Com Net

Term : 01

Fighting naksa ayuda ☺

11 August 2020

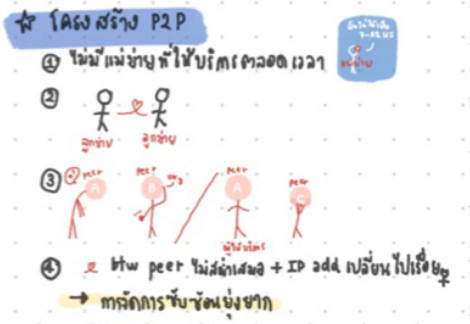
Chapter 2 Application Layer Part1

2.1) Principles of network applications — app architectures / app requirements
2.2) Web and HTTP
2.3) FTP
2.4) Electronic Mail — SMTP / POP3 / IMAP
2.5) DNS
2.6) P2P applications
2.7) Socket programming with TCP
2.8) Socket programming with UDP

Some network apps — email, web, remote login, P2P file sharing, etc. } we call "app พิเคชั่น"

เอ็คคิว

App พิเคชั่น layer คือ layer ที่ take care Application that excute on that เครื่องที่เราเชื่อ
└ take care end system



Creating a network app

2.1) Principles of network applications

Application architectures

1. Client-server architectures (or) client model — * always on host * (on all the time)
   คือ คอมเราป that we have 2 row of host เอ็คตัง in side the network.
2. Peer-to-peer (P2P) — พวก fild downlode ??
   คือ we have only 1 row in side the network * no always-on server * (ไม่on all the time)
3. Hybrid of client-server and P2P
   - Skype : app ที่ working at ins แทน massage and also can call
     is very popular เพราะ เพราะ use hybrid system / connecting of client-server and P2P
   - Instant messaging : chat btw 2 users is P2P

program running within a host

Processes communicating #some processes it use for communicat

Processes will be crate a message and then message send over the network.

create by software

Sockets : the door of the processes. (address, port)
when processes make the request so the sockets will be create connecting btw the appcation layer and the another layer ที่เราneed.

a lot of software we have the sockets served Ex. JAVA

Addressing processes (ขี้แกยข้าม)

App-layer protocal defines (ขี้แกยข้าม)

What transport service does an app need?

(ความสมบูรณ์ของข้อมูล)

1. **Data loss** : require some connecting that can incase to data loss ex. radio steaming
   Ex music stop is เพราะ data loss มัน should be ok to you. ไม่ก็เวลาเราคุยกับเพื่อนแล้วเราสื่อพูดบางคำแต่เข้าใจ
2. **Timing** : some app require low deis of data
   Ex online game เราคลิก and เซิฟเวอร์ก็จะโดดตามที่เรากด โดยที่ไม่ต้องรอ 1-2 s ∴ call "real delay"
3. **Throughput** : to make sure that all from end-to-end connecting u have อัตราพ แบนวิด to run to all of data that u going to use in side the app
4. **Security** : very importan that u have a make sure ว่า no one trap u data บลาๆ

Example เป็นตาราง (ใน slide)

very importan in connecting servide (วัด)

Internet transport protocols services : have 2 any of word (TCP, UDP)

slow - **TCP** : คือเซอวิสthat working as the เซอกิตสวิตชิ่ง
- connection-oriented ต้อง setup การเชื่อมต่อก่อนการให้บริการ
- reliable transport (ส่งข้อมูลที่น่าเชื่อถือได้) btw. process ผู้ส่ง - รับ
- flow control (ควบคุมการไหลของข้อมูล) ผู้ส่งไม่ส่งข้อความไปหาผู้รับ
- congestion control (ควบคุมความคับคั่ง)
- does not provide (สิ่งที่ไม่ได้ให้นำ) ไม่รับประกันเรื่อง ⏲ แบนวิดขั้นต่ำ + ความปลอดภัยของข้อมูล

TCP&UDP
- ไม่มีการเข้ารหัส
- ข้อมูล Passที่ส่งผ่านเครือข่ายถูกดูได้

speed

UDP : คือ service ที่ can เทอไวเรน with the data loss (ส่งแบบไม่น่าเชื่อถือ)
- use คอนเซป เซอกิตสวิตชิ่ง

fastter than TCP

ไม่ได้ให้บริการแบบ TCP สักอย่าง เอาง่ายๆคือมีดีแค่เร็ว แต่พัง อย่างอื่นเสี่ยง+อันตรายข้อมูลอาจหายได้

Example เป็นตาราง

2.2) Web and HTTP

can เป็น / อาจจะเป็น
can be HTML file, JPEG image, etc.
or Java applet, audio file

web page ปก.ไปด้วย file HTML หลัก ๆ หลาย ๆ

Web and HTTP. is web page is consists of objects
- the base HTML-file
- eash object will be addressable by a URL

ที่อยู่ของแต่ละ object ถูกกน. โดย

ex. www.someschool.edu/someDept/pic.gif
(www.someschool.edu = host name; /someDept/pic.gif = path name)

HTTP ใช้ในการติดต่อรับส่งข้อมูลชนิดไฮเปอร์เท็กซ์ระหว่างเครื่องลูกข่ายกับ WWW Server (World Wide Web) โดยเอกสารจะอยู่ในรูปแบบที่เขียนในภาษา HTMLเอกสารแต่ละชิ้น สามารถเชื่อมโยงไปยังเอกสารชิ้นอื่นได้ ซึ่งเอกสารที่ถูกเชื่อมโยงนี้ อาจจะอยู่บนเครื่องคอมพิวเตอร์เครื่องเดียวกันหรือต่างเครื่องกันก็ได้

เป็น Protocal ในชั้น Application ที่ใช้กับ web.

server: รอรับร้องขอ + ส่งข้อมูลที่ถูกร้องขอกลับไป
clients: มี browser ร้องขอ + รับข้อมูล (ใช้ HTTP Protocal) จาก server มาแสดงผล

HTTP overview : HTTP (Hypertext transfer protocol) is server/clients model
- so เปลือก HTML file all of web page obj will be เก็บ on web server

\# Apache Web server =

connect ขึ้น

- **uses TCP** = make รีเควส / server open the connecting send it over and then close
  - uses port 80 # ส่วนใหญ่
- **HTTP is "stateless"** (พี่แกบ้าม)

HTTP connections : have 2 kind of sending HTTP รีเควส

การเชื่อมต่อหลายครั้ง

- **Nonpersistent HTTP** : At most one object is sent over a TCP connection.
  - You hold the connecting and then send it untill finish.
- **Persistent HTTP** : Multiple objects can be sent over single TCP connection btw client and server.

การเชื่อมแบบคงอยู่

Nonpersistent HTTP : at first (1a) HTTP initiates the connection btw host & server
(1b) server will be "accepts", and send accepts notifying client
(2) Client รับ the accepts notifying from server start makeing a request message to contain URL
(3) server receives the request message forms response massage and then send over
(4) HTTP server closes TCP connection.
(5) HTTP client receives response message contain html file, displays html
(6) steps 1-5 repeated for each of 10 jpeg objects

when u not finish รับ the first object u can not be เอาเงิน to start รับing second object #u should go one by one จนจบ.

So it mean u have to do something like this if 10 obj in this webpage so it mean u have to รีเพีท this one 10 time with nonpersistent

เวลาตอบกลับ
Non-Persistent HTTP : Response time is the same thing (33.50)

- **RTT** = the round trip of เวลา ≈ ⏱ ใช้ในการเดินทางของ package เล็กๆ จากฝั่งผ่ายไปยัง server แล้ว กลับมา

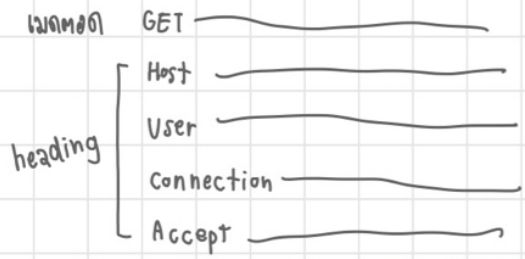
ระยะเวลาตอบกลับ
Persistent HTTP

take slow

- **Nonpersistent HTTP issues** : have some problem because requires 2 RTTs per obj
- **Persistent HTTP** : leaves one connection open and server send มัลติเพิล obj in one connection # this is different
  so they use lit RTT and they เซ็ต it more faster
  (1RTT ในการดึงข้อมูล obj all)

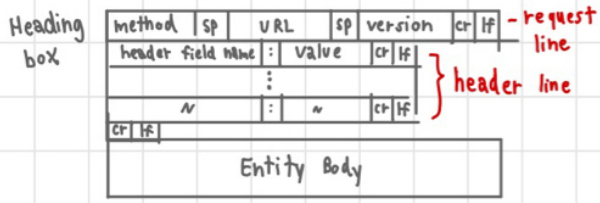
HTTP request message : 2 types of HTTP messages < request (ขอบริการ) / response (ตอบกลับ)

- **HTTP request message** อ่านได้
  - ASCII (human-readable format)



- ~ : general format

comพาย this one
≈ ให้ดูได้ง่ายกว่า / clean กว่า



ส่งข้อมูลไปยัง server
Uploading from input

หน้า web ส่วนมากมีช่องให้ใส่ข้อมูล
ข้อมูล input ที่เรียก ถูกใส่ในส่วน body

- **Post method** : use for upload data frome ไคเอ้น
- **URL method** : use get method (for URL เรเดอส์)

ข้อมูล input ถูกใส่ลงไปใน field URL ที่อยู่ในheader ของrequest

Ex. www.somesite.com/animalsearch?monkeys&banana

Method type : different method

- HTTP/1.0
  - get
  - post
  - head — ไม่ค่อยใช้ use for server to leave requested when out of response.

ปจบ. ใช้นี้ ↓

- HTTP/1.1
  - get, post, head
  - put — upload file
  - delete — delete specified file in the URL field

HTTP response message

สถานะ	HTTP/1.1 200 OK
Date บลาๆๆ	connection close
	Data :
	server :
	Last- :
	Co— :
	Co— :
HTML สคริป	data data data data ...

※ 200 OK = สำเร็จ / ทุกอย่าง OK

HTTP response status codes

1. 200 = Ok // succeeded
2. 301 = Moved Permanently // obj moved new location
3. 400 = Bad Request // message not understand
4. 404 = Not found // file not found
5. 505 = HTTP Version Not Supported // ร้องขอ version ของ HTTP ที่ server ไม่ได้ support

Trying out HTTP (client side) for yourself

1. Telnet
2.
3.

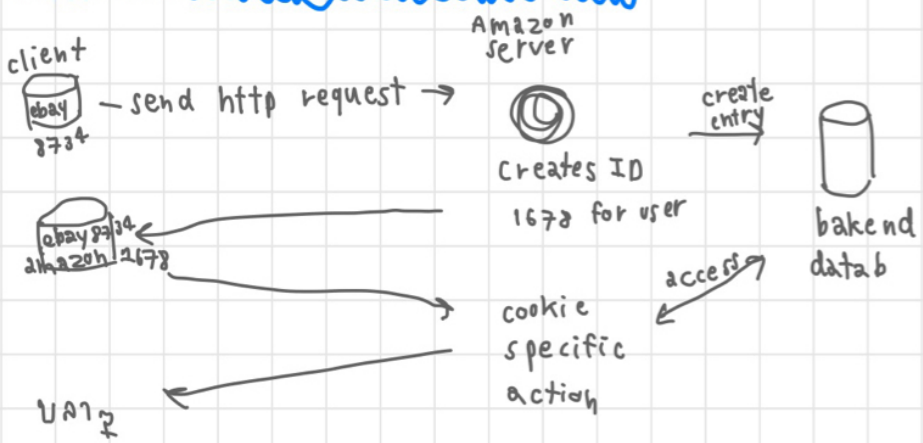
← ลองเชื่อมด้วยตัวเอง (P. 33)

User-server state : cookies

: one of the importent part of the website คือวาริอะเบิล that website write down in to your maching จะคอนเทน u personหอล information ex. login ก็จะสร้าง cookie มาเวลาเข้าใหม่ก็จะได้ไม่ต้อง login ใหม่อีกเพราะมัน read cookie ที่มันสร้างมา (ป่ะ??)

website จน. มากใช้ cookies

Cookies: keeping "state" (cont.)



cookies & privacy
- ทำให้ web รู้ข้อมูลของ น เยอะ
- อาจรวมถึง e-mail ชื่อ-นามสกุล

Cookies (cont.)

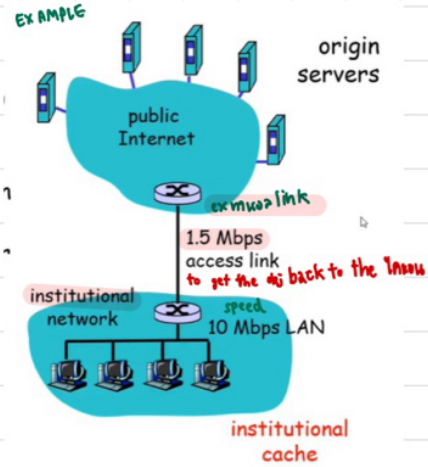
- What cookies can bring (ทำอะไรได้บ้าง)
  - authorization (การระบุตัวตน)
  - shopping carts
  - recommendations (แนะนำสินค้า/ข้อมูล)
  - user session state (Web e-mail) ปลาย (สถานะของขั้นตอนการใช้งานของผู้ใช้)

Chapter 2 Application Layer Part 2

Web caches (proxy server) : working as the บัฟเฟอ information (it in the middle) # at slide u see

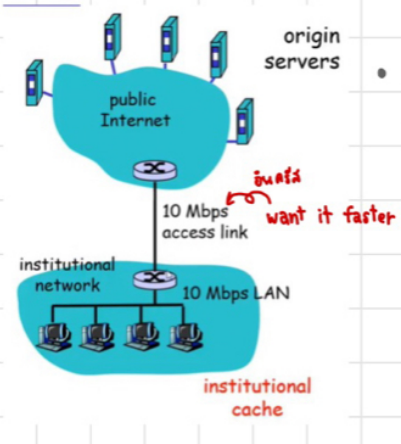
① Proxy server have information of the จุดหมาย (เคยมีแน่นั่น) it เก็บit for the second or for another request that going to the same จุดหมาย so u don't have to go out that very singnia time that u request

More about web caching : Why we have to caching because reduce the time request
: reduce traffic

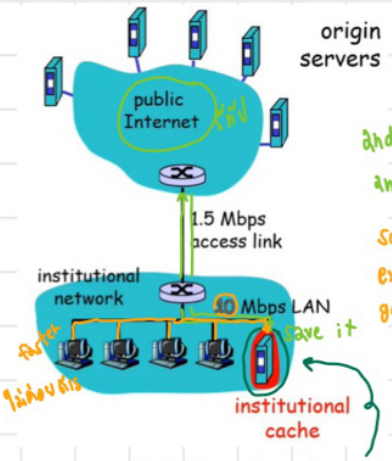


Caching example

- assumptions
  consequences



- possible solution
  consequences



and ที่กับ what every they want and save it in here

so every cliend จะร้องขอ the same ex.ร้องขัง the content it always go here

but if we have the proxy server

so it mean the proxy server will go out

Conditional GET

: GET method send to the proxy server we have 2 สถานการณ์

① When they get the request and the server find out object isn't modified

so it mean proxy เก็บ web content → มันต้องเช็คตัวมันเองว่าเก็บมาอัพยัง or uptodate ไหม ถ้า uptodate มัน can over ตอบกลับ to the cliend

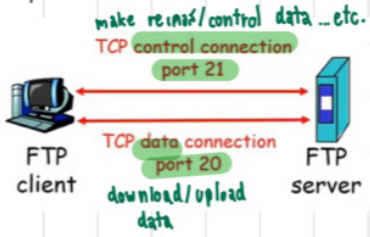
but if it not uptoday proxy จะ go out and ที่กับ new information and save it to the proxy again.

2.3) FTP คือ another app ที่ very popular and useful in the network

FTP : the file transfer protocol : is protocal s app that runing on this protocol it working for upload file from cliend in to the server or download (up and down)

#ftp server : port 21

FTP: Separate control, data connections : they have 2 port to use



: use TCP connection

FTP commands, responses

Sample commands:
- sent as ASCII text over control channel
- USER *username*
- PASS *password*
- LIST return list of file in current directory
- RETR filename retrieves (gets) file
- STOR filename stores (puts) file onto remote host

Sample return codes
- status code and phrase (as in HTTP)
- 331 Username OK, password required
- 125 data connection already open; transfer starting
- 425 Can't open data connection
- 452 Error writing file

: FTP use (ASCII text commands) = normal text commands

Ex. USER username , PASS password , LIST return list of file

: so when the ที่ส่ง get reมัน from the server

Ex. 331 Username OK , 425 / 452 is error massage

we use at the time that we want to upload the big chan of the file like 1 website etc.

2.4) Electronic Mail

- SMTP, POP3, IMAP

Electronic Mail (E-mail) : use SMTP Protocal to send the e-mail

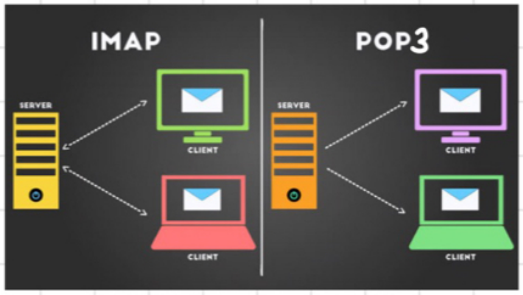
user agent =

agent =

we use agent สร้าง e-mail and send to the server , mail server จะรับ u email and then make resend to the network to another mail server.

Mail server

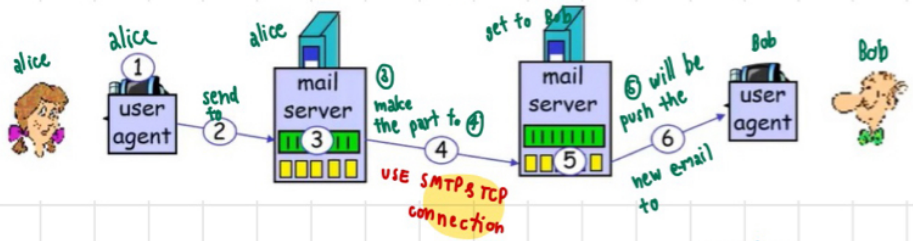
- mailbox
- message queue
- SMTP protocol



Electronic Mail: SMTP [RFC 2821]

* use the ASCII text commands (same as MTP)
* we have 3 phase of transfer — handshaking
  - transfer of messages
  - closure
* messages must be in 7-bit ASCII (is well ☺)
  ไม่ใช่ 8-bit

Scenario : Alice sends message to Bob



Sample SMTP interaction : Ex. : start with hard~~chacking~~ chacking process

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

— hello
— hello back
} hardchacking process that the hamburger.edu มัน ex-ist/sw

and then they makeing a resend

— finish every thing

Try SMTP interaction for yourself

⇐ ลองด้วยตัวเอง

* telnet servername 25
* see 220 reply from server
* enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

above lets you send email without using email client (reader)

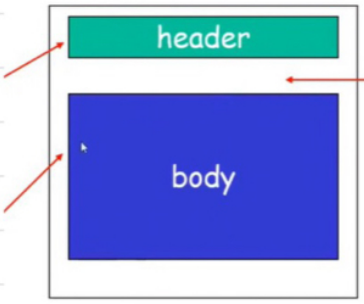
SMTP : final words

- SMTP uses persistent connections
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses CRLF.CRLF to determine end of message

Comparison with HTTP:

- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

Mail message format



it same as HTTP

Message format : multimedia extensions

is the extension additional for e-mail to have more contro and more ฟังก์ชั่น

* MIME : multimedia mail extension

Mail access protocols

is we have * POP : Post office Protocol [RFC 1939]

User connect to mail server to download the e-mail back to the user

* IMAP : Internet Mail Access Protocol [RFC 1730] (more complex) more features — manipulation of stored msgs on server
  - it is another ภาษา for ... they have the detail ในนั้น
* HTTP : gmail, Hotmail, Yahoo! Mail, etc. — free email
  - it very popular that we use HTTP to working at the mail agent so u don't have to ติดตั้ง any mail agent to u PC ที่ e-mail u can login g-mail, hotmail etc. use mail agent to connect with the mail server.

POP3 protocal

is protocal ที่ makeing the request from the mail เธอร์ to the mail server to request เอกสาร from server and then end the list and end email that can request เสร็จ.

POP3 (more) and IMAP

More about POP3

- Previous example uses "download and delete" mode.
- Bob cannot re-read e-mail if he changes client
- "Download-and-keep": copies of messages on different clients
- POP3 is stateless across sessions

IMAP

- Keep all messages in one place: the server
- Allows user to organize messages in folders
- IMAP keeps user state across sessions:
  - names of folders and mappings between message IDs and folder name

– can สร้าง folder, move, request เอกสารใน control by IMAP

Chapter 2 Application Layer Part 3 [End]

2.5) DNS

DNS: Domain Name System

it's like a name system for network ex. Id address (เพื่อนๆ)

* DNS is when people very hard to remember only number so the invent เหตุผลนี้ that we ... name the num (เลขๆ) ...
* What the DNS system มันเหตุผลนี้ distributed database so it mean we have database and we have many database request together

People: many identifiers:
- SSN, name, passport #

Internet hosts, routers:
- IP address (32 bit) - host used for addressing datagrams
- "name", e.g., dns ww.yahoo.com - used by humans

Q: map between IP addresses and name ?

Domain Name System:
- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

DNS

DNS services
- hostname to IP address translation
- host aliasing
  - Canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?
- single point of failure
- traffic volume
- distant centralized database
- maintenance

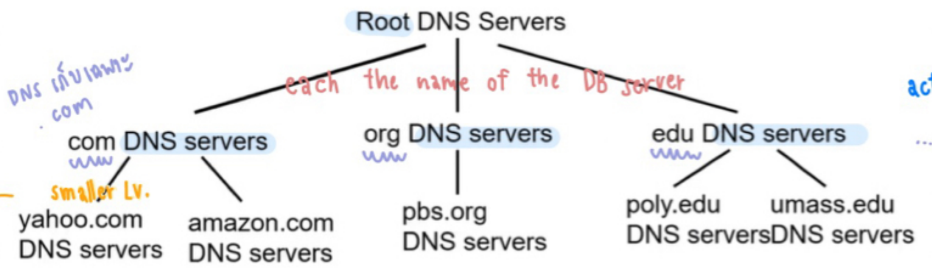
doesn't *scale!*

if we set the DNS server at a center it means when the server down all of the DNS server ก็จะ down ไปด้วย

บางที่ไกลจาก database

it good if we have database sit in each contry and they go step-to-step would be better

Distributed, Hierarchical Database



server Lv. it help user that under this พวกตัวล่าง domain name to เรียก/search to the DNS system.

Client wants IP for www.amazon.com; 1st approx:
- client queries a root server to find com DNS server
- client queries com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com

DNS : Root name servers

* it the biggest โหด of the DNS ทุกเซอร์เวอร์ต้องเรียก
* it have many สถานที่ in the world

for example:

13 root name servers worldwide

TLD and Authoritative Servers

- Top-level domain (TLD) servers: edu เท่านั้น
  - responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
  - Network Solutions maintains servers for com TLD
  - Educause for edu TLD
- Authoritative DNS servers: (Low - LV.)
  - organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web, mail).
  - can be maintained by organization or service provider — CMU

als or anything that we provide internet connecting to cliend

*TLD = Top- LV. - domain

e it เก็บ all หมวดหมู่ under .com or .net ... etc. (.org)

a big node

Local Name Server

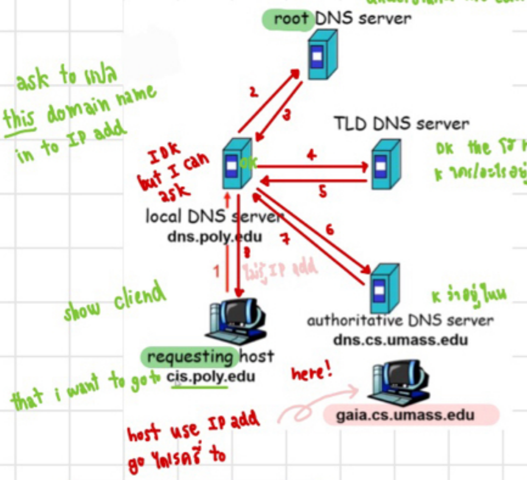
: so each host have to make the DNS query*

#so when every single time that the cliend in side the network request to connect to some where they have to send request to DNS server and DNS server send out the DNS query to upper layer of the DNS server

DNS name resolution example

* we have DNS ... in 2 way

① iterated query : 1 way that we make the DNS query to get the ...



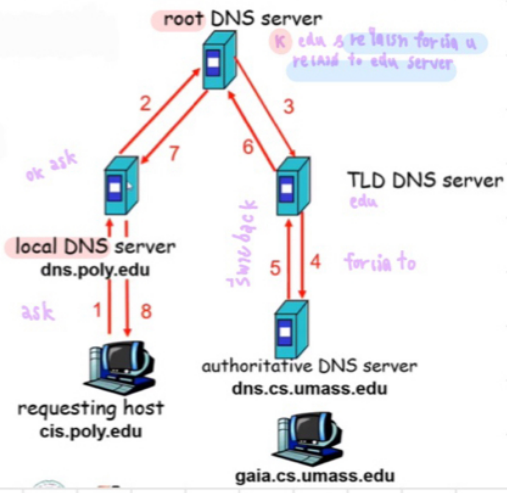
* ถ้ารู้แค่ only name it mean ... in the network because the network only ... the IP add

(พี่แก show in window)

② recursive query : this's another way to ... the DNS ...

* Q & A : it more heavy load more than the last one or not?

: yes, it more heavy load.



DNS: caching and updating records

- ❐ once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time
  - TLD servers typically cached in local name servers
    - Thus root name servers not often visited
- ❐ update/notify mechanisms under design by IETF
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

ยกตัวอย่าง เช่น : Aj. รู้คำตอบ Aj can ตอบได้โดย useing ความจำ ของ Aj เอง it that good idea!

DNS records

: have 4 type — A, CNAME, NS, MX

it look like RR format: (name, value, type, ttl) — ttl: contro

- * type A — name is hostname; value is IP address
- * Type NS — name is domain name (sub domain / swicht domain under need the main domain); value is hostname — that it under the this domain name
- * Type CNAME — name is alias name (when u want to แปลง the wrong name in to short name.); value is canonical name
- * Type MX — value is name of mailserver

DNS protocal, messages

= in each messages of the DNS it will look like this when it send out to get the transaction

* DNS protocol: query and reply messages, both with same message format.

msg header
- identification : 16 bit #
- flags :
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative

header →

identification (16 bit)	flags (have identifile)
number of questions	number of answer RRs
number of authority RRs	number of additional RRs

12 bytes

data itself @ the คิวรี่ s Q →

questions (variable number of questions)
answers (variable number of resource records)
authority (variable number of resource records)
additional information (variable number of resource records)

data

questions (variable number of questions)	Name, type fields for a query
answers (variable number of resource records)	RRs in response to query
authority (variable number of resource records)	records authoritative
additional information (variable number of resource records)	helpful info

Inserting records into DNS

= u register new name with new IP add — have 2 kind:
- สแตติก = fix IP not เปลี่ยน
- ไดนามิก = ยุ่งยาก, good for client use already — # it เปลี่ยน in sometime.

the name of the domain name it have to be uni : have to be a new one not the same name

must be แมพ with the IP add — have to be (ซื้อเพิ่ม) เจ้าของ IP add

2.6) P2P applications

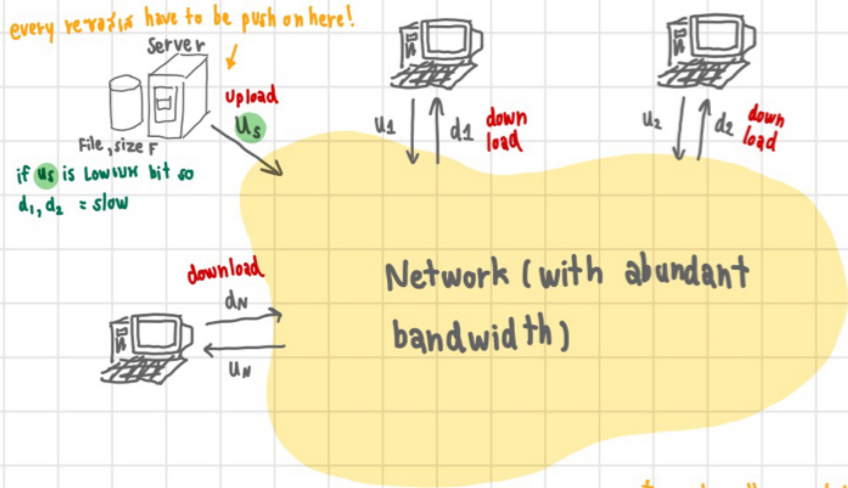
Pure P2P architecture

= is the สถาปัตยกรรม that we have 1 row in the network

Cliend connect ไปหากัน to Cliend . so not alway on server in side the network and end system connect to end system

* 3 topics
  - File distribution
  - Searching for information
  - Case study : skype (รีวิวของ แชร์ing)

File Distribution : Server-Client vs P2P



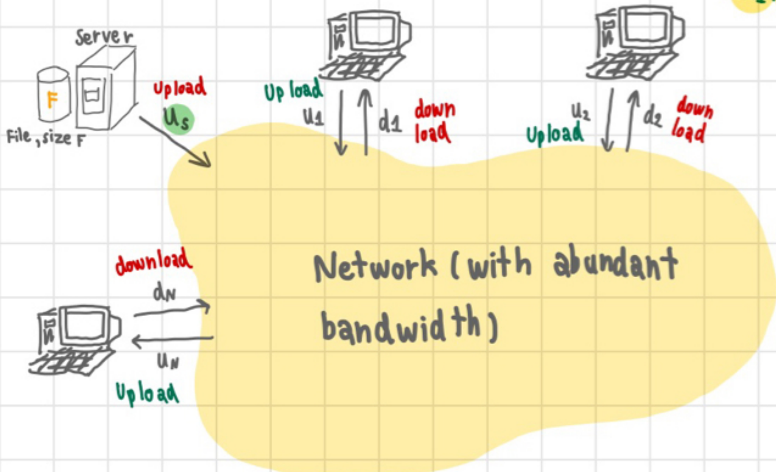
not แชร์ Us with someone แล้ว

* U_s : server upload bandwidth
* U_i : peer i upload ——
* d_i : —— download ——

to make all คนอื่น download fast

File distribution time : server-client

: many way of upload but download only Us (u_1, u_2, u_N)



// พี่แกข้าม Part calcu ไป

File distribution time : P2P

= we have upload & download equly.

* helping upload for each other so when download d_1 may be connect to $U_{1,2}$ or b_2 connect to dn or $d_{4,5}$ so it

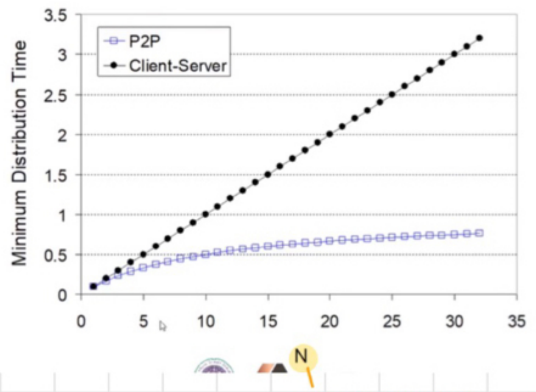
(U_1, U_2, U_n)

ร้องขอ dis ที่มันรัน to help database calm out from the network so it faster because the $U_s + \Sigma U_i$ must faster than this one !!

Server-client vs. P2P : example

this is คอมแพร์ชั่น ชาด (แทนฟังก์/ดูมั่)

Client upload rate = u, F/u = 1 hour, u_s = 10u, $d_{min} \geq u_s$



: Client-server take more time when n increase

if N increase the time will be more

more client = more upload เยอะ

└ because when client get ส่วน1 ของ info that download it. They also upload what they have in to the network

so it mean more time when more N

not just download!

N is ตัวแทน for number of client that connect → that make เยอะ for download

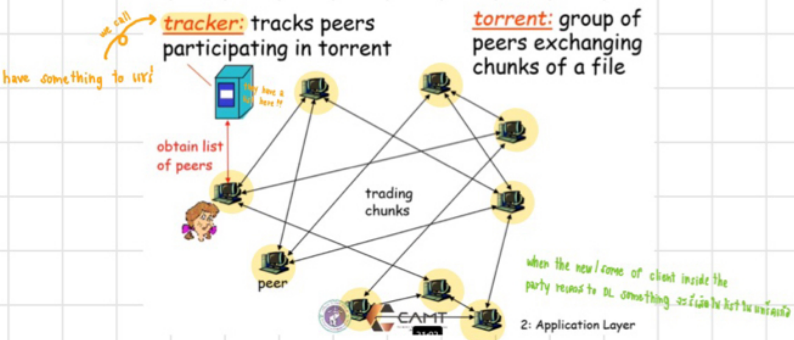
* When we have the ส่วนแรกของ info, it take ใกล้ๆ the same time because client don't have the info เพียงพอที่จะ แชร์ but when they have more client it will be faster because each client also help upload the ส่วน1 ของ info that they already have to another one who come later as well.

more client = more สเกลเบิ้ล download ← must better than this ↗

File distribution : BitTorrent

* at first bitTorrent have to know who has ส่วนของ file that เราจะ แชร์

so it mean we have to create a list of file / info in each client that who has it and how many what they have



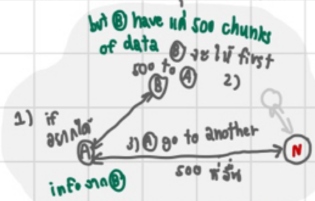
* tracker just help to ให้ the ส่วนของfile / info where it is who have it and then the client have go to by themself by P2P

BitTorrent (1)

: ถูกตัดให้เป็นส่วนที่เล็ก เราเรียกส่วนนั้นว่า "chunks" — size : 256 KB

* so If Ⓐ ask Ⓑ for a chunks Ⓑ...

if we have 1000 chunks as a one ส่วนของinfo ... person ที่ have chunks can provide info to the 1 who get DL



they don't have a ส่วนของinfo

ดูรูป Pic. จากด้านบนมา

so when the person Ⓐ DL in the chunks in the same ⏱

if another person ask to the chunks this person alreay have → it will be เอเบิล to ให้ the chunks that the person Ⓐ here alreay have to the new guy as well.

in the same time ⏱

BitTorrent (2)

: we have * Pulling chunks (we cell a "DL chunks")

* Sending Chunks

P2P: searching for information

: we have something each

* file sharing (like e-mule system) — for Index / have แทร็กเกอร์ itself :)

when client ติดตั้ง e-mule PG เราแค่ย้าย file ที่เรา want to แชร์ใน file upload folder and e-mule จะ get a list to another P that can see u UL folder info and then they can ดู from u UL folder — very easy system to use

ถ้าหา สิ่งที่จะ file ไม่เจอ ใน bittorrent system บางที่อาจจะเจอใน e-mule

* Instant massaging : it also the ระบบ of P2P
  - it working in the same thing เพราะ it มี server working at the index

ex. when u want to คุย with someone the server จะให้ IP add of that พาทิศวงล่า member that u want to คุย and then u just go P2P to them.

P2P: centralized index

P2P: problems with centralized directory

(พี่แกข้าม)

2.7) Socket programming with TCP

Socket programming

is the PG we สร้าง ให้ send / send a ส่วนของ contro / something info over the network

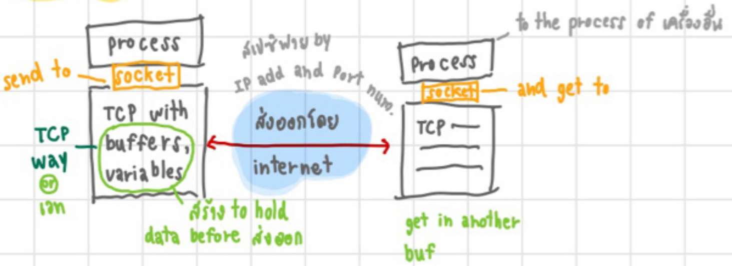
โดยปกติแล้ว we need socket API → to use with
  - use java API สร้าง by UNIX (1981)

Socket-programming using TCP

* Socket = door btw app and under need a protocal. We have 2 kind ระบบ < TCP / UCP } ทำงานต่างกัน

TCP = connection ออเรนเต็ด (u have to รอรับ connect / hold connection until the end of ทรานแซ็คชัน)

UCP = u don't have รอรับ anything แค่ send out it ส่งไป a way on the network

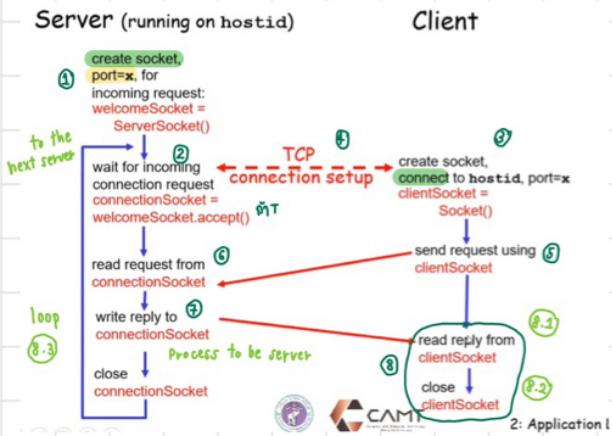


Socket programming with TCP

* Client must contact server (it work on client server model & server model ต้อง on (running), สร้าง socket that welcomes client contact)
* Client contacts server by:
  - have to สร้าง contact to the server by สร้าง client TCP socket ส่งไปยัง server socket it must be the same boor (เช่น Specifying IP address, port num have to be the same)
  - When client สร้าง socket → establishes connection when server เจอ เลยโปรเซส

* start ส่งขนาดอย่างออกไป and then close the connection

Client/server socket interaction : TCP



Stream jargon

KEYBOARD

see box = have some variable buf

for รอรับ

Client part

Example : Java client (TCP) : พี่แกให้ code ไปรันเอง (46.54)

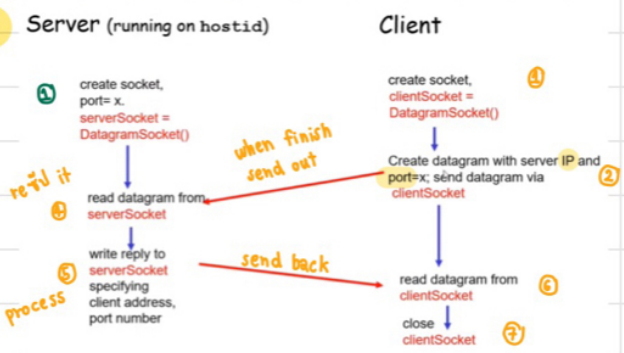
Java server (TCP)

2.8 Socket programming with UDP

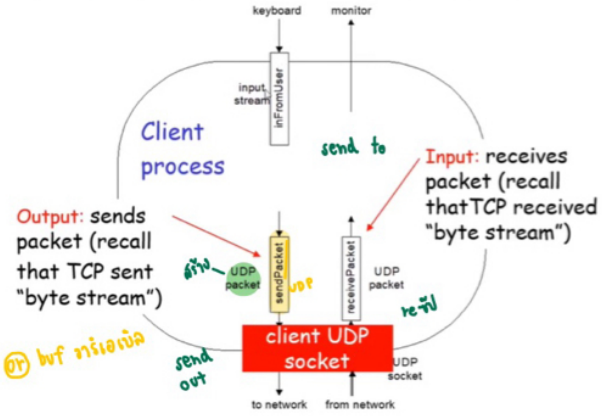
Socket programming with UDP : คล้าย TCP แตกต่างกันตรง connection btw server & client

: UDP no connection btw client & server

: no handshaking



Example : Java client (UDP)



Example : Java client (UDP) : Code ให้ฝึกทำ / ทวน (S1.20)

Java server (UDP)

Chapter 2 : Summary

* our study of network app now complete! *

- application architectures
  - client-server
  - P2P
  - hybrid
- application service requirements:
  - reliability, bandwidth, delay
- Internet transport service model
  - connection-oriented, reliable: TCP
  - unreliable, datagram UDP

- specific protocols:
  - HTTP
  - FTP
  - SMTP, POP, IMAP
  - DNS
  - P2P: BitTorrent, Skype
- socket programming

CAMT

Most importantly: learned about *protocols*

- typical request/reply message exchange:
  - client requests info or service
  - server responds with data, status code
- message formats:
  - headers: fields giving info about data
  - data: info being communicated

Important themes:

- control vs. data msgs
  - in-band, out-of-band
- centralized vs. decentralized
- stateless vs. stateful
- reliable vs. unreliable msg transfer
- "complexity at network edge"

Chapter 3 Transport Layer Part1

2.1) Principles of network applications

CHAPTER 3 : P1

logical
L differ
is mien end to end to ser
fisic
——→ to software

★ **segments**

★ **transport layer** = comม in the real life.

care end-end

★ **network layer** = แอนเถือ เดอะ way to send ?

สเปก it out
↑
Demultiplexing

★ ส่ง ดาต้า . . .   contro by ____

UDP = อันตรายไกลเปิด ควบคุม check . . . . . .
↓
need
↓
some kind

SP = พอร์ตของ = ทาง
DP = = รับ.

read before connect = no handshaking.

☆ **flipped bits** = bits have 0, 1 but it read 1

data ผิด
sum num.

☆ **check sum**

17 29 34
5 12 11 6 34

+ all = 34 แล้วใส่ -

ค่าของ data = 0 = ไม่มี error
ถ้ามี 0 = have error in data