ANKARA
HACI BAYRAM VELİ ÜNİVERSİTESİ
UZAKTAN EĞİTİM UYGULAMA VE
ARAŞTIRMA MERKEZİ

İŞLETMELERDE VERİ İLETİŞİMİ

Doç. Dr. Şaban Öztürk

saban.ozturk@hbv.edu.tr

Dersin Genel Hedefleri

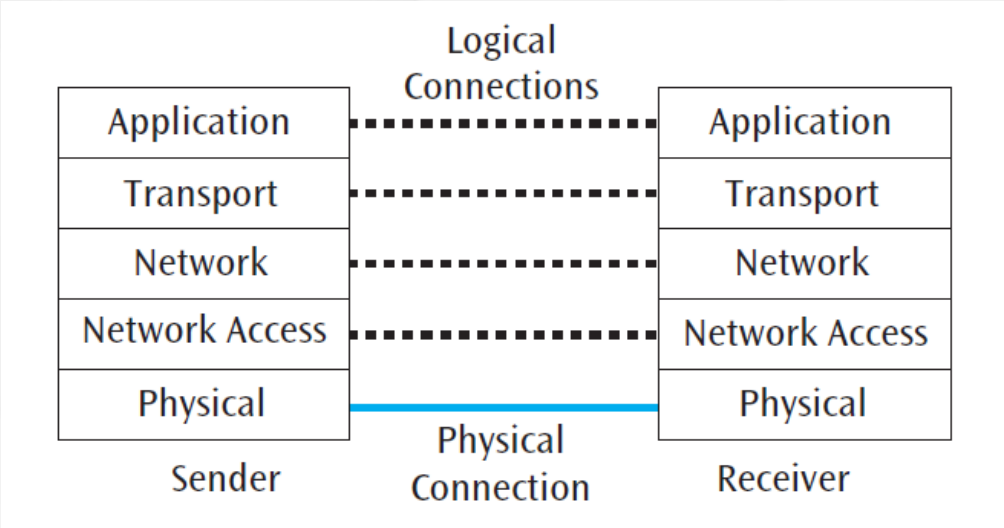
- Öğrencilerin veri iletişiminin temel kavramlarını, protokollerini ve teknolojilerini anlamalarını sağlamak.
- İşletmelerde kullanılan çeşitli iletişim ağlarının yapısını, bileşenlerini ve işlevlerini incelemek.
- Günümüz veri iletişimi teknolojilerini (örneğin, Ethernet, Wi-Fi, Bluetooth) ve bunların işletme ortamındaki uygulama alanlarını öğrenmek.
- Veri iletişimi ağlarının güvenliğini sağlamak ve yönetmek için gerekli stratejileri ve araçları öğrenmek.
- İşletmelerde veri iletişimi ile ilgili ortaya çıkan sorunları tanımlamak ve bu sorunlara etkili çözümler geliştirmek.

Doç. Dr. Şaban Öztürk
saban.ozturk@hbv.edu.tr

Bölüm Hedefleri

- Uygulama Katmanı Çalışma Prensibinin kavranması
- Örnek Uygulamalar ile katmanların pekiştirilmesi

1. Application Layer



Kaynak: https://mrrichardtrinh.wordpress.com/2016/09/05/first-blog-post/

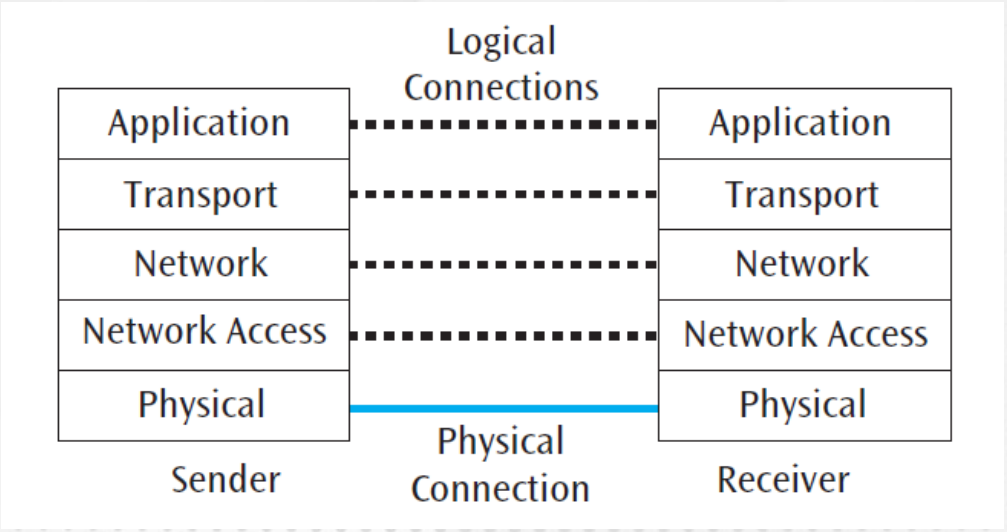
Bir ağdaki en üst katmanlardan biridir ve kullanıcıların ağdaki uygulamalara erişmesini sağlar. Bu katman, kullanıcıların veri alışverişinde bulunmalarını sağlayan çeşitli uygulamaları barındırır. Aşağıdaki gibi işlevleri vardır:

Protokol Dönüşümü: Farklı uygulamalar farklı iletişim protokolleri kullanabilir. Örneğin, bir tarayıcı HTTP (Hypertext Transfer Protocol) kullanırken, e-posta istemcisi SMTP (Simple Mail Transfer Protocol) kullanabilir. Application layer, farklı protokolleri birbirine dönüştürebilir, böylece uygulamaların birbiriyle iletişim kurmasını sağlar.

Kullanıcı Arayüzü: Kullanıcıların ağ uygulamalarına erişmesini sağlar. Web tarayıcıları, e-posta istemcileri, anlık mesajlaşma uygulamaları gibi birçok yaygın uygulama, bu katmanda yer alır.

saban.ozturk@hbv.edu.tr

1. Application Layer



Kaynak: https://mrrichardtrinh.wordpress.com/2016/09/05/first-blog-post/

Veri Kodlama ve Formatlama: Verilerin belirli bir formatta kodlanması ve sunulması gerekir. Örneğin, web sayfaları HTML (HyperText Markup Language) kullanır. Application layer, bu tür formatlama ve kodlamayı gerçekleştirir.

Kullanıcı Kimlik Doğrulama ve Yetkilendirme: Kullanıcıların kimlik doğrulama sürecini yönetir ve yetkilendirme sağlar. Örneğin, bir web sitesine giriş yaparken kullanıcı adı ve şifre ile kimlik doğrulama gerçekleştirilir.

Veri Güvenliği ve Şifreleme: Veri güvenliği, application layer'da önemli bir konudur. Özellikle web uygulamaları gibi açık ağlarda iletişimde bulunan uygulamalar, verilerin güvenliğini sağlamak için şifreleme ve diğer güvenlik önlemlerini kullanır.

1. 1. Application Layer Temel Yapı Taşları

1- Veri depolama. (hertürlü veri için)

2- Veri erişim mantığı, (veritabanı sorgularını içerir).

3- İşlev uygulama mantığı.

4- İşlev sunum mantığı (bazen kullanıcı arayüzü olarak da adlandırılır), bilgilerin kullanıcıya sunulması ve kullanıcının komutlarını kabul edilmesi.

5- İşlev hizmetler mantığı, bu da diğer uygulamalara hizmetlerin sağlanmasıdır (örneğin, uygulama programlama arayüzleri (API)). Bu, kullanıcı arayüzüne benzer, ancak kullanıcılar yerine diğer uygulama yazılım paketlerinin isteklerini yapmasını sağlar.

1. 2. Yaygın Application Layer Mimarileri

Host-Based Architectures

Client-Based Architectures

Client-Server Architectures

Cloud Computing Architectures

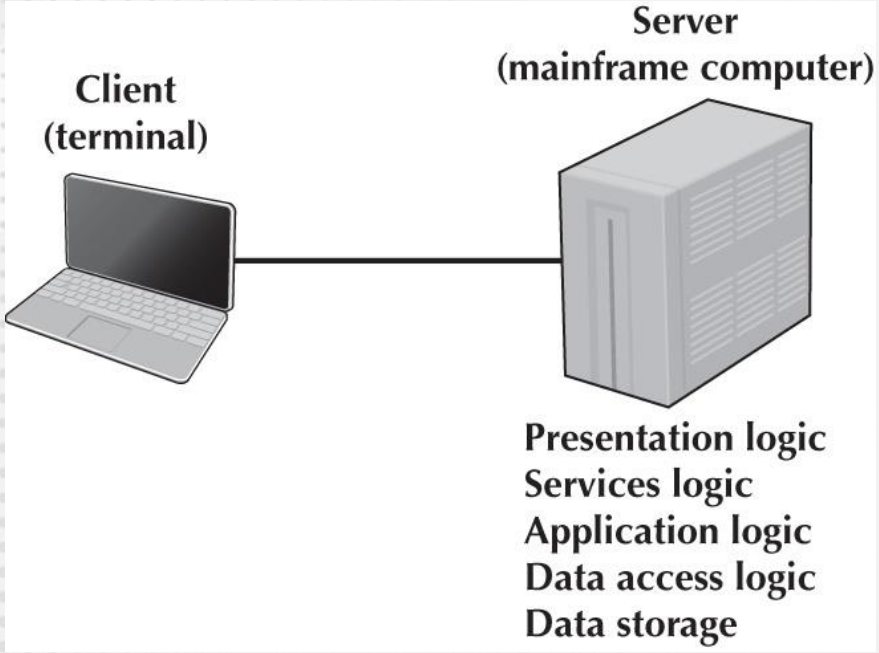
Peer-to-Peer Architectures

Doç. Dr. Şaban Öztürk

saban.ozturk@hbv.edu.tr

1. 3. Host-Based Architectures

İlkel veri iletişim ağları, genellikle büyük ana bilgisayarın tüm işlevleri gerçekleştirdiği ana bilgisayar tabanlıydı. İstemciler kullanıcılara ana bilgisayara mesaj gönderme ve almayı sağlardı. İstemciler sadece tuş vuruşlarını yakalardı, bunları işlemek için sunucuya gönderirdi ve sunucudan hangi bilgilerin görüntüleneceğine dair talimatlar alırdı.



Kaynak: https://quizlet.com/197467111/it-280-ch-02-application-layer-application-architectures-flash-cards/

saban.ozturk@hbv.edu.tr

1. 3. Host-Based Architectures

Bu çok basit mimari genellikle çok iyi çalışır. Uygulama yazılımı, tüm verilerle birlikte tek bir sunucuda geliştirilir ve depolanır.

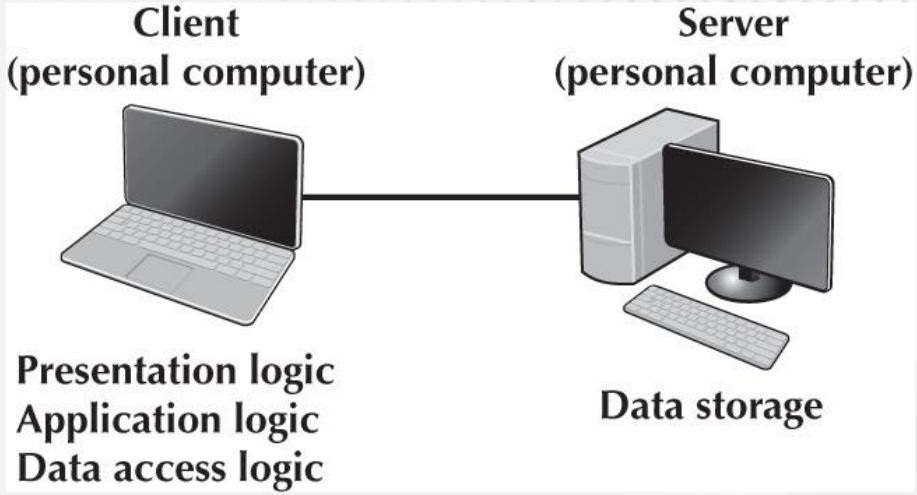
Öğrencilerden Örnekler???????

Ana bilgisayar tabanlı ağların iki temel sorunu vardır. İlk olarak, sunucunun tüm mesajları işlemesi gerekir. Daha fazla ağ uygulamasına olan talep arttıkça, birçok sunucu aşırı yüklenir ve tüm kullanıcı taleplerini hızla işleyemez hale gelir. Kullanıcıların erişimini önceliklendirme zorlaşır. Yanıt süresi daha yavaş hale gelir ve ağ yöneticileri sunucuyu yükseltmek için giderek daha fazla para harcamak zorunda kalır. Maalesef, genellikle bu mimaride sunucular olan ana bilgisayarların yükseltmeleri "büyük" olur. Yani, yükseltmeler büyük artışlarla gelir ve pahalıdır.

1. 4. Client-Based Architectures

Kişisel bilgisayarların yaygınlaşmasından sonra, çoğu organizasyonun toplam bilgisayar işlem gücünün %90'dan fazlası merkezi ana bilgisayar değil, kişisel bilgisayarlarda bulunmaktadır. Bu genişlemenin bir kısmı, kelime işlemcileri, elektronik tablolar ve sunum grafik programları gibi düşük maliyetli, oldukça popüler uygulamaların sayısının artmasıyla desteklendi. Ayrıca, çoğu ana bilgisayar yazılımı, kişisel bilgisayar yazılımı kadar kolay kullanılmaz, çok daha pahalıdır ve yıllarca geliştirilebilir. 1980'lerin sonlarında, birçok büyük organizasyonun 2-3 yıllık bir uygulama geliştirme gecikmesi vardı; yani, herhangi bir yeni ana bilgisayar uygulama programının yazılması yıllar alırdı.

İstemci-tabanlı mimarilerde, istemciler bir LAN'daki kişisel bilgisayarlardır ve sunucu genellikle aynı ağdaki başka bir kişisel bilgisayardır. İstemci bilgisayarlarındaki uygulama yazılımı, sunucu sadece verileri depolar. Hizmet mantığı yoktur.



Kaynak: https://quizlet.com/197467111/it-280-ch-02-application-layer-application-architectures-flash-cards/

saban.ozturk@hbv.edu.tr

1. 4. Client-Based Architectures

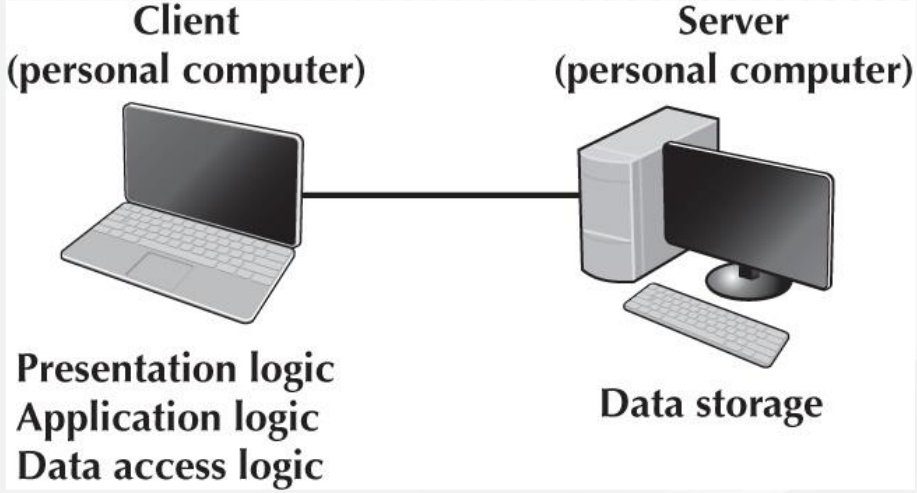
Bu basit mimari genellikle çok iyi çalışır. Eğer bir kelime işlemcisini kullandıysanız ve belge dosyanızı bir sunucuda sakladıysanız (veya kendi bilgisayarınızda çalışan ancak verileri bir sunucuda saklayan Visual Basic veya C programı yazdıysanız), bir istemci-tabanlı mimari kullanmış olursunuz.

İstemci-tabanlı ağlardaki temel sorun, sunucudaki tüm verilerin işlenmek üzere istemciye gitmesi gerekliliğidir. Örneğin, kullanıcının şirket hayat sigortası olan tüm çalışanları listelemek istediğini varsayalım. Veritabanındaki tüm verilerin (veya tüm dizinlerin) sunucudan, veritabanının depolandığı yerden ağ devresi üzerinden istemciye gitmesi gerekir, ardından istemci, her kaydı kullanıcının istediği veriyle eşleşip eşleşmediğini kontrol eder. Bu, ağ devrelerini aşırı yükleyebilir çünkü sunucudan istemciye aktarılan veriler, istemcinin aslında ihtiyaç duyduğundan çok daha fazladır.

1. 4. Client-Based Architectures

Bugün yazılan çoğu uygulama, istemci-sunucu mimarilerini kullanır. İstemci-sunucu mimarileri, işlemi istemci ve sunucu arasında dengeli bir şekilde dağıtmaya çalışırken her ikisinin de bazı mantığı yerine getirmesini sağlar. Bu ağlarda, istemci sunum mantığından sorumlu iken sunucu veri erişim mantığı ve veri depolamadan sorumludur. Uygulama mantığı ya istemci üzerinde bulunabilir, ya sunucuda bulunabilir, ya da her ikisi arasında bölünebilir.

Web sunucusundan sayfalar alırken, bir sunucudaki bir veritabanı ile konuşmak için SQL kullanan bir program



Kaynak: https://quizlet.com/197467111/it-280-ch-02-application-layer-application-architectures-flash-cards/

saban.ozturk@hbv.edu.tr

1. 4. Client-Based Architectures

İşletmeler için bir örnek;

Kullanıcı şirket hayat sigortası olan tüm çalışanların bir listesini istediğinde, istemci isteği kabul eder, bu isteği sunucu tarafından anlaşılabilir bir şekilde biçimlendirir ve sunucuya ileterek gönderir. İstek alındığında, sunucu veritabanında istenen tüm kayıtları arar ve ardından sadece eşleşen kayıtları istemciye ileterek kullanıcıya sunar. Veritabanı güncellemeleri için de aynısı geçerlidir; istemci isteği kabul eder ve bunu sunucuya gönderir. Sunucu güncellemeyi işler ve istemciye yanıt verir (güncellemeyi kabul eder veya neden kabul edilmediğini açıklar), istemci de bunu kullanıcıya gösterir.

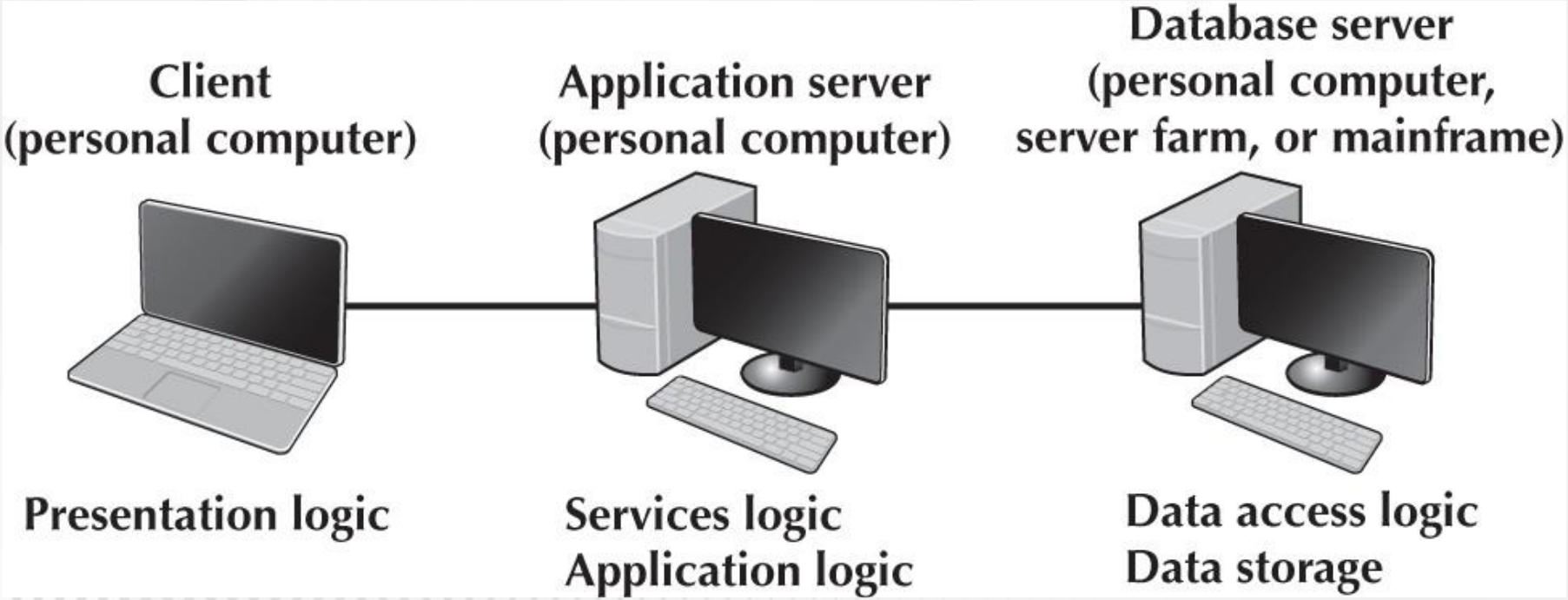
Öğrencilerden örnekler?????????

1. 4. Client-Based Architectures

İstemci-sunucu ağlarının güçlerinden biri, farklı satıcılardan gelen yazılım ve donanımın birlikte kullanılmasına olanak tanımasıdır. Ancak, bu aynı zamanda dezavantajlarından biridir, çünkü farklı satıcılardan gelen yazılımların birlikte çalışması zor olabilir. Bu sorunun bir çözümü, middleware'dir, istemci üzerindeki uygulama yazılımı ile sunucudaki uygulama yazılımı arasına yerleştirilen yazılımdır. Middleware iki şey yapar. İlk olarak, farklı satıcılardan gelen yazılımlar arasında çeviri yapabilen standart bir iletişim sağlar. Birçok middleware aracı, belirli bir istemci aracından gönderilen mesajların belirli bir sunucu aracı tarafından anlaşılabilir bir forma çevrilmesini sağlayan çeviri araçları olarak başladı.

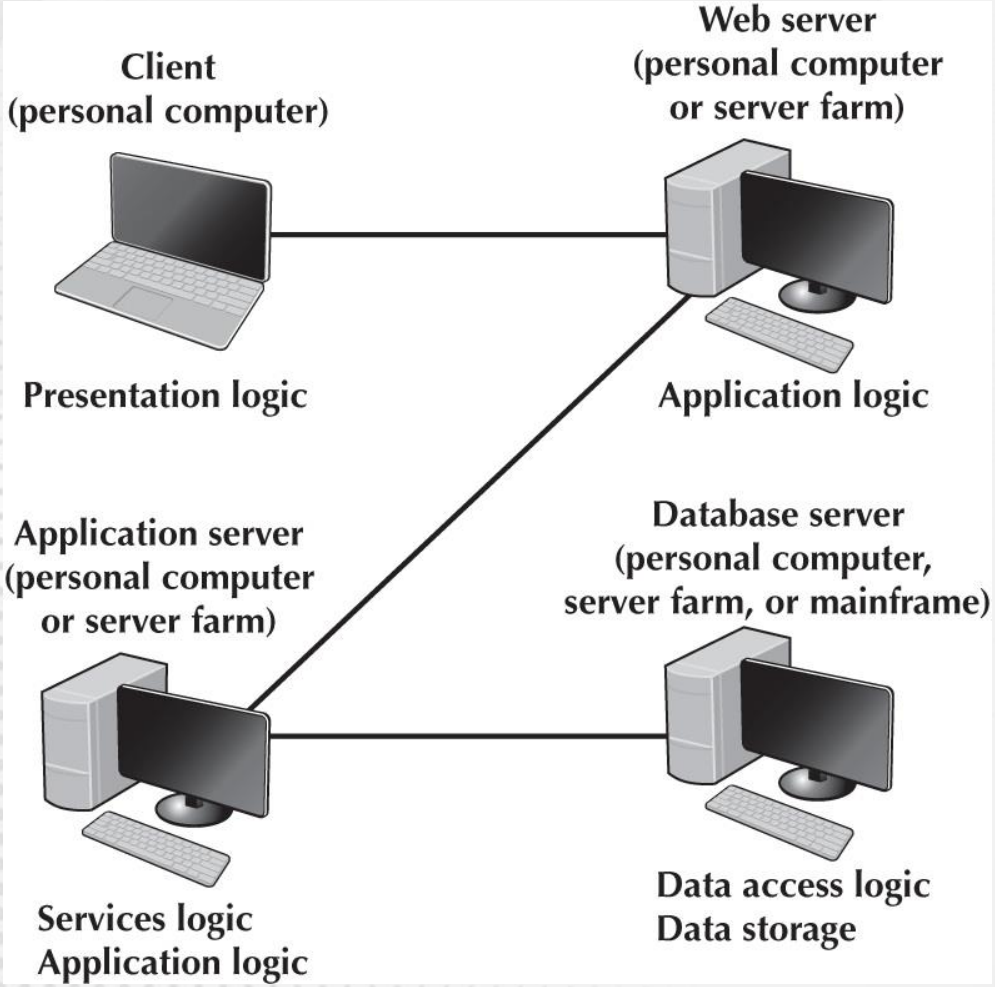
Middleware'ın ikinci işlevi, istemciden sunuculara (ve tersi) mesaj transferini yönetmektir, böylece istemcilerin uygulamanın verilerini içeren belirli sunucuyu bilmesi gerekmez. İstemci üzerindeki uygulama yazılımı tüm mesajları middleware'a gönderir, middleware bunları doğru sunucuya ileterek yönlendirir. Böylece, istemci üzerindeki uygulama yazılımı fiziksel ağda herhangi bir değişiklikten korunur. Ağ düzeni değişirse (örneğin, yeni bir sunucu eklenirse), sadece middleware güncellenmelidir.

1. 4. Client-Based Architectures



Kaynak: https://quizlet.com/197467111/it-280-ch-02-application-layer-application-architectures-flash-cards/

1. 4. Client-Based Architectures



Kaynak: https://quizlet.com/168722424/it-280-02-application-architectures-flash-cards/

1. 5. Cloud Computing Architectures

Geleneksel istemci-sunucu mimarisi uygulamak karmaşık ve maliyetli olabilir. Her uygulama, potansiyel olarak binlerce istemciden gelen istekleri karşılayabilmek için bir sunucuda barındırılmalıdır. Bir organizasyonun yüzlerce uygulaması olduğundan, başarılı bir istemci-sunucu mimarisini yürütmek çeşitli yazılım ve donanım ile bu mimariyi inşa edip sürdürebilecek deneyimli personeli gerektirir.

Buna karşılık, bulut bilişim mimarileri, altyapının bir kısmını veya tamamını yönetmeye odaklanmış diğer firmalardan yararlanır. Dolayısıyla, bulut bilişim, verileri depolamak, yönetmek ve işlemek için bir organizasyon tarafından barındırılan uzak sunucu ağını kullanır. Sunucu, büyük bir bilgisayar veya bir sunucu çiftliği olabilir. Bir sunucu çiftliği, bir araya bağlanmış bir grup bilgisayardır, böylece bir bilgisayar gibi davranırlar. İstekler sunucu çiftliğine (örneğin, Web istekleri) ulaşır ve bilgisayarlar arasında dağıtılır, böylece hiçbir bilgisayar aşırı yüklenmez. Her bilgisayar ayrıdır, böylece biri başarısız olursa, sunucu çiftliği onu basitçe atlar. Sunucu çiftlikleri, işin hızlı bir şekilde koordine edilmesi ve bireysel bilgisayarlar arasında paylaşılması gerektiğinden tek bir sunucudan daha karmaşıktır. Sunucu çiftlikleri, her zaman başka bir bilgisayar ekleyebilme esnekliğine sahiptir.

1. 6. Peer-to-Peer Architectures

Bir P2P mimarisinde, tüm bilgisayarlar hem istemci hem de sunucu olarak hareket eder. Dolayısıyla, tüm bilgisayarlar dört işlevi de gerçekleştirir: sunum mantığı, uygulama mantığı, veri erişim mantığı ve veri depolama. Bir P2P dosya paylaşım uygulaması ile, bir kullanıcı ağdaki başka bir bilgisayarda depolanan verilere erişmek için kendi bilgisayarında kurulu sunum, uygulama ve veri erişim mantığını kullanır. Bir P2P uygulama paylaşım ağı ile, ağdaki diğer kullanıcılar başkalarının bilgisayarlarını uygulama mantığına erişmek için kullanabilir. P2P ağlarının avantajı, verilerin ağın herhangi bir yerine kurulabilmesidir. Zorluk, verileri bulmaktır. İhtiyacınız olan verileri bulmanızı sağlayan bazı merkezi sunucular olmalıdır, bu nedenle P2P mimarileri genellikle bir istemci-sunucu mimarisi ile birleştirilir. Güvenlik, çoğu P2P ağı için önemli bir endişedir, bu nedenle P2P mimarileri genellikle kuruluşlarda kullanılmaz.

1. 6. Peer-to-Peer Architectures

Bir P2P mimarisinde, tüm bilgisayarlar hem istemci hem de sunucu olarak hareket eder. Dolayısıyla, tüm bilgisayarlar dört işlevi de gerçekleştirir: sunum mantığı, uygulama mantığı, veri erişim mantığı ve veri depolama. Bir P2P dosya paylaşım uygulaması ile, bir kullanıcı ağdaki başka bir bilgisayarda depolanan verilere erişmek için kendi bilgisayarında kurulu sunum, uygulama ve veri erişim mantığını kullanır. Bir P2P uygulama paylaşım ağı ile, ağdaki diğer kullanıcılar başkalarının bilgisayarlarını uygulama mantığına erişmek için kullanabilir. P2P ağlarının avantajı, verilerin ağın herhangi bir yerine kurulabilmesidir. Zorluk, verileri bulmaktır. İhtiyacınız olan verileri bulmanızı sağlayan bazı merkezi sunucular olmalıdır, bu nedenle P2P mimarileri genellikle bir istemci-sunucu mimarisi ile birleştirilir. Güvenlik, çoğu P2P ağı için önemli bir endişedir, bu nedenle P2P mimarileri genellikle kuruluşlarda kullanılmaz.

1. 7. Hangi Mimariyi Seçmeliyim?

Mimarilerin her birinin belirli maliyetleri ve faydaları vardır, peki "doğru" mimariyi nasıl seçersiniz? Birçok durumda, mimari basitçe verilmiştir; kuruluşun belirli bir mimarisi vardır ve kişi sadece onu kullanmalıdır. Diğer durumlarda, kuruluş yeni ekipmanlar edinmekte ve yeni yazılımlar yazmakta ve en azından kuruluşun bazı bölümlerinde yeni bir mimari geliştirme fırsatına sahiptir. Bugün neredeyse tüm yeni uygulamalar istemci–sunucu uygulamalarıdır. İstemci–sunucu mimarileri, en iyi ölçeklenebilirliği sağlar, sunucuların kapasitesini değişen ihtiyaçları karşılamak için artırma (veya azaltma) yeteneğini sunar. Örneğin, uygulama yazılımı veya veritabanı yazılımı ve depolama için daha fazla veya daha az kapasiteye ihtiyacımız olup olmadığına bağlı olarak kolayca uygulama sunucularını veya veritabanı sunucularını ekleyebilir veya kaldırabiliriz. İstemci–sunucu mimarileri aynı zamanda en güvenilir olanlardır. Aynı görevleri yerine getirmek için birden fazla sunucu kullanabiliriz, böylece bir sunucu başarısız olduğunda, kalan sunucular işlemlere devam eder ve kullanıcılar sorunları fark etmez.

(İstemci–sunucu mimarileri ayrıca bulut bilişimini de sağlar mı?)

2. WORLD WIDE WEB

Web ilk olarak 1989'da Sir Tim Berners-Lee tarafından Cenevre'deki Avrupa Parçacık Fiziği Laboratuvarı'nda (CERN) tasarlandı. Orijinal fikri, fizik araştırmaları hakkında bir bilgi veritabanı geliştirmekti, ancak bilgileri geleneksel bir veritabanına sığdırmak zor buldu. Bunun yerine, bilgi ağı olarak bir **hipertext** kullanmaya karar verdi. **Hipertext** ile herhangi bir belge, herhangi bir başka belgeye bir bağlantı içerebilir. CERN'in ilk Web tarayıcısı 1990'da oluşturuldu, ancak diğer organizasyonların kullanımı için İnternet'te 1991'e kadar kullanılabilir hale gelmedi. 1992'nin sonuna gelindiğinde, CERN ve birkaç Avrupa ve Amerikan üniversitesi tarafından UNIX bilgisayarları için birkaç tarayıcı oluşturulmuştu ve dünyada yaklaşık 30 Web sunucusu vardı.

Web, iki katmanlı bir istemci-sunucu mimarisinin iyi bir örneğidir.

2. WORLD WIDE WEB

Her istemci bilgisayar, bir Web tarayıcısı olarak adlandırılan bir uygulama katmanı yazılım paketine ihtiyaç duyar. Microsoft'un Internet Explorer gibi birçok farklı tarayıcı bulunmaktadır. Ağdaki her sunucu, bir Web sunucusu olarak işlev görecekse, bir Web sunucusu olarak adlandırılan bir uygulama katmanı yazılım paketine ihtiyaç duyar. Microsoft ve Apache gibi birçok farklı Web sunucusu bulunmaktadır.

Web'den bir sayfa almak için, kullanıcı istediği sayfanın İnternet birleşik kaynak konum belirleyicisi (URL) adresini yazmalıdır (örneğin, www.yahoo.com) veya URL adresini sağlayan bir bağlantıya tıklamalıdır. URL, istenen belirli sayfanın İnternet adresini ve dizinini ve adını belirtir. Dizin ve sayfa belirtilmemişse, Web sunucusu sitenin ana sayfası olarak tanımlanmış olan herhangi bir sayfayı sağlar. Web tarayıcısından gelen isteklerin Web sunucusu tarafından anlaşılabilmesi için aynı standart protokol veya dil kullanılmalıdır. Eğer bir standart olmasaydı ve her Web tarayıcısı farklı bir protokol kullanıyorsa, örneğin bir Microsoft Web tarayıcısının Apache Web sunucusuyla iletişim kurması imkansız olurdu.

2. WORLD WIDE WEB

Bir Web tarayıcısı ile Web sunucusu arasındaki iletişim için standart protokol, **Hypertext Transfer Protocol (HTTP)** 'dir. Bir Web sunucusundan bir sayfa almak için, Web tarayıcısı, URL ve istenen Web sayfası hakkında diğer bilgileri içeren bir HTTP isteği adı verilen özel bir paket gönderir. Sunucu isteği aldığında, işler ve isteğe bir HTTP yanıtı gönderir, bu ya istenen sayfa ya da bir hata mesajı olacaktır. Bu istek-yanıt diyaloğu, istemci ve sunucu arasında her dosya transferi için gerçekleşir. Örneğin, istemci iki grafik görüntüsü olan bir Web sayfası talep ederse. Grafikler, Web sayfasından farklı bir dosya biçimi kullanılarak ayrı dosyalarda saklanır (örneğin, JPEG [Joint Photographic Experts Group] biçiminde). Bu durumda, üç istek-yanıt çifti olurdu. İlk olarak, tarayıcı Web sayfası için bir istek gönderir ve sunucu yanıtı gönderir. Daha sonra, tarayıcı Web sayfasını görüntülemeye başlar ve iki grafik dosyasını fark eder. Tarayıcı daha sonra birinci grafik için bir istek ve ikinci grafik için bir istek gönderir ve sunucu iki ayrı HTTP yanıtı gönderir, her biri için bir tane.

3. HTTP

HTTP, İnternet üzerinde belgelerin ve kaynakların (örneğin web sayfalarının, resim dosyalarının, videoların vb.) aktarılması için kullanılan bir iletişim protokolüdür. HTTP, bir istemci-sunucu modeli kullanır, yani istemci (genellikle bir web tarayıcısı) bir sunucuya (web sunucusu) istek gönderir ve sunucu da isteği karşılar ve yanıtı geri gönderir.

HTTP, genellikle web tarayıcıları tarafından kullanılır. Bir kullanıcı bir web tarayıcısı aracılığıyla bir web sitesini ziyaret ettiğinde, tarayıcı otomatik olarak web sunucusuna bir HTTP isteği gönderir. Bu istek, kullanıcının istediği belge veya kaynağı (örneğin bir web sayfası) içerir. Sunucu, isteği alır, istenilen belgeyi veya kaynağı bulur ve bir HTTP yanıtı olarak tarayıcıya geri gönderir. Tarayıcı, aldığı yanıtı kullanıcıya gösterir.

3.1. HTTP Request

HTTP isteği ve HTTP yanıtı, ağ katmanları boyunca iletilmek üzere uygulama katmanı tarafından oluşturulan ve gönderilen paketlerin örnekleridir. Bu istekler ve yanıtlar, uygulama tarafından sağlanan bilgileri (örneğin, alınacak URL gibi) alır ve alıcının mesajı net bir şekilde anlayabilmesi için yapılandırılmış bir şekilde biçimlendirilmiş basit metin dosyalarıdır. Bir Web tarayıcısından bir Web sunucusuna yapılan bir HTTP isteği üç bölümden oluşur. İlk iki bölüm zorunludur; sonuncusu ise isteğe bağlıdır. Bu bölümler şunlardır:

- İstek satırı: Bir komutla başlar (örneğin, get), Web sayfasını sağlar ve tarayıcının anladığı HTTP sürüm numarası ile biter; sürüm numarası, Web sunucusunun, tarayıcının anlamadığı daha gelişmiş veya yeni bir HTTP standardı sürümünü kullanmaya çalışmasını önler.

- İstek başlığı: İsteğin yapıldığı tarihi ve kullanılan Web tarayıcısının (örneğin, Internet Explorer) gibi çeşitli isteğe bağlı bilgileri içerir.

- İstek gövdesi: Sunucuya gönderilen bilgileri içerir, örneğin, kullanıcının bir formda yazdığı bilgiler gibi.

3.1. HTTP Request

Şekilde, HTTP standardının 1.1 sürümü kullanılarak biçimlendirilmiş, kendi Web sunucumuzdaki bir sayfa için bir HTTP isteğinin bir örneğini göstermektedir. Bu istek sadece istek satırını ve istek başlığını içerir, çünkü bu istek için bir istek gövdesine ihtiyaç yoktur. Bu istek, isteğin yapıldığı tarih ve saatini (Greenwich Ortalama Zamanı [GMT], Londra üzerinden geçen saat dilimi) ve kullanılan tarayıcının adını (Mozilla, tarayıcının kod adıdır) içerir. "Referrer" alanı, kullanıcının bu Web sayfasının URL'sini başka bir sayfadaki bir bağlantıya tıklayarak aldığını belirtir; bu durumda, bu, Indiana Üniversitesi'ndeki öğretim üyelerinin bir listesi olan başka bir sayfada bulunmaktadır (yani, www.indiana.edu/~isdept/faculty.htm). Eğer "referrer" alanı boşsa, bu kullanıcının URL'yi kendisi yazdığı anlamına gelir.

Şekil için dersin sorumlusuna ulaşınız

3.1. HTTP Request

Sunucudan tarayıcıya yapılan bir HTTP yanıtının formatı, HTTP isteğinin formatına çok benzer.

1. Yanıt durumu, sunucunun kullandığı HTTP sürüm numarasını, bir durum kodunu içerir (örneğin, 200 "Tamam" anlamına gelir; 404 "bulunamadı" anlamına gelir) ve bir neden ifadesini (durum kodunun metin açıklaması).
2. Yanıt başlığı, Web sunucusunun kullanıldığı gibi çeşitli isteğe bağlı bilgiler içerir (örneğin, Apache), tarih ve yanıttaki sayfanın tam URL'si.
3. Yanıt gövdesi, Web sayfası kendisidir.

3.2. HTTP Örnek

GET /index.html HTTP/1.1
Host: www.example.com
Connection: close

Bu istekte, istemci (tarayıcı) "/index.html" adlı bir belgeyi (web sayfasını) istiyor. İstek, "www.example.com" adlı bir sunucuya yapılıyor ve "Connection: close" başlığı, bağlantının isteğin tamamlanmasının ardından kapatılacağını belirtiyor.

```
HTTP/1.1 200 OK
Date: Fri, 20 Feb 2024 12:00:00 GMT
Content-Type: text/html
Content-Length: 1234
<!DOCTYPE html>
<html>
<head>
    <title>Welcome to Example.com</title>
</head>
<body>
    <h1>Hello, world!</h1>
</body>
</html>
```

Bu yanıtta, sunucu isteği başarıyla karşıladı ve HTTP durum kodu "200 OK" ile yanıt veriyor. Yanıt, HTML içeriğine sahip bir belge içeriyor ve içeriğin uzunluğu "Content-Length" başlığı ile belirtiliyor.
HTTP'nin kullanımı, web sayfalarının ve diğer çevrimiçi içeriğin güvenilir bir şekilde alınması ve gösterilmesi için temel bir mekanizmadır. Web tarayıcıları, HTTP'yi kullanarak web sayfalarını indirir ve kullanıcıların görüntülemesine olanak tanır. Ayrıca, web sunucuları HTTP'yi kullanarak istemcilere içerik sağlar ve isteklere yanıt verir.

3.2. HTTP Örnek

```
// app.js

const http = require('http');

// Sunucu oluştur
const server = http.createServer((req, res) => {
  // İstek alındığında
  if (req.url === '/request') {
    // Hedef tarayıcıya istek gönder
    sendRequestToBrowser();

    // Başarılı yanıtı gönder
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('İstek gönderildi!');
  } else {
    // Bilinmeyen yol için 404 yanıtı gönder
    res.writeHead(404, {'Content-Type': 'text/plain'});
    res.end('404 Not Found');
  }
});

// Sunucuyu dinle
const PORT = 3000;
server.listen(PORT, () => {
  console.log(`Sunucu ${PORT} portunda çalışıyor.`);
});

// Hedef tarayıcıya istek gönderen fonksiyon
function sendRequestToBrowser() {
  // Bu kısımda isteği hedef tarayıcıya göndermek için uygun bir kütüphane veya araç kullanılmalıdır.
  // Örneğin, puppeteer gibi bir araçla tarayıcı açılabilir ve istek gönderilebilir.
}
```

4. E-Mail Sistemi

Farklı e-posta yazılımı paketleri arasındaki uyumluluğu sağlamak için birkaç standart geliştirilmiştir. Belirli bir standarta uyumlu herhangi bir yazılım paketi, kendi kurallarına göre biçimlendirilmiş mesajlar gönderebilir. Bu belirli standardı anlayan herhangi bir diğer paket daha sonra mesajı doğru hedefine iletir; ancak bir e-posta paketi farklı bir formatta bir posta mesajı alırsa, onu doğru şekilde işleyemeyebilir. Birçok e-posta paketi, bir standart kullanarak gönderirken, birkaç farklı standartta gönderilen mesajları anlayabilir. En yaygın kullanılan standart SMTP (Basit Posta Aktarım Protokolü) 'dür.

Basit Posta Aktarım Protokolü (SMTP), sadece İnternet üzerinde kullanılan e-posta standardı olduğu için en yaygın olarak kullanılan e-posta standardıdır. E-posta, Web'in nasıl çalıştığına benzer şekilde çalışır, ancak biraz daha karmaşıktır. SMTP e-postası genellikle iki katmanlı kalın istemci-sunucu uygulaması olarak uygulanır.

4. E-Mail Sistemi

Farklı e-posta yazılımı paketleri arasındaki uyumluluğu sağlamak için birkaç standart geliştirilmiştir. Belirli bir standarta uyumlu herhangi bir yazılım paketi, kendi kurallarına göre biçimlendirilmiş mesajlar gönderebilir. Bu belirli standardı anlayan herhangi bir diğer paket daha sonra mesajı doğru hedefine iletir; ancak bir e-posta paketi farklı bir formatta bir posta mesajı alırsa, onu doğru şekilde işleyemeyebilir. Birçok e-posta paketi, bir standart kullanarak gönderirken, birkaç farklı standartta gönderilen mesajları anlayabilir. En yaygın kullanılan standart SMTP (Basit Posta Aktarım Protokolü) 'dür.

Basit Posta Aktarım Protokolü (SMTP), sadece İnternet üzerinde kullanılan e-posta standardı olduğu için en yaygın olarak kullanılan e-posta standardıdır. E-posta, Web'in nasıl çalıştığına benzer şekilde çalışır, ancak biraz daha karmaşıktır. SMTP e-postası genellikle iki katmanlı kalın istemci-sunucu uygulaması olarak uygulanır.

4. E-Mail Sistemi

Her istemci bilgisayar genellikle bir e-posta istemcisi olarak adlandırılan bir uygulama katmanı yazılım paketi çalıştırır. Eudora ve Outlook gibi birçok yaygın e-posta istemcisi yazılım paketi bulunmaktadır. Kullanıcı, bu e-posta istemcilerinden birini kullanarak e-posta iletişimini oluşturur; bu, gönderenin adresi ve alıcının adresi gibi bilgileri içeren bir SMTP paketine biçimlendirilir.

Kullanıcı istemci daha sonra SMTP paketini, posta sunucusu yazılımı olarak da adlandırılan özel bir uygulama katmanı yazılım paketi çalıştıran bir posta sunucusuna gönderir. Bu e-posta sunucusu, hedef adresi bulmak için SMTP paketini okur ve ardından paketi ağ üzerinden -genellikle İnternet üzerinden- posta sunucusundan posta sunucusuna gönderir. Paket, hedef adresinde belirtilen posta sunucusuna ulaşana kadar posta sunucuları arasında dolaşır. Hedef sunucudaki posta transfer ajanı daha sonra iletiyi alıcının posta kutusuna kaydeder. İleti, alıcının yeni posta kontrol etmesi kadar alıcıya atanan posta kutusunda bekler.

4. E-Mail Sistemi

SMTP standardı, posta sunucuları arasındaki ileti iletimini (yani, posta sunucusu ile posta sunucusu arasındaki iletişimi) ve başlangıçtaki e-posta istemci ile posta sunucusu arasındaki iletişimi kapsar. Alıcının e-posta istemcisi ile posta sunucusu arasındaki iletişim için farklı bir standart kullanılır. E-posta istemci ve posta sunucusu arasındaki iletişim için kullanılan iki yaygın standart Posta Protokolü (POP) ve İnternet Mesaj Erişim Protokolü (IMAP) 'tir. POP ve IMAP arasında birçok önemli teknik fark vardır, ancak en belirgin fark, bir kullanıcının POP (sürüm 3) e-posta istemcisi ile bir posta iletisini okuyabilmesi için e-posta iletisinin istemci bilgisayarının sabit diske kopyalanması ve posta sunucusundan silinmesi gerekliliğidir. IMAP ile, e-posta iletisi okunduktan sonra posta sunucusunda saklanabilir. Bu nedenle, e-postalarını birçok farklı bilgisayardan (örneğin, ev, ofis, bilgisayar laboratuvarları) okuyan kullanıcılar için IMAP önemli avantajlar sunar.

4.1. E-Mail Sistemi (SMTP Paketi)

SMTP, mesaj aktarım ajanlarının nasıl çalıştığını ve diğer mesaj aktarım ajanlarına gönderilen iletileri nasıl biçimlendirdiklerini tanımlar. Bir SMTP paketinin iki parçası vardır:

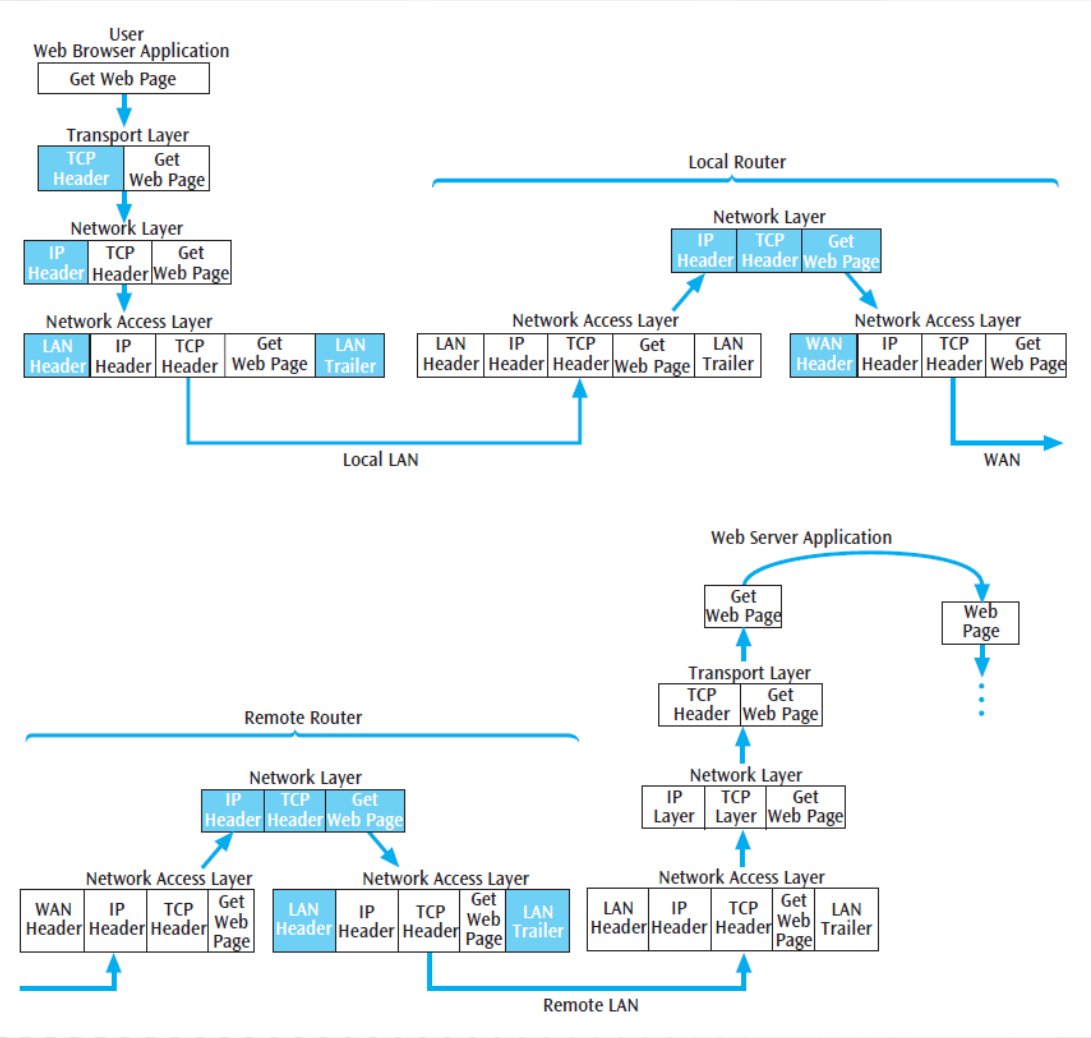
Başlık, kaynak ve hedef e-posta adreslerini (muhtemelen metin biçiminde [örneğin, "Pat Smith"]) ve adresi kendisi (örneğin, psmith@somewhere.com), tarihi, konuyu ve benzerlerini listeleyen bölümdür.

Gövde, mesajın kendisidir.

4.1. E-Mail Sistemi (SMTP Paketi)

SMTP kullanılarak biçimlendirilmiş basit bir e-posta iletişimini gösterir. Bir SMTP iletişiminin başlığında, göndericinin e-posta adresi, alıcının adresi, tarih vb. gibi belirli bilgileri sağlayan bir dizi alan bulunur. from ve to satırlarındaki tırnak içindeki bilgiler SMTP tarafından görmezden gelinir; e-posta adreslerinde yalnızca açılmış parantez içindeki bilgi kullanılır. Mesaj Kimliği alanı, iletiyi izlemek için benzersiz bir tanımlama kodu sağlamak için kullanılır. Mesaj gövdesi, iletişimin gerçek metnini içerir.

4.1. E-Mail Sistemi (SMTP Paketi)



Kaynak: https://slideplayer.biz.tr/slide/3177832/

Bölüm Özeti

- Uygulama Katmanı Çalışma Prensibine yönelik temel kavramlar
- Örnek Uygulamalar ile katmanların pekiştirilmesi

Değerlendirme Soruları

1) Uygulama Katmanı Çalışma Prensibini açıklayınız?

Kaynaklar

1) Stallings, W. (2007). Data and computer communications. Pearson Education India.

2) Forouzan, B. A. (2007). Data communications and networking. Huga Media.

3) Stallings, W., & Case, T. L. (2012). Business Data Communications-Infrastructure, Networking and Security.

4) Freer, J. (1996). Computer communications and networks. CRC Press.

5) Walrand, J., & Parekh, S. (2022). Communication networks: a concise introduction. Springer Nature.

6) .