ekim 1991 tarihinde 0.02 sürümü ile Linux ilk defa tanıtılmıştır. Linus, comp.os.minix haber grubuna gönderdiği yazıda yeni bir işletim sistemi geliştirmekte olduğunu ve ilgilenen herkesin yardımını beklediğini yazmıştı. İşletim sisteminin çekirdeği için verilen numaralar kısa sürede bir standart kazandı. "*a.x.y*" seklinde belirtilen çekirdek türevlerinde "*y*" bulunulan seviyeyi, "*x*" gelişim aşamasını göstermektedir. Tek sayılı "*x*" 'ler geliştirme aşamalarını çift sayılı "*x*" 'ler ise güvenilir Linux çekirdeklerini göstermektedirler. "*a*" ise değişik Linux sürümlerini belirtir. [2]

Linux gerçekten son yıllarda hızlı bir gelişme göstermiş, çesitli ülkelerden birçok kullanıcıya erişmiş ve yazılım desteği günden güne artmıştır. Değişik kuruluşlar Linux sistemi ve uygulama yazılımlarını biraraya getirerek dağıtımlar oluşturmuşlar ve kullanımını yaygınlaştırmışlardır. [2]

3 TCP/IP

OSI (Open System Interconnection) katmanlarına göre her katman birbirinden bağımsız olarak birbirlerine daha önceden tanımlanmış fonksiyonlar sağlamaktadırlar (Şekil 3.1).

<table>
<tr><th>OSI</th><th>TCP/IP</th><th colspan="2">Protokoller</th></tr>
<tr><td>Uygulama</td><td rowspan="3">Uygulama Kümesi</td><td colspan="2" rowspan="3">HTTP FTP TELNET
SMTP SNMP</td></tr>
<tr><td>Sunum</td></tr>
<tr><td>Oturum</td></tr>
<tr><td>Aktarım</td><td>Aktarım Kümesi</td><td>TCP</td><td>UDP</td></tr>
<tr><td>Ağ</td><td rowspan="2">Ağ Kümesi</td><td colspan="2" rowspan="2">IP</td></tr>
<tr><td>Veri Bağlantı</td></tr>
<tr><td>Fiziksel</td><td>Fiziksel Katman</td><td colspan="2">IEEE 802.X, X25, PPP</td></tr>
</table>

Şekil 3.1 OSI ve TCP/IP protokol modeli [3]

TCP/IP'nin doğuşu, 1969 yılında, ABD'nin savunma sistemleri araştırma grubu (DARPA) tarafından geliştirilen bir proje ile başlamıştır. ARPANET, deneysel bir ağ durumundan, başarısı kanıtlandıkdan sonra, 1975 yılında işlevsel (kullanılan) bir ağ haline gelmiştir. 1983 yılında ise, TCP/IP yeni bir ağ standardı olarak kabul edilmiş ve ağ üzerindeki tüm konakların kullanması zorunlu hale gelmiştir. ARPANET'in, Internet haline gelmesiyle de TCP/IP, Internet dışındaki ağlarda da kullanımaya başladı. Bugün birçok şirket, TCP/IP ağları oluşturmakta ve bunun sonucu olarakda Internet, oldukça geniş kullanıcısı olan bir tüketici pazarı olarak görülmektedir.[3]

TCP/IP'nin özellikleri [3]:

- Üreticiden bağımsız olması.
- Değişik ölçekli bilgisayarları birbirine bağlayabilmesi.
- Farklı işletim sistemleri arasında veri alışverişi için kullanılabilmesi.
- UNIX sistemleriyle tam uyumluluk.
- Birçok firma tarafından birinci protokol olarak tanınması ve kullanılması.
- Internet üzerinde kullanılması.

- Minimum veri kullanımı.

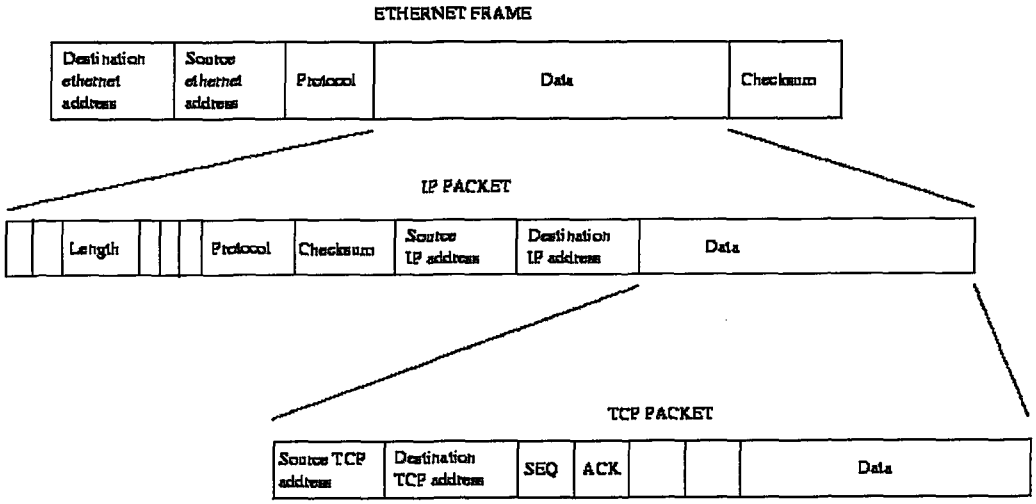
Protokollerin görevi iki bilgisayar arasındaki iletişim kurallarını düzenlemek ve verilerin gönderilmesini sağlamaktır. Bu anlamda OSI modeli içindeki yedi katmandaki görevleri yerine getirmek için gereken TCP/IP protokoller katmanı üç bölümden oluşur [3]:

1. Uygulama (Application)
2. Aktarım (Transport)
3. Ağ (Network)

Uygulama kümesinde uygulamadan-uygulamaya verilerin iletimi sağlanır. Bu kümeye örnek olarak SMTP protokolü verilebilir. Aktarım kümesinde ise bilgisayarlar arasındaki iletişim oturumu başlatılır ve güvenilir bir şekilde verilerin gönderilmesine zemin hazırlar. Ağ kümesinde ise bağlantı servisleri oluşturulur. Bu protokoller adresleme ve yönlendirme (routing) bilgilerini işlerler[3].

3.1 TCP (Transmission Control Protocol)

TCP, birbirinden farklı noktalardaki iki ağ uygulamasının sanal bir bağlantı kurarak birbirleri ile haberleştiği bağlantı temelli bir protokoldür. TCP, IP ile paketlerini gönderdiğinde, IP katmanının esas taşıdığı veri TCP verisidir (Şekil 3-2). IP paketleri karşı bilgisayarda değerlendirilirken hangi protokole ait veriyi taşıdığını belirten "Protokol" alanından faydalanmaktadır. İki bilgisayar TCP/IP protokolü ile haberleşirken hem IP hem de port adreslerinin belirtilmesi gerekmektedir. IP ve port adresi ikilisi hangi makinada hangi ağ programı ile haberleşildiğini gösteren eşsiz bir adres bütünüdür. IP katmanının altında bulunan fiziksel katman IP paketinin iletiminden sorumludur. Şekil 3.2'de fiziksel katman olarak Ethernet kullanıldığında iletilen verinin yapısı gösterilmiştir.[3]



Şekil 3.2 TCP/IP protokolünde paket yapısı [4]

TCP, güvenilir ve bağlantı (connection-oriented) temelli bir servistir. Bağlanı temelli olması bağlantının bilgisayarlar arasında veri değişiminden önce yapılması anlamına gelir. Güvenilir olması ise iletimin kontrolünün yapılması ile ilgilidir. Belli aralıklarla ACK bilgisi ile veri gönderimi kontrol edilir[3].

TCP, byte-akım (byte-stream) iletişimi kullanır. Bu yöntemde TCP segmentlerindeki veriler bir bayt dizisi olarak işlenir. Aşağıdaki tabloda TCP header içindeki ana alanlar yer almaktadır[3]:

Çizelge 3.1 TCP başlığı içindeki ana alanlar [3]

Alan	İşlevi
Source Port	Gönderenin TCP portu
Destination Port	Alanın (hedefin) TCP portu
Sequence Number	TCP segmenti içindeki birinci byte'ın sıra numarası
Window	TCP ara bellek (buffer) alanının şu anki mevcut büyüklüğü
TCP Checksum	TCP header ve TCP datanın bütünlüğünü kontrol etmek için kullanılır

3.2 IP (Internet Protocol)

Hedef bilgisayarın ağ üzerindeki yerini bulur. Paketlerin adreslenmesi ve ağ üzerindeki bilgisayarlar arasında yönlendirilmesini sağlar. IP iletimi de UDP protokolü gibi gönderimin garanti edilmediği bağlantısız bir iletişim kurar [3].

IP, iki bilgisayar (aygıt) paketlerin yönlendirilmesini sağlayan bağlantısız bir protokoldür. Bağlantısız (connectionless) olması oturumun iletişimden önce kurulmamasıyla ilgilidir. Bununla birlikte veri iletimindeki başarı da garantili olmaz. İletimin garantisi daha üst düzey protokol olan TCP ile sağlanır [3].

Bir IP paketi bir IP Header (başlık bilgisi) ve bir IP payload'tan oluşur. Aşağıdaki tabloda IP header paketinin alanları yer almaktadır [3]:

Çizelge 3.2 IP başlığı içindeki alanlar [3]

IP Başlık Alanı	İşlevi
Kaynak IP Adresi	Kaynak verinin IP adresi
Hedef IP Adresi	Gideceği yerin IP adresi
Tanımlama	Bir spesifik IP datagramını tanımlamak için kullanılır
Protokol	Paketlerin TCP, UDP, ICMP ya da diğer protokollerle iletişimi ile ilgili alan
Checksum	IP header'ın bütünlüğünü kontrol etmek için kullanılan matematiksel hesaplama
Time-to-Live (TTL)	Datagramın dolaşacağı ağ sayısını belirler. TTL sayesinde paketlerin sürekli olarak dolaşması engellenir

4 MOBİL IP

Hareketli cihazlarda (kablosuz ağ erişim kartı ile Internet'e bağlanan bir diz üstü bilgisayarı ele alırsak) IP paketlerini yönlendirme sorun oluşturmaktadır. Cihaz her farklı bağlantı noktasında farklı bir IP adresi alacağından, önceki IP ile kurulmuş olan tüm bağlantılar sona erecektir.

Mobil cihazlar için bu sorunun çözümü Internet Engineering Task Force (IETF) kuruluşu tarafından Mobile IP standardı belirlenmiştir.

Bu bölümde, Mobil IPv4 yapısı ve işleyişi anlatılacaktır. Ayrıca birden fazla fiziksel cihaz kullanılması durumunda bu cihazlar arasındaki geçişin nasıl olacağına dair bilgi verilecektir.

4.1 Mobil IPv4

IP versiyon 4 'de bir cihazın IP adresi cihazın Internet bağlantı noktasını gösteren eşsiz bir değerdir. Bu yüzden, cihaz kendisine gelen datagramları alabilmesi için IP adresinin belirtmiş olduğu ağ alanında bulunmalıdır (C. Perkins,2002). Diğer durumda bu datagramlar cihaza ulaşamayacaktır. Bağlantı noktasını sık değiştiren bir cihaz için iletişim özelliklerini kaybetmeden sürdürebilmesi için iki çözüm söz konusu olabilir :

1. Her bağlantı noktasında IP adresini değiştirebilir
2. Internet üzerinde cihaza özgü yönlendirme bilgileri yayınlanacaktır

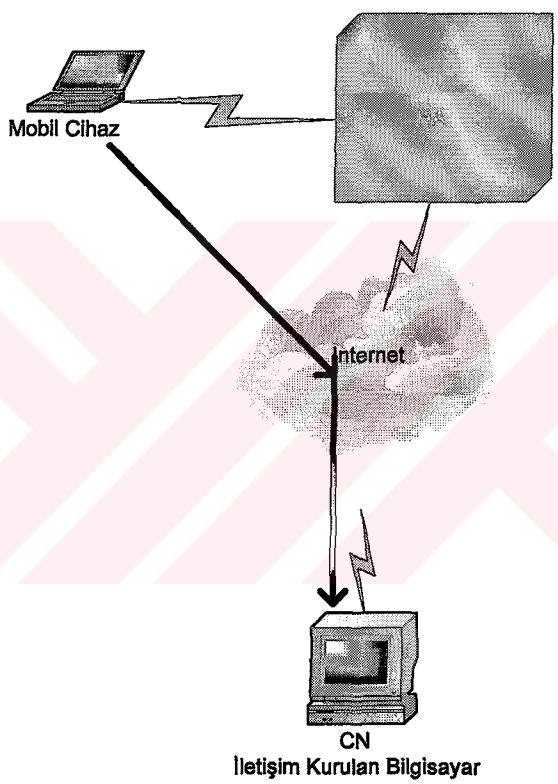
Bu iki yöntem, kullanılabilirlik açısından uygun değildir. Birinci alternatifte, aktarım katmanı ve üzerinde bulunan katmanlardaki (Şekil 3.1) bağlantının devam edebilmesi mümkün olmayacaktır. İkinci alternatifte ise mobil cihazların artışı göz önüne alınırsa ölçeklendirme problemi yaşanacaktır (C. Perkins,2002).

Mobil IP protokolü ile mobil cihaz, Internet bağlantı noktasının değişmesi durumunda IP adresi değişmeyerek diğer mobil veya mobil olmayan cihazlarla iletişim kurabilecektir.

Mobil IP standardına göre (C. Perkins,2002)

- Mobil Cihaz sabit bir adrese sahiptir (home address)
- Her ziyaret edilen ağda bir de geçici adres edinir (care-of address)
- Her yeni geçici adresi yerel ağındaki (home network) "yerel acenteye" (Home Agent - HA) bildirir.

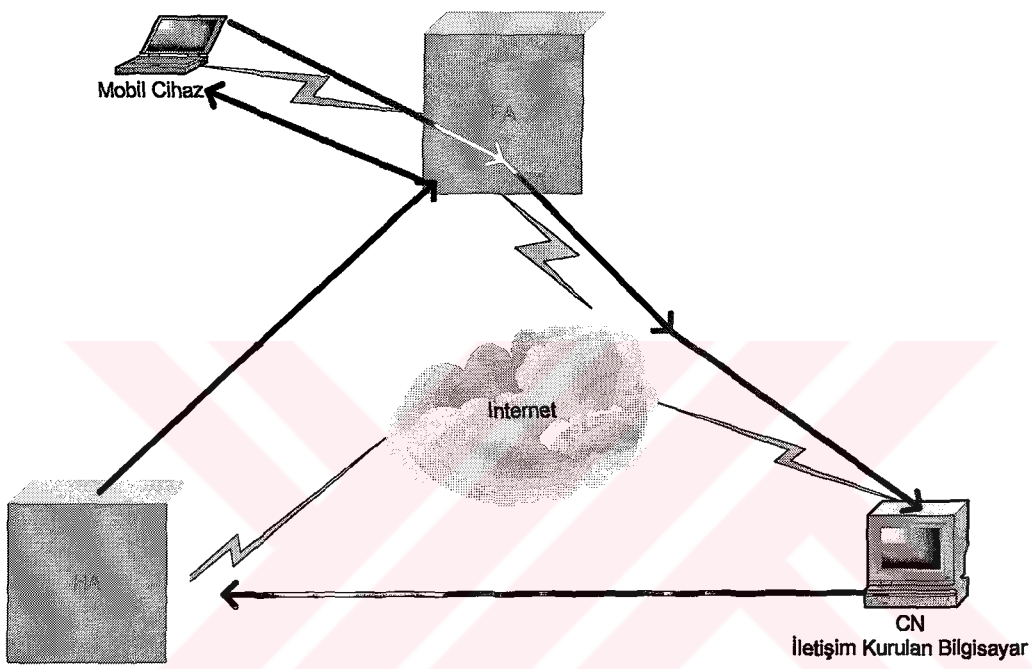
Bundan sonra paketler geçici adres üzerinden yollanır.



Şekil 4.1 Mobil Cihazın yerel ağında iletişimi

Mobil cihaz, yerel ağında olduğunu tespit ederse, cihaz mobil servisler olmadan çalışmasına devam eder. Şekil 4.1'de cihazın yerel ağında olduğu durumda iletişimi gösterilmiştir.

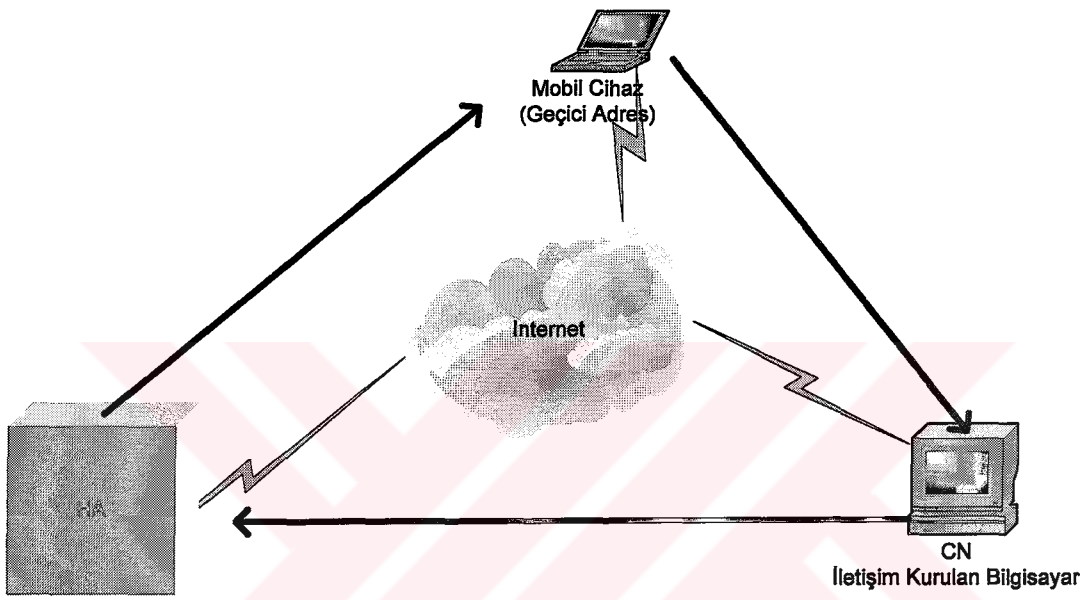
Mobil cihaz, kendi ağı dışında bir ağda ise ve ziyaret edilen ağda yabancı acente (Foreign Agent - FA) (bu bir bilgisayar veya router olabilir) varsa iletişim yabancı acente üzerinden de gerçekleştirilebilir. HA, iletişim için mobil cihaza gelen paketleri FA'ya iletir. FA ise mobil cihaza gelen tüm paketleri kendi ağında bulunan mobil cihaza iletir (Şekil 4.2) (C. Perkins,2002).



Şekil 4.2 Mobil Cihazın yabancı acente olan bir ağda haberleşmesi

Mobil cihaz farklı bir ağda iken geçici adres alır. Bu adres, eğer farklı ağda yabancı acente mevcut ise "foreign agent care of address" tir. Bu adres aslında mobil cihazın kayıt olduğu yabancı acentenin adresidir. Mobil cihaz iletişim kurmak istediği bilgisayara (Correspondent Node - CN) IP paketlerini gönderirken, gönderen kısmına (source address) sabit adresini (home address) yerleştirir. İletişim kurulan bilgisayar ise gönderilecek adres kısmına (destination address) mobil cihazın sabit adresini (home address) koyar. Yerel acenteye gelen paketler mobil cihazın kayıtlı geçici adresine (care

of address) yönlendirilir. Bu durumda kayıtlı geçici adres yabancı acente tarafından verilmiş olan IP adresi olacaktır. Bu adrese aktarılan paketler, yabancı acente tarafından mobil cihaza aktarılır. (C. Perkins,2002)



Şekil 4.3 Mobil Cihazın FA olmayan bir ağda haberleşmesi

Mobil cihazın bağlantı kurduğu ağ içerisinde yabancı acente bulunmayabilir. Mobil cihaz buna rağmen iletişimini devam ettirebilir (Şekil 4.3). Bu durumda mobil cihaz alacağı IP adresi ile (örneğin DHCP'den alınacak IP adresi) kendisini yerel acentesine tanıtır. Mobil cihazdan giden paketler doğrudan iletişim kurulan bilgisayara ulaşırken, iletişim kurulan bilgisayardan gelen paketler ise yerel acente tarafından mobil cihazın geçici adresine tünellenir. (C. Perkins,2002)

4.2 Mobil IPv4 Fonksiyonel Mekanizmaları

Mobil IP fonksiyonal olarak üç ana mekanizmadan oluşmaktadır (C. Perkins,2002):

1. Geçici adresi keşfetme (Discovering the care-of address)

2. Geçici adresi home agent'a kaydettirme (Registering the care-of address)
3. Geçici adresi tünelleme (Tunneling to the care-of address)

Geçici adresi keşfetme, iki algoritma ile gerçekleştirilebilir. Bunlar (C. Perkins,2002):

1. Yerel acente veya yabancı acente tarafından gönderilen ICMP Router duyuru paketlerinin içerisinde bulunan "Lifetime" yaşam zamanı alanı mobil cihaz tarafından tutulur. Bu süre doluncaya kadar yeni bir duyuru paketi gelmez ise mobil cihaz bu agent ile bağlantısının koptuğunu kabul edebilir. Zaman dolmadan başka bir acenteden duyuru paketi gelirse, mobil cihaz bu acenteden kayıt işlemine başlamalıdır.

2. İkinci algoritma ise ağ öneklerine (prefix) dayanmaktadır. Eğer önek uzunluğu, mobil cihazın geçici adresinin en son gelen acente duyuru paketleri ile aynı subnet içerisinde olduğunu göstermiyorsa mobil cihaz hareket ettiğini kabul edebilir.

Geçici adres, mobil cihaz tarafından yerel acenteye kayıt ettirilmelidir. Bu kayıt işlemi sırasında mobil cihaz güvenlik nedeni ile MD5 algoritması ile anahtar çiftleri kullanarak gerçekten bu kayıt işleminin sahibinin kendisi olduğunu yerel acenteye gösterir (C. Perkins,2002).

Geçici adrese tünelleme işlemi, yerel acente tarafından iletişim kurulan bilgisayardan gelen paketler için gerçekleştirilmektedir (C. Perkins,2002).

5 LİNUX NETLİNK SOKETLERİ

Netlink, Linux çekirdeği tarafından kullanıcı uzayı ve çekirdek uzayı arasında mesaj alışverişi sağlamak amacı ile verilen bir servistir. Tüm mesajlaşma ağ programlama da olduğu gibi gerçekleşmektedir. Üç sınıf mesaj tipi mevcuttur [4] :

1. Bilgi alan mesajlar,
2. Yeni bir bilgi kaydeden mesajlar ve
3. Olan bir bilgiyi silen mesajlar.

Bu çalışma boyunca çekirdekten fiziksel cihazlar ile ilgili bilgi alınması gerekmektedir. Netlink soketlerinden bilgi alan mesajlar ile dört tip istekte bulunabilir. Bunlar [4] :

1. Fiziksel arabirimler ile ilgili bilgi alma
2. Konfigüre edilmiş adres bilgilerini alma
3. Yönlendirme tablosu ile ilgili bilgi alma
4. ARP önbelleği ile ilgili bilgi alma

Netlink soketleri ile ilgili olarak çekirdekte tanımlı fonksiyonlar "linux/rtnetlink.h" dosyasında mevcuttur. Netlink soketi açmak normal bir soket açmak ile aynı kod parçacığına sahiptir (Salim vd.,2003). Netlink soketi açmak için aşağıdaki komut kullanır:

socket(AF_NETLINK, SOCK_DGRAM, NETLINK_ROUTE)

Soket tipi olarak SOCK_DGRAM veya SOCK_RAW tipleri kullanılabilir. SOCKET_RAW soket tipi sadece yönetici (root) tarafından kullanılabilmektedir. Bu soket tipinin amacı sistemde parametre değiştirmek veya yeni parametre eklemektir. SOCK_DGRAM soket tipi ise sistemden bilgi almak amaçlı kullanılan bir soket tipidir (Salim vd.,2003).

5.1 Netlink Soket Kullanımı

Netlink mesaj temelli bir protokoldür. Kısa işleyişi, çekirdek kendisine gönderdiğimiz mesajları alır, işler ve sonuçlarını asenkron olarak geri döner. Netlink soketi diğer soket işlemlerinde olduğu gibi aynı işlevlere sahiptir. Tüm giden gelen veri ortak bir yapıya sahiptir [4].

Bir netlink soketine gönderilen ve alınan tüm mesajlar Netlink mesaj başlığı içerir (Salim vd.,2003). Netlink mesaj başlığı Şekil 5.1'de belirtilmiştir.

```
struct nlmsghdr
{
    __u32       nlmsg_len;    /* Başlık dahil mesajın boyutu */
    __u16       nlmsg_type;   /* Mesaj içeriği */
    __u16       nlmsg_flags;  /* Ek bayraklar */
    __u32       nlmsg_seq;    /* Sıra numarası */
    __u32       nlmsg_pid;    /* Gönderen süreç PID */
};
```

Şekil 5.1 Netlink mesaj başlık yapısı

Bu başlık alanları bundan sonra gelen mesaj ile ilgili bilgileri içermektedir. Her farklı istek için *nlmsg_type* alanı bundan sonra gelecek mesaj yapısının belirlenmesini sağlar. Bu alan dört farklı değer alabilir. Bu değerlerin tanımlanabilmesi için baz bir değişken tanımlanmıştır (Şekil 5.2). *nlmsg_type* alanının sahip olabileceği değerler Çizelge 5.1'de gösterilmiştir (Salim vd.,2003).

```
#define RTM_BASE 0x10
```

Şekil 5.2 *nlmsg_type* alanının baz değişkeni

Sabit	Değeri	Açıklaması
RTM_GETLINK	RTM_BASE + 2	Arabirimler ile ilgili bilgi alma
RTM_GETADDR	RTM_BASE + 6	Adresler ile ilgili bilgi alma
RTM_GETROUTE	RTM_BASE + 10	Yönlendirme tablosundan bilgi alma
RTM_GETARPD	RTM_BASE + 14	ARP önbelleğinden bilgi alma

Çizelge 5.1 Netlink başlığı *nlmsg_type* alanı değerleri

Netlink mesaj başlığında bulunan *nlmsg_len* alanı, başlık veri yapısı uzunluğu ve isteğe bağlı veri yapısının uzunluğunun toplamını göstermektedir. Uzunluk alanının hesaplanması için işletim sistemi tarafından sağlanan makrolar kullanılır [4].

Netlink başlığında bulunan *nlmsg_flags* alanı Şekil 5.3'te belirtilen değerlerden bir veya birkaçını alabilir [4].

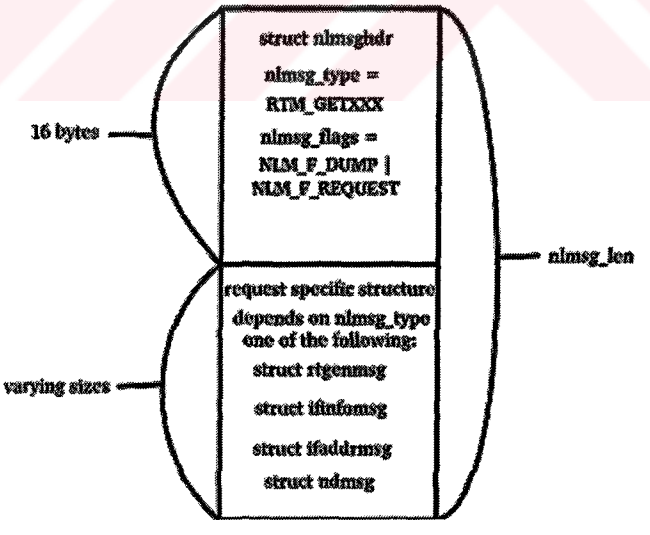
```
#define NLM_F_REQUEST      1     /* İstek Mesajı */
#define NLM_F_MULTI        2     /* Birden çok parçası olan mesaj, NLMSG_DONE ile
sonlanır */
#define NLM_F_ACK          4     /* ACK ile cevap verilip, sıfır veya hata kodu */
#define NLM_F_ECHO         8     /* Bu isteği geri gönder (echo) */

/* GET isteğinin niteleyicisi */
#define NLM_F_ROOT    0x100  /* kök ağacını gösterir */
#define NLM_F_MATCH   0x200  /* Tüm uyan sonuçları getirir */
#define NLM_F_ATOMIC  0x400  /* atomik GET */
#define NLM_F_DUMP    (NLM_F_ROOT|NLM_F_MATCH)
```

Şekil 5.3 *nlmsg_flags* alanının alabileceği tanımlı sabit değerler [4]

Çekirdekten bir bilgi isteminde bulunulduğunda, NLM_F_DUMP | NLM_F_REQUEST bayrakları kullanılmalıdır. Bu isteğe uygun tüm bilgilerin çekirdek tarafından gönderilmesini sağlar. F_MULTI bayrağı gelen mesajları soketten okurken kullanılır ve

çekirdekten gelen cevabın birden çok parçadan oluştuğunu gösterir. Maksimum UDP datagram boyutu 512 byte olduğundan çekirdekten dönen bilgi 512 byte'tan büyük ise bilgi parça parça soketten okunmalıdır. ACK ve ECHO bayrakları ise uygulama katmanı güvenilirliğini sağlamak amacı ile kullanılmaktadır. UDP protokolü güvenilir bir protokol olmadığı için paketlerin kaybolma olasılığı mevcuttur. Netlink soketleri ağ üzerinden haberleşmeye uygun yapıda tasarlandığından bu bayraklar güvenilirlik sağlamak amacı için konulmuştur. Netlink lokal makinadan kullanıldığında bu problem göz ardı edilebilir. Netlink başlığında bulunan *nlmsg_seq* ve *nlmsg_pid* alanları mesajın etiket alanı olarak görev yapmaktadır. SOCK_RAW soket tipi kullanıldığında çekirdeğe gönderilen istekler dışında da paket alma olasığı mevcuttu. Bu nedenle *nlmsg_seq* ve *nlmsg_pid* alanları ile dönen mesajın gönderilen isteğe ait olup olmadığı anlaşılabilir. *nlmsg_pid* alanı Linux sistem fonksiyonu *getpid()* ile doldurulabilir. *nlmsg_seq* alanı ise istenilen bir sayı olabilmektedir [4].

Gönderilen mesaj formatı Şekil 5.4'teki gibidir.



Şekil 5.4 Netlink istek mesajı yapısı [4]

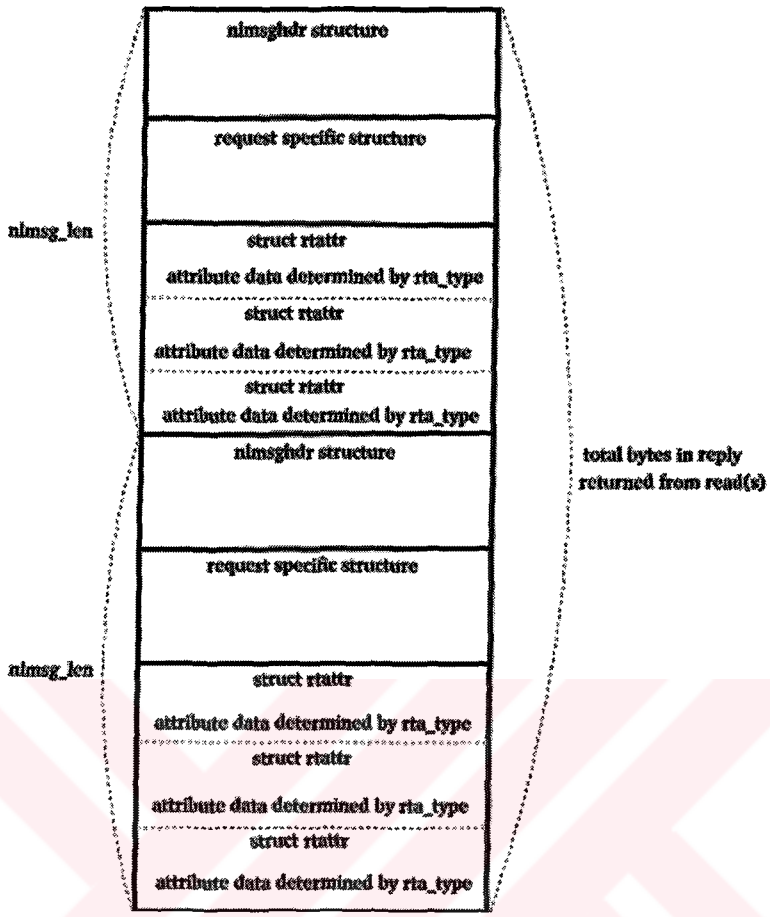
Çekirdek tarafından gönderilen cevap Netlink başlığı ile istekte gönderdiğimiz aynı yapıyı geri döner. Bunun devamında *rtattr* yapısında bir cevap dizisi döner. *rtattr* yapısı Şekil 5.5'te belirtilmiştir.

```
struct rtattr
{
        unsigned short  rta_len;
        unsigned short  rta_type;
};
```

Şekil 5.5 *rtattr* netlink cevap yapısı

Şekil 5.5'te belirtilen yapı, istenen veri ile ilgili başlık bilgilerinin saklanmasını sağlar. Bu yapının içerisinde veri bulunmamaktadır, veri ile ilgili olarak başlık bilgilerini içermektedir. Bu yapının hemen sonrasında veri mevcuttur. Bu yapıdan sonra gelen verinin tipi değişkendir. Bilgi numerik, karekter veya farklı bir veri yapısı olabilir. *rta_len* alanı *rtattr* başlığı ile bundan sonra gelen verinin toplam uzunluğunu vermektedir [4].

Gelen cevap birden fazla netlink mesajı içerebilir. Her bir cevabın, netlink başlığı, isteğe özgü veri yapısı ve bir veya birden fazla rtattr veri yapısı mevcuttur. Gelen cevabın *nlmsg_flags* alanı içerisinde *NLM_F_MULTI* bayrağı var ise soketten birden fazla veri okunmalıdır. Soketten veri okuma netlink başlığındaki *nlmsg_type* alanı NLMSG_DONE içerene kadar devam etmelidir. Gelen cevap okunurken netlink başlığındaki *nlmsg_flags* alanı birden çok okuma gerekliliğini, *nlmsg_type* alanı ise okuma sonunun gelip gelmediğini belirtir. Şekil 5.6'da örnek bir cevap mesaj arabelleği (buffer) gösterilmiştir [4].



Şekil 5.6 Netlink cevap mesajı yapısı [4]

Netlink soketleri kullanılırken veri yapılarının ve verinin doğruluğu işleyiş bakımından önemlidir. Çekirdek gelen istekleri işlerken, bazı kabullenmeler yapar. Kullanılan veri yapılarının çekirdeğe işleyişine uygun olduğunu tespit etmek için bir çok makro mevcuttur. İstek çekirdeğe gönderilmeden önce veri yapısı ile ilgili kontroller makrolar ile gerçekleştirilebilir.

Şekil 5.7 ve Şekil 5.8'de örnek iki makro prototipi gösterilmiştir.

```
void *NLMSG_DATA(struct nlmsghdr *nlh)
```

Şekil 5.7 NLMSG_DATA() makrosu [4]

Şekil 5.7'de prototipi gösterilen NLMSG_DATA makrosu netlink başlık veri tipinin işaretçisini parametre olarak alır ve veri alanında doğru düzenlenmiş bir işaretçi geri döner.

```
int NLMSG_LENGTH(size t len)
```

Şekil 5.8 NLMSG_LENGTH() makrosu [4]

Şekil 5.8'de prototipi gösterilen NLMSG_LENGTH makrosu netlink başlığından sonra gelen isteğe özgü veri tipini uzunluğunu parametre olarak alır ve mesajın toplam boyutunu döner.

6 BENZER ÇALIŞMALAR

Mobil IP protokolü ortaya atıldıktan sonra mobil cihazın ağ geçiş işlemi ile ilgili birçok çalışma gerçekleştirilmiştir. G.P. Pollini geçiş performansı ve kontrolü ile ilgili bir çalışma yayınlamıştır. Ayrıca geçişler ile ilgili yapılan araştırmaları ele almıştır. Ayrıca sinyal şiddetine göre değişik geçiş metodları ortaya koymuştur (Pollini,1996).

Mobil IP protokolü sadece mikro-mobiliteyi sağlamak amacı için geliştirilmemiş (Montavont ve Noel, 2002), ağ seviyesinde mikro-mobilite için iyi bir çözüm olarak kabul edilmemiştir (Brown ve Singh,1996;Caceres ve Padmanabhan,1998; Tan vd.,1999). Bu nedenle Mobil IP ile birlikte ve Mobil IP olmadan birçok mikro-mobilite önerileri sunulmuştur (McCann vd.,1999;Ramjee vd.,1999).

Bantgenişliğini kullanan bir yazılım veya kullanıcı sadece ağ üzerindeki veri akışına bakarak arabirim/ağ geçişini fark etmiyor ise bu geçiş görünmez "seamless" olarak tanımlanır. Bu tür geçişlerde oluşan gecikmeler veri akışından fark edilmez. Bu tür geçişi sağlamak için arabellek kullanımı (buffering) ve multicast yayın yapma yöntemleri kullanılır. Caceres ve V. N. Padmanabhan bir ağ erişim noktasında dört-paket arabellek (four-packet buffer) ile bir arabellek kullanım çözümü geliştirmiştir (Caceres ve Padmanabhan,1998). Tasarlanan geçiş protokolünde mobil cihaz geçiş işlemini başlatmakta ve yeni ağ erişim noktası eski erişim noktasına geçiş ile ilgili haber mesajı göndermektedir. Test sistemlerinde aynı alt ağlara (subnet) ait IP adresleri kullanılmıştır. Erişim noktalarının kablolu tarafında paketlerin doğru erişim noktasına yönlendirmek için proxy ve gratuitous ARP mesajları kullanılmıştır. Ölçümlenen kayıp periyodu 10 milisaniye ile 1 saniye arasındadır.

C. Perkins ve K.Y. Wang optimize edilmiş akıcı geçişler için bir yapı geliştirmiştir (Perkins ve Wang, 1999). Geçiş işleminin temeli olarak Mobil IP protokolünde arabellekler kullanılmıştır. Yabancı acenteler (Foreign Agent - FA) mobil cihaz için paketleri arabelleğe almakta ve mobil cihaz FA değişikliği gerçekleştirdiği zaman eski FA'ya bir sinyal gönderilmektedir. Bu sinyal ile eski FA arabelleğindeki tüm paketleri yeni FA'ya gönderir ve mobil cihaza gelecek tüm paketleri de mobil cihaza yönlendirir.

Paketlerin çift gitmesini önlemek amacı ile paket tanımlayıcıları kullanılmıştır. Paket arabellekleri tüm mobil cihazlar için ve birçok ağ erişim noktası için gereklidir. Bu kaynak ihtiyaçlarını arttırmakta ve sistemin genişleyebilirliğini azaltmaktadır.

S. Tekinay ve B. Jabbari, mobil hücresel ağlardaki geçiş işlemi için önceliklendirme şeması tanımlamışlardır (Tekinay ve Jabbari,1992). Önceliklendirme kablolu ağ tarafında gerçekleştirilmektedir. Sistemden daha iyi performans sağlamak amacı ile SQ (Sinyal Kalitesi – Signal Quality) değerlerinin geçişlerin önceliklendirilmesinde kullanılabileceğini göstermiştir.

Farklı geçiş politikaları, farklı önceliklendirme şemaları ile ilişkilidir. H. J. Wang ve diğerleri geçiş işleminin birçok farklı yönden ele alan geçiş politikaları ortaya koymuştur (Wang vd., 1999). Örnek olarak, herhangi bir andaki en iyi kablosuz ağ sistemi için performans, güç tüketimi ve maliyet özellikleri hesaplanıp karşılaştırılmaktadır. Wang ve diğerleri, "en" erişilebilir ağ için politikaların tanımlanması ve geçiş zamanının belirlenmesinin karmaşık olabileceğini vurgulamıştır. Tek ve elle girilmiş bir politikanın uygun olacağı kararına varmıştır.

H.J. Wang ve diğerleri, geçiş işlemi için bir senkronizasyon problemini ortaya koymuştur (Wang vd., 1999). Birçok mobil cihaz aynı yerde aynı geçiş politikasını uygular ise ağı aynı anda değiştirir ve ağın dinamik parametrelerine etki edebilir. Bu değişen dinamik parametreler politikalara etki edecek ve mobil cihazın farklı ağlar arasında salınımına neden olacaktır. Bu aynı ağ üzerinde bulunan ağ erişim noktalarını da etkileyebilmektedir. Bu probleme çözüm olarak "denge periyodu" ortaya atılmıştır. Bu süre mobil cihazın geçiş gerçekleştirmeden önce beklediği denge süresidir. Bu süre zarfında beklemesinin nedeni kararlılığın sağlanmasıdır. Denge periyodu boyunca diğer ağlar sürekli olan geçerli olandan daha iyi ise mobil cihaz geçiş işlemini gerçekleştirmektedir. Bu tür bir yaklaşım geçiş süresini gizli olarak arttırmaktadır.

7 SİSTEM TASARIMI

Sistemin tasarımı yapılırken ilk olarak sistemin ihtiyaçları belirlenmiştir. Sistemin ihtiyaçları belirlenirken mobil cihazın sağlaması gereken işlevler göz önünde bulundurulmuştur. Sistemin kısıtlamaları tasarımda aşamasında tespit edilmişdir.

7.1 Sistemin İhtiyaçları

Sistemin aşağıda listelenen ihtiyaçları karşılaması öngörülmüştür :

- Yerel veya yabancı bir ağ üzerinde olan mobil cihaz bağlantılarının ağ değişiminden bağımsız olarak kalıcı olması,
- Ethernet, Modem ve kablosuz ağ kartının bağlantı durumlarının otomatik kontrolü,
- Bu üç fiziksel cihaza kullanıcı tarafından önceliklerin tanımlanabilmesi ve bu tanımlanan önceliklere göre cihazlar arasında bağlantıların kopmadan geçmesi,
- Fiziksel cihazlar arasında geçiş olduktan sonra eski cihazın bağlantısının kesilmesi.

7.2 Sistemin Kısıtları

Sistemde kısıtlama olarak tüm fiziksel arabirimlerin (Ethernet, Modem, kablosuz ağ kart vb.) mobil cihaza sabit olduğu kabul edilmiştir. Özellikle kablosuz bağlantı arabiriminin PCMCI arabirimünden takılıp çıkartılabilmesi göz önünde bulundurulmamıştır. Mobil IP protokolü gereği mobil cihazın yabancı bir ağda mobil işlemleri sağlayabilmesi için iki seçeneği mevcuttur.

Birinci seçenek doğrudan yerel acenteye bağlanarak iletişimin kurulmasıdır. İkinci seçenek ise yabancı acente aracılığı ile yerel acenteye bağlantının sağlanmasıdır.

Yabancı acente kullanımının sağlayacağı avantaj ve dezavantajlar (Baker vd., 1996):

Avantajlar :

- Ana avantaj mobil cihazın FA olmayan ağları da kullanabilir olmasıdır,
- **Hata toleransı** : Mobil cihazın iletişimi FA' a bağımlı olmayacaktır. HA yerel ağda olduğu için bir problem durumunda kolaylıkla yeniden başlatılarak veya hata saptanarak giderilebilir. FA, yabancı bir ağda olduğu için bir problem durumunda erişim olanağı olmayacaktır.
- **Genişliyebilirlik** : Her ağda FA olmasına gerek olmayacaktır.
- **Sade protokol** : Tam bir FA modülü uygulamasının olmaması protokolün daha sade olmasını sağlayacaktır.

Dezavantajlar:

- Ana dezavantaj mobil cihazın yabancı ağda geçici bir adres alması gerekliliğidir.
- **Güvenlik** : Mobil cihaz yabancı ağdan çıktığı zaman, mobil cihaza gelen paketler ağa yeni gelmiş olan ve mobil cihazın eski adresini kullanan cihaza ulaşacaktır. Pratikte DHCP sunucusu mobil cihazın eski adresini uzun süre koruyacaktır. Bu problemden daha önemli olarak HA ve MN arasında yeni adresin kaydı sırasındaki paketlerin şifrelenme sorunudur. Bunun için kayıt paketleri Kerberos, PGP gibi şifreleme yöntemleri ile şifrelenmelidir.
- **Paket kaybı** : FA'lar paket kaybını azaltmaktadır. Yerini değiştiren MN, yeni adresini HA'a bildirmek zorundadır. CN tarafından HA'a yeni kayıt paketi ulaşmadan gönderilen tüm paketler MN'un eski adresine gönderileceğinden bu paketler MN'a ulaşmayacaktır. Eski ağda bulunan FA, bu durumda MN'a gelen paketleri MN'un yeni adresine yönlendirebilir.
- **Daha karmaşık MN modülü** : FA olmadığından dolayı mobil cihaz, FA tarafınan yerine getirilen görevleri gerçekleştirmek zorunda kalacaktır.

Yukarıda verilen kıyaslama sonucunda test platformu için mobil cihazın doğrudan yerel acenteye bağlanmasına karar verilmiştir.

7.3 Mobil IP Protokol Uygulamaları

Mobil cihazın, yerel veya yabancı bir ağ üzerinde ağ değişiminden bağımsız olarak bağlantılarının kalıcı olması için mobil IP protokolü kullanılmıştır. Mobil IP protokolü üçüncü bölümde anlatıldığı üzere hareketli cihazlarda yönlendirme sorununu çözmek amacı ile geliştirilmiş olan bir protokoldür. Bu protokolün kullanılması ile yerel acente ihtiyacı ortaya çıkmaktadır. Yerel acente, yerel ağ üzerinde bulunan ve mobil cihazın yabancı bir ağda olduğu durumlarda mobil cihaza gelen paketleri mobil cihazın bulunduğu adrese ileten cihazdır.

Farklı fiziksel cihazlar arasında geçişin sağlanması mobil cihaz üzerinde gerçekleşeceği için mobil cihaz üzerine Mobil IP protokol yığını (stack) yüklenecektir.

Mobil IP protokolü için dört farklı yazılım incelenmiştir. Bunlar :

1. Manuel Rodríguez tarafından HP laboratuvarında geliştirilmiş olan Mobil IP yazılımı [5]
2. The Portland State Üniversitesi tarafından geliştirilen Mobil IP yazılımı [6]
3. Stanford Üniversitesi MosquitoNet Mobile Computing Group tarafından geliştirilen MosquitoNet Mobil IP yazılımı [7]
4. Helsinki University of Technology (HUT) tarafından geliştirilen Mobil IP yazılımı [8]

Manual Rodríguez tarafından geliştirilen yazılımın özellikleri [5]:

- RFC 2002[11] ile tamamen uyumluluk (Yabancı acente ARP – Address Resolution Protocol - isteklerine cevap verme problemine karşın belirli periyodik aralıklarla ARP cevapları gönderilmektedir).
- Linux çekirdeğine ek olarak çalışmadığı için çekirdeğe yük getirmemektedir.
- Platform bağımsızdır. Çekirdek değişiklikleri içermediği için kolayca diğer platformlara taşınabilir.
- Güvenliği zorunlu tutar. Yerel acenteye yer bildirimleri içeren tüm yer kaydı paketleri yabancı acente üzerinden gerçekleşmek zorundadır.

- Küçük program parçalarından oluşur (30-40 KB).
- Birden fazla arabirim desteği mevcut değildir.

The Portland State Üniversitesi tarafından geliştirilen Mobil IP yazılımının özellikleri ise [6]:

- FreeBSD için geliştirilmiştir.
- Linux için kısıtlı bir versiyonu mevcuttur. Bu versiyon sadece mobil cihaza ait yazılımı içermektedir, HA ve FA modülleri mevcut değildir.
- NAT (Ağ Adres Çözümleme - Network Address Translation) desteği sadece FreeBSD sistemi üzerinde geliştirilmiş olup Linux versiyonunda bulunmamaktadır.
- Linux ile birlikte çalıştığında yerel acente veya yabancı acenteden bir paket almadığı zaman DHCP protokolü ile bir IP adresi alıp otomatik olarak yerel acente bağlantısı kurmaya çalışır.
- Linux ile birlikte gelen DHCP istemcisi ile beraber çalışmamaktadır.

Stanford Universitesi MosquitoNet Mobile Computing Group tarafından geliştirilen MosquitoNet Mobil IP yazılımının özellikleri [7] :

- Redhat Linux 7.0 işletim sisteminde çalışmaktadır.
- Linux çekirdeği üzerinde bazı değişiklikler içermektedir. Ayrıca kullanıcı uzayında çalışan yazılıma sahiptir.
- RFC 2002 (C. Perkins,1996) ile uyumludur.
- Linux 2.4 çekirdeği desteği içermemektedir.

Helsinki University of Technology (HUT) tarafından geliştirilen Mobil IP yazılımı özellikleri [8]:

- Linux çekirdeğinde değişiklik içermemektedir.
- Platform bağımsızdır. Çekirdek değişiklikleri içermediği için kolayca diğer platformlara taşınabilir .
- Birden fazla fiziksel arabirim desteği mevcuttur.

- Kablosuz ağlarda önceliğe göre geçiş desteği mevcuttur.
- Yabancı acente kullanımı isteğe bağlıdır.
- RFC 2002 (C. Perkins,1996)], RFC 2344 (Montenegro,1998) desteği mevcuttur.
- Yazılımla sağlanan API aracılığı ile genişletilebilir.

Kullanılacak Mobil IP yazılımı seçiminde gereksinimlerin yanı sıra test platformunda kullanılacak cihazlar ile uyumlu çalışması gerekliliği göz önünde bulundurulmuştur. Test platformunda kullanılacak olan kablosuz ağ kartının Linux sürücüleri çekirdek versiyonun 2.4 ve üstü ile çalışmaktadır. Ayrıca test platformunda HA aracılığı ile Mobil IP özellikleri sağlanacağından HA modülü gereksinimi mevcuttur.

Bu yazılımlar arasında Helsinki University of Technolgy tarafından geliştirilmiş olan Dynamics adlı Mobil IP yazılımı tercih edilmiştir. Bunun nedeni HA ve MN modül desteği sağlamasının yanı sıra test sisteminde FA kullanım zorunluluğu getirmemesidir. Ayrıca test platformunda kullanılacak olan kablosuz kartın çalıştığı çekirdek versiyonu ile uyumludur. Bunun yanı sıra birden fazla fiziksel arabirim desteği sunması, yazılımının modüler olması ve sağlanan API ile ek geliştirmelerin kolayca yapılabilir olması tercihin Dynamics yazılımı yönünde olmasını sağlamıştır.

Dynamic yazılımı farklı fiziksel cihazlar üzerinden mobil IP işlevselliği sağlayabilmektedir. Ayrıca kablosuz cihazlar için kablosuz erişim noktaları arasında verilen öncelikler ile geçiş gerçekleştirebilmektedir.

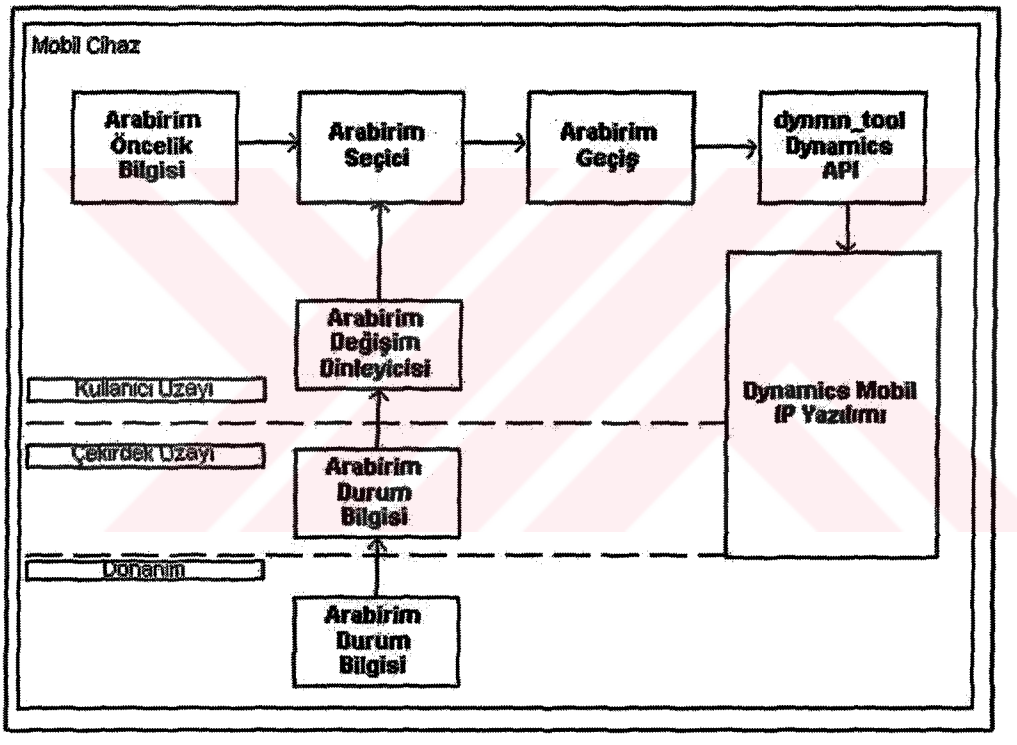
Mobil cihaz, geçiş kararını fiziksel arabirimlerin durumu ile bu arabirime ait önceliğe bakarak karar vermektedir. Farklı fiziksel arabiriminlerin iletişim kalitesi, hızı gibi kriterler göz önünde bulundurulmamaktadır.

7.4 Dynamics yazılımı ile entegre çalışan uygulama tasarımı

Sistem tasarımı yapılırken iki farklı tasarım yapılmıştır. Bunlardan birincisi Dynamics yazılımının mobil cihaz modülünde bir değişiklik gerçekleştirmeden geçiş işleminin

otomatik gerçekleştirilmesidir. Bunun için Şekil 7.1'de ki yapı tasarlanmıştır. Bu yapıda bulunan modüller :

- Arabirim Değişim Dinleyicisi
- Arabirim Öncelik Bilgisi
- Arabirim Seçici
- Arabirim Geçiş
- dynmn_tool arabirimi ile Dynamics yazılımında geçişin gerçeklenmesi sağlayan modüllerdir.

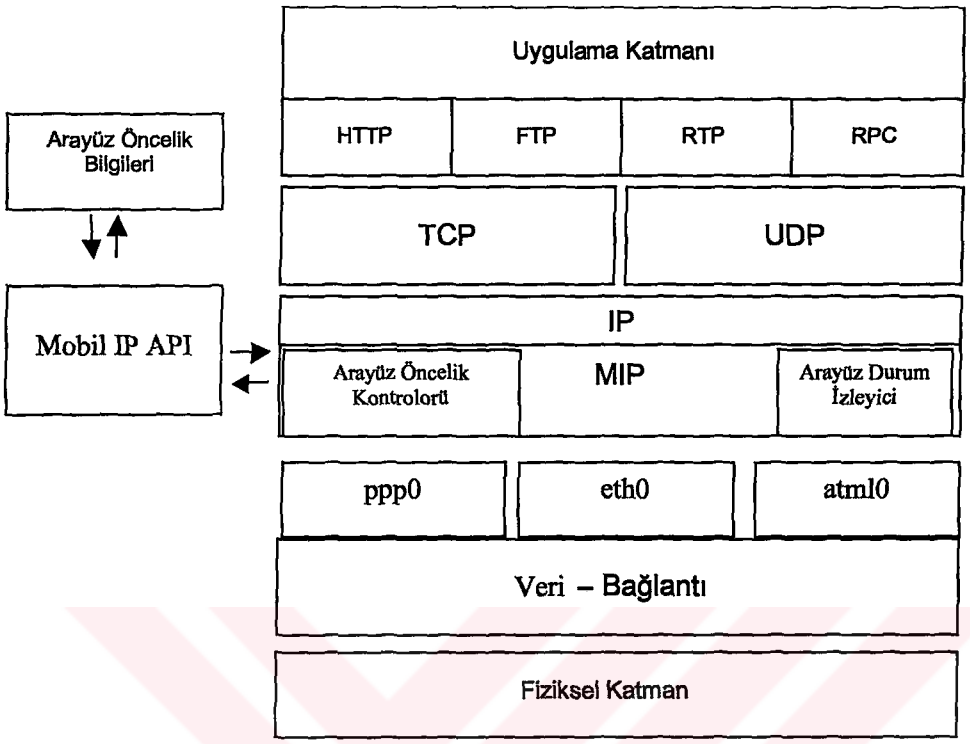


Şekil 7.1 Tasarlanan Birinci Sistemin Genel Yapısı

7.5 Dynamics yazılımında yapılan geliştirmelerin tasarımı

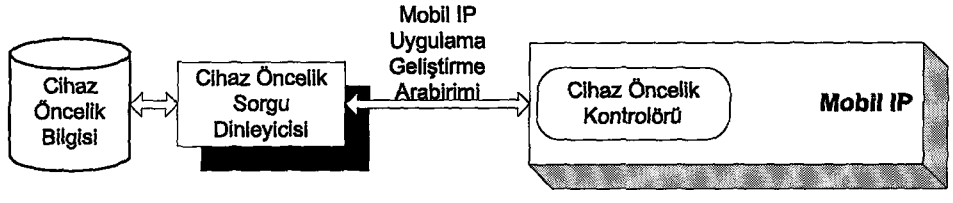
Sistem tasarımında yapılan ikinci tasarım ise Dynamics yazılımında geliştirme yapılmasıdır. Dynamic yazılımında gerçekleştirilecek değişiklik ile Şekil 7.2'de gösterilen yapı oluşturulacaktır. Mobil IP protokol yığınına arabirimler ile ilgili olarak

kullanıcıdan gelen bilgilerin aktarılacağı bölüm ile Mobil IP yığınında bu bilgilerin işleneceği bölümdür.



Şekil 7.2 Tasarlanan İkinci Sistemin Genel Yapısı

Mobil cihazın ethernet, modem ve kablosuz ağ kartı ile ağa bağlantı sağlayacağı kabul edilmiştir. Bu cihazlara öncelik verebilmek amacı ile Şekil 7.3'de gösterilen yapı tasarlanmıştır. Bu yapı ile yeni bir cihaz tespit edildiğinde Mobil IP protokolünden gelen sorgu paketi "Cihaz Öncelik Sorgu Dinleyicisi" tarafından alınır ve cihazla ilgili olarak daha önce kullanıcı tarafından verilen öncelik dönülür.

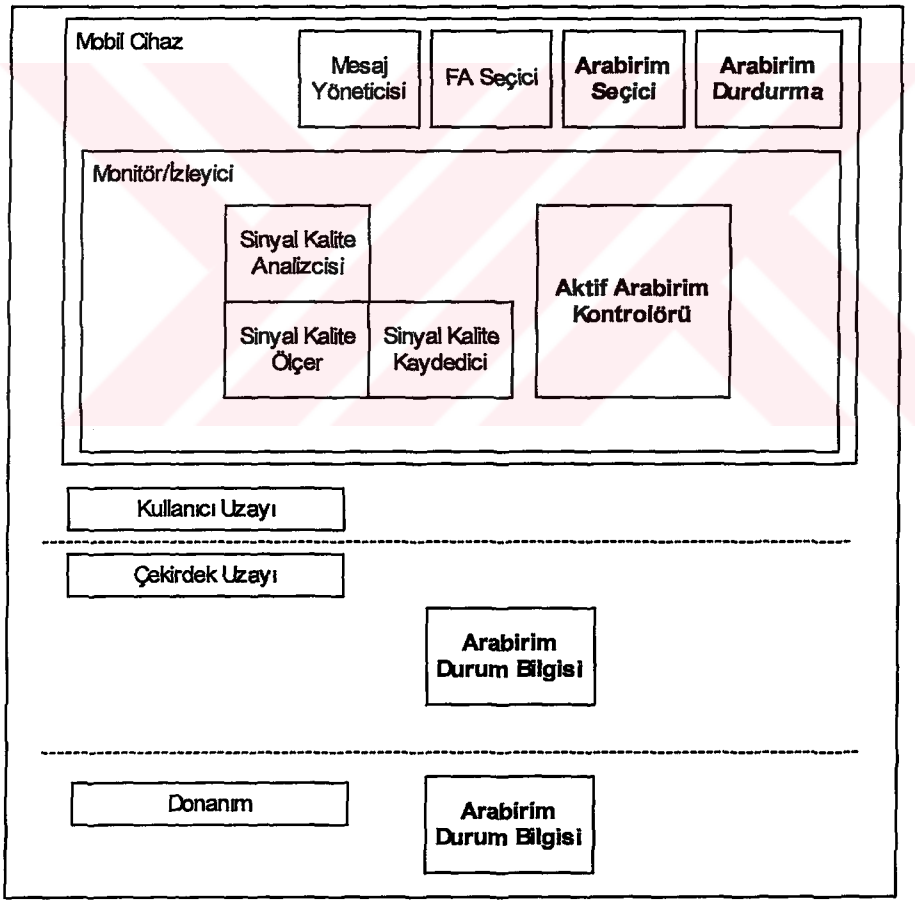


Şekil 7.3 Dynamics Mobil IP yazılımında cihaz öncelik tanımlaması

Fiziksel cihazlar arasında geçiş gerçekleştikten sonra erişim maliyetini düşürmek ve mobil cihazın pil tüketimini azaltmak amacı ile geçişin gerçekleştiği cihazın bağlantısı kesilmektedir. Bağlantı ancak yüksek öncelikli cihazın bağlantısın kopması durumunda düşük öncelikli cihaza geçmektedir.

Fiziksel cihazlar arasında geçiş tamamlandıktan sonra aktif bağlantıların koparılması sistem tarafından otomatik olarak gerçekleştirilecektir.

Mobil cihaz üzerinde Dynamics yazılımı tarafından sağlanan dinleyici arabirimi genişletilerek cihaz durum bilgisininde alınması sağlanmıştır. Bu yapı şekil 7.4'te görüldüğü üzere gerçekleştirilmiştir.



Şekil 7.4 Dynamics Mobil IP yazılımında arabirim durum bilginin alınması [13]

Mobil cihazda gerçekleşen farklı olaylarda çengel (hook) desteği için özel yönetim arabirimleri Dynamics yazılımında mevcuttur. Acente duyuru mesajları ve fiziksel arabirim cihazlarının çıkartılıp takılması gibi olaylar bu yönetim arabirimleri tarafından kaydedilmektedir. Bu yapının genişleyebilir olması sayesinde yeni olay tipleri için yeni çengeller tanımlanabilmektedir (Forsberg,2000).

Dynamics yazılımında iki tür olay mevcuttur. Bunlar Çizelge 7.1'de listelenen fiziksel arabirim olayları ve FA olaylarıdır. Fiziksel arabirim olayları yeni fiziksel arabirim tespiti (INTERFACE_INIT) ve fiziksel arabirim devreden çıkma tespiti (INTERFACE_DOWN) şeklinde iki çeşittir. FA olayları kategorisi ise FA acente duyurusu süre sonu (FA_ADV_EXPIRE), FA yeni duyuru alımı (FA_ADV_RECEIVE) ve en iyi FA adayı seçme (FA_GET) olaylarıdır (Forsberg,2000).

Bu olaylardan fiziksel arabirim olayları, fiziksel arabirimin çalışmasını kontrol etmemektedir. Örnek olarak kablolu Ethernet kartının aktif olduğu durumda kablo durumu kontrolü mevcut değildir. Bu durumda Ethernet kartının kablosu takılı olmasa dahi arabirim aktif kabul edilmektedir. Bu durumu engellemek amacı ile yeni fiziksel arabirim tespiti (INTERFACE_INIT) bölümünde bu kontrollerle ilgili geliştirmeler gerçekleştirilmiştir.

Çizelge 7.1 Dynamics yazılımında bulunan çengeller(Forsberg,2000)

Olay	Çengel
INTERFACE_INIT	mn_default_INTERFACE_INIT_handler monitor_interface_up
INTERFACE_DOWN	mn_default_INTERFACE_DOWN_handler monitor_interface_down
FA_ADV_EXP	monitor_del_fa
FA_ADV_RECEIVE	monitor_add_fa
FA_GET	mn_default_FA_GET_handler monitor_get_fa

Dynamics yazılımında bulunan olay-çengel yönetim mekanizması kablosuz ağlar için geliştirilmiş ve özelleştirilmiştir. Bu mekanizma sonradan geliştirilip yazılıma eklendiğinden dolayı bir modül olarak çalışmaktadır. Bu modül Dynamics yazılımında

bulunan fiziksel arabirim olayları ile paralel çalışmaktadır. Dynamics yazılımı Linux çekirdeğinden Netlink soketleri yardımı ile fiziksel arabirim değişim olaylarını almaktadır.

Dynamics yazılımında yer alan özelliklere ek olarak gerçekleştirilen sistem üç bölümden oluşmaktadır. Fiziksel arabirim öncelik bilgi sorgularının cevaplanacağı arabirimin başlatılması, Dynamics yazılımında fiziksel arabirimlerin durum bilgilerinin detaylı olarak algılanması ve durum bilgisi ile birlikte öncelik bilgisi ile arabirim arası geçişin gerçeklenmesidir.

7.5.1 Öncelik Bilgileri Sorgu Dinleyicisi

Öncelik bilgileri sorgu dinleyicisi Dynamics yazılımına arabirimler ile ilgili öncelik bilgilerini sağlar. Yazılım sırası ile

1. Konfigürasyon dosyalarını okuyarak fiziksel arabirim öncelik bilgilerini alır, veri yapılarına yerleştirir,
2. Dynamics yazılımından gelecek isteklere yanıt verebilmek amacı ile Dynamics API ortak arabirimine kendini kaydeder,
3. API aracılığı ile gelen öncelik bilgisi isteklerini cevaplar.

7.5.2 Arabirim Detaylı Durum Kontrolorü

Dynamics yazılımında çekirdekten gelen fiziksel arabirim durum değişim bilgileri detaylı durum bilgileri içermez. Detaylı durum bilgileri arabirim türüne göre arabirimin aktif olup olmaması bilgisidir. Yazılım arabirimle ilgili olarak detaylı durum bilgisini sağlamakla görevlidir. Yazılım arabirim değişim bilgisi çekirdek tarafından iletildikten sonra çalışır.

7.5.3 Arabirim Seçici

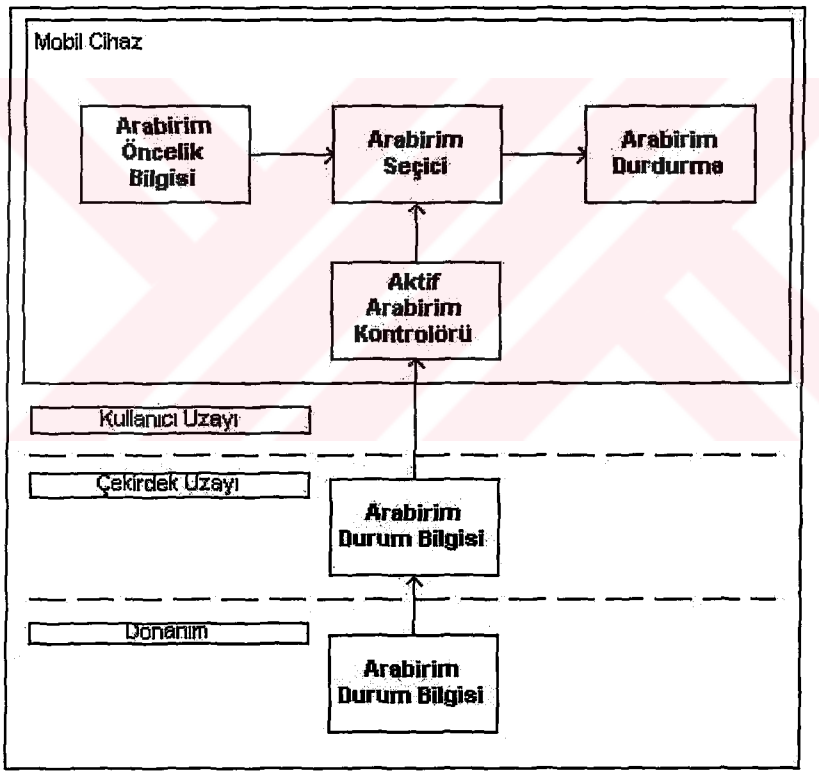
Arabirim seçici, Arabirim detaylı durum kontrolörüden elde edilen bilgilere göre çalışır. Arabirim çalışıyor, durumu aktif ise arabirimin önceliğine bakılır. En yüksek öncelikli arabirim seçilir. Eğer kullanılan arabirim en yüksek önceliğe sahip ise bir değişiklik

gerçekleşmez. Eğer farklı bir cihaz yüksek önceliğe sahip ise bağlantılar yüksek öncelikli cihaza aktarılır ve bu konuda arabirim sonlandırıcıya bilgi verilir (Şekil 7.5).

7.5.4 Arabirim Sonlandırıcı

Arabirim seçiçi tarafından gelen veriye göre ilgili arabirimin sonlandırılmasını sağlar. Bir arabirimin sonlandırılması pil tüketimi, bağlantı ücreti gibi noktalarda önem kazanmaktadır.

Arabirim sonlandırma işlemi sonrasında sonlanan arabirim kullanıcı tarafından tekrar aktive edilmedikçe kullanılamayacaktır.



Şekil 7.5 Tasarlanan İkinci Sistemin Genel Yapısı

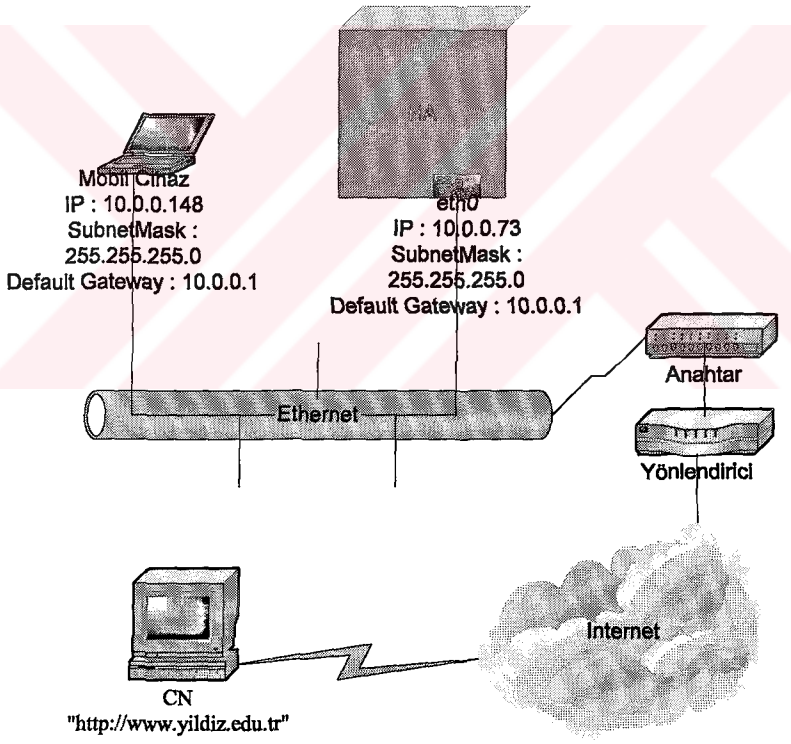
7.6 Sistem Ağ Tasarımı

Sistem tasarımı gerçekleştirilirken kullanılacak ağ yapısı da belirlenmiştir. İlk olarak Dynamics yazılımında değişiklik gerçekleştirilmeden çalıştırılmıştır. Bunun için mobil cihaz ve yerel acente (Home Agent - HA), Şekil 7.6'da olduğu gibi kurulmuştur.

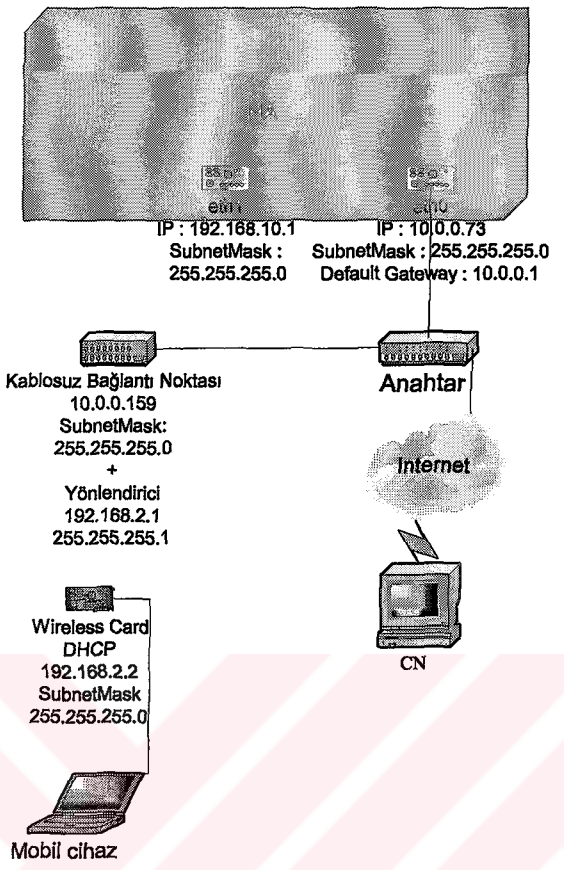
HA, Redhat Linux Advanced Server 2.1 işletim sistemine sahip 10.0.0.0/24 ağında statik olarak 10.0.0.73 numaralı IP'si olan bir cihazdır. Bu cihaz üzerine Dynamics yazılımının HA modülü yüklenmiştir.

Mobil cihaza ise Redhat 9.0 işletim sistemi yüklenmiştir. Mobil cihaza statik olarak 10.0.0.148 numaralı IP adresi verilerek Dynamics yazılımının mobile cihaz modülü kurulmuştur.

Bu yapıda HA tarafından gönderilen acente duyuru mesajları mobil cihaz tarafından alınmış ve mobil cihaz yerel ağda olduğu tespit etmiş, mobilite özellikleri olmadan iletişimini sağlamıştır.



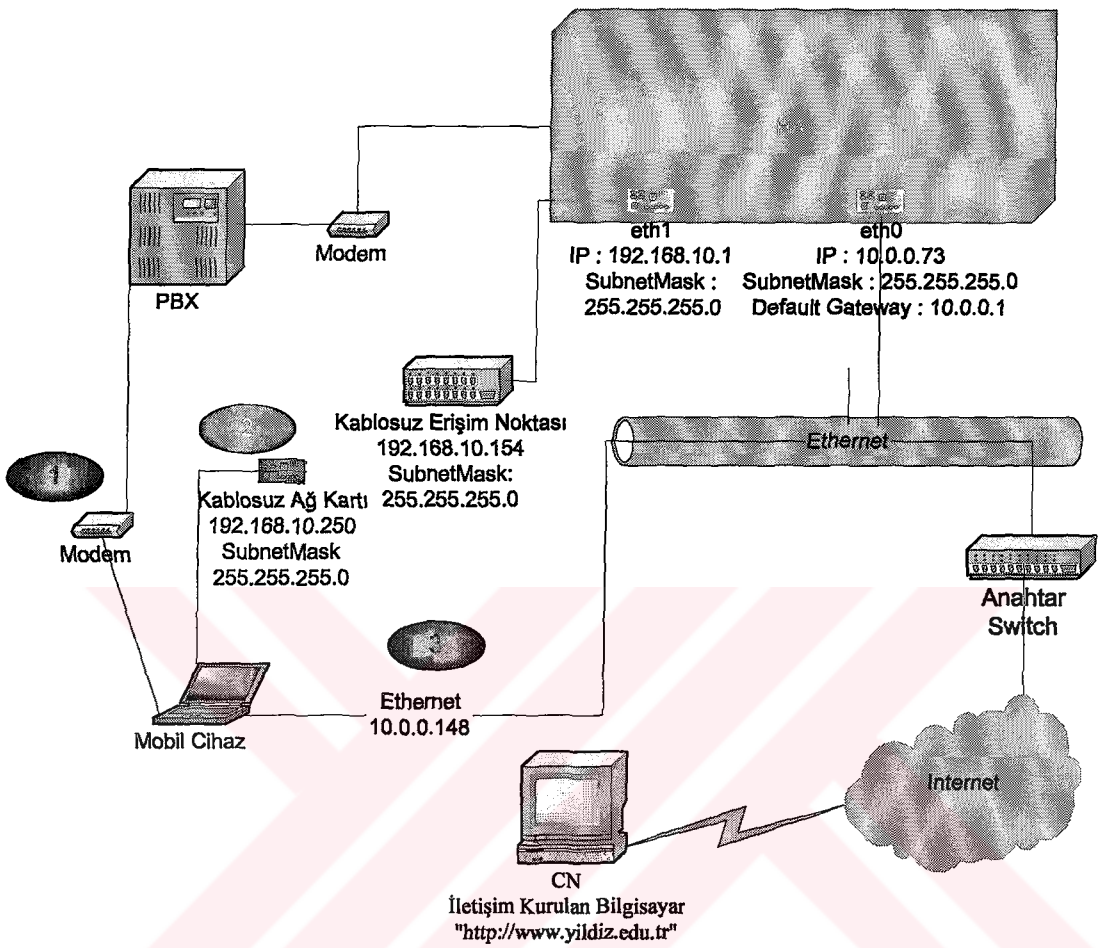
Şekil 7.6 Mobil Cihazın yerel ağında çalışması



Şekil 7.7 Mobil Cihazın yabancı ağda iletişimi

Mobil cihazın mobilite özelliklerini Internet gibi genel bir ağ üzerinde test edebilmek Internet Servis Sağlayıcısının güvenlik ayarları ile ilgili bilgi gerektirdiğinden sistem dahili bir ağ içerisinde test edilecektir. Bunun için Şekil 7.7'de bulunan yapı kurulmuştur. Bu yapı dahilinde kullanılan kablosuz bağlantı noktası aynı zamanda yönlendirici olduğundan dolayı farklı bir ağ segmenti daha yaratılabilmiştir.

Mobil cihaz, yabancı ağda DHCP ile bir IP adresi aldıktan sonra HA'a bağlanarak geçici adresini bildirmiş ve iletişimini HA'ya tünelleme yaparak sağlamıştır.



Şekil 7.8 İkinci Tasarımın Genel Ağ Yapısı

Sistemde mobil cihazın kablosuz Ethernet, kablolu Ethernet ve modem arasındaki geçişlerinin tespiti amacı ile Şekil 7.8'deki ağ yapısı kurulmuştur. Bu ağ yapısında ana hedef mobil cihazın üç farklı fiziksel arabirim arasında geçişinin sağlanmasıdır.

Mobil cihaz üç farklı fiziksel arabirim ile doğrudan HA'ya bağlanmaktadır. Farklı fiziksel arabirimler farklı ağlar üzerinden HA'ya bağlanmaktadır.

Dynamics yazılımının konfigürasyonu hem mobil cihazda hem de HA üzerinde gerçekleştirilmiştir. Bunun ile ilgili konfigürasyon detayları EK 1 ve EK 2' de belirtilmiştir.

8 UYGULAMA

Sisteme ait uygulama, tasarımda ortaya çıkan iki farklı yapıya göre geliştirilmiştir. Yapılan iki tasarımın uygulamaları Redhat Linux 9.0 işletim sisteminde ve C programlama dili ile gerçekleştirilmiştir. Yapılan birinci tasarıma göre Dynamics yazılımında bir değişikliklik gerçekleştirilmeyecektir. İkinci yapıda ise Dynamics yazılımında gerçekleştirilen geliştirmeler ile arabirimler detay bilgileri alınıp geçiş kararı verilmiştir.

Her iki uygulamanın FA olmadan çalışabilmesi amacı ile Dynamics yazılımında değişiklik gerçekleştirilmiştir. Dynamics yazılımı HA'ya tünelleme işlemini sadece API aracılığı ile verilen komut sonrası gerçekleştirmektedir. Bu komut "dynmn_tool" yazılımı ile verilen "connect HA" komutudur. Diğer durumlarda mobil cihaz modülü HA veya FA'nın gönderdiği acente duyurularını beklemektedir. Mobil cihaz bir acente duyurusu almadığı sürece bağlantısını sağlayamamaktadır. Yapılan tasarımda (Şekil 7.8) HA sadece üçüncü erişim noktası olan Ethernet üzerinden acente duyuru mesajları göndermektedir. Bu nedenle sadece üçüncü bağlantı noktasında mobil cihaz Internet erişimi sağlayabilecektir. Bu nedenle Dynamics yazılımında doğrudan HA'ya tünelleme yapılması sağlanmıştır. Bunun için Dynamics yazılımında "api/dyn_api.h" dosyasında bulunan *tunneling_modes* enum tipi kullanılmıştır. Bu enum tipinin API_TUNNEL_FULL_HA değeri ana tünelleme tipi olarak atanmıştır.

8.1 Dynamics yazılımı ile entegre çalışan uygulama

Gerçekleştirilen ilk uygulamada Dynamics yazılımının "dynmnd_tool" arabirimi ile haberleşmektedir. "dynmnd_tool" arabirimi mobil cihazın izlenmesi ve kontrol edilebilmesi sağlayan bir arabirimdir. Bu arabirim Dynamics yazılımının API fonksiyonlarını çağıran bir yazılımdır. Bu yazılıma verilebilecek parametreler Ek 3'te detaylandırılmıştır.

Uygulama ilk başladığında "/etc/device_info.conf" dosyasından arabirimlere ait öncelik bilgilerini alıp *device_priorities* veri yapısına yerleştirmektedir. Daha sonra yeni bir

process yaratarak *interfaceListener* fonksiyonunu çağırmaktadır. Bu fonksiyon *netlinkOpen* fonksiyonunu çağırarak çekirdek ile haberleşmeyi sağlayan bir netlink soket açılmasını sağlar.

Çekirdekten arabirim cihaz listesi alınarak *device_priorities* veri yapısındaki en yüksek önceliğe ait cihazdan başlayarak arabirim durumunu *interfaceStatus* adlı fonksiyondan alınır.

Bir arabirim için tanımlanan durum bilgisi üç farklı tipten biri olabilir. Bunlar arabirim aktif, arabirim aktif değil ve arabirim erişim hatası şeklindedir. Bunun için *interface_status_type* enum yapısı kullanılmaktadır

interfaceStatus fonksiyonu arabirimin durumu ile ilgili olarak link durum bilgisini dönmektedir. Bunun için üç farklı fonksiyon kullanmaktadır. Bu fonksiyonlar :

***interface_detect__WLAN*:** Kablosuz ağ kartı için kablosuz erişim noktasına bağlı olduğunu, sinyal seviyesinin belirlenen aralıklarda olduğunu kontrol eder.
***interface_detect_ ETHTOOL_GLINK*:** Kablolu ağ kartı için arabirimin aktif olduğu ve Ethernet kablosunun takılı olduğu bilgisini kontrol eder
***interface_detect_ PPP*:** Parametre olarak verilen arabirimin çevirmeli bağlantı arabirimi olup olmadığını kontrol eder.

interfaceListener fonksiyonu en yüksek önceliğe sahip arabirimin durum bilgisini kontrol ettikten sonra eğer arabirim durumuda aktif ise *makeHandover* fonksiyonu ile arabirim geçişini gerçekleştirir.

makeHandover fonksiyonu aktif arabirim ile geçilmek istenen arabirimin aynı olup olmadığını kontrol eder. Eğer geçiş yapılmak istenen arabirim farklı ise "dynmn_tool" yazılımı *execl* sistem fonksiyonu kullanılarak çağrılır. *execl* sistem fonksiyonu çalışan prosesi parametresinde verilen yeni bir proses ile değiştirir (Matthew ve Stones, 2004). "dynmn_tool" yazılımına verilen parametre istenilen arabirime geçiş yapılmasının sağlanmasıdır.

Arabirim değişim bilgileri çekirdekten alınmaktadır. Çekirdekten gelen arabirim değişim bilgilerini almak amacı ile *select* sistem fonksiyonu kullanılmaktadır. Bu sistem fonksiyonu bir programın birden çok noktadan girdi beklemesini sağlar (Matthew ve Stones, 2004). Netlink soketi tarafından gelen arabirim değişim bilgisi *interfaceListener* fonksiyonu tarafından kontrol edilmektedir. Netlink soketi ile çekirdekten gelen arabirim değişim bilgisi *iface_info* veri yapısına yerleştirilmektedir.

Yazılımı sonlandırmak için CTRL+C tuşlarına basılmalıdır. CTRL+C tuşlarına basıldığının anlamak için Linux işletim sisteminde sinyalleşme mekanizması kullanılmıştır. Sinyal bir proseste oluşan bir duruma karşı Linux sistemi tarafından yaratılan bir olaydır (Matthew ve Stones, 2004). CTRL+C tuşlarına basılması Linux sisteminde SIGINT sinyalinin oluşmasını sağlar. Bu sinyal oluştuğunda gerçekleşecek işlemi değiştirmek amacı ile *signal* sistem fonksiyonu kullanılır. Bu fonksiyon ile SIGINT sinyali alındığında *destroyProcess* fonksiyonunun çağrılması sağlanır.

destroyProcess fonksiyonu daha önce açılmış olan netlink soketini kapatır. Ayrıca veri yapıları için tahsis edilmiş olan bellek alanını işletim sistemine iade eder.

Yazılımda kullanılan fonksiyon prototipleri ve veri yapıları Ek 4'te detaylandırılmıştır.

8.2 Dynamics yazılımında yapılan geliştirmeler

Bu yapıda 8.1'de kullanılan ayrı bir yazılım yerine Dynamics yazılımı içinde yeni fonksiyonlar geliştirilerek arabirimler arasında otomatik geçiş sağlanmıştır.

8.2.1 Öncelik Bilgileri Sorgu Dinleyicisi Modülü

Öncelik bilgileri sorgu dinleyicisi Dynamics yazılımına arabirimler ile ilgili öncelik bilgilerini sağlar. Dynamics yazılımı yeni bir arabirim tespit ettiğinde sağlamış olduğu API aracılığı ile yeni arabirimin öncelik bilgisini almaktadır. Bu modül API aracılığı ile Dynamics yazılımına kayıt olmaktadır. Ardından gelen öncelik bilgisi isteklerini cevaplamaktadır.

Yazılım, başladığı zaman "/etc/device_info.conf" dosyasını okuyup cihaz önceliklerini bu dosyadan almaktadır. Ardından yazılım kendini "/var/run/dynamics_device_info" arabirimi ile Dynamics API birimine kaydeder. Dynamics yazılımı ile haberleşme soketler üzerinden gerçekleşmektedir. Bu yüzden *select* sistem fonksiyonu ile açılan soket üzerinden istek beklenir. Bir istek geldiğinde gelen istek dosyadan okunan öncelik bilgileri ile cevaplanır. Gelen arabirim ile ilgili olarak öncelik bilgisi mevcut değil ise -2 değeri dönülür.

8.2.2 Arabirim Detaylı Durum Kontrolorü

Dynamics yazılımına bir arabirim aktif olduğunda arabirim ile ilgili detaylı durum bilgisini sağlayan modüldür. Bu modül daha önceki uygulamada kullanılan *interfaceStatus* fonksiyonunu kullanmaktadır. Bu fonksiyon arabirimin durumu ile ilgili olarak link durum bilgisi dönmektedir.

Bu modül Dynamics yazılımında *check_interfaces* fonksiyonu tarafından çağrılmaktadır. Bu metod yeni bir arabirim tespit edildiğinde çalışmaktadır. Yeni bir arabirim olduğunu veya olan bir arabirimin devre dışı kalmasında yazılım içindeki ilgili fonksiyonları çağırmaktadır. Yeni bir arabirim tespit edildiğinde *modify_new_interface* fonksiyonunu çağırmaktadır. Eğer arabirim durum detaylı kontrolorü tarafından cihaz aktif bilgisi alınmaz ise *modify_new_interface* fonksiyonu çağrılmamaktadır. Arabirimlerden bir tanesi devre dışı kaldığında ise *modify_removed_interface* fonksiyonu çağrılmaktadır.

8.2.3 Arabirim Seçici

Arabirimler arasında geçiş yapılıp yapılmamasına karar veren ve bu karara göre geçiş işleminin gerçekleşmesini sağlayan modüldür. Bu modül *check_interfaces* fonksiyonu tarafından çağrılmaktadır. Yeni bir arabirim tespit edildiğinde arabirim detaylı durum kontrolorü tarafından cihaz aktif bilgisi alındıktan sonra *modify_new_interface* fonksiyonu çağrılır. Ardından arabirimler arasında geçiş kararının verilmesi için *makeHandoverDecision* fonksiyonu çağrılır. Bu fonksiyon yeni arabirimin öncelik

değerlerini diğer aktif arabirimlerle karşılaştırır. Eğer yeni arabirim en yüksek önceliğe sahip ise arabirim geçişi yapılması üzere *makeHandover* fonksiyonu çağrılır. Bu fonksiyon kendisine parametre olarak gelen arabirime geçişi gerçekleştirir.

8.2.4 Arabirim Sonlandırıcı

Geçiş işleminden sonra diğer arabirimlerin sonlandırılması için arabirim sonlandırıcı modülü çağrılır. Bu modül *makeHandover* fonksiyonundan sonra çağrılmaktadır. Geçiş işlemi gerçekleştikten sonra diğer arabirimlerin devre dışı bırakılması için *deactivateInterfaces* fonksiyonu çağrılır.

deactivateInterfaces fonksiyonu aktif arabirim dışındaki tüm arabirimleri devre dışı bırakır. Bu sayede kullanıcı geçiş işleminden sonra eski bağlantılarını yeni arabirim üzerinden sürdürebilmekte ve eski arabirimlerin de otomatik olarak sonlanması sağlanmaktadır.

9 SİSTEMİN TEST EDİLMESİ

Tasarımı yapılan ağ yapısı (Şekil 7.8) üzerinde her iki uygulama test edilmiştir. Yapılan testler sonucunda arabirim değişimleri otomatik olarak yazılımlar tarafından gerçekleştirilmiştir. Ayrıca kullanıcı tarafından cihazlara verilen öncelikler sistem tarafından işlenip arabirimlerin durumları ile birlikte değerlendirilmiştir.

Yapılan testlerde "Ethereal" adlı açık kaynak kodlu yazılım kullanılmıştır. Bu yazılım ile mobil cihazın HA'ya gönderdiği paketler incelenmiştir. Bu paketlerde HA'ya kayıt mesajları ve HA'dan gelen cevap mesajları tespit edilmiştir.

Mobil cihazın yerel ağında HA'dan gelen acente duyuru mesajlarını aldığı ve kayıt iptal paketini gönderdiği tespit edilmiştir.

Yapılan testlerde aşağıdaki değişkenlere yönelik değerler elde edilmiştir:

- Arabirim aktivasyonunun algılanması, karar mekanizmasının çalışması ve yeni arabirim için gerekli veri yapılarının kurulması, kayıt işlemi için geçen süre – geçiş süresi (t1 milisaniye),
- Arabirimün devre dışı olmasının algılanması ve aktif başka bağlantı tespit süresi – bağlantı olmaması durumu (t2 milisaniye),
- Arabirimün devre dışı olmasının algılanması ve aktif başka bağlantı tespit süresi – başka arabirim tespit edilip geçiş durumunda (t3 milisaniye).

Test aşamasında her iki uygulama için bu değerler tespit edilmiştir. Testler beş kere tekrarlanmış ve ortalama sonuçları alınmıştır. Bu değerler birinci uygulama için Çizelge 9.1'de, ikinci uygulama için Çizelge 9.2'de verilmiştir.

Çizelge 9.1 Uygulama 8.1 için test sonuçları

Test No / Süre	t1	t2	t3
1	3201 milisaniye	53 milisaniye	9363 milisaniye
2	2539 milisaniye	51 milisaniye	8209 milisaniye
3	2010 milisaniye	56 milisaniye	9746 milisaniye
4	2505 milisaniye	54 milisaniye	9960 milisaniye
5	2173 milisaniye	53 milisaniye	8978 milisaniye
Ortalama Değer	2485 milisaniye	53,4 milisaniye	9251,2 milisaniye

Çizelge 9.2 Uygulama 8.2 için test sonuçları

Test No / Süre	t1	t2	t3
1	704 milisaniye	64 milisaniye	3906 milisaniye
2	695 milisaniye	58 milisaniye	4607 milisaniye
3	695 milisaniye	66 milisaniye	4847 milisaniye
4	572 milisaniye	59 milisaniye	4504 milisaniye
5	695 milisaniye	62 milisaniye	4078 milisaniye
Ortalama Değer	672 milisaniye	61,8 milisaniye	4388,4 milisaniye

Çizelge 9.1 ve 9.2'deki "t1" alanı Dynamics yazılımı tarafında gerçekleşen yeni arabirim veri yapılarının kurulması ve kayıt işlemi süresi ile ek yazılımın arabirim değişimini algılaması ve karar mekanizmasının geçiş kararı alma süreleri toplamıdır. "t1" süresinin ölçümlemesinde Şekil 7.8'de tasarlanan ağ üzerinde sadece ethernet bağlantısı aktive edilmiştir. Cihaz öncelikleri

1. Kablosuz Ağ Arabirim
2. Ethernet arabirim
3. Çevirmeli Ağ Arabirim

olarak sistemde tanımlanmıştır. Kablosuz ağ kartının aktive edilmesi ve önceliğe göre bu arabirime geçişin gerçekleşmesi süresi ölçümlenmiştir. Bu süreye kablosuz ağ kartının deaktive süresi eklenmemiştir.

"t2" alanı ise aktif bağlantının kopması durumunda bu durumun algılanması ve geçiş yapılabilecek farklı arabirim arama süresidir. Bu süre dahilinde yeni arabirime geçiş işlemi mevcut değildir. Sistem "t2" zaman ölçümü için tek bir arabirim ile çalıştırılmıştır. "t2" süresinin ölçümlemesinde Şekil 7.8'de tasarlanan ağ üzerinde sadece ethernet bağlantısı aktive edilmiştir. Cihaz öncelikleri

1. Ethernet arabirim
2. Çevirmeli Ağ Arabirim
3. Kablosuz Ağ Arabirim

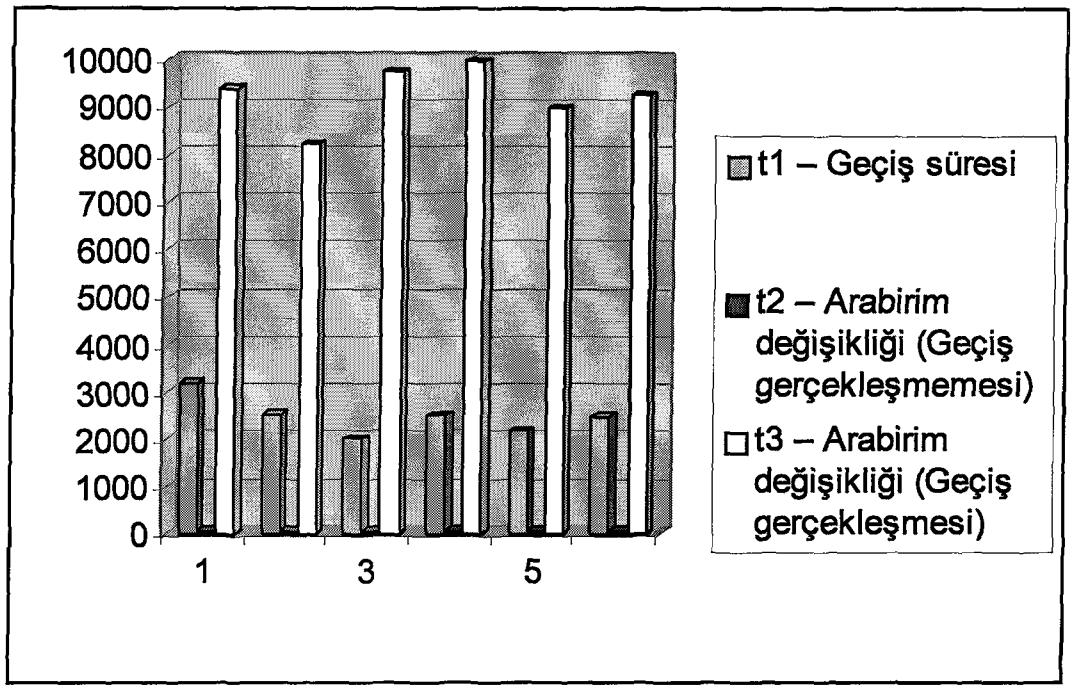
olarak sistemde tanımlanmıştır. Ethernet kablosu çıkartıldığı ve diğer bağlantıların aktif olmaması durumunda sisteminin algılama süresi ölçülmüştür.

"t3" süresi ise aktif bağlantının koptuğu durumda bunun otomatik algılanması ve yeni bir arabirime geçiş için geçen süre miktarıdır. "t3" süresinin ölçümlemesinde Şekil 7.8'de tasarlanan ağ üzerinde sadece kablosuz ağ bağlantısı aktive edilmiştir. Cihaz öncelikleri

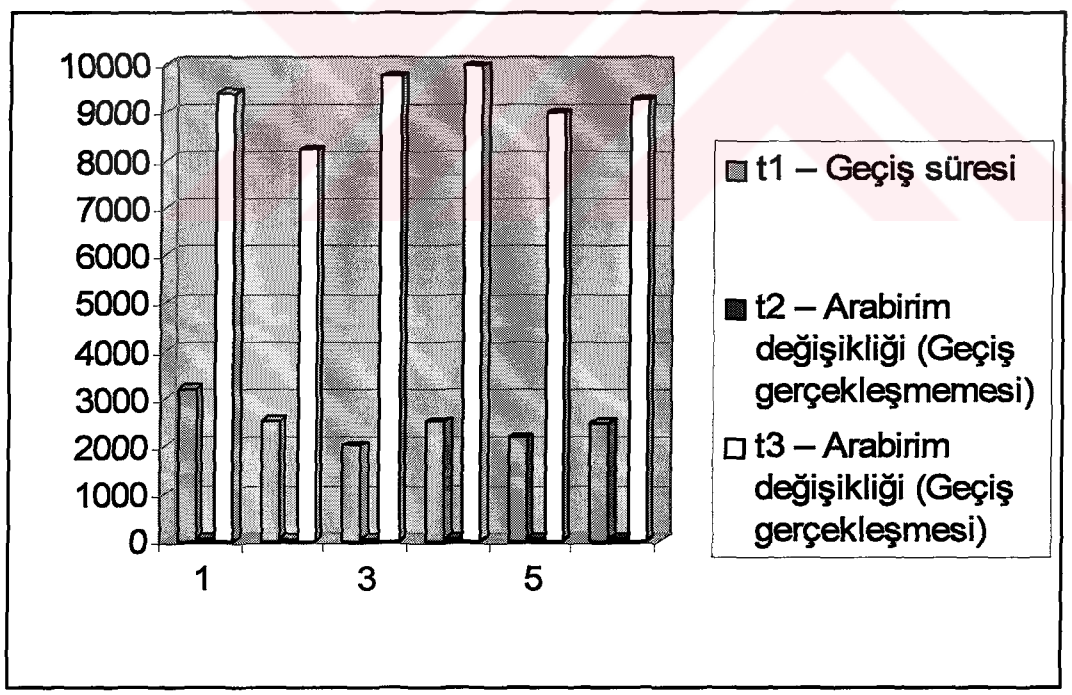
1. Kablosuz Ağ Arabirim
2. Çevirmeli Ağ Arabirim
3. Ethernet arabirim

olarak sistemde tanımlanmıştır. Kablosuz erişim noktası kapatılarak, kablosuz ağ arabiriminin bağlantısı kesilmiştir. Sistemin bu durumda aktif ethernet arabirime geçiş süresi ölçümlenmiştir.

İki uygulamanın sonuçları Şekil 9.1 ve Şekil 9.2'de grafiksel olarak gösterilmiştir.

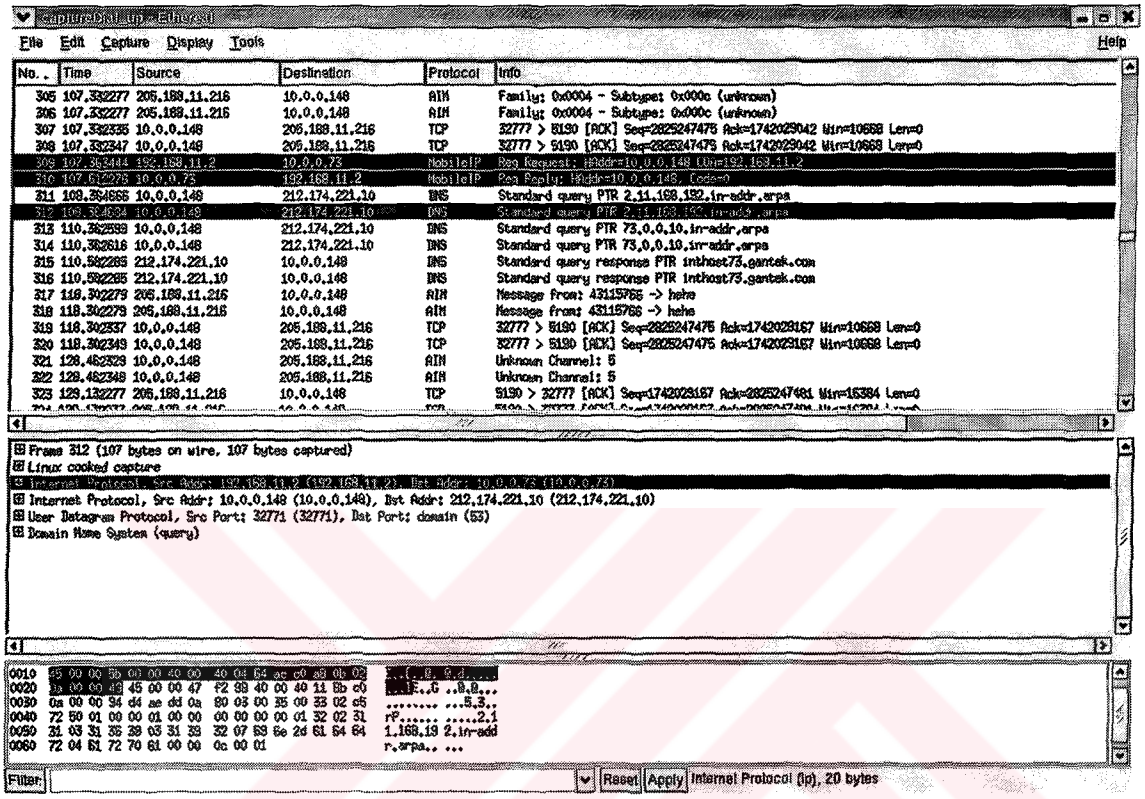


Şekil 9.1 Birinci Uygulamanın Test Sonuçları



Şekil 9.2 İkinci Uygulamanın Test Sonuçları

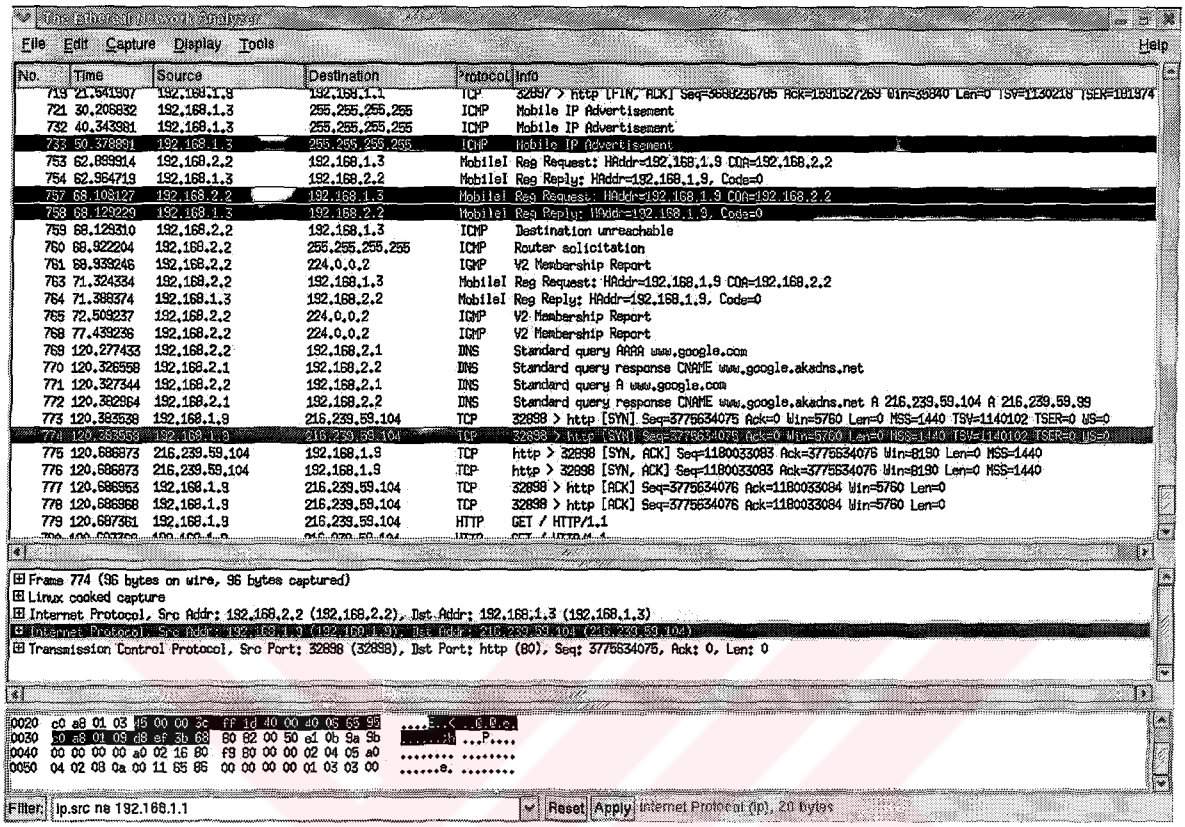
Uygulamada yerel acente ile mobil cihazın haberleşmesi bir paket dinleyici program olan "ethereal" yazılımı ile gerçekleştirilmiştir. Bu yazılımın ekran görüntüleri aşağıdaki gibidir.



Şekil 9.3 Modem ile bağlantıda kayıt işlemi

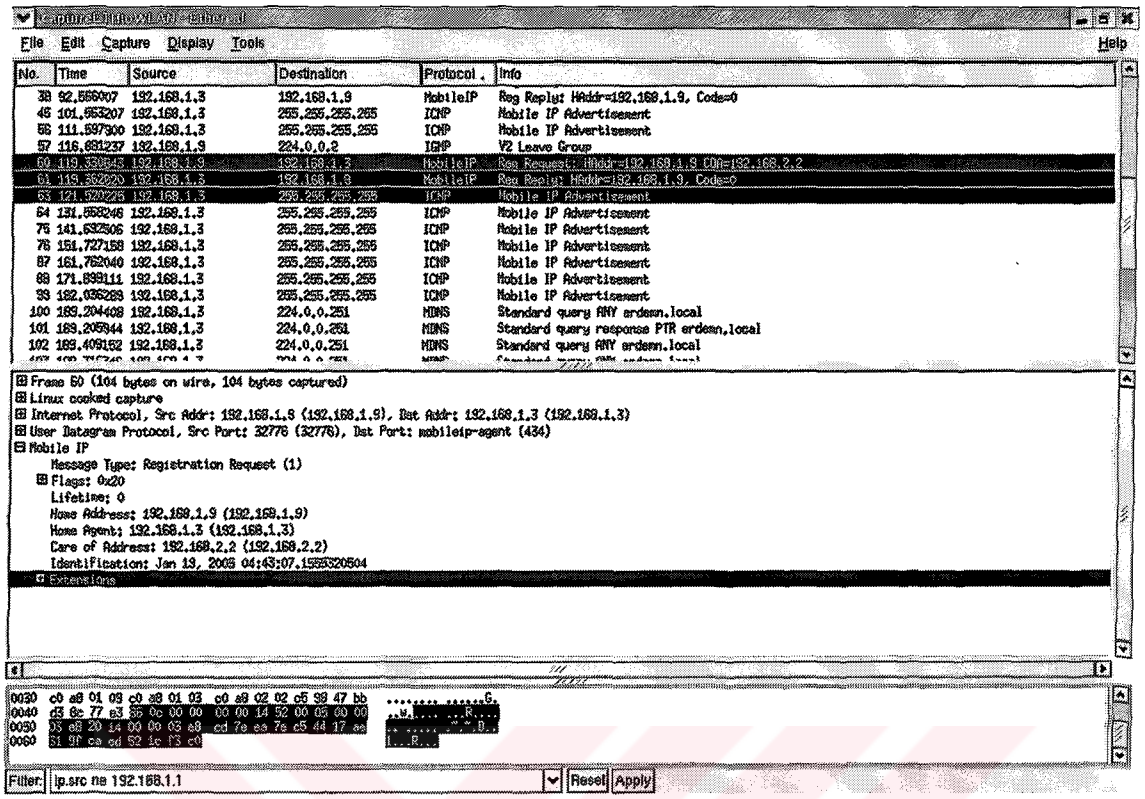
Şekil 9.3'de çevirmeli bağlantı sonrasında mobil cihazın HA bağlanarak yer kayıt paketini göndermesi gösterilmiştir. Ayrıca iletişim sırasındaki IP paketi içindeki IP paketleri de 312 numaralı satırda gösterilmiştir. 309 numaralı satırda mobil cihazın çevirmeli ağdan gönderdiği kayıt paketi gösterilmiştir. 310 numaralı satırda ise yerel acenteden gelen kayıt kabul paketi gösterilmiştir.

Kablosuz erişim kartı ile kablolu Ethernet kartı arasındaki geçiş işlemi ise Şekil 9.4'de gösterilmiştir.



Şekil 9.4 Arabirim geçişleri I

Şekil 9.4'de kablosuz erişim kartından bağlı olan mobil cihazın Ethernet kartının aktif olması ile gerçekleşen geçişin paketleri gösterilmiştir. Bu paketlerden ilk seçili olan paket HA tarafından gönderilen acente duyuru paketidir. Mobil cihaz bu durumda yerel ağında olduğundan mobilite özelliklerini kullanmamıştır. Ethernet kartı aktif olduğunda geçişin gerçekleştiği ve sonrasında HA'ya kayıt işlemi için kayıt paketi gönderimi (757-758 numaralı paketler) gerçekleştirildiği seçili paketlerden görülmektedir.



Şekil 9.5 Arabirim geçişleri II

Şekil 9.5'te Ethernet kartı ile HA'ya bağlı olan cihazın kablosuz erişim kartının aktif olması ile geçişin gerçekleştiği (60 ve 61 numaralı paketler) ve yerel ağına geçiş yapan mobil cihazın acente duyuru mesajları (63 numaralı paket) aldığı seçili mesajlardan görülmektedir.

10 SONUÇ

Mobil cihaza sahip kullanıcılar Ethernet, modem, kablosuz ağ erişim kartı gibi arabirimler yardımı ile Internet'e erişim sağlayabilmektedir. Farklı arabirimler arasında bağlantıların kopmadan geçiş işlemi, arabirimlerden alınan detaylı durum bilgisi ile kullanıcının belirlediği öncelik bilgisi ile birlikte kullanılarak gerçekleştirilmiştir. Yapılan çalışmada bu amaca yönelik bir sistem tasarlanmış ve Linux işletim sisteminde C programlama dili ile bu tasarım gerçeklenmiştir.

- Sistemde kullanıcıların cihazlara öncelik verebilmesi amacı ile bir arabirim öncelik bilgi dosyası oluşturularak sistemin kullanıcı ile etkileşi sağlanmıştır.
- Gerçekleştirilen uygulamada Mobil IP protokolünde mobil cihazın FA olmadan otomatik olarak doğrudan HA'ya bağlanması için geliştirme gerçekleştirilmiştir. Bu sayede kullanıcı etkileşimi olmadan mobil cihaz HA'ya bağlanmaya çalışacaktır.
- Aktif arabirimlerin detaylı durum bilgisi çekirdekten alınmıştır. Bunun için arabirim türüne bağlı olarak farklı bilgiler elde edilmiştir. Ethernet kartı için ethernet kablosunun takılı olup olmadığı bilgisi, kablosuz ağ erişim kartı için bağlı olduğu ağ erişim noktası bilgisi, modem için arabirimün aktif olduğu bilgisi sistemden alınmıştır.
- Kullanıcıların arabirimlere verdikleri öncelikler arabirim değişimlerinde kullanılmıştır. Kullanıcı tarafından verilen öncelik bilgisi ile arabirim detaylı durum bilgisi birleştirilerek geçiş yapılacak arabirim belirlenmektedir. Bu sayede karar mekanizması otomatik olarak çalışabilmekte ve kullanıcı isteklerine göre karar verebilmektedir.
- İki farklı sistem tasarımı yapılmış ve buna bağlı iki farklı yazılım geliştirilmiştir. Dynamics yazılımından bağımsız çalışan uygulama, Mobil IP protokol uygulamalarından bağımsız olarak geçiş işlem kararlarını gerçekleştirmekte ve uygulayabilmektedir. Dynamics yazılımında gerçekleştirilen uygulama ile sistemin geçiş performansı arttırılmıştır.
- Yapılan testler Dynamics yazılımında gerçekleştirilen uygulamanın daha performanslı olduğunu ortaya çıkarmıştır. Bu Dynamics yazılımı ile haberleşme için soket arabiriminin ortaya çıkarmış olduğu bir yüktür. Fakat arabirim

değişim bilgilerinin işlenmesi, Dynamics yazılımında daha yavaş kalmaktadır. Bunun nedeni kablosuz erişim noktaları arasındaki önceliklere göre geçişin gerçekleştirilmesini sağlayan "monitor" ek modülüdür. Bu modül arabirim değişim bilgilerinin iki defa işlenmesini sağlamaktadır.

- Arabirimün sonlanmasının algılanması ve aktif bir bağlantı bulma işlemi diğer işlemlere göre daha fazla zaman almaktadır. Bunun nedeni aktif arabirimün sonlanması sonucunda ilgili verilerin güncellenmesi ve istatiklerin yeniden elde edilmesi gerekliliğidir.

Tasarlanan iki sistemin çalışması ve yapılan testler sonucunda sistemde ileride yapılabilecek çalışmalarda aşağıdaki değişikliklerin uygulanabileceği sonucuna varılmıştır.

FA olan ağlarda, arabirim geçişlerinin kullanıcı tarafından belirlenecek öncelikler ve acente duyuruları ile birlikte gerçekleştirilmesi sağlanabilir.

Arabirimler ile ilgili olarak öncelik bilgisi ile birlikte arabirim bağlantı tipi tanımı yapılabilir. Bu bağlantı tipi arabirim detaylı durum bilgisi değerlendirilirken arabirim tipine uygun detaylı durum bilgilerinin daha hızlı getirilmesini sağlayabilir.

Kullanıcıya detaylı bir yönetim arabirimü ile arabirim öncelik grupları oluşturması sağlanabilir. Bu öncelik grupları dinamik olarak yazılıma yüklenerek kullanıcının seçtiği öncelik grubunun aktif olması sağlanabilir.

KAYNAKLAR

Brown, K. ve Singh., S., "M-UDP: UDP for Mobile Networks", ACM Computer Communication Review, 26(5):60-78,1996

Caceres, R. ve Padmanabhan, V.N., "Fast and Scalable Wireless Handoffs in Supoort of Mobile Internet Audio", ACM Journal on Mobile Networks and Applications, 3(4), Aralık 1998

C. Perkins, Ed., (1996), "IP Mobility Support", IETF Standartı, RFC 2002

C. Perkins, Ed. (2002), "IP Mobility Support for IPv4.", IETF Standartı, RFC 3220

Forsberg, Dan, (2000), "Communication Availability with Mobile IP Wireless LANs", Y.Lisans Tezi, Helsinki University of Technology

G. Baker, Mary , Zhao, Xinhua, Cheshire, Stuart, Stone, Jonathan (1996), "Supporting Mobility in MosquitoNet.", USENIX Teknik Konferansı, 1996

Matthew, Neil ve Stones, Richard (2004), Beginning Linux Programming, Wiley Publishing, Indianapolis

Montavont, N. ve Noel, T. (2002), "MIPv6 for Multiple Interfaces.", Internet Taslağı (Çalışma devam ediyor)

Montenegro, G. , (1998), "Reverse Tunneling for Mobile IP", IETF Standartı, RFC 2344

Perkins, C. E.ve Wang, K. Y., "Optimized smooth handoff in Mobile IP", Bilgisayar ve İletişimle ilgili dördüncü IEEE Sempozyumu, Sf. 340-346, 1999

P. McCann, T.H.J. Wang, A. Casati, C.Perkins ve P. Calhoun, "Transparent Hierarchical Mobility Agents (THEMA)", Internet taslağı, Mart 1999

Pollini, G.P., (1996), "Trends in Handover Design", IEEE Communications Magazine, 34(3):82-90

Ramjee, R., Porta, T. L., Thuel, S., Varadhan K. ve Salgarelli, L., "IP micro-mobility support using HAWAII", Internet taslağı, Haziran 1999

Salim, J., Khosravi, H., Kleen, A., Kuznetsov, A. (2003), "Linux Netlink as an IP Services Protocol", RFC 3549

Tan, C., Pink, S. ve Lye, K., "A Fast Handoff Scheme for Wireless Networks", 2nd ACM International Workshop on Wireless Mobile Multimedia (WoWMoM'99), Seattle, Washington, sf. 83-90, Ağustos 1999

Tekinay, S., Jabbari, B., "A measurement-based prioritization schema for handovers in mobile cellular networks", IEEE dergisi İletişimde Seçili alanlar, 10:1343:1350,1992

Wang, H. J., Katz, R. H., Giese, J., "Policy-Enabled Handoff Across Heterogeneous Wireless Networks", II. IEEE Mobile Computing Systems and Applications Workshop, Sf. 51-60, Şubat 1999

INTERNET KAYNAKLARI

[1] Yeğin, Alper E., (2001), "Mobil Internet Protokolü"

http://www.teknoturk.org/docking/yazilar/tt000077-yazi.htm.

[2] Linux Belgelendirme Çalışma Grubu, "Linux İşletim Sistemi",

http://www.belgeler.org/lis/archive-tlkg-lis-1g.html

[3] http://www.farukcubukcu.com

[4] "Netlink Routing Sockets Tutorial", http://www.infsec.net/

[5] Manuel Rodríguez, "An implementation of Mobile IP under Linux",

http://www.hpl.hp.com/personal/Jean_Tourrilhes/MobileIP/

[6] The Portland State University Secure Mobile Networking Project, Linux Mobile-IP partial release,

http://www.cs.pdx.edu/research/SMN/

[7] MosquitoNet Mobile Computing Group, Stanford University,

http://mosquitonet.stanford.edu/mip/

[8] Helsinki University of Technology, Dynamics - HUT Mobile IP,

http://dynamics.sourceforge.net/

EKLER

EK 1 HA ÜZERİNDE GERÇEKLEŞTİRİLEN KONFİGÜRASYON AYARLARI

"/usr/local/etc/dynhad.conf" dosyası içinde HA ağ ayarları gerçekleştirilmiştir. Bu dosyanın içeriğinde kullanılan gerekli parametreler ve anlamları aşağıda anlatılmıştır. "#" ile başlayan satırlar dosyada açıklama satırlardır.

"dynhad.conf"

```
# Mobile IP servisleri için kullanılan arabirim listesi.
# Dikkat edilmesi gereken nokta kayıt mesajı alımı ve
# gönderimi gerçekleştirilecek olan tüm arabirimlerin
# bu bölümde tanımlanması gerekmektedir.
# interface: Arabirimün adı, eth0 gibi
# ha_disc:
#   0 = HA'ın dinamik bulunmasına izin verme
#   1 = HA'ın genel yayın (broadcast) mesajları ile dinamik
#       olarak bulunmasına izin ver
# agentadv:
#   0 = Acente duyuru mesajlarını istek olmadan gönderme
#   1 = Acente duyuru mesajlarını düzenli olarak gönder
#  -1 = İstek olsa dahi duyuru mesajı gönderme
# interval: İki duyuru mesajı arasında saniye cinsinden bekleme değeri
#       (eğer arabirime izin verilmiş ise)
# force_IP_addr: Bu arabirim için zorunlu kullanılacak lokal adres
#                (Birden çok sanal adresin seçimi için kullanılabilir);
#                eğer girilmez ise arabirimün ana adresi kullanılır
INTERFACES_BEGIN
# interface  ha_disc  agentadv  interval  force_IP_addr
eth0         1        1        10
#eth1        1        1        20        192.168.240.2
INTERFACES_END
```

```
# Kayıt isteklerinin dinleneceği UDP portu
# Standart değer 434'tür
UDPPort 434

# HA'ya aynı anda bağlı olabilecek maksimum mobil cihaz sayısı
# Standart değer 20'dir.
MaxBindings 20

# HA tarafından tavsiye edilen tünel ömrü (tunnel lifetime)
# Standart değer 500'dür.
HADefaultTunnelLifetime 600

# Kayıt hata dönüş değeri.
# Çok fazla kabul edilmemiş kayıt istek mesajlarının
# sisteme yük getirmesini önlemek amacı ile kayıt
# hata cevap aralığı kullanılacaktır.
# Standart değer 1 saniyedir.
RegErrorReplyInterval 1

# Üçgen tünelleme, MN'ye gelen paketlerin HA üzerinden gelmesi
# fakat MN'den giden paketlerin doğrudan yönlendirilmesidir. (örnek olarak FA normal
# IP  yönlendirmesi yapmaktadır)
# EnableTriangleTunneling < TRUE | FALSE >
EnableTriangleTunneling TRUE

# Zıt tünelleme, MN'den çıkan ve MN'a gelen tüm paketlerin
# HA üzerinden geçerek çift yönlü tünellenmesidir
# EnableReverseTunneling < TRUE | FALSE >
EnableReverseTunneling TRUE

# HA, tüm mobil cihazların ne tür güvenlik parametrelerini kullandığını bilmelidir.
# Bu nedenle bir tabloda çoktan çoka ilişki ile SPI numaraları
```

```
# (veya SPI numara aralıkları) ağ adresi ve ağ maskesi ile
# IP adresine (veya IP adres aralığına)  ilişkilendirilmiştir
# < SPI | SPI-range       IP | network/netmask  >
AUTHORIZEDLIST_BEGIN
# SPI          IP
1000    10.0.0.148
AUTHORIZEDLIST_END

# HA, tüm mobil cihazlar için bir güvenlik eşleştirmesi yapmalıdır.
# Bu eşleştirme aşağıdaki bilgileri içerir.
#
# SPI (Security Parameter Index): Diğer alanlar için indeks alanı.
#
# Doğrulama Algoritması (Authentication Algorithm):
#    1: MD5/prefix+suffix [RFC 2002]
#    4: HMAC-MD5 [RFC 2104]
#    5: SHA-1 [FIPS 180-1]
#    6: HMAC-SHA1 [RFC 2104]
# Dikkat edilmesi gereken nokta MD5/prefix+suffix algoritmasının
# zayıf yönleri mevcuttur ve HMAC-MD5 kullanımı tavsiye edilir.
#
# Tekrarlama Koruma Metodu (Replay Protection Method):
#    0: Yok
#    1: timestamp
#    2: nonce
#
# Zaman etiketi toleransı mobil cihaz zaman etiketinin HA'nın zamanından
# ne kadar farklı olabileceğini göstermektedir. Standart olarak 7 saniye
# önerilmektedir. Bu değer tekrarlama koruma metodu olarak zaman
# etiketi kullanıldığında kontrol edilmektedir.
```

```
# Saniye olarak kayıt bağlantı süresi
# 65535 (veya daha fazla) saniye sınırsız olarak kabul edilmektedir.
#
# Ortak Şifre :MN ve HA tarafından bilinen ortak veri(shared secret)
#
# SPI alanı diğer güvenlik parametreleri için anathar görevini üstlenmiştir.
#
SECURITY_BEGIN
#      auth.   replay  timestamp      max           shared
# SPI  alg.    meth.   tolerance      lifetime      secret
1000   4        1       120             600            "test"
SECURITY_END
END
```

EK 2 MOBİL CİHAZDA GERÇEKLEŞTİRİLEN KONFİGÜRASYON AYARLARI

"/usr/local/etc/dynmn.conf" dosyası üzerinde mobil cihaz Mobil IP protokol konfigürasyonu gerçekleştirilmiştir. Bu dosyanın içeriğinde kullanılan gerekli parametreler ve anlamları aşağıda anlatılmıştır. "#" ile başlayan satırlar dosyada açıklama satırlardır.

dynmnd.conf

```
# Mobil cihazın yerel ağındaki IP adresi
# Eğer AAA kullanımı mevcut ise bu parametre 0.0.0.0 olarak parametre edilebilir.
# Bu durumda mobil cihaz AAA yapısına bir yerel adres isteğinde bulunacaktır.
# AAA yapısında NAI parametresinin girilmesi gerekmektedir.
MNHomeIPAddress 10.0.0.148

# Mobil cihazın tanımlı olduğu HA'nın IP adresi.
# Eğer HA'nın birden fazla arabirimü mevcut ise burada
# dış ağlardan erişilebilir olan adresi kullanılmalıdır.
HAIPAddress 10.0.0.73

# EnableFADecapsulation < TRUE | FALSE >.
# TRUE durumunda içiçe IP  (IP-within-IP)  paketleri FA tarafından
# işlenmektedir.
# FALSE durumunda mobil cihaz bu paketleri alıp işlemektedir.
# FA'nın paketleri işleme durumunda mobil cihaz yabancı ağda
# dahi olsa yerel adresini kullanır. Mobil cihazın paketleri işlemesi
# durumunda mobil cihaz yabancı ağ üzerinde geçici bir adres
# (co-located care-of address) alması gerekmektedir
EnableFADecapsulation FALSE
```

```
# Yerel ağın ağ adresi (Network address of home network)
# (Formatı: a.b.c.d/prefix_length)
# Örnek: 192.168.242.0/24
HomeNetPrefix 10.0.0.0/24
##############################################################################
########
# SPI (Security Parameter Index) tüm mobil cihazlar tarafından
# konfigüre edilmiştir.
# HA tarafında güvenlik bağlantılarını indeks ile gösterme amacı ile kullanılır
SPI 1000

# Ortak Şifre HA ile mobil cihaz tarafında ortak kullanılan şifredir.
# SharedSecret < shared secret >
# SharedSecret 016A352B2F235E
SharedSecret "test"

# Tünelleme Modu < 1 | 2 | 3 | 4 >
# Mobil cihaz ve iletişim kurulan bilgisayar arasındaki
# paketler farklı yollardan yönlendirilebilirler.
# Bu parametre kullanılacak yönlendirme modunu belirler
# Olası değerler:
# 1 = otomatik, zıt tünellemeyi tercih et (tünelleme her iki yönde MN->HA, HA->MN)
# 2 = otomatik, üçgen tünellemeyi tercih et (tünelleme sadece CN->MN yönünde)
# 3 = sadece zıt tünellemeyi kabul et
# 4 = sadece üçgen tünellemeyi kabul et
TunnelingMode 1

# Mobil cihaz zıt tünellemeyi kullandığı ve geçici adres aldığı durumda
# ana yölendirme tünele gerçekleşir. Bunun anlamı bulunan alt ağa gönderilen
# paketlerin dışındaki diğer tüm noktalar gönderilen paketler HA'ya gönderilir.
# Eğer geçici adres genel (public) bir adres ise sabit IP adresi istemeyen oturumlar
# için geçici adres kullanılabilir (internette sürf yapmak gibi).
```

Bu parametre geçici adres ile ilgili yapılacak yönlendirmeyi belirler.
Olası değerler:
0 = ana yönlendirmeyi tünel sistemine olarak belirle
1 = sadece yerel ağa olan paketleri tünele yönlendir
2 = yönlendirme kayıtlarını değişiklik gerçekleştirme
(kullanıcı dışardan yönlendirme tablosunu değiştirir)
MNDecapsRouteHandling 0

EK 3 "DYNMN_TOOL" YAZILIMININ KOMUT PARAMETRELERİ

"dynmn_tool" yazılımı mobil cihazın kontrol edilmesi ve izlenebilmesi amacı ile geliştirilmiş bir yazılımdır. "dynmn_tool" yazılarak interaktif modda başlatılabilir veya komut satırından verilen parametreler ile istenilen komutu yerine getirmesi sağlanabilir. Bu komutlar :

confirm : Tünellemenin aktif olduğu ve HA'ya kayıt mesajlarının gönderildiğini teyit eder.

Careof : Mobil cihazın geçici adresini gösterir.

"tunnel" ve "connect" : Aynı amaçla kullanılırlar. Mobil cihaz bağlı durumda değil iken bir FA'ya kayıt işlemini başlatır. "tunnel HA" komutu HA'ya doğrudan bağlantı kurulmasını sağlar.

disconnect : Mobil cihazı tünelleme sisteminden çıkartır ve "Bağlı Değil" durumuna taşır.

status : Mobil cihazın durum bilgisini döner.

update *interface* : Yer güncellemesi gerçekleştirir ve parametre olarak verilen arabirimün ana IP adresinin kullanılmasını sağlar.

update *ip_addr* : Yer güncellemesi gerçekleştirir ve parametre olarak verilen adresi ağ adresi olarak kullanır.

list : En son gelen yabancı acente duyurularını listeler.

EK 4 UYGULAMANIN FONKSİYON PROTOTİPLERİ VE VERİ YAPILARI

Fonksiyon Prototipleri

interfaceListener Fonksiyonu

Uygulamanın ana fonksiyonudur. Netlink soketi açılmasını sağlar ve çekirdekten gelen arabirim değişim bilgilerini bu soket üzerinden kontrol eder. Değişim tespit edildiğinde en yüksek öncelikli arabirimün aktif olup olmadığını *interfaceStatus* fonksiyonu ile alıp geçiş için *makeHandover* fonksiyonunu çağırır. Fonksiyonun prototipi aşağıdaki gibidir.

void interfaceListener(void)

netlinkOpen Fonksiyonu

Çekirdek ile haberleşmeyi sağlayan netlink soketini açar. Bu netlink soketi yazılım tarafından arabirim bilgilerini almak ve sonrasında da değişim bilgilerini dinlemek amacı ile kullanılır. Fonksiyonun prototipi aşağıdaki gibidir.

int netlinkOpen(void)

interfaceStatus Fonksiyonu

Parametre olarak gelen arabirimün cihazın aktif bilgisini döner. Bunun için *interface_detect_ETHTOOL_GLINK, interface_detect__WLAN, interface_detect_ PPP* fonksiyonlarını çağırır. Fonksiyonun prototipi aşağıdaki gibidir.

*interface_status_type interfaceStatus(int fd, char *iface)*

interface_detect__WLAN Fonksiyonu

Kablosuz ağ erişim kartı için bir kablosuz erişim noktasına bağlı olup olmadığını kontrol eder. Fonksiyonun prototipi aşağıdaki gibidir.

*interface_status_type interface_detect__WLAN(int fd, char *iface)*

interface_detect_ ETHTOOL_GLINK Fonksiyonu

Kablolu ağ erişim kartının aktif olduğunu ve kablosunun takılı olduğunu kontrol eder. Fonksiyonun prototipi aşağıdaki gibidir.

*interface_status_type interface_detect_ ETHTOOL_GLINK(int fd, char *iface)*

interface_detect_ PPP Fonksiyonu

Çevirmeli bağlantı durumunun aktif olduğunu kontrol eder. Fonksiyonun prototipi aşağıdaki gibidir.

*interface_status_type interface_detect_ PPP(int fd, char *iface)*

makeHandover Fonksiyonu

Aktif arabirim ile değişim yapılmak istenen arabirimü karşılaştırır. Eğer iki arabirim aynı ise geçiş gerçekleşmez. Arabirimler farklı ise "dynmn_tool" yazılımı aracılığı ile istenilen cihaza geçiş için Dynamics yazılımına komut gönderir. Fonksiyonun prototipi aşağıdaki gibidir.

*int makeHandover(char *iface)*

destroyProcess Fonksiyonu

Çekirdekten gelen arabirim değişim bilgilerini dinlemek için açılan Netlink soketi kapatır. veri yapıları için tahsis edilmiş olan bellek alanını işletim sistemine iade eder. Fonksiyonun prototipi aşağıdaki gibidir.

void destroyProcess(int fd)

Veri Yapıları

device_priorities Veri Yapısı

Arabirim öncelik bilgilerinin dosyadan alınıp bellekte saklandığı veri yapısıdır. Veri yapısı ve içerdiği alanların anlamları aşağıdaki gibidir.

```
struct device_priorities {
        unsigned char hw[ETH_ALEN];      // Arabirim adı
        int priority;                    // Öncelik Değeri
        struct device_priorities *next;  // Sonraki arabirimün işaretçisi
};
```

iface_info Veri Yapısı

Çekirdekten gelen arabirim değişiklik bilgisinin saklandığı veri yapısıdır. Veri yapısı ve içerdiği alanların anlamları aşağıdaki gibidir.

```
struct iface_info{
        char    name[IF_NAMESIZE];
        u_int   index;
        u_int   flags;
        struct iface_info       *next;
};
```

interface_status_type Veri Tipi

Arabirim durum bilgisi ile ilgili veri tipidir. Prototipi aşağıdaki gibidir.

typedef enum { IFSTATUS_UP, IFSTATUS_DOWN, IFSTATUS_ERR } interface_status_type;

ÖZGEÇMİŞ

Doğum Tarih	19.09.1979	
Doğum Yeri	Hatay	
Lise	1991- 1997	Hatay Osman Ötken Anadolu Lisesi
Lisans	1997-2001	Ege Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü

Çalıştığı kurum(lar)

2002-2002	Erenet Bilgisayar A.Ş.
2002-Devam ediyor	Gantek Teknoloji