T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ

MÜHENDİSLİK ve FEN BİLİMLERİ ENSTİTÜSÜ

MOBİL CİHAZLARDA FARKLI AĞ BAĞLANTILARINI KULLANARAK TCP PERFORMANSININ ARTTIRILMASI

NECATİ ERSEN ŞİŞECİ

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

GEBZE

2013

T.C.
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK ve FEN BİLİMLERİ ENSTİTÜSÜ

MOBİL CİHAZLARDA FARKLI AĞ BAĞLANTILARINI KULLANARAK TCP PERFORMANSININ ARTTIRILMASI

NECATİ ERSEN ŞİŞECİ
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
DOÇ. DR. HACI ALİ MANTAR

GEBZE
2013

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ	YÜKSEK LİSANS JÜRİ ONAY FORMU

GYTE Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..17..06..2013.. tarih ve ..2013/..22.. sayılı kararıyla oluşturulan jüri tarafından 27/06/2013 tarihinde tez savunma sınavı yapılan Necati Ersen ŞİŞECİ'nin tez çalışması Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE (TEZ DANIŞMANI)	:	Doç. Dr. Hacı Ali MANTAR	
ÜYE	:	Yrd. Doç. Dr. Hasari ÇELEBİ	
ÜYE	:	Yrd. Doç. Dr. Serdar Süer ERDEM	

ONAY

GYTE Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... tarih ve ...../..... sayılı kararı.

İMZA/MÜHÜR

ÖZET

Mobil cihazların farklı ağalara bağlanma yeteneklerinin artması, aynı anda birden fazla ağ arayüzünü kullanabilme yeteneğine ihtiyaç doğurmuştur. Bu tez kapsamında birden fazla ağ arayüzünün aynı anda kullanılabilmesi için bir yöntem önerilmiştir.

Önerilen yöntem, mobil cihazlarda bağlantıların izlenmesi ve çeşitli parametreler aracılığı ile daha hızlı sonuç vereceği düşünülen ağ bağlantısının anlık olarak kullanılmasına dayanmaktadır. Bu yöntemle mobil cihazlarda TCP performansının arttırılması hedeflenmiştir.

Önerilen yöntemde sadece mobil cihazların TCP/IP yığınında değişiklik yapılması yeterli olmaktadır. Hücresel ağ bağlantı cihazları, kablosuz modemler, anahtarlama ve yönlendirme cihazları, güvenlik duvarları, saldırı tespit sistemleri gibi herhangi bir sistemde değişiklik yapılması gerekmemektedir. Kullanıcı uygulamalarında da sistem uygulamalarında da herhangi bir değişiklik yapılması gerekmemektedir. Yapılması gereken tek değişiklik mobil cihazların TCP/IP yığınında ağ katmanındadır.

Önerilen yöntem, FreeBSD üzerinde bir çekirdek modülü olarak gerçeklenmiş ve çeşitli performans testleri yapılmıştır.

Sonuç olarak, önerilen yöntemin zaman zaman sadece tek ağ bağlantısının kullanılması durumuna göre avantajları olduğu tespit edilmiştir.

Anahtar Kelimeler: Multihome; Transmission Control Protocol (TCP) Performansı; Mobil Cihazlar; Kablosuz Ağ; Freebsd

SUMMARY

Increasing number of network interfaces of mobile devices created a demand on capability of using simultaneous use of multiple network interfaces. In this thesis, we have proposed a method to use multiple network interfaces simultaneously.

The proposed method is based on instant use of network which is considered to give better performance by tracking connections and using several parameters on mobile devices. The objective is to increase the TCP performance of mobile devices.

The method proposes that only updating TCP/IP stack of mobile devices is enough. There is no need to make changes on devices like cellular network devices, wireless modems, routers and network switches, firewalls, intrusion detection and prevention systems. There is a need to update or change neither in user applications nor in system applications. The only change needed is on the network layer of TCP/IP stack in mobile devices.

The proposed method is implemented as a kernel module on a FreeBSD box and several performance tests are applied. As a result, tests results showed that the proposed method is better than using single network interface on some conditions.

Keywords: Multihome; Transmission Control Protocol (TCP) Performance; Mobile Devices; Wireless; FreeBSD

TEŞEKKÜR

Tez çalışmamda bana sürekli yol gösteren ve yardımlarını esirgemeyen saygıdeğer hocam Sayın Doç. Dr. Hacı Ali MANTAR Bey'e, tez çalışmam sırasında bana gösterdikleri anlayış ve sabırdan ötürü tüm çalışma arkadaşlarıma ve benden hiçbir zaman desteğini esirgemeyen eşim Sayın Songül ŞİŞECİ Hanım'a ve aileme teşekkürlerimi sunarım.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler ve Kısaltmalar		**Açıklamalar**
3G	:	3ncü nesil
ARP	:	Address Resolution Protocol (Adres Çözümleme Protokolü)
DHCP	:	Dynamic Host Configuration Protokol (Dinamik Host Yapılandırma Protokolü)
DNS	:	Domain Name System (Alan Adı Sistemi)
DSL	:	Digital Subscriber Line (Sayısal Abone Hattı)
FIB	:	Forwarding Information Base (Yönlendirme tablosu)
FTP	:	File Transfer Protocol (Dosya Aktarım Protokolü)
GPRS	:	General Packet Radio Service (Genel Paket Radyo Servisi)
HSPA	:	High Speed Packet Access (Yüksek Hızlı Paket Erişimi)
IANA	:	Internet Assigned Numbers Authority
ICMP	:	Internet control messaging protocol (Internet Kontrol Mesaj İletişim Kuralı)
IEEE	:	The Institute of Electrical and Electronics Engineers (Elektrik ve Elektronik Mühendisleri Enstitüsü)
IP	:	Internet Protocol (Internet Protokolü)
ISS	:	Internet Servis Sağlayıcı (Internet Service Provider)
OSI	:	Open Systems Interconnection
POP3	:	Post Office Protokol 3 (Postane Protokolü 3)
RFC	:	Request For Comments (Yorumlar için talep)
RTT	:	Round Trip Time (Gidiş geliş süresi)
SCTP	:	Stream Control Transmission Protocol (Akış Kontrol İletişim Protokolü)
TCP	:	Transmission Control Protocol (İletişim Kontrol Protokolü)
TTL	:	Time-to-Live (Yaşam Süresi)
UDP	:	User Datagram Protocol (Kullanıcı Veribloğu İletişim Kuralları)
VLAN	:	Virtual Lan (Sanal Ağ)

ŞEKİLLER DİZİNİ

1. GİRİŞ

Teknolojinin ilerlemesi ve mobil cihazların daha akıllı hale gelmesi ile her ortamda Internet'e bağlanma imkânı doğmuştur. 3G, 4G ve kablosuz internet gibi teknolojiler ile mobil cihazlar kesintisiz olarak Internet'e ulaşılabilmektedir. Taşınabilir bilgisayarlarda kablolu, kablosuz (IEEE 802.11) ve hücresel ağ bağlantıları (HSPA+, HSPA, 3G vb) kullanılabilmektedir [Web 1]. Mobil cihazlarda ise kablosuz ve hücresel ağ bağlantıları kullanılabilmektedir. HSPA, GPRS gibi teknolojilerin, IEEE 802.11 tabanlı kablosuz ağlara göre kapsama alanı çok geniş olmasına rağmen bant genişlikleri düşüktür. Bu bağlantı teknolojileri ayrı ayrı kullanılabildikleri gibi aynı anda da kullanılabilmektedir. Kullanıcı için Internet erişiminde tek değişen şey fiziksel katmandır.

Aynı anda birden fazla ağ bağlantısı yapılabilen cihazlar aracılığı ile farklı Internet servis sağlayıcılar üzerinden Internet'e erişmek mümkündür. Bir cihaz, ağ bağlantısı yaptığı tüm arayüzleri için bağlandığı Internet servis sağlayıcısının verdiği ağ erişim bilgilerini (IP adresi, alt ağ maskesi, ön tanımlı ağ geçidi, DNS sunucusu vb.) alacaktır. Örneğin iş yeri, ev, kafe gibi ortamlarda bulunan kablosuz ağ bağlantıları kullanıldığında ortamda bulunan DHCP sunucusu ağ bağlantısı yapan cihazlara ağ erişim bilgilerini yollayacaktır. Aynı şekilde hücresel ağ bağlantısı kullanıldığı zaman ise, şebeke operatörü tarafından ağ erişim bilgileri cihaza yollanacaktır. Atanan IP adresleri, RFC1918'ye göre dağıtılabileceği gibi gerçek IP adresleri olarak da dağıtılabilmektedir [Web 2]. RFC1918'e göre dağıtılan IP adresleri, Internet erişimi için RFC3022'de anlatıldığı gibi basit ağ adres dönüşümünden geçirilmelidir [Web 3]. Bu işlem genellikle ağ geçidi görevi gören DSL modemler, kablosuz erişim noktaları veya ağ geçidi olan cihazlar tarafından yapılmaktadır.

Aynı cihaz üzerinde birden fazla ağ bağlantısına ihtiyaç duyulmasının en önemli sebeplerinden birisi bantn genişliğinin arttırılmasıdır. Bunun yanı sıra, bağlantı yedekliği, hat güvenliği, erişilebilirlik gibi sebepler de birden fazla ağ arayüzü kullanım ihtiyacını doğurmuştur. Bazı kurumlar, ana yönlendirici cihazlarında birden fazla Internet Servis Sağlayıcı (ISS)'den hizmet alarak Internet'e bağlanmaktadırlar. Bağlantı hatlarından herhangi birisinde problem olması durumunda, kurumun Internet'i etkilenmeyeceği için hem kurum Internet

kullanıcıları hem de kuruma Internet üzerinden erişen kullanıcılar bu kesintiden etkilenmeyecektir. Aynı şekilde mobil cihazlarda birden fazla ağ arayüzü kullanımında da benzer sebepler ek olarak ek olarak hücresel ağ bağlantı ücretlerinin kablosuz ağ bağlantılarına göre daha pahalı olması ve farklı ağ tiplerinin kapsama alanlarının farklı olması da eklenebilir.

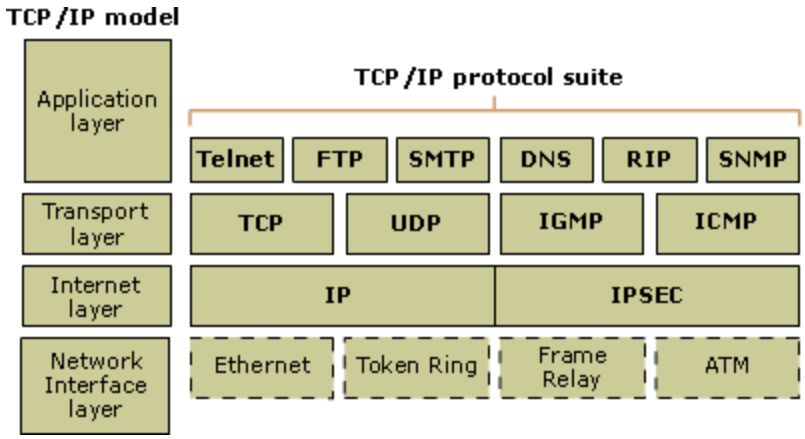
Birden fazla farklı ağ bağlantısının aynı anda kullanılması ile ilgili yaptığımız bu çalışmadaki amaçlarımızdan birisi yapılacak değişikliklerin sadece kullanıcı tarafında yapılmasıdır. Kullanıcı cihazlarında güncelleştirme yapmak, ağ geçidi olan ve Internet üzerinde bulunan aktif cihazların TCP/IP yığınında değişiklik yapmaktan çok daha kolaydır. Diğer bir amaç ise kullanıcı tarafından belirlenecek politika ile ağ arayüzlerine ağırlık verilebilmesidir. Örneğin 802.11 tabanlı kablosuz ağdaki kayıp oranı arttığında, sistem otomatik olarak hücresel ağ bağlantısını daha fazla kullandırmalıdır.

2. TEMEL BİLGİLER VE KULLANILAN TEKNOLOJİLER

Bu bölümde TCP/IP, mobil cihazların Internet erişiminde kullandığı haberleşme teknolojilerinden ve tezde önerilen yöntemin gerçeklenmesinde kullanılan yazılımlardan bahsedilecektir.

2.1. TCP/IP Protokol Ailesi

TCP/IP protokol ailesi, Internet'in çalışmasını sağlayan protokollerin bütünüdür. TCP (Transmission Control Protocol) ve IP (Internet Protocol)'ün kısaltmalarıdır. TCP/IP'nin katmanlı bir yapısı bulunmaktadır. TCP/IP kullanarak başka bir noktaya gönderilecek olan paketler sırayla katmanlardan geçerek ilgili katman başlık bilgilerini eklenir ve en alt katman olan fiziksel katman aracılığı ile hedef noktaya gönderilir.



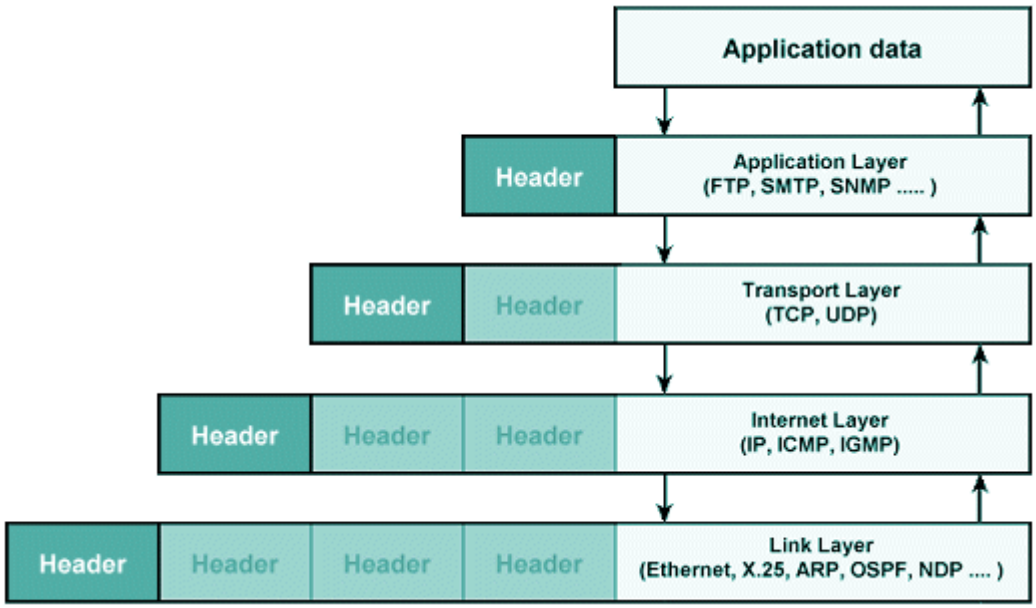
Şekil 2.1: TCP/IP Modeli.

Şekil 2.1'de görüldüğü gibi TCP/IP modelinde 4 katman bulunmaktadır [Web 4]. Bu katmanlar aşağıda listelenmiştir.

- Uygulama katmanı
- Taşıma katmanı
- Internet katmanı
- Fiziksel katman

TCP/IP referans modeline ek olarak 7 katmanlı OSI (Open Systems Interconnection) modeli de mevcuttur. Konsept bir modeldir. Bu modelde bulunan katmanlar aşağıda listelenmiştir.

- Uygulama katmanı
- Sunum katmanı
- Oturum katmanı
- Taşıma katmanı
- Ağ katmanı
- Veri bağı katmanı
- Fiziksel katman



Şekil 2.2: Katmanlarda eklenen başlıklar.

Şekil 2.2'de her katmanda eklenen başlıklar görülmektedir [Web 5]. Alt katmanlarda veri anlamına gelen bilgiler, üst katmanlarda başlık anlamına gelebilmektedir.

Gönderilen bir paketin karşı uca ulaşması ve pakete cevabın gelmesi arasında geçen süreye gidiş geliş zamanı (Round Trip Time – RTT) denir. RTT değerinin büyük olması, uzaklık, bant genişliği kullanımı, geçilen yönlendirici sayısı gibi parametrelere bağlı olarak anlık olarak değişebilir.

2.1.1. Fiziksel Katman

Bu katman aracılığı ile kullanılan bilgisayar veya cihazdan gönderilecek olan paketler elektrik sinyallerine dönüştürülerek, fiziksel ağ iletişim ortamına bırakılır. Koaksiyel kablo, UTP kablo, fiber optik kablo, telefon kablosu, RS-232 gibi kablolama yöntemleri dışında, 3G, Wi-fi, Wi-Max gibi kablosuz iletişim ortamları da bu katman aracılığı ile sisteme bağlanır.

2.1.2. Internet Katmanı

Bu katmanda tüm paketlere yeni başlık bilgisi eklenir. Gönderilecek paketler, eklenen başlık bilgisi içerisinde alıcı ve gönderi adresleri gibi bilgiler pakete eklenerek fiziksel katmana gönderilir. Alınan paketlerde ise, başlık bilgisinin doğruluğu, alıcı adresi gibi bilgiler kontrol edilir. Uygunsa paket bir üst katman olan taşıma katmanına iletilir. Uygun olmayan paketler düşürülür. IP, ICMP (Internet control messaging protocol) ve ARP (Address resolution protokol)'de bu katmanda çalışmaktadır. Bu katmanda en çok kullanılan protokoller aşağıda anlatılmıştır.

2.1.2.1. Internet Protokolü v4 (IPv4)

Internet protokolü aracılığı ile farklı cihazlar birbirleri ile görüşebilmektedir. Dünya üzerindeki her bir cihazın farklı bir IP adresi bulunmaktadır. IP adresine ait farklı iki bilgisayar ve cihaz farklı ağlarda olsalar bile yönlendirici cihazlar sayesinde birbirleri ile haberleşebilmektedir. Günümüzde IPv4 ve IPv6 olmak üzere 2 farklı sürümü kullanılmaktadır. IPv4 protokolünde IP adresleri 4 adet 8 bitlik sayılarla gösterilir. Gösterimde sayılar onluk düzende gösterilir. *172.16.0.10, 192.168.0.255* ve *10.0.0.0* örnek IP adresleri olarak gösterilebilir.

Bazı IP adresileri IANA (Internet Assigned Numbers Authority) tarafından farklı amaçlarla kullanım için ayrılmıştır. Internet'e direk bağlı olan cihazlar bu IP adreslerini kullanamazlar [Web 6]. Bu adreslerin bir kısmı aşağıda listelenmiştir.

- 0.0.0.0 – 0.255.255.255 Lokal kimlik
- 10.0.0.0 – 10.255.255.255 Özel kullanım
- 127.0.0.0 – 127.255.255.255 Loopback
- 169.254.0.0 – 169.254.255.255 – APIPA (Automatic private IP addressing)

- 172.16.0.0 – 172.31.255.255 Özel kullanım
- 192.168.0.0 – 192.168.255.255 Özel kullanım
- 240.0.0.0 – 255.255.255.255 Gelecekte kullanım

IP adreslerinin gösterimlerinde farklı yöntemler bulunmaktadır. Bu yöntemler yazım ve anlaşılma kolaylığı sağlamaktadır. Örneğin 192.168.10.0 – 192.168.10.255 arasındaki ağı göstermek için aşağıdaki gösterimler kullanılabilir.

- 192.168.10.0 – 192.168.10.255
- 192.168.10.0/255.255.255.0
- 192.168.10.0/24

IP aralıkları gösterilirken alt ağ maskesi kullanılır. Alt ağ maskesi, 32 bitlik IP adresinin ilk kaç bitinin sabit, sonraki kaç bitinin değişken değer alabileceğini göstermektedir. Örneğin 256 adet IP adres aralığını, ilk 24 bitin sabit (1), sonraki 8 bitin değişken (0 veya 1) olması anlamına gelmektedir. Bu durumda alt ağ maskesi *255.255.255.0* veya */24* olarak gösterilir. İkinci bir örnek olarak 16 IP adresli bir ağ belirtilmek istenirse, ilk 28 bitin sabit, sonraki 4 bitin değişebileceğini gösteren *255.255.255.240* veya */28* gösterimi kullanılır.

IP adresleri 5 farklı sınıf olarak olarak kullanılmaktadır. Bu sınıflar aşağıda gösterilmiştir.

- A sınıfı: 0.0.0.0 – 127.0.0.0 (126 ağ, ağ maskesi /8)
- B sınıfı: 128.0.0.0 – 191.255.0.0 (16.384 ağ, ağ maskesi /16)
- C sınıfı: 192.0.0.0 – 223.255.255.255 (2M+ ağ, ağ maskesi /24)
- Multicast (D sınıfı): 224.0.0.0 – 239.255.255.255 (RFC 5771)
- E sınıfı: 240.0.0.0 – 255.255.255.255

IPv4 protokolünde en fazla 2^{32} (4.294.967.296) tane IP adresi kullanılabilmektedir. Internet'in dünya üzerinde hızla yaygınlaşması, Internet'e bağlı olarak kullanılan cihaz sayısının artması sebebiyle IPv4'de bulunan IP adresleri yetmemeye başlamıştır. Bu sebeple IPv6 geliştirilmiştir. IPv6 ile ilgili bilgiler ilerleyen bölümlerde verilmiştir. IPv4 başlığı Şekil 2.3'de gösterilmiştir.

Şekil 2.3'de görüldüğü gibi en düşük IP başlığı 20 bayttır. IP başlık bilgisinde aşağıdaki alanlar bulunmaktadır.

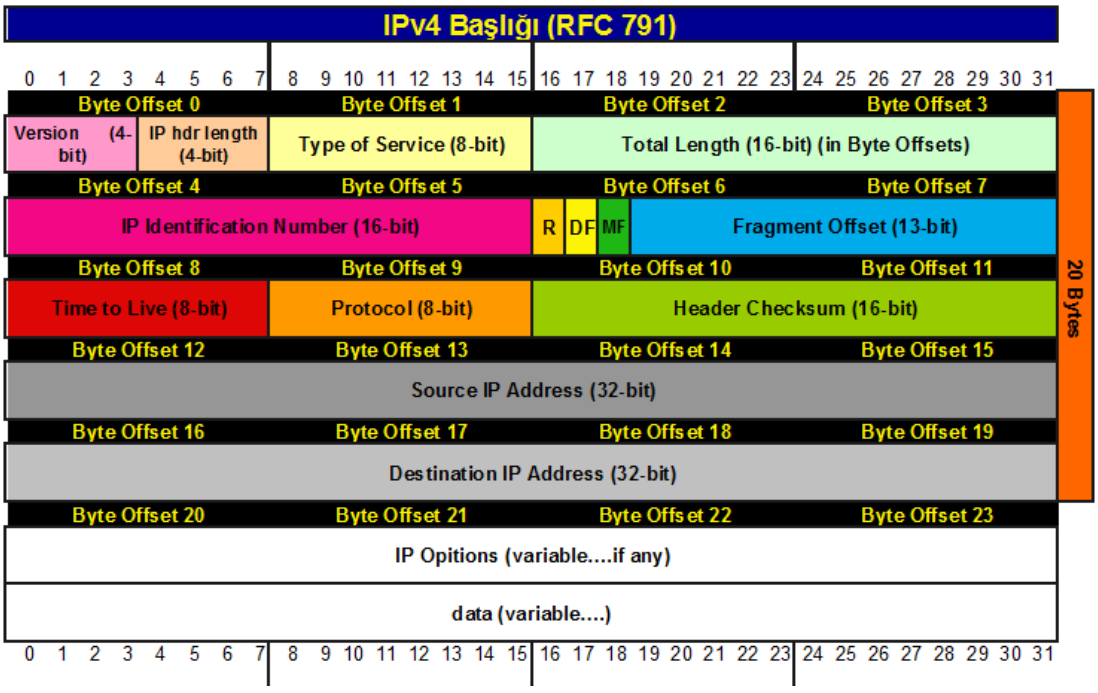
- IP sürümü
- Başlık boyutu
- Servis tipi
- Toplam boyut
- IP ID bilgisi
- Parçalanma bilgileri
- TTL (time to live) bilgisi
- Protokol bilgisi
- Başlık bütünlük kontrol bilgisi
- Kaynak ve hedef IP bilgileri

IP sürüm bilgisi, IPv4 veya IPv6 olduğunu göstermek için kullanılır.

IP başlık bilgisinde bulunan protokol bilgisi, bir üst katman tarafından kullanılan protokolü göstermek içindir.

TTL bilgisi, bir paketin en fazla kaç yönlendirmeden geçebileceğini belirtmek için kullanılır.

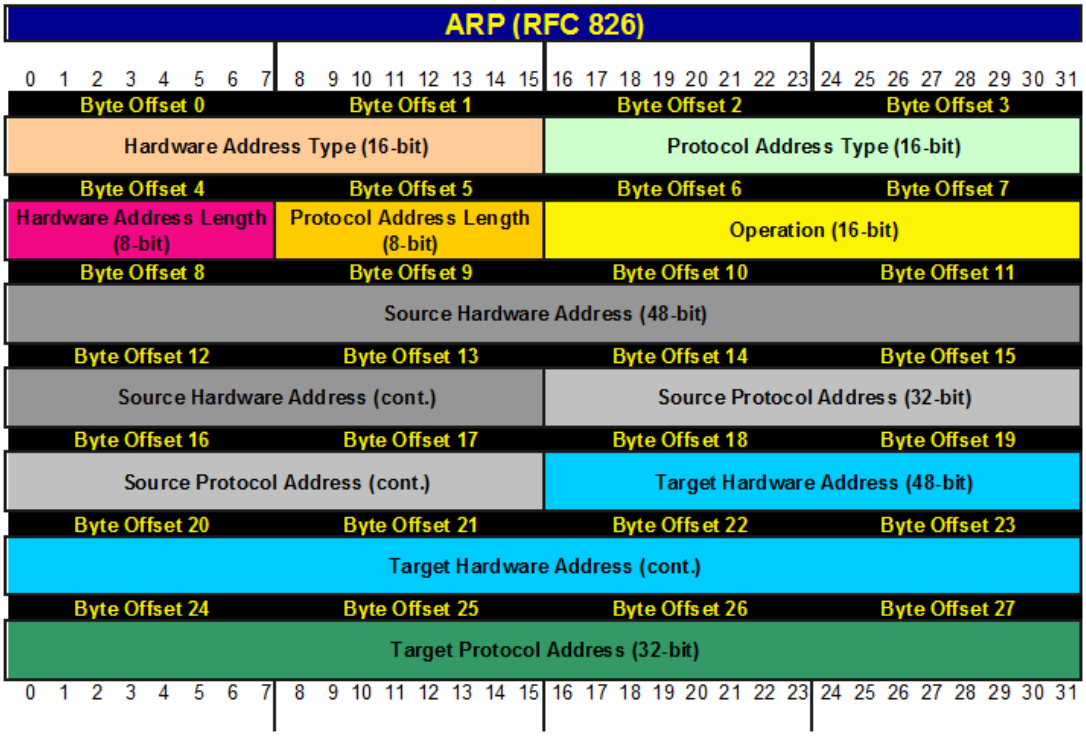
20 baytlık IP başlık bilgisinin ardından varsa, ek IP seçenekleri gelir. Güvenlik, yönlendirme gibi amaçlarla kullanılır.



Şekil 2.3: IP Başlığı.

2.1.2.2. Adres Çözümleme Protokolü (ARP)

Yerel ağlarda en çok kullanılan ağ arayüzü Ethernet'tir. Ethernet arayüzü veya kablosuz Ethernet arayüzü olan ağ kartları ile bilgisayarlar veya cihazlar bu ağa kolayca dâhil edilebilmektedir. Bu ağlarda haberleşme IP adreslerinin bu kartların fiziksel adreslerine dönüştürülmesi ile sağlanır. Bir cihaz kendi ağında bulunan bir IP adresine herhangi bir paket göndermek istediği zaman, işletim sistemi ile olarak bir ARP sorgusu ile hedef IP adresine ait cihazın ağ arayüzünün fiziksel adresini alır. Gönderilecek pakete IP başlık bilgisi eklendikten sonra ARP başlık bilgisi eklenerek paket gönderilir. ARP başlığı Şekil 2.4: ARP başlığı'nda gösterilmiştir.

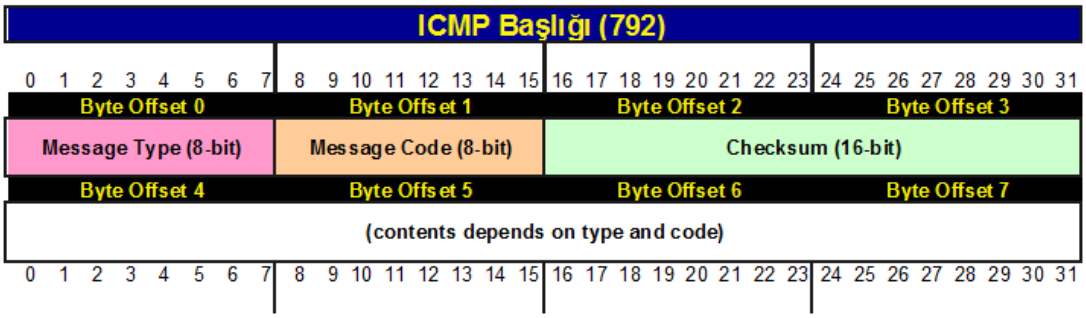


Şekil 2.4: ARP başlığı.

2.1.2.3. Internet Kontrol Mesaj İletişim Protokolü (ICMP)

TCP/IP ağlarında genel olarak kontrol ve hata bildirimi amaçlı olarak kullanılır. Örneğin bir yönlendiricinin erişime izin vermemesi durumunda, yönlendirici tarafından trafiğin engellendiğine dair bir ICMP mesajı alınır. İkinci bir örnek olarak, bir cihaz üzerinde hizmet vermeyen bir porta bağlanılmak istendiğinde benzer bir hata mesajı alınır.

ICMP en çok karşı cihazın açık olup olmadığını anlamak için kullanılır. Karşı cihaza gönderilen "ICMP Echo Request" mesajı, karşı cihaza ulaştığında "ICMP Echo Reply" mesajı ile yanıtlanır. Cevap mesajını alan cihaz, karşı cihazın açık olduğunu anlayacaktır. Şekil 2.5'de görüldüğü üzere, ICMP paketleri tip, kod ve bütünlük kontrol ve içerik alanlarından oluşmaktadır. En düşük 4 bayt olarak gönderilir



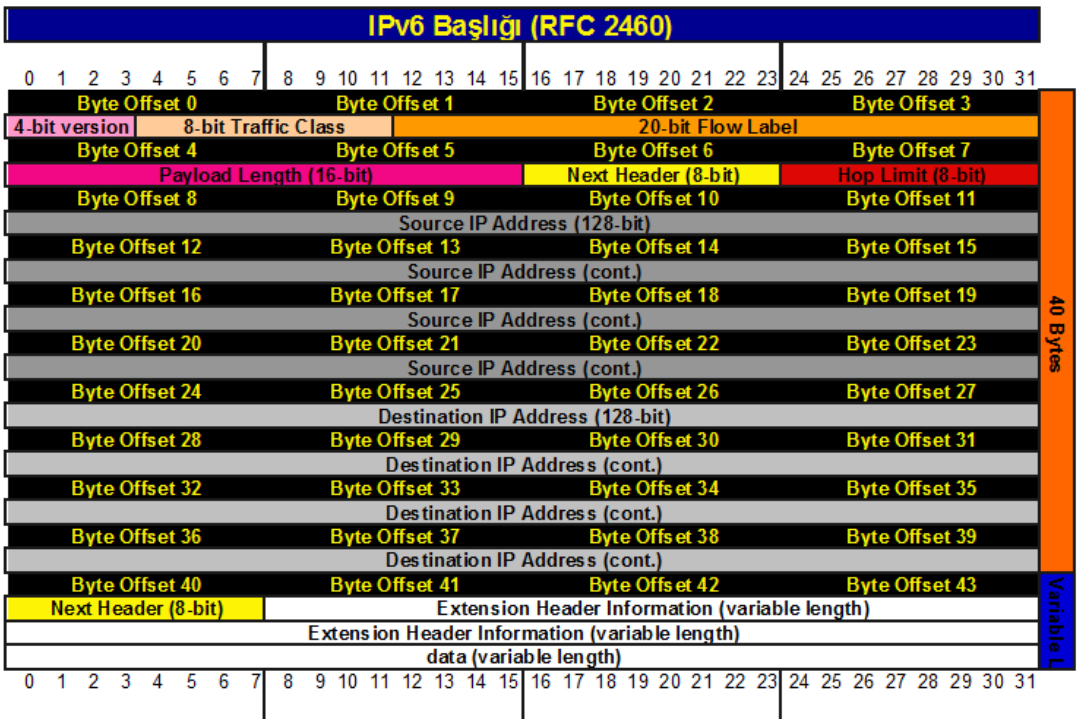
Şekil 2.5: ICMP Başlığı.

2.1.2.4. Internet Protokolü v6 (IPv6)

IP adres sayısının yetersiz olacağının öngörülmesi üzerine IPv6 ile ilgili çalışmalara başlanmıştır. IPv6'da IP adresi alanı için 128 bitlik yer ayrılmıştır. Bu da 2^{128} IP adresi anlamına gelir. IPv4'e göre çok fazla IP adres uzayı mevcuttur.

IPv4'de kullanılan ARP protokolü yerini IPv6'da NDP (Neighbor Discovery Protocol)'ye bırakmıştır. IPv4'de kullanılan yayın paketleri(broadcast), IPv6'da yoktur. IPv6 başlığı Şekil 2.6'da gösterilmiştir.

IPv6'da IP adresleri *fe80:0000:0000:0000:250:26ff:ceb1:aaa1* veya *fe80::250:26ff:ceb1:aaa1* şeklinde gösterilir. Ardışıl olarak gelen sıfırlar, bir kereye mahsus olmak üzere "::" ile değiştirilebilmektedir.



Şekil 2.6: IPv6 Başlığı.

Şekil 2.6'da görüldüğü gibi en küçük IPv6 başlığı 40 bayttır. IPv4 de olduğu gibi IPv6 başlığının da ilk 4 biti, IP sürümünü göstermektedir. IPv4 de bulunan *protokol* alanı IPv6'da *next header* olarak bulunmaktadır. Benzer şekilde IPv4'de bulunan TTL, IPv6'da *hop count* olarak görülmektedir.

TCP ve UDP gibi taşıma katmanı protokolleri IPv6'da da sorunsuz çalışmaktadır. Ancak IPv6'da ICMP protokolü yerine ICMPv6 kullanılmaktadır.

2.1.2.5. Internet Kontrol Mesaj İletişim Protokolü v6 (ICMPv6)

ICMP'ye göre daha fazla kontrol ve hata mesajı içermektedir. IPv6 için hayati önem taşımaktadır. ICMPv6'nın güvenlik duvarları aracılığı ile kesilmesi durumunda IPv6 çalışmayacaktır. Başlık bilgisi olarak ICMP ile aynıdır.

ICMP'nin protokol numarası 1 iken ICMPv6'nın protokol numarası 58'dir.

2.1.3. Taşıma Katmanı

Bilgisayarlar arasında yapılan bağlantının oturumunu yöneten katmandır. Taşınan verinin hizmet düzeyini ve durumunu tanımlar. Uygulama katmanından gelen verinin parçalanarak uygun boyutlarda karşı uca gönderilmesinden sorumludur.

Taşıma katmanı, bağlantı yönetimi, verilerin sıralı gitmesi, güvenilirlik, tıkanıklıktan kaçınma, port numaraları gibi servisler sunar. Taşıma katmanında, tüm paketlere kullandıkları protokole göre bir başlık daha eklenir.

Taşıma katmanında en çok kullanılan protokoller TCP ve UDP'dir. Stream Control Transmission Protocol (SCTP)'de taşıma katmanı protokolüdür.

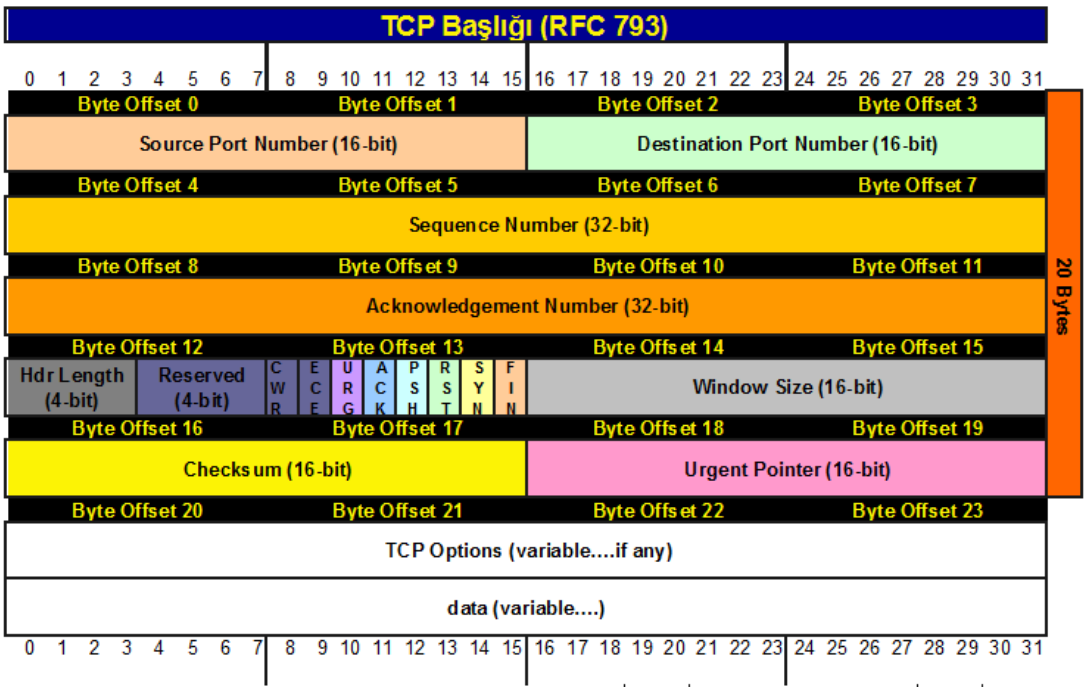
2.1.3.1. Transmission Control Protocol (TCP)

Kayıpsız ve hatasız veri gönderimi sağlamak amacıyla geliştirilmiştir. Bugün Internet'te kullanılan uygulamaların çoğu TCP ile çalışmaktadır. Bu sebeple Internet trafiğinin önemli bir kısmını TCP trafiği oluşturmaktadır.

Uygulama katmanında kullanılan, HTTP, HTTPS, FTP, SMTP, IMAP, POP3 gibi bir çok protokol TCP kullanarak çalışmaktadır. Bu protokollerin yanı sıra DNS protokolü de, ihtiyaç olması durumunda TCP ile haberleşmektedir. TCP ile gönderilen paketlerin doğru ve sıralı olarak karşı noktaya iletilmesi garanti edilir. Herhangi bir paket kaybı veya pakette bozulma olması durumunda ilgili paket yeniden gönderilir.

TCP, bağlantılı bir protokoldür ve iki noktanın veri iletişimine başlayabilmesi için, veriler gönderilmeden önce her iki noktanın anlaşması gerekmektedir. Buna üçlü el sıkışma (3 way handshaking – 3 whs) denir.

Şekil 2.7'de TCP başlığı gösterilmektedir.

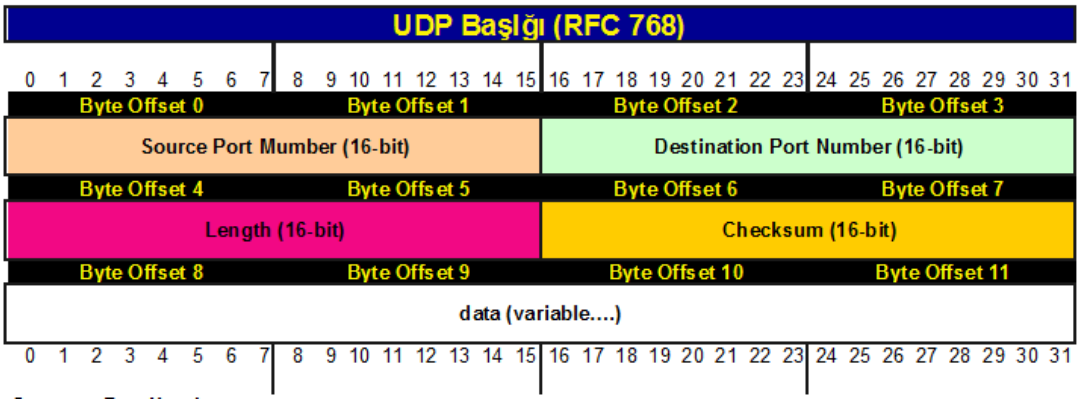


Şekil 2.7: TCP Başlığı.

2.1.3.2. User Datagram Protocol (UDP)

TCP protokolüne göre daha hızlı olan UDP'de paketlerin karşı noktaya ulaştığının, doğru ulaştığının veya sıralı ulaştığının garantisi verilmez. UDP, bağlantı tabanlı olmadığı için veri transferi yapacak noktaların karşılıklı olarak anlaşması gerekmez. UDP'de TCP de olduğu gibi üçlü el sıkışma yapılmaz.

En çok kullanılan UDP protokolleri DNS, SNMP ve TFTP'dir. Ses ve görüntü iletiminde de UDP kullanılır Şekil 2.8'de UDP Başlığı gösterilmiştir.



Şekil 2.8: UDP Başlığı.

2.2. Mobil Cihazlarda Kullanılan Haberleşme Teknolojileri

Mobil cihazların Internet'e bağlanması, 1991 yılında 2nci nesil kablosuz telefon teknolojisi (2G) ile başlamıştır. GSM teknolojisindeki Internet erişim hızı 9.6 kbit/s'dır. Internet'in yaygınlaşması ve bu hızın yeterli olmaması sebebi ile HSCSD ve GPRS standartları çıkartılmıştır.

Uluslararası Telekomünikasyon Birliği (International Telecommunication Union – ITU)'nün 3ncü nesil standartları ile mobil cihazlardan yapılan Internet bağlantıları hızlanmıştır. 3G teknolojisi, Türkiye'de 2009 yılında kullanıma girmiştir.

Dördüncü nesil kablosuz telefon teknolojisi olan 4G, 3G ve 2G standartlarının devamıdır. Bağlantı hızı 3G den çok daha fazladır. Ülkemizde henüz 4G teknolojisi kullanılmamaktadır.

2.3. Sanallaştırma

Sanallaştırma, genel olarak bir işletim sisteminin çeşitli yazılımlar kullanarak bir başka işletim sistemi içerisinde bir pencerede çalıştırılması olarak tanımlanabilir. Farklı işletim sistemleri üzerinde çalıştırılan sanallaştırma yazılımları, hem kişisel bilgisayarlarda sanallaştırılmış işletim sistemi kullanımına hem de sunucu üzerine kurulan sürümleri ile aynı sunucu üzerinde bir çok sanallaştırılmış işletim sisteminin çalıştırılmasına imkân sağlar. Sanallaştırmanın bir çok çeşidi olmasına rağmen bu tez kapsamında sadece işletim sistemi ve ağ arayüzü sanallaştırması kullanılmıştır.

Sanallaştırma ile sadece işletim sistemlerini değil, Android gibi mobil işletim sistemlerini veya Cisco IOS gibi ağ cihazlarının işletim sistemlerini de sanallaştırma ortamında çalıştırmak mümkündür. Sanallaştırma sisteminin en büyük faydaları, ek bir donanım maliyeti olmadan birden fazla işletim sisteminin çalıştırılabilmesi ve sistemin durumunun kaydedilip istenildiği zaman bu duruma dönülebilmesidir. Aşağıda sanallaştırmanın en çok kullanıldığı alanlardan bahsedilmiştir.

2.3.1. Kullanım Alanları

Günümüzde sallaştırma teknolojisinin gelişmesiyle birlikte birçok kurumda kritik sunucular bile sanallaştırılmış olarak kullanılabilmektedir. Aşağıda en çok kullanıldığı alanlar anlatılmıştır.

2.3.1.1. Sunucu Yazılımları

Bazı kurumlar, geliştirme ve test işlemlerini sanallaştırılmış işletim sistemleri üzerinde yapıp, gerçek ortamda ise donanım kullanmaktadır. Bazı kurumlarda ise bir çok sunucu yazılımı, yedeklilik ve yük dağılımı sağlamak açısından sanallaştırılmış ortamda çalıştırılmaktadır.

2.3.1.2. Geliştirme ve Test Ortamları

Bazı donanımların her zaman kullanımı mümkün olmamasından veya ilgili donanım üzerinde test yapılma zorluğundan dolayı sanal işletim sistem olarak çalıştırılmakta ve geliştirme işlemleri sanal işletim sistemleri üzerinde yapılmaktadır. Örnek olarak mobil cihazlarda kullanılan Android işletim sistemi için emülatör yazılımları kullanılmaktadır.

2.3.2. Kullanılan Sanallaştırma Yazılımları

Sunucu veya kişisel bilgisayarlarda en yaygın olarak kullanılan sanallaştırma yazılımları VMware ve VirtualBox'dır.

VMware firması tarafından geliştirilen VMware isimli ürün ailesi oldukça yetenekli olup kişisel kullanım ve sunucu üzerinde kullanılan sürümleri mevcuttur. VMware'in ESX sürümü, sanallaştırma için sunucu platformudur ve üzerinde bir çok sanal işletim sistemi çalıştırabilmektedir. ESX'in verdiği ağ desteği ile sanal anahtarlama cihazları ve VLAN (Virtual Lan - Sanal ağ)'lar oluşturabilmek mümkündür. ESX ile birden fazla sanallaştırma sunucusunu bir küme olarak çalıştırmak ve anlık olarak çalışan sanallaştırılmış işletim sistemlerini diğer sanallaştırma sunucularına hizmet kesintisi olmadan taşımak mümkündür.

Bu tez kapsamında VMware'in Workstation 9 sürümü kullanılmıştır. Bu sürüm kişisel bilgisayarlarda kullanım için uygundur. Aynı anda birden fazla sanallaştırılmış işletim sistemi çalıştırılabilir. Sanallaştırılmış işletim sisteminin ağ arayüzlerini, fiziksel ağlara veya sadece sanallaştırılmış işletim sistemleri arasında oluşturulan ağlara bağlamak mümkündür.

VirtualBox, Oracle firmasına ait sanallaştırma çözümüdür ve açık kaynak kodlu olarak dağıtılmaktadır. Kişisel bilgisayarlarda ve sunucu işletim sistemleri üzerinde çalışabilmektedir. VirtualBox'ın küme çözümü bulunmamaktadır.

2.4. FreeBSD İşletim Sistemi

FreeBSD [Web 7] işletim sistemi x86 uyumlu bir işletim sistemidir. Farklı işlemci mimarileri ile çalışabilen ileri seviye bir işletim sistemidir. Berkeley'deki Kaliforniya Üniversitesinde geliştirilmiş olup, UNIX türevi BSD'yi temel almıştır. Kararlı çalışması, ağ ve sistem performansı, kolay paket yönetimi gibi konularda öne çıkmaktadır. 30 yılı aşkın süredir geliştirilmekte ve iyileştirilmektedir.

FreeBSD'nin gelişmiş ağ özellikleri gömülü cihazlarda da kullanılabilmektedir. FreeBSD, Internet'te mail sunuculardan yönlendirici cihazlara, web sunuculardan gömülü cihazlara kadar birçok alanda kullanılmaktadır.

Port ağacı yapısı ile binlerce yazılım sistem üzerinde kolaylıkla derlenebilmekte veya Internet üzerinde daha önceden derlenmiş olan paketler kurulabilmektedir.

FreeBSD çekirdeği birden fazla TCP tıkanıklık kontrol mekanizmasını desteklemektedir. Çekirdek modülü olarak istenilen tıkanıklık kontrol yöntemi kolaylıkla kullanılabilmektedir.

Açık kaynak kodlu bir işletim sistemi olan FreeBSD, BSD lisanslıdır.

2.5. Çekirdek Programlama

FreeBSD çekirdeği yüklenebilir çekirdek modüllerine izin vermektedir. Bu sebeple geliştiriciler ve sistem yöneticileri için yazılan veya kullanılacak olan çekirdek modülünün dinamik olarak çalışan sisteme eklenmesi ve çıkarılması mümkün olmaktadır. İlerleyen bölümlerde tez çalışmam kapsamında geliştirilen FreeBSD çekirdek modülünde kullanılan, çekirdek programlama tekniklerinden bahsedilmiştir.

2.5.1. En Basit Çekirdek Modülü

FreeBSD üzerindeki çekirdek modüllerine, Dynamic Kernel Linker (KLD) ismi verilir. Bir çekirdek modülün sisteme yüklenebilmesi için o modüle ait bir olay denetimcisi (module event handler) olmalıdır. Olay denetimcisi bir çekirdek modülünün ilk yüklenirken ve bellekten atılırken yapması gereken işlemleri yapar. Örneğin bir bellek ayrılması ve ayrılan belleğin tekrar işletim sistemine geri verilmesi gibi. Bu işlem yükleme ve bellekten atma sırasında *load* isimli fonksiyonun çağrılmasıyla gerçeklenir.

Yüklenen çekirdek modülün kendisini çalışan çekirdeğe kayıt ettirmesi gerekmektedir. Bu kayıt işlemi DECLARE_MODULE makrosu aracılığı ile kolaylıkla yapılabilmektedir.

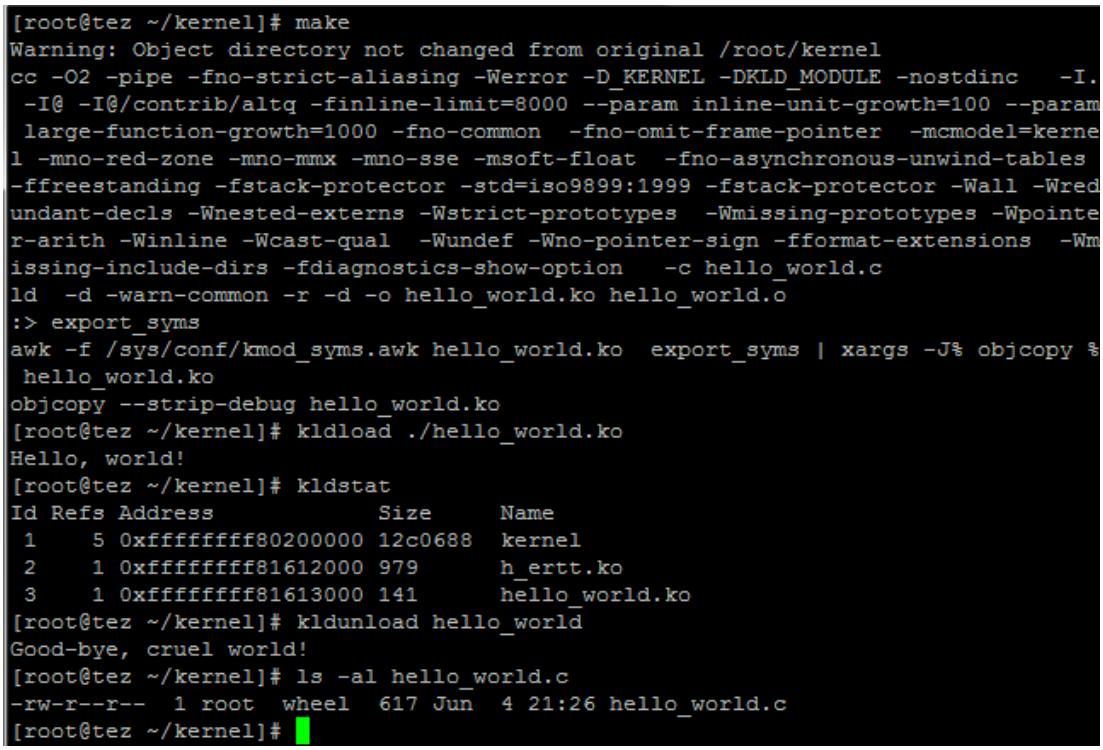
Bir çekirdek modülü yazıp sisteme yüklemek için olay denetimcisi ve kayıt işleminin yapılması yeterlidir. Yazılan çekirdek modülü derlendikten sonra kldload komutu ile sisteme yüklenebilir. Şekil 2.9'da basit bir çekirdek modülün derlenmesi, sisteme yüklenmesi ve kaldırılması gösterilmektedir.

2.5.2. Sistem Çağrıları

İşletim sistemlerinde, uygulamanın işletim sistemi çekirdeğinden istediği servislere sistem çağrıları denir.

Sistem çağrısı modülleri en basit anlamda, çekirdeğe bir sistem çağrısı ekleyen modüllerdir. Sistem çağrıları *sysent[]* isimli bir yapıda tutulurlar. Her sistem çağrısının *sysent* tipinde bir değişkeni bulunmaktadır ve bu değişken parametre sayısı, sistem çağrısı olan fonksiyonun adresi gibi bilgileri tutar.

İşletim sisteminin çekirdeğine yeni bir sistem çağrısı eklenmek istenildiğinde *SYSCALL_MACRO* isimli makro kullanılır.



Şekil 2.9: hello_world çekirdek modülü.

2.5.3. Sistem Çağrılarını Çengelleme

Çengellenen sistem çağrıları aracılığı ile çekirdeğe veya kullanıcıya farklı bilgiler gönderilebilmektedir.

FreeBSD'de bir sistem çağrısını çengellemek için, *sysent[]* dizisindeki hedefteki sistem çağrısının adresi, uygun şekilde yazılmış olan yeni sistem çağrısının adresi ile değiştirilir. İlgili çekirdek modülü bellekten atılırken, çengellenen sistem çağrısının adresi eski haline getirilmelidir. Aksi takdirde sistemin kararsızlaşması ve çökmesi kaçınılmazdır.

2.5.4. Çekirdek Durumları

FreeBSD'de çekirdek ile ilgili veya çekirdek modülleri ile ilgili bazı parametreler kullanıcı tarafından girilebilmektedir. Benzer şekilde çekirdek veya modüllerin durumu hakkında bilgiler kullanıcı tarafından görülebilmektedir. Bu tür çekirdek durumları, *sysctl* komutu ile kullanıcı tarafından okunup, değiştirilebilir.

FreeBSD'de geliştirilen çekirdek modülüne çekirdek durumları eklemek için *SYSCTL_ADD_NODE, SYSCTL_ADD_INT* gibi makrolar kullanılır.

2.5.5. Paket Filtreleme Arayüzü - pfil

Pfil mekanizması, gelen veya giden her paket için özel olarak belirtilmiş fonksiyonun çalıştırılmasına imkân sağlar. Bu konu ile ilgili en güzel örnek paket filtreleme yazılımlarıdır. Pfil mekanizması sayesin gelen ve giden tüm paketlerin başlık bilgileri ve içerik bilgileri kontrol edilerek paketin kabul edilip edilmeyeceğine karar verilebilmektedir [Web 8].

2.5.6. Yardımcı Çengelleme Uygulama İskeleti – hhook

Khook uygulama iskeleti, çekirdekte tanımlanan noktalarda çeşitli çengelleme fonksiyonlarının çalıştırılmasına imkân sağlar. pfil mekanizmasından esinlenilmiştir ve pfil mekanizmasının daha geniş halidir [Web 9].

2.5.7. Çekirdek Yardımcı Uygulama İskeleti – khelp

Khelp, hhook'u dolaylı olarak kullanarak çekirdekte ilgilenilen noktalara çengelleme yapılmasını sağlayan uygulama iskeleti sunar [Web 10] .

Khelp ve hhook uygulama iskeletleri oldukça sıkı entegredirler. Khelp, hhook çengelleme noktalarına khelp modüllerindeki çengelleme fonksiyonları kayıt etmekten ve bellekten atmaktan sorumludur.

Khelp'de de çekirdek modüllerinde olduğu gibi yüklenirken ve bellekten atılırken çalıştırılması gereken fonksiyonların yazılması gerekmektedir.

```
cc -O2 -pipe -fno-strict-aliasing -Werror -D_KERNEL -DKLD_MODULE -nostdinc
00 --param large-function-growth=1000 -fno-common  -fno-omit-frame-pointer  -
onous-unwind-tables -ffreestanding -fstack-protector -std=iso9899:1999 -fstac
 -Wmissing-prototypes -Wpointer-arith -Winline -Wcast-qual  -Wundef -Wno-poin
ption   -c 10.c
ld  -d -warn-common -r -d -o 10.ko 10.o
:> export_syms
awk -f /sys/conf/kmod_syms.awk 10.ko  export_syms | xargs -J% objcopy % 10.ko
objcopy --strip-debug 10.ko
[root@tez ~/tez]# kldload ./10.ko
Hello, world!
KHelp Module init
[root@tez ~/tez]# kldstat
Id Refs Address            Size     Name
 1    6 0xffffffff80200000 12c0688  kernel
 2    1 0xffffffff81612000 979      h_ertt.ko
 3    1 0xffffffff81613000 11ed     10.ko
[root@tez ~/tez]# kldunload 10
KHelp Module destroy
Bye bye!
[root@tez ~/tez]#
```

Şekil 2.10: KHelp modülünün yüklenmesi.

Şekil 2.10'da khelp kullanan bir çekirdek modülünün yüklenmesi ve bellekten atılması gösterilmiştir.

2.5.8. Gelişmiş RTT KHelp Modülü

RTT, h_ertt modülü khelp uygulama iskeletini kullanarak geliştirilmiştir. Bağlantı başına istenildiği anda RTT değerinin hesaplanılmasını sağlar. FreeBSD 9.0 ile gelen bir özellik olan h_ertt modülü, 2010 yılında, Swinburne University of Technology'de NewTCP araştırma projesinde geliştirilmiştir [Web 11].

3. YÖNTEM

Bu başlık altında konu ile ilgili incelenmiş yayınlar ve önerilen yöntem anlatılacaktır.

3.1. Giriş

Benzer çalışmalar incelendiğinde farklı katmanlarda önerilmiş çözümler görülmektedir. Konu ile ilgili olarak önerilen çözümler genel olarak yeni bir TCP eklentisi geliştirilmesi veya yeni bir taşıma katmanı protokolü geliştirilmesi şeklindedir. Yeni bir TCP eklentisi geliştirilmesi veya yeni bir protokol geliştirilmesi durumunda bu geliştirilen protokolün karşı nokta tarafından da desteklenmesi gerekmektedir. Aynı şekilde ağ güvenliği ile ilgili cihazların da bu yeni protokolü veya eklentiyi desteklemesi gerekmektedir. Örneğin geliştirilen bir TCP eklentisi, güvenlik duvarı tarafından tanınmadığı için engellenebilir veya saldırı tespit ve engelleme sistemleri, TCP başlığında desteklemediği bir TCP eklentisi göreceği için ilgili paketleri anormallik olarak algılayıp, alarm üretebilir ve paketi düşürebilir. Yeni bir protokolün ve yeni bir eklentinin olgunlaşması zaman almaktadır ve tam olarak olgunlaşmadan bir çok işletim sistemi ve ağ güvenliği cihazları desteklememektedir.

El değiştirme (vertical handover) için yapılan çalışmalar da ise genel olarak, enerji kullanımının düşük tutulması ve farklı ağ arayüzleri arasında geçiş yapılmasından bahsedilmektedir. Karar mekanizması RTT, jitter,bit hata oranı, ağ bağlantılarının enerji tüketimi, alınan sinyal gücü, kullanıcı öncelikleri, güvenlik gibi parametreleri kullanılmaktadır [Bathich vd., 2012], [Zekri vd. 2011]. Bu alanda yapılan çalışmalarda, ağ arayüzleri arasında yapılacak geçişin kesintisiz olması hedeflenmektedir. Ancak bu alanda yapılan çalışmalar birden fazla ağ arayüzünün aynı anda kullanımına odaklanmamıştır.

Bu tez çalışması kapsamında önerilen yöntemin avantajı, aynı anda her iki ağ arayüzünü birden kullanmasına rağmen sadece istemci tarafındaki ağ yığınında değişiklik yapılmasıdır. Böylece, sadece istemcinin ağ yığınına yapılan eklentinin, karşı nokta veya ağ güvenlik cihazları tarafından bilinmesine ihtiyaç duyulmamaktadır. Aynı şekilde, sistemce çalışan uygulamalarda da herhangi bir

güncelleme yapılması gerekmemektedir. Yapılan eklenti kısaca, TCP bağlantılarını kurmak için kullanılan *connect* sistem çağrısının yeteneklerinin arttırılması ve yapılacak yeni bağlantılar hakkında daha önceden çeşitli bilgilere sahip olmasına dayanır. Böylece mobil cihazlarda daha önceden bağlanılmış adreslere, yeni bir bağlantı açılacağı zaman daha önceden hangi ağ arayüzü üzerinden hedef adrese kaç tane bağlantı yapıldığı, ne kadar veri gönderilip alındığı, ortalama RTT değerleri bilindiği için daha iyi sonuç vereceği düşünülen arayüz seçilerek bağlantı yapılır.

Valdovinos vd. [Valdovinos vd., 2009]'da yeni bir TCP eklentisi geliştirmiştir ancak geliştirilen TCP eklentisini tanımayan güvenlik duvarları ilgili paketleri düşürebilir. Aynı şekilde saldırı tespit sistemleri de bu paketler için çeşitli alarmlar üretebilir. Bu yöntemde birden farklı ağ arayüzlerindeki IP adreslerine sahip cihazların, yeni bir TCP bağlantısı kurmak istediklerinde, önerilen TCP eklentisi aracılığı ile bu IP adreslerinin karşı noktaya bildirilmesine dayanmaktadır. Eğer karşı noktada bu özelliği destekliyorsa, aynı TCP eklentisi ile cevap vermektedir. Bu çözümün avantajı uygulamalarda değişiklik gerektirmemesidir. Sadece TCP nin bu eklentiyi destekleyecek şekilde güncellenmesi gerekmektedir.

Dreibholz vd. [Dreibholz vd., 2011]'de, Natarajan vd. [Natarajan vd., 2009]'da Atiquzzaman vd. [Atiquzzaman vd., 2002]'de *Stream Transmission Control Protocol*'ünden bahsetmiştir. Bu protokolün getirdiği bir çok avantaj olmasına rağmen henüz standartlaşmamış olması ve bazı işletim sistemleri tarafından tam olarak desteklememesi en büyük dezavantajlarındandır. Aynı şekilde SCTP protokolünün sağlıklı çalışabilmesi için güvenlik duvarı gibi cihazların da bu protokolü desteklemesi gerekmektedir. Aynı şekilde uygulamalarında protokol olarak SCTP'yi kullanması gerekmektedir. SCTP'de de TCP'de olduğu gibi çeşitli eklentiler bulunmaktadır. Örneğin dinamik adres yeniden yapılandırma eklentisi sayesinde, cihazların arayüzlerinin bağlantılarının kopması, yeni IP adresi almaları, yeni ağ arayüzlerine sahip olmaları gibi IP değişiklikleri karşı noktaya bildirilmektedir.

Ribeiro vd. [Ribeiro vd., 2006]'da önerdikleri yöntem, birden fazla ağ bağlantısı olan cihazlar arasında gidip elen paketler için asimetrik yol kullanımına dayanmaktadır. Bu yöntemde her nokta, karşı nokta ile arasındaki tüm gidiş-geliş yolları için bir matriste RTT değerlerini tutar ve en uygun arayüzünden paketi gönderir. Dönüş paketi farklı bir arayüzünden gelebilmektedir. Bu yöntemde de karşı

noktanın bu özelliği desteklemesi gerekmektedir. Ayrıca durum korumalı güvenlik duvarları bu paketlerin geçmesine izin vermeyebilir.

3.2. Önerilen Yöntem

Önerilen yöntem, yeni açılacak olan TCP bağlantılarının en uygun ağ arayüzü üzerinden gönderilmesine dayanmaktadır. Uygun ağ arayüzünün seçimi için daha önceden açılmış ve izlenen TCP bağlantılara ait *Round-trip time (RTT)* değerleri ve ağ arayüzü öncelikleri gibi parametreler kullanılmaktadır.

Herhangi bir cihaz, bir ağa bağlandığı zaman, o ağda olmayan cihazlara IP kullanarak ulaşabilmek için yönlendirme tablosuna ihtiyaç duymaktadır. Yönlendirme tablosunda, belirli ağ veya cihaz adresleri için özel olarak girilmiş yönlendirme girdileri bulunabileceği gibi tanımlı olmayan tüm ağ adresleri için kullanılacak olan ön tanımlı ağ geçidi de bulunabilmektedir. Cihaz üzerinde çalışan işletim sistemi, oluşturulan paketin hangi ağ arayüzü üzerinden ve hangi cihaza gönderileceğine, yönlendirme tablosuna bakarak karar vermektedir. Bir işletim sistemi üzerinde herhangi bir ek ayar yapılmadan sadece bir tane ön tanımlı ağ geçidi bulunabilmektedir. Örnek bir yönlendirme tablosu Şekil 3.1'de gösterilmektedir. Bazı işletim sistemlerinde birden fazla ön tanımlı ağ geçidi tanımlanabilmesine rağmen sadece bir tanesi kullanılabilmektedir.

```
Internet:
Destination        Gateway            Flags    Refs      Use  Netif
default            192.168.200.1      UGS         0     7715   em0
127.0.0.1          link#4             UH          0       56   lo0
192.168.200.0/24   link#1             U           0        0   em0
192.168.200.129    link#1             UHS         0        0   lo0
```

Şekil 3.1: Örnek yönlendirme tablosu.

Birden fazla ağ bağlantısı olan cihazlar da aynı yönlendirme tablosu kullanılmaktadır ve tüm yönlendirme işlemleri bu tabloya göre yapılmaktadır. Bu durumda ön tanımlı ağ geçidi için bir öncelik veya bir politika kullanılması gerekmektedir. Bazı işletim sistemleri birden fazla yönlendirme tablosuna *(Forwarding Information Base - FIB)* izin vermektedir. Bu durumda her yönlendirme

tablosuna bir adet ön tanımlı ağ geçidi eklenebilmektedir. *FIB*, genelde yönlendirici cihazlar üzerinde kullanılmaktadır.

FreeBSD işletim sistemi 16 farklı yönlendirme tablosuna izin vermektedir [Web 12]. FreeBSD'de bulunan *setfib* [Web 13] komutu ile çalıştırılacak olan program için *FIB* tablosu önceden belirlenebilmektedir.

Uygun olduğuna karar verilen ağ arayüzü seçildikten sonra, yapılacak olan bağlantı ile ilgili bilgileri tutan *socket* değişkeninin yönlendirme tablosu (Forwarding Information Base – FIB) ayarı değiştirilir ve paketin FIB'de tanımlanmış olan ağ arayüzünden gönderilmesi sağlanır. Önerilen yöntemde her ağ arayüzü için bir FIB tablosu tutulmaktadır. Tutulan FIB tablolarının hepsinde ön tanımlı ağ geçidi farklıdır. Şekil 3.2'de görüldüğü üzere, ön tanımlı ağ geçidi *192.168.42.129* IP adresidir ve bu IP adresine ulaşabilmek için *em1* ağ arayüzü kullanılmalıdır.

```
[root@tez ~]# setfib 1 netstat -nr
Routing tables

Internet:
Destination        Gateway            Flags    Refs      Use  Netif
default            192.168.42.129     UGS         0     5161    em1
10.100.0.0/16      link#1             U           0        0    em0
127.0.0.1          link#4             UH          0        0    lo0
192.168.42.0/24    link#2             U           0        0    em1
```

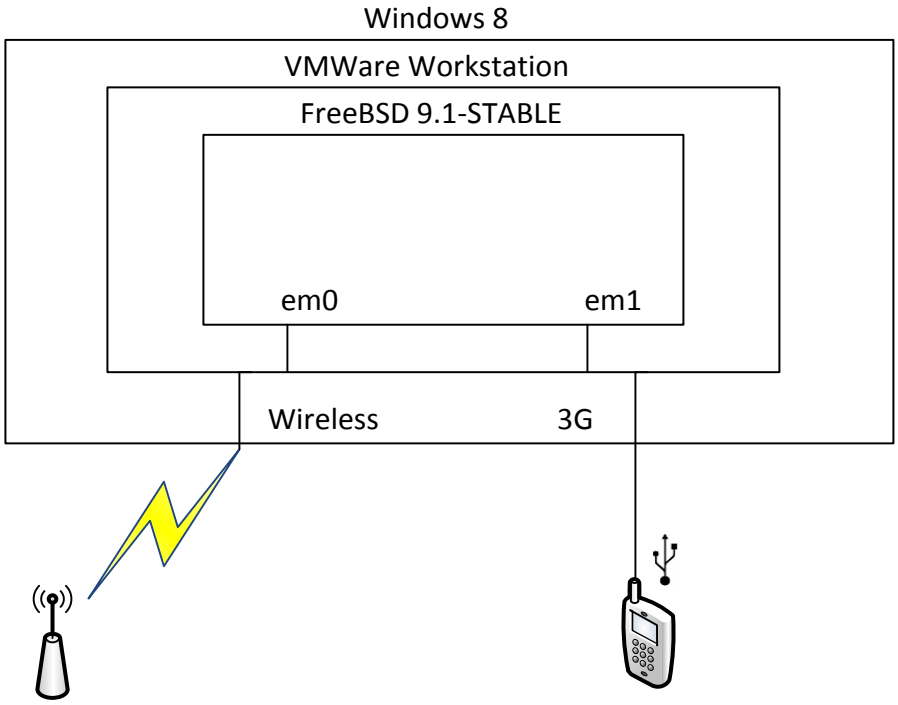
Şekil 3.2: FIB tablosu örneği.

RTT, bir paketin karşı cihaza gönderilip, karşı cihazdan gelen alındı bilgisi alınana kadar geçen zaman olarak nitelendirilir. Ping süresi olarak da bilinir ve *ping* komutu kullanılarak ölçülebilir. *RTT*'nin büyük olması karşı cihaz ile aramızdaki bağlantının yavaş olması anlamına gelir. Önerilen yöntemde, yapılan bağlantıların *RTT* değerlerinin sürekli takip edilmesi gerekmektedir.

NewTCP araştırmaları sırasında, gecikme ve hız tabanlı tıkanıklık algoritmalarında kullanılmak üzere *ERTT* [Web 14] isimli bir çekirdek modülü geliştirilmiştir. Bu modül FreeBSD 9.0 [Web 15] ile gelen *hhook* ve *khelp* çekirdek programlama arayüzleri aracılığı ile çalışmaktadır. Bu modül sayesinde TCP bağlantılarına ait *RTT* değerleri sürekli alınabilmektedir.

Önerilen yöntem için geliştirilen kodlar sanallaştırılmış ortamda, FreeBSD üzerinde C dili kullanılarak çekirdek modülü olarak geliştirilmiştir. Açılan bağlantıların izlenebilmesi ve yeni açılacak bağlantıların kullanacağı arayüzün

değiştirilebilmesi için FreeBSD'nin *connect* [Web 16] sistem çağrısı değiştirilmiştir. Test ve geliştirme ortamı Şekil 3.3'de gösterilmiştir.



Şekil 3.3: Test ve geliştirme ortamı.

Önerilen yöntemde istemci tarafından yapılan tüm TCP bağlantılarının izlenmesi ve çeşitli parametrelere göre yeni açılacak olan bağlantıların kullanması gereken ağ arayüzünün seçimi hedeflenmektedir. Bu parametrelerden arayüz önceliği, ön tanımlı olarak verilebildiği gibi daha sonradan da *sysctl* [Web 17] komutu yardımı ile değiştirilebilmektedir. Diğer tutulan parametreler ise *Time-To-Live (TTL), RTT*, gönderilen ve alınan paket sayıları gibi bağlantıya özgü parametrelerdir.

İşletim sisteminin *connect* sistem çağrısı değiştirildiği için, herhangi bir yazılım *connect* sistem çağrısını kullanarak bir bağlantı açmak istediğinde ilk olarak geliştirilmiş olan kod çalışmaktadır. Daha önceden bağlanılmış bir hedefe tekrar bağlanılmak istendiğinde IP adresi ve hedef port numarasına göre tablodan en uygun *FIB* tablosu bulunmaktadır. *FIB* tablosu bulunurken, daha önceden aynı hedefe doğru yapılmış olan bağlantıların *RTT* değerleri, daha önceki bağlantıların paket sayıları ve ağ arayüzlerinin öncelik değerleri kullanılmaktadır. Daha uygun olduğuna karar verilen FIB tablosu yardımı ile bağlantı isteğinin gönderileceği ağ arayüzü seçilir ve

soket parametreleri değiştirilir. Değiştirilmiş soket parametreleri kullanılarak işletim sisteminin gerçek *connect* sistem çağrısı çalıştırılır.

Bağlantı sırasında hedefe gidip gelen paketlerin *RTT* değerleri *h_ertt* yardımı ile hesaplanır ve daha sonra aynı hedefe açılacak olan yeni bağlantılara referans olması için tablo güncellenir. Herhangi bir hedefe yapılan ilk bağlantı ilk ağ arayüzü üzerinden gönderilir. Daha sonra aynı hedefe yapılan sonraki bağlantılar sırasıyla diğer ağ arayüzlerinden yapılarak, tüm ağ arayüzlerinin ilgili hedefe doğru olan bağlantısının parametreleri alınır. Tüm ağ arayüzlerinin *RTT* değerleri alındıktan sonra yeni yapılacak bağlantılar için kullanılacak *FIB* tablosu bulunarak ilgili ağ arayüzünden bağlanılması sağlanır.

3.2.1. Çekirdek Modülü

Geliştirilen çekirdek modülünün önemli kısımları aşağıda anlatılmıştır.

3.2.1.1. Çekirdek Modülünün Tanıtılması

Geliştirilen çekirdek modülünün sisteme yüklenirken DECLARE_MODULE makrosunu çalıştırılması gerekmektedir. İlgili kod satırı Şekil 3.4'de gösterilmiştir.

```
DECLARE_MODULE(connect_hook, connect_hook_mod,  SI_SUB_DRIVERS,
SI_ORDER_MIDDLE);
```

Şekil 3.4: Çekirdek modülünün tanıtılması.

3.2.1.2. KHelp Modülünün Tanıtılması

KHelp modülünün sisteme tanıtılması KHELP_DECLARE_MOD_UMA makrosunun çalıştırılması gerekmektedir. İlgili kod satırı Şekil 3.5'de gösterilmiştir.

```
KHELP_DECLARE_MOD_UMA(tertt, &tertt_helper, tertt_hooks, 1,
sizeof(struct tertt), NULL, NULL);
```

Şekil 3.5: KHelp modülünün tanıtılması.

3.2.1.3. Bağlantı Tablosu

Yapılan bağlantıların tutulması için kullanılan bağlantı tablosunun veri yapısı Şekil 3.6'da verilmiştir.

```
// Connection Table
struct conentry  {
        struct in_addr ip_dst;
        struct in_addr ip_src;
        uint32_t con_count;
        u_short th_dport;
        u_short rtt_last_in;
        u_short rtt_last_out;
        u_short rtt_avg_in;
        u_short rtt_avg_out;
        u_short pkt_count_in;
        u_short pkt_count_out;
        u_short fib;
        u_char ttl;
} ;
struct conentry **contable;
```

Şekil 3.6: Bağlantı tablosu.

Bağlantı tablosu giden ve gelen her paket için güncellenmektedir.

3.2.1.4. Sistem Çağrısı

Geliştirilen çekirdek modülü *connect* sistem çağrısını değiştirmek için *connect_hook* isimli bir fonksiyon yazılmıştır.

Çekirdek modülü sisteme yüklendiğinde Şekil 3.7'de gösterilen C kod satırı çalıştırılarak, connect sistem çağrısının bellekteki adresi değiştirilir.

```
sysent[SYS_connect].sy_call = (sy_call_t *)connect_hook;
```

Şekil 3.7: Sistem cağrısının değiştirilmesi.

Çekirdek modülü bellekten atılırken Şekil 3.8'da gösterilen C kod satırı çalıştırılarak, connect sistem çağrısının bellekteki adresi eski haline getirilir.

```
sysent[SYS_connect].sy_call = (sy_call_t *)sys_connect;
```

Şekil 3.8: Sistem çağrısının eski haline getirilmesi.

Değiştirilen yeni sistem çağrısı, gelen tüm bağlantı istekleri için Internet katmanı ve taşıma katmanı protokollerine bakmaktadır. IP ve TCP protokolleri haricinde gelen tüm istekler sistemin orijinal sistem çağrısına yönlendirilir.

IP ve TCP için gelen bağlantı istekleri için, daha öncede hedef IP adresine veya portuna bağlantı yapılıp yapılmadığı bağlantı tablosundan aranır. Bağlantı yapılmamışsa, sistemde tanımlanmış ilk FIB kullanılarak orijinal sistem çağrısı çağrılır ve bağlantı kurulur. Bağlantı başarılı bir şekilde kurulmuş ise, bağlantı tablosuna bu bağlantı ile ilgili bir kayıt eklenir. Daha önceden bağlantı yapılmışsa, hangi FIB'ler kullanılarak bağlantı yapıldığına bakılır. Eğer hiç kullanılmamış bir FIB varsa, ilgili FIB kullanılarak bağlantı kurulur. Bağlantı başarılı bir şekilde kurulmuş ise, bağlantı tablosuna bu bağlantı ile ilgili bir kayıt eklenir. Eğer daha önce tüm FIB'ler kullanılarak bağlanılmışsa, en uygun FIB'in seçilmesi gerekmektedir. Uygun FIB seçiminde tutulan bilgiler ve kullanıcı tarafından belirlenen ağ arayüzü öncelik değeri etkendir. Uygun FIB seçildikten sonra, bu bağlantının FIB değeri değiştirilerek istenilen ağ arayüzünden paketin gönderilmesi sağlanmaktadır.

Bu işlem sadece yeni bağlantı yapılmak istendiği zaman çalışmaktadır. Bağlantının devamındaki paketler, işletim sistemi bu bağlantıyı hangi ağ arayüzünden yaptığını bildiği için o arayüzden göndermeye devam edecektir. Bu sayede giden paketler, gönderildikleri arayüzün IP adresine sahip olarak karşı noktaya ulaşacaktır.

3.2.1.5. RTT Hesaplanması

2.5.8'de anlatılan gelişmiş RTT KHelp Modülü sayesinde gelen ve giden paketler için RTT değerleri hesaplanmakta ve alınabilmektedir. Yapılan çengellemede kullanılan yapılar Şekil 3.9'da gösterilmiştir.

```
struct helper tertt_helper = {
        .mod_init = tertt_mod_init,
        .mod_destroy = tertt_mod_destroy,
        .h_flags = HELPER_NEEDS_OSD,
        .h_classes = HELPER_CLASS_TCP
};
struct hookinfo tertt_hooks[] = {
        {
                .hook_type = HHOOK_TYPE_TCP,
                .hook_id = HHOOK_TCP_EST_IN,
                .hook_udata = NULL,
                .hook_func = &tertt_hook
        },
        {
                .hook_type = HHOOK_TYPE_TCP,
                .hook_id = HHOOK_TCP_EST_OUT,
                .hook_udata = NULL,
                .hook_func = &tertt_hook
        }
};
```

Şekil 3.9: ERTT çengellemesinde kullanılan yapılar.

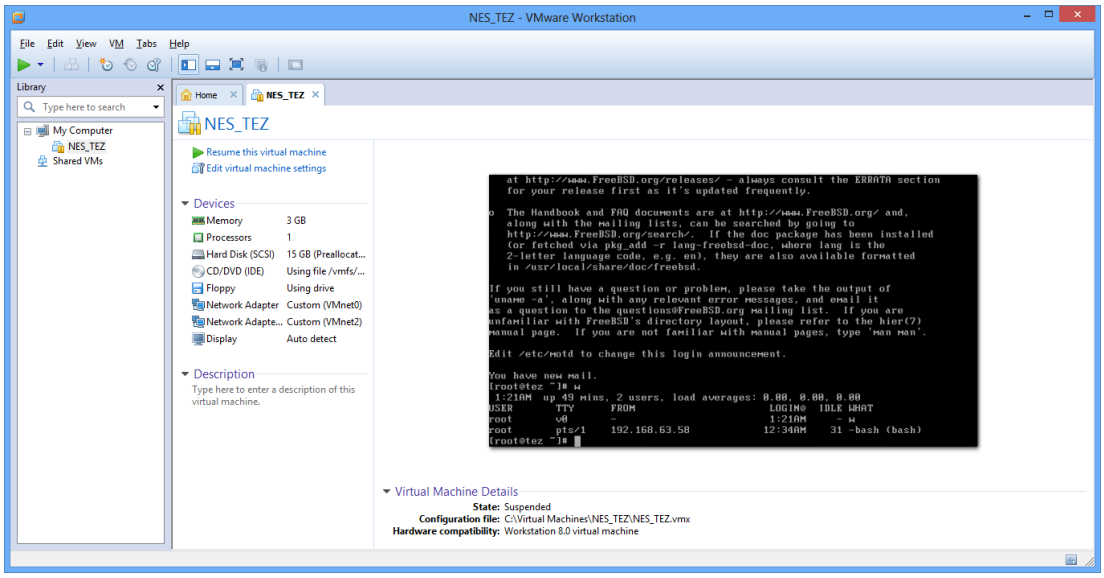
Şekil 3.9'da gösterilen *helper* yapısı, modül sisteme yüklendiğinde ve bellekten atıldığında çalıştırılacak olan fonksiyonları ve çengellemenin sınıfını içermektedir. Bir sonraki yapı olan *hookinfo* yapısı ise, yapılan çengellemenin tipi ve çalıştırılacak fonksiyonu tanımlamaktadır.

Bir paket geldiğinde veya sistemden gönderildiğinde ilgili fonksiyon çalıştırılır. Gelişmiş RTT modülünün sistemde yüklenmiş olan son sürümü, ilgili fonksiyona sadece TCP ile ilgili başlıkları göndermektedir. Ancak tez kapsamında TCP başlık bilgisinin yanı sıra, IP başlık bilgisini de göndermesi için h_ertt çekirdek modülünde değişiklikler yapılmıştır.

tertt_hook fonksiyonu, kendisine parametre olarak gelen veri yapısından TCP ve IP başlıklarını almaktadır. Yapılan kontrollerde fonksiyona parametre olarak gelen paketin, hangi bağlantıya ait olduğu, bağlantı tablosundan bulunmaktadır. Bulunan kayıt güncellenmektedir.

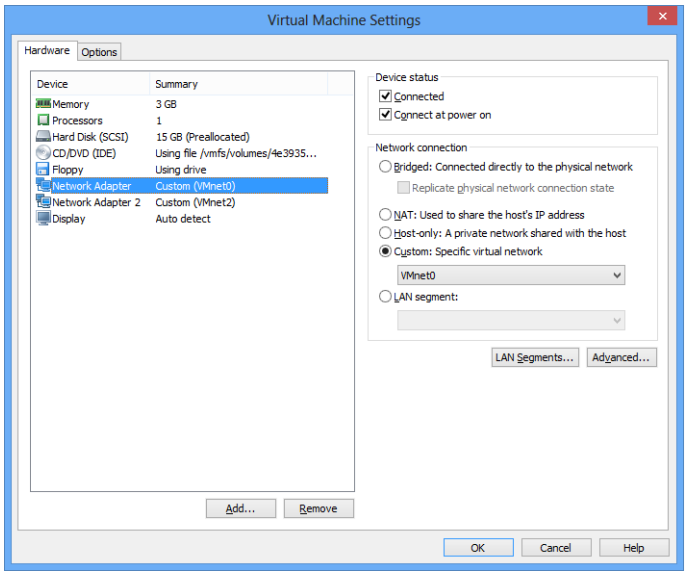
3.2.2. Geliştirme ve Test Ortamı

Bu tez kapsamında kullanılan sanallaştırma yazılımının çalıştığı işletim sistemi Windows 8'dir. VMware Workstation 9 aracılığı ile FreeBSD 9.1 işletim sistemi sanallaştırılmıştır. Şekil 3.10'de VMware Workstation 9'un ana ekranı ve sanallaştırılmış FreeBSD işletim sisteminin donanım özellikleri görülmektedir.



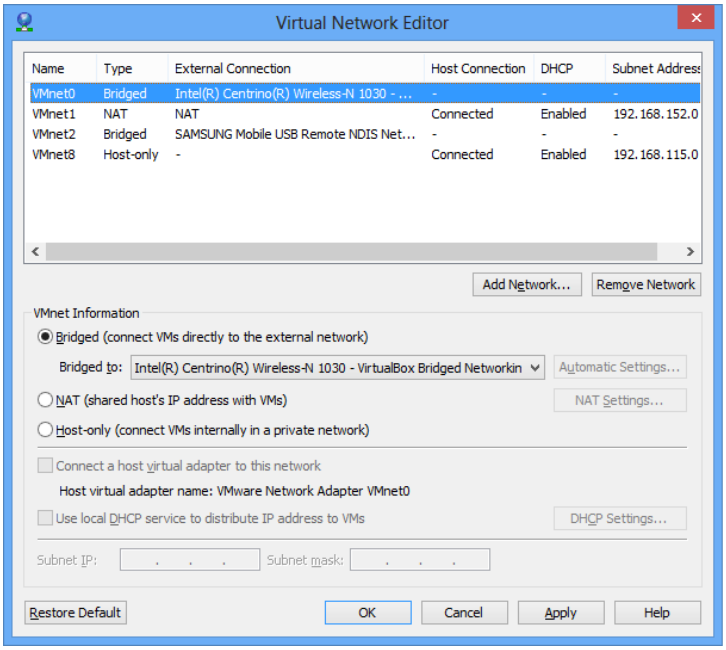
Şekil 3.10: VMWare Workstation genel görünüm.

Sanallaştırılmış FreeBSD işletim sisteminde 2 adet ağ arayüzü bulunmaktadır. Bu ağ arayüzleri Windows 8'de bulunan kablosuz ağ arayüzüne ve USB kablo ile bağlanan mobil cihazın kullandığı 3G ağ arayüzüne direk olarak bağlanmıştır. Yapılan çalışmada kullanılan ağ arayüzlerinin her ikiside köprü(bridge) modunda fiziksel ağ arayüzlerine bağlanmıştır. Sanallaştırılmış işletim sisteminde her iki ağ arayüzünün de IP adreslerini DHCP aracılığı ile alması sağlanmıştır. Ağ arayüzleri köprü modunda bağlandığı için, alınan IP adresleri fiziksel ağa ait IP adresleridir.



Şekil 3.11: Sanal işletim sistemi donanım özellikleri.

Şekil 3.11'de ağ arayüzlerinin bağlantı şekilleri ve sanal makinenin donanim özellikleri, Şekil 3.12'de ise VMware Workstation 9'un sanal ağ yapılandırmaları görülmektedir.



Şekil 3.12: VMware Workstation 9 sanal ağ yapılandırmaları.

3.2.3. Test Metodolojisi

Yapılan tüm testler Internet ortamında yapılmıştır. Kablosuz ağ ve cep telefonu aracılığı ile USB üzerinden Internet erişimi ile testler yapılmıştır.

Tüm testlerde 10 adet HTTP GET isteği paralel olarak gönderilmiştir ve bu işleme toplam hedef istek sayısına ulaşana kadar devam edilmiştir. Hedef adres olarak Amerika'da sunucu barındırma hizmeti veren firmada tutulan bir web adresi seçilmiştir. Yapılacak testlerin yoğun olması ve hedef sunucuya saldırı olarak düşünülmemesi için çok kullanılan web siteleri test için seçilmemiştir. Gönderilen isteklerin, normal HTTP GET isteklerinden herhangi bir farkı bulunmamaktadır. Ardışıl olarak 20 adet bağlantı açan bir betik, paralel olarak 10 kez çağrılmıştır. Bu sayede toplamda 200 adet istek gönderilmiştir. Sunucuda Apache web sunucusu çalışmaktadır. İsteklere cevap olarak gelen dosya boyutu 20KB'dır. Yapılan HTTP isteklerin sıkıştırma etkinleştirilmemiştir.

3.2.4. Test Sonuçlarının Yorumlanması

Çekirdek modülü tarafından her yeni bağlantı açıldığında, bağlantı tablosunda bulunan girdiler *syslog* mesajı olarak gönderilmiştir. Modülün çalıştığı test makinesinde bulunan *syslog* yazılımı, bu mesajlar */var/log/messages* dosyasına eklemektedir. Test yapılmadan önce bu dosya temizlenmekte ve *syslog* servisi yeniden başlatılmaktadır. Yapılan testlerde alınan bir bağlantı tablosu çıktısı Şekil 3.13'de gösterilmiştir.

```
Apr 28 00:44:28 tez kernel: ----------------------[ TABLE BEGIN ]-----------------------------------
Apr 28 00:44:28 tez kernel: GCC: 40 ENTRY 0 Count: 24 192.168.63.158 -> 64.90.191.122:80 FIB: 0 RTT: 258 TTL: 37
Apr 28 00:44:28 tez kernel: GCC: 40 ENTRY 1 Count: 16 192.168.42.131 -> 64.90.191.122:80 FIB: 1 RTT: 359 TTL: 43
Apr 28 00:44:28 tez kernel: ----------------------[ TABLE END ]-------------------------------------
```

Şekil 3.13: Bağlantı tablosu.

Test ortamında 2 farklı ağ arayüzü bulunduğu için bağlantılar bu ağ arayüzleri arasında dağıtılmaktadır.

Şekil 3.13'de gösterilen tabloda, *GCC*, çekirdek modülü tarafından kontrol edilerek yapılan bağlantı sayısını göstermektedir. *Entry*'ler tabloda bulunan girdiler anlamına gelmektedir. Bu tabloda toplamda 2 adet girdi bulunmaktadır. *Count* ile gösterilen kısım, ilgili girdinin ait olduğu FIB'in kaç kez kullanıldığını

göstermektedir. Daha sonra sırasıyla, istemci ve sunucu IP ve port bilgileri, kullanılan FIB, ortalama RTT değeri ve bağlantının TTL değeri bulunmaktadır. Test başlatılmadan önce başlatılan başka bir betik, saniyede 10 kez sistemde bulunan güvenlik duvarının durum tablosunu kontrol ederek hedef adrese kaç adet bağlantı yapıldığını kayıt altına almaktadır.

```
23:05:46 CCC0: 2 CCC1: 8
23:05:47 CCC0: 2 CCC1: 8
23:05:47 CCC0: 2 CCC1: 8
23:05:47 CCC0: 2 CCC1: 8
23:05:47 CCC0: 1 CCC1: 8
23:05:47 CCC0: 1 CCC1: 8
23:05:47 CCC0: 2 CCC1: 8
23:05:47 CCC0: 2 CCC1: 8
23:05:47 CCC0: 2 CCC1: 8
23:05:47 CCC0: 2 CCC1: 8
23:05:47 CCC0: 2 CCC1: 8
23:05:47 CCC0: 2 CCC1: 8
23:05:47 CCC0: 2 CCC1: 8
23:05:47 CCC0: 2 CCC1: 8
23:05:47 CCC0: 2 CCC1: 8
23:05:48 CCC0: 2 CCC1: 7
```

Şekil 3.14: Anlık bağlantı sayıları.

Anlık bağlantı sayıları ve çekirdek mesajları kayıt altına alındıktan sonra, yazılmış olan başka bir betik kullanılarak bu bilgiler anlamlı hale getirilir.

TIME	GCC	CCC0	CCC1	COUNT0	COUNT1	RTT0	RTT1	TTL0	TTL1
23:05:26	1	1	0	0	0	0	0	64	0
23:05:30	2	0	1	0	0	266	0	37	64
23:05:33	3	9	0	1	0	266	884	37	43
23:05:33	4	9	0	2	0	266	884	37	43
23:05:33	5	9	0	3	0	266	884	37	43
23:05:33	6	9	0	4	0	266	884	37	43
23:05:33	7	9	0	5	0	266	884	37	43
23:05:33	8	9	0	6	0	266	884	37	43
23:05:33	9	9	0	7	0	266	884	37	43
23:05:33	10	9	0	8	0	266	884	37	43
23:05:33	11	9	0	9	0	266	884	37	43
23:05:33	12	9	0	10	0	266	884	37	43
23:05:36	13	2	7	10	1	266	884	37	43
23:05:36	14	2	7	10	2	266	884	37	43
23:05:36	15	2	7	10	3	266	884	37	43
23:05:36	16	2	7	10	4	266	884	37	43

Şekil 3.15: Birleştirilmiş sonuçlar.

Şekil 3.15'de gösterilen her satır bir bağlantı anlamına gelmektedir. GCC toplam bağlantı sayısını, CCC0 ve CCC1 anlık bağlantı sayılarını, COUNT0 ve COUNT1 ilgili FIB'lere ait toplam bağlantı sayısını, RTT0 ve RTT1 bağlantıların RTT değerlerini, TTL0 ve TTL1 ise bağlantıların TTL değerlerini göstermektedir.

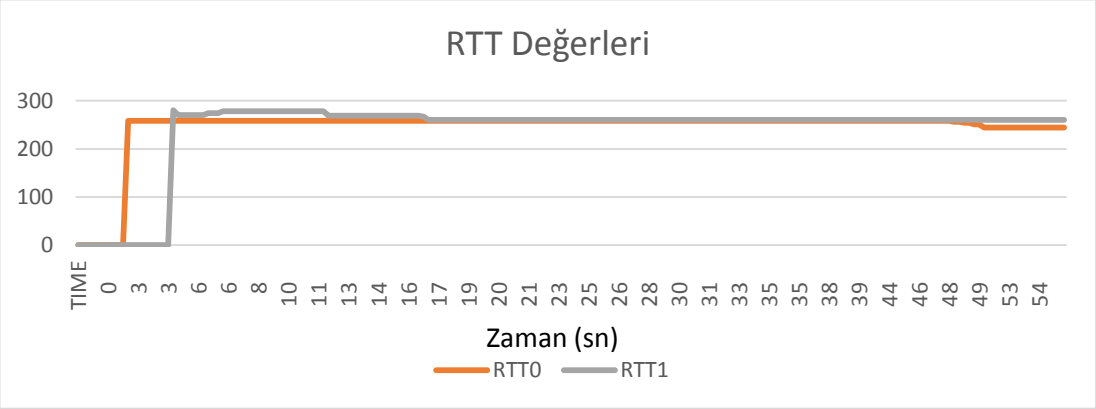
Bu sonuçlar, Microsoft Excel'e aktarılarak çeşitli performans grafikleri üretilmektedir.

Yapılan tüm testlerde yukarıda bahsedilen adımlar uygulanmıştır. Geliştirilen betikler sonuçların birleştirilmesinde ve yorumlanmasında büyük rol oynamaktadır.

4. DENEYSEL ÇALIŞMALAR

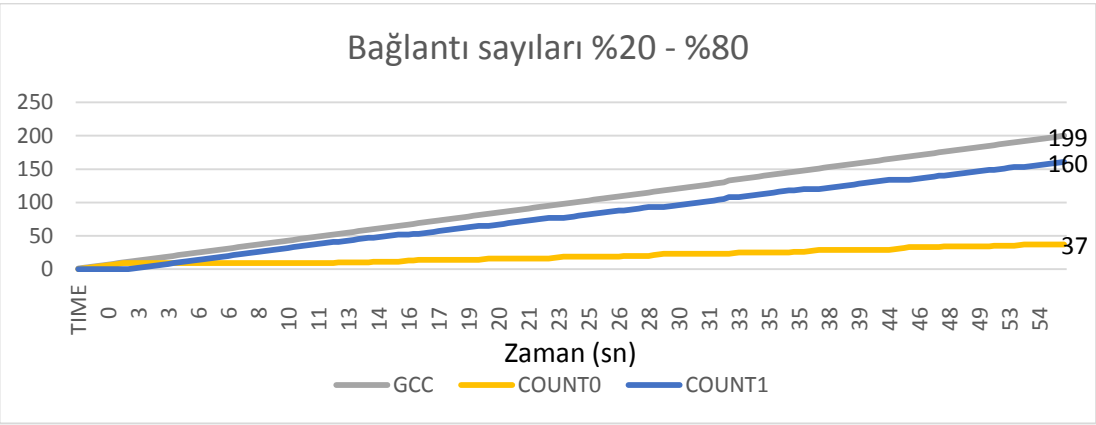
Bu başlık altında, yapılan testler ve test sonuçları anlatılmıştır.

Yapılan ilk testte toplam 200 adet bağlantı açılmıştır. Bu test 3G bağlantısının hızlı olduğu bir ortamda yapılmıştır. Ağ arayüzlerinde öncelik olarak kablosuz ağ bağlantısı %20, 3G ağ bağlantısı %80 olarak verilmiştir.



Şekil 4.1: Test 1: %20-%80 RTT değerleri.

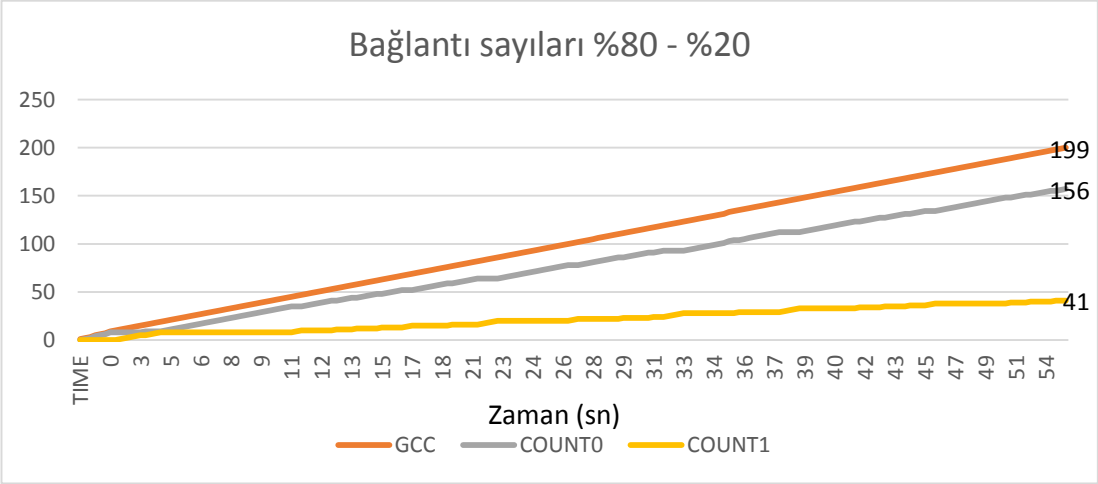
Şekil 4.2'de görüldüğü üzere, her iki ağa ait RTT değerleri birbirine çok yakındır.



Şekil 4.2: Test 1: %20-%80 bağlantı sayıları.

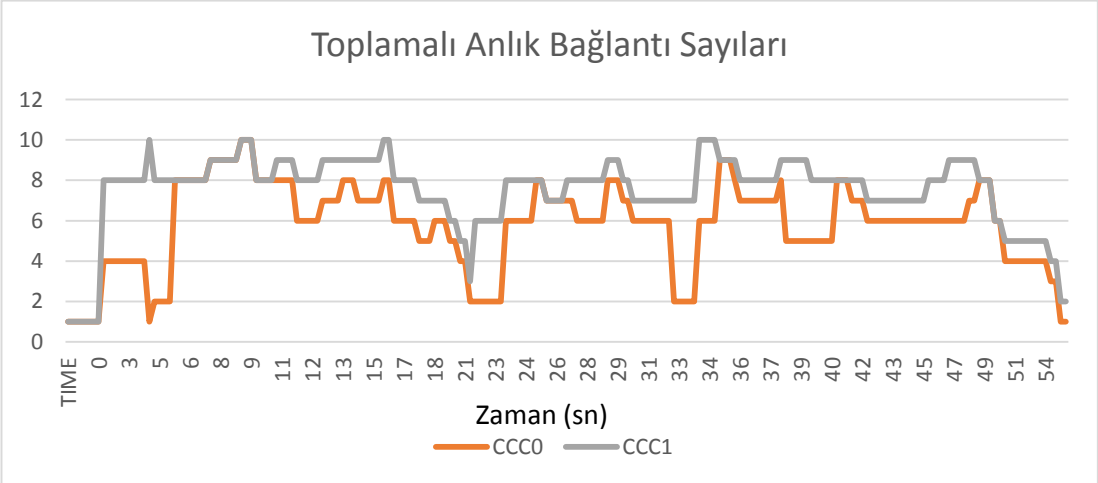
Şekil 4.2'den anlaşılacağı gibi RTT değerlerinin birbirine çok yakın olması dağılımın öncelik oranına çok yakın yapılmasına sebep olmuştur.

Aynı ortamda yapılan, kablosuz ağ önceliği %80 iken, 3G önceliği %20 verildiğinde sonuçlar bir önceki teste benzer çıkmaktadır.



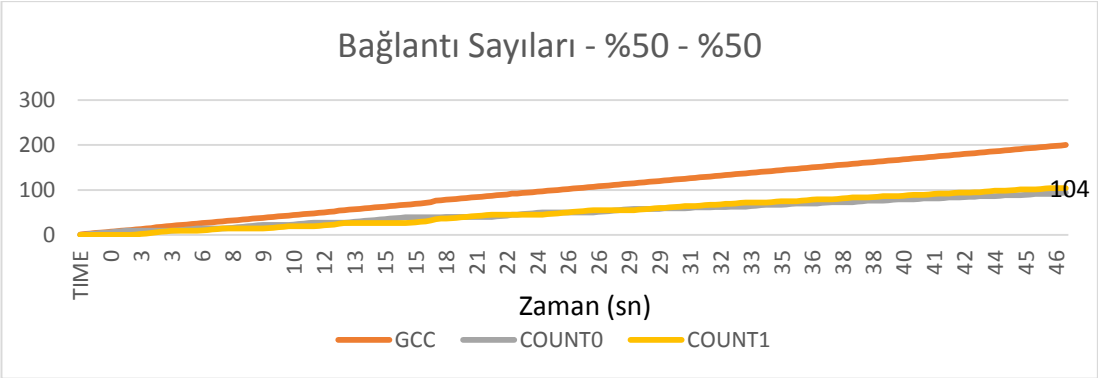
Şekil 4.3: Test 2: %80-%20 bağlantı sayıları.

Şekil 4.4'da bu teste ait anlık bağlantı sayıları gösterilmektedir.



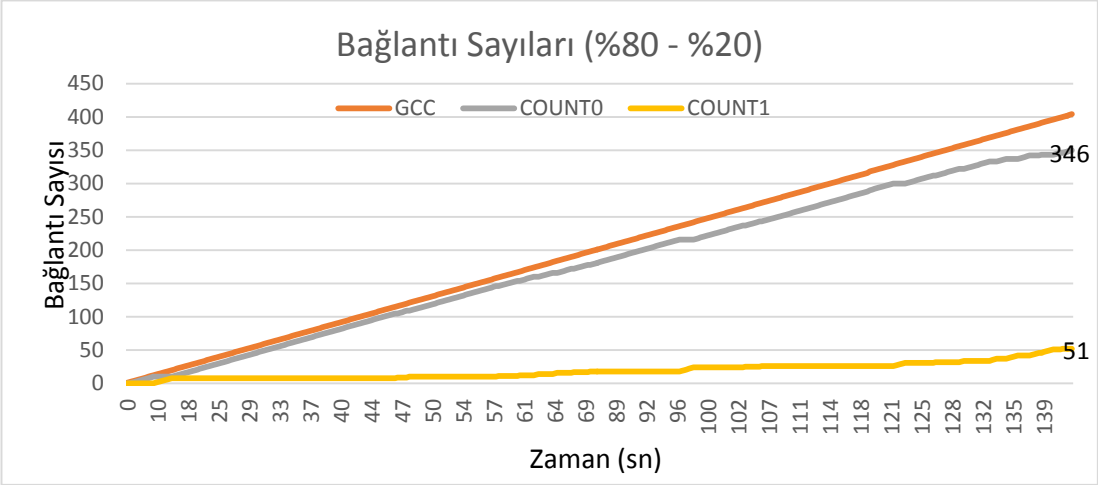
Şekil 4.4: Test 2: %80-%20 Anlık bağlantı sayıları.

Şekil 4.5'da aynı testin her iki ağ arayüzüne de %50 öncelik tanınması durumundaki sonuçları gösterilmiştir. RTT değerlerinin yakın olması bağlantı sayılarının yakın olmasına sebep olmuştur.

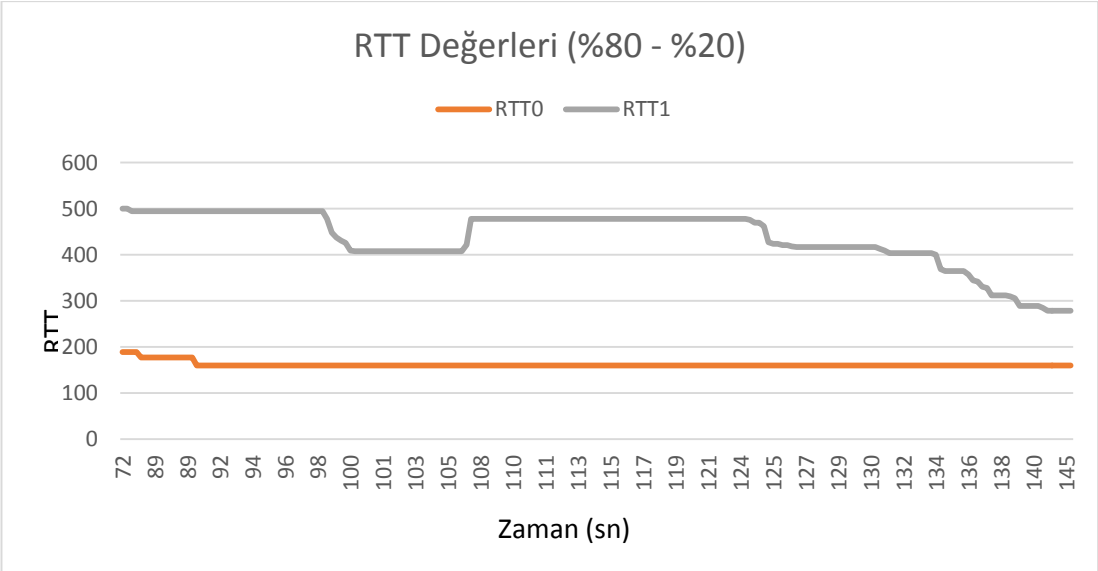


Şekil 4.5: Test 3: %50-%50 Bağlantı sayıları.

3G ağ bağlantısının kablosuz ağa göre daha yavaş olduğu bir noktada kablosuz ağ bağlantısına %80 öncelik, 3G ağ bağlantısına %20 öncelik verilerek test tekrarlanmıştır.



Şekil 4.6: Test 4: %80-%20 Bağlantı sayıları.

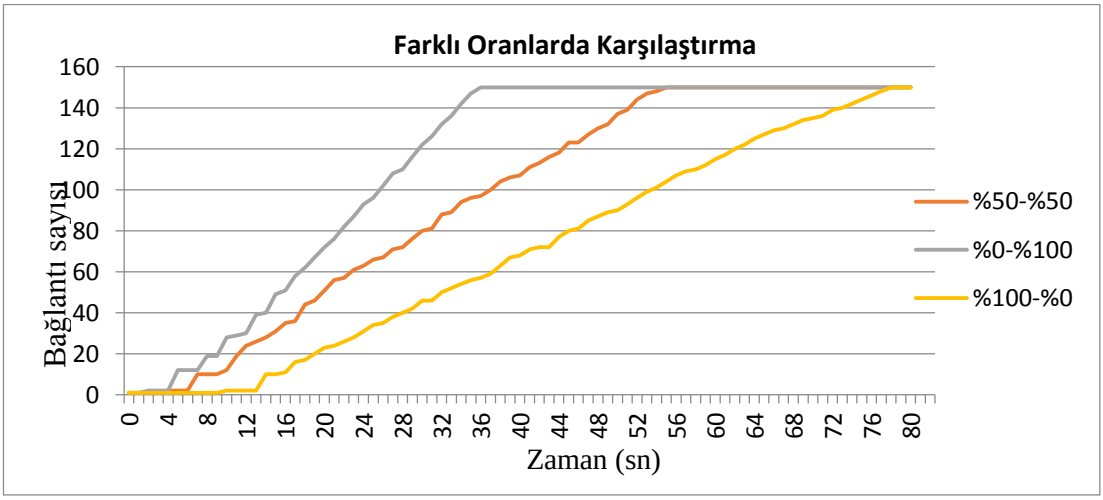


Şekil 4.7: Test 4: %80 - %20 RTT değerleri.

Test 4'e ait RTT değerleri Şekil 4.7'de gösterilmiştir.

Test 5'de aynı test tekrarlanmıştır. Yapılan ilk testte kablosuz ağ bağlantısına %100 öncelik verilmiştir ve test toplam **77** saniye sürmüştür. Öncelik değerleri 3G ağ bağlantısına %100 olarak değiştirildiğinde, toplam test **49** saniye sürmüştür (Test 6). Her iki ağ bağlantısına %50 öncelik verildiğinde ise testin tamamlanması **63** saniye sürmüştür (Test 7).

Test 8, 9 ve 10'da benzer testler yapılmış ve test sonuçları bir araya getirilmiştir. Bu testlerde toplam 150 istek gönderilmiştir.



Şekil 4.8: Test 8,9,10 sonuçları.

Şekil 4.8’de görüldüğü üzere 3G ağ bağlantısının daha hızlı olması durumunda test sonuçla, sadece kablosuz ağ ile yapılan testlerden daha önce bitmektedir.

5. SONUÇLAR ve ÖNERİLER

Kablosuz iletişim teknolojilerinin gelişmesiyle kullanım alanları da kullanıcı sayısı da önemli ölçüde artmıştır. Yeni teknolojilerin geliştirilmesi, yeni araştırma alanlarının doğmasına neden olmuştur. Kablosuz ağ kullanımı ise oldukça geniş bir araştırma alanıdır. Birden fazla ağ arayüzünün aynı anda kullanılması da bu araştırma alanlarının başında gelmektedir.

Yapılan testler, aynı anda farklı ağ arayüzlerinin birlikte kullanılması sonucu istemcinin TCP performansının zaman zaman arttığını göstermiştir. Sadece bir ağ arayüzünün kullanılması durumunda alınan performansın, diğer ağ bağlantılarının da kullanılmaya başlanması ile arttığı gözlemlenmiştir. Özellikle kapsama alanı çok daha geniş olan hücresel bağlantı hızının, kablosuz bağlantı hızına göre daha iyi olması durumunda test sonuçları performansın arttığını göstermiştir. Test sonuçlarına göre hücresel ağ önceliği yüksek verilmişken ve hücresel bağlantı performansın düşük olması sebebiyle, tüm performansın düştüğü gözlemlenmiştir. Ağ arayüzü önceliklerinin düzgün olarak ayarlanması tüm performansın artışını sağlayacaktır.

Önerilen yöntem, 2 farklı ağ arayüzü kullanılarak gerçeklenmiştir. Ancak bu sayı geliştirilen çekirdek modülünde sadece birkaç tanımlama değiştirerek çok daha fazla ağ arayüzü ile de çalışabilir.

Bu yöntemin dezavantajı, ağ bağlantılarının kopması durumunda, o ağ bağlantısını kullanan TCP bağlantılarının da kopmasıdır. Bu durumda uygulama yeniden bağlantı açmaya çalışacak ve bu sefer diğer ağ bağlantısı üzerinden gönderilecektir. HTTP veya HTTPS gibi protokollerde sunucu yazılımlarında her kullanıcının aynı anda sadece bir IP adresinden istekte bulunabileceği ile ilgili çalışmalar yapılmış olabilir. Bu durumda sunucu, istemcinin isteklerine cevap vermeyebilir.

TCP performansı, hücresel ve kablosuz ağ bağlantılarının kayıp oranları, bant genişliği gibi parametrelerden etkilemektedir. Ancak her ağ arayüzüne ayrı ayrı öncelik verilebildiği için performansın optimize edilmesi sağlanabilir. Sistemde çalışacak ek bir servis, ağ arayüzlerindeki paket kayıp oranları, bit hataları, RTT dalgalanmaları veya jitter gibi parametrelere bakarak ağ arayüzlerinin önceliğini otomatik olarak değiştirebilir. Bu durumda, sabit bir önceliğe göre çok daha iyi performans alınabilecektir.

El değiştirme araştırmalarına benzer şekilde önerilen yöntemde de arayüzler için seçilecek olan öncelik değerleri hesaplanırken, RTT, jitter, bit hata oranı, paket kaybı oranı, ağ bağlantılarının enerji tüketimi, alınan sinyal gücü, kullanıcı öncelikleri ve kullanım ücreti parametreleri kullanılabilir.

Önceden tanımlanan politikalar aracılığı ile istenen uygulama katmanı protokolünün, performans ihtiyacına göre hep aynı ağ arayüzünü kullanması sağlanabilir. Örneğin HTTP protokolü her zaman kablosuz ağdan, SMTP protokolü her zaman hücresel ağdan Internet'e bağlanması sağlanabilir.

Benzer şekilde, TCP ye uygulanan yöntem, UDP'ye de uygulanabilir. Ancak UDP, işletim sisteminin *connect* sistem çağrısını kullanmadığı için uygun sistem çağrısının çengellenmesi gerekmektedir. Bu durumda, DNS, SNMP ve VOIP gibi protokollerde de performans artışı sağlanacaktır.

Hücresel ağların, kablosuz ağlara göre daha pahalı olması sebebiyle, ücretlendirmeye göre de öncelik dağıtımı yapılabilir. Ağ arayüzlerine kota konularak belirlenen miktarda veri transferi yapıldıktan sonra bu ağ bağlantısının kullanılmaması sağlanabilir.

KAYNAKLAR

Atiquzzaman M., (2002), SCTP: state of the art in research, products, and technical challenges. 14th International Conference on Ion Implantation Technology Proceedings (IEEE Cat. No.02EX505), 85–91. doi:10.1109/CCW.2003.1240794

Bathich A.A., Baba M. D., Ibrahim M., (2012), "IEEE 802.21 based vertical handover in WiFi and WiMAX networks," Computers & Informatics (ISCI), 2012 IEEE Symposium on , vol., no., pp.140,144, 18-20 March 2012

Dreibholz T., Rathgeb E. P., Rüngeler I., Seggelmann R., Tüxen M., Stewart R. R., (2011), "Stream control transmission protocol: Past, current, and future standardization activities," Communications Magazine, IEEE , vol.49, no.4, pp.82,88, April 2011

Natarajan P., Baker F., Amer P. D., Leighton J. T., (2009), "SCTP: What, Why, and How," Internet Computing, IEEE , vol.13, no.5, pp.81,85, Sept.-Oct. 2009 doi: 10.1109/MIC.2009.114

Ribeiro E., & Leung V., (2006). "Minimum delay path selection in multi-homed systems with path asymmetry", Communications Letters, IEEE, 10(3), 135–137.

Valdovinos, I. A., & Díaz, J. A. P., (2009), "TCP Extension to Send Traffic Simultaneously through Multiple Heterogeneous Network Interfaces", Mexican International Conference on Computer Science, 3, 89–94. doi:10.1109/ENC.2009.28

Zekri M., Pokhrel J., Jouaber B., Zeghlache D., (2011), "Reputation for Vertical Handover decision making," Communications (APCC), 2011 17th Asia-Pacific Conference on , vol., no., pp.318,323, 2-5 Oct. 2011

Web 1, (2013), https://en.wikipedia.org/wiki/Template:Comparison_of_mobile_Internet_standards (Erişim Tarihi: 01/03/2013)

Web 2, (2013), https://www.ietf.org/rfc/rfc1918.txt (Erişim Tarihi: 02/03/2013)

Web 3, (2013), https://www.ietf.org/rfc/rfc3022.txt (Erişim Tarihi: 03/03/2013)

Web 4, (2013), http://technet.microsoft.com/tr-tr/library/cc786900%28v=ws.10%29.aspx (Erişim Tarihi: 04/032013)

Web 5, (2013), http://www.technologyuk.net/the_internet/internet/tcp_ip_stack.shtml (Erişim Tarihi: 05/03/2013)

Web 6, (2013), https://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml (Erişim Tarihi: 06/05/2013)

Web 7, (2013), http://www.freebsd.org (Erişim Tarihi: 07/01/2013)

Web 8, (2013), http://www.freebsd.org/cgi/man.cgi?query=pfil&sektion=9&apropos=0&manpath=FreeBSD+9.1-RELEASE (Erişim Tarihi: 08/05/2013)

Web 9, (2013), man sayfası http://www.freebsd.org/cgi/man.cgi?query=hhook&sektion=9&manpath=FreeBSD+9.0-RELEASE (Erişim Tarihi: 09/01/2013)

Web 10, (2013), http://www.freebsd.org/cgi/man.cgi?query=khelp&sektion=9&manpath=FreeBSD+9.0-RELEASE (Erişim Tarihi: 10/01/2013)

Web 11, (2013), http://caia.swin.edu.au/urp/newtcp/ (Erişim Tarihi: 11/01/2013)

Web 12,(2013), http://www.freebsd.org/releases/8.0R/relnotes-detailed.html (Erişim Tarihi: 12/02/2013)

Web 13, (2013), http://www.freebsd.org/cgi/man.cgi?query=setfib&apropos=0&sektion=0&manpath=FreeBSD+8.0-RELEASE&arch=default&format=html (Erişim Tarihi: 13/01/2013)

Web 14, (2013), http://caia.swin.edu.au/urp/newtcp/tools.html (Erişim Tarihi: 14/01/2013)

Web 15, (2013), http://www.freebsd.org/releases/9.0R/relnotes.html (Erişim Tarihi: 15/02/2013)

Web 16, (2013), http://www.freebsd.org/cgi/man.cgi?query=connect&apropos=0&sektion=0&manpath=FreeBSD+9.1-RELEASE&arch=default&format=html (Erişim Tarihi: 16/01/2013)

Web 17, (2013), http://www.freebsd.org/cgi/man.cgi?query=sysctl&apropos=0&sektion=0&manpath=FreeBSD+9.1-RELEASE&arch=default&format=html (Erişim Tarihi: 17/01/2013)

ÖZGEÇMİŞ

1979 yılında Sivas'ta doğan Necati Ersen ŞİŞECİ, 2000 yılında Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği bölümünden mezun olmuştur. Sistem mühendisliği, yazılım mühendisliği ve sistem yöneticiliği gibi farklı alanlarda çalıştıktan sonra 2004 yılında TÜBİTAK UEKAE'de çalışmaya başlamıştır. 2009 yılında Gebze Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği bölümünde Yüksek Lisans eğitimine başlamıştır. Halen TÜBİTAK Siber Güvenlik Enstitüsünde Siber Güvenlik Uzmanı olarak çalışmaktadır.

EKLER

Ek 1: Tez Çalışması Kapsamında Yapılan Yayınlar

Şişeci N. E., Mantar H. A., (2013) "Mobil Cihazlarda Farklı Ağ Bağlantılarını Kullanarak TCP Performansının Arttırılması", ISITES 2013, Sakarya, Turkiye, 7-9 Haziran