Bilgisayar Ağları Ders Notları

21100011010

Metin Furkan Yaman

İçindekiler

CHAPTER 1 4
Computer Networks and the Internet 5
1.İnternet Nedir? 6
1.1 ISP Nedir? 7
1.2 IETF Nedir? 7
1.3 Distributed Applications Nedir? 7
1.4 Protokol Nedir? 8
1.2 Protokol Nedir? 8
1.3 Network Edge 9
1.3.1 Data Centers 9
2. Access Network Nedir? 10
2.1 Home Access: DSL, Cable, FTTH ve 5G Fixed Wireless 10
2.1.1 DSL Nedir? 10
2.1.2 Cable İnternet Nedir? 12
2.1.3 FTTH Nedir? 13
2.2 Kurumsal (ve Ev) Erişim: Ethernet ve WiFi 13
2.3 Wide-Area Wireless Access: 3G, LTE 4G ve 5G 14
3. Fiziksel Medya 14
3.1 Twisted-Pair Copper Wire 15
3.2 Coaxial Cable 15
3.3 Fiber Optics 16
3.4 Terrestrial Radio Channels 16
3.5 Satellite Radio Channels 17
4. Network Core 18
4.1 Packet Switching 18
4.1.1 Store-and-Forward Transmission 18
4.1.1.1 Queuing Delays and Packet Loss 20
4.1.1.2 Forwarding Tables and Routing Protocols 21
4.2.1 Packet Switching vs Circut Switching 22
4.2.2 Multiplexing in Circuit-Switched Networks 23
4.3 Networks of Networks 24
4.4 Delay, Loss, and Throughput in Packet-Switched Networks 27
4.4.1 Processing Delay 27
4.4.2 Queuing Delay 28
4.4.3 Transmission Delay 28

4.4.4 Propagation Delay....28
4.4.5 Propagation Delay vs Transmission Delay....28
4.4.6 Queuing Delay and Packet Loss....29
4.4.7 End-to-End Delay....30
4.4.8 Ek Not....31
4.4.9 Throughput in Computer Networks....31
5.0 Protocol Layers ve Servis Modelleri....34
5.1 Layered Architecture....34
5.1.2 Protocol Layering....34
5.2 Encapsulation....38
6.0 Networks Under Attack....40
7.0 Network Applications'ın Prensipleri....43
7.1....44
7.1.1 Process Communicating....45
7.1.1.1 Client and Server Processes....45
7.1.2 Transport Services Available to Applications....47
7.1.3 Transport Services Provided by the Internet....49
7.1.4 Services Not Provided by Internet Transport Protocols....51
7.1.5 Application-Layer Protocols....51
7.2 The Web and HTTP....52
7.2.1 Overview of HTTP....52
7.2.2 Non-Persistent and Persistent Connections (Kalıcı Olmayan ve Kalıcı Bağlantılar)....53
7.2.3 HTTP Message Format....56
7.2.4 User-Server Interaction: Cookies....59
7.2.5 Web Caching....61
7.2.6 HTTP/2....66
7.2.7 HTTP/3....68
7.3 Electronic Mail in the Internet....68
7.3.1 SMTP....69
7.3.2 Mail Message Formats....71
2.3.3 Mail Accsess Protocols....72
7.4 DNS—The Internet's Directory Service....73
7.4.1 Services Provided by DNS....73
7.4.2 Overview of How DNS Works....75
7.4.3 DNS Records and Messages....80
7.5 Peer-to-Peer File Distribution....83

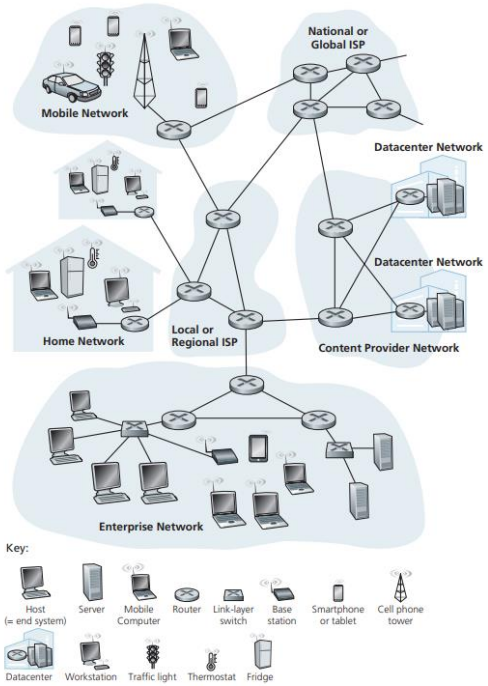
7.6 Video Streaming and Content Distribution Networks 84
7.6.1 Internet Video 84
7.6.2 HTTP Streaming and DASH 84
2.6.3 Content Distribution Networks 85
2.6.4 Socket Programming (Network Applications).....88
3.0 Transport Layer.....92

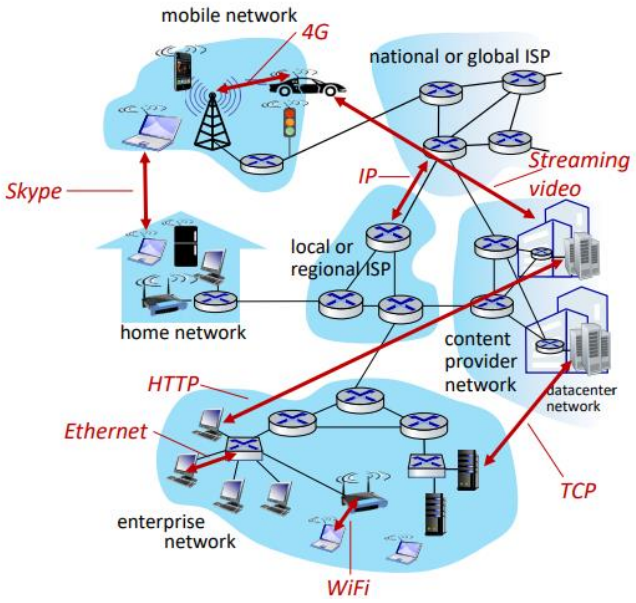
CHAPTER 1

Computer Networks and the Internet

1.İnternet Nedir?

İnternet kısaca ağların ağıdır. Yazılım-donanım ve servis sağlayan hizmet olarak 2'ye ayırabiliriz. Uç sistemler bir communication links ve packet switches ağıyla birbirine bağlanır. Farklı bağlantılar, verileri farklı hızlarda iletebilir ve bir bağlantının transmission rate'i bit/saniye cinsinden ölçülür. Transmission rate saniyede gönderilen bit sayısıdır. Bir uç sistemin başka bir uç sisteme gönderilecek verileri olduğunda, gönderen uç sistem verileri bölümlere ayırır ve her bir bölüme header byte ekler. Çıkan bilgi paketleri, daha sonra ağ üzerinden hedef uç sisteme gönderilir ve burada orijinal veriler halinde yeniden birleştirilir. Packet swtichler: gelen iletişim bağlantılarından birine gelen bir paketi alır ve bu paketi giden iletişim bağlantılarından birine iletir. Packet switchlerin pek çok şekli ve çeşidi vardır, en öne çıkan iki tür, routerlar ve link-layer switchlerdir. Her iki switch de gelen paketleri nihai hedefe yönlendirir ancak bazı farklılıklar vardır. Link layer swtichler genellikle erişim ağlarında kullanılırken, routerlar genellikle network core (ağ çekirdeğinde) kullanılır. Bir paketin gönderen uç sistemden alıcı uç sisteme geçtiği communication links ve packet switchs, ağdaki route veya path olarak bilinir.





1.1 ISP Nedir?

Uç sistemler internete Internet Service Providers (ISP'ler) aracılığıyla erişir. Her ISP kendi içinde packet switchler ve communication linklerden oluşan bir ağdır. İSP'ler, kablolu modem veya DSL gibi konut geniş bant erişimi, yüksek hızlı yerel alan ağı erişimi ve mobil kablosuz erişim dahil olmak üzere, uç sistemlere çeşitli ağ erişimi türleri sağlar. İSP'ler sunucuları doğrudan İnternet'e bağlayarak içerik sağlayıcılara İnternet erişimi sağlar. İnternet tamamen uç sistemleri birbirine bağlamakla ilgilidir, dolayısıyla uç sistemlere erişim sağlayan İSP'lerin de birbirine bağlı olması gerekir. Üst düzey bir ISP, yüksek hızlı fiber optik bağlantılarla birbirine bağlanan yüksek hızlı routerlardan oluşur. Her ISP ağı, üst katman veya alt katman olsun, bağımsız olarak yönetilir, IP protokolünü çalıştırır ve belirli adlandırma ve adresleme kurallarına uyar. Uç sistemleri, packet switchs ve İnternet'teki diğer parçalar, İnternet içinde bilgi gönderme ve alma işlemlerini kontrol eden protokoller çalıştırır. Transmission Control Protocol (TCP) ve Internet Protocol (IP), internet'teki en önemli protokollerden ikisidir. IP protokolü, yönlendiriciler ve end sistemleri arasında gönderilen ve alınan paketlerin formatını belirtir. İnternet'in başlıca protokolleri genellikle TCP/IP olarak adlandırılır.

1.2 IETF Nedir?

İnternet standartları IETF tarafından geliştirilmiştir. IETF standart belgelerine RFC adı verilir. RFC'ler genellikle oldukça teknik ve detaylıdır. Bunlar, TCP, IP, HTTP (Web için) ve SMTP (e-posta için) gibi protokolleri tanımlar.

1.3 Distributed Applications Nedir?

İnternet uygulamaları arasında mobil akıllı telefon ve tablet uygulamaları da bulunur. Bunlar, Internet mesajlaşması, gerçek zamanlı yol trafiği bilgisi ile haritalama, müzik ve film akışı, çevrimiçi sosyal medya, video konferans, çoklu oyunculu oyunlar ve konum tabanlı öneri sistemlerini içerir. Bu uygulamalar, veri alışverişi yapan birden fazla uç sistem içerdiği için distributed applications (dağıtılmış uygulamalar) olarak adlandırılır. Önemli bir nokta

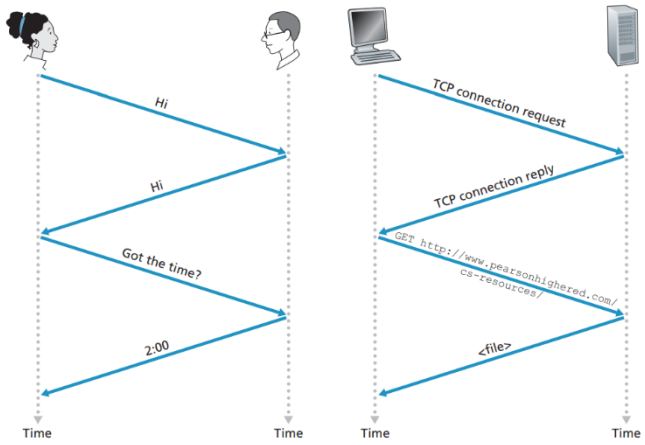
olarak, İnternet uygulamaları uç sistemlerde çalışır- ağ çekirdeğindeki paket anahtarlarında çalışmazlar-. Packet switchler, verinin kaynağı veya hedefi olan uygulama ile ilgilenmezken, veri alışverişini kolaylaştırır.

1.4 Protokol Nedir?

İnternete bağlı olan uç sistemler, bir programın bir uç sistemde çalışan belirli bir hedef programına veri iletilmesini istediğinde internet altyapısına nasıl sorması gerektiğini belirleyen bir socket interface sağlar. Bu İnternet socket interface, gönderen programın veriyi hedef programın alabilmesi için izlemesi gereken kurallar kümesidir. Diyelim ki Alice, posta servisini kullanarak Bob'a bir mektup göndermek istiyor. Elbette, Alice mektubu (veriyi) yazıp penceresinden dışarı bırakamaz. Bunun yerine, posta servisi, Alice'in mektubu bir zarfa koymasını; zarfın ortasına Bob'un tam adını, adresini ve posta kodunu yazmasını, zarfı kapatmasını, zarfın sağ üst köşesine bir pul yapıştırmasını ve son olarak, zarfı resmi bir posta servisi posta kutusuna bırakmasını gerektirir. Böylece, posta servisinin kendi "posta servisi arayüzü" veya kuralları vardır ve Alice'in mektubunun Bob'a ulaştırılması için izlemesi gereken kurallar kümesidir. Benzer şekilde, İnternet'in, veri gönderen programın, veriyi alacak olan programa İnternet'in teslim etmesini sağlamak için izlemesi gereken bir socket interface vardır.

1.2 Protokol Nedir?

İnsanlar farklı protokoller çalıştırıyorsa (örneğin, bir kişinin nezaketi varken diğerinin yoksa, veya birinin zaman kavramını anladığı halde diğerinin anlamadığı durumlar) protokoller uyumlu çalışmaz ve faydalı bir iş yapılamaz. Aynısı ağda da geçerlidir- bir görevi tamamlamak için aynı protokolü çalıştıran iki (veya daha fazla) iletişim kuran varlık gereklidir.



İnternet'de iki veya daha fazla iletişim kuran uzak varlık içeren tüm etkinlikler bir protokol tarafından yönetilir. Örneğin, iki fiziksel olarak bağlı bilgisayarda donanım tarafından uygulanan protokoller, iki ağ arabirim kartı arasındaki "tel (wire)" üzerindeki bit akışını kontrol eder; son sistemlerdeki tıkanıklık kontrol protokolleri (congestion-control protocols), paketlerin

gönderici ile alıcı arasında iletim hızını kontrol eder; routerlardaki protokoller, bir paketin kaynaktan hedefe giden pathi belirler.

Bir protokol, iki veya daha fazla iletişim kuran varlık arasında alınıp verilen mesajların formatını ve sırasını, ayrıca bir mesajın veya başka bir olayın iletilmesi ve/veya alınması üzerine gerçekleştirilen eylemleri tanımlar.

1.3 Network Edge

Son sistemler barındırdıkları (yani, çalıştırdıkları) uygulama programları nedeniyle "ana bilgisayarlar (host)" olarak da adlandırılır. Bu derste end edge= host gibi düşünebiliriz ancak Ana bilgisayarlar bazen daha fazla iki kategoriye ayrılır: istemciler (clients) ve sunucular (servers). Gayri resmi olarak, istemciler genellikle masaüstü bilgisayarlar, dizüstü bilgisayarlar, akıllı telefonlar vb. olurken, sunucular genellikle web sayfalarını depolayan ve dağıtan daha güçlü makineler olur, video akışı yapar, e-postayı aktarır vb.

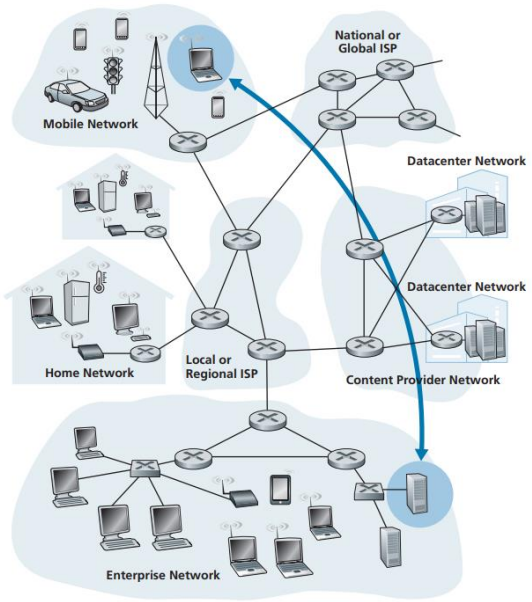


Figure 1.3 ♦ End-system interaction

1.3.1 Data Centers

Genel olarak, veri merkezleri üç amaçla hizmet verir, bunları somutlaştırmak için Amazon bağlamında burada açıklıyoruz. İlk olarak, kullanıcılara Amazon e-ticaret sayfalarını sunarlar,

örneğin, ürünleri ve satın alma bilgilerini açıklayan sayfalar. İkinci olarak, Amazon özel veri işleme görevleri için büyük ölçekli paralel hesaplama altyapısı olarak hizmet ederler. Üçüncü olarak, diğer şirketlere bulut bilişim hizmeti sağlarlar. Bir veri merkezindeki işçi arılar (bees), ana bilgisayarlardır. İçerik sunarlar (örneğin, Web sayfaları ve videolar), e-posta ve belgeleri depolarlar ve toplu olarak dağıtılmış hesaplamaları gerçekleştirirler. Veri merkezindeki ana bilgisayarlar, bıçaklar (blades) olarak adlandırılır ve pizza kutularını andırır. Genellikle CPU, bellek ve disk depolama içeren standart ana bilgisayarlar içerirler. Ana bilgisayarlar raflarda istiflenir ve her raf genellikle 20 ila 40 bıçak içerir. Raflar, sofistike ve gelişen veri merkezi ağ tasarımları kullanılarak birbirine bağlanır.

2. Access Network Nedir?

Access Network, bir uç sistemi, diğer herhangi bir uzak uç sisteme giden path üzerindeki ilk routera (ayrıca 'uç yönlendirici' olarak da bilinir) fiziksel olarak bağlayan ağdır. Şekil 1.4, kalın, gölgeli çizgilerle gösterilen çeşitli erişim ağlarını ve bunların ev, kurumsal ve geniş alan mobil kablosuz gibi kullanıldığı ortamları göstermektedir.

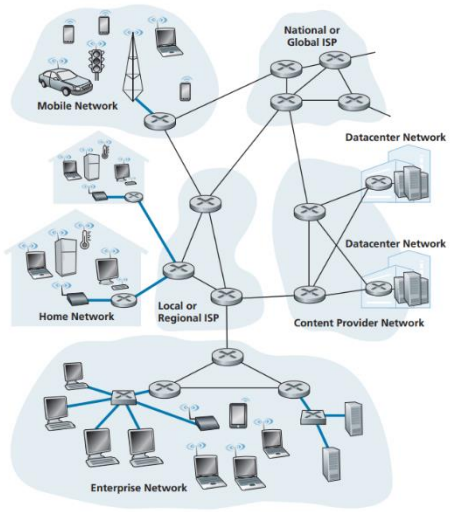


Figure 1.4 ♦ Access networks

2.1 Home Access: DSL, Cable, FTTH ve 5G Fixed Wireless

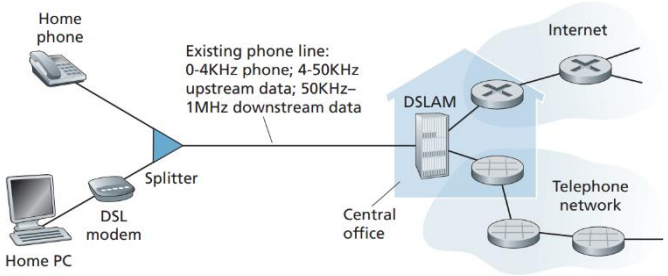
2.1.1 DSL Nedir?

Geniş bantlı konut erişiminin en yaygın iki türü digital subscriber line (DSL) ve cabledır. Bir konut genellikle kablolu yerel telefon erişimini sağlayan yerel telefon şirketi (telco) tarafından DSL İnternet erişimi sağlar. Bu nedenle, DSL kullanıldığında, bir müşterinin telco'su aynı zamanda İSP'sidir. Şekil 1.5'te gösterildiği gibi, her müşterinin DSL modemi, telco'nun yerel merkez ofisinde (local central Office) (CO) bulunan bir digital subscriber line access

multiplexer (DSLAM) ile veri alışverişi için mevcut telefon hattını kullanır. Evdeki DSL modemi, dijital verileri alır ve bunları CO'ya telefon telleri üzerinden iletim için yüksek frekanslı tonlara çevirir; birçok evin analog sinyalleri, DSLAM'de tekrar dijital formata çevrilir.

Konut telefon hattı hem veri hem de geleneksel telefon sinyallerini aynı anda taşır ve bunlar farklı frekanslarda kodlanır:

- Yüksek hızlı akış yönlü kanal, 50 kHz ile 1 MHz aralığında
- Orta hızlı yük akış yönlü kanal, 4 kHz ile 50 kHz aralığında
- Sıradan iki yönlü telefon kanalı, 0 ile 4 kHz aralığında



Bu yaklaşım, tek bir DSL bağlantısının üç ayrı bağlantı gibi görünmesini sağlar, böylece bir telefon görüşmesi ve bir İnternet bağlantısı aynı anda DSL bağlantısını paylaşabilir. Müşteri tarafında, bir splitter, eve gelen veri ve telefon sinyallerini ayırır ve veri sinyalini DSL modeme iletilir. Telco tarafında, CO'da, DSLAM, veri ve telefon sinyallerini ayırır ve veriyi İnternet'e gönderir. Yüzlerce hatta binlerce ev, tek bir DSLAM'a bağlanır.

DSL standartları, 24 Mbs ve 52 Mbs gibi downstream iletim hızlarını ve 3.5 Mbps ve 16 Mbps gibi upstream hızlarını içeren birden fazla iletim hızını tanımlar; en yeni standart, 1 Gbps'lik toplam upstream artı downstream hızları sağlar. Downstream ve upstream hızları farklı olduğu için, erişimin asimetrik olduğu söylenir. Gerçek downstream ve upstream iletim hızları, DSL sağlayıcısı konut hızını kademeli hizmet sunduğunda (farklı hızlar, farklı fiyatlarla sunulan) yukarıda belirtilen hızlardan daha az olabilir. Maksimum hız, ev ile CO arasındaki mesafe, çift hat hattının kalınlığı ve elektriksel girişim derecesi tarafından da sınırlıdır.

DSL, telco'nun mevcut yerel telefon altyapısını kullanırken, cable internet erişimi kablo televizyon şirketinin mevcut kablo televizyon altyapısını kullanır. Bir konut, kablo televizyonunu sağlayan aynı şirketten kablo internet erişimi sağlar. Şekil 1.6'da gösterildiği gibi, fiber optikler kablo baş ucuyla mahalle seviyesindeki kavşakları birbirine bağlar, bu kavşaklardan geleneksel coaxial kablolar kullanılarak bireysel evlere ve dairelere ulaşılır. Her

mahalle kavşağı genellikle 500 ila 5.000 evi destekler. Bu sistemde hem fiber hem de coaxial kablo kullanıldığından, sıklıkla hibrit fiber coaxial (HFC) olarak adlandırılır.

2.1.2 Cable İnternet Nedir?

Kablo İnternet erişimi özel modemler gerektirir, buna kablo modemleri denir. Bir DSL modemine benzer şekilde, kablo modem genellikle harici bir cihazdır ve ev PC'sine bir Ethernet bağlantı noktası aracılığıyla bağlanır. Kablonun baş ucu, cable modem termination system (CMTS), DSL ağının DSLAM'ı ile benzer bir işlev görür- birçok alt akımdaki evdeki kablo modemlerden gönderilen analog sinyali dijital formata geri dönüştürür. Kablo modemleri, HFC ağını downstream ve upstream kanallarına böler-. DSL gibi, erişim genellikle asimetrik olup, downstream kanal genellikle upstream kanaldan daha yüksek bir iletim hızına sahiptir. Sırasıyla DOCSIS 2.0 ve 3.0 standartları, downstream bit hızlarını sırasıyla 40 Mbps ve 1.2 Gbps ve upstream hızlarını 30 Mbps ve 100 Mbps olarak tanımlar. DSL ağları durumunda olduğu gibi, maksimum elde edilebilir hız, daha düşük sözleşmeli veri hızları veya ortam bozulmaları nedeniyle gerçekleşmeyebilir.

Cable internet erişiminin önemli bir özelliği, bunun paylaşımlı bir yayın ortamı olmasıdır. Özellikle, ana merkez tarafından gönderilen her paket, tüm bağlantılarda downstream her eve gider ve bir ev tarafından gönderilen her paket, upstream kanalında ana merkeze gider. Bu nedenle, birkaç kullanıcı aynı anda downstream kanalında bir video dosyası indiriyorsa, her kullanıcının video dosyasını aldığı gerçek hız, toplam cable downstream önemli ölçüde daha düşük olacaktır. Öte yandan, yalnızca birkaç aktif kullanıcı varsa ve bunların hepsi web'de gezinme yapıyorsa, kullanıcıların neredeyse aynı anda bir web sayfası talep etme olasılığı düşük olduğundan, kullanıcıların her biri gerçekten tam cable downstream web sayfaları alabilir. Upstream de paylaşıldığı için, iletileri koordine etmek ve çarpışmaları önlemek için distributed multiple access protocole (dağıtık bir çoklu erişim protokolüne) ihtiyaç vardır.

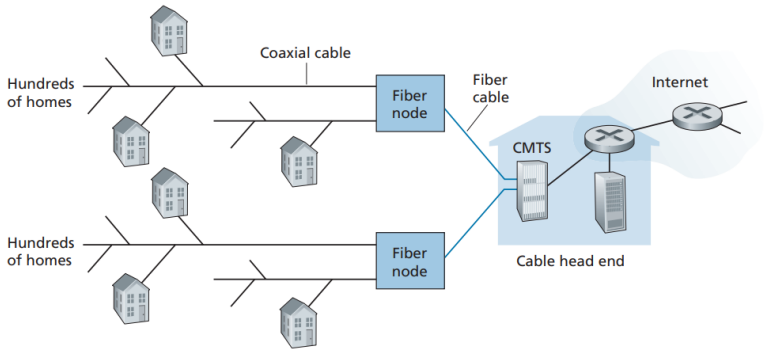


Figure 1.6 ♦ A hybrid fiber-coaxial access network

2.1.3 FTTH Nedir?

FTTH kavramı basittir CO'dan doğrudan eve optik fiber bir yol sağlamaktır. FTTH'de, teorik olarak gigabitler per saniye aralığında internet erişim hızları sağlayabilir. CO'dan evlere optik dağıtım için birkaç farklı teknoloji bulunmaktadır. En basit optik dağıtım ağı direct fiber olarak adlandırılır ve her ev için bir fiber CO'dan çıkar. Daha yaygın olarak, merkezi ofisten çıkan her fiber aslında birçok eve paylaşılmaktadır; fiber evlere oldukça yaklaştığında, fiberler müşteriye özel ayrı fiberlere bölünür. Bölünme işlemini gerçekleştiren iki farklı optik dağıtım ağı mimarisi bulunmaktadır: active optical netwrokd (AON'lar) ve passive optical networks (PON'lar). AON, temelde anahtarlanmış Ethernet'tir.

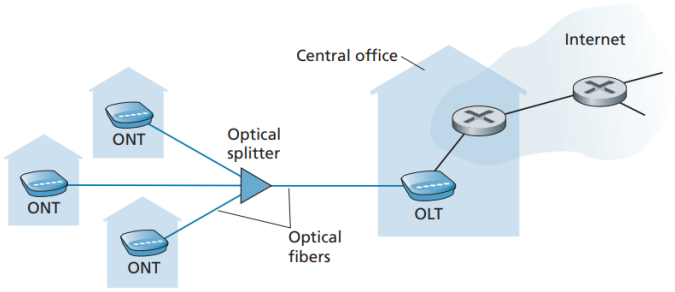


Figure 1.7 ♦ FTTH Internet access

Şekil 1.7, PON dağıtım mimarisini kullanarak FTTH'yi göstermektedir. Her evde bir optic network terminator (ONT) bulunur ve bu özel optik fiberle mahalle splitterına bağlanır. Splitter, birkaç evi (genellikle 100'den az) tek bir, paylaşılan optik fiber üzerine birleştirir ve bu fiber, telco'nun CO'sundaki bir optic line terminatore (OLT) bağlanır. Optik ve elektrik sinyalleri arasındaki dönüşümü sağlayan OLT, bir telco routeri aracılığıyla İnternet'e bağlanır. Evde, kullanıcılar bir ev router ile (genellikle bir kablosuz yönlendirici) ONT'ye bağlanırlar ve nternet'e bu ev routerı aracılığıyla erişir. PON mimarisinde, OLT'den splitter'a gönderilen tüm paketler splitter'da çoğaltılır (kablo baş ucuyla benzer şekilde).

2.2 Kurumsal (ve Ev) Erişim: Ethernet ve WiFi

Bir local area network (LAN) bir uç sistemi (edge system) edge routera bağlamak için kullanılır. Şekil 1.8'de gösterildiği gibi, Ethernet kullanıcıları, Ethernet anahtarıyla bağlantı kurmak için örgü çift bakır tel kullanır. Ethernet anahtarı (switchs) veya bu tür birbirine bağlı anahtarların ağı daha büyük internet'e bağlanır.

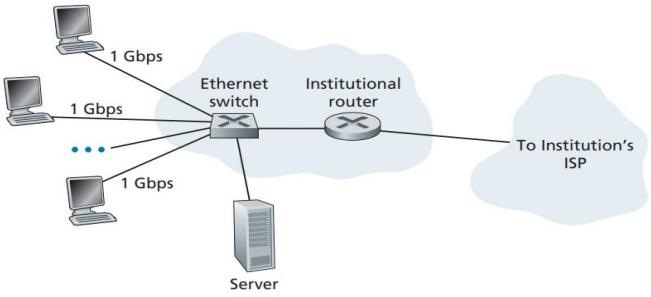


Figure 1.8 ♦ Ethernet Internet access

Kablosuz bir LAN ortamında, kablosuz kullanıcılar erişim noktasına paket gönderir/alır ve bu access point (erişim noktası) kuruluşun ağına bağlıdır (muhtemelen kablolu ethernet kullanılarak), bu ağ da kablolu internet'e bağlıdır.

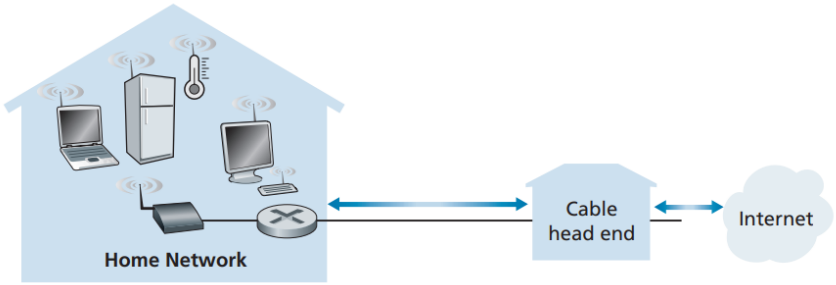


Figure 1.9 ♦ A typical home network

2.3 Wide-Area Wireless Access: 3G, LTE 4G ve 5G

Akıllı telefonlar, hücresel telefonculuk için kullanılan aynı kablosuz altyapıyı kullanarak, bir hücresel ağ sağlayıcısı tarafından işletilen bir baz istasyonu aracılığıyla paket gönderme/alma işlemleri gerçekleştirir. WiFi'nin aksine, bir kullanıcının sadece baz istasyonuna birkaç on kilometre (birkaç on metreden farklı olarak) yakınlıkta olması yeterlidir.

3. Fiziksel Medya

Bir bit, bir uç sistemden, bir dizi bağlantı ve yönlendiriciden geçerek başka bir uç sistemine doğru seyahat ederken birçok kez iletilir ve dolaştırılır. Kaynak uç sistem önce biti ileterek, ardından serideki ilk router biti alır; ilk router, biti iletir ve kısa bir süre sonra ikinci router, biti

alır ve böyle devam eder. Bu nedenle, kaynaktan hedefe seyahat ederken, bit bir dizi verici-alıcı çiftinden geçer. Her verici-alıcı çifti için, bit, elektromanyetik dalgaların veya optik darbelerin bir fiziksel ortam boyunca yayılması yoluyla gönderilir. Fiziksel ortam, birçok şekil ve form alabilir ve yol boyunca her verici-alıcı çifti için aynı türde olmak zorunda değildir. Fiziksel ortamların örnekleri arasında:

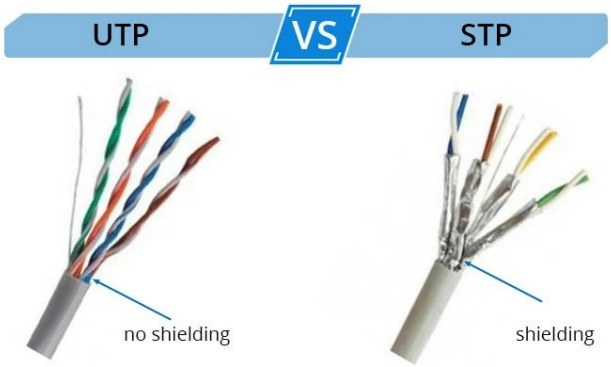
- twisted-pair copper wire (örgü tel bakır tel)
- coaxial cable (koaksiyel kablo)
- multimode fiber-optic cable (çok modlu fiber optik kablo)
- terrestrial radio spectrum (yeryüzü radyo spektrumu)
- satellite radio spectrum (uydu radyo spektrumu) bulunur.

Fiziksel ortamlar iki kategoriye ayrılır: guided media (yönlendirilmiş ortam) ve unguided media (yönlendirilmemiş ortam). Yönlendirilmiş ortamda, dalgalar katı bir ortam boyunca yönlendirilir,örneğin fiber optik kablo, örgü tel bakır tel veya koaksiyel kablo gibi. Yönlendirilmemiş ortamda, dalgalar atmosfer ve uzayda yayılır, örneğin kablosuz bir LAN veya bir dijital uydu kanalında olduğu gibi.

Fiziksel bağlantının (bakır kablo, fiber optik kablo vb.) gerçek maliyeti, diğer ağ maliyetleriyle karşılaştırıldığında genellikle nispeten azdır.

3.1 Twisted-Pair Copper Wire

En ucuz ve en yaygın kullanılan yönlendirilmiş iletim ortamı örgü tel bakır teldir. Telefon ağları tarafından uzun zamandır kullanımdadır. Tellikler, yan yana benzer tellerden elektriksel parazitleri azaltmak için birlikte sarılır. Genellikle, birkaç çift, çiftleri koruyucu bir kalkanla sarmak suretiyle bir kabloda bir araya getirilir. Bir tel çifti, tek bir iletişim bağlantısını oluşturur. Korumasız örgü tel (UTP), bir binadaki bilgisayar ağları için yaygın olarak kullanılır, yani LAN'lar için kullanılır. Ulaşılabilecek veri hızları, telin kalınlığına ve verici ile alıcı arasındaki mesafeye bağlıdır.



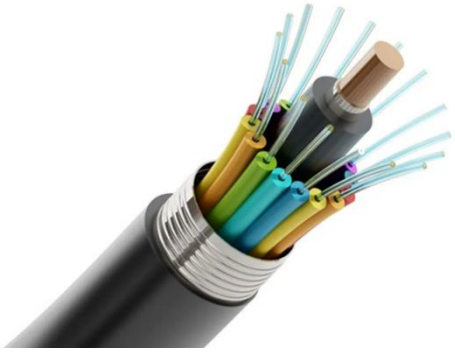
3.2 Coaxial Cable

Örgü tel gibi, koaksiyel kablo da iki bakır ileticiden oluşur, ancak bu ileticiler birbirine paralel değil, konsantrik bir yapıya sahiptir. Bu yapı ve özel izolasyon ve kalkanlama ile, koaksiyel kablo yüksek veri iletim hızlarına ulaşabilir. Koaksiyel kablo, kablo televizyon sistemlerinde

oldukça yaygındır. Kablo televizyonu ve kablo İnternet erişiminde, verici dijital sinyali belirli bir frekans bandına kaydırır ve sonuç olarak analog sinyal vericiden bir veya daha fazla alıcıya gönderilir. Koaksiyel kablo, yönlendirilmiş paylaşılan bir ortam (guided shared medium) olarak kullanılabilir. Özellikle, bir dizi uç sistem doğrudan kabloya bağlanabilir ve her bir uç sistem diğer uç sistemler tarafından gönderilen her şeyi alır.

3.3 Fiber Optics

Bir optik fiber, her biri bir biti temsil eden ışık darbelerini iletirken ince, esnek bir ortamdır. Tek bir optik fiber, saniyede onlarca hatta yüzlerce gigabitlik muazzam bit hızlarını destekleyebilir.. Elektromanyetik parazitlere karşı bağışıklıdır, 100 kilometreye kadar çok düşük sinyal zayıflamasına sahiptir (optik fiberin sinyalin iletimi sırasında çok az sinyal güç kaybına uğradığını ifade eder. Bu özellik, optik fiberin uzun mesafelerde güvenilir bir şekilde veri iletimi için ideal bir seçim olmasını sağlar.) Ve çok zor tespit edilir (Bu, fiber optik kablolardan bilgi çalmanın zor olduğunu belirtir.). Bu özellikler, özellikle deniz aşırı bağlantılar için fiber optiği tercih edilen uzun menzilli yönlendirilmiş iletim ortamı haline getirmiştir. Fiber optik ayrıca İnternet'in omurgasında yaygındır. Ancak, optik cihazların- vericiler, alıcılar ve anahtarlar gibi -yüksek maliyeti, bunların LAN veya konut erişim ağındaki gibi kısa mesafe taşımı için kullanımını engellemiştir.



3.4 Terrestrial Radio Channels

Radyo kanalları elektromanyetik spektrumda sinyalleri iletir. Fiziksel tel kurulumu gerektirmezler, duvarları geçebilir, hareketli bir kullanıcıya bağlantı sağlayabilir ve potansiyel olarak sinyali uzun mesafelere taşıyabilirler. Bir radyo kanalının özellikleri, sinyalin taşınacağı ortamın yayılma ortamı ve taşınacak mesafenin büyük ölçüde belirler. Çevresel faktörleri (propagation environment), yol kaybı, gölgeleme kaybı (sinyalin bir mesafe boyunca ve

engelleyici nesneler etrafından/dan geçerken sinyal gücünün azalması), çoklu yoldan gelen kaybı (sinyalin engelleyici nesnelerden yansıması nedeniyle) ve girişimi (diğer iletimler ve elektromanyetik sinyaller nedeniyle) belirler. Kara radyo kanalları genel olarak üç gruba ayrılabilir:

- Çok kısa mesafede çalışanlar, bir veya iki metre (Kablosuz kulaklıklar, klavyeler ve tıbbi cihazlar)
- Yerel alanlarda çalışanlar, genellikle on ila birkaç yüz metre arasında yayılanlar. (Kablosuz LAN teknolojileri)
- Geniş alanlarda, onlarca kilometreyi kapsayanlar. (hücresel erişim teknolojileri)

3.5 Satellite Radio Channels

Bir iletişim uydusu, iki veya daha fazla Dünya merkezli mikrodalga verici/alıcıyı, yani karasal istasyonları, birbirine bağlar. Uydu, bir frekans bandında iletimleri alır, sinyali bir tekrarlayıcı kullanarak yeniden oluşturur ve sinyali başka bir frekansta iletim yapar. İletişimde iki tür uydu kullanılmaktadır:

1. Jeostatik uydular: Jeostatik uydular, sürekli olarak Dünya'nın aynı noktasının üzerinde kalır. Karasal istasyondan uydudan karasal istasyona olan bu büyük mesafe, 280 milisaniyelik önemli bir sinyal yayılma gecikmesiyle sonuçlanır. Bununla birlikte, yüzlerce Mbps hızında çalışabilen uydu bağlantıları, DSL veya kablo tabanlı İnternet erişimine sahip olmayan bölgelerde sıkça kullanılır.
2. Düşük yörüngeli (LEO) uydular: LEO (Low-Earth Orbit) uyduları Dünya'ya çok daha yakın bir konuma yerleştirilir ve Dünya'nın bir noktasının üzerinde sürekli olarak kalmazlar. (Ay'ın yaptığı gibi) Dünya etrafında dönerler ve birbirleriyle iletişim kurabilirler, aynı zamanda karasal istasyonlarla da iletişim kurabilirler. Bir bölgeye sürekli kapsama sağlamak için birçok uydunun yörüngeye yerleştirilmesi gerekir.

4. Network Core

İnternet'in uç sistemlerini birbirine bağlayan paket anahtarları ve bağlantıların (links) örgüsüdür. Şekil 1.10, ağı kalın gölgeli çizgilerle vurgulayarak ağın çekirdeğini (network core) göstermektedir.

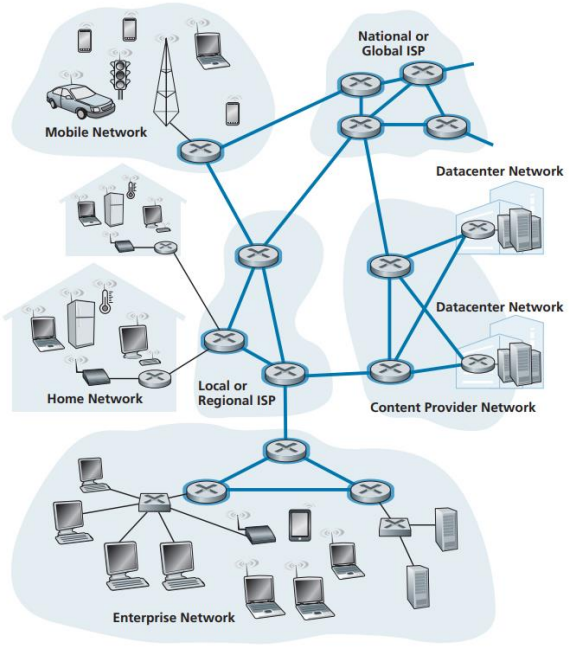


Figure 1.10 ♦ The network core

4.1 Packet Switching

Bir ağ uygulamasında, uç sistemler birbirleriyle iletişim kurarlar. Mesajlar, uygulama tasarımcısının istediği herhangi bir şeyi içerebilir. Mesajlar bir kontrol işlevi, e-posta mesajı, bir JPEG görüntüsü veya bir MP3 ses dosyası gibi verileri içerebilir. Bir kaynak uç sisteminden bir hedef uç sistemine bir mesaj göndermek için, kaynak uzun mesajları paketler adı verilen daha küçük veri parçalarına böler. Kaynak ve hedef arasında, her paket iletişim bağlantıları ve paket anahtarları (routers ve link layer switchs olarak iki türü bulunmaktadır) üzerinden seyahat eder. Bir kaynak uç sistem veya bir paket anahtarı, R bit/sn iletim hızına sahip bir bağlantı üzerinden L bitlik bir paket gönderiyorsa, paketi iletmek için geçen süre L/R saniyedir.

4.1.1 Store-and-Forward Transmission

Çoğu packet switchs store-and-foward transmission yöntemini kullanır. store-and-foward transmission yöntemini, packet switchlerin çıkış bağlantısına ilk biti aktarmaya başlamadan önce tüm paketi alması gerektiği anlamına gelir. Bu yöntemi anlamak için, iki uç sistemi arasında bir tek yönlü bağlantı ile bağlı basit bir ağ düşünelim (Şekil 1.11).

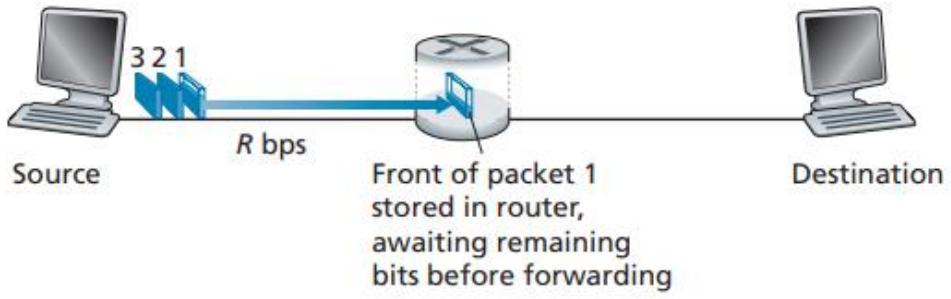


Figure 1.11 ♦ Store-and-forward packet switching

Bir yönlendiricinin genellikle birçok giriş bağlantısı olacaktır, çünkü görevi gelen bir paketi çıkış bağlantısına geçirmektir; bu basit örnekte, yönlendiricinin bir (giriş) bağlantıdan diğerine tek bir paketi aktarması oldukça basit bir görevdir. Bu örnekte, kaynağın göndermek için üç paketi var, her biri L bitlik. Şekil 1.11'de gösterilen zaman anında, kaynak paket 1'in bazılarını iletti ve paket 1'in ön kısmı zaten yönlendiricide ulaştı. Yönlendirici mağaza ve ileri iletim kullandığı için, bu anlık zaman diliminde, yönlendirici aldığı bitleri iletemez bunun yerine, önce paketin bitlerini önbelleğe ("store") almalıdır. Yönlendirici tüm paketin bitlerini aldıktan sonra ("forward") paketi çıkış bağlantısına aktarabilir. Kaynak zamanında 0'da iletimi başlatır; L/R saniye sonra, kaynak tüm paketi iletti ve tüm paket yönlendirici tarafından alındı ve depolandı. L/R saniye sonra, yönlendirici tüm paketi yeni aldığı için paketi çıkış bağlantısına doğru iletmeye başlayabilir; 2L/R'de, yönlendirici tüm paketi iletti ve tüm paket varış noktası tarafından alındı. Dolayısıyla, toplam gecikme 2L/R'dir. Anahtar, bitleri tüm paketi almadan önce (önce tam paketi almadan) ilettiği takdirde, toplam gecikme L/R olurdu çünkü bitler yönlendiricide bekletilmez.

Şimdi, kaynağın ilk paketi göndermeye başladığı andan, hedefin üç paketi de almış olana kadar geçen süreyi hesaplayalım. Daha önce olduğu gibi, L/R zamanında, yönlendirici ilk paketi iletmeye başlar. Ancak aynı zamanda, L/R zamanında kaynak ikinci paketi göndermeye başlayacak, çünkü tam ilk paketi göndermeyi bitirmiştir. Dolayısıyla, 2L/R zamanında, hedef ilk paketi almış ve yönlendirici ikinci paketi almıştır. Benzer şekilde, 3L/R zamanında, hedef ilk iki paketi almış ve yönlendirici üçüncü paketi almıştır. Son olarak, 4L/R zamanında hedef tüm üç paketi almış olur. Kaynaktan hedefe her biri R hızındaki N bağlantıdan (links) oluşan bir yol üzerinden bir paket gönderme durumunda kaynak ile hedef arasında N-1 yönlendirici bulunmaktadır. Tüm bu mantıklardan yola çıkarak end-to-end delay aşağıdaki formül ile hesaplanır.

$$d_{\text{end-to-end}} = N\frac{L}{R}$$

N tane inputum var ve

Swtich (forwarding) hızı > N * input hızı ise

durumunda bekleme olmaz.

4.1.1.1 Queuing Delays and Packet Loss

Her paket anahtarı, bağlı olduğu her bağlantı(kink) için bir output buffera (aynı zamanda bir output queue olarak da adlandırılır) sahiptir. Bu çıkış bufferları, routerların bağlantıya göndereceği paketleri depolar. Output bufferlar, paket anahtarlamada önemli bir rol oynar. Gelen paketin bir bağlantıya iletilmesi gerekiyorsa ve bağlantı başka bir paketin iletimi ile meşgulse, gelen paket çıkış tamponunda beklemek zorunda kalır. Dolayısıyla, store-and-foward gecikmelerine ek olarak, paketler output buffer kuyruğunda gecikmeler yaşarlar. Bu gecikmeler değişkendir ve ağdaki tıkanıklık seviyesine bağlıdır. Buffer alanının miktarı sınırlı olduğu için, gelen bir paket, başka paketlerin iletimi için bekleyen bufferı tamamen dolu bulabilir. Bu durumda, paket kaybı veya ya gelen paket ya da zaten kuyrukta bekleyen paketlerden biri düşürülecektir.

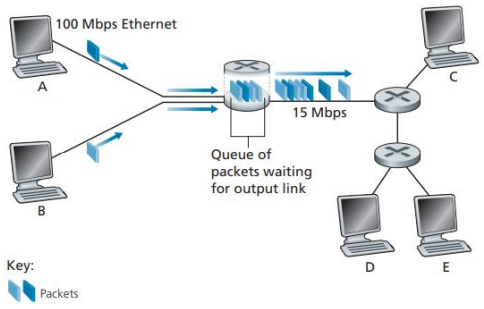


Figure 1.12 ♦ Packet switching

Şekil 1.12, basit bir packet-switched networkü göstermektedir. Şekil 1.11'de olduğu gibi, paketler üç boyutlu plakalarla temsil edilir. Bir plakanın genişliği, paketteki bit sayısını temsil eder. Bu şekilde, tüm paketler aynı genişliğe ve dolayısıyla aynı uzunluğa sahiptir. A ve B Ana Makineleri'nin paketleri E Ana Makinesi'ne gönderdiğini varsayalım. Ana Bilgisayarlar A ve B, önceki şekilde olduğu gibi, paketlerini 100 Mbps Ethernet bağlantıları üzerinden ilk routera gönderirler. Router daha sonra bu paketleri 15 Mbps bağlantıya yönlendirir. Kısa bir zaman aralığında, routera gelen paketlerin (saniyedeki bit cinsinden dönüştürüldüğünde) geliş hızı 15 Mbps'yi aşıyorsa, paketlerin bağlantı üzerine iletilmeden önce bağlantının output bufferda sıraya girmesi nedeniyle routerda tıkanıklık oluşur. Örneğin, A ve B Ana Bilgisayarları aynı anda arka arkaya beş paket gönderirse, bu paketlerin çoğu bir süre bufferda beklemek zorunda kalır.

4.1.1.2 Forwarding Tables and Routing Protocols

İnternet'te, her uç sistem, bir IP adresi olarak adlandırılan bir adrese sahiptir. Bir kaynak uç sistemi, bir paketi bir hedef uç sistemine göndermek istediğinde, kaynak paketin başlığına hedefin IP adresini ekler. Posta adresleri gibi, bu adresin de hiyerarşik bir yapısı vardır. Bir paket bir ağdaki bir yönlendiriciye ulaştığında, yönlendirici paketin hedef adresinin bir kısmını inceleyerek paketi bir yanındaki yönlendiriciye iletiyor. Daha spesifik olarak, her yönlendiricinin, hedef adresleri (veya hedef adreslerinin bir kısmı) ile bu yönlendiricinin çıkış bağlantıları arasında bir eşleştirme yapan bir yönlendirme tablosu vardır. Bir paket bir yönlendiriciye ulaştığında, yönlendirici adresi inceleyip bu hedef adresi kullanarak yönlendirme tablosunu arar ve uygun çıkış bağlantısını bulur. Daha sonra yönlendirici, paketi bu çıkış bağlantısına yönlendirir.

Forwarding: Diğer adıyla switching. Local anlamda hangi çıkışı seçmem gerektiğini gösterir. Yönlendirme, router tablolar ile yapılır. Gelen paketleri yönlendiricinin giriş bağlantısından uygun yönlendirici çıkış bağlantısına taşır.

Routing: Store ve forward eder. Nereden gidileceğini belirler. Bunu en kısa yol veya ağın yoğunluğuna göre belirler. Paketler tarafından alınan kaynak-hedef yollarını belirle. Routing algoritmaları.

Routera gelen noktaya inputlink veya inbound denir. Routerın çıkış noktasına ise outputlink veya outbound denir.

4.2 Circut Switching

Circut switching ağlarda, iletişim uç sistemler arasında gerçekleşirken kullanılan yollar boyunca gereken kaynaklar (arabellekler, link transmission rate), iletişim oturumu süresince rezerve edilir. Packet switnching ağlarda ise bu kaynaklar rezerve edilmez; bir oturumun iletileri, gerektiğinde kaynakları kullanır ve sonuç olarak, iletişim bağlantısına erişim için beklemek gerekebilir. Geleneksel telefon ağları, Circut switching ağların örnekleridir.

Ağ devreyi kurduğunda, aynı zamanda ağın bağlantının süresi boyunca (her bağlantının iletim kapasitesinin bir kısmını temsil eden) sabit bir iletim hızınıda (constant transmission rate) rezerve eder. Belirli bir iletim hızı bu gönderici-alıcı bağlantısı için ayrıldığından, gönderici verileri garanti edilen sabit hızda alıcıya iletebilir.

Şekil 1.13'teki, ağda, dört devre anahtarı dört bağlantı ile birbirine bağlanmıştır. Bu bağlantılardan her birinde dört devre bulunur, böylece her bağlantı dört eşzamanlı bağlantıyı destekleyebilir. Her bir bağlantının dört devresi olduğundan, uçtan uca bağlantı tarafından kullanılan her bağlantı için bağlantı, bağlantının süresi boyunca bağlantının toplam iletim kapasitesinin dörtte birini alır. Dolayısıyla, örneğin, bitişik anahtarlar arasındaki her bağlantının 1 Mbps iletim hızına sahip olduğunu varsayarsak, her bir uçtan uca devre anahtarlamalı bağlantı 250 kbps'lik özel bir iletim hızı alır.

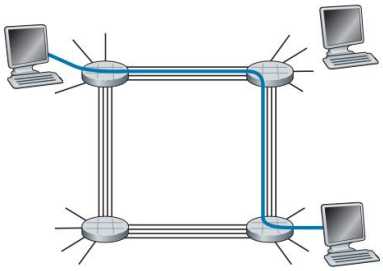


Figure 1.13 ♦ A simple circuit-switched network consisting of four switches and four links

4.2.1 Packet Switching vs Circut Switching

Devre anahtarlamada olduğu gibi, paket bir dizi iletişim bağlantısı üzerinden iletilir. Ancak devre anahtarlamadan farklı olarak, paket herhangi bir bağlantı kaynağını rezerve etmeden ağa gönderilir. Diğer paketlerin aynı anda aynı bağlantı üzerinden iletilmesi gerektiği için bağlantılardan biri yoğunsa, paket iletim bağlantısının gönderme tarafında bir tampona beklemek zorunda kalır ve gecikme yaşar. İnternet, paketleri zamanında teslim etmek için elinden geleni yapar, ancak herhangi bir garanti vermez.

Paket anahtarlamayı eleştirenler, paket anahtarlamasının gerçek zamanlı hizmetler için (örneğin, telefon görüşmeleri ve video konferans görüşmeleri) uygun olmadığını sıklıkla savunmuşlardır, çünkü değişken ve öngörülemeyen uçtan uca gecikmelere (genellikle değişken

ve öngörülemeyen sıra gecikmelerine) sahiptir. Paket anahtarlamasının destekçileri ise (1) devre anahtarlamasına kıyasla daha iyi bir iletim kapasitesi paylaşımı sağladığını ve (2) devre anahtarlamasından daha basit, daha verimli ve daha az maliyetli olduğunu savunurlar.

4.2.2 Multiplexing in Circuit-Switched Networks

Bir bağlantıdaki devre, frekans bölümleme çoğullama (FDM) veya zaman bölümleme çoğullama (TDM) ile uygulanır.

4.2.2.1 FDM

Farklı frekans aralığında veri gönderir. Aynı frekanstan hem video, hem ses göndermek için kullanılır. FDM ile, bir bağlantının frekans spektrumu, bağlantılar arasında kurulan bağlantılar arasında bölünür. Özellikle, bağlantı her bir bağlantı için bir frekans bandına, bağlantının süresi boyunca bağlantıya adanır. Telefon ağlarında, bu frekans bandının genişliği genellikle 4 kHz'dir (yani, saniyede 4.000 hertz veya 4.000 döngü). FM radyo istasyonları da 88 MHz ile 108 MHz arasında frekans spektrumunu paylaşmak için FDM kullanır, her istasyon belirli bir frekans bandına tahsis edilir.

4.2.2.2 TDM

TDM bağlantısı için, zaman sabit bir süre dilimine bölünür ve her frame sabit sayıda zaman dilimine bölünür. Ağ, bir bağlantıyı bir bağlantı boyunca bir bağlantıya adar, bir bağlantının verilerini iletmek için her framede bir zaman dilimini bu bağlantıya adar. Bu yuvalar, bu bağlantının verilerinin iletimi için (her framede) kullanılmak üzere sadece bu bağlantının özel kullanımı içindir.

A, B, C makinelerinden D'ye aynı anda gönderim yapmak istiyorum. Her 30 ms de bir 10ms A'ya, 10ms B'ye, 10ms C'ye bu hattı tahsis ederim. Bu yöntem verimsizdir. Diyelim ki sadece A makinesi iletişim kurmak istiyor. O zaman A'ya kısıtlama getirip B ve C ye gereksiz zaman ayırıyoruz.

Bunlar yerine packet switching yapmak mantıklı.

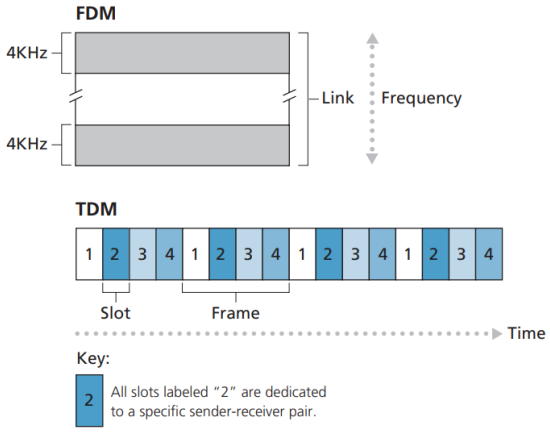


Figure 1.14 ♦ With FDM, each circuit continuously gets a fraction of the bandwidth. With TDM, each circuit gets all of the bandwidth periodically during brief intervals of time (that is, during slots)

Şekil 1.14, en fazla dört devreyi destekleyen belirli bir ağı göstermektedir. FDM için, frekans alanı dört bant halinde bölünmüştür, her biri 4 kHz bant genişliğine sahiptir. TDM için zaman alanı, her karede dört zaman dilimi olacak şekilde karelere bölünür; her devreye aynı tahsis edilmiş yuva atanır. TDM için, bir devrenin iletim hızı, bir yuvadaki bit sayısıyla frame hızının çarpımına eşittir. Örneğin, eğer bağlantı saniyede 8.000 frame iletimi sağlarsa ve her yuva 8 bit içeriyorsa, her devrenin iletim hızı 64 kbps'dir.

Bir dosyanın 640.000 bitinin Host A'dan Host B'ye devre anahtarlama ağı üzerinden gönderilmesinin ne kadar sürdüğünü düşünelim. Ağdaki tüm linklerin 24 yuvayla TDM kullandığını ve bir bit hızına sahip olduğunu varsayalım. Ayrıca, Host A'nın dosyayı iletmeye başlamadan önce uçtan uca bir devre kurulması için 500 milisaniye gerektiğini varsayalım. Dosyayı göndermek ne kadar sürer? Her devrenin iletim hızı (1.536 Mbps) / 24 = 64 kbps olduğundan, dosyanın iletilmesi (640.000 bit) / (64 kbps) = 10 saniye sürer. Buna devre kurma süresini ekleriz ve dosyayı göndermek için 10,5 saniye alırız. İletim süresinin link sayısından bağımsızdır. Uçtan uca devrenin bir veya yüz linkten geçtiği durumda iletim süresi 10 saniye olacaktır.

4.3 Networks of Networks

Uç sistemler (PC'ler, akıllı telefonlar, Web sunucuları, posta sunucuları vb.) bir erişim İSP'si aracılığıyla internet'e bağlanır. Erişim İSP'si, DSL, kablo, FTTH, Wi-Fi ve hücresel gibi bir dizi erişim teknolojisini kullanarak kablosal veya kablosuz bağlantı sağlayabilir. Erişim İSP'sinin bir telekom şirketi veya kablo şirketi olması gerekmez bunun yerine, örneğin bir üniversite (öğrencilere, personel ve öğretim üyelerine internet erişimi sağlayarak) veya bir şirket olabilir.

Genel amaç, tüm son sistemlerin birbirine paket gönderebilmesi için erişim İSP'lerini birbirine bağlamaktır. Her erişim İSP'sinin doğrudan her bir diğer erişim İSP'siyle bağlanması gibi her bir erişim İSP'sinin her bir diğer erişim İSP'siyle doğrudan bağlanması gibi saf bir yaklaşım, elbette erişim İSP'leri için çok maliyetlidir, çünkü her bir erişim İSP'sinin, dünya çapındaki yüz binlerce diğer erişim İSP'sine ayrı bir iletişim bağlantısına sahip olması gerekir.

İlk ağ yapımız, Ağ Yapısı 1, tüm erişim İSP'lerini tek bir global transit İSP ile birbirine bağlar (hayali). Global transit İSP'miz, dünyayı kaplayan bir dizi yönlendirici ve iletişim bağlantısından oluşur ve yalnızca dünya genelindeki yüz binlerce erişim İSP'sinin her birine yakın en az bir yönlendiriciye sahiptir. Tabii ki, böyle geniş bir ağı inşa etmek global İSP için çok maliyetli olurdu. Karlı olabilmesi için, her bir erişim İSP'sine bağlantı için ücret talep eder ve fiyatlandırma, erişim İSP'sinin global İSP ile alışveriş yaptığı trafiğin miktarını yansıtır (ancak bunun doğrudan orantılı olması gerekmez). Erişim İSP'si global transit İSP'ye ödeme yapar, bu nedenle erişim İSP'si müşteri olarak adlandırılır ve global transit İSP'si sağlayıcı olarak adlandırılır.

Şimdi, eğer bir şirket karlı bir global transit İSP inşa eder ve işletirse, diğer şirketlerin kendi global transit İSP'lerini inşa etmeleri ve orijinal global transit İSP ile rekabet etmeleri doğal olur. Bu, Ağ Yapısı 2'ye yol açar, yani yüz binlerce erişim İSP'si ve birden fazla global transit İSP'si içerir. Erişim İSP'leri kesinlikle Ağ Yapısı 2'yi Ağ Yapısı 1'e tercih eder çünkü artık fiyatlandırma ve hizmetlerine göre rekabet eden global transit sağlayıcılarını seçebilirler. Ancak, global transit İSP'lerinin kendilerinin de birbirine bağlanması gerekir: Aksi takdirde, bir global transit sağlayıcıya bağlı erişim İSP'leri, diğer global transit sağlayıcılara bağlı erişim İSP'leri ile iletişim kuramaz.

Yukarıda tanımlanan Ağ Yapısı 2, üst katmanda global transit sağlayıcılarını ve alt katmanda erişim İSP'lerini içeren iki katlı bir hiyerarşiye sahiptir. Bu, global transit İSP'lerinin her bir erişim İSP'sine yakın olmakla kalmayıp aynı zamanda bunu ekonomik olarak arzu edilebilir bulduğunu varsayar. Gerçekte, bazı İSP'ler etkileyici bir küresel kapsamı olabilir ve birçok erişim İSP'sine doğrudan bağlanabilirken, hiçbir İSP'nin dünyadaki her şehirde varlık göstermez. Bunun yerine, herhangi bir bölgede, bölgesel erişim İSP'lerinin bağlandığı bir bölgesel İSP olabilir. Her bir bölgesel İSP daha sonra birinci katman İSP'lerine bağlanır. Birinci katman İSP'leri, (hayali) global transit İSP'imize benzer; ancak gerçekten var olan tier-1 İSP'leri, dünyadaki her şehirde varlık göstermezler.

Günümüzde kullanılan yapı Points of Presence (PoP'ler), çoklu yönlendiriciler (multi homing), peering ve İnternet değişim noktalarının (IXP'ler) Ağ Yapısı-3' e eklenmiş halidir.. PoP'ler hiyerarşinin tüm seviyelerinde mevcuttur, ancak alt seviye (access İSP'si) seviyesinde değildir. Bir PoP, müşteri İSP'lerinin sağlayıcı İSP'ye bağlanabileceği sağlayıcı İSP'nin ağındaki bir veya daha fazla yönlendirici grubudur. Bir müşteri ağı, sağlayıcının PoP'sine bağlanmak için, bir üçüncü taraf telekomünikasyon sağlayıcısından bir yüksek hızlı bağlantı kiralayarak bir yönlendiricisini PoP'daki bir yönlendiriciye doğrudan bağlayabilir. Herhangi bir İSP (birinci

katman İSP'leri hariç) çoklu yönlendiricilere bağlanmayı tercih edebilir, yani iki veya daha fazla sağlayıcı İSP'ye bağlanabilir. Örneğin, bir erişim İSP'si iki bölgesel İSP ile çoklu yönlendiricilere bağlanabilir veya iki bölgesel İSP ve birinci katman İSP ile çoklu yönlendiricilere bağlanabilir. Benzer şekilde, bir bölgesel İSP birden fazla birinci katman İSP'sine çoklu yönlendiricilere bağlanabilir. Bir İSP çoklu yönlendiricilere bağlandığında, sağlayıcılarından biri arızalandığında bile İnternet'e paket göndermeye ve almayı sürdürebilir.

Bir müşteri İSP'nin bir sağlayıcı İSP'ye ödediği miktar, sağlayıcıyla değiştirdiği trafiğin miktarını yansıtır. Bu maliyetleri azaltmak için, aynı hiyerarşinin aynı seviyesinde bulunan iki yakındaki İSP, peering yapabilir, yani ağlarını birbirine doğrudan bağlayabilir, böylece aralarındaki tüm trafiğin doğrudan bağlantı üzerinden, yukarı akış aracılar aracılığıyla değil, geçmesini sağlar. İki İSP peer yaptığında, genellikle karşılıksızdır, yani bir İSP diğerine ödeme yapmaz. Birinci katman İSP'lerinin de birbirleriyle karşılıksız peer yaptığına dikkat edilmelidir. Ayrıca, bir üçüncü taraf şirket, birden fazla İSP'nin bir araya gelebileceği bir İnternet Değişim Noktası (ISP) oluşturabilir. Bir ISP genellikle kendi anahtarlarına sahip bağımsız bir binada bulunur. Bugün İnternet'te 600'den fazla ISP bulunmaktadır. Bu ekosistemi- erişim İSP'leri, bölgesel İSP'ler, birinci katman İSP'ler, PoP'ler, çoklu yönlendiricilere bağlanma, peering ve ISP'ler- Ağ Yapısı 4 olarak adlandırıyoruz.

Günümüzde Ağ Yapısı 5'te firmaların özel sunucuları bulunabilmektedir. Google'ın birkaç yüz sunucuya sahip daha küçük veri merkezleri bulunmaktadır; bu daha küçük veri merkezleri genellikle IXPler içinde bulunur. Google veri merkezleri, dünya genelinde yayılan ancak yine de genel İnternet'ten ayrı olan Google'ın özel TCP/IP ağı aracılığıyla birbirine bağlıdır. Önemli bir nokta olarak, Google özel ağı sadece Google sunucularına gelen/giden trafiği taşır. Şekil 1.15'te gösterildiği gibi, Google özel ağı, üst katmanlardaki İnternet'i bypass etmeye çalışarak alt katman İSP'leriyle ya doğrudan bağlanarak ya da ISPler aracılığıyla bağlanarak peering yapar. Ancak, birçok erişim İSP'sine hala tier-1 ağlar aracılığıyla ulaşılabilir olduğu için, Google ağı ayrıca tier-1 İSP'lerine bağlanır ve bu İSP'lerle trafiği değiştirmek için bu İSP'lere ödeme yapar. Kendi ağını oluşturarak, bir içerik sağlayıcı yalnızca üst katman İSP'lere olan ödemelerini azaltmakla kalmaz, aynı zamanda hizmetlerinin son kullanıcılara nasıl ulaştırılacağı konusunda daha büyük bir kontrol sahibi olur.

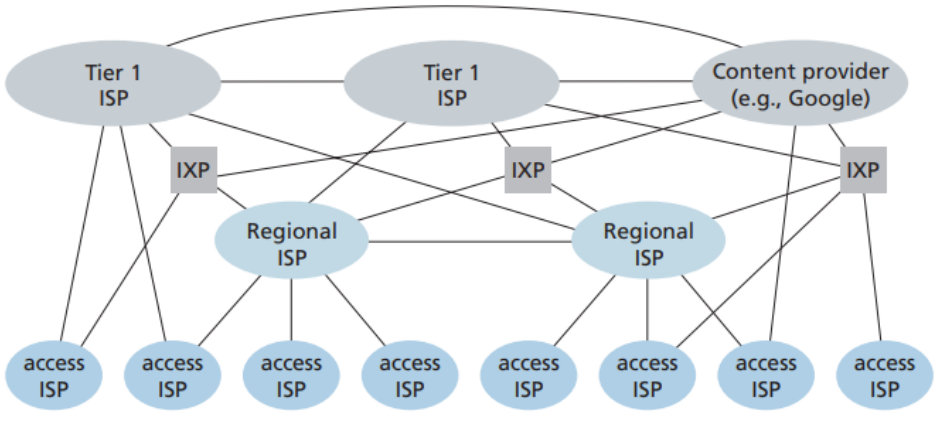


Figure 1.15 ♦ Interconnection of ISPs

4.4 Delay, Loss, and Throughput in Packet-Switched Networks

Gecikmelerin en önemlileri nodal processing delay, queuing delay, transmission delay ve propagation delay'dir. Bu gecikmeler bir araya gelerek total nodal delayi verir.

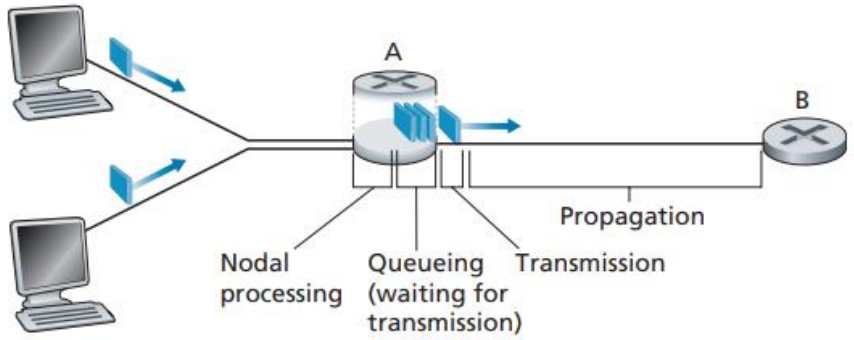


Figure 1.16 ♦ The nodal delay at router A

Şekil 1.16 bağlamında bu gecikmeleri inceleyelim. Kaynak ve hedef arasındaki uçtan uca rotasının bir parçası olarak, bir paket yukarı akış düğümünden router A üzerinden router B'ye gönderilir. Amacımız, router A'daki nodal delayi belirlemektir. Router A'nın, router B'ye çıkan bir bağlantısı bulunmaktadır. Bu bağlantının önünde bir queue bulunur (buffer olarak da bilinir). Paket yukarı akış düğümünden router A'ya ulaştığında, router A paketin başlığını inceleyerek paket için uygun çıkan bağlantıyı belirler ve ardından paketi bu bağlantıya yönlendirir. Bu örnekte, paket için çıkan bağlantı, router B'ye götüren bağlantıdır. Bir paket, linkte şu anda başka bir paketin iletilmediği ve kuyrukta öncesinde gelen başka paketlerin olmadığı durumda linkte iletilir; eğer link şu anda meşgulse veya link için zaten kuyrukta olan başka paketler varsa, yeni gelen paket kuyruğa katılır.

$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

4.4.1 Processing Delay

Paketin headerini incelemek ve paketi nereye yönlendireceğini belirlemek için gereken süre, processing delayin bir parçasıdır. Processing delay ayrıca, paketin bitlerinin yukarı akış düğümünden router A'ya iletilirken oluşan bit düzeyinde hataları kontrol etmek için gereken süre gibi diğer faktörleri de içerebilir. Yüksek hızlı yönlendiricilerde processing delayler genellikle mikrosaniye veya daha az düzeydedir. Bu nodal processingden sonra, router paketi router B'ye giden bağlantıya önceden gelen kuyruğa yönlendirir.

4.4.2 Queuing Delay

Kuyrukta, paket, bağlantıya iletilmeyi beklerken bir queuing delay yaşar. Belirli bir paketin queuing delayi, kuyrukta sıraya girmiş ve bağlantıya iletilmek için bekleyen önceki paketlerin sayısına bağlı olacaktır. Eğer kuyruk boşsa ve şu anda başka bir paket iletilmiyorsa, o zaman paketimizin queuing delay sıfır olacaktır. Diğer yandan, eğer trafik yoğunsa ve birçok başka paket de iletilmeyi bekliyorsa, queuing delay uzun olacaktır.

4.4.3 Transmission Delay

Paketlerin genellikle ilk gelen ilk hizmet (first-come-first-served) prensibine göre iletilmesi varsayıldığında, paketimiz sadece önünde gelen tüm paketler iletilmiş olduktan sonra iletilir. Paketin uzunluğunu L bitlerle gösterelim ve router A'dan router B'ye olan bağlantının iletim hızını (transmission rate) R bit/saniye olarak gösterelim. Örneğin, bir 10 Mbps Ethernet bağlantısı için, hız R = 10 Mbps; bir 100 Mbps Ethernet bağlantısı için, hız R = 100 Mbps'dir. İletim gecikmesi L/R'dir. Bu, paketin tüm bitlerinin bağlantıya (yani, iletilmesine) itilmesi için gereken süredir. Transmission delay uygulamada genellikle mikrosaniyelerden milisaniyelere kadar olabilir.

4.4.4 Propagation Delay

Bağlantıya itilen (push edilen) bir bit, router B'ye yayılmak üzere hareket etmesi gerekmektedir. Bağlantının başından router B'ye yayılması için gereken süre, propagation delaydir. Bit, bağlantının yayılma hızında (propagation speed) yayılır. Yayılma hızı, bağlantının fiziksel ortamına bağlıdır (yani, fiber optik, örgülü bakır tel vb.) ve ışık hızına eşit veya biraz daha az olan bir aralıktadır. Propagation delay, iki router arasındaki mesafenin yayılma hızına bölünmesiyle hesaplanır. Yani, yayılma gecikmesi d/s'dir, burada d router A ve router B arasındaki mesafeyi ve s bağlantının yayılma hızını temsil eder. Paketin son biti router B'ye ulaştığında, paketin tüm önceki bitleriyle birlikte router B'de depolanır. Daha sonra tüm süreç, B yönlendiricisinin artık yönlendirmeyi gerçekleştirmesiyle devam eder. Geniş alan ağlarında, yayılma gecikmeleri milisaniyeler düzeyindedir.

4.4.5 Propagation Delay vs Transmission Delay

Transmission delay, yönlendiricinin paketi göndermesi için gereken süredir; paketin uzunluğu ve bağlantının iletim hızının bir fonksiyonudur, ancak iki yönlendirici arasındaki mesafe ile ilgili değildir. Diğer yandan, propagation delay, bir bitin bir yönlendiriciden diğerine yayılması için gereken zamandır; iki yönlendirici arasındaki mesafenin bir fonksiyonudur, ancak paketin uzunluğu veya bağlantının iletim hızı ile ilgili değildir. Bu kısımla alakalı kitapta sayfa 49da konvoy örneği var okuyabilirsin.

Dprop, iki yönlendiriciyi aynı üniversite kampüsünde birbirine bağlayan bir bağlantı için ihmal edilebilir olabilir (örneğin, birkaç mikrosaniye); ancak, dprop, birbirine bağlı iki yönlendirici için bir geostatik uydu bağlantısıyla yüzlerce milisaniye olabilir ve dnodal'da baskın terim olabilir. Benzer şekilde, dtrans ihmal edilebilir olabileceği gibi önemli olabilir. Katkısı genellikle 10 Mbps ve daha yüksek iletim hızları için ihmal edilebilirken (örneğin, LAN'lar için), büyük İnternet paketleri üzerinden düşük hızlı dial-up modem bağlantılarına

gönderilenler için yüzlerce milisaniye olabilir. Dproc, genellikle ihmal edilebilir; ancak, bir yönlendiricinin maksimum verimliliğini etkileyen, yani bir yönlendiricinin paketleri yönlendirme hızı, önemli ölçüde etkiler.

4.4.6 Queuing Delay and Packet Loss

4.4.6.1 Queuing Delay

Diğer üç gecikmenin (yani, dproc, dtrans ve dprop) aksine, kuyruk gecikmesi paketten pakete değişebilir. Örneğin, aynı anda boş bir kuyruğa 10 paket gelirse, iletilen ilk paket kuyruk gecikmesi yaşamazken, iletilen son paket (diğer dokuz paketin iletilmesini beklerken) göreceli olarak büyük bir kuyruk gecikmesi yaşayacaktır. Kuyruk gecikmesini karakterize ederken, genellikle ortalama kuyruk gecikmesi, kuyruk gecikmesinin varyansı ve kuyruk gecikmesinin belirli bir değeri aşma olasılığı gibi istatistiksel ölçüleri kullanılır.

Kuyruk gecikmesi ne zaman büyük olur ve ne zaman önemsizdir?

- Trafiğin kuyruğa varış hızı
- Bağlantının iletim hızı
- Gelen trafik türüne yani trafiğin periyodik olarak mı yoksa aralıklarla mı geldiğine bağlıdır.

a kuyruğa paketlerin ortalama hızını temsil eder (paket/saniye). R'nin iletim hızıdır yani, bağlantıdan itilen bitlerin hızı (bit/saniye cinsinden). Basitlik için, tüm paketlerin L bitlerinden oluştuğunu varsayalım. Ardından, kuyruğa gelen bitlerin ortalama hızı La bit/saniye olur. La/R olarak adlandırılan bu oran, trafik yoğunluğunu tahmin etmemizi sağlar. Eğer La/R > 1 ise, kuyruğa gelen bitlerin ortalama hızı, bitlerin kuyruktan iletilme hızını aşar. Bu durumda, kuyruk sınırsız bir şekilde artacaktır ve kuyruk gecikmesi sonsuza yaklaşacaktır. La/R ≤ 1 durumunda gelen trafik türünün kuyruk gecikmesine etkisi var. Örneğin, paketler periyodik olarak gelirse- yani, her L/R saniyede bir paket gelirse- her paket boş bir kuyruğa ulaşır ve kuyruk gecikmesi olmaz. Öte yandan, paketler aralıklı olarak gelirse ama periyodik olarak, önemli bir ortalama kuyruk gecikmesi olabilir. Örneğin, her (L/R)N saniyede N paket aynı anda gelirse. O zaman iletilen ilk paketin kuyruk gecikmesi olmaz; ikinci iletilen paketin kuyruk gecikmesi L/R saniye olur.

Tipik olarak, bir kuyruğa gelen geliş süreci rastgeledir; yani, gelişler herhangi bir deseni takip etmez ve paketler rastgele zaman aralıklarıyla ayrılır. Bu daha gerçekçi durumda, La/R miktarı genellikle kuyruk gecikme istatistiklerini tam olarak karakterize etmek için yeterli değildir. Bununla birlikte, kuyruk gecikme derecesini sezgisel olarak anlamakta faydalıdır. Özellikle, eğer trafik yoğunluğu sıfıra yakınsa, o zaman paket gelişleri az ve uzak aralıklarla olacak ve bir gelen paketin kuyrukta başka bir paket bulması olası değildir. Bu nedenle, ortalama kuyruk gecikmesi sıfıra yakın olacaktır. Diğer yandan, trafik yoğunluğu 1'e yaklaştığında, geliş hızının iletim kapasitesini (paket geliş hızındaki değişkenlere bağlı olarak) aşacağı zaman aralıkları olacaktır ve bu zaman aralıklarında bir kuyruk oluşacaktır; geliş hızı iletim kapasitesinden daha

düşük olduğunda, kuyruğun uzunluğu azalacaktır. Bununla birlikte, trafik yoğunluğu 1'e yaklaştıkça, ortalama kuyruk uzunluğu giderek artar. Ortalama kuyruk gecikmesinin trafik yoğunluğuna bağlı kalitatif bağımlılığı Şekil 1.18'de gösterilmiştir.

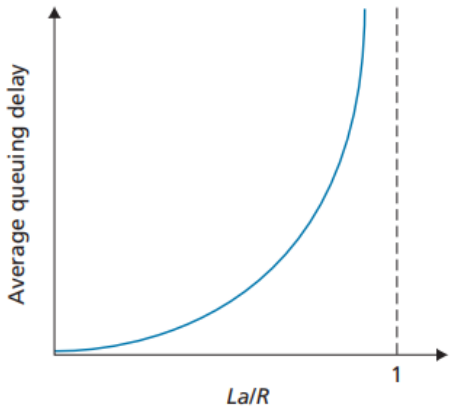


Figure 1.18 ♦ Dependence of average queuing delay on traffic intensity

Şekil 1.18'in önemli bir yönü, trafik yoğunluğunun 1'e yaklaştıkça, ortalama kuyruk gecikmesinin hızla artmasıdır. Yoğunluğun küçük bir yüzdesindeki artış, gecikide çok daha büyük bir yüzde artışına neden olacaktır.

4.4.6.2 Packet Loss

Kuyruk kapasitesinin büyük ölçüde yönlendirici tasarımı ve maliyetine bağlıdır. Kuyruk kapasitesi sonlu olduğundan, trafik yoğunluğu 1'e yaklaştıkça paket gecikmeleri gerçekte sonsuzluğa yaklaşmaz. Bunun yerine, bir paketin tam bir kuyrukla karşılaşması mümkündür. Böyle bir paketi saklamak için yer olmadığında, bir yönlendirici o paketi düşürür; yani, paket kaybolur. Bir uç sistem bakış açısından, bir paket kaybı, bir paketin ağ çekirdeğine iletilmiş ancak hedefteki ağdan hiç çıkmamış gibi görünecektir. Kaybolan paketlerin oranı, trafik yoğunluğu arttıkça artar. Bu nedenle, bir düğümde performans genellikle sadece gecikme açısından değil, aynı zamanda paket kaybı olasılığı açısından da ölçülür.

4.4.7 End-to-End Delay

Kaynak ana makine ile hedef ana makine arasında N - 1 yönlendirici olduğunu varsayalım. Ayrıca, bir süre için ağın kalabalık olmadığını (bu nedenle kuyruk gecikmelerinin ihmal edilebilir olduğunu), her yönlendiricide ve kaynak ana makinede işleme gecikmesinin dproc olduğunu, her yönlendiriciden ve kaynak ana makineden çıkan iletim hızının R bit/saniye olduğunu ve her bir bağlantıda yayılma gecikmesinin dprop olduğunu varsayalım. Nodal gecikmeler birikir ve bir uçtan uca gecikme verir

$$d_{\text{end-end}} = N(d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}})$$

4.4.7.1 Traceroute

Kaynak ve hedef arasında N - 1 yönlendirici olduğunu varsayalım. Kaynak, her biri son hedefe adreslenmiş N özel paket gönderir. Bu N özel paket 1'den N'ye kadar işaretlenir, ilk paket 1 işareti ile işaretlenir ve son paket N işareti ile işaretlenir. n'inci yönlendirici, n'inci, n ile işaretlenmiş paketi aldığında, paketi hedefine doğru yönlendirmez, ancak kaynağa bir mesaj gönderir. Hedef ana bilgisayar, N'inci paketi aldığında, o da bir mesajı kaynağa geri gönderir. Kaynak, bir paket gönderdiği zaman ve karşılık gelen geri dönüş mesajını aldığı zaman arasında geçen süreyi kaydeder; ayrıca mesajı geri gönderen yönlendiricinin (veya hedef ana bilgisayarın) adını ve adresini kaydeder. Bu şekilde, kaynak, kaynaktan hedefe akan paketlerin aldığı rotayı yeniden oluşturabilir ve kaynağın aradaki tüm yönlendiricilere gidiş-dönüş gecikmelerini belirleyebilir. Traceroute aslında yukarıda açıklanan deneyi üç kez tekrarlar, bu nedenle kaynak aslında hedefe 3 • N paket gönderir.

4.4.8 Ek Not

Processing, transmission, and propagation delaylere ek olarak önemli gecikmeler olabilir. Örneğin, bir son sistem, paylaşılan bir ortama bir paket iletmek istediğinde (örneğin, bir WiFi veya kablo modem senaryosunda olduğu gibi), ortamı diğer son sistemlerle paylaşma protokolünün bir parçası olarak iletimini bilinçli olarak geciktirebilir. Başka bir önemli gecikme ise Medya Paketleme Gecikmesi'dir, bu gecikme ses üzerinden internet protokolü (VoIP) uygulamalarında mevcuttur. VoIP'te, gönderme tarafı önce bir paketi kodlanmış dijitalleştirilmiş konuşma ile doldurmalı ve ardından paketi internet'e iletmelidir. Bir paketi doldurma süresi - paketleme gecikmesi olarak adlandırılan - önemli olabilir ve VoIP aramasının kullanıcı tarafından algılanan kalitesini etkileyebilir.

4.4.9 Throughput in Computer Networks

Verimliliği tanımlamak için, bir bilgisayar ağı üzerinden Ana Bilgisayar A'dan Ana Bilgisayar B'ye büyük bir dosyanın transferini düşünün. Bu transfer, örneğin bir bilgisayardan diğerine büyük bir video klibi olabilir. Anlık verimlilik herhangi bir anda, Ana Bilgisayar B'nin dosyayı almakta olduğu hızdır (bit/saniye cinsinden) Dosya F bitinden oluşuyorsa ve aktarım Ana Bilgisayar B'nin tüm F bitini alması için T saniye sürerse, dosya transferinin ortalama verimliliği F/T bit/saniyedir.

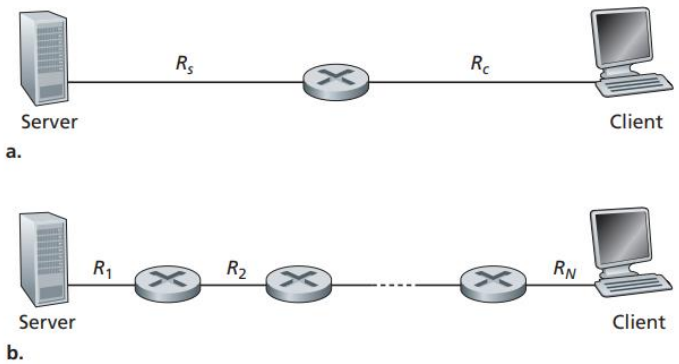


Figure 1.19 ♦ Throughput for a file transfer from server to client

Şekil 1.19(a), iki uç sistemi, bir sunucu ve bir istemciyi, iki iletişim bağlantısı ve bir yönlendiriciyle bağlı olarak göstermektedir. Sunucudan istemciye dosya aktarımı için verimliliği düşünelim. Rs, sunucu ile yönlendirici arasındaki bağlantı hızını ve Rc, yönlendirici ile istemci arasındaki bağlantı hızını ifade etsin. Tüm ağ boyunca gönderilen tek bitlerin sunucudan istemciye olduğunu varsayalım.

Bu ideal senaryoda sunucu-istemci verimliliği nedir? Bu soruyu yanıtlamak için, bitleri sıvı ve iletişim bağlantılarını borular olarak düşünebiliriz. Açıkça, sunucu Rs bps hızından daha hızlı bir hızda bitleri bağlantısına pompalayamaz ve yönlendirici bitleri Rc bps hızından daha hızlı bir hızda iletemez. Eğer Rs ≤ Rc ise, sunucu tarafından pompalanan bitler yönlendiriciyi doğruca "akıp" geçecek ve istemciye Rs bps hızında ulaşacak, bu da bir verimlilik sağlayacak. Diğer taraftan, eğer Rc ≤ Rs ise, yönlendirici bitleri alır almaz iletemeyecektir. Bu durumda, bitler yalnızca yönlendiriciden Rc hızında ayrılacak ve sonuç olarak uçtan uca verimlilik Rc olacaktır. (Ayrıca, eğer bitler hala Rs hızında yönlendiriciye ulaşıyorsa ve hala yönlendiriciden Rc hızında ayrılıyorsa, istemciye iletim için bekleyen yönlendiricideki bitlerin birikimi artacaktır- bu en istenmeyen durumdur!) Dolayısıyla bu basit iki bağlantılı ağ için verim min{Rc, Rs}'dir, yani darboğaz hattının iletim hızıdır (bottleneck of transmission rate). Verimliliği belirledikten sonra, büyük bir F bit dosyasının sunucudan istemciye transfer edilmesi için gereken zamanı F/min{Rs, Rc} olarak yaklaşıklayabiliriz.

Şekil 1.19(b) sunucu ve istemci arasındaki N bağlantıya sahip bir ağı gösterir ve bu N bağlantının iletim hızları R1, R2, ..., RN olarak belirtilir. İki bağlantılı ağ için yapılan analiz gibi aynı analizi uyguladığımızda, sunucudan istemciye dosya transferi için verimliliğin min{R1, R2, ..., RN} olduğunu buluruz, bu da yine sunucu ve istemci arasındaki yol boyunca şişen bağlantının iletim hızıdır.

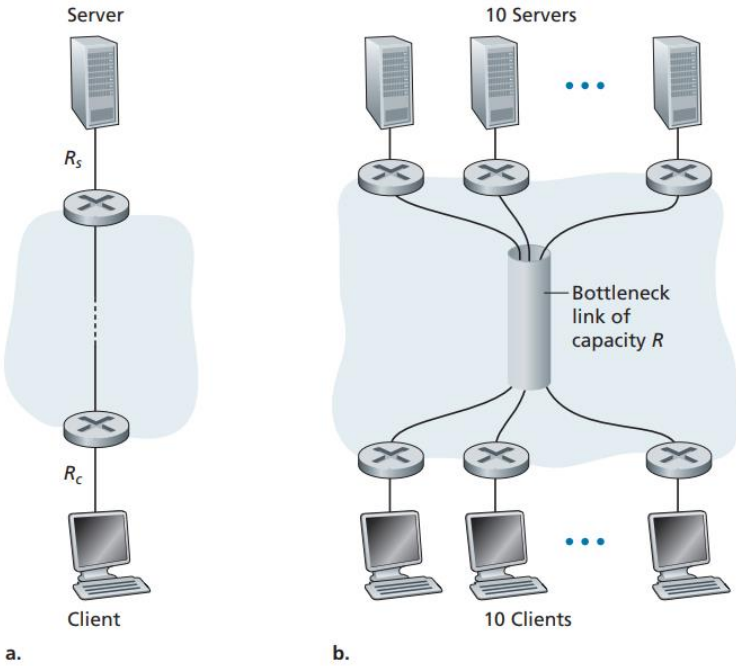


Figure 1.20 ♦ End-to-end throughput: (a) Client downloads a file from server; (b) 10 clients downloading with 10 servers

Şekil 1.20(a), bir sunucu ve bir istemci olmak üzere iki uç sistemini bir bilgisayar ağına bağlı gösterir. Sunucu, Rs hızında bir erişim bağlantısıyla ağa bağlıdır ve istemci, Rc hızında bir erişim bağlantısıyla ağa bağlıdır. Şimdi, iletişim ağının çekirdeğindeki tüm bağlantıların iletim hızlarının, Rs ve Rc'den çok daha yüksek olduğunu varsayalım. Ayrıca, ağdaki tüm bitlerin sunucudan istemciye gönderildiğini varsayalım. Bu örnekte, bilgisayar ağının çekirdeği geniş bir boruya benzediği için, kaynaktan hedefe bitlerin akabileceği hız, tekrar min{Rs, Rc}, yani verimlilik = min{Rs, Rc} olacaktır. Dolayısıyla, günümüz İnternet'inin verimlilik için kısıtlayıcı faktörü genellikle erişim ağıdır.

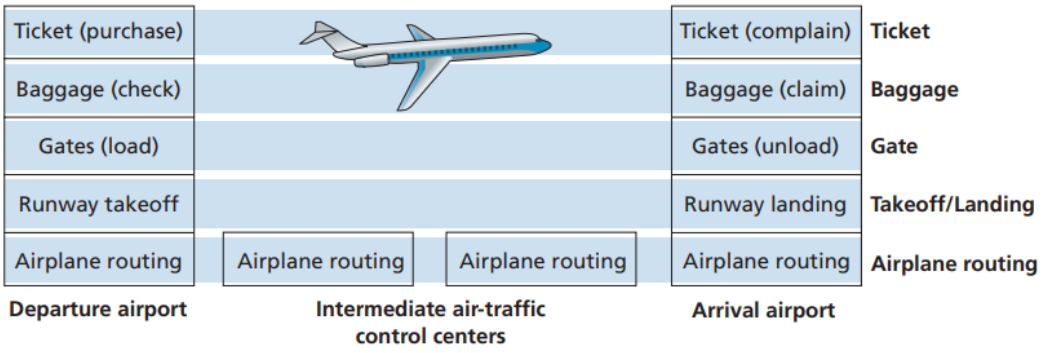
Şekil 1.20(b)'de bilgisayar ağının çekirdeğine bağlı 10 sunucu ve 10 istemci bulunmaktadır. Bu örnekte, 10 eşzamanlı indirme gerçekleşmektedir ve 10 istemci-sunucu çiftini içerir. Bu 10 indirme, şu anda ağdaki tek trafiktir. Şekilde gösterildiği gibi, tüm 10 indirmenin geçtiği çekirdek bir bağlantı vardır. Bu bağlantının iletim hızı R olarak gösterilsin. Tüm sunucu erişim bağlantılarının aynı hızda Rs'ye, tüm istemci erişim bağlantılarının aynı hızda Rc'ye ve çekirdek bağlantılarının iletim hızlarının Rs, Rc ve R'den çok daha büyük olduğunu varsayalım -R ile bağlantıların biri dışında. Şimdi soruyoruz, indirmelerin verimlilikleri nedir? Açıkça, ortak bağlantının hızı, R, büyükse -mesela Rs ve Rc'nin her ikisinden de yüz kat daha büyükse- her indirme için verimlilik tekrar min{Rs, Rc} olacaktır. Ancak ortak bağlantının hızı Rs ve Rc ile aynı düzeydeyse ne olacak? Bu durumda verimlilik ne olur? Belirli bir örneğe bakalım. Diyelim ki Rs = 2 Mbps, Rc = 1 Mbps, R = 5 Mbps ve ortak bağlantının hızının, R, 10 indirmeye eşit olarak bölündüğünü varsayalım. O zaman her bir indirme için şişen nokta artık erişim ağında değil, ancak her bir indirmeye yalnızca 500 kbps veren çekirdekteki paylaşılan bağlantıdır. Dolayısıyla, her bir indirme için uçtan uca verimlilik şimdi 500 kbps'ye düşürülür.

Şekil 1.19 ve Şekil 1.20(a) daki örnekler, verimliliğin veri akışının üzerinden geçtiği bağlantıların iletim hızlarına bağlı olduğunu göstermektedir. Başka bir müdahale eden trafik olmadığında, verimliliğin basitçe kaynak ve hedef arasındaki yol boyunca en düşük iletim hızı olarak yaklaşıklandığını gördük. Şekil 1.20(b)'deki örnek, daha genel olarak verimliliğin yalnızca yol boyunca bağlantıların iletim hızlarına değil, aynı zamanda müdahil olan trafiklere de bağlı olduğunu göstermektedir. Özellikle, yüksek bir iletim hızına sahip bir bağlantı, birçok başka veri akışı da bu bağlantıdan geçiyorsa bile, dosya transferi için bottleneck link olabilir.

5.0 Protocol Layers ve Servis Modelleri

5.1 Layered Architecture

Katmanın sağladığı hizmetin uygulamasını değiştirmeyi çok daha kolay hale getirir. Bir katmanın, üstündeki katmana aynı hizmeti sağladığı ve altındaki katmandan aynı hizmetleri kullandığı sürece, katmanın uygulaması değiştirildiğinde sistem geri kalanı değişmez kalır. Hizmetin uygulamasını değiştirmek, hizmeti kendisini değiştirmekten çok farklıdır. Örneğin, kapı işlevleri değiştirilirse (örneğin, insanların boyuna göre binmelerini ve inmelerini sağlamak için), havayolu sisteminin geri kalanı değişmez kalır çünkü kapı katmanı hala aynı işlevi sağlar (insanları yüklemek ve indirmek); değişiklikten sonra sadece bu işlevi farklı bir şekilde uygular. Sürekli olarak güncellenen büyük ve karmaşık sistemler için, bir hizmetin uygulamasını değiştirme yeteneği, diğer sistem bileşenlerini etkilemeden bir hizmetin uygulamasını değiştirmenin başka bir önemli avantajıdır.



5.1.2 Protocol Layering

Ağ protokollerinin tasarımına yapı sağlamak için ağ tasarımcıları, protokolleri ve bu protokolleri uygulayan ağ donanımını ve yazılımını katmanlar halinde düzenler. Her protokol, şekil 1.22'deki havayolu mimarisinde olduğu gibi bir katmana aittir. Bir katmanın üstündeki katmana sunduğu hizmete katmanın hizmet modeli denir. Havayolu örneğimizde olduğu gibi, her katman, (1) kendi katmanında belirli eylemleri gerçekleştirerek ve (2) doğrudan altındaki katmanın hizmetlerini kullanarak hizmetini sağlar. Örneğin, katman n tarafından sağlanan hizmetler şunları içerebilir:

Mesajların ağın bir ucundan diğer ucuna güvenilir şekilde iletilmesi. Bu, katman n-1'in güvenilmez bir uçtan uca mesaj dağıtım hizmeti kullanılarak ve kayıp mesajları tespit etmek ve yeniden iletmek için katman n işlevselliği eklenerek uygulanabilir.

Bir protokol katmanı yazılımda, donanımda veya ikisinin birleşimiyle uygulanabilir. HTTP ve SMTP gibi uygulama katmanı protokolleri neredeyse her zaman son sistemlerdeki yazılımlara (oftware in the end systems) uygulanır; transport-layer protocolleride öyle.

Physical layer ve data link layerlar, belirli bir bağlantı üzerinden iletişimin yönetilmesinden sorumlu olduğundan, bunlar genellikle belirli bir bağlantıyla ilişkili bir ağ arayüz kartında (örneğin, Ethernet veya WiFi arayüz kartları) uygulanır.

Ağ katmanı genellikle donanım ve yazılımın karma bir uygulamasıdır. Ayrıca, katmanlı havayolu mimarisindeki işlevlerin sistemi oluşturan çeşitli havalimanları ve uçuş kontrol merkezleri arasında dağıtıldığı gibi, katman n protokolünün de uç sistemler, paket anahtarları ve sistemi oluşturan diğer bileşenler arasında *dağıtılır (distributed)*. Yani, bu ağ bileşenlerinin her birinde genellikle bir katman n protokolü parçası bulunur.

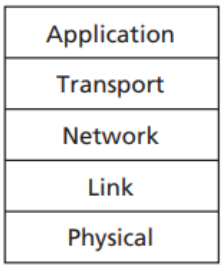
Protocol Layer Avantajı:

Protokol katmanlamanın kavramsal ve yapısal avantajları vardır. Katmanlama, sistem bileşenlerini tartışmak için yapılandırılmış bir yol sağlar. Modülerlik, sistem bileşenlerinin güncellenmesini kolaylaştırır.

Protocol Layer Dezavantajı:

Katmanlamanın olası bir dezavantajı, bir katmanın alt katman işlevselliğini kopyalayabilmesidir. Örneğin, birçok protokol yığını hem bağlantı başına hem de uçtan uca temelde hata kurtarma sağlar. İkinci potansiyel dezavantaj, bir katmandaki işlevselliğin yalnızca başka bir katmanda mevcut olan bilgiye (örneğin, zaman damgası değerine) ihtiyaç duyabilmesidir; bu, katmanların ayrılması amacını ihlal eder.

Bir araya getirildiğinde çeşitli katmanların protokollerine protokol yığını adı verilir. İnternet protokol yığını beş katmandan oluşur: Şekil 1.23'te gösterildiği gibi physical (fiziksel), link (bağlantı), network (ağ), transport (taşıma) ve application (uygulama) layerlarıdır.



Five-layer Internet protocol stack

Figure 1.23 ♦ The Internet protocol stack

5.1.2.1 Application Layer

Uygulama katmanı, ağ uygulamalarının ve bunların uygulama katmanı protokollerinin bulunduğu yerdir. İnternet'in uygulama katmanı, HTTP protokolü (Web belgesi isteği ve aktarımını sağlar), SMTP (e-posta mesajlarının aktarımını sağlar) ve FTP (dosyaların uç sistem arasında aktarımını sağlar) gibi birçok protokolü içerir

www.ietf.org gibi internet uç sistemleri için insan dostu adların 32 bitlik bir ağ adresine çevrilmesi gibi belirli ağ işlevlerinin de belirli bir uygulama katmanı protokolünün yardımıyla yapılır. Bu protokol, alan adı sistemi (DNS) olarak adlandırılır.

Bir uygulama katmanı protokolü birden fazla uç sisteme dağıtılır; bir uç sistemdeki uygulama, başka bir uç sistemdeki uygulama ile bilgi paketleri alışverişinde bulunmak için protokolü kullanır. Uygulama katmanındaki bu bilgi paketine mesaj (message) olarak değineceğiz.

5.1.2.2 Transport Layer

İnternetin taşıma katmanı, uygulama katmanı mesajlarını uygulama uç noktaları arasında taşır. İnternette, her biri uygulama katmanı mesajlarını taşıyabilen TCP ve UDP olmak üzere iki aktarım (transport) protokolü vardır.

TCP

TCP, uygulamalarına bağlantı odaklı bir hizmet sağlar. Bu hizmet, uygulama katmanı mesajlarının hedefe garantili teslimini ve akış kontrolünü (yani gönderen/alıcı hız eşleştirmesini) içerir. TCP aynı zamanda uzun mesajları daha kısa bölümlere ayırır ve bir tıkanıklık kontrol mekanizması sağlar, böylece ağ tıkandığında kaynak iletim hızını azaltır.

UDP

UDP protokolü, uygulamalarına bağlantısız bir hizmet sağlar. Bu, güvenilirlik, akış kontrolü ve tıkanıklık kontrolü sağlamayan, gösterişten uzak bir hizmettir.

Not => Kitapta taşıma katmanı paketini segment olarak adlandırılacak.

5.1.2.2 Network Layer

İnternet ağ katmanı, datagramlar olarak bilinen ağ katmanı paketlerinin bir ana bilgisayardan diğerine taşınmasından sorumludur. Kaynak ana bilgisayardaki internet aktarım katmanı protokolü (TCP veya UDP), tıpkı posta servisine hedef adresi olan bir mektup verdiğiniz gibi, bir aktarım katmanı segmentini ve bir hedef adresi ağ katmanına geçirir. Ağ katmanı daha sonra segmentin hedef ana bilgisayardaki taşıma katmanına teslim edilmesi hizmetini sağlar.

İnternet ağ katmanı, datagramdaki alanların yanı sıra uç sistemlerin ve yönlendiricilerin bu alanlar üzerinde nasıl hareket ettiğini tanımlayan ünlü IP protokolünü içerir. Yalnızca bir IP protokolü vardır ve ağ katmanına sahip tüm internet bileşenlerinin IP protokolünü çalıştırması gerekir. İnternetin ağ katmanı aynı zamanda datagramların kaynaklar ve hedefler arasında izlediği yolları belirleyen yönlendirme protokollerini de içerir. İnternetin birçok yönlendirme

protokolü vardır. Ağ katmanı hem IP protokolünü hem de çok sayıda yönlendirme protokolünü içermesine rağmen, genellikle basitçe IP katmanı olarak anılır ve IP'nin İnternet'i birbirine bağlayan yapıştırıcı olduğu gerçeğini yansıtır.

5.1.2.3 Link Layer

İnternet ağ katmanı, bir datagramı kaynak ve hedef arasındaki bir dizi yönlendirici aracılığıyla yönlendirir. Bir paketi rotadaki bir düğümden (ana bilgisayar veya yönlendirici) sonraki düğüme taşımak için ağ katmanı, bağlantı katmanının hizmetlerine güvenir. Özellikle, her düğümde, ağ katmanı datagramı bağlantı katmanına iletir ve bu da datagramı rota boyunca bir sonraki düğüme iletir. Bu sonraki düğümde, bağlantı katmanı datagramı ağ katmanına iletir.

Bağlantı katmanı tarafından sağlanan hizmetler, bağlantı üzerinde kullanılan belirli bağlantı katmanı protokolüne bağlıdır. Örneğin, bazı bağlantı katmanı protokolleri, tek bir bağlantı üzerinden verici düğümden alıcı düğüme kadar güvenilir teslimat sağlar. Bu güvenilir dağıtım hizmetinin, bir uç sistemden diğerine güvenilir teslimat sağlayan TCP'nin güvenilir dağıtım hizmetinden farklı olduğunu unutmayın. Bağlantı katmanı protokollerine örnek olarak Ethernet, WiFi ve kablolu erişim ağının DOCSIS protokolü verilebilir. Datagramların genellikle kaynaktan hedefe seyahat etmek için çeşitli bağlantılardan geçmesi gerektiğinden, bir datagram, rotası boyunca farklı bağlantılarda farklı bağlantı katmanı protokolleri tarafından işlenebilir. Örneğin, bir datagram bir bağlantıda Ethernet tarafından ve bir sonraki bağlantıda PPP tarafından işlenebilir. Ağ katmanı, farklı bağlantı katmanı protokollerinin her birinden farklı bir hizmet alacaktır.

Not => Kitapta bağlantı katmanı paketlerini frames olarak adlandırılmış.

5.1.2.4 Physical Layer

Bağlantı katmanının işi tüm çerçeveleri bir ağ elemanından bitişik ağ elemanına taşımak iken, fiziksel katmanın işi çerçeve içindeki bireysel bitleri bir düğümden diğerine taşımaktır. Bu katmandaki protokoller yine bağlantıya bağımlıdır ve ayrıca bağlantının gerçek iletim ortamına (örneğin, çift bükümlü bakır tel, tek modlu fiber optik) bağlıdır. Örneğin, Ethernet'in birçok fiziksel katman protokolü vardır: biri çift bükümlü bakır tel için, diğeri koaksiyel kablo için, diğeri fiber için vb. Her durumda, bir bit bağlantı boyunca farklı bir şekilde hareket eder.

5.2 Encapsulation

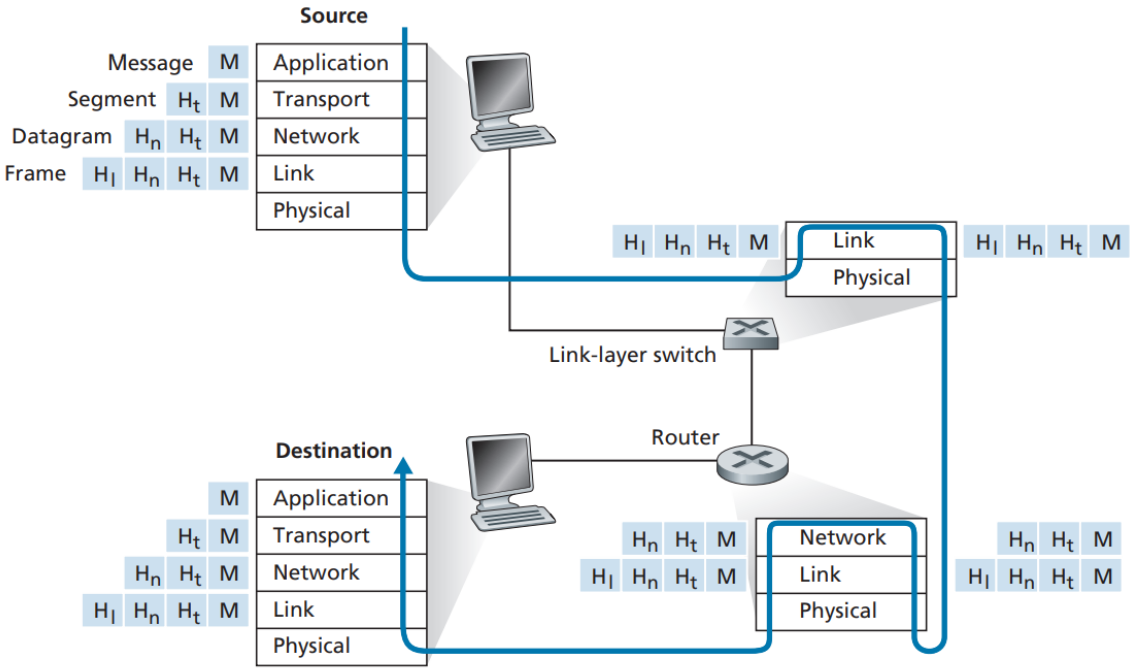


Figure 1.24 ♦ Hosts, routers, and link-layer switches; each contains a different set of layers, reflecting their differences in functionality

Şekil 1.24, verinin gönderen uç sistemin protokol yığınında izlediği, araya giren bağlantı katmanı anahtarı ve yönlendiricinin protokol yığınlarında yukarı ve aşağı ve ardından alıcı uç sistemdeki protokol yığınında yukarı doğru izlediği fiziksel yolu gösterir.

Hem yönlendiriciler hem de bağlantı katmanı anahtarları paket anahtarlardır. Uç sistemlere benzer şekilde yönlendiriciler ve bağlantı katmanı anahtarları (link layer switch), ağ donanımlarını ve yazılımlarını katmanlar halinde düzenler. Ancak yönlendiriciler ve bağlantı katmanı anahtarları, protokol yığınındaki tüm katmanları uygulamaz; genellikle yalnızca alt katmanları uygularlar. Şekil 1.24'te gösterildiği gibi, bağlantı katmanı anahtarları katman 1 ve 2'yi uygular; yönlendiriciler 1'den 3'e kadar olan katmanları uygular. Bu, örneğin, İnternet yönlendiricilerinin IP protokolünü (katman 3 protokolü) uygulayabildiği, ancak bağlantı katmanı anahtarlarının uygulayamadığı anlamına gelir. Bağlantı katmanı anahtarlarının IP adreslerini tanımamasına rağmen Ethernet adresleri gibi katman 2 adreslerini tanıyabilir. Ana bilgisayarların beş katmanın tümünü uygular bu, İnternet mimarisinin karmaşıklığının çoğunu ağın kenarlarına koyduğu görüşüyle tutarlıdır.

Şekil 1.24 aynı zamanda önemli kapsülleme kavramını da göstermektedir. Gönderen ana bilgisayarda, bir uygulama katmanı mesajı (Şekil 1.24'te M) taşıma katmanına iletilir. En basit durumda, taşıma katmanı mesajı alır ve alıcı tarafındaki taşıma katmanı tarafından kullanılacak ek bilgileri (taşıma katmanı başlık bilgisi olarak adlandırılan, Şekil 1.24'te Ht) ekler. Uygulama katmanı mesajı ve taşıma katmanı başlık bilgisi birlikte taşıma katmanı bölümünü oluşturur.

Taşıma katmanı bölümü böylece uygulama katmanı mesajını kapsüller. Eklenen bilgiler, alıcı tarafındaki taşıma katmanının mesajı uygun uygulamaya iletmesine olanak tanıyan bilgileri ve alıcının, mesajdaki bitlerin rotada değiştirilip değiştirilmediğini belirlemesine olanak tanıyan hata tespit bitlerini içerebilir. Daha sonra taşıma katmanı segmenti, kaynak ve hedef uç sistem adresleri gibi ağ katmanı başlık bilgilerini (Şekil 1.24'te Hn) ekleyen ağ katmanına iletir ve bir ağ katmanı datagramı oluşturur.

Datagram daha sonra kendi bağlantı katmanı başlık bilgisini ekleyecek ve bir bağlantı katmanı çerçevesi oluşturacak olan bağlantı katmanına iletilir. Böylece, her katmanda bir paketin iki tür alana sahip olduğunu görüyoruz: başlık alanları (header fields) ve bir yük alanı (payload fields). Yük genellikle bir yukarıdaki katmanın paketidir.

Örneklendirme:
Ofisler arası bir notun bir şirket şubesinden diğerine kamu posta hizmeti aracılığıyla gönderilmesidir. Bir şubede bulunan Alice'in başka bir şubede bulunan Bob'a bir not göndermek istediğini varsayalım. Not, uygulama katmanı mesajına benzer. Alice, notu, zarfın ön tarafında Bob'un adının ve departmanının yazılı olduğu ofisler arası bir zarfa koyar. Ofisler arası zarf, aktarım katmanı segmentine benzer; başlık bilgilerini (Bob'un adı ve departman numarası) içerir ve uygulama katmanı mesajını (not) kapsüller. Gönderen şube posta odası, ofisler arası zarfı aldığında, ofisler arası zarfı, kamu posta servisi aracılığıyla gönderilmeye uygun başka bir zarfın içine koyar. Gönderen posta odası, gönderen ve alan şubelerin posta adresini de posta zarfının üzerine yazar. Burada, posta zarfı datagrama benzer; orijinal mesajı (not) kapsülleyen taşıma katmanı bölümünü (ofisler arası zarf) kapsüller. Posta servisi, posta zarfını alıcı şubenin posta odasına teslim eder. Orada kapsülden arındırma süreci başlıyor. Posta odası ofisler arası notu çıkarır ve Bob'a iletir. Sonunda Bob zarfı açar ve notu çıkarır.

Kapsülleme işlemi yukarıda açıklanandan daha karmaşık olabilir. Örneğin, büyük bir mesaj birden fazla taşıma katmanı segmentine bölünebilir (bunların her biri birden fazla ağ katmanı datagramına bölünebilir). Alıcı uçta, böyle bir segmentin daha sonra kendisini oluşturan datagramlardan yeniden yapılandırılması gerekir.

6.0 Networks Under Attack

Cihazlarımıza girip bulaşabilen kötü amaçlı yazılımlara malware denir. Güvenliği ihlal edilmiş ana makinemiz, toplu olarak botnet olarak bilinen, kötü adamların kontrol ettiği ve hedeflenen ana bilgisayarlara karşı spam e-posta dağıtımı veya dağıtılmış hizmet reddi saldırıları için kullandığı, benzer şekilde güvenliği ihlal edilmiş binlerce cihazdan oluşan bir ağa kayıtlı olabilir.

Günümüzde kötü amaçlı yazılımların çoğu kendi kendini kopyalıyor (self-replicating). Bir ana bilgisayara bulaştığında, o ana bilgisayardan İnternet üzerinden diğer ana bilgisayarlara giriş yapmaya çalışıyor ve yeni virüs bulaşan ana bilgisayarlardan da daha fazla ana bilgisayara giriş yapmaya çalışıyor. Bu şekilde, kendi kendini kopyalayan kötü amaçlı yazılımlar katlanarak hızla yayılabilir.

Başka bir geniş güvenlik tehditleri sınıfı, denial of service (DoS) saldırıları olarak bilinir. DoS saldırısı, bir ağı, ana bilgisayarı veya diğer bir altyapı parçasını meşru kullanıcılar tarafından kullanılamaz hale getirir. Web sunucuları, e-posta sunucuları, DNS sunucularıve kurumsal ağlar, tümü DoS saldırılarına maruz kalabilir. DoS saldırıları 3'e ayrılır:

• Güvenlik Açığı Saldırısı (Vulnerability Attack): Bu, hedeflenen bir ana bilgisayarda çalışan bir açık uygulamaya veya işletim sistemine birkaç iyi hazırlanmış ileti göndermeyi içerir. Doğru paket dizisi bir açık uygulamaya veya işletim sistemine gönderilirse, hizmet durabilir veya daha kötüsü, ana bilgisayar çökebilir.

• Bant Genişliği Taşkını (Bandwidth Flooding): Saldırgan, hedeflenen ana bilgisayara bir sel paketi (deluge) gönderir- o kadar çok paket ki, hedefin erişim bağlantısı tıkanır ve meşru paketler sunucuya ulaşamaz.

Sunucunun bir erişim hızına sahip olduğunu düşünürsek, saldırganın zarar vermek için yaklaşık olarak R bps hızında trafiği göndermesi gerekeceği açıktır. Eğer R çok büyükse, tek bir saldırı kaynağı sunucuya zarar verecek kadar fazla trafik üretemeyebilir. Ayrıca, eğer tüm trafik tek bir kaynaktan geliyorsa, bir yukarı akış yönlendiricisi saldırıyı tespit edebilir ve trafik sunucuya yaklaşmadan önce bu kaynaktan gelen tüm trafikleri engelleyebilir. Şekil 1.25'te gösterilen distributed DoS (DDoS) saldırısında, saldırgan birden fazla kaynağı kontrol eder ve her bir kaynağı hedefe trafiği sıçratmak için kullanır. Bu yaklaşımla, kontrol edilen tüm kaynaklardaki toplam trafik hızının yaklaşık olarak R olması gerekmektedir. Binlerce ele geçirilmiş ana bilgisayar ile botnetleri kullanan DDoS saldırıları bugün yaygın bir olaydır.

Yakındaki kablosuz vericiye pasif bir alıcı yerleştirerek, bu alıcı her gönderilen paketin bir kopyasını alabilir! Bu paketler, şifreler, sosyal güvenlik numaraları, ticari sırlar ve özel kişisel mesajlar da dahil olmak üzere hassas bilgiler içerebilir. Uçan her paketin bir kopyasını kaydeden pasif bir alıcıya bir paket sniffer denir. Sniffer'lar kablolu ortamlarda da kullanılabilir. Birçok Ethernet LAN'ında olduğu gibi kablolu yayın ortamlarında, bir paket sniffer, LAN

üzerinden gönderilen yayın paketlerinin kopyalarını alabilir. Kablo erişim teknolojileri de paket yayını yapar ve bu nedenle sniffing'e karşı savunmasızdır.

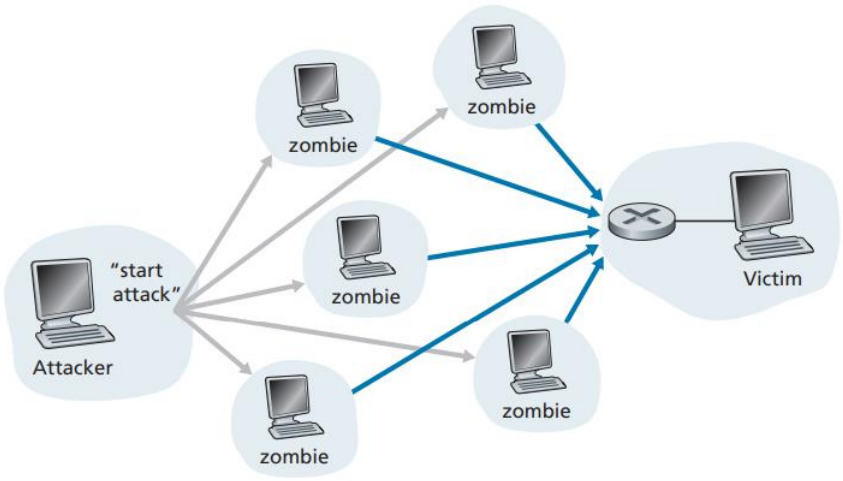


Figure 1.25 ♦ A distributed denial-of-service attack

• Bağlantı taşkını (Connection Flooding): Saldırgan, hedef ana bilgisayarda birçok yarı açık veya tamamen açık TCP bağlantısı kurar. Ana bilgisayar bu sahte bağlantılarla o kadar meşgul olur ki, meşru bağlantıları kabul etmeyi bırakır.

CHAPTER 2

Application Layer

7.0 Network Applications'ın Prensipleri

Ağ uygulama geliştirme sürecinin temelinde, farklı uç sistemlerde çalışan ve ağ üzerinde iletişim kuran programları yazmak yer alır. Örneğin, Web uygulamasında birbirleriyle iletişim kuran iki ayrı program bulunur: kullanıcının bilgisayarında (masaüstü, dizüstü bilgisayar, tablet, akıllı telefon vb.) çalışan tarayıcı programı ve Web sunucusu bilgisayarında çalışan Web sunucu programı. Başka bir örnek olarak, Netflix gibi bir Video On Demand uygulamasında, kullanıcının akıllı telefonunda, tabletinde veya bilgisayarında çalışan bir Netflix sağlayıcısı programı ve Netflix sunucu bilgisayarında çalışan bir Netflix sunucu programı bulunur. Sunucular genellikle (ancak kesinlikle her zaman değil) bir veri merkezinde bulunur (Şekil 2.1). Bu nedenle, yeni uygulamanızı geliştirirken, birden çok uç sistemde çalışacak yazılım yazmanız gerekir. Bu yazılım örneğin, C, Java veya Python gibi dillerde yazılabilir. Önemli olan, ağ ana cihazlarında, yani yönlendiricilerde veya bağlantı katmanı anahtarlarında çalışan yazılım yazmanıza gerek olmamasıdır. Hatta bu tür ağ ana cihazları için uygulama yazılımı yazmak isteseniz bile, bunu yapamazsınız. Şekil 1.24'te gösterildiği gibi, ağ ana cihazları ağ katmanı ve altındaki katmanlarda çalışır, uygulama katmanında değil. Bu temel tasarım, yani uygulama yazılımını uç sistemlerle sınırlama, Şekil 2.1'de gösterildiği gibi, çok çeşitli ağ uygulamalarının hızlı bir şekilde geliştirilmesini ve dağıtılmasını kolaylaştırmıştır.

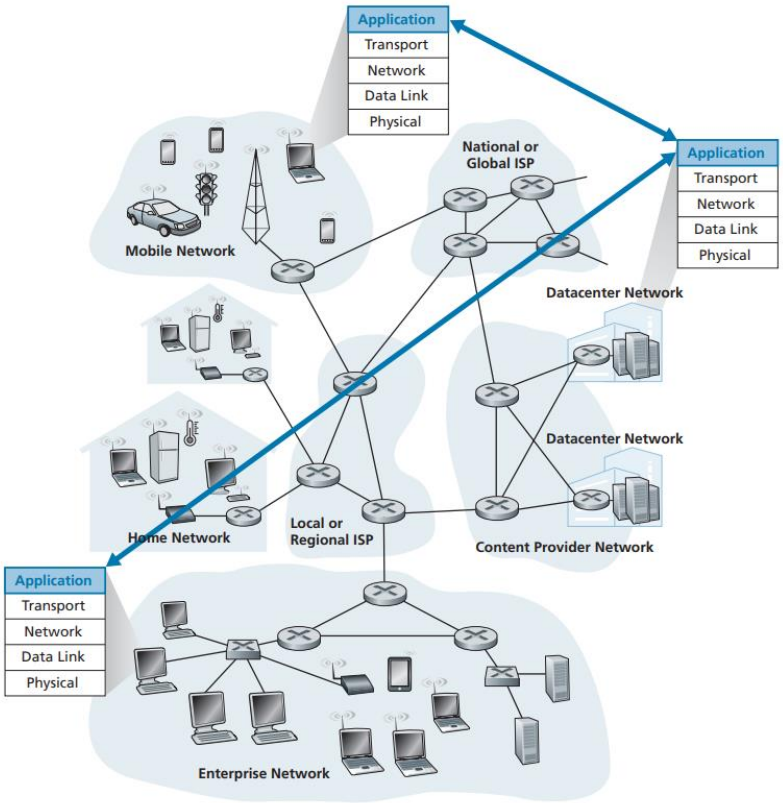


Figure 2.1 ♦ Communication for a network application takes place between end systems at the application layer

7.1

Yazılım kodlamaya başlamadan önce, uygulamanız için geniş bir mimari plana sahip olmanız gerekir. Bir uygulamanın mimarisi, ağ mimarisinden (örneğin, Bölüm 1'de tartışılan beş katmanlı İnternet mimarisinden) kesinlikle farklıdır. Uygulama geliştiricisinin bakış açısından, ağ mimarisi sabittir ve uygulamalara belirli bir hizmet kümesi sağlar. Diğer taraftan, uygulama mimarisi, uygulama geliştirici tarafından tasarlanır ve uygulamanın farklı uç sistemler üzerinde nasıl yapılandırıldığını belirler. Uygulama mimarisini seçerken, bir uygulama geliştiricisi muhtemelen modern ağ uygulamalarında kullanılan iki baskın mimari paradigmadan birine dayanacaktır: client-server (istemci-sunucu) mimarisi veya eşler arası (P2P) mimarisi.

İstemci-sunucu mimarisinde, birçok başka bilgisayar olan istemcilerden gelen isteklere hizmet veren her zaman açık bir ana bilgisayar, yani sunucu bulunur. Klasik bir örnek, her zaman açık bir Web sunucusunun, istemci bilgisayarlarında çalışan tarayıcılardan gelen isteklere hizmet verdiği Web uygulamasıdır. Bir Web sunucusu, bir istemci bilgisayarından bir nesne isteği aldığında, istemci bilgisayarına istenilen nesneyi göndererek yanıt verir. İstemci-sunucu mimarisinde, istemciler doğrudan birbirleriyle iletişim etmezler; örneğin, Web uygulamasında iki tarayıcı doğrudan iletişim kurmaz. İstemci-sunucu mimarisinin diğer bir özelliği, sunucunun sabit, iyi bilinen bir adrese, yani bir IP adresine sahip olmasıdır. Sunucunun sabit, iyi bilinen bir adresi olduğu ve sunucunun her zaman açık olduğu için, bir istemci her zaman sunucuya paket göndererek sunucuya ulaşabilir. İstemci-sunucu mimarisine sahip daha iyi bilinen uygulamalardan bazıları Web, FTP, Telnet ve e-posta içerir. İstemci-sunucu mimarisi Şekil 2.2(a)'de gösterilmiştir. Çoğu zaman, bir istemci-sunucu uygulamasında, tek bir sunucu ana bilgisayarı, tüm istemcilerden gelen tüm istekleri karşılamakta yetersiz kalır. Örneğin, popüler bir sosyal ağ sitesi, tüm isteklerini işleyen yalnızca bir sunucuya sahipse hızla aşırı yük altında kalabilir. Bu nedenle, güçlü bir sanal sunucu oluşturmak için genellikle büyük bir ana bilgisayar grubunu barındıran bir veri merkezi kullanılır.

P2P mimarisinde, veri merkezlerindeki özel sunuculara minimal (veya hiç) güvenilmez. Bunun yerine, uygulama, eşler olarak adlandırılan kesintili olarak bağlı ana bilgisayarlar arasındaki doğrudan iletişimi kullanır. Eşler, hizmet sağlayıcısına ait değildir, ancak kullanıcılar tarafından kontrol edilen masaüstü ve dizüstü bilgisayarlar olup, çoğu eş evlerde, üniversitelerde ve ofislerde bulunur. Eşler, özel bir sunucu üzerinden geçmeden iletişim kurduğu için mimariye eşler arası denir. Popüler bir P2P uygulaması örneği, dosya paylaşım uygulaması BitTorrent'tir. P2P mimarisinin en çekici özelliklerinden biri, **self-scalability (kendiliğinden ölçeklenebilir)** olmasıdır. Örneğin, bir P2P dosya paylaşım uygulamasında, her eş dosya talepleri oluştursa da her eş aynı zamanda dosyaları diğer eşlere dağıtarak sistemde hizmet kapasitesi ekler. P2P mimarileri, genellikle önemli ölçüde sunucu altyapısı ve sunucu bant genişliği gerektirmediği için maliyet etkilidir (veri merkezleriyle karşılaştırıldığında). Ancak, P2P uygulamaları, son derece merkezi olmayan yapıları nedeniyle güvenlik, performans ve güvenilirlik sorunlarıyla karşılaşır

7.1.1 Process Communicating

İşletim sistemleri jargonunda, aslında programlar değil işlemler arasında iletişim sağlanır. Bir işlem, bir son sistem içinde çalışan bir program olarak düşünülebilir. İşlemler aynı son sistemde çalışırken, son sistemlerin işletim sistemi tarafından belirlenen kurallara göre birbirleriyle iletişim kurabilirler. İki farklı son sistemdeki işlemler, bilgisayar ağı üzerinde mesaj alışverişi yaparak birbirleriyle iletişim kurarlar. Gönderme işlemi, mesajları oluşturur ve ağa gönderir; alıcı işlem bu mesajları alır ve belki de yanıt olarak mesaj göndererek cevap verir. Şekil 2.1, birbirleriyle iletişim kuran işlemlerin, beş katmanlı protokol yığınının uygulama katmanında bulunduğunu göstermektedir.

7.1.1.1 Client and Server Processes

Bir ağ uygulaması, birbirleriyle ağ üzerinden mesajlar gönderen işlem çiftlerinden oluşur. Örneğin, web uygulamasında bir istemci tarayıcı işlemi, bir web sunucusu işlemiyle mesaj alışverişi yapar. Bir P2P dosya paylaşım sisteminde, bir dosya bir eşteki bir işlemden diğer bir eşteki bir işleme aktarılır. İletişim kuran her işlem çifti için, genellikle iki işlemden birini istemci olarak etiketleriz ve diğer işlemi sunucu olarak etiketleriz. Web'de, bir tarayıcı bir istemci işlemi olarak etiketlenirken, bir Web sunucusu bir sunucu işlemidir. P2P dosya paylaşımında, dosyayı indiren eş istemci olarak etiketlenir ve dosyayı yükleyen eş sunucu olarak etiketlenir.

Bazı uygulamalarda, örneğin P2P dosya paylaşımında olduğu gibi, bir işlem hem istemci hem de sunucu olabilir. Gerçekten de, P2P dosya paylaşım sistemindeki bir işlem hem dosya yükleme hem de indirme işlemleri yapabilir. Bununla birlikte, herhangi bir iletişim oturumu bağlamında, bir işlem çifti arasındaki herhangi bir iletişim oturumunda bir işlemi istemci olarak ve diğer işlemi sunucu olarak etiketleyebiliriz. İstemci ve sunucu işlemleri şu şekilde tanımlanır:

Bir işlem çifti arasındaki iletişim oturumunun bağlamında, iletişimi başlatan işlem (yani, oturumun başlangıcında diğer işlemle ilk teması kuran işlem) istemci olarak etiketlenir. Oturumu başlatmak için bekleyen işlem sunucudur.

Web'de, bir tarayıcı işlemi bir Web sunucusu işlemi ile teması başlatır; bu nedenle tarayıcı işlemi istemci ve Web sunucusu işlemi sunucu olarak etiketlenir. P2P dosya paylaşımında, Eş A'nın belirli bir dosyayı göndermesini istediği zaman, Eş A, bu belirli iletişim oturumunun bağlamında istemci ve Eş B sunucu olarak etiketlenir. Kafa karışıklığı olmadığında, bazen "uygulamanın istemci ve sunucu tarafı" terimlerini de kullanılır.

The Interface Between the Process and the Computer Network

Bir işlemden diğerine gönderilen herhangi bir ileti, alttaki ağ üzerinden geçmelidir. Bir işlem, bir soket adı verilen bir yazılım arayüzü aracılığıyla ağa mesaj gönderir ve ağdan mesaj alır. Bir işlem, bir eve benzer ve onun soketi, kapısına benzer. Bir işlem, başka bir host'taki başka bir işleme bir mesaj göndermek istediğinde, mesajı kapısından (soket) dışarı iter. Bu gönderme işlemi, mesajın hedef işlemin kapısına taşıyacak bir taşıma altyapısı olduğunu varsayar. Mesaj

hedef ana bilgisayarına ulaştığında, mesaj alım işleminin kapısından (soket) geçer ve alım işlemi daha sonra mesajı işler.

Şekil 2.3, İnternet üzerinden iletişim kuran iki süreç arasındaki soket iletişimini göstermektedir. (Şekil 2.3, işlemler tarafından kullanılan temel aktarım protokolünün İnternet'in TCP protokolü olduğunu varsayar.) Bu şekilde gösterildiği gibi, soket, bir ana bilgisayardaki uygulama katmanı ile aktarım katmanı arasındaki arayüzdür. Soket, ağ uygulamalarının oluşturulduğu programlama arayüzü olduğundan, uygulama ile ağ arasındaki Uygulama Programlama Arayüzü (API) olarak da anılır. Uygulama geliştiricisi, soketin uygulama katmanı tarafındaki her şeyin kontrolüne sahiptir ancak soketin taşıma katmanı tarafında çok az kontrole sahiptir. Uygulama geliştiricisinin taşıma katmanı tarafında sahip olduğu tek kontrol (1) aktarım protokolünün seçimi ve (2) belki de birkaçını düzeltme yeteneğidir. Uygulama geliştiricisi bir aktarım protokolü seçtiğinde (eğer bir seçenek mevcutsa), uygulama bu protokol tarafından sağlanan aktarım katmanı hizmetleri kullanılarak oluşturulur.

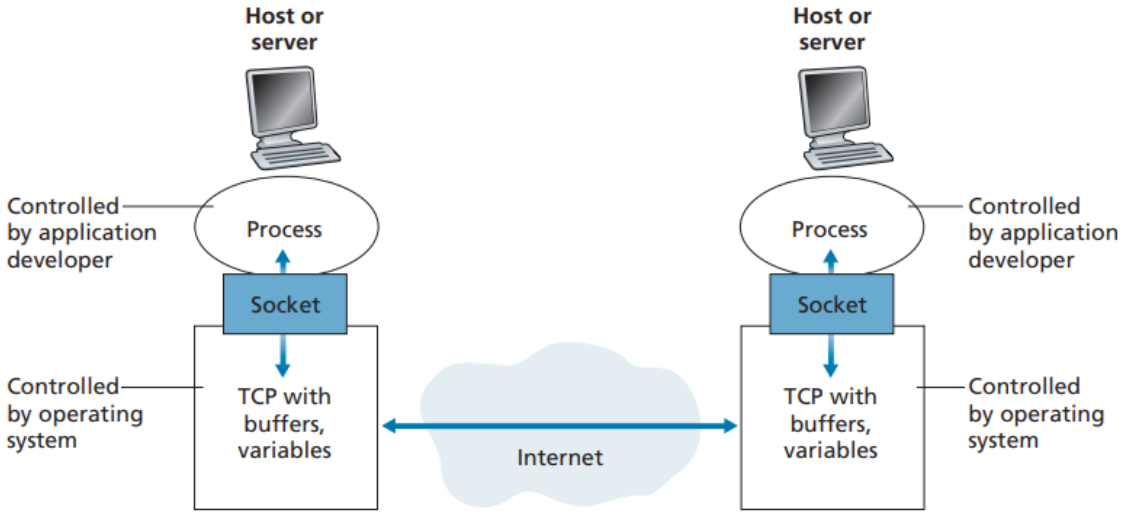


Figure 2.3 ♦ Application processes, sockets, and underlying transport protocol

Addressing Processes

Belirli bir hedefe posta göndermek için, hedefin bir adresi olması gerekir. Benzer şekilde, bir ana bilgisayar üzerinde çalışan bir işlemin, başka bir ana bilgisayar üzerinde çalışan bir işleme paket göndermesi için, alıcı işlemin bir adresi olmalıdır. Alıcı işlemi tanımlamak için iki bilgi parçası belirtilmelidir:

1) Ana bilgisayarın adresi

2) Hedef ana bilgisayarda çalışan alıcı işlemi belirleyen bir tanımlayıcı.

İnternet'te, ana bilgisayar IP adresi ile tanımlanır. Bir IP adresinin ana bilgisayarı benzersiz bir şekilde tanımlayan 32 bitlik bir miktardır. Bir mesajın yönlendirileceği ana bilgisayarın adresini bilmekle kalmaz, gönderen işlem aynı zamanda ana bilgisayarda çalışan alıcı işlemi (daha spesifik olarak, alıcı soket) de tanımlamalıdır. Bu bilgiye ihtiyaç duyulmasının nedeni, genel olarak bir ana bilgisayarın birçok ağ uygulamasını çalıştırabilmesidir. Bir hedef bağlantı noktası numarası bu işe yarar. Popüler uygulamalara özel bağlantı noktası numaraları atanmıştır. Örneğin, bir Web sunucusu bağlantı noktası numarası 80 ile tanımlanır. Bir posta sunucusu işlemi (SMTP protokolünü kullanarak) bağlantı noktası numarası 25 ile tanımlanır.

7.1.2 Transport Services Available to Applications

Gönderen taraftaki uygulama mesajları sokete iletir. Soketin diğer tarafında, taşıma katmanı protokolü, mesajları alma işleminin soketine ulaştırma sorumluluğuna sahiptir. Bir uygulama geliştirdiğinizde mevcut taşıma katmanı protokollerinden birini seçmelisiniz.Bir taşıma katmanı protokolünün onu çağıran uygulamalara sunabileceği hizmetler nelerdir? Olası hizmetleri genel olarak dört boyuta göre sınıflandırılır:

- Güvenilir veri aktarımı
- Verim
- Zamanlama
- Güvenlik

Reliable Data Transfer (Güvenilir veri aktarımı)

Paketler bir bilgisayar ağı içinde kaybolabilir. Örneğin, bir paket, yönlendiricideki arabelleğin dışına taşabilir veya bazı bitleri bozulduktan sonra bir ana bilgisayar veya yönlendirici tarafından atılabilir. Uygulamaların desteklenmesi için uygulamanın bir ucundan gönderilen verinin, uygulamanın diğer ucuna doğru ve eksiksiz olarak iletilmesini garanti altına alacak bir şeyler yapılması gerekmektedir. Bir protokolün böyle garantili bir veri dağıtım hizmeti sağlaması durumunda güvenilir veri aktarımı sağladığı söylenir. Bir taşıma katmanı protokolünün bir uygulamaya potansiyel olarak sağlayabileceği önemli bir hizmet, işlemden işleme güvenilir veri aktarımıdır. Bir aktarım protokolü bu hizmeti sağladığında, gönderme işlemi verilerini sokete aktarabilir ve verilerin alma işlemine hatasız olarak ulaşacağını tam bir güvenle bilir. Bir taşıma katmanı protokolü güvenilir veri iletimi sağlamadığında, gönderme işlemi tarafından gönderilen bazı veriler alıcı işleme asla ulaşmayabilir. Bu, veri kaybına dayanıklı uygulamalar (loss-tolerant applications) için kabul edilebilir olabilir.

Throughput (Verim)

Diğer oturumlar(sessions) ağ yolu boyunca bant genişliğini paylaşacağından ve bu diğer oturumlar gelip gideceğinden, mevcut verim zamanla dalgalanabilir. Bu gözlemler, bir taşıma katmanı protokolünün sağlayabileceği başka bir doğal hizmete, yani belirli bir oranda garantili kullanılabilir çıktıya yol açar. Böyle bir hizmetle, uygulama r bit/sn'lik garantili bir çıktı talep edebilir ve taşıma protokolü daha sonra mevcut çıktının her zaman en az r bit/sn olmasını sağlayacaktır. Böyle garantili bir üretim hizmeti birçok uygulamaya hitap edecektir. Örneğin, bir İnternet telefonu uygulaması sesi 32 kbps hızında kodluyorsa, ağa veri göndermesi ve alıcı uygulamaya bu hızda veri iletmesi gerekir. Aktarım protokolü bu verimi sağlayamazsa, uygulamanın daha düşük bir hızda kodlaması (ve bu düşük kodlama hızını sürdürmek için yeterli verimi alması) gerekecektir veya örneğin ihtiyaç duyulan verimin yarısı alındığından vazgeçmek zorunda kalabilir. Bu İnternet telefon uygulamasının çok az faydası var veya hiç faydası yok. Verim gereksinimleri olan uygulamaların **bant genişliğine duyarlı uygulamalar (bandwidth-sensitive applications)** olduğu söylenir. Mevcut multimedya uygulamalarının çoğu bant genişliğine duyarlıdır, ancak bazı multimedya uygulamaları uyarlanabilir bant genişliği kullanabilir. Bant genişliğine duyarlı uygulamaların belirli aktarım hızı gereksinimleri olsa da **elastik uygulamalar (elastic applications)** mevcut olduğu kadar çok veya az aktarımdan yararlanabilir. Elektronik posta, dosya aktarımı ve web aktarımlarının tümü esnek uygulamalardır.

Timig (Zamanlama)

Veri hızı garantileriyle benzer şekilde, zamanlama garantileri çeşitli biçimlerde olabilir. Bir örnek garanti, gönderenin sokete pompaladığı her bitin, alıcının soketine en fazla 100 ms daha sonra ulaşması olabilir. Bu tür bir hizmet, İnternet telefonu, sanal ortamlar, telekonferans ve çok oyunculu oyunlar gibi etkili olmak için veri teslimatında sıkı zamanlama kısıtlamaları gerektiren etkileşimli gerçek zamanlı uygulamalar için cazip olabilir. Örneğin, İnternet telefonundaki uzun gecikmeler, konuşmada doğal olmayan duraklamalara neden olma eğilimindedir; çok oyunculu bir oyunda veya sanal etkileşimli ortamda, bir eylem almanın ve ortamdan yanıtı görmek arasındaki uzun gecikme (örneğin, bir başka oyuncudan bir end-to-end bağlantısının sonundaki yanıt) uygulamanın daha az gerçekçi hissettirmesine neden olur. Gerçek zamanlı olmayan uygulamalar için, daha düşük gecikme her zaman daha tercih edilir, ancak uçtan uca gecikmelere sıkı kısıtlamalar getirilmez.

Security (Güvenlik)

Bir aktarım protokolü, bir uygulamaya bir veya daha fazla güvenlik hizmeti sağlayabilir. Örneğin, gönderen ana bilgisayarda, bir aktarım protokolü, gönderme işlemi tarafından iletilen tüm verileri şifreleyebilir ve alıcı ana bilgisayarda, taşıma katmanı protokolü, verileri alma işlemine iletmeden önce verilerin şifresini çözebilir. Böyle bir hizmet, veri gönderme ve alma süreçleri arasında bir şekilde gözlemlense bile, iki süreç arasında gizlilik sağlayacaktır. Bir aktarım protokolü, gizliliğin yanı sıra veri bütünlüğü ve uç nokta kimlik doğrulaması da dahil olmak üzere güvenlik hizmetlerini de sağlayabilir.

7.1.3 Transport Services Provided by the Internet

İnternet (ve daha genel olarak TCP/ IP ağları), UDP ve TCP olmak üzere iki aktarım protokolünü uygulamalar için kullanılabilir hale getirir. (Bir uygulama geliştiricisi olarak) için yeni bir ağ uygulaması oluşturduğunuzda UDP veya TCP'yi kullanmak konusunda yapmanız gereken ilk kararlardan biri budur. Her iki protokol de çağrılan uygulamalara farklı bir hizmet seti sunar. Şekil 2.4, bazı seçilmiş uygulamalar için hizmet gereksinimlerini göstermektedir.

Application	Data Loss	Throughput	Time-Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/ Video conferencing	Loss-tolerant	Audio: few kbps–1 Mbps Video: 10 kbps–5 Mbps	Yes: 100s of msec
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of msec
Smartphone messaging	No loss	Elastic	Yes and no

Figure 2.4 ♦ Requirements of selected network applications

7.1.3.1 TCP Services

TCP hizmet modeli, bir bağlantı odaklı hizmet ve güvenilir veri transferi hizmetini içerir. Bir uygulama TCP'yi taşıma protokolü olarak çağırdığında, uygulama TCP'den her iki hizmeti de alır.

- **Bağlantı odaklı hizmet (Connection-oriented service):** TCP, uygulama düzeyindeki mesajların akmaya başlamasından önce istemci ve sunucunun aktarım katmanı kontrol bilgilerini birbirleriyle paylaşmasını sağlar. Bu sözde el sıkışma prosedürü, istemciyi ve sunucuyu uyararak onların bir paket saldırısına hazırlanmalarına olanak tanır. El sıkışma aşamasından sonra, iki işlemin soketleri arasında bir TCP bağlantısının var olduğu söylenir. Bağlantı, iki işlemin aynı anda bağlantı üzerinden birbirlerine mesaj gönderebildiği tam çift yönlü bir bağlantıdır. Uygulama mesaj göndermeyi bitirdiğinde, bağlantıyı sonlandırmalıdır.
- **Güvenilir veri aktarım hizmeti (Reliable data transfer service):** İletişim halindeki işlemler, TCP'ye gönderilen tüm verilerin hatasız ve doğru sırayla iletilmesine güvenebilir. Uygulamanın bir tarafı bir bayt akışını bir sokete aktardığında, TCP'nin aynı bayt akışını alıcı sokete, eksik veya çoğaltılmış baytlar olmadan ileteceğine güvenebilir. TCP ayrıca, doğrudan iletişim halindeki işlemlerin faydası yerine, genel olarak İnternet'in refahı için bir hizmet olarak tıkanıklık kontrol mekanizması da içerir.

TCP tıkanıklık kontrol mekanizması, gönderici ve alıcı arasındaki ağın tıkanık olduğunda bir gönderim işlemini (istemci veya sunucu) sınırlar.

7.1.3.2 UDP Services

UDP, minimal hizmetler sağlayan basit ve hafif bir taşıma protokolüdür. UDP bağlantısızdır, bu nedenle iki işlem iletişime başlamadan önce el sıkışma yoktur. UDP, güvensiz veri transfer hizmeti sağlar; yani bir işlem bir mesajı bir UDP soketine gönderdiğinde, UDP, mesajın alıcı işleme ulaşacağı konusunda hiçbir garanti sağlamaz. Dahası, alıcı işleme ulaşan mesajlar sırasız bir şekilde gelebilir.

Ne TCP ne de UDP herhangi bir şifreleme sağlamaz - gönderen işlemin soketine iletilen veri, ağ üzerindeki hedef işleme giden aynı veridir. Örneğin, gönderen işlem soketine açık metin (yani, şifrelenmemiş) bir şifre gönderirse, açık metin şifreleme yolundaki tüm bağlantılarda seyahat eder ve müdahale eden bağlantılardan herhangi birinde izlenip keşfedilebilir. Gizlilik ve diğer güvenlik konuları birçok uygulama için kritik hale geldiğinden, İnternet topluluğu, TLS olarak adlandırılan bir TCP geliştirmesi geliştirdi. TCP ile güçlendirilmiş-TLS, geleneksel TCP'nin yaptığı her şeyi yapmanın yanı sıra, şifreleme, veri bütünlüğü ve uç nokta kimlik doğrulama da dahil olmak üzere kritik işlem işlemi güvenlik hizmetleri sağlar. Özellikle, bir uygulama TLS hizmetlerini kullanmak istiyorsa, TLS kodunu (mevcut, son derece optimize edilmiş kitaplıklar ve sınıflar) hem istemci hem de sunucu tarafında uygulamaya dahil etmesi gerekir. TLS'nin kendi soket API'si, geleneksel TCP soket API'sine benzer. Bir uygulama TLS kullandığında, gönderen işlem açık metin verileri TLS soketine iletiyor; Gönderen ana bilgisayarında TLS daha sonra verileri şifreler ve şifrelenmiş verileri TCP soketine ileter. Şifrelenmiş veri İnternet üzerinde alıcı işlemin TCP soketine seyahat eder. Alıcı soket şifreli verileri TLS'ye iletir, TLS verileri şifreler. Son olarak, TLS, açık metin verileri TLS soketi aracılığıyla alıcı işleme (process) iletir.

UDP, bir tıkanıklık kontrol mekanizması içermez, bu nedenle UDP'nin gönderme tarafı veriyi alt katmana (ağ katmanına) herhangi bir hızda pompalayabilir. (Ancak, araya giren bağlantıların sınırlı iletim kapasitesi veya tıkanıklık nedeniyle gerçek uçtan uca verimin bu orandan daha düşük olabilir.).

7.1.4 Services Not Provided by Internet Transport Protocols

Günümüz internetinde gecikme aşırı olduğunda veya uçtan uca üretim sınırlı olduğunda akıllı tasarımın sınırlamaları vardır. Özetle, günümüzün İnternet'i zamana duyarlı uygulamalara genellikle tatmin edici hizmet sunabilmektedir ancak herhangi bir zamanlama veya verim garantisi sağlayamaz.

E-postanın, uzaktan terminal erişiminin, Web'in ve dosya aktarımının hepsinin TCP kullanır.Bu uygulamalar öncelikle TCP'yi seçmiştir çünkü TCP güvenilir veri aktarımı sağlar ve sonunda tüm verilerin hedefine ulaşmasını garanti eder. İnternet telefonu uygulamaları (Skype gibi) çoğu zaman bir miktar kaybı tolere edebildiğinden ancak etkili olabilmesi için minimum ücrete ihtiyaç duymasından dolayı, İnternet telefonu uygulamalarının geliştiriciler Genellikle uygulamalarını UDP üzerinden çalıştırmayı tercih ederek TCP'nin tıkanıklık kontrol mekanizmasını ve paket yüklerini atlatırlar. Ancak birçok güvenlik duvarı UDP trafiğini (çoğu tür) engelleyecek şekilde yapılandırıldığından, İnternet telefon uygulamaları genellikle UDP iletişimi başarısız olursa yedek olarak TCP'yi kullanacak şekilde tasarlanmıştır.

7.1.5 Application-Layer Protocols

Ağ işlemleri birbirleriyle mesajları soketlere göndererek iletişirler. Ancak bu mesajlar nasıl yapılandırılır? Mesajdaki çeşitli alanların anlamları nedir? İşlemler ne zaman mesaj gönderir? Bu sorular bizi uygulama katmanı protokollerinin alanına sokar. Bir uygulama katmanı protokolü, farklı uç sistemlerinde çalışan bir uygulamanın işlemlerinin birbirlerine mesaj göndermelerini nasıl belirlediğini tanımlar. Özellikle, bir uygulama katmanı protokolü şunları tanımlar:

- Değiş tokuş edilen mesaj türleri, örneğin, istek mesajları ve yanıt mesajları
- Çeşitli mesaj türlerindeki sözdizimi, yani mesajdaki alanlar ve alanların nasıl belirtildiği
- Alanların anlamsal bilgileri, yani alanlardaki bilgilerin anlamı
- Bir işlemin ne zaman ve nasıl mesaj gönderdiğini ve mesajlara nasıl yanıt verdiğini belirleyen kurallar

Bazı uygulama katmanı protokolleri RFC'lerde belirtilir ve bu nedenle kamuya açıktır. Örneğin, Web'in uygulama katmanı protokolü olan http, bir RFC olarak mevcuttur. Bir tarayıcı geliştirici, HTTP RFC'nin kurallarını takip ederse, tarayıcı, HTTP RFC'nin kurallarını takip eden herhangi bir Web sunucusundan web sayfalarını alabilir. Birçok diğer uygulama katmanı protokolü patentli ve bilerek kamuya açık değildir.

Ağ uygulamaları ile uygulama katmanı protokolleri arasındaki farkı ayırt etmek önemlidir. Bir uygulama katmanı protokolü, bir ağ uygulamasının sadece bir parçasıdır (ancak bizim bakış açımızdan uygulamanın çok önemli bir parçası!). Birkaç örneğe bakalım. Web, kullanıcıların talep üzerine belgeleri Web sunucularından almasına izin veren bir istemci-sunucu uygulamasıdır. Web uygulaması, HTML gibi belge biçimi standartlarını, Web tarayıcılarını

(örneğin, Chrome ve Microsoft Internet Explorer), Web sunucularını (örneğin, Apache ve Microsoft sunucuları) ve bir uygulama katmanı protokolünü içeren birçok bileşenden oluşur. Web'in uygulama katmanı protokolü olan HTTP, tarayıcı ve Web sunucusu arasında değiş tokuş edilen mesajların formatını ve sıralamasını tanımlar. Dolayısıyla, HTTP, Web uygulamasının yalnızca bir parçasıdır (ancak önemli bir parçası).

7.2 The Web and HTTP

7.2.1 Overview of HTTP

HyperMetin Transfer Protokolü (HTTP), Web'in uygulama katmanı protokolünün kalbidir. HTTP, iki programda uygulanır: bir istemci programı ve bir sunucu programı. Farklı uç sistemlerinde çalışan istemci programı ve sunucu programı, HTTP mesajlarını değiş tokuş ederek birbirleriyle iletişim kurarlar. HTTP, bu mesajların yapısını ve istemci ile sunucunun mesajları nasıl değiş tokuş ettiklerini tanımlar.

Bir Web sayfası (aynı zamanda bir belge olarak da adlandırılır), nesnelerden oluşur. Bir nesne, bir URL tarafından adreslenebilen bir dosyadır. Bu dosya, bir HTML dosyası, bir JPEG görüntüsü, bir JavaScript dosyası, bir CSS stil sayfası dosyası veya bir video klibi olabilir. Çoğu Web sayfası, bir **temel HTML dosyası (base HTML file)** ve birkaç referans alınan nesne içerir. Örneğin, bir Web sayfası HTML metni ve beş JPEG görüntüsü içeriyorsa, Web sayfasında altı nesne bulunur. Temel HTML dosyası, sayfadaki diğer nesnelere nesnelerin URL'leri ile referans verir. Her URL'nin iki bileşeni vardır: nesnenin bulunduğu sunucunun ana bilgisayarı ve nesnenin yol adı. Örneğin, URL www.someSchool.edu/someDepartment/picture.gif, www.someSchool.edu için bir ana bilgisayar adı ve /someDepartment/picture.gif için bir yol adı içerebilir.

HTTP, Web istemcilerinin Web sunucularından Web sayfaları istemesini ve sunucuların sayfaları istemcilere aktarmasını tanımlar. İstemci ve sunucu arasındaki etkileşim Bir kullanıcı bir Web sayfası istediğinde (örneğin, bir bağlantıya tıkladığında), tarayıcı sayfadaki nesneler için sunucuya HTTP istek mesajları gönderir. Sunucu istekleri alır ve nesneleri içeren HTTP yanıt mesajlarıyla yanıt verir. Şekil 2.6'da gösterilmiştir. HTTP, altta yatan taşıma protokolü olarak TCP'yi kullanır (UDP'nin üzerinde çalışmaz). HTTP istemcisi önce sunucu ile bir TCP bağlantısı başlatır. Bağlantı kurulduktan sonra, tarayıcı ve sunucu işlemleri TCP'ye soket arayüzleri üzerinden erişirler. İstemci, HTTP istek mesajlarını soket arayüzüne gönderir ve soket arayüzünden HTTP yanıt mesajları alır. Benzer şekilde, HTTP sunucusu istek mesajlarını soket arayüzünden alır ve yanıt mesajlarını soket arayüzüne gönderir. İstemci bir mesajı soket arayüzüne gönderdiğinde, mesaj istemcinin elinden çıkar ve "TCP'nin elinde" olur. TCP HTTP'ye güvenilir veri aktarımı hizmeti sağlar. Bu, bir istemci işlemi tarafından gönderilen her HTTP istek mesajının sonunda sunucuya sağlam bir şekilde ulaştığı anlamına gelir; benzer şekilde, sunucu işlemi tarafından gönderilen her HTTP yanıt mesajı sonunda istemciye sağlam bir şekilde ulaşır. Burada katmanlı bir mimarinin büyük avantajlarından birini görüyoruz HTTP'nin kayıp verilerle veya TCP'nin ağ içindeki veri kaybını veya yeniden sıralamasını nasıl

telafi ettiği ile ilgili ayrıntıları dert etmesi gerekmez. Bu, TCP'nin ve protokol yığınının alt katmanlarındaki protokollerin işidir. Önemli bir nokta olarak, sunucu istemcilere istenen dosyaları gönderirken istemcilerle ilgili herhangi bir durum bilgisini depolamaz. Belirli bir istemcinin birkaç saniye içinde aynı nesneyi iki kez istemesi durumunda, sunucu nesneyi daha önce istemciye hizmet ettiğini belirterek yanıt vermez; bunun yerine, sunucu nesneyi yeniden gönderir, çünkü daha önce yaptıklarını tamamen unutmuştur. Bir HTTP sunucusunun istemciler hakkında herhangi bir bilgiyi saklamaması nedeniyle, HTTP'nin **durumsuz(Stateless)** bir protokol olduğu söylenir.

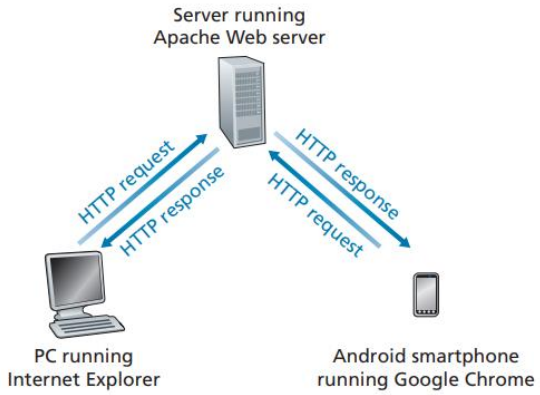


Figure 2.6 ♦ HTTP request-response behavior

7.2.2 Non-Persistent and Persistent Connections (Kalıcı Olmayan ve Kalıcı Bağlantılar)

Birçok İnternet uygulamasında, istemci ve sunucu, istemcinin bir dizi istekte bulunması ve sunucunun isteklerin her birine yanıt vermesiyle uzun bir süre iletişim kurar. Uygulamaya ve uygulamanın kullanım şekline bağlı olarak, isteklerin serisi sırayla, düzenli aralıklarla periyodik olarak veya aralıklı olarak yapılabilir. Bu istemci-sunucu etkileşimi TCP üzerinden gerçekleştiğinde, uygulama geliştiricisi önemli bir karar vermek zorundadır. Her istek/yanıt çifti ayrı bir TCP bağlantısı üzerinden mi gönderilmelidir, yoksa tüm istekler ve bunların karşılık gelen yanıtları aynı TCP bağlantısı üzerinden mi gönderilmelidir? İlk yaklaşımda, uygulama **non-persistent (kalıcı olmayan)** bağlantılar kullanırken, ikinci yaklaşımda **persistent (kalıcı)** bağlantılar kullanılır. HTTP, hem non-persistent bağlantılar hem de persistent bağlantılar kullanabilen bir uygulama olmasına rağmen, varsayılan olarak persistent bağlantıları kullanır. Ancak, HTTP istemcileri ve sunucuları non-persistent bağlantıları kullanmak üzere yapılandırılabilir.

HTTP with Non-Persistent Connections

Kalıcı olmayan bağlantılar için bir Web sayfasını sunucudan istemciye aktarma adımlarını inceleyelim. Sayfanın bir temel HTML dosyası ve 10 JPEG görüntüsünden oluştuğunu ve bu nesnelerin 11'inin tamamının aynı sunucuda bulunduğunu varsayalım. Ayrıca, temel HTML dosyası için URL'nin http://www.someSchool.edu/someDepartment/home.index olduğunu varsayalım. Sırasıyla aşağıdaki durumlar gerçekleşir :

1. HTTP istemci işlemi, varsayılan HTTP bağlantı noktası olan 80 numaralı portta www.someSchool.edu sunucusuna bir TCP bağlantısı başlatır. TCP bağlantısı ile ilişkilendirilmiş bir soket istemci ve sunucuda olacaktır.
2. HTTP istemcisi, soket aracılığıyla sunucuya bir HTTP istek iletişim mesajı gönderir. İstek mesajı, /someDepartment/home.index yol adını içerir.
3. HTTP sunucusu işlemi istek mesajını soketi aracılığıyla alır, /someDepartment/home.index nesnesini deposundan (RAM veya disk) alır, bu nesneyi bir HTTP yanıt mesajına kapsüller ve mesajı soket aracılığıyla istemciye gönderir.
4. HTTP sunucu işlemi, TCP bağlantısını kapatması için TCP'ye bildirim gönderir. (Ancak, TCP bağlantısını gerçekten sonlandırmaz, bağlantının istemcinin yanıt mesajını bütün olarak aldığından emin olana kadar bekler.)
5. HTTP istemcisi yanıt mesajını alır. TCP bağlantısı sona erer. Mesaj, kapsülenmiş nesnenin bir HTML dosyası olduğunu belirtir. İstemci, yanıt mesajından dosyayı çıkarır, HTML dosyasını inceleyerek 10 JPEG nesnesine yönelik referansları bulur.
6. İlk dört adım, her bir referans gösterilen JPEG nesnesi için tekrar edilir.

Tarayıcı Web sayfasını aldıkça, sayfayı kullanıcıya gösterir. İki farklı tarayıcı, bir Web sayfasını kullanıcıya gösterme (yani, yorumlama) konusunda biraz farklı olabilir. HTTP, bir Web sayfasının istemci tarafından nasıl yorumlandığıyla hiçbir ilgisi yoktur. HTTP spesifikasyonları yalnızca istemci HTTP programı ile sunucu HTTP programı arasındaki iletişim protokolünü tanımlar.

Yukarıdaki adımlar, her TCP bağlantısının sunucu nesneyi gönderdikten sonra kapatıldığı non-persistent bağlantıların kullanımını gösterir. HTTP/1.0, non-persistent TCP bağlantılarını kullanır. Her non-persistent TCP bağlantısı tam olarak bir istek mesajı ve bir yanıt mesajı taşır. Bu nedenle, bu örnekte, bir kullanıcı Web sayfasını istediğinde, 11 TCP bağlantısı oluşturulur.

Yukarıda açıklanan adımlarda, istemcinin 10 seri TCP bağlantısı üzerinden 10 JPEG elde edip etmediği veya bazı JPEG'lerin paralel TCP bağlantıları üzerinden elde edilip edilmediği konusunda kasıtlı olarak belirsiz kaldık. Aslında, kullanıcılar bazı tarayıcıları paralellik derecesini kontrol etmek için yapılandırabilirler. Tarayıcılar, çoklu TCP bağlantıları açabilir ve Web sayfasının farklı kısımlarını bu çoklu bağlantılar üzerinden isteyebilirler. Paralel bağlantıların kullanımı yanıt süresini kısaltır.

Devam etmeden önce, istemcinin temel HTML dosyasını istediği andan itibaren dosyanın tamamının istemci tarafından alınmasına kadar geçen süreyi tahmin etmek için bir hesaplama yapalım. Bu amaçla, yuvarlama süresini (RTT) tanımlarız, bu, küçük bir paketin istemciden

sunucuya ve ardından tekrar istemciye kadar gitmesi için gereken süredir. RTT, paket iletim gecikmelerini, ara yönlendiriciler ve anahtarlamalı cihazlardaki paket sıralama gecikmelerini ve paket işleme gecikmelerini içerir. Bir kullanıcının bir bağlantıya tıkladığında ne olacağını düşünelim. Şekil 2.7'de gösterildiği gibi, bu, tarayıcının tarayıcı ve Web sunucusu arasında bir TCP bağlantısı başlatmasına neden olur; bu, bir "**three way handshake** " içerir - istemci küçük bir TCP segmentini sunucuya gönderir, sunucu bunu onaylar ve küçük bir TCP segmentiyle yanıt verir ve son olarak istemci sunucuya geri yanıt verir. El sıkışmanın ilk iki kısmı bir RTT alır. El sıkışmanın ilk iki kısmını tamamladıktan sonra, istemci HTTP istek mesajını ve üçlü el sıkışmanın üçüncü kısmını (onayı) birleştirerek TCP bağlantısına gönderir. İstek mesajı sunucuya ulaştığında, sunucu HTML dosyasını TCP bağlantısına gönderir. Bu HTTP istek/yanıtı başka bir RTT'yi tüketir. Dolayısıyla, yaklaşık olarak, toplam yanıt süresi iki RTT artı sunucuda HTML dosyasının iletim süresidir.

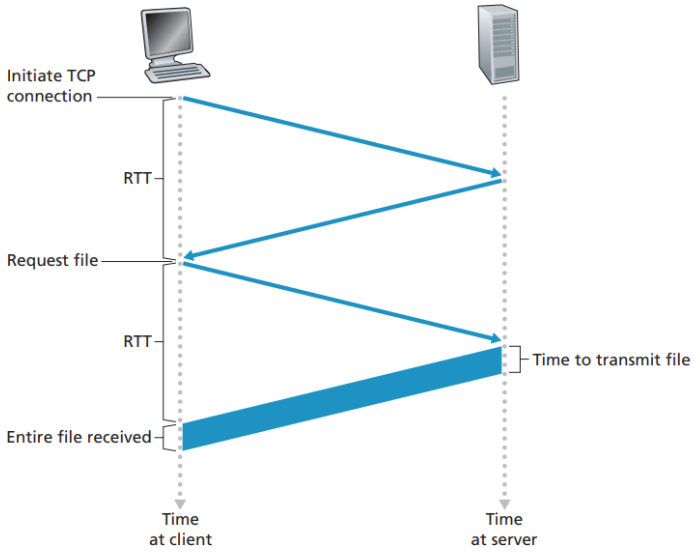


Figure 2.7 ♦ Back-of-the-envelope calculation for the time needed to request and receive an HTML file

HTTP with Persistent Connections

Non-persistent bağlantıların bazı eksiklikleri bulunmaktadır. İlk olarak, her istenilen nesne için yeni bir bağlantı kurulması ve sürdürülmesi gerekir. Bu bağlantılar için TCP önbellekleri tahsis edilmeli ve TCP değişkenleri hem istemci hem de sunucuda tutulmalıdır. Bu durum, aynı anda yüzlerce farklı istemciden gelen istekleri karşılayan Web sunucusuna önemli bir yük getirebilir. İkinci olarak, şu anda açıkladığımız gibi, her nesne iki RTT'lik bir teslimat gecikmesine maruz kalır - bir RTT TCP bağlantısını kurmak ve bir nesneyi istemek ve almak için ve bir RTT. HTTP/1.1 kalıcı bağlantılarla, sunucu yanıt gönderdikten sonra TCP bağlantısını açık bırakır. Aynı istemci ve sunucu arasındaki sonraki istekler ve yanıtlar aynı bağlantı üzerinden gönderilebilir. Özellikle, bir bütün Web sayfası (yukarıdaki örnekte temel HTML dosyası ve 10 resim) tek bir kalıcı TCP bağlantısı üzerinden gönderilebilir. Dahası, aynı sunucuda bulunan birden fazla Web sayfası, sunucudan aynı istemciye tek bir kalıcı TCP bağlantısı üzerinden gönderilebilir. Bu nesneler için yapılan istekler peşi sıra yapılabilir ve bekleyen isteklere yanıt beklenmeden (pipelining) yapılabilir. Tipik olarak, HTTP sunucusu bir bağlantı belirli bir süre

kullanılmadığında kapatır (ayarlanabilir bir zaman aşımı aralığı). Sunucu peş peşe istekler aldığında, nesneleri peş peşe gönderir. HTTP'nin varsayılan modu, pipelining ile kalıcı bağlantıları kullanır.

7.2.3 HTTP Message Format

HTTP özellikleri, HTTP ileti biçimlerinin tanımlarını içerir. İki tür HTTP ileti bulunmaktadır: istek iletileri ve yanıt iletileri, her ikisi de aşağıda açıklanmıştır.

HTTP Request Message

Aşağıda tipik bir HTTP istek mesajı bulunmaktadır.

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

İlk olarak, iletinin sıradan ASCII metin formatında olduğunu görüyoruz, böylece normal bilgisayar okuryazarı insanlar bunu okuyabilir. İkinci olarak, ileti beş satırdan oluşur, her biri bir taşıma dönüşü ve bir satır beslemesi ile sona erer. Son satır ek bir taşıma dönüşü ve satır beslemesi ile sona erer. Bu belirli istek iletisinin beş satırı olsa da, bir istek iletisi çok daha fazla satıra veya bir satıra sahip olabilir. Bir HTTP istek iletisinin ilk satırına **istek satırı (request line)** denir; sonraki satırlara **başlık satırları (header line)** denir. İstek satırının üç alanı vardır: yöntem alanı, URL alanı ve HTTP sürümü alanı. Yöntem alanı GET, POST, HEAD, PUT ve DELETE gibi çeşitli değerler alabilir.

HTTP istek iletilerinin büyük çoğunluğu GET yöntemini kullanır. GET yöntemi, tarayıcı istenen nesneyi URL alanında tanımlanmışken kullanılır. Bu örnekte, tarayıcı /somedir/page.html nesnesini istemektedir. Sürüm açıklayıcıdır; bu örnekte, tarayıcı sürümü HTTP/1.1 olarak uygular. Şimdi örnekteki başlık satırlarına bir göz atalım. Host: www.someschool.edu başlık satırı, nesnenin bulunduğu ana bilgisayar belirtilmiştir. Bu başlık satırının gereksiz olduğunu düşünebilirsiniz, çünkü ana bilgisayara zaten yerinde bir TCP bağlantısı bulunmaktadır. Ancak, Web proxy önbellekleri tarafından gerektirildiğini göreceğiz. Connection: close başlık satırı ekleyerek, tarayıcı sunucuya sürekli bağlantılarla uğraşmak istemediğini belirtir; istenen nesneyi gönderdikten sonra sunucunun bağlantıyı kapatmasını ister. User-agent: başlık satırı kullanıcı ajanını belirtir, yani sunucuya istekte bulunan tarayıcı türüdür. Burada kullanıcı ajanı Mozilla/5.0, bir Firefox tarayıcısıdır. Bu başlık satırı, sunucunun farklı kullanıcı ajanlarına farklı sürümleri aynı nesnenin gönderebilmesi açısından faydalıdır. Son olarak, Accept-language: başlığı, kullanıcının sunucudaki bir Fransızca sürümünü tercih ettiğini belirtir; aksi takdirde, sunucunun varsayılan sürümünü göndermesi gerekir. Accept-language: başlığı, HTTP'de bulunan birçok içerik müzakere başlıklarından sadece biridir.

Bir örneğe baktıktan sonra şimdi Şekil 2.8'de gösterildiği gibi istek mesajının genel formatına bakalım. Genel formatın önceki örneğimizi yakından takip ettiğini görüyoruz. Bununla birlikte,

başlık satırlarından (ve ek satır başı ve satır beslemesinden) sonra bir "varlık gövdesi" bulunduğunu fark etmiş olabilirsiniz. Varlık gövdesi GET yöntemi ile boştur, ancak POST yöntemi ile kullanılır. Bir HTTP istemcisi, genellikle bir kullanıcı bir form doldurduğunda POST yöntemini kullanır - örneğin, bir kullanıcının bir arama motoruna arama kelimelerini sağladığında. POST iletiyle, kullanıcı hala sunucudan bir Web sayfası istemekte, ancak Web sayfasının belirli içeriği, kullanıcının form alanlarına girdiği şeye bağlıdır. Yöntem alanının değeri POST ise, varlık gövdesi, kullanıcının form alanlarına girdiği bilgiyi içerir. Bir form ile oluşturulan bir isteğin zorunlu olarak POST yöntemini kullanması gerekmez. HTML formları genellikle GET yöntemini kullanır ve girilen verileri (form alanlarında) istenen URL'de gönderir. Örneğin, bir form GET yöntemini kullanıyorsa, iki alanı olan bir formu ve iki alan için girilen verileri (maymunlar ve muzlar) içerir, URL'nin yapısı şu şekildedir: www.somesite.com/animalsearch?maymunlar&muzlar. Günlük Web gezintinizde, bu tür genişletilmiş URL'leri muhtemelen fark etmişsinizdir.

HEAD yöntemi GET yöntemine benzer. Sunucu bir HEAD yöntemiyle bir istek aldığında, istenen nesneyi hariç tutarak bir HTTP ileti ile yanıt verir. Uygulama geliştiriciler sıklıkla HEAD yöntemini hata ayıklamada kullanır. PUT yöntemi genellikle Web yayınlama araçlarıyla birlikte kullanılır. Kullanıcıya belirli bir Web sunucusundaki belirli bir yol (dizin) üzerine bir nesne yüklemesine izin verir. PUT yöntemi ayrıca nesneleri Web sunucularına yüklemesi gereken uygulamalar tarafından da kullanılır. DELETE yöntemi bir kullanıcının veya bir uygulamanın bir Web sunucusundaki bir nesneyi silmesine izin verir.

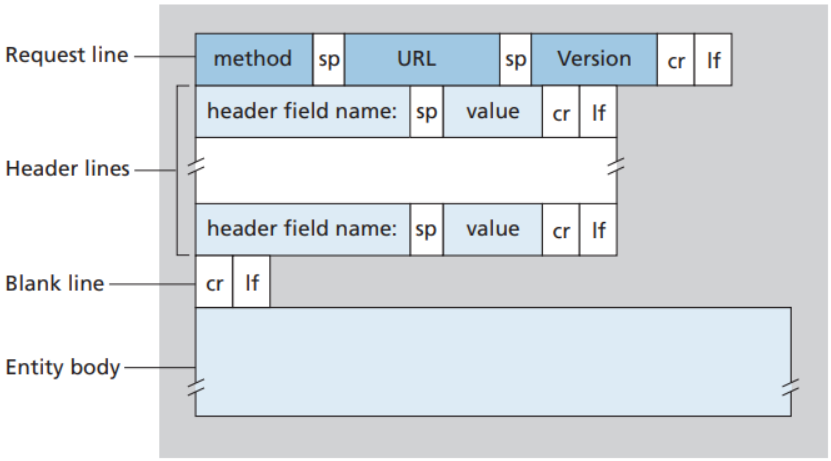


Figure 2.8 ♦ General format of an HTTP request message

HTTP Response Message

Aşağıda tipik bir HTTP istek iletisi bulunmaktadır.

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data ...)
```

Bu yanıt iletisine dikkatlice bir göz atalım. Üç bölümden oluşur: başlangıçta bir durum satırı, altı başlık satırı ve ardından varlık gövdesi. Varlık gövdesi, iletişimin özüdür istenen nesneyi içerir (veri veri veri veri veri ...). Durum satırı üç alan içerir: protokol sürümü alanı, bir durum kodu ve buna karşılık gelen bir durum mesajı. Bu örnekte, durum satırı sunucunun HTTP/1.1 kullandığını ve her şeyin yolunda olduğunu (yani, sunucunun istenen nesneyi bulduğunu ve gönderdiğini) belirtir.

Şimdi başlık satırlarına bakalım. Sunucu, Connection: close başlık satırını kullanarak, mesajı gönderdikten sonra TCP bağlantısını kapatacağını istemciye bildirir. Date: başlık satırı, HTTP yanıtının sunucu tarafından oluşturulduğu ve gönderildiği zamanı ve tarihi gösterir. Unutulmamalıdır ki, bu, nesnenin oluşturulduğu veya son olarak değiştirildiği zaman değil; sunucunun nesneyi dosya sisteminde bulduğu, nesneyi yanıt iletisine yerleştirdiği ve yanıt iletisini gönderdiği zamandır. Server: başlık satırı, mesajın bir Apache Web sunucusu tarafından oluşturulduğunu belirtir; bu, HTTP istek iletisindeki User-agent: başlık satırına benzerdir. Last-Modified: başlık satırı, nesnenin oluşturulduğu veya son olarak değiştirildiği zamanı ve tarihi belirtir. Last-Modified başlığı, yerel istemci ve ağ önbellek sunucularında (ayrıca proxy sunucuları olarak da bilinir) nesne önbellekleme için kritiktir. Content-Length : başlık satırı, gönderilen nesnedeki bayt sayısını belirtir. Content-Type: başlık satırı, varlık gövdesindeki nesnenin HTML metni olduğunu belirtir. (Nesne türü resmi olarak Content-Type: başlığı ile belirtilir ve dosya uzantısı ile değil.)

Bir örneğe baktıktan sonra, şimdi bir yanıt iletisinin genel formatına bakalım, bu da önceki örnekteki bir yanıt iletisinin formatıdır. Bir örneğe baktıktan sonra şimdi Şekil 2.9'da gösterilen yanıt mesajının genel formatını inceleyelim. Yanıt mesajının bu genel formatı, önceki yanıt mesajı örneğiyle eşleşir. Durum kodları ve bunların cümleleri hakkında birkaç kelime daha söyleyelim. Durum kodu ve ilgili ifade, isteğin sonucunu gösterir. Bazı yaygın durum kodları ve ilgili ifadeler şunları içerir:

• 200 OK: İstek başarılı oldu ve bilgi yanıtta döndürülür.

• 301 Moved Permanently: İstenen nesne kalıcı olarak taşındı; yeni URL, yanıt iletisinin Location: başlığında belirtilir. İstemci yazılımı otomatik olarak yeni URL'yi alacak.

• 400 Bad Request: Bu, sunucunun anlayamadığı bir genel hata kodudur.

• 404 Not Found: İstenen belge bu sunucuda mevcut değil.

• 505 HTTP Version Not Supported: Sunucu tarafından desteklenmeyen istenen HTTP protokol sürümü.

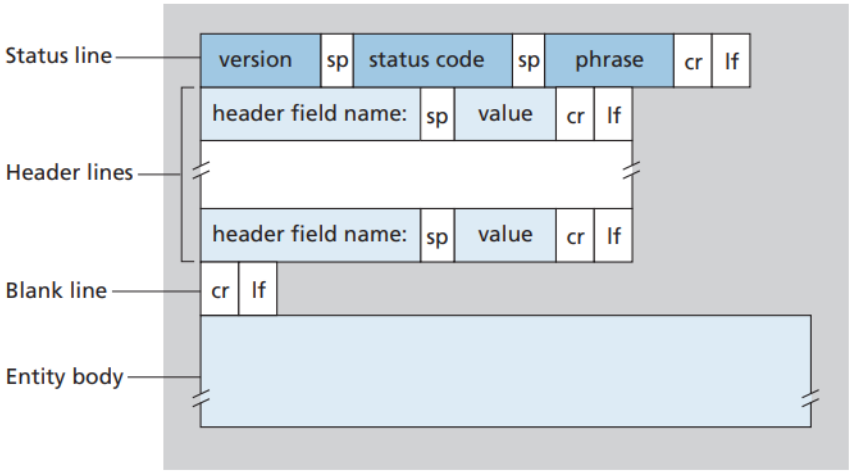


Figure 2.9 ♦ General format of an HTTP response message

Bir tarayıcı, hangi başlık satırlarını bir istek iletişine dahil edeceğine nasıl karar verir? Bir Web sunucusu, bir yanıt iletişine hangi başlık satırlarının dahil edileceğine nasıl karar verir? Bir tarayıcı, başlık satırlarını tarayıcı türü ve sürümü, tarayıcının kullanıcı yapılandırması ve tarayıcının şu anda önbellekte tutulan, ancak muhtemelen güncel olmayan nesnenin bir sürümüne sahip olup olmadığı gibi faktörlere bağlı olarak oluşturur. Web sunucuları benzer şekilde davranır: Farklı ürünler, sürümler ve yapılandırmalar, hangi başlık satırlarının yanıt iletilerine dahil edileceğini etkiler.

7.2.4 User-Server Interaction: Cookies

Yukarıda bir HTTP sunucusunun durum bilgisi olmayan bir sunucu olduğundan bahsetmiştik. Bu, sunucu tasarımını basitleştirir ve mühendislerin binlerce eşzamanlı TCP bağlantısını yönetebilecek yüksek performanslı Web sunucuları geliştirmelerine olanak tanır. Ancak, bir Web sitesinin kullanıcıları tanımlaması sıklıkla istenir, ya çünkü sunucu kullanıcı erişimini sınırlamak ister ya da içeriği kullanıcı kimliği olarak işlev olarak sunmak istemektedir. Bu amaçlar için HTTP **çerezleri** kullanır. Çerezler, sitelerin kullanıcıları takip etmesine izin verir.

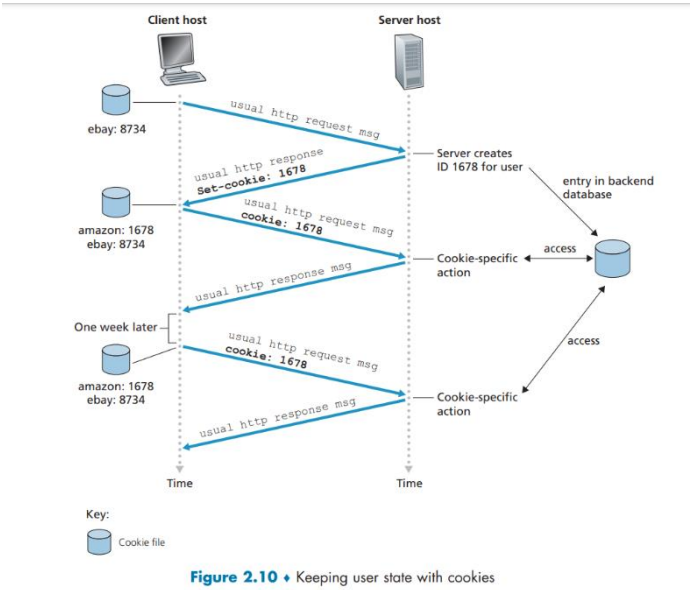


Figure 2.10 ♦ Keeping user state with cookies

Şekil 2.10'da gösterildiği gibi, çerez teknolojisinin dört bileşeni bulunmaktadır:

1. HTTP yanıt iletilerinde bir çerez başlık satırı
2. HTTP istek iletilerinde bir çerez başlık satırı
3. Kullanıcının uç sistemde tutulan ve kullanıcının tarayıcısı tarafından yönetilen bir çerez dosyası
4. Web sitesindeki bir arka uç veri tabanı.

Diyelim ki her zaman ev PC'sinden Internet Explorer kullanarak Web'e erişen Susan, ilk kez Amazon.com ile iletişime geçiyor. Geçmişte eBay sitesini ziyaret ettiğini varsayalım. İstek Amazon Web sunucusuna ulaştığında, sunucu benzersiz bir kimlik numarası oluşturur ve bu numarayla dizinlenmiş bir giriş oluşturur. Amazon Web sunucusu daha sonra Susan'ın tarayıcısına yanıt verirken, HTTP yanıtında bir Set-cookie: başlık satırı ekler ve bu başlık satırı kimlik numarasını içerir. Örneğin, başlık satırı şöyle olabilir:

Set-cookie: 1678

Susan'ın tarayıcısı HTTP yanıt iletilerini aldığında, Set-cookie: başlığını görür. Tarayıcı daha sonra yönettiği özel çerez dosyasına bir satır ekler. Bu satır, sunucunun ana bilgisayar adını ve Set-cookie: başlığındaki kimlik numarasını içerir. Susan daha önce bu siteyi ziyaret ettiği için çerez dosyasında zaten bir giriş bulunmaktadır. Susan Amazon sitesinde gezinmeye devam ettikçe, her bir web sayfası isteği yaptığında, tarayıcısı çerez dosyasına danışır, bu site için kimlik numarasını çıkarır ve HTTP isteğinde kimlik numarasını içeren bir çerez başlık satırını ekler. Özellikle, Amazon sunucusuna yaptığı her HTTP isteği, aşağıdaki başlık satırını içerir:

Cookie: 1678

Bu şekilde, Amazon sunucusu Susan'ın Amazon sitesindeki etkinliklerini takip edebilir. Amazon Web sitesinin Susan'ın adını kesin olarak bilmese de, kullanıcı 1678'in hangi sayfaları ziyaret ettiğini, hangi sırayla ve hangi saatlerde ziyaret ettiğini tam olarak bilir! Amazon, alışveriş sepeti hizmetini sağlamak için çerezleri kullanır - Amazon, Susan'ın niyet ettiği tüm satın alımların bir listesini tutabilir, böylece oturumun sonunda bunları toplu olarak ödeyebilir.

Eğer Susan bir hafta sonra Amazon'un sitesine geri dönerse, tarayıcısı istek iletilerinde hala Cookie: 1678 başlık satırını eklemeye devam eder. Amazon, Susan'a daha önce Amazon'da ziyaret ettiği web sayfalarına dayanarak ürün önerileri sunar. Susan ayrıca Amazon'a kayıt olursa tam adını, e-posta adresini, posta adresini ve kredi kartı bilgilerini sağlarsa - Amazon bu bilgileri veritabanına ekleyebilir ve böylece Susan'ın adını kimlik numarasıyla ilişkilendirebilir (ve daha önce siteyi ziyaret ettiği tüm sayfalar!). Amazon ve diğer e-ticaret siteleri "tek tıklamalı alışveriş" sunar. Susan bir sonraki ziyaretinde bir ürün satın almaya karar verdiğinde, adını, kredi kartı numarasını veya adresini yeniden girmesi gerekmez.

Bir kullanıcı bir siteyi ilk ziyaret ettiğinde, kullanıcı bir kullanıcı kimliği (muhtemelen adı veya soyadı) sağlayabilir. Sonraki oturumlarda, tarayıcı bir çerez başlığı geçirerek kullanıcıyı sunucuya tanıtır. Bu nedenle, çerezler, stateless HTTP'nin üstüne bir kullanıcı oturum katmanı oluşturmak için kullanılabilir. Örneğin, bir kullanıcı Web tabanlı bir e-posta uygulamasına (örneğin, Hotmail) giriş yaptığında, tarayıcı çerez bilgilerini sunucuya ileterek sunucunun kullanıcıyı uygulama ile oturum boyunca tanımlamasına izin verir.

7.2.5 Web Caching

Proxy sunucusu olarak da adlandırılan Web cache, kaynak Web sunucusu adına HTTP isteklerini karşılayan bir ağ varlığıdır. Web önbelleğinin kendi disk deposu vardır ve en son istenen nesnelerin kopyalarını bu depoda tutar. Şekil 2.11'de gösterildiği gibi, bir kullanıcının tarayıcısı, kullanıcının tüm HTTP isteklerinin ilk olarak Web önbelleğine yönlendirileceği şekilde yapılandırılabilir. Bir tarayıcı yapılandırıldıktan sonra, bir nesneye yönelik her tarayıcı isteği ilk olarak Web önbelleğine yönlendirilir. Örneğin, bir tarayıcının http://www.someSchool.edu/campus.gif nesnesini istediğini varsayalım. Sırasıyla aşağıdaki durumlar gerçekleşir :

1. Tarayıcı, web önbelleğine bir TCP bağlantısı kurar ve nesne için bir HTTP isteği gönderir.
2. Web önbelleği, nesnenin yerel olarak saklanmış bir kopyasının olup olmadığını kontrol eder. Eğer varsa, Web önbelleği nesneyi bir HTTP yanıt ile istemci tarayıcısına gönderir.
3. Web önbelleğinde nesne yoksa, web önbelleği kaynak sunucuya, yani www.someschool.edu'ya bir TCP bağlantısı açar. Web önbelleği daha sonra nesne için önbellekten sunucuya TCP bağlantısına bir HTTP isteği gönderir. Bu isteği aldıktan sonra, kaynak sunucu nesneyi bir HTTP yanıtı içinde Web önbelleğine gönderir.
4. Web önbelleği nesneyi aldığında, bir kopyasını yerel depolamasına kaydeder ve bir HTTP yanıtı içinde, istemci tarayıcısına (istemci tarayıcı ile Web önbelleği arasındaki mevcut TCP bağlantısı üzerinden) bir kopya gönderir.

Web cache, aynı anda hem bir sunucu hem de bir istemci olarak görev yapar. Bir tarayıcıdan gelen istekleri alıp yanıtlar gönderdiğinde bir sunucu olurken, bir kaynak sunucuya istekler gönderip yanıtlar aldığında bir istemci olur.

Genellikle bir İnternet Servis Sağlayıcısı (ISS) tarafından satın alınır ve kurulur. Örneğin, bir üniversite, kampüs ağına bir önbellek kurabilir ve tüm kampüs tarayıcılarını önbelleğe yönlendirebilir. Veya büyük bir konut ISS'si (örneğin, Comcast) ağına bir veya daha fazla önbellek kurabilir ve gönderilen tarayıcıları kurulu önbelleklere yönlendirebilir.

Web önbelleklemesi, İnternet'te iki nedenle yaygın olarak kullanılmaktadır. İlk olarak, bir Web önbelleği, özellikle istemci isteği ve köken sunucu arasındaki darboğaz bant genişliği, istemci ile önbellek arasındaki darboğaz bant genişliğinden çok daha az olduğunda, bir istemci isteğinin yanıt süresini önemli ölçüde azaltabilir. İstemci ile önbellek arasında genellikle yüksek hızlı bir bağlantı olduğu ve önbelleğin istenen nesneye sahip olduğu durumda, önbellek nesneyi hızlı bir şekilde istemciye teslim edebilir. İkinci olarak, bir kurumun (örneğin, bir şirket veya bir üniversite) İnternet'e erişim bağlantısındaki trafiği önemli ölçüde azaltabilirler. Trafik azaltıldığında, kurumun bant genişliğini daha hızlı bir şekilde yükseltmesi gerekmez, bu da maliyetleri azaltır. Ayrıca, Web önbelleklemesi, İnternet genelinde Web trafiğini önemli ölçüde azaltabilir ve bu da tüm uygulamalar için performansı artırır.

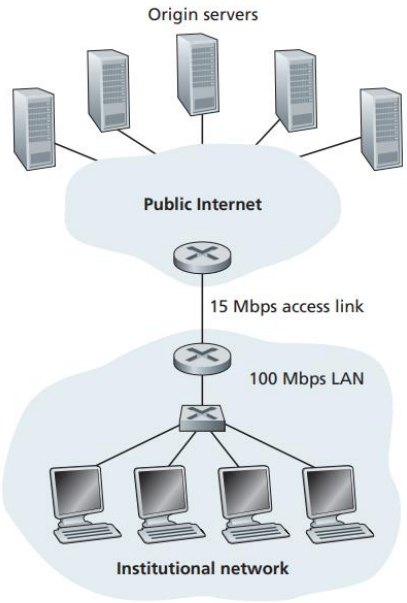


Figure 2.12 ♦ Bottleneck between an institutional network and the Internet

Önbelleklerin faydalarını daha derinlemesine anlamak için, Şekil 2.12 bağlamında bir örneği ele alalım. Bu şekil, kurumsal ağı ve genel İnternet'i göstermektedir. Kurumsal ağ, yüksek hızlı bir yerel ağdır. Kurumsal ağdaki bir yönlendirici ile İnternet'teki bir yönlendirici, 15 Mbps'lik bir bağlantı ile bağlanmıştır. Köken sunucuları İnternet'e bağlıdır ancak dünyanın dört bir yanına dağılmışlardır. Ortalama nesne boyutunun 1 Mbit ve kurumun tarayıcılarından köken sunucularına ortalama istek oranının saniyede 15 istek olduğunu varsayalım. HTTP istek mesajlarının önemsiz derecede küçük olduğunu ve bu nedenle ağlarda veya erişim bağlantısında (kurumsal yönlendiriciden İnternet yönlendiricisine) trafik oluşturmadığını varsayalım. Ayrıca, Şekil 2.12'deki erişim bağlantısının İnternet tarafındaki yönlendiriciden bir HTTP isteğini (bir IP veri gramı içinde) ilettiği ve yanıtı (genellikle birçok IP veri gramı içinde) aldığı sürenin ortalama iki saniye olduğunu varsayalım. Gayri resmi olarak, bu son gecikmeyi "İnternet gecikmesi" olarak adlandırırız.

Toplam yanıt süresi - yani bir nesnenin istemci tarafından talep edilmesinden alınmasına kadar geçen süre - LAN gecikmesi, erişim gecikmesi (yani iki yönlendirici arasındaki gecikme) ve İnternet gecikmesi toplamından oluşur. Şimdi bu gecikmeyi tahmin etmek için çok basit bir hesaplama yapalım.

LAN üzerindeki trafik yoğunluğu

$$(15 \text{ requests/sec}) \cdot (1 \text{ Mbits/request})/(100 \text{ Mbps}) = 0.15$$

Erişim bağlantısındaki trafik yoğunluğu (İnternet yönlendiricisinden kurum yönlendiricisine) ise

$$(15 \text{ requests/sec}) \# (1 \text{ Mbits/request})/(15 \text{ Mbps}) = 1$$

Bir LAN'da 0.15 trafik yoğunluğu tipik olarak en fazla onlarca milisaniyelik gecikmelere neden olur; bu nedenle, LAN gecikmesini ihmal edebiliriz. Trafik yoğunluğu 1'e yaklaştıkça (Şekil 2.12'deki erişim bağlantısının durumunda olduğu gibi), bir bağlantıdaki gecikme çok büyük hale gelir ve sınırsız bir şekilde artar. Bu nedenle, istekleri karşılamak için ortalama yanıt süresi dakikalar düzeyinde olacak ve kurumun kullanıcıları için kabul edilemez olacaktır. Bir olası çözüm, erişim hızını 15 Mbps'den 100 Mbps'e çıkarmaktır. Bu, erişim bağlantısındaki trafik yoğunluğunu 0.15'e düşürür, bu da iki yönlendirici arasındaki ihmal edilebilir gecikmelere karşılık gelir. Bu durumda, toplam yanıt süresi yaklaşık olarak iki saniye olacaktır, yani İnternet gecikmesi. Ancak, bu çözüm aynı zamanda kurumun erişim bağlantısını 15 Mbps'den 100 Mbps'e yükseltmesi anlamına gelir, ki bu da maliyetli bir tekliftir.

Şimdi, erişim bağlantısını yükseltmek yerine kurumsal ağa bir Web önbelleği kurmak alternatif bir çözümü düşünelim. Bu çözüm Şekil 2.13'te gösterilmiştir. Önbelleğin karşıladığı isteklerin fraksiyonu olan vuruş oranları, pratikte genellikle 0.2 ile 0,7 arasında değişir. İllüstratif amaçlarla, önbelleğin bu kurum için %0,4'lük bir vuruş oranı sağladığını varsayalım. İstemciler

ve önbellek aynı yüksek hızlı LAN'a bağlı olduğundan, isteklerin %40'ı, önbellek tarafından neredeyse hemen, yani 10 milisaniye içinde karşılanır. Bununla birlikte, kalan %60'lık istekler hala köken sunucular tarafından karşılanması gerekmektedir. Ancak, istenen nesnelerin yalnızca %60'ı erişim bağlantısından geçtiğinde, erişim bağlantısındaki trafik yoğunluğu 1.0'den 0.6'ya düşer. Genellikle, 0.8'den küçük bir trafik yoğunluğu, 15 Mbps'lik bir bağlantıda küçük gecikmelere, örneğin onlarca milisaniye, karşılık gelir. Bu gecikme, iki saniyelik İnternet gecikmesine kıyasla ihmal edilebilir. Bu hususlar göz önüne alındığında, ortalama gecikme dolayısıyla

$$0.4x(0.01 \text{ seconds}) + 0.6x(2.01 \text{ seconds})$$

bu da 1,2 saniyeden biraz daha fazladır.

Bu ikinci çözüm, birinci çözümden daha düşük bir yanıt süresi sağlar ve kurumun internet bağlantısını yükseltmesini gerektirmez. Kurumun elbette bir web önbellegi satın alıp kurması gerekir. Ancak bu maliyet düşüktür birçok ön bellek, ucuz PC'lerde çalışan genel alandaki yazılımları kullanır.

Content Distribution Networks (İçerik Dağıtım Ağları (CDN'ler))kullanılarak, Web ön belleklerinin İnternet'te giderek önemli bir rol oynamasıdır. Bir CDN şirketi, İnternet üzerinde coğrafi olarak dağıtılmış birçok ön bellek kurar ve bu şekilde trafiğin büyük bir kısmını yerelleştirir. Paylaşılan CDN'ler (örneğin Akamai ve Limelight) ve özel CDN'ler (örneğin Google ve Netflix) bulunmaktadır.

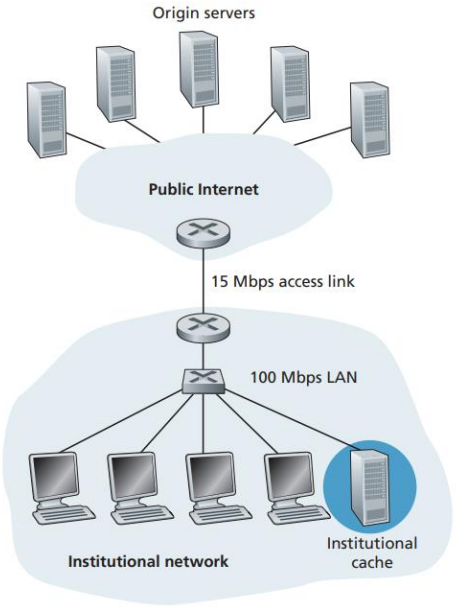


Figure 2.13 ♦ Adding a cache to the institutional network

The Conditional GET (Koşullu GET)

Özellikle web önbelleği kullanıcı tarafından algılanan yanıt sürelerini azaltabilirken, yeni bir sorun ortaya çıkarır önbellekteki bir nesnenin kopyası eski olabilir. Başka bir deyişle, Web sunucusunda barındırılan nesne, istemcide önbelleğe alındığından beri değiştirilmiş olabilir. HTTP'nin önbelleğin nesnelerinin güncel olup olmadığını doğrulamaya izin veren bir mekanizması vardır. Bu mekanizma, **koşullu GET** olarak adlandırılır. Bir HTTP istek mesajı, GET yöntemi kullanılarak (1) ve **If-Modified-Since:** başlık satırını içeriyorsa (2), söz konusu koşullu GET mesajı olarak adlandırılır.

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
```

İkinci olarak, Web sunucusu istenen nesneyi içeren bir yanıt mesajını önbelleğe gönderir:

```
HTTP/1.1 200 OK
Date: Sat, 3 Oct 2015 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 9 Sep 2015 09:23:24
Content-Type: image/gif

(data data data data data ...)
```

Önbellek, nesneyi istekte bulunan tarayıcıya iletir ancak aynı zamanda nesneyi yerel olarak önbelleğe alır. Daha da önemlisi, önbellek nesneyle birlikte son değiştirilme tarihini de saklar. Üçüncüsü, bir hafta sonra, başka bir tarayıcı aynı nesneyi önbellek aracılığıyla ister ve nesne hâlâ önbellektedir. Bu nesne geçen hafta Web sunucusunda değiştirilmiş olabileceğinden, önbellek koşullu bir GET vererek güncellik kontrolü gerçekleştirir. Özellikle önbellek şunları gönderir:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 9 Sep 2015 09:23:24
```

If-modified-since: başlık satırının değerinin, sunucu tarafından bir hafta önce gönderilen LastModified: başlık satırının değerine tam olarak eşit olduğunu unutmayın. Bu koşullu GET, sunucuya nesneyi yalnızca nesne belirtilen tarihten bu yana değiştirilmişse göndermesini söyler. Nesnenin 9 Eylül 2015 09:23:24 tarihinden bu yana değiştirilmediğini varsayalım. Daha sonra dördüncü olarak Web sunucusu önbelleğe bir yanıt mesajı gönderir:

```
HTTP/1.1 304 Not Modified
Date: Sat, 10 Oct 2015 15:39:29
Server: Apache/1.3.0 (Unix)

(empty entity body)
```

Koşullu GET'e yanıt olarak Web sunucusunun hala bir yanıt mesajı gönderdiğini ancak istenen nesneyi yanıt mesajına dahil etmediğini görüyoruz.

Koşullu GET'e yanıt olarak, Web sunucusu hala bir yanıt mesajı gönderir ancak istenen nesneyi yanıt mesajına dahil etmez. İstenen nesnenin yanıt mesajına dahil edilmesi sadece bant genişliğini israf eder ve kullanıcı tarafından algılanan yanıt süresini artırır, özellikle nesne büyükse. Son yanıt mesajının durum satırında 304 Not Modified olduğuna dikkat edin; bu, önbelleğin (proxy önbelleğinin) kendi önbelleğindeki nesneyi isteyen tarayıcıya iletebileceğini belirtir.

7.2.6 HTTP/2

HTTP/2'nin temel hedefleri, tek bir TCP bağlantısı üzerinden istek ve yanıt çoğullama, istek önceliklendirme ve sunucu itme, ve HTTP başlık alanlarının etkin sıkıştırılması yoluyla algılanan gecikmeyi azaltmaktır. HTTP/2, HTTP yöntemlerini, durum kodlarını, URL'leri veya başlık alanlarını değiştirmez. Bunun yerine, HTTP/2, verinin nasıl biçimlendirildiğini ve istemci ile sunucu arasında nasıl taşındığını değiştirir.

HTTP/2'nin gerekliliğini motive etmek için, HTTP/1.1'in sürekli TCP bağlantıları kullandığını hatırlayın, böylece bir web sayfası tek bir TCP bağlantısı üzerinden sunucudan istemciye gönderilir. Her bir web sayfası için sadece bir TCP bağlantısı olduğunda, sunucuda soket sayısı azalır ve her taşınan web sayfası ağ bant genişliğinin adil bir payını alır. Ancak geliştiriciler, bir web sayfasındaki tüm nesneleri tek bir TCP bağlantısı üzerinden göndermenin bir a Head of Line Blocking (Kuyruk Başı (HOL) engelleme) sorunu olduğunu keşfettiler.

HOL engellemeyi anlamak için, HTML taban sayfası, web sayfasının üst kısmında büyük bir video klibi ve video klibinin altında birçok küçük nesneyi içeren bir web sayfasını düşünün. Ayrıca, sunucu ve istemci arasındaki yol üzerinde düşük-orta hızda bir darboğaz bağlantısı (örneğin, düşük hızlı kablosuz bağlantı) olduğunu varsayalım. Tek bir TCP bağlantısı kullanarak, video klibi darboğaz bağlantısı üzerinden geçerken uzun bir süre alırken, küçük nesneler video klibinin arkasında bekleyerek gecikir; yani, kuyruğun başındaki video klibi, arkasındaki küçük nesneleri engeller. HTTP/1.1 tarayıcıları genellikle bu sorunu çözmek için birden fazla paralel TCP bağlantısı açarak çalışır, böylece aynı web sayfasındaki nesneleri tarayıcıya paralel olarak gönderirler. Bu şekilde, küçük nesneler tarayıcıya daha hızlı ulaşabilir ve tarayıcıda daha hızlı bir şekilde işlenir, böylece kullanıcı tarafından algılanan gecikme azalır.

TCP tıkanıklık kontrolü, tarayıcılara tek bir kalıcı bağlantı yerine birden fazla paralel TCP bağlantısı kullanma istenmeyen bir teşvik sağlar. Çok kabaca konuşursak, TCP tıkanıklık kontrolü, bir darboğaz bağlantısını paylaşan her TCP bağlantısına bu bağlantının mevcut bant genişliğinin eşit bir payını vermek ister; bu nedenle, bir darboğaz üzerinde çalışan n TCP bağlantısı varsa, her bağlantı yaklaşık olarak bant genişliğinin 1/n'ini alır. Bir web sayfasını taşımak için birden fazla paralel TCP bağlantısı açarak, tarayıcı bağlantıların bağlantı bant

genişliğinin daha büyük bir kısmını "aldatabilir". Birçok HTTP/1.1 tarayıcısı, HOL engellemeyi atlatmanın yanı sıra daha fazla bant genişliği elde etmek için altı paralel TCP bağlantısı açar.

HTTP/2'nin temel hedeflerinden biri, tek bir web sayfasını taşımak için birden fazla paralel TCP bağlantısını ortadan kaldırmaktır (veya en azından sayısını azaltmaktır). Bu, sadece sunucuda açık ve bakımlı tutulması gereken soket sayısını azaltmakla kalmaz, aynı zamanda TCP tıkanıklık kontrolünün amaçlandığı gibi çalışmasına da izin verir. Ancak, bir web sayfasını taşımak için yalnızca bir TCP bağlantısının kullanılmasıyla, HTTP/2, HOL engellemeyi önlemek için dikkatlice tasarlanmış mekanizmalara ihtiyaç duyar.

HTTP/2 Framing

HOL engellemeyi çözmek için HTTP/2'nin çözümü, her mesajı küçük çerçevelere bölmek ve istek ve yanıt mesajlarını aynı TCP bağlantısında sıralı olarak göndermektir. Bunun anlaşılması için, bir web sayfasının bir büyük video klibi ve 8 küçük nesneden oluştuğu bir örneği yeniden düşünelim. Dolayısıyla sunucu, bu web sayfasını görmek isteyen herhangi bir tarayıcıdan eşzamanlı olarak 9 istek alır. Bu isteklerin her biri için, sunucunun tarayıcıya 9 rekabet eden HTTP yanıt mesajı göndermesi gerekir. Tüm çerçevelerin sabit uzunlukta olduğunu varsayalım, video klibi 1000 çerçeveden oluşur ve her bir küçük nesne iki çerçeveden oluşur. Çerçeve sıralaması kullanılarak, video klibinden bir çerçeve gönderildikten sonra, her bir küçük nesnenin ilk çerçeveleri gönderilir. Ardından, video klibinin ikinci çerçevesi gönderildikten sonra, her bir küçük nesnenin son çerçeveleri gönderilir. Böylece, tüm küçük nesneler, toplam 18 çerçeve gönderildikten sonra gönderilir. Eğer sıralama kullanılmazsa, küçük nesneler yalnızca 1016 çerçeve gönderildikten sonra gönderilir. Bu nedenle, HTTP/2 çerçeveleme mekanizması kullanıcı tarafından algılanan gecikmeyi önemli ölçüde azaltabilir. Bir HTTP mesajını bağımsız çerçevelere bölmek, bunları sıralamak ve ardından diğer ucunda yeniden birleştirmek için yeteneği, HTTP/2'nin tek en önemli gelişmesidir. Çerçeveleme, HTTP/2 protokolünün çerçeve alt katmanı tarafından yapılır. Bir sunucu bir HTTP yanıtı göndermek istediğinde, yanıt, çerçeve alt katmanı tarafından işlenir ve çerçevelere bölünür. Yanıtın başlık alanı bir çerçeve haline gelir ve mesajın gövdesi bir veya daha fazla ek çerçeveye bölünür. Yanıtın çerçeveleri, sunucudaki çerçeve alt katmanı tarafından diğer yanıt çerçeveleriyle sıralanır ve tek kalıcı TCP bağlantısı üzerinden gönderilir. Çerçeveler istemciye ulaştığında, önce çerçeve alt katmanında orijinal yanıt mesajlarına yeniden birleştirilir ve ardından tarayıcı tarafından normal şekilde işlenir. Benzer şekilde, bir istemcinin HTTP istekleri çerçevelere bölünür ve sıralanır.

Bir HTTP mesajını bağımsız çerçevelere bölmekle kalmayıp, çerçeve alt katmanı ayrıca çerçeveleri ikili olarak kodlar. İkili protokoller, daha verimli ayrıştırılır, biraz daha küçük çerçevelere yol açar ve daha az hata eğilimlidirler.

Response Message Prioritization and Server Pushing

Mesaj önceliklendirme, geliştiricilere isteklerin göreceli önceliğini özelleştirerek uygulama performansını daha iyi optimize etme imkânı tanır. Çerçeve alt katmanının mesajları aynı istek yapan kişiye gidecek şekilde paralel veri akışlarına düzenler.Bir istemci bir sunucuya eşzamanlı istekler gönderdiğinde, istediği yanıtlara öncelik verebilir, her bir mesaja 1 ile 256 arasında bir ağırlık atayarak. Daha yüksek sayı daha yüksek önceliği gösterir. Bu ağırlıkları kullanarak, sunucu en yüksek önceliğe sahip yanıtların çerçevelerini önce gönderebilir. Buna ek olarak, istemci, her mesajın diğer mesajlara olan bağımlılığını belirterek, her mesajın hangi mesaja bağlı olduğunu belirtir.

HTTP/2'nin bir başka özelliği de sunucunun tek bir istemci isteği için birden fazla yanıt gönderme yeteneğidir. Yani, orijinal isteğin yanıtına ek olarak, sunucu her birini ayrı ayrı talep etmeden istemciye ek nesneleri iletebilir. Bu, HTML taban sayfasının tamamen oluşturulması için gereken nesneleri belirttiği için mümkündür. Bu nesneler için HTTP isteklerinin beklenmesi yerine, sunucu HTML sayfasını analiz edebilir, gereken nesneleri belirleyebilir ve bu nesneleri açıkça talep edilmeden önce istemciye gönderebilir. Sunucu itmesi, isteklerin beklenmesinden kaynaklanan ek gecikmeyi ortadan kaldırır.

7.2.7 HTTP/3

QUIC, uygulama katmanında çıplak kemik UDP protokolü üzerinden uygulanan yeni bir "aktarım" protokolüdür. QUIC, mesaj çoğullama (serpiştirme), akış başına akış kontrolü ve düşük gecikmeli bağlantı kurulumu gibi HTTP için arzu edilen çeşitli özelliklere sahiptir. HTTP/3 henüz QUIC üzerinden çalışacak şekilde tasarlanmış yeni bir HTTP protokolüdür. 2020 itibariyle HTTP/3 İnternet taslaklarında açıklanmıştır ve henüz tam olarak standartlaştırılmamıştır. HTTP/2 özelliklerinin çoğu (mesaj serpiştirme gibi) QUIC tarafından kapsanarak HTTP/3 için daha basit ve akıcı bir tasarıma olanak tanır.

7.3 Electronic Mail in the Internet

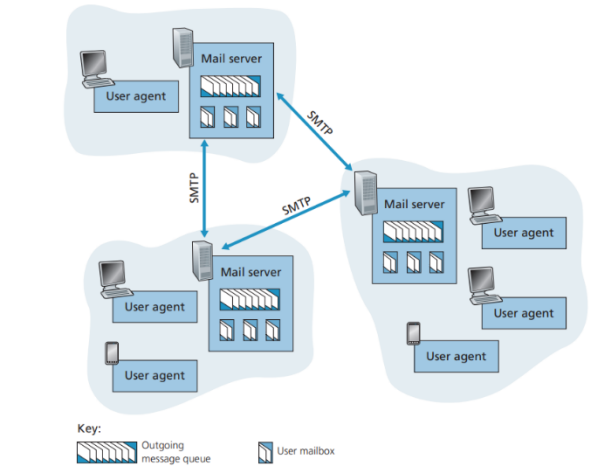


Figure 2.14 ♦ A high-level view of the Internet e-mail system

Şekil 2.14, İnternet posta sistemine yüksek seviyeli bir bakış sunar. Bu diyagramdan görüyoruz ki üç ana bileşeni vardır: kullanıcı araçları, posta sunucuları ve Basit Posta Aktarım Protokolü (SMTP). Gönderici, Alice'in bir e-posta mesajını alıcı, Bob'a gönderdiği bağlamında, bu bileşenlerden her birini tanımlayalım. Kullanıcı araçları, kullanıcıların mesajları okumasına, yanıtlamasına, iletmesine, kaydetmesine ve oluşturmasına izin verir. Alice mesajını oluşturmayı bitirdiğinde, kullanıcı aracı mesajı posta sunucusuna gönderir ve mesaj posta sunucusunun çıkış kuyruğuna yerleştirilir. Bob bir mesajı okumak istediğinde, kullanıcı aracı mesajı posta sunucusundaki posta kutusundan alır. Posta sunucuları, e-posta altyapısının çekirdeğini oluşturur. Bob gibi her alıcı, posta sunucularından birinde bulunan bir posta kutusuna sahiptir. Bob'un posta kutusu, kendisine gönderilen mesajları yönetir ve saklar. Bir mesaj genellikle göndericinin kullanıcı aracından başlar, ardından göndericinin posta sunucusuna ve ardından alıcının posta sunucusuna, burada alıcının posta kutusuna yerleştirilir. Bob, posta kutusundaki mesajlara erişmek istediğinde, posta sunucusu Bob'u (kullanıcı adı ve şifresiyle) doğrular. Ayrıca, Alice'in posta sunucusu Bob'un posta sunucusundaki arızalarla da ilgilenmek zorundadır. Eğer Alice'in sunucusu postayı Bob'un sunucusuna teslim edemezse, Alice'in sunucusu mesajı bir mesaj kuyruğunda (message queue) tutar ve daha sonra mesajı aktarmayı dener. Tekrar denemeler genellikle her 30 dakikada bir yapılır; birkaç gün boyunca başarı olmazsa, sunucu mesajı kaldırır ve göndericiye (Alice) bir e-posta mesajı ile bildirir. SMTP, İnternet elektronik postası için ana uygulama katmanı protokolüdür. Göndericinin posta sunucusundan alıcının posta sunucusuna posta aktarmak için TCP'nin güvenilir veri transfer hizmetini kullanır. Çoğu uygulama katmanı protokolü gibi, SMTP'nin iki tarafı vardır: biri, göndericinin posta sunucusunda çalışan istemci tarafı ve diğeri, alıcının posta sunucusunda çalışan sunucu tarafı. SMTP'nin istemci ve sunucu tarafları her posta sunucusunda çalışır. Bir posta sunucusu, diğer posta sunucularına posta gönderirken SMTP istemcisi olarak hareket eder. Bir posta sunucusu, diğer posta sunucularından posta alırken SMTP sunucusu olarak hareket eder.

7.3.1 SMTP

SMTP, internet elektronik postasının temelidir. SMTP, göndericilerin posta sunucularından alıcıların posta sunucularına mesaj transfer eder. SMTP, HTTP'den çok daha eski bir teknolojidir.

SMTP'nin İnternet'teki yaygınlığı göz önüne alındığında birçok harika özelliğe sahip olmasına rağmen, yine de bazı eski özelliklere sahip eski bir teknolojidir. Örneğin, tüm posta mesajlarının gövdesini (sadece başlıkları değil) basit 7-bit ASCII'ye sınırlar. Bu kısıtlama, iletim kapasitesinin sınırlı olduğu ve kimse büyük ekler veya büyük görüntü, ses veya video dosyaları göndermiyorken erken 1980'lerde mantıklıydı. Ancak bugün, multimedya çağında, 7-bit ASCII kısıtlaması biraz sıkıntılıdır. SMTP üzerinden gönderilmeden önce ikili multimedya verilerinin ASCII'ye kodlanmasını gerektirir ve karşılık gelen ASCII mesajının SMTP taşımadan sonra ikili olarak kodlanması gerektirir. Http ise multimedya verilerinin transferinden önce ASCII'ye kodlanmasını gerektirmez.

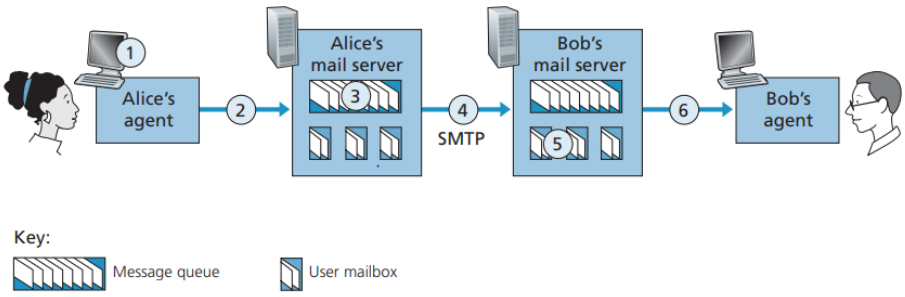


Figure 2.15 ♦ Alice sends a message to Bob

1. Alice, e-posta için kullanıcı aracını başlatır, Bob'un e-posta adresini sağlar (örneğin, bob@someschool.edu), bir mesaj oluşturur ve kullanıcı aracına mesajı göndermesi talimatını verir.
2. Alice'in kullanıcı aracı, mesajı posta sunucusuna gönderir ve mesaj bir mesaj kuyruğuna yerleştirilir.
3. SMTP'nin istemci tarafı, Alice'in posta sunucusunda çalışırken, mesajı mesaj kuyruğunda görür. Bob'un posta sunucusunda çalışan bir SMTP sunucusuna TCP bağlantısı açar.
4. İlk SMTP el sıkışma işlemlerinin ardından, SMTP istemcisi Alice'in mesajını TCP bağlantısına gönderir.
5. Bob'un posta sunucusunda, SMTP'nin sunucu tarafı mesajı alır. Bob'un posta sunucusu daha sonra mesajı Bob'un posta kutusuna yerleştirir.
6. Bob, mesajı istediği zaman okumak için kullanıcı aracını başlatır.

SMTP, normalde iki posta sunucusu dünyanın tam zıt uçlarında bile olsa, posta göndermek için ara posta sunucularını kullanmaz. Diyelim ki Alice'in sunucusu Hong Kong'da ve Bob'un sunucusu St. Louis'de olsun, TCP bağlantısı doğrudan Hong Kong ve St. Louis sunucuları arasında kurulur. Özellikle, Bob'un posta sunucusu çalışmıyorsa, mesaj Alice'in posta sunucusunda kalır ve yeni bir deneme için bekler - mesaj ara bir posta sunucusuna yerleştirilmez. . İlk olarak, istemci SMTP (gönderen posta sunucusunda çalışan) TCP'nin alıcı posta sunucusundaki 25 numaralı bağlantı noktasına bir bağlantı kurmasını sağlar. Sunucu kapalıysa, istemci daha sonra tekrar dener. Bu bağlantı kurulduğunda, sunucu ve istemci bazı uygulama katmanı el sıkışma işlemleri gerçekleştirir. Bu SMTP el sıkışma aşamasında, SMTP istemcisi mesajı oluşturan kişinin e-posta adresini (gönderen) ve alıcının e-posta adresini belirtir. SMTP istemcisi ve sunucusu birbirlerini tanıttıktan sonra, istemci mesajı gönderir. SMTP, TCP'nin güvenilir veri transfer hizmetine güvenebilir ve mesajı hata olmadan sunucuya ulaştırabilir. İstemci daha sonra diğer mesajları sunucuya göndermek için aynı TCP bağlantısı üzerinde bu işlemi tekrarlar; aksi takdirde, bağlantıyı kapatmak için TCP'ye talimat verir.

Şimdi bir SMTP istemcisi (C) ve bir SMTP sunucusu (S) arasında değiş tokuş edilen mesajların bir örnek geçişine bir göz atalım. İstemcinin ana bilgisayar adı crepes.fr ve sunucunun ana bilgisayar adı hamburger.edu'dur. C: ile başlayan ASCII metin satırları, istemcinin TCP soketine gönderdiği satırlardır ve S: ile başlayan ASCII metin satırları, sunucunun TCP soketine gönderdiği satırlardır. Aşağıdaki geçiş, TCP bağlantısı kurulduktan hemen sonra başlar.

```
S:  220 hamburger.edu
C:  HELO crepes.fr
S:  250 Hello crepes.fr, pleased to meet you
C:  MAIL FROM: <alice@crepes.fr>
S:  250 alice@crepes.fr ... Sender ok
C:  RCPT TO: <bob@hamburger.edu>
S:  250 bob@hamburger.edu ... Recipient ok
C:  DATA
S:  354 Enter mail, end with "." on a line by itself
C:  Do you like ketchup?
C:  How about pickles?
C:  .
S:  250 Message accepted for delivery
C:  QUIT
S:  221 hamburger.edu closing connection
```

Yukarıdaki örnekte, istemci ("Do you like ketchup? How about pickles?") mesajını crepes.fr posta sunucusundan hamburger.edu posta sunucusuna gönderir. Diyalogun bir parçası olarak, istemci beş komut verir: HELO (HELLO'nun kısaltması), MAIL FROM, RCPT TO, DATA ve QUIT. Bu komutlar açıklayıcıdır. İstemci ayrıca, mesajın sonunu belirten tek bir noktadan oluşan bir satır da gönderir. (ASCII jargonunda, her mesaj CRLF.CRLF ile biter, burada CR ve LF sırasıyla "carriage return" ve "line feed"i temsil eder.) Sunucu, her komuta yanıtlar verir, her yanıtın bir yanıt kodu ve bazı (isteğe bağlı) İngilizce açıklamaları vardır. Burada belirtmek gerekir ki SMTP kalıcı bağlantıları kullanır: Gönderen posta sunucusunun aynı alıcı posta sunucusuna gönderilecek birkaç mesajı varsa, tüm mesajları aynı TCP bağlantısı üzerinden gönderebilir. Her mesaj için, istemci yeni bir MAIL FROM: crepes.fr ile işlemi başlatır, mesajın sonunu izole bir nokta ile belirtir ve tüm mesajlar gönderildikten sonra QUIT komutunu verir.

7.3.2 Mail Message Formats

Alice, bir mektubu posta ile Bob'a gönderdiğinde, mektubun üst kısmına Bob'un adresi, kendi geri dönüş adresi ve tarih gibi çeşitli yan bilgileri ekleyebilir. Benzer şekilde, bir kişiden diğerine bir e-posta mesajı gönderildiğinde, mesajın kendisi öncesinde, periferik bilgileri içeren bir başlık bulunur. Bu periferik bilgiler, RFC 5322'de tanımlanan bir dizi başlık satırında bulunur ve mesajın gövdesinden boş bir satır (yani CRLF tarafından) ayırt edilir. RFC 5322, posta başlık satırları için kesin formatı ve anlamsal yorumlarını belirtir. HTTP'de olduğu gibi, her başlık satırı, bir anahtarın ardından gelen iki nokta üst üste ve bir değerden oluşan okunabilir metni içerir. Bazı anahtar kelimeler zorunludur ve diğerleri isteğe bağlıdır. Her başlık, bir From: başlık satırı ve bir To: başlık satırı içermelidir; bir başlık, bir Subject: başlık satırı ve diğer isteğe bağlı başlık satırlarını da içerebilir. Bu başlık satırlarının, 7.3.1 Bölümündeki SMTP komutlarından farklı olduğunu belirtmek önemlidir ("from" ve "to" gibi bazı ortak kelimeleri

içerirler). O bölümdeki komutlar, SMTP el sıkışma protokolünün bir parçasıydı; bu bölümde incelenen başlık satırları ise posta mesajının bir parçasıdır.

2.3.3 Mail Accsess Protocols

SMTP, mesajı Alice'in posta sunucusundan Bob'un posta sunucusuna ilettikten sonra, mesaj Bob'un posta kutusuna yerleştirilir. Alıcı olan Bob (örneğin, akıllı telefon veya PC üzerinde) kullanıcı ajanını yerel ana bilgisayarında çalıştırırken, posta sunucusunu da yerel ana bilgisayarına yerleştirmek doğal olarak düşünülebilir. Bu yaklaşımda, Alice'in posta sunucusu doğrudan Bob'un PC'siyle diyalog kurabilir. Ancak bu yaklaşımın bir sorunu vardır. Bir posta sunucusu, posta kutularını yönetir ve SMTP'nin istemci ve sunucu taraflarını çalıştırır. Bob'un posta sunucusu yerel ana bilgisayarında bulunursa, Bob'un ana bilgisayarının her zaman açık kalması ve İnternet'e bağlı kalması gerekecektir, çünkü yeni posta her zaman gelebilir. Bu, birçok İnternet kullanıcısı için pratik değildir. Bunun yerine, tipik bir kullanıcı, yerel ana bilgisayarında bir kullanıcı ajanını çalıştırır ancak posta kutusuna her zaman açık olan paylaşılan bir posta sunucusuna erişir. Bu posta sunucusu, diğer kullanıcılarla paylaşılır.

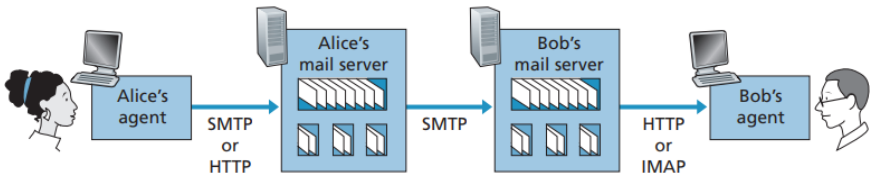


Figure 2.16 ♦ E-mail protocols and their communicating entities

Şimdi, bir e-posta mesajının Alice'ten Bob'a gönderildiğinde aldığı yol düşünülecek. E-posta mesajının bir noktada Bob'un posta sunucusuna yerleştirilmesi gerektiğini öğrendik. Bu, Alice'in kullanıcı ajanının mesajı doğrudan Bob'un posta sunucusuna göndermesiyle yapılabilir. Ancak genellikle gönderen kullanıcı ajanı doğrudan alıcı posta sunucusuyla iletişim kurmaz. Bunun yerine, Şekil 2.16'da gösterildiği gibi, Alice'in kullanıcı ajanı e-posta mesajını kendi posta sunucusuna teslim etmek için SMTP veya HTTP kullanır, ardından Alice'in posta sunucusu (bir SMTP istemcisi olarak) e-posta mesajını Bob'un posta sunucusuna iletir. İki adımlı işlem neden? Öncelikle, Alice'in posta sunucusu aracı olmadan, Alice'in kullanıcı ajanının ulaşılamayan bir hedef posta sunucusuna başvurması mümkün değildir. Alice'in önce e-postayı kendi posta sunucusuna yatırmasını sağlayarak, Alice'in posta sunucusu, Bob'un posta sunucusuna mesajı tekrar tekrar göndermeye çalışabilir, örneğin her 30 dakikada bir, Bob'un posta sunucusu işlevsel hale gelene kadar. (Ve eğer Alice'in posta sunucusu devre dışı kalırsa, o zaman sistem yöneticisine şikayet etme imkanı vardır!) Bob gibi bir alıcı, mesajlarının, posta sunucusunda oturduğu bir yerel ana bilgisayardan nasıl alır? Bob'un kullanıcı ajanı, mesajları almak için SMTP'yi kullanamaz çünkü mesajları almak bir çekme işlemidir, oysa SMTP bir itme protokolüdür.

Bugün, Bob'un e-postasını bir posta sunucusundan almak için iki yaygın yöntem vardır. Bob, Web tabanlı e-posta veya bir akıllı telefon uygulaması (örneğin, Gmail) kullanıyorsa, kullanıcı ajanı Bob'un e-postasını almak için HTTP'yi kullanır. Bu durumda, Bob'un posta sunucusunun HTTP arayüzüne ve Alice'in posta sunucusuyla iletişim kurmak için bir SMTP arayüzüne sahip olması gerekir. Alternatif yöntem, genellikle Microsoft Outlook gibi posta istemcileriyle kullanılan İnternet Posta Erişim Protokolü'nü (IMAP), kullanmaktır. Hem HTTP hem de IMAP yaklaşımları, Bob'un klasörleri yönetmesine izin verir; Bob'un posta sunucusunda bulunan klasörler. Bob, mesajları oluşturabilir, klasörler oluşturabilir, mesajları siler, mesajları önemli olarak işaretler ve benzeri işlemleri yapabilir.

7.4 DNS—The Internet's Directory Service

Bir ana bilgisayar için bir tanımlayıcı, hostname (ana bilgisayar) adıdır. Ana bilgisayar adları www.facebook.com, www.google.com, gaia.cs.umass.edu gibi. Ana bilgisayar adları değişken uzunlukta alfasayısal karakterlerden oluşabilir, bu nedenle yönlendiriciler tarafından işlenmesi zor olabilir. Bu nedenlerle, hostlar aynı zamanda "**IP adresleri**" olarak adlandırılan tanımlayıcılarla da tanımlanır.

Bir IP adresi dört byte'tan oluşur ve katı bir hiyerarşik yapıya sahiptir. Bir IP adresi, her biri 0 ila 255 arasında ondalık gösterimde ifade edilen her bir byte'ı ayıran noktalardan oluşur. Bir IP adresi hiyerarşiktir çünkü adresi soldan sağa taradığımızda, İnternet'teki konumu hakkında giderek daha spesifik bilgiler elde ederiz (yani, hangi ağda, ağların ağında). Benzer şekilde, bir posta adresini alttan üste taradığımızda, alıcının konumu hakkında giderek daha spesifik bilgiler elde ederiz.

7.4.1 Services Provided by DNS

Bir ana bilgisayarı tanımlamanın iki yolu vardır bir ana bilgisayar adıyla ve bir IP adresiyle. İnsanlar daha ezberlenebilir ana bilgisayar adı tanımlayıcısını tercih ederken, yönlendiriciler sabit uzunluklu, hiyerarşik yapıdaki IP adreslerini tercih ederler. Bu tercihleri uzlaştırmak için, ana bilgisayar adlarını IP adreslerine çeviren bir dizin hizmetine ihtiyacımız var. İşte **İnternet'in alan adı sisteminin (DNS)** temel görevi budur.

DNS, (1) DNS sunucularının bir hiyerarşi içinde uygulanan dağıtılmış bir veritabanı ve (2) ana bilgisayarların dağıtılmış veritabanını sorgulamasına izin veren bir uygulama katmanı protokolüdür. DNS sunucuları genellikle BIND (Berkeley Internet Name Domain) yazılımını çalıştıran UNIX makineleridir. DNS protokolü UDP üzerinden çalışır ve 53 numaralı bağlantı noktasını kullanır.

DNS, kullanıcı tarafından sağlanan ana bilgisayar adlarını IP adreslerine çevirmek için HTTP ve SMTP gibi diğer uygulama katmanı protokoller tarafından yaygın olarak kullanılır. Bir örnek olarak, bir kullanıcının ana bilgisayarında çalışan bir tarayıcı (yani, bir HTTP istemcisi), www.someschool.edu/index.html URL'sini istediğinde ne olduğunu düşünelim. Kullanıcının ana bilgisayarının www.someschool.edu web sunucusuna bir HTTP istek iletilmesi için, kullanıcının ana bilgisayarının önce www.someschool.edu'nun IP adresini alması gerekir. Bu şu şekilde yapılır:

1. Aynı kullanıcı makinesi, DNS uygulamasının istemci tarafını çalıştırır.
2. Tarayıcı, URL'den (Uniform Resource Locator) www.someschool.edu ana bilgisayar adını çıkarır ve ana bilgisayar adını DNS uygulamasının istemci tarafına iletilir.
3. DNS istemcisi, ana bilgisayar adını içeren bir sorguyu bir DNS sunucusuna gönderir.
4. DNS istemcisi sonunda bir cevap alır, bu cevap ana bilgisayar adı için IP adresini içerir.
5. Tarayıcı, DNS'ten IP adresini aldıktan sonra, o IP adresindeki port 80'de bulunan HTTP sunucusu işlemine bir TCP bağlantısı başlatabilir.

Bu örneğe göre, DNS kullanan İnternet uygulamalarına ek bir gecikme ekler bazen önemli bir gecikme. İstenen IP adresi genellikle "yakındaki" bir DNS sunucusunda önbelleğe alınır, bu da DNS ağ trafiğini azaltmaya ve ortalama DNS gecikmesini azaltmaya yardımcı olur. DNS, ana bilgisayar adlarını IP adreslerine çevirmenin yanı sıra birkaç önemli hizmet sunar:

Host Aliasing: Karmaşık bir ana bilgisayar adına sahip bir ana bilgisayara bir veya daha fazla takma ad atanabilir. Örneğin, *relay1.west-coast.enterprise.com* gibi bir ana bilgisayar adı, *enterprise.com* ve *www.enterprise.com* gibi iki takma adı olabilir. Bu durumda, relay1.west-coast.enterprise.com ana bilgisayar adı canonical hostname olarak adlandırılır. Alias hostnameler, varsa, genellikle kanonik ana bilgisayar adlarından daha mnemoniktir. Bir uygulama, sağlanan bir takma ad ana bilgisayar adı için kanonik ana bilgisayar adını ve ana bilgisayarın IP adresini almak için DNS'yi çağırabilir.

Mail Server Aliasing: Açıkça görüleceği gibi, e-posta adreslerinin mnemonik olması son derece arzu edilir. Örneğin, Bob'un Yahoo Mail'de bir hesabı varsa, Bob'un e-posta adresi basitçe bob@yahoo.com olabilir. Ancak, Yahoo mail sunucusunun ana bilgisayar adı, basitçe yahoo.com'dan çok daha karmaşıktır ve daha az mnemoniktir (örneğin, kanonik ana bilgisayar adı relay1.west-coast.yahoo.com gibi bir şey olabilir). DNS, bir posta uygulaması tarafından, sağlanan bir takma ad ana bilgisayar adı için kanonik ana bilgisayar adını ve ana bilgisayarın IP adresini almak için çağrılabilir. Aslında, MX kaydı, bir şirketin posta sunucusunun ve Web sunucusunun aynı (takma adlandırılmış) ana bilgisayar adlarına sahip olmasına izin verir; örneğin, bir şirketin web sunucusu ve posta sunucusu her ikisi de enterprise.com olarak adlandırılabilir.

Load Distribution: DNS, örneğin çoğaltılmış Web sunucuları arasında yük dağıtımı yapmak için de kullanılır. Cnn.com gibi yoğun siteler, her biri farklı bir son sistemde çalışan ve her biri farklı bir IP adresine sahip olan çeşitli sunucular üzerinde çoğaltılır. Çoğaltılmış Web sunucuları için, bir takma ad ana bilgisayar adı ile ilişkilendirilmiş bir IP adresler kümesi bulunur. DNS veritabanı bu IP adresleri kümesini içerir. İstemciler, bir adın bir küme IP adresine eşlendiği bir DNS sorgusu yaptıklarında, sunucu yanıt olarak tüm IP adres kümesini verir, ancak her yanıt içinde adreslerin sıralamasını döndürür. Bir istemci tipik olarak isteğini IP adresine gönderir, bu da kümede ilk sırada listelenen IP adresine gönderilir, DNS dönüşümü ise trafiği çoğaltılmış sunucular arasında dağıtır. DNS dönüşümü ayrıca e-posta için de kullanılır, böylece birden fazla posta sunucusu aynı takma adı alabilir.

NOT: HTTP, FTP ve SMTP gibi, DNS protokolü bir uygulama katmanı protokolüdür çünkü (1) iletişim kuran son sistemler arasında istemci-sunucu paradigması kullanılarak çalışır ve (2) iletişim kuran son sistemler arasında DNS iletilerini transfer etmek için temel bir uçtan uca

taşıma protokolüne güvenir. Ancak, DNS'nin rolü Web, dosya transferi ve e-posta uygulamalarından oldukça farklıdır. Bu uygulamaların aksine, DNS, kullanıcıların doğrudan etkileşimde bulunmadığı bir uygulama değildir. Bunun yerine, DNS, host adlarını temel IP adreslerine çevirerek, Internet'teki kullanıcı uygulamaları ve diğer yazılımlar için temel bir Internet işlevi sağlar.

NOT: internet mimarisindeki karmaşıklığın çoğunun ağın "kenarlarında" bulunduğunu belirttik. DNS, isimden adres çevrimi sürecini kritik bir şekilde uygulayan ve ağın kenarında bulunan istemciler ve sunucuları kullanarak, bu tasarım felsefesinin bir başka örneğidir.

7.4.2 Overview of How DNS Works

Bir kullanıcının makinesinde çalışan bir Web tarayıcısı veya bir posta istemcisi gibi bir uygulamanın bir host adını bir IP adresine çevirmesi gerektiğini varsayalım. Uygulama, çevrilmek istenen host adını belirterek DNS'nin istemci tarafını çağırır. (Birçok UNIX tabanlı makinede, bir uygulamanın çevrimi yapmak için çağırdığı işlev çağrısı gethostbyname() işlevi çağrısıdır.) Kullanıcının makinesindeki DNS daha sonra ağa bir sorgu ileti gönderir. Tüm DNS sorgu ve yanıt iletileri, port 53'e UDP veragramları içinde gönderilir. Birkaç milisaniyeden birkaç saniyeye kadar süren bir gecikmeden sonra, kullanıcının makinesindeki DNS bir DNS yanıt iletisi alır ve istenen eşlemeyi sağlar. Bu eşleme daha sonra çağıran uygulamaya iletilir. Dolayısıyla, kullanıcının makinesindeki çağıran uygulamanın bakış açısından DNS, basit ve doğrudan bir çeviri hizmeti sağlayan bir siyah kutudur. Ancak aslında, hizmeti uygulayan siyah kutu karmaşık bir yapıya sahiptir ve dünya çapında dağıtılan birçok DNS sunucusunu ve sorgulayan hostların nasıl iletişim kurduğunu belirleyen bir uygulama katmanı protokolünü içerir.

DNS için basit bir tasarım, tüm eşlemeleri içeren tek bir DNS sunucusuna sahip olurdu. Bu merkezi tasarımında, istemciler tüm sorguları tek bir DNS sunucusuna yönlendirir ve DNS sunucusu doğrudan sorgulayan istemcilere yanıt verir. Bu tasarımın basitliği cazip olsa da, günümüz interneti için uygun değildir, çünkü internetin çok sayıda (ve artan) anahtarı vardır. Merkezi bir tasarımın sorunları şunları içerir:

- **A Single Point Of Failure:** DNS sunucusu çökerse tüm İnternet de çöker!
- **Traffic Volume:** Tek bir DNS sunucusunun tüm DNS sorgularını (yüz milyonlarca ana bilgisayardan oluşturulan tüm HTTP istekleri ve e-posta mesajları için) işlemesi gerekir.
- **Distant Centralized Database:** Tek DNS sunucusunu New York City'ye koyarsak, Avustralya'dan gelen tüm sorguların belki de yavaş ve sıkışık bağlantılar üzerinden dünyanın diğer tarafına gitmesi gerekir. Bu önemli gecikmelere yol açabilir.
- **Mantinince:** Tek DNS sunucusunun tüm İnternet ana bilgisayarlarının kayıtlarını tutması gerekir. Bu merkezi veri tabanı sadece çok büyük olmakla kalmayacak, aynı zamanda her yeni ana makineyi hesaba katacak şekilde sık sık güncellenmesi gerekecekti.

A Distributed, Hierarchical Database

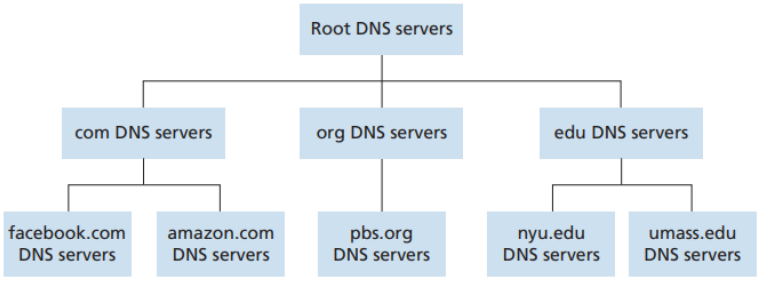


Figure 2.17 ♦ Portion of the hierarchy of DNS servers

Ölçekleme sorunuyla başa çıkmak için DNS, hiyerarşik bir şekilde düzenlenmiş ve dünya çapında dağıtılmış büyük bir sunucu ağı kullanır. İnternetin tüm anahtarlarının eşlemelerini tek bir DNS sunucusunda bulundurmaz. Bunun yerine, eşlemeler DNS sunucuları arasında dağıtılır. Birinci yaklaşıma göre, kök DNS sunucuları, top-level domain (üst düzey alan (TLD)) DNS sunucuları ve yetkili DNS sunucuları olmak üzere üç sınıf DNS sunucusu vardır ve bu sunucular hiyerarşik bir düzende düzenlenir ve dünya çapında dağıtılır. Bu üç sunucu sınıfının nasıl etkileşime girdiğini anlamak için, bir DNS istemcisi www.amazon.com host adının IP adresini belirlemek istediğinde, birinci yaklaşıma göre aşağıdaki olaylar gerçekleşir. İstemci öncelikle kök sunuculardan birine başvurur, bu sunucu com üst düzey alanı için TLD sunucularının IP adreslerini döndürür. Ardından, istemci bu TLD sunuculardan birine başvurur, bu sunucu amazon.com için bir yetkili sunucunun IP adresini döndürür. Son olarak, istemci amazon.com için bir yetkili sunucuya başvurur, bu sunucu da www.amazon.com host adının IP adresini döndürür.

- **Root DNS Servers:** Dünya çapında dağınık olarak bulunan 1000'den fazla kök sunucu örneği bulunmaktadır. Bu kök sunucuları, 13 farklı kök sunucusunun kopyalarıdır ve 12 farklı kuruluş tarafından yönetilmektedir. Bu kuruluşlar, İnternet Tahsisli Sayılar Otoritesi (IANA) aracılığıyla koordine edilir. Kök isim sunucularının tam listesi, bunları yöneten kuruluşlar ve IP adresleri [Root Servers 2020] adresinde bulunabilir. Kök isim sunucuları, TLD sunucularının IP adreslerini sağlar.

- **Top-level domain (TLD) servers:** Her bir üst düzey alan için - com, org, net, edu, gov gibi üst düzey alanlar ve uk, fr, ca, jp gibi tüm ülke üst düzey alanları için - bir TLD sunucusu (veya sunucu kümesi) bulunmaktadır. Verisign Global Registry Services şirketi, com üst düzey alanı için TLD sunucularını yönetirken, Educause şirketi edu üst düzey alanı için TLD sunucularını yönetir. Bir TLD'yi destekleyen ağ altyapısı geniş ve karmaşık olabilir;

- **Authoritative DNS servers:** İnternet'te halka açık sunucuları (Web sunucuları ve posta sunucuları gibi) olan her organizasyon, bu sunucuların isimlerini IP adreslerine eşleyen halka açık DNS kayıtları sağlamalıdır. Bir organizasyonun yetkili DNS sunucusu, bu

DNS kayıtlarını barındırır. Bir organizasyon, bu kayıtları tutmak için kendi yetkili DNS sunucusunu uygulamayı seçebilir; alternatif olarak, organizasyon bu kayıtların bir servis sağlayıcısının yetkili DNS sunucusunda saklanmasını sağlamak için ödeme yapabilir. Çoğu üniversite ve büyük şirket, kendi birincil ve yedek (yedek) yetkili DNS sunucularını uygular ve sürdürür.

Root, TLD ve Authoritative DNS sunucuları, Figür 2.17'de gösterildiği gibi DNS sunucularının hiyerarşisine aittir. Ancak, DNS mimarisinde merkezi bir konuma sahip olmasına rağmen, hiyerarşinin katı bir parçası olmayan önemli bir DNS sunucu türü daha vardır. Bu sunucuya yerel DNS sunucusu denir. Her ISP - bir ev kullanıcıları için ISP veya kurumsal bir ISP gibi - bir yerel DNS sunucusuna (ayrıca varsayılan ad sunucusu olarak da adlandırılır) sahiptir. Bir ana bilgisayar bir ISS'ye bağlandığında, ISS, ana bilgisayara bir veya daha fazla yerel DNS sunucusunun IP adresini sağlar . Windows veya UNIX'te ağ durumu pencerelerine erişerek yerel DNS sunucunuzun IP adresini kolayca belirleyebilirsiniz. Bir ana bilgisayarın yerel DNS sunucusu genellikle ana bilgisayara "yakın" konumdadır. Kurumsal bir ISS için, yerel DNS sunucusu ana bilgisayarla aynı yerel ağda olabilir; bir ev kullanıcıları için, genellikle yerel DNS sunucusu ana bilgisayardan birkaç yönlendiriciden daha fazla uzaklıkta olur. Bir ana bilgisayar bir DNS sorgusu yaptığında, sorgu yerel DNS sunucusuna gönderilir ve bu sunucu, sorguyu DNS sunucu hiyerarşisine ileten bir proxy olarak hareket eder.

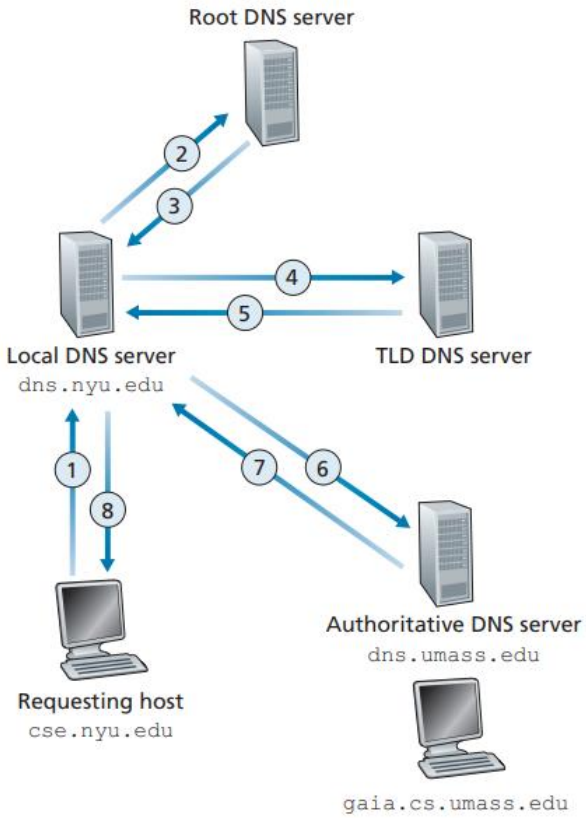


Figure 2.19 ♦ Interaction of the various DNS servers

Diyelim ki cse.nyu.edu ana bilgisayarı gaia.cs.umass.edu ana bilgisayarının IP adresini istiyor. Ayrıca, NYU'nun cse.nyu.edu için yerel (local) DNS sunucusunun dns.nyu.edu olduğunu ve gaia.cs.umass.edu için yetkili bir DNS sunucusunun dns.umass.edu olduğunu varsayalım. Şekil 2.19'da gösterildiği gibi, cse.nyu.edu ana bilgisayarı öncelikle DNS sorgu iletişim kutusunu yerel DNS sunucusu dns.nyu.edu'ya gönderir. Sorgu iletisi, çevrilmek istenen ana bilgisayar adını, yani gaia.cs.umass.edu'yu içerir. Yerel DNS sunucusu sorgu iletişini bir kök DNS sunucusuna yönlendirir. Kök DNS sunucusu, edu eklerini dikkate alır ve edu için sorumlu olan TLD sunucularının IP adreslerini yerel DNS sunucusuna döndürür. Yerel DNS sunucusu daha sonra sorgu iletisini bu TLD sunucularından birine yeniden gönderir. TLD sunucusu umass.edu eklerini dikkate alır ve Massachusetts Üniversitesi için yetkili DNS sunucusunun IP adresini, yani dns.umass.edu'yu yanıt olarak döndürür. Son olarak, yerel DNS sunucusu sorgu iletişim kutusunu doğrudan dns.umass.edu'ya yeniden gönderir ve bu, gaia.cs.umass.edu'nun IP adresini yanıt olarak döndürür. Bu örnekte, bir ana bilgisayar adı eşlemesi elde etmek için sekiz DNS ileti gönderildi: dört sorgu ileti ve dört yanıt ileti.

Önceki örneğimizde, TLD sunucusunun ana bilgisayar adı eşlemesi için yetkili DNS sunucusunu bildiği varsayıldı. Genel olarak, bu her zaman doğru değildir. Bunun yerine, TLD sunucusu genellikle yalnızca ana bilgisayar adı için yetkili DNS sunucusunu bilen ara DNS sunucusunu bilir. Örneğin, Massachusetts Üniversitesi'nin üniversite için bir DNS sunucusu olan dns.umass.edu olduğunu tekrar varsayalım. Ayrıca, Massachusetts Üniversitesi'ndeki her bölümün kendi DNS sunucusuna sahip olduğunu ve her bölüm DNS sunucusunun bölümdeki tüm ana bilgisayarlar için yetkili olduğunu varsayalım. Bu durumda, ara DNS sunucusu dns.umass.edu, cs.umass.edu ile biten bir ana bilgisayar adı için bir sorgu aldığında, dns.nyu.edu'ya, cs.umass.edu ile biten tüm ana bilgisayar adları için yetkili olan dns.cs.umass.edu'nun IP adresini döndürür. Yerel DNS sunucusu dns.nyu.edu daha sonra yetkili DNS sunucusuna sorguyu gönderir, bu da istenen eşlemeyi yerel DNS sunucusuna, ardından da isteği yapan ana bilgisayara geri döndürür. Bu durumda, toplamda 10 DNS ileti gönderilir! Şekil 2.19'da gösterilen örnek, hem yinelemeli sorguları hem de yinelemeli sorguları kullanır. cse.nyu.edu'dan dns.nyu.edu'ya gönderilen sorgu, dns.nyu.edu'nun adına eşlemeyi alması için yinelemeli bir sorgudur. Ancak, sonraki üç sorgu yinelemelidir çünkü tüm yanıtlar doğrudan dns.nyu.edu'ya geri döndürülür. Teoride, herhangi bir DNS sorgusu yinelemeli veya yinelemeli olabilir. Örneğin, Şekil 2.20'de tüm sorguların yinelemeli olduğu bir DNS sorgu zinciri gösterilmektedir. Uygulamada, sorgular genellikle Şekil 2.19'da gösterilen modeli izler: İsteği yapan ana bilgisayarın yerel DNS sunucusuna giden sorgu yinelemeli olarak, kalan sorgular ise yinelemeli olarak gerçekleştirilir.

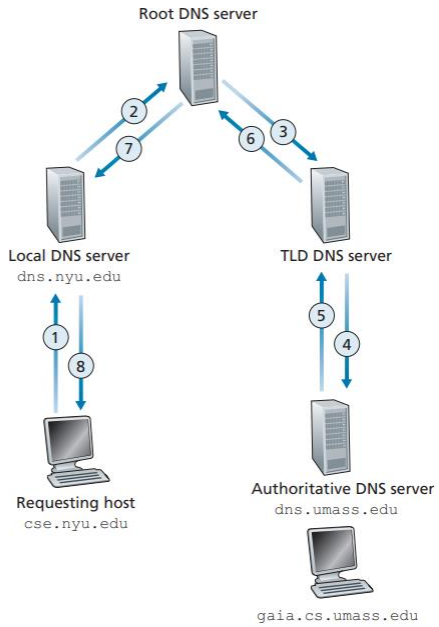


Figure 2.20 ♦ Recursive queries in DNS

DNS Caching

DNS önbelleğini, DNS sisteminin son derece önemli bir özelliğini ihmal etmiştir. Gerçekte, DNS, gecikme performansını iyileştirmek ve İnternet üzerinde dolaşan DNS iletilerinin sayısını azaltmak için DNS önbelleğinden büyük ölçüde yararlanır. DNS önbelleğinin temel fikri çok basittir. Bir sorgu zincirinde, bir DNS sunucusu bir DNS yanıtı aldığında (örneğin, bir ana bilgisayar adından IP adresine bir eşleme içeren), yerel belleğinde eşlemeyi önbelleğe alabilir. Örneğin, Şekil 2.19'da, yerel DNS sunucusu dns.nyu.edu herhangi bir DNS sunucusundan bir yanıt aldığında, yanıtta bulunan herhangi bir bilgiyi önbelleğe alabilir. Bir ana bilgisayar adı/IP adresi çifti bir DNS sunucusunda önbelleğe alındığında ve başka bir sorgu aynı ana bilgisayar adı için DNS sunucusuna geldiğinde, DNS sunucusu istenen IP adresini sağlayabilir, hatta ana bilgisayar adı için yetkili değilse bile. Ana bilgisayarlar ve ana bilgisayar adları ile IP adresleri arasındaki eşlemeler kesinlikle kalıcı değildir, bu nedenle DNS sunucuları önbelleğe alınan bilgileri genellikle iki gün olarak ayarlanan bir süre sonra siler.

Bir örnek olarak, apricot.nyu.edu ana bilgisayarının dns.nyu.edu'ya cnn.com ana bilgisayar adının IP adresini sorması durumunu düşünelim. Dahası, birkaç saat sonra, diyelim ki başka bir NYU ana bilgisayarı olan kiwi.nyu.edu da dns.nyu.edu'ya aynı ana bilgisayar adı için sorgu gönderiyor. Önbellekleme sayesinde, yerel DNS sunucusu ikinci talep eden ana bilgisayara cnn.com'un IP adresini sorgu yapmadan hemen sağlayabilir. Bir yerel DNS sunucusu, TLD sunucularının IP adreslerini de önbelleğe alabilir, böylece yerel DNS sunucusunun bir sorgu zincirinde kök DNS sunucularını atlamasına izin verir. Aslında, önbellekleme nedeniyle, kök sunucular, DNS sorgularının çok küçük bir kesimi hariç olmak üzere atlanır.

7.4.3 DNS Records and Messages

DNS dağıtılmış veritabanını birlikte uygulayan DNS sunucuları, ana bilgisayar adından IP adresine eşlemeleri sağlayan kaynak kayıtlarını (Resource Records - RR) saklar. Her DNS yanıt iletiği bir veya daha fazla kaynak kaydını taşır. Resource Records, aşağıdaki alanları içeren dörtlü bir kayıttır: (Name, Value, Type, TTL)

TTL, kaynak kaydının ömrüdür; bir kaynağın bir önbellekten ne zaman kaldırılması gerektiğini belirler. Aşağıdaki örnek kayıtlarda TTL alanını dikkate almıyoruz. Ad ve Değer alanlarının anlamı, Tür'e bağlıdır:

- Eğer Tür=A ise, Ad bir ana bilgisayar adıdır ve Değer, bu ana bilgisayar adının IP adresidir. Böylece, bir A Türü kaydı standart ana bilgisayar adı-IP adresi eşlemesi sağlar. Örneğin, (relay1.bar.foo.com, 145.37.93.126, A), bir A Türü kaydıdır.

- Eğer Tür=NS ise, Ad bir alan adıdır (örneğin foo.com) ve Değer, alanın ana bilgisayarlarının IP adreslerini nasıl alacağını bilen yetkili bir DNS sunucusunun ana bilgisayar adıdır. Bu kayıt, DNS sorgularını sorgu zincirinde daha ileri yönlendirmek için kullanılır. Örneğin, (foo.com, dns.foo.com, NS), bir NS Türü kaydıdır.

- Eğer Tür=CNAME ise, Değer, takma ad ana bilgisayar adı için bir kanonik ana bilgisayar adıdır. Bu kayıt, sorgu yapan ana bilgisayarlar için bir ana bilgisayar adı için kanonik ad sağlayabilir. Örneğin, (foo.com, relay1.bar.foo.com, CNAME), bir CNAME kaydıdır.

- Eğer Tür=MX ise, Değer, bir takma ad ana bilgisayar adı olan bir posta sunucusunun kanonik adıdır. Örneğin, (foo.com, mail.bar.foo.com, MX), bir MX kaydıdır. MX kayıtları, posta sunucularının ana bilgisayar adlarının basit takma adlara sahip olmasını sağlar. MX kaydını kullanarak, bir şirket, posta sunucusu için ve diğer sunucularından biri için (örneğin, Web sunucusu) aynı takma adı kullanabilir. Bir DNS istemcisi için posta sunucusunun kanonik adını almak için bir MX kaydı için sorgu yapılır; diğer sunucunun kanonik adını almak için DNS istemcisi CNAME kaydı için sorgu yapar.

Eğer bir DNS sunucusu belirli bir ana bilgisayar adı için yetkili ise, o zaman DNS sunucusu bu ana bilgisayar adı için bir A Türü kaydı içerecektir. (DNS sunucusu yetkili değilse bile, önbelleğinde bir A Türü kaydı bulunabilir.) Bir sunucu belirli bir ana bilgisayar adı için yetkili değilse, o zaman sunucu, bu ana bilgisayar adını içeren alan için bir NS Türü kaydı içerecektir; ayrıca NS kaydının Değer alanında DNS sunucusunun IP adresini sağlayan bir A Türü kaydı da içerecektir. Örneğin, varsayalım ki bir edu üst alan adı sunucusu gaia.cs.umass.edu ana bilgisayar adı için yetkili değil. Bu durumda, bu sunucu, ana bilgisayar adını içeren bir alan için bir kayda sahip olacaktır, örneğin, (umass.edu, dns.umass.edu, NS). Edu üst alan adı sunucusu, DNS sunucusu dns.umass.edu'yu bir IP adresine eşleyen bir A Türü kaydı da içerecektir, örneğin, (dns.umass.edu, 128.119.40.111, A).

DNS Messages

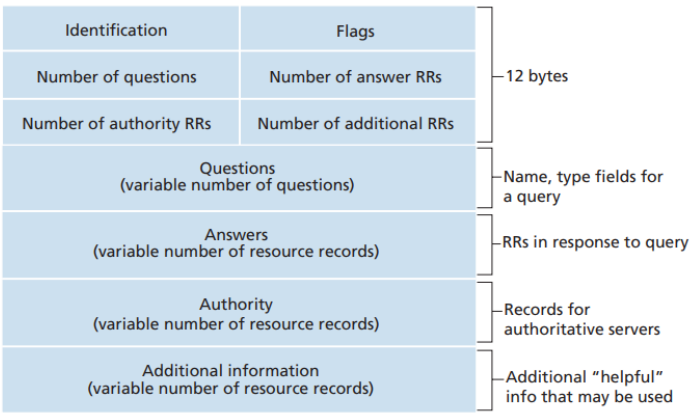


Figure 2.21 ♦ DNS message format

DNS sorgu ve yanıt mesajları DNS mesajlarının yalnızca iki türüdür. Ayrıca, hem sorgu hem de yanıt mesajlarının aynı biçimi vardır, Şekil 2.21'de gösterildiği gibi. Bir DNS mesajındaki çeşitli alanların anlamları şunlardır:

İlk 12 bayt **header** bölümüdür ve birkaç alan içerir. İlk alan, sorguyu tanımlayan 16 bitlik bir numaradır. Bu tanımlayıcı, alınan yanıtları gönderilen sorgularla eşleştirmek için yanıt mesajına kopyalanır. Bayrak alanında bir dizi bayrak bulunmaktadır. Bir 1 bitlik sorgu/yanıt bayrağı, mesajın bir sorgu (0) mu yoksa bir yanıt (1) mi olduğunu belirtir. Bir 1 bitlik yetkili bayrağı, bir DNS sunucusunun sorgulanan bir isim için yetkili bir sunucu olduğunda bir yanıt mesajında ayarlanır. Bir 1 bitlik rekürsif istek bayrağı, bir istemcinin (ana makine veya DNS sunucusu) kayıt olmadığında DNS sunucusunun rekürsif bir işlem yapmasını istediğinde ayarlanır. Bir 1 bitlik rekürsif mevcut bayrağı, DNS sunucusunun rekürsifi desteklediğinde bir yanıtta ayarlanır. Başlıkta ayrıca dört adet sayı alanı bulunur. Bu alanlar, başlığı takip eden dört tür veri bölümünün sayısını gösterir.

Question bölümü, yapılan sorgu hakkında bilgiler içerir. Bu bölüm, (1) sorgulanan ismi içeren bir isim alanı ve (2) isimle ilgili sorulan sorunun türünü belirten bir tür alanı içerir - örneğin, bir ana makine adresi (Type A) ile ilişkilendirilmiş bir isim veya bir isim için posta sunucusu (Type MX).

Bir DNS sunucusundan gelen bir yanıtta, **Answer** bölümü, başlangıçta sorgulanan isim için kaynak kayıtlarını içerir. Her kaynak kaydında Tür (örneğin, A, NS, CNAME ve MX), Değer ve TTL bulunur. Bir yanıt, bir ana makinenin birden fazla IP adresine sahip olabileceği için yanıtta birden fazla RR (kaynak kaydı) döndürebilir (örneğin, bu bölümün daha önce tartışıldığı gibi, çoğaltılmış Web sunucuları için).

Authority bölümü, diğer yetkili sunucuların kayıtlarını içerir.

Additional bölüm, diğer yararlı kayıtları içerir. Örneğin, bir MX sorgusuna yanıt olarak gelen yanıtta, bir posta sunucusunun kanonik ana makinesini sağlayan bir kaynak kaydı bulunur. Ek bölüm, posta sunucusunun kanonik ana makinesinin IP adresini sağlayan bir Tür A kaydını içerir.

Kullandığınız ana bilgisayardan bir DNS sorgusu doğrudan bir DNS sunucusuna Windows ve UNIX platformunda bulunan nslookup programı ile kolayca gerçekleştirilebilir.

Inserting Records into the DNS Database

Yukarıdaki tartışma, kayıtların DNS veritabanından nasıl alındığına odaklandı. Şimdi belirli bir örnek bağlamında bu kayıtların veritabanına nasıl girildiğine bakalım.
Diyelim ki heyecan verici yeni bir startup şirketi olan Network Utopia'yı oluşturdunuz. İlk yapmak isteyeceğiniz şey, domain adınızı bir kayıt kurumunda kaydetmektir. Bir kayıt kurumu, domain adının benzersizliğini doğrular, domain adını DNS veritabanına girer (aşağıda tartışıldığı gibi) ve hizmetleri için sizden küçük bir ücret alır. 1999'dan önce, tek bir kayıt kurumu olan Network Solutions, com, net ve org alan adları için domain adı kaydının bir tekeli vardı. Ancak şimdi birçok kayıt kurumu müşteriler için rekabet ediyor ve İnternet İsim ve Numaraları Atama Kurumu (ICANN), çeşitli kayıt kurumlarını akredite ediyor. Networkutopia.com alan adını bir kayıt kurumuyla kaydettiğinizde, ayrıca ana ve ikincil yetkili DNS sunucularınızın adlarını ve IP adreslerini de kayıt kurumuna sağlamanız gerekecektir. Diyelim ki bu isimler ve IP adresleri dns1.networkutopia.com, dns2.networkutopia.com, 212.2.212.1 ve 212.212.212.2'dir. Her iki yetkili DNS sunucusu için kayıt kurumu, TLD com sunucularına Type NS ve Type A kayıtlarının girildiğinden emin olacaktır. Özellikle, networkutopia.com için birincil yetkili sunucu için kayıt kurumu, aşağıdaki iki kayıt kaydını DNS sistemi içine ekleyecektir:

(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)

Ayrıca, Web sunucunuz www.networkutopia.com için bir Type A kayıt kaydı ve posta sunucunuz mail.networkutopia.com için bir Type MX kayıt kaydı da yetkili DNS sunucularınıza girilmiş olmalıdır. (Son zamanlara kadar, her bir DNS sunucusunun içeriği statik olarak yapılandırılmıştı, örneğin, bir sistem yöneticisi tarafından oluşturulan bir yapılandırma dosyasından. Daha sonra, DNS protokolüne veri dinamik olarak eklenebilmesine veya veri tabanından silinebilmesine olanak tanımak için bir GÜNCELLEME seçeneği eklendi. Bu adımlar tamamlandığında, insanlar web sitenizi ziyaret edebilecek ve şirketinizdeki çalışanlara e-posta gönderebileceklerdir. Diyelim ki Avustralya'dan Alice, www.networkutopia.com web sayfasını görmek istiyor. Önceki tartışmada belirtildiği gibi, bilgisayarı önce DNS sorgusunu yerel DNS sunucusuna gönderir. Yerel DNS sunucusu daha sonra bir TLD com sunucusu ile iletişime geçer. (Yerel DNS sunucusu, bir TLD com sunucusunun adresi önbelleğe alınmamışsa bir kök DNS sunucusuna da başvurmak zorunda kalır.) Bu TLD sunucusu, kayıt kurumu tarafından tüm TLD com sunucularına bu kayıt kayıtlarının girildiği için yukarıda listelenen iki

kayıt kaydını içerir. TLD com sunucusu, Alice'in yerel DNS sunucusuna yanıt gönderirken, yanıt iki kayıt kaydını içerir. Yerel DNS sunucusu daha sonra www.networkutopia.com için karşılık gelen Type A kaydını sağlayan 212.212.212.1'e bir DNS sorgusu gönderir. Bu kayıt, istenen Web sunucusunun IP adresini, örneğin 212.212.71.4'ü sağlar ve yerel DNS sunucusu bunu Alice'in bilgisayarına iletir. Alice'in tarayıcısı artık 212.212.71.4 ana bilgisayarına bir TCP bağlantısı başlatabilir ve bağlantı üzerinden bir HTTP isteği gönderebilir.

DNS Zayıflıkları-Saldırıları

Bir saldırgan, her bir DNS kök sunucusuna çok sayıda paket göndermeye çalışabilir, bu da çoğu meşru DNS sorgusunun cevaplanmamasına neden olur. çoğu yerel DNS sunucusu, üst düzey alan sunucularının IP adreslerini önbelleğe alır, bu da sorgu sürecinin genellikle DNS kök sunucularını atlamasına olanak tanır. DNS'ye karşı potansiyel olarak daha etkili bir DDoS saldırısı, üst düzey alan sunucularına, örneğin .com alanını işleyen üst düzey alan sunucularına bir DNS sorgusu seli göndermektir. DNS sunucularına yönlendirilen DNS sorgularını filtrelemek daha zordur ve üst düzey alan sunucuları kök sunucuları kadar kolay atlanamaz. DNS potansiyel olarak başka yollarla da saldırıya uğrayabilir. Bir ara bağlantı saldırısında, saldırgan ana bilgisayarlardan gelen sorguları engeller ve yanlış cevaplar döndürür. DNS zehirlenmesi saldırısında, saldırgan bir DNS sunucusuna yanıltıcı cevaplar gönderir ve sunucunun yanlış kayıtları önbelleğine almasını sağlar. Bu saldırılardan herhangi biri, örneğin, bir web kullanıcısını saldırganın web sitesine yönlendirmek için kullanılabilir. DNS Güvenlik Uzantıları (DNSSEC), bu tür saldırılara karşı koruma sağlamak üzere tasarlanmış ve dağıtılmıştır. DNSSEC, DNS'in güvenli bir sürümüdür.

7.5 Peer-to-Peer File Distribution

Bugüne kadar tanımlanan uygulamalar - Web, e-posta ve DNS dahil - hepsi önemli ölçüde her zaman açık altyapı sunucularına bağlı olan istemci-sunucu mimarilerini kullanmaktadır. P2P mimarisi ile, her zaman açık altyapı sunucularına minimal (veya hiç) bağımlılık bulunmamaktadır. Bunun yerine, ara sıra bağlanan çiftlerden oluşan, eşler olarak adlandırılan ana bilgisayarlar doğrudan birbirleriyle iletişim kurarlar. Eşler bir hizmet sağlayıcı tarafından sahip olunmaz, bunun yerine kullanıcılar tarafından kontrol edilen PC'ler, dizüstü bilgisayarlar ve akıllı telefonlardır. Dosya, Linux işletim sisteminin yeni bir sürümü, mevcut bir işletim sistemi için bir yazılım yaması veya bir MPEG video dosyası olabilir. İstemci-sunucu dosya dağıtımında, sunucunun her bir eşe bir kopya göndermesi gerekir - bu, sunucu üzerinde büyük bir yük oluşturur ve büyük miktarda sunucu bant genişliği tüketir. P2P dosya dağıtımında, her eş, aldığı dosyanın herhangi bir bölümünü diğer eşlere yeniden dağıtabilir ve bu şekilde sunucuya dağıtım sürecine yardımcı olur. 2020 itibarıyla, en popüler P2P dosya dağıtım protokolü BitTorrent'dir.

7.6 Video Streaming and Content Distribution Networks

7.6.1 Internet Video

Önceden kaydedilmiş videolar sunuculara yerleştirilir ve kullanıcılar videoları istediklerinde görüntülemek için sunuculara istek gönderirler. Bir video, genellikle sabit bir hızda gösterilen görüntülerin bir dizisidir, örneğin saniyede 24 veya 30 görüntü. Videonun önemli bir özelliği, video kalitesini bit hızıyla değiştirebilecek şekilde sıkıştırılabilmesidir. bit hızı ne kadar yüksek olursa, görüntü kalitesi ve genel kullanıcı izleme deneyimi o kadar iyidir. Sıkıştırılmış İnternet videoları genellikle düşük kaliteli video için 100 kbps'den yüksek çözünürlüklü filmler için 4 Mbps ‘ye kadar değişir; 4K akışı 10 Mbps ‘den fazla bir bit hızını öngörür. Akışlı video için şimdiye kadar en önemli performans ölçüsü ortalama uçtan uca bant genişliğidir. Sürekli oynatma sağlamak için ağ, sıkıştırılmış vidyonun bit hızından en azından büyük bir ortalama bant genişliği sağlamalıdır. Aynı videoyu 300 kbps, 1 Mbps ve 3 Mbps hızlarında üç farklı sürüm oluşturabilinir. Kullanıcılar ardından mevcut bant genişliklerine bağlı olarak izlemek istedikleri sürümü seçebilirler.

7.6.2 HTTP Streaming and DASH

HTTP akışında, video sadece belirli bir URL'e sahip bir HTTP sunucusunda sıradan bir dosya olarak depolanır. Bir kullanıcı videoyu görmek istediğinde, istemci sunucuyla bir TCP bağlantısı kurar ve o URL için bir HTTP GET isteği gönderir. Sunucu daha sonra, temel ağ protokolleri ve trafik koşulları izin verdiği kadar hızlı bir şekilde video dosyasını HTTP yanıt ile gönderir. İstemci tarafında, baytlar bir istemci uygulama önbelleğinde toplanır. Bu önbellekteki bayt sayısı belirli bir eşik değerini aştığında, istemci uygulaması oynatmayı başlatır. Özellikle, akışlı video uygulaması periyodik olarak video karelerini istemci uygulama önbelleğinden alır, kareleri sıkıştırır ve kullanıcının ekranında gösterir. Böylece, video akış uygulaması, videoyu alırken ve videoya ait sonraki bölümlere karşılık gelen kareleri önbelleğe alırken videoyu gösterir.

HTTP akışının geniş ölçüde uygulanmasına rağmen, önemli bir kusuru vardır: Tüm istemciler, hem farklı istemciler arasında hem de aynı istemci için zaman içinde mevcut bant genişliğinde büyük varyasyonlar olmasına rağmen, videonun aynı kodlamasını alırlar. Bu durum HTTP tabanlı yeni bir akış türü olan **Dynamic Adaptive Streaming over HTTP (DASH)**’in ortaya çıkmasına sebep olmuştur.

DASH’ de, video birkaç farklı sürüme kodlanır, her sürümün farklı bir bit hızı ve dolayısıyla farklı bir kalite seviyesi vardır. İstemci, birkaç saniye uzunluğundaki video segmentlerinin parçalarını dinamik olarak istek eder. Mevcut bant genişliği yüksek olduğunda, istemci doğal olarak yüksek hızlı bir versiyondan parçaları seçer ve mevcut bant genişliği düşük olduğunda, düşük hızlı bir versiyondan seçer. İstemci, bir seferde birkaç parça seçerken HTTP GET istek mesajları kullanır. DASH, farklı İnternet erişim hızlarına sahip istemcilerin farklı kodlama hızlarında video akışı yapmasına olanak tanır. DASH ayrıca, oturum sırasında mevcut uçtan uca bant genişliği değişirse istemcinin buna uyum sağlamasına olanak tanır. Bu özellik özellikle mobil kullanıcılar için önemlidir, çünkü genellikle baz istasyonlarına göre hareket ettikçe bant genişliklerinin dalgalanmasını görürler. DASH ile her video sürümü, her biri farklı bir URL ile

HTTP sunucusunda depolanır. HTTP sunucusu ayrıca her sürüm için bir URL ve bit hızıyla birlikte bir manifest dosyasına sahiptir. İstemci önce manifest dosyasını (Manifest dosyası, bir medya akışında bulunan içeriğin organizasyonunu, yapısını ve bölümlerini tanımlayan bir metin dosyası veya XML dosyasıdır.) isteyerek ve çeşitli sürümler hakkında bilgi alır. İstemci daha sonra her bir parçayı belirterek HTTP GET istek mesajında bir URL ve bayt aralığı belirterek bir seferde bir parça seçer. Parçaları indirirken, istemci ayrıca alınan bant genişliğini ölçer ve bir sonraki parçayı istemek için bir oran belirleme algoritması çalıştırır. Eğer istemcinin çok fazla videoyu önbelleğe almış ve ölçülen alınan bant genişliği yüksekse, yüksek bit hızlı bir versiyondan parça seçer. Eğer istemcinin az miktarda video önbelleği varsa ve ölçülen alınan bant genişliği düşükse, düşük bit hızlı bir versiyondan parça seçer.

2.6.3 Content Distribution Networks

Bir İnternet video şirketi için, akışlı video hizmeti sağlamanın en basit yaklaşımı tek bir devasa veri merkezi oluşturmaktır, Tüm videolarını veri merkezinde depolamak ve videoları doğrudan veri merkezinden dünya çapındaki istemcilere akıtmaktır. Ancak, bu yaklaşımın üç büyük sorunu vardır.

- İlk olarak, istemci veri merkezinden uzakta ise, sunucudan istemciye paketler birçok iletişim bağlantısını geçecek ve muhtemelen birçok İSP üzerinden geçecektir, bazı İSP'lerin farklı kıtalarda olabileceği düşünülürse bu bağlantılardan biri video tüketim hızından daha az bir geçiş sağlarsa, uçtan uca bant genişliği de tüketim hızının altında olacaktır, bu da kullanıcı için sinir bozucu donma gecikmelerine neden olacaktır. (Bir akışın uçtan uca bant genişliği, darboğaz bağlantısındaki bant genişliği tarafından belirlenir.) Bu durumun gerçekleşme olasılığı, uçtan uca yol üzerindeki bağlantı sayısı arttıkça artar.
- İkinci bir dezavantaj, popüler bir videonun aynı iletişim bağlantıları üzerinden muhtemelen birçok kez gönderilecek olmasıdır. Bu, sadece ağ bant genişliğini boşa harcamakla kalmaz, aynı zamanda İnternet video şirketi de aynı baytları tekrar tekrar İnternet'e gönderdiği için sağlayıcı İSP'ye ödeme yapar.
- Üçüncü bir sorunu ise, tek bir veri merkezinin tek bir arıza noktasını temsil etmesidir. Eğer veri merkezi veya İnternet'e olan bağlantıları çökerse, hiçbir video akışı dağıtamaz.

Dünya çapında dağıtılmış kullanıcılara büyük miktarda video verisinin dağıtılma zorluğunu karşılamak için, neredeyse tüm büyük video akışı şirketleri İçerik Dağıtım Ağları (CDN'ler) kullanır. Bir CDN, coğrafi olarak dağıtılmış birden fazla konumda sunucuları yönetir, videoların kopyalarını (ve belgeler, resimler ve sesler de dahil olmak üzere diğer türdeki Web içeriğini) sunucularında saklar ve her kullanıcı isteğini en iyi kullanıcı deneyimini sağlayacak bir CDN konumuna yönlendirmeye çalışır. CDN, özel bir CDN olabilir, yani içerik sağlayıcı tarafından sahip olunan; örneğin, Google'ın CDN'si YouTube videolarını ve diğer türdeki içeriği dağıtır. Alternatif olarak, CDN, birden fazla içerik sağlayıcı adına içeriği dağıtan üçüncü taraf bir CDN olabilir; Akamai, Limelight ve Level-3 tümü üçüncü taraf CDN'lerini işletir.

CDN'ler genellikle iki farklı sunucu yerleştirme felsefesinden birini benimser :

Enter Deep: Erişim İnternet Servis Sağlayıcılarının ağlarına derinlemesine girerek, dünya çapındaki erişim İSP'lerinde sunucu kümeleri dağıtarak gerçekleştirilir. Binlerce konumda kümelerle bu yaklaşımı benimser. Amaç, son kullanıcılara yakın olmak, böylece kullanıcı algılanan gecikmeyi ve bant genişliğini azaltarak son kullanıcı ile CDN sunucusu arasındaki bağlantıların ve yönlendiricilerin sayısını azaltarak iyileştirmektir. Bu son derece dağıtılmış tasarım nedeniyle, kümelerin bakımını ve yönetimini sağlamak zorlu hale gelir.

Bring Home: Erişim İnternet Servis Sağlayıcılarını (İSP'lerini) evlerine getirmeyi hedefler. Erişim İSP'lerinin içine girmek yerine, genellikle İnternet Değişim Noktaları (IXP'ler) gibi büyük kümelerini inşa ederler. Enter Deep tasarım felsefesiyle karşılaştırıldığında, eve getirme tasarımı genellikle daha düşük bakım ve yönetim giderleri ile son kullanıcılara daha yüksek gecikme ve daha düşük bant genişliği sağlar.

Küme yerleştirmeleri yapıldıktan sonra, CDN içeriği kümeleri arasında kopyalar. CDN, her videoyu her küme içine koymak istemeyebilir, çünkü bazı videolar nadiren görüntülenir veya sadece bazı ülkelerde popülerdir. Aslında, birçok CDN, videoları kümelerine yerleştirmek yerine basit bir çekme stratejisi kullanır: Bir istemci, videoyu saklamayan bir kümeye bir video isteğinde bulunursa, o zaman küme videoyu (merkezi bir depodan veya başka bir kümelerden) alır ve yerel olarak bir kopya saklar, aynı anda videoyu istemciye akıtır. Benzer şekilde, bir kümenin depolama alanı dolduğunda, nadiren istenen videoları kaldırır.

CDN Operation

Bir kullanıcının ana makinesindeki bir tarayıcı, belirli bir videoyu (URL ile tanımlanan) almak üzere talimatlandırıldığında, CDN'nin isteği onaylaması ve yönlendirmesi gerekir, böylece (1) o anda o istemci için uygun bir CDN sunucu kümesini belirleyebilir ve (2) istemcinin isteğini o kümedeki bir sunucuya yönlendirebilir. Çoğu CDN, istekleri onaylamak ve yönlendirmek için DNS'ten yararlanır. Bir içerik sağlayıcı olan NetCinema, videolarını müşterilerine dağıtmak için üçüncü taraf bir CDN şirketi olan KingCDN'yi kullanıyor olsun. NetCinema Web sayfalarında, her bir videosu, videonun kendisi için benzersiz bir tanımlayıcı içeren bir URL'ye atanır; örneğin, Transformers 7 için http://video.netcinema.com/6Y7B23V gibi. Şimdi, Şekil 2.25'te gösterildiği gibi altı adım gerçekleşir:

1. Kullanıcı NetCinema'nın Web sayfasını ziyaret eder.
2. Kullanıcı, http://video.netcinema.com/6Y7B23V bağlantısına tıkladığında, kullanıcının ana bilgisayarı video.netcinema.com için bir DNS sorgusu gönderir.
3. Kullanıcının Yerel DNS Sunucusu (LDNS), video.netcinema.com için yetkili bir DNS sunucusuna DNS sorgusunu iletir. NetCinema'nın yetkili DNS sunucusu, hostname video.netcinema.com'daki "video" dizesini gözlemleyerek, DNS sorgusunu KingCDN'e "devretmek" için, bir IP adresi döndürmek yerine, LDNS'ye KingCDN'in alan adında bir hostname döndürür, örneğin, a1105.kingcdn.com.
4. Bu noktadan itibaren, DNS sorgusu KingCDN'nin özel DNS altyapısına girer. Kullanıcının LDNS'si daha sonra ikinci bir sorgu gönderir, şimdi a1105.kingcdn.com için ve KingCDN'nin DNS sistemi sonunda bir KingCDN içerik sunucusunun IP

adreslerini LDNS'ye döndürür. Bu nedenle, CDN sunucusu, müşterinin içeriğini alacağı yer, burada, KingCDN'nin DNS sistemi içinde belirlenir.

5. LDNS, içerik sunan CDN düğümünün IP adresini kullanıcının ana bilgisayarına iletilir.
6. Bir kez müşteri, bir KingCDN içerik sunucusunun IP adresini aldığında, o IP adresiyle doğrudan bir TCP bağlantısı kurar ve video için bir HTTP GET isteği gönderir. Eğer DASH kullanılıyorsa, sunucu önce müşteriye her bir video sürümü için bir URL listesi olan bir manifest dosyası gönderecek ve müşteri farklı sürümlerden parçaları dinamik olarak seçecektir.

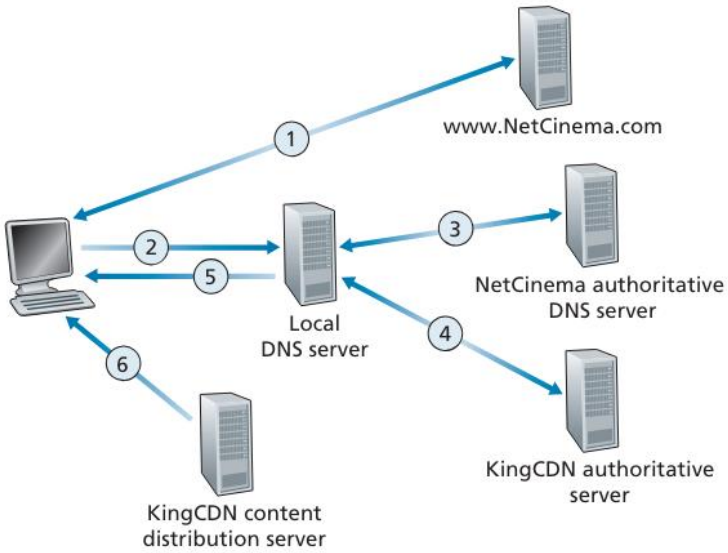


Figure 2.25 ♦ DNS redirects a user's request to a CDN server

Cluster Selection Strategies

CDN, müşterinin DNS sorgulaması aracılığıyla müşterinin LDNS (Yerel Alan Adı Sunucusu) sunucusunun IP adresini öğrenir. Bu IP adresini öğrendikten sonra, CDN bu IP adresine göre uygun bir küme seçmelidir. CDNLer genellikle özel küme seçimi stratejileri kullanır.

1) Müşteriyi coğrafi olarak en yakın kümeye atamaktır. CDN coğrafi olarak en yakın kümeyi, yani LDNS'ye "kuş uçuşu" en az kilometre uzaklıkta olan kümeyi seçer. bazı müşteriler için çözüm kötü performans gösterebilir çünkü coğrafi olarak en yakın küme, ağ yolu uzunluğu veya atlama sayısı açısından en yakın küme olmayabilir. bu basit strateji, İnternet yollarındaki gecikme ve mevcut bant genişliğindeki zaman içindeki değişiklikleri göz ardı eder ve her zaman belirli bir müşteriye aynı kümeyi atar.

2) Bir müşteriye mevcut trafik koşullarına göre en iyi kümeyi belirlemek için, CDNLer bunun yerine kümeleri ile müşteriler arasındaki gecikme ve kayıp performansının periyodik gerçek zamanlı ölçümlerini yapabilirler. Örneğin, bir CDN, kümesinin her birinin dünya çapındaki tüm LDNS'lere periyodik olarak prob göndermesini (örneğin, ping mesajları veya DNS sorguları)

sağlayabilir. Bu yaklaşımın bir dezavantajı, birçok LDNS'nin bu tür prob isteklerine yanıt vermeyecek şekilde yapılandırılmış olmasıdır.

OTT (Over The Top)

İnternet üzerinden doğrudan kullanıcıya sunulan hizmetleri tanımlar. Örneğin, Netflix, Amazon Prime Video gibi platformlar OTT hizmetlerine örnektir.

OTT hizmetleri, yoğun internet trafiği ile başa çıkmak zorundadır. Bu zorluklar arasında:

Hangi içeriğin hangi CDN düğümüne yerleştirileceği: İçeriğin kullanıcıya en hızlı şekilde ulaştırılabilmesi için doğru CDN düğümüne yerleştirilmesi gereklidir.

İçeriğin hangi CDN düğümünden ve hangi hızda alınacağı: Kullanıcı talebine göre en uygun CDN düğümünden içerik alınmalı ve bu süreç en verimli şekilde yönetilmelidir.

2.6.4 Socket Programming (Network Applications)

Iki program çalıştırıldığında, bir istemci süreci ve bir sunucu süreci oluşturulur ve bu süreçler, soketlerden okuyup yazarak birbirleriyle iletişim kurar. İki tür ağ uygulaması vardır

Bir RFC veya başka bir standartlar belgesinde belirtilen bir protokol standardına göre işletimi belirlenen bir uygulamadır; böyle bir uygulama bazen "açık" olarak adlandırılır, çünkü işletimini belirleyen kurallar herkes tarafından bilinir.

Diğer tür ağ uygulaması ise özel ağ uygulamasıdır. Bu durumda, istemci ve sunucu programları, bir RFC'de veya başka bir yerde açıkça yayınlanmamış bir uygulama katmanı protokolü kullanır. Tek bir geliştirici (veya geliştirme ekibi) hem istemci hem de sunucu programlarını oluşturur ve geliştirici, kodda ne olacağı üzerinde tam kontrol sahibidir. Ancak kod, açık bir protokolü uygulamadığı için, diğer bağımsız geliştiriciler, uygulamayla birlikte çalışabilen kodlar geliştiremeyeceklerdir.

Geliştirme aşamasında, geliştiricinin vermesi gereken ilk kararlardan biri, uygulamanın TCP üzerinden mi yoksa UDP üzerinden mi çalışacağıdır.

TCP'nin bağlantı odaklı olduğunu ve iki uç sistem arasında veri akışının güvenilir bir bayt akışı kanalı sağladığını hatırlayın. UDP ise bağlantısızdır ve bir uç sistemden diğerine teslimat garantisi olmadan bağımsız veri paketleri gönderir.

Bir istemci veya sunucu programı bir RFC ile tanımlanan bir protokolü uyguladığında, protokolle ilişkili iyi bilinen port numarasını kullanması gerektiğini, buna karşılık, özel bir uygulama geliştirirken, geliştiricinin bu tür iyi bilinen port numaralarını kullanmaktan kaçınması gereklidir.

2.6.4.1 Socket Programming with UDP

Gönderen süreç, UDP kullanırken, veri paketini soket kapısından dışarı itmeden önce, pakete bir hedef adres eklemelidir. Paket, gönderenin soketinden geçtikten sonra, İnternet bu hedef adresi kullanarak paketi İnternet üzerinden alıcı süreçteki sokete yönlendirecektir. Paket alıcı sokete ulaştığında, alıcı süreç paketi soket aracılığıyla alacak, paket içeriğini inceleyecek ve uygun işlemi yapacaktır.

Hedef ev sahibinin IP adresi, hedef adresin bir parçasıdır. Pakete hedef IP adresini ekleyerek, İnternet'teki yönlendiriciler paketi İnternet üzerinden hedef ev sahibine yönlendirebilecektir. Ancak, bir ev sahibi, her biri bir veya daha fazla sokete sahip birçok ağ uygulama süreci çalıştırabileceğinden, hedef ev sahibindeki belirli soketi de tanımlamak gereklidir. Bir soket oluşturulduğunda, ona bir tanımlayıcı, yani bir port numarası atanır. Dolayısıyla, paketin hedef adresi aynı zamanda soketin port numarasını da içerir. Gönderen süreç pakete, hedef ev sahibinin IP adresi ve hedef soketin port numarasından oluşan bir hedef adres ekler. Göndericinin kaynak adresi, kaynak ev sahibinin IP adresi ve kaynak soketin port numarasından oluşan pakete eklenir. Ancak, kaynak adresi pakete eklemek tipik olarak UDP uygulama kodu tarafından yapılmaz; bunun yerine, işletim sistemi tarafından otomatik olarak yapılır.

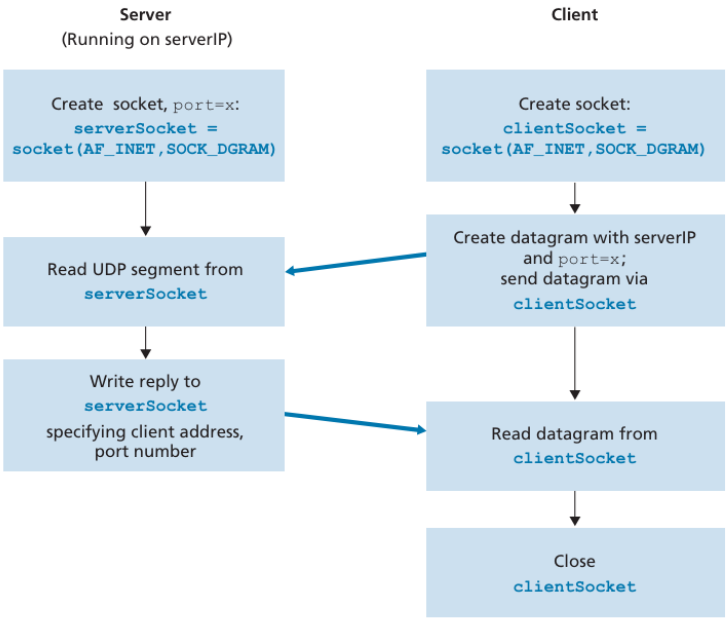


Figure 2.27 ♦ The client-server application using UDP

2.6.4.2 Socket Programming with TCP

UDP'nin aksine, TCP bağlantı odaklı bir protokoldür. Bu, istemci ve sunucunun birbirine veri göndermeye başlamadan önce el sıkışması ve bir TCP bağlantısı kurması gerektiği anlamına gelir. TCP bağlantısının bir ucu istemci soketine ve diğer ucu sunucu soketine bağlıdır. TCP bağlantısı oluşturulurken, istemci soket adresi (IP adresi ve port numarası) ve sunucu soket adresi (IP adresi ve port numarası) ile ilişkilendirilir. TCP bağlantısı kurulduğunda, bir taraf diğer tarafa veri göndermek istediğinde, veriyi soket aracılığıyla TCP bağlantısına bırakır. Bu, UDP'den farklıdır; UDP'de sunucu, paketi sokete bırakmadan önce bir hedef adres eklemelidir.

İstemcinin görevi sunucuyla iletişimi başlatmaktır. Sunucunun istemcinin ilk iletişimine yanıt verebilmesi için hazır olması gerekir. Bu iki şey anlamına gelir. Birincisi, UDP durumunda olduğu gibi, TCP sunucusu, istemci iletişimi başlatmaya çalışmadan önce bir süreç olarak çalışıyor olmalıdır. İkincisi, sunucu programının, istemci sürecinin rastgele bir ana makinede çalışarak başlatacağı ilk iletişimi karşılayan özel bir kapıya—daha doğrusu, özel bir sokete—sahip olması gerekmektedir. Süreç/soket için ev/kapı benzetimimizi kullanarak, istemcinin ilk iletişimini bazen "karşılama kapısına vurmak" olarak adlandıracağız.

Sunucu süreci çalışırken, istemci süreci sunucuya bir TCP bağlantısı başlatabilir. Bu, istemci programında bir TCP soketi oluşturularak yapılır. İstemci, TCP soketini oluşturduğunda, sunucudaki karşılama soketinin adresini, yani sunucu ana bilgisayarının IP adresini ve soketin port numarasını belirtir. Soketini oluşturduktan sonra, istemci üç yönlü el sıkışmayı başlatır ve sunucu ile bir TCP bağlantısı kurar. Taşıma katmanında gerçekleşen bu üç yönlü el sıkışma, istemci ve sunucu programları için tamamen görünmezdir.

Üç yönlü el sıkışma sırasında, istemci süreci sunucu sürecinin karşılama kapısına vurur. Sunucu "vurmayı duyduğunda", o belirli istemciye adanmış yeni bir kapı daha doğrusu, yeni bir soket oluşturur. Aşağıdaki örneğimizde, karşılama kapısı **serverSocket** adını verdiğimiz bir TCP soket nesnesidir; bağlantı kuran istemciye adanmış yeni oluşturulan soket ise **connectionSocket** olarak adlandırılır. Karşılama soketi (welcoming socket) (sunucuyla iletişim kurmak isteyen tüm istemciler için ilk temas noktası) daha sonra her istemciyle iletişim kurmak için oluşturulan her yeni sunucu tarafı bağlantı soketidir (server-side connection socket).

Uygulamanın perspektifinden, istemcinin soketi ile sunucunun bağlantı soketi doğrudan bir boru hattı ile bağlanmış gibidir. Şekil 2.28'de gösterildiği gibi, istemci süreci kendi soketine rastgele baytlar gönderebilir ve TCP, sunucu sürecinin (bağlantı (connection) soketi aracılığıyla) her baytı gönderildiği sırayla almasını garanti eder. Bu şekilde TCP, istemci ve sunucu süreçleri arasında güvenilir bir hizmet sağlar. Ayrıca, insanların aynı kapıdan girip çıkabildiği gibi, istemci süreci yalnızca bayt göndermekle kalmaz, aynı zamanda soketinden bayt alır; benzer şekilde, sunucu süreci yalnızca bayt almakla kalmaz, aynı zamanda bağlantı soketine bayt da gönderir.

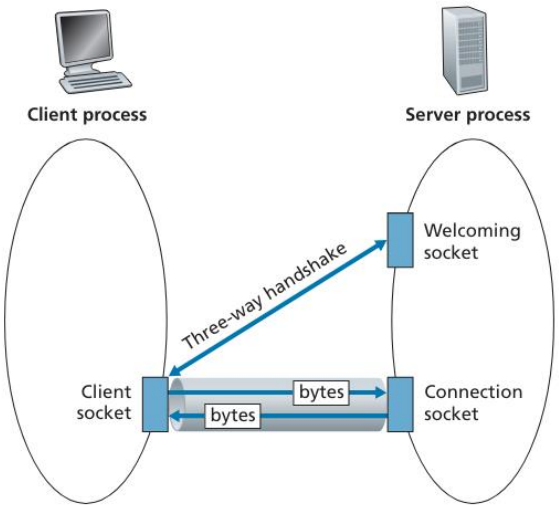


Figure 2.28 ♦ The TCPServer process has two sockets

UDP ile soket programlama

UDP: istemci ile sunucu arasında

"bağlantı" yok:

- veri göndermeden önce el sıkışma yok
- gönderen, IP hedef adresini ve port numarasını her pakete açıkça ekler
- alıcı, alınan paketten gönderenin IP adresini ve bağlantı noktası numarasını çıkarır

UDP: iletilen veriler kaybolabilir veya sıra dışı olarak alınabilir

Uygulama bakış açısı:

- UDP, istemci ve sunucu işlemleri arasında bayt gruplarının ("datagramlar") güvenilir olmayan aktarımını sağlar

TCP ile soket programlama

İstemci sunucuyla iletişime geçmelidir

- öncelikle sunucu işlemi çalışıyor olmalıdır
- sunucu, müşterinin iletişimini karşılayan bir yuva (kapı) oluşturmuş olmalıdır

İstemci sunucuyla şu şekilde iletişim kurar:

- TCP soketinin oluşturulması, IP adresinin belirtilmesi, sunucunun port numarasının belirlenmesi
- istemci soket oluşturduğunda: istemci TCP'si sunucu TCP'sine bağlantı kurar

- istemci tarafından iletişim kurulduğunda, sunucu TCP'si, sunucu işleminin söz konusu istemciyle iletişim kurması için yeni bir yuva oluşturur
  - sunucunun birden fazla istemciyle konuşmasına olanak tanır
  - istemcileri ayırt etmek için kullanılan kaynak bağlantı noktası numaraları (daha fazlası Bölüm 3'te)

Uygulama bakış açısı

TCP, istemci ve sunucu işlemleri arasında güvenilir, sıralı bayt akışı aktarımı ("boru") sağlar

3.0 Transport Layer

Farklı ana bilgisayarlarda çalışan uygulama işlemleri arasında mantıksal iletişim sağlar. End system (uç sistemlerdeki) transport protokol eylemleri:

-Sender: Application mesajlarını segmentlere ayırır, network layera aktarır burada segment bir ağ katmanı paketi (bir datagram) içine kapsüllenir ve hedefe gönderilir.
-Receiver: Segmentleri mesaj halinde yeniden birleştirir. Application layer katmanına geçer.

Taşıma katmanı protokolü, farklı ana bilgisayarlarda çalışan süreçler arasında mantıksal iletişim sağlarken, ağ katmanı protokolü ana bilgisayarlar arasında mantıksal iletişim sağlar. Bu ayrımı bir ev analojisi yardımıyla inceleyelim.

Doğu Kıyısı'nda ve Batı Kıyısı'nda iki ev olduğunu düşünün; her evde bir düzine çocuk yaşıyor. Doğu Kıyısı'ndaki evde yaşayan çocuklar, Batı Kıyısı'ndaki evde yaşayan çocukların kuzenleridir. İki evdeki çocuklar birbirlerine yazı yazmayı çok severler—her çocuk her hafta her bir kuzenine mektup yazar, her bir mektup ayrı bir zarfta geleneksel posta hizmeti ile teslim edilir. Böylece, her ev diğer eve her hafta 144 mektup gönderir. (Bu çocuklar e-posta kullanarak çok para biriktirirdi!) Her evde, posta toplama ve dağıtımından sorumlu bir çocuk vardır—Batı Kıyısı evinde Ann ve Doğu Kıyısı evinde Bill. Her hafta Ann, tüm kardeşlerini ziyaret eder, postaları toplar ve eve günlük ziyaretler yapan posta servisi görevlisine verir. Mektuplar Batı Kıyısı evine ulaştığında, Ann aynı zamanda postayı kardeşlerine dağıtma görevine de sahiptir. Bill, Doğu Kıyısı'nda benzer bir işe sahiptir.

Bu örnekte, posta servisi iki ev arasında mantıksal iletişim sağlar—posta servisi mektupları evden eve taşır, kişiden kişiye değil. Öte yandan, Ann ve Bill kuzenler arasında mantıksal iletişim sağlar—Ann ve Bill postayı kardeşlerinden alır ve kardeşlerine teslim eder. Kuzenlerin perspektifinden bakıldığında, Ann ve Bill posta servisi gibi görünür, oysa Ann ve Bill sadece uçtan uca teslimat sürecinin bir parçasıdır (uç sistem kısmı). Bu ev örneği, taşıma katmanının ağ katmanıyla nasıl ilişkilendirildiğini açıklamak için güzel bir benzetme sağlar:

Application messages: Zarflardaki mektuplar

Processes = Kuzenler

Host (end systems) = Evler

Transport Layer Protocol = Ann ve Bill

Network Layer Protocol = Posta Servisi (Posta taşıyıcıları dahil)

Bu benzetmeye devam edersek, Ann ve Bill'in tüm işlerini kendi evlerinde yaptıklarını fark edin; örneğin, ara posta merkezlerinde postaları ayırma veya bir posta merkezinden diğerine taşıma işine dahil değildirler. Benzer şekilde, taşıma katmanı protokolleri uç sistemlerde bulunur. Bir uç sistemde, taşıma protokolü mesajları uygulama süreçlerinden ağın kenarına (yani, ağ katmanına) ve tersi yönde taşır, ancak mesajların ağın çekirdeği içinde nasıl taşındığı konusunda bir etkisi yoktur. Ara yönlendiriciler (router) taşıma katmanının uygulama mesajlarına ekleyebileceği bilgileri ne tanır ne de bu bilgilere göre hareket eder.

Aile hikayemize devam edersek, şimdi Ann ve Bill tatile gittiklerinde, yerlerine başka bir kuzen çifti örneğin, Susan ve Harvey geçip ev içi posta toplama ve dağıtımını sağlasınlar. Ne yazık ki iki aile için, Susan ve Harvey toplama ve dağıtımı Ann ve Bill ile aynı şekilde yapmazlar. Daha küçük çocuklar oldukları için, Susan ve Harvey postaları daha az sıklıkla toplar ve dağıtır ve bazen mektupları kaybederler (bazen mektuplar aile köpeği tarafından çiğnenir). Dolayısıyla, Susan ve Harvey kuzen çifti, Ann ve Bill ile aynı hizmet modelini sağlamaz. Benzer şekilde, bir bilgisayar ağı birden fazla taşıma protokolü sağlayabilir ve her protokol uygulamalara farklı bir hizmet modeli sunar.

Ann ve Bill'in sağlayabileceği hizmetler açıkça posta hizmetinin sağlayabileceği hizmetlerle sınırlıdır. Örneğin, posta hizmeti iki ev arasındaki posta teslimatı için maksimum bir süre garantisi vermezse (örneğin, üç gün), Ann ve Bill kuzen çiftleri arasındaki posta teslimatı için maksimum bir gecikme garantisi veremezler. Benzer şekilde, bir taşıma protokolünün sağlayabileceği hizmetler genellikle altındaki ağ katmanı protokolünün hizmet modeli tarafından sınırlanır. Ağ katmanı protokolü, ana bilgisayarlar arasında gönderilen taşıma katmanı segmentleri için gecikme veya bant genişliği garantileri sağlayamazsa, taşıma katmanı protokolü, süreçler arasında gönderilen uygulama mesajları için gecikme veya bant genişliği garantileri sağlayamaz.

Yine de taşıma protokolü, altındaki ağ protokolü ağ katmanında karşılık gelen hizmeti sunmadığında bile belirli hizmetler sunabilir. Örneğin, bu bölümde göreceğimiz gibi, taşıma protokolü, ağ protokolü güvenilmez olsa bile, yani ağ protokolü paketleri kaybetse, bozulsa veya çoğaltılsa bile bir uygulamaya güvenilir veri transferi hizmeti sunabilir. Başka bir örnek olarak taşıma protokolü, ağ katmanı segmentlerinin gizliliğini garanti edemese bile, uygulama mesajlarının izinsiz kişiler tarafından okunmamasını sağlamak için şifreleme kullanabilir.