

**AĞ (NETWORK)**

## İçindekiler

**SORU 1: Sıcak Patates Algoritması (Hot Potato Routing Algorithm)**

**SORU 2: ADSL**

**SORU 3: CDMA (code division multiple access)**

**SORU 4: İstatistiksel Çoklama (Statistical Multiplexing)**

**SORU 5: Jitter (Dalga Bozulumu)**

**SORU 6: ICMP (Internet Kontrol Mesajı Protokolü)**

**SORU 7: Eşlik Kontrol Matrisi (Parity Check Matrix)**

**SORU 8: Kod Kelimesi**

**SORU 9: Birbirini Dışlama (Mutually Exclusive)**

**SORU 10: Overhead (Ek Yük)**

**SORU 11: JAVA ile Sohbet İstemci/Sunucusu**

**SORU 12: peer to peer (uçtan uca iletişim)**

**SORU 13: Güvenli Ağ Protokolü (Reliable Network Protocol)**

**SORU 14: VLAN (Sanal Yerel Ağ, Virtual Local Area Network)**

**SORU 15: Çerezler (Cookies)**

**SORU 16: MIME**

**SORU 17: Ara kilit Protokolü (Interlock Protocol)**

**SORU 18: Protokol (Protocol, Teşrifat)**

**SORU 19: CSMA (Carrier Sense Multiple Access)**

**SORU 20: Atomluluk (Atomicity)**

**SORU 21: Entropi (Entropy, Dağınım, Dağıntı)**

**SORU 22: SMTP ( Simple Mail Transport Protocol)**

**SORU 23: DHCP Sunucu (DHCP Server)**

**SORU 24: Uniform Dağılım ( Uniform Distribution, Yeknesak, Tekdüze, Biteviye)**

**SORU 25: Traceroute**

**SORU 26: Extranet (Dış ağ)**

**SORU 27: Intranet (İç Ağ)**

**SORU 28: Çift Yönlü İletişim (Duplex Communication)**

**SORU 29: POP3**

**SORU 30: Noktadan Noktaya İletişim (Point to Point Protocol PPP)**

**SORU 31: DNS (Domain Name System, Alan İsim Sistemi)**

**SORU 32: Köprü (Bridge)**

**SORU 33: Tekrarlayıcı (Repeater)**

**SORU 34: Sıralama Algoritmaları (Sorting Algorithms)**

**SORU 35: Düğüm (Node)**

**SORU 36: Eşlik biti kontrolü (parity bit check)**

**SORU 37: Aloha**

**SORU 38: Çoklamak (multiplexing)**

**SORU 39: BGP (Sınır Kapısı Protokolü, Gümrük Protokolü, Border Gateway Protocol)**

**SORU 40: Otonom Sistem (Autonomous System)**

**SORU 41: Ters Zehir (Reverse Poison)**

**SORU 42: Mesafe Vektörü (Distance Vector)**

**SORU 43: CSMA ( Çoklu Erişimde Hat Kontrolü, Carrier Sense Multiple Access)**

**SORU 44: CRC (cyclic redundancy check, çevrimsel fazlalık sınaması)**

**SORU 45: İnternet Toplam Kontrolü (Internet Checksum)**

**SORU 46: Ortam Erişim Kontrolü (Media Access Control)**

**SORU 47: İnternet Katman Kümesi (Internet Layer Stack)**

**SORU 48: IP Tünelleme (IP Tunnelling)**

**SORU 49: Çift Küme (çift yığıt, Dual-Stack)**

**SORU 50: IPv4'den IPv6 geçişi ( Transition from IPv4 to IPv6)**

**SORU 51: Geçiş Günü (Flag Day)**

**SORU 52: Protokol Kümesi (Protocol Stack , Protokol Yığıtı)**

**SORU 53: Sanal Devre (virtual circuit)**

**SORU 54: Gönderme Gecikmesi (transmission delay)**

**SORU 55: alt ağ (subnetwork)**

**SORU 56: Kruskal Asgari Tarama Ağacı Algoritması**

**SORU 57: Prim asgari tarama ağacı Algoritması**

**SORU 58: asgari tarama ağacı (en kısa örten ağaç, minimum spanning tree)**

**SORU 59: yönlendirici (router)**

**SORU 60: en uzun önek eşleşmesi (longest prefix matching)**

**SORU 61: TCP AIMD (additive increase multiplicative decrease, toplanarak artan çarpılarak azalan)**

**SORU 62: TCP Reno , Tahoe**

**SORU 63: tıkanıklık önleme (congestion avoidance)**

**SORU 64: tıkanıklık (congestion)**

**SORU 65: hızlı kurtarma (fast recovery)**

**SORU 66: yavaş başlangıç (slow start)**

**SORU 67: tıkanıklık penceresi (congestion window)**

**SORU 68: http (hyper text transfer protocol, hipermetin transfer protokolü)**

**SORU 69: kapsülleme (encapsulation)**

**SORU 70: Çıktı (throughput)**

**SORU 71: paket kaybı (packet loss)**

**SORU 72: sıra gecikmesi (queueing delay)**

**SIRA 73: noktasal gecikme (nodal delay)**

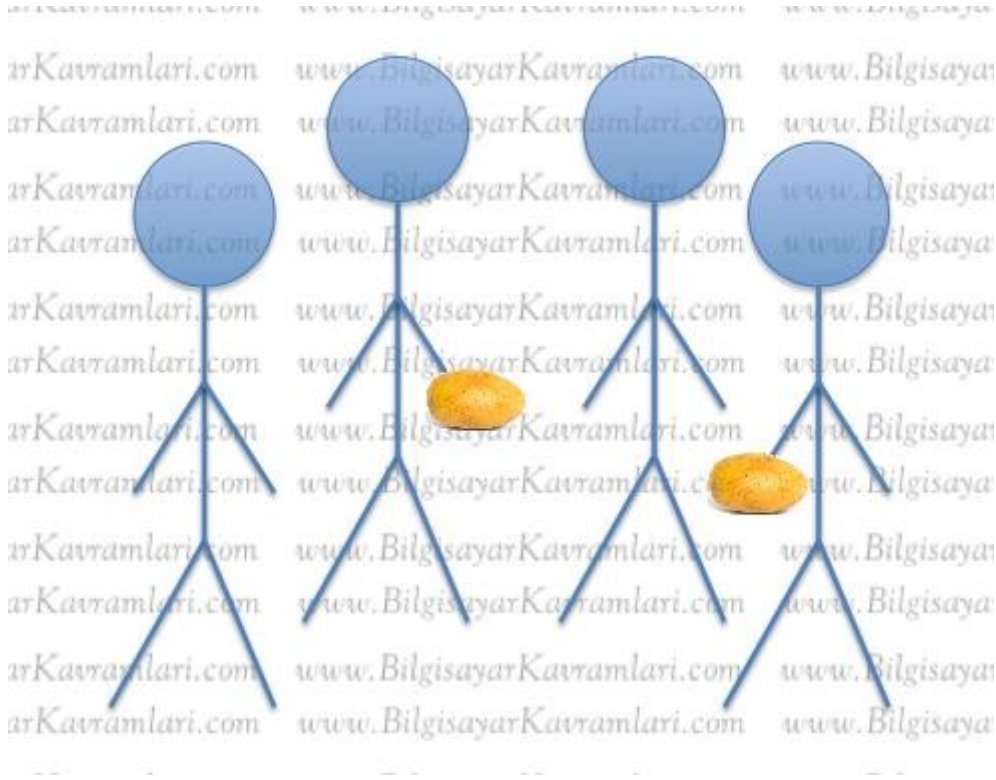
**SORU 74: frekans bölmeli çoklama (sıklık bölmeli çoklayıcı, frequency division multiplexing, fdm)**

**SORU 75: zaman bölmeli çoklama (time division multiplexing, tdm)**

## SORU 1: Sıcak Patates Algoritması (Hot Potato Routing Algorithm)

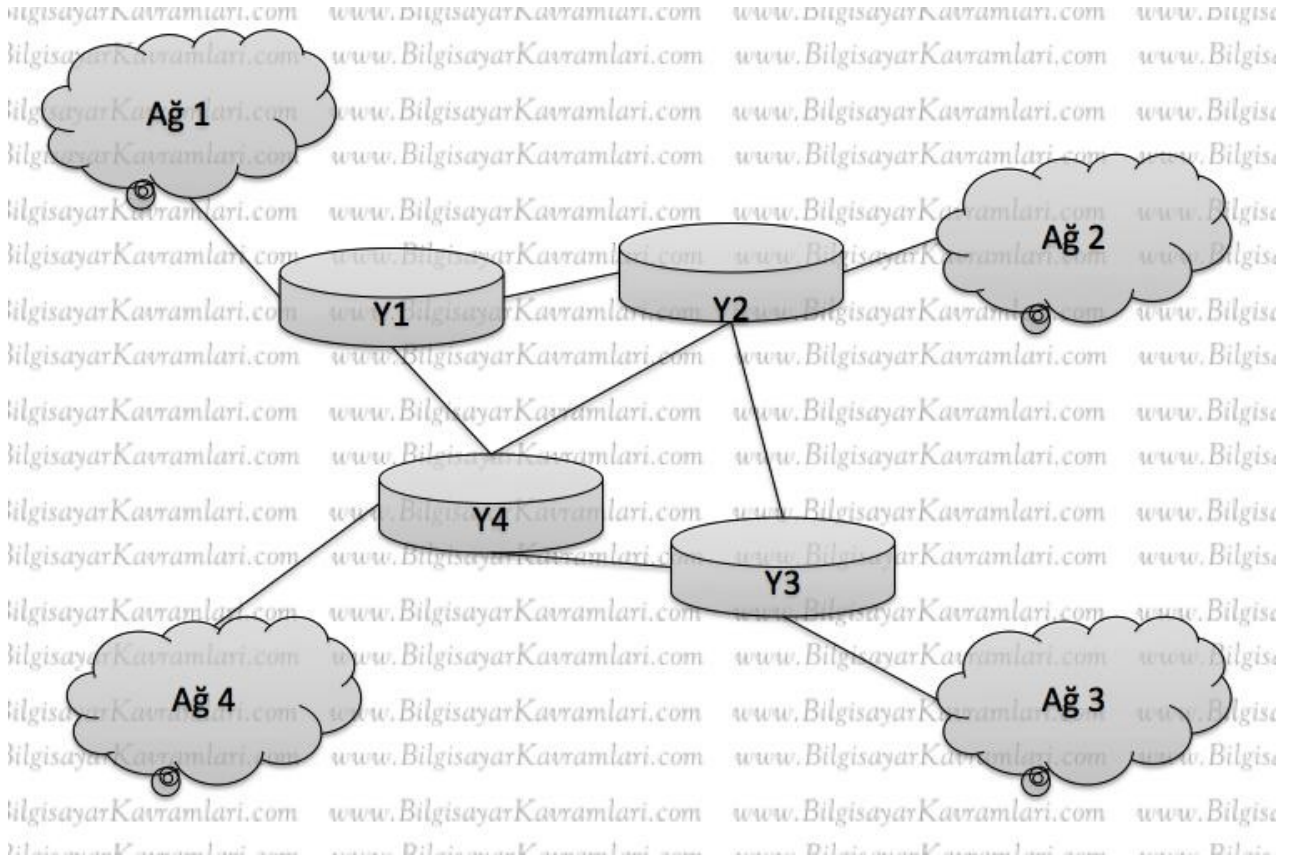
Bu yazının amacı, sıcak patates (hot potato) ve soğuk patates (cold potato) yönlendirme algoritmalarını (routing algorithms) açıklamaktır.

Basitçe konuyu açıklayacak olursak, yönlendirme algoritmaları (routing algorithms) internet paketlerinin yönlendiriciler (routers) arasında gönderilmesi temeline dayanır. Sıcak patates algoritması bir yönlendirme algoritması olarak, bir router üzerine bir paket geldiğinde, bu yönlendiricinin bağlantıda olduğu yönlendiricilerden birisine düşünmeden hızlıca paketi geçirmesine dayanır. İsmi de buradan gelmektedir. Yani elinde sıcak bir patates tutan kişinin patatesi alelacele yakınındaki birisine atması ve onunda bir başkasına atması olarak düşünülebilir.

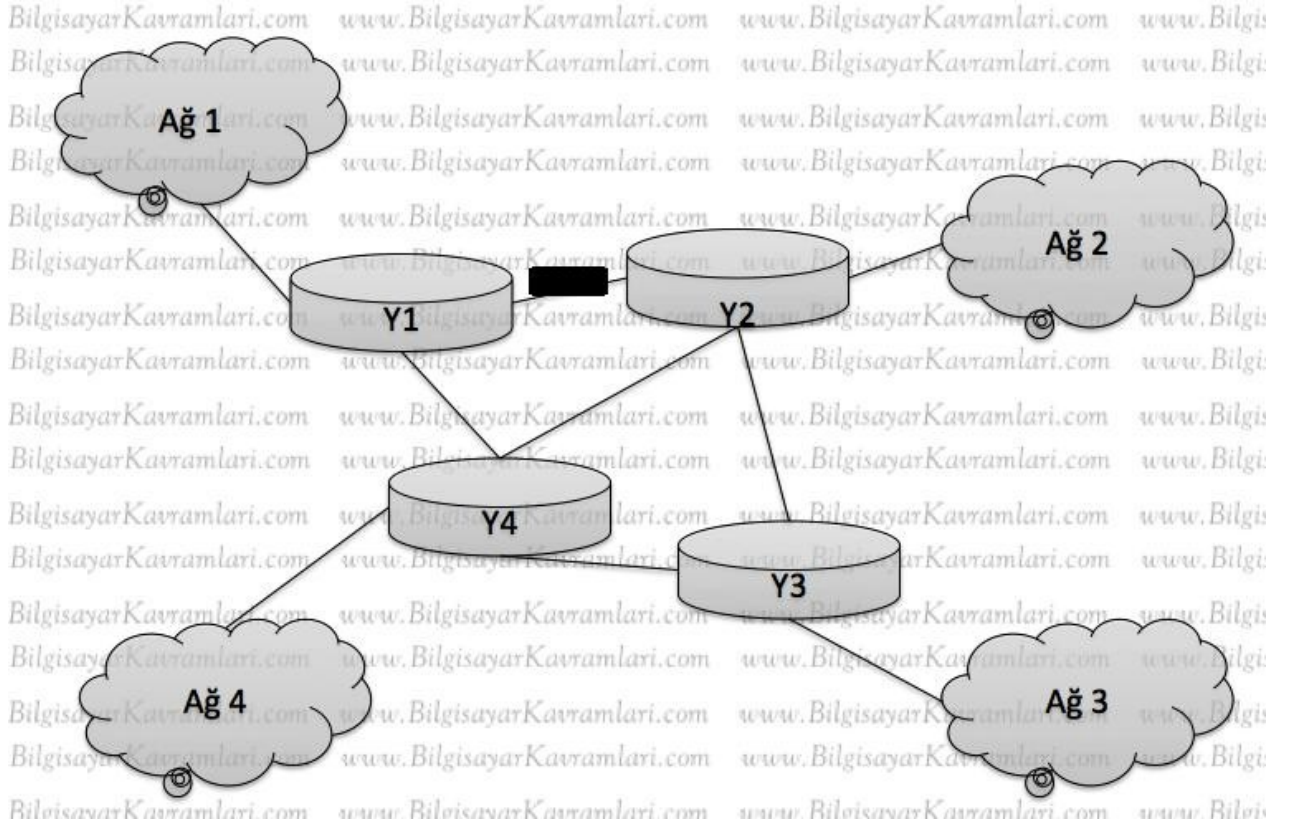


Yukarıdaki şekilde gösterildiği üzere, elinde patates tutan kişi, bunu en kısa sürede elinden çıkararak en yakındaki kişiye yollamaya çalışır. Bu sayede patates, rast gele seçilen kişileri dolaşarak bu kişilerin herhangi birisinin kendisini kabul etmesini bekler.

Şayet paket, rast gele olarak kabul edileceği bir kişiye gelirse bu durumda daha fazla yönlendirilmez ve o kişinin arkasındaki ağa dağıtım yapılır.

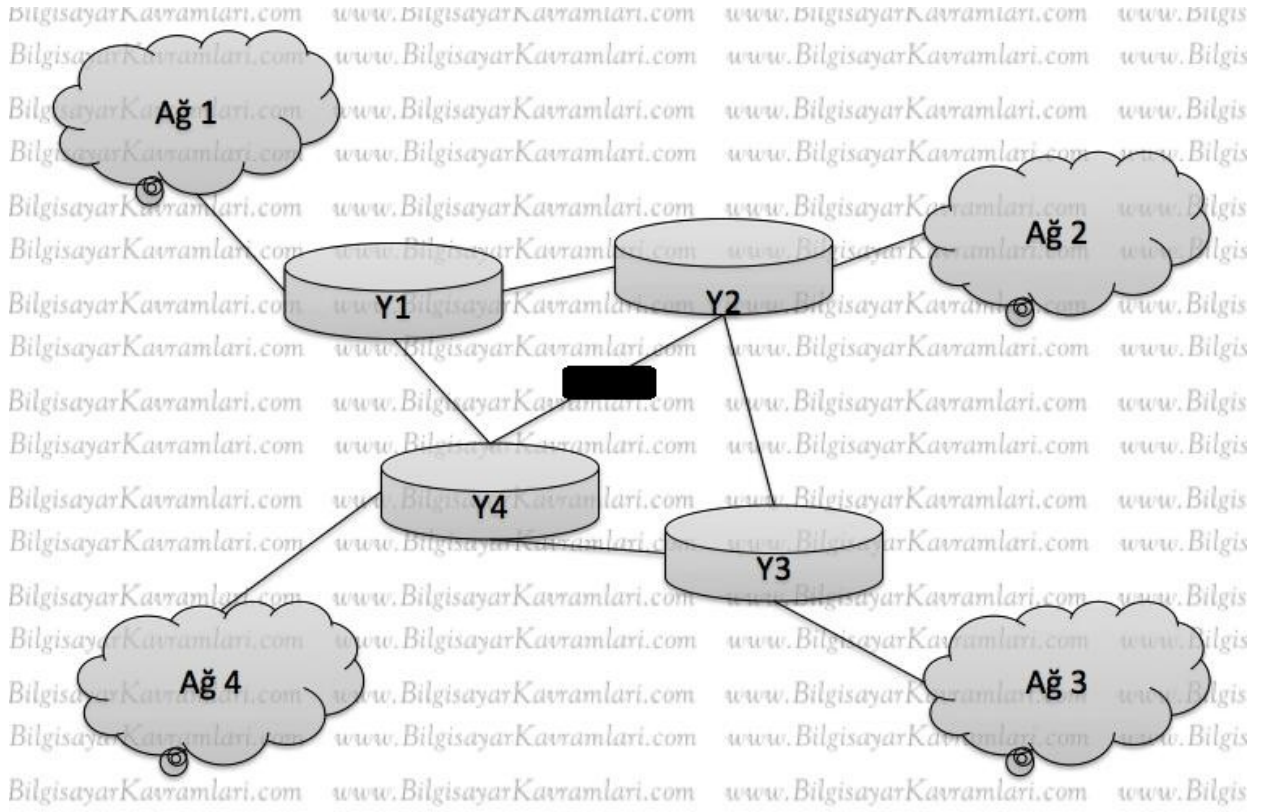


Yukarıdaki şekilde 4 farklı ağa hizmet veren 4 farklı yönlendirici olduğunu düşünelim. Örneğin Ağ1'den bir bilgisayar, Ağ3'e bir paket yollamak istiyor olsun. Bu durumda Y1 paketi komşularından birisine rast gele olarak yollayacaktır. Y1'in komşuları Y2 ve Y4'tür ve diyelim ki Y2'ye rast gele olarak yollamış olsun :

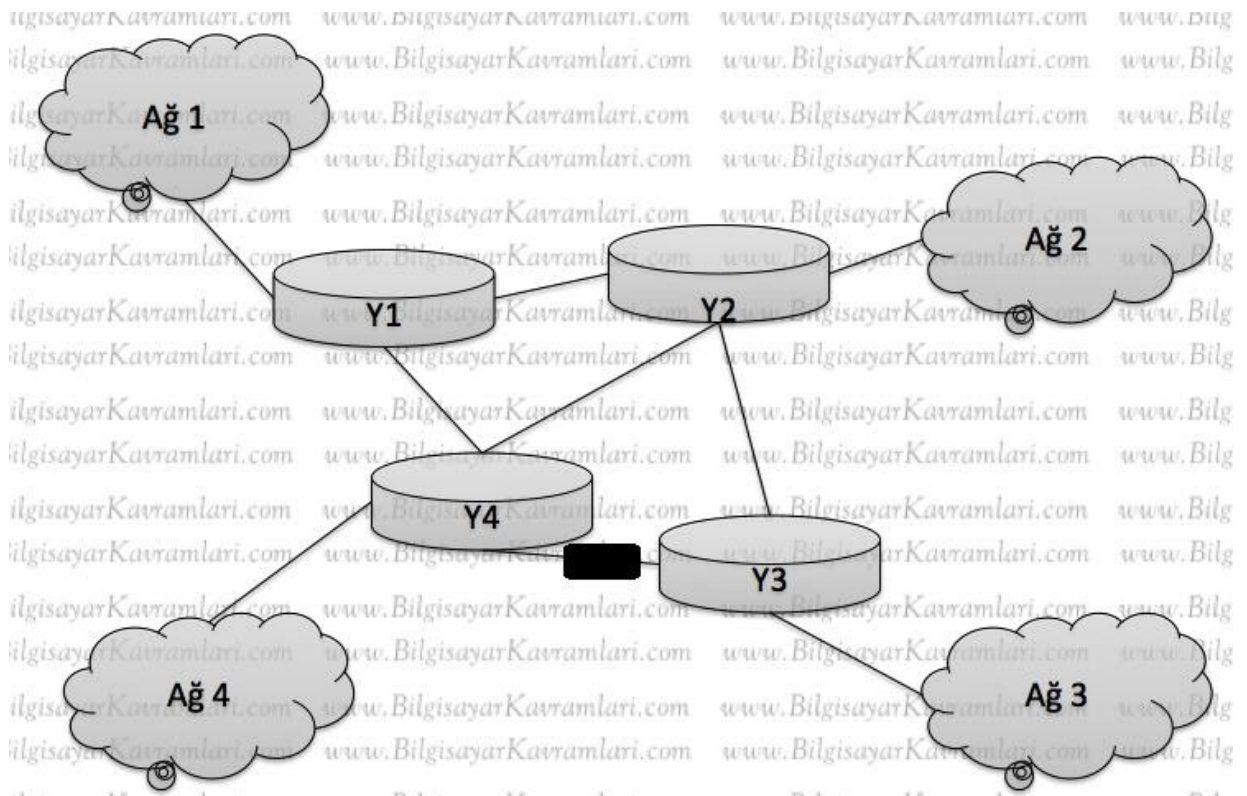


Y2 bu paketi aldıktan sonra, kendi ağında dağıtamaz çünkü paketin hedefi Ağ3 içerisinde. Bu durumda Y2'nin komşuları arasında bir seçim yapıp paketi yollaması beklenir. Y2'nin komşuları, Y1, Y4 ve Y3 olduğu için bu komşulardan birisine rast gele olarak paketi yollar (burada paketin geldiği Y1 yönlendiricisinin de listede olduğuna dikkat ediniz). Diyelim ki Y4 yönlendiricisine yollamış olsun:





Y4 yönlendiricisi paketi aldıktan sonra paket kendisine ait olmadığı için yine komşularından birisini rast gele seçerek dağıtacaktır. Komşuları, Y1, Y2 ve Y3 olduğuna göre bunlardan birisini rast gele olarak seçip yollar. Diyelim ki Y3 yönlendiricisini seçmiş olsun:





Nihayi olarak paketi alan Y3 yönlendiricisi, paketi kendi ağı içerisinde dağıtır ve Ağ3 içerisindeki alıcı paketi teslim alır. Bu durumda paketin daha fazla yönlendirilmesine gerek kalmaz.

Görüldüğü üzere paketin teslimi tamamen rast gele bir mantıkla yapılmaktadır. Bu açıdan yönlendirme algoritmasının kodlanması veya anlaşılması gayet basittir. Herhangi bir veri yapısı (Data structure) tutma zorunluluğu yoktur. Ancak algoritmanın çalışma başarısı oldukça düşüktür. Paketin sürekli olarak ağda dolaşması söz konusudur.

Buradaki temel mantık, paketin hiçbir zaman için kaybolmamasıdır. Yani paket gideceği hedeften uzaklaşsa bile ağda sürüklenmeye devam eder. Hedefe ulaşana kadar da dolaşmaya devam eder.

Soğuk patates algoritması (cold potato algorithm) sıcak patates algoritmasının tam tersi gibi düşünülebilir. Buna göre sıcak patates algoritmasında, yönlendiriciler arasında hiçbir bilgi tutulmamaktadır. Tamamen rast gele bir yaklaşımla paketler birbirine gönderilebilmektedir. Buna karşılık soğuk patates algoritmasında, sistem yöneticileri tam olarak paketlerin yönlendirilme şeklini yönlendiriciler üzerinde tanımlar ve yönlendiriciler buna göre çalışırlar. Bu algorithmada sistemlerin birbirine güvenmesi esas alınır ve rast gele olarak çalışan hiçbir unsur bulunmaz.

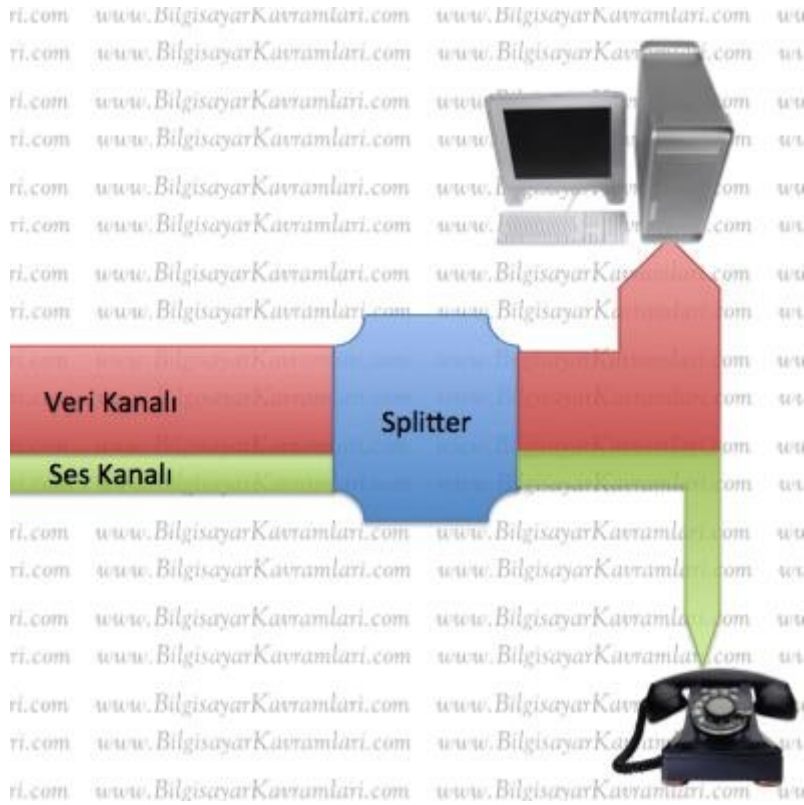
## **SORU 2: ADSL**

Bu yazının amacı, günümüzde sıkça internet bağlantısı için tercih edilen ADSL teknolojisini açıklamaktır. ADSL kelimesi, İngilizce Asymmetric Digital Subscriber Line kelimelerinin baş harflerinden oluşmaktadır ve Türkçede asimetrik dijital üye hattı gibi bir terim ile karşılanabilir.

Teknolojinin en belirgin özelliği, giden ve gelen verilere ayrılan bant genişliklerinin simetrik olmamasıdır. Yani örneğin indirme (download) için 1Mbit hat tahsisi yapılırken yükleme (upload) için 256Kbit hat tahsisi yapılıyor olabilir. Bu açıdan, genelde indirme (download) ağırlıklı kullanıcılardan oluşan ev kullanıcıları için oldukça cazip bir teknolojidir.

Teknoloji basitçe bakır telefon hatları üzerinden çalışabilir. Frekans paylaşımli bir ortamda (frequency division multiplexing (FDM)) veri iletimi olduğu için, telefon hattı üzerinde, aynı anda hem telefon konuşmaları hem de veri iletimi gerçekleştirilebilmektedir. Yani veri iletişimi için ayrılan bant genişliğinin farklı frekansları veri ve ses için farklı olarak tahsis edilir. Bu durumu, aynı anda farklı frekanslardan yayın yapan radyo sinyallerine benzetmek mümkündür. Aslında herhangi bir FM alıcısı radyo, bütün frekansları o anda almaktadır ancak sadece ayarlanmış olduğu frekanstaki gelen sinyalleri sese dönüştürerek dinlememize imkan sağlar. Diğer frekansları almak istiyorsak, diğer frekanslara ayarlı farklı radyolara ihtiyaç duyarız.

Benzer durum telefon hattı üzerinden taşınan veriler için de geçerlidir. Uygun alıcı ayarı yapıldıktan sonra, hatta taşınan ses veya veri kısmı ayrıştırılabilir. Bu ayrıştırma işlemi ayırıcı (splitter) ismi verilen özel bir donanım ile yapılmaktadır.



Yukarıdaki şekilde görüldüğü üzere tek bir bakır kablo hattıyla taşınan veri ve ses hatları, splitter marifeti ile iki ayrı hatta bölünmekte ve hatlardan birisi telefona diğeri ise veri iletişimi için bilgisayara veya ADSL MODEM'e yönlendirilmektedir.

Santral tarafında, hem ses hem de veri birleştirilerek aynı hatta indirgenir. Yani evimize kadar gelen bakır kabloların aynı anda hem internet verisi hem de konuşma verisini taşıması için, ses verilerinin geldiği telefon santrali ile internet verilerinin geldiği yönlendirici (router) aynı hatta ve farklı frekanslarda birleştirilmelidir. Bu birleştirme işlemi DSLAM (Digital Subscriber Line Access Multiplexer) ismi verilen ve Türkçede “dijital abone hat erişimi çoklayıcısı” olarak çevrilebilecek bir ilave cihaz ile yapılmaktadır.



Yukarıdaki resimde, bir DSLAM sunucusu ve üzerinde bağlı olan modemler görülmektedir.

Bu işlem, genelde telefon firması tarafından telefon hattının bağlı bulunduğu santralde yapılmaktadır. Ayrıca her telefon hattı için santral kısmında ilave bir modem bulundurulmaktadır.

## ADSL teknolojisi

Yukarıdaki giriş kısmından sonra teknolojinin çalışma detaylarına girebiliriz. Teknoloji basit olarak FDM veya TDM yaklaşımlarından birisini kullanır. Yani veri farklı frekans aralığından kesintisiz olarak veya farklı zamanlarda paylaşımlı olarak iletilmektedir.

Her iki teknoloji tercihi için de yükleme akışı (upstream) için ayrılan aralık, indirme akışı (downstream) için ayrılan aralıktan çok daha azdır (genelde  $\frac{1}{4}$  oranında).

Örneğin frekans paylaşımı yapılan bir ortamda, annex A tipi iletişim için yükleme akışına 26,000 ile 137,825kHz arasındaki frekans bandı ayrılırken, indirme akışı için çok daha geniş bir aralık olan 138kHz ile 1104kHz arasındaki bant ayrılmaktadır.

ADSL teknolojisi, bu alanları da daha küçük parçalara bölmektedir. Yaklaşık 4.3kHz genişliğindeki bu alt kanallara terminolojide kutu anlamında “bin” ismi verilmektedir.

ADSL teknolojisi, iletişim kurma aşamasında her bini ayrı ayrı test etmekte ve SNR (signal to noise ratio, sinyal gürültü oranı) değerlerine göre bu binleri kullanıp kullanmamaya karar vermektedir. Genelde mesafeye bağlı olarak gürültünün artacağını düşünürsek, ADSL teknolojisinin neden mesafeye bağlı olarak hızının değiştiği anlaşılabilir. Yani mesafe arttıkça bazı binler daha gürültülü olduğu için kullanıma kapatılacak ve neticede de kullanılabilir bant genişliği düşecektir.

Ayrıca ADSL MODEM’ler, gönderim veya alım için bant genişliğini farklı değerlerle bu binlere dağıtabilmektedir. Örneğin bir bin’in taşıyacağı veri diğerine göre 2 veya 3 misli fazla olabilir. ADSL MODEM bu değerlere yapmış olduğu SNR testlerine göre karar vermektedir.

ADSL2+ teknolojisinde ise her binde tek bit taşıma yaklaşımı kullanılmaktadır. Bu yaklaşımda gürültülü binler hiç kullanılmaz.

Bu aşamada ADSL teknolojisi muhafazakar bir yaklaşımla bit per bin (bin başına bit) değerini düşük tutabilir. Bu yaklaşımda veri iletişiminin yavaşlaması baştan kabul edilmiş olunur ancak amaç, iletişim sırasındaki veri kaybını asgariye indirmektir. Öte yandan biraz daha cesur bir yaklaşımla daha yüksek bin başına bit değeri ile daha fazla veri transferi ve dolayısıyla daha hızlı bir iletişim hedeflenebilir. Ancak bu durumda verinin kaybolma riski daha da artacaktır. İşte bu durumda daha üstte çalışan TCP/IP gibi protokoller paket kayıpları yaşayacağından verinin tekrar ve tekrar yollanması yüzünden hızın yine yavaşlaması söz konusu olabilecektir. Burada iki uç arasında dengeli bir seçim yapılması en hızlı çözümü getirecektir.

ADSL2+ teknolojisi burada dikişsiz oran uyumu (seamless rate adaptation (SRA)) ismi verilen bir çözüm önermektedir. Bu çözüme göre iki taraf arasındaki bin başına bit haberleşmesi çok daha az iletişim ile çözülebilmektedir. Yani ADSL teknolojisi üzerinden iletişim halinde olan taraflar (ev kullanıcısı ve ADSL sunucusu) ilgili bin başına ayrılan bit değerini değiştirmeye karar verdiklerinde bu bilginin iki taraf tarafından da bilinmesi gerekmekte ve bu bilginin taraflar arasında taşınması çok daha az haberleşme aşamasında sağlanmaktadır.

ADSL teknolojisini iyileştirmek için denenen yollardan birisi de, ADSL için ayrılmış olan özel frekans aralıklarının ötesindeki frekansların kullanılmasıdır. Bu yaklaşımda birinci

problem, iki tarafta da (ADSL kullanıcısı tarafında da) özel bir cihaz bulunması ve bu yüksek frekans değerlerini algılaması gerekliliğidir. Bu ilave bir maliyet getirir.

Ayrıca ADSL verisinin telefon hatları üzerinden taşındığı unutulmamalıdır. Dolayısıyla çoğu yerde birbirine yakın geçen telefon hatları üzerinden yüksek frekans değerlerinde veri iletimi, çoğu zaman çarpaz konuşma (crosstalk) ismi verilen ve bir kablodaki veri iletişimi sırasında oluşan manyetik alanın diğer hatta etkilemesi olarak anlaşılabilecek problemin doğmasına sebep olur.

### **SORU 3: CDMA (code division multiple access)**

Bilgisayar bilimlerinde, özellikle ağ (network) konusunda geçen ve bir ortamı, birden fazla veri kanalının iletişimi için kullanılan yöntemlerden birisidir. Literatürde sıkça geçen diğer çok kanallı veri iletişim yöntemleri, [TDMA \(time division multiple access, zaman paylaşımli çoklu erişim\)](#) ve [FDMA \(frequency division multiple access, frekans paylaşımli çoklu erişim\)](#) yöntemleridir.

CDMA yöntemini bu diğer meşhur iki yöntem ile karşılaştırmak için genelde şu şekilde bir örnek verilir. Örneğin bir odada birden çok kişinin konuşarak haberleştiğini düşünelim. TDM yaklaşımında, kişiler sırayla ve teker teker konuşmakta, ilgili alıcı konuşan kişinin mesajını almaktadır. FDM yaklaşımında, kişiler farklı ses tonları ile konuşmakta ve dolayısıyla alıcı olan kişi, ilgili ses tonuna dikkatini vererek iletilen mesajı almaktadır. CDMA yaklaşımında ise, kişiler farklı lisanlarda konuşmakta, dolayısıyla o lisanı bilen kişiler tarafından algılanmakta, diğer kişiler tarafından iletilen veri gürültü olarak algılanıp dikkate alınmamaktadır.

#### **Örnek**

Konuyu bir örnek üzerinden açıklamaya çalışalım. Örneğin 4 farklı veri kanalı üzerinden veri akmakta olsun ve bunları CDMA yöntemi ile tek bir kanaldan taşımak isteyelim.

- V1: 1101
- V2: 0010
- V3: 1010
- V4: 0011

Yukarıdaki şekilde verilen 4 farklı verinin CDMA ile nasıl taşındığını anlatalım. Verileri ilk adımda farklı frekans değerine sahip işaretler ile kodluyoruz (code). Örneğimizde kullanacağımız 4 farklı kodumuz aşağıdaki şekilde olsun:

- K1: 1111
- K2: 1010
- K3: 1100
- K4: 1001

Verilerin, kodlar tarafından işlenebilmesi için ve 4 farklı verimiz olduğu için, verilerin genliğini 4 misli şeklinde düşünebiliriz. Buna göre örnek olarak son veri için kodlamayı anlatalım:

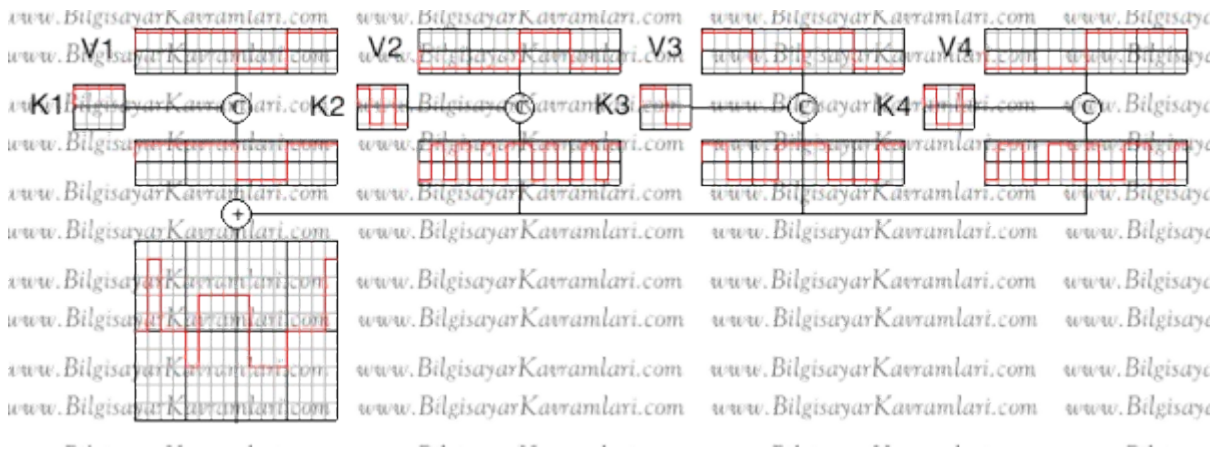
V4 : 0000 0000 1111 1111 (Gösterim için 0011 verisinin her elemanını 4 kere tekrarladım.)

K(V4,K4) : 0110 0110 1001 1001 (V4'ün, K4 ile kodlanması sonucunda, V4 üzerindeki 1 değerleri için K4'ün kendisi, V4 üzerindeki 0 değerleri için ise K4'ün tersi gelmektedir. Daha basit anlamda her V4 dördlüsü (uzun şekilde yazılmış halini düşünün) ile K4 değerlerinin özel veyasının (XOR) tersi alınır !(( 0000 0000 1111 1111 ) XOR ( 1001 1001 1001 1001)) şeklinde)

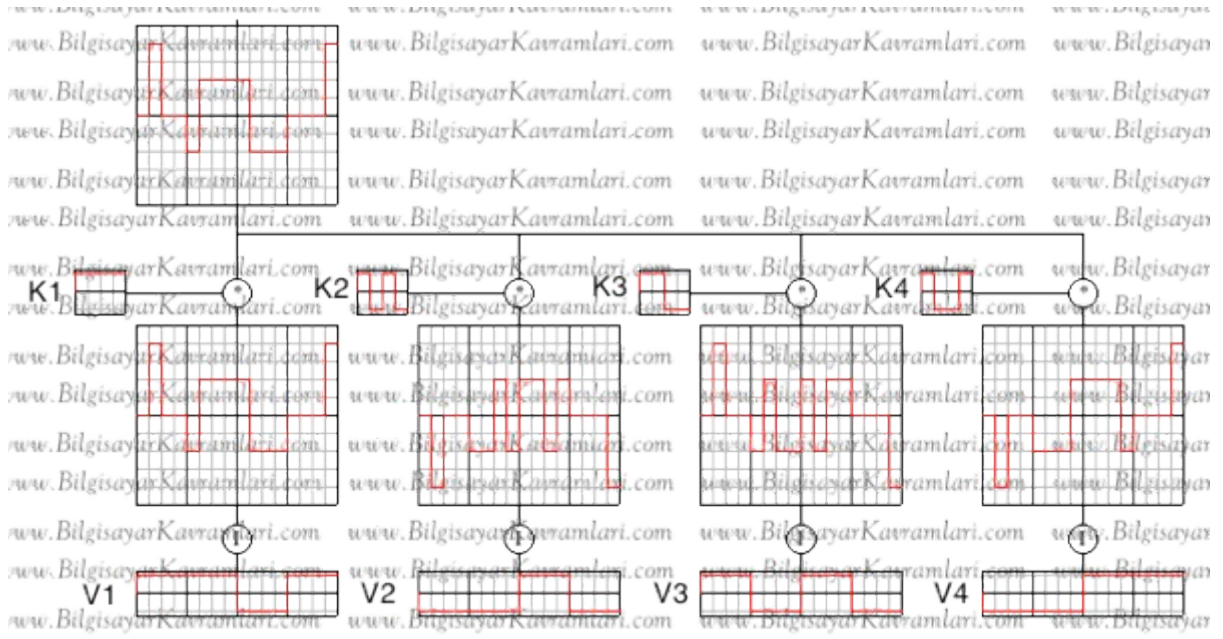
Sonuçta 4 farklı veri ve 4 farklı kodlama için aşağıdaki sonuçlara ulaşılır:

- K(V1,K1) : 1111 1111 0000 1111
- K(V2,K2) : 0101 0101 1010 0101
- K(V3,K3) : 1100 0011 1100 0011
- K(V4,K4) : 0110 0110 1001 1001

CDMA algoritmamızda, son adım olarak yukarıdaki değerleri topluyoruz. Toplamın ve yukarıdaki işlemlerin görsel olarak ifadesi aşağıdaki şekildedir:



Yukarıdaki toplama işlemi sonucunda elde edilen verilerin, her birisinin farklı alıcılar tarafından alınmak istediğini düşünelim. Bu durumda her alıcı, almak istediği göndericinin kodlama değerini kendisinde ayarlayacak ve yukarıda elde edilen sonuç verisini kendisinde işleyecektir. Bu durum aşağıdaki şekilde gösterilmektedir:



Yukarıda görüldüğü üzere her kodlama değeri sonucunda açılan veri, orjinal olarak kodlanan ilgili veridir. Örneğin K1 kodlamasından açılan veri V1 olarak bulunmuştur. Bu işlem diğer kodlamaları engellememektedir.

CDMA yöntemi, günümüzde de kullanılan UMTS teknolojisinin temelini oluşturur. UMTS (universal mobile telecommunication system, evrensel hareketli telekomunikasyon sistemi) teknolojisi, CDMA2000 teknolojisinden sonra (IMT Multi Carrier, inter mobile telecommunications, hareketli telekomunikasyonlar arası çoklu taşıyıcı olarak da bilinir) geliştirilen ve CDMA 2000 teknolojisi ile rekabeti amaçlayan bir teknolojidir. CDMA2000 de, UMTS'in temeli olan W-CDMA de birer 3G teknolojisidir ve cep telefonlarının aynı anda iletişimi için kullanılmaktadır.

#### **SORU 4: İstatistiksel Çoklama (Statistical Multiplexing)**

Bilgisayar bilimlerinde, özellikle ağ (network) konusunda kullanılan bir terimdir. Ağ iletişimi için kullanılan kanalın, birden fazla veriyi aynı anda iletmesi istendiği durumlarda genel olarak TDM (zaman paylaşımı çoklamalı) veya FDM (frekans paylaşımı çoklamalı) alternatiflerinden birisi kullanılır.

İstatistiksel çoklama ise, TDM paylaşımının özel bir çeşididir. Basitçe ağ genişliğini (bandwidth), istatistiksel yöntemlere göre, transfer edilecek veriler arasında paylaştırır. Bu yüzden literatürde istatistiksel zaman bölmeli çoklama (statistical time division multiplexing, STDM) olarak da geçmektedir.

STDM yöntemi, birden fazla veri kanalını aynı anda taşıırken, iletilmek istenen verilerin arasında rastgele bir değerle tercih yapar. Bu yüzden, klasik TDM yöntemine göre, kanalın boş kalması ihtimali yoktur. Daha iyi anlatabilmek için bir örnek verelim:

İletilmek istenen veriler aşağıdaki şekilde olsun:



- V1 50Kb
- V2 20Kb
- V3 50kb

Örneğin bir paketle iletebileceğimiz veri miktarı da 20Kb olsun. Yukarıdaki verilerin TDM ile iletilmesi durumunda, aşağıdakine benzer bir tablo ortaya çıkacaktır:

T1:V1(20Kb)	
T2:V2(20Kb)	
T3:V3(20Kb)	Bu noktada bütün verilerin 20Kb'lık parçaları iletildi
T4:V1(20Kb)	
T5:V2(20Kb)	
T6:V3(20Kb)	
T7:V1(10Kb)	ilk 40kb zaten iletildiği için kalan 10Kb iletiliyor ve pakette 10Kb boşluk bulunuyor
T8:V2(0Kb)	V2'nin bittiğini anlıyoruz
T9:V3(10Kb)	

Yukarıda görüldüğü üzere T8 anında, 20Kb kapasitesi olan paketimiz gereksiz yere vakit kaybına sebep olmuştur. Çözüm olarak istatistiksel yöntemin kullanılması halinde, aşağıdakine benzer bir durum ortaya çıkacaktır.

Yukarıdaki veriler ve paket genişliği için aynı örneği STDM üzerinden çözmeye çalışalım. Öncelikle bir veri yapısı olarak [sıra \(queue\)](#) belirliyor ve bu sırada verileri ve kalan boyutlarının bilgisini tutuyoruz:

Q	->	V1(50Kb)	->	V2(40Kb)	->	V3 (50Kb)
T1:V1(20Kb)	Q	->	V2(40Kb)	->	V3(50Kb)	-> V1 (30Kb)
T2:V2(20Kb)	Q	->	V3(30Kb)	->	V1(30Kb)	-> V2 (20Kb)
T3:V3(20Kb)	Q	->	V1(30Kb)	->	V2(20Kb)	-> V3 (30Kb)
T4:V1(20Kb)	Q	->	V2(40Kb)	->	V3(50Kb)	-> V1 (10Kb)
T5:V2(20Kb)	Q	->	V3(30Kb)	->	V1(10Kb)	
T6:V3(20Kb)	Q	->	V1(10Kb)	->	V3(10Kb)	
T7:V1(10Kb)		Q	->	V3(10Kb)		
T9:V3(10Kb)	Q ->	Boş				

Yukarıda görüldüğü üzere, sıra (queue) kullanılarak erişilecek verilerin listesi tutulmuş ve bu verilere [ilk gelen çalışır \(first come first serve, FIFO\)](#) yaklaşımı ile erişilmiştir.

STDM yönteminde, erişim şekli, farklı zamanlama algoritmaları (scheduling algorithms) kullanılarak iyileştirilebilir. Örneğin öncelik sırası (priority queue) kullanılarak bazı veri kanallarına öncelik verilmesi mümkündür. [Adil zamanlama algoritmaları ile \(fair share scheduling\)](#) veri iletimini dengelemek veya bütün verilerin aynı zamanda bitmesi veya küçük verinin daha hızlı iletilmesi, kullanılacak olan zamanlama algoritması ile ayarlanabilir. Hatta tamamen rast gele değerler üreterek veri iletimi rast gele bir sürece dönüştürülebilir.



**Sarnıçlama Dalga Bozulumu (Sampling Jitter):** Bu kavram, genelde işaret (sinyal) üzerinde uygulanan çevirimler sırasında ortaya çıkar. DAC (digital to analog converter, dijital verinin analog veriye çevirimi) veya tersi olan ADC (analog to digital converter, analog verinin dijital veriye çevirimi) işlemleri belirli bir zaman almaktadır. O halde sinyal işlenirken, beklenen zamana göre gecikmeli olarak sonuç elde edilecek ve nihayetinde bir dalga bozulumu yaşanacaktır.

Örneğin sarnıçlama yapılan (belirli aralıklarla örnekler alınan, sampling) bir sistemin, ses, ışık veya hız gibi sürekli (conitinous) bir işaret (signal) olduğunu kabul edelim. Bu işaretin belirli zamanlarda değerinin okunarak dijital ortama çevirimi, burada bahsedilen gecikmeler ve kaymalara neticede de sarnıçlama dalga bozulumuna sebep olacaktır.

**Paket Dalga Bozulumu:** Bilgisayar ağlarında, bazı durumlarda, paketlerin belirli sıklıkta (frequency) iletilmesi beklenir. Bu sıklığın bozulması da bir dalga bozulumu (jitter) olarak kabul edilebilir. Bilgisayar ağlarındaki dalga bozulumu (jitter) aslında başlı başına bir hizmet kalitesi (quality of service) konusudur ve daha çok kabul gören PDV (packet delay variation) terimi altında kullanılmaktadır.

Yukarıda verilen örnekler daha da arttırılabilir. Örneğin bir CD-ROM'dan okuma sırasında, CD üzerindeki verinin aranması sırasında geçen süre, herhangi bir veri transfer yazılımı veya devresinin, veriyi göndermeye başlamasında geçen süre, kuantum kapılarının (qunatum gates), elektron dönüşünden kaynaklanan (spin based) çalışma gecikmesi veya aktarılacak istenen verinin kanal kapasitesinin çok üzerinde olasıdan dolayı, verinin bir kısmının tıkanıklık (congestion) ile karşılaşması ve bu yüzden beklenen zamandan daha geç transfer edilmesi gibi durumlar birer dalga bozulumu (jitter) örneğidir.

## **SORU 6: ICMP (Internet Kontrol Mesajı Protokolü)**

İngilizce, Internet Control Message Protocol kelimelerinin baş harflerinden oluşan kısaltmadır. Türkçe olarak İnternet Tespit Mesajı Teşrifatı olarak çevrilebilir.

Genel olarak işletim sistemleri tarafından, ağda bulunan cihazların durumunu tespit amaçlı kullanılan bir [teşrifattır \(protocol\)](#). Örneğin bir cihaza erişilip erişilemediğini tespit için gönderilen mesaj tipidir.

ICMP mesajları, birer IP paketi halinde yollanmaktadır. ICMP hem UDP hem de TCP üzerinde çalışabilmektedir ancak çalıştığı protokole göre farklılık gösterebilir. Örneğin UDP paketlerine itibar etmek doğru olmaz çünkü paket kaybı olabilir.

ICMP paketleri ayrıca üzerinde çalıştıkları [IP sürümüne göre \(IP Version\)](#) ismlendirilmektedirler. Örneğin IPv4 için olan paketlere ICMPv4, [IPv6](#) için olanlara ise ICMPv6 ismi verilmektedir.

ICMP paketlerinin 8 byte uzunluğunda bir başlığı (header) ve bunu takip eden ve değişken boyutta veri kısmı bulunur. Klasik bir windows ICMP paketi 32 byte uzunluğundayken, klasik bir UNIX / Linux paketi ise 64 byte uzunluğundadır. Bu durumda windows için veri uzunluğu 24 byte, linux için ise 56 byte olmaktadır.

ICMP paketinin başlığında ilk byte, ICMP tipini belirtir. İkinci byte ICMP kodunu, üçüncü ve dördüncü bytelar ise paketin tamamının, [toplam kontrolünü \(check sum\)](#) belirtir. Başlık boyutunun 8 byte olduğunu belirtimiştik, geri kalan 4 byte ise ICMP tip ve koduna göre değişiklik göstermektedir. Bu durumda ICMP paketi aşağıdaki yapıda olacaktır:

Bitle r	0–7	8–15	16–23	24–31
0	Tip	Kod	<a href="#">Toplam Kontrolü</a>	

**Yukarıdaki tiplerin değerine örnek olması açısından aşağıdaki tablodan istifade edilebilir:**

Ti p	Açıklama
0	Eko yanıt-ping yanıtı(Echo Reply)
3	Hedefe Erişilemedi(Destination Not Reachable)
4	Kaynak Kapatmak(Source Quench)
5	Yeniden Yönlendirme(Redirection Required)
8	Eko yanıt-ping isteği(Echo Request)
9	Yönlendirici tanıtımı
10	Yönlendirici istemi
11	Zaman aşımı–traceroute kullanır(Time to Live Exceeded)
12	Parametre Problemi(Parameter Problem)
13	Timestamp İstemi(Timestamp Request)
14	Timestamp Yanıtı(Timestamp Reply)
15	Bilgi İstemi(Information Request)
16	Bilgi Yanıtı(Information Reply)
17	Addres Maskesi istemi(Address Mask Request)
18	Addres Maskesi yanıtı(Address Mask Reply)

Yukarıdaki her tip için ayrıca kodlar bulunmaktadır.

Örneğin 5 numaralı tip olan yeniden yönlendirme tipinin (redirection required) alt kodları olarak aşağıdaki tabloda yer alan değerlerden birisi atanabilir:

Kod	Açıklama
0	Ağ için yönlendir (Redirect Datagram for the Network )
1	Sahibi için yönlendir (Redirect Datagram for the Host)
2	Servis tipine göre ve ağa göre yönlendir (Redirect Datagram for the TOS & network)
3	Servis tipine ve sahibine göre yönlendir (Redirect Datagram for the TOS & host)

Ayrıca diğer tiplerinde benzer şekilde alt kodları bulunmaktadır. ICMP paketi, ayrıca internet standartlarını belirleyen kurum olan IETF (internet engineerint task force) tarafından yayınlanan ve standartların yayınlandığı RFC dokümanlarında (yorum için talep, request for comment) 792 numaralı yayında geçmektedir. İlgili dokümana aşağıdaki bağlantıdan erişilebilir:

<http://tools.ietf.org/html/rfc792>

## SORU 7: Eşlik Kontrol Matrisi (Parity Check Matrix)

Hata kontrolü için kullanılan yöntemlerden birisidir. Veri güvenliği, veri iletimi veya veri sıkıştırma gibi alanlarda kullanılır. Genelde H sembolü ile gösterilir. Basitçe sistemde kullanılan üreteç matristen (generating matrix) çıkarılabilir. Bir eşlik kontrol matrisinin yapısı aşağıda verilmiştir:

$G = [I|P]$  şeklinde bir üreteç matris olmak üzere

$H = [P^T|I]$  şeklinde bir eşlik kontrol matrisi üretilebilir.

Örneğin aşağıdaki şekilde bir üreteç matrisimiz olsun:

Örneğin aşağıdaki üreteç matrisi ele alalım:

$$\left| \begin{array}{cccccc|c} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array} \right| = G$$

Bu matrisin ilk kısmı olan  $3 \times 3$  boyutlarındaki bölüm birim matristir

$$\left| \begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{array} \right| = I$$

İkinci kısım ise P ile gösterilen üreteç matrisin özel parçasıdır:

$$\left| \begin{array}{cc} 1 & 1 \end{array} \right|$$

$$\left| \begin{array}{cc} 1 & 0 \end{array} \right| = P$$

$$\left| \begin{array}{cc} 1 & 1 \end{array} \right|$$

Yukarıdaki parçaları birleştirmeden önce P matrisinin tersyüzünü (transpose) alıyoruz:

$$|111|$$

$$|101| = P^T$$

Ardından birim matris ile birleştiriyoruz:

$$|11110|$$

$$|10101| = H$$

Şeklinde yukarıda verilen üreteç matrisin, eşlik kontrol matrisi bulunur.

## SORU 8: Kod Kelimesi

Haberleşmede kullanılan bir terimdir. Bir kod kelimesi (code word), belirli bir teşrifatın (protocol, protokol) anlamlı en küçük parçasıdır. Her kod kendi başına tek bir anlam ifade eder ve bu anlam yeganedir (unique).

Aynı yaklaşım programlama dilleri için de geçerlidir. Her programlama dilinde bulunan her kelime tek bir anlam ifade eder.

Örneğin bir programlama dilindeki “if” kelimesi, bu dildeki kaynak kodda bulunan (source code) kod kelimesidir (code word).

Benzer durum herhangi bir haberleşme teşrifatında da olabilir.

Kod kelimeleri kullanıldıkları yere göre kanal kelimeleri (channel code words) veya kaynak kelimeleri (source code words) olarak isimlendirilebilir. İlki haberleşme ikincisi ise programlama tabiridir.

Ancak kavramsal olarak kaynak kelimelerinin veri sıkıştırma (data compression) veya veri güvenliği (cryptography) gibi alanlarda kullanılması da mümkündür. Örneğin uzun bir kelimeyi, daha kısa bir kelime ile ifade etmenin anlamı, bu kelimenin yerine geçen bir kaynak kod kullanılmasıdır.

Benzer şekilde kanal kodları, gürültülü ortamlarda veri iletişimini güvenli hale getirmek için gereksiz ilave bilgiler içerebilir. Yani kod kelimeleri, yerine kullanıldıkları anlamdan uzun veya kısa olabilmektedirler.

Veri güvenliği açısından da bir kod kelimesi, yerine kullanıldığı kelimeye dönüşü sadece belirli kişiler tarafından yapılabilen şifreli metindir.

## SORU 9: Birbirini Dışlama (Mutually Exclusive)

Birbirini dışlama özelliği, birden fazla işin birbiri ile ilişkisizliğini belirtmek için kullanılan bir terimdir. Örneğin iki [işlem \(process\)](#) veya iki [lifin \(thread\)](#) birbirinden bağımsız çalışmasını, aynı anda bir işlemi yapmamasını istediğimiz zaman birbirini dışlama özelliğini kullanabiliriz. Bazı kaynaklarda, kısaca mutex (mutually exclusive kısaltması) olarak da geçer.

İki adet [birbirine paralel ilerleyen iş için \(concurrent\)](#) aynı anda bir kaynağa erişme veya birbirleri için kritik olan işlemler yapma ihtimali her zaman bulunur. Örneğin bilgisayarda çalışan iki ayrı [lifin \(thread\)](#) JAVA dilinde ekrana aynı anda bir şeyler basmaya çalışması veya paylaşılan bir dosyaya aynı anda yazmaya çalışması, eş zamanlı programlamalarda, sıkça karşılaşılan problemlerdendir. Bu problemin çözümü için, [işlemlerin senkronize edilmesi](#) gerekir ve temel işletim sistemleri teorisinde 4 yöntem önerilir:

- Koşullu Değişkenler (Conditional Variable)
- [Semaforlar \(Semaphores\)](#)
- Kilitler (Locks)
- Monitörler (Monitors)

Yukarıda sayılan bu 4 yöntem, basitçe iki farklı işi senkronize hale getirmeye yarar. Şayet iki ayrı işin birbirine hiçbir şekilde karışmaması isteniyorsa (mutex) bu durumda çeşitli algoritmaların kullanılmasıyla sistemdeki işlerin ayrılması sağlanabilir. Örneğin aşağıda, iki çok kullanılan algoritmaya bağlantı verilmiştir:

- [Dekker Algoritması](#)
- [Peterson Algoritması](#)

### **SORU 10: Overhead (Ek Yük)**

Genel olarak bir işin yapılması için, gereken ek maliyetlere verilen isimdir. Örneğin bir kamyonun, bir yükü taşıması için, kendisini de taşıması gerekir. Kendisini taşımasının maliyeti, bu işlemdeki ek yüküdür (overhead).

Bilgisayar bilimlerinde, çeşitli alanlarda farklı anlamlarla kullanılmaktadır.

Örneğin veri iletişimi (network) konusunda ek yük (overhead) denildiğinde genelde bir veriyi iletmek için kullanılan teşrifatın (protokol) kendi içindeki ilave haberleşmeleri kast edilir. Örneğin TCP/IP protokolünü ele alalım. Veriyi doğrudan iletmek yerine öncelikle 3 yönlü el sıkışma (three way hand shaking) işlemi ile taraflar arasında iletişim kurulur. Ardından her taşınan veri için TCP/IP paketinin başlık ve sonluk bilgileri de taşınır (ki bu bilgilerin içerisinde örneğin paketin nereden gelip nereye gittiği bilgisi bulunur). İşte veri iletmek için ayrılan kanal veya kaynakların bir kısmının, veriyi taşımak yerine protokole özgü ilave bilgileri taşıması genelde veri iletişimi (network) açısından ek yük (overhead) olarak isimlendirilir.

Programlama dilleri açısından çağırma ek yükü (call overhead) ismi verilen kavram, bir fonksiyonun çağırılması sırasında yaşanan ek yüküdür. Bir programlama dilinde, herhangi bir fonksiyon çağırıldığında, bu fonksiyonun çalışması sonucunda döneceği yer bir yığında (stack) tutulur. Fonksiyon çalışıp işi bittikten sonra program akışına bu noktadan devam edilir. Bu şekilde fonksiyonlar birbirini çağırarak devam edebilir. Örneğin A fonksiyonu B'yi, B fonksiyonu C'yi ... şeklinde birbirilerini çağırdıklarında fonksiyonun çalışması için gereken yere ilave olarak fonksiyon bilgisi için ilave bir yer tutulması gerekir. Bu yere ve bu yer üzerine yapılan işlemlere çağırma ek yükü (call overhead) ismi verilir.

İşletim sistemleri açısından zamanlama ek yükü (scheduling overhead) ismi verilen kavram, bir zamanlama algoritmasının, işlemler arasında geçiş yaparken kaybettiği kaynaklardır. Örneğin kesintili zamanlama (preemptive scheduling) kullanılan algoritmalarda bu ek yük miktarı (overhead) artacaktır.

### **SORU 11: JAVA ile Sohbet İstemci/Sunucusu**

1. [Giriş](#)
2. [Sunucu / İstemci Mimarisi \(Client /Server\)](#)
3. [JAVA ile ağ programlama](#)
4. [JAVA dilinde veri iletişimi için akışların \(streams\) kullanımı](#)
5. [JAVA dilinde Temel bir istemci sunucu \(client / server\) kodlaması](#)
6. [Kodların derlenmesi ve çalıştırılması](#)
7. [Java dilinde çok lifli bir istemci / sunucu kodlaması](#)
8. [Kodların derlenmesi ve çalıştırılması](#)

## 1. Giriş

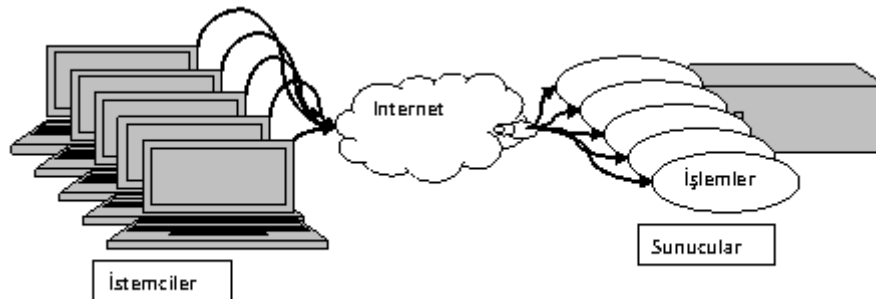
Bu yazının amacı, JAVA programlama dili kullanarak basit bir istemci/sunucu mimarisinin kodlanmasıdır. Sunucu / istemci mimarilerinde birden fazla istemcinin bağlanması durumunda [çok işlemlili \(multiprocess\)](#) veya [çok lifli \(multithreaded\)](#) kodlama uygulanır.

## 2. Sunucu / İstemci Mimarisi (Client /Server)



İstemci / sunucu (client / server) mimarisinde bir bilgisayarın istemci veya sunucu olması oynadığı role göre değişir. Yani bir sunucu bir ağ üzerinde bir hizmet sunan bilgisayardır. Örneğin bir web sitesi yada email sunucusu basitçe bu hizmetleri sunan bilgisayarlardır. Bu anlamda bir bilgisayar duruma göre istemci (client) ve duruma göre de sunucu (server) olabilir.

Birden fazla [istemicinin](#) aynı anda bağlandığı durumlarda, [bir istemicinin](#), [sunucu](#) üzerinde yaptığı işlemlerin, diğer istemcileri engellememesi için (non-blocking); diğer programlamanın ve işletim sistemlerinin bize sunduğu birden fazla işlemin aynı anda çalışmasını sağlayan çok işlemsellikten istifade edilir.



Örneğin basit bir sisteme giriş işlemi sırasında bir kullanıcının ismini ve şifresini girmesi beklenirken, diğer kullanıcıların çalışmasına devam edebilmesi için çok işlemli programlama gerekir. Bu durumda [istemcilerin](#) her birisi için [sunucu tarafında](#) ayrı bir işlem oluşturulmalıdır.

JAVA programlama dilinde ne yazık ki işlem(process) oluşturma imkanı bulunmaz. Bunun sebebi JAVA programlarının üzerinde çalıştığı JAVA Çalışma Ortamının (JRE , Java Runtime Environment) zaten bir işlem olması ve bu işlem üzerinden yeni işlemler çalıştırıldığında kontrolünün mümkün olmamasıdır. Bu duruma çözüm olarak JAVA geliştiricileri, [çok lifli \(multi threaded\)](#) programlamaya izin veren bir ortam geliştirmişlerdir.

Kısaca java ortamında, bizim örneğimizdeki sohbet programı gibi, aynı anda birden fazla işin yapılması istendiğinde [çok lifli \(multithreaded\)](#) programlama kullanılır.



Yukarıdaki temel konuların açıklamasından sonra programlamaya başlayabiliriz. Programlama sırasında öncelikle ağ bağlantılarını kodlayacak, tek lifli bir ağ programını istemci / sunucu mimarisi ile geliştirdikten sonra çok lifli hale getireceğiz. Ağ olarak IP (internet protocol) üzerinde çalışan TCP (transport control protocol) kullanacağız (kısaca TCP/IP protokolü)

### 3. JAVA ile ağ programlama

JAVA dilinde ağ ile ilgili [sınıflar \(Class\)](#) java.net paketinin içerisinde toplanmıştır. Bu paketten ServerSocket ve Socket sınıflarını kullanacağız. Ağ programlama bilgisi olmayan okuyucular için burada belirtmekte yarar gördüğüm bir nokta, ağdaki IP (internet protocol) iletişiminin bilgisayarlarda bulunan portlardan yapılmasıdır. Bir portun bağlanması (binding) sonucu bu port socket (socket) halini alır ve bu socket üzerinden artık veri iletişimi olabilir. Yani bu durumu bilgisayarımızda çok sayıdaki deliğe benzetebiliriz (port). Bu deliklerden birisinin bir boru ile başka bir bilgisayara bağlanması durumunda, artık bu delikten bırakılan veriler diğer bilgisayara ulaşacaktır. Dolayısıyla bu delik artık bir socket olmuştur ve ucu bağlıdır.

```
Socket istemci;  
istemci = new Socket("Bilgisayar", PortNo);
```

Yukarıdaki kod, JAVA dilinde basit bir socket oluşturmaya yarar. Buna göre verilen parametrelerden ilki bilgisayar adıdır. Bu isim bilgisayarın ağ üzerinde bulabileceği bir isim (bilgisayar ismi (host name)) veya DNS üzerinden sorgulayabileceği bir isim (örn. bir web sitesi ismi) olabileceği gibi bir IP adresi de olabilir.

JAVA dilinde bulunan [istisnalar \(exceptions\)](#) gereği yukarıdaki satırı tek başına kullanmak mümkün olmaz. Bunun sebebi bir socket açılması sırasında karşılaşılabilecek giriş çıkış problemleridir (I/O). Bu durumu engellemek için JAVA'nın bize sunduğu imkanlardan try / catch bloklarını kullanmamız gerekir:

```
try{  
    Socket istemci;  
    istemci = new Socket("Bilgisayar", PortNo);  
}catch(IOException e){  
    System.out.println("Soket acilamadi");  
}
```

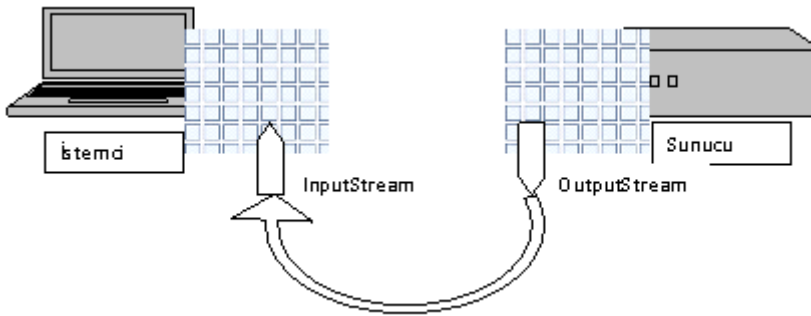
Yukarıdaki kodun yeni halinde, try bloğunda bir problem olması durumunda catch bloğu çalışmakta dolayısıyla hata ekrana basılmaktadır.

Şimdiye kadar öğrendiklerimiz doğrultusunda artık iki bilgisayarın üzerinde bulunan deliklerden (port) birbirine bağlanabildiklerini ve bu bağlantı üzerinden veri akıtılabildiklerini biliyoruz. Tam bu noktada bir problem söz konusudur. Şayet sunucu, [istemcileri \(clients\)](#) beklediği deliği (port) bir istemciye ayırırsa, bu delikten başka istemci bağlanamaz. Yani tek bir istemci gelip bu deliği tıkar ve yeni istemciler için bu delik artık doludur hatası oluşur.

Bu durumu çözmek için [sunucunun](#) konuştuğu ve istemcilerin bağlanmasını beklediği delik boş tutulmalıdır. Bu deliğin tek görevi gelen bağlanma taleplerini cevaplamaktır, sunucunun sunduğu hizmet ise pek ala diğer deliklerden sunulabilir.

Örneğin 800 numaralı portta çalışan bir sunucuyu ele alalım. Sunucu 800 numaralı porttan bir bağlantı gelmesini bekler, bağlantı olunca bu istemciyi hemen farklı bir porta, örneğin 801 alır ve iletişim bu port üzerinden devam eder. Yeni bir istemci bağlanınca da bir sonraki , örneğin 802 alarak bu işleme devam eder. İlk istemcinin işi bittikten sonra artık yeni gelen istemciyi 801 numaralı porta alabilir.

Bir anlamda 800 numaralı portta bekleyen sunucu gelen bağlantı taleplerini farklı portlara yönlendirerek bu portlardan hizmet sunar. Bu işleme kısaca port atlatma denilebilir. Bu işlemi yapmak için oturup baştan kod yazmak yerine JAVA’da bize sunulan nimetlerden birisi olan ServerSocket [sınıfını](#) kullanabiliriz. Bu sınıf bütün bu açıkladığımız işlemleri zaten yapmaktadır. Yani ServerSocket sınıfı verilen port numarasında beklemekte gelen bağlantıları bizim bilmediğimiz , bilmemiz de gerekmeyen portlara yönlendirmektedir.



Yukarıdaki temsili resimde gösterilen her iki bilgisayardaki deliklerin (port) bağlanarak soket haline gelmesidir. Bu resimde anlatılmak istenen, iki bilgisayar arasındaki deliklerin bir boru benzeri [giriş ve çıkış akışlarının \(InputStream / OutputStream\)](#) bağlanmasıdır.

ServerSocket sınıfından oluşan bir [nesnenin](#) kullanımı iki aşamadan oluşur. İlk aşamada nesneyi oluştururuz:

```
ServerSocket Sunucu;  
try {  
    Sunucu = new ServerSocket(PortNo);  
}catch (IOException e) {  
    System.out.println("Sunucu soketi acilamadi");  
}
```

Bu yazı şadi evren şeker tarafından yazılmış ve bilgisayar kavramları.com sitesinde yayınlanmıştır. Bu içeriğin kopyalanması veya farklı bir sitede yayınlanması hırsızlıktır ve telif hakları yasası gereği suçtur.

Yukarıdaki şekilde açılan “Sunucu” isimli nesneyi kullanarak [borularımızı \(streams\)](#) bağlayabileceğimiz bir [nesne oluşturmak](#) için aşağıdaki kod yazılır:

```
Socket baglanti = null;  
try {  
    baglanti = Sunucu.accept();  
}  
catch (IOException e) {  
    System.out.println(e);  
}
```

Yukarıda, ilk aşamada oluşturulan Sunucu [nesnesinin](#) accept metodu çağrılarak bir bağlantı oluşturulmuş ve bu bağlantı bir sokete yönlendirilmiştir. Artık kodun bundan sonraki bölümünde bağlantı nesnesine erişilerek [sunucudaki](#) veri iletişimi kontrol edilebilir.

#### 4. JAVA dilinde veri iletişimi için akışların (streams) kullanımı

Ağdaki iki bilgisayarın birbiri ile konuşması için JAVA programlama dilinde bulunan [akışlar \(streams\)](#) kullanılır. Biz bu uygulamada DataInputStream ve PrintStream [sınıflarını \(class\)](#) kullanacağız. Bu uygulamanın geliştirilmesinde veya benzer ağ uygulamalarında farklı akış sınıfları kullanılabilir. Bu akış sınıflarını birer boruya benzetmek mümkündür. Basitçe iki bilgisayardaki delikler (ports) arasında köprü kurarlar ve birinden bırakılan veri diğerine ulaşır.

JAVA dilinde verilerin akması için kullanılan [boruları \(streams\)](#) ikiye ayırmak mümkündür:

- Giriş akışları (InputStreams)
- Çıkış Akışları (OutputStreams)

Öncelikle giriş akışlarını tanımlayalım. Bunun için iki ayrı tanım yapmamız gerekecek. İstemci tarafında “istemci” isimli soket nesnesinde bağlanan kodu aşağıdaki şekilde yazabiliriz:

```
InputStream input;
try {
    input = new InputStream(istemci.getInputStream());
}
catch (IOException e) {
    System.out.println();
}
```

Öte yandan [sunucu tarafında](#) “baglanti” isimli nesneden bir giriş akışı oluşturulacaktır:

```
InputStream input;
try {
    input = new InputStream(baglanti.getInputStream());
}
catch (IOException e) {
    System.out.println(e);
}
```

Yukarıdaki kodların yazılması sonucu hem [sunucu](#) hem de [istemci tarafında](#) soketlerimizin arkasına boruları bağlamış oluyoruz. Artık bu input [nesnesinden](#) herhangi bir değer okunduğunda, istemci için sunucudan, sunucu için ise istemciden bir veri okunmuş olur.

Yukarıdaki üçüncü şekilde bulunan boruların bağlanması hatırlanırsa, giriş borularına benzer şekilde (ve karşılık gelecek şekilde) çıkış borularının da bağlanması gerekir. Bu bağlantı için aşağıdaki kodlar kullanılabilir:

```
PrintStream output;
try {
    output = new PrintStream(istemci.getOutputStream());
}
catch (IOException e) {
    System.out.println(e);
}
```

```
}
```

Yukarıdaki bağlama işlemi `PrintStream` [sınıfı](#) ile yapılmıştır. Bu sınıf aslında bir `OutputStream` tipidir. Uygulamamızda ve JAVA ağ iletişimi sırasında en çok yollanan veri tipinin `String` (dizgi) olduğunu kabul edebiliriz. Yukarıdaki `printstream` sınıfı bize java'nın en temel ekrana bilgi yazdıran `System.out.println` benzeri bir yazım imkanı sunar. Bu sayede yollamak istediğimiz verileri `println` fonksiyonunu çağırarak basitçe karşıya gönderebiliriz.

JAVA'da diğer bir alternatif ise yine bir `OutputStream` tipi olan `DataOutputStream` sınıfıdır. Bu sınıfta ise `PrintStream`'den farklı olarak `String` yerine `byte` tipi verileri yollamamıza izin verir.

```
DataOutputStream output;  
try {  
    output = new DataOutputStream(istemci.getOutputStream());  
}  
catch (IOException e) {  
    System.out.println(e);  
}
```

## 5. JAVA dilinde Temel bir istemci sunucu (client / server) kodlaması

Yukarıdaki temel kodlamaları öğrendikten sonra çok lifli (multithreaded) programlamayı yine bir kenara bırakarak basit bir istemci / sunucu iletişimini kodlamaya çalışalım. Öncelikle çok tanıdık olan bir yankı sunucusu (echo server) yazalım. Literatürde çok geçen bu sunucu, kendisine gelen mesajları aynen [istemciye](#) geri yollar. Dolayısıyla istemcinin [sunucuya](#) yolladığı mesajlar bir anlamda yankılanmış olur. Basit bir şekilde teknolojiyi anlamak için oldukça elverişli bir örnek olan bu kodun istemci tarafı aşağıdaki şekilde kodlanabilir:

```

1  import java.io.*;
2  import java.net.*;
3  //www.bilgisayarkavramlari.com , Şadi Evren ŞEKER
4  public class istemci {
5      public static void main(String[] args) {
6          Socket istemciSoket = null;
7          DataInputStream is = null;
8          PrintStream os = null;
9          DataInputStream inputLine =null;
10         try {
11             istemciSoket = new Socket("127.0.0.1", 1234);
12             os = new PrintStream(istemciSoket.getOutputStream());
13             is = new DataInputStream(istemciSoket.getInputStream());
14             inputLine = new DataInputStream(new BufferedInputStream(System.in));
15         } catch (UnknownHostException e) {
16             System.err.println("Sunucu bulamama hatasi");
17         } catch (IOException e) {
18             System.err.println("Sunucu baglanma hatasi");
19         }
20         if (istemciSoket != null && os != null && is != null) {
21             try {
22                 String satir;
23                 os.println(inputLine.readLine());
24                 while ((satir = is.readLine()) != null) {
25                     System.out.println(satir);
26                     if (satir.indexOf("ACK") != -1) {
27                         break;
28                     }
29                 }
30                 os.println(inputLine.readLine());
31             }
32             os.close();
33             is.close();
34             istemciSoket.close();
35         } catch (UnknownHostException e) {
36             System.err.println("Sunucu bulamama hatasi");
37         } catch (IOException e) {
38             System.err.println("Sunucu baglanma hatasi");
39         }
40     }
41 }

```

Yukarıda, kodumuzun [istemci tarafı](#) bulunuyor. Kodu inceleyecek olursak, ilk iki satırında ağ ve giriş çıkış işlemleri için gereken importlar yapılmıştır. Ardından 6 ile 19. Satırlar arasında kullanacağımız ağ bağlantısı ve bu bağlantıdaki akışlar (streams, borular) tanımlanarak birbirine bağlanmıştır. Buradaki bağlantıya dikkat edilecek olursa istemciSoket oluşturulmuş ve bu soket üzerine hem PrintStream hem de DataInputStream bağlanmıştır.

Sunucu olarak verilen 127.0.0.1 IP adresi localhost veya loopback olarak geçen bilgisayarın kendisidir. Yani bağlanmaya çalıştığımız sunucunun aynı bilgisayar olduğunu kabul ediyoruz. Şayet farklı bir bilgisayara bağlanılmak isteniyorsa buradaki IP veya bilgisayar ismi değiştirilecektir.

```

1  import java.io.*;
2  import java.net.*;
3  //www.bilgisayarkavramlari.com , Şadi Evren ŞEKER
4  public class sunucu {
5      public static void main(String args[]) {
6          ServerSocket yankisunucu = null;
7          String line;
8          DataInputStream is;
9          PrintStream os;
10         Socket istemciSoket = null;
11         try {
12             yankisunucu = new ServerSocket(1234);
13         }
14         catch (IOException e) {
15             System.out.println("sunucu acilma hatasi");
16         }
17         try {
18             istemciSoket = yankisunucu.accept();
19             is = new DataInputStream(istemciSoket.getInputStream());
20             os = new PrintStream(istemciSoket.getOutputStream());
21             while (true) {
22                 line = is.readLine();
23                 os.println("Sunucudan gelen: " + line);
24             }
25         }
26         catch (IOException e) {
27             System.out.println(e);
28         }
29     }
30 }

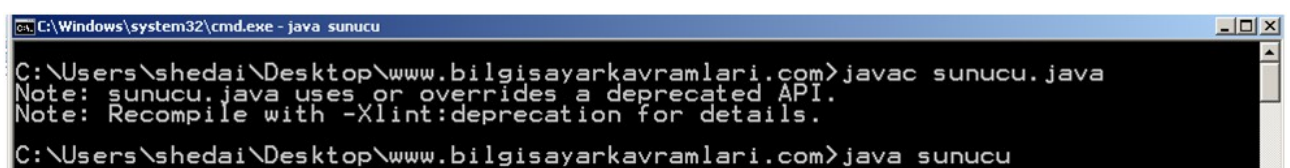
```

Yukarıdaki kod, bir önceki istemciye benzer şekilde 1234 numaralı portta bir soket oluşturmakta ve buradan gelen bağlantıları dinlemektedir. Sunucu, kodun 23. Satırında bulunan `os.println` fonksiyonu ile gelen mesajı geri yollamakta ve bu işi 21. Satırda açılan bir sonsuz [döngü \(loop\)](#) içerisinde yapmaktadır.

## 6. Kodların derlenmesi ve çalıştırılması

Kodların [derlenmesi \(compile\)](#) ve çalıştırılması aşağıdaki şekildedir:

Sunucu tarafı için:



```

C:\Windows\system32\cmd.exe - java sunucu
C:\Users\shedai\Desktop\www.bilgisayarkavramlari.com>javac sunucu.java
Note: sunucu.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
C:\Users\shedai\Desktop\www.bilgisayarkavramlari.com>java sunucu

```

İstemci tarafı için:

```
C:\Windows\system32\cmd.exe - java istemci
C:\Users\shedai\Desktop\www.bilgisayarkavramlari.com>javac istemci.java
Note: istemci.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
C:\Users\shedai\Desktop\www.bilgisayarkavramlari.com>java istemci
```

Unutulmaması gereken bir husus, sunucunun istemciden önce çalıştırılmasıdır. Aksi halde aşağıdaki şekilde bir hata alınabilir:

```
C:\Windows\system32\cmd.exe
C:\Users\shedai\Desktop\www.bilgisayarkavramlari.com>java istemci
Sunucu baglanma hatasi
C:\Users\shedai\Desktop\www.bilgisayarkavramlari.com>
```

Bunun sebebi kodumuzda bulunan [istisnanın \(exception\)](#) tetiklenmiş olmasıdır. Çünkü bağlanılmak istenen bilgisayardaki ilgili portun arkasında soket bağlanmamıştır (bind).

Kodu test ederken, [istemci tarafından](#) bir mesajın yazılması yeterlidir. Bu mesaj sunucu tarafına ağ üzerinden iletilecek ve cevabı yine ağ üzerinden dönerek [istemci tarafında](#) ekrana yazılacaktır:

```
C:\Users\shedai\Desktop\www.bilgisayarkavramlari.com>java istemci
deneme mesaji
Sunucudan gelen: deneme mesaji
www.bilgisayarkavramlari.com
Sunucudan gelen: www.bilgisayarkavramlari.com
```

## 7. Java dilinde çok lifli bir istemci / sunucu kodlaması

Yukarıda basit bir ağ bağlantısının nasıl kodlandığını gördükten sonra esas amacımız olan sohbet programımızı yazmaya başlayabiliriz. Kodumuz, şimdiye kadar yazdığımız koda çok benzeyecek, sadece çok lifli (multithreaded) olmasını sağlayacağız. Dolayısıyla önce kodu verip sonra açıklamasını yapalım:



```

1  import java.io.*;
2  import java.net.*;
3  //www.bilgisayarkavramlari.com , Şadi Evren ŞEKER
4  public class istemci implements Runnable{
5      static Socket clientSocket = null;
6      static PrintStream os = null;
7      static DataInputStream is = null;
8      static BufferedReader inputLine = null;
9      static boolean closed = false;
10     public static void main(String[] args) {
11         int port_number=1234;
12         String host="127.0.0.1";
13         try {
14             clientSocket = new Socket(host, port_number);
15             inputLine = new BufferedReader(new InputStreamReader(System.in));
16             os = new PrintStream(clientSocket.getOutputStream());
17             is = new DataInputStream(clientSocket.getInputStream());
18         } catch (UnknownHostException e) {
19             System.err.println("Sunucu bulma hatasi");
20         } catch (IOException e) {
21             System.err.println("Sunucu baglanti hatasi");
22         }
23         if (clientSocket != null && os != null && is != null) {
24             try {
25                 new Thread(new istemci()).start();
26                 while (!closed) {
27                     os.println(inputLine.readLine());
28                 }
29                 os.close();
30                 is.close();
31                 clientSocket.close();
32             } catch (IOException e) {
33                 System.err.println("iletisim hatasi");
34             }
35         }
36     }
37     public void run() {
38         String responseLine;
39         try{
40             while ((responseLine = is.readLine()) != null) {
41                 System.out.println(responseLine);
42                 if (responseLine.indexOf("*** Bye") != -1) break;
43             }
44             closed=true;
45         } catch (IOException e) {
46             System.err.println("iletisim hatasi");
47         }
48     }
49 }

```

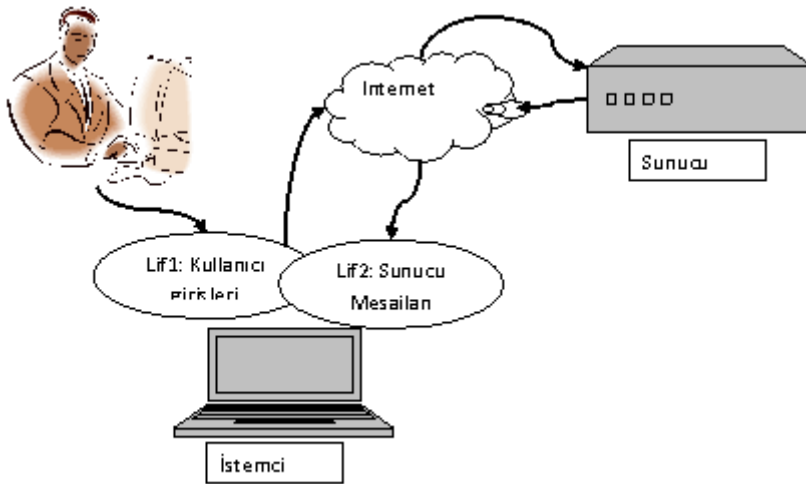
Yukarıdaki kodun büyük kısmı bir önceki uygulamamızla aynı olmakla birlikte, kodda yeni olan lif (thread) eklentileri bulunmaktadır. Öncelikle kodun bir lif (Thread) olarak çalışmasını sağlayan Runnable [arayüzünü \(interface\)](#) implement etmesi gerekmektedir. (kodun 4. Satırında)

Ayrıca bu [ara yüzün \(interface\)](#) gereği olarak sınıfımızda (class) bir run() metodu bulunması gerekir. Bu metod, kodun 37. Satırından itibaren kodlanmıştır. Bu metodun özelliği, lif çalıştığında bu fonksiyonun çalışmasıdır.

Kodun yeni olan bir diğer özelliği 25. Satırdaki yeni lif oluşturma satırıdır. Bu satırda new Thread() yazılarak yeni bir lif (Thread) oluşturulur ve içerisine sınıfımız (istemci sınıfı) olduğu gibi verilir.

Burada lif oluşturulup içerisine bir sınıfı verebilmek için sınıfın Runnable özelliğine haiz olması gerekir. Yani her sınıfı, bir lif oluşturup içine koymak mümkün değildir. Bunun için sınıfta bir run() metodu bulunmalıdır ve bu metodun bulunduğundan emin olmak için de Runnable ara yüzünü implement etmelidir.

Kodun 25. Satırından sonra artık bilgisayarımızda iki ayrı lif (thread) varlığından bahsedebiliriz. Birincisi [sunucudan](#) gelen mesajları ekrana basarken diğeri kullanıcının ekranda yazdıklarını (klavyeden yazıklarını) sunucuya yollamaktadır.



Yukarıdaki temsili şekilde de gösterildiği üzere birinci lif yani kodun kendisi, kullanıcı girişlerini okuyup sunucuya gönderirken (kodun 27. Satırı) ikinci lif ise, sunucudan gelen mesajları ekrana basmaktadır (kodun 40. Satırı) bu sayede bu iki iş birbirini engellememektedir.

[Sunucu tarafında](#) ise iki ayrı sınıf kullanacağız. Sınıflardan birisi sunucunun asıl çalışan ve bağlantıları toplayan lifi (thread) olacaktır. Diğer sınıf ise sunucunun gelen her bağlantıda oluşturduğu ve sadece ilgili istemciye cevap vermesi için kullandığı liften (thread) oluşacaktır. Önce ana sunucu sınıfımızın kodunu inceleyelim:

```

1  import java.io.*;
2  import java.net.*;
3
4  public class anaSunucu{
5
6      static Socket istemciSocket = null;
7      static ServerSocket sunucuSocket = null;
8      static clientThread t[] = new clientThread[10];
9      public static void main(String args[]) {
10         int port_number=1234;
11         try
12         {
13             sunucuSocket = new ServerSocket(port_number);
14         }
15         catch (IOException e)
16         {
17             System.out.println(e);
18         }
19         while(true){
20             try {
21                 istemciSocket = sunucuSocket.accept();
22                 for(int i=0; i<=9; i++){
23                     if(t[i]==null)
24                     {
25                         (t[i] = new istemciLif(istemciSocket,t)).start();
26                         break;
27                     }
28                 }
29             }
30             catch (IOException e) {
31                 System.out.println(e);
32             }
33         }
34     }

```

Yukarıdaki kod, daha önceden kullandığımız bağlantı ve [borulama](#) yaklaşımları ile 1234 numaralı porttan bir bağlantı olmasını bekler. Bu bekleme işlemi kodun 21. Satırında olmaktadır ve her bağlantı alınmasında, yeni bir istemciLif [nesnesi](#) oluşturulur.

Burada dikkat edilecek bir nokta toplam 10 adet liften oluşan bir [dizinin \(Array\)](#) varlığıdır. Kodun 8. Satırında tanımlanan ve 25. Satırında içerisine birer istemciLif nesnesi atanan bu dizide, bağlanan her [istemci](#) için bir nesne referansı (object referrer, [gösterici \(pointer\)](#) ) tutulur. Bu nesne göstericileri daha sonra istemciLifini incelerken de göreceğimiz üzere istemciler arası haberleşme sırasında kullanılır.

Diğer bir deyişle [sunucuya](#) bir istemci bağlandıktan sonra, bu istemci için oluşturulan lif (thread) artık sohbet ederken diğer liflerle haberleşir ve bizim ağ üzerindeki sohbet programımız aslında lifler arası sohbete dönüşür. Elbette her lif kendisine bağlanan istemciye, mesajları anında iletmekte ve dolayısıyla farklı ağ konumundaki kullanıcılar da bu mesajları almaktadır.

```

1  import java.io.*;
2  import java.net.*;
3  class istemciLif extends Thread{
4      DataInputStream is = null;
5      PrintStream os = null;
6      Socket clientSocket = null;
7      istemciLif t[];
8  public istemciLif(Socket clientSocket, istemciLif[] t){
9      this.clientSocket=clientSocket;
10     this.t=t;
11 }
12 public void run() {
13     String line;
14     String isim;
15     try{
16         is = new DataInputStream(clientSocket.getInputStream());
17         os = new PrintStream(clientSocket.getOutputStream());
18         os.println("İsminiz:");
19         isim = is.readLine();
20         os.println("Sohbetten cikmak icin /cik yaziniz");
21         for(int i=0; i<=9; i++)
22             if (t[i]!=null && t[i]!=this)
23                 t[i].os.println("Uyari:"+isim+" isimli kullanıcı sohbete katildi" );
24         while (true) {
25             line = is.readLine();
26             if(line.startsWith("/cik")) break;
27             for(int i=0; i<=9; i++)
28                 if (t[i]!=null) t[i].os.println("<"+isim+> "+line);
29         }
30         for(int i=0; i<=9; i++)
31             if (t[i]!=null && t[i]!=this)
32                 t[i].os.println("Uyari:"+isim+" isimli kullanıcı sohbetten ayrildi" );
33         for(int i=0; i<=9; i++)
34             if (t[i]==this) t[i]=null;
35         is.close();
36         os.close();
37         clientSocket.close();
38     }
39     catch(IOException e){};
40 }
41 }

```

Yukarıdaki kodda, [yapıcı metodun \(constructor\)](#) içerisinde, anaSunucudan gelen lif bilgileri ve soket bilgisi, bu [sınıf](#)a geçirilmektedir. Yani yeni bir [nesne \(object\)](#) üretilirken, bu nesne anaSunucuda kurulmuş olan bağlantıyı almakta ve bu bağlantı üzerinden çalışmasına devam etmektedir. Ayrıca anaSunucuda bulunan diğer liflerin (threads) bilgisi de bu life geçmekte ve bu sayede lifler arasında mesajlaşma ile diğer liflerden gelen mesajları alabilmekte veya yeni bir mesajı diğer liflere iletebilmektedir.

Kodun 18. Satırında, istemciden yeni bir isim istenmekte 19. Satırda bu isim okunmakta, 21-23. Satırlarda ise bu bağlantı diğer liflere haber verilmektedir.

Bilindiği üzere anaSunucuda 10 elemandan oluşan bir lif dizimiz bulunuyor, bu liflerin tamamı o anda çalışmayabilir. Çalışanlara ise bağlanan kişiyi haber vermek üzere 21. Satırda basit bir for [döngüsü \(loop\)](#) kurulmuş, 22. Satırda ise o lifin varlığı sınanarak şayet böyle bir lif varsa 23. Satırda o life mesaj geçirilmiştir.

Bu satırdan sonra yeni gelen kişinin bağlandığı o anda çalışan herkese duyurulur. Mesajlaşmasının da bundan pek bir farkı yoktur. 24. Satırdaki sonsuz döngü içerisinde 25. Satırda ağdaki istemciden okunan mesaj, 27-28. Satırlarda benzer bir döngü ile bütün liflere iletilmektedir.

Yukarıdaki kodda, ayrıca /cik komutu ile kullanıcının bağlantıyı koparması hedeflenmiştir. Bunun için kullanıcı bağlanır bağlanmaz, istemcisine bilgilendirme mesajı 20. Satırda yollanmıştır. Ayrıca kullanıcıdan gelen her mesaj 25. Satırda okunduktan hemen sonra 26. Satırda bu mesajın /cik harfleri ile başlayıp başlamadığı kontrol edilmiştir. Şayet başlıyorsa, 24. Satırdaki sonsuz döngü kırılarak 31. Ve 32. Satırlarla kullanıcının sohbetten ayrıldığı diğer liflere bildirilmiştir.

## 8. Kodların derlenmesi ve çalıştırılması

Kodların çalışması ve derlenmesi aşağıda gösterilmiştir:

Sunucu tarafında:

```
C:\Users\shedai\Desktop\www.bilgisayarkavramlari.com>javac anaSunucu.java
C:\Users\shedai\Desktop\www.bilgisayarkavramlari.com>java anaSunucu
```

İstemci tarafında:

```
C:\Users\shedai\Desktop\www.bilgisayarkavramlari.com>java istemci
?sminiz:
sadi
Sohbetten cikmak icin /cik yaziniz
merhaba
<sadi> merhaba
Uyari:evren isimli kullanıcı sohbe te katildi
<evren> merhaba
```

Yukarıdaki kodlarda, Anatol URSU'nun sitesinden faydalanılmıştır. Bu kodlara <http://www.ase.md/~aursu/ClientServerThreads.html> adresinden de erişilebilir.

## SORU 12: peer to peer (uçtan uca iletişim)

Bilgisayar bilimlerinde, özellikle ağ yönetiminde (network) sıkça kullanılan bir terimdir. Buna göre iki uç bilgisyaar herhangi bir sunucu (Server) ihtiyacı olmadan birbiri ile doğrudan iletişim kurar.

Normalde ağ yapılarında sunucu /istemci (client /server) modeli sıklıkla kullanılır. Bu model yönetim açısından tek bir sunucuya müdahale edilme kolaylığı sağlamanın yanında bütün ağ yapısı ile ilgili bilgi edinmeyi de kolaylaştırır. Ancak ağ trafiğinin tek sunucu üzerinde birikmesi ve sunucudaki bir problemin bütün ağa zarar vermesi istemci / sunucu mimarisinin dez avantajıdır.

Bunun yerine uçtan uca (peer to peer, bazı kaynaklarda kısaca P2P olarak da geçer) mimari kullanılabilir. Bu mimaride, ağdaki bilgisayarlar bir sunucuya bağlanmaz. Bunun yerine her bilgisayarda ağdaki diğer bilgisayarların adresleri bulunur. Ağda hizmet almak isteyen bir bilgisayar elindeki listedeki bilgisayarları dolaşır ve istediği hizmeti bularak işini tamamlar. Böylelikle trafik sadece bu iki bilgisayar arasında yaşanır ve sunucu gibi tek bir noktada trafik sorunu olmaz. Ancak tahmin edileceği üzere sistemin yönetimi oldukça zordur ve sistemdeki işlemlerin takip edilmesi kimin ne yaptığıнын izlenmesi neredeyse imkansızdır.

Yukarıdaki bu iki sistemin yani istemci/sunucu mimarisi ve uçtan uca mimarinin dezavantajlarını ortadan kaldırmak için ara bir yol olarak karışık (hibrid (hybrid)) sistemler geliştirilmiştir. Bu sistemlerde genelde ağdaki bilgisayarların ve hizmetlerin bir listesi sunucuda durur, ancak hizmet iki bilgisayar arasında gerçekleşir.

Örneğin hibrid p2p ağlarının sıkça kullanıldığı torrent, kazaa veya emule gibi dosya paylaşım ağlarını ele alalım. Burada kullanıcılar kendi dosyalarını sunucuya kaydedirler. Bir dosya arayan kişi sunucudaki bu kayıtlar üzerinden arar. Daha sonra dosyayı transfer etmek istediğinde sunucudan değil, dosyanın sahibi kişiden doğrudan transfer eder. Dolayısıyla sunucu üzerinde trafiğin yoğunlaştığı dosya transferi olmaz. Sunucular dosya listesini tutarak bağlanan kişilerin işini kolaylaştırır çünkü sunucu olmasaydı bütün ağdaki bütün bilgisayarlara teker teker dosya var mı diye sorulması gerekirdi.

### **SORU 13: Güvenli Ağ Protokolü (Reliable Network Protocol)**

Bilgisayar bilimlerinde ağ güvenliğinde (network security) kullanılan terimlerden birisidir. Buradaki güvenlik kelimesi saldırılara karşı sağlanan güvenlikten daha çok ağdaki problemlere karşı sağlanan güvenlik anlamındadır.

Yazının burasında Güvenlik kelimesine biraz açıklık getirmek istiyorum. İngilizcedeki 5 ayrı kelimeyi Türkçede tek bir güvenlik kelimesi ile karşılamamız anlam karmaşasına yol açıyor yani security, trust, reliability, confidentiality ve safety kelimelerinin tamamı güvenlik değildir. Aslında Türkçe çok zengin bir dil olmasına karşılık zamanla pek çok kelime dilden çıkarılıyor ve Türkçe fakirleştiriliyor. Ben anlam olarak reliability için Türkçeden bir erozyon ile kaydırılmış kelimelerden birisi olan mutmain kelimesini öneriyorum. Mutmain anlam olarak içi rahat, tatmin olmuş, emin anlamlarına geliyor. Bu anlam olarak reliability kelimesine daha yakın çünkü örneğin bu yazıdaki anlamı da ağ üzerindeki paketlerin emin bir şekilde karşı tarafa erişmesi ve protokolün bu anlamda bize sağladığı emniyet ve rahatlaktır. Ayrıca protokol kelimesi de Türkçeye yeni girmiş ve Türkçe gibi binlerce yıllık bir dil açısından en fazla 50 yıllık geçmişi olan bir kelimedir. Bunun yerine de çok daha uzun süreler Türkçede kullanılmış olan teşrifat kelimesi kullanılabilir. Dolayısıyla yazının bundan sonraki kısmında “mutmain ağ teşrifatı” terimini kullanacağım.

Mutmain ağ teşrifatı ile anlatılmak istenen, tanım olarak, iletişim içerisinde bulunan iki tarafın, yani gönderici ve alıcı tarafların, birbirine ağdaki problemlerden beri olarak veriyi iletmeleridir. Ağdaki problemler çok çeşitli olabilir. Örneğin paketin yanlış yönlendirilmesi (routing), ağda [tkanıklık oluşması \(congestion\)](#), ağdaki cihazlarda [sıra gecikmesi \(queue delay\)](#) gibi hatalar gönderilen [paketin alıcıya ulaşmadan kaybolmasına yol açar \(packet loss\)](#). İşte mutmain ağ teşrifatı burada devreye girerek alıcıya paketler tam ve eksiksiz olarak ulaşana kadar çalışır.

Bir ağ teŖrifatının mutmain etmesi yada etmemesi paketlerin karŖı tarafa iletilmesinin [teŖrifat \(protocol\)](#) tarafından garanti edilip edilmemesine baėlıdır. Yani bir [aė teŖrifatı \(network protocol\)](#) Ŗayet aėdaki paketlerin kaybolmasına g z yumuyor veya bunun takibini yapmıyorsa bu durumda bu aė teŖrifatının mutmain etmediėini s yleyebiliriz.

Bu anlamda  rneėin [TCP \(transport control protocol\)](#) paket teslimatını garanti etmesi y n nden mutmain aė teŖrifatıdır denilebilir. Diėer taraftan UDP ise paket kayıplı bir aė teŖrifatıdır ve bu aė teŖrifatının mutmain olmadıėını (unreliable network protocol) s yleyebiliriz.

Genellikle mutmain bir aė teŖrifatında aŖaėıdaki  zellikler bulunur:

- Baėlantı kurulması
- AkıŖ Kontrol 
- Baėlantı kapatılması

Yukarıdaki baėlantı kurulması ve kapatılması sırasında her iki tarafında iletimi bitirdiėini garanti etmek i in [   y nl  el sıkıŖma algoritması \(three way hand shaking algorithm\)](#) en  ok kullanılan algoritmalarındandır. Yine paketlerin doėru sırayla karŖı tarafa ulaŖmasını temin etmek i in kayan pencere algoritması (sliding window algorithm) en meŖhur algoritmalarından birisidir.

#### **SORU 14: VLAN (Sanal Yerel Aė, Virtual Local Area Network)**

Bilgisayar aėlarında baėlayıcı cihazların (aė anahtarı, switch) kullanımının artmasıyla g nl k hayata girmiŖ bir terimdir. Temel olarak herhangi bir boyuttaki baėlantı Ŗekli  zerinde sanal bir yerel aė baėlantısında (local area network) oluŖturmanın ismidir.

Buna g re  rneėin internet  zerinde d nyanın iki ucundaki kiŖi aynı yerel aėdaymıŖ gibi  alıŖabilir. Benzer Ŗekilde bir iŖ yerinde yan yana  alıŖan kiŖiler farklı iki sanal yerel aė baėlantısına dahil olabilirler. Yani sanal yerel aė baėlantısı, aėın  l eėinden ve topografyasından baėımsız olarak sanal yerel aėlar oluŖturma imkanı saėlar.

Herhangi bir kullanıcının sanal yerel aėına girmek i in donanımsal olarak bir switch'e baėlantısı gerekmektedir. Ancak internet paketleri bu cihaza eriŖtikten sonra ilgili yetkilendirme ve y nlendirmeler  er evesinde paketleri sadece kendi yerel aėı olarak ayarlanmıŖ sanal yerel aėda (vlan) dolaŖabilmektedir.

Bu donanımsal baėlantı bazı durumlarda internet baėlantısı  zerinden de taŖınabilir.

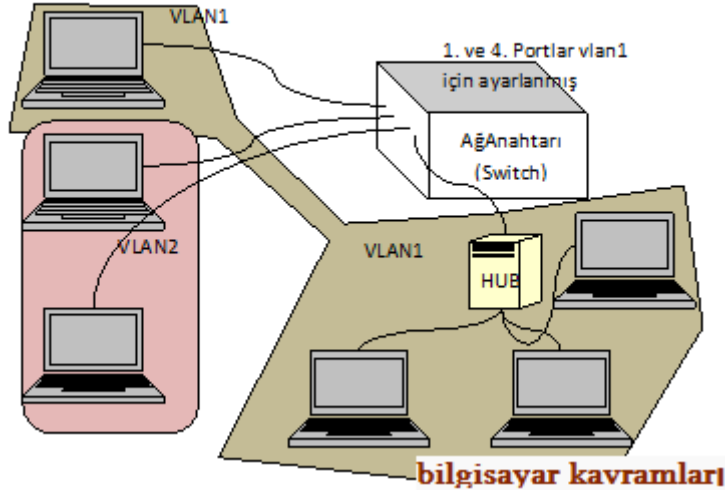
Bu yazı Ŗadi evren Ŗeker tarafından yazılmıŖ ve bilgisayararkavramlari.com sitesinde yayınlanmıŖtır. Bu i eriėin kopyalanması veya farklı bir sitede yayınlanması hırsızlıktır ve telif hakları yasası gereėi su tur.

VLAN teknolojisi genel olarak g venlik amacıyla kullanılır. Buna g re yetki veya kullanım ihtiya larına g re bilgisayarlar  eŖitli sanal aėlara daėıtılır. Bir bilgisayar ancak kendi sanal aėındaki bilgisayarlar ile g venli iletiŖimde olabilir diėer aėlara izni dahilinde eriŖebilir veya hi  eriŖemez.



VLAN teknolojisini sık kullanılan diğer bir şekli de internet üzerinden uzaktaki bir ağa, sanki o ağdaki bir bilgisayar gibi bağlanmasıdır.

Vlan teknolojisini port vlan ile daha kolay anlayabiliriz. Buna göre bir switch üzerindeki portların, switch üzerinde yapılan ayar ile hangi vlan'a ait olduğu atanabilir. Örneğin 10 portlu bir vlan için ilk 5 port 1.vlan, sonraki 5 port is 2.vlan olarak atanabilir.



Örneğin yukarıdaki ağ kurulumunda 4 portlu bir switch olduğunu düşünelim. Şayet 1. Ve 4. Portlar 1.vlan olarak ve 2. Ve 3. Portlar ise 2.vlan olarak ayarlandıysa yukarıdaki şekilde sanal yerel ağlar oluşturulmuş demektir.

### **SORU 15: Çerezler (Cookies)**

İçerik

Çerezlerin	internet	gezininde	ayarlanması
Çerezlerin	HTTP	protokolü	çalışması
PHP	dilinde	çerez	kullanımı
JSP	dilinde	çerez	kullanımı
ASP	dilinde	çerez	kullanımı
Çerezler ve güvenlik			

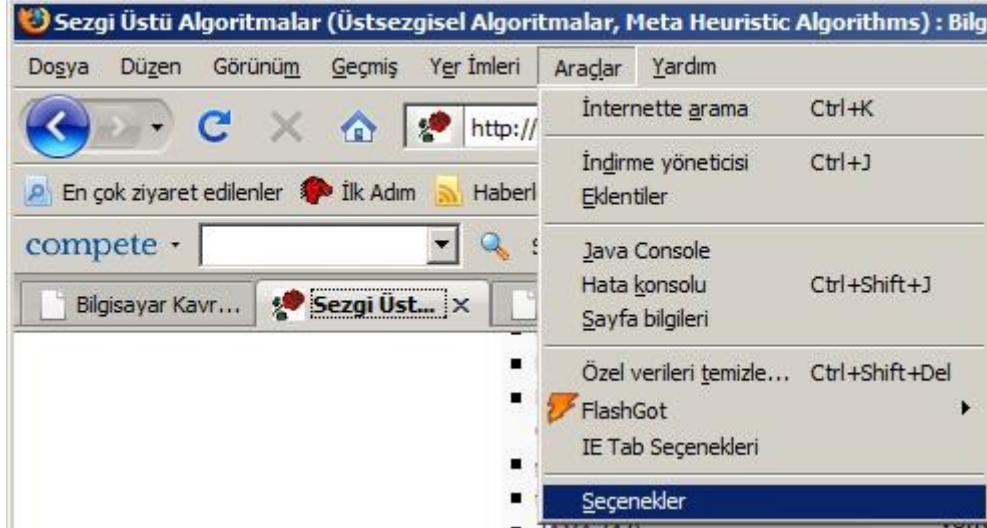
İnternet üzerinde, özellikle de web sayfaları üzerinde gezinirken kullanılan ufak kayıt dosyalarına verilen isimdir. Basitçe bir web sitesi internet üzerinden yayın yaparken bazan bağlanan kullanıcılar hakkında bilgi tutma ihtiyacı duyar. Genelde bu bilgiyi tutmanın iki yolu vardır. Birincisi sunucu üzerindeki bir veri tabanı veya farklı bir veri saklama yapısı içinde tutulması. Diğeri ise istemci (client) üzerinde saklamak.

Bu yazının konusu istemci (client) üzerinde veri saklama teknolojilerinden çerez ismi verilen (cookie) dosyaları anlatmaktır.

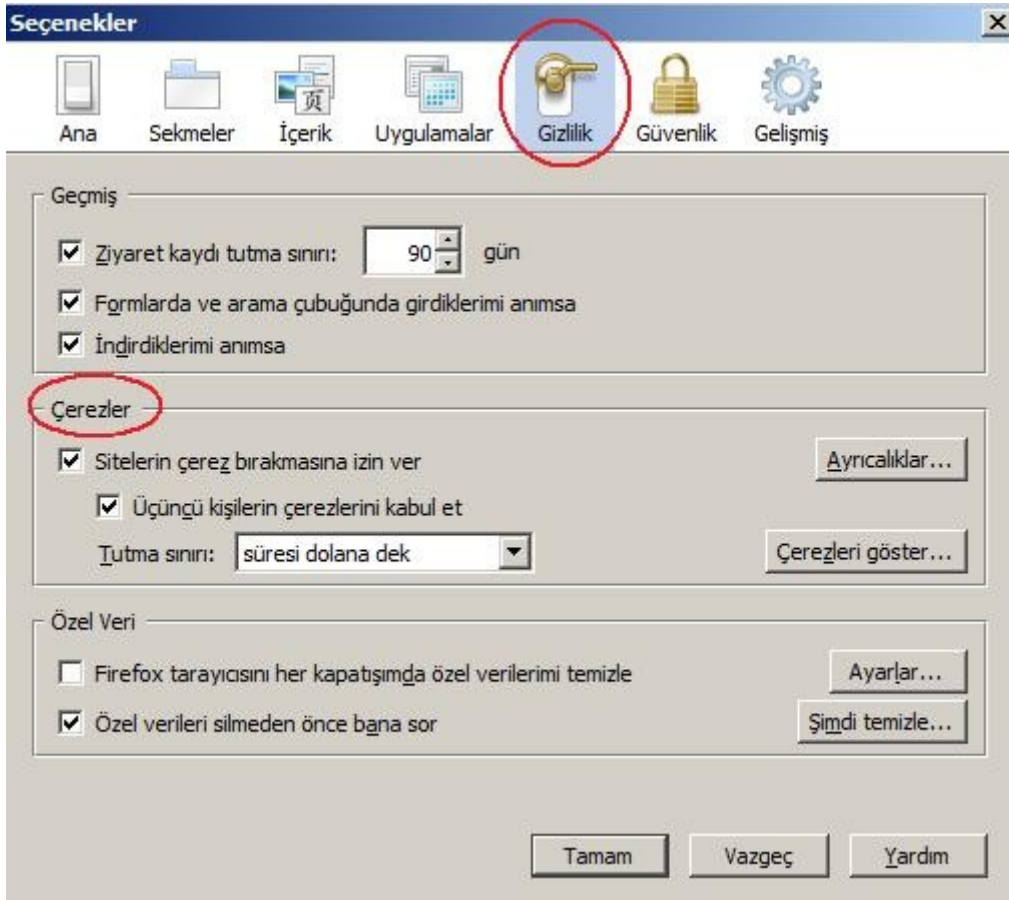
Literatürde, çerezler HTTP protokolü üzerinden taşındıkları için HTTP çerezleri (http cookie) olarak da geçmektedirler.

### **Çerezlerin internet gezininde ayarlanması**

Çerezlerin detayına ve programlamasına geçmeden önce çerezleri kullanıcıların kendi bilgisayarlarında nasıl ayarlayabileceklerini açıklayalım (ekran görüntüleri ve menü yerleri Firefox 3 sürümünden alınmıştır) . Örneğin firefox internet gezginindeki çerez ayarları aşağıdaki şekilde yapılabilir :



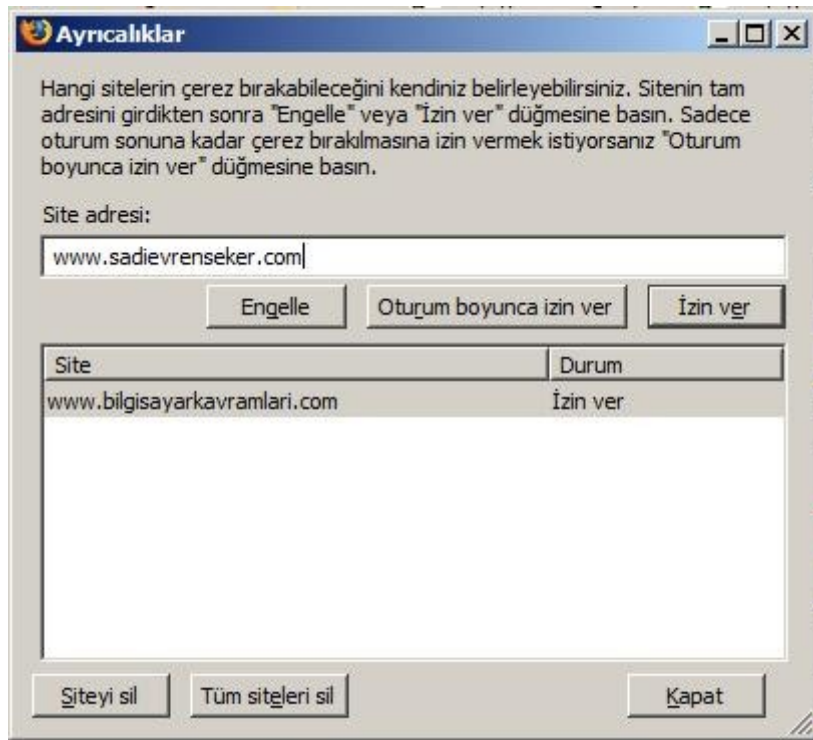
Öncelikle ayarların yapılacağı ekrana Araçlar > Seçenekler tıklanarak girilebilir:



Yukarıda görüldüğü şekilde açılan seçenekler diyalogunda Gizlilik sekmesi altında Çerezler bölümü bulunmaktadır. Bu bölümde istenirse sitelerin çerez bırakmasına izin verilebilir veya bu izin kaldırılabilir.

Temel olarak kullanıcıların böyle bir izni verme veya kaldırma hakkı bulunmaktadır çünkü sonuçta site tarafından kullanıcının bilgisayarına bir dosya kaydedilecektir. Yazının ilerleyen kısımlarında da anlatılacağı üzere bu izin kötü amaçla kullanılabilir ve kullanıcılar için tehdit oluşturabilmektedir. Dolayısıyla kullanıcı dilerse bu seçeneği kapatabilir.

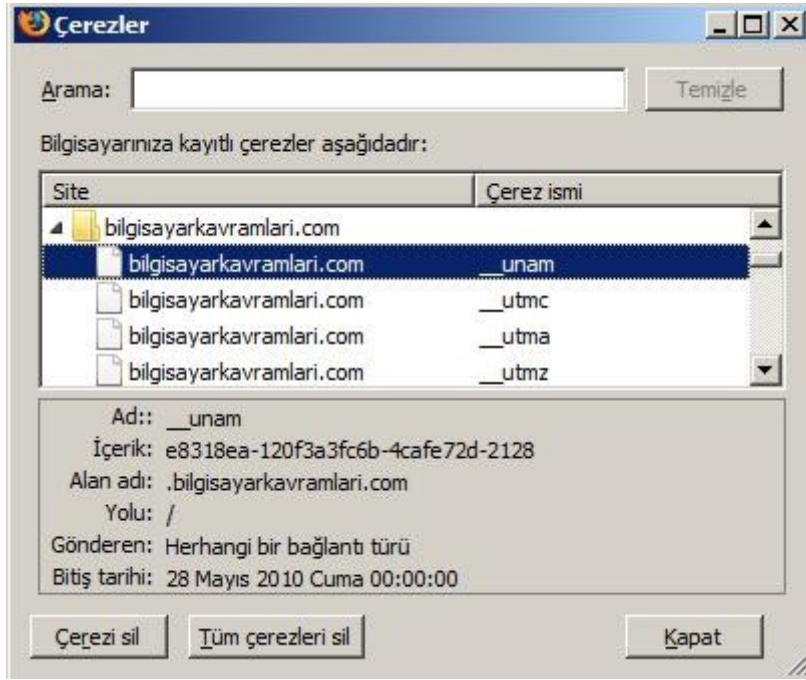
Bu temel özellik bütün internet gezginlerinde ortak olmakla beraber yukarıdaki resimde de görüldüğü üzere Firefox internet gezgininde ilave bazı özellikler bulunmaktadır. Bunlardan birincisi ayrıcalık tanıma özelliğidir. Kullanıcılar isterlerse site bazlı olarak özel ayar yapabilirler ve kuralı sadece belirli sitelere uygulayabilirler.



Örneğin yukarıdaki resimde ayrıcalık diyalogu açıldığında çıkan ekran bulunmaktadır. Burada www.bilgisayarkavramlari.com sitesine izin verilmiştir. Benzer şekilde www.sadievrenseker.com sitesi eklenmek üzere yazılmıştır. Dilenirse Engelle düğmesi ile engellenebilir veya izin ver düğmesi ile izin verilebilir. Ortada bulunan Oturum boyunca izin ver düğmesi ise siteye bir girişlik izin vermek ve siteden çıkıldıktan sonra çerezin temizlenmesi anlamına gelir.

Eklenen siteler istenirse alt tarafta bulunan siteyi sil veya tüm siteleri sil düğmeleri ile silinebilir.

Ayarlar ekranından çerezleri göster düğmesi ile de aşağıdaki ekrana geçilebilir:



Bu ekranda site bazlı olarak çerezlere ve bilgilerine erişmek mümkündür. Bilgisayarımıza bir sitenin bıraktığı çerezi ve detaylarını buradan görebiliriz. Elbette çoğu site çerez içeriği olarak şifreli bilgi tutmaktadır. Bunun sebebini güvenlik kısmında anlayacağız.

Çerezlerin yukarıdaki ekranda da görüldüğü üzere bitiş tarihleri bulunmaktadır. Yani bir çerez istenirse belirli bir süreliğine yollanabilir. Örneğin sitemize giren kişinin alışveriş sırasında sepetine eklediği eşyaların sadece 1 saat boyunca geçerli olmasını bundan sonra tekrar sitemize girerse sepetinin boşalmasını istiyor olalım. Bu durumda çerezi oluştururken bitiş tarihi olarak 1 saat ileri tarihi eklememiz yeterli olacaktır.

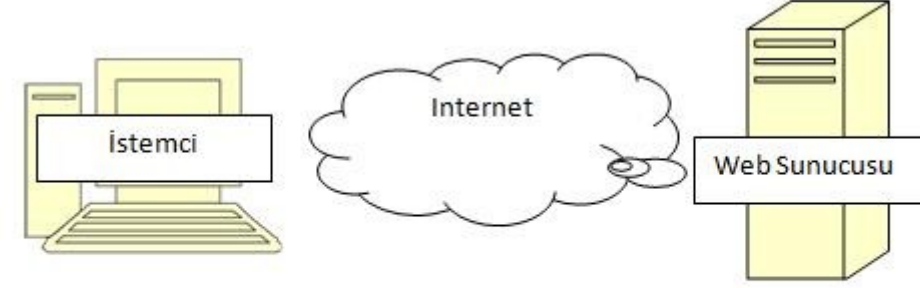
İnternet gezginleri süresi dolan çerezleri saklamaz (veya saklamayabilir) yani bu tip süresi dolmuş çerezlerin silinip silinmemesi internet gezgininin insiyatifindedir ancak yine de süres dolan çerezlerin kullanılamayacağını bilmemiz yeterlidir.

### Çerezlerin çalışması

Bir çerez, normal bir HTTP paketi ile kullanıcı tarafından talep edilir ve sunucu tarafından oluşturularak kullanıcıya yollanır.

HTTP protokolü üzerinden yapılan veri transferi request / response (talep / arz (istek cevap) ) şeklinde olmaktadır. Yani istemci (müşteri , client) bilgisayar sunucu (server) bilgisayarından bir bilgiyi talep eder (request) ve sunucu bilgisayar bu bilgiye cevap olarak bir sonuç arz eder.

Bu iletişim şeklinde arz edilen (sunucudan dönen) bilgi içerisinde bir çerez bilgisi bulunabilir. Bu bilgiyi HTTP protokolü desteklemektedir:



```
GET /index.html HTTP/1.1
Host: www.bilgisayarkavramlari.com
```

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: isim=deger
(Sayfanın istenen içeriği)
```

```
GET /spec.html HTTP/1.1
Host: www.bilgisayarkavramlari.com
Cookie: isim=deger
Accept: */*
```

Örneğin yukarıdaki şekilde adım adım bir HTTP protokolü üzerinden iletişim temsil edilmiştir. Önce istemci tarafı internet üzerinden bir HTTP paketi ile [www.bilgisayarkavramlari.com](http://www.bilgisayarkavramlari.com) sitesinde bulunan index.html dosyasını talep etmektedir.

Ardından sunucu bu talebe yine bir HTTP paketi ile cevap vermektedir. Bu paketin içeriğine dikkat edilecek olursa cevap HTTP 1.1 sürümü ile yapıldığını göstermektedir. Ayrıca 200 OK mesajı dönmüştür. Yani istenen sayfanın bulunduğu ve başarılı bir talep olduğu anlamında bir HTTP kodu ile talep cevaplanmıştır.

Burada HTTP paketinde bizim için önemli olan Set-Cookie bökümüdür. Bu bölümde bir çerez'in istemciye yollandığı ifade edilmektedir ve isim=değer ibaresiyle herhangi bir bilgi istemciye çerez olarak yüklenmi olur.

Buradaki isim=değer bizim belirlediğimiz bir ismin değeridir. Örneğin : “Giriştarihi = 28102009 ” şeklinde bir bilgi olabilir.

Bu şekilde istemci tarafı bilgiyi talep ettikten ve içeriğinde çerez bulunan bir bilgi geldikten sonra bir önceki bölümde açıkladığımız üzere bu bilgi bilgisayarımızda bir dosya olarak saklanır.

Bundan sonraki talepler (farklı sayfa veya aynı sayfanın tekrar talep edilmesi gibi), bu çerez bilgisi de HTTP paketine ilave edilir.

## PHP dilinde çerez programlama

Sunucu tarafı betik dili (server side scripting language) php ile çerez programlamak mümkündür. Bir php sayfasında çerez belirlemek için aşağıdaki satırlar kullanılır:

```
<?php
setcookie("kullanici", "Şadi Evren ŞEKER", time()+3600);
?>
```

Yukarıdaki örnek kodda php sayfasında setcookie fonksiyonu marifetiyle “kullanici” adında bir değişken tanımlanmış ve içeriğine Şadi Evren ŞEKER bilgisi eklenmiştir. Fonksiyonun son parametresi ise çerezin yaşam süresidir. Burada yine php içerisinde tanımlı olan time() fonksiyonu kullanılarak mevcut zaman bilgisi sistemden (sunucunun saatinden) okunmuş ve bu süreye 3600 saniye (tam olarak 60 dakika veya 1 saat) ilave edilerek çerezin geçerli olacağı son an belirlenmiştir.

Yukarıdaki kodda belki dikkat çekmeyebilir ancak setcookie fonksiyonu her zaman için sayfanın en başında bulunmalıdır. Bunun sebebi daha önce de açıkladığımız HTTP paketinin başlık kısmında yer alan bir bilgi olmasıdır. Şayet sayfanın içeriğinde bir yerlerde bu bilgi gönderilirse HTTP protokolünün bu bilgiyi ayırma şansı kalmaz.

Sitemizde yukarıdaki sayfa ile bir çerez üretilerek bu çerezin içerisinde kullanıcı=Şadi Evren ŞEKER bilgisi konulmuştur. Bu bilgiye erişilmek istendiğinde (yine bu sayfadan veya aynı sitedeki herhangi başka bir sayfadan) aşağıdaki şekilde ulaşılabilir:

```
<?php
echo $_COOKIE["kullanici"];
print_r($_COOKIE);
?>
```

Yukarıdaki kodun ilk satırı ile \$\_COOKIE sistem değişkeni (ki bu değişken bir dizidir (array) ) içerisinde bulunan “kullanici” bilgisine erişilmiştir. Bu dizinin tamamının içeriğini görmek için ikinci satırda bulunan ve dizi içeriğini basmaya yarayan print\_r fonksiyonundan yararlanılabilir.

## JSP dilinde çerez programlama

JAVA’nın web tabanlı arayüzü kabul edebileceğimiz java server pages (jsp) ile de yukarıdaki php koduna benzer çerez tanımları yapmak mümkündür.

Örneğin aşağıdaki sayfa kodunu inceleyerek JSP üzerinden nasıl çerez kullanıldığını anlamaya çalışalım :

```
<%@ page language="java" import="java.util.*"%>
<%
User username = new User();
Date now = new Date();
String timestamp = now.toString();
Cookie cookie = new Cookie ("username",username);
cookie.setMaxAge(365 * 24 * 60 * 60);
response.addCookie(cookie);
%>
<html>
<body>
<p><a href="iki.jsp">ikinci sayfa icin buraya basiniz</a><p>
</body>
```



```
</html>
```

Yukarıdaki kodda basit bir web sayfası html dilinde kodlanmıştır. JSP dilinden hatırlanacağı üzere `<% %>` blokları arasındaki kod JSP'ye aittir. Bu alan kolay okunsun diye yukarıda kırmızı renkle belirlenmiştir.

Sayfamızda JSP alanı içerisinde amacımız Cookie sınıfından (class) bir nesne (object) üretmektir. Bu nesnenin özelliği bir dizgi (string) ve bir nesne (object) parametre almasıdır. Yukarıdaki kodda bulunan :

```
Cookie cookie = new Cookie ("username",username);
```

Satırı aslında JSP için çerez kodlamanın yapıldığı satırdır. Bu satıra dikkat ederseniz Cookie yapıcısının (constructor) içerisine birinci parametre olarak bir dizgi (string) verilmiştir. Bu yazı yani "username" ilerde çerezimize erişmek için kullanacağımız bir etiket olarak düşünülebilir. Bu etiketle erişilecek olan bilgi ise Cookie yapıcısının (constructor) ikinci parametresi olan ve daha önceden bir nesne olarak tanımlanmış olan username değişkenidir.

Yukarıdaki örnekte username nesnesi, User ismindeki bir sınıftan türetilmiştir. Siz uygulamanızda cookie olarak saklamak istediğiniz bir nesneyi buraya yerleştirebilirsiniz.

Yukarıdaki şekilde HTTP paketine yerleştirilen bir çereze yine JSP kodunu kullanarak erişmek için aşağıdaki kodlama işinize yarayabilir:

```
<%@ page language="java" %>
<%
    String cookieName = "username";
    Cookie cookies [] = request.getCookies ();
    Cookie myCookie = null;
    if (cookies != null)
    {
        for (int i = 0; i < cookies.length; i++)
        {
            if (cookies [i].getName().equals
(cookieName))
            {
                myCookie = cookies[i];
                break;
            }
        }
    }
%>

<html>
<body>
<%
    if (myCookie == null) {
%>
        <%=cookieName%> isminde bir çerez bulunamadı.

    } else {
%>
        <p>Merhaba: <%=myCookie.getValue() %>.

    }
%>
</body>
```



</html>

Yukarıdaki kodda request.getCookies() fonksiyonu ile, sitemizden erişilebilen bütün çerezler alınmıştır. Ardından bir döngü ile bu çerezler arasında ismi “username” olan çerez aranmıştır. Bulunan bu çerez myCookie isimindeki çerezin içerisine konularak HTML sayfasının içine myCookie.getValue() ile ekrana yazılmıştır.

### ASP dilinde çerezlerin kullanımı

ASP Microsoft tarafından geliştirilen bir sunucu tarafı betik dilidir (server side scripting language). Bu anlamda JSP ve PHP’ye benzemektedir. Aşağıdaki örnek kod ile bir ASP sayfası üzerinden nasıl çerez üretildiğini anlayabiliriz:

```
<%  
Response.Cookies("kullanici")="Şadi Evren ŞEKER"  
Response.Cookies("kullanici").Expires=#May 10,2010#  
%>
```

Yukarıdaki kodda daha önceki dillerde de gördüğümüz üzere çerezin ismi ve değeri atanmıştır. Çerezimize isim olarak kullanıcı ismi verilmiş ve değer olarak ilk satırda içeriğine “Şadi Evren ŞEKER” değeri konulmuştur. Çerezin yaşam süresi ise 10 Mayıs 2010 olarak ikinci satırda atanmıştır.

Aşağıdaki kod ile, yukarıda atanan içeriğe farklı bir sayfadan erişebiliriz:

```
<%  
abc=Request.Cookies("kullanici")  
response.write("Çerez bilgisi=" & abc)  
%>
```

Yukarıdaki kodda, abc isimindeki değişkene öncelikle HTTP paketinden kullanıcı isimli değişken içeriği okunmuştur. Kodun ikinci satırında bu bilgi response.write ile istemciye geri yollanmış ve ekranda görüntülenmiştir.

### Çerezler ve güvenlik

Yukarıda da açıklandığı üzere, çerezlerin siteler tarafından serbestçe erişilebilir olması bazı güvenlik sorunlarını da beraberinde getirmektedir. Aşağıda bu sorunlardan bazıları açıklanmıştır:

İz takibi : Bu güvenlik zaafiyeti birden fazla siteye tek bir elde yerleştirilen çerezlerde olur. Örneğin reklam yayını yapan bir şirket, reklamını yayınladığı yerlerde aynı zamanda çerezini de yayınlatabilir. Bu tip çerezlere, üçüncü parti çerezler (third party cookies) ismi verilir. Bunun sebebi sitenin esas yayıncısı ve siteyi o anda ziyaret etmekte olan istemci (client) dışında üçüncü bir kişinin çerezi olmasıdır.

İşte bu üçüncü parti çerezler siteyi ziyaret eden kişinin bilgisayarına kaydedilir. Şayet reklam veren şirket isterse ziyaretçinin girdiği bütün sitelerin izini sürebilir. Yani reklamının yayımlandığı hangi sitelerin kullanıcı tarafından ziyaret edildiğini takip etmesi mümkündür.

Çerez kaçırlması (cookie hijacking): Bilindiği üzere çerezlerin içerisinde site ve kullanıcı ile ilgili çeşitli bilgiler tutulmaktadır. Saldırgan bir taraf sunucu ve istemci arasında gidip gelen

bu çerezleri takip ederek veya istemcinin bilgisayarında saklanan çerez dosyalarına erişerek çeşitli bilgileri ele geçirebilir. Bu noktada sunucu üzerinden çerez programlayan tarafın oldukça hassas davranması ve kişisel bilgileri, şifre kullanıcı detayı gibi bilgileri çerez üzerinde tutmaması gerekir. Elbette internet gibi bir ortamda bu hassasiyet herkesten beklenemez bu da çerezlerin güvenlik açısından oluşturduğu bir problemdir. Diğer bir çözüm ise çerezlerin içindeki bilgilerin şifreli (encrypted) tutulmasıdır. Elbette bütün şifreler bir gün kırılabilir ancak bu vakit alacaktır ve daha organize bir saldırı gerektirecektir.

Çerez zehirlenmesi (cookie poisoning): Çerez zehirlenmesindeki amaç, istemci taraftan sunucuya giden bilgilerin amaçlı olarak değiştirilmesidir. Örneğin bir alışveriş sitesinde, müşterilerin sepet bilgileri çerezde tutuluyor olsun. Ve büyük bir hata olarak müşterilerin sepetlerindeki eşyaların toplam fiyatı ve dolayısıyla müşterinin ödeyeceği fiyatın da çerezde tutulduğunu düşünelim. Kötü niyetli birisi bu değeri çerez üzerine elle müdahale ederek değiştirebilir ve aslında ödemesi gereken değerden çok daha ucuza sepetindeki eşyaları satın alabilir. Bu tip saldırılara çerez zehirlenmesi ismi verilir.

Oturum saldırısı (session hijacking): Çerezlerin sıkça kullanıldığı yerlerden birisi de oturum bilgilerinin saklanmasıdır. Örneğin kullanıcının en son eriştiği sayfa, kullanıcının site üzerindeki ayarları veya kullanıcıya ait sitede tutulan bilgiler çerezlerde tutularak kullanıcının sonraki bağlantılarında sitede kaldığı yerden devam etmesi amaçlanır. Bu durum bir güvenlik zaaflığı doğurur. Örneğin saldırgan bir kişi bu bilgileri kullanarak aslında hiç olmayan bir kullanıcı ile sitede erişme izni olmayan sayfalara sanki en son bu sayfada kalmış gibi erişebilir. Veya sahip olmadığı yetkiye çerez üzerinde değişiklikler yaparak erişmeye çalışabilir. Bu saldırının tek mantıklı çözümü ise daha dikkatli web siteleri programlamak ve programcıların bu tip saldırılara karşı dikkatli olmasıdır.

## **SORU 16: MIME**

MIME , internet üzerinde kullanılan bir posta (mail) protokolüdür (protocol , teşrifat). Kelime anlamı olarak multi purpose internet mail extensions (mime, çok amaçlı internet posta uzantıları) kelimelerinin baş harflerinden oluşmaktadır.

Yapı olarak açık ve ASCII karakter kodları ile çalışan protokolde istenirse ASCII olmayan posta ekleri, veya mesaj içerikleri de yollanabilir.SMTP ile yakın ilişkisinden dolayı çoğu yerde SMTP/MIME tabiri de kullanılmaktadır.

Örneğin aşağıda bir örnek MIME mesajının içeriği verilmiştir:

```
From: Sadi Evren SEKER <sadi@bilgisayarkavramlari.com>
MIME-Version: 1.0
Content-Type: multipart/mixed;
    boundary="XXXXKESME"
```

Örnek bir mime postasının başlığı.

```
--XXXXKESME
Content-Type: text/plain
```

Postanın içeriği.

```
--XXXXKESME
Content-Type: text/plain;
Content-Disposition: attachment;
```

```
filename="ilave.txt"
```

Posta ile gelecek olan ilave.txt dosyasının icerigi.

--XXXXKESME--

Yukarıdaki mesaj örneğinde görüleceği üzere MIME protokolü ile yollanan bir mesajda çeşitli bölümler bulunmaktadır. Bunlar şöyle sıralanabilir:

Başlık bölümü (header) bu bölümde MIME sürümü (version) ve içerik bilgileri yollanır. Ayrıca mesajın konu kısmı buradadır. Dikkat edilirse başlık kısmında verilen boundary = "" bilgisi mesaj içinde tekrarlanmaktadır. Bu bilgi mesajdaki bölümleri birbirinden ayırmak için kullanılan yazıdır. Tercihen mesajın içerisinde geçmeyecek bir yazı olmalıdır.

Mesajın içeriği bölümü. Bu bölüm, ilk kesmeden sonra başlar ve ikinci kesmeye kadar sürer. Basitçe postamızda yazıların bulunduğu ana bölümdür ve postanın içeriğini belirler.

Ekler bölmü. Bu bölüm 2. kesme ile başlar ve 3. kesmeye kadar devam eder. Burada postanın ekleri yer alır. Birden fazla ek varsa bu kesme sayısı arttırılarak her dosya eki için bir yeni alan açılır. Nihayet son kesme ile MIME protokolündeki posta sona erdirilir.

Bu yazı şadi evren şeker tarafından yazılmış ve bilgisayar kavramları.com sitesinde yayınlanmıştır. Bu içeriğin kopyalanması veya farklı bir sitede yayınlanması hırsızlıktır ve telif hakları yasası gereği suçtur.

## **JAVA ile MIME programlama**

JAVA dilinde MIME protokolü için hazır olarak bulunan java.mail. sınıfı bulunmaktadır. Bu sınıfı kullanarak hızlı bir şekilde e-posta oluşturmak ve yollamak mümkündür.

Bu işlem için öncelikle J2EE içerisinde de yer alan JavaMail paketinin indirilerek kurulması gerekir. Paketin son sürümüne <http://java.sun.com/products/javamail/downloads/index.html> adresinden erişebilirsiniz.

İlgili paket indirildikten ve kurulduktan sonra aşağıdaki kod çalıştırılabilir ve SMTP sunucu üzerinden mail yollanması mümkün hale gelir. Elbette postanın yollanması için bir SMTP sunucunun kurulu olması veya internet üzerindeki herhangi bir SMTP sunucuda hesabımızın olması gerekiyor:

```
import javax.mail.Message;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.URLName;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

import com.sun.mail.smtp.SMTPTransport;

public class MIMEDeneme {
```

```

private Transport tr;
private Message msg;
private Session sn;
public void setMessage(String fromAddress,String toAddress,
String subject,String content) throws Exception {
String[] addresses = {toAddress};
setMessage(fromAddress,addresses,subject,content);
}

public void setMessage(String fromAddress,String[] toAddresses,
String subject,String content) throws Exception {

sn = Session.getInstance(System.getProperties());

msg = new MimeMessage(sn);
msg.setFrom(new InternetAddress(fromAddress));
InternetAddress[] toIntAdds = new InternetAddress[toAddresses.length];

for (int i=0;i<toAddresses.length;i++)
toIntAdds[i] = new InternetAddress(toAddresses[i]);

msg.setRecipients(Message.RecipientType.TO,toIntAdds);
msg.setSubject(subject);
msg.setSentDate(new java.util.Date());
msg.setText(content);
}

public void setSMTPServer(String host,int port,
String user,String password) throws Exception{
tr = new SMTPTransport(sn,new URLName(host));
tr.connect(host,port,null,null);
}

public void send() throws Exception{
msg.saveChanges();
tr.sendMessage(msg,msg.getAllRecipients());
tr.close();
}

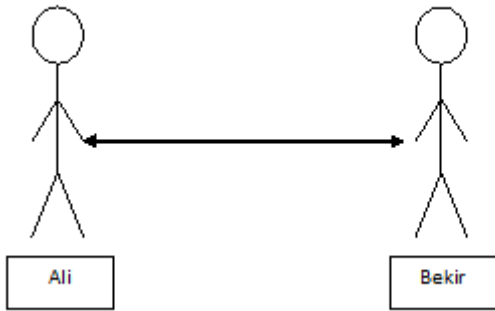
public static void main(String[] args) throws Exception{
MIMEDeneme tms = new MIMEDeneme();
tms.setMessage("sadi@bilgisayarkavramlari.com",
new String[]{"test@bilgisayarkavramlari.com","test@sadievrenseker.com"},"Deneme
Postasi","MIME ile mektup yollama denemesidir" +
"içerik kısmını yazıyoruz");
tms.setSMTPServer("localhost",25,null,null); //smtp sunucunun adresi,port
numarasi,kullanici,şifre
tms.send();
}
}

```

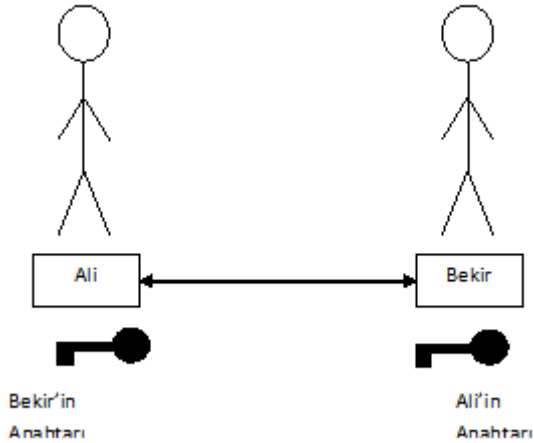
## SORU 17: Ara kilit Protokolü (Interlock Protocol)

Veri güvenliğinde kullanılan bir protokol (teşrifat, protocol) çeşididir. Temel olarak ortadaki kişi (man in middle) saldırılarına karşı geliştirilmiştir. Amacı, şifreli veri iletişimi sırasında, iletişimi izleyerek sabotaj etmek isteyen kötü niyetli kişilere karşı güvenlik sağlamaktır.

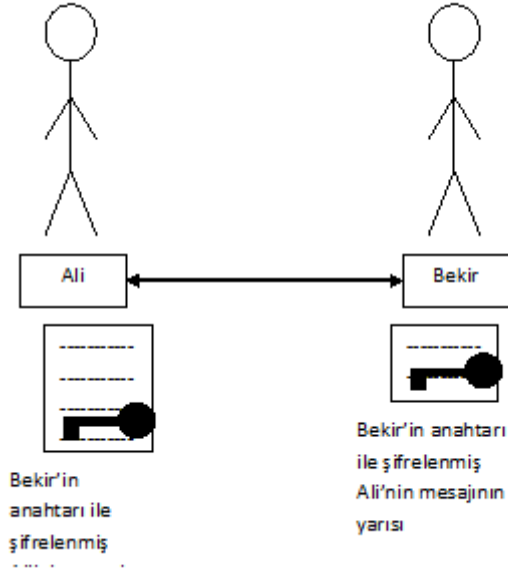
Bilindiği üzere ortadaki kişi saldırısında, mesajlaşmanın üzerinden geçtiği bir kişi, her iki tarafa da sahte anahtarlar yollayarak , açık anahtar şifrelemesi (public key cryptography) kullanılan bir mesajlaşmaya saldırabilmekteydi. Bu saldırının nasıl önlenildiği ve dolayısıyla ara kilit protokolü (interlock protocol) aşağıda açıklanmıştır:



Ali ve Bekir arasında mesajlaşma yapılmak isteniyor olsun. Mesajı açık anahtar şifrelemesi (public key cryptography) ile şifrelemek isteyen taraflar öncelikle umumi anahtarlarını (public key) karşı tarafa geçirirler:



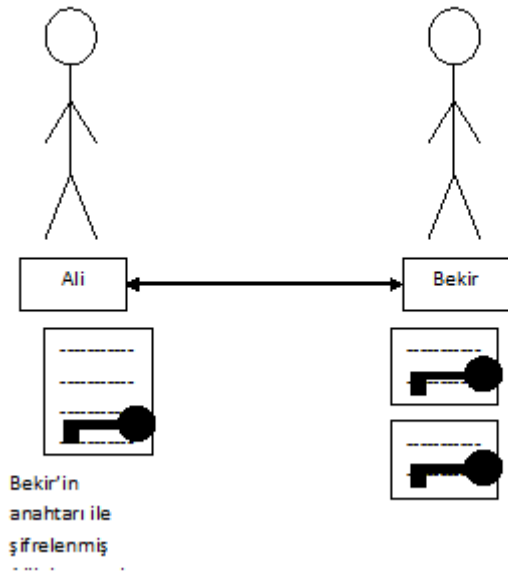
Karşılıklı anahtar geçişinin ardından mesajlaşma başlar. Ancak buradaki mesajlaşma klasik açık anahtar şifrelemesinden biraz farklıdır. Örneğin Bekir'e mesaj iletmek isteyen Ali, mesajını klasik bir şekilde Bekir'in umumi anahtarı ile şifreleyerek yollamaz. Bunun yerine şifrelenmiş mesajın yarısını yollar. Ve bu adımla ara kilit protokolü başlamış olur.



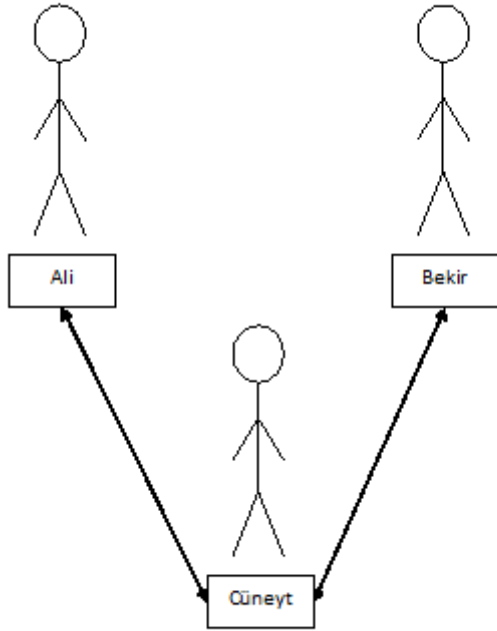
İlk adımda yarısı yollanan mesaj Bekir tarafından alınır. Elbette mesaj yarım olduğu için Bekir'in hususi anahtarı (private key) bu mesajı açmak için kullanılamaz. Mesajın tamamı elde edilmeden de mesaj açılmayacaktır.

Bu yazı şadi evren şeker tarafından yazılmış ve bilgisayar kavramları.com sitesinde yayınlanmıştır. Bu içeriğin kopyalanması veya farklı bir sitede yayınlanması hırsızlıktır ve telif hakları yasası gereği suçtur.

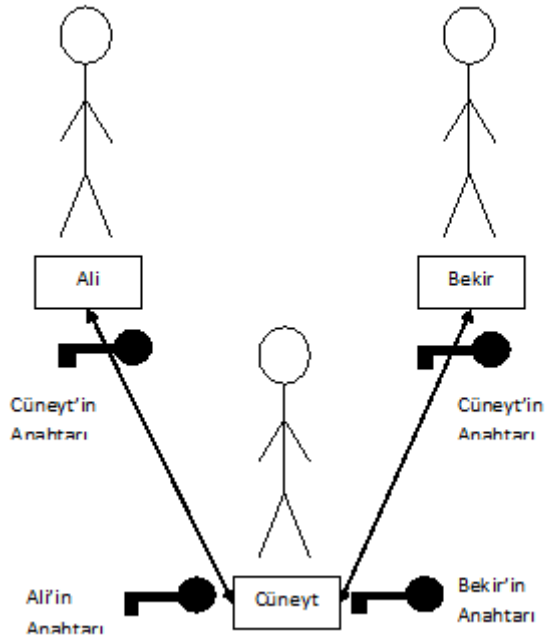
Bu adımda Ali den mesajın ikinci yarısı beklenenecektir. İşte tam bu noktada ara kilit protokolünün kuvveti ortaya çıkmaktadır. Ali'nin ikinci mesajı yollaması ancak ve ancak Bekir'in bir mesaj parçası yollamasından sonra gerçekleşecektir. Ali, Bekir'den bir mesaj aldıktan sonra mesajın diğer yarısını yollayacaktır ve mesajın tamamını elde eden Bekir, mesajı kendi hususi anahtarı ile açacaktır. Benzer şekilde Ali'ye de mesajın ikinci yarısını yollayacak ve Ali de mesajın tamamını açacaktır.



Bu yeni durumda Bekir mesajı açabilecektir. Şayet sistemde örneğin Cüneyt isiminde saldırgan olan bir kişi bulunsaydı ve aradaki mesajlaşmayı dinliyor olsaydı:



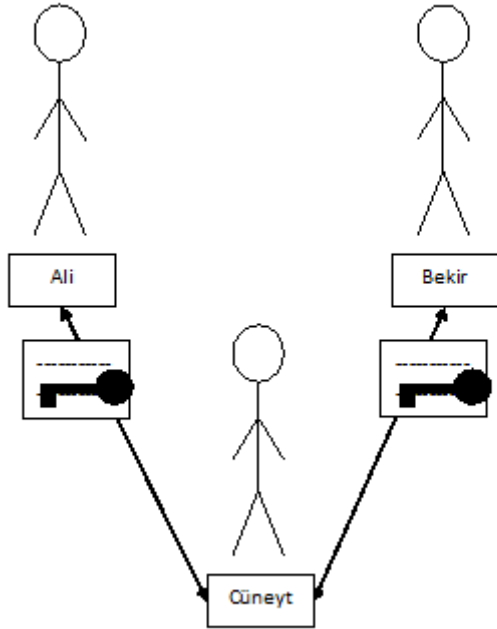
Bu durumda Cüneyt saldırı için taraflara kendi anahtarını sanki karşı tarafın anahtarıymış gibi dağıtacaktır:



Ancak Cüneyt'in anahtarı ile şifrelenen mesajlar, Cüneyt tarafından okunamayacaktı çünkü mesajın tamamına erişme şansı bulunmayacaktı. Buradaki kritik nokta mesajın yarısının



açılmamasıdır. Ayrıca ikinci kritik nokta mesajların karşılıklı olarak yarım yollanmasıdır. Yan örneğin Ali, Bekir'e mesaj yollarken arka arkaya iki yarıyı yollasaydı, Cüneyt başarılı bir şekilde saldırılabilecekti. Ancak Ali, Bekir'den bir mesaj yarısı gelene kadar beklemektedir. Bu durumda iki tarafta da mesajlar yarım halde olacaktır. Bu durumda Cüneyt saldırıyı gerçekleştiremez çünkü yarım mesajları açamaz.



Bu durumda ortadaki adam saldırısı başarısız hale getirilmiş olmaktadır ve açık anahtar şifrelemesi kullanılarak daha güvenli mesajlaşma gerçekleştirilebilir.

Elbette bu noktada kritik olan bir iki durum şudur. Örneğin Ali, Bekir'e bir mesaj yolladığı sırada Bekir de Ali'ye mesaj yollamak istemiyor olabilir. Yani teşrifatımızın (protokol) sağlıklı işlemesi için tarafların karşılıklı mesaj alışverişinde bulunması beklenir. Bu durumda Bekir cevap olarak başarılı bir şekilde iletişimi gerçekleştirdiğini bildiren bir onay (acknowledgement) mesajı geri yollarsa bu mesajın içeriğinin tahmin edilebilmesi ve sistemde yeni zafiyetler oluşturması mümkündür. Çözüm olarak anlamlı rastgele bir mesaj yollanabilir.

Diğer bir durum, mesajlaşmada kullanılan yarım mesaj kavramıdır. Örneğin blok şifreleme (block cipher) için bu durum bir problem oluşturabilir. 100 blokluk bir mesajın 50 bloğunun yollanması bir tehdit oluşturur. Bunun yerine örneğin blok boyutu 10 byte ise her bloğun ilk 5 byte'ını yollamak gerekir ki mesajın bir yarısına saldırı yapan kişi açamasın.

Ayrıca şifreleme sırasında başlangıç yöneyi (initialization vector) kullanılıyorsa bu vektör mesajın ikinci yarısında yollanmalıdır ki ilk yarısına saldırı yapan kişiye bir avantaj sağlamasın. Ayrıca ikinci yarıya saldırı yapan kişinin de işine yaramasın.

## SORU 18: Protokol (Protocol, Teşrifat)

Bilgisayar bilimleri açısından protokoller (teşrifatlar) genelde iki veya daha fazla taraf için yapılması gereken bir dizi işi belirtir.

Özellikle veri iletişimi ve birden çok işin aynı anda yapıldığı [çok işlemlili \(multi-process\)](#) ve [çok izli \(multi threaded\)](#) sistemlerde oldukça sık kullanılan bir terimdir. Bilgisayar bilimlerindeki kullanımı da günlük hayattan pek farklı değildir. Örneğin günlük hayatta iki kişinin selamlaşma protokolünü aşağıdaki diyalog ile tanımlayalım:

A: Merhaba

B: Merhaba

A: Nasılsın?

B: İyiyim ya sen nasılsın?

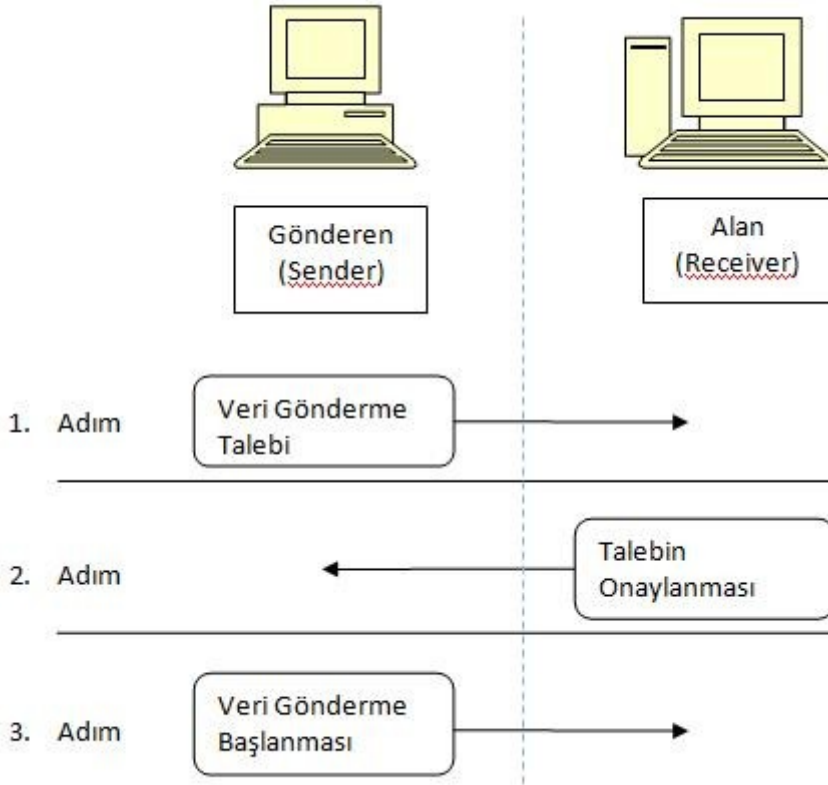
A: Ben de iyiyim.

iki kişi arasında geçen bu diyalog, standart bir hal alıyorsa ve iki kişinin birbirini selamlaması için kullandığı bir yöntemse bu yöntemle selamlama protokolü diyebiliriz.

Elbette bilgisayarlar, güncel hayata göre çok daha [belirli \(Deterministic\)](#) yapılardır. Yani gerçek hayatta iki insan pek çok farklı selamlama protokolü kullanabilir hatta ihtiyaç durumunda daha önceden hiç kullanmadıkları yeni bir protokolü icad edebilir. Ancak bu durum bilgisayarlar için ne yazık ki mümkün değildir.

### Üç kereli el sıkışma protokolü ( Three way handshaking protocol)

Örneğin iki bilgisayarın birbirini selamlaması için kullandıkları üç kereli el sıkışma protokolünü (three way hand shaking protocol) ele alalım. Bu protokolde iki bilgisayar (yada [işlem \(process\)](#) ) birbiri ile iletişime başlamadan önce selamlaşırlar. Bunun için iletişimi yapacak taraf alıcıya bir mesaj yollar. Alıcı mesajı alıp iletişime hazır olduğunu belirten bir mesajı geri yollar ve iletişim başlar.



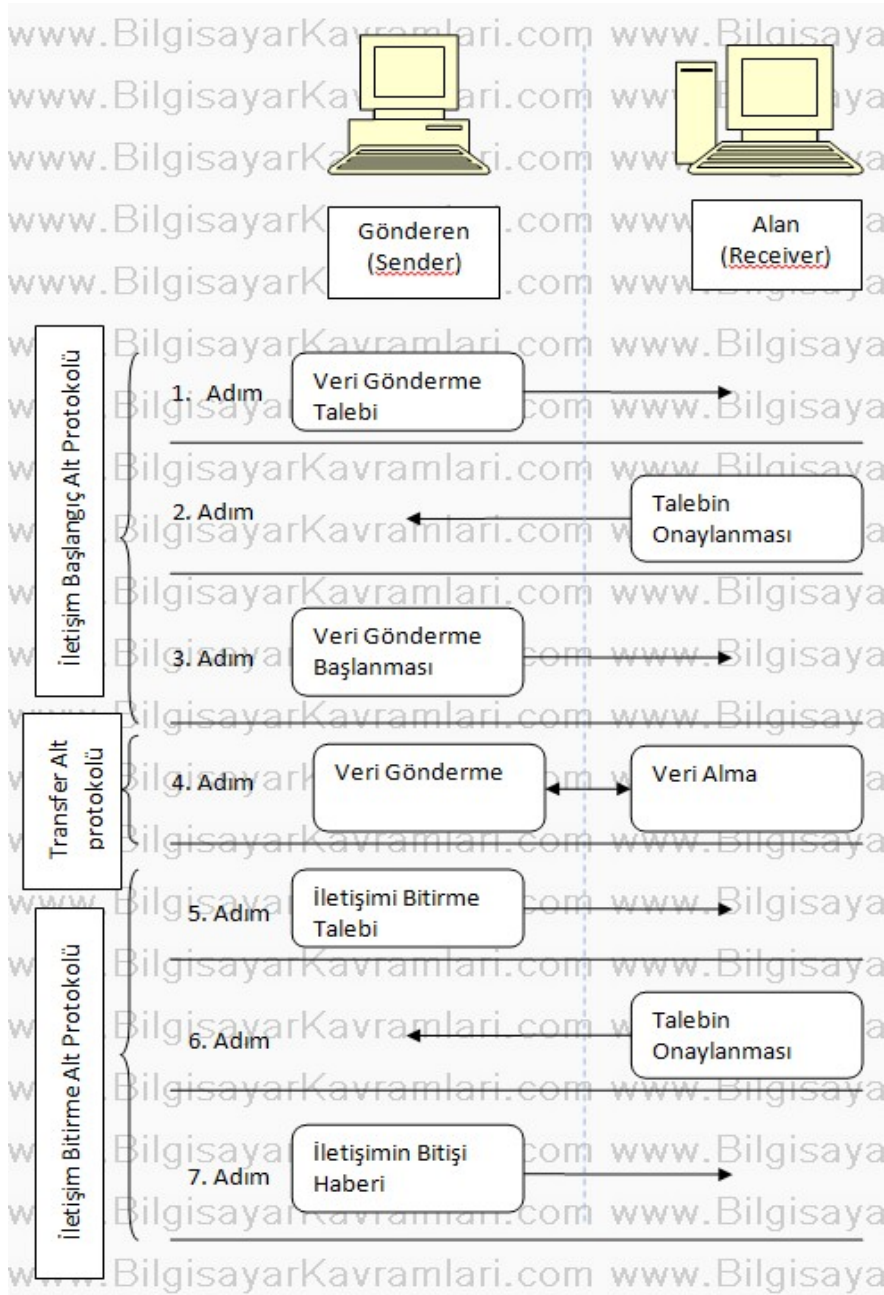
Yukarıdaki şekilde iki bilgisayar arasında yaşanan bu selamlama protokolü gösterilmiştir.

Protokollerin bilgisayar dünyası açısından özel olarak aşağıdaki özellikleri taşıması beklenir :

- Protokolü kullanan tarafların tamamı, protokolü biliyor olmalıdır.
- Protokolü kullanan tarafların tamamı bu protokolü kullanmak istiyor olmalıdır.
- Protokol tam olmalıdır (complete). Yani her durum için tanımlı bir sonuç bulunmalıdır.
- Protokol kesin (unambiguous) olmalıdır. Yani protokoldeki herhangi bir durumun belirsizliği söz konusu olamaz.
- Protokol belirli (deterministic) olmalıdır. Yani her durumdan geçilebilecek bir sonraki durum tanımlanmış olmalıdır.

### Alt protokoller (sub protocols)

Bazan protokollerin alt protokollere bölünmesi mümkün olabilir. Örneğin tanımlı olan protokolümüz, iki bilgisayar arasında veri iletişimi yapmayı amaçlıyor olsun. Bu durumda iki bilgisayarın bir önceki örnekte gösterildiği üzere selamlaşması, ardından veri iletişiminde bulunmaları ve nihayetinde iletişimi kapatmak için tekrar selamlaşmaları birer alt protokol olacaktır.



Yukarıdaki şekilde kabaca bir iletişim protokolünün üç ayrı alt protokolü gösterilmektedir. Yukarıdaki örnekten de görüldüğü ve anlaşılabilceği üzere protokoller alt parçalara bölünürse bu parçalara alt protokol ismi verilir.

### Protokollere saldırılar ( attacks against protocols)

Bir protokolün yapısı itibariyle birden fazla taraf arasında kullanıldığını dolayısıyla tarafların bir şekilde iletişimine yaradığını biliyoruz. Dolayısıyla veri güvenliği açısından oldukça önemli olan protokoller gerek saldırı gerekse korunma yöntemleri açısından bilgisayar bilimlerinin konusu olmuşlardır.

Bu durumda protokollere yapılan saldırılar ile kast edilen, iletişim halinde bulunan iki veya daha fazla tarafın iletişim bilgilerini ele geçirmek, bozmak veya yanıltmak olarak sayılabilir. Bu noktada protokollere yapılan saldırıları aktif ve pasif olarak ikiye ayırmak mümkündür.

Aktif saldırılar (active attacks) : Bu saldırı türünde saldırgan kişi, saldırının olduğunu gizleme veya kendi varlığını gizleme ihtiyacı duymaz. İletişime kendi mesajını taraflardan birisiymiş gibi koyması veya mesajlardan birini silmesi veya mesajlardan birini anlaşılmaz hale getirmesi gibi yöntemler bu saldırı türündendir. Taraflar bu yöntemde saldırganı bulma şansına sahiptir.

Pasif saldırılar (passive attacks): Bu saldırı türünde saldırgan kişi, saldırıyı ve kendi varlığını gizler. Örneğin iletişimden bir bilgi parçasını çalarak bu bilgiyi okumak ve dolayısıyla taraflar hakkında bilgi toplamak bu tip bir saldırı sayılabilir. ( [sadece şifreli mesaj saldırısı \( cipher text attack only\)](#))

### **SORU 19: CSMA (Carrier Sense Multiple Access)**

Bilgisayar bilimlerinin ağ yönetimi çalışmalarının bir konusudur. Basitçe bir ağ ortamına aynı anda birden fazla bilgisayarın erişmesi durumunda ortamın algılanmasını ifade eder.

CSMA, OSI katmanlarından veri bağlantı katmanında (datalink layer ) çalışan MAC (Ortam erişim kontrolü, media access control) katmanının özel bir halidir. Bu protokol aşağıda açıklanacak olan olasılıksal bazı değerleri kullanarak, aynı ortama birden fazla erişimin olmasını sağlamaktadır.

Basitçe bir ortamda tek bir bilgisayarın sinyal yollayabildiği ortamlarda kullanılan ve bilgisayarların sırayla birbirini bekleyerek ya da ortamın boş olup olmadığını kontrol ederek iletişim yapmasını sağlayan protokoldür.

#### **CSMA çeşitleri**

csma protokolleri, ortam erişimindeki ısrarlarına (harun, persistent) göre 3 grupta incelenebilir

Israrcı olmayan CSMA (Non-persistent CSMA) : Bu yaklaşımda ortama erişip veri yollayacak olan bilgisayar ortamı kontrol eder. Şayet ortam boşsa veriyi yollar, şayet ortam boş değilse rastgele bir süre bekler ve tekrar ortamı kontrol eder.

Bu kontrol sırasında başka bir bilgisayar daha ortamı kontrol etmiş ve boş bulmuş olabilir. Bu durumda aynı anda iki bilgisayar da ortamdaki veri yollamaya çalışacaktır. Bu durumda çakışma (collision) oluşacaktır ve çözüm olarak çakışma olması durumunda her iki bilgisayarda kendi ürettikleri rastgele zamanlar kadar bekleyerek ortamı yeniden kontrol ederler.

1-ısrarlı CSMA (1-persistent CSMA): Bu yaklaşımda veriyi yollayacak olan bilgisayar ortamı kontrol eder. Şayet boşsa yollar, doluyorsa sürekli olarak kontrole devam eder. Ortam boş olunca veriyi yollar. Şayet bir çakışma (collision) oluşursa rastgele bir süre beklenerek tekrar yollanır.

p-ısrarlı CSMA (p-persistent CSMA) : Bu yaklaşımda 1-ısrarlı yaklaşıma göre p değerinde bir olasılık çarpanı bulunur. Buna göre gönderecek olan bilgisayar 1-ısrarlı yaklaşımda olduğu gibi ortamı kontrol eder. Şayet ortam doluyorsa boş olana kadar bekler, şayet ortam boşsa 1-ısrarlı CSMA'de olduğu gibi doğrudan veriyi yollamaz bunun yerine p değerinde bir olasılık hesabı yaparak bu olasılıksal oranda yollar. Ardından algoritma baştan ortamı kontrol eder ve bu şekilde devam eder.

Basitçe 0.2 ısrarlı bir CSMA örneğinde her 5 yollama imkanından sadece birisinde yollama olacağını söyleyebiliriz. Ancak bu 5te 1 oranı tekrarlı değildir. Yani her yollamadan önce 5 yüzlü bir zar atıp, yollanıp yollanmayacağına bakılır. Şayet zar yollama yönüdeyse yollanır. İstatistiksel olarak bu ihtimal örneğin 10 denemede de olabilir arka arkaya iki denemede de.

## **SORU 20: Atomluluk (Atomicity)**

Latince bölünemez anlamına gelen atom kökünden üretilen bu kelime, bilgisayar bilimlerinde çeşitli alanlarda bir bilginin veya bir varlığın bölünemediğini ifade eder.

Örneğin programlama dillerinde bir dilin atomic (bölünemez) en küçük üyesi bu anlama gelmektedir. Mesela C dilinde her satır (statement) atomic (bölünemez) bir varlıktır.

Benzer şekilde bir verinin bölünemezliğini ifade etmek için de veri tabanı, veri güvenliği veya veri iletimi konularında kullanılabilir.

Örneğin veri tabanında bir işlemin (transaction) tamamlanmasının bölünemez olması gerekir. Yani basit bir örnekle bir para transferi bir hesabın değerinin artması ve diğer hesabın değerinin azalmasıdır (havale yapılan kaynak hesaptan havale yapılan hedef hesaba doğru paranın yer değiştirmesi) bu sıradaki işlemlerin bölünmeden tamamlanması (atomic olması) gerekir ve bir hesaptan para eksildikten sonra, diğer hesaba para eklenmeden araya başka işlem giremez.

Benzer şekilde işletim sistemi tasarımı, paralel programlama gibi konularda da bir işlemin atomic olması araya başka işlemlerin girmemesi anlamına gelir.

Örneğin sistem tasarımında kullanılan check and set fonksiyonu önce bir değişkeni kontrol edip sonra değerini değiştirmektedir. Bir değişkenin değeri kontrol edildikten sonra içerisine değer atanmadan farklı işlemler araya girerse bu sırada problem yaşanması mümkündür. Pek çok işlemci tasarımında buna benzer fonksiyonlar sunulmaktadır.

Genel olarak bölünemezlik (atomicity) geliştirilen ortamda daha düşük seviyeli kontroller ile sağlanır. Örneğin işletim sistemlerinde kullanılan [semafor'lar \(semaphores\)](#), kilitler (locks), koşullu değişkenler (conditional variables) ve monitörler (monitors) bunlar örnektir ve işletim sisteminde bir işlemin yapılması öncesinde bölünmezlik sağlayabilirler.

Kullanılan ortama göre farklı yöntemlerle benzer bölünmezlikler geliştirilebilir. Örneğin veritabanı programlama sırasında koşul (condition) veya kilit (lock) kullanımı bölünmezliği sağlayabilir.

## **SORU 21: Entropi (Entropy, Dağılım, Dağıntı)**

Bir sistemin düzensizliğini ifade eden terimdir. Örneğin entropi terimini bir yazı tura atma işleminde 1 bitlik (ikil) ve %50 ihtimallik bir değer olarak görebiliriz. Burada paranın adil olduğunu ve yazı tura işleminin dengeli bir şekilde gerçekleştiğini düşünüyoruz. Şayet para hileli ise o zaman sistemin entropisi (üretilen sayıların entropisi) %50'den daha düşüktür. Çünkü daha az düzensizdir. Yani hileli olan tarafa doğru daha düzenli sonuç üretir. Örnek olarak sürekli tura gelen bir paranın ürettiği sayıların entropisi 0'dır (sıfırdır).

Entropi terimi ilk kez shannon tarafından bilgisayar bilimlerinde veri iletişimde kullanılmıştır. Dolayısıyla literatürde Shannon Entropisi (Shannon's Entropy) olarak da geçen kavrama göre bir mesajı kodlamak için gereken en kısa ihtimallerin ortalama değeri alfabede bulunan sembollerin logaritmasının entropiye bölümüdür. Yani kabaca alfabemizde 256 karakter varsa bu sayının logaritmasını ( $\log 256 = 8$ 'dir) mesajın entropisine böleriz. Yani mesajdaki değişim ne kadar fazla ise o kadar fazla kodlamaya ihtiyacımız vardır. Diğer bir deyişle alfabemiz 256 karakterse ama biz sadece tek karakter yolluyorsak o zaman entropi 0 olduğundan  $0/256 = 0$  farklı kodlamaya (0 bite) ihtiyacımız vardır. Veya benzer olarak her harften aynı sıklıkta yolluyorsak bu durumda  $256/8 = 8$  bitlik kodlamaya ihtiyaç duyulur.

Bilgisayar bilimleri açısından daha kesin bir tanım yapmak gerekirse elimizdeki veriyi kaç bit ile (ikil) kodlayabileceğimize entropi ismi verilir. Örneğin bir yılda bulunan ayları kodlamak için kaç ikile ihtiyacımız olduğu ayların dağılımıdır.

Toplam 12 ay vardır ve bu ayları 4 ikil ile kodlayabiliriz:

0000 Ocak

0001 Şubat

0010 Mart

0011 Nisan

0100 Mayıs

0101 Haziran

0110 Temmuz

0111 Ağustos

1000 Eylül

1001 Ekim

1010 Kasım

1011 Aralık

Görüldüğü üzere her ay için farklı bir bilgi girilmiş ve girilen 12 ay için 4 bit yeterli olmuştur. Dolayısıyla yılın aylarının entropisi 4'tür.

Genellikle bir bilginin entropisi hesaplanırken  $\log_2 n$  formülü kullanılır. Burada n birbirinden farklı ihtimal sayısını belirler. Örneğin yılın aylarında bu sayı 12'dir ve  $\log_2 12 = 3.58$  olmaktadır. 0.58 gibi bir bit olamayacağı için yani bilgisayar kesikli matematik (discrete math) kullandığı için 4 bit gerektiğini söyleyebiliriz.

Farklı bir örnek olarak veri tabanında bulunan kişilerin cinsiyetinin tutulacağı alan 1 bitlik olacaktır. Çünkü kadın/erkek alternatifleri tek bit ile tutulabilir:



0 Kadın

1 Erkek

şeklinde. dolayısıyla cinsiyet alanının entropisi 1'dir.

Yukarıdaki örnekte veritabanında 5 karakterlik bir dizgi (string) alanı tutmak gereksizdir. Çünkü entropi bilgisi bize 1 bitin yeterli olduğunu söyler. 5 karakterlik bilgi (ascii tablosunun kullanıldığı düşünülürse)  $5 \times 8 = 40$  bitlik alan demektir ve 1 bite göre 40 misli fazla gereksiz demektir.

Entropi terimi veri güvenliğinde genelde belirsizlik (uncertainty) terimi ile birlikte kullanılır. Belirsizlik bir mesajda farklılığı oluşturan ve saldırgan kişi açısından belirsiz olan durumdur. Örneğin bir önceki örnekteki gibi veri tabanında Kadın ve Erkek bilgilerini yazı olarak tuttuğumuzu düşünelim. Şifreli mesajımız da “fjass” olsun. Saldırgan kişi bu mesajdan tek bir biti bulursa tutulan bilgiye ulaşabilir. Örneğin 3. bitin karşılığının k olduğunu bulursa verinin erkek olduğunu anlayabilir. Dolayısıyla bu örnekte belirsizliğimiz 1 bittir.

## **SORU 22: SMTP ( Simple Mail Transport Protocol)**

SMTP, Simple Mail Transport Protocol , Basit Mektup İletim Merasimi kelimelerinin baş harflerinden oluşan ve isminden de anlaşılacağı üzere internet üzerinde mektuplaşmaya (mailing) yarayan bir protokoldür.

İnternet üzerinde mektup okuyan kişiler bilindiği üzere anlık olarak internette bulunmayabilirler. Bu tip mesajlaşmalar anlık ileti (instant mesagging) amacına yönelik özel yazılımlarla yapılmaktadır. Bunun yerine her kullanıcının mektup sunucusu (mail server) üzerinde bir posta kutusu bulunmakta ve bu kutular üzerinde mektupları birikmektedir. Ardından kullanıcı bağlanarak kutusundaki mektupları okumaktadır. Bu işin yapıldığı ve kullanıcıların mektuplarını okuduğu protokol ise [POP3 \(post office protocol\)](#) olarak bilinmektedir.

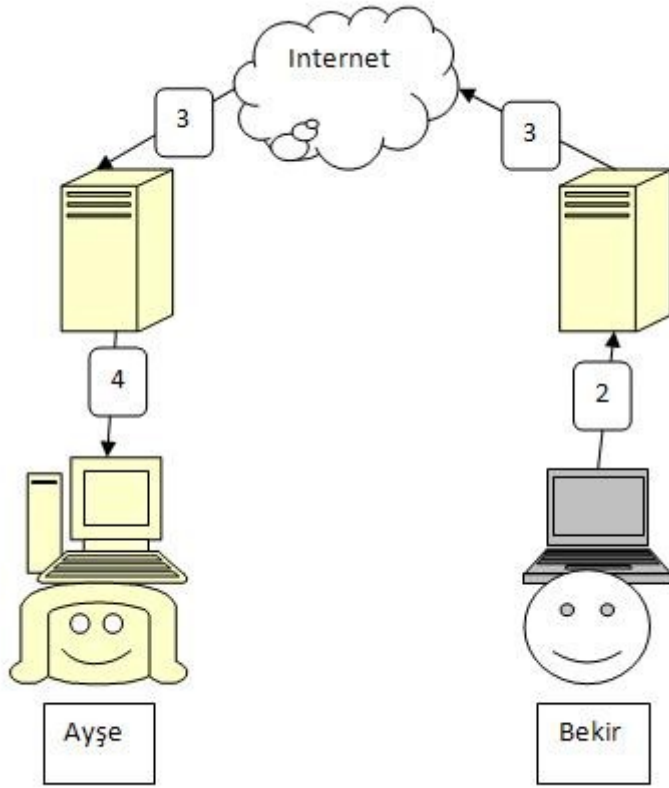
SMTP protokolü mektupları okurken değil; mektupları gönderirken kullanılan protokoldür. Bu protokolde amaç gönderilecek olan mektubun hedefteki alıcının sunucusuna iletilmesidir. Bu iletim yapıldıktan sonra zaten kullanıcının posta kutusunda mektup bekleyecektir. Dolayısıyla SMTP'nin birinci görevi hedef sunucuyu bulmak ve alıcının kutusuna mektubu bırakmaktır.

SMTP protokolü 25. port üzerinden çalışır. 3 temel aşamadan oluşur:

- Selamlaşma (handshaking)
- İletim (message transfer)
- Kapatma (closure)

ilk adımda sunucular aralarında iletişimi başlatmak için gerekli olan bilgileri geçirirler. Ardından mesaj transferi başlar ve mesaj (ya da mesajlar) yollanır. Son olarak aradaki bağlantı kapatılır.

SMTP protokolünü kullanan sunucular arasında mesajlaşma yapılırken bir sıra (queue) kullanılır. Yani bir sunucunun ileteceği birden çok mesaj varsa bu mesajlar sırasıyla iletilirler ve beklemekte olan mesajlar için bir sıra (queue) kullanılır.



Yukarıdaki şekilde bulunan ve numaralandırılmış olan 6 adımı aşağıda açıklamaya çalışalım:

1. adımda Bekir, Ayşe'ye bilgisayarında bulunna program üzerinden (user agent) mektup yazmaktadır.
2. adımda Bekir mektubunu yollar ve Bekir'in bilgisayarındaki program, Bekir'in SMTP sunucusuna (smtp server) bağlanarak mektubu sunucuya yükler
3. adımda Bekir'in SMTP sunucusu internet üzerinden Ayşe'nin sunucusunu bulur ve bu sunucuya mektubu teslim eder.
4. adımda Ayşe mektubu kendi bilgisayarındaki program marifetiyle (user agent) görüntüler.

Bu adımlar sırasında olan mesajlaşma SMTP protokolünde aşağıdakine benzer bir haldedir:

```
S: 220 bilgisayar kavramlari.com
İ: HELO sadievrenseker.com
S: 250 Hello sadievrenseker.com, pleased to meet you
İ: MAIL FROM: <bekir@sadievrenseker.com>
S: 250 bekir@sadievrenseker.com... Sender ok
İ: RCPT TO: <ayse@bilgisayarkavramlari.com>
S: 250 ayse@bilgisayarkavramlari.com ... Recipient ok
İ: DATA
S: 354 Enter mail, end with "." on a line by itself
```

```
İ: Ben Bekir;  
İ: Bilgisayar kavramlarında yeni neler var?  
İ: .  
S: 250 Message accepted for delivery  
İ: QUIT  
S: 221 sadievrenseker.com closing connection
```

Yukarıdaki mesajlaşmada, bilgisayar kavramlari.com sunucusuna bağlanarak bekir isimli kullanıcının mesajı bekir@sadievrenseker.com adresinden ayse@bilgisayarkavramlari.com adresine gönderilmiştir (İ: istemci (sadievrenseker.com), S: sunucu tarafıdır (bilgisayarkavramlari.com ) ). mesaj:

“Ben Bekir;

Bilgisayar kavramlarında yeni neler var?”

şeklinde. Bu mesajlaşmanın ilk 3 satırı sunucular arası selamlaşma (handshaking), son 2 satırı da kapatma (closure) işlemidir. Bu satırlar dışındaki satırlar mesajın gönderilmesi işlemidir. Şayet birden fazla mesaj gönderilecek olsaydı istemci yeniden MAIL FROM komutuyla yeni bir mektup transferi başlatacaktı.

### **SORU 23: DHCP Sunucu (DHCP Server)**

Dynamic Host Configuration Protocol (Dinamik sunucu konfigürasyon protokolü, müteharrik hancı teşkilat merasimi)

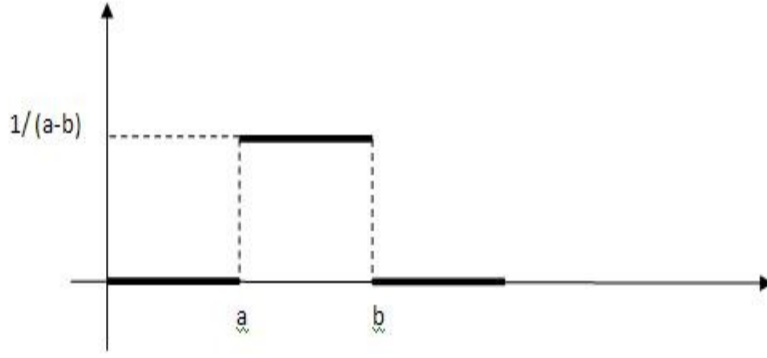
Bu protokolün amacı IP (Internet Protocol) dağılımını dinamik olarak yapmak ve böylece anlık olarak bağlı olmayan bilgisayarların IP kaynağını bağlı olan bilgisayarlara kaydırmaktır.

Örneğin elimizde 10 IP numaramız bulunsun, ağımda bu 10 bilgisayar ile aktif olmalarına bağlı olarak 10dan fazla bilgisayarı bağlamamız mümkündür. Şayet statik IP adresi atayacak olursak (yani her bilgisayara bir IP adresi verecek olursak) bu durumda ancak 10 bilgisayar bağlayabilirken dinamik olarak bilgisayara IP adresi vererek o anda bağlı olmayan IP adreslerini başkasının kullanması sağlanabilir.

DHCP sunucu ayarlanırken bir IP adres aralığı belirlenir. Bu aralıktan otomatik olarak talepte bulunan bilgisayarlara IP numarası ataması yapılır.

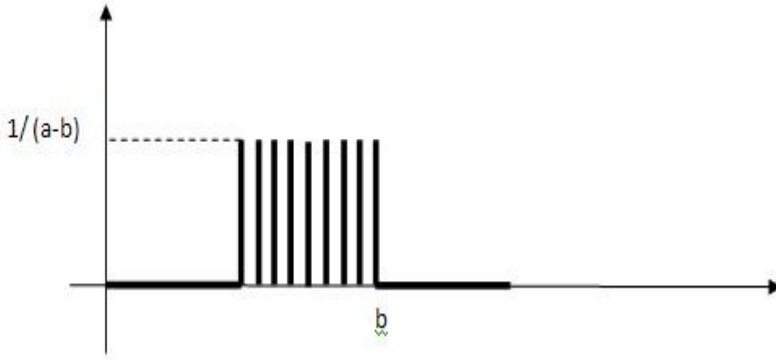
### **SORU 24: Uniform Dağılım ( Uniform Distribution, Yeknesak, Tekdüze, Biteviye)**

Matematiksel olarak rastgele üretilen sayıların belirli bir düzen içerisinde olması durumudur. Dağılım grafiği aşağıdaki şekilde beklenir:



Yukarıdaki şekilde bir uniform dağılımın grafiği verilmiştir. Buna göre dağılımda yer alan sayı aralığı  $a$  ve  $b$  sayıları arasında yer almaktadır.

Basit bir ifade ile örneğin bir bilgisayarda üretilen gürültünün veya rastgele sayıların uniform dağılımda olması demek bütün değerlerin  $a$  ile  $b$  arasında eşit dağılması demektir. Bu durum aşağıdaki örnekte daha net görülmektedir:



Yukarıdaki ilk resimde sürekli (continuous) bir dağılım varken hemen yukarıdaki ikinci resimde kesikli (discrete) bir dağılım söz konusudur. Bilgisayar bilimlerinde kullanılan tahmin edileceği üzere daha çok kesikli olan dağılımdır. Yani verilen  $a$  ve  $b$  aralığındaki her sayı için bir karşılık üretilmekte ve bu üretilen karşılıkların miktarı birbiri ile aynı olmaktadır.

Örneğin bir resimdeki histogramın çıkarılması sonucu bütün resmin verilen aralıktaki renk kodlarından aynı miktarda alması veya bir ağ iletişimi sırasında sinyale üzerinde oluşan gürültünün sinyalin aynı aralığında aynı miktarda uygulanması gibi.

### **SORU 25: Traceroute**

Bir konumdan başka bir konuma giden internet paketlerinin (IP) geçtikleri güzergahtaki düğümleri (nodes) görmeye yarayan yazılımın ismidir.

Örneğin bulunduğumuz konumdan, internet üzerindeki herhangi bir konuma paketlerin hangi yolu izleyerek geçtiğini görmek için çoğu işletim sisteminde desteklenen (windows, linux, unix gibi) aşağıdaki komutu çalıştıralım:

```

tracert
En          fazla          30          atlamanın          72.55.146.14
ip-72-55-146-14.static.privatedns.com [72.55.146.14]'ye izleme yolu :          üstünde

```

1	<1	ms	<1	ms	<1	ms	192.168.1.1
2	11	ms	9	ms	10	ms	85.99.239.1
3	11	ms	10	ms	12	ms	mstr81212-18313.dial-in.ttnet.net.tr [81.212.71.137]
4	885	ms	814	ms	400	ms	acibadem_t2_2-kadikoy_t3_1.turktelekom.com.tr [212.156.118.253]
5	12	ms	32	ms	11	ms	acibadem_t2_1-acibadem_t2_2.turktelekom.com.tr [212.156.117.245]
6	68	ms	68	ms	66	ms	ams_col_1-acibadem_t2_1.turktelekom.com.tr [212.156.102.9]
7	67	ms	67	ms	66	ms	ams-sara-cor-1.peer1.net [195.69.145.209]
8	73	ms	74	ms	73	ms	oc48-so2-1-0.ldn-teleh-dis-1.peer1.net [216.187.115.49]
9	166	ms	165	ms	165	ms	oc48-so-3-1-0.nyc-telx-dis-1.peer1.net [216.187.115.225]
10	173	ms	174	ms	173	ms	oc192.xe-1-0-0.mtl-bvh-cor-1.peer1.net [216.187.115.53]
11	175	ms	174	ms	174	ms	10ge.xe-0-0-0.mtl-bvh-cor-2.peer1.net [216.187.115.86]
12	179	ms	178	ms	179	ms	g9-8.hd-core02.peer1.mtl.iweb.com [216.187.90.134]
13	179	ms	179	ms	180	ms	ip-72-55-146-14.static.privatedns.com [72.55.146.14]

İzleme tamamlandı.

Yukarıda da görüldüğü üzere bilgisayarımızın internete bağlı olduğu noktadan başlayarak hedef bilgisayara kadar olan bir paketin geçtiği bütün düğümler (atlamalar, nodes, hops) gösterilmiştir. Ayrıca bu düğümler arasındaki geçiş zamanı en başta yer alan ms (milisecond, mili saniye) cinsinden gösterilmektedir.

## SORU 26: Extranet (Dış ağ)

Kabaca bir kurumun dışarıya açık ağı anlamına gelmektedir. Örneğin bir firmanın kendi sunucularına erişim için kurmuş olduğu kurumsal ağı (Intranet) dışarıya açılması ve iş yaptığı çeşitli firmaların erişimi için dışarıdan erişilebilir bir ağ sağlaması durumudur.

Farklı bir bakışla Intranet'in (iç ağ) bir parçası olarak görülüp, dışarıdan bağlanacaklara açılmış bir parça olarak yorumlamak mümkündür.

Genellikle kurumsal yapıları bağlayan işten işe (Business to Business (B2B)) ağlar ve son kullanıcıları kurumlara bağlayan işletim müşteriye (Business to consumer (B2C)) kurulumlar bu yapının altında düşünülebilir.

Ağ yapılandırması açısından Intranet'i tek bir VPN (Virtual Private Network, Sanal Kişisel Ağ) olarak düşünecek olursak, dış ağ (extranet) kavramını bu VPN'e ilave dış VPN'ler olarak düşünmek mümkündür.

## SORU 27: Intranet (İç Ağ)

Gelişen ağ teknolojileri ile birlikte İnternet'in özelleştirilmesi de mümkün olmuştur. Örneğin bir şirketin posta sunucuları (mail servers), web sunucuları (web servers) DNS'i, FTP sunucuları ve benzeri pekçok sunucusu bulunmaktadır. Şayet şirket bu sunucuları kendisine

özgü olarak sadece kendi çalışanlarının erişebileceği şekilde ayarlarsa bu ağ tipine Intranet (iç ağ) ismi verilmektedir.

Kısaca bir kurumun (şirket, üniversite, dernek gibi) kendisine özgü bir internet kurması durumudur.

Tek merkezli olup bütün sunucuların bir merkezden yönetildiği ve kullanıcıların çeşitli yerlerden bağlanabildiği sistemler olduğu gibi, çok merkezli olup her merkezde farklı yönetimlerin olduğu uygulamalar da bulunmaktadır. Örneğin çok uluslu ve çok şubeli bir yapıda her şubenin kendi sunucularını barındırması mümkündür.

Intranet uygulamalarının erişimi için tamamen özel hatların kullanılması mümkün olduğu gibi İnternet üzerinden de erişmek mümkün olabilmektedir.

Örneğin tamamen özel bir Intranet uygulamasında kullanıcılar ya şirketlerinden ağa özel kablolama ile bağlanmakta ya da şirketin özel telefon hatlarına bağlanarak Intranete ulaşmaktadır.

İnternet üzerinden erişilen uygulamalarda ise kullanıcılar çeşitli güvenlik aşamalarını geçerek ağa ulaşmaktadır. Bu tip erişimde iki farklı ağda paketler gittiği için tünelleme (tunnelling) kullanılabilir. Yani Intranette gidecek olan paketler İnternet paketlerinin içerisine konularak ulaştırılabilir.

## **SORU 28: Çift Yönlü İletişim (Duplex Communication)**

Bir iletişimin tipini belirlemek için kullanılan terimdir. Kabaca bir otoyolun tek yönlü veya çift yönlü olması mümkündür. Çift yönlü otoyolun ise tek şerit veya çift şerit olması mümkündür.

Duplex terimi aynı anda iki tarafında iletişim kurduğu sistemler için kullanılmıştır. İkiye ayırmak mümkündür:

- Full Duplexing (aynı anda çift yönlü iletişim)
- Half Duplexing (paylaşımlı çift yönlü iletişim)

Full duplex iletişimde aynı anda iki taraf da konuşabilmektedir. Örneğin telefon konuşmalarında her iki tarafta aynı anda konuşabilmekte ve iki tarafta birbirini duyabilmektedir. Genellikle tek hat üzerinde birden fazla tarafın iletişim kurması [frekans paylaşımlı iletişimle mümkün olmaktadır \(Frequency division multiplexing\)](#)

Half duplex iletişimde ise bir taraf konuşurken diğer taraf beklemek zorundadır. Örneğin telsiz konuşmalarında bir taraf konuşurken diğer taraf konuşamaz. Konuşmaya çalışırsa iki taraf da birbirini duyamaz. Bu durumda hattın doğru zamanlamasının yapılması gerekir ([Zaman paylaşımlı iletişim, Time division multiplexing](#))

## **SORU 29: POP3**

POP3 protokolü, post office protocol (postahane, postane protokolünün) kısaltılmışıdır. Bu protokolün çalışma mantığı sürekli bağlı kalmayan kullanıcıların gelen iletilerinin sunucuda saklanmasına dayanır.

Buna göre her kullanıcının, sunucu üzerinde bir posta kutusu bulunur ve gelen iletiler (e-postalar, e-mails) bu kutuda biriktirilir. Kullanıcı sunucuya bağlandığı zaman kutusunda bulunan mektupları kendi bilgisayarına çekerek bağlı olmadığı süre içerisinde de okuyabilir. Sunucuda bulunan mektupları ise ister siler ister olduğu gibi bırakır.

POP1 ve POP2 protokollerinin gelişmiş hali olan POP3'ten sonra günümüzde henüz yaygın olarak kullanılmayan ve klasör özelliği, mektubu parçalara ayırmak gibi özellikleri içeren POP4 protokolü de bulunmaktadır.

POP3 protokolü çok özel bir ayar yapılmadıktan sonra 110 numaralı port üzerinden hizmet vermektedir. Aşağıda örnek bir POP3 protokolü iletişimi gösterilmiştir:

```
S: <wait for connection on TCP port 110>
C: <open connection>
S: +OK POP3 server ready <sadi@bilgisayarkavramlari.com>
C: APOP sadi c4c9334bac560ecc979e58001b3e22fb
S: +OK sadi's maildrop has 2 messages (320 octets)
C: STAT
S: +OK 2 320
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
C: RETR 1
S: +OK 120 octets
S: <the POP3 server sends message 1>
S: .
C: DELE 1
S: +OK message 1 deleted
C: RETR 2
S: +OK 200 octets
S: <the POP3 server sends message 2>
C: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
C: <close connection>
S: <wait for next connection>
```

Yukarıdaki örnekte S harfi ile başlayan satırlar Sunucu (server) mesajları ve C harfiyle başlayanlar İstemci (Client) mesajları olmaktadır. Yukarıdaki iletişimde öncelikle sadi@bilgisayarkavramlari.com adresinin sunucu üzerinde sisteme girişi yapılmakta ardından sunucuda bulunan 2 mesajdan ilkinin okuyup silmekte ve son olarak 2. mesajı okuyarak sistemden çıkmaktadır.

### **SORU 30: Noktadan Noktaya İletişim (Point to Point Protocol PPP)**

Veri bağlama katmanı (Data Link Layer) protokollerinden birisi olan PPP üzerinden iki noktanın iletişim kurması mümkündür. Bu protokol sayesinde kullanıcı kontrolü, veri sıkıştırma ve şifreli iletişim mümkündür.

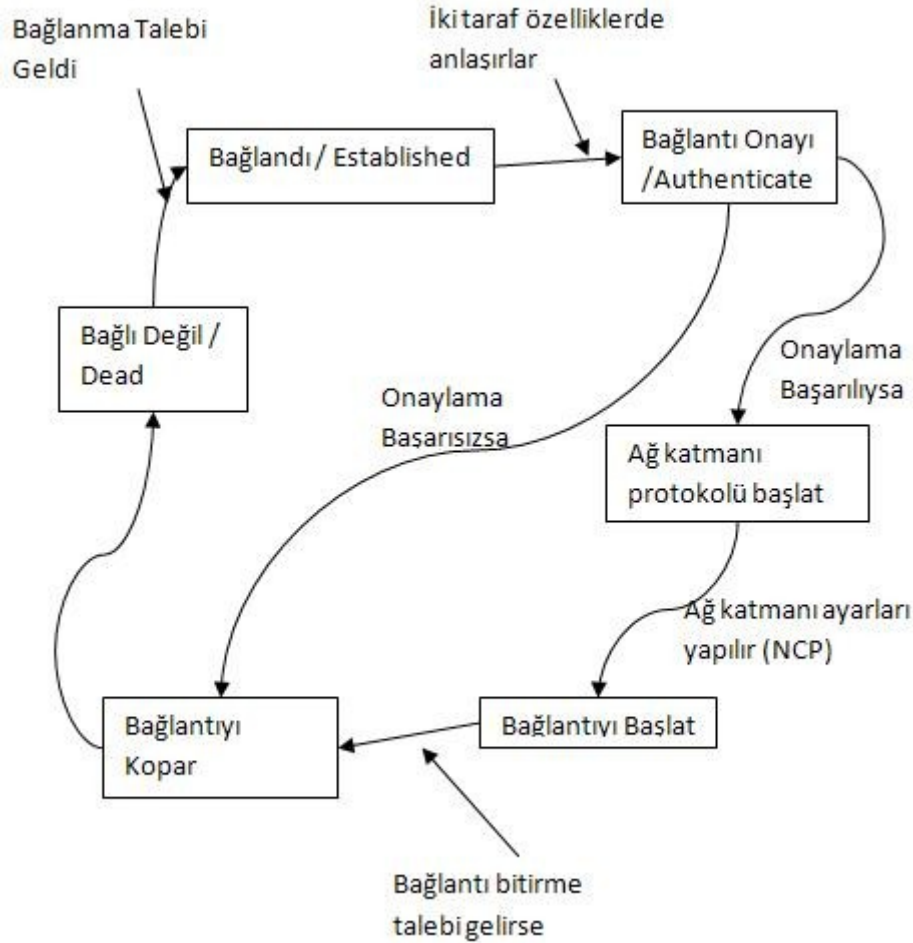
PPP protokolü oldukça fazla alanda kullanılmaktadır, telefon iletişimi, fiber optik kablo ortamı, radio iletişim ortamı, cep telefonları gibi ortamlarda kullanılır. Örneğin ADSL hatların yaygınlaşması öncesinde Türkiye'de de İnternet Servis Sağlayıcıları (İnternet Service



Providers, ISP) tarafından sunulan telefon hattı üzerinden bağlantı (Dial up) için telefon bağlantısı sağlandıktan sonra modem üzerinde PPP protokolü yaygın olarak kullanılmaktaydı.

Günümüzde ise ADSL hatların üzerinde yaygın olarak PPPoE (Point to point protocol over Ethernet, Ethernet üzerinden Noktadan Noktaya iletişim) veya PPPoA ( Point to Point Protocol over ATM, ATM üzerinden Noktadan noktaya iletişim) bu protokolün birer örneğidir.

PPP'nin çalışmasını gösteren durum şeması aşağıda verilmiştir. Buna göre çeşitli durumlar ve bu durumlar arasında geçişler açıklanmıştır (State Transition Diagram)



Yukarıda da görüldüğü üzere hat ya bağlı ya da değildir. Bağlı olmadığı durumda (yukarıdaki Bağlı değil / Dead durumu) bağlantıya geçmek için öncelikle hatta bir talep gelmelidir (carrier sense). Ardından bağlantı onayı ve ağ katmanı protokolünün devreye girmesi takip eder.

### **SORU 31: DNS (Domain Name System, Alan İsim Sistemi)**

DNS'in amacı insanların anlaması kolay olan alan isimlerini (domain names) bilgisayarların iletişiminde kullanılan [IP adres](#) çevirmektir.

Basitçe, internette gezen birisi girmek istediği sitenin adresini web tarayıcısına (web browser) girerek bu adrese bağlanmak ister. Ancak bu adresin konumu bağlı olduğu IP adresine göre

belirlenebilir. İşte adresin karşılığı olan IP adresine dönüşüm için DNS sunucuları kullanılmalıdır.

İnternet üzerinde bir DNS şebekesinden bahsetmek mümkündür. Buna göre bir DNS üzerinde İnternette bulunan bütün alan adlarının bulunması imkansızdır. Bu yüzden bu bilgi pek çok DNS üzerinde paylaştırılmıştır. Kısaca DNS yapısı dağıtık bir veritabanı (Database) benzetmek mümkündür. Bu yapıya göre bir alan adı sorgulaması sırasında DNS biliyorsa bu bilgiyi isteyen kişiye verir. Şayet bilmediği bir adres söz konusuysa bu sorguyu kendisinin bir üstündeki (veya kendisinden bir sonraki) domain name server'a (DNS) yönlendirir (forward).

DNS'leri ikiye ayırmak mümkündür. TLD ismi verilen Top Level Domain (Üst seviye alanları) tutan ve nispeten büyük isim sunucuları ve Otokratif (authoritative) şahsi yönetim altındaki diğer isim sunucuları. TLD'lerde genel olarak .com, .org, .net gibi üst alan isimleri ile bütün ülke isimleri (.tr , .fr, .uk gibi) durmaktadır. Bunlar dışındaki bütün alan adları otokratif sunucular üzerinde tutulurlar ve yönetimi kişisel veya kurum bazlıdır.

Bir DNS sunucusunda yapılan işleri aşağıdaki şekilde sıralamak mümkündür:

- istemci isminden(hostname) IP adresine dönüşüm sağlarlar
- istemci takma ismine izin verirler (host aliasing, canonical (normal gösterim))
- posta sunucusu takma ismi sağlarlar (mail server aliasing)
- Yük dağıtımı sağlarlar (web sunuculara eşit dağılım sağlayarak birden fazla web server'ın tek sunucu gibi çalışmasını ve yükün bu sunucular üzerine eşit dağılmasını sağlarlar)

Bir DNS üzerinde yapılan sorgular iki şekilde olabilmektedir:

Özyineli sorgular (Recursive queries):

Bu sorgu tipinde bir sunucuya sorulan alan adı bu sunucu tarafından cevaplanır. Ancak sunucu sorguyu bilmiyorsa istemci adına bir üstteki sunucuya sorar ve bu bir cevap bulunana kadar böylece devam eder. Sorguya bir cevap bir sunucudan gelince bütün sunucular bu cevabı ilk sunucuya geri iletirler. En sonunca sorguyu ilk alan sunucu istemciye cevabı dönmüş olur.

Tekrarlanan sorgular (İterative queries)

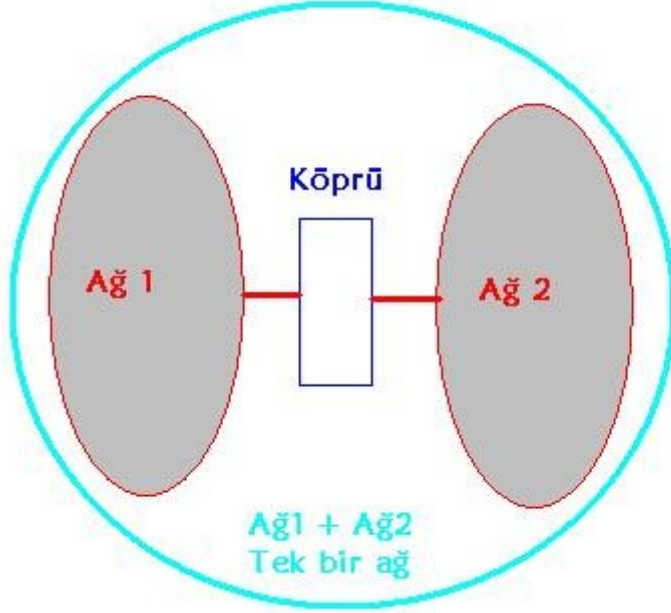
Bu yöntemde ise sorunun muhatabı olan DNS sorguyu sırasıyla diğer DNS'lere sorar ve cevabı bulunca cevabı ilgili istemciye yönlendirir.

### **SORU 32: Köprü (Bridge)**

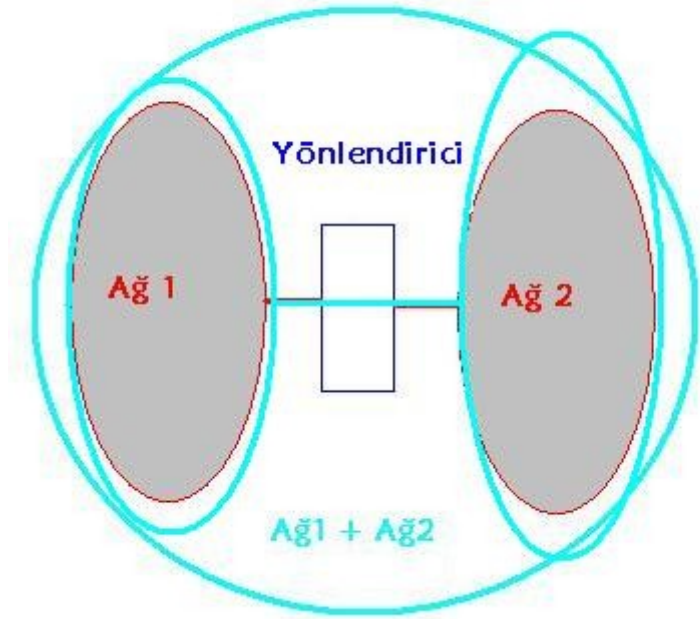
Aktif bir ağ (network) cihazı olan köprünün amacı iki farklı ağı birbirine bağlamaktır. Bu cihaz yönlendiricilerin (routers) sık kullanılması ve ucuzlayan maliyetleri sonucunda günümüzde çok sık kullanılmamaktadır.

Kısaca bağlı olduğu iki ağdaki paketleri birbirine aktarır ve bu sayede sanki tek bir ağ varmış gibi çalışmasını sağlar. Bu cihazın yönlendirici (router) ile değiştirilmesi ile iki ağ arasında gereksiz trafik engellenmiş olur. Yani bir ağdaki paketin diğer ağa gitmesi gerekmiyorsa bu paket ilgili ağda kalarak diğer ağda trafik oluşturmaz.

Yukarıdaki şekilde köprülenmiş (bridged) ağların ikisi birden tek bir ağ gibi çalışmaktadır. Arada köprü ismi verilen bir cihaz bulunmakta ve ağların birbirine bağlamaktadır. Bir ağdaki paket bütün ağı dolaşabilmektedir.



Yukarıdaki şekilde ise arada yönlendirici (Routers) konulmakta ve iki ağ ayrı ağlar olarak çalışmakta ancak ve ancak diğer ağına gönderilen bir paket olduğunda yönlendirici üzerinden paket geçerek diğer ağıda trafik oluşturmaktadır.



Routerlar (yönlendiriciler), birdgelerden (köprülerden) farklı olarak OSI katman 2 yerine OSI katmanı 3'e kadar yönlendirme yapabilmektedir. Bunun anlamı bir router ile frame (çerçevelerin) yönlendirilmesi yerine paket (packet) yönlendirme yapılabilmektedir. Yani Bridgeler MAC (machine address code) seviyesinde yönlendirme yapabilmekteyken yönlendiriciler (routers) bir seviye üstü olan IP adresi bazında yönlendirme yapabilmektedir.

### **SORU 33: Tekrarlayıcı (Repeater)**

Aktif bir ağ (network) cihazıdır. Bu cihazın amacı çeşitli sebeplerle zayıflamış olan sinyali kuvvetlendirerek ortama geri salmaktır.

Örneğin UTP kablo üzerinden taşınan sinyallerin 180m ve üzeri mesafelerde zayıflaması ve kullanılan sinyal kaynağı ve alıcının türüne göre hiç okunamaması söz konusudur. Bu durumda 180m üzerindeki mesafelere UTP kablo ile sinyal taşımak için belirli aralıklarla tekrarlayıcılar yerleştirilebilir. Örneğin 1km mesafeye sinyal taşımak için her 100m mesafede bir tekrarlayıcı kullanılarak toplam 10 tekrarlayıcı ile sinyal bozulmadan taşınabilir.

Tekrarlayıcıların ( repeaters ) sık kullanıldığı ortamlar kablolu ortamların aksine kablosuz ortamlardır. Çünkü kablolu ortamlarda her aktif cihaz (router, switch, hub vs.) birer tekrarlayıcı olarak çalışır. Kablosuz ortamlar ise daha çok sinyalin uzun mesafeli taşınması istenen (genelde geniş ağ ( wide area network ( wan ) ) ve genelde engelli ( coğrafi şartlar , dağlar, bulutlar vs.) ortamlardır.

### **SORU 34: Sıralama Algoritmaları (Sorting Algorithms)**

Bilgisayar bilimlerinde verilmiş olan bir grup sayının küçükten büyüğe (veya tersi) sıralanması işlemini yapan algoritmalara verilen isimdir. Örneğin aşağıdaki düzensiz sayıları ele alalım:

5 9 2 3 7 11 -4 6

Bu sayıların sıralanmış hali

-4 2 3 5 6 7 11

olacaktır. Bu sıralama işlemini yapmanın çok farklı yolları vardır ancak bilgisayar mühendisliğinin temel olarak üzerine oturduğu iki performans kriteri buradaki sonuçları değerlendirmede önemli rol oynar.

- Hafıza verimliliği (memory efficiency)
- Zaman verimliliği (Time efficiency)

Temel olarak algoritma analizindeki iki önemli kriter bunlardır. Bir algoritmanın hızlı çalışması demek daha çok hafızaya ihtiyaç duyması demektir. Ters durumda da bir algoritmanın daha az yere ihtiyaç duyması daha yavaş çalışması demektir. Ancak bir algoritma hem zaman hem de hafıza olarak verimliyse bu durumda diğer algoritmalarından başarılı sayılabilir.

Genellikle verinin hafızada saklanması sırasında veriyi tutan bir belirleyici özelliğinin olması istenir. Veritabanı teorisinde birincil anahtar (primary key) ismi de verilen bu özellik kullanılarak hafızada bulunan veriye erişilebilir. Bu erişim sırasında şayet belirleyici alan sıralı ise erişimin logaritmik zamanda olması mümkündür. Şayet veri sıralı değilse erişim süresi doğrusal (linear) zamanda olmaktadır.

Aşağıda bazı sıralama algoritmaları verilmiştir:

- [Seçerek Sıralama \(Selection Sort\)](#)
- [Hızlı Sıralama Algoritması \(Quick Sort Algorithm\)](#)
- [Birleştirme Sıralaması \(Merge Sort\)](#)
- [Yığınlama Sıralaması \(Heap Sort\)](#)
- [Sayarak Sıralama \(Counting Sort\)](#)
- [Kabarcık Sıralaması \(Baloncuk sıralaması, Bubble Sort\)](#)
- [Taban Sıralaması \(Radix Sort\)](#)
- [Sokma Sıralaması \( Insertion Sort\)](#)
- [Sallama Sıralaması \(Shaker Sort\)](#)
- [Kabuk Sıralaması \(Shell Sort\)](#)
- [Rastgele Sıralama \(Bogo Sort\)](#)
- [Şanslı Sıralama \(Lucky Sort\)](#)
- [Serileri Sıralaması \(Stooge Sort\)](#)
- [Şimşek Sıralaması \(Flahs Sort, Bora Sıralaması\)](#)
- [Tarak Sıralaması \(Comb Sort\)](#)
- [Gnome Sıralaması \(Gnome Sort\)](#)
- [Permütasyon Sıralaması \(Permutation Sort\)](#)

- [Strand Sort \(İplik Sıralaması\)](#)

Yukarıda verilen veya herhangi başka bir sıralama algoritması genelde küçükten büyüğe doğru (ascending) sıralama yapar. Ancak bunun tam tersine çevirmek (descending) genelde algoritma için oldukça basittir. Yapılması gereken çoğu zaman sadece kontrol işleminin yönünü değiştirmektir.

Ayrıca sıralama işleminin yapılması sırasında hafızanın kullanımına göre de sıralama algoritmaları :

- [Harici Sıralama \(External Sort\)](#)
- Dahili Sıralama (Internal Sort)

şeklinde iki grupta incelenebilir.

Algoritmaların karşılaştırılması için aşağıdaki tablo hazırlanmıştır:

Algoritma	İngilizces i	Algoritma Analizi			Kararlılı	Yöntem	Açıklama
		En İyi	Ortalama	En Kötü			
Seçerek Sıralama	Selection Sort	$n^2$	$n^2$	$n^2$	Kararsız	Seçerek	
Hızlı Sıralama	Quick Sort	$n \log(n)$	$n \log(n)$	$n^2$	Kararsız	Parçala Fethet	
Birleştirme Sıralaması	Merge Srot	$n \log(n)$	$n \log(n)$	$n \log(n)$	Kararlı	Parçala Fethet	
Yığınlama Sıralaması	Heap Sort	$n \log(n)$	$n \log(n)$	$n \log(n)$	Kararsız	Seçerek	
Sayarak Sıralama	Counting Sort	$n + 2^k$	$n + 2^k$	$n + 2^k$	Kararsız	Sayarak	k ikinci dizinin boyutu.
Kabarcık Sıralaması	Bubble Sort	$n$	$n^2$	$n^2$	Kararlı	Yer Değiştirme	
Kokteyl Sıralması	Coctail Sort	$n$	$n^2$	$n^2$	Kararlı	Yer Değiştirme	Çift yönlü kabarcık sıralaması (bidirectional bubble sort) olarak da bilinir ve dizinin iki ucundan işleyen kabarcık

							sıralamasıdır.
Taban Sıralaması	Radix Sort	$n(k/t)$	$n(k/t)$	$n(k/t)$	Kararlı	Gruplama / Sayma	k, en büyük eleman değeri, t ise tabandır
Sokma Sıralaması	Insertion Sort	n	d+n	$n^2$	Kararlı	Sokma	d yer değiştirme sayısıdır ve $n^2$ cinsindendir
Sallama Sıralaması	Shaker Sort	$n^2$	$n^2$	$n^2$	Kararsız	Seçme	Çift yönlü seçme sıralaması (bidirectional selection sort) olarak da bilinir.
Kabuk Sıralaması	Shell Sort	$n^{3/2}$	$n^{3/2}$	$n^{3/2}$	Kararsız	Sokma	
Rastgele Sıralama	Bogo Sort	1	n n!	sonsuz	Kararsız	Rastgele	Algoritma olduğu tartışmalıdır. Knuth karıştırması (knuth shuffle) süresinde sonuca ulaşması beklenir.
Bozo Sıralaması	Bozo Sort	1	n!	sonsuz	Kararsız	Rastgele	Rastgele sıralamanın özel bir halidir. Rastgele olarak diziyi karıştırdıktan sonra, dizi sıralanmamışsa, yine rastgele iki sayının yeri değiştirilip denenir.
Goro Sıralaması	Goro Sort	$2^{(\log(d)/\log(2))}$	$2^{(\log(d)/\log(2))}$	$2^{(\log(d)/\log(2))}$	Kararsız	Rastgele	2011 Google kod yarışı (google code jam) sırasında ortaya çıkmıştır. Sıralanana kadar her alt küme permüte edilir. Buradaki

							performans değeri ispatlanmamıştır ve d dereceyi ifade eder.
Şanslı Sıralama	Lucky Sort	1	1	1	Kararsız	Rastgele	Algoritma olarak kabul edilmemelidir.
Serseri Sıralama	Stooge Sort	$n^3$	$n^3$	$n^3$	Kararsız	Yer değiştirme	e, doğal logaritma sayısıdır (2,71)
Şimşek Sıralaması	Partial Flash Sort	n	$n + d$	$n + d$	Kararsız	Yer Değiştirme	d, kullanılan ikinci algoritmanın performansıdır, bu algoritma bu listedekilerden herhangi birisi olabilir.
Permütasyon Sıralaması	Perm Sort	n	$n \cdot n!$	$n \cdot n!$	Kararsız	Yer Değiştirme	
Bazı Yegane Sıralaması	Several Unique Sort	n	$n^2$	$n^2$	Kararsız	Yer Değiştirme	Bir bilgisayar programı tarafından bulunmuştur.
Tarak Sıralaması	Comb Sort	$n \log(n)$	$n \log(n)$	$n \log(n)$	Kararsız	Yer Değiştirme	Kabarcık ve hızlı sıralama algoritmalarının birleşimi şeklinde düşünülebilir

Yukarıdaki yazıda geçen kararlılık kolonu ile, bir algoritmanın bitiş kontrolüne dayanmaktadır. Örneğin sıralı bir dizi verilse bile sıralama işlemi yapmaya çalışır mı?

### **SORU 35: Düğüm (Node)**

Bir graf üzerindeki her noktaya düğüm adı verilir. Düğümler, kenarlar kendi üzerlerinde birleştiği için bu ismi almışlardır.

Graf teorisine göre bir düğümün derecesi o düğümde bulunan kenar sayısıdır. Örneğin aşağıdaki grapta A düğümünün derecesi 3'tür.



### SORU 36: Eşlik biti kontrolü (parity bit check)

Eşlik biti yada parity bit olarak bilinen bu bit, ikilik tabandaki bitlerin tek veya çift olması esasına göre kontrol amaçlı olarak kullanılan bittir.

Örneğin 7 bitlik bir mesajın bit değerlerinin toplamı tek ise 1 çift ise 0 bilgisini de bu 7bitlik mesaja ekleyerek 8 bitlik bir mesaj elde edilir. Karşı taraf mesajı aldıktan sonra bit değerlerini toplayarak 8. bit olan eşlik biti ile karşılaştırır. Şayet eşlik biti tutuyorsa sorun yok demektir, şayet tutuşmuyorsa mesajda veya eşlik bitinde bir bozulma var demektir.

Örnek:

Mesaj: 1011011

Mesajın toplam değeri (kaç tane 1 olduğu) : 5 , 5 bir tek sayı olduğu için pariti bit olarak karşıya 1 yollanacak

Parity bit ile mesaj: 10110111

Alan taraf mesajın ilk 7 bitini toplayara 5 sayısını bulur bu tek sayı olduğu için parity bit olarak 1 olması gerektiğini düşünür ve parity bit 1 olduğu için sorunsuz aldığına karar verir.

parity bit hesaplanmasında binary toplama işlemi de yapılabilir. Yukarıdaki örnek için basitçe:

$1+0+1+1+0+1+1 = 1$  olarak bulunur (onluk tabana çevirmeksizin ikilik tabanda toplamın son biti alınır)

Bu kontrol işleminin başarısız olması için en az 2 bitte birden bozulma olması gerekmektedir.

### SORU 37: Aloha

CSMA ve Token Ring gibi bir Yerel Ağ bağlantı protokolü olan Aloha 1970 yılında Hawaii Üniversitesinde geliştirilmiştir. Geliştirme sürecinde temel alınan ortam kablosuz ağlardır bu yüzden örneğin uydu iletişimi de dahil olmak üzere pekçok yerde kullanılabilir. Çalışma mantığı basittir:

Gidecek veri varsa yolla  
Şayet çakışma olursa daha sonra tekrar yolla

Yukarıdaki iki adımlık çalışmada en çok tartışılan kısım “daha sonra” kısmıdır. En verimli “daha sonra”nın ne olacağı bir araştırma konusudur. Aloha protokolü aslında ikiye ayrılabilir. Bir saf aloha (pure aloha) diye adlandırılan ve ilk çıktığında kullanılan çeşidi varken bir de bölümlü aloha (slotted aloha) adı verilen ikinci bir çeşitten bahsedilebilir. İlk çeşidinin hat kullanım başarısı %18 seviyesindeyken ikincisi %36 gibi başarılarla ulaşabilmektedir, ilk durumda hattın %82’si ikinci durumda ise %64’ü israf edilmektedir.

Aloha’nın en çok karşılaştırıldığı protokol CSMA protokolüdür çünkü yerl ağda en çok kullanılan ve en başarılı protokollerden birisidir. CSMA yaklaşımında her bilgisayar, ağda bir

çakışma olduğundan haberdar edilmektedir, aynı zamanda CSMA yaklaşımında her bilgisayar hatta bir bilgi koymadan önce hattı sorgulamakta ve ancak hat serbestse hatta bilgi koymaktadır, bu durum özellikle bant genişliğinin düşük olduğu kablosuz bağlantılarda oldukça zordur.

#### Artıları:

uygulaması kolaydır  
Hattı tek bir bilgisayar sürekli olarak tam kapasitede kullanabilmektedir  
Merkezî yönetim gerektirmeyen ağda sadece slotların uyumlu olması yeterlidir

#### Eksileri:

Çakışma ihtimali ve israf olan zaman bölümleri vardır  
Verimliliği tartışma konusudur

Aloha protokolünde çakışma olmama olasılığı şu şekilde hesaplanabilir. Örneğin hatta bir noktadan veri gönderilme olasılığı  $p$  olsun, bu durumda  $n$  adet nokta için veri gönderilmeme olasılığı  $(1-p)^n$  olacaktır. Anlık olarak veri yolladığımız için yollanan zamandan bir önceki ve bir sonraki zamanlarda da çakışma olmamasına dikkat edilmelidir. Öyleyse örneğin  $t_n$  zamanında veri yollanıyorsa  $t_{n-1}$  ve  $t_{n+1}$  zamanlarında da çakışma olmamalıdır. Ve başarılı yollama olasılığı  $p \cdot (1-p)^{n-1} \cdot (1-p)^{n-1} = p \cdot (1-p)^{2n-2}$  olarak bulunur. İlk  $p$  olasılığı bir noktadan veri gönderilme olasılığı, ikinci  $(1-p)^{n-1}$  ile  $t_{n-1}$  zamanı arasındaki çakışma olmama olasılığı ve son  $(1-p)^{n-1}$  değeri ise  $t_n$  ile  $t_{n+1}$  zamanları arasında çakışma olmama olasılığını vermektedir.

Bu değer  $n$  sonsuza giderken, yani çok fazla iletişim olduğunda,  $1/(2e)$  değerine yaklaşmaktadır o da 0.18 değerini verir.

### **SORU 38: Çoklamak (multiplexing)**

Basitçe birşeyin çoklanmasıdır. Örneğin bir hattı birden fazla kişinin kullanması hattın çoklanmasıdır. Devre tasarımında çoklayıcı (multiplexer) devrelerin tasarım temeli de budur.

Ağ dünyasında çoklamak ile kast edilen ise bir hattı birden fazla bilgisayarın veya birden fazla programın kullanmasıdır. Örneğin bir bilgisayarda bulunan birden fazla programın tek bir bağlantı üzerinden konuşmasına TCP veya UDP kullanılan ağlar için TCP çoklaması (TCP multiplexing) veya UDP çoklaması (UDP multiplexing) adı verilir. Bu durum TCP veya UDP kullanan ağlar için aynı IP numarası üzerinden ama farklı port numaralarını bağlayarak (port binding) yani farklı soketlerden olur. Bu durumda birden fazla program aslında aynı IP adresinden çıkan ve aynı IP numarasına sahip paketler ile internet üzerinde dolaşırlar. Benzer şekilde pek çok farklı paket de aynı bilgisayara gelirler, bu paketlerin dağıtımı gelmiş oldukları port numaralarına göre ilgili programa dağıtılır.

Bu işlem aynı adresteki farklı kişiler gibi düşünülebilir. Yani örneğin aynı adrese farklı mektuplar gelmekte fakat farklı kişilere ait olabilmektedir. Bu durumda mektupların geldiği adres dışında mektubun kime ait olduğuna da bakılır. Bu örnekte adres bilgisi IP adresine, kişi bilgisi ise port numarasına benzetilebilir. Bir paket aynı adreste farklı programlara gelebilmektedir.

Diğer bir çoklama örneği de IP çoklama (IP multiplexing) olabilir. IP çoklama ile kast edilen tek bir IP üzerinden birden fazla bilgisayarın bağlanmasıdır. Bu işlem NAT (network address

table, ağ adres tablosu) sayesinde yapılır. Bu yapıda, bir alt ağda (subnet) bulunan bilgisayarlar ortak bir kapıdan (gateway) internete bağlanmaktadır ve her çıkan bilgisayar için NAT üzerinde farklı bir kayıt tutulur. NAT yapısı port numaralarından faydalanmaktadır. Yani her bilgisayar için NAT üzerinde farklı bir port numarası atanmakta ve bu kapıya gelen her paket hangi port numarasına geldiyse o port numarasının arkasındaki bilgisayara iletilmektedir.

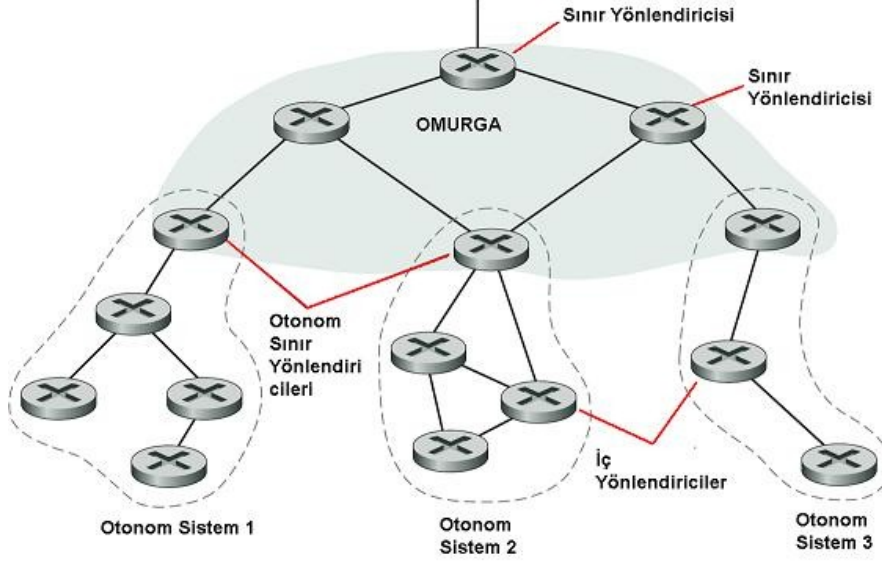
### **SORU 39: BGP (Sınır Kapısı Protokolü, Gümrük Protokolü, Border Gateway Protocol)**

Paket anahtarlama ağılarda kullanılan iki önemli yönlendirme protokolünden birisi mesafe vektörü (distance vector) diğeri ise bağlantı durumudur (link state). Mesafe vektörü yaklaşımında en kısa yolu bulmakta kullanılan bellman-ford algoritması uygulanmaktadır. Bağlantı durumu çözümünde ise asgari tarama ağacı benzeri bir arama algoritması olan Dijkstra algoritması kullanılmaktadır.

Bir mesafe vektörü uygulaması olan BGP protokolünün ana uygulama alanı otonom sistemler arası yönlendirmelerdir. Aslında tam bir mesafe vektörü olmayan BGP protokolü yapı olarak çok benzemektedir. Çünkü bu sistemde mesafe vektörüne benzer şekilde komşuların sürekli olarak haberdar edilmesi ve değişikliklerin bildirilmesi gerekir. BGP'nin çıkışı EGP protokolünün yerine alternatif olarak merkezî olmayan bir yönlendirme yapabilmektir. BGP genel olarak büyük ölçekli ağları bir araya getirmek için kullanılabilir, örneğin OSPF kullanan ağların BGP üzerinden birbirine bağlanması mümkündür. Genellikle otonom sistemleri birbirine bağlaması hasebiyle çoğu kullanıcı BGP ile doğrudan muhatab olmaz ancak hemen hemen her internet servis sağlayıcısı ana omurgada bir BGP bağlantısı kullanmak zorunda olduğu için BGP İnternetin en önemli protokollerinden birisi olarak sayılabilir.

BGP kurulumu el ile müdahaleyi gerektirir ve 179 porttan TCP ile bağlantı sağlar. BGP ayarlı bir yönlendirici sürekli olarak normal kurulumda her 60 saniyede bir 19byte uzunluğunda bir paket yollayarak komşularını haberdar eder. Diğer yönlendirme protokollerinden farklı olarak TCP protokolünü kullanan neredeyse tek yönlendirme protokolü BGP'dir. Otonom sistemleri bağlamanın dışında bir otonom sistem içerisinde de kullanılabilen BGP bu durumda IBGP ( İç sınır kapısı protokolü, Interior Border Gateway Protocol) ismini alır. Otonom sistemler arası bağlantı durumunda ise EBGP ( Dış sınır kapısı protokolü, Exterior Border Gateway Protocol) ismini alır.

Aşağıdaki grafikte bu iki duruma örnek yönlendirici de gösterilmiştir:

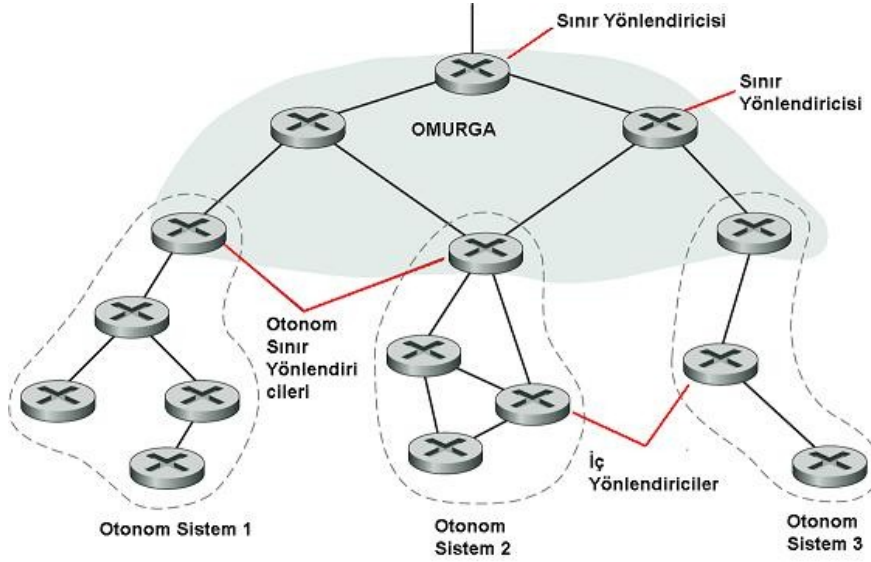


Otonom sistem seviyesinde döngülerin yakalanması da BGP'nin önemli fonksiyonlarından birisidir. Bu işlem sırasında BGP protokolü daha karmaşık hale gelmektedir ancak döngülerin sebep olduğu problemler düşünüldüğünde oldukça kullanışlı bir çözümdür. AS\_PATH özelliği olarak bilinen bu özellik ile paketin geçtiği otonom sistemlerin listesi tutulur. Şayet bu AS\_PATH listesinde kendi bilgisine rastlanırsa bu bir döngü olduğunu gösterir. Bu durumda yönlendirici AS\_PATH üzerindeki en kısa yolu tercih ederek diğer yoldan vaz geçer.

#### **SORU 40: Otonom Sistem (Autonomous System)**

Internet üzerinde aslında her yönlendiricinin bir yöneticisi veya sahibi bulunmaktadır. Bu yönlendiricilerin nasıl bir protok kullanacağından nasıl çalışacağına kadar her türlü kararı bu yönetici verebilir. İşte birden fazla yönlendiriciden oluşan bir grup tek bir yöneticinin idaresindeyse bu gruba otonom sistem adı verilmektedir. Bu yönlendirici grubunun içlerinde tek bir algoritma ile çalıştıkları kabul edilebilir (mesafe vektörü (distance vector) veya link state) bu durum ilk başta internetin yapısını basitleştirdiği için bir kolaylık olarak görülebilir ancak otonom sistemler arasında iletişim yapılması problemi doğmuştur.

Aşağıdaki şekili inceleyelim:



Yukarıdaki şekilde 3 farklı otonom sistem verilmiştir ve her sistemin yönetimi bağımsız bir yönetici eliyle yapılmaktadır. Dikkat edilirse yukarıdaki şekilden de anlaşılacağı üzere problem otonom sistemler arası bağlantıların nasıl sağlanacağıdır. Şekilde bu bağlantı, otonom sistemler dışındaki omurga üzerinden yapılmıştır ve sorun bu bağlantıda hangi protokolün nasıl kullanılacağıdır. Örneğin BGP4 protokolü bu tarz problemler için geliştirilmiş melez bir protokoldür (otonom sistemler arası yönlendirme protokolü (inter autonomous system routing protocol)).

#### **SORU 41: Ters Zehir (Reverse Poison)**

Mesafe vektörü (Distance Vector) üzerinde çalışan bir döngü engelleyici algoritmadır. Yani ağda fasit daireye (kısır döngüye) girme ihtimalini kaldırmayı hedefler.

Çalışma mantığı dairesel bir ağda, bir döngünün yakalanması üzerine döngüyü kırmak için düğümlerden birisinin mesafe olarak doğru olmadığı halde diğer düğüme ulaşmanın sonsuz olduğunu yani ulaşamadığını beyan eder. Bu yalan beyan ve ağda ulaşılabilir olmasına rağmen ulaşamaz olarak işaretlenmesi ters zehir anlamındadır.

Bu yaklaşımın iki önemli sebebi vardır. Birincisi ağın ölçeğidir. Yani çok büyük ağlar üzerinde yönlendirici tablolarının güncellenmesi zor hale gelmektedir. Diğer sebebi ise ağın yönetilebilirliğidir. Yani bir yönetici ağını istediği gibi yönetebilmelidir. Oysaki bu yaklaşımda ağın otomatik olarak yönetilmesi söz konusudur.

#### **SORU 42: Mesafe Vektörü (Distance Vector)**

yönlendiriciler (routers) üzerinde çalışan iki önemli yönlendirme algoritmasından birisidir. Diğer önemli algoritma ise bağlantı durumu (link state) algoritmasıdır.

Amaçyönlendirici üzerine gelen bir paketin hangi hedefe gittiğine bakılarak hedefe en kısa gidebileceği bir sonraki noktaya yönlendirilmesidir.

Mesafe Vektörünü kullanan en yaygın protokoller RIPv1 veya 2 , IGRP, EGP ve BGP protokolleridir.

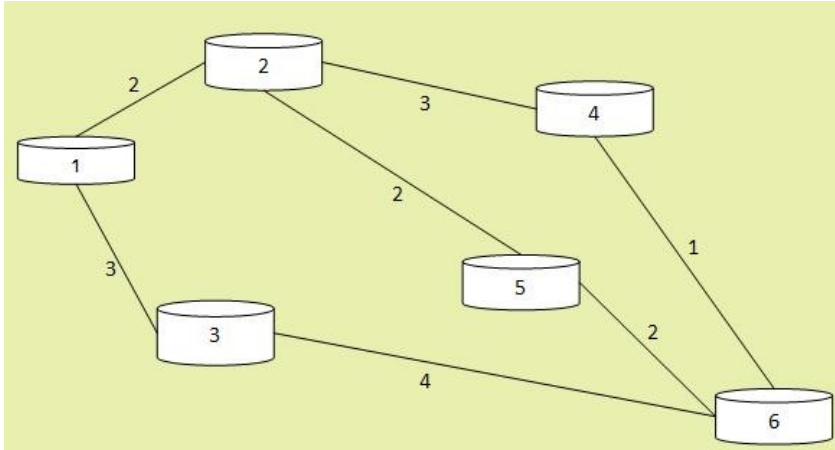
Bu algoritmada, ağda bulunan diğer yönlendiricilerin ağ hakkında bilgi edindirilmesi hedeflenmiştir. Yani bir yönlendirici bağlı bulunduğu ağlar hakkındaki bilgiyi diğer yönlendiricilere periyodik olarak yollamaktadır. Bağlantı durumu (link state) algoritması ile karşılaştırıldığında hesaplanması ve yönetilmesi daha basit olan ancak ağ üzerinde daha fazla trafik oluşturan bu algoritmanın çalışma mantığı aşağıda verilmiştir:

Öncelikle ağ hakkında bilgi edinilerek ağda bulunan diğer noktalar ile mesafe hesaplanır. Örneğin RIP protokolü nokta sayısını kullanırken IGRP ağdaki gecikme veya bant genişliği gibi bilgileri kullanır. Bir kez ağ tanındıktan sonra yönlendirici bu ağ bilgisini ağda bulunan diğer yönlendiricilere gönderir ve diğer yönlendiriciden benzer bilgileri alır. Mesafe vektörü algoritmasına dedikodu algoritması denilmesinin bir sebebi de her yönlendiricinin ağdaki diğer yönlendiriciden aldığı bilginin doğruluğuna güvenmesi ve bu bilgiyi teyid etme ihtiyacı duymamasıdır. Alınan bilgileri kendi yönlendirme tablosuna yazan yönlendirici bundan sonraki paket yönlendirmelerini bu bilgilere göre yapmaktadır.

Aşağıda bu uygulamanın bir örneği verilmiştir: Aşağıda bir ağ yapısı verilmiştir, her düğümün bir numarayla gösterildiği ağda aşağıdaki tabloda hangi noktadan hangi noktaya ne kadar maliyetle gidildiği verilmiştir:

1	2	2
1	3	3
2	4	3
2	5	2
3	4	1
3	6	4
4	6	1
5	6	2

bu noktaların çizilmiş hali aşağıdadır:



Yukarıdaki bu ağda dikkat edilirse yollar yönsüzdür. Yani bir yoldan geçmenin maliyetiyle tersinden geçmek arasında mesafe olarak fark yoktur. Bu algoritma tersinin de farklı olması durumunda çalışabilmektedir.

Yukarıdaki bu ağ üzerinde çıkarılmış olan her düğüm için yönlendirme tabloları aşağıda verilmiştir. Algoritma ilk başta doğrudan komşuluğu olan noktaları tablolarına işaretlemektedirler. Tabloların yapısı, hangi düğümden hangi düğüme ne kadar maliyet olduğudur.

1.	noktanın	yönlendirme	tablosu:
1		1	0
2		2	2
3		3	3
4		0	1000
5		0	1000
6 0 1000			

2.	noktanın	yönlendirme	tablosu:
1		1	2
2		2	0
3		0	1000
4		4	3
5		5	2
6 0 1000			

3.	noktanın	yönlendirme	tablosu:
1		1	3
2		0	1000
3		3	0
4		4	1
5		0	1000
6 6 4			

4.	noktanın	yönlendirme	tablosu:
1		0	1000
2		2	3
3		3	1
4		4	0
5		0	1000
6 6 1			

5.	noktanın	yönlendirme	tablosu:
1		0	1000
2		2	2
3		0	1000
4		0	1000
5		5	0
6 6 2			

6.	noktanın	yönlendirme	tablosu:
1		0	1000
2		0	1000
3		3	4
4		4	1
5		5	2
6 6 0			

Yukarıdaki tabloda 1000 olarak verilen sayılar ulaşılması imkansız olan noktalardır. Yani bu noktalara ulaşım bulunmamaktadır. Formal olarak algoritmanın tanımında bu noktaların değeri sonsuz olarak verilmektedir.

Bu ilk değerler atandıktan sonra, ağdaki diğer yönlendiriciler ile iletişime geçilerek yönlendirme tabloları güncellenmiş ve aşağıdaki sonuçlar elde edilmiştir.

1.	noktanın	yönlendirme	tablosu:
1		1	0
2		2	2
3		3	3
4		3	4
5		2	4
6 3 5			

2.	noktanın	yönlendirme	tablosu:
1		1	2
2		2	0
3		4	4
4		4	3
5		5	2
6 4 4			

3.	noktanın	yönlendirme	tablosu:
1		1	3
2		4	4
3		3	0
4		4	1
5		4	4
6 4 2			

4.	noktanın	yönlendirme	tablosu:
1		3	4
2		2	3
3		3	1
4		4	0
5		6	3
6 6 1			

5.	noktanın	yönlendirme	tablosu:
1		2	4
2		2	2
3		6	4
4		6	3
5		5	0
6 6 2			

6.	noktanın	yönlendirme	tablosu:
1		4	5
2		4	4



3	4	2
4	4	1
5	5	2
6 6 0		

### **SORU 43: CSMA ( Çoklu Erişimde Hat Kontrolü, Carrier Sense Multiple Access)**

Bu yöntem bir hatta birden fazla erişim olması durumunda kullanılmaktadır. Yaklaşım hatta bir paket veya sinyal konulmadan önce hattın müsait olup olmadığının kontrolüne dayanır. Bu yöntem, switch cihazlarının kullanımının artmasıyla önemini yitirmiştir çünkü switch iletişime geçen iki uç arasında sadece bu iki uca has bir bağlantı kurabilmektedir. Güncel olarak IEEE 802.3 ve Ethernet uygulamalarında kullanılmaktadır. Gigabit Ethernet ile kullanımdan kalkmıştır.

Bu yöntem kullanılırken aynı ortamda sinyal iletimi olduğu için sinyallerin çakışması (collision) ihtimali bulunmaktadır. Bu ihtimali engellemek için aşağıdaki çözümler önerilmiştir, unutulmamalıdır ki , CSMA yönteminde çakışmayı engellemek imkansızdır. Bunun en basit sebebi, örneğin aynı anda boş olan bir hatta iki farklı bilgisayar veri koymak isterse ikisi de hattı boş kabul edeceği için hatta koyacaklar ve bu veriler hatta çakışacaktır.

CSMA/CA (Çoklu Erişimde Hat Kontrolü / Çakışma Mümkün, Carrier Sense Multipla Access / Collision Avoidance) : Çakışma için en basit yaklaşımdır. Hattı kullanacak olan herkes, hattı dinler ve şayet boş olduğunu görürlerse hatta verilerini koyarlar. Çakışma ihtimali vardır ve CSMA'ın kendisi ile sonuçları aynıdır. CSMA/CD (Çoklu Erişimde Hat Kontrolü / Çakışma Tesbit, Carrier Sense Multipla Access / Collision Detection) : Çakışmayı engellemeyi hedefler. Hattı kullanan taraflar, bir çakışma olup olmadığını kontrol ederler ve çakışmayı algıladıkları anda iletimi durdururlar. Tekrar iletim için herkes rasgele (random) bir miktar bekler ve tekrar hattı kullanmaya başlarlar (tabi kullanmadan önce yine hattı başkasının kullanıp kullanmadığını kontrol ederek). Çakışmaya sebep olacak tarafların rasgele sayı tutması sayesinde tekrar aynı anda başlama ihtimallerinin az olması mantığına dayanarak çakışma engellenmiş olur.

CSMA/CR (Çoklu Erişimde Hat Kontrolü / Çakışma Çözüm, Carrier Sense Multipla Access / Collision Resolution) : Çakışmayı engellemeyi hedefler. Aynı zamanda CSMA/BA (bitwise arbitration, ikilik çözümleme) de denilmektedir. Hattı kullanan herkes için birer sayı veya öncelik numarası atanmıştır. Hattı kullanan taraflardan kaynaklanan bir çakışma olduğunda, sahip olunan bu numaraya göre veri gönderme önceliği verilerek önceliği düşük olan tarafın bekletilmesini söyler.

### **SORU 44: CRC (cyclic redundancy check, çevrimsel fazlalık sınaması)**

Hata algılama yaklaşımlarından birisidir. Bu yöntemde işlenmekte olan verinin dışında ilave bir kontrol verisi daha bulunur. Bu ilave bilgi ile bütün bilgi kontrol edilerek bilgide bir bozulma olup olmadığı kontrol edilir. Örneğin ağ iletişiminde gidip gelen bilginin kontrol edilmesinde veya CD gibi kayıt ortamlarında verinin bozulup bozulmadığının kontrol edilmesinde kullanılır.

#### **Çalışması:**

Her iki tarafın da bildiği bir sayı bölen olarak kullanılır. Örneğin bölen sayımız 13 olsun ve bunu her iki tarafta başlangıçta biliyor olsun. (Bu sayı standarta bağlıdır lütfen standartların

anlatıldığı  
Gidecek  
19  
CRC  
CRC hesaplanırken daha önceden bildiğimiz sayıya bu sonuç bölünür ve kalan alınır: 250 %  
13  
Dolayısıyla yukarıdaki veriler yollanırken CRC olarak 3 sayısı yollanmaktadır.  
Veri alındıktan sonra kontrol edilmesi:  
yukarıdaki veriler alındıktan sonra alan taraf da verileri toplar, bu toplamdan CRC bilgisini çıkarır ve  $250-3=247$  sayısını bulur. Daha önceden bildiği 13 sayısına böler ve 0 sonucunu bulursa iletim hatasızdır yargısına varır  $247 \% 13 = 0$   
şayet hata olsaydı sonuç 0'dan farklı çıkardır. Burada görüldüğü üzere CRC'nin de hata yapma ihtimali vardır örneğin veri bozulması verinin ilk kısmının 1 fazla olması şeklindeyse:  
20 54 89 22 03 44 19  
verisi alındığında bu hata algılanır :  $251 - 3 = 248 \% 13 = 1$  ve sonuç 0 olmadığı için hata kararına varılacaktı ancak hata miktarı daha önceden bilinen sayının (ki bu örnekte 13) katı şeklinde olursa hatanın yakalanması imkansızdır. Örneğin veri bozulması sonucu:  
32 54 89 22 03 44 19  
sayıları çıkmış olsun. Bu sayıların toplamı 263 olacak ve CRC kısmı olan 3 değeri çıktıktan sonra  $260 \% 13 = 0$  olacaktır. Görüldüğü üzere orjinal verimizden farklı olmasına rağmen hatasız olarak kabul edilmiştir.

#### **SORU 45: Internet Toplam Kontrolü (Internet Checksum)**

İnternet üzerinde iletilen paketlerin bozulup bozulmadığının kontrolünüdür. Bu yöntem paketin veri kısmındaki bilgileri toplar ve elde edilen bilgiyi ayrıca yollar. Alan kişi de veriyi toplayarak aynı kontrol toplamını (check sum) elde edip etmediğine bakar.

Bu bozulup bozulmama kontrolünü bilindiği üzere TCP protokolü kullanmaktadır.

Örneğin:

Gönderilecek veri : 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 olarak verilmiş olsun bu veri 32 bit uzunluğundadır ve bizim paketlerimiz 16 bit uzunluğunda kabul edelim. Dolayısıyla veriyi 16 bitlik parçalara bölüp toplayacağız.

1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	
+																															

1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 (toplam değeri ve 17 bit uzunluğunda dolayısıyla 17. bitteki fazla olan veri ile kendisini tekrar topluyoruz)

1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0 (verinin 17. taşan bit ile toplanmış hali)

0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1 (son olarak verinin tersini alıyoruz, yani 1 olan bitleri 0 ve 0 olan bitleri 1 yapıyoruz)

Yukarıdaki son satırda elde edilen değer 32 bitlik orjinal paketin toplam kontrol ( checksum ) değeridir.

bilgi yollanırken bu bilgi de ilave olarak pakete konulmakta ve karşı taraf aldığı zaman paketdeki verilere aynı işlemleri uygulayarak sonucun doğru olup olmadığını kontrol etmektedir.

#### **SORU 46: Ortam Erişim Kontrolü (Media Access Control)**

OSI katmanlarına göre 2. katman olan veri bağlama katmanı (data link layer) protokolüdür. Temel görevi aynı ağda bulunan ve birbiri ile temas halindeki (örneğin bir LAN'da veya MAN'da çalışan) bilgisayarların veri iletişimini sağlamaktır. Kısaca MAC adı da verilen bu kontrol aynı zamanda bilgisayarların üzerindeki ortama erişirken kullanılan cihazlar için bir de adresleme yapmaktadır. Bu adrese MAC adresi adı verilmektedir ve dünya üzerinde sadece tek bir cihaz (NIC, Network Interface Card, Ağ bağlantı Kartı) üzerinde olabilecek bu numara IEEE tarafından dağıtılmaktadır.

Ortam erişim kontrolünün diğer bir görevi ağ üzerine aynı anda erişmekte olan birden fazla bilgisayar arasındaki uyumu sağlamaktır. Bu erişim kontrolünü paket anahtarlamalı ağlarda çözen çözümlerin en meşhurları aşağıdadır:

- \* CSMA/CD (Carrier Sense Multiple Access / Collision Detect, ) Ethernet (gigabit öncesi) ve IEEE 802.3 ağlarda kullanılır,
- \* Token bus (Jeton Sıralı) (IEEE 802.4)
- \* Token ring (Jeton Göngüsü) (IEEE 802.5)
- \* Token passing (Jeton Geçirmeli) (FDDI ağlarda kullanılır).

MAC Adresleri IP veya IPX gibi, host (bilgisayar) ve ağ (network) olarak parçalara ayrılmamıştır. Dolayısıyla bir MAC adresine bakarak bir bilgisayar ile aynı ağda olup olunulmadığı anlaşılamaz. Bir MAC adresi 48 bittten oluşur ve onaltılık (hexadecimal) tabanda 6 adet ikili olarak ifade edilir. Örneğin 00-00-00-00-00-00 şeklinde gösterilir.

#### **SORU 47: Internet Katman Kümesi (Internet Layer Stack)**

Internet üzerinde kullanılan ağ katmanlarını ifade eder. Internet üzerinde gidip gelen paketler 4 farklı katmana bölünürler ve her katmanda farklı bir protokol konuşur. Yanlış olmasına karşılık çok fazla kullanıldığı için bu tanıma TCP/IP modeli de denilmektedir.

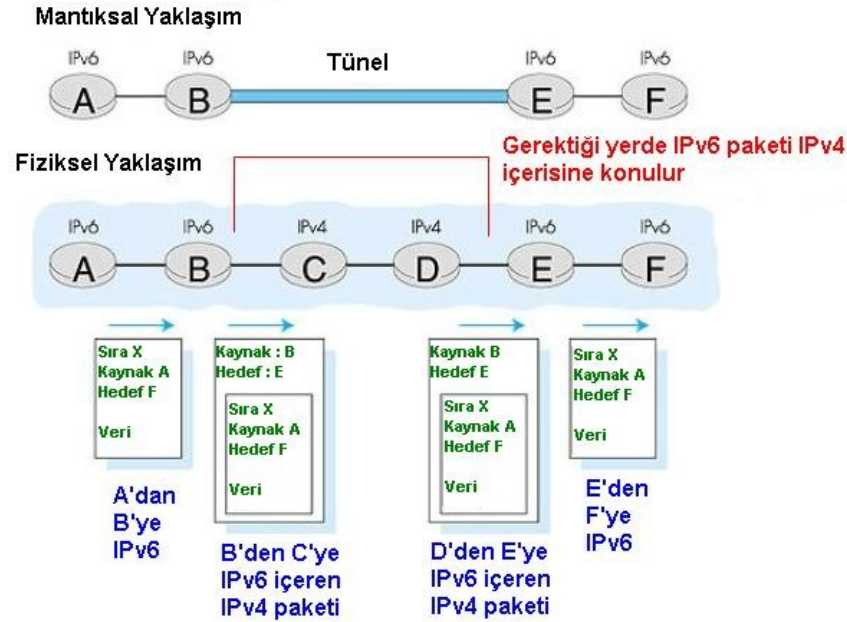
Internet Katman Kümesinde Aşağıdaki katmanlar Bulunmaktadır , her katmanın yanında bu katmanda çalışan örnek protokol isimleri de verilmiştir:

4. Uygulama Katmanı (application layer) | FTP, HTTP, Telnet, SMTP, POP3
3. İletim Katmanı (Transport Layer) | TCP , UDP
2. Ağ Katmanı (Network Layer) | IP
2. Veri Bağlama Katmanı (Data Link Layer) : Bu katman IP altında çalışmaktadır ve birlikte tek bir katman gibi düşünülebilir. | ARP
1. Fiziksel Katman (Physical Layer) | Ethernet , Wi-Fi

Yukarıdaki ikinci katmanın ayrılıp ayrılmaması kaynaktan kaynağa değişiklik göstermektedir.

## SORU 48: IP Tünelleme (IP Tunnelling)

Bir IP paketinin içerisinde farklı bir protokolün bilgisinin yollanması anlamına gelir. Örneğin IPv4 paketinin içerisinde IPv6 paketi yollanabilir. Bu yaklaşıma göre IPv4 ile konuşan uçlar arasında yollanan paketin içeriği açılmaz ve bilinmez, bu paket veri gibi kabul edilir. IPv4 üzerinden IPv6 yollayan uçlar ise birbirlerinin paketlerini IPv4 ağından alarak anlayabilirler. Bu durum aşağıdaki temsili resimde gösterilmiştir:



Yukarıdaki resimde, iki IPv6 konuşan nokta daha altta, IPv4 konuşan noktalar üzerinden konuşmaktadır. Ancak bu durumdan haberi olan B ve E noktaları paketlerinin IPv4 paketlerinin içerisine koyarak karşı tarafa yollamakta ve gelen paketi de açarak bir IPv6 paketi olarak kullanılmaktadırlar. Bu sayede çift küme (dual-stack) yaklaşımındakine benzer bir veri kaybı olmaksızın veri karşı tarafa ulaşmakta ve paket yoluna devam etmektedir.

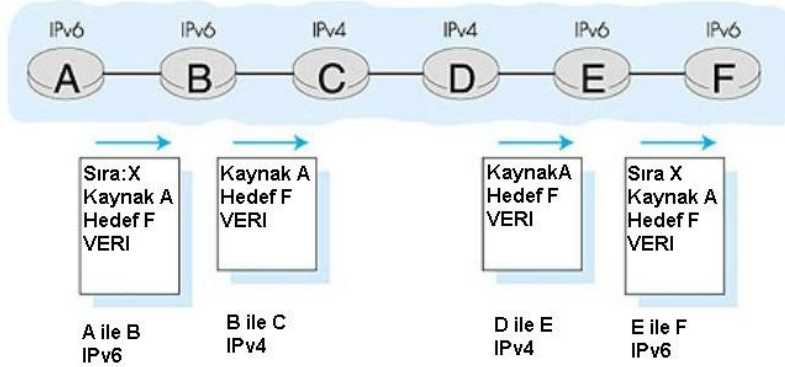
## SORU 49: Çift Küme (çift yığıt, Dual-Stack)

Bir ağ içerisinde aynı anda birden fazla protokol kümesinin (ailesinin,yığıtının ) kullanılması durumudur. Buna göre örneğin aynı ağ içerisinde IPv4 ile IPv6 beraber çalışabilmektedir.

Bu yaklaşıma göre IPv4'den IPv6 geçişi ( Transition from IPv4 to IPv6) işleminin problemsiz olarak yapılması hedeflenmektedir. Bu yaklaşımda bazı noktaların her iki protokol kümesini de konuşabilmesi gerekmektedir. Bu çift kümeye hitâb eden noktalarda bir aileden gelen paketler diğer aileye çevrilebilmektedir ve bu noktalar karşı tarafın konuştuğu protokolü anlayarak bundan sonraki konuşmalarını bu protokol üzerinden yaparlar.

Karşı tarafın hangi protokol kümesinden olduğunun anlaşılması için örneğin DNS'den faydalanılabilir. Buna göre örneğin DNS üzerinde yapılan bir sorgu sonucunda adresin karşılığı olan IP adresi v6 olarak destekleniyorsa cevap da IPv6 olarak döndürülür. Şayet desteklenmiyorsa IPv4 olarak cevap dönülür.

Aşağıda birden fazla ağda birden fazla protokol kümesinin kullanılması durumunda paketlerin geçişleri ve geçişler sırasında hangi kümeye ait oldukları verilmiştir:



Yukarıdaki grafikte her noktadan bir sonraki noktaya geçiş sırasında paketin içeriği gösterilmiştir. Dikkat edilirse ortamlardan birisinin IPv4 olması durumunda iletişim IPv4 üzerinden yapılmakta, her ikisinin de IPv6'yı desteklemesi durumunda iletişim IPv6 üzerine taşınmaktadır. Hem IPv4 hem de IPv6 için ortak olan hedef ve kaynak bilgileri paketlerde kopyalanırken, IPv6'ya özgü olan sıra bilgisi ne yazık ki IPv4 ağ içerisinde paketler geçtikten sonra kaybedilmektedir.

#### **SORU 50: IPv4'den IPv6 geçişi ( Transition from IPv4 to IPv6)**

IP (internet protocol) için günümüzde kullanılan dördüncü versiyondan altıncı versiyona geçiş için alternatif yollar araştırılmış ve çözüm olarak çeşitli öneriler bulunmuştur. Bunlardan birincisi çift küme (dual-stack) adında aynı anda iki protokolün de çalıştığı bir ağ yapısı geliştirmektir. Diğer alternatif yollardan birisi ise IP-Tünelleme (IP Tunneling) kullanmaktır.

#### **SORU 51: Geçiş Günü (Flag Day)**

IPv4'ten IPv6'ya geçiş için kullanılan bir terimdir. Buna göre bütün cihazların ve İnternet dünyasının aynı anda IPv4'ten IPv6'ya geçmesi hedeflenmiştir. Bu sayede hiçbir bilgisayar IPv4 çalıştırmayacağı için karmaşa olmayacak ve geçiş sorunsuz olarak tamamlanmış olacaktır. Ne yazık ki İnternetin homojen olmayan yapısı yüzünden ve maliyet sorunlarından dolayı bütün internetin bir anda bu geçişi yapması olası görülmemektedir.

#### **SORU 52: Protokol Kümesi (Protocol Stack , Protokol Yığıtı)**

Bir ağda çalışan bir yazılımın kullandığı protokol grubudur. Genellikle her protokolün katmanlı olmasından kaynaklanan, her protokol için birden fazla alt protokol bulunması durumu için zaman içerisinde gelişmiş bir terimdir. Buna göre örneğin OSI modeline bir protokol kümesi denilebilir. Aslında OSI modelinin içerisinde 7 farklı protokol (7 farklı katman) birbirini ile iletişim içerisinde.

Benzer şekilde TCP/IP protokol kümesinde 5 farklı protokol birbirini ile iletişim halindedir.

#### **SORU 53: Sanal Devre (virtual circuit)**

Bilgisayar ağlarından, paket anahtarlama ağ yapısından bir tanesidir. Diğer ağ yapısı datagram ağlardır. Sanal devrede bir ağ üzerinde iki nokta arasında sanki bir hat döşenmiş gibi bir devre kurulur. Yani aslında fiziksel olarak bir hat ayrımı veya daha basit anlamda iki uç arasında bir kablo döşenmesi söz konusu değildir. Ancak paketlerin hep aynı yolu izlemesi sayesinde sanki bir hat döşenmiş gibi bir ağ elde edilir. Bu yapının kurulması için yol

üzerinde bulunan bütün yönlendiriciler (router) ayarlanarak bu ağ için her paketin aynı yönde yönlendirilmesi sağlanmaktadır.

Datagram ağlarda, sanal devreden farklı olarak paketler duruma göre farklı yollar izleyebilirler. Özellikle tıkanıklık önlenme ve mesafe hesaplamalarına göre yönlendiriciler (router) paket yönlendirmesini datagram ağlarda değiştirirler. Ancak sanal devre uygulamasında bu mümkün değildir.

Sanal devre uygulanan ağlarda paketlerin yönlendirilmesi ve takibi açısından her alt ağda (subnetwork) paketlere bir sanal devre numarası (VC number) atanmakta ve yönlendiriciler (router) bu numaralara göre yönlendirme yapmaktadırlar.

#### **SORU 54: Gönderme Gecikmesi (transmission delay)**

Hat kapasitesine bağlı olarak bir uçtan diğer uca verinin gönderilme süresidir. Aşağıdaki ağ yapısında iki bilgisayarın basit bir bağlantı ile birbirine bağlı olduğunu düşünelim.

Yukarıdaki ağda hat kapasitesi (bant genişliği veya band width)  $R$  olarak kabul edilsin. Hat kapasitesinin birimi genelde KB/s (kilobit per second, saniyede gönderilen kilobit. kbps olarak da yazılır. Veya mbps , megabit per second (saniyede 1 milyon bit)) olarak gösterilebilir.

Bu kapasite kullanılarak  $L$  uzunluğunda bir bilginin karşı tarafa yollanacağını kabul edelim. Ver birimi benzer şekilde kilobit ,megabit veya kilobyte, megabyte olabilir (bir byte 8 bittir)

Gönderim süresi genelde saniye cinsinden hesaplanır. Birimleri bölecek olursak mbps / megabit = second (saniye) olarak bulunur.

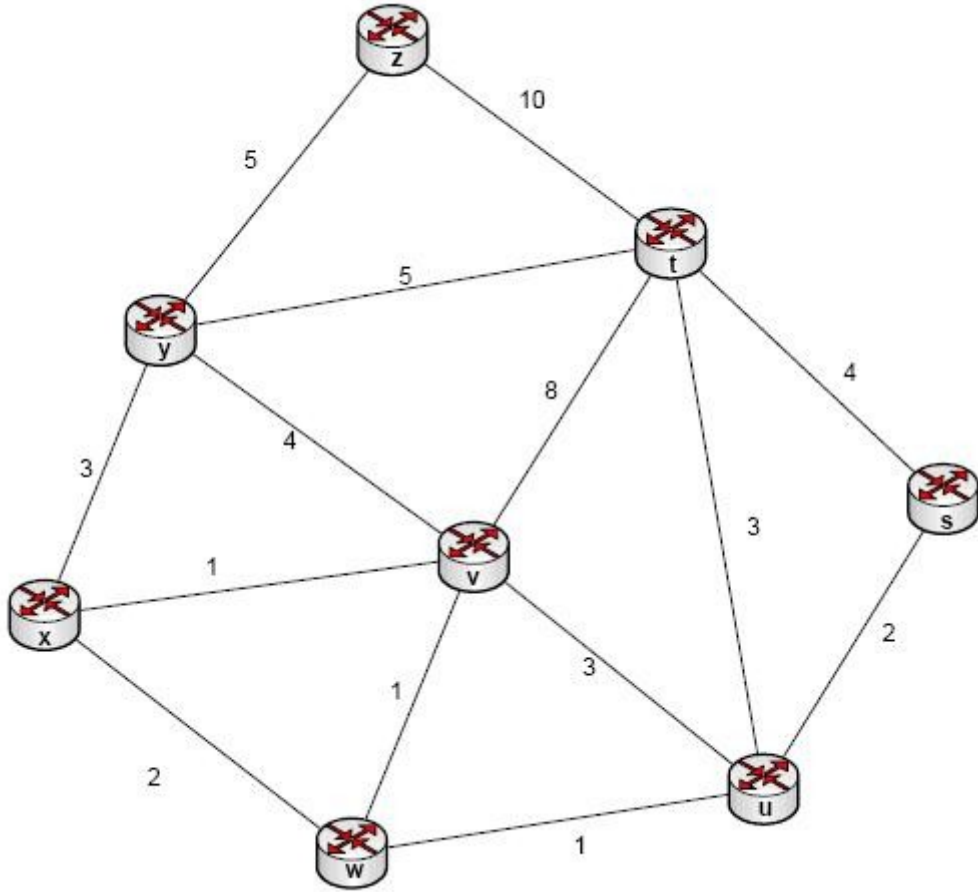
Gönderme gecikmesinin formülü yukarıdaki değişkenlere göre  $L/R$  olarak kabul edilebilir.

#### **SORU 55: alt ağ (subnetwork)**

IP (internet protokol) kullanılan Paket anahtarlama ağlarda, her ağ parçası için mantıksal olarak atanan ağ adresidir.

#### **SORU 56: Kruskal Asgari Tarama Ağacı Algoritması**

Bir asgari tarama ağacı (minimum spanning tree) algoritması olan Dijkstra algoritması, işaretlemiş olduğu komşuluklara en yakın düğümü bünyesine katarak ilerler. Buna göre aşağıdaki grafiğin asgari tarama ağacını çıkaralım:



Yukarıdaki grafikte her düğüm için bir temsili harf ve her bağlantı için bir ağırlık değeri atanmıştır. Buna göre her düğümünden diğerine gitmenin maliyeti belirlenmiştir.

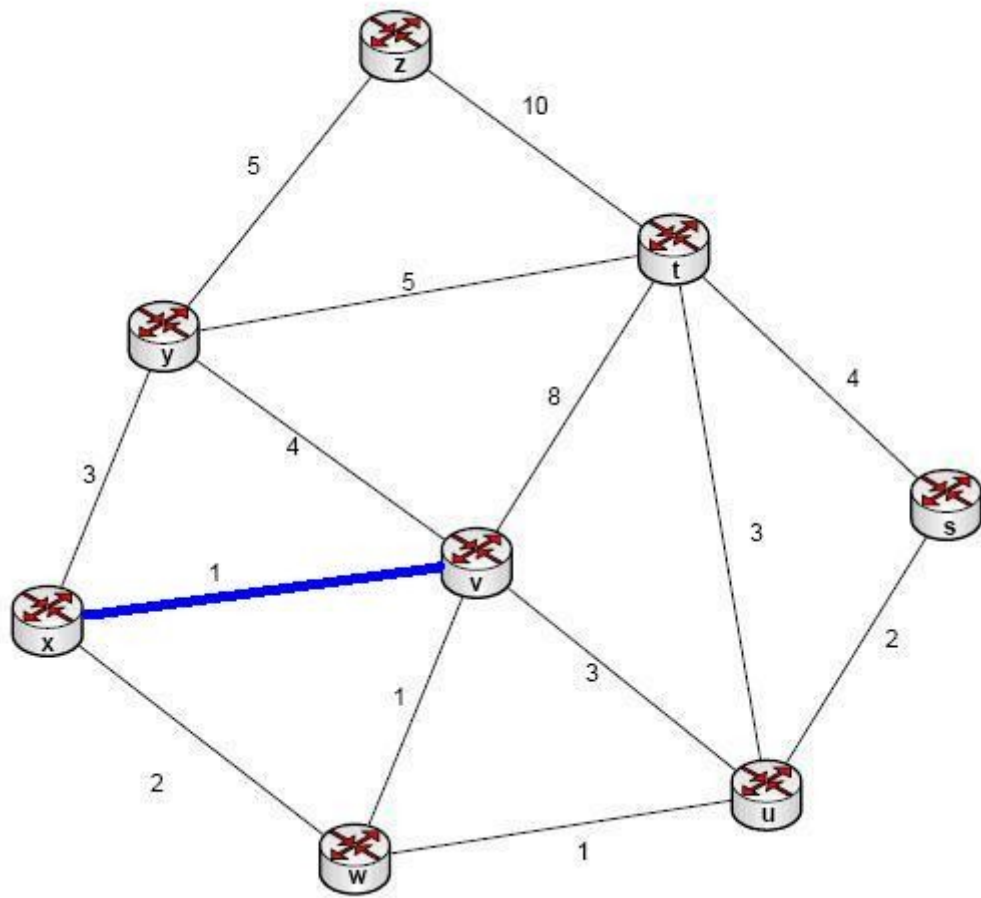
Kruskal algoritmasında bütün yollar listelenip küçükten büyüğe doğru sıralanır. Bu liste yukarıdaki grafik için aşağıda verilmiştir:

x-v:1  
 w-v:1  
 w-u:1  
 x-w:2  
 u-s:2  
 x-y:3  
 t-u:3  
 u-v:3  
 y-v:4  
 s-t:4  
 y-z:5  
 y-t:5  
 z-t:10

Yukarıdaki liste çıkarıldıktan sonra sırasıyla en küçükten en büyüğe doğru komşuluklar işaretlenir. Bu işaretleme sırasında ada grupları ve grupların birbiri ile ilişkisine dikkat edilir. Yani şayet listedeki iki düğüm harfi de aynı adadan ise bu bağlantı atlanır. Aşağıda sırasıyla bu grafikteki adaların oluşması ve asgari tarama ağacının çıkarılması gösterilmiştir:

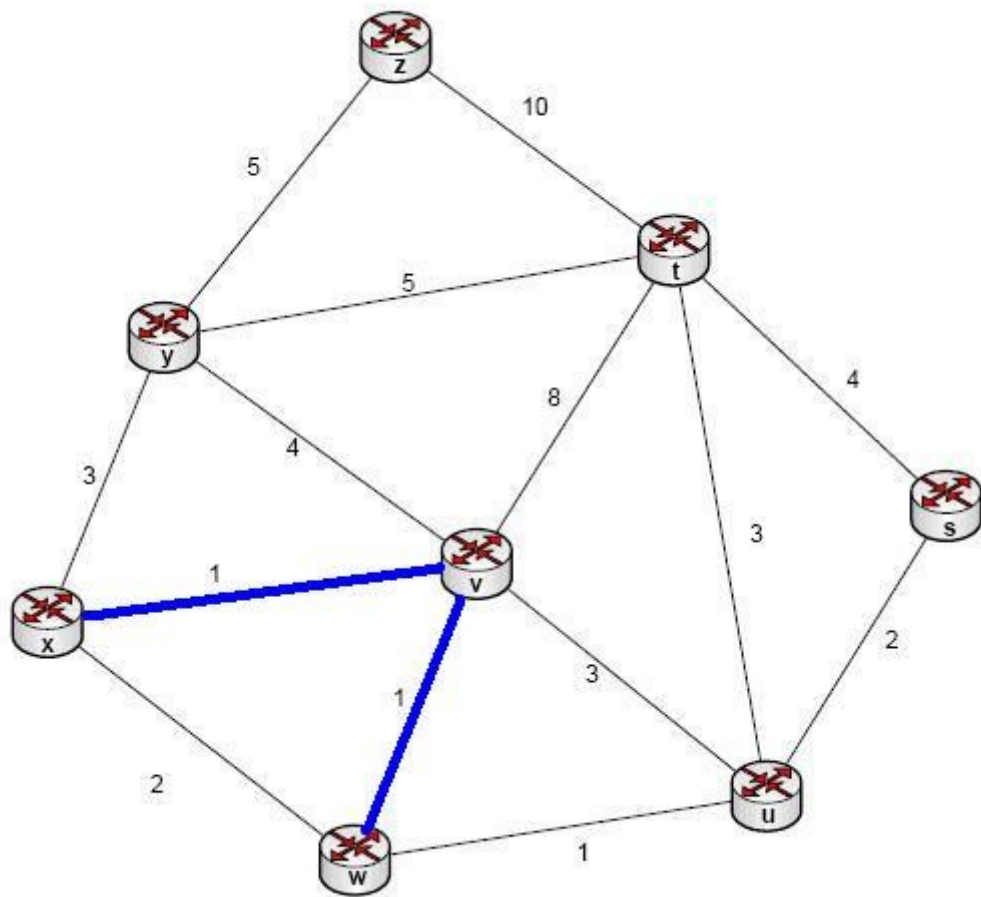


x-v:1

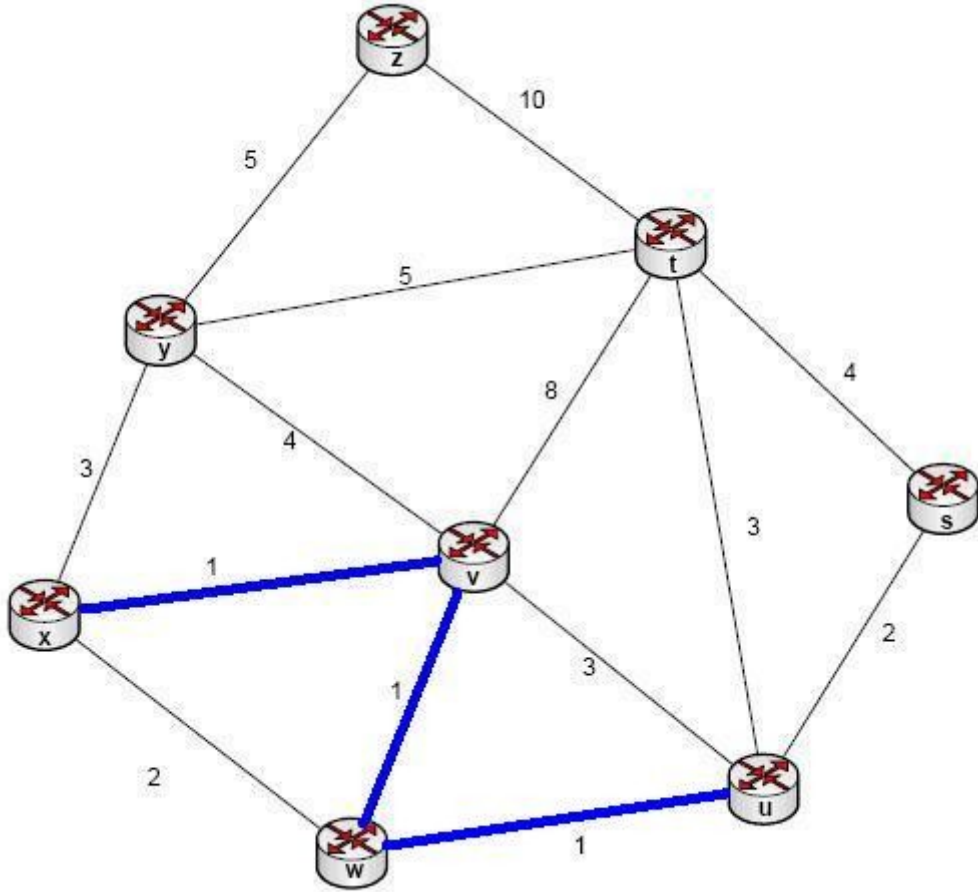


w-v:1

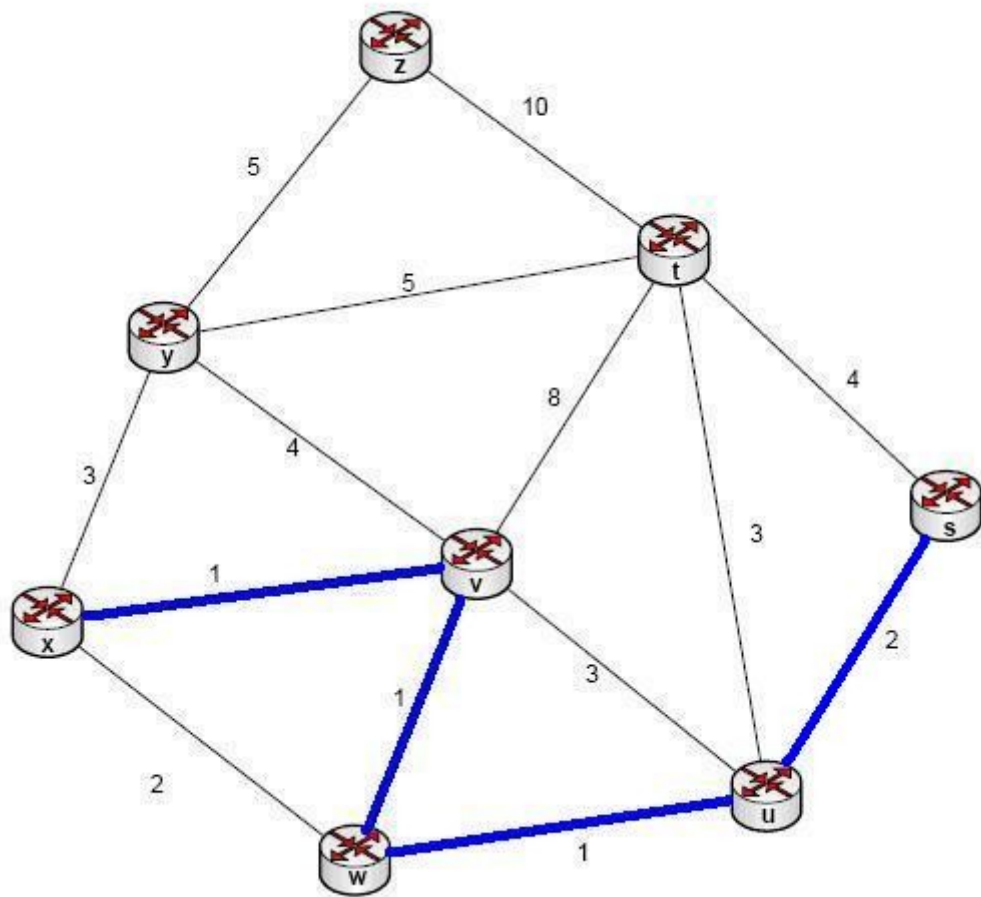




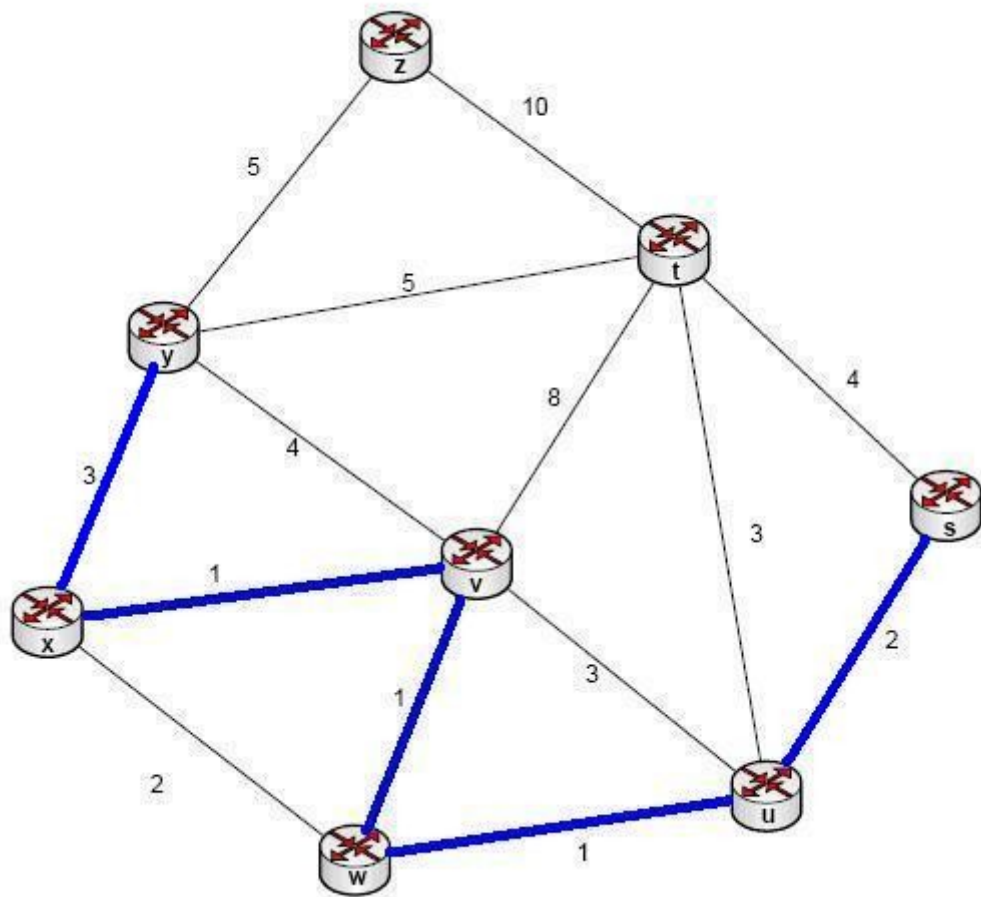
w-u:1



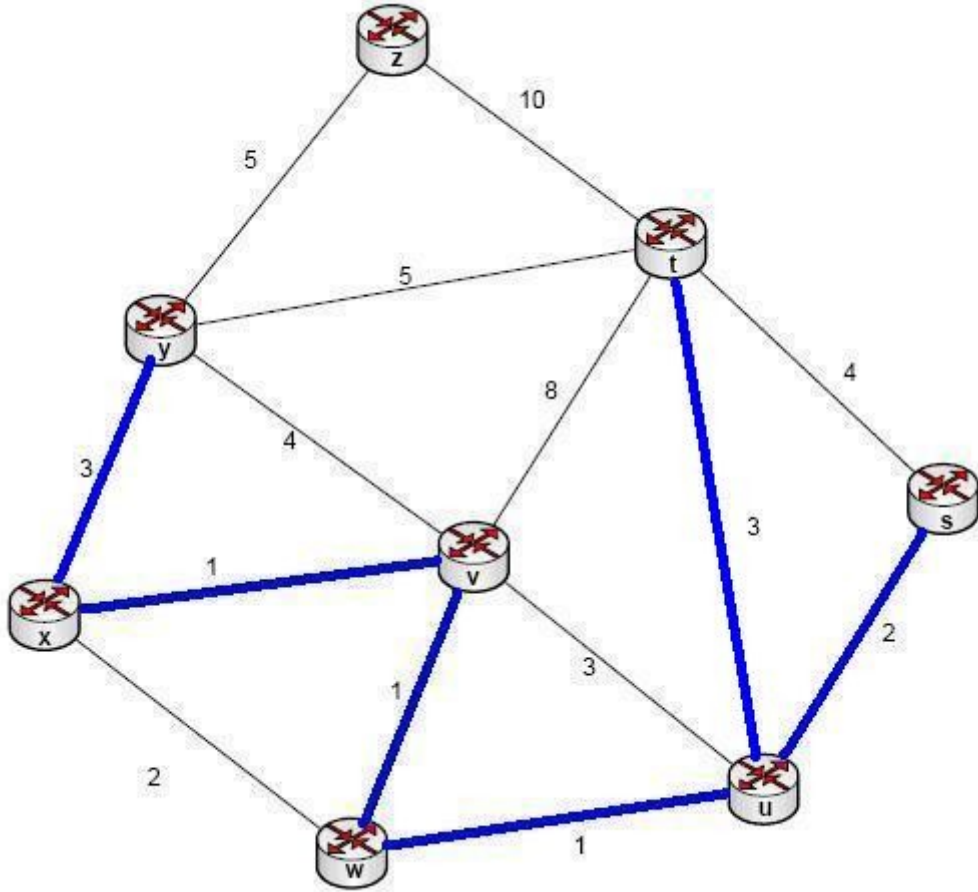
x-w bağlantısı atlanır, çünkü iki düğüm de zaten dolaşmıştır bunun yerine u-s:2 bağlantısına atlanır.



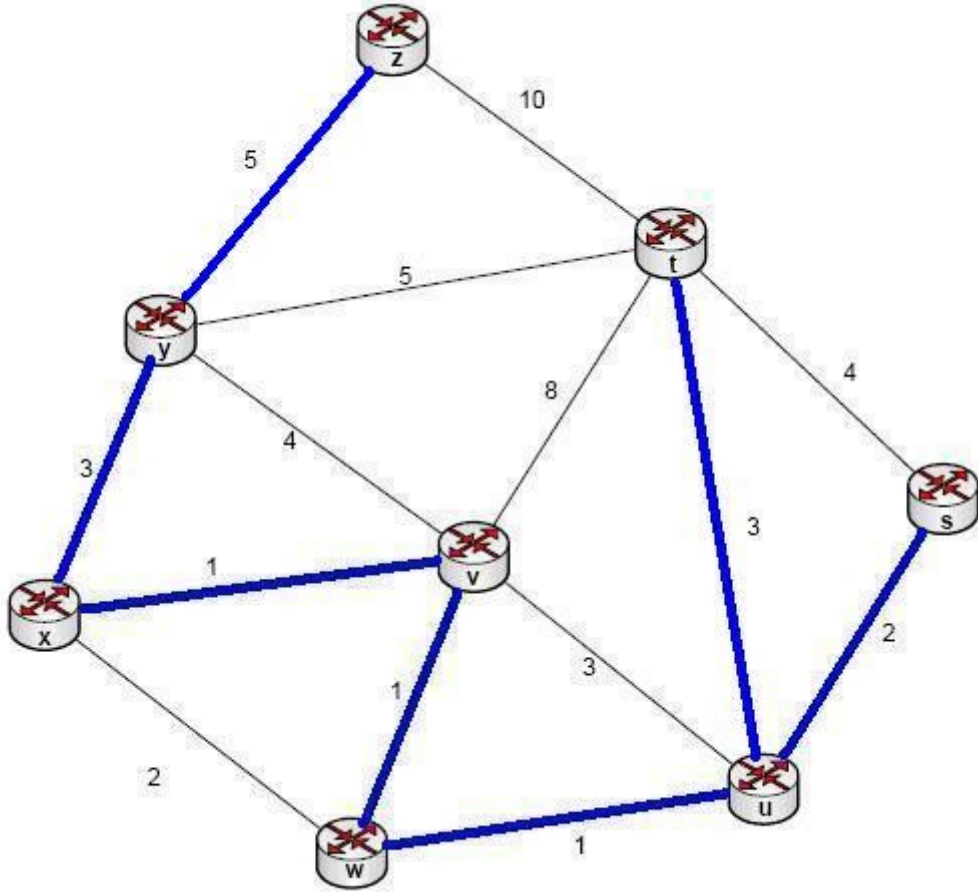
x-y:3



t-u:3



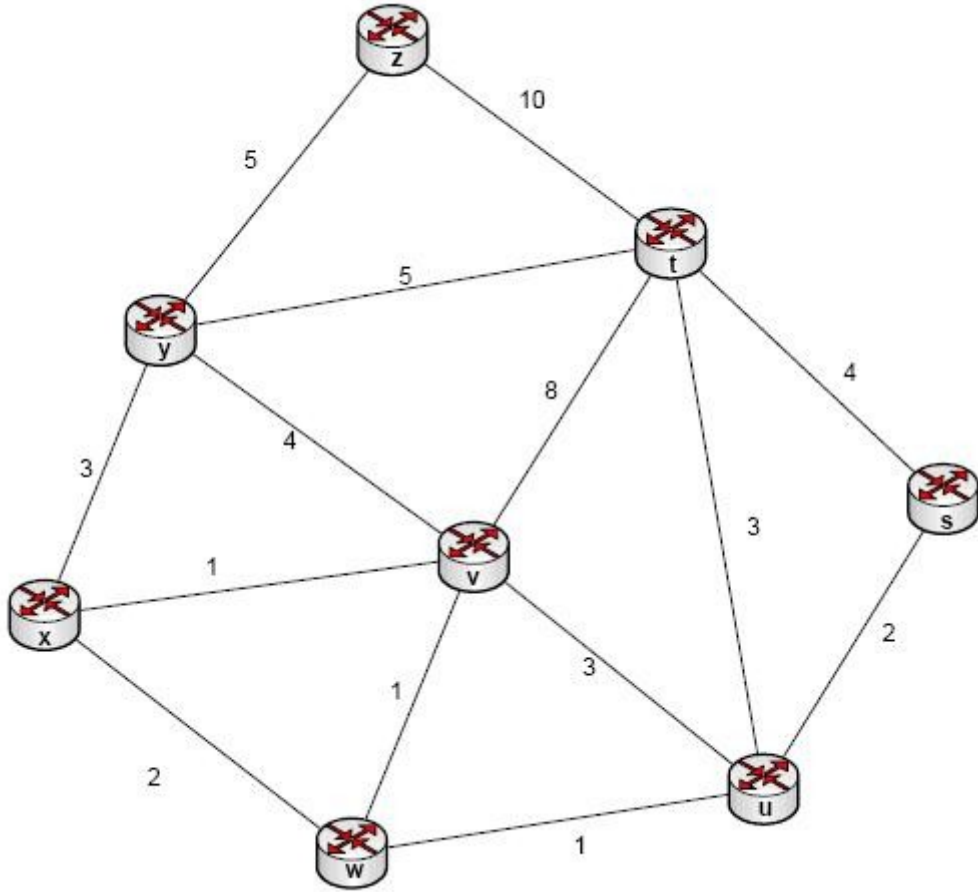
Bu noktadan sonra u-v:3 , y-v:4 , s-t:4 , y-z:5 bağlantılarındaki her iki düğümde aynı adada olduğu için atlanır ve y-t:5 bağlantısına geçilir.



z-t:10 bağlantısı ise iki düğüm de gezildiği için yine gereksizdir.

### **SORU 57: Prim asgari tarama ağacı Algoritması**

Bir asgari tarama ağacı (minimum spanning tree) algoritması olan Prim algoritması, işaretlemiş olduğu komşuluklara en yakın düğümü bünyesine katarak ilerler. Buna göre aşağıdaki grafiğin asgari tarama ağacını çıkaralım:



Yukarıdaki grafikte her düğüm için bir temsili harf ve her bağlantı için bir ağırlık değeri atanmıştır. Buna göre her düğümün diğerine gitmenin maliyeti belirlenmiştir.

Prim algoritmasında rasgele bir başlangıç noktası seçilir. Örneğin bizim başlangıç noktamız “z” düğümü olsun. Bu durumda ilk inceleyeceğimiz komşuluk, “z” düğümünden gidilebilen düğümler ve maliyetleri olacaktır.

z düğümünden gidilebilen düğümler ve maliyetleri aşağıda listelenmiştir:

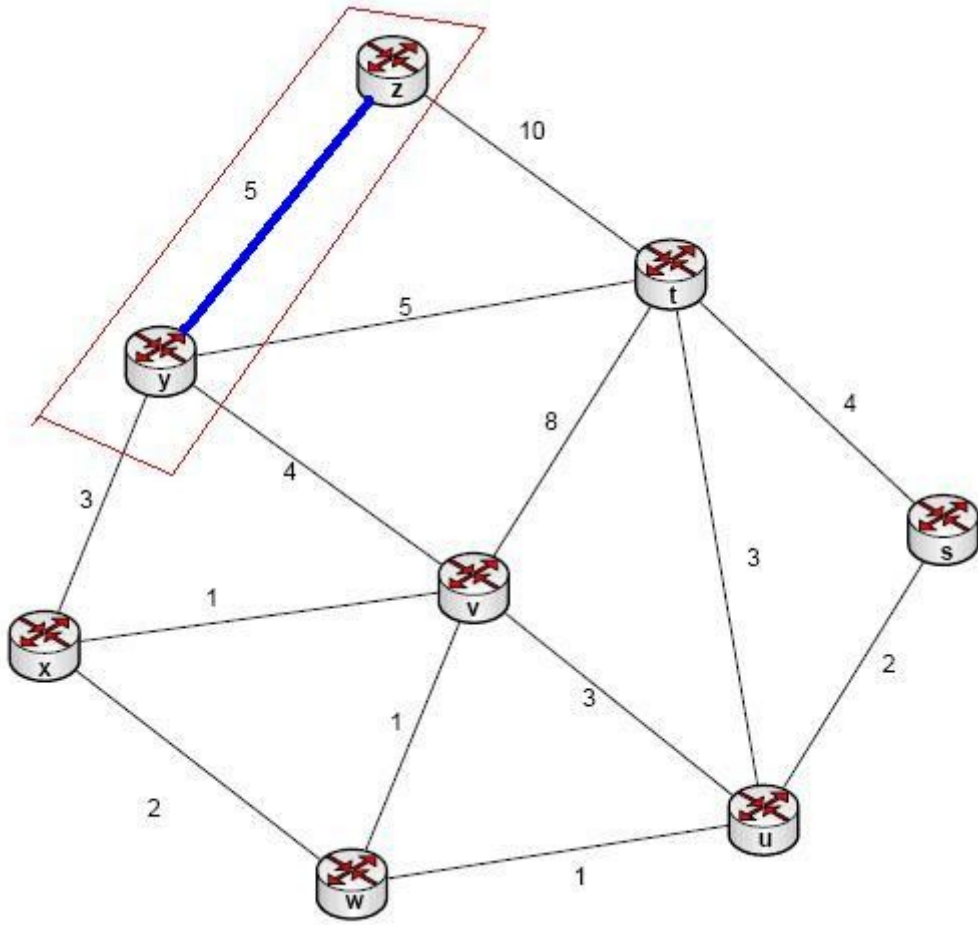
y:5

t:10

Prim algoritması bu listedeki en küçük maliyetli komşuyu bünyesine dahil eder. Buna göre yeni üyelerimiz {z,y} olacaktır ve gidilen yollar {z-y:5} olacaktır. (ilk üyeler listesinde şimdiye kadar ziyaret edilmiş düğümler bulunur. Bu düğümler listesinde zaten olan bir düğüm listeye eklenemez. yollar listesinde ise hangi düğümün hangi düğüme ne kadar maliyetle gidildiği tutulur.) Dolayısıyla grafiğimizde Prim algoritması tarafından işaretlenen düğümler

aşağıda

gösterilmiştir:



şimdi üyelerimizin durduğu listedeki bütün düğümlerin komşularını listeleyelim:

t:5

t:10

v:4

x:3

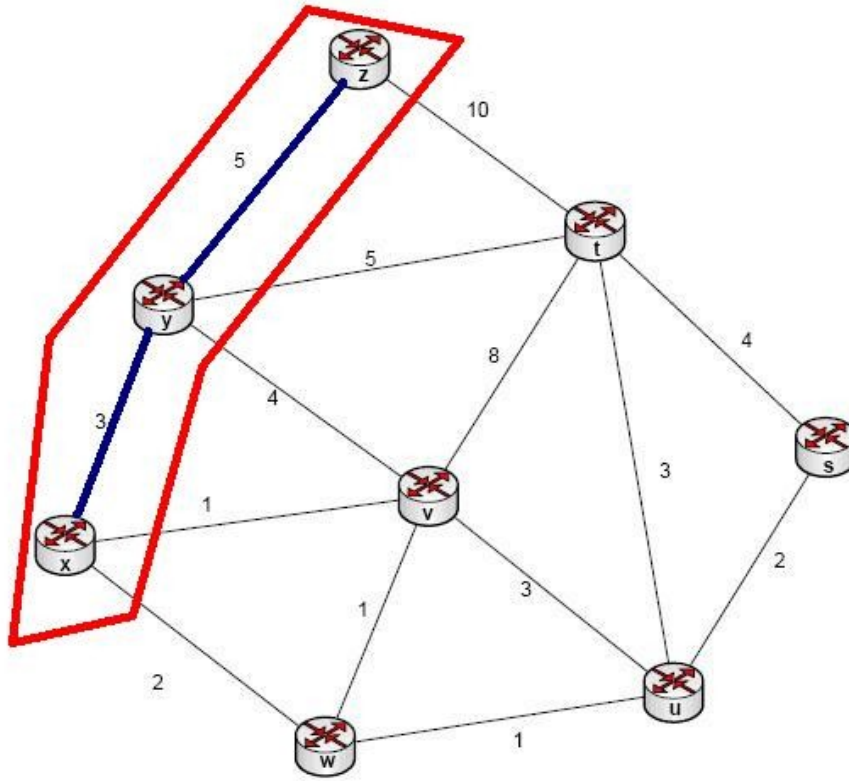
yukarıdaki listede t düğümüne iki farklı gidiş bulunmaktadır ( hem z hem de y üzerinden). Biz algoritmamıza devam edip en küçük yolu bünyemize dahil edelim. En yakın komşu x:3'tür. Bu durumda üyelerimiz {z,y,x} olacak ve yollarımız {z-y:5,y-x:3} olacaktır. Bu durum



aşağıdaki

grafikte

gösterilmiştir:



Yeniden

komşularımızı

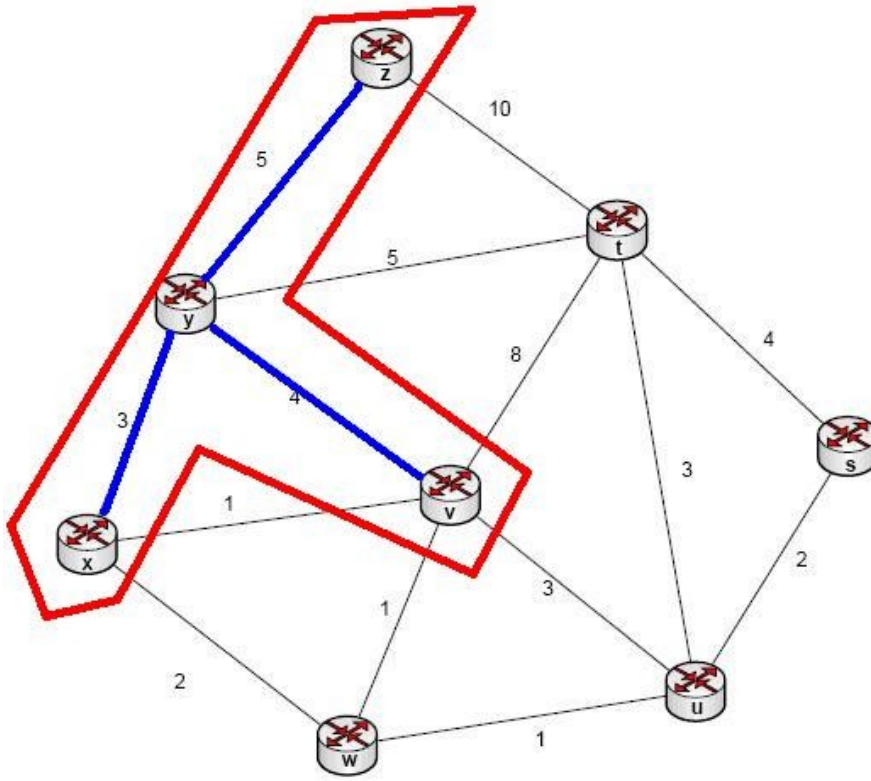
listelersek:

t:5

v:1

w:2

yukarıdaki listede bünyemize aldığımız adadan, bir düğüme giden birden fazla yol bulunması durumunda en kısıası alınmıştır. Bu durumda listenin en küçük elemanı olan v:1 tercih edilir ve üyelerimiz {z,y,x,v} yollarımız ise {z-y:5,y-x:3,x-v:1} olur. Durum aşağıdaki grafikte gösterilmiştir:



Yeniden

komşularımızı

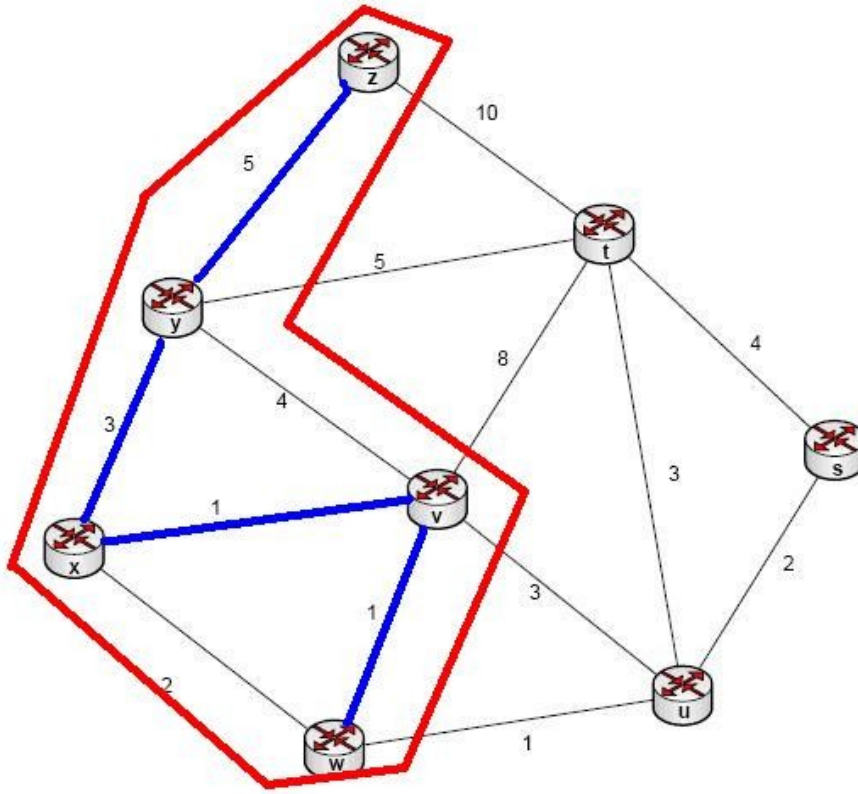
listelersek:

t:5

u:3

w:1

yukarıdaki listede bünyemize aldığımız adadan, bir düğüme giden birden fazla yol bulunması durumunda en kısıası alınmıştır. Bu durumda listenin en küçük elemanı olan w:1 tercih edilir ve üyelerimiz {z,y,x,v,w} yollarımız ise {z-y:5,y-x:3,x-v:1,v-w:1} olur. Durum aşağıdaki grafikte gösterilmiştir:



Yeniden

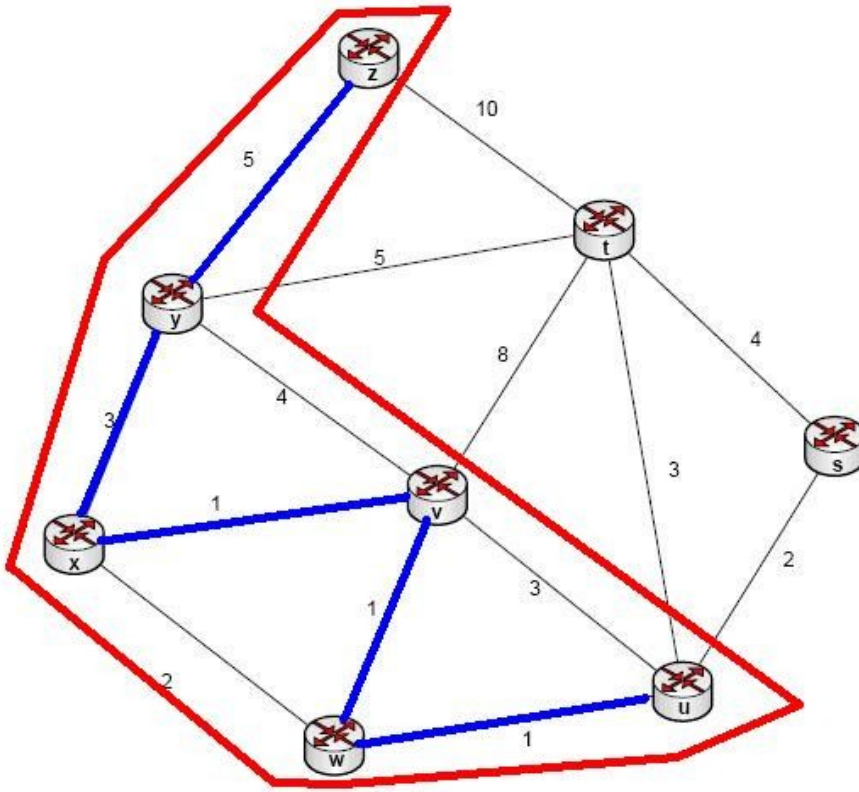
komşularımızı

listelersek:

t:5

u:1

yukarıdaki listede bünyemize aldığımız adadan, bir düğüme giden birden fazla yol bulunması durumunda en kısası alınmıştır. Bu durumda listenin en küçük elemanı olan u:1 tercih edilir ve üyelerimiz {z,y,x,v,w,u} yollarımız ise {z-y:5,y-x:3,x-v:1,v-w:1,w-u:1} olur. Durum aşağıdaki grafikte gösterilmiştir:



Yeniden

komşularımızı

listelersek:

t:3

s:2

yukarıdaki listede bünyemize aldığımız adadan, bir düğüme giden birden fazla yol bulunması durumunda en kısası alınmıştır. Bu durumda listenin en küçük elemanı olan s:2 tercih edilir ve üyelerimiz {z,y,x,v,w,u,s} yollarımız ise {z-y:5,y-x:3,x-v:1,v-w:1,w-u:1,u-s:2} olur. Durum

aşağıdaki

grafikte

gösterilmiştir:

Son komşumuz olan  $t$  için en kısa ulaşım  $t:3$  değeridir ve  $u$  üzerinden sağlanır. Bu durumda listenin en küçük elemanı olan  $t:3$  tercih edilir ve üyelerimiz  $\{z,y,x,v,w,u,s,t\}$  yollarımız ise  $\{z-y:5,y-x:3,x-v:1,v-w:1,w-u:1,u-s:2,u-t:3\}$  olur. Sonuçta elde edilen asgari tarama ağacı aşağıda verilmiştir:

asgari	tarama	ağacını	veren	en	meşhur	algoritmalar:
Kruskal						Algoritması
Prims						Algoritması
Dijkstra Algoritması (Asgari tarama ağacının benzeri olarak bir arama algoritması örneği)						

### **SORU 59: yönlendirici (router)**

paket değişmeli (packet switched) kullanılan ağlarda kesişim noktalarında paketlerin güzergah seçmeleri için yol gösteren aletlerdir. Basitçe yol ayrımlarında yönlendirme yaparak paketlerin ulaşacakları noktalara daha hızlı ulaşmalarını hedeflerler. Aslında yönlendiriciler basit birer bilgisayardır ve üzerlerinde birer işletim sistemi yüklüdür. Genelde bu işletim sistemi sadece yönlendirme amacıyla yazılmış ve daha hızlı çalışması için diğer bütün özellikleri kaldırılmıştır. Yönlendiriciler doğası gereği birden fazla ağ arayüzüne (network interface) sahiptirler ve bu arayüzler arasında yönlendirme işlemi yaparlar. Bu arayüzlerin baktığı ağlara alt ağ (subnet) adı verilir ve amaç bu ağlar arasında gidip gelen paketlere yönlendirme yapmaktır. Temel olarak bir yönlendirici kendi ağında dolaşan paketler için yönlendirme yapmaz bu paketleri o ağda bırakır, böylelikle ağ trafiğini azaltmış olur.

Bir yönlendirici internet protokolü seviyesinde IP adreslerine bakarak yönlendirme yapar. Yani TCP/IP protokolünde 3. seviye olan ağ katmanına (network layer) kadar paketleri açarak yönlendirme yapar.

{bu konu ileride daha detaylı anlatılmalıdır}

### **SORU 60: en uzun önek eşleşmesi (longest prefix matching)**

IP (internet protocol, internet protokolü) kullanan yönlendiriciler (router) tarafından yönlendirme tablosundan (routing table) bir kayıt bulurken kullanılan bir hesaplama yöntemidir. Bu yöntemle göre, bir yönlendirme tablosunda (routing table) birden fazla alt ağ kaydı (subnet) bir IP adresini kapsayabilir. Bu durumda bu alt ağ kayıtlarından (Subnet) en belirleyicisi en uzun olanıdır ve bu alt ağ kaydı tercih edilmelidir.

Örnek olarak yönlendirme tablomuzda aşağıdaki iki kayıt bulunsun.  
192.168.20.16/28  
192.168.0.0/16

Ve yönlendirmek istediğimiz, dolayısıyla aramak istediğimiz adres 192.168.20.19 adresi olsun. Bu durumda, üstteki kayıt yönlendirilme sırasında tercih edilir, sebebi alttaki kayda göre daha uzun olmasıdır. (28, 16'dan daha uzundur ve daha belirleyicidir)

### **SORU 61: TCP AIMD (additive increase multiplicative decrease, toplanarak artan çarpılarak azalan)**

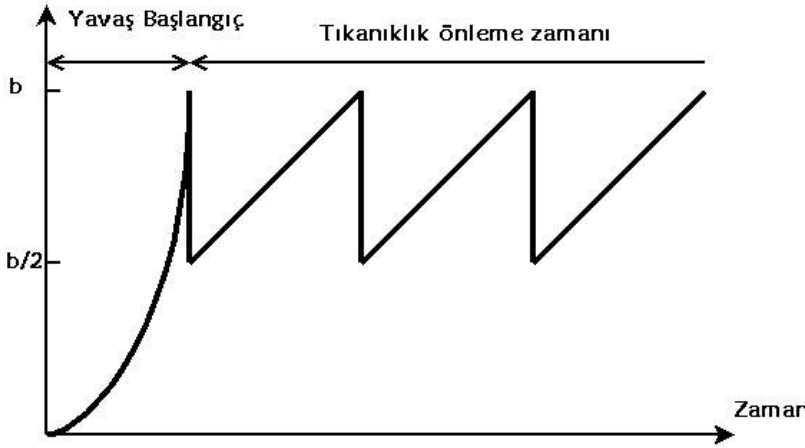
AIMD, TCP algoritması üzerinde çalışan bir tıkanıklık önleme yöntemidir. Buna göre tıkanıklık penceresinin (congestion window) sayısal değeri toplanarak yani doğrusal (linear) olarak artarken, ortamda bir tıkanıklık olması durumunda ise üssel olarak azalmasıdır.(yani yarılanması diye de yorumlanabilir).

Algoritma basitçe, bir paket kaybı oluşana kadar pencere boyutunun her RTT (round trip time, yani bir paketin hedefe gidip geri gelmesi için geçen süre) boyunca 1 MSS (Maximum Segment Size, yani o anda elimizde bulunan en büyük pencere boyutu) kadar artırılmasını hedefler.

Şayet bir paket kaybı oluşursa bu durumda da pencere boyutunun yarıya indirilmesini hedefler.

Sonuç grafiği çizildiğinde, testere dişlerine benzer bir grafik elde edilir.

Pencere boyutu



Yukarıdaki grafiğe bu testere dışı örnek olarak gösterilmiştir. Grafiğin solunda bulunan eğimli alan, yavaş başlangıç (slow start) kısmı olup bu kısımda pencere boyutu üssel olarak arttırılmaktadır. Daha sonraki testere dışı kısımlarında doğrusal artış yapılmış ve bir paket kaybı olması durumunda paket boyutu yarısına düşürülmüştür. Algoritmanın çalışması basitçe:

$b = b - ab$  (paket kaybı olduğunda)

$b = b + c/b$  (paket kaybı olmadığında)

formülleri ile özetlenebilir. Yukarıdaki formüllerde  $b$ , paket boyutu,  $a$  ve  $c$  ise birer sabit sayı olarak düşünülebilir. Bu algoritmayı kullanan TCP, paket kaybı önleme yöntemlerinden birisi de TCP Reno yöntemidir.

## SORU 62: TCP Reno , Tahoe

TCP protokolü kullanılan ağlarda, birden fazla [tıkanıklık önleme \(congestion avoidance\)](#) yöntemi bulunmaktadır. Bunlardan bir tanesi de TCP Reno algoritmasıdır. Basitçe özellikleri aşağıdaki şekilde listelenebilir:

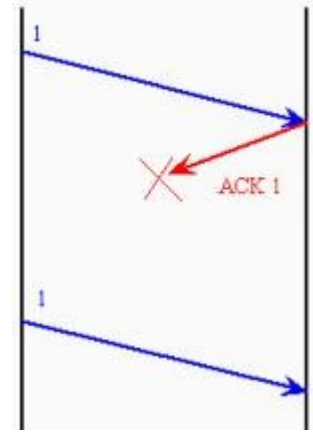
- [yavaş başlangıç \(slow start\)](#)
- [Hızlı kurtarma \(fast recovery\)](#)

### Çalışma mantığı:

- Basitçe her başarılı onay paketinden (acknowledgement ack) sonra [tıkanıklık penceresini \(congestion window\)](#) 1 artırır.
- Şayet [paket kaybı](#) olursa ([packet loss](#), beklenen zamanda cevabının gelmemesi şeklinde (timeout)) bu durumda [tıkanıklık penceresi \(congestion window\)](#)'nin değeri başlangıç değeri olan 1'e geri döner.
- Şayet tıkanıklık olduysa [tıkanıklık penceresini \(congestion window\)](#)'nin kapasitesi dolduysa ve onay paketi gelmediyse, bu durumda [tıkanıklık penceresini \(congestion window\)](#)'nin kapasitesi yarısına indirilir.

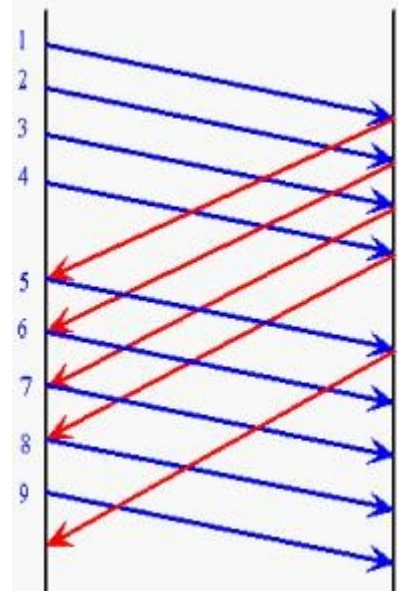


Yukarıdaki bu durumlar aşağıdaki şekillerde anlatılmıştır:



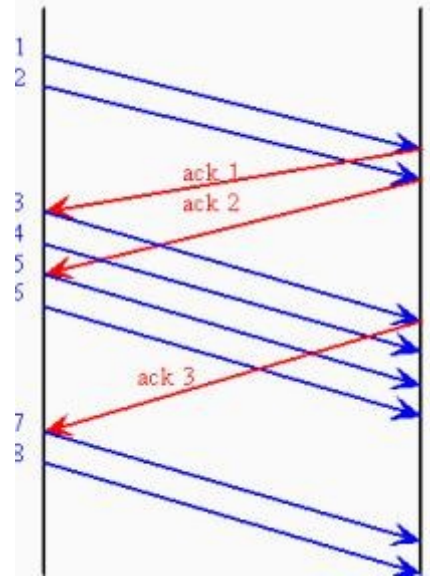
1)

Yukarıdaki grafikte bir paketin zaman aşımına uğraması (onay mesajının beklenen zamanda ulaşmaması) durumu gösterilmiştir. Buna göre TCP Reno algoritması, [tıkanıklık penceresinin \(congestion window\)](#) kapasitesini yarisına indirir.



2)

yukarıdaki resimde, TCP Reno algoritmasında, her gönderilen mesajın zamanında ve sorunsuz olarak onaylandığı (acknowledgement) gösterilmektedir. Resimde de görüldüğü üzere sorunsuz gelen onay mesajları sonunda [tıkanıklık penceresinin \(congestion window\)](#) boyutu arttırılmıştır.



3)

Yukarıdaki resimde ise, gönderilen onay paketleri (ack) gönderen tarafın beklediği hızdan yavaş olarak ulaşmaktadır. Bu durumda TCP Reno algoritması, [tıkanklık penceresinin \(congestion window\)](#) boyutu azaltmıştır.

### TCP Tahoe

TCP reno benzeri bir şekilde tıkanklık önleme algoritmalarından birisi de tcp tahoe algoritmasıdır. Bu algoritma Reno'dan farklı bir şekilde tıkanklığı algılar ve çözüm üretir.

Tahoe algoritmasında, paket kaybı, ACK alınmasındaki zaman aşımında algılanır. Paket kaybının algılanması durumunda da tahoe algoritmasında [tıkanklık penceresinin \(congestion window\)](#) boyutu 1 mss 'e (maximum segment size) indirilir. Ardından iletişim sıfırlanarak [yavaş başlangıç \(slow start\)](#) yapılır ve iletişime devam edilir.

Aynı durum, yani paket kaybının algılanması, Reno algoritmasında 3 adet tekrarlı ACK mesajının alınması ile olur. Reno'nun bu durumda ürettiği çözüm ise Tahoe'den farklı olarak [tıkanklık penceresini \(congestion window\)](#) yarıya indirmek ve yukarıdaki yazıda da anlatılan [hızlı kurtarma \(fast recovery\)](#) yapmaktır.

Bu anlamda, hızlı kurtarma işleminin sadece TCP reno tarafından kullanıldığını ve Tahoe tarafından kullanılmadığını söyleyebiliriz. Ayrıca zaman aşımı durumunda (time out) iki algoritma da tıkanklık penceresinin boyutunu 1 MSS indirmektedir.

### SORU 63: tıkanklık önleme (congestion avoidance)

Amaç bilgisayar ağlarında oluşan tıkanklıkların önlenmesidir. Bunun için gönderen tarafın, paket kaybı veya noktasal gecikmeler gibi ağdaki tıkanklık sebeplerini hesaba katarak gönderim hızını azaltması veya arttırması esasına dayanır.

TCP ağlar için kullanılan tıkanklık kontrol yöntemi (istatistiksel olarak hattan yararlanacak azami kullanıcının hesaplanarak bu sayıdan fazlasının engellenmesi gibi) veya aktif sıra yönetimi (active queueing management, bu yöntemde sıra teorisinde (queueing theory) yapılan istatistiksel yaklaşımlar ile tıkanklığı önceden anlamak hedef alınmıştır) veya rasgele erken tespit (random early detection , bu yöntem active queueing management'ın bir alt türüdür) gibi istatistiksel yöntemler ile tıkanklık algılanır ve çözüm olarak gönderim hızı veya noktalarda bulunan sıraların kontrol yöntemi değiştirilir.

### **SORU 64: tıkanıklık (congestion)**

Paket değişimli bilgisayar ağlarında paket yollanması sırasında yaşanan paketlerin istenilen hedefe geç ulaşması veya hiç ulaşamaması sonucunu doğuran olaydır. Basitçe trafikte giden araçların trafik sıkışıklığı yüzünden istedikleri yere geç ulaşması gibi düşünülebilir.

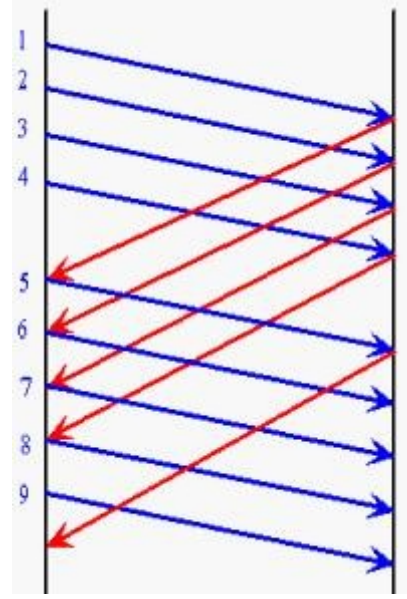
Temel sebepleri paket kaybı (packet loss) ve sıra gecikmesidir (queueing delay).

### **SORU 65: hızlı kurtarma (fast recovery)**

Bu yaklaşım yavaş başlangıç yönetiminin bir alt uygulamasıdır. Bu yaklaşımda paket onaylarındaki gecikmeden (congestion, tıkanıklık) kaynaklanan pencere boyutunu düşürme işlemi, pencerenin boyutunu daha yavaş azaltmakla olmaktadır. Yani üssel olarak artmakta olan pencere boyutunu diyeliki  $2n$  boyutuna çıkarttık, bir sonraki adımda  $4n$  olacaktır. Ancak  $4n$  yaptıktan sonra pencerenin dolduğunu gözlemledik. Bu durumda yavaş başlangıç algoritmasının  $2n$  boyutundan doğrusal arttırım yaparak devam etmesi gerekmektedir. Bunun yerine hızlı devam yönteminde  $4n$  sayısından azaltarak pencere boyutu değiştirilmeye devam eder.

### **SORU 66: yavaş başlangıç (slow start)**

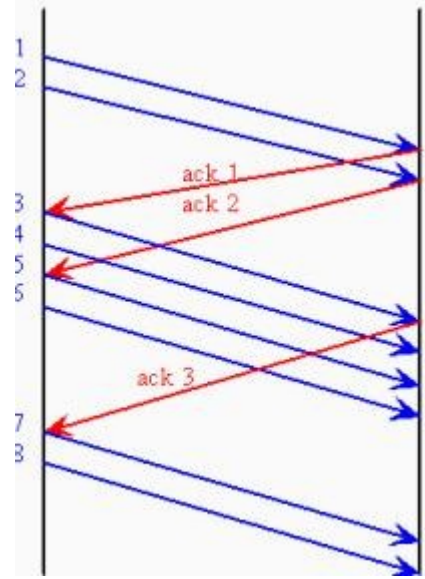
TCP protokolü için geliştirilen bir tıkanıklık önleme yöntemidir. Amaç hakkında bilgi bulunmayan bir ağda paket yollanırken, paket kayıplarını asgariye indirmek için hattı test ederek paket gönderim hızını arttırmaktır. Doğrusal (linear) veya üssel (growth) şeklinde artan iki farklı yaklaşımdır: Doğrusal yaklaşımda tıkanıklık penceresi (congestion window) içerisindeki bir bilginin onaylanması durumunda pencerenin kapasitesi onaylanmış paketler kadar arttırılır (veya sabit bir sayı kadar arttırılır (artış miktarı sabitse doğrusal artım, değişkense üssel artım şeklinde yorumlanabilir)) dolayısıyla her başarılı transferden sonra pencere değişken miktarda büyütülerek anlık akatarım miktarı arttırılmış olur. Ters durumda, yani penceredeki bir paket için nack (negative acknowledgement, hata bildirisi) ulaşırsa veya pencerenin başındaki paket onaylanmadan pencere dolarsa (henüz onaylanmamış (ack) pencere kapasitesi kadar paket oluşursa) bu durumda TCP, problemin tıkanıklıktan kaynaklandığını düşünerek pencere boyutunu bir küçültür (veya sabit bir sayı kadar azaltır). Doğrusal arttırım yukarıda da bahsedildiği gibi pencere boyutunun 1er 1er arttırılması durumudur. Bu durum için TCP, RTT (round trip time, bir paketin karşı tarafa ulaşip geri gelmesi için geçen süre) zamanı boyunca pencere boyutunu 1 arttırır. Üssel arttırım yaklaşımında ise her onay paketinden sonra onaylanmış paketler kadar arttırır. Bu durum aşağıdaki iki resimde gösterilmiştir:



**durum**

**1)**

yukarıdaki resimde, TCP Reno algoritmasında, her gönderilen mesajın zamanında ve sorunsuz olarak onaylandığı (acknowledgement) gösterilmektedir. Resimde de görüldüğü üzere sorunsuz gelen onay mesajları sonunda tıkanıklık penceresinin (congestion window) boyutu arttırılmıştır.



**durum**

**2)**

Yukarıdaki resimde ise, gönderilen onay paketleri (ack) gönderen tarafın beklediği hızdan yavaş olarak ulaşmaktadır. Bu durumda TCP Reno algoritması, tıkanıklık penceresinin (congestion window) boyutu azaltmıştır.

Hızlı devam (fast recovery) Bu yaklaşım yavaş başlangıç yönetiminin bir alt uygulamasıdır. Bu yaklaşımda paket onaylarındaki gecikmeden (congestion, tıkanıklık) kaynaklanan pencere boyutunu düşürme işlemi, pencerenin boyutunu daha yavaş azaltmakla olmaktadır. Yani üssel olarak artmakta olan pencere boyutunu diyeliki  $2n$  boyutuna çıkarttık, bir sonraki adımda  $4n$  olacaktır. Ancak  $4n$  yaptıktan sonra pencerenin dolduğunu gözlemledik. Bu durumda yavaş başlangıç algoritmasının  $2n$  boyutundan doğrusal arttırım yaparak devam etmesi gerekmektedir. Bunun yerine hızlı devam yönteminde  $4n$  sayısından azaltarak pencere boyutu değiştirilmeye devam eder.

Yavaş başlangıç (slow start) algoritmasının eksik yanı bütün paket onay sorunlarını tıkanıklık sebebine bağlamasıdır. Oysaki paketler ortam kalitesinden kaynaklanan sorunlardan dolayı geç onaylanabilir veya hiç onaylanmayabilir. Örneğin kablosuz ağlarda bu tip paket kayıpları çok yaşanmaktadır. Dolayısıyla kablosuz ağlarda bir anda paket kayıpları yüksek miktardayken pencere boyutu aniden çok küçük değerlere getirilecek ve sonraki anda çok kaliteli bir iletişim varken pencere boyutunun eski değerlere ulaşması vakit alacaktır.

### **SORU 67: tıkanıklık penceresi (congestion window)**

TCP protokolünde kullanılan ve onaylanmamış paketlerin (gönderilmiş ama henüz acknowledgement almamış paketlerin ) durumunu tutan penceredir. Pencerenin boyutu, bir paket onaylanmadan önce ne kadar paketin tıkanıklığa takılacağına bağlı olarak artar veya azalır. Bu boyut hesaplandıktan sonra, ilk bilginin onaylanmasından sonra ne kadar bilginin gönderilebileceği bulunmuş olur. Yani basitçe kayan pencere algoritması şeklinde çalışır ve pencerenin başında bulunan paket onaylanınca pencere bir kaydırılır. Bu ilk paket onaylanmadığı sürece pencere kaymaz ve pencerenin kapasitesinin dışındakiler gönderilmez.

Bu pencereinin büyüklüğü transfer hızına ve kalitesine bağlıdır, gönderim yapılan kanal ne kadar kaliteli ve genişse pencere o kadar büyüktür.

### **SORU 68: http (hyper text transfer protocol, hipermetin transfer protokolü)**

Internet üzerinde web sayfalarının görüntülenmesi için kullanılan protokoldür. Genel olarak web sunucusunun 80. portundan yapılan iletişimidir. Buna göre bilgilerin sunucudan nasıl isteneceği ve sunucunun vereceği cevaplar bir standarda oturulmuştur. HTTP'nin kullanılan iki versiyonu şunlardır:

HTTP 1.0 : Bu versiyonunda sunucuya istekler (request) ve sunucu cevapları (response) birer bire ulaşmaktadır. Yani bağlantı açık kalmamakta her nesne için kapanıp açılmaktadır (nonpersistent)

HTTP 1.1: Bu versiyonunda sunucuya istekler ardışık olarak ulaşmakta ve sunucu her isteğe cevabı yollamakta ancak bağlantıyı kapatmamaktadır. Bağlantı bütün talepler bittikten sonra istemci tarafından gelen kapama isteği ile kapatılmaktadır. Dolayısıyla her nesne için yeniden bağlantı kurma ve bağlantı kapatma maliyeti bulunmamaktadır. Bunun yanında her istemcinin hangi nesnelerden sonra hangi nesneleri istediği gibi bilgilere de ulaşmak mümkündür.

HTTP							kodları:
Aşağıda	anlatılan	kodları	3	gruba	ayırmak		mümkündür:
200	ile	başlayan		grup	hatasız		bağlantı
300	ile	başlayan		grup			yönlendirme
400	ile	başlayan			grup		hata
500	ile başlayan	grup					

sunucu hataları

Bu kodlar aşağıdadır:

200				—			OK
The	request	sent	by	the	client	was	successful.

Başarılı iletişim.

201	–	Created
The request was successful and a new resource was created. Talep başarılı ve yeni bir kaynak hazırlandı.		
202	–	Accepted
The request has been accepted for processing, but has not yet been processed. Talep işlem için kabul edilmiştir ancak henüz işlenmemiştir.		
203	– Non-Authoritative	Information
The returned meta information in the entity-header is not the definitive set as available from the origin server.		
204	– No	Content
The request was successful but does not require the return of an entity-body.		
205	– Reset	Content
The request was successful but the User-Agent should reset the document view that caused the request.		
206	– Partial	Content
The partial GET request has been successful.		
300	– Multiple	Choices
The requested resource has multiple possibilities, each with different locations.		
301	– Moved	Permanently
The resource has permanently moved to a different URI.		
302	–	Found
The requested resource has been found under a different URI but the client should continue to use the original URI.		
303	– See	Other
The requested response is at a different URI and should be accessed using a GET command at the given URI.		
304	– Not	Modified
The resource has not been modified since the last request.		
305	– Use	Proxy
The requested resource can only be accessed through the proxy specified in the location field.		
306	– No Longer	Used
Reserved for future use.		
307	– Temporary	Redirect
The resource has temporarily been moved to a different URI. The client should use the original URI to access the resource in future as the URI may change.		

400	—	Bad	Request
The syntax of the request was not understood by the server.			
401	—	Not	Authorised
The request needs user authentication			
402	—	Payment	Required
Reserved for future use.			
403	—		Forbidden
The server has refused to fulfill the request.			
404	—	Not	Found
The document/file requested by the client was not found.			
405	—	Method	Not Allowed
The method specified in the Request-Line is not allowed for the specified resource.			
406	—	Not	Acceptable
The resource requested is only capable of generating response entities which have content characteristics not specified in the accept headers sent in the request.			
407	—	Proxy	Authentication Required
The request first requires authentication with the proxy.			
408	—	Request	Timeout
The client failed to send a request in the time allowed by the server.			
409	—		Conflict
The request was unsuccessful due to a conflict in the state of the resource.			
410	—		Gone
The resource requested is no longer available and no forwarding address is available.			
411	—	Length	Required
The server will not accept the request without a valid Content-Length header field.			
412	—	Precondition	Failed
A precondition specified in one or more Request-Header fields returned false.			
413	—	Request	Entity Too Large
The request was unsuccessful because the request entity is larger than the server will allow.			
414	—	Request	URI Too Long
The request was unsuccessful because the URI specified is longer than the server is willing to process.			
415	—	Unsupported	Media Type
The request was unsuccessful because the entity of the request is in a format not supported by the requested resource for the method requested.			

416	–	Requested	Range	Not	Satisfiable
The request included a Range request-header field, and not any of the range-specifier values in this field overlap the current extent of the selected resource, and also the request did not include an If-Range request-header field.					
417	–		Expectation		Failed
The expectation given in the Expect request-header could not be fulfilled by the server.					
500	–	Internal		Server	Error
The request was unsuccessful due to an unexpected condition encountered by the server.					
501	–		Not		Implemented
The request was unsuccessful because the server can not support the functionality needed to fulfill the request.					
502	–		Bad		Gateway
The server received an invalid response from the upstream server while trying to fulfill the request.					
503	–		Service		Unavailable
The request was unsuccessful to the server being down or overloaded.					
504	–		Gateway		Timeout
The upstream server failed to send a request in the time allowed by the server.					
505	–	HTTP	Version	Not	Supported
The server does not support or is not allowing the HTTP protocol version specified in the request.					

### **SORU 69: kapsülleme (encapsulation)**

genel olarak bir bilginin soyut bir yapı içerisine konulmasına verilen isimdir. En çok ağ teknolojilerinde ve nesne yönelimli programlama dünyasında kullanılır.

Nesne Yönelimli Programlama için anlamı bir sınıfın (class) bilgilerinin dışarıya kapalı olması ve bu sınıfın her türlü veri iletişiminin kontrol altındaki metodlar ile yapılmasıdır.

Ağ teknolojileri için anlamı, katmanlı mimaride (OSI katmanı veya Internet katmanları gibi), her katman arasında verinin bir kapsüle konularak diğer (alt veya üst) katmana geçirilmesidir. Buna göre gönderilmek istenen veri her alt katmana indikçe, indiği katmandaki başlık bilgileri eklenerek yeni bir kutuya konulmuş gibi paketlenir. Ulaştığı yerde ise her katman kendisi ile ilgili kutuyu açarak görevini icra eder.

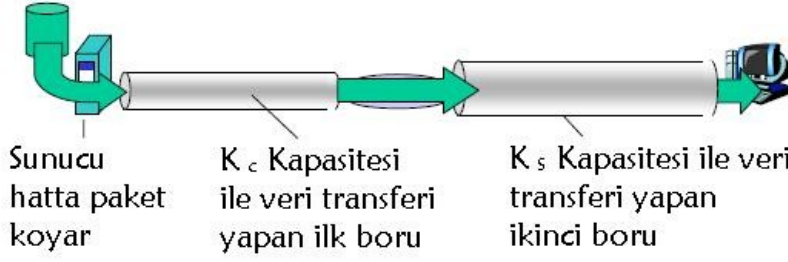
### **SORU 70: Çıktı (throughput)**

Ağ üzerinde bir göndericiden bir alıcıya kadar olan hattın çıktısıdır. Yani hatta giren ve çıkan bilgi (bit) kapasitesidir. Basitçe aşağıdaki formül ile hesaplanabilir:

(gönderilen bilgi (bit) ) / zaman = bit / saniye = bits / second



Örneğin bir ADSL hattının kapasitesi 1024 Kbps denilirken kastedilen hattın saniyede 1024 kbit veri geçtiğidir.



Yukarıdaki şekilde sunucu ile istemci arasında veri transferi yapılması sırasında iki farklı hattın veri transferini gösteren temsili resim verilmiştir. Bu hatlar sıvı taşıyan borulara benzetilirse borulardan akan suyun miktarı hattaki en dar olan boru kadardır.

Anlık çıktı (instantaneous throughput): Anlık olarak hattaki veri transfer oranını verir

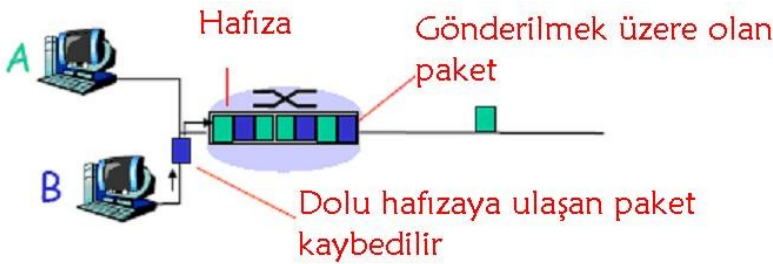
Ortalama çıktı (average throughput): Uzun süreli bir aralıktaki veri transfer oranını verir.

### **SORU 71: paket kaybı (packet loss)**

Ağ üzerinde gönderilen bir paketin ulaştığı noktalardaki hafıza kısıtlarından dolayı nokta tarafından kabul edilmemesi veya ağdaki ulaşım yollarında karşılaşılan sorunlardan dolayı hedefe ulaşmadan kaybedilmesine verilen addır.

Yani paket bir hedefe ulaşmak için bir kaynaktan ayrılır ve hedefe asla ulaşamazsa kaybolmuş olarak kabul edilir. Şayet gecikme olursa bunun adı paket gecikmesidir (packet delay).

kaybolan paketler, emin bir protokol ile (reliable protocol) gönderme yapılıyorsa yeniden yollanarak telafi edilirler.



yukarıdaki şekilde iki farklı bilgisayardan paket gelen bir noktada paket gönderme hızı alınan paketleri işlemeye yetmemektedir. Nuktada bulunan hafıza (buffer) yettiği sürece gönderilemeyen paketleri sıraya sokarak bekletir. Ancak bu hafıza da dolduktan sonraki paketler kaybedilir.

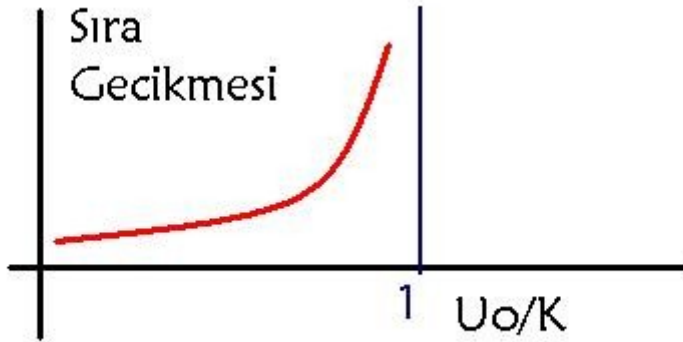
### **SORU 72: sıra gecikmesi (queueing delay)**

Ağ üzerinde bulunan bir noktada alınan paket miktarından daha azının yollanabilmesi durumunda birikme olur. Bu birikmeler paket kaybına sebep olabilmektedir. Sıra gecikmesi,

paket kaybı olmaksızın bir paketin bir noktaya erişmesinden sonra kendisine sıra gelip noktayı terk edene kadar geçen zamandır. Buna göre bir noktaya erişen paketler ve terk eden paketler arasındaki kapasite farkına göre hesaplama yapmak mümkündür. Bu farka trafik yoğunluğu adı verilir ve aşağıdaki formülle hesaplanır:

Trafik yoğunluğu =  $U_o/K$  (traffic intensity = (Packet Length) x (Average packet arrival rate) / Bandwidth

yukarıdaki formülde, U= Paket Uzunluğu, o= ortalama paket varma oranı, K = hattın kapasitesi şeklindedir.



yukarıdaki grafikte, paket gecikmesi ile trafik yoğunluğu arasındaki değişim verilmiştir. buna göre trafik yoğunluğu 1 olduğunda (yani noktada işlenen paketlerin üzerinde paket ulaştığında) noktasal gecikme sonsuz olmaktadır. Benzer şekilde trafik yoğunluğu 0 olduğunda, bekleme gecikmesi de 0 olmaktadır.

### **SIRA 73: noktasal gecikme (nodal delay)**

Ağ üzerindeki bir noktada oluşan gecikmedir. Noktasal gecikme, ağ üzerinde bir uçtan bir uca ulaşana kadar paketlerin geçtiği her noktada oluşan gecikmedir. Noktasal gecikmeye sebep olan faktörler şunlardır:

gi = işleme gecikmesi (processing delay) = paketlerin geçtikleri nokta tarafından alınıp, belirli bir seviyeye kadar açılması ve sonunda işlenerek karar verilmesi süresidir.  
gs = sıra gecikmesi (queueing delay) = paketlerin geçtikleri noktada önlerindeki paketleri bekleme süresidir. Geçilen her noktanın belirli bir işleme ve paket yollama kapasitesi bulunmaktadır. Bu kapasiteden daha fazla paket gelirse, işlenemeyeceği için hafızada bekletilir. Bu bekleme süresine sırada bekledikleri için sıra gecikmesi adı verilir.  
gg = gönderme gecikmesi (transmission delay) = paketlerin hat kapasitesine uygun olarak gönderilmesi süresinde geçen vakittir.  
go = ortam gecikmesi (propagation delay) = paketlerin ortam şartlarına bağlı olarak gecikme süresidir. Örneğin kablosuz bir iletişim sırasında atmosfer şartlarına bağlı olarak veya bakır kablo üzerinden yapılan iletişimde kablonun mesafesi ve kalitesine bağlı olarak gecikme süresidir.

Bir noktadaki toplam gecikme yukarıda sayılanların toplamıdır:  $gi + gs + gg + go$

#### **SORU 74: frekans bölmeli çoklama (sıklık bölmeli çoklayıcı, frequency division multiplexing, fdm)**

Literatürde sıklık bölmeli çoklayıcı, dalga bölmeli çoklayıcı veya frekans bölmeli çoklayıcı olarak geçmektedir. Tek hat üzerinden birden fazla kişinin iletişimine izin veren devre anahtarlamalı ağ tipidir. Buna göre hattı birden fazla kişi kullanmak için farklı frekans aralıklarını kullanırlar. Örneğin bir hattı 4 farklı kişi kullanmak istiyorlar. Hattın kullanımı farklı frekanslardan alış verişi ile 4 farklı kişiye bölünerek her frekans aralığında farklı kişinin kullanması ile sağlanır.

Yukarıdaki şekilde birim zamanda aynı hat üzerinden 4 farklı kişinin iletişim şekli gösterilmiştir. Buna göre 1. kişi (kırmızı renkle veri transferi yapan kişi) diğer 3 kişi verisinin yolladıkları hattan verisini yollayabilmektedir. Aynı hattan 4 farklı kişinin yollayabilmesi için farklı frekans aralıklarını kullanmaları gerekir. Bu yöntemde diğer devre anahtarlamalı yöntemlerde olduğu gibi herhangi bir kişi veri yollamasa bile ona ayrılan frekans dilimini başka kimse kullanamaz ve hattın bu kısmı boş kalır.

#### **SORU 75: zaman bölmeli çoklama (time division multiplexing, tdm)**

Tek hat üzerinden birden fazla kişinin iletişimine izin veren devre anahtarlamalı ağ tipidir. Buna göre hattı birden fazla kişi kullanmak için farklı zaman aralıklarını kullanırlar. Örneğin bir hattı 4 farklı kişi kullanmak istiyorlar. Hattın kullanımı birim zamanı 4 farklı kişiye bölünerek her birim zamanda farklı kişinin kullanması ile sağlanır.

Yukarıdaki şekilde birim zamanda aynı hat üzerinden 4 farklı kişinin iletişim şekli gösterilmiştir. Buna göre 1. kişi (kırmızı renkle veri transferi yapan kişi) diğer 3 kişi verisinin yolladıktan sonra hattan verisini yollayabilmektedir. Bu yöntemde diğer devre anahtarlamalı yöntemlerde olduğu gibi herhangi bir kişi veri yollamasa bile ona ayrılan zaman dilimini başka kimse kullanamaz ve hat boş kalır.

Bu bağlantı şekli ikiye ayrılır:

**eş zamanlı (zaman uyumlu) zaman bölmeli çoklama (synchronous TDM):** Buna göre gönderen ve alan taraflar tam olarak hangi zamanda göndereceklerini ve alacaklarını bilmektedirler. Bu zaman aralıklarında iletişimi yaparlar. Bu sistemin tam çalışabilmesi için alıcı ve verici sistemlerin eş zamanlı olmaları ve birbirlerinden haberdar olmaları gerekir.

**zaman uyumsuz zaman bölmeli çoklama (asynchronous TDM):** Bu yapıya göre gönderen ve alan tarafların birbiri ile uyumlu olması gerekmemektedir. Bu sistemde her zaman biriminde gönderilen bilgi ilave olarak kimden geldiği ve kime gittiği gibi kimlik bilgilerini tutmaktadır. Bu sayede alıcı ve vericiler arasında iletişim yapılabilir.

### **SORU 76: sunucu (server)**

Bir hizmet yada kaynağı arz eden bilgisayara verilen isimdir. Buna göre hizmet veya kaynağı sunan bir sunucu bulunmakta ve istemciler bu sunucuya bağlanarak bu hizmetten faydalanmaktadır. Bu bağlantı sistemine istemci /sunucu ( arz /talep, client/server) ismi verilmektedir.

Öneğin internete bağlı bir kullanıcı, internet üzerinde bir web sayfasını görüntülemek istesin. Bu durumda sayfaya bağlanmakta ve sayfanın içeriğini kendi bilgisayarına çekerek internet görüntüleyicisi ile göstermektedir. Bu görüntüleme esnasında talep edilen web sayfası için istemci taraf ilgili sunucuya bağlanmakta ve sayfayı talep etmektedir. Sunucu ise bu talebe cevap olarak sayfayı istemciye yollamaktadır. Görüldüğü üzere talep eden taraf istemci arz eden taraf ise sunucudur. Sonuçta bir sunucuya çok sayıda istemci bağlanarak taleplerde bulunabilmekte ve sunucu da bu talepleri cevaplamaktadır. Dolayısıyla istemci/sunucu mimarisi çoktan teke bir bağlantı şeklidir. Bu bağlantı şekli aynı zamanda özdeş olmayan uçlar arasında (istemci/sunucu) yapıldığı için asimetrik olarak da kabul edilir.

### **SORU 77: istemci (client, talebe)**

bir hizmet yada kaynağı talep eden istekte bulunan taraftır. Buna göre hizmet veya kaynağı sunan bir sunucu bulunmakta ve istemciler bu sunucuya bağlanarak bu hizmetten faydalanmaktadır. Bu bağlantı sistemine istemci /sunucu ( arz /talep, client/server) ismi verilmektedir.

Öneğin internete bağlı bir kullanıcı, internet üzerinde bir web sayfasını görüntülemek istesin. Bu durumda sayfaya bağlanmakta ve sayfanın içeriğini kendi bilgisayarına çekerek internet görüntüleyicisi ile göstermektedir. Bu görüntüleme esnasında talep edilen web sayfası için istemci taraf ilgili sunucuya bağlanmakta ve sayfayı talep etmektedir. Sunucu ise bu talebe cevap olarak sayfayı istemciye yollamaktadır. Görüldüğü üzere talep eden taraf istemci arz eden taraf ise sunucudur. Sonuçta bir sunucuya çok sayıda istemci bağlanarak taleplerde bulunabilmekte ve sunucu da bu talepleri cevaplamaktadır. Dolayısıyla istemci/sunucu mimarisi çoktan teke bir bağlantı şeklidir. Bu bağlantı şekli aynı zamanda özdeş olmayan uçlar arasında (istemci/sunucu) yapıldığı için asimetrik olarak da kabul edilir.

### **SORU 78: vekil sunucu (proxy server)**

vekil sunucunun görevi, istemcilerin (client) taleplerini farklı sunuculara taşımaktır. Diğer bir deyişle sunucu / istemci (client / server ) mimarisindeki paketlerin üçüncü bir sunuc üzerinden taşınması durumunda bu sunucuya vekil sunucu adı verilir.

Paketlerin üzerinden geçmesi sayesinde vekil sunucular aşağıdaki görevleri yerine getirebilecek bir avantaj elde etmiş olurlar.

**İçerik bazlı filitreleme (content based filtering):** Bütün paketler sunucudan geçtiği için geçen paketlerin içeriklerine bakarak istenilmeyen bir bilgi veya adres taşınması durumunda paketin engellenmesi sağlanabilir. Bu yaklaşımda vekil sunucuların üzerinde kara liste

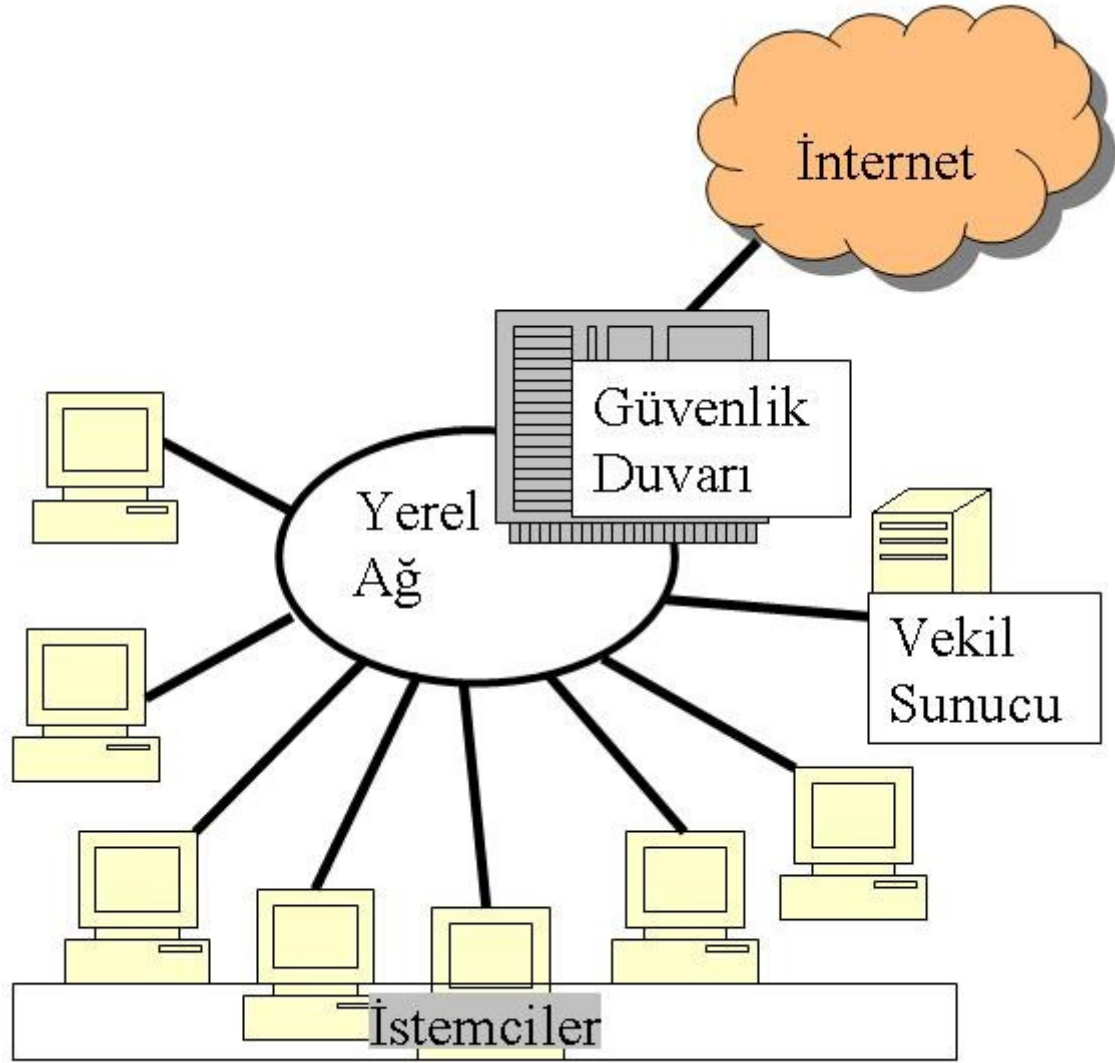
(black list) üzerinde yazılı olan kelimeleri içeren adreslere girilmesi engellenmiş olur. (örneğin adresin içerisinde “oyun” kelimesi geçiyorsa ve engellenmesi isteniyorsa, sunucunun kara listesine oyun kelimesi eklenir. ) Kara liste kullanılımasının oluşturduğu bir problem, kara listede olan kelimelerin istenmeyen adresleri de engellemesidir. Örneğin oyun kelimesi engellenince, hiç alakası olmayan “koyun” kelimesi de engellenmiş olur. Bunu çözmek için beyaz liste (white list) tanımlanarak bu kelimeler izin verilenler arasına eklenir.

**Ön Bellekleme (Caching):** Bütün bilgiler vekil sunucu üzerinden geçtiği için, geçen bu bilgilerin birer kopyası vekil sunucuda saklanarak ileride yapılan taleplere vekil sunucudan cevap verilebilir. Örneğin bir istemcinin (Client) internet üzerinde bulunan bir siteyi talep ettiğini düşünelim. Bu sitenin içeriği istemciye (client) ulaştırılırken vekil sunucu üzerinden taşınacak ve bu sırada bir kopyası vekil sunu üzerinde tutulacaktır. Ağda bulunan farklı bir bilgisayarın veya aynı bilgisayarın bu bilgiyi tekrar talep etmesi durumunda, vekil sunucu web sayfasının değişikliğini kontrol ederek şayet sitede bir değişiklik yapılmadıysa kendi üzerindeki kopyadan istemciye dosyaları iletir. Şayet sitede değişiklik varsa bu bilgileri yeniden ön hafızasına (cache) alarak daha sonraki talepler için tutar.

**Diğer kullanım alanları:** Günümüzde vekil sunucular, yukarıda bahsedilen iki ana amacının dışında amaçlar içinde kullanılmaktadırlar. Örneğin kimliksiz (anonymous) kayıt için kullanılmaktadırlar. Buna göre internet üzerinde dolaşan her bilgi bir kimlik bilgisi ile dolaşmaktadır. Bu durumdan rahatsız olan kişiler veya kimliğini gizlemek isteyen kişiler paketlerini bir vekil sunucu üzerinden taşıyarak internet üzerinde dolaşan paketlerini sanki bu sunucunun paketleriymiş gibi gösterebilirler.

Vekil sunucular kurulum tiplerine göre saydam (transparent) veya saydam olmayan (non-transparent) şeklinde sınıflandırılabilirler. Basitçe saydam sunucular (Transparent proxy server) internet üzerinde herhangi bir ayar gerektirmeksizin paket trafiğini kendi üzerilerine alırlar ve paket kontrolünü kendi üzerilerinden gerçekleştirirler. Saydam olmayan vekil sunucular (non-transparent proxy server) ise tam tersi şeklinde istemcilerde (client) bir sunucu ayarlaması ve hatta bazan kullanıcı adı ve şifre girilmesini gerektirirler.

Aşağıda vekil sunucu kullanılan örnek bir ağ yapısı verilmiştir:



Yukarıdaki şekle göre, yerel ağda bulunan bilgisayarların internet çıkışları tek bir nokta üzerinden yapılmaktadır. (Genelde bu nokta gateway veya ağ geçidi olarak adlandırılır). Dolayısıyla ağda bulunan her bilgisayarın internet paketleri bu noktadan geçmektedir. Tam bu noktaya yani ağdaki bilgisayarları internete bağlayan bu ağ geçidine bir güvenlik duvarı (firewall) kurularak bütün bilgisayarların geçişi engellenir. Bu engelleminin yanında tek izin verilen geçiş ağda bulunan vekil sunucu olarak ayarlanır. Yani içeriden vekil sunucu dışında kimse internete erişememektedir. Yukarıdaki ayarlama sonucunda, içerideki her bilgisayarın internet erişimi için vekil sunucuya başvurmakta ve vekil sunucu da her paket talebine kendisine güvenlik duvarında tanınan hak sayesinde internet üzerinden talep edilen bilgileri içerideki bilgisayarlara taşımaktadır. Bu sırada da yukarıda bahsedilen ön bellekleme ve içerik bazlı filtreleme işlemlerini yapmaktadır.

#### **SORU 79: Bir tümleyeni**

Konunun diğer isimleri : (1 tümleyeni, İşaretili sayı gösterimi, Ones' Complement, 1's Complement, Signed number representations)

ikilik tabandaki bir sayının 1 tümleyeni her sayının tersidir. Örneğin sayı:

10110011

olarak verilmiş olsun. Bu sayının 1 tümleyeni:

01001100

olarak bulunur.

Bu bilgi 2 tümleyeninin hesabında da kullanılmaktadır.

Bir tümleyeni aynı zamanda sayının eksi değeri olarak gösterilmesine de yararmaktadır. Aşağıdaki tabloda sayıların alabileceği değerler ve bu sayıların ikili ve onluk tabandaki gösterimleri verilmiştir:

İkili sayı	1 tümleyeninin 10luk	Yönsüz sayı
00000000	0	0
00000001	1	1
...	...	...
01111101	125	125
01111110	126	126
01111111	127	127
10000000	-127	128
10000001	-126	129
10000010	-125	130
...	...	...
11111110	-1	254
11111111	-0	255

yukarıdaki tabloda ikilik sistemde bir takım sayılar verilmiştir. Tablonun ikinci kolonunda bu sayıları 1 tümleyeni olarak yorumladığımızda 10luk sistemdeki karşılıkları, üçüncü kolonda ise bu sayıları normal birer ikilik sayı gibi görüp, 10luk sisteme çevirince çıkan karşılıkları verilmiştir.

Buna göre 0 sayısının bütün bitlerinin tersi -0 veya örneğin 10 sayısının bütün bitlerinin tam tersi -10 olmaktadır.

Bu ayrımı ilk bit belirlemektedir bu bit'e sign bit (yön biti) de denilmektedir.

### **SORU 80: İki tümleyeni**

Konunun diğer başlıkları: 2 tümleyeni, two's complement

Bilgisayar bilimlerinde, sayılar genelde ikilik tabanda tutulmaktadır. Değerleri ikilik tabanda göstermenin bir devamı olarak eksi sayı ve artı sayıları da ayırmak gerekmektedir. bir tümleyeni gibi iki tümleyeni de eksi sayıları gösterim biçimlerinden birisidir. iki tümleyenini almak için önce bir tümleyeni alınır ardında sayıya ikilik tabanda 1 eklenir.

### Örneğin

11011001

sayısının bir tümleyeni aşağıda verilmiştir:

00100110

bu sayıya 1 eklenerek, iki tümleyeni elde edilir:

00100111

Bu sayı aynı zamanda orjinal sayı olan 11011001 sayısının da negatifi gösterimi olarak kullanılabilir.

Bunu bir örnek ile göstermek gerekirse, aşağıdaki çıkarma işlemini ele alalım:

11001001

10110101

-

-----

00010100

bilindiği üzere aslında çıkarma işlemini, çıkarılan sayının negatifi alıp toplama olarak da yorumlayabiliriz.

Bu durumu aynı örnek için tecrübe edelim. Öncelikle çıkarılan sayı olan 10110101 sayısının negatifi alalım, yani iki tümleyeni: 01001010 sayısı elde edilir. Şimdi bu sayının gerçekten negatif olduğunu yukarıdaki örneği toplamaya çevirerek görelim:

11001001

01001010

+

-----

100010100

Görüldüğü üzere elde edilen sonucun başında bulunan 1 atılırsa, ilk işlemde çıkan sonuç ile aynıdır.

### **SORU 81: sanal devre (Virtual Circuit)**

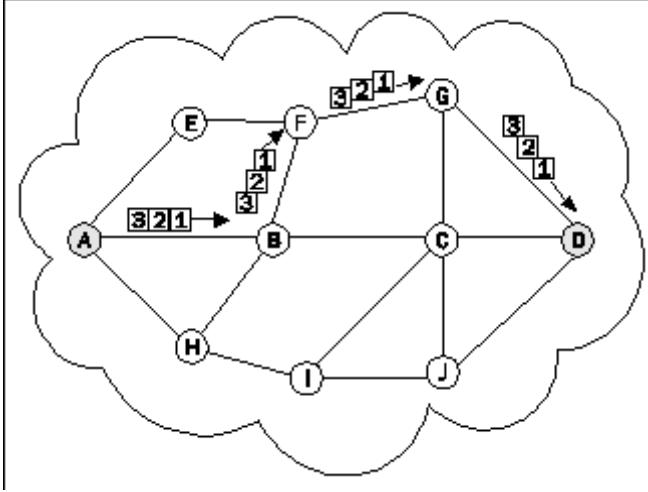
Bilgisayar ağlarından, paket anahtarlama ağı tipinde kullanılan iki alternatiften birisidir. (diğeri datagram tipi paketlerdir.)

sanal devre uygulamasında bir paket vasıtası ile yol boyunca geçilecek olan bütün cihazlara paket bilgileri tanıtılarak bundan sonra geçecek olan paketlerin tam olarak nereye yollanacaklarını bildirilir. Bu ilk paketten sonra geçen her paket belirli bir sıra ile tam olarak hedefe yollanırlar.

sanal devre uygulamaları genelde güvenilir (reliable) bir ortam sunarlar (istisnaları mevcuttur).

Sonuçta oluşturulan ortam devre anahtarlama (circuit switched) ağlara benzemektedir. Her ikisi de bağlantı yönlendirmeli (connection oriented) ağlardır.





şekilde sanal devre oluşturulmuş bir ağda A bilgisayarından D bilgisayarına paket akışı gösterilmiştir. Buna göre paketlerin her birisi aynı yerden ve sırayla geçmektedir. Bu ağın hazırlanması aşamasında yolda bulunan her cihaz bu akış için özel olarak ayarlanmıştır.

Bu ağ ayarlanması işlemi (yani yolda bulunan cihazların ayarlanması) değişik seviyelerde yapılabilir. TCP katmanları düşünülürse 4

### **SORU 82: Devre anahtarlama (Circuit Switching)**

### **SORU 83: Paket anahtarlama (Packet Switching)**

Bu iletim yönteminde, kanal üzerinde gönderilecek olan bilgiler paketlere konularak iletilir. Örneğin uzun bir yazının karşı tarafa yollanacağını düşünelim. Bu durumda yazı paket boyutlarında kesilerek her pakete yazının bir kısmı konulur. Paketlerin üzerine kullanılan alt iletim yöntemlerine göre (örneğin datagram veya sanal devre (Virtual Circuit) ) değişik bilgiler eklenmektedir bu eklenen bilgiler başlık (header) veya sonluk (footer ) olarak da isimlendirilmektedir, ve paketin geldiği yeri gideceği yeri gibi bilgileri içerir.

Verinin paketler halinde yollanmasının bir avantajı, herhangi bir şekilde hattın bir veya birden fazla kullanıcıya tahsis edilmemesidir. Yani hat kullanıcı tarafından kullanılmadığı zamanlarda başka kullanıcılar tarafından kullanılabilir.

Örneğin iletişim yapacak olan iki uç arasındaki bağlantı hızı 1000 kps olsun. Kullanıcıların %10 aktif olduklarını düşünelim. Yani kullanıcılar zamanın %10'unda iletişime geçiyor geri kalan zamanlarda ise iletişim ihtiyaçları olmuyor. Ve kullanıcıların 100kps iletişim ihtiyaçları olduğunu düşünelim. Yani kullanıcılar aktif olduklarında hat kullanımları 100kps olarak verilmiş. Bu durumda kaç kullanıcı hat üzerinde taşınabilir?

Bu soruda örneğin 10 kişinin aynı anda paket anahtarlama kullandığı düşünülürse, binom dağılımı kullanılabilir. Bilindiği üzere N denemedeki başarı oranı binom dağılımında aşağıdaki şekilde modellenir:

$$P\{y=k\} = \binom{n}{k} p^k q^{n-k}, p+q=1, k=0,1,\dots,n$$

Yukarıdaki formülde, N denemedeki k başarının oluşma sayısı

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

bu yolların her birinin olasılığını yazarsak:

$p^k(1-p)^{n-k}$  denkleminde, (1-p) başarısızlık oranı n-k ise başarısızlık sayısını verir. dolayısıyla 10 kişi için başarı oranı, yani 10 kişinin aynı anda paket transfer etme olasılığı:

$$\begin{aligned} &= \binom{n}{k} (p)^k (1-p)^{n-k} \\ &= \binom{10}{1} \left(\frac{1}{10}\right)^1 \left(\frac{9}{10}\right)^9 \\ &= \left(\frac{10!}{1!.9!}\right) \left(\frac{1}{10}\right)^1 \left(\frac{9}{10}\right)^9 \\ &= 0,387420489 \end{aligned}$$

dolayısıyla 0,387420489 olasılıkla (yani %38) aynı anda 10 kişi birden transfer yapabiliyormuş. Bu denklemi bir de 40 kişiden 10 kişinin transferi için denersek:

$$\begin{aligned} &= \binom{n}{k} (p)^k (1-p)^{n-k} \\ &= \binom{40}{10} \left(\frac{1}{10}\right)^{10} \left(\frac{9}{10}\right)^{30} \\ &= \left(\frac{40!}{10!.30!}\right) \left(\frac{1}{10}\right)^{10} \left(\frac{9}{10}\right)^{30} \\ &= 847660528 \cdot 10^{-10} \cdot 0,04239115 \\ &= 0.03 \end{aligned}$$

Dikkat edilirse yukarıdaki denklem sadece 40 kişiden 10 kişinin aynı anda transfer yapması durumudur. Dolayısıyla 10'dan fazla kişinin aynı anda transfer yapması durumunu hesaplamak için aynı denklem 11, 12, 13 ... 40 kişi için ayrı ayrı çözülmeli ve çıkan sonuçlar toplanmalıdır.

Bu işlem yapıldığında örneğin 35 kişi için 0.004 çakışma olasılığı ile aynı anda hattın kullanılabileceği anlaşılmıştır.

aynı durum devre anahtarlama yöntemi kullanılarak yapılsaydı hattı sadece 10 kişinin (1000kbs / 100kbs) kullanabileceği düşünüldüğünde bu sayı oldukça başarılı kabul edilebilir.

Paket anahtarlama sistemlerinin bir dez avantajı maliyetlerini yüksek olmasıdır.

#### **SORU 84: Kişisel AĞ (personal area network, PAN)**

Günümüzde gelişen kişisel cihazların sayısı ve kullanışlılığı ile literatüre giren bir terimdir. Kişisel ağ ile kast edilen kişisel cihazların bağlanması sonucu elde edilen ağıdır. Örneğin bir

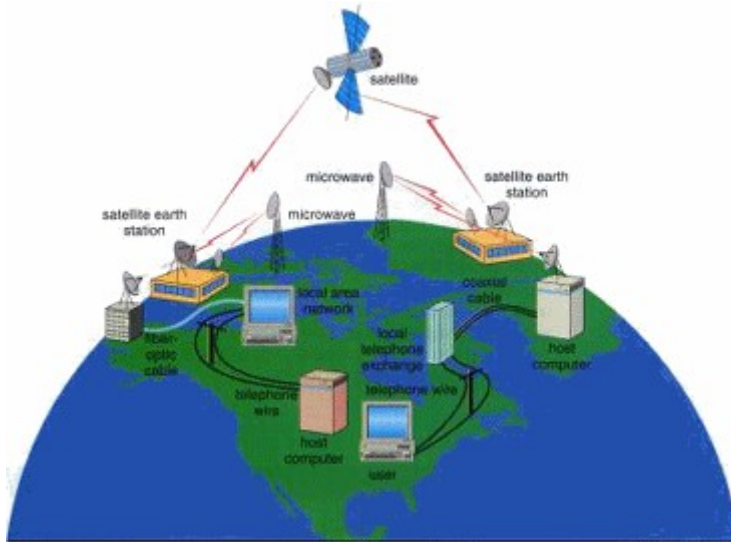
kişinin, masa üstü bilgisayar, diz üstü bilgisayar, cep telefonu, cep bilgisayar gibi aygıtlarını birbiri ile konuşturmak için kurduğu ağıdır. Genellikle bir iki bilgisayar ve kişisel cihazdan oluşur. Bağlantı ortamı olarak, kablosuz bağlantı teknolojileri veya USB gibi bağlantılar kullanılır.

şekilde kişisel cihazların (printer, scanner, fotoğraf makinası, dizüstü bilgisayar gibi) birbiri ile olan bağlantısı temsil edilmiştir.

Örneğin 8 aygıtın bağlanabildiği bluetooth bağlantısı (piconet denilmektedir) veya 10 adet piconet'in bağlandığı scatternet gibi ağlar bu kişisel ağ uygulamalarıdır.

### **SORU 85: Geniş Ağ, WAN (Wide Area Network)**

Coğrafi olarak en geniş ölçekli ağ çeşididir. Dünya üzerine dağılmış ağ yapılarını ifade eder. Genelde çok sayıda yerel ağ, yerleşke ağı gibi ağların birleşmesinden oluşan büyük yapıdır.



geniş ağ'a bağlı milyonlarca bilgisayar olabileceği gibi dünyanın iki ucundaki iki bilgisayar da bir geniş ağ kabul edilebilir.

geniş ağlar üzerinde iki farklı iletişim yöntemi kullanılabilir bunlar aşağıda listelenmiştir:

Paket	anahtarlama	(Packet	Switching)
Devre	anahtarlama	(Circuit	Switching)

Diğer		ağ	ölçekleri:
Kişisel	Ağ,	PAN (Personal	Area Network)
Yerleşke	Ağı,	MAN (Metropolitan	Area Network)
Yerel Ağ Bağlantısı	LAN, (Local Area Network)		

### SORU 86: Yerleşke Ağı, MAN (Metropolitan Area Network)

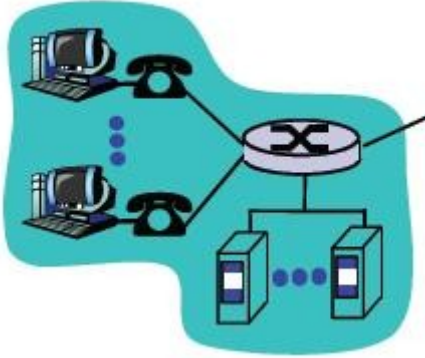
Yerel ağdan daha büyük ancak geniş ağdan daha küçük, genelde bir üniversite veya şirket yerleşkesi içerisinde birden fazla bina arasında kurulan ağlara verilen isimdir.

şekilde çeşitli binalar arası kurulmuş olan bir ağ yapısı gösterilmektedir.

Diğer			bağlantı		ölçekleri
Kişisel	Ağ,	PAN	(Personal	Area	Network)
Yerel	Ağ	Bağlantısı	LAN,	(Local	Area
Geniş Ağ,	WAN (Wide Area Network)				Network)

### SORU 87: Yerel Ağ Bağlantısı (YAB, Local Area Network, LAN)

küçük çaptaki bilgisayar ağlarına verilen isimdir. Ağların ölçeklendirilmesi için kullanılan terimlerden birisidir. Genelde birbirine yakın, ortalama 10 ile 50 arası bilgisayarın bağlı olduğu ağlara verilen isimdir. Teorik olarak 2 bilgisayarın birbirine bağlanması da yüzlerce bilgisayarın birbirine bağlanması da yerel ağ olarak isimlendirilebilir. Burada ölçekleme daha çok bağlanan bilgisayarların birbirine olan uzaklığıyla ölçülür. Örneğin aynı binadaki bilgisayarların bağlanması gibi. Bağlantıda kullanılan cihazlarda ölçeklendirmede önemlidir. Genelde yerel ağ tanımı altında basit HUB, Switch gibi cihazlarla bağlanan 1. seviyeden komşu bilgisayarlar kast edilir. Coğrafi mesafe büyüdükçe yerel ağ yerine başka terimler kullanılır. Örneğin dünyanın iki ucundaki 2 bilgisayar birbirine bağlanırsa buna yerel ağ denilemez.



Şekilde basit bir yerel ağ bağlantısı görülüyor. Bir bağlayıcı cihaz (Switch veya HUB) üzerinden birden fazla bilgisayar birbiri ile iletişim halindedir. Ayrıca bu yerel ağ, dışarıya da bağlanmıştır.

Diğer			bağlantılar	
Kişisel	Ağ,	PAN	(Personal	Area
Yerleşke	Ağı,	MAN	(Metropolitan	Area
Geniş Ağ,	WAN (Wide Area Network)			Network)

### SORU 88: SNR (Signal to Noise Ratio, İşaret Gürültü Oranı, S/N)

SNR-> Kısaca işaret üzerinde olan bozulmanın işaretin ne kadarına tekabül ettiğini hesaplamak için kullanılan bir yöntemdir. Bu yöntemde negatif ve pozitif gürültülerin işarete yaptığı etkiyi sıfırlamaması için, işaretlere yapılan etkinin karesi alınmıştır.

$$SNR = \frac{\sum \sum \text{Gürültülü Re sim}(x, y)^2}{\sum \sum (\text{Gürültülü Re sim}(x, y) - \text{Orjinal Re sim}(x, y))^2}$$

verilen bir resimin gürültü uygulanmış hali ve orjinal resim arasındaki bozulma miktarı hakkında fikir edinmeye yarayan formüldür. Her imgecik (piksel)için ayrı ayrı farklar hesaplanarak karelerinin oranının toplamını verir.

SNR değeri ne kadar küçükse resimdeki bozulma o kadar çoktur yorumu yapılabilir. Resimdeki bozulma sonsuza giderken SNR değeri 0'a yaklaşır.

### SORU 89: Gauss Gürültüsü (Gaussian Noise)

Sinyal işleme veya resim işleme gibi konularda işlenen veri üzerinde istenmeyen değişimler olmasına gürültü denilir. Buna göre örneğin bir kaynaktan bir hedefe giden resim veya ses üzerinde yolda istenmeyen değişimler oluşursa resim ve seste bozulmalar olur. Örneğin aşağıdaki orjinal resmi ele alalım:

Bu resime gauss gürültüsü (gaussian noise) uygulanırsa aşağıdaki şekilde olur:

yukarıdaki resim orjinal resim üzerine sigma = 14 değeri ile gaus gürültüsü uygulanmış halidir. Görüldüğü üzere resimde belirgin bir bozulma bulunmaktadır. Bu bozulma orjinal resimlerdeki piksel değerlerinin değişmesi şeklinde yorumlanabilir.

Bu değişim oranlarının belirli bir histogram değeri içinde olması durumunda gürültü hakkında yorum yapılabilir. Örneğin gauss gürültüsünde resime yapılan etki, resimdeki ortalama gri ton üzerinde sıfır (0) 'dır. Yani gauss dağılımı düşünüldüğünde bu dağılımın ortalama değerde en yüksek olduğunu biliyoruz. İşte en yüksek bu değer ortalama gri tondan uzaklığı da sıfırdır. Örneğin ortalama değere en uzak olan gri ton ise resimde en çok değişim oranını belirler.

Bu durum basitçe şöyle ifade edilebilir. Verilen sigma değeri için bir gauss dağılımı üretilir. Bu dağılım uygulanacak olan gürültünün histogramıdır. Yani bu değerler resmi değiştirmede kullanılacaktır.

Üretilen gürültü değerleri teker teker resime toplama olarak uygulanırlar.

Yani örneğin 255 gri tonlu 256×256 boyutlarında (256 x 256 = 65536 piksellik ) bir resimde ortalama gri ton değeri 127 bulunmuş olsun (resmin histogramı çizildiğinde çıkan ortalama değer). Bu ortalama değer 127 sağında ve solunda kalan her sayı için (örneğin ortalama 127 ise hemen sağında 128 ve hemen solunda 126 var demektir ve diğer sayılar bu şekilde sıralanmıştır) kaçar tane oldukları dağılımda görülmektedir. Dolayısıyla dağılım aslında 0-255 aralığındaki her gri ton için bir değer üretmiştir.

Bu üretilen değerlerin ortalama değere (bu örnekte 127) olan uzaklığı ise gürültü uygulandığında resimdeki piksel değerlerinin ne kadar değişeceğini verir. Örneğin 97 sayısı resimdeki bir piksele uygulandığında, bu o pikselin değerinin 30 azalacağını ifade eder ( $127 - 97 = 30$ ).

Bu üretilen gürültü dağılımındaki her sayıya karşılık gelen değer ise o sayıdan kaç tane uygulanacağını verir. Örneğin dağılım üretildikten sonra 97 sayısının karşılığı olarak 614 bulunduğunu düşünelim. Bu, resimde bulunan rasgele 614 pikselden 30 değerinin çıkarılacağını anlatmaktadır. Aynı durum dağılımdaki diğer her gri ton değeri için de geçerlidir.

Yani benzer şekilde örneğin üretilen gauss dağılımında 154 numaralı sayının karşılığı 50 olarak bulunmuşsa, bu durum resimdeki rasgele 50 piksele 27 ekleneceği anlamına gelmektedir ( $154 - 127 = 27$ ).

### **SORU 90: Gürültü (Noise)**

Sinyal işleme veya resim işleme gibi konularda işlenen veri üzerinde istenmeyen değişimler olmasına gürültü denilir. Buna göre örneğin bir kaynaktan bir hedefe giden resim veya ses üzerinde yolda istenmeyen değişimler oluştursa resim ve seste bozulmalar olur. Örneğin aşağıdaki orjinal resimi ele alalım:

Bu resime gauss gürültüsü (gaussian noise) uygulanırsa aşağıdaki şekilde olur:

yukarıdaki resim orjinal resim üzerine  $\sigma = 14$  değeri ile gauss gürültüsü uygulanmış halidir. Görüldüğü üzere resimde belirgin bir bozulma bulunmaktadır. Bu bozulma orjinal resimlerdeki piksel değerlerinin değişmesi şeklinde yorumlanabilir.

### **SORU 91: bit (ikil)**

Bilgisayar dünyasında ikili tabandaki (binary) tek haneli bir sayıyı ifade eder. Yani bir bit değeri 1 veya 0 olabilir. Bu aslında elektronik sinyali olarak yüksek (1) veya düşük (0) gerilimde akım demektir.

bir bit, 1 veya 0 değeri alabildiğine göre her bit değerinin 2 farklı değer alması mümkündür. Bu durumda örneğin iki sistemde yazılmış 8 haneli bir sayı (8 bitlik bir sayı) 2 üzeri 8 farklı değer = 256 farklı değer alabilir. Örneğin 11010010 sayısı, 8 bitlik bir sayıdır.

8 bitlik sayılara bilgisayar dünyasında kısaca byte adı da verilmektedir.

Bit değerleri için kullanılan birimler aşağıda listelenmiştir:

<u>Metrik</u> Gösterim		<u>İkili</u> (binary) Gösterim	
İsim	Metrik Değeri	İsim	Değeri
<u>kilobit</u> (kbit)	$10^3$	<u>kibibit</u> (Kibit)	$2^{10}$
<u>megabit</u> (Mbit)	$10^6$	<u>mebibit</u> (Mibit)	$2^{20}$
<u>gigabit</u> (Gbit)	$10^9$	<u>gibibit</u> (Gibit)	$2^{30}$
<u>terabit</u> (Tbit)	$10^{12}$	<u>tebibit</u> (Tibit)	$2^{40}$
<u>petabit</u> (Pbit)	$10^{15}$	<u>pebibit</u> (Pibit)	$2^{50}$
<u>exabit</u> (Ebit)	$10^{18}$	<u>exbibit</u> (Eibit)	$2^{60}$
<u>zettabit</u> (Zbit)	$10^{21}$	<u>zebibit</u> (Zibit)	$2^{70}$
<u>yottabit</u> (Ybit)	$10^{24}$	<u>yobibit</u> (Yibit)	$2^{80}$

### **SORU 92: Histogram ( tekrar dağılımı, histogram)**

tekrarlı sayılar içeren bir dizideki her sayının tekrar miktarını veren dağılım grafiğidir. Örneğin 1'den 3'e kadar sayıların olabildiği aşağıdaki seriyi ele alalım:

1,3,1,2,3,2,1,1,1,2,2,3,1,3,3

yukarıdaki 15 sayıda sadece 1,2 ve 3 sayıları geçmektedir. Bu sayıların histogramı:

1->6

2->4

3->5

olarak gösterilebilir. Yani seride kaç kere 1, kaç kere 2 geçtiğinin gösterilmesidir. Grafik olarak çizilecek olursa:

Dolayısıyla bir bilginin histogram değeri o bilgiden kaç tane olduğu ile ifade edilebilir.

### **SORU 93: DHCP (Dynamic Host Configuration Protocol, Dinamik Bağlantı Ayarlama Protokolü)**

Protokol IP (internet protocol, internet protokolü) için bir hizmet protokolüdür. Amacı bilgisayarın bağlantı sırasında kullanacağı IP Adresini dinamik olarak bir sunucudan alması bu sayede de her bağlantıda farklı bir IP adresi almasıdır. Bu kullanımın bir diğer avantajı da her bilgisayara sabit bir IP adresi atanmayarak mevcut IP adreslerinin daha tasarruflu kullanılmasıdır.

Sağladığı diğer bir avantaj, her bilgisayarda teker teker IP adreslerinin yazılması gerekmemektedir. Bu sayede ağ yönetimi daha kolay hale getirilmektedir.

### **SORU 94: IP Adress (Internet Protokolü Adresi, Internet Protocol Address)**

Adresin amacı internet üzerinde bağlı olan bilgisayarların, ağ üzerinde tanınmalarını sağlayan bir kimlik bilgisinin olmasıdır. IP adresi 4 parçalıdır:

xxx.xxx.xxx.xxx

şeklinde

Bu numaralar 0 ile 255 arasında yazılabilir. olabilir.

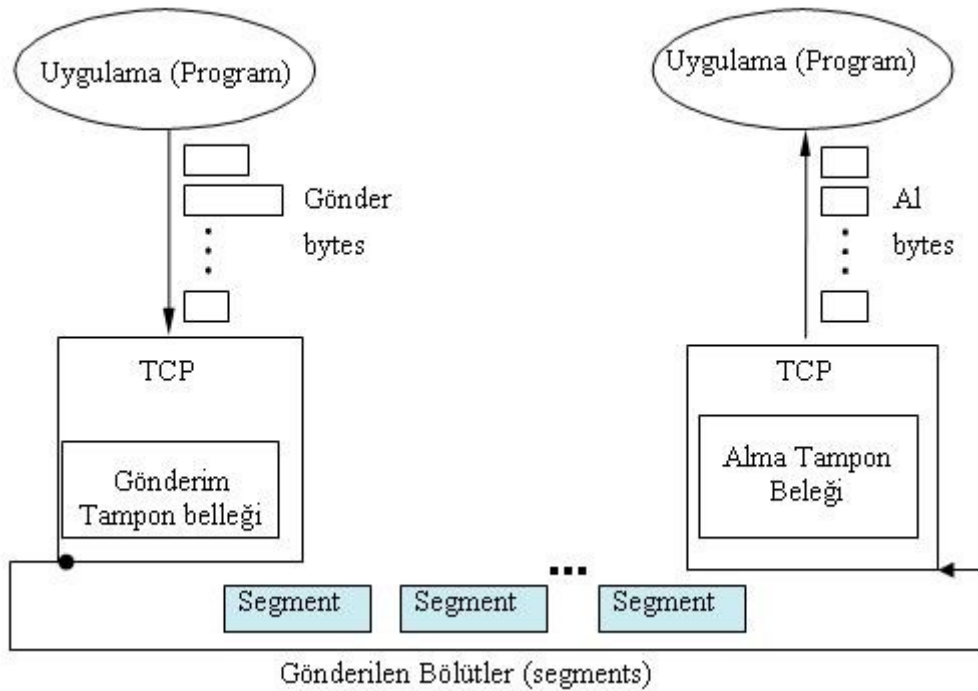


IP v.4 (Internet protokolü versiyon 4) için bu sayı 32 bitlik ulaşılan azami sayıdır. IP v.6 (Internet protokolü versiyon 6) için bu sayı 128 bitlik adreslemeye çıkarılmıştır. Karşılaştırma amaçlı olarak: IPv4 için sahip olunabilen adres sayısı: 4,294,967,296 iken IPv6 ile bu sayı 3,402,823,669,293,846,346,337,467,431,768,211,456 kadar adrese çıkarılmıştır.

IPv4 adresine örnek: 192.168.0.1 şeklinde 0-255 arasında 4 sayıdan oluşmakta ve sayılar onluk tabanda olmaktadır. IPv6 adresine örnek: 2001:0db8:85a3:08d3:1319:8a2e:0370:7334 şeklinde 8 farklı 4 grup hexadecimal (16lık tabanda) sayıdan oluşmaktadır.

### **SORU 95: TCP (Transmission Control Protocol (Nakil Hakimiyet Anlaşması , İletim Kontrol Protokolü))**

Ağ üzerinde emin (reliable) bir nakil ortamı sağlar. TCP, paket değişmeli (packet switching) bir protokol olup amaç gönderilmek istenen paketlerin karşı tarafa güvenli bir şekilde ulaştırılmasını sağlamaktır. tcp protokolü uygulamalar ile ağ üzerindeki alt protokoller arasında bir ara katmandır. Bu durum aşağıdaki grafikte gösterilmiştir:



OSI katmanına alternatif olarak 5 seviyeli bir protokoldür. Ağ katmanı (network layer) olarak IP (internet packet) protokolünün üzerinde çalışan TCP, gönderilen paketlerin karşı tarafa ulaştıklarının onaylanması ile daha emin bir ortam sağlamış olur.

kayan pencere (sliding window) algoritması ile gönderilen paketler sıralı olarak karşı tarafa yollanır ve karşı taraftan toptan onaylama (cumulative acknowledgement) yöntemi ile sağlıklı ulaştığından emin olunur.



Aynı zamanda ulaşan her paketin içeriğinin bozulup bozulmadığı da kontrol edilmektedir. Bu işlem için Internet toplam kontrolü (Internet Checksum) yaklaşımı kullanılır.

### **SORU 96: Paket değişimli (Packet Switching) Ağlar**

Paket değişimli ağlar (Packet switching networks), üzerlerinde bilgilerin paketler ile yollandığı ağ tipleridir. Bu ağ tipinde veri kapsülleme (encapsulation) yöntemi ile bir veya birden çok katmanda paketlenir ve kapüller bu geçtikleri katmanların bilgilerini içerir. Örneğin OSI modelinde 7 farklı katmanın her birinde ilave bilgiler eklenerek paketler hazırlanır.

Bu hazırlanan bilgi ağ üzerinde her düğüm (node)'a uğrayacak şekilde gönderilir. Yani paketlerin üzerinde nereye gidecekleri ve kime teslim edilecekleri yazılıdır ve her noktada bu paketler bir sonraki daha yakın noktaya yönlendirilir.

### **SORU 97: Devre değişimli (Circuit Switching) ağlar**

Ağ bağlantı çeşitlerinden birisidir. Paket değişimli ağlara alternatiftir ve iki uç arasında (bağlantı yapan iki bilgisayar gibi), özel bir hat kurulmuş gibi çalışır. En klasik örneği ilkel telefon santrallerinde bir operatörün, konuşmak isteyen iki kişiyi tek bir kablo üzerinden bağlı hale getirmesi olarak düşünülebilir.

Hat bir kere iki uç arasında tahsis edildikten sonra iletişim bitip bağlantı kapatılana kadar açık kalmak zorundadır, bu durum sistem kaynaklarının uzun süre bir bağlantıya tahsis edilmesinden dolayı kullanışsız olarak görülür, ancak uzun süre sabit hızda bağlantı kuran sistemlerde kullanışlı olabilir.

Daha fazla bilgi için:  
Paket değişimli ağlar  
frekans bölmeli çoklama (sıklık bölmeli çoklayıcı, frequency division multiplexing, fdm)  
zaman bölmeli çoklama (time division multiplexing, tdm)

### **SORU 98: OSI (Open System Interconnection (Açık sistem bağlantı))**

OSI, ISO (International Standard Organisation (Uluslararası standart organizasyonu) ) tarafından 1982 yılında, ağ için bir standart getirmek amacıyla tasarlanmış 7 katmanlı bir ağ yapısıdır.

5 katmanlı TCP/IP yapısıyla yakın benzerliği olan bu yapının katmanları aşağıda listelenmiştir:

Katman 7: Application layer (Uygulama katmanı) : Kısaca bilgisayarda çalışan işlemler (processler) için hazırlanmış olan katmandır.

Katman 6: Presentation layer (Gösterim katmanı): Uygulama katmanından gelen bilgilerin bir alt katman olan oturum katmanına geçmesi sırasında, şifreleme verilerin tutuluş biçimleri (ASCII veya EBCDIC gibi) ve bu biçimler arası dönüşümler bu katmanda yapılır

Katman 5: Session layer (Oturum katmanı) Bu katman bir bilgisayarda bulunan işlemler arasında ayırım yapmaya yarar. Bilgisayara gelen paketlerin hangi işleme geldiği veya bilgisayardan giden paketlerin hangi işleme ait olduğunu tutar. (örneğin hem web hem de eposta paketleri aynı bilgisayara geliyor ama farklı programlar bu paketleri alıyor. işte bu ayırım oturum katmanında yapılır)

Katman 4: Transport layer (Taşıma katmanı) bu katman, ağ üzerinde gidecek veriler için emin bir iletişim ortamı sunar. Gitmeyen paketlerin tekrar gönderimi, gelen paketlerin bozuk olup olmadığının kontrolü bu seviyede yapılır.

Katman 3: Network layer: (Ağ katmanı) bu katman ağ üzerinde dolaşan paketlerin adreslemesi ve bilgisayarlar arasında yönlendirilmesini yapar. TCP/IP protokolündeki IP protokolüne benzetilebilir ve örneğin paketlerin hangi IP adresinden gelip hangi IP adresine gittiği tutulur.

Katman 2: Data Link layer (Veri bağı katmanı) Bu katmanın görevi aynı ağda bulunan bilgisayar veya cihazlar arasında iletişimi sağlamaktır. bir üst katman olan ağ katmanına çok benzer ancak etki alanı sadece direk bağlı cihazlar ile sınırlıdır.

Katman 1: Physical Layer (Fiziksel Katman): Bu katman verinin elektrik sinyallerine dönüştürüldüğü katmandır. Gelen ve giden verilerin sinyal işlemesi yapılır. Bağlı bulunulan ortama göre (örneğin ethernet, uydu, kablosuz bağlantı gibi) paketler işlenerek ağa konulur veya ağdan alınarak anlamlı hale getirilir.

### **SORU 99: İşlemler arası iletişim (Inter process communication (IPC))**

Bir bilgisayarda çalışan birden fazla işlemin (process) bir biri ile haberleşmesini hedefleyen teknolojidir. Hız açısından düşünüldüğünde en hızlı iletişim yöntemi hafıza (RAM) üzerinde veri paylaşımıdır.

Dolayısıyla bir işlemin hafızaya yazdığı bilgi başka bir işlem tarafından okunarak bu iletişim sağlanmış olur.

Günümüz işletim sistemlerinde bu paylaşım işlemi aşağıdaki 4 işlemten birisi ile yapılmaktadır:

mesajlaşma (message passing)

senkronizasyon (synchronization)

paylaşılmış hafıza (shared memory)

uzak prosedür çağırımı (remote procedure calls)

Yukarıdaki bu yöntemlerin hepsinin ortak yanı hafıza temelli iletişim yöntemleri olmalarıdır.

### **SORU 100: FTP (File Transfer Protocol)(Dosya Transferi Protokolü)**

FTP, “File Transfer Protocol” (Dosya Transferi Protokolü) kelimelerinin baş harflerinden oluşan bir kısaltmadır. FTP, internete bağlı iki bilgisayar arasında dosya aktarımı yapmak için geliştirilen bir internet protokolüdür.

Sunucu (server) / İstemci (Client) mimarisi üzerine kuruludur. Yani dosyaları yayınlayan bir sunucu ve bu sunucudan dosyaları almak isteyen bir yada daha çok sayıdaki bilgisayara hizmet verir.

TCP protokolü üzerinde çalışan ftp protokolünün ana amacı dosya almak ve göndermektir. Bu amaç için tanımlı olan FTP komutları aşağıdaki şekildedir:

Jomut

Tanımı

ABOR

Abort data connection process. (ver iletişimini iptal et)

## ACCT

Account for system privileges. (sistem yetkilendirmesi için parametresi ver)

## ALLO

Allocate bytes for file storage on server. (Sunucu üzerinde kadar dosya saklamak için yer ayır)

## APPE

Append file to file of same name on server. ( isimli aynı dosyaya gönderilen dosyayı ilave et)

## CDUP

Change to parent directory on server. (Sunucudaki ana dizine (parametre olarak verilen dizine) dön)

## CWD

Change working directory on server. (Sunucudaki çalışma dizinini değiştir)

## DELE

Delete specified file on server. (Sunucudan isimli dosyayı sil)

## HELP

Return information on specified command. ( isimli komut için yardım göster)

## LIST

List information if name is a file or list files if name is a directory. ( şayet bir dosya ismiyse bu dosya hakkında bilgi ver, şayet bir dizin ise bu dizindeki dosyaları göster.)

## MODE

Transfer mode (S=stream, B=block, C=compressed). (Dosya alma/gönderme şeklini değiştir (S= Akış (aralıksız) B = bloklar şeklinde (parça parça) C = Sıkıştırılmış)

## MKD

Create specified directory on server. (Sunucuda ile belirtilen dizini oluşturur)

## NLST

List contents of specified directory. (Sunucuda bulunan isimli dizinin içeriğini gösterir)

## NOOP

Cause no action other than acknowledgement from server. (Hiç bir işlem yapmaz 😊(aslında kullanım amacı bağlantıyı canlı tutmak ve belirli aralıklarla hiç bir iş yapılmazsa bağlantının kesilmesini engellemektir))

PASS Password for system log-in. (Sunucuya girişte verilen şifre)

## PASV

Request server wait for data connection. (Sunucunun veri almak için beklemesini söyler)

## PORT

IP address and two-byte system port ID. (sunucunun belirtine porta zıplamasını sağlar)

## PWD

Display current working directory. (Şu anda sunucu üzerinde çalışılan dizini gösterir)

## QUIT

Log off from the FTP server. (Sunucudan çıkmak için kullanılır)

## REIN

Reinitialize connection to log-in status. (Bağlantıyı tazelemek amacıyla şifre giriş ekranına kadar bağlantıyı koparıp tekrar bağlanır)

## REST

Restart file transfer from given offset. (dosya transferini belirtilen değerinden yeniden başlatır (bütün sunucular desteklememektedir ama büyük dosya transferlerinde bağlantının kopması durumunda kalan yerden devam etmek için oldukça faydalıdır))

## RETR

Retrieve (copy) file from server. (sunucudan belirtilen dosyayı almaya yarar)

## RMD

Remove specified directory on server. (Bir dizini komple silmek için kullanılır)

RNFR Rename from old path. (eski dizini yeniden adlandırmak için kullanılır)

## RNTO

Rename to new path. (yeni dizine verilecek adı belirler)

## SITE

Site specific parameters provided by server. (Sunucuya özel bu komut listesi dışında bulunan komutları gösterir)

SMNT Mount the specified file structure. (bir dosya yapısını (cdrom floppy disket gibi) sisteme bağlamak için kullanılır)

## STAT

Return information on current process or directory. (sistemin istatistik değerlerini gösterir (ftp bir processtir ve bu process hakkında veya içinde bulunulan dizin hakkında bilgiler))

### *STOR*

*Store (copy) file to server. (Belirtilen dosyayı sunucuya atmak için kullanılır)*

### *STOU*

*Store file to server name. (belirtilen sunucuya dosyayı kopyalamak için kullanılır)*

### *STRU*

*Data structure (F=file, R=record, P=page). (veri transferleri sırasında kullanılacak olan veri yapısını belirlemeye yarar (F:Dosya R: Kayıt P: Sayfa bkz. veri yapıları)*

### *SYST*

*Return operating system used by server. (Sunucunun kullandığı işletim sisteminin bilgilerini gösterir)*

### *TYPE*

*Data type (A=ASCII, E=EBCDIC, I=binary). (transfer sırasında kullanılan veri yapısını değiştirmeye yarar (A: [Ascii dosyalar düz metin ve yazılardan oluşan dosyalar](#), E: EBCDIC , ascii benzeri farklı bir karakter tablosudur, I: Binary, resim, müzik gibi ikili dosyalar için kullanılır)*

### *USER*

*User name for system log-in. (sisteme girişte kullanılan kullanıcı adını tanımlar)*