

Bilgisayar Ağları

Slayt 6

Protokol: Herhangi bir bilginin ağdan nasıl alınabileceğini anlatan komutlar dizisi

Public: Internet

RFC: Kurallar dizisi. Tüm protokoller bu kurallara uyacağını beyan eder.

IETF: Kuralları koyan kuruluş

PROTOKOL

Human Protokol:

Bir kişi ile iletişime geçmek için belirli bir sıra vardır.

Network Protokol:

Network'ün Sınırları

Network'ün sınırları = host and application

Network'ün aklındaki = birbirleri ile bağlantılı olan router'lar var (yönlendiriciler)

end systems (hosts): ağı kullanan programları çalıştıran cihazlar.

client / server modeli: biz istiyoruz (istenciyiz)

bize o sayfayı sağlayan server

peer to peer modeli: aynı düzeydeki istenciler arasında dosya paylaşımı sağlayan server. (Skype, BitTorrent)



Bir yere bağlantı yaptığım zaman sadece ben mi bağlanıyorum?

2 tip bağlantı vardır:

- 1) Devre bağlantımlı bağlantılar: Tüm hattı reserve edersiniz.
- 2) Paket bağlantımlı yapı: Ağ paketlere bölünür, paketler ağda kendi sıralarına girer. (İnternette paket bağlantımlı yapı)

KULLANDIĞIMIZ ALT YAPILAR

1) Dial-up Modem: (Çevirmeli)

- Telefon hattını kullanır
 - Veri indirme hızı çok düşük
 - Aynı anda telefonu kullanamayız
- } Dezavantajları

2) DSL :

- Hem telefon hem de internet kullanılabilir.

3) Kablolu Modem:

- Hybrid, fiber ve coax korisimli kablodur.
- Asimetrik tip
- DSL'de sizin için ayrılmış bir hat var fakat bu modemde paylaşım var.

FDMA: Frekans Division Multiplexing

Herkes farklı bir frekans bandı bölünür ve herkes kendi bandını kullanır.

4) Fiber to the Home: (FTTH)

ONT: modemlerin içindeki yapı

OLT: ıskı bilgisini alıp, ağı elektrikscl bilgi gönderir.

- Download ve upload kapasiteleri ortmıstır
- PON 'da OLT'den gelen bilgiler splitter ile çoğaltılıp gönderilir
- PAN'lorda splitter yok, direkt bağlantı var

5) Ethernet İnternet Bağlantısı:

- Twisted pair copper wire kullanılır.

6) Kablosuz Bağlantı Ayarları:

- Baz istasyonları mevcuttur. Yo da değişik noktalara bağlantı noktası konulmuştur.

- * WiFi = kapsama alanı küçük olan bir teknoloji
- * EVDO ve HSDPA = kapsama alanı biraz daha geniş
- * WiMAX = kapsama alanı kilometrelerce

= Home Networks =

- DSL yada kablolu modem
- router
- - -

Fiziksel Medya

fiziksel link: kullanıcılar arası link

guided media: nereye gideceğini biliniyor

unguided media: nereye gideceği bilinmeyen sinyaller (radyo dalgası)

coaxial kablo: iki yönlü

baseband: kablo üzerinde bir kanal

broadband: kablo üzerinde çok kanal

uydu:

- gecikme, zaman kaybı olacaktır.

Network (Ağ) Gekirdeği

- Router: İşlevi olan elemanlar. Üzerindeki algoritmalarla en kısa bilgi alışverişini gerçekleştirir.

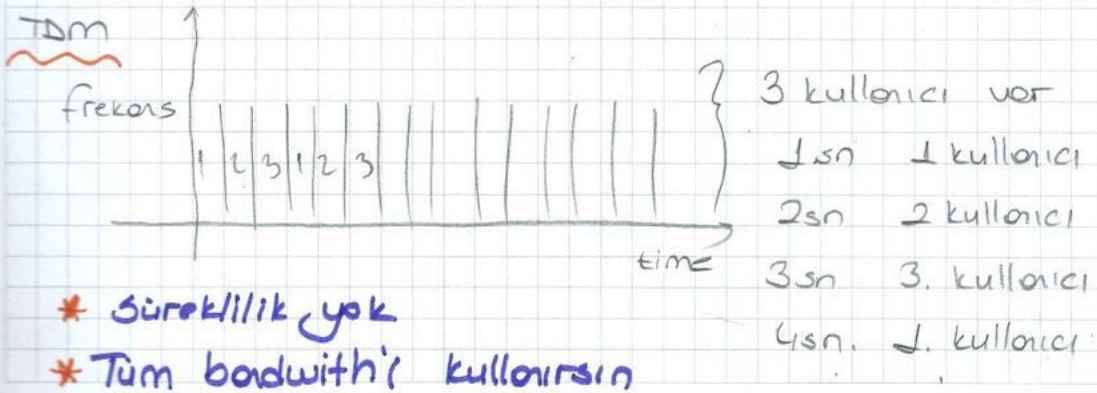
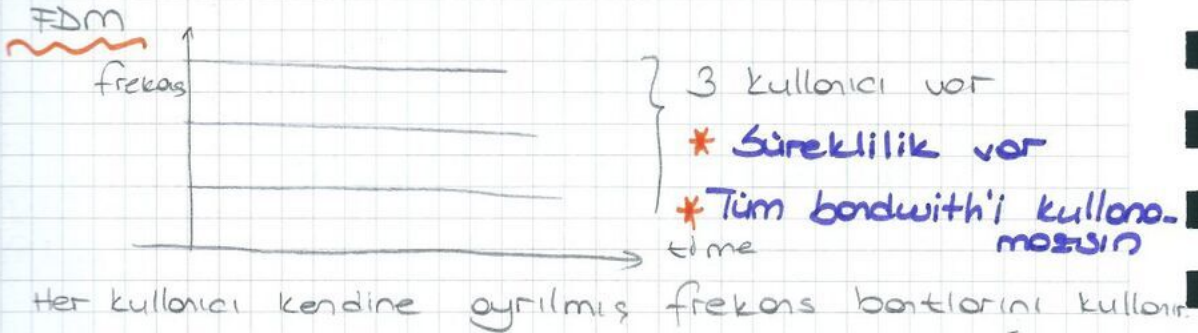
Router'lar arası bilgi alışverişi nasıl sağlanır?

- 1) Daire bağlantılarına
- 2) Paket bağlantılarına

Devre ihtiyaclarlomo:

Öncelikle iki end sistem arasında bilgi alışverişi için o hatın tamamen onlara verilmesi söz konusudur. O hat ya zaman ya da frekans olarak bölün-
lendirilerek diğer bilgi alışverişi olacak ekipmana
da ayrılır. (Restoranda rezervasyon ile oturma
sekli. Rezervasyonu olmayan bekler) Herkes kendine
ayrılmış bandwidth'i kullanır. Bandwidth'i olmayan
bekler.

- Bağlantı kuracağın garantidir. (Call setup ile öncelikle bağlantı kurulur)
- Total bandwidth'in bölünmüş parçaları kullanılır.



örneğe
Örn: 640 000 bits A hostundan B hostuna
doğrudan anahtarlanmalı^{ya} ile göndermek istiyoruz.

- A linki 1,536 Mbps (1 gün)
- Her link 24 slot'a bölünmüş
- Call setup işlemleri 500msec sürüyor.

$$\frac{1,536}{24} = 64,000$$

$$\frac{640.000}{64.000} = 10$$

$$10 + 0,5 = 10,5 \text{ sn}$$

Paket Anahtarlama:

Bir restoran var fakat rezervasyon usulü ile çalışmıyor. Yeryükseğe masanın boşalmasını bekliyorlar. Dolayısıyla elimizdeki bantwidth'i en verimli şekilde kullanıyoruz.

Uygulama Katmanı Protokolleri

- Mesajı tanımlar
- Mesajın syntaxını tanımlar
- RF-C kurallarını tanımlar

Gönderimi Sağlayan Servisler Hangi Uygulamaları Tercih Eder?

- Bazı uygulamalarda oradaki sesler vs. gibi detaylar kayıp olsa da göz ardı edilebilir.
- Fakat word vb. datalarda kayıp göz ardı edilemez.

* Bazı uygulamalarda zaman önemlidir ve bu uygulamalarda gecikmenin belirli süreyi aşmaması gerekir.

* **throughput**: veri aktarımı

Örneğin; mail gönderilecekse ne timing ne de throughput önemlidir. fakat data kaybı göz ardı edilemez.

İnternetteki Veri Gönderim Protokolleri

1) TCP

2) UDP

TCP:

- connection-oriented: karşılıklı olarak bağlantıya ihtiyacımız var. (UDP için ihtiyacımız yok)

- reliable transport: güvenli veri iletişimi, gönderme istediğinizi herşeyi gönderir. Yolda karşılaşılabilecek olan

veri bozulmalarından TCP sorumlu değildir.

- flow control: Asla alıcının alabileceğinden fazlasını göndermez. Gönderim hızını buna göre ayarlar.

- congestion control: Ağda yoğunluk varsa ^{veri} kontrolünü yapar. (Bunu flowunu azaltarak yapar)

- does not provide: timing ve minimum throughput garantisi vermez. İçeriğin değiştirilemeyeceğine garanti vermez.

UDP:

- UDP için öncelikli bağlantıya gerek yoktur.

- güvenlik yok

- timing yok

- sadece hız var

- pahalı olmayışı cezbedicidir.

deniz çok real time uygulamalarda kullanılır.

Canlı TV bağlantısı → UDP

Mail alışverişi → TCP

UDP ve TCP → transport layer'deki iki protokoldür.



Web and HTTP

* HTTP'yi kullandığımda ne yapıyorum?

- client / server modeli

client: bizleriz

server: objelerin tutulduğu sunucular

- TCP'yi kullanır

- server buna cevap verir.

- HTTP "stateless"tir. Geçmişle alakalı hiç bir durumu tutmaz.

HTTP Bağlantıları



Kollu HTTP

- Bağlantı bir kez

kurulur tüm objeler

indikten sonra kapatılır.



Kollu Olmayan HTTP

- Web sayfasındaki her bir

objeyi indirmek için tekrar

girip tekrar kapatmak gerekir.

Nonpersistent HTTP: (Kollu Olmayan)

1) TCP coll setup başlar

2) Kabul edilir

3) İstegimiz neyse gönderilir.

4) Geri cevap bize ulaştırılır ve kapatılır.

5) Diğer istek için bir daha girilip kapatılır.

RTT: (round trip time) Bir paketin istenciden servera serverdan tekrardan istenciye dönmesidir.

Response time: Tüm RTT ve dosyanın gelmesi için geçen süre.

Persestant HTTP

- 1 kez call setup yapılır.

HTTP request message

→ request line

header lines

- Host
- User-agent
- Connection
- Accept-encoding

→ bitiş mesajı

HTTP /1.0

- Get
- Post
- Head

HTTP /1.1

- Get, Post, Head
- Put

Post, entity kısmına girerek anahtar kelimeyi
oracık

HTTP response status codes

200 OK

301 moved Permanently

1
1
1

Bunları bil

COOKIES

Server'in sizi tanımlayan ID atanması

Örnek; Ayşe her zaman internete kendi Pc'sinden bağlanır. Alışveriş sitesine ilk girdiğinde Ayşe o ID bilgisi ile tanınır.

Cookies Neler Sağlar, Nereelerde Kullanılır?

- Girmiş olduğunuz sitenin data base'ini ulasılmasını sağlar
- Cookies sizi takip etmek durumundadır.

Cashes (Proxy Server)

Daha önceden girdiğiniz web sayfasına birdaha girmek istediğinizde sayfayı cache'den sağlar.

Böylelikle

- oğın trafğini rahatlatır
- sayfaya ulaşacağımız süre azalır.

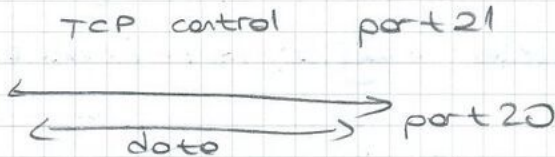
Cashe'in tek sorunu;

→ Bozen cashe'de eski bilgiler olabilir.

Conditional GET; coshe'in icindeki bilgilerin modifiye olup, olmadigini tespit eden komut.

FTP (the file transfer protocol)

- FTP'yi kullanabilmek icin FTP programina ihtiyac var
- FTP dosya transferi icin kullanilir.
- Outoff band: FTP'ye serverla baglanti kurmak istedigimi soyluyorum (kontrol icin), baglanti kuruluyor.



Data kullanan connection bordinin disinda bir baglanti ile gonderiliyor. Bu nedenle outoff bandedir.

FTP komutlarina bak!

* * * Electronic Mail:

3 tane major elemanı var

- Kullanıcı arayüzü (user agent) (maili almak vermek oluşturmak vb...)
- Mail servers
- SMTP (simple mail transfer protocol (mail oluşturma ve taşıma))

mail serverlar arası iletişimi sağlar.

Mail Servers: Kişilerin gönderdiği maileri tutar

Senaryo

- 1) Alice Bob'a mesajı göndericek. Mesajı oluşturdu.
- 2) UA Alice'in mailini mail server'a gönderiyor.
- 3) Client olan o an mail server (işlenide olan)
- 4) Bob'un mail server'ı ile contact kuruluyor TCP aracılığıyla
- 5) Bob kendi user agent'ından maili alıyor ve okuyor.

SMTP ile ilgili

- * persistent bağlantı kullanılır.
- * ASCII karakteri

Bosluk satırı bilgilerine bakıyor.

- http pull, SMTP push
- ASCII komutları ve cevaplarını kullanır her ikisi de
- http'de her obje kendi içinde, SMTP'de tüm objeler tek bir yerde toplanır.

http ve SMTP'nin ilişkisi filmi

Mail access protocols

→ Maili SMTP ile gönderdim fakat maili mailbox'tan almak için ne yapacağız?

- 1) POP:
 - 2) IMAP:
 - 3) HTTP:
- } mailbox'tan maileri almak için kullanılan protokoller

* → HTTP ile kendi user agent'imizden server'a
* mail gönderilebilir fakat sender's mail server ve receiver's mail server arasında mutlaka SMTP kullanılır.

= POP3 protokolleri =

✓ client commands

*user: declare username

*pass: password

✓ server response

*OK

*-ERR



POP 3

- İndirdikten sonra sil "download and delete"
- Oturumlar prosesi durum bilgisi tutulmaz
- Eğer mesajları serverde tutmak istiyorsak özel bir komut kullanmamız lazım

IMAP

- Tüm mesajlar serverde tutulur
- tüm durum bilgileri tutulur.

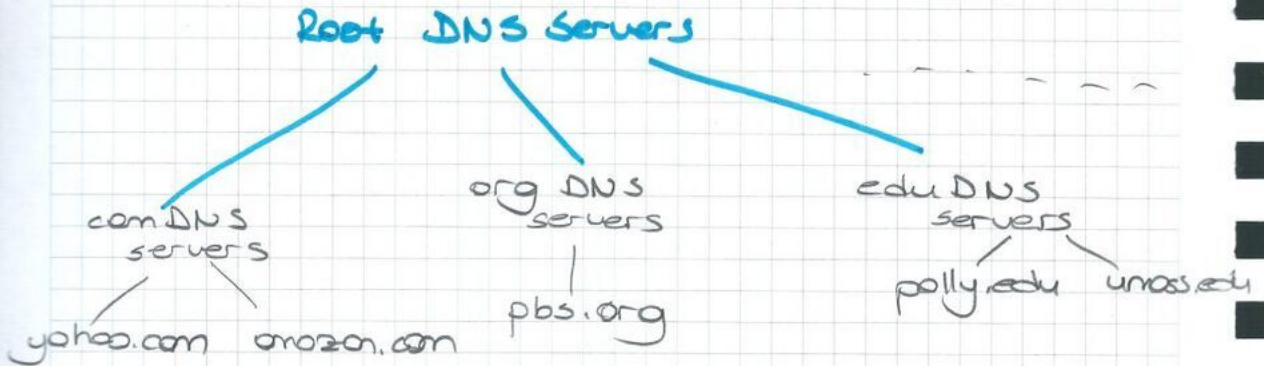
DNS: Domain Name System

baskent.edu.tr 'nin arkasında 10. 395 gibi bir numara var fakat bu numara okula tutulması zor olduğundan bu ismi kullanırız. Bir yerde bu ismin bu numaraya eşdeğer olduğunu bildiği bir yer olmalı Bu ismi yapan yapıya DNS diyoruz. IP adres 32 bit. Her isme karşılık 32 bitlik bir IP adres var. DNS bir application layer protokolünü gerçekleştirir.

DNS servisleri

- host isimleri IP adresine çevirirler
- bazı hostlar için birden fazla isimleri olabilir ve bu isimler için tek bir IP adresi olabilir. DNS servisleri bu olayı aszomler

- birden fazla DNS 'server'ı var çünkü, yığılmayı azaltmak için.



istenecek IP adresini istiyor ^{www.}amazon.com'un

- root'a diyor ki .com uzantılı serverların IP adresini ver
- com uzantılı servera amazon.com'un IP'adresini ver diye sorgular.
- amazon.com dan ^{www.}amazon.com'un IP adresini öğrenmiş olur.

DNS none resolution example

iterated yapı:

- kendi local server'imize sorulacak. local serverimiz root'a sordu. Root yetkili olan DNS serverının adresini verdi (top level DNS serverı), top level DNS servera yetkilendirilmiş DNS serverına gitti. Yetkili adresi verdi.

recursive yapı: Bu yapı kullanılan bir yapı değil. Her seferinde root yapıyor. sorgulamayı bu da pek mümkün değil.

DNS: caching and updating records

DNS kayıtları nasıl tutuluyor?

RR formatı (name, value, type, ttl)

↳ time to live
kaydın yaşama
süresi

Buradaki type'lara bak

DNS protokolü Nasıl Çalışır?

16 bit ile istekmi cevap mı bilgisi verir.

4 bitlik bayraklar var

P2P (peer to peer) : hostların kendi aralarında iletişimini sağlayan yapılardır

Dosya PAYLAŞIMI:

u_1 : upload hızı

d_1 : download hızı

} Büyüklüğü f olan bir dosyayı
N client'a göndermek için

N kadar kişi için server N kere upload
edilecektir.

(Server client
yapısı)

$$dcs = \max \{ NF/u_s, F/\min(d_i) \}$$

Peer to peer yapısında;

server dosyayı birine gönderdikten sonra, peerler kendi kütüphanelerinde eklerini yapılıyor.

$$d_{202} = \max \{ F/u_s, F/\min(d_i), NF(u_s + \sum u_i) \}$$

Bu nedenle P2P ile client-server'i karşılaştırırsak kullanıcı sayısı arttığında P2P daha az sürede halleder.

File Distribution: Bit torrent

Dosyaları parçalara bölerek alıverme yapılıyor.

tracker: torrentlere üye olan peerlerin listesi burada bulunur.

BIT TORRENT

- Bir peer bir torrente üye olursa parçaları peerde toplanmışsa o peer de paylaşabilir (dosyayı)
- "Ne kadar eklemek o kadar kofte" karşılıklı olarak dosya paylaşımı ne kadar çok olursa sistemin önceliğiniz de o peer olur.



* Bu yapıya üye olanlar değişik zamanlarda üye olabilirler

* Daha çok dosya paylaşımı yapar peer'la sende daha çok dosya alıp verebilirsin.

→ Tit for tat

1) Biri ile iletişime başladıysanız onun topluluğine girmeye bir adaysınız.

→ Herhangi bir sort yok.

→ Ne kadar çok dosya paylaşımı yaparsanız okadar hızlanırsınız.

DHT

- Anahtar ; sosyal sigortalar numarası vb...

- değer ; IP numarası

peerlar bu database'in içerisine sorgu gerçekleştirebilirler bunu da anahtarla yaparlar.

Her peer değer ve anahtar girerek kayıt yapabilirler.

Anahtar ve değer database'de match edilirse paylaşımına ulaşılabilir.

h'lona ? $key = h("Led Zeppelin - IV")$

bu dosyayı h'ladığımızda doğru key'i vermesi lazım eğer doğru key ise.

① (Doğru dosya mı ?) öğrenmek için kullanılır.

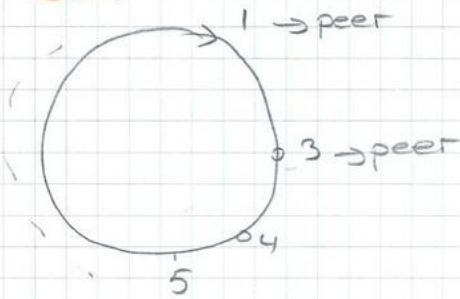
② torrent yapısına sadece 1 dosya indirip almak için kullanıcılarda var buna engel olmak için de yapılır. (SİFRELEME için)

? Dht neden kullanılır?

① Dosya doğru dosya mı bulmak için

② 1 dosya indirmeyi engellenek için (sifreleme)

Circular DHT



Biz peerlere ulaşmak istediğimizde mesela 1'e sorarız tanımiyorsa 3'e - - - gider.

Doğru kişiyi bulmak zorun gerektirir.

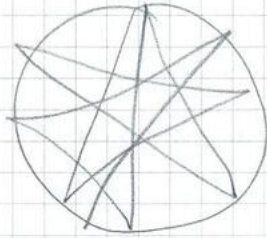
Her peer kendinden önceki ve sonrakini bilir. Peerlere değer atanırken buna dikkat edilir.

Ex: $n=4$ peers: 1, 3, 4, 5, 8, 10, 12, 14

13. key için peer = 14

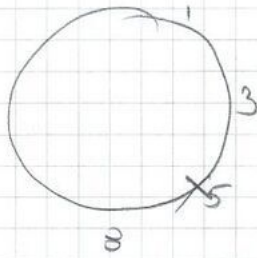
key 15 için peer = 1

N sayısı yüksekse döngü sonsuza gider
bunun yerine shortcut sistemi kullanılır.



→ Shortcut

Örneğin 5 çıktı



Birbirlerinin var olup
olmadığını sorarak
algırlarlar. Bu durumda
3' 8'e yönelir.

Bir peer girmek isterse ona en yakın
peerler aralarında bağlantı kurar.

= Soket Programlama =

Bu yazılımlar aracılığı ile app'leri doğru
uygulamalara yönlendirir.

Gelen bilgiyi doğru katmana yönlendiren
yapılar. 2 tip var.

1) TCP için

2) UDP için

Hem server hem de client tarafından bizim
soketlerimizin tanınması gerekir.

Soketler bir port numarası ile tanımlanırlar.

→ UDP ve TCP arasında;

- biri handshaking yapar biri yapmaz.

Bunedenle TCP'de 2 adet soket programlanmalıdır.

UDP

→ Her client hemde server datagram soketini kullanırlar. (UDP'de)

→ Birden çok client aynı anda aynı servere kullanabilir mi?

UDP buna müsaade eder yani evet.

→ Client ^{serverin} IP adresini ^{ve port numarasını} bilmeden server'a birsey gollayabilir mi?

Gollayamaz.

TCP (Transport layer katmanında bir protokol App'de değil)

- IP adreslerinin ve port numaralarının tanımlanmış olması gerekiyor.

- Birden çok client aynı anda server'ı kullanabilir.

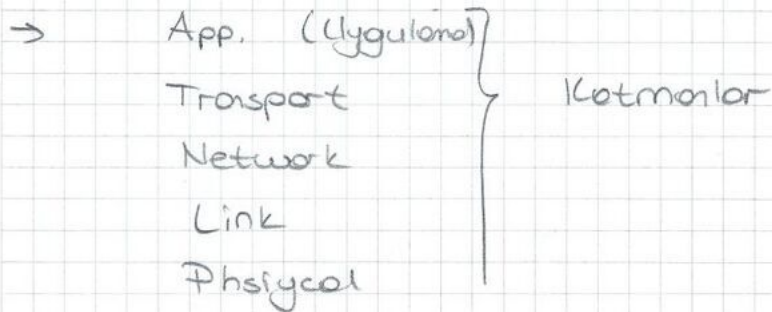
Server Soket: } Biri handshaking için kullanılır.
Soket:

= CHAPTER 3 Transport Layer =

(Tasima Katmanı)

→ UDP ve TCP bu katmandadır.

→ iki ^{end} nokte sistem arasında verilerin alisverisini yapar (transport layer)



→ processler arasında sayısal iletişimden sorumludur (transport layer)

→ TCP güvenilir bir taşıma protokolüdür, yani aldığı mailin tamamını karşı tarafa gönderir fakat içeriğine karışmaz.

TCP

- congestion control
- flow control
- connection setup

→ *gecikme
* bandwidth garantisi

} TCP ve UDP bunların garantisini vermez.

→ UDP ile daha hızlı gönderim yapılır.

Multiplexing / Demultiplexing

Gönderici
kismi

Alıcı
kismi

Değişik soketlerden gelen datayı toplayıp ~~birlikte~~ ile göndermek (multiplexing), bunun tam tersi (demultiplexing)

Demultiplexing:

Host IP datagramları olur.

1) - - -

2) Her datagram bir segment taşıır.

3) - - - - -

UDP için demultiplexing:

dest IP adresi, dest port numarası var UDP için

✓ Eger bir alıcı UDP segmenti olmuş ise dest port numarasını kontrol eder.

TCP ,

source IP, source port, dest IP adresi, dest port number

✓ Server birden fazla TCP soketini kullanabilirler

UDP (User Datagram Protocol)

- * UDP ile gönderdiğiniz her segment kaybolma riskini içerir.
- * TCP'den daha hızlıdır. (Real time ses, görüntü için hız çok önemlidir) Bir videodaki herhangi bir karenin kaybolması insan gözünde fark edilmez bu nedenle hız bu işlemler için daha önemlidir.
- * Hızlı olmasının sebebi TCP'nin yaptığı birçok sorgulamayı yapmaz.
- * App katmanında yazılımlar aracılığı ile hem UDP'yi kullanıp hemde TCP'nin özellikleri ile UDP'ye optimizebiliriz.

Neden UDP hızlıdır?

- sıkışıklık kontrolü yapmaz
- handshaking yapmaz.
- yapı olarak daha basittir.

UDP'yi kullanan yapılar;

- DNS
- SNMP

UDP checksum;

Gönderilmiş verideki hataları bulur.



sum 1 1 0 0
checksum
bunun tersidir
yani 0'lar 1 1'ler 0

Alıcı kısmı sum ve checksum değerlerini toplar hepsinin 1 olması gerekir.

Gönderici kısmın checksum'ı ile kendi topladığı sum'ı toplar ve hepsinin 1 olmasını ister.

→ Eğer 0 varsa overide bir hata vardır.

Güvenilir Data Transferi Nasıl Olur?

(Principles of Reliable data trans)

RDT → reliable data transfer

→ Güvenilir olabilmesi bizim için çok önemli

App. layer'dan transport layer'a incek. Aradaki hattın güvenli olması gerekir. Internet açısından düşünürsek oğ güvenilir değildir ve kontrol yapılması gerekir.

udt → uni directional transfer (tek yönlü transfer)

* App layer'dan transport layer'a gelen data
rdt_send

* Transport layer'dan app layer'a gönderilen
data deliver_data()

RDT 1.0: güvenli bir layer üzerinden güvenilir bir data transferi

Gönderici kısmı;

App.
Katmanı



rdt_send

packet = make_pkt(data)
udt_send

Alıcı kısmı

Link
Katmanı



rdt_rcv

extract(packet, data)
deliver_data

RDT 2.0:

Kanal güvenilir değilse;

- Checksum yapmam lazım. (TCP'de checksumdan sonra hata mesajını bildirir fakat UDP yapmaz)
- Doğru gönderilip gönderilmediğini nasıl anlarım?
2 adet bildirim ihtiyacım var
 - * ACK → düzgün almışsa veriyi (alıcı) bu mesaj
 - * NAK → düzgün veriyi almadıysa (alıcı) bu mesaj

Alıcı NAK gönderirse veriyi tekrar göndericek.

! Eger gönderdiğiniz ACK ve NAK'larda sorun varsa

Siz bir ACK yada NAK gönderdiğiniz ama gitmedi yada bir paket gitti ve siz onun için ACK gönderdiğiniz, karşı taraf ACK'yi almadan 2. paketi gönderdi yada ACK yada NAK bozuldu

- O zaman bu paketlere bir sıra numarası koymalıyız.

- Sadece 0 ve 1 seq. numaraları yeterli mi?

Gönderip bekleyecekse yeterli

Sürekli gönderim için yeterli değil.

→ NAK'ı hic göndermeden de karşı tarafa dosyanın hatalı geldiği bilgisini karşı tarafa bildirebilir miyiz? Yani sadece ACK'larla bu işleni yapabilir miyiz?

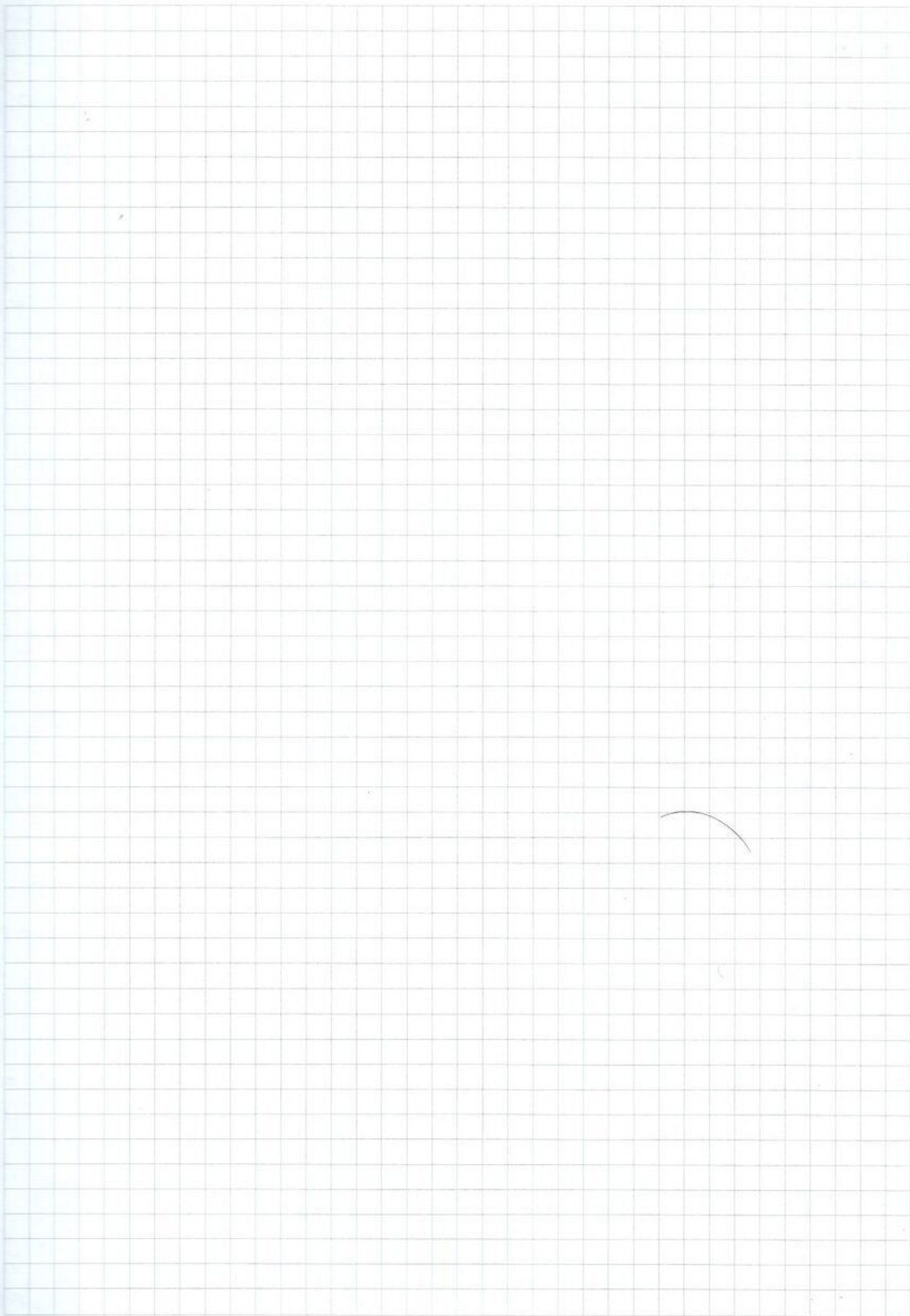
Eğer paket doğru geldi ACK gönderdik, 2. paket yanlış geldi NAK göndermek yerine bir kez daha ACK gönderirsek aynı paket için 2 kez ACK gelmişse o paket NAK olarak algılanır. Yani bir önceki için ACK gönderilmelidir.

RST 3.0:

Kanal gönderdiğimiz paketi kaybetti varsayalım karşı taraf ne yapmalı. Örneğin 0. paket gönderildi 1. paketi bekliyorum paketin gelmediğini ne zaman anlarım? Paketlerin kaybolması ya da gelmediği durumlarda belli bir zaman bekleme gerekir fakat ne kadar beklemek gerekir? Timer eklememiz gerekir ve o timer süresince ACK beklenir.

Timer o paketin gönderilmesinden sonra ACK bekleyecek, ACK gelmezse veri kaybolmuştur denir. Timer sender'dadır.

Timer'ın süresi RTT'ye göre ayarlanır.



Hello Kitty

© 1991, 2011 SANRIO CO., LTD.

= ATM ABR congestion control =

ATM → deure onotterlomoli bir yopidir. (Asenkron transfer modu)

ABR: available bit rate

Hotto bir sikisiklik yokse gonderici tum bandwidth'i kullanarak gonderir.

Hotto sikisiklik varsa gonderici rate'i kiser.

→ 32 data yolluyoruz 1 tane RM hücresi yolluyoruz.
(1/32)

bitler (RM'nin icinde)

NI ve CI biti var.

NI: Rate'inde bir arttirim yapma der (set edilmis)
Belli bir sikisiklik var ama sorun degil

CI: Hotto ciddi sikisiklik var rate'i dusur.

Ayrice RM hücresinin icinde 2 byte'lik ER
olani var. Eger switchde tikaniklik varsa switch
ER'yi azaltiyor.

Datolarin icinde ise EFCT 1 bitlik olan
vardir. Switch sikisiksa bu olan setlenir.

Sıkışıklık olduğunu anladık ne yapılacaktır?

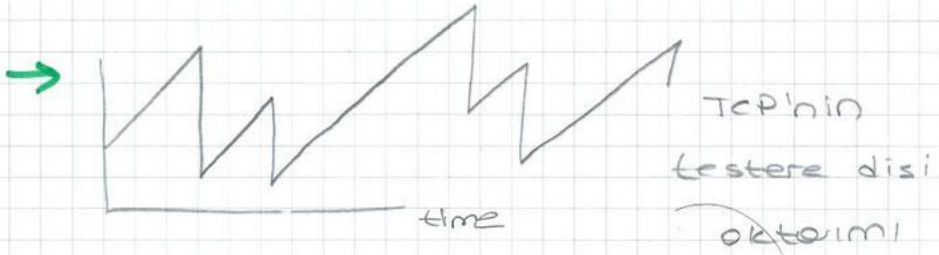
TCP Congestion Control

- Her TCP sender'ı kendi rate'ini kendi ayarlar.
- ACK ve LOST segment

ACK: eğer bir datayı gönderdik ve ACK geliyorsa hatta bir sorun yoktur, Rate'i ↑

LOST SEGMENT: Yo ACK dönmez yo da timeout olduysa RATE'i azalt.

* ACK'yi aldığı sürece Rate'i arttırmaya devam eder toki lost'u gördüğü sürece kadar



* Sabit Rate yok.

* Ne kadar arttırıp ne kadar azaltmalı ?

Henüz ACK'sı alınmayan segment ile ACK gönderilen segment arası.

cwnd: gönderim windowunun içi.

$$\text{rate} = \frac{\text{cwnd}}{\text{RTT}} \text{ byte/second}$$

rwnd: akış kontrolü

cwnd, rwnd'den hangisi en küçükse ona göre ayarlar.

→ segment kayıp olduğunda ne yapacağız. (time-out)

* cwnd'yi 1'e set ederiz !
↳ 1ms

→ 3'tane duplicate ACK ile karşılaşırsak

* cwnd'yi yarisına indiririz !

→ Normal ACK aldysa

Rate'i arttırmak lazım, 2 mod var

* Slowstart phase → hızlı arttırım fazı

+ Congestion avoidance → sıkışıklıktan kaçınma

→ Bağlantı

→ Connection başladığında başlar. Gönderim hızı üstel olarak artacak. Time out sonrasında süreçte başlar.

→ Artık lineerdir. Bir threshold değeri var

Eğer cwnd değeri $\geq$ threshold olursa slowstart'tan congestion avoidance moduna geçer.

Slow Start:

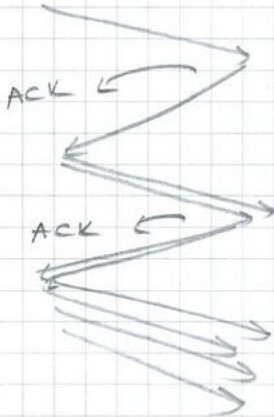
cwnd = 1 mss

MSS = 500

RTT = 200

} initial rate
20 kbps

Dönen her bir ACK için gönderim sayısını
bir öncekinin 2 katı yapar.



* Bağlantının başında
veya cwnd'yi 1 mss'ye
kadar indirmişsek
bu bağlantı olur.

Congestion Avoidance:

cwnd lineer artar. Her RTT için cwnd'yi
1 mss artırır.

AIMD: toplamsal artış, aritmetik azalış.

Eğer paket kaybı olursa cwnd'yi yarıya indirir.

Eğer time out oluyorsa;

cwnd 1'e indirilir ve slowstarttan başlar.

$$\text{threshold} = \frac{\text{cwnd}}{2}$$

→ TCP Reno timeout durumunda 1'e indirmez daha yüksek bir değerde bırakır.

* cwnd < threshold slowstart

cwnd: gönderici kısmındaki gönderim windowumuzu.

TCP Fairness

Adil olma. TCP adil çalışan bir protokoldür. Hostları paylaştırırken.

2 TCP varsa bandwidth 2'ye bölünür. $\frac{B}{2}$

Congestion Controlün bize sağladığı yarar;

Bu adil durumun devami sağlanmasıdır. Cong. cont. yapılmıyorsa adil durum olmazdı.

UDP'de durum;

- Real time işlemlerde UDP kullanılır. UDP sabit bir rate'te gönderim yapar. Rate'ı azaltıp, artırma. Çünkü olabilecek kayıpları önemsemez.
- 2 host arasında paralel bağlantı yapılıyorsa adil dağılım söz konusu.

R/TCP connection

Bandwidth uygulamaya sayısına bölünmez. Uygulamaya için gerektiği TCP sayısına bölünür.

Hello Kitty 



Hello Kitty

© 1999 SANRIO CO., LTD.

Distance Vector

Linkin maliyeti değişirse, update edilir ve distance hesaplamaları yeniden yapılır.

Maliyet = linkteki trafik, kullanıcı sayısı

→ İyi haber çokuk yayılır (maliyet 4'ten 1'e indi)

→ Kötü " yavaş yayılır (maliyet 4'ten 60'la çıktı)

Bu durumda y aye

maliyetin sonsuz olması = bağlantının kopması

LS

* nE kodlar mesaj gönderilir

* n^2 kodlar algoritma gerekir

* Dayanıklılık;

Routerda bir sorun çıkarsa, yanlış hesap tüm herkeze bildirilecektir.

DV

* Router sadece kendi komşularına bilgi verir.

* kendi komşu sayısına kodlar

* Dayanıklı;

* Node to node varsa ^{herkes} kendisi komşularına gönderecektir.

= Hierarchical Routing =

- Bir router'in kendi forwarding tablosunda tüm routerlara ait bilgi tutması mümkün değildir.
- Belirli bölgedeki routerları bir araya topluyoruz ve bunlara AS diyoruz.
- Bosko AS'lerin içindeki routerlarda vardır.
- Bir AS ve bosko bir AS haberleşecekse, bir routeri gateway routeri olarak belirliyoruz.
- AS'ler arası veri alışverişi inter AS protokolleri belirler.

INTER AS

- AS'ler arası demektir.
- Gateway routerlar AS'ler arası routerlar ile iletişimi sağlar.

INTRA AS Routing (AS içi)

- RIP: Routing Information Protocol
- OSPF: Open Shortest Path First
- IGRP: Interior Gateway Routing Protocol.

RIP:

- distance vector algoritmasını kullanır.
- distance metric: # of hops (max = 15 hops)
- distance vectorleri bir response mesajı yollarak kendi distance tablosunu düzenler.
- UDP paketlerini kullanırlar.

OSPF:

- Link state'i kullanır.
- Dijkstra algoritmasını kullanır rotolama için.
- OSPF kimlik doğrulama mesajları içerir.
- Bir hedefe olan birden fazla yolu birden fazla entry olarak kullanılabılır.
- TOS içeriklerine göre maliyetleri değiştirir.
- OSPF daha özellikli bir algoritmadır.

2 seviyeli hiyerarsi vardır.

area border routers:

backbone router:

boundary router: Başka AS'ye ulaşmak için kullanılır.

De facto: En çok kullanılan, en çok bilinen

eBGP → AS'ler arası }
iBGP → AS'ler içi }

Path orb... BGP

- * Sadece IP'nin prefixini göndermeyiz. O ağa giderken bosko hangi AS'ler üzerinden geçeceğiz bunun bilgisini verir. (AS-PATH ve NEXT-HOP)
- * O AS'nin içindeki routerin numarasını da gönderir.
- * Farklı AS'ler arasındaki protokolü kontrol eden protokoldür.

→ Router 1'den fazla yolu olduğunu öğrenebilir.
Fakat 1'ini seçmek zorunda bunu seçerken;

Sırayla
gibi bilir.

- 1) local preference value
- 2) En kısa AS-PATH
- 3) En yakın NEXT-HOP
- 4) - - - - -

= BGP Messages =

OPEN, NOTIFICATION - - - -

AS'ler içi ve AS'ler arası protokoller neden farklı?

✓ AS'ler arası protokol ağıdaki trafiği bilmek ister, AS'ler içi lokaldır ve böyle bir durum içermez.

✓ Ölçekleme

✓ Performans * AS içi önemli olan performans

* AS'ler arası " " kuralları

→ Bilginin her yere herkese bildirilmesi

Broadcast Routing

in network duplicationunda

her router kendi komşusuna gönderir.

- Bir rout kendisine komşusundan gelen her paketi kontrol edecek (daha önce gönderilmiş mi diye)
- Bu sorun için broadcast numarasının saklı olması gerekir ve bunu tutar.

Spanning Tree

- ✓ Önce ağac oluşturulur.
- ✓ Ağacın dalları arasında paket ilerler.
- ✓ Koyu yollar ağacın dallarıdır.
- ✓ Genel yayın mesajı kime geldiyse oradan gönderim başlar (A'dan ise $A \rightarrow B \rightarrow D \rightarrow G$)
 $\downarrow \quad \searrow$
 $F \quad E$
- ✓ Spanning Tree nasıl oluşturulur?
 - Merkez noktası seçilir.
 - Her node bir mesaj gönderir.

→ Shared Tree: steiner tree karmaşık bir yapıdır (NP) Uygulama alanında karmaşıktır ve gerçek hayatta kullanılmaz. Bir tane router bile girilse tüm yapının bozulan kurulması beklenir.

Bu islemleri yürüten protocoiler.

→ DVMRP

→ RPF yapısı kullanılır. En kısa yol

soft state: 1 dk'da bir kendini yeniler. Eğer router a ait bir bilgi yokse dallas kendini korur.

Her 1 dk'da bir ağa yenileniyor. Bunu yapmak için IGMP protokolü kullanılır.

Tunneling

mcast mesajı unicast mesajı gibi kullanılır.
Karsi taraf mesajı alır ve gönder.

PIM protokolü

Dense:

- Grubun üyeleri birbirine yakın
- Bandwidth geniştir (Savunma)

Sparse:

- wrt #: with respect to
 - Grubun üyeleri doğrudan bir vaziyette bulunur.
 - Bandwidth geniş değil.
- (Ölçülü)

→ Gruba girmek için join mesajı kullanılır.

Link Layer and LAN's

- Bu katmanda datanın adı frame
- Tüm host ve routerlara node diyoruz.
- İki komşu node'u bağlayanlara link diyoruz.
- Aşağıdan yukarı 2. katmandır.

data-link layer → bir datagramın bir node'tan komşu node link üzerinden aktarılmasından sorumludur

Datagram ^{dönüştürken} → frame (boşaltılır)

Datagramın başına header konulur.

MAC (Media Access Control - 48 bits)

Bu katmanda IP kullanmıyoruz. Headerın içerisinde MAC adresi olacaktır.

Güvenli Veri Transferi

Bağlantı tipine göre değişir. Kablolu kullanırsak hata payı düşük. Wireless ise hata payı yüksek.

Güvenilir Veri Transferi için bağlantı yapıları farklılık gösterir.

flow control: Alıcının yükünü aşmayacak şekilde
gönderimde bulunmak.

error detection:

Hatırları bulur ve buna göre işlem yapar.

error correction:

1 bitlik veriyi düzeltir.

Link Layer

NIC kartların içinde uygulanır.

Gönderici tarafı

datagram + header = frame
error bitleri ... eklenir.

ERROR DETECTION

EDC → kuyruk kısmı

datagram



datobits + EDC

1) Parity Checking

1 0 1 0 | ① → tek yapmaya çalışıyorsan

1 0 1 0 | 0 → çift yapmaya çalışıyorsan.

ikisini de yapabiliriz



Hello Kitty

FEC: forward error correction

hem hatayı bulduk, hem de düzelttik

Sınavda

Cyclic Redundancy Check (CRC)

D: data bits (d)

R: r bits CRC $\rightarrow$ CRC'yi nasıl buluyoruz?

$$D * 2^r$$

10 bitlik data'nın arkasına 3 bitlik CRS koyarsak

13 toplam

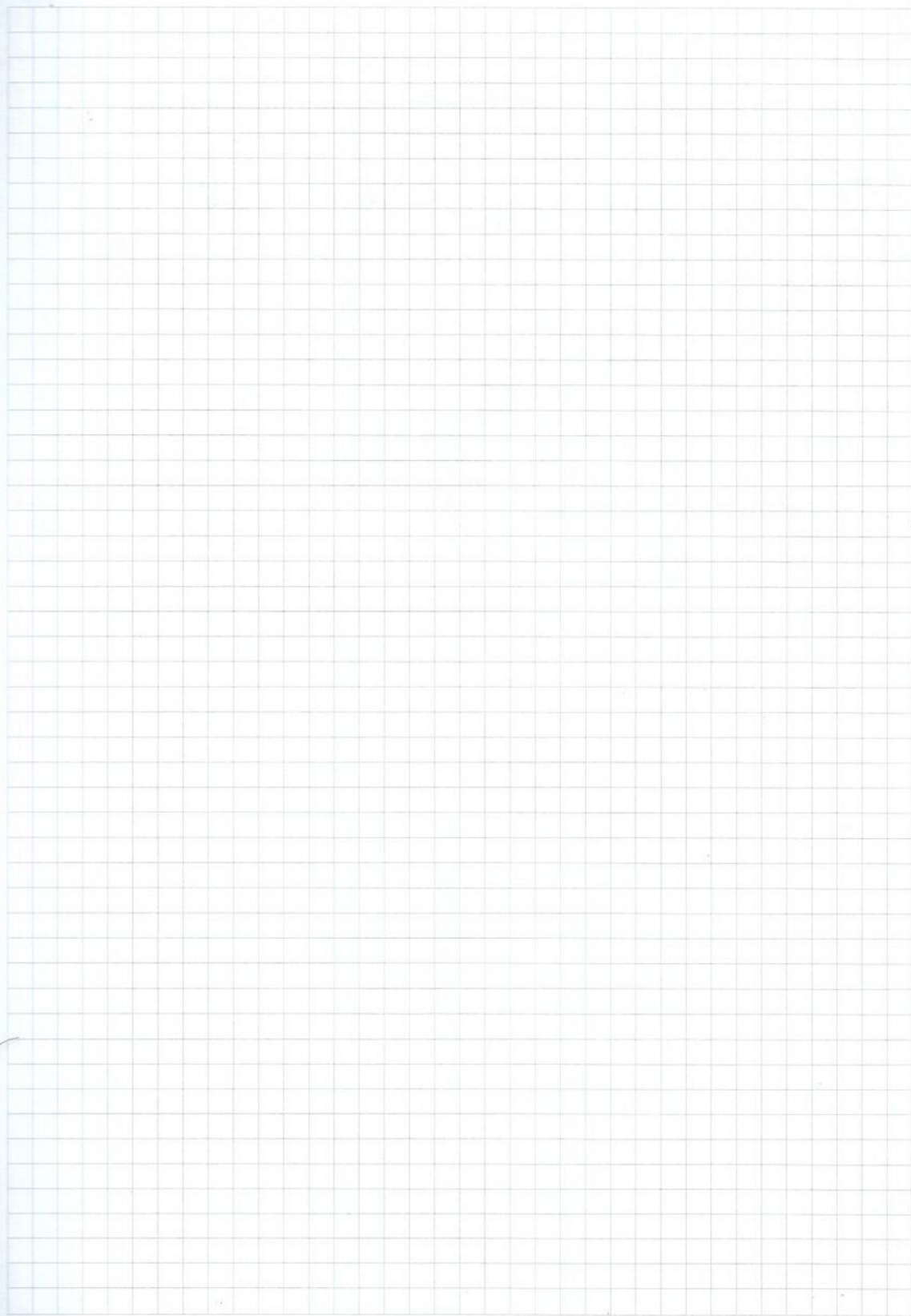
13 sayısı: $G = r + 1$ 'e kolonsiz bölünmeli.

$$D = 101011$$

$r = 3$ bilmiyoruz 000 dedik

$$G = 1001 \text{ olsun}$$

$D / G \rightarrow$ kalan R'yi verir.



Hello Kitty

© 1991, 2011 SANRIO CO., LTD.

DHCP
UDP
IP
Ethernet
Phy



Karşı tarafın routeri
aşağıdan yukarı
bu paketleri alır
tekrar paketler olarak
laptop'a gönderir.

→ IP adresimiz artık var internete girebiliriz.

HTTP kullanırız.

Fakat google'in IP'sini öğrenmemiz gerekir (DNS)

DNS
UDP
IP
Ethernet
Phy



Paketler. → google'in serverından
google'in IP
adresini alır.

Bu sırada ^{ilk routerin} MAC adresine ihtiyacımız var ARP,
kullanmalıyız. Bu adres bulununca DNS isteği
gönderebiliriz.

Artık IP adresimiz de var HTTP'yı kullanabiliriz

HTTP
TCP
IP
Ethernet
Phys.

Artık google girdik.

5100

-

Routing Algorithm

-

Sensory



Hello Kitty

© 1991, 2011 SANRIO CO., LTD.

* forwarding işle routing işlemleri ile gerçekleştirilir.

→ Diyelim ki bir switch yetmedi;
Bir switch daha bağlarız.



kullanılmayan port (kullanılmaz)

Ağın sonallştırılması

Üstten baktığımızda alt ile ilgilenmiyoruz.

→ Çok fazla sayıda birbirleri ile bağlantısı olmayan yapılar var.

→ Bu yapılar farklı olduğu için kendi içlerindeki adresler farklı, algoritmaları farklı.

→ Yukarıdan baktığımızda eşit bir usulle yapılması lazım.

→ Herkesi EŞİT görmek işimizi kolaylaştırır.

ATM ve MPLS

→ farklı ağ, farklı protokolleri var.

→ Network açısından bakarsak bu 2 ağı logic olarak bağlamak zorun dayız

ATM

- Voice, video ve datanın hızlı bir şekilde iletilmesini sağlamak.

MPLS

- Amac forwarding işlemini daha hızlı yapmak.
- switchler ve routerlar birlikte çalıştığı için onlara etiket vermemiz gerekiyor.

SYNTHESIS

Bugüne kadar ki tüm yapıları kullanırsak 

- Öğrenci kampüsün dışına laptop ile girecek
- google'a girecek.
- Laptop ile yapılan bağlantı kablolu.
- Ağın sürekli kullanıcısı değilim bunun için IP lazım.

DHCP → IP adresini sağlar.

ilk bağlantı routerın adresini sağlar.

UDP → DHCP isteği oluşturur.

UDP tarafından bölünür.

IP tarafından

Ethernet

9

"



* Hublar ve switchler tek cildir cihaz.
Routerlar tokildiginda calismaya hemen basla-
maz.

* Switchler tabile'i olustururken en kisa yolu
olmazlar

- Routerlar en kisa yolu bulan algoritmalar.
Hublar da da en kisa yol yok.

* Hublar trofigi duzenlenmez.

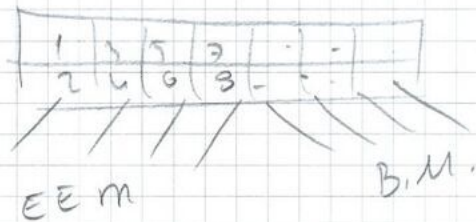
Routerlar trofigi duzenlerler. Hattaki trofik
routerlar uzerinden duzenlenir.

Switchler de routerlar kadar olmasada hattaki
trofigi duzenlerler.

VLANs: motivation

(Sanal yerel ogleri)

→ Tek bir switch'i kullanirlar fakat kendilerine
bolunmus olani (portlara ayrilir). Bu sebeple sanki
farkli 2 switch kullaniyormus gibi



Dolayisiyla EEM
trofigi belirli
portlar araciligi
ile kontrol edilir

→ Trofik izolasyonu
saglanır

Hubs

Gelen datayı çoğaltarak linklere dağıtır.

- Detection yok

Switch

Daha akıllılar

- Frame'leri store ederler.
- Kendi switching tabloları var.
- CSMA'yı kullanırlar.
- Switchlerin varlığından hostlar haberdar değildir.
- Table'ında hedef adresi yoksa her hosta gönderir.

Daha sonra A'ya gönderirken daha önce gönderdiği için bir daha hepsine göndermez. Buna self learning denir.

Switches vs. Routers

- her ikisinde store ve forward yapar.
(bridge = switch)
- Switchler link layer'ın elemanı
- Routerlar network layer elemanı
- Routing tabloları var. Her 2'side tablolarını kendi oluşturur. Switch → self learning
Router → router algoritma