Sivas Cumhuriyet Üniversitesi Bilgisayar Ağları Dersi

Bölüm 1
Bilgisayar Ağları Temelleri

Dr. Halil ARSLAN

Değerlendirme ve Kaynaklar

- Değerlendirme
  - (Arasınav + Ödev + Kısa Sınav)
  - Yıl içi çalışmalar : 40%
  - Final : 60%

- Ders kitabı
  - Larry L. Peterson and Bruce S. Davie, Computer Networks a systems approach, Fifth Edition, 2012.

- Diğer kaynaklar
  - James F. Kurose, Keith W. Ross, Computer Networking A Top-Down Approach, Sixth Edition, 2012.

Ders İçeriği

- Bilgisayar Ağları Temelleri
- Fiziksel Katman
  - Fiziksel ortamlar, veri kodlama
- Veri Bağı Katmanı
  - Çerçeveleme, MAC, Hata bulma/düzeltme
- Ağ Katmanı
  - IPv4, IPv6, SubNet, Yönlendirme, Protokoller
- Taşıma Katmanı
  - UDP, TCP, Tıkanıklık
- Uygulama Katmanı
  - HTTP
- Güvenlik

Bölümün hedefleri

- Temeller, Uygulamalar, Bileşenler
- Anahtarlama
- Adresleme
- Çoğullama
- Servisler
- Güvenirlik
- Yönetim
- Mimari
- Ağ uygulanması geliştirme
  - API, Örnek Uygulama
- Performans
  - Bandwidth, Latency, Delay, Throughput
- Uygulama Performansı

Bilgisayar Ağları - Temeller

Ağ kavramı;

- Başlangıçta telefon ağı ve TV yayıncılığı,
  - Ancak bu ağ türleri özel amaçlar içindir,
- Bilgisayar ağları ise geneldir,
  - Bilgisayar ağları birçok farklı veri türünü taşıyabilir,
  - Genişletilebilir ve farklı uygulamaları destekler

Bilgisayar Ağları - Temeller

Bilgisayar Ağları, 3 temel bileşeni içermektedir;

- Ağ kullanıcıları - users,
- Ağ operatörleri – ISS vb.,
- Ağ mimarileri ve protokoller – IEEE, IETF vb.,

Bilgisayar Ağları - Uygulamalar

- **WWW –** World Wide Web, temel internet uygulamasıdır. Internetin kendisi değildir.
  - Uniform Resource Locator (**URL**) ile erişimler tanımlanır.
  - Hypertext Transfer Protocol (**HTTP**) kullanılır.
  - IP ve TCP protokolleri adresleme ve taşıma içindir
- **Streaming** (video ve ses), www'den farklıdır. Bazen tarayıcı üzerinden olsa da RTP, UDP gibi farklı mimariler kullanır.
  - Skype, Hangout, WebRTC vb. gerçek zamanlı (real-time) iletim önemlidir.

Bilgisayar Ağları - Perspektif

Bilgisayar Ağları 3 perspektiften ele alınır.

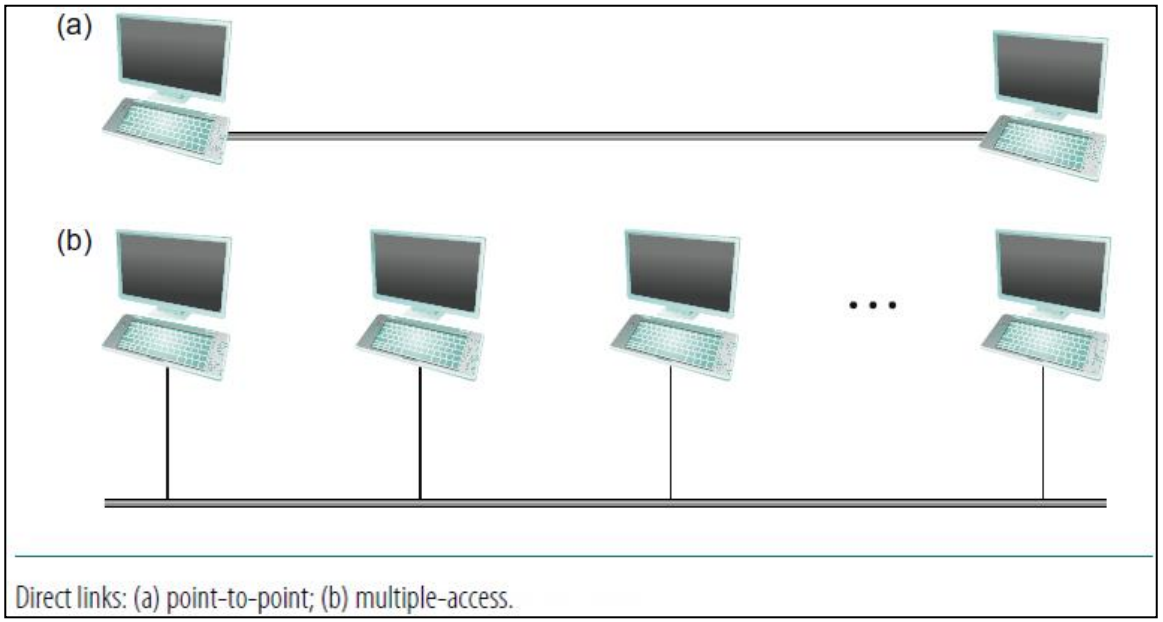
1. **Uygulama Geliştirici:** Ağ uygulamalarının geliştirilmesi için ihtiyaç duyulan temeller. Mesaj iletimi ve kullanıcı hareketlerini tanımlar
2. **Ağ Operatörü:** Ağ kurulumu, genişletilmesi ve yönetimi aşamalarını tanımlar
3. **Ağ Tasarımcısı:** Ağ kaynaklarının verimliliği, performansı ve ağın dağıtımı süreçlerini tanımlar

Bilgisayar Ağları - Bileşenler

- Bir bilgisayar ağı bir dizi bilgisayar arasında bağlantı sağlayan yapıdır.
- Bir ağ gizlilik nedeniyle sınırlandırılabilir, yada internette olduğu gibi sınırsız olabilir.
- Bir bilgisayar ağını oluşturan en temel bileşenler bağlantılar (**links**), düğümler (**nodes**) ve ağ bulutları (**clouds**) olarak tanımlanabilir.
- **Düğümler** ağa bağlanan terminaller yada ağa servis veren aktif bileşenlerken, **clouds** ise ağı soyut olarak tanımlayan kurallar bütünüdür.

Bilgisayar Ağları - Bileşenler

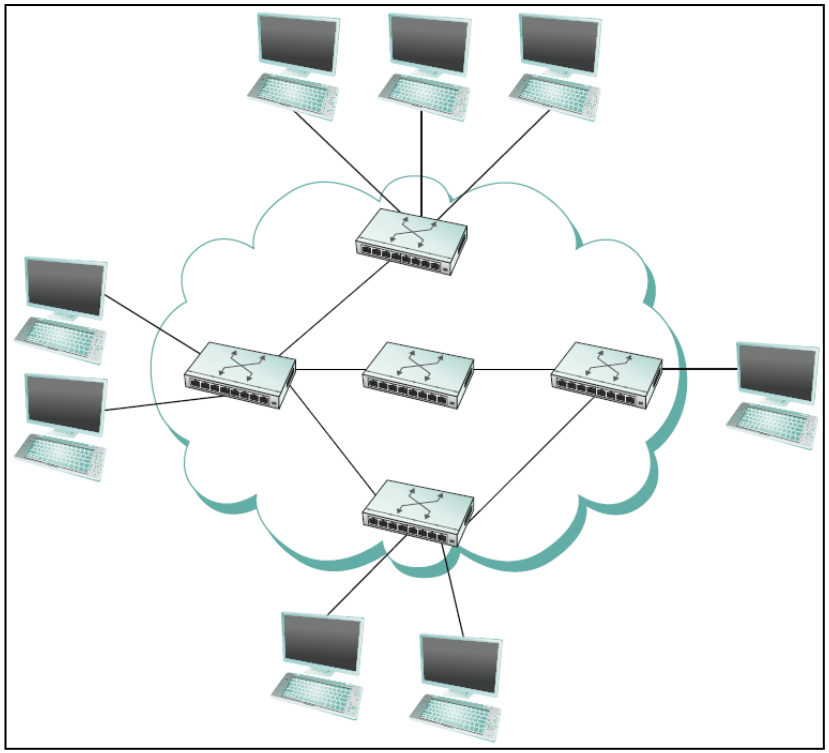
- **Links (bağlantılar),** Koaksiyel yada optik lif olabileceği gibi kablosuz ortam da olabilir.
- Bağlantılar bir düğüm çifti (point-to-point) ile sınırlı olabileceği gibi ikiden fazla düğümün tek bir fiziksel bağlantıyı (multiple-access) kullanabilir. Wi-fi ve hücresel bağlantı gibi.



Direct links: (a) point-to-point; (b) multiple-access.

Bilgisayar Ağları - Anahtarlama

- Düğümlerin doğrudan bağlantılı olma zorluğu iki düğüm arasındaki bağlantının dolaylı yoldan sağlanması ile aşılır.
- Bu işbirliğine anahtarlamalı ağ (**switched network**) denir.

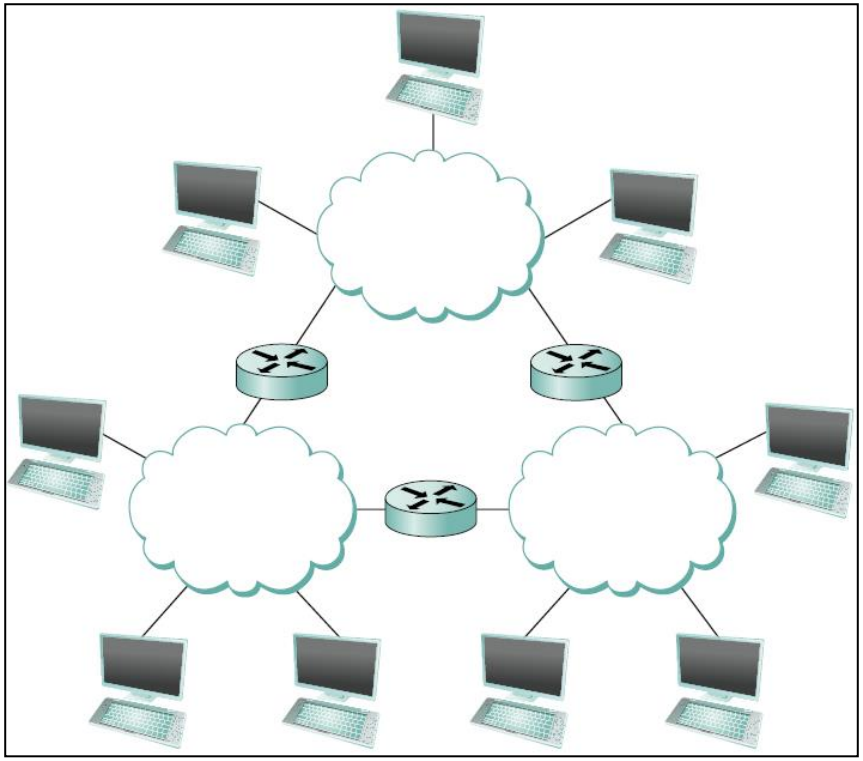


Bilgisayar Ağları - Anahtarlama

- En yaygın kullanılan anahtarlamalı ağ örnekleri;
- **Devre anahtarlama**: Telefon sistemlerinin temeli, ancak dsl ve fiber gibi sistemlerde de giderek artan kullanımı vardır.
- **Paket anahtarlama**: Bilgisayar Ağlarının temelidir. Düğümler arasında iletişim (dosya, e-posta vb.) paket yada mesaj adı verilen küçük parçalar halinde parça parça gerçekleşir.
- Paket anahtarlamalı ağlar depola ve ilet (**store-and-forward**) stratejisini kullanır.
- Devre anahtarlamada ise iki düğüm arasında devre kurulur ve iletim düğümler arasında gerçekleştirilir.

Bilgisayar Ağları - Anahtarlama

- Bağımsız ağlar kümesinin ağlar arası bağlantısına internetwork yada kısaca internet (Internet farklı) denir.
- İki veya daha fazla ağa bağlı düğüme **router** yada **gateway** denir. Temelde bir switch ile benzer rolü oynar.

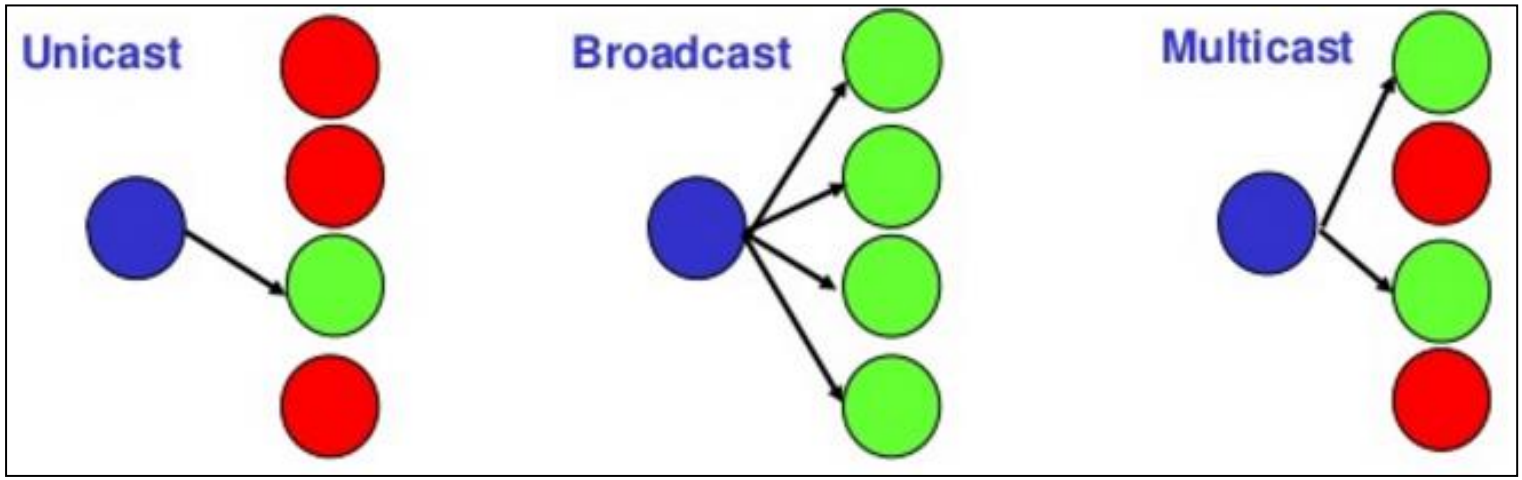


Bilgisayar Ağları - Adresleme

- Bir dizi bilgisayarın doğrudan veya dolaylı olarak birbirine bağlı olması, bir bilgisayardan başka bir bilgisayara bağlanabileceğimiz anlamına gelmez.
- Bir düğümün, ağdaki diğer düğümlerle iletişim kurabilmesi için bir adrese sahip olması gerekir.
- Adres, bir düğümü tanımlayan bir bayt dizesidir.
- Bir kaynak düğüm, belirli bir hedef düğüme bir mesaj teslim etmek isterse, hedef düğümün adresini belirtir.
- Gönderen ve alan düğümler doğrudan bağlı değilse, ağın anahtarları ve yönlendiricileri, mesajın hedefe doğru nasıl iletileceğine karar vermek için bu adresi kullanır.
- Adrese dayalı olarak hedef düğüme iletilerin nerden gönderileceğinin belirlenmesi sürecine **yönlendirme** denir.

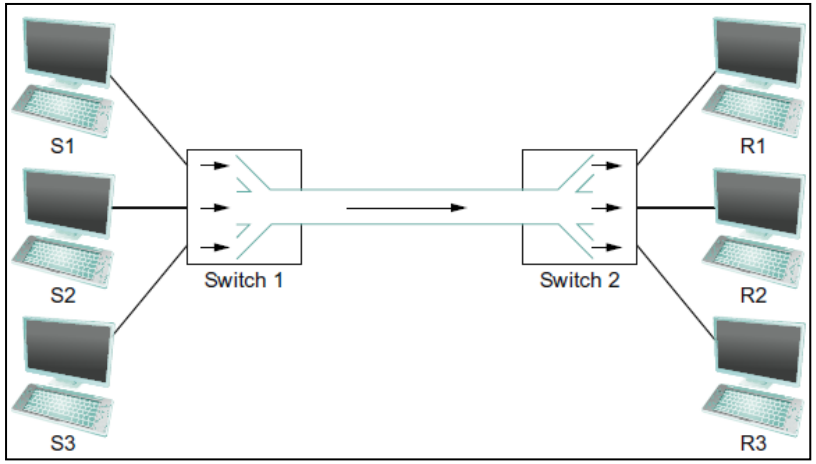
Bilgisayar Ağları - Adresleme

- Yönlendirme bir düğümden bir başka düğüme olabileceği gibi (**unicast**),
- Bir kaynak düğüm o ağdaki tüm düğümlere de paket iletimi yapabilir (**broadcast**)
- Yada bir kaynak düğüm diğer düğümlerin bir alt grubuna bir ileti göndermek isteyebilir (**multicast**)



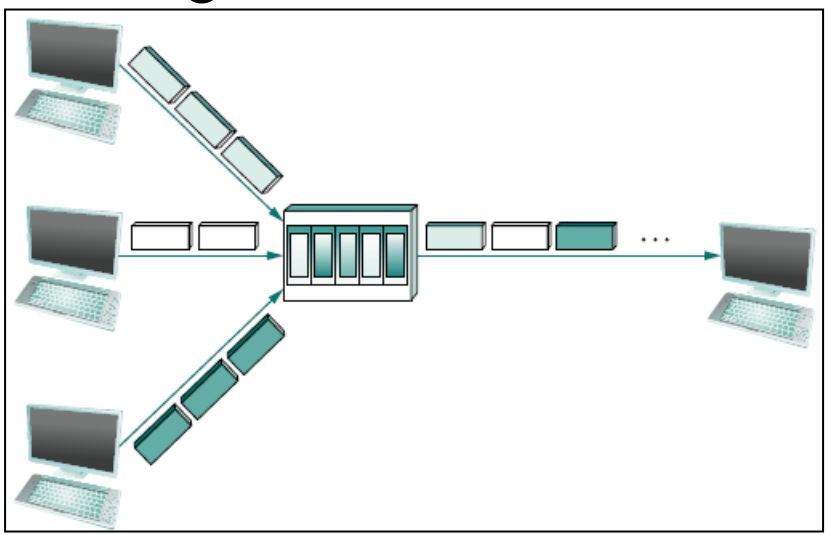
Bilgisayar Ağları - Çoğullama

- İletişim kurmak isteyen düğümlerin ortak bağlantıları paylaşmasına **multiplexing** denir.
- Çoğullama yaklaşımlarından birinci senkron zaman bölmeli çoğullamada (synchronous time-division multiplexing - **STDM**), kanal belli quantum zamanlarında paylaştırılır.
- Diğer bir yöntem frekans bölmeli çoğullamada (frequency-division multiplexing - **FDM**), farklı frekans boyları farklı kaynaklara ayrılır. Koaksiyel kablo üzerinden TV yayınları.



Bilgisayar Ağları - Çoğullama

- Hem **STDM** hem de **FDM** bilgisayar ağları için verimlilik açısından uygun değildir. Hattın boşta kalma süresi fazla.
- Bunun yerine bilgisayar ağları özel **multiplexing** yöntemleri kullanır.
- Ağın boşta kalmaması için isteyen düğüm iletime başlayabilir.
- Ancak bir düğümün hattı tamamen meşgul etmemesi için bazı sınırlamalar getirmek gerekir. MTU.



Bilgisayar Ağları – Coğrafik yapılar

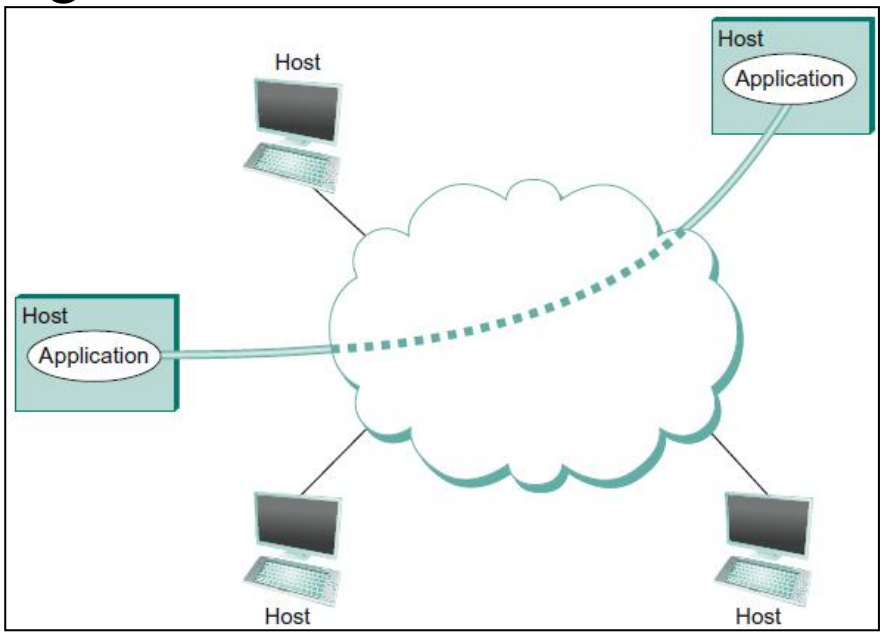
- Bilgisayar ağları kapladıkları fiziksel alanlar açısından PAN, LAN, SAN, MAN ve WAN gibi isimlerle anılırlar.
  - PAN – Personel Area Network
  - LAN – Local Area Network
  - SAN – Storage/system Area Network
  - MAN – Metropolitan Area Network
  - WAN – Wide Area Network
- Bu tür sınıflandırmanın önemi, bir ağın boyutunun kullanılan temel teknolojiler açısından etkili olmasıdır.
- Ancak genellikle ağlar LAN ve WAN konumundadır.

Bilgisayar Ağları – Servisler

- Bir bilgisayar ağı, ağa bağlı olan bilgisayarlarda çalışan uygulama programlarının anlamlı bir şekilde iletişim kurabilmesini sağlamalıdır.
- İki uygulama programının birbiriyle iletişim kurması için, bir bilgisayardan diğerine basitçe mesaj göndermenin ötesinde bir takım karmaşık işlemler gerçekleşir.
- Bu karmaşık işlemlerin her uygulama programı için yeniden inşaa edilmesi zordur.
- Oysa pekçok uygulama ortak servislere ihtiyaç duyar.
- Bir bilgisayar ağının temel hedefi, ağın karmaşıklığını, uygulama tasarımcısından, tasarımcıyı aşırı derecede kısıtmalamadan soyutlamaktır.

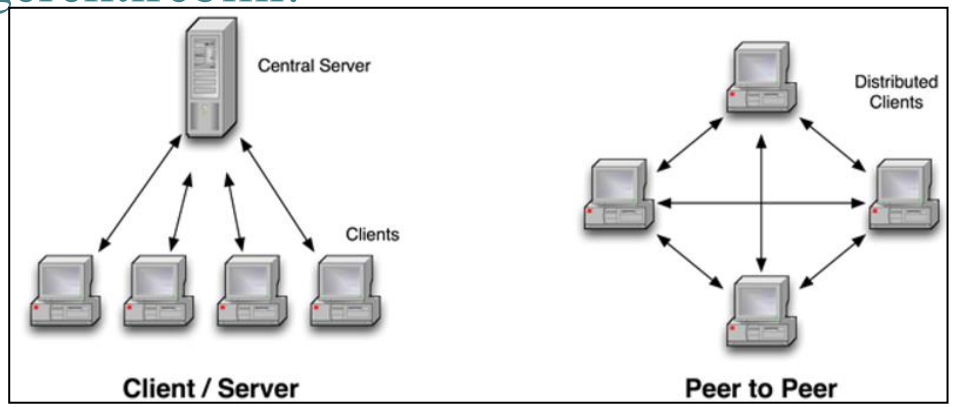
Bilgisayar Ağları – Servisler

- İki uygulama arasında oluşturulan soyut kanal uygulamanın gerektirdiği hizmet kümesini sunar.
- Örneğin, iletim garantisi gereksinimi, ileti kayıplarının önemi, paket sıralaması, gizlilik ve güvenlik gibi.
- Genel yaklaşımda ağ farklı hizmet türlerini sunar ve uygulama ihtiyaç duyduğu hizmet türünü tercih eder.



Bilgisayar Ağları – Servisler

- Örneğin ağ servisleri tasarlanan soyut kanallar üzerinden istemci/sunucu (client/server) mimarisinde bir hizmet türünü kullanabilir (FTP, NFS, HTTP vb.)
  - Bu hizmet türleri gevenlikli ve veri kayıpsız olmalıdır.
- Başka bir ağ servisi ise video konferans gibi eş düğümler arası (P2P – peer to peer yada client/server) bir hizmet türü kullanabilir.
  - Bu hizmet türü veri kaybını önemsemeyebilir. Ancak birden fazla bağlantı gerektirebilir.



Bilgisayar Ağları – Güvenirlik

- Bir ağın önemli gereksinimlerinden biri de belirli türdeki arıza ve veri kayıplarından uygulama programlarını yalıtmaktır.
- Bu açıdan ağ tasarımcıları 3 temel güvenirlik gereksinimini ele almalıdır.
  - Bir veri iletildiğinde, verideki bazı bitler bozulmuş olabilir. Yıldırım çarpması, güç dalgalanması, hatların zarar görmesi gibi dış etkenler olabilir. (bakır kabloda $10^6 – 10^7$, fiberde $10^{12} – 10^{14}$ bitte bir olur)
  - İletilen paket ağda kaybolabilir. Bunun nedeni genellikle bozulan bitlerden dolayı paketin atılması, anahtarlama cihazlarının belleklerinin dolması vb. olabilir.
  - Düğüm yada bağlantı seviyesinde bağlantı kesilebilir yada düğümlerden biri çökebilir. Bunun nedeni elektrik kesintileri, servis sağlayıcının hizmetinin durması, bir ağ aygıtının yanlış yapılandırılması gibi nedenlerle olabilir.

Bilgisayar Ağları – Yönetim

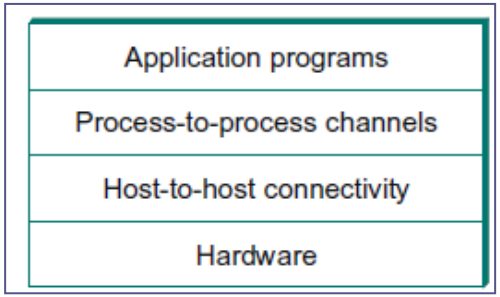
- Bir ağın yönetimi;
  - ağda daha fazla trafiği taşımak
  - ağın daha fazla kullanıcıya hizmet vermesini sağlamak
  - ağ büyüdükçe değişiklikler yapmak
  - işler ters gittiğinde veya performans düştüğünde ağın sorunlarını gidermek, gibi gereksinimleri kapsar.
- Yönetim kısmen ölçeklenebilirlik kavramıyla ilgilidir.
- Ağı oluşturan bileşenlerin yapılandırılması ve kurulumu yönetim aşamalarının temelini oluşturur.

Bilgisayar Ağları – Mimari

- Bir bilgisayar ağı, çok sayıda bilgisayar arasında genel, düşük maliyetli, adil ve sağlam bağlantılar sağlamalıdır.
- Ağların, değişikliklere, uygulama programlarına ve birlikte çalışabilirliğe uyum sağlayabilmesi için genel **mimari taslağının** olması gerekir.
- Yaygın olarak bilinen 2 ağ mimarisi **OSI** ve **Internet** (**DoD**)'dir.
- Bu yaklaşım genel olarak soyutlama (**Abstraction**) olarak bilinir.
- **Soyutlama**, tasarımcıların karmaşıklığı azaltmak için ayrıntıları iyi tanımlanmış arayüzlerin arkasına saklaması yaklaşımıdır.

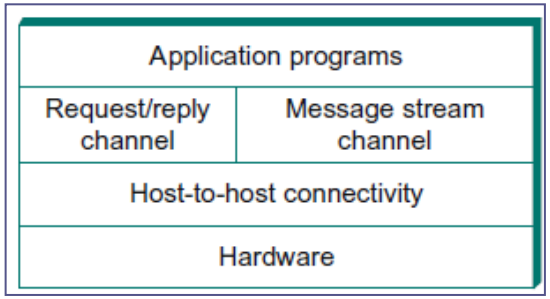
Bilgisayar Ağları – Mimari

- **Soyutlama**, ağ sistemleri açısından katmanlamayla (**layering**) sağlanır.
- **Katmanlama**, altta yatan donanımdan alınan hizmetle başlayıp alınan çıktıların daha üstteki katmanlara sunulmasıdır.
- Katmanlamanın en önemli iki avantajı;
  - Bir ağ açısından daha yönetilebilir bileşenler sunarak, sorunların sistemin belli katmanlarına indirgenmesini sağlar
  - Daha modüler bir tasarımla, yeni işlevlerin eklenmesini sadece belirli katmanlara indirgeyebiliriz



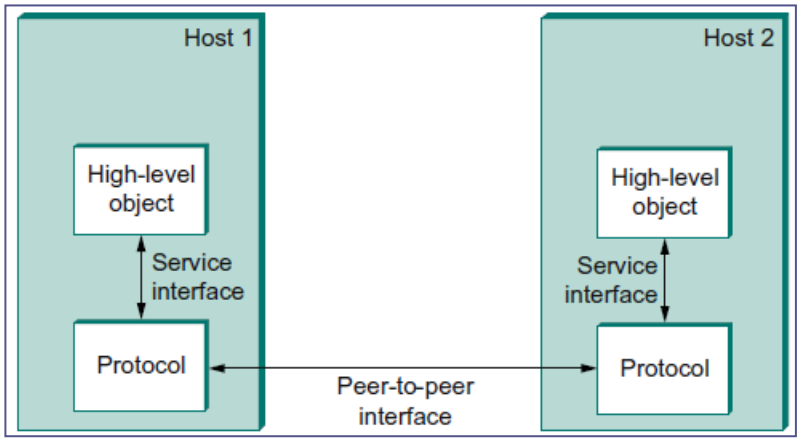
Bilgisayar Ağları – Mimari

- **Soyutlama'**da, sistemin herhangi bir katmanında birden fazla soyutlama yer alabilir.
- Bu katman içi soyutlamaların her biri, üst katmanlara farklı bir hizmet sunarken, aynı alt seviyedeki soyut katman üzerine kuruludur.
- Şekildeki gibi 3. katmanda soyutlamalardan biri istek/yanıt servisini sağlarken, bir diğeri mesaj akış hizmetini sunabilmektedir.
- Bir ağ sisteminin katmanlarını oluşturan soyut nesnelere **protokol** denir.



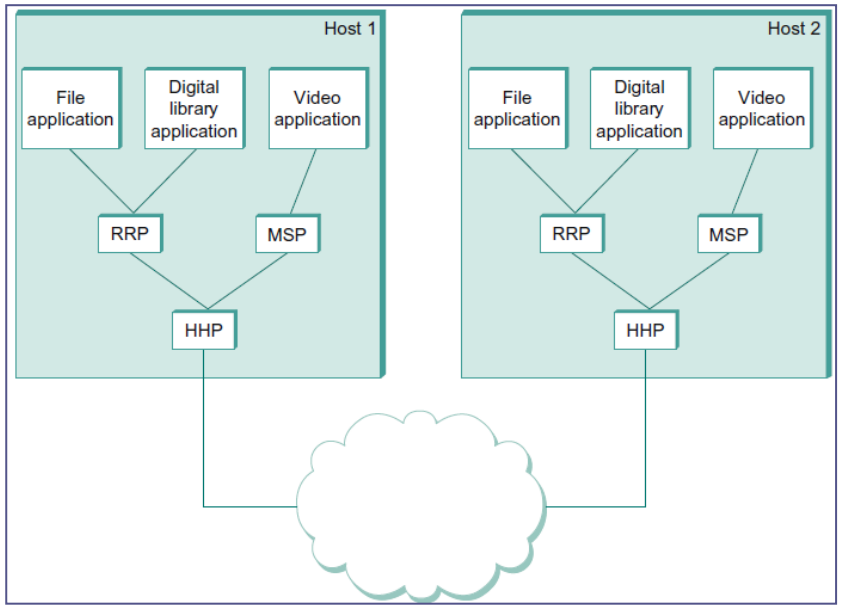
Bilgisayar Ağları – Mimari

- Her **protokol**, iki farklı arabirim sunar.
  - Aynı makine içinde iletişim servislerini kullanmak isteyen diğer nesneler için bir hizmet arayüzü sunar. Örn. http protokolünün sunduğu link vb. hizmetler.
  - Farklı makinelerdeki akran protokü için arabirim sunar. Örneğin HTTP protokolünün GET metodunun web sunucusunda nasıl işlenmesi gerektiğine karar vermesi gibi.



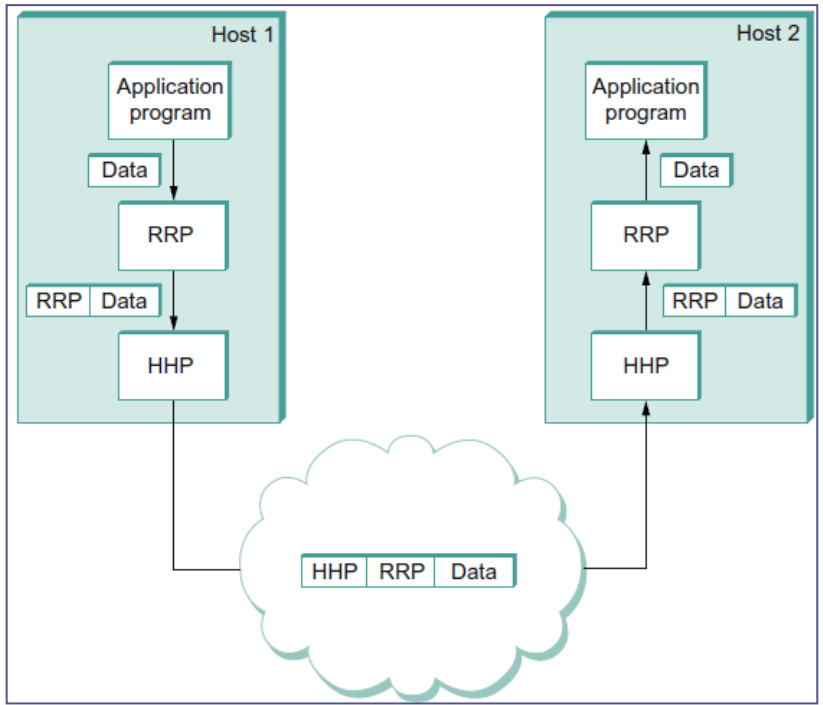
Bilgisayar Ağları – Mimari

- Bu örnekte, Host 1'deki Dosya (file app) Uygulaması, Host 2 üzerindeki hizmeti kullanmak için RRP (request/reply protocol) tarafından sunulan hizmeti kullanarak bir ileti göndermek ister.
- Bu durumda Dosya Uygulaması RRP'ye mesajı onun adına göndermesini ister. RRP, diğer makineyle iletşim için HHP'yi (Host-Host-Protocol) çağırır. Mesaj Host 2'ye geldiğinde HHP'yi çağırır, HHP mesajı RRP'ye geçer ve mesaj Dosya Uygulamasına iletilir.



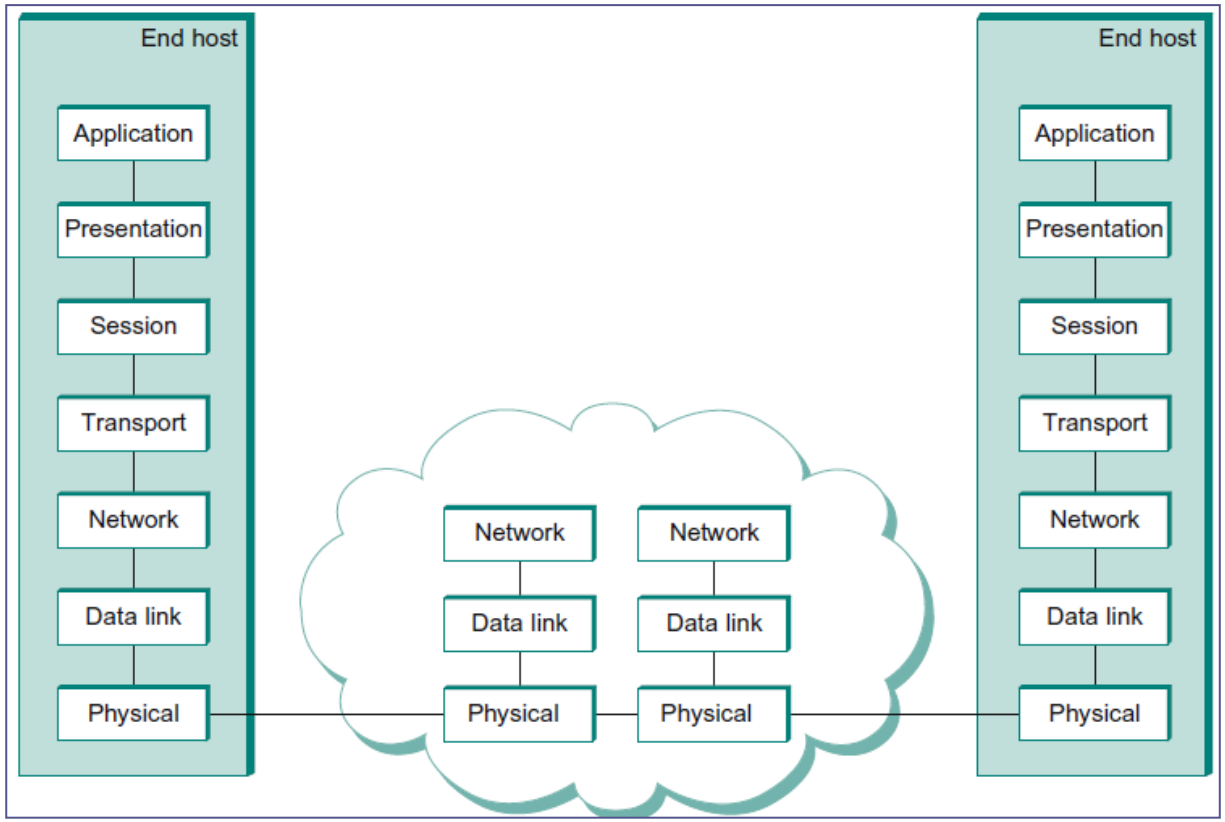
Bilgisayar Ağları – Mimari

- Kapsülleme – (**Encapsulation**), Üst katman protokollerinden alınan yükün (**payload**), karşılıklı olarak konuşabilmeleri için alt katman protolü tarafından kontrol bilgisi, adres vb. başlık (**header**) veya kuyruk (**trailer**)'ların eklenmesi işlemine denir.
- Kapsülleme işlemi protokol yığınının her seviyesinde gerçekleşir.



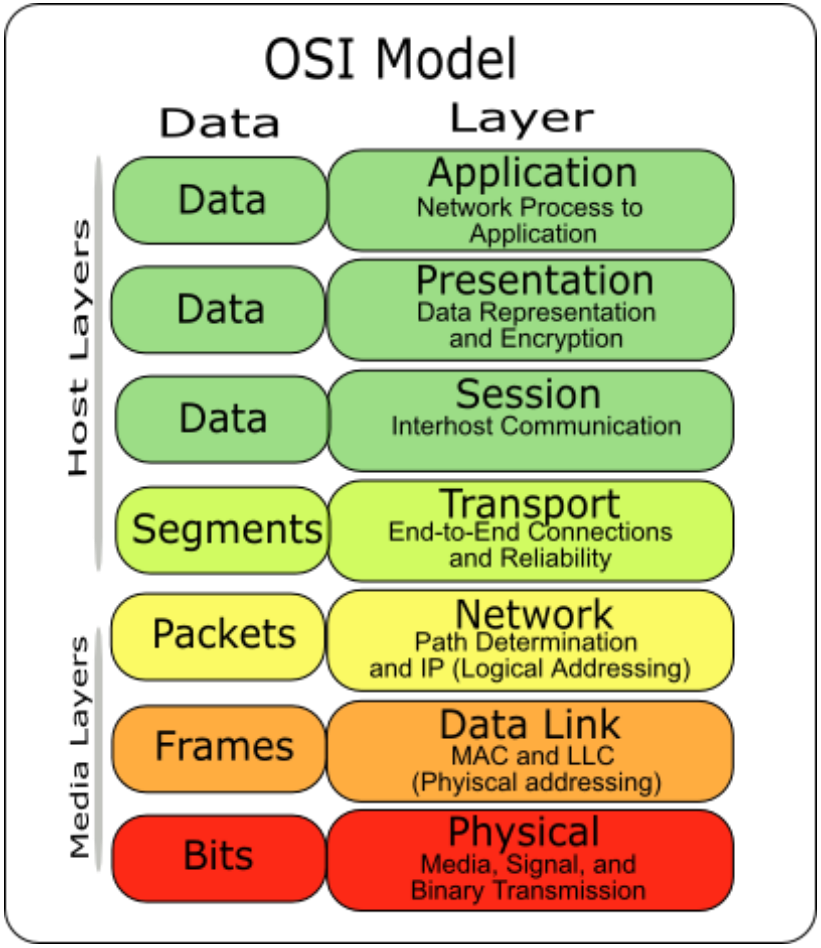
Bilgisayar Ağları – Mimari / OSI

- ISO tarafından ilk kez ağlar için tanımlanan katmansal mimari Open Systems Interconnection - OSI olarak bilinir.
- Bu model bir protokol yığınından çok bir mimari olarak tanımlanmıştır.



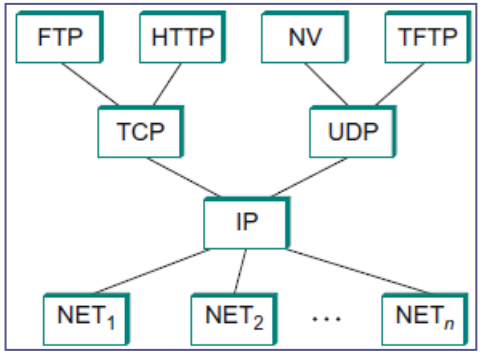
Bilgisayar Ağları – Mimari / OSI

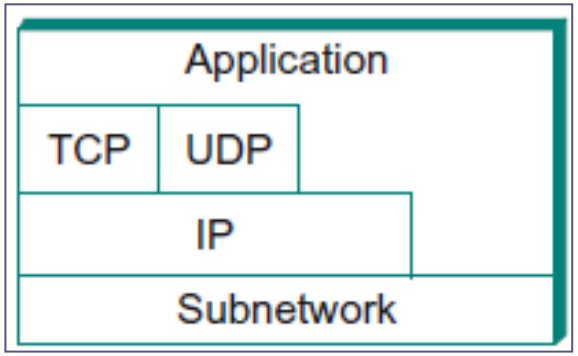
- Fiziksel katman, iletişim hattı üzerindeki bit iletimini,
- Veri bağı, çerçeve adı verilen bit yığınını ağ bağdaştırıcıları üzerinden işletim sistemindeki sürücülerle düzenler
- Ağ katmanı, paket anahtarlamalı ağlarda yönlendirme işlemini tanımlar.
- Taşıma katmanı, host bilgisayarlarda çalışan uygulama istek/cevap arayüzlerini tanımlar
- Diğer 3 katman uygulamaya özgüdür ve soyutlanmamış olabilir.



Bilgisayar Ağları – Mimari / DoD

- **TCP/IP** mimarisi olarak da ifade edilen İnternet mimarisi, **ARPANET** olarak bilinen ilk paket anahtarlamalı ağlar için ABD savunma bakanlığı (**D**epartment **o**f **D**efense) tarafından geliştirilmiştir.
- OSI'nin internete uygulanmasındaki zorluklardan dolayı 4 katmanlı model önerilmiştir.
- NET1, NET2 olarak sunulan en alt katmanda **802.x** gibi **ethernet** ve **kablosuz iletişimi** tanımlayan protokoller bulunur.
- İkinci katman sadece **IP** (Internet Protocol) protokolünden oluşur.
- Üçüncü katman ise **TCP** ve **UDP** gibi uçtan uca aktarım protokollerini tanımlar.





Bilgisayar Ağları – ISO/ DoD

OSI (Open Source Interconnection) 7 Layer Model

Layer	Application/Example		Central Device/ Protocols		DOD4 Model
Application (7) Serves as the window for users and application processes to access the network services.	**End User layer** Program that opens what was sent or creates what is to be sent Resource sharing • Remote file access • Remote printer access • Directory services • Network management		**User Applications** SMTP	GATEWAY Can be used on all layers	Process
Presentation (6) Formats the data to be presented to the Application layer. It can be viewed as the "Translator" for the network.	**Syntax layer** encrypt & decrypt (if needed) Character code translation • Data conversion • Data compression • Data encryption • **Character Set Translation**		JPEG/ASCII EBDIC/TIFF/GIF PICT		
Session (5) Allows session establishment between processes running on different stations.	**Synch & send to ports** (logical ports) Session establishment, maintenance and termination • Session support - perform security, name recognition, logging, etc.		**Logical Ports** RPC/SQL/NFS NetBIOS names		
Transport (4) Ensures that messages are delivered error-free, in sequence, and with no losses or duplications.	**TCP** Host to Host, Flow Control Message segmentation • Message acknowledgement • Message traffic control • Session multiplexing	PACKET FILTERING	TCP/SPX/UDP		Host to Host
Network (3) Controls the operations of the subnet, deciding which physical path the data takes.	**Packets** ("letter", contains IP address) Routing • Subnet traffic control • Frame fragmentation • Logical-physical address mapping • Subnet usage accounting		**Routers** IP/IPX/ICMP		Internet
Data Link (2) Provides error-free transfer of data frames from one node to another over the Physical layer.	**Frames** ("envelopes", contains MAC address) [NIC card — Switch — NIC card] (end to end) Establishes & terminates the logical link between nodes • Frame traffic control • Frame sequencing • Frame acknowledgment • Frame delimiting • Frame error checking • Media access control		**Switch Bridge WAP** PPP/SLIP	Land Based Layers	Network
Physical (1) Concerned with the transmission and reception of the unstructured raw bit stream over the physical medium.	**Physical structure** Cables, hubs, etc. Data Encoding • Physical medium attachment • Transmission technique - Baseband or Broadband • Physical medium transmission Bits & Volts		**Hub**		

Ağ Uygulaması Geliştirme

- Bilgisayar ağlarındaki bu hızlı gelişimin en önemli faktörlerinden biri de ağ uygulama yazılımı geliştirmedeki standartlardır.
- Bir ağ yazılımı, işletim sisteminin ağ arayüzleri için standart olarak sağladığı Ağ API'ları ile kolayca geliştirilebilmektedir.
- Her işletim sistemi kendi ağ API'ını tanımlayabilmekte serbesttir. Ancak Unix Berkeley tarafından geliştirilen **socket API**'yı neredeyse tüm sistemler tarafından desteklenir durumdadır.
- Ayrıca Java gibi platform bağımsız soket uygulamaları için standart oluşturmuştur.
- Soket arayüzünün ana soyutlaması **soket**'lerdir. Soketler, yerel uygulamaların ağa eklendiği noktalardır.
- Soket API'ları soketi oluşturmak, soketi ağa bağlamak, ileti alma/gönderme yapmak ve soketi kapatmak için işlemleri tanımlar.

Ağ Uygulaması Geliştirme

- Socket API'yı kullanarak TCP tabanlı ağ uygulaması geliştirmek için.
  - int socket(int domain, int type, int protocol) ile soket tanımı yapılır
    - domain argümanı: PF_INET, PF_UNIX, PF_PACKET
    - type argümanı: SOCK_STREAM (TCP), SOCK_DGRAM (UDP)
    - protokol argümanı: üst seviye protokoller tanımlanır.
- Sonraki adımda sunucu yada istemci olma durumuna göre bağlantı tanımlanır. Sunucu için passive open durumunda
  - int bind(int socket, struct sockaddr *address, int addr_len)
    - İstenilen sunucu adresine ait soket kurulur
  - int listen(int socket, int backlog)
    - Soket dinlenir
  - int accept(int socket, struct sockaddr *address, int *addr_len)
    - Yeni bir bağlantı kuruluncaya kadar passive open durumunda beklenir.

Ağ Uygulaması Geliştirme

- İstemci bir sokete bağlanmak istediğinde;
  - int connect(int socket, struct sockaddr *address, int addr_len)
    - Sunucunun soketine bağlanılmaya çalışılır. Başarılı bir şekilde bağlantı kurulunca soket işlenebilir.
- Bir bağlantı kurulduktan sonra oluşturulan soket üzerinden veri göndermek ve almak için;
  - int send(int socket, char *message, int msg len, int flags)
    - İlgili mesaj belirtilen sokete iletilir.
  - int recv(int socket, char *buffer, int buf len, int flags)
    - Belirtilen soketten arabelleğe mesaj okunur.

Örnek Uygulama - Server

```
#include<stdio.h>
#include<string.h>    //strlen
#include<sys/socket.h>
#include<arpa/inet.h> //inet_addr
#include<unistd.h>    //write
int main(int argc , char *argv[]){
    int socket_desc , client_sock , c , read_size;
    struct sockaddr_in server , client;
    char client_message[2000];
    socket_desc = socket(AF_INET , SOCK_STREAM , 0); //Create socket
    if (socket_desc == -1) printf("Could not create socket");
    puts("Socket created");
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = INADDR_ANY;
    server.sin_port = htons( 8888 );
    if( bind(socket_desc,(struct sockaddr *)&server , sizeof(server)) < 0){
        perror("bind hatasi"); return 1;}
    listen(socket_desc , 3);
    puts("Baglanti bekleniyor...");
    c = sizeof(struct sockaddr_in);
    client_sock = accept(socket_desc, (struct sockaddr *)&client, (socklen_t*)&c);
    if (client_sock < 0){
        perror("accept hatasi"); return 1;
    }
    puts("Baglanti basarili…");
    while( (read_size = recv(client_sock , client_message , 2000 , 0)) > 0 ) {
          write(client_sock , client_message , strlen(client_message));
    }
    if(read_size == 0) {
        puts("Client ayrildi");
        fflush(stdout);
    }
    else if(read_size == -1) {
        perror("okuma hatasi");
    }
    return 0;
}
```

```
$ gcc -c server.c
$ gcc server.o -o server
$ ./server
```

Örnek Uygulama - Client

```
#include<stdio.h>      //printf
#include<string.h>     //strlen
#include<sys/socket.h> //socket
#include<arpa/inet.h>  //inet_addr
int main(int argc , char *argv[]){
    int sock;
    struct sockaddr_in server;
    char message[1000] , server_reply[2000];

    sock = socket(AF_INET , SOCK_STREAM , 0);
    if (sock == -1) {
        printf("socket olusturulamiyor");
    }
    puts("Soket olusturuldu");
    server.sin_addr.s_addr = inet_addr("127.0.0.1");
    server.sin_family = AF_INET;
    server.sin_port = htons( 8888 );

    if (connect(sock , (struct sockaddr *)&server , sizeof(server)) < 0) {
        perror("baglanti hatasi"); return 1;
    }
    puts("baglanti kuruldu..\n");
    while(1) {
        printf("Mesaj gir: ");  scanf("%s" , message);
        if( send(sock , message , strlen(message) , 0) < 0) {
            puts("Mesaj iletilemedi!"); return 1;
        }

        if( recv(sock , server_reply , 2000 , 0) < 0) {
            puts("okuma hatasi"); break;
        }
        puts("Server cevabi: "); puts(server_reply);
    }
    close(sock);
    return 0;
}
```

```
$ gcc -c client.c
$ gcc client.o -o client
$ ./client
```

Performans

- Herhangi bir bilgisayar sistemi gibi bilgisayar ağlarının da iyi performans göstermesi gerekir.
- Bu nedenle ağın performansını etkileyen faktörler önemlidir.
- Ağ performansı 2 temel yolla ölçülür.
- **Bandgenişliği** (bandwidth - throughput olarak da isimlendirilebilir)
- Belirli bir süre içerisinde ağda iletilebilen bit miktarını verir.
- Örneğin 10 Mbps'lik bir ağda, saniyede 10 milyon bit iletilebilir. Bu ağda her bitin iletim zamanı 0,1 mikrosaniyedir.
- Bandwidth (band genişliği) ile throughput (verimlilik) terimleri temelde farklıdır.
- Birincisi elektriksel olarak frekansı ifade ederken, verimlilikte ise gerçekte iletilebilecek bit sayısı ifade edilir.

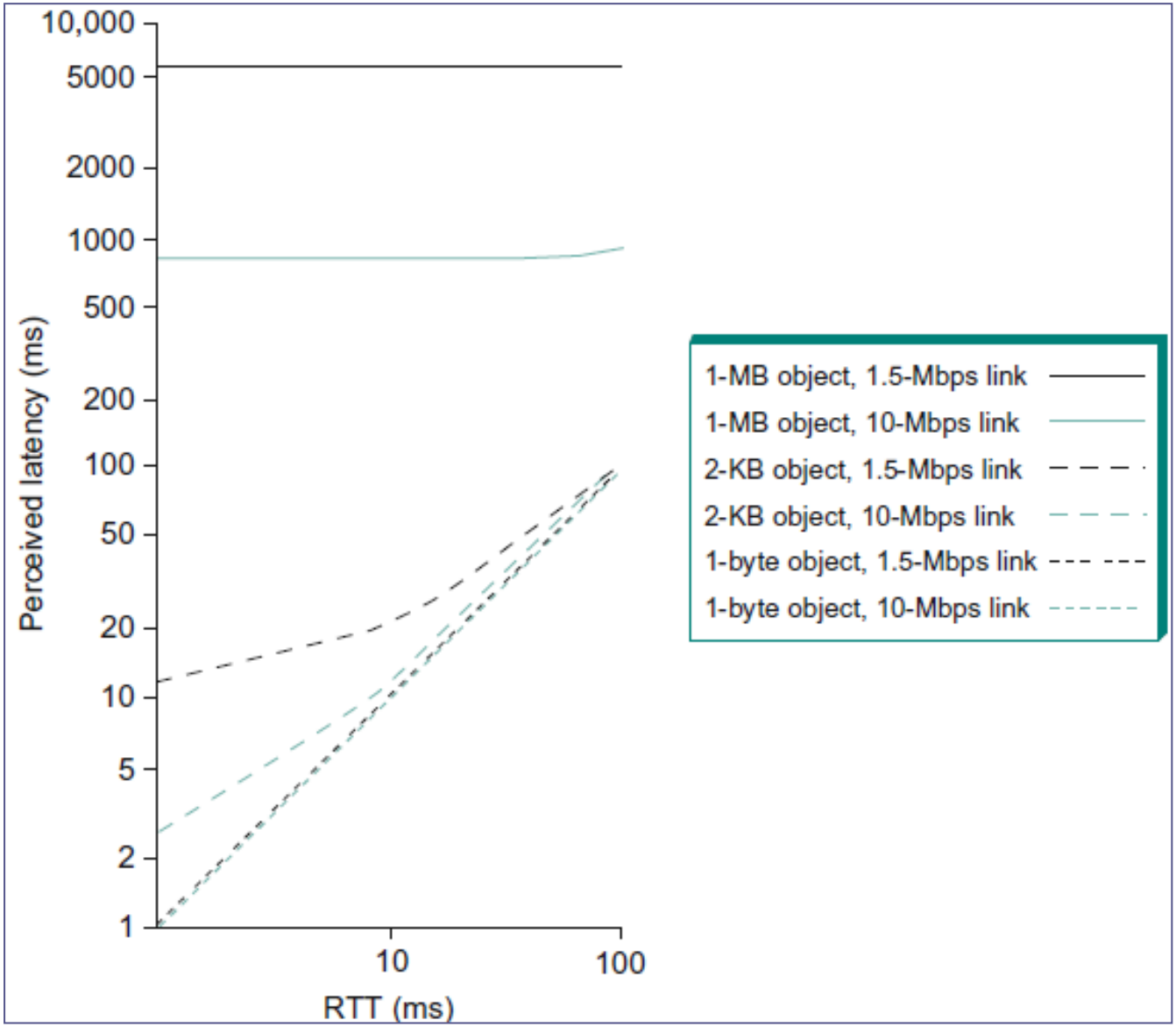
Performans

- **Gecikme** (latency – delay olarak da isimlendirilir), bir iletinin, ağın bir ucundan diğerine ne kadar sürede gittiğini tanımlar
- Tek yönlü gecikme yerine bir ağın bir ucundan diğerine gidiş-dönüş süresini bilmek daha önemli olabilir. Buna round-trip time (**RTT**) denir.
- Gecikme genellikle 3 bileşene sahiptir.
  - Birincisi ışığın yayılım gecikmesidir. Mesafe ve ortama göre yayılım hızı biliniyorsa bu hesaplanabilir. Havasız ortamda 3.0×10^8 m/s, bakır kabloda 2.3×10^8 m/s ve fiberde 2.0×10^8 m/s'dir.
  - İkincisi bir veri birimini iletmek için geçen süre miktarıdır. Band genişliği ile ilgilidir.
  - Üçüncüsü, paket anahtarlamalı ağlardan kaynaklanan yönlendirme yaklaşımı olan store-and-forward yaklaşımdan kaynaklı kuyruk (Queue) gecikmesidir.

Performans

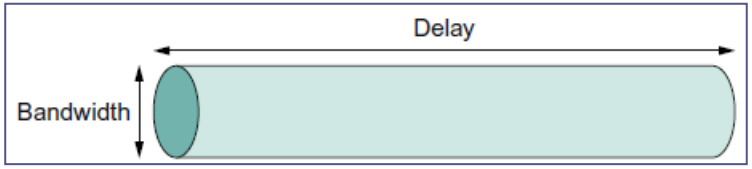
- Dolayısıyla toplam gecikmeyi aşağıdaki gibi hesaplayabiliriz;
  - **Latency** = Propagation+Transmit+Queue
    - Propagation = Distance/SpeedOfLight
    - Transmit = Size/Bandwidth
- Bandgenişliği ve gecikme belirli bir bağlantının performansını tanımlamak için birlikte ele alınır.
- Bazı uygulamalar için gecikme band genişliğinden önemliyken bazı uygulamalar için band genişliği önem kazanır.
- Örneğin iletilecek veri 1byte ise band genişliğinin önemi olmaz.
- Ancak iletilecek veri 25 MB olduğunda band genişliği önem kazanacaktır. 10 Mbps'lık bir bağlantı için ($25\times10^6\times8$ bits $\div 10\times10^6$ Mbps = 20 saniye)
- Bu durumda 100ms yada 1ms'lik gecikmenin çok da bir önemi olmaz.

Performans


10,000
5000
2000
1000
500
200
100
50
20
10
5
2
1
Perceived latency (ms)
10
100
RTT (ms)
1-MB object, 1.5-Mbps link
1-MB object, 10-Mbps link
2-KB object, 1.5-Mbps link
2-KB object, 10-Mbps link
1-byte object, 1.5-Mbps link
1-byte object, 10-Mbps link

Performans

- Gecikme ve bandgenişliği (**bandwidth x delay**) ilişkisi, gecikmenin hattın uzunluğunu, band genişliğinin de hattın genişliğini tanımlarsa bu iki değerin çarpımı hattın hacmini yani hattan geçebilecek maksimum bit sayısını tanımlar.
  - Örneğin tek yönlü gecikme süresi 50ms, band genişliği 45Mbps olan bir hattın kapasitesi: $50\times10^{-3}s \times 45\times10^{6}$ bits/s = 2.25×10^{6} bits = 280 KB olur.
  - Bu kavram alıcıya ulaşmadan göndericinin kaç bit gönderebileceğini ifade eder.
  - Bu durum çift taraflı olarak, gönderene alıcının bitlerin geldiğini ifade etmesi gerektiğinde **bandwidth x RTT** şeklinde tanımlanır.



Performans

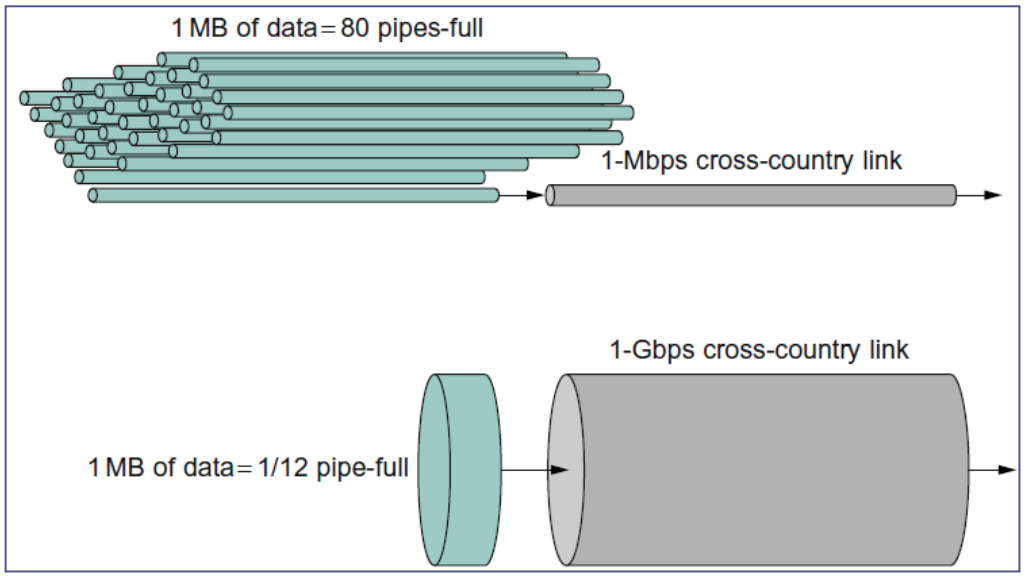
Link type	Bandwidth (typical)	One-way distance (typical)	Round-trip delay	RTT × Bandwidth
Dial-up	56 kbps	10 km	87 μs	5 bits
Wireless LAN	54 Mbps	50 m	0.33 μs	18 bits
Satellite	45 Mbps	35,000 km	230 ms	10 Mb
Cross-country fiber	10 Gbps	4,000 km	40 ms	400 Mb

Performans

- Band genişliğindeki hızlı artış aynı zamanda gecikmenin de iyileşmesi anlamına **gelmez**.
- 1Gbps'lık bir bağlantı ile 1Mbps'lık bir bağlantının kıtalar arası RTT değeri aynıdır. (100 ms civarında)
- Aynı gecikmeye sahip 1Gbps ve 1Mbps'lık iki ağ üzerinden 1 MB'lık bir dosya iletildiğinde

1 Mbps için;
100ms gecikme 1Mbps bw
0,1s x 1 x 10^6bit/s = 1x10^6 bit
1000000bit=125000byte≈12,5KB
1 MB için ≈ 80 RTT

1 Gbps için;
100ms gecikme 1Gbps bw
0,1s x 1 x 10^9bit/s = 12,5MB
1 MB için 1 RTT



Performans

- Bir ağ üzerinden elde edilebilecek etkin uçtan uca işlem hacmine **verim** (**Throughput**) denir ve aşağıdaki ilişki ile tanımlanır.
  - Throughput = TransferSize/TransferTime
- Burada TransferTime tek yönlü gecikmenin yanında aktarım sürecinde harcanan diğer zamanları da içerir.
  - TransferTime = RTT+((1/Bandwidth) × TransferSize)
- Örneğin kullanıcın 100ms RTT ile **1 Gbps'lık** bir ağ üzerinden 1MB'lık bir dosyayı almak istediğinde;
  - TransferTime = 100 + (1/1Gbps x 1 MB = 8 ms) = 108 ms'dir.
  - Throughput = 1 MB/108 ms = **74.1 Mbps** (1 Gbps olmadığı görülüyor)
- Band genişliği (Bandwidth) ile Verim (Throughput) arasındaki ilişki net olarak görülmektedir.

Uygulama Performansı

- Ağ performansı o ağda koşan uygulamanın ne kadarlık bir ağ performansına ihtiyaç duyduğuyla ilgilidir.
- Örneğin standart bir TV ekranı 352 x 240 piksel çözünürlüğe 24 bit derinliğine ve her saniyede 30 frame oynatsın
  - Her frame'in boyutu (352×240×24)/8 = **247.5 KB**
  - Throughput Rate 247.5 × 30 = **58 Mbps** olur
- Gerçekte böyle bir verime ihtiyaç duyulmaz. Sıkıştırma ve kodek?
- İletilen iki çerçeve arasındaki farklar iletilerek boyut azaltılır.
- Haliyle değişken performans ihtiyaçlarına gereksinim olacaktır.
- Bir uygulamanın bandgenişliği gereksinimi, olabildiğince fazla ve gecikme gereksinimi, olabildiğince az şeklinde basitçe özetlenebilir.

Uygulama Performansı

- Ağın tek yönlü gecikmesinden çok ağın paketler arasında ne kadar gecikme değişimi gösterdiği daha önemli olabilir. Bu değere **jitter** denir.
- Saniyede 30 frame iletim yapan bir video uygulaması için;
  - Kaynağın her 33ms'de bir bir frame gönderdiğini varsayalım,
  - Paketler hedefe 33ms'de bir ulaşırsa ağdaki her paketin gecikmesi aynıdır,
  - Ancak paketlerin varış noktasına geliş aralığı değişken olursa (**inter-packet gap** olarak da adlandırılır) ağda **jitter** oluştuğu ifade edilir.
- Bu tür durumlar devre anahtarlamalı ağlarda oluşmaz. Paket anahtarlamalı ağlarda yönlendirme durumlarından kaynaklanır.
- Jitter, paketlerin alıcıda tamponlanmasına ve oynatımın geciktirilmesine/kalitenin düşürülmesine neden olacaktır.