«بسمه»

مرجع درسی

Computer Networking : A Top-Down Approach

The Internet : a "nuts and bolts" view

* bilions of connected computing devices :

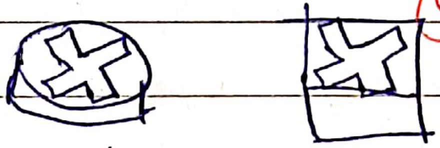
▫ hosts = end systems (به دستگاه‌های که با اینترنت کار می‌کنند) میزبان

در لبه‌های اینترنت هستند

▫ running network apps at internet's "edge"

▫ packet switches : forward packets (chunks of data) بسته‌های داده را دریافت و ارسال می‌کنند

* routers , switches



router — switch

مسئولیت دریافت و ارسال بسته‌ها

فیبر نوری

▫ communication links ← خطوط ارتباطی — ماهواره

* fiber, copper, radio, satellite

* transmition rate : bandwidth

خطوط ارتباطی که قسمت‌های مختلف شبکه را به هم متصل کرده‌اند و از طریق این لینک‌ها ، بسته‌ها می‌توانند بین نودها جابجا شوند

Networks : collection of devices, routers, links: managed by an organization

local or regional ISP, national or global ISP

home network, mobile network, enterprise network

Internet: network of networks

اینترنت: مجموعه‌ای از شبکه‌ها که با کمک ISP ها به هم متصل شده اند

→ الگوریتم توزیع شده

protocols: are everywhere

* control sending, receiving of messages
  HTTP (web), streaming video, skype,
* TCP, IP, wifi, 4G, Ethernet

قسمتی از الگوریتم روی یک دستگاه و قسمت دیگر روی دستگاه دیگر
از طریق ارسال پیام به هم دیگر کاری که قرار است انجام دهند را انجام
میدهند

زبانی که دستگاه‌های مختلف با هم صحبت میکنند از طریق آن ← protocol

Internet standards: استانداردهای اینترنت

RFC: Request for comments

IETF: Internet Engineering Task Force
موسسه

The Internet: a "services" view

Infrastructure: that provides services to applications

* web, streaming video, multimedia teleconfrencing, email, games, e-commerce, social media, interconnected appliances.

بستری که به توسعه اپلیکیشن ها کمک میکند (در دو بعد اقدامات بین کامپیوترها)

→ ایجاد اپلیکیشن ها

provides programming interface to distributed applications:

* "hooks" allowing sending / receiving apps to "connect" to, use Internet transport service.
* provides service options, analogous to postal service.

اینترنت ← شبیه سرویس پست ← حمل و نقل نامه‌ها یا بسته‌هایی
که بین کامپیوترها یا device های مختلف

what's a protocol?   یه سری کارای مشخص

→ specific messages sent
specific actions taken   یک سری پیام‌ها ارسال میشوند
when message received, or other events.   به ازای اون پیام‌ها یک سری کار انجام میشود

network protocols: * computers (devices) rather than humans.

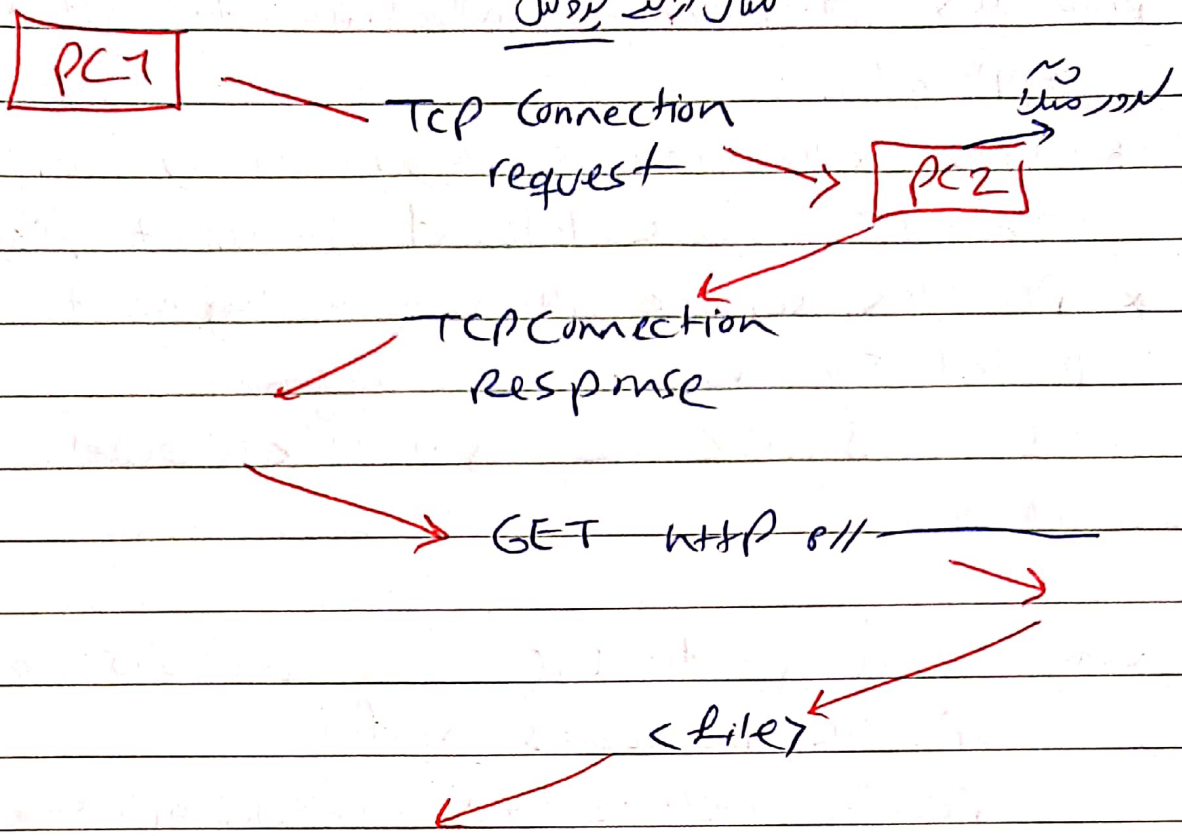
* all communication activity in Internet goveKned by protocols

فعالیت‌ها توسط پروتکل‌ها مدیریت میشوند

Protocols define the format, order of messages sent and received among network entities, and actions taken on message transmission, receipt

برای مجموعه پروتکل میگیم

مثال از یک پروتکل



قسمت هایی از نت که در لبه ها قرار دارد Network edge

قسمت ۱ و End_system /hosts

* Clients and servers
* servers often in data centers

Access Networks & connects hosts to the "edge router"

wired & wireless communication links

switches.

مجموعه‌ای از لینک‌ها و روترها

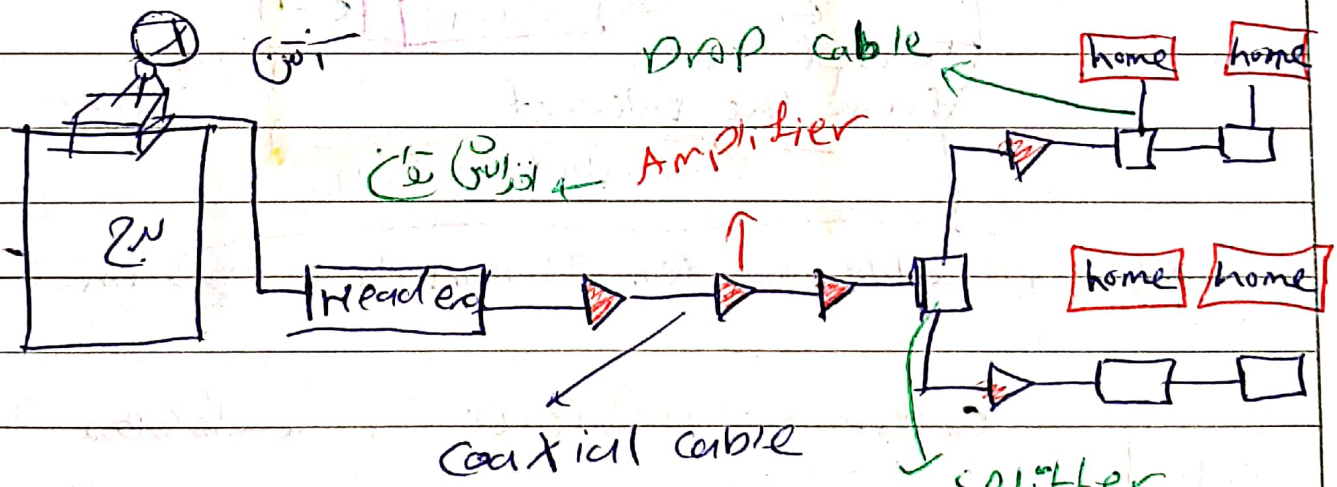
(اولین روتری که ما را به بقیه خارج وصل می‌کند)

که End system را به اولین روتری که می‌بیند وصل می‌کند.

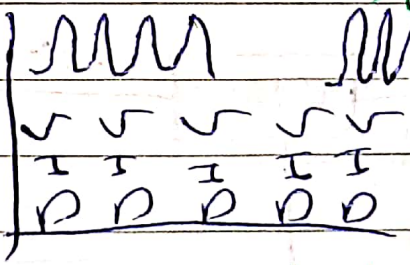
Network core: مجموعه‌ای از روترهای به هم متصل
interconnected routers
network of networks شبکه‌ای از شبکه‌ها
معمولا یک تعدادی از این روترها و شبکه‌ها متعلق به یک ISP ای
regional or national

How to connect end systems to edge router
① residential access nets منافق مسکونی
② institutional access networks (school, company) دانشگاه و موسسه‌ها
③ mobile access networks (wifi, 4G/5G)

Access networks: cable-based access



کانال‌های جدا برای سیگنال‌های جدا

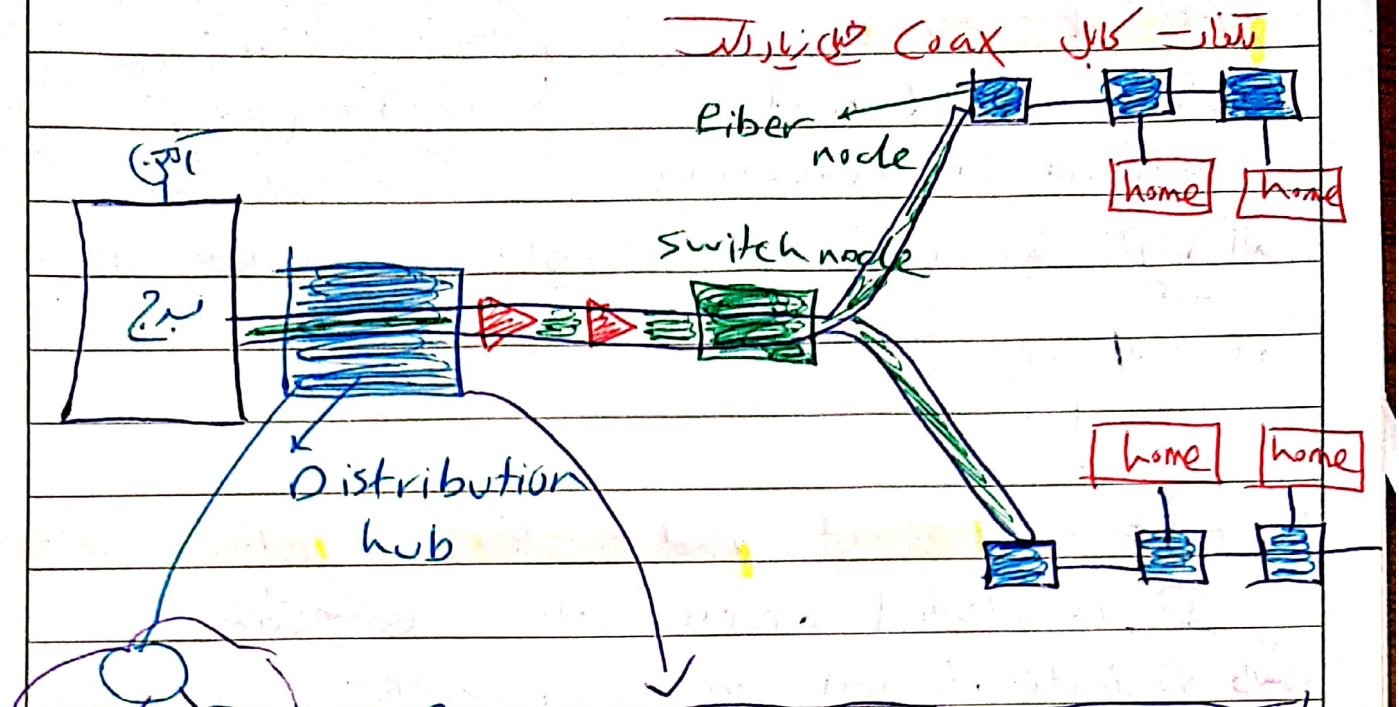


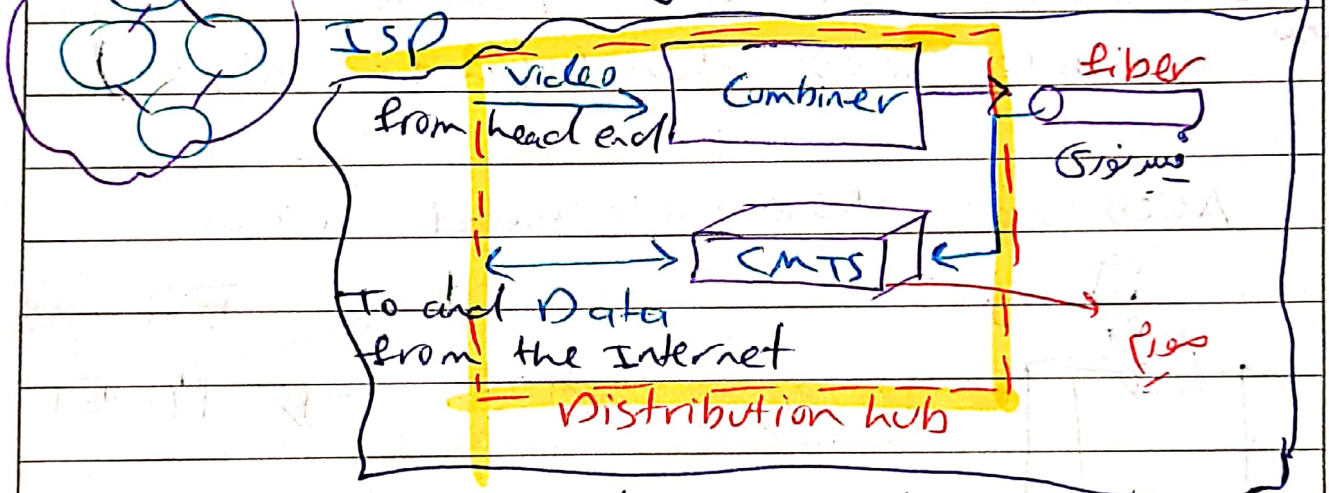
FDM: frequency division multiplexing → different channels transmitted in different frequency bands

از سه نوع مختلف سیگنال با فرکانس

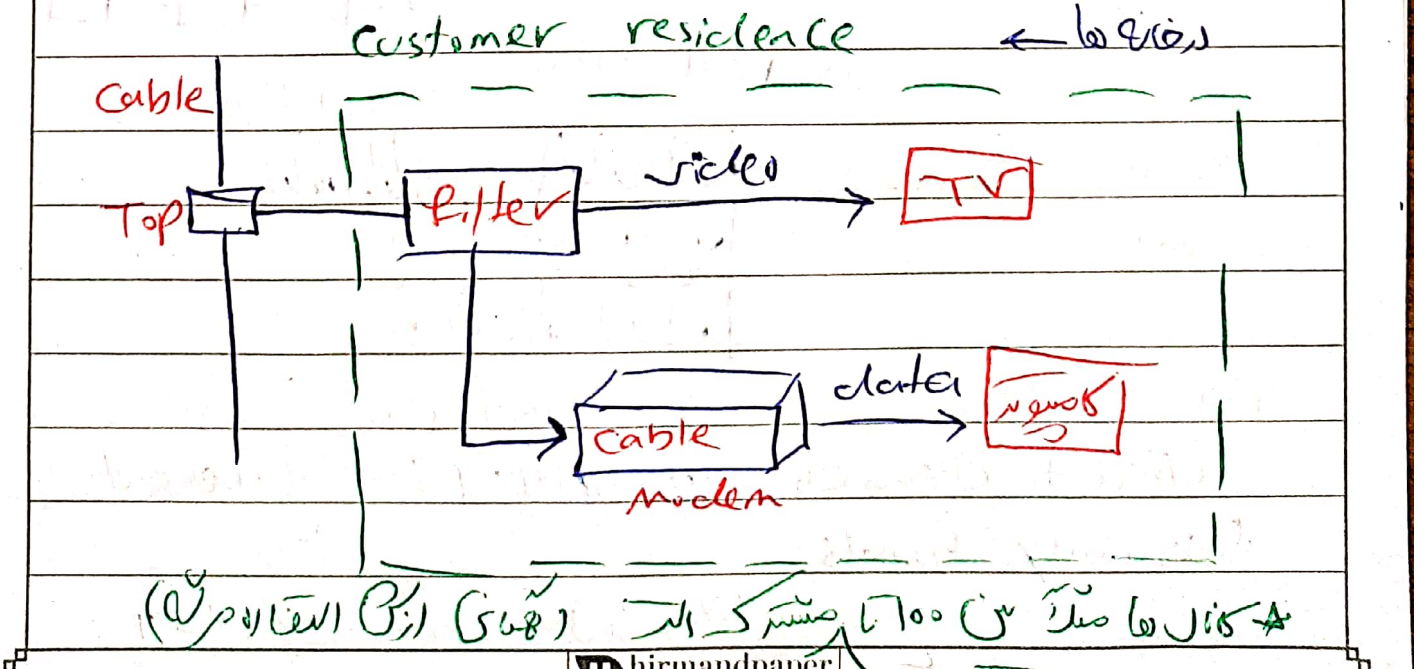
FDM

HFC → Hybird fiber coax





CMTS → Cable modem Termination system



* کانال ها معمولا بین ۱۰۰ تا مشترک دارند (بقیه از این اطلاعات)

* Each channel is shared among a number of subscribers

Downlink: multicasting → (یعنی) سیگنالی که مرخواهد برای یک یوزر ارسال بشه چون کابل coax بین چندتا یوزر مشترک است همه آن بسته را دریافت مرکنند اما اون مودم (modem) خانه‌ای که قرار است بسته را بگیرد آدرسی دارد که داخل بسته‌ی ارسالی قرار داده شده ← با نگاه کردن به آدرس بسته‌ها modem ها متوجه مرشوند که این بسته‌ها برای آن‌ها است یا نه. اگر برای خودش باشد داده استخراج مرشود و برای دستگاه‌های دیجیتال آن خانه ارسال مرشود و modem های دیگر آن بسته‌ها را دور مرریزند

از یک کانال تعداد زیادی مشترک استفاده مرکنند | طبق آدرس‌ها مرفهمند که بسته برای‌شون هست یا نه

Uplink: multiple_access

Down

	uplink	downlink	سرعت

DL: 40 Mbps – 1.2 Gbps ; UL: 30 – 100 Mbps

(استانداردها با Docsis نام گذاری مرشوند)

→ Because of different channel bandwidths / modulations (پهنای باند)

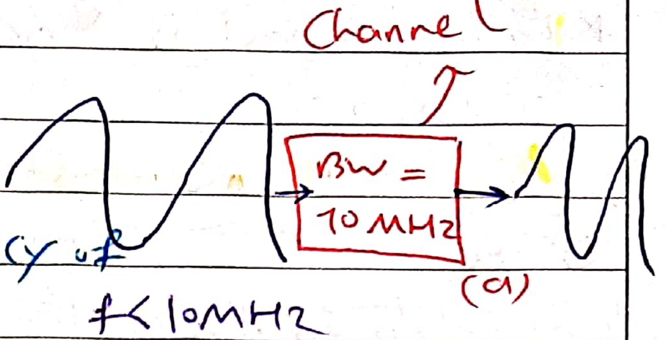
کانال‌ها اشتراکی است → سرعت خیلی هم بالا نیست

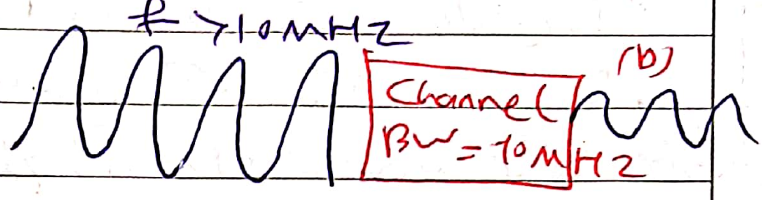
بستگی دارد چه کاربرها در آن زمان از نت استفاده مرکنند یا نه
اگر تعداد زیادی باشند → سرعت نت کم مرشود

سرعت * به زمان روز و به نحوه‌ی استفاده‌ی بقیه‌ی مشترکین بستگی دارد

Bandwidth

unite Hertz (Hz)
limits the max frequency of passing sines.

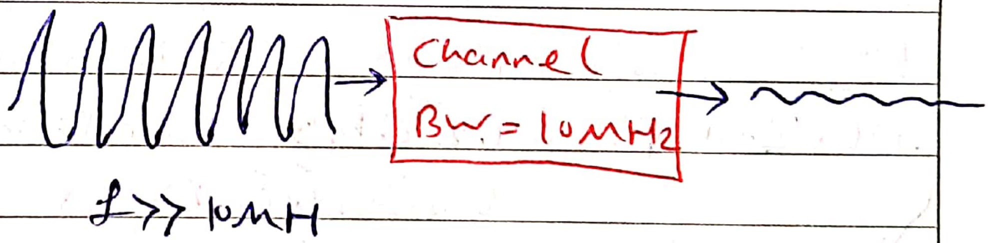




★ اگر سیگنال ورودی به کانال ۵ کمتر از 10MHz باشد ← بدون اینکه دچار تلفات بشه و قدرتش را از دست بدهد سیگنال ورودی را دریافت می‌کند (a)

★ با افزایش فرکانس ورودی (بیش از 10MHz شود) سیگنال خروجی دچار تضعیف می‌شود (b)

این قضیه با افزایش شدت فرکانس ورودی تشدید هم می‌شود ← خروجی تقریباً صفر می‌شود



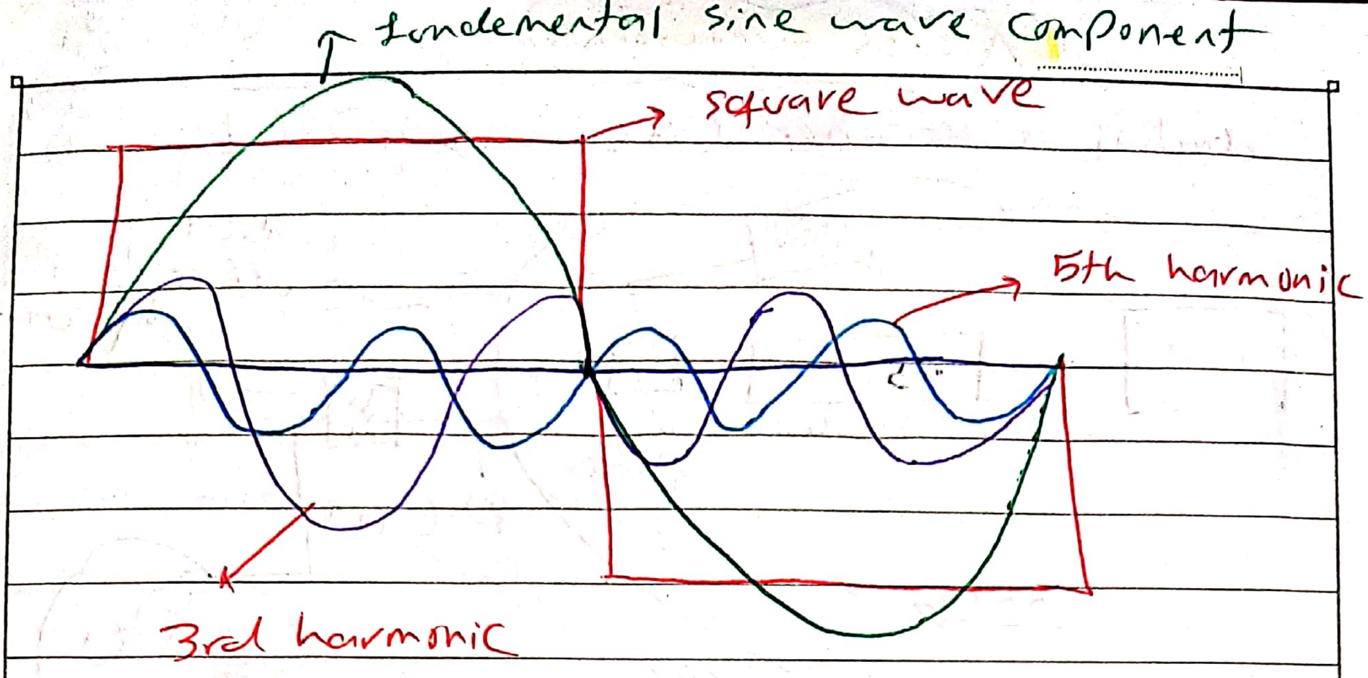
A pulse (carries a 0/1) can be written as the sum of infinite sines (سینوس) → * first harmony, second harmony, ...

★ As the pulse's width decreases, the frequency of the first harmony increases.

★ The BW of the channel limits the pulse rate
aka (also known as) symbol rate
aka (~) baud rate

Baud-rate ∝ BW

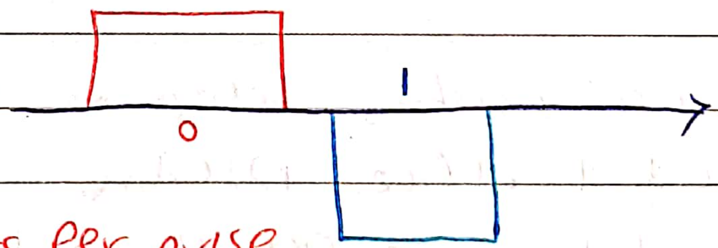
به هر کدام از این ها میگن



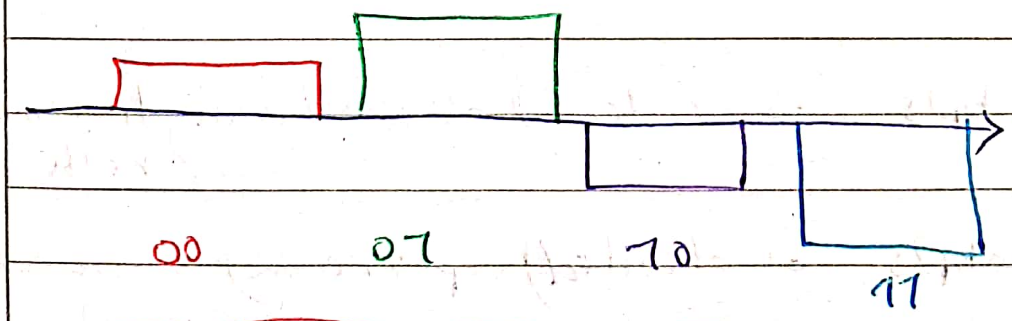
اگر بتونیم کانالی ۴MHz ، band width داره ، میتونیم ۴M پالس در هر ثانیه بفرستیم

Modulation: A pulse can carry a single bit or more than one bit.

→ the number of levels



m = Number of bits per pulse
= log (number of levels)



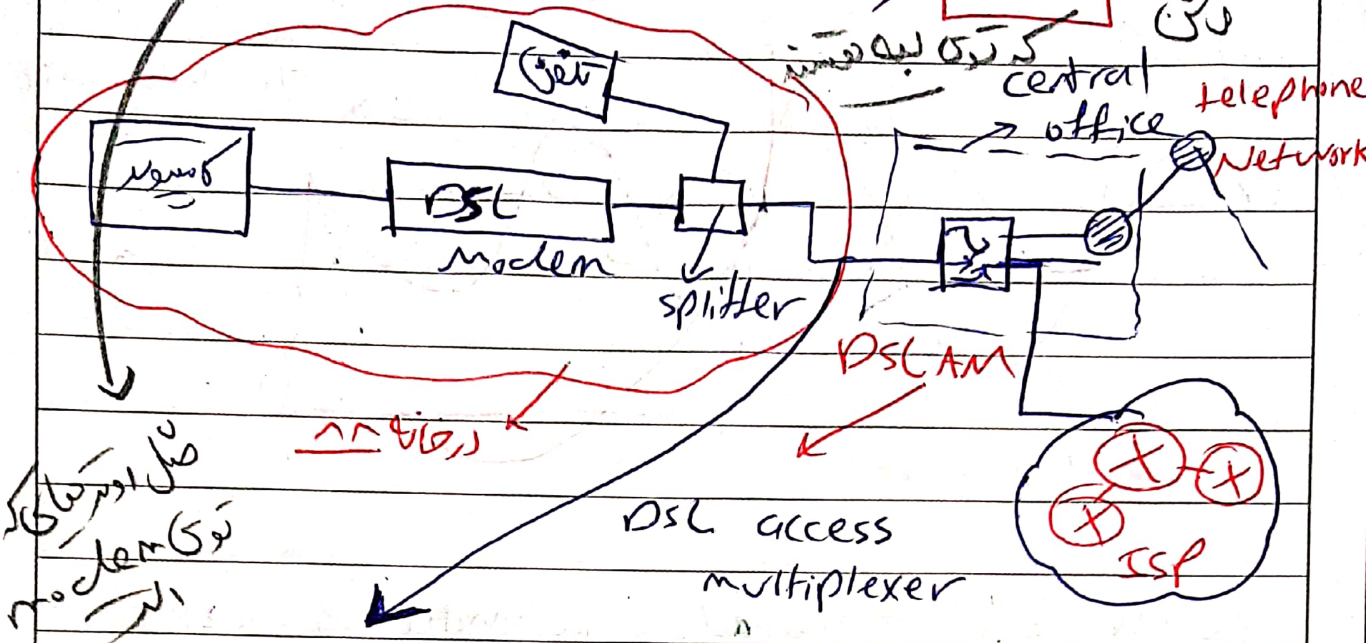
Bit rate = m × Baud-rate

m → چند بیت در واحد زمان

Baud-rate → هر پالس ؟ چند بیت = هر پالس چند بیت رو حمل میکند

Access networks → اولین روترهایی که از لبه دنیا

digital subscriber line → DSL بیشترین وصل است



voice, data transmitted at different frequencies over (dedicated) line to central office
خط ها اختصاصی است

★ use existing telephone line (local loop) to central office DSLAM.
- data over DSL phone line goes to Internet
- voice 〃 〃 〃 telephone net

★ ۲۴ - ۵۲ mbps (dedicated) (downstream) transmition rate
★ ۳.۵ - ۱۶ mbps (dedicated) (upstream) 〃 〃

→ Asymmetric → ADSL → نامتقارن

حجم داده از مرکز به خانه بیشتر است → یعنی Downlink بیشتر است

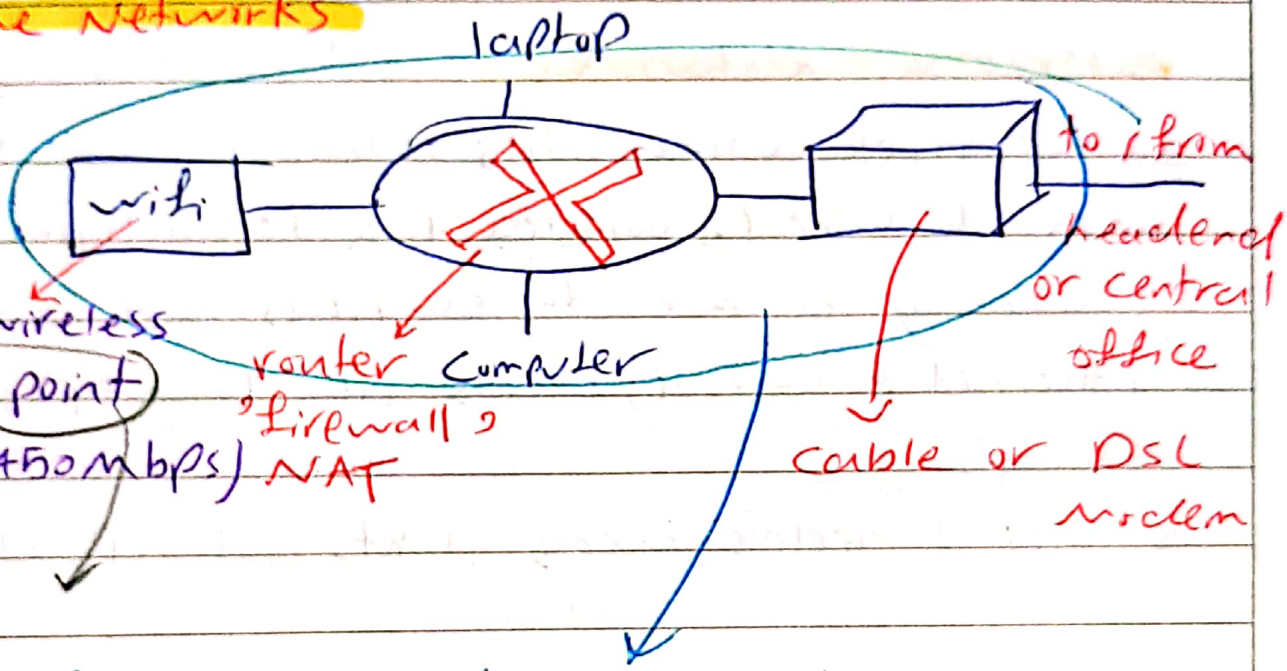
تفاوت سرعت ها به خاطر

>> ADSL is dedicated but not independent!

why? چرا → کاربران کاربرهای مختلف روی کیبل اینترنتی هم اثر میگذارد با وجود اختصاری بودن

>> ADSL Channels vary → ADSL modems work adaptively → time of the day
→ Distance to CO
→ متفاوت است

home Networks



often combined in one box

wireless access Networks

shared wireless access network connects end systems to router

* via base station → aka → access point

(WLANs) → wireless local Area Networks

* typically within or around building (~100 ft)
* 802.11 b/g/n (wifi)
  - 11, 54, 450 mbps transmition rate

wide-area cellular access networks

* provided by mobile, cellular network operator (10's km)
* 10's Mbps 2G, 3G, 4G
* 4G cellular networks (5G coming)

enterprise networks

* companies, universities, etc....
* mix of wired, wireless link technologies, connecting a mix of switches and routers
* Ethernet: wired access at 100Mbps, 1Gps, 10 Gps
* WiFi: wireless access points at 11, 54, 450 Mbps

Links (physical media)

bit: propagates between transmitter/receiver pairs

physical link: what lies between transmitter & receiver,

guided media:
- signals propagate in solid media: copper, fiber, coax

unguided media:
- signals propagate freely → radio

جنس خوب و قطر بزرگ و لرکت بار را و کلمه ...

مس

Twisted pair (TP)

زوج سیم

- two insulated copper wires → سیم مس
  - category 5 : 100 Mbps , 1 Gbps Ethernet
  - 6 : 10 Gbps Ethernet

Coaxial Cable :

هدایت کننده

★ two concentric copper conductors

★ bidirectional — به شکل مسی و الستر... جنس

★ broadband :

★ multiple frequency channels on cable

★ 100's mbps per channel.

Fiber optic cable :

① ★ glass fiber carrying light pulses, each pulse a bit

② ★ high-speed operation :

. high-speed point-to-point transmition (10's - 100's Gbps)

③ ★ low error rate :

★ repeaters spaced far apart

★ immune to electromagnetic noise

④ تضعیف سیگنال داخل فیبر نوری خیلی پایین است ، پهنای باند و سرعت خیلی بالایی دارند

★ احتمال خطا خیلی کم است

دلیل ← noise کم است و امواج تداخل ندارند → مشکل بیلینگ

از فیبر نوری در Data Center ها استفاده می‌شود ⑤

مشکل : گران و نصب کار ندارند | لینک‌های بین قاره‌ای و ...

hirmandpaper

wireless radio

از فرکانس مختلف امواج مغناطیسی برای ایجاد کانال‌ها و یا لینک‌های مختلف

* signal carried in various "bands" in electromagnetic spectrum
* no physical wire
* broadcast , "half-duplex" (sender to receiver)

چالش‌ها * propagation environment effects :

1. * reflection → یک سیگنال می‌فرستیم اما گیرنده چندین کپی از آن را دریافت کند (چالش اول)
2. * obstruction by objects
3. * interference / noise

* هر فرکانس و باند را به یک کار خاص ، اختصاص می‌دهیم.

* چرا سیگنال‌ها روی هم مزاحمت ایجاد نمی‌کنند؟ چون در فرکانس‌های مختلف و باندهای مختلف این امواج را ارسال و دریافت می‌کنند.

* ارزان * در دسترس * احتیاج به نصب ندارند

چالش دوم ← وجود موانع بین فرستنده و گیرنده — باعث تضعیف سیگنال می‌شوند

چالش سوم ← ایجاد تداخل ، noise ، رباندهای یکسان

* اهمیتی کم است * در یک باند فرکانسی ، لینک‌ها یک طرفه هستند

فقط یک فرستنده از یک باند فرکانسی برای ارسال داده‌هایش استفاده کند

مگر اینکه از هم خیلی دور باشند

Radio Link Types

محیط‌های داخلی خانه و اداره

wireless LAN (wifi) → 10-100's Mbps ; 10's of meters

wide-area (e.g , 4G cellular) → 10's Mbps over ~10 Km

→ محیط‌های خارج ساختمان — رنج زیادی (range) را لازم دارند

Bluetooth : cable replacement → short distances , limited rates

جایگزین کابل‌ها

terrestrial microwave

point to point ; 45 Mbps channels

satellite up to 45 Mbps per channel

ارتباط ماهواره‌ای 270 msec end-end delay

* تأخیر زیادی دارند و لینک زیاد

Network Core

جلسه ۴

* A network at core : mesh of interconnected routers/switches که روتر و سوئیچ‌ها + لینک‌هایی که آنها را متصل می‌کنند

* operating based on — ① packet switching / ① circuit switching

بر اساس کدوم عملکرد — دو نوع مختلف معماری

packet - switching

* hosts break application-layer messages into packets

برای ارسال مستقیم پیام به packet های مختلف تقسیم می‌کنند و بعد ارسال می‌کنند.

packets = payload + header

قسمت اطلاعات کاربری — قسمت دارد

آدرس فرستنده و گیرنده داخل header قرار دارد.

network forwards packets from one router to the next : across links on path from source to destinertion

فرستادن packet ها از طریق لینک‌ها از مبدا به مقصد

Packet switching network : functionalities

forwarding: → به صورت محلی
aka "switching"
local action: move arriving packets from router's input link to appropriate router output link

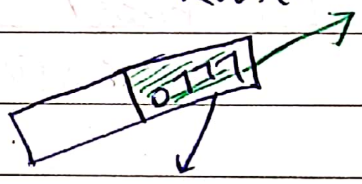
routing algorithm

local forwarding table

header value	output link
0100	3
0101	2
0111	2
1001	1

پورت های خروجی روتر

Router



destination address in arriving packet's header

Routing: **global** → action
- ☐ determine source - destination paths taken by packets
- ☐ routing algorithms

مقادیر این جدول های forwarding را از کجا می‌آوریم؟

اگر ابعاد شبکه کوچک باشد و شبکه مدام در حال تغییر نباشد این کار را می‌شود به صورت دستی انجام داد ← بین هر فرستنده و گیرنده یک مسیری را به صورت الگاریتم تعریف کرد و این جدول های که داخل روترها هستند را طوری مقدار دهی کرد که اگر یک روتر به یک مقصدی، بسته ای ارسال می‌کند بسته را به مقصد های برسانند.

به دلیل خرابی لینک ها و روترها و سوئیچ ها ممکن است باشد

ولی اگر شبکه ها زیاد باشد ← به دلیل ترافیک شبکه هم ممکن است باشد

algorithm

routing ← با الگوریتم ها تعیین می‌کنیم ← به صورت دستی نمی‌شود

routing algorithm ها ← protocol ها ← مقادیر این جدول خود را پر کنند
را به گونه بهینه تعیین می‌کنند و مسیریابی های بهینه را انجام دهند.

>> No Resource reservation (no call setup)

منابع و تعیین مسیر و اختصاص قسمتی از لینک ها، پهنای باند یا یک اتفاقی که می‌خواهد
ایجاد شده ← رزرو و هماهنگی در packet switch ها (نداریم)

Reserve اینها نداریم → + (مزیت) (Resource sharing) (statistical multiplexing)

منابع به طور مشترک استفاده می‌شوند ← آبرومندی

چندتایی کردن
(مثلا تلفن همگانی)

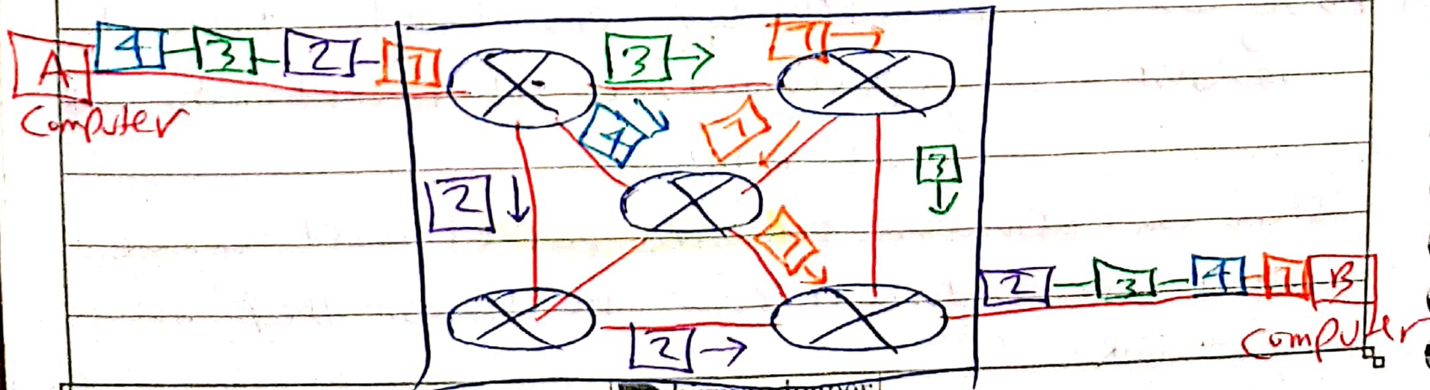
(ضرر) → Different packets are routed independently and may

① take different paths ← ترتیب ارسال با دریافت packet ها یکی نیست
(از مسیرهای مختلف استفاده کنند)

② have different delays (missing order) ← تاخیرهای مختلف

③ be dropped (lost) گم شدن بسته ها

BEST EFFORT! → تضمینی برای ارسال بسته و میزان تاخیر بسته ها نمی‌دهد



۸۸ (ترتیب بسته ها کوهن /۰/)

delays

① تاخیر انتشار امواج الکترومغناطیسی در لینک‌ها که به طول لینک و جنس لینک بستگی دارد

② تاخیر ارسال : بسته به ظرفیت لینک

③ تاخیر صف که در داخل Router ها

switch ها ناشی از تاخیر ارسال و انتقالی

هم‌زمان بیش از یک کاربر در شبکه

چون لینک ها و روتر های مختلف در مسیرها وجود دارند ← تاخیرها می‌توانند متفاوت باشد

Store and forward ← کاری که Router باید انجام دهد

در شبکه های packet switch

کل بسته را باید دریافت کند و بعد عمل forward را انجام دهد.

وقتی مشخص شد برای چه مقصدی باید ارسال کنند بسته ها را (از طریق) forwarding table ← ظرفیت لینک

یعنی مشخص می‌کند چه مقدار طول می‌کشد تا بسته ارسال شود

مثلاً اگر طول packet : L بیت باشد و ظرفیت لینکی که می‌خواهد از آن ارسال شود R bps باشد ←

$\frac{L}{R}$ (s)

تا بسته روتر را ترک کند و همه‌اش ارسال شود

packet transmission delaye

takes L/R seconds to transmit (push out) L-bit packet into link at R bps

store and forward : entire packet must arrive at router before it can be transmitted on next link

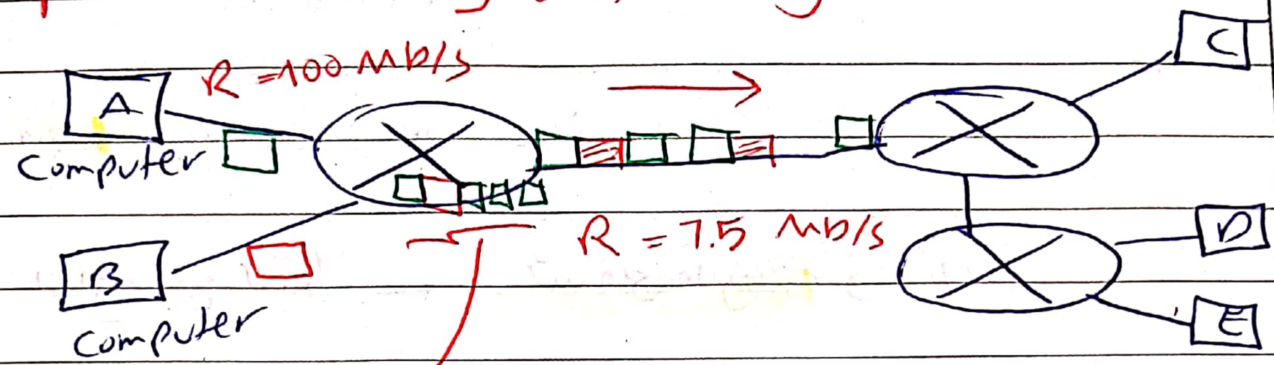
ex: L = 10 kbits R = 100 Mbps

* one-hop transmission delay = **0.1 msec**

صف می‌تواند تشکیل شود فقط وقتی ترافیک زیاد شود ← یعنی rate سرویس دهی
کمتر از rate تقاضا شود

تأخیر صف ← وابسته به ترافیک شبکه

packet switching: queueing



queue of packets waiting for transmission over output link

Packet queuing and loss: if arrival rate (in bps) to link exceeds transmission rate (bps) of link for some period of time:

* packets will queue, waiting to be transmitted on output link
* packets can be dropped or lost if memory (buffer) in router fills up.

یکی از دلایل گم شدن بسته‌ها در داخل شبکه ← ایجاد صف در داخل روترها که رخ می‌دهد و پر می‌شود

این نوع تأخیر وابسته به ترافیک شبکه است. ← تأخیر صف

* چون بسته‌ها از مسیرهای مختلفی می‌آیند به مقصد می‌رسند، میزان تأخیر صف متفاوت است.

* اگر بافر روتر پر شده باشد ← بسته drop می‌شود.

« Circuit switching »

در این روش برخلاف روش قبلی ، منابع به صورت end to end ، رزرو می‌شود
و این منابع به صورت انحصاری در اختیار اون ارتباط قرار می‌گیره

مزیت ← اگر یک ارتباط شکل گرفت و منابع موجود بود و در اختیار یک ارتباط قرار گرفت
ترافیک شبکه روی عملکرد اون ارتباط دیگه تاثیری نداره

ضعف ← از منابع خوب استفاده نمی‌شود

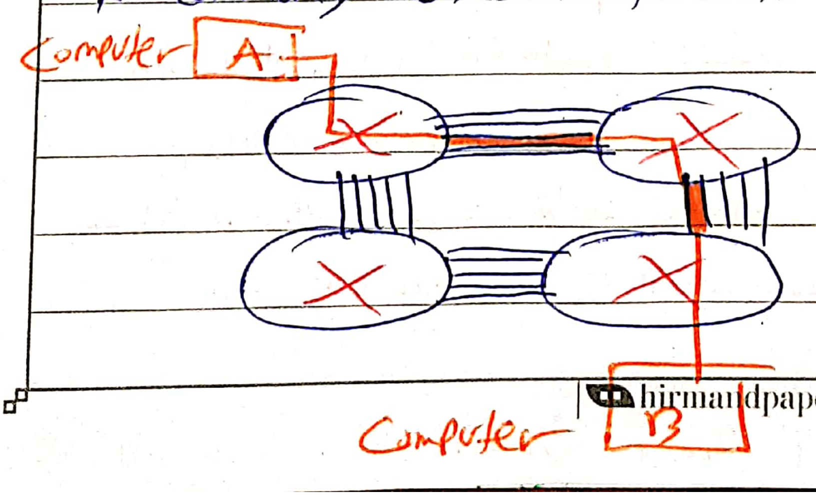
از این نوع ارتباط ← در شبکه‌های تلفن ، روش اصلی است

* end-end resorces allocated to , reserved for call between source and destination

* dedicated resources , no sharing
  - circuit-like (guaranteed) performance

* Circuit segment idle if not used by call (no sharing)

* commonly used in traditional telephone networks.



→ each links has 4 circuits

(3) call gets 2nd circuit in top link and 1st circuit in right link.

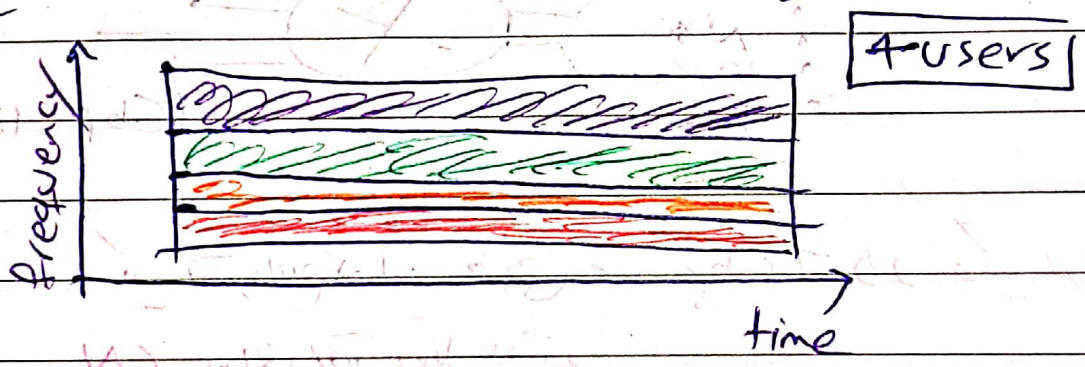
می‌توانیم به ۲ روش منابع یک لینک را تقسیم‌بندی کنیم
FDM ①
TDM ②

Frequency Division Multiplexing (FDM)

* optical, electromagnetic frequencies divided into (narrow) frequency bands.

در این روش پهنای باندی که لینک دارد به چند کانال (sub-band) تقسیم می‌شود و هرکدام از این کانال‌ها را اختیار یک ارتباط قرار می‌گیرد.

مثلاً اگر ۴ کاربر داشته باشیم پهنای باندمان را به ۴ قسمت تقسیم می‌کنیم



روش بعدی تقسیم منابع (حوزه‌ی) زمان است

Time Division Multiplexing (TDM)

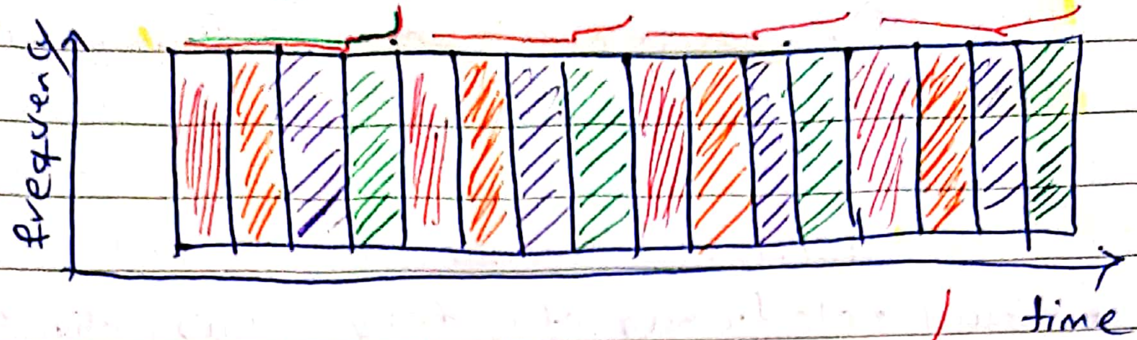
* time divided into slots
* each call allocated periodic slot(s) can transmit at maximum rate of frequency band (only) during its time slots.

یک فریم را تعریف می‌کنیم که هر فریم تشکیل شده از یک یا چند slot و هر slot را به یک کاربر اختصاص می‌دهیم.

مثلاً اگر ۴ کاربر داشته باشیم ← یک فریم ۴ تا slot تقسیم می‌شود

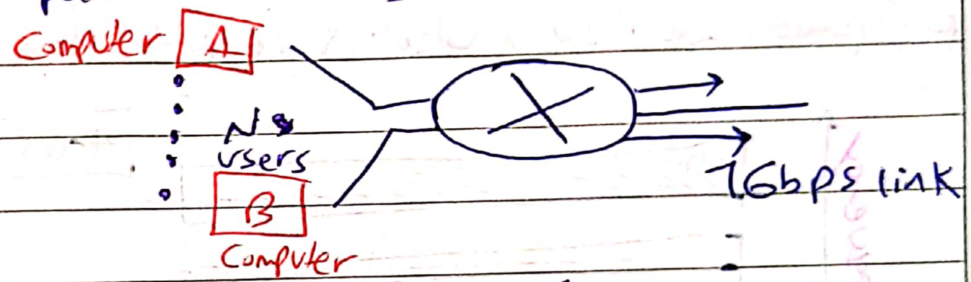
هر فریم ۱۶ تا slot دارد

۴ تا کاربر داریم ←



مثل line بندی خیابان‌ها برای استفاده‌ی همزمان ماشین‌های مختلف
- → درحوزه‌ی فرکانس
- → درحوزه‌ی زمان

example packet switching vs circuit switching



یک روتر داریم که لینک خروجی $1 Gb/s$ دارد و یک سری کاربر به تعداد N، به این روتر وصل هستند

* هرکدام از این کاربرها وقتی active هستند و می‌خواهند از اینترنت استفاده کنند $100 Mb/s$ نیاز دارند

* هر node فقط ۱۰٪ زمان فعال است

how many users can use this network under circuit-switching and packet switching?

→ circuit switching: 10 users $\frac{1G}{100M} = \frac{10^9}{10^8}$

Packet switching: with 35 users,

probability > 10 active at same time is less than 0.0004*

احتمال اینکه بیش‌تر از ۱۰ نفر در یک لحظه فعال باشند کمتر از 0.0004 می‌شود

اگر تعداد بیش از ۱۰ شود مشکل پیدا می‌کنیم چرا؟ چون

< sum rate > ورودی < خروجی

اما احتمال کم است

می‌توانیم از buffering → استفاده کنیم تا مشکل را حداقل کنیم

یعنی اگر تعداد کاربرها > ۱۰ بودند بسته‌ها در صف‌ها می‌مانند. مگر اینکه تعداد کاربرها انقدر زیاد شود که بسته‌هایی که هستند buffer را پر کنند و بسته‌های جدید drop شوند.

Q: How did we get value 0.0004 ?

بازیکن های بسکتبال که به حلقه آویزان می‌رسونه

is packet switching a "slum dunk winner"?

great for "bursty" data - sometimes has data to send, but at other times not

- resource sharing
- simpler, no call setup.

excessive congestion possible: packet delay and loss due to buffer overflow

- protocols needed for reliable data transfer, congestion control.

Q: How to provide circuit-like behavior with packet-switching?

"its complicated" we'll study various techniques that try to make packet switching as "circuit-like" as possible

Internet structure

«هر یک نحوه ی اتصال شبکه های مختلف با هم»
یا شبکه های دیگری

* hosts connect to Internet via <u>access</u> Internet service providers (ISPs)

«host ها از طریق شبکه های دیگری به ISP ها متصل می شوند»

* access ISPs in turn must be interconnected.

خود ISP ها هم باید به هم متصل شوند ← تا host ها بتوانند برای هم بسته ارسال کنند و با هم در ارتباط باشند

• so that any two hosts (anywhere) can send packets to each other.

→ «یک شبکه پیچیده داریم به نام اینترنت»

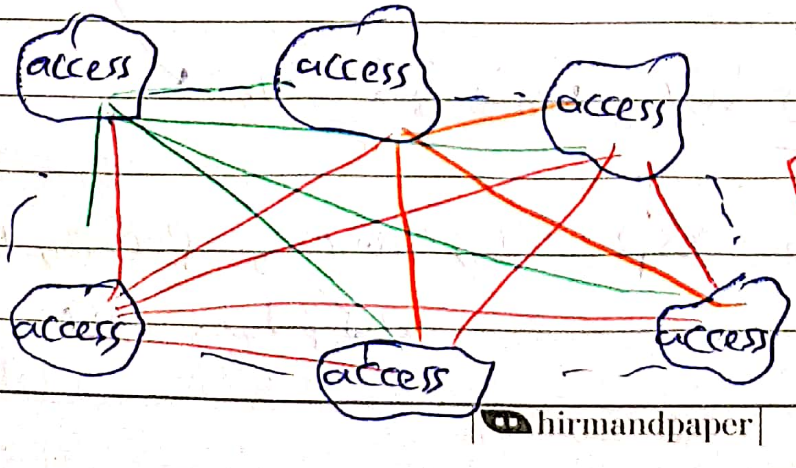
* resulting network of networks is very complex
• evolution driven by economics, national policies

Question: given millions of access ISPs, how to connect them together?

① یک روش برای ارتباط ISP ها با هم ← بین هر کدام از اونا یک لینک وجود داشته باشه

→ (یک گراف کامل باشه) → node ها = access Network ها

یال ها = بین هر کدام از اون access network ها



گراف کامل

→ قابل پیاده سازی نیست چرا؟

$O(n^2)$ connections

به جای ارتباط ۲ به ۲ ← از ساختار حالت ستاره استفاده می‌کنیم.

یک Network به نام global ISP ایجاد می‌کنیم که در کشورهای مختلف تجهیزات دارد و access ها می‌توانند به روتر های این global ISP متصل شوند تا بتوانند به هم متصل شوند

هر بین شرکت را داریم که سرویس پوشش جهانی را ایجاد می‌کنند

و access networks ها هم آن آزادی را دارند که انتخاب کنند از کدام ISP استفاده کنند

به این ISP ها که global هستند ← tier-1 گفته می‌شود

IXP ← internet exchange point

تولید کننده شرکتی برای ایجاد ارتباط بین ISP ها

Peering link ارتباط دو تا ISP هم‌رده و برای سرویس که به هم می‌دهند پول می‌گیرند

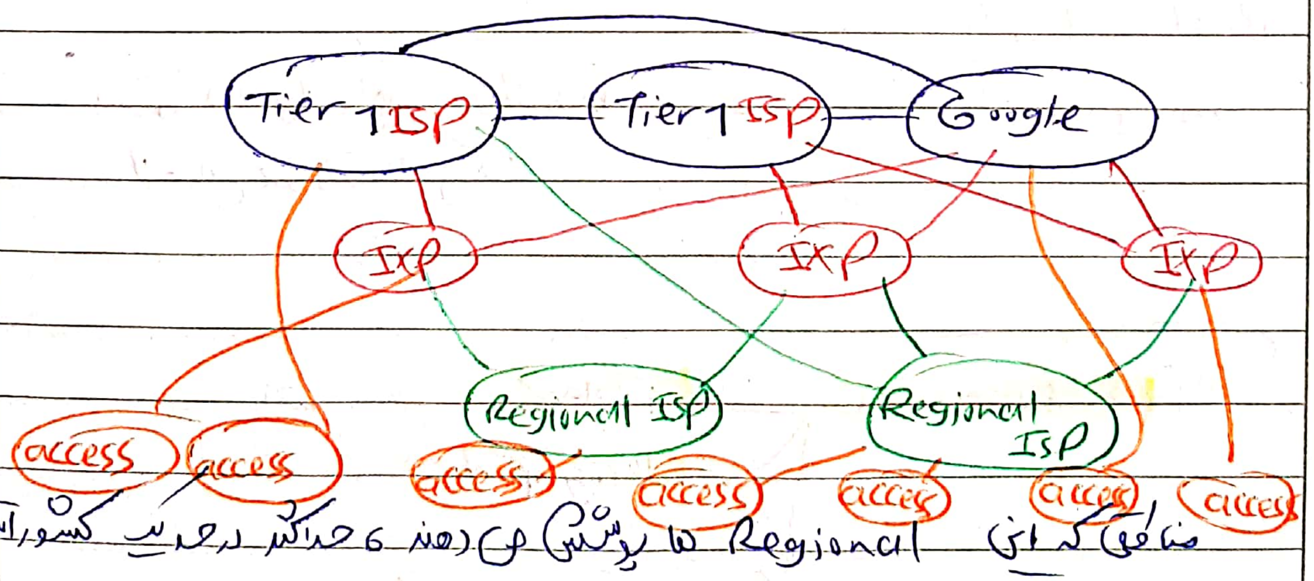
regional ISP ← ISP منطقه ای ← access ها معمولاً Network
این ISP ها متصل می‌شوند و بعد
این regional ISP ها به Tier-1 متصل می‌شوند

در کنار ISP ها یک سری Network های شرکت‌های بزرگ هم داریم

Content provider Network

شرکت‌های مثل google و Facebook و microsoft
برای اینکه بتوانند سرویس‌های خود را هم ارزان‌تر و هم مناسب‌تر از نظر تاخیر
ارائه دهند data center های مختلف در جاهای مختلف زمین است و برای این ارتباط
بین data center ها از شبکه خصوصی استفاده می‌کنند

ترجیح می‌دهند به ISP های Tier-1 متصل نشوند به دلیل (تاخیر، هزینه، سرویس‌ها)
↓
به tier 2 وصل می‌شوند



مناطقی که این Regional ها پوشش می‌دهند حداکثر در حد یک کشور است

access ISP ها به صورت معمول از regional ISP ها سرویس می‌گیرند
ولی می‌توانند اتصال مستقیم به Tier-1 هم داشته باشند.

"tier-1" commercial ISPs (e.g. level 3, Sprint, ...)
, national & international coverge.

Content provider networks (Google, Facebook)
: private network that connects its data centers to Internet, often by passing tier-1, regional ISPs

session 5

circuit switch → FDM
→ TDM

(Guaranteed) Reserved

چون که منابع از قبل رزرو شده، گارانتی هست

(مثل رزرو کردن رستوران ها)

packet را ارسال می کنیم بدون اجازه گرفتن
و منتظر نمی مانیم

packet switch → statistical multiplexing

منابع به صورت مشترک بوده و بر حسب تقاضا تخصیص داده می شوند

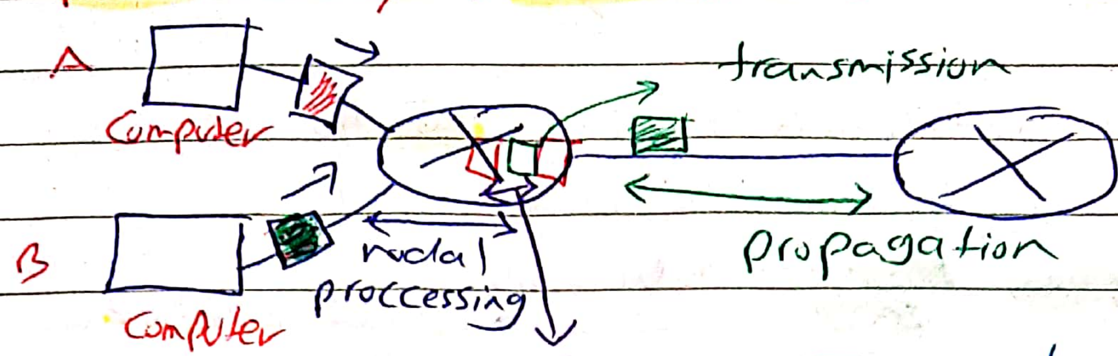
Best Effort
تعیین می دهند

در صورتی که حجم ترافیک bursty باشد،
از منابع بهتر استفاده می کنند.

Performance → Delay (تاخیر)
→ Throughput (گذردهی)
→ Loss (گم شدن بسته ها)

روتر یا سوییچ ← node

Packet delay : four sources



queueing ← پیچیده است

معمولاً صرف نظر میکنیم ← ... → فاصله دو نقطه

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

پردازشی — اگر طول لینک کم باشه میتونیم صرف نظر کنیم

d proc : nodal proccessing

- □ check bit errors — چک کردن اینکه خطای هست یا نه ①
- □ determine output link — مشخص کردن اینکه کدام لینک برای خروجی ②
- □ typically < microsecs — مثلا

پیش که در حد میکروثانیه order ← تاخیر پردازشی — زیاد نیست — dp

الف

d_{queue} : queueing delay

- □ time waiting at output link for transmission
- □ depends on congestion level of router

چون ظرفیت یک لینک محدود است اگر نرخ بسته های که میخواهند از اون لینک استفاده کنند > ظرفیت اون لینک باشه یک تاخیر داریم

پس یک Buffer میگذاریم و بسته ها را داخل اون ذخیره میکنیم و هر وقت بسته‌های ابتدای صف که کارشون تمام شده بقیه را ارسال کنیم.

تاخیر صف ← فرمول مشخصی ندارد

d_{trans} : transmission delay :

* L : packet length (bits) : طول بسته
* R : link transmission rate (bps)
* $d_{trans} = \frac{L}{R}$ → واحد = ثانیه

→ مدت زمانیه طول می‌کشد تا یک بسته کامل ارسال شود

d_{prop} : Propagation delay :

d : length of physical link — طول لینک

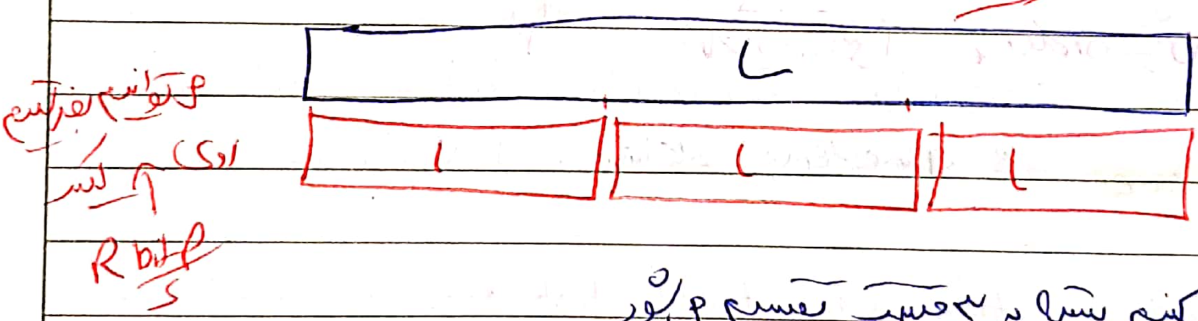
s : propagation speed ($\sim 2 \times 10^8$ m/sec) — سرعت انتشار

$d_{prop} = \frac{d}{s}$ → d: distance, s: speed

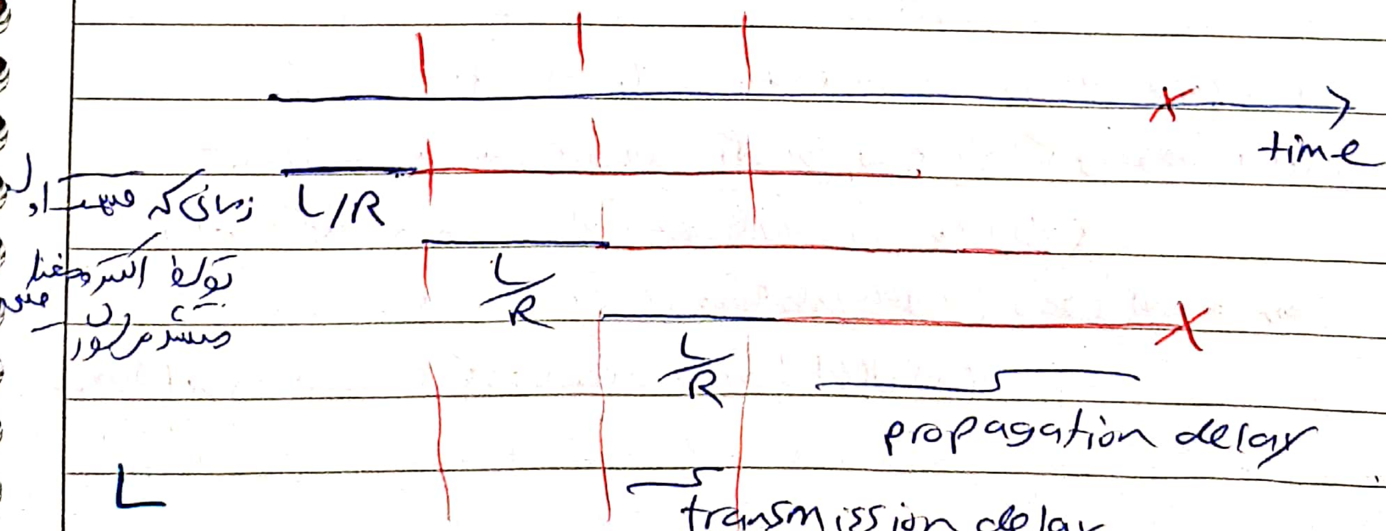
d_{trans} and d_{prop}

روی هم افتادن — انتشار

overlap. propagation — بسته



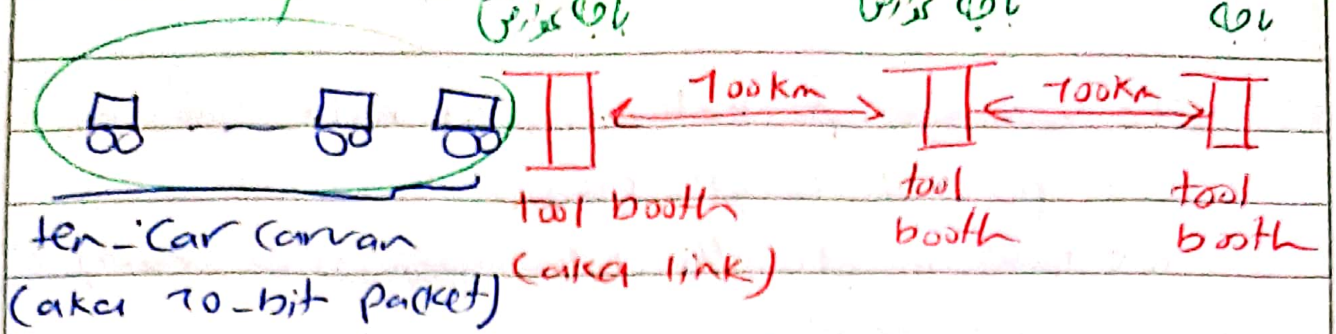
فرض می‌کنیم بسته به ۳ قسمت تقسیم می‌شود



3 L/R + propagation delay = کل زمانی که طول کشیده ارسال شود :

کاروان مثل بسته   ماشین ← بیت

Carvan analogy



هر بسته ← ۱۰ بیت

- car ~ bit و carvan ~ packet;
  toll service ~ link transmission

- toll booth takes 12 sec to service car (bit transmission time)

- "propagate" at 100 km/hr

Q: How long until carvan is lined up before 2nd toll booth?

- time to "push" entire carvan through toll booth onto highway = 12 * 10 = 120 sec

زمانی که کل کاروان از باجه عبور می‌کند

- time for last car to propagate from 1st to 2nd toll booth: $\frac{100km}{100km/hr}$ = [1hr] → 60 min

A: 60 + 120 sec (2min) = [62] min

کل کاروان delay که از باجه اول عبور کنه تا باجه دوم برسه

این اول های قبلی overlap را می‌گیرد

(اگر یک ماشین زودتر رسید کار پردازشش (انتقالش) انجام می‌شه تا بقیه برسند)

Packet queueing delay (revisited)

a : average packet arrival rate ورود پکت - rate

L : packet length (bits) تعداد بیت های داخل بسته

R : link bandwidth (bit transmission rate) → بیت ریت

$\frac{L \cdot a}{R}$: $\frac{\text{arrival rate of bits}}{\text{service rate of bits}}$ "traffic intensity"

نزدیک به صفر (شدت ترافیک) — شدت ترافیک

$\frac{L \cdot a}{R} \sim 0$: avg. queueing delay small

$\frac{L \cdot a}{R} > 1$: more "work" arriving is more than can be serviced, average delay infinite.

(بیت ریت ورودی) > (بیت ریت خروجی) → تاخیر صف بی نهایت

۲ عامل در ایجاد تاخیر صف ← ۱) نرخ داده ورودی

۲) ... خروجی

۲ ← این مولفه ها در اینکه چقدر طول صف بزرگ باشد موثر اند.

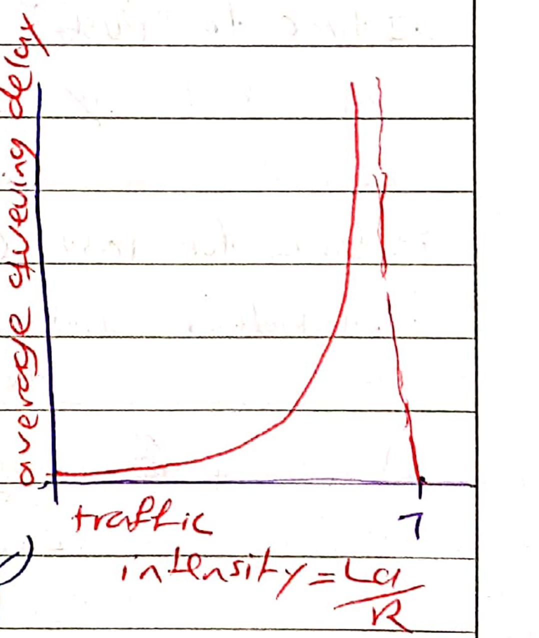
تاخیر صف به جنس ترافیک بستگی دارد

مثلا نرخ ورود ماشین > خروج ← ترافیک دیگر جاده ای

(delay) به بی نهایت میرود

نزدیک به ۱ میشه

$\frac{La}{R} \to 1$: avg. queueing delay large (به سمت بی نهایت میل میکنه delay)



نموداری که رابطه ی بین مولفه delay و شدت ترافیک را نشان میدهد.

"Real" Internet delays and routes

* what do "real" Internet delay & Loss look like?

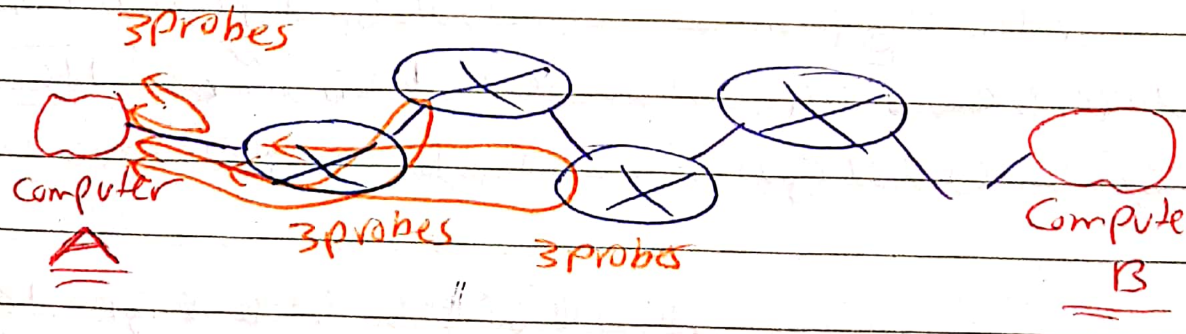
■ **traceroute program**: provides delay measurement from source to router along end-end Internet path towards destination.

for all i:

- sends **three packets** that will reach router i on path towards destination (with time to live filed value of i)

- router i will return packets to sender

- sender measures time interval between transmission and reply.



وقتی ما به عنوان ورودی یک آدرس Host ای را به برنامه بدهیم، هدفش اینه بیاد مسیر بین مبدا و مقصد را trace کنه و به ازای هر کدوم روترها که توی مسیر هستند مشخص کنه که چه مقدار تاخیر بین مبدا و روتر وجود داره.

برای اینکار میاد به ازای هر روتری که بین فرستنده و گیرنده هستند ۳تا packet ارسال می کنه

چرا ۳ تا؟ تا میانگین گیری کنه

این کار را به ازای تمام روترهایی که در مسیر هستند تکرار می کنند

time to live

*در header بسته هایی ttl داریم که یک مقدار اولیه به این ttl اختصاص داده می شود و بعد هر موقع این بسته به روتر می رسد اون روتر یکی از مقدار ttl کم می کند اگر مقدارش هم چنان بزرگ تر از صفر باشد طبق جدول forwarding می بیند که اون بسته باید کجا برسد و ارسالش می کند اما اگر ttl = 0 شده دیگر اون بسته را ارسال نمی کند و به جایش یک پیغام خطا برای مبدا می فرستد

در traceroute هم از همین مکانیزم استفاده می شود. وقتی می خواهد اولین روتر حوالیش را ببیند ← ttl = 1 می گذارد و یک بسته ارسال می کند

روتر اول ttl را کم می کند و دیگر ارسال نمی کند و یک پیغام خطا به مبدا می فرستد و برنامه traceroute این زمان رفت و برگشت را به عنوان تاخیر تا روتر اول حساب می کند

و این کار را ۳ بار تکرار می کند تا بتوانیم average بگیریم

بسته های بعدی (بسته ی ۴) برای پیدا کردن روتر دوم ←

ttl = 2 می گذاریم ← روتر اول یکی از ttl کم می کند ← ttl = 1 چون صفر نشده ارسالش می کند برای روتر بعدی ← روتر دوم یکی از ttl کم می کند ←

ttl = 0 ← دیگر بسته را به ادامه ی مسیر ارسال نمی کند و بسته را drop می کند و یک پیغام خطا برای مبدا ارسال می کند ← یک تخمین از زمان رفت و برگشت

traceroute این کارها را تکرار می کند تا بسته به مقصد برسد

traceroute از یک مکانیزم به جز ttl استفاده می کند تا مقصد را مجبور کند پیغام خطا بفرستد

مکانیزمی که این برنامه استفاده می کند برای اینکه مقصد Q را مجبور شود پیغام خطا به مبدا ارسال کند و تا یک زمانی برآورد کند چی هست؟

سوال

مکانیزم روترها ← الان ttl برای مقصد چی؟

traceroute uwaterloo.ca : توی محیط ترمینال

چرا بعضی از روترها ستاره اند ← یعنی جوابی دریافت نکردیم از اونها

① چون بعضی روترها کانفیگ نشده‌اند تا پیغام خطا را ارسال کنند و وقتی ttl صفر شد، بسته را drop می‌کنند و چیزی برای مقصد ارسال نمی‌کنند

اینکه روتری پیغام خطا بفرسته یا نه بستگی به کانفیگ روتر داره.

② یا روترها جواب دادند ولی جواب‌ها گم شده اند

جهش در delay به خاطر چیه؟ لینک‌های بین قاره‌ای trans-oceanic link → propagation delay

« Throughput » گذردهی

throughput : rate ($\frac{bits}{time\ unit}$) at which bits are being sent from sender to receiver

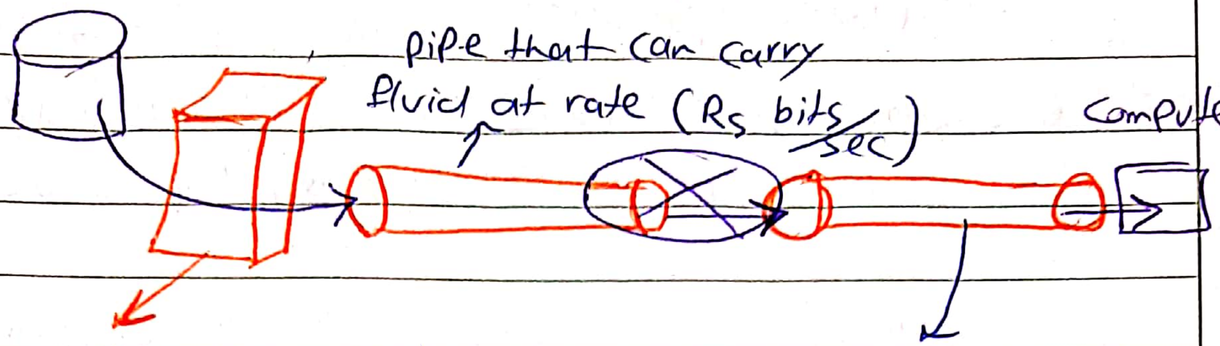
instantaneous : rate at given point in time (حالت آنی)

average : rate over longer period of time (حالت میانگین)

$$\text{Throughput} = \frac{\text{packet size (bit)}}{\text{delay (s)}}$$

→ طول بسته / تاخیر

Throughput calculation : Approximate , accurate



server sends bits (fluid) into pipe

pipe that can carry fluid at rate (R_c bits/sec)

Links as pipes

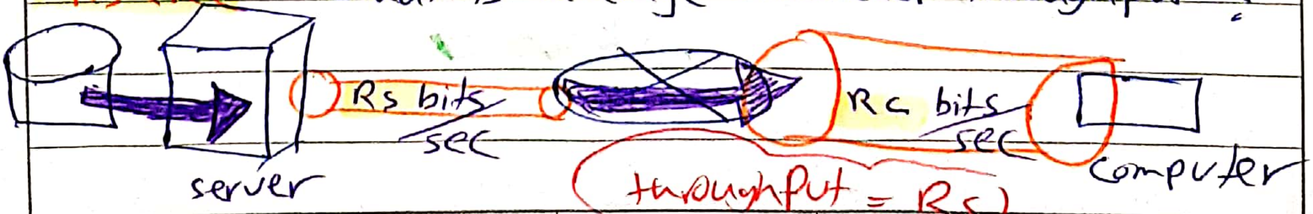
اگر روترها ← نقاط اتصال برای یک سری لوله درنظر بگیریم که این لوله‌ها جانشین لینک‌ها در شبکه شده‌اند و قطر لوله‌ها متناسب با ظرفیت آن‌هاست

(یا برعکس) اگر لوله‌ها رو الکی بزرگ‌تر کنیم (همه‌جوره هم بزرگ کنیم)

وقتی که قطر لوله‌ی سمت چپ (ابتدای مسیر کم است) ← فایده‌ای ندارد و ما به اندازه‌ی حداکثر ظرفیت لوله‌ی چپ می‌توانیم آب پمپ کنیم داخل این شبکه ←

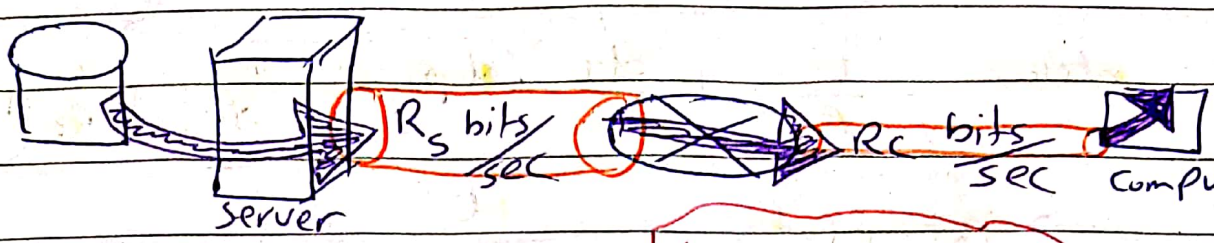
bottleneck (گلوگاه) ← شبکه — لینکی که ظرفیت کمتر دارد = throughput

$R_s < R_c$ what is average end-end throughput ?



throughput = R_s → Minimum

$R_s > R_c$ what is average end-end throughput?

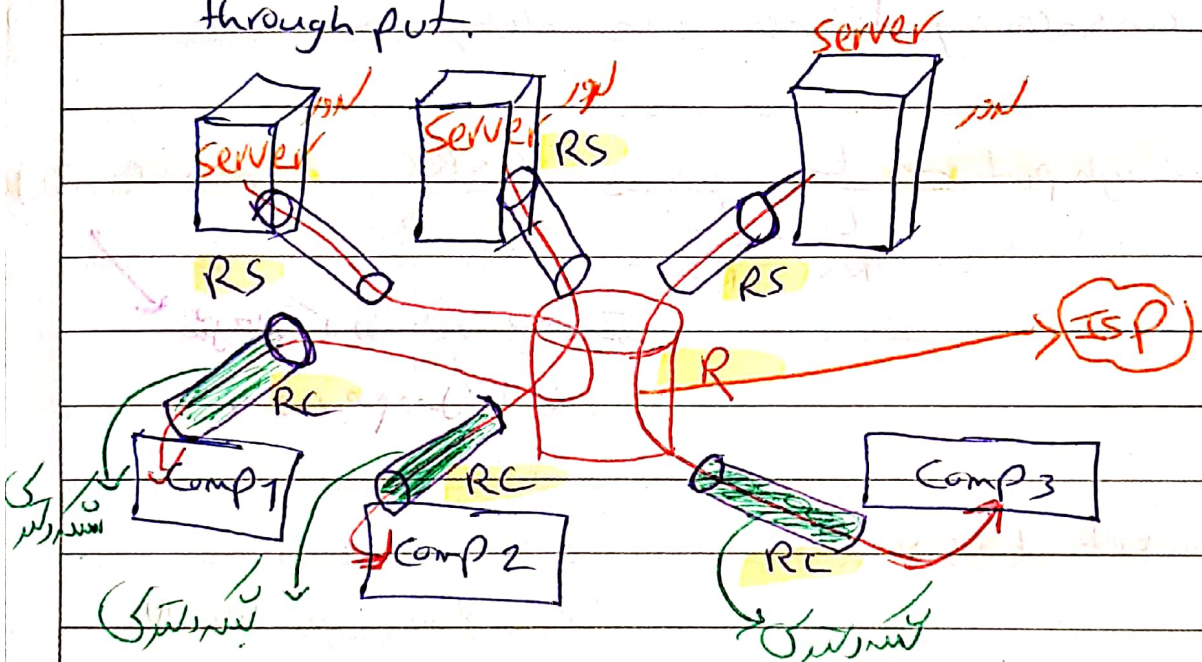


→ throughput = R_c

minimum

bottleneck link

↳ link on end-end path that constrains end-end throughput.



- per-connection end-end throughput:

$\min(R_c, R_s, R_s/10)$ → end-end throughput

- in practice: R_c or R_s is often bottleneck

10 connections (fairly) share backbone bottleneck link R bits/sec

زمان propagation در لینک دوم = زمان propagation در لینک اول

Throughput calculation & accurate

distance

$$delay = \frac{L}{R_s} + \frac{d_1}{s_1} + d_{proc} + d_{queue} + \frac{L}{R_c} + \frac{d_2}{s_2}$$

(سرعت انتشار سیگنال در محیط)

$$\rightarrow Throughput = \frac{L}{\frac{L}{R_s} + \frac{d_1}{s_1} + d_{proc} + d_{queue} + \frac{L}{R_c} + \frac{d_2}{s_2}}$$

$d_{propagation}$ ، d_{queue} ، d_{proc} ↑ صرف نظر از

just considering transmission delay ⇒

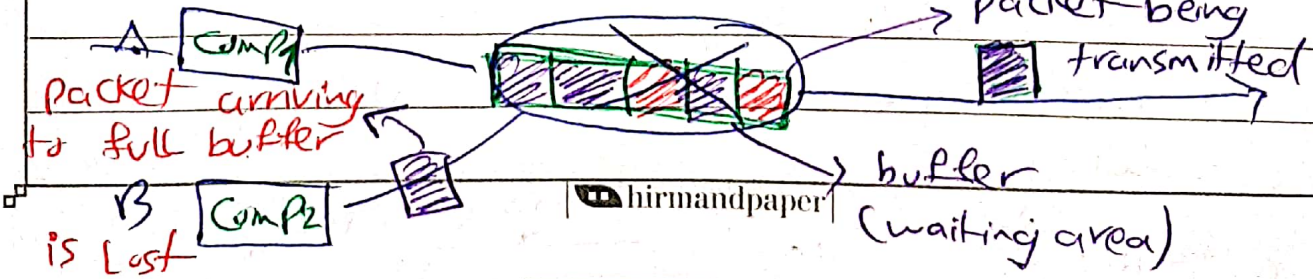
$$Throughput \simeq \frac{L}{\frac{L}{R_s} + \frac{L}{R_c}} = \frac{R_s R_c}{R_s + R_c} \simeq \min(R_s, R_c)$$

حتی حالتی که دو مقاومت به صورت موازی به هم وصل شده اند

« Packet Loss »

① ظرفیت بافر برای انتقال بسته‌ها

- queue (aka buffer) preceding link in buffer has finite capacity (queue delay) ② سرعت ارسال بسته‌ها
- packet arriving to full queue dropped (aka Lost) اگر سرعت ورودی از ظرفیت خروجی بالاتر باشد بسته‌ها گم می‌شوند (drop)
- Lost packet may be transmitted by previous node, by source end system, or not at all.



Session 6 — protocol "layers" and reference models.

Networks are complex with many "pieces":

- hosts
- routers
- links of various media
- applications
- protocols
- hardware, software

is there any hope of organizing structure of network!!!

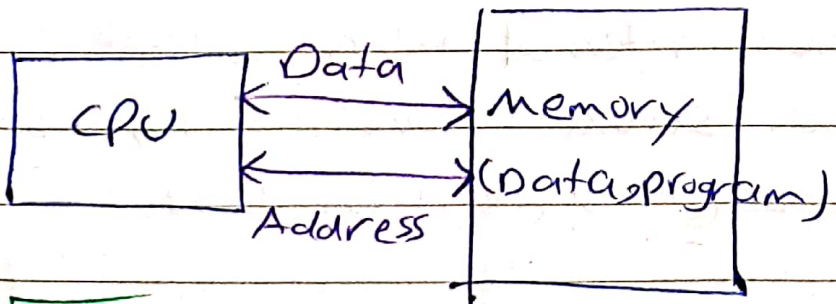
Layering is every where !!

protocol layering (stack)

we usually organize our tasks into sub_tasks and define the interfaces.

Ex1

- Task ⇐ Computation
- sub_tasks ⇐ Hardware design & software design
- Interfaces ⇐ ISA ⟵ زبان ماشین یا زبان



Ex2

Task ⇐ programming

sub_tasks ⇐ procedures

Interfaces ⇐ Arguments/return values.

ex3 Task: travel — مسافرت هوایی

sub-tasks: agency, check in, gates

Interfaces: passport, ticket, serial number

ticket (purchase)
baggage (check)
gates (load)
runway takeoff
airplane routing

Example: organization of air travel

فرودگاه مبدا		فرودگاه مقصد
ticket (purchase)	ticketing service	ticket (complain)
baggage (check)	baggage service	baggage (claim)
gates (load)	gate service	gates (unload)
runway takeoff (پرواز)	runway service	runway landing (فرود)
airplane routing	routing service	airplane routing
راهبری و هدایت مکانی		راهبری و هدایت مکانی

layers: each layer implements a service

- via its own internal-layer actions.
- relying on services provided by layer below

هر لایه ای فقط با لایه های بالا و پایینی خودش در ارتباط است

Why layering?! Approach to designing/discussing complex systems:

1. این لایه ها به ما اجازه ی فهم و شناخت بهتر سیستم را می دهد
2. نگهداری، آپدیت سیستم راحت تر می شود

☐ explicit structure allows identification, relationship of system's pieces.

- layered reference model for discussion

☐ modularization eases maintenance, updating of system

- change in layer's service implementation transparent to rest of system.
- change in gate procedure doesn't affect rest of system.

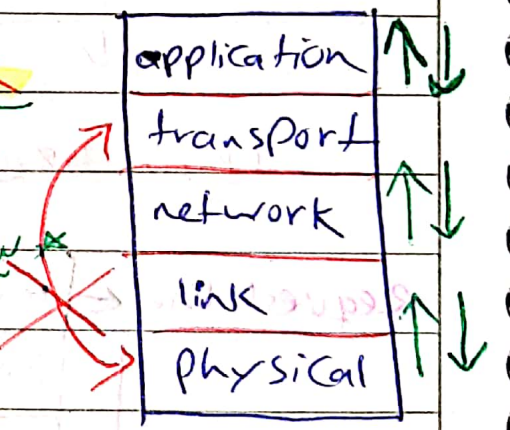
اگر توی قسمت پایانی مثلاً تغییری اتفاق بیفته نیاز نیست به قسمت های بالایی دست بزنیم

Layered Internet protocol stack

هر لایه یک وظیفه ای دارد

پروتوکل های هر لایه وقتی با دو تا لایه در ارتباط هستند

لایه ی بالا و پایین



Gets service from the below

Gives ~ to ~ above

Each layer does his jobs by adding its header to the packet. (Encapsulation)

Packet به Header (لایه ها) با اضافه کردن کارهای لایه

application : supporting network applications

• HTTP • IMAP • SMTP • DNS

HTTP Demo using (telnet) → صفحه بلک در سرور

در ترمینال ویندوز بزن :

* telnet www.example.com 80

enter بزن — روی پورت ۸۰

→ الان می‌تونم با وب سرور صحبت کنم

GET /index.html HTTP/1.1

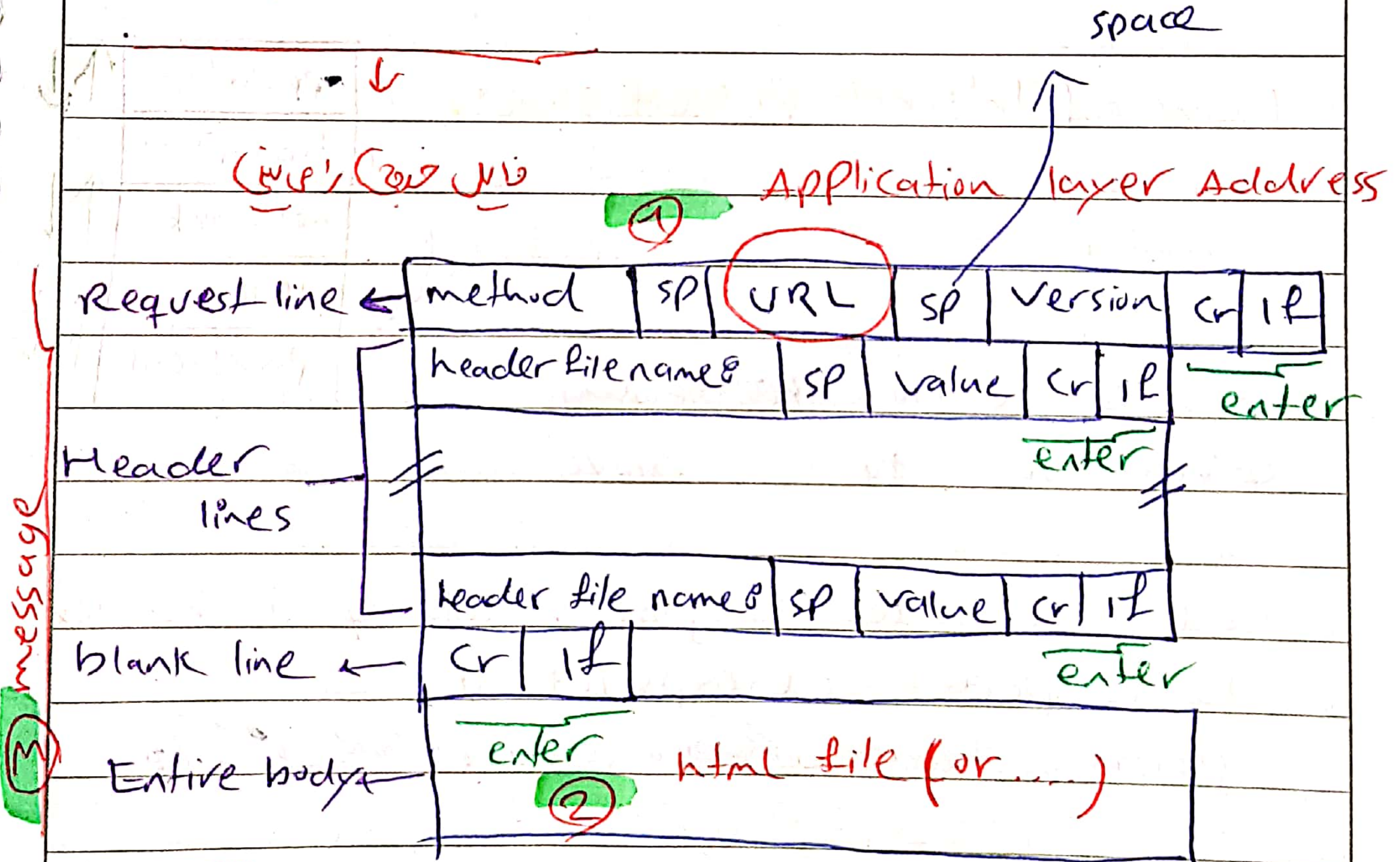
Host: www.example.com

enter بزن

enter بزن

فایل خروجی را ببین

① Application layer Address



④ only in end-systems. پروتکل‌هایی که متعلق به لایه application

به بسته‌ها یا پیام‌هایی که در لایه application هستند ← Message می‌گوییم

هستند ← فقط داخل end-system ها هستند.

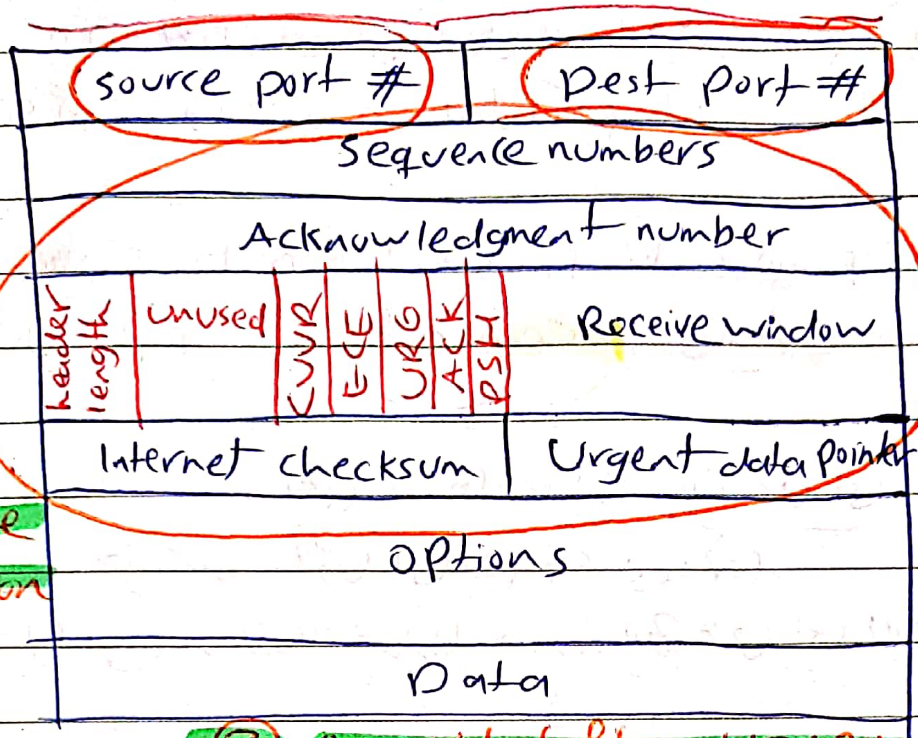
(5) only in end-system → در داخل روتر و سوئیچ نیست

transport: process-process data transfer

• TCP, UDP

32 bits — Transport layer Address

(1)



(2) making a reliable connection

ساختار داخلی header TCP که به message اضافه می‌شود

(4) message segment

(3) e.g: (html file + HTTP)

لایه‌ی application باید بسته‌هایی که از لایه‌ی transport دریافت می‌کند را بر اساس اعداد متفاوت در نظر بگیرد که به این اعداد port number می‌گویند و با قرار دادن این اعداد در header خودش به لایه‌ی transport طرف مقابل اطلاعات لازم برای اینکه بسته را منتقل کند را برساند

مثلاً وقتی یک بسته مربوط به web دریافت می‌کند از پورت 80 استفاده می‌کند.

multiplex کردن بسته‌هایی که از application های مختلف میاد سمت فرستنده توسط پورت number های مختلف

و سمت گیرنده لایه‌ی transport عمل demultiplex را انجام دهد

کار دیگر این لایه کانال ناامنی که توسط اینترنت بین فرستنده و گیرنده وجود دارد را به یک کانال امن تبدیل کند

TCP که در داخل end system گیرنده هاست به sequence number نگاه می‌کند و بسته‌ها را مرتب می‌کند

کاربرد 8. Acknowledge number: منتظر هستیم تا ack بسته‌ها را دریافت کنیم

← اگر بعد از مدتی این ack را دریافت نکنیم فرض را بر این می‌گذاریم که بسته گم شده و دوباره بسته را ارسال می‌کنیم

به بسته‌هایی که داخل transport layer شکل می‌گیرند با اضافه شدن header ـ پروتکل TCP ← (segment) می‌گوییم

only in end system ← یعنی روترها و سوییچ‌ها فیلدهای داخل این header را چک نکنند؛ برایشان این پیام‌ها بی مفهوم است

چرا؟ چون در روترها و سوییچ‌ها به صورت پیش‌فرض این لایه (transport) وجود ندارد.

(لایه‌ی شبکه یا لایه‌ی IP)

network: routing of datagrams from source to destination — وظیفه: بسته‌هایی که قرار است از مبدأ به مقصد برسند را ببرد

مثال: در داخل شبکه مسیرهایی کشف تا بسته‌ها به مقصد برسند ـ IP و routing protocols

32 bits

version	Header length	Type of service	Datagram length (bytes)	
16-bit Identifier			flags	13-bit Fragmentation offset
Time to live	upper-layer protocol		Header checksum	
32-bit Source IP address				
32-bit destination ~ ~				
options (if any)				
Data				

(فیلدها) این آدرس‌ها این است که مقصد و مبدأ را به صورت یکتا مشخص کنند

3) Datagram

1) Network layer address

2) Data

<html file + HTTP, TCP>

4) in end-system & routers — (IP) header ← اطلاعات داخلی ↑

در داخل پروتوکل IP ← فرمت header های که باید به segment های که به دلیل لایه transport اضافه شود، تعریف می‌شود و از طریق اطلاعاتی که در این header ها قرار می‌گیرد، روترها و end_system ها می‌توانند بفهمند که با بسته چکار کنند.

* به بسته‌هایی که شکل می‌گیرند در داخل Network layer از طریق اضافه کردن این header ها Datagram گفته می‌شود.

IP address ها به صورت یکتا هستند

link: data transfer between neighboring network elements

• Ethernet, • 802.11 (wifi) • PPP

فرمت header Ethernet ← شکل زیر

preamble	Dest. address	source address	Type	Data	....	CRC

① link layer Address

Trailers Error Checking ④

③ html, http, TCP, IP

⑤ in all nodes → آدرس‌های فیزیکی (MAC)

Frame ④

لایه‌ای که وظیفه‌ی انتقال بسته‌ها را بین هر کدام از نودها و همسایه‌اش دارد ← Link layer

چرا به این لایه می‌گویند Link؟ چون پروتوکل‌های که در این لایه هستند به لینکی که داریم برای ارتباط دو node استفاده می‌کنیم بستگی دارد.

مثلاً زوج سیم یا فیبر نوری ← Link layer = Ethernet

لینک رادیویی ← 802.11 (wifi)

لینک به لینک مقادیر آدرس مبدا و مقصد تغییر می کنند

(2) Link address (e.g. MAC) are different from IP address. IP address are fixed from source to destination. Link addresses change node to node (exception: switch)

آدرس های فیزیکی (MAC) های گیرنده (و فرستنده) لودر لینک را مشخص می کنند

→ مقادیرشون هم تغییر کند ← وقتی از یک node عبور می کند

مثل IP آدرس نیست که در کل مسیر مقدارش ثابت است

(3) Data = <html file, HTTP, TCP, IP> → لایه Network

encapsulate کردن لایه های بالایی — پیغام اصلی — لایه application — لایه Transport

(4) Trailer: نهایتا بعد از قسمت داده یک سری بیت اضافه می کنیم → که بهشون می گوییم CRC وظیفه: error checking — در Ethernet هست

(5) بسته هایی که در داخل لایه Link لینک می گیرند Frame می گوییم
لایه

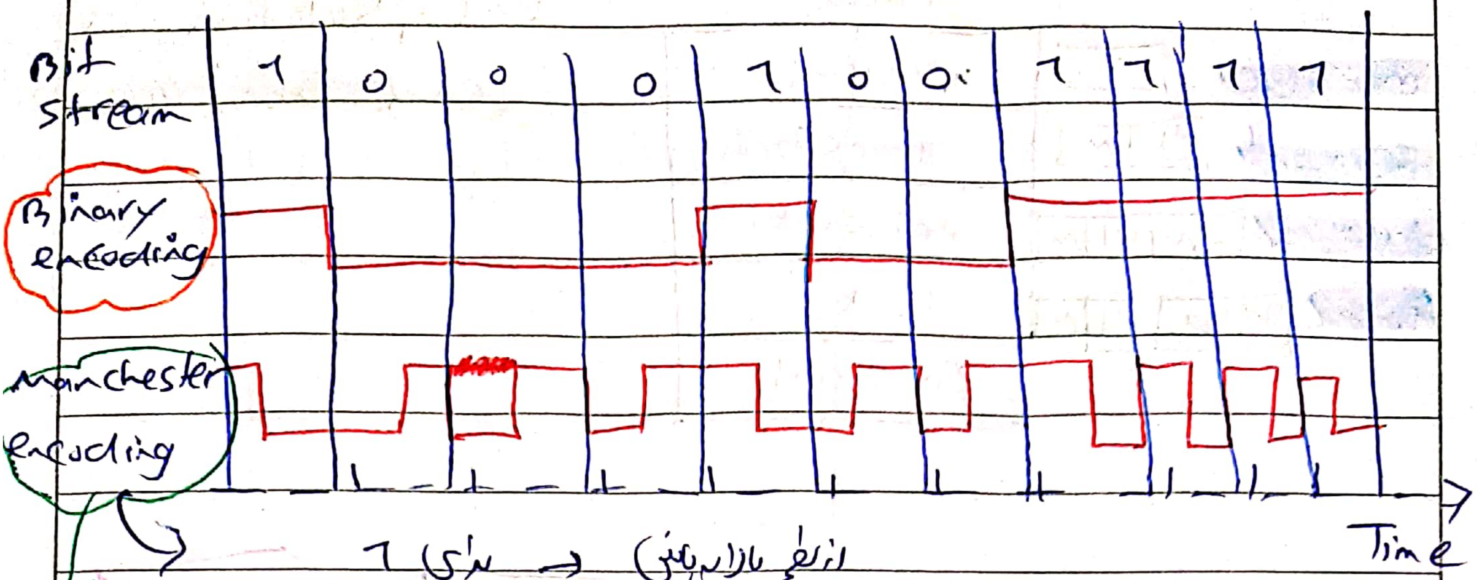
Frame → بسته هایی که header لینک دارند

(6) In all nodes (end-system, router, switches)
که چرا؟ چون همه این دیوایس ها باید بتوانند بسته را برای همسایه خود ارسال کنند.

Physical: bits "on the wire"

بیت ها را با شکل موج مناسب در داخل media ای که از دستگاه بعدی متصل هست را هدایت کند. (حداقل انتقال می دهد + انتقال شرط)

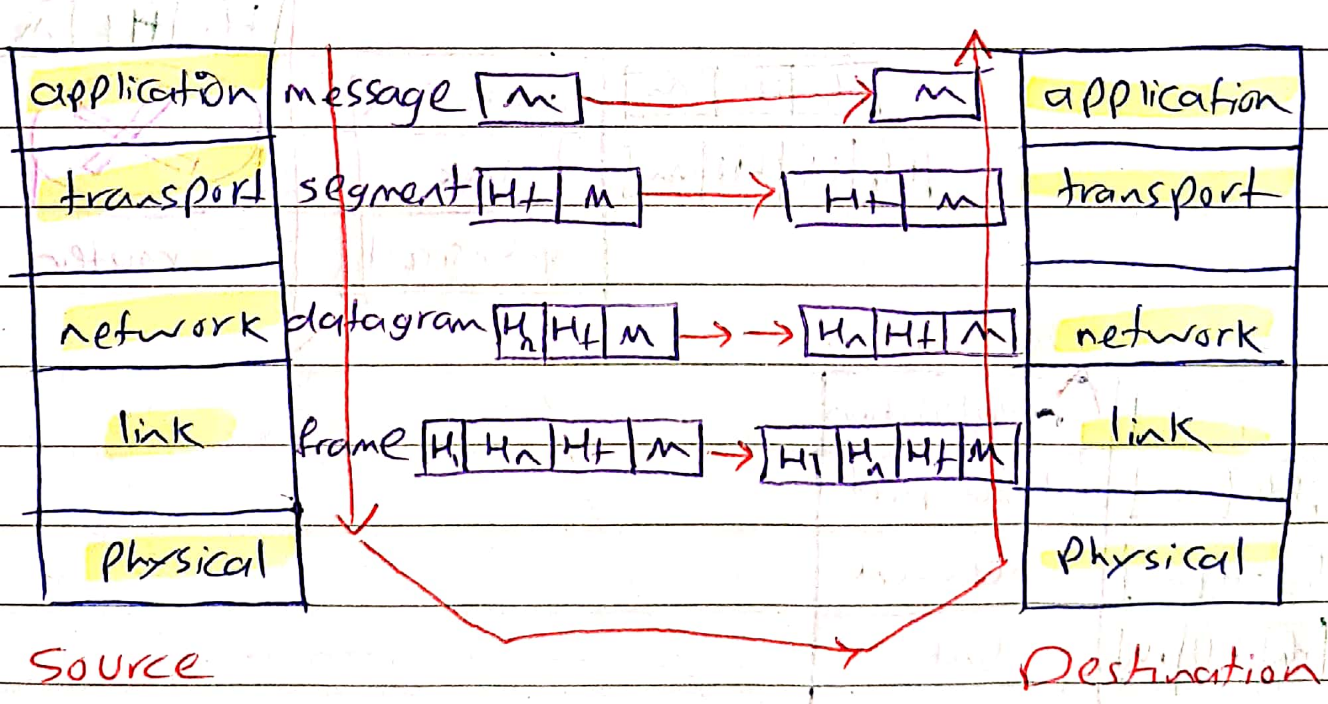
مثلا کابل از امواج الکترومغناطیسی استفاده کنی، یا موج فرکانس، با چه نوع ...



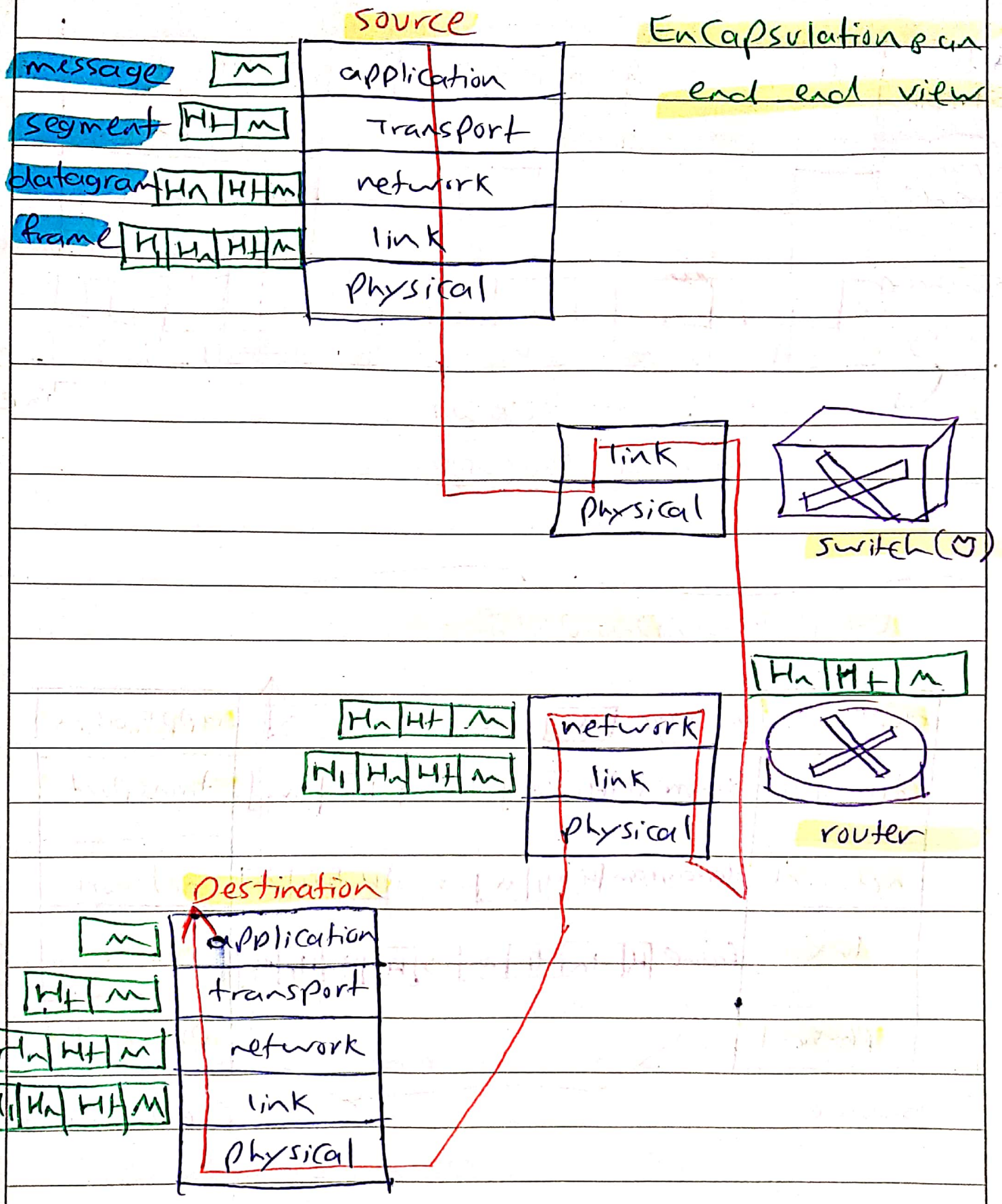
برای 1 → از لبه بالا به پایین

برای 0 → از لبه پایین به بالا

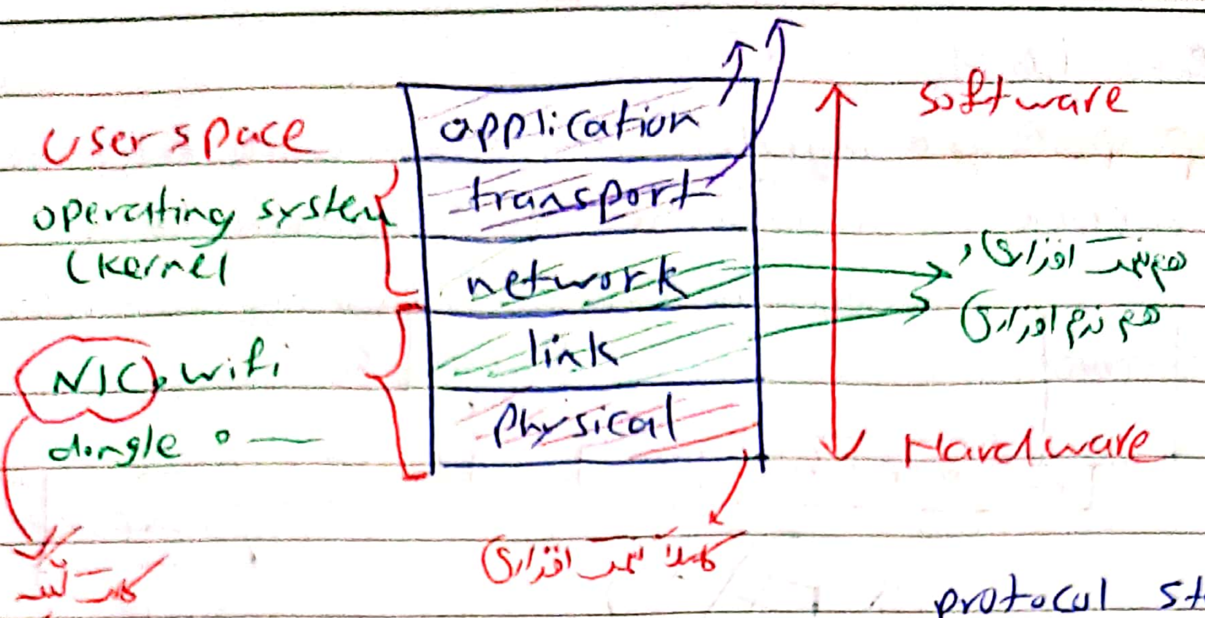
Encapsulation/Decapsulation



Encapsulation & un encapsulation end to end view



کاملاً نرم افزاری



protocol stack
شکل ادیه ها

network interface cart

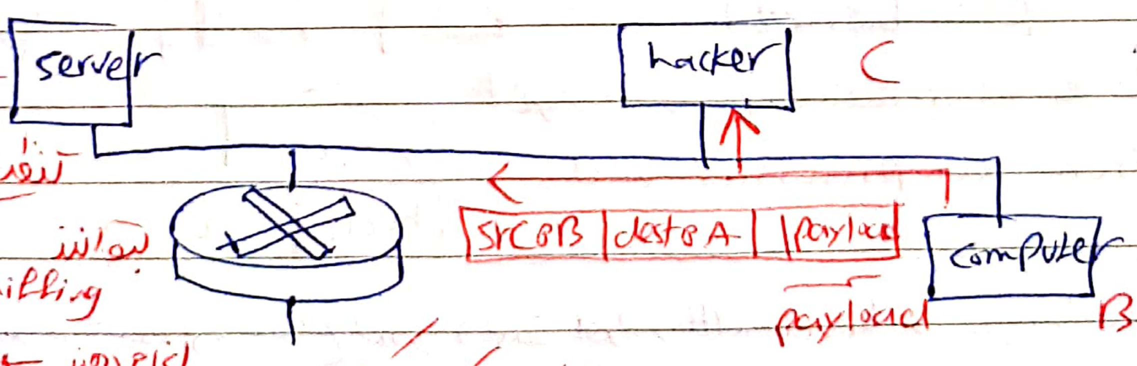
session 7

Network security

بو کشیدن

① packet "sniffing" : وسط راه ارسال بسته یک نفر بسته را دریافت می کند و اطلاعات را می بیند

- broadcast media (shared Ethernet, wireless) — در این
- promiscuous network interface reads/records all packets (including passwords!) — راحت است، حالتی که کارت شبکه روی می گذارد



اول A تقلب کردند تا packet sniffing بتوانند انجام دهند ← promiscuous

می توانند این کار را بکنند

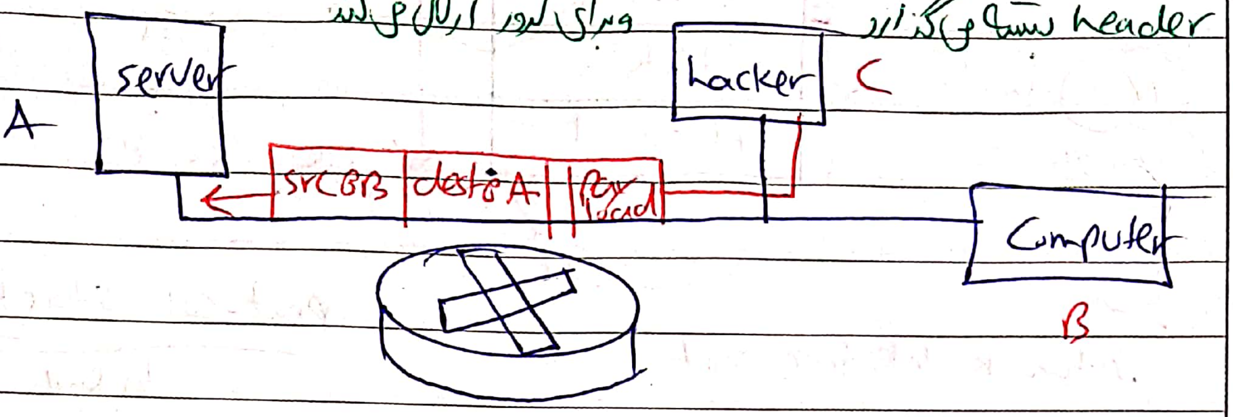
* wireshark : used for our end-of chapter labs is a (free) packet sniffer

جعل آیدی یا آیپی

Fake Identity

IP spoofing: injection of packet with false source address

هکر میاد آیپی جعلی را بجای آیپی فرستنده توی header بسته می‌گذارد و برای سرور ارسال می‌کند



Denial of service (DoS):

attackers make resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic

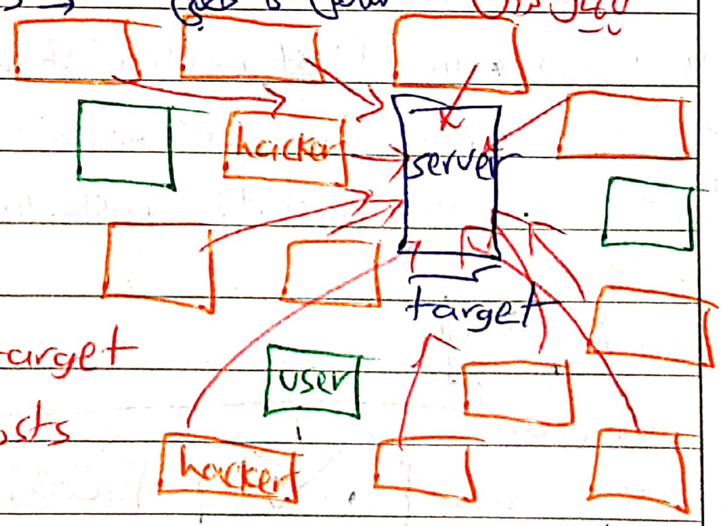
overwhelming: درهم شکستن، پایمال کردن

bogus → ساختگی و جعلی

① select target

② break into hosts around the network (see botnet)

③ send packets to target from compromised hosts



هکر با اختصاص منابع سرور (مثلاً bandwidth، نرم افزار و ...) را به ترافیک‌های جعلی اختصاص دهد ← بدین کاربرهای واقعی منبعی باقی نمی‌ماند تا به آنها سرویس داده شود

botnet: شبکه‌ای از host‌ها که هکر در آنها نفوذ کرده

«Creating a network app»

write programs that to
- run on (different) end systems
- communicate over network
- e.g., web server software communicates with browser software.

«no need to write software for network-core devices»

- network-core devices do not run user applications.
- applications on end systems allows for rapid app development, propagation.

how to write a network app?

1. choose an architecture (client/server or P2P)
2. Decide about the transport-layer service
3. use Socket API

client – server paradigm

servers:

① always – on host

② permanent IP address.

③ often in data centers, for scaling

① سرورها همیشه روشن هستند

② آدرس آی‌پی ثابت دارند (همیشگی)

③ تعداد زیادی data center داریم که مجموعه‌ای آنها یک سرور مجازی قدرتمند را می‌سازد

clients:

① contact, communicate with server

② may be intermittently connected

③ may have dynamic IP address

④ do not communicate directly with each other.

⑤ examples: HTTP, IMAP, FTP

① ارتباط با سرورها

② می‌توانند قطع و وصل شوند ارتباطشون با سرورها

③ آی‌پی آدرس آنها (client) dynamic است (هربار ممکن است فرق کند)

④ client ها مستقیم به هم وصل نمی‌شوند (از طریق سرورها وصل می‌شوند)

⑤ نمونه‌های از معماری client-server: HTTP, ...

Peer-Peer architecture

① no always-on server — سروری نداریم که همیشه روشن باشد

② arbitrary end systems directly communicate. — end system ها به صورت مستقیم با هم در ارتباط هستند

③ peers request service from other peers, provide service in return to other peers.

- self scalability – new peers bring new service capacity, as well as new service demands.

④ peers are intermittently connected and change IP address, security, incentives — حضوری باید

- complex management — پروتکل خودمان را طراحی کنیم تا

هرکدام از این peer ها در کنار سرویسی که از بقیه می‌گیرند مجبور شوند سرویس هم ارائه کنند

⑤ example: P2P file sharing

حالتی از این نوع روش

① سروری نداریم که همیشه روشن باشد

② end system ها به صورت مستقیم باهم در ارتباط هستند و

چون همه باهم در راه هستند (سلسله مراتبی وجود ندارد) به هرکدام از این device های end-system و peer گفته می‌شود

③ به همدیگر سرویس می‌دهند و از هم سرویس می‌گیرند — شبکه های که معماری peer-peer دارند self-scale باشند. یعنی با افزایش تعداد کاربران مشکلی پیش نیاید و اون سرویس ارائه شود. ← برخلاف روش client-server

یک مثال از این نوع شبکه ها P2P file sharing

Processes Communicating

process: program running within a host

- within same host, two processes communicate using inter-process communication (defined by OS)
  روش متصل کردن پروسس‌ها با هم

- processes in different hosts communicate by exchanging messages

* در معماری peer_peer اون برنامه‌ای که داره اجرا می‌شه هم سرور الان و هم client (یعنی اوقات سرویس می‌ده و بعضی اوقات تقاضای سرویس می‌کنه.*)

- note: applications with P2P architectures have client processes & server processes.

وقتی داریم برنامه‌ای که تحت شبکه می‌نویسیم به اون بخش از برنامه که قراره روی client‌ها اجرا بشه client می‌گوییم و به اون قسمت که قراره تحت سرور اجرا بشه server می‌گوییم

client process: process that initiates communication
شروع کننده‌ی ارتباط

server process: process that waits to be ~~connected~~ contacted
منتظره الان بقیه باهاش ارتباط برقرار کنند

* اپلیکیشن‌ها در قالب برنامه‌هایی هستند که وقتی در حالت اجرا در میان در end system‌های مختلف، به اینها process می‌گوییم.

هدف شبکه‌های کامپیوتری: پروسس‌هایی که در داخل end system‌های مختلف هستند بتوانند با هم ارتباط داشته باشند

process: برنامه‌ی در حال اجرا در end system

An application layer protocol defines:

- types of messages exchanged, e.g., request, response
- message syntax: فرمت پیام‌ها
  - what fields in messages & how fields are delineated.
- message semantics: معنی پیام‌ها
  - meaning of information in fields
- rules for when and how processes send & respond to messages. اعمالی که در حین دریافت پیام‌ها باید انجام دهند

تعریف protocol

- open protocols: defined in RFC, everyone has access to protocol definition. → open-source
  - allows for interoperability
  - e.g., HTTP, SMTP

پروتکل‌ها

- Proprietary protocols: اختصاصی
  - e.g., Skype, Zoom

elastic؟ منعطف

what transport service does an app need?

data integrity (سرویس امنیت داره)

- some apps (e.g., file transfer, web transactions) require 100% reliable data transfer.

حتی یک بیت هم نباید خطا داشته باشد

- other apps (audio, ...) can tolerate some loss

می‌توانند تا حدی گم شوند (یا خطا داشته باشند)

timing (تاخیر)

some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

نسبت به تاخیر حساس هستند و اگر میزان تاخیر از یک حدی بیشتر شود به توی کیفیت

throughput

سرویسی که کاربر از آن استفاده می‌کند تاثیر می‌گذارد.

- some apps (e.g., multimedia) require minimum amount of throughput to be "effective"

حداقل بیت ریت را لازم دارد

تا سرویس را به صورت موثر

- other apps ("elastic apps") make use of whatever throughput they get.

ارائه کنند

(می‌توانند منعطف باشند) منعطف (app ها) → مثل file transfer

security

encryption, data integrity, ...

رمز کردن اطلاعات

کسی اطلاعات ما را برندارد و به نام ما

به طرف مقابل برساند

③ TCP ها وقتی می‌بینند که میزان تاخیر لینک از یک حدی بیشتر شد به صورت داوطلبانه rate ـِ ارسال بسته را کم می‌کنند تا وضع لینک به حالت مطلوب برگردد.

Internet transport protocols services

TCP service:

① reliable transport between sending and receiving process

① ضمانت می‌دهیم که هر داده‌ای که ارسال می‌شود عیناً به طرف مقابل می‌رسد بدون رخ دادن خطا

② flow control: sender won't overwhelm receiver

② مثلاً فرستنده با سرعت خوبی می‌تواند بفرستد اما گیرنده محدودیت بافر دارد، TCP تعریف می‌کند

③ Congestion control: throttle sender when network overloaded

کنترل ازدحام

سرعت فرستنده را منطبق کند بر میزان حجم بافر گیرنده

③ کم کردن rate ـِ ارسال بسته‌ها می‌شود

④ Connection-oriented: setup required between client and server processes

لینک

اتصال گرا

⑤ does not provide: timing, minimum throughput guarantee, security

این سرویس‌ها را نمی‌دهد

UDP service:

unreliable data transfer between sending and receiving process.

does not provide: reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup.

نقطه ضعف سرویس‌های TCP: ① تاخیر زیاد دارد ② سربار زیاد (header بزرگتری) نسبت به UDP دارد

hirmandpaper

((اگر تاخیر برای یک app خیلی مهم باشد ترجیح می‌دهد از UDP استفاده کند))

بعضی از firewall ها اجازه‌ی به ترافیک‌های UDP رو نمی‌دهند

Securing TCP → نسخه‌ی اولیه‌ی TCP

Vanilla TCP & UDP sockets:
- no encryption
- cleartext passwords sent into socket traverse Internet in cleartext (!)

مکانیزم‌های برای برآورده کردن امنیت در TCP ← پروتکل TLS

Transport layer security (TLS)
- provides encrypted TCP connections.
- data integrity
- end-point authentication.

همان TCP است که یک سری چیزها مازاد بر TCP دارد (مثل امنیت)

TLS implemented in application layer.
- apps use TLS libraries, that use TCP in turn.

- cleartext sent into "socket" traverse Internet encrypted.

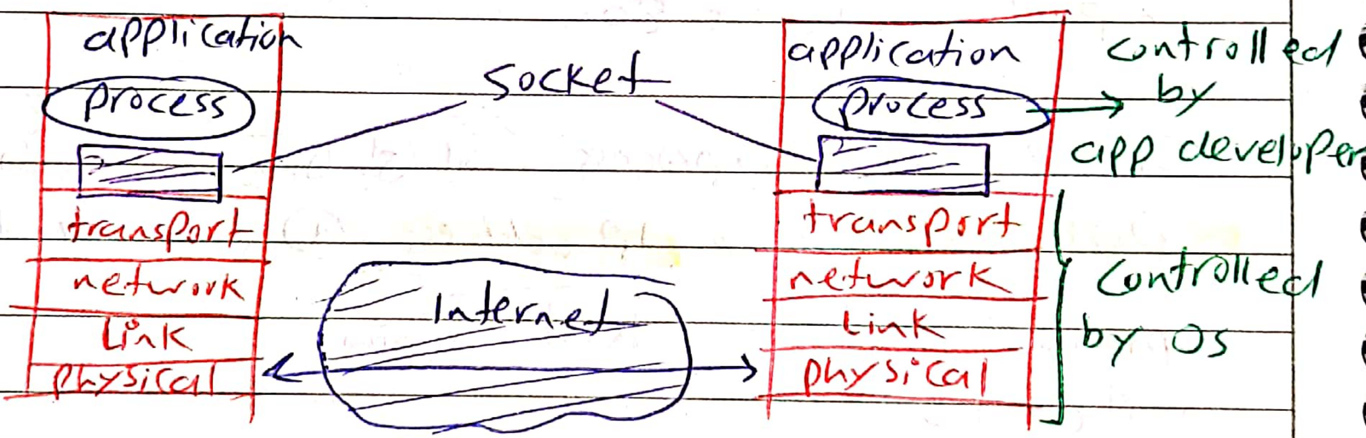
پیاده‌سازی مکانیزم‌های TLS در لایه‌ی application

ابتدا توسط فانکشن‌های TLS ، پیام امن می‌شود سپس پیام در اختیار socket TCP گذاشته می‌شود تا ارسال کند.

Sockets

- Process sends/receives messages to/from its socket
- socket analogous to door
  - Sending process shoves message out door
  - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process

  - two sockets involved: one on each side



socket یک interface ای است بین لایه‌ی application و transport
که توسط اون می‌توانیم پیام ها را از لایه‌ی app به لایه‌ی trans انتقال دهیم و یا برعکس
* مثل آرایه‌ها

~~Addressing~~ Addressing processes

- to receive messages, process must have identifier
- host device has unique 32-bit IP address
- Q: does IP address of host on which process runs suffice for identifying the process?

A: no, many processes can be running on same host.

- identifier includes both IP address and port numbers associated with process on host

- example port numbers:
  - HTTP server: 80
  - mail server: 25

- to send HTTP message to gaia.cs.umass.edu web server:

  (HTTP → 80) (gaia.cs.umass.edu: domain) (→ از طریق DNS)

  - IP address: 128.119.245.12
  - port number: 80

در شبکه‌های کامپیوتری برای اینکه process به‌صورت یکتا مشخص شود، دو تا identifier داریم ① IP address و ② Port numbers

IP address: مشخص کردن host

Port numbers: مشخص کردن process در داخل اون host

port number: دو بایت است ← از صفر تا ۶۵ هزار (تقریباً)

server های معروف، application های معروف، به‌شون یک‌سری port number تخصیص داده‌اند تا client ها ضمن اینکه IP آدرس آنها را می‌دانند، port number آنها را هم بلد باشند.

لایه transport، با توجه به این port number ها، بسته‌ها را ارسال می‌کند.

TCP/UDP Socket

ارسال امن

TCP & Application viewpoint

TCP provides reliable, in-order byte-stream transfer ("pipe") between client and server processes

مثل لوله

UDP & Application viewpoint ← تاخیر کم است

UDP provides unreliable transfer of groups of bytes ("datagrams") between client and server processes.

session 8

web

web changed the world!!

- until early 1990's internet for experts
- soon after Berners-Lee introduced web in 1989 & Internet for public
- on-demand
- everyone can become a publisher at extremly low cost
- videos, images, ... simulate our scens.

xdg-open file.html ... لینوکس

Web and HTTP

- web page consists of objects, each of which can be stored on different web servers
- object can be HTML file, JPEG image, Java applet, audio file, ...
- web page consists of base HTML-file which includes several referenced objects, each addressable by a URL e.g.,

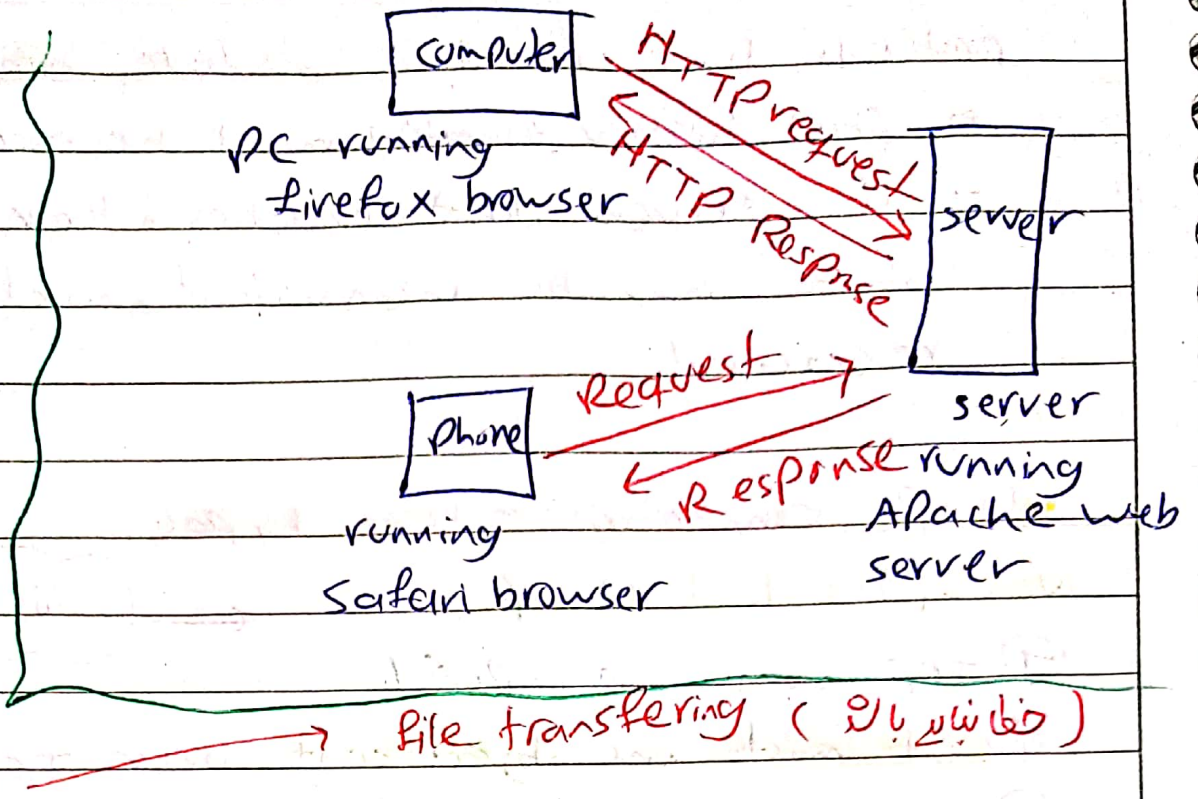
www.someschool.edu/somedept/pic.gif

host name | path name

HTTP overview

HTTP: hypertext transfer protocol

- web's application-layer protocol
- client/server model:
  - client: browser that requests, receives, (using HTTP protocol) and "displays" web projects
  - server: web server sends (using HTTP protocol) objects in response to requests



HTTP uses TCP

- client initiates TCP connection (creates socket) to server, port 80.
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and web server (HTTP server)
- TCP connection closed.

HTTP is "stateless" ← بدون حالة

- server maintains no information about past client requests

protocols that maintain "state" are complex.

- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled.

HTTP connections: two types

Non-persistent HTTP ← (ناپایدار)

1. TCP connection opened
2. at most one object sent over TCP connection
3. TCP connection closed.

برای دریافت هر object یک TCP connection ایجاد کنیم و بعد از دریافت اون object، اون connection بسته میشود.

↓

به تعداد object های که اون page دارد باید connection TCP ایجاد کنیم.

persistent HTTP: ← (پایدار)

1. TCP connection opened to a server
2. multiple objects can be sent over single TCP connection between client, and that server.
3. TCP connection closed.

یک ارتباط TCP ایجاد کنیم و بعد متن base html را دریافت کردیم و متوجه شدیم چند object دیگر لازم داریم تا کل این web page را داشته باشیم، در قالب همان ارتباط TCP، بقیه‌ی object ها را دریافت می‌کنیم و سپس ارتباط بسته می‌شود.

Non persistent HTTP (example)

User enters URL: www.someschool.edu/someDept/home.index (containing text, references to 10 jpeg images)

Comp	Server
1a. HTTP client initiates TCP connection to HTTP server (process) at www.someschool.edu on port 80	1b. HTTP server at host www.someschool.edu waiting for TCP connection at port 80 "accepts" connection, notifying client
2. HTTP client sends HTTP request message (containing URL) into TCP connection socket.	3. HTTP server receives request message, forms response message containing requested object and sends message into its socket.
5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects.	4. HTTP server closes TCP connection.
6. steps 1-5 repeated for each of 10 jpeg objects	

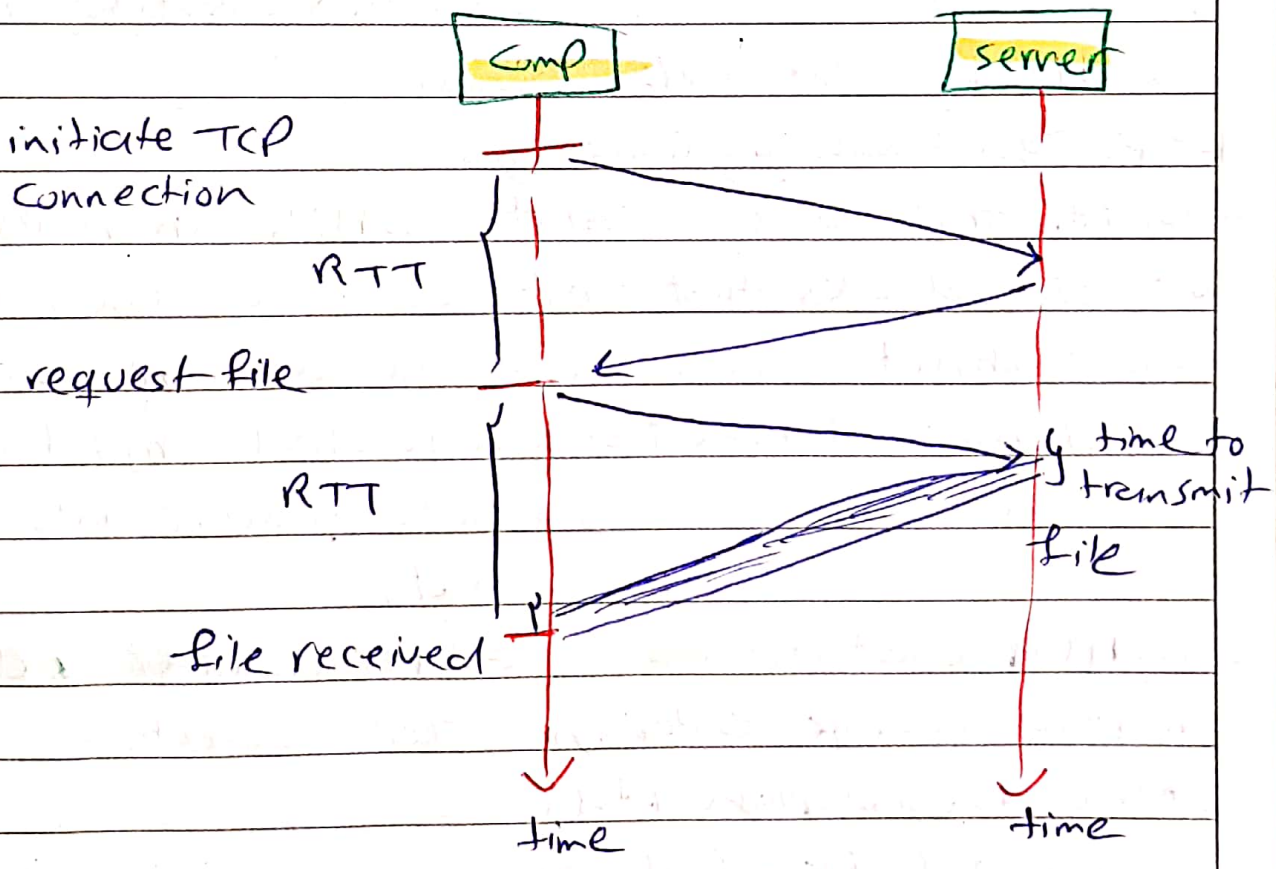
Non-persistent HTTP & response time

RTT (definition): time for a small packet to travel from client to server and back.

زمانی که طول می‌کشد تا یک بسته کوچک مسیر بین client و server را برود و برگردد

HTTP response time (per object):

- one RTT to initiate TCP connection
- ~ ~ for HTTP request and first few bytes of HTTP response to return.
- object / file transmission time



Non-persistent HTTP (response time) =
2RTT + file transmission time

زمان ارسال object

persistent HTTP (HTTP. 1.1)

non-persistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for each TCP connection
- browsers often open multiple parallel TCP connections to fetch referenced objects in parallel.

persistent HTTP:

- server leaves connection open after sending response.
- subsequent HTTP messages between same client/server sent over open connection.
- client sends requests as soon as it encounters a referenced object.
- as little as one RTT for all the referenced objects (cutting response time in half)

non-persistent: $\text{delay} = 2 \times RTT + \sum_{i=0}^{10} \frac{L_i}{\text{Throughput}}$

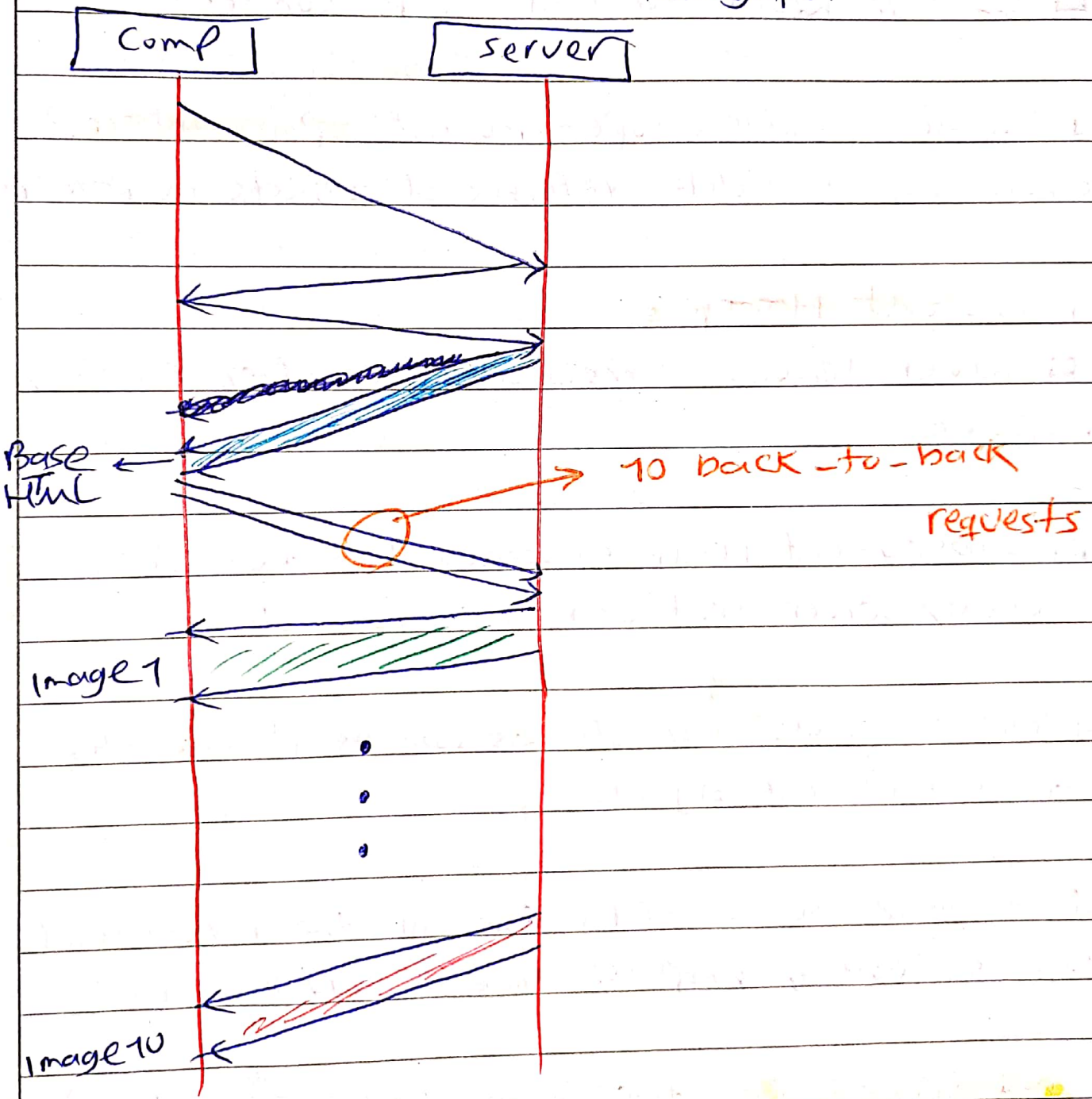
parallel: $\text{delay} = 4 \times RTT + \frac{L_0}{\text{Throughput}} + \max_{i=1,\ldots,10} \frac{L_i}{\text{Throughput}}$

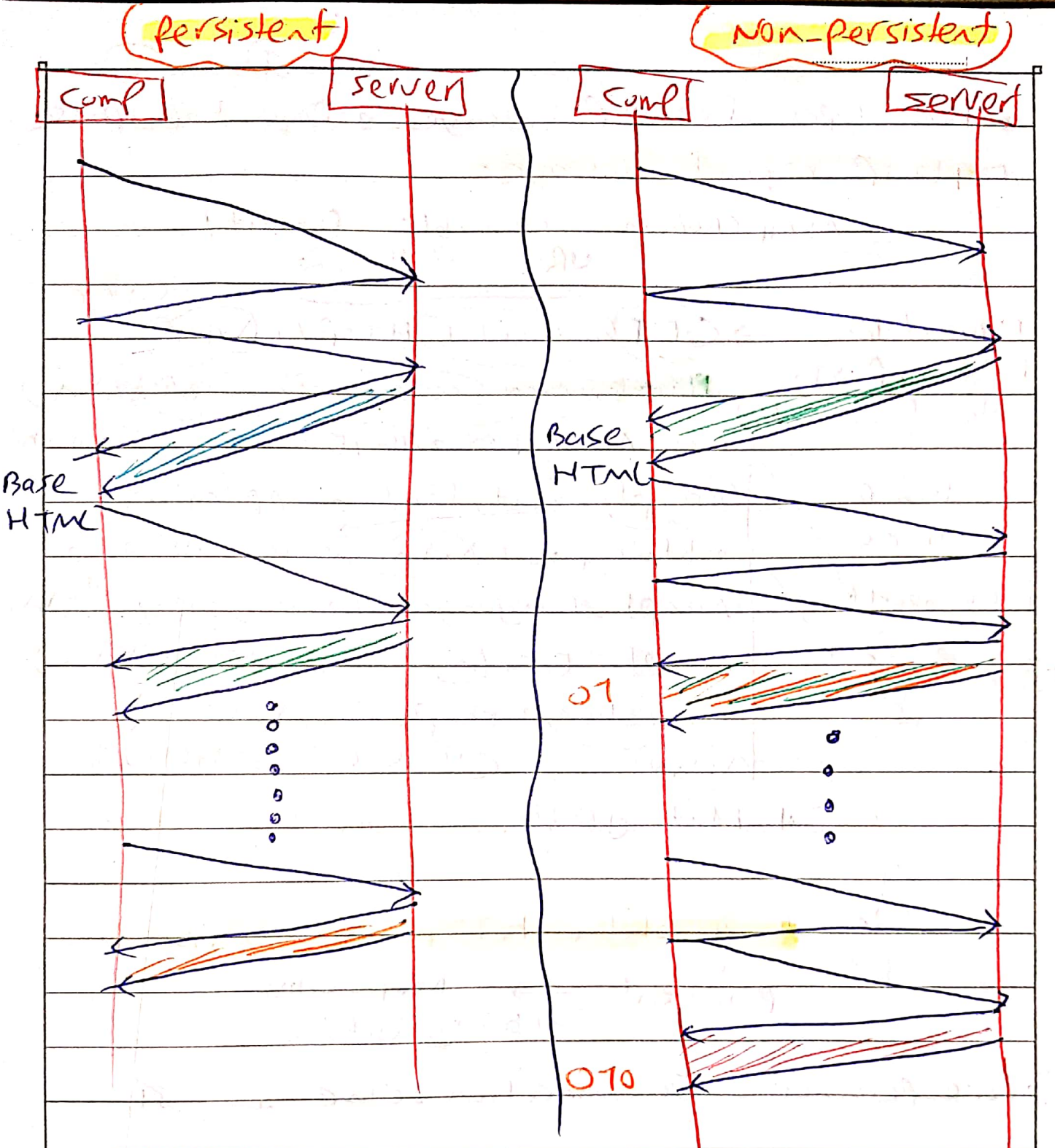
Q: when do we have sum instead of max?

persistento delay $= 12 \times RTT + \sum_{i=0}^{10} \frac{L_i}{\text{Throughput}}$

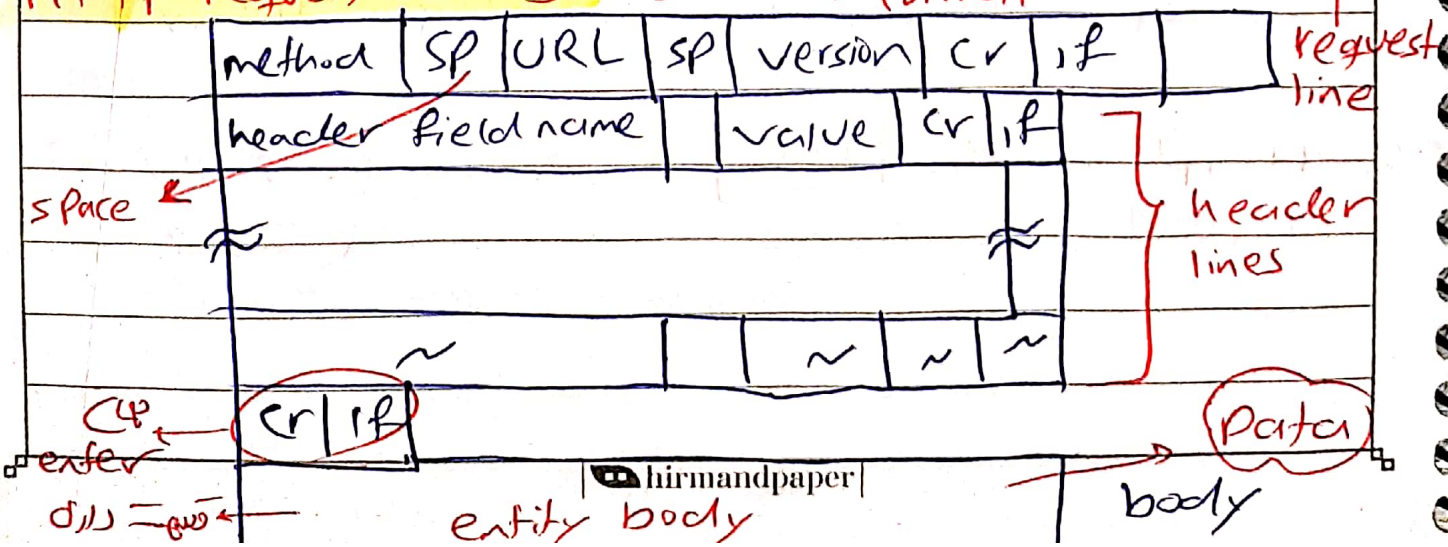
persistent _ pipeline

delay $= 3 \times RTT + \sum_{i=0}^{10} \frac{L_i}{\text{Throughput}}$





HTTP request message & general format



① ..... ②

- two types of HTTP messages: Request, Response

HTTP request message:

- ASCII (human readable format)

فرمت

URL = path مسیر

enter

request line (GET, POST, Head commands) → GET /index.html HTTP/1.1\r\n

header lines:

Host: www.net.cs.umass.edu\r\n — hostname

User_Agent: Mozilla/5.0

Accept: text/html, application/xhtml+xml\r\n — فرمت های که مرورگر می تواند قبول کند

Accept_language: en-US, en; q=0.5\r\n

Accept_Encoding: gzip, deflate\r\n — فرمت های قابل قبول برای encode کردن

Connection: keep_alive\r\n

\r\n

این خط باید باشد

نوع مرورگر را مشخص می کند

حداقل header line ← خط شامل Host است

پیش فرض در HTTP/1.1 ارتباطات پایدار است

persistent → keep_alive

متناظر با این نوع ارتباط است

اگر ← connection: close باشد ← none persistent

دو خط اول پیام بالا اجباری هستند (بقیه خط ها optional هستند)

سوال ؟ دلیل اینکه همچنان Host را داریم در HTTP request چی هست؟

other HTTP request messages

POST method :

- web page often includes form input
- user input sent from client to server in entity body of HTTP POST request message

GET method : (for sending data to server)

- include user data in URL field of HTTP GET request message (following a '?'):

www.somesite.com/animalsearch?monkeys&bena

Head (HEAD method):

- requests headers (only) that would be returned if specified URL were requested with an HTTP GET method.

PUT method :

- uploads new file (object) to server
- completely replaces file that exists at specified URL with content in entity body of PUT HTTP request message.

HTTP response message

status line (protocol status code status phrase)

header lines

```
← HTTP/1.1 200 OK
Date : Tue, 08 sep 2020
server : Apache/2.4.6 (centos)
last_modified : Tue, 01 mar 2016
ETag : "a5b-52d~
Accept_Ranges : bytes
content_length : 2657
Content_Type : text/html; charset=
                                UTF-8
\r\n
data data data data data
```

data, e.g., requested HTML file

HTTP response status codes

- status code appears in 1st line in server-to-client response message

200 OK

- request succeeded, requested object later in this message.

301 Moved permanently

- requested object moved, new location specified later in this message (in Location: field)

400 Bad Request
- request msg not understood by server

404 Not found
- requested document not found on this server

505 HTTP version Not supported.

Trying out HTTP (client side) for yourself
① netcat to your favorite web server
netcat
%nc -c -v example.com 80 (مثال)
- opens TCP connection to port 80 (default HTTP server Port) at example.com
- anything typed in will be sent to port 80 at example.com

② type in a GET HTTP request
GET /kurse_ross/index.php HTTP/1.1
Host: example.com

③ look at response message sent by HTTP server (or use wireshark to look at captured HTTP request/response)

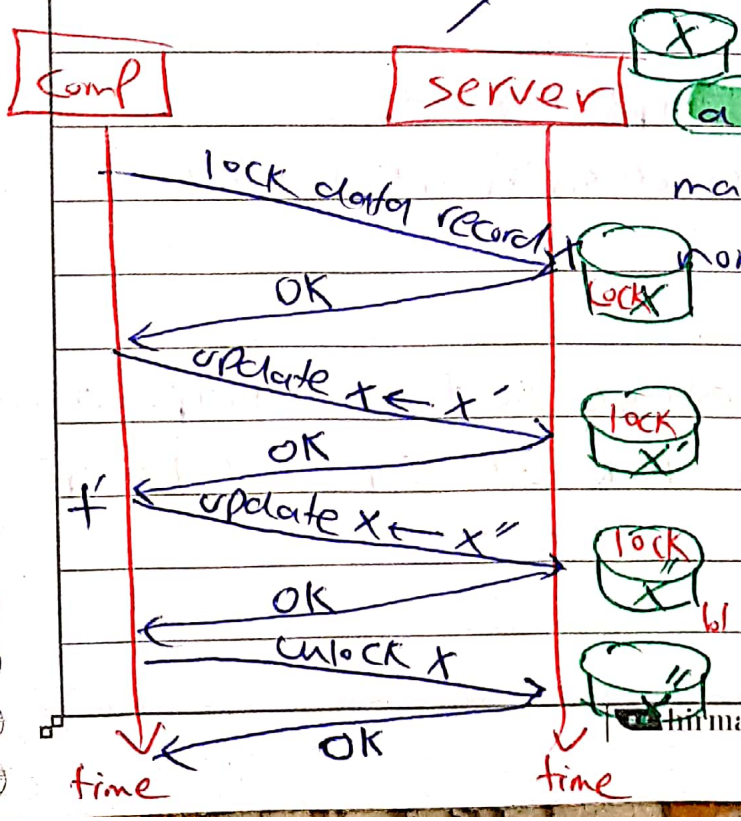
Session 9

Maintaining user/server state: cookies

HTTP GET/response interaction is **stateless**

- no notion of multi-step exchanges of HTTP messages to complete a web "transaction"
  - no need for client/server to track "state" of multi-step exchange
  - all HTTP requests are independent of each other.
  - no need for client/server to "recover" from a partially-completed-but-never-completed transaction.

a stateful protocol: client makes two changes to X, or none at all



what happens if network connection or client crashes at t'?

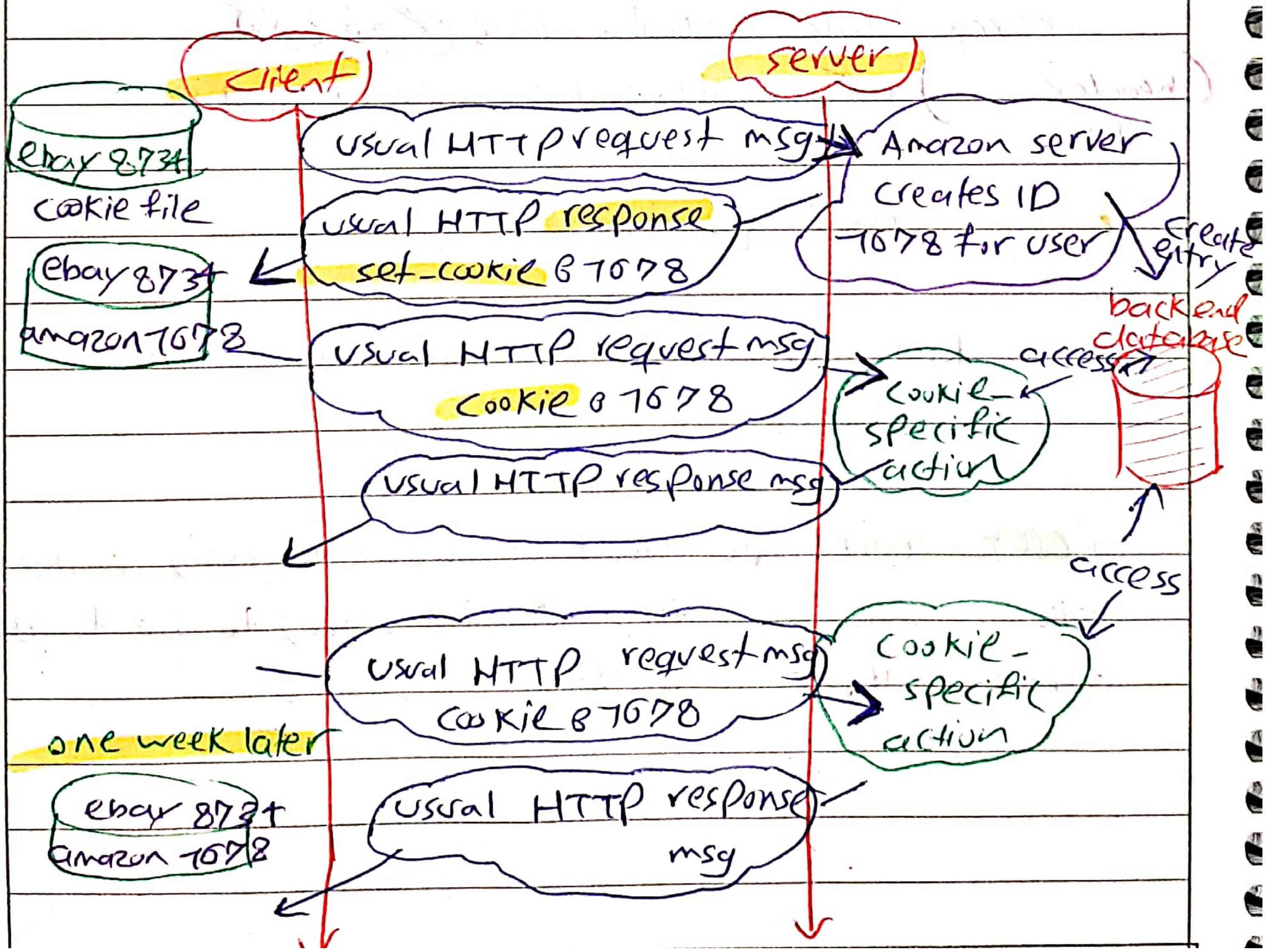
تشالاکی و رصد کارهایی که کاربر انجام می‌دهند   track* کردن فعالیت کاربر برای

در بعضی از app ها الزامی اند   ارائه‌ی سرویس *

web sites and client browser use cookies to maintain some state between transactions.

four (4) components:

1. cookie header line of HTTP response message
2. cookie ~ ~ in next HTTP request ~
3. cookie file kept on user's host, managed by user's browser
4. back end database at web site



what cookies can be used for?

- authorization
- shopping carts
- recommendations
- user session state (web e-mail)

از زمانی که به server وصل می‌شویم تا انتهای آن

حضوری state را ذخیره کنیم تا بتوانیم تاریخچه کارها را رصد کنیم.

Challenge: How to keep state?

- at protocol endpoints: maintain state at sender/receiver over multiple transactions

- in messages: cookies in HTTP messages carry state. client پیچیدگی کمتری دارد. کم کردن

(header line) از جهت افزایش در cookie ها استفاده می‌کنیم

cookies and privacy

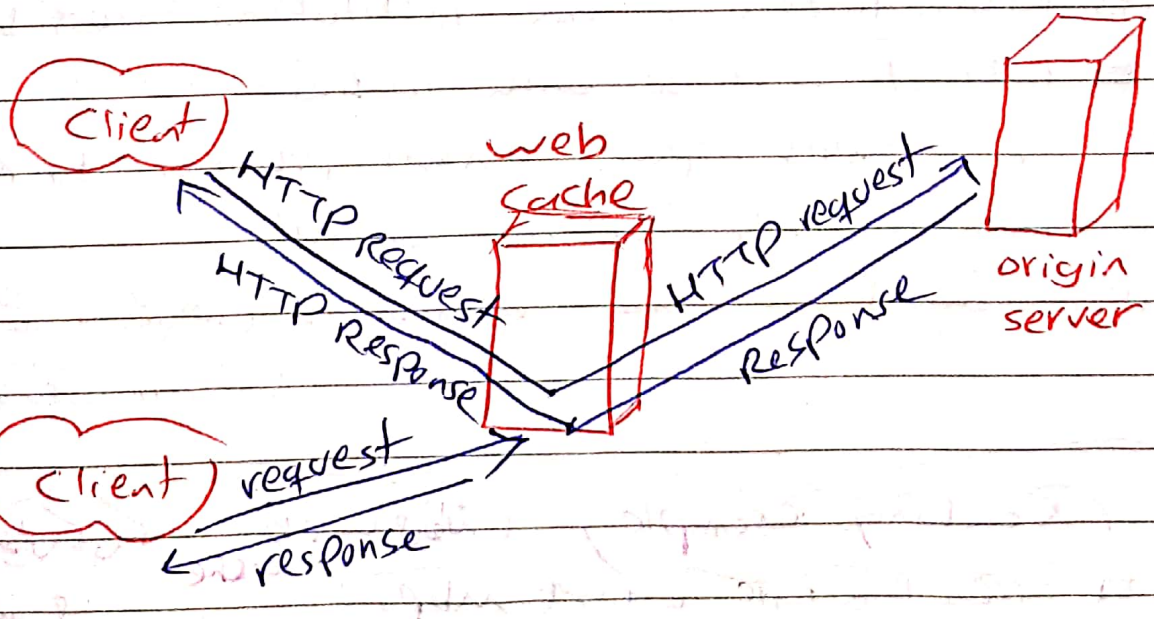
- cookies permit sites to learn a lot about you on their site.

- third party persistent cookies (tracking cookies) allow common identity (cookie value) to be tracked across multiple web sites.

Web caches

Goal: satisfy client requests without involving origin server.

- user configures browser to point to a (local) web cache
- browser sends all HTTP requests to cache
  - if object in cache: cache returns object to client
  - else: cache requests object from origin server, caches received object, then returns object to client.



web caches (aka proxy servers)

client → web cache → server

- web cache acts as both client and server
  - server for original requesting client
  - client to original server

server tells cache about object's allowable caching in response headers

cache_control: max-age = <seconds> → مدت زمان اعتبار دارد

cache_control: no_cache → قابل cache نباشد

why web caching?

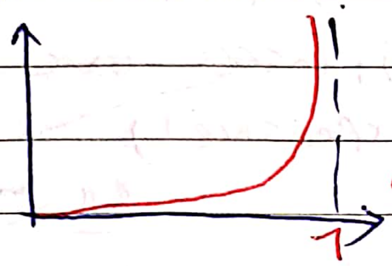
- reduce response time for client request
  - cache is closer to client
- reduce traffic on an institution's access link.
- Internet is dense with caches
  - enables "poor" content providers to more effectively deliver content.

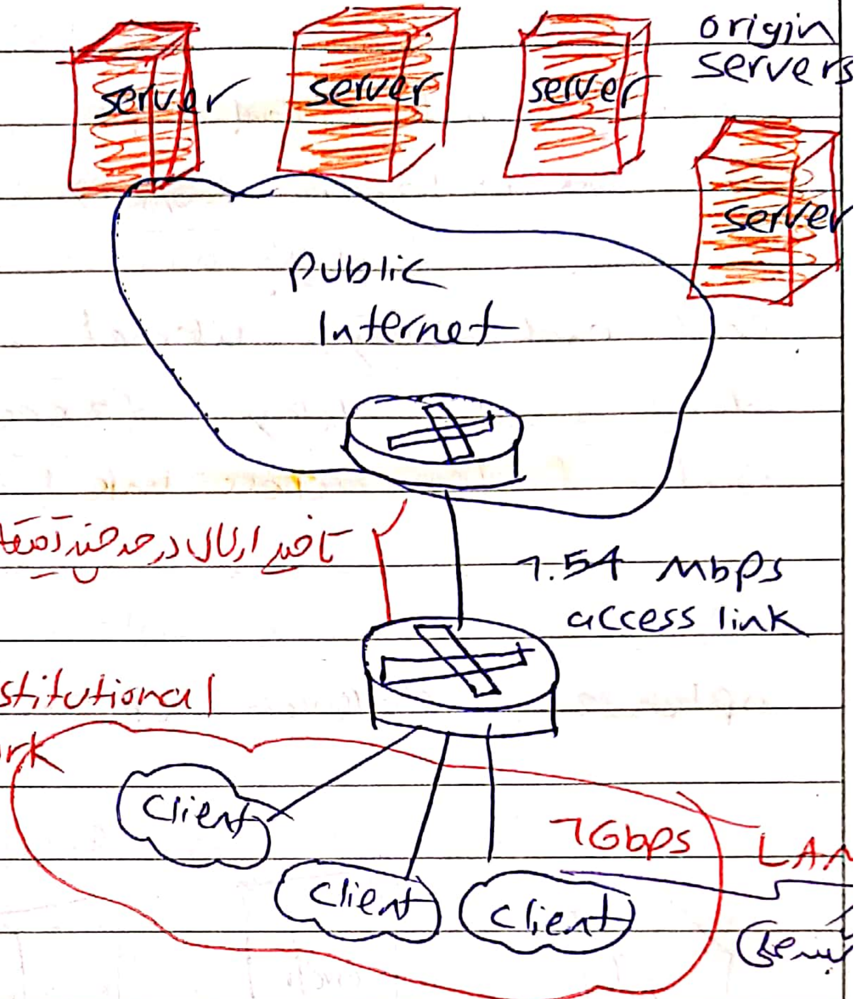
Caching example

- access link rate: 1.54 Mbps
- RTT from Internet router to server: 2sec
- web object size: 100k bits
- average request rate from browsers to origin servers: 15/sec

Ex: avg data rate to browsers: 1.5 Mbps

problem: large queueing delays at high utilization!





performance:

- access link utilization = 0.97 $\left(\frac{1.5}{1.54} \simeq 0.97\right)$
- LAN utilization: 0.0015
- end-end delay = Internet delay + access link delay + LAN delay = 2 sec + minutes + usecs (میکروثانیه)

زمانی که یک request از روتر به public internet می‌فرستند
به server ها حدود [2sec] است
و پاسخ server به روتر می‌آید

خواسته‌ی سوال: محاسبه‌ی تاخیر end-end
** تاخیرهای زیاد در لینک ← در حد چند دقیقه

option 1 : buy a faster access link (راه حل اول)

access link rate : 154 Mbps تغییرات در پارامتر

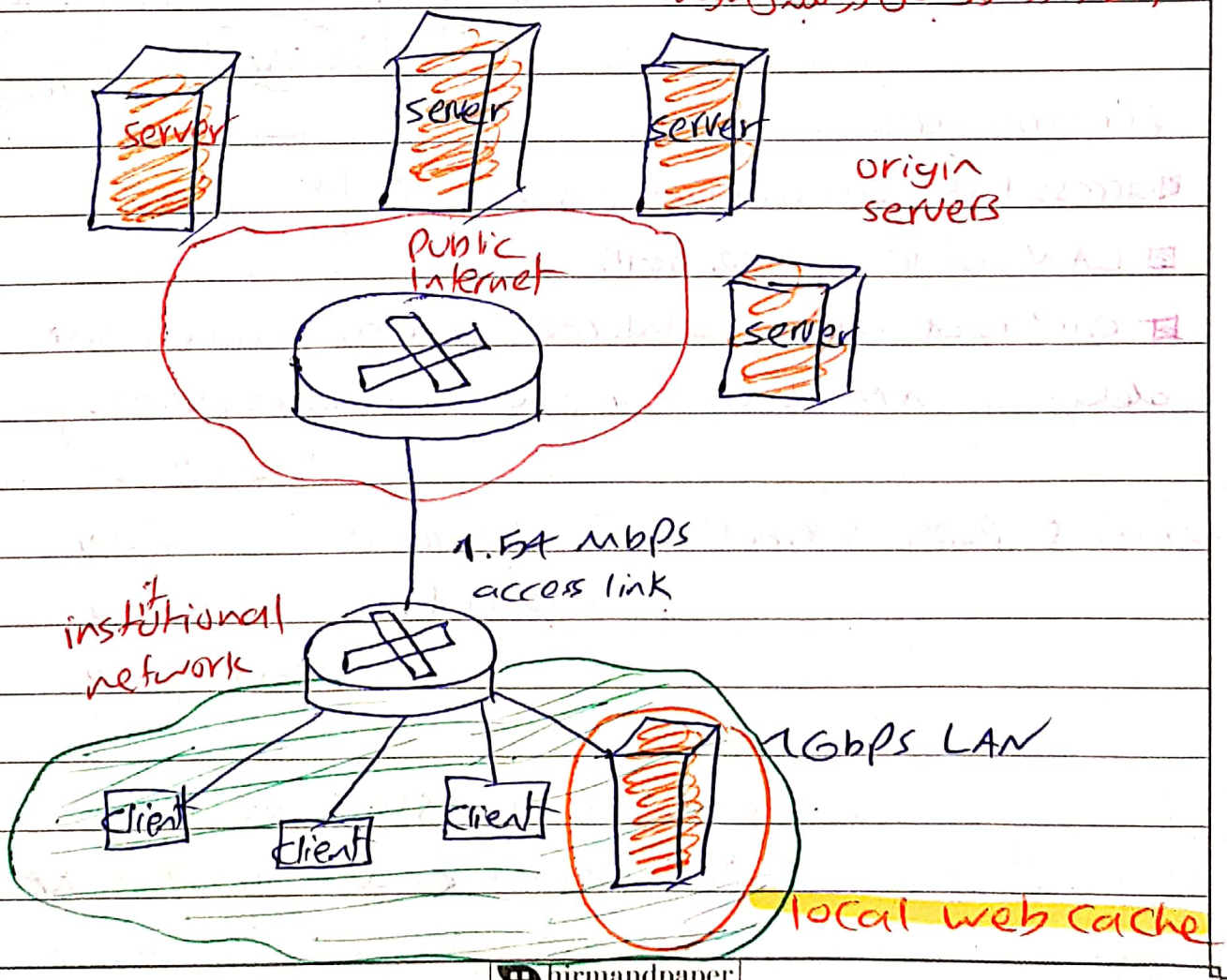
access link utilization : 0/0097 : performance ~

end_end delay = internet delay + access link delay + LAN delay = 2sec + msecs + usecs

cost : faster access link (expensive!)

option 2 : install a web cache (راه حل دوم)

استفاده از حافظه کش در شبکه موبایل



calculating access link utilization, end-end delay with cache:

suppose cache hit rate is 0.4

40% requests served by cache, with low delay (msecs)

60% requests satisfied at origin

rate to browsers over access link

$= 0.6 * 1.50 \text{ Mbps} = 0.9 \text{ Mbps}$

access link utilization $= 0.9/1.54 = 0.58$

low (msec) queueing delay at access link

average end-end delay:

$= 0.6 * (\text{delay from origin servers}) + 0.4 * (\text{delay when satisfied at cache})$

$= 0.6 (2.07) + 0.4 (\sim \text{msecs}) = \sim 1.2 \text{ secs}$

lower average end-end delay than with 154 Mbps link and cheaper too

URL به همراه دو طولک

چون زمانی miss می‌شود و نشونه قسمتی الـ تا اصلا وجود ندارد به اون request باید به سرورهای اصلی ارسال بشود و اون سرورها جواب http request را بدهند.

0/2 < hit rate < 0/7

وقتی می‌گوییم hit rate = 0/2 یعنی 20% تعدادها web cache از http requestها جواب داده می‌شود

* استفاده از web cache ← ↑ ارزان‌تر ، delay ↓

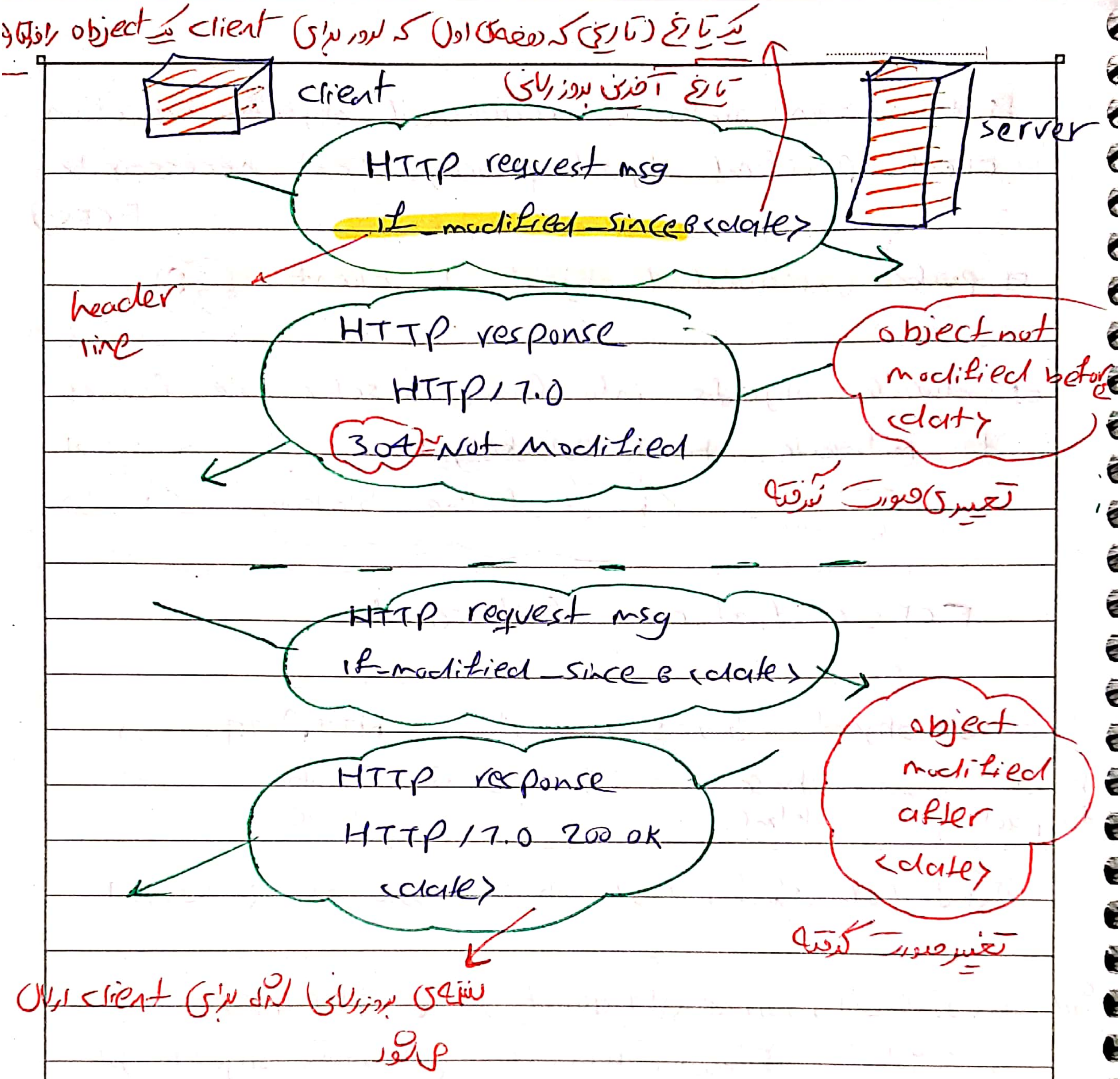
Conditional GET

Goal: don't send object if cache has up-to-date cached version.

- no object transmission delay (or use of network resources)

- client: specify date of cached copy in HTTP request.
  if-modified-since: <date>

- server: response contains no object if cached ~~object~~ copy is up-to-date:
  HTTP/1.0 304 Not modified



HTTP/2

key goal: decreased delay in multi-object HTTP requests.

HTTP/2: [RFC 7540, 2015] increased flexibility at server in sending objects to client:

- methods, status codes, most header fields unchanged from HTTP 1.1

- transmission order of requested objects based on client specified object priority (not necessarily FCFS)

- push unrequested objects to client (۴)

- divide objects into frames, schedule frames to mitigate HOL blocking. → ابتدای خط و بلاک شده

head of line blocking — تکه کردن

FCFS : first come first send

در HTTP/1.1 ، کروم web page هستند و بعد از دریافت base html request های مربوط به object های مختلف را back to back به صورت pack ارسال میکنیم و بعد سرور object ها را به ترتیب رسیدن های ارسال می‌کند.

در HTTP/1.1 ، هر request ای که زودتر دریافت شود، اول object متناظر با آن request ارسال می‌شود.

★ در HTTP/2 سرور می‌تواند اولویت های دیگری در ارسال object ها داشته باشد (نه لزوماً بر اساس زودتر اومدن)

★ سرور می‌تواند بعضی از object ها را به client ارسال کند با مکانیزم push حتی قبل از آنکه request آنها بیاد

★ سرور بجای آنکه منتظر باشد تا client تقاضای دریافت object ها را ارسال کند، خودش با فاصله پس از ارسال base html، خودش object ها را ارسال کند. به این کار می‌گویند (push) ← باعث تاخیر ↓

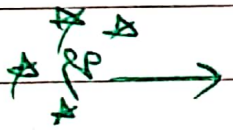
HoL Blocking یعنی چی؟! یعنی یک object بزرگی ابتدای خط قرار گرفته و تا ارسال آن تمام نشود، بقیه‌ی object ها ارسال نمی‌شوند

↓

در HTTP/2 برای رفع این موضوع، object ها را به frame ها تقسیم می‌کنند و به صورت یکی در میان از frame های object های مختلف ارسال می‌کنند.

↓

object های خیلی کوچک و سریع دریافت می‌شوند و معمولا object های بزرگ می‌مانند.



در HTTP/2 زمان دریافت object های کوچک تر می‌شود نسبت به HTTP/1

mitigating HOL blocking

session 10

Domain Name System
DNS

people: many identifiers:
- SSN, name, passport

Internet hosts, routers:
- IP address (32 bit) - used for addressing datagrams
- "name", e.g., cs.umass.edu - used by humans

Q: how to map between IP address and name, and vice versa?

DNS: Domain Name system:

- distributed database implemented in hierarchy of many name servers.

- application-layer protocol:
  hosts, DNS servers communicate to resolve names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

DNS: services, structure

DNS services:

1. hostname-to-IP address translation
2. host aliasing
   - one host, multiple names
     - (canonical, alias names)
     - relay1.west-coast.enterprise.com
     - www.enterprise.com or enterprise.com
   - one name, multiple host
     - mail server aliasing
       - yahoo.com
     - replicated web servers

replicated / تکرار کردن

③ load distribution for replicated servers

تکرار شده

توزیع تقسیم بار DNS IP آدرس‌ها توسط http request ها پخش می‌شود ← DNS

incrediby important Internet function
. Internet couldn't function without it

Q : why not centralize DNS ?

سوال : چرا نیایم از یک سرور واحد و در یک نقطه برای DNS استفاده کنیم ؟

① single point of failure

اگر برای این سرور اتفاقی بیفتد کل کارکرد DNS مختل می‌شود ← اینترنت دچار مشکل می‌شود

② traffic volume

↑ میزان ترافیک (برای یک سرور)

③ distant centralized database

فاصله‌ی یک سرور تا نقاط مختلف متفاوت ← ↑ فاصله از سرور ← ↑ زمان

④ Maintenance (a huge , frequently updated database)

↓ ↑ تاخیر

نگهداری از دیتابیس خیلی سخت می‌شود

ما با یک دیتابیس بزرگ با تغییرات زیاد روبرو هستیم

↓

مدیریت سخت است

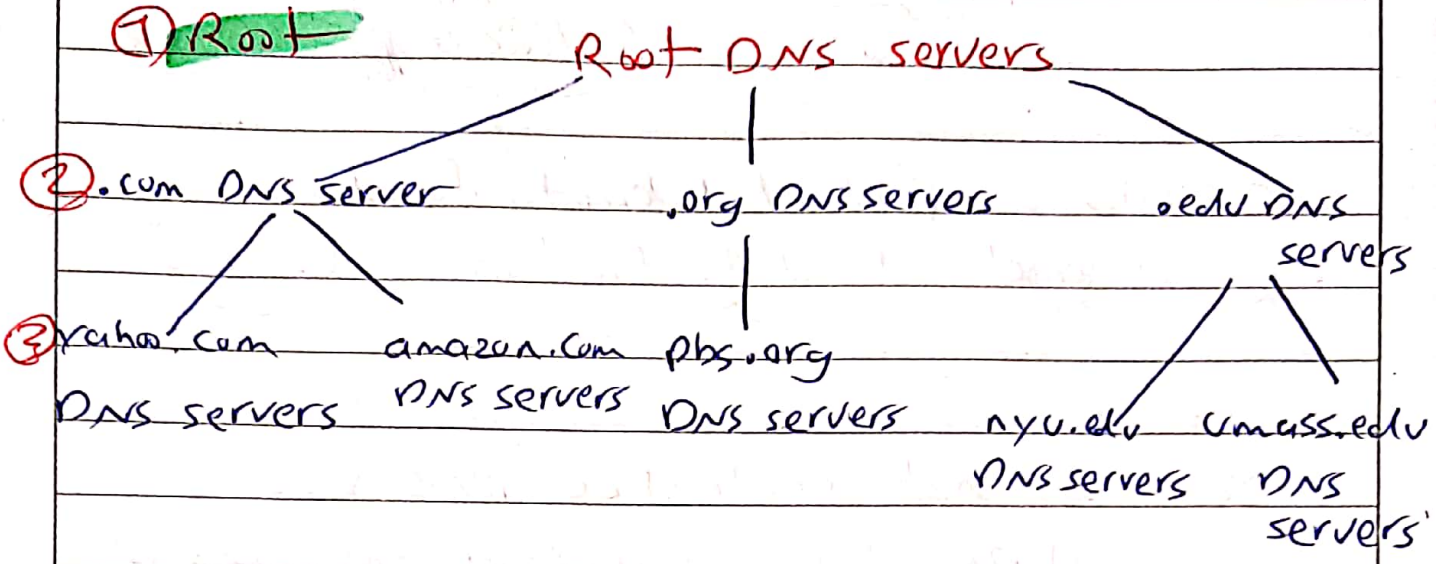
A : doesn't scale !!

* قابلیت مقیاس پذیری ندارد *

② → Top Level Domain ③ Authoritative

DNS: a distributed, hierarchical database

① Root



Ex: client wants IP address for www.amazon.com

① client queries root server to find .com DNS server

② client queries .com DNS server to get amazon.com DNS server

③ client queries amazon.com Dns server to get IP address for www.amazon.com

root name server

- official, contact-of-last-resort by name servers that can not resolve name
- managed by 12 organizations
- coordinated by ICANN
- visit root-servers.org

Top-Level Domain (TLD) servers:

- responsible for .com, .org, .net, .edu, .aero, .jobs, .museums and all top-level country domains: .cn, .uk, .fr, .ca, .jp, ---

سرورهایی که TLDها را مدیریت می‌کنند

- Network Solutions: authoritative registry for .com, .net TLD (این شرکت مالک .net و .com های TLD است)

یک شرکت است

- educause: edu TLD (این شرکت TLD edu را مدیریت می‌کند.)

authoritative DNS servers:

سرورهایی که اطلاعات نام‌های دامنه

- organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts. (قابل دارند)

یا خودشون دارند

- can be maintained by organization or service provider (این سرورها، یا توسط خود سازمان مدیریت می‌شوند.)

منظور ISP است

local DNS servers:

- when host makes DNS query, it is sent to its local DNS server.

* (الزامی نیست) وحوصله‌اش خیلی اختیاری الکه

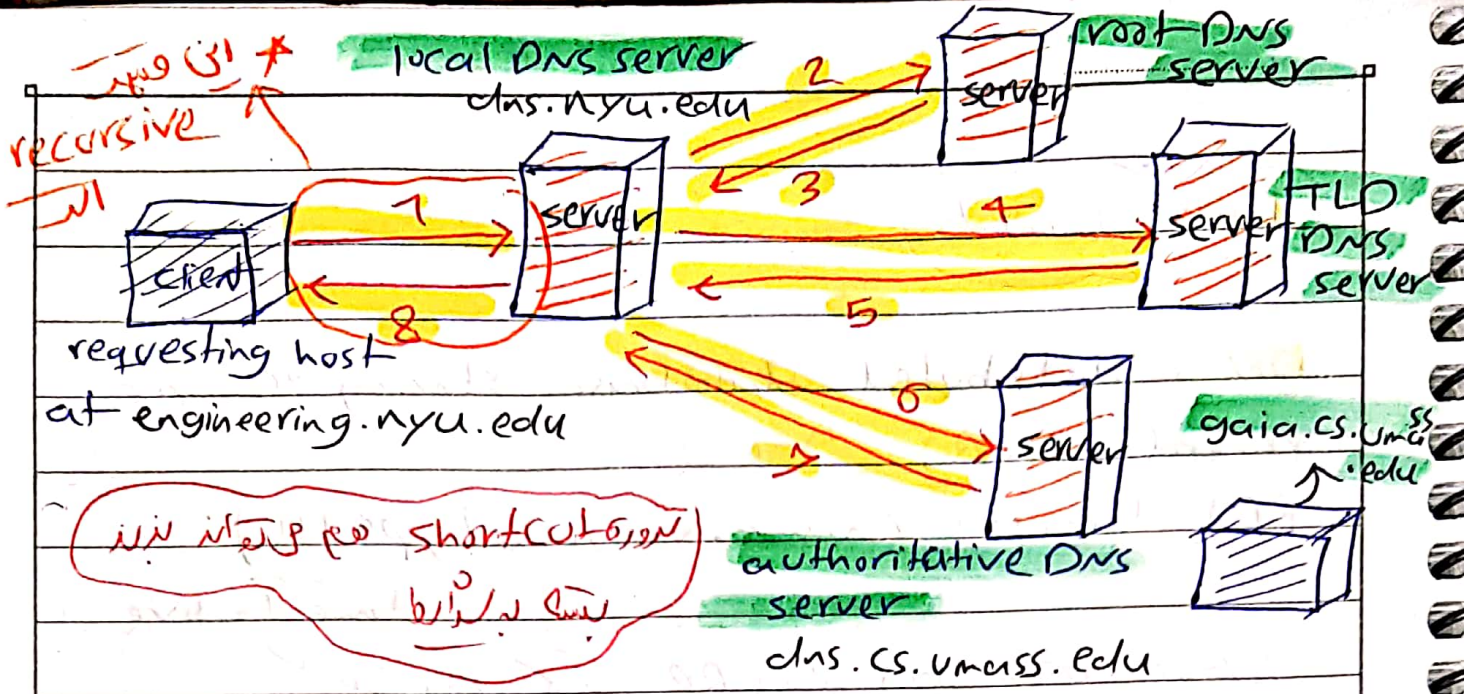
local DNS server returns reply, answering:
• from its local cache of recent name-to-address translation pair — local DNS server اگر یک موسسه یک
داشته باشه client ها که دارن از موسسه هستند بجای اینکه مستقیماً با DNS server ها
• forwarding request into DNS hierarchy for resolution — تماس بگیرند، سوال‌شون رو بپرسند می‌توانند سوال‌شون را
از اون local DNS server بپرسند → که اتفاق می‌افتد
• each ISP has local DNS name server.
to find yours: ① خود local server از سوال جواب بده
• macos: scutil --dns (در cache برود موجود باشه)
• ubuntu: nmcli device show <interface_name>
• windows: ipconfig /all
local server که سیستم تو که می‌توانند از ② برود و از سرور اصلی بپرسد
□ local DNS server doesn't strictly belong to hierarchy — اما DNS و ← پس به تاخیر در رسیدن جواب های سازگاری کار در موسسه‌ها و ISP ها کمک می‌کند
اطلاعات مربوط به local DNS از طریق DHCP در اختیار host ها قرار می‌گیرند.

---

* فرایند رسیدن سوال از DNS server ها * DNS name resolution

example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu (آدرس یک دانشگاه است)

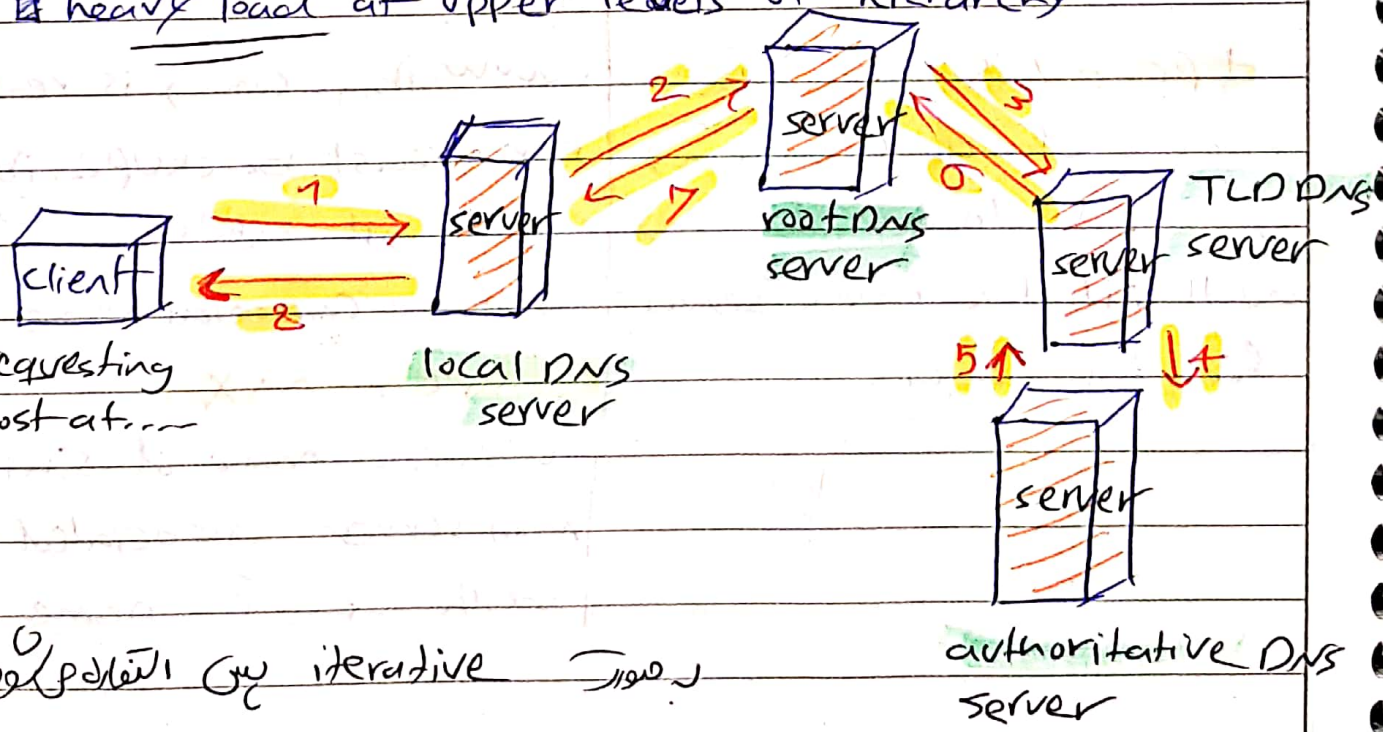
iterated query:
□ contacted server replies with name of server to contact
□ "I don't know this name, but ask this server"



DNS name resolution & recursive query

Recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy



DNS records

DNS : distributed database storing resource records (RR)

(RR) format : (name, value, type, ttl)

ttl = time to live

تا مدت زمانی که یک RR می‌تواند داخل cache ذخیره شود
تا لازم نباشد مجدد از سرور اصلی سوال بپرسیم

وقتی ttl منقضی (منقضی) شد ← RR از داخل cache بیرون ریخته می‌شود
و مجدداً از سرور اصلی سوال پرسیده شود

type = A → وب سرور

name = hostname

value = IP address

type = NS مثلا

name = domain (foo.com)

value = hostname of authoritative name server for this domain

type = CNAME

name = alias name for some canonial (the real) name.

www.ibm.com → is really serverreast.backup2.ibm.com

نام اصلی یک سرور

value = canonial name

type = MX ← اصلی

value = name of SMTP mail server associated with ~~name~~ name

DNS ← از UDP استفاده می کند

DNS protocol message

~~۲ بایت~~ بایت ~~۲~~ بایت

DNS query & reply messages, both have same format

identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions	
answers	
authority	
additional info	

(# questions → تعداد سوال ها ; # answer RRs → تعداد جواب ها)

message header:

1. identification: 16 bit # for query, reply to query uses same #
2. flags:
   - query or reply
   - recursion desired → recursive ترجیح می دهد به صورت
   - recursion available → (در جای) iterative کار کند
   - reply is authoritative

recursion ← اگر سرور بیت recursion را صفر بگذارد یعنی نمی تواند یا صفر بگذارد و خودِ کلاینت باید به صورت iterative سوال خودش را از طریق سوال از سرورهای دیگر به دست آورد

DNS سروری که به ما جواب می دهد authoritative است → DNS server

اگر فرض کنیم هیچ اطلاعاتی درباره ی DNS server نداریم و می خواهیم ip دانشگاه را بدست بیاوریم.
و تنها اطلاعاتی که داریم IPv4 یک root server است. ←

198.41.0.4

dig iut.ac.ir @198.41.0.4

hirmandpaper

Caching DNS information

- once (any) server learns mapping, it caches mapping, and immediately returns a cached mapping in response to a query.

- caching improves response time
- cache entries timeout (disappear) after time (TTL) time to live
- TLD servers typically cached in local name server.

- cached entries may be out_of_date
  - if named host changes IP address, may not be known Internet-wide until all TTLs expire!

- best_effort name_to_address translation.

session TTL DNS security

① DDoS attacks
- bombard root servers with traffic
  - not successful to date.
  - traffic filtering
  - local DNS servers cache IPs of TLD servers & allowing root server bypass.
- bombard TLD servers so more dangerous

Spoofing attacks

- intercept DNS queries, — —

peer_to_peer (P2P) architecture

- no always_on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers.
- self scalability _ new peers bring new service capacity, and new service demands

↓

Client-server برخلاف

- peers are intermittently connected and change IP addresses.

متناوب و IP برای

- complex management.
- examples of P2P file sharing (Bit Torrent), streaming (Kankan), VoIP (Skype).