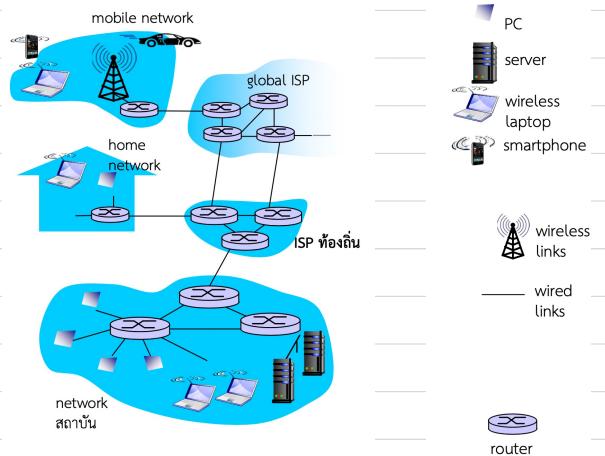


09 : Introduction

និច្ចមានការបន្ទាន់រាល់ទៅលើការពិនិត្យ។ “តួនបានដោយបាន”

- hosts : ພົມວິທະຍາກົດລົງໄຊເອງກ່າວນຸ້ມ
 - communication link : ສໍາຕັບຖານນີ້ໃຫຍ່ກ່າວນຸ້ມ
 - Server , computer , server , monitor
 - ດົວລະນີ້ວິວ : Bandwidth.
 - Packet Switch : ປົມວິທະຍາກົດ (ຮົ້ວວິວຈຳ)
 - ອັນນີ້ : routers ແລະ switches.
 - Protocols : ນັກໂນໂລຢີ - ຂໍ້ນິ້ງຈຳ
 - e.g. TCP IP HTTP Skype etc. 802.11



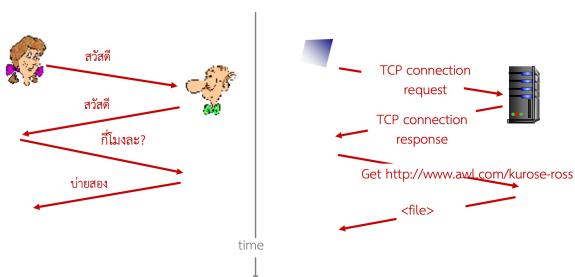
କେବେଳିନ୍ତରୁମାର୍ଫିଟ୍ସ୍ (Network) ମାଗନ୍ଦିଲୋକ ?

- Network edge : ឧបករណីដែលរាយសម្រាប់បន្ថែមទិន្នន័យ
និងចេញផ្សាយពេលវេលាដែលមានតម្លៃខ្ពស់។ នៅក្នុងគម្រោងនេះ មានពីរ។
 - Access Internet : ឧបករណីដែលរាយសម្រាប់បន្ថែមទិន្នន័យ
និងចេញផ្សាយពេលវេលាដែលមានតម្លៃខ្ពស់។ នៅក្នុងគម្រោងនេះ មានពីរ。
e.g. router / switcher / Access Point

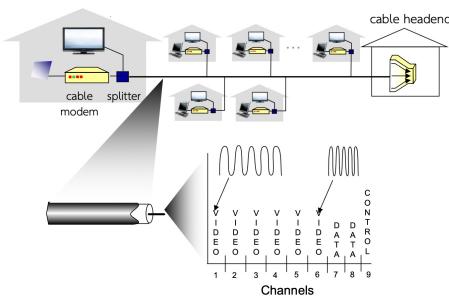
Protocol តើខ្លួនទៀត ?

ବୀଜିଲାକ ମନ୍ଦିରପାଇନ ଟର୍ମ - ଏ ବୀଜିଲାକ ମନ୍ଦିରପାଇନ ଟର୍ମ - ଏ
ବୀଜିଲାକ ମନ୍ଦିରପାଇନ ଟର୍ମ ଏହା କେବଳ ଏଠାକା ମନ୍ଦିରପାଇନ ଟର୍ମ ନାହିଁ.

Human protocol \Rightarrow Computer network protocol:

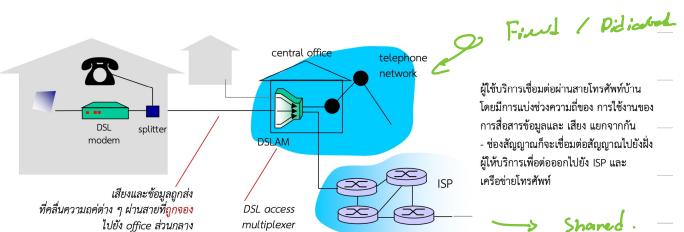


Access Net : Cable Network



frequency division multiplexing: ช่องทางที่แตกต่างกันถูกส่งในคลื่นความถี่ที่แตกต่างกัน

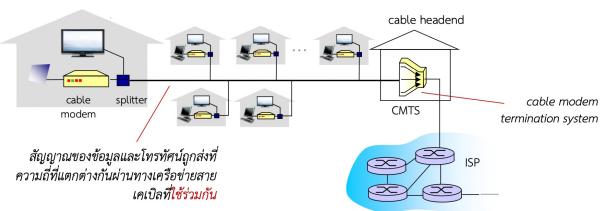
Access Net : Digital subscriber line (DSL)



៥៥. សារព័ន្ធឌីជីថលទៅវគ្គប្រើបាន នៃការអភិវឌ្ឍន៍របស់ខ្លួន និងការអភិវឌ្ឍន៍របស់ប្រជាជាតិ

- ພົມບູນ : ສັງເກດມານີກສິນໄປສູງ, ເພື່ອຕົວແທນ
 - ດິຈຸນ : ດັ່ງນີ້ແລ້ວ ໄກສະນິກ ແລ້ວ ເຫດຜົນ (ໂຄສະນາ).

- போகியுமின்ஸ் என்றும் (Up stream) தொடு < 2.5 Mbps
(மாதிரி 1 Mbps)
 - வெடியுஸ்ட்ரீம் (Down stream) தொடு < 2^a Mbps
(மாதிரி 10 Mbps)

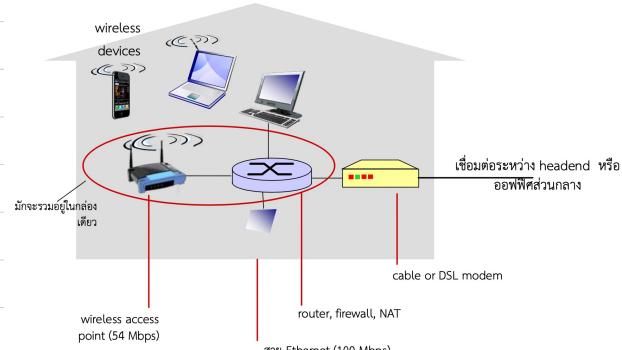


HFC : Hybrid fiber coax

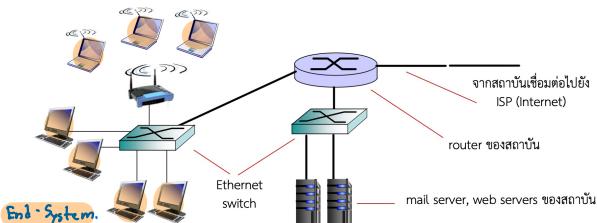
ระบบนำไฟเบอร์จาก ISP ไป Fiber ก่อนถึง CMDS ลงสู่บ้านผ่าน coax และกลับสู่ ISP ผ่านบ้านอีกด้วย.

ขั้นตอนขึ้นบ้าน (Upstream) : Max 2 Mbps / s
ขั้นตอนลงบ้าน (Down stream) : Max 30 Mbps / s.

Access Net : จัดการเข้าสู่เครือข่าย



มาตรฐานสำหรับเครือข่าย LAN (Ethernet)



มาตรฐาน ISO/IEC : บริษัท ดูแลมาตรฐาน

- ขั้นตอนขึ้นบ้าน : 10 / 100 Mbps - 1 / 10 Gbps.
- End-System ต่ออยู่กับตัว Ethernet Switch.

สาย Fiber Optic Cable : 95% เนื่องจากความต้องการในการสื่อสารในบ้าน 1 bits

- ความจุสูง 10 - 1000 Gbps
- สามารถขยายตัว : ใช้ Repeater หรือ Router
- ความต้องการสื่อสารในบ้านเพิ่มมากขึ้น

โครงสร้างสื่อสารแบบไร้สาย (Wireless)



Wireless LANs

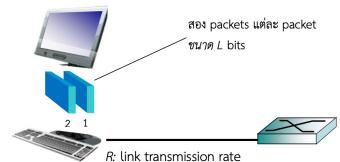
Local Area Network.

- ระยะทาง 100 m
- ความเร็ว ประมาณ 802.11 b/g.
(WiFi) 11, 54 Mbps

Wide Area Wireless Access (WAN)

- ผู้ให้บริการ ISP มาก 10 km
- ความเร็วในการรับส่ง 1 - 10 Mbps
- ตัวตน : 3G 4G : LTE

Host : ส่ง packets บนบ้าน



1. ตอบสนองของ Application

2. ไมโครโปรเซสเซอร์ (CPU)

packets มีขนาด L bits)

3. ส่ง package ผ่านชั้นประสาท
ความเร็ว R

$$\text{ความเร็วในการถ่าย}\quad \text{เวลาที่ต้องใช้ในการส่ง L-bit} \\ \text{โอนแพคเกจ} = \quad \text{packet ผ่าน link} = \frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$$

สื่อสารทางกายภาพ (Physical Media)

- bit : BITS บิต (Binary digit 0 1 0 1)

- Physical Link : ลิงค์ที่ต่อระหว่าง Source หรือ Destination

- Guided Media : Fiber Optic จาระตัว สาย Coax

- Unguided Media : ไม่มีทางเดินสัญญาณ เช่น อากาศ แม่เหล็ก

- Twisted Pair (TP) : สายไฟฟ้าหุ้นส่วนที่มีความต้านทานต่ำ

e.g. Cat 5 : 100 Mbps / 1 Gbps Ethernet

Cat 6 : 10 Gbps

สาย Coax vs สาย Fiber Optic

- สาย Coax : สายเดียวที่ต้องมีตัวกลาง

Bidirectional : สองทางเดียว (Up-Down) ที่ต้องมีตัวกลาง

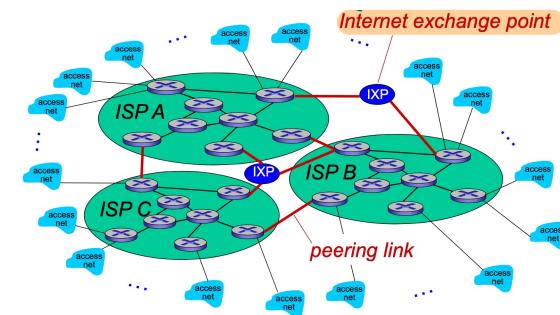
Broadband : จัดการความจุสูงโดยการต่อตัวกลาง

HFC (Hybrid Fiber Coaxial Network) คือการนำ Optical Fiber Cable และ Coaxial Cable.

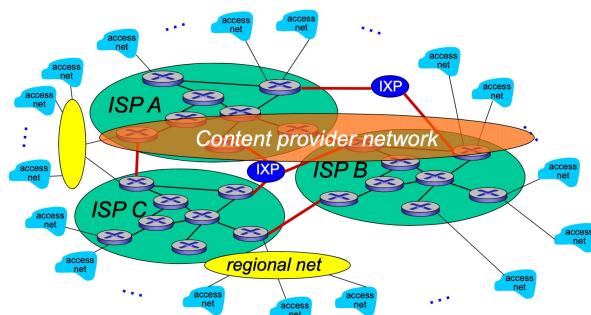
Internet Structure : Network of Network.

- End System ဆုတေသနတွင်ပါဝင်ပါမည့်စူးပေးဆိုလော်မှု
- ISP ဆုတေသနတွင်ပါဝင်ပါမည့် ISP တွင်ပါဝင်ပါမည့်စူးပေးဆိုလော်မှု
- အောက်ပါတော်းချွဲတွင် အောက်ဖော်လော်မှုများ ဖြစ်ပါသည်။

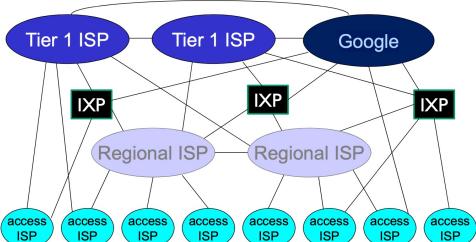
Internet Structure : Network of networks.



IXP (Internet exchange point) : အောက်မှုများ ဖြစ်ပါသည်။
- Peering link ရှိပါသော ISP တွင်ပါဝင်ပါမည့် IXP.



Content Provider Network : အောက်မှုများ ဖြစ်ပါသည်။
- Server ရှိပါသော အောက်မှုများ ဖြစ်ပါသည်။



1. access net (ISP)

2. regional ISPs

3. IXP (Internet Exchange Point)

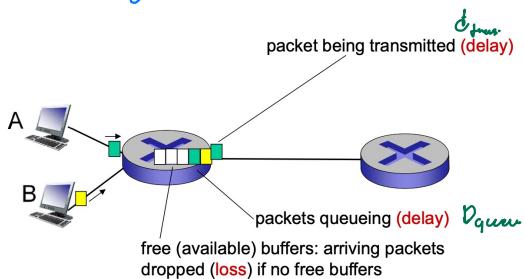
4. Tier 3 / National ISP

5. Content Provider Network.

minimum Delay Optimization.

မြန်မာစာတွင် Drop ဟိုမှု
1. packet transmission delay \rightarrow မြန်မာစာတွင် packet loss.

2. packet queueing delay



4. Minimizing packet delay.

1. d_{proc} : nodal processing. ($\sim \text{ms}$)

- access link resources.

- output link

- Routing table.

2. d_{queue} : queuing delay

- resource reservation (wait for transmission)

- bottleneck queue Congestion via routing.

3. d_{trans} : transmission delay ($\sim \text{microseconds}$)

- L : unicast packets (bit)

$$d_{trans} = \frac{L}{R} = d_{trans}$$

- R : bandwidth (Link bandwidth)

4. d_{prop} : Propagation Delay (microseconds)

- d : distance Physical Link

- s : propagation speed ($\sim 2 \times 10^8 \text{ m/sec}$)

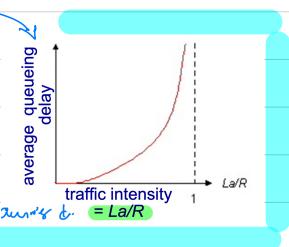
$$d_{prop} = d/s$$

Queuing Delay

- R : bandwidth link (bps)

- L : unicast packets (bits)

- a : acknowledgement packets



$La/R \sim 0$: Queue length 0 (no queuing delay)

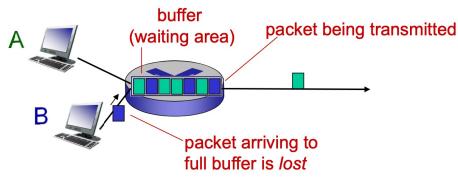
$La/R \rightarrow 1$: Queue length 1 (minimum delay)

$La/R > 1$: Very long queue length. Queue infinity.



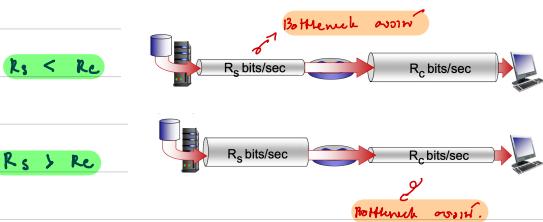
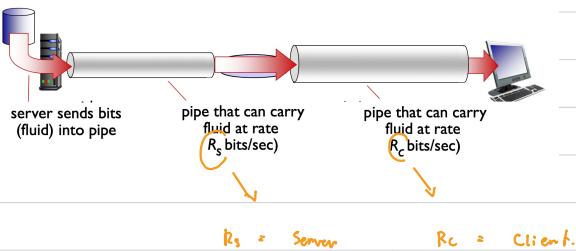
Packet loss (dropping packets)

- នៅលើ Buffer បានបង្កើតឡើង។
- នៅលើ Buffer ត្រូវបាន Drop Packets ដូចខាងក្រោម Lost.
- lost packet នឹងបានបញ្ជាក់ (Retransmitted) នៅពាណិជ្ជកម្ម។



Throughput

- **Throughput** : សម្រាប់បាន ទំនួរទំនួរ NW នៃ .
- **Bandwidth** : សម្រាប់បាន និរន័យទំនួរ NW នៃ .



Network Model : ISO / OSI

"All People Seem To Need Data Processing" នៅលើ OSI Layer.



7. Application Layer : ទទួលទាញការណ៍យោង និងផ្តល់ការណ៍យោង នៃការប្រើប្រាស់ការពារិក និងការប្រើប្រាស់ការពារិក។ (FTP, SMTP, HTTP)

6. Presentation Layer : រួចរាល់ការពារិក និងការប្រើប្រាស់ការពារិក។ និងការបញ្ចូនការពារិក។ (Encryption) និងការប្រើប្រាស់ការពារិក។ (Compress, compress)

5. Session Layer : រួចរាល់ការពារិក និងការប្រើប្រាស់ការពារិក។ និងការបញ្ចូនការពារិក។ (Checkpoint, recovery of data exchange.)

4. Transport Layer : ផ្តល់ការពារិកនៃការប្រើប្រាស់ការពារិក។ TCP, IP.

3. Network Layer : ផ្តល់ការពារិកនៃ datagram នៃ Server ទៅ Destination. IP, routing protocols.

2. Link (Data) Layer : ផ្តល់ IP Address នៃគម្រោងមែនមែន ដែលបានបង្កើតឡើង។ Ethernet, 802.11 (WiFi), PPP

1. physical Layer : សំរួចរាល់ស្ថាបន ឬសុវត្ថិភាព នៃ 0 និង 1

Encapsulation

ឯកសារ

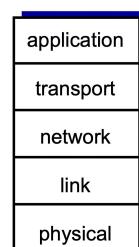
message

segment

datagram

frame

physical



Message

$H_1 \rightarrow$ Message

$H_2 \rightarrow H_1 \rightarrow$ Message

$H_3 \rightarrow H_2 \rightarrow H_1 \rightarrow$ Message



1. Transport Header (H_1) : Port Number

2. Network Header (H_2) : Logical Address

ឬ IP Address.

3. Link Header (H_3) : Physical Address / Mac (Ethernet Address)

Network Security

- សំណង់សារ និងការបញ្ចប់ការពារិកដែលត្រូវបានពារិក។

និងការបញ្ចប់ការពារិក។

- **Virus** : និងការបញ្ចប់ការពារិក និងការបញ្ចប់ការពារិក។ e.g. Email / File.

- **Worm** : និងការបញ្ចប់ការពារិក និងការបញ្ចប់ការពារិក។ e.g. Trojan.

- **Spyware** : និងការបញ្ចប់ការពារិក និងការបញ្ចប់ការពារិក។ e.g. key logger.

- **Botnet** : និងការបញ្ចប់ការពារិក និងការបញ្ចប់ការពារិក។

Denial of Services (DDoS) : និងការបញ្ចប់ការពារិក និងការបញ្ចប់ការពារិក។

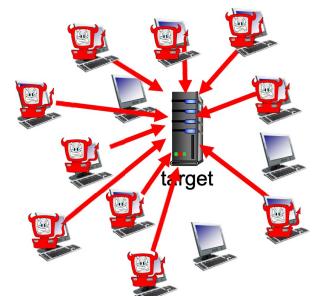
ឬស្ថាបន ឬសុវត្ថិភាព និងការបញ្ចប់ការពារិក។

Steps

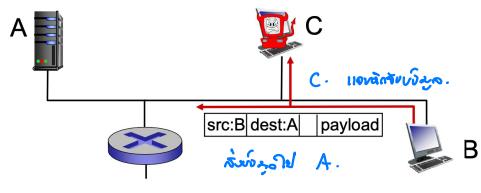
1. ការបញ្ចប់ការពារិក។

2. ការបញ្ចប់ការពារិក និងការបញ្ចប់ការពារិក។

3. ការបញ្ចប់ការពារិក និងការបញ្ចប់ការពារិក។

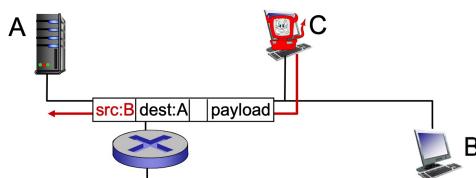


Sniff packet : និងការបញ្ចប់ការពារិក។



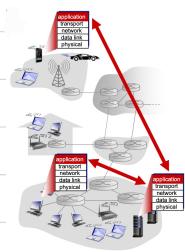
02: Application Layer.

malicious IP Spoofing: mislabeled packet fälschlicherweise.



Network Application: Interaktionen End-System zwischen Netzwerk und dem Network

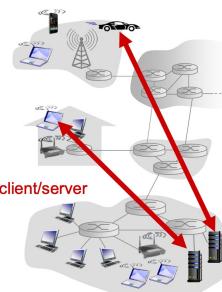
e.g. Web Server, E-mail, Browser



Auch von Internet Application.

- Client Server
- P2P (Peer-to-Peer)

Client-Server Architecture.

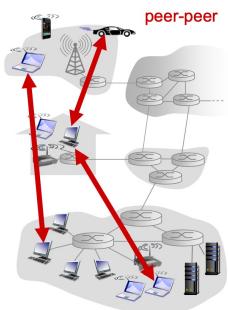


- Server**
- Programm host
 - IP Address nurmas fällin
 - zentralis Data Center
 - Scaling Fähigkeit

- Client**
- Anwendungsservern
 - mitsamtressourcenfördern
 - mit IP Address im Dynamic
 - Wiederholungsfähigkeit

P2P Architecture.

- Tiers Server möglich
- end System fördern können
- Self Scalability: Leistungsfähigkeit
- IP Address von Peer übernehmen



Process Communication, interprocess Process.

Process: Interaktionen End-System.

- inter-process communication zwischen Prozessen
- zwischen OS.

Client Process: Prozess am Clientseitigen Server

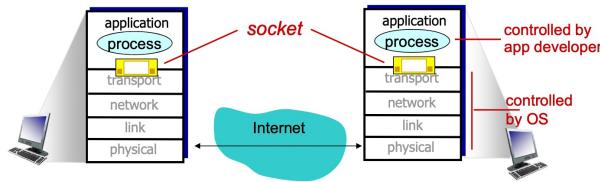
Server Process: Prozess am Clientseitigen Client.

Sockets

ສະແດງ ໂຄງນາໂຄງການ ສະແດງ ອັນດີ ທີ່ ດີວິເລີນ ຖ້າ ດີວິເລີນ ຕໍ່ ດີວິເລີນ

ອັນດີ e.g. TCP / UDP

- ມີສະແດງທີ່ ສະແດງ host ມີ Client



ນຳມືມາດຫຼຸດ Process (Process Addressing)

- ອັນດີ - ສະແດງ Process ມີວິທີ identify IP Address Port number.
- ມີຫຼັກ host ຢັດ IP ມີ 32 bit ພິບຂອງມີຫຼັກ.
- ມີຫຼັກ Process ສະແດງໂທນີ້ ນີ້ໄດ້ຮັດເຫັນໄດ້.

TCP Services.

- Reliable transport : ຝ່າຍຕົວໜ້າ ຢຶດເລືອດ.
- Flow control : ມີມາດອຸປະກອດໃຫ້ລົງຈະບົບໄດ້ ມີນີ້

ທີ່ ອົນຫວັງ.

- Congestion control : ມີມາດອຸປະກອດ ອົນຫວັງຜົນກົດ overflown.
- Connection-oriented : ມີມາດຕິດຕະຫຼອດຕືມກີ່ ໃນ ມີພົມການຂົ້ນຫຼັກ host ມີ Client ມີກົດກົດຫຼັກ.

UPP Services.

- ISO's TCP ຮັດໃດ Reliable Data transfer.
- TCP Flow / Congestion control.

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Securing TCP Connection. SSL ສູ່ Application Layer.

- TCP / UDP ຖັນຍາກົດຫຼັກ

- ຖັນຍາກົດຫຼັກ SSL ຖັນຍາກົດຫຼັກ - ສູ່

Web and HTTP.

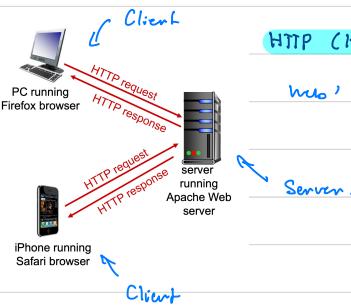
- Web page ລົດໃດ Object

- URL Object or URL (Uniform Resource Locator) ປົກສອງ ດິນີ້ວິຊາ ດີວິເລີນ.

www.someschool.edu/someDept/pic.gif

host name

path name



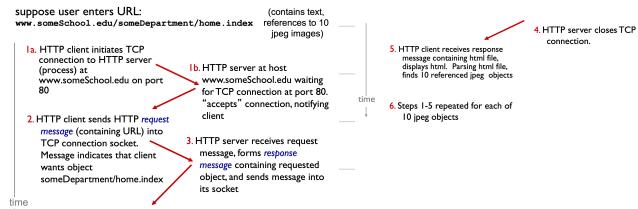
HTTP (Hypertext transfer protocol)

Web = application layer protocol.

HTTP Connections

- non-persistent HTTP : ສະແດງ object ແລ້ວ ສະແດງ connection.
- persistent connection.

ມີ download ອະນຸຍາວ ສະແດງ non-persistent connection.



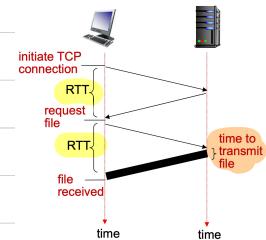
Time RTT (Round trip time)

ຫຼັກທີ່ ອົນຫວັງ - ຮັນໄວ້ພັກເລີກ.

ສິນເກີດ non-persistent HTTP Response

Time = $2 \times RTT + \text{File Transmission}$

Time ($\frac{\text{ວິທະຍາດ}}{\text{ອົນຫວັງ}}$)



- persistent connection : ມີຫຼັກ object ສະແດງໃດໆ

1 connection.

HTTP request message.

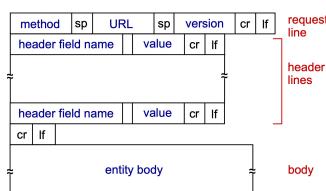
2 elements of HTTP

- request

- response

request line (GET, POST, HEAD commands)
header lines
carriage return, line feed at start of line indicates end of header lines

GET /index.htm HTTP/1.1\r\nHost: www.net.cs.umass.edu\r\nUser-Agent: Firefox/3.6.10\r\nAccept: text/html,application/xhtml+xml\r\nAccept-Language: en-us,en;q=0.5\r\nAccept-Encoding: gzip,deflate\r\nKeep-Alive: 115\r\nConnection: keep-alive\r\n\r\n



ຕົວຢ່າງ HTTP request message.

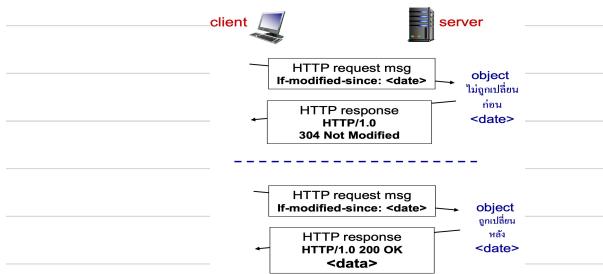
Condition GET : ~~request~~ request ~~from~~ Origin server ~~from~~

និងនៅលើ Cache ផ្សាយ.

- Total Delay (Transmission + Propagation) (D_{trans})
 - Propagation delay along link.

- Cache : սպառակ կամք : If-modified-since : <date>
- server : զետեղություն (object) reservationին Client ըստօպուր

my server HTTP/1.0 304 Not Modified.

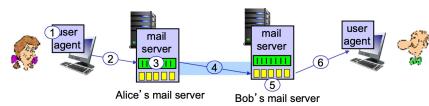


3 Phases of transfer

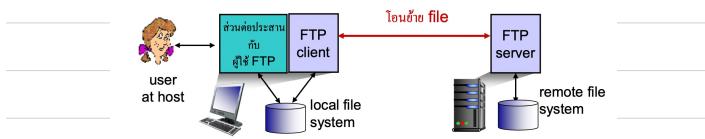
1. handshaking (greeting)
 2. transfer of message.
 3. closure.

Example of Alice's signature 13010.

1. Alice's User Agent initiates Email Transaction with Bob.
 2. Alice's User Agent connects Mail Server managed by the organization of Bob.
 3. SMTP Client Alice initiates TCP Connection with Mail Server of Bob.
 4. SMTP Client Alice sends Message via TCP Connection.
 5. Mail Server uses Bob's IP address to forward message via SMTP to Alice's IP.
 6. Bob receives message from Alice via User Agent.



FTP: File Transfer Protocol



- 9096 port -> 7000, 1024-1024 Port
 - Client / Server
 - FTP : RFC 959 / Port 20

Electronic Mail (Email)

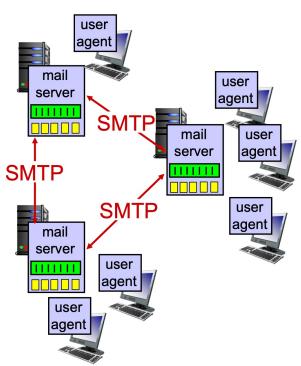
៣ សំណងជនុសាស្ត្រ

- User agents
 - mail servers
 - simple mail transfer protocol (SMTP)

Pont : 25 (SMTP)

User Agent Persistent.

- a.k.a. "mail reader"
 - ~~now~~, ~~then~~, ~~choose~~ mail
 - ~~choose~~ open - ~~use~~ read the mail when



Mail Servers - mailbox សេវាសម្រាប់ផ្តល់ព័ត៌មាន

- ការទិន្នន័យរបស់វាមួយនាក់ នៅតីប៊ូតិ៍.
 - មានចំណាំរបស់វានៅក្នុង 128 ពាណិជ្ជកម្ម ដើម្បីរួម SMTP Protocol
 - Client : ស៊ីតីឡាតិវិជ្ជកម្មរបស់ខ្លួន
 - Server : ប្រគល់ពាណិជ្ជកម្ម នៃការបញ្ចូនទំនាក់ទំនង.

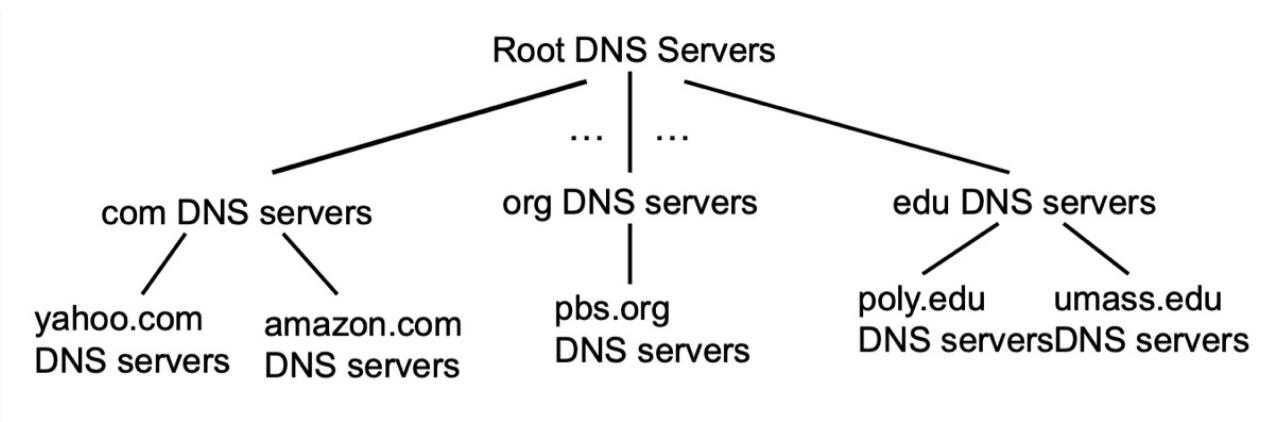
DNS: Domain Name System นำหน้าที่แปลงจาก Hostname ไปยัง IP Address

- แทนที่เราจะต้องมานั่งจำหมายเลข IP Address ของแต่ละเว็บไซต์ DNS จะทำหน้าที่จำ และแปลงให้เราเอง เราแค่ใช้ hostname
- e.g. Google.com → IP Address

ทำไม DNS ถึงไม่ควบคุมแบบรวมศูนย์

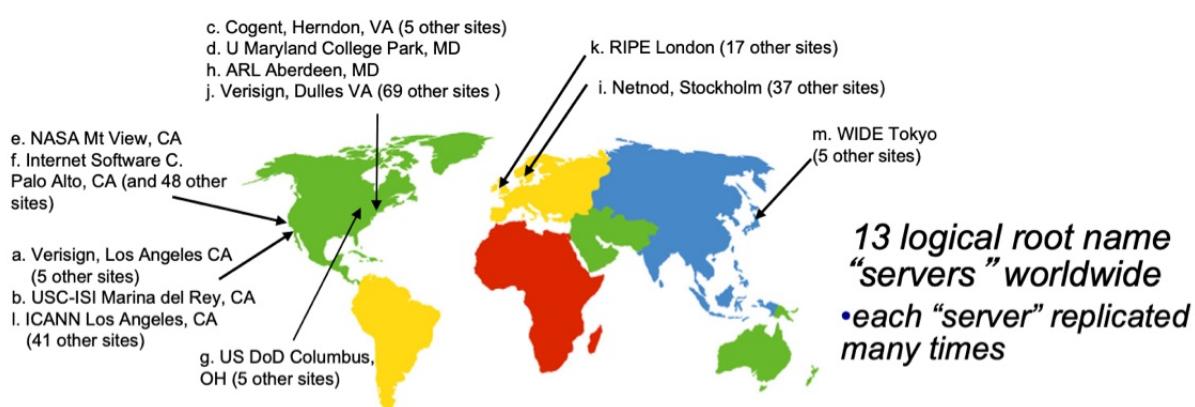
- ป้องกันข้อผิดพลาดเมื่อมี DNS ไดเกิดข้อผิดพลาด ตัวอื่นก็ยังสามารถทำงานได้ปกติ
- ลดการใช้ traffic ไม่ให้ traffic มันวิ่งไปทาง DNS ตัวเดียวมากเกินไป
- ทำให้การดูแลรักษาและการ Scaling ในอนาคตทำได้ง่ายและสะดวกมากขึ้น

ลำดับการเรียกหาข้อมูลภายใน DNS



- Root DNS Server
- TLD (Top-Level Domains DNS) e.g. .com / ac.th / .net
- Domain DNS (Authoritative) e.g. kmitl / google / pornhub

DNS (Domain Name System): Root name servers



เป็นการที่ local name server ไม่สามารถทำการ resolver ได้

- Root name server จะทำการติดต่อ Authoritative Name Server ในการ mapping ระหว่าง hostname และ IP Address

TLD, Authoritative Servers

Top-Level Domain (TLD) Servers

- เป็น DNS ของ TLD เช่น org, net, edu, ac, ... and all top-level country domain e.g. th, uk, fr
- .com TLD ดูแลรับผิดชอบโดย Network Solutions
- .edu TLD ดูแลรับผิดชอบโดย Educause

Authoritative DNS Server

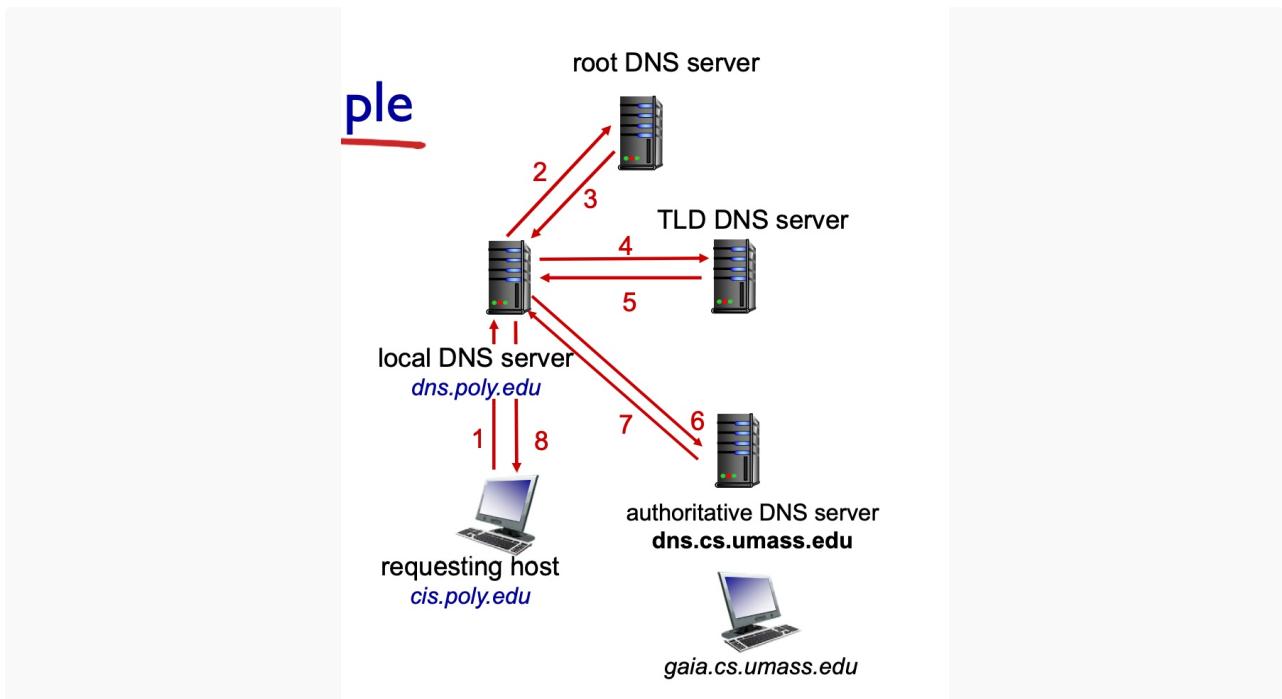
- องค์กรนัดแจ้งการ DNS Server ของตัวเอง

Local DNS Name Server

- เป็น DNS Server ที่จัดตั้งโดยแต่ละ ISPs เรียกว่า Default Name Server
- โดยปกติเมื่อไม่ได้กำหนดค่า DNS ตัว hostname จะถูก mapping ที่ Local DNS เป็นที่แรก
- มี Local Cache ในตัว

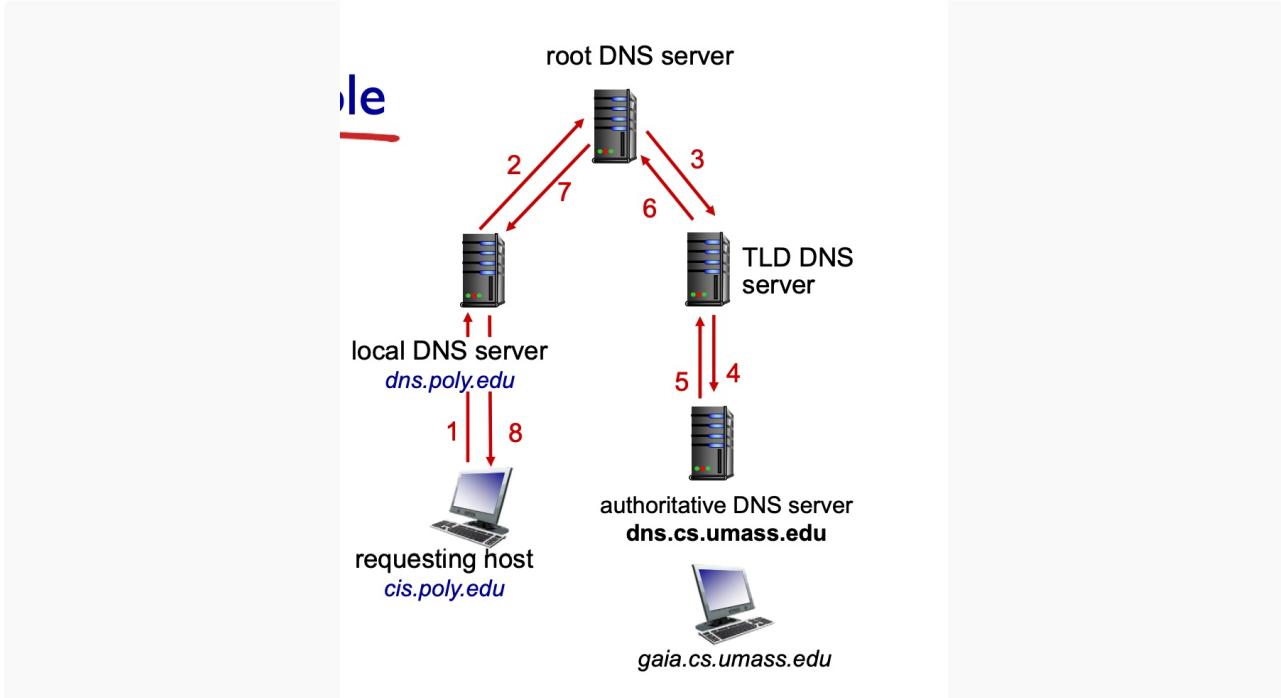
DNS Name Resolution Example

Host: cis.poly.edu ต้องการหา IP Address ของ gais.cs.umass.edu



- **แบบ Iterated Query**
 - ส่งไปตามที่ Local DNS Server ก่อนเป็นอันดับแรก (*dns.poly.edu*)

- ถ้ายังไม่ได้ IP Address ไปหาที่ root DNS Server ต่อ
- จากนั้นไป TLD DNS Server
- จากนั้นถ้ายังไม่เจออีกไป Authoritative DNS Server (dns.cs.umass.edu)
- พอดีเลข IP Address แล้ว วิ่งกลับมา



- **Recursive Query**

- ส่งไปถามที่ Local DNS Server ก่อนเป็นอันดับแรก (*dns.poly.edu*)
- แล้วส่งเป็น Chains ต่อๆ กันไปเรื่อยๆ ที่ DNS ตัวอื่นๆ
- พอดีเลข IP Address แล้ว วิ่งกลับมา

DNS: Caching, updating records

- เมื่อ DNS Server ทำการ mapping hostname to IP Address และ จะทำการ Caching ผลลัพธ์นั้นไว้ เมื่อมีการร้องขอครั้งต่อไป จะทำการส่งผลลัพธ์ที่ทำการ Caching ไว้ออกไป
- แต่ละ Cache มี TTL (Time to Live ระยะเวลาที่จะเก็บ Cache ของ Record ไว้ (Default = 1 Hr))
- Cached สามารถหมดอายุได้ (Best offert name-to-address translation)
 - ถ้ามีการเปลี่ยนแปลง hostname ข้อมูลที่ถูก Cache ไว้จะไม่ถูกเปลี่ยนแปลงจนกว่าจะหมดเวลา TTL
- การแก้ไข ถูกเสนอตอน IETF Standart RFC 2136

DNS Records

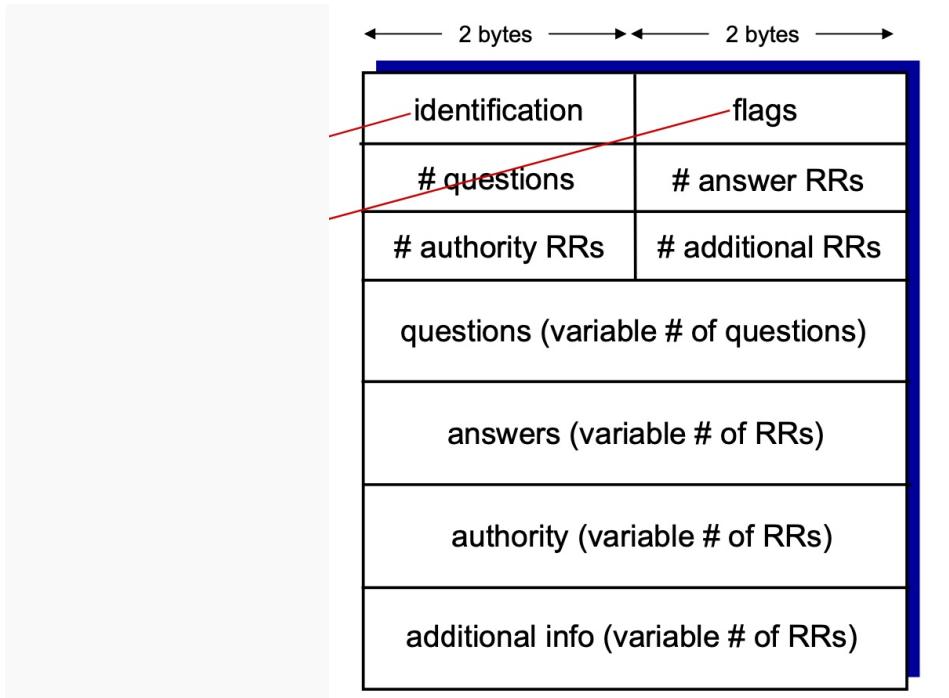
DNS: เหมือนฐานข้อมูลที่เก็บข้อมูล IP Address ของทุก hostname เอาไว้ (RR) \rightarrow (name, value, type, until)

- **Type: A**

- name is hostname
- value is IP Address

- **Type: CNAME**
 - name is alias name
 - www.ibm.com is really servereast.backup2.ibm.com
 - value is canonical name → ชื่อที่ใช้เป็นมาตรฐาน (e.g. กำหนด CNAME ให้ [api.aaa.com](#) ชี้ไปที่ [ccc.com](#) เมื่อเข้าเว็บด้วย [api.aaa.com](#) มันก็จะวิ่งไปที่ [ccc.com](#) แล้วแสดงหน้าเว็บนั้นขึ้นมาทันที)
- **Type=NS**
 - name is domain (e.g. facebook.com / google.com)
 - value is hostname of authoritative name server (facebook.com → facebook)
- **Type: MX**
 - value is name id mail server

Message: DNS Protocol



- รูปแบบการ ส่ง (query) และ ข้อความตอบกลับ (reply) อยู่ในรูปแบบเดียวกันทั้งหมด
- **Message Header** ประกอบไปด้วย
 - identification number (16 bits)
 - flags ใช้ในการ query หรือ reply
 - recursion desired / available
 - reply is authoritative
- **Message Body** ประกอบไปด้วย
 - questions (name, type and fields)
 - RRs ภายใน response สำหรับการ query
 - records for authoritative servers
 - additional information that client may use it.

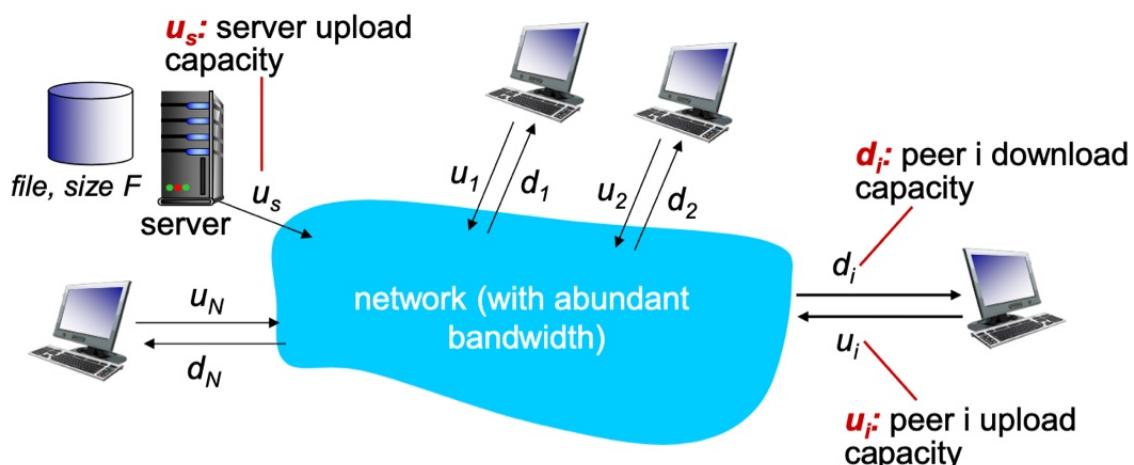
การเพิ่ม records (hostname) ใหม่ลงใน DNS Server

- ทำการลงทะเบียน Domains ใหม่ที่ DNS Registrar (e.g. Network Solutions)
- ทำการเพิ่ม name, IP Address ของ Authoritative name server (Primary and Secondary)
- เพิ่ม RRs ลงไว้ใน DNS ของ Top-Level Domain (TLD)
- สร้าง Authoritative Server (Type A → www.example.com) และ (Type MX → example.com)

การโจมตี DNS Server

- DDoS (Denial of Services)**
 - การทำให้ DNS Server ปลายทางมี Traffic จำนวนมากวิ่งเข้ามานั่นเอง
- Redirect Attacks**
 - Man-in-Middle: การปลอมแปลง DNS ให้ไปยังปลายทางที่ Hackers กำหนด
 - DNS Poisoning: การโจมตีโดยอาศัยการ Cache ของ DNS Server

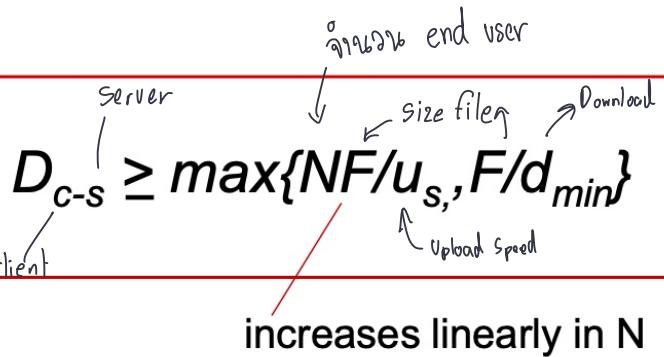
File distribution: Client-Server vs P2P



Client-Server transmission

- server transmission:** ส่งไฟล์แบบเป็นลำดับ ต่อเนื่องกัน (Upload ไปยัง Client)
 - ระยะเวลาที่ใช้ในการส่ง 1 Copy: F/U_s
 - ระยะเวลาที่ใช้ในการส่ง N Copies: NF/U_s

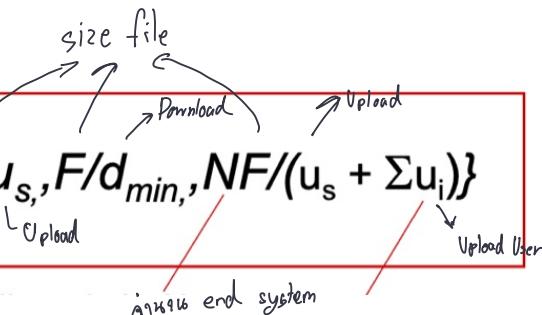
*time to distribute F
to N clients using
client-server approach*



- **Client:** ดำเนินการดาวน์โหลดไฟล์ที่ได้มาจากการ Server
 - D_{min} : ความเร็วในการดาวน์โหลดต่ำสุดของผู้ใช้
 - การหาระยะเวลาดาวน์โหลดที่สูงสุดของผู้ใช้: F/D_{min}

*time to distribute F
to N clients using
P2P approach*

$$D_{P2P} \geq \max\{F/u_s, F/d_{min}, NF/(u_s + \sum u_i)\}$$



Pure P2P Architecture

- ไม่จำเป็นต้องใช้ Server กลาง
- end system เชื่อมต่อ รับส่งข้อมูลได้โดยตรง
- แต่ละ peers สามารถเปลี่ยน IP Address ได้ตลอดเวลา
- ตัวอย่างของ P2P
 - File Distribution (Bit-Torrent)
 - Streaming (KanKan)
 - VoIP (Skype)

Example การคำนวณเปรียบเทียบระหว่างการส่งแบบ Client-Server และ P2P

File distribution time: Client-Server

$$F = 10 \text{ MB}$$

$$u_s = 2 \text{ MB/s}$$

$$d = 200 \text{ kB/s} = 0.2 \text{ MB/s}$$

$$u = 100 \text{ kB/s} = 0.1 \text{ MB/s}$$

$$N = 100$$

Client-server time $\geq 500 \text{ sec}$

$$\frac{NF}{u_s} = \frac{100 \cdot 10}{2} = 500 \text{ sec}$$

$$\frac{F}{d} = \frac{10}{0.2} = 50 \text{ sec}$$

scb 997-238902-1

File distribution time: Peer-to-Peer (P2P)

$$F = 10 \text{ MB}$$

$$u_s = 2 \text{ MB/s}$$

$$d = 200 \text{ kB/s} = 0.2 \text{ MB/s}$$

$$u = 100 \text{ kB/s} = 0.1 \text{ MB/s}$$

$$N = 100$$

$$\sum u_i = 10 \text{ MB/s}$$

Peer-to-peer

$$\frac{F}{u_s} = \frac{10}{2} = 5 \text{ sec}$$

$$\frac{F}{d} = \frac{10}{0.2} = 50 \text{ sec}$$

$$\frac{NF}{(u_s + \sum u_i)}$$

scb 997-238902-1

$$F = 10 MB$$

$$u_s = 2 MB/s$$

$$d = 200 KB/s = 0.2 MB/s$$

$$u = 100 KB/s = 0.1 MB/s$$

$$N = 100$$

$$\sum u_i = 10 MB/s$$

Peer-to-peer

$$\frac{F}{u_s} = \frac{10}{2} = 5 \text{ sec}$$

$$\frac{NF}{(u_s + \sum u_i)} = \frac{100 \cdot 10}{(2 + 10)} = 83.33 \text{ sec}$$

$$\frac{F}{I} = \frac{10}{0.2} = 50 \text{ sec}$$

scb 997-238902-1

การแปลงหน่วย

$$TB = 10^{12}$$

$$GB = 10^9$$

$$MB = 10^6$$

$$KB = 10^3$$

scb 997-238902-1

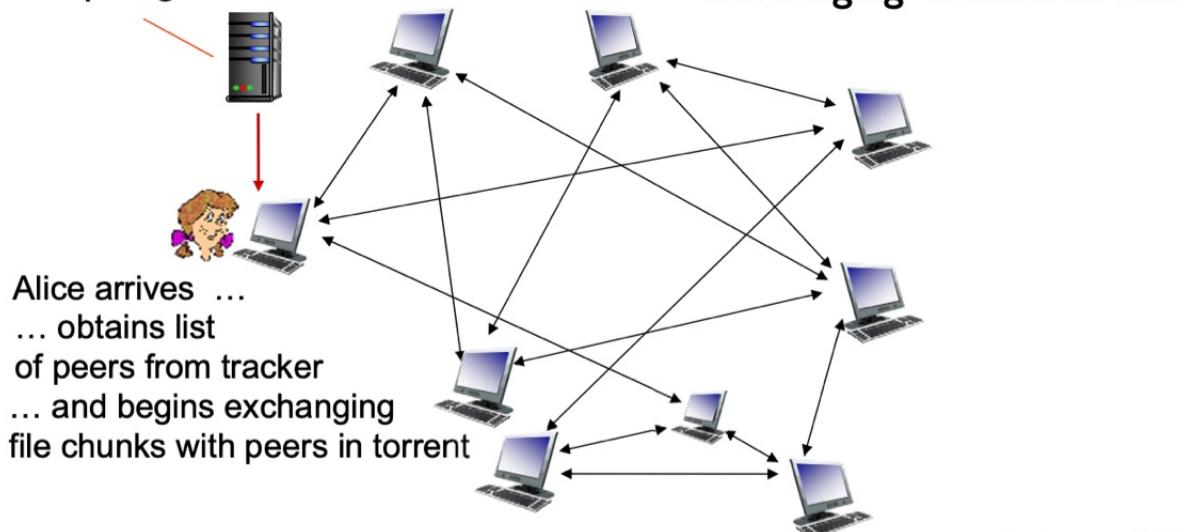
P2P File distribution: BitTorrent

- แบ่งไฟล์เป็นส่วนๆ (chunks) ส่วนละ 256kb และจากนั้น peers ทำการรับ-ส่งไฟล์
- Peers ที่เข้ามาเป็นส่วนร่วมจะเริ่มรับไฟล์ และกระจายให้กับคนอื่นต่อได้
- ใช้ Trackers ในการหาลิสต์ของ Peers ทั้งหมดที่มีในการกระจายไฟล์

T ติดตามไฟล์อย่าง

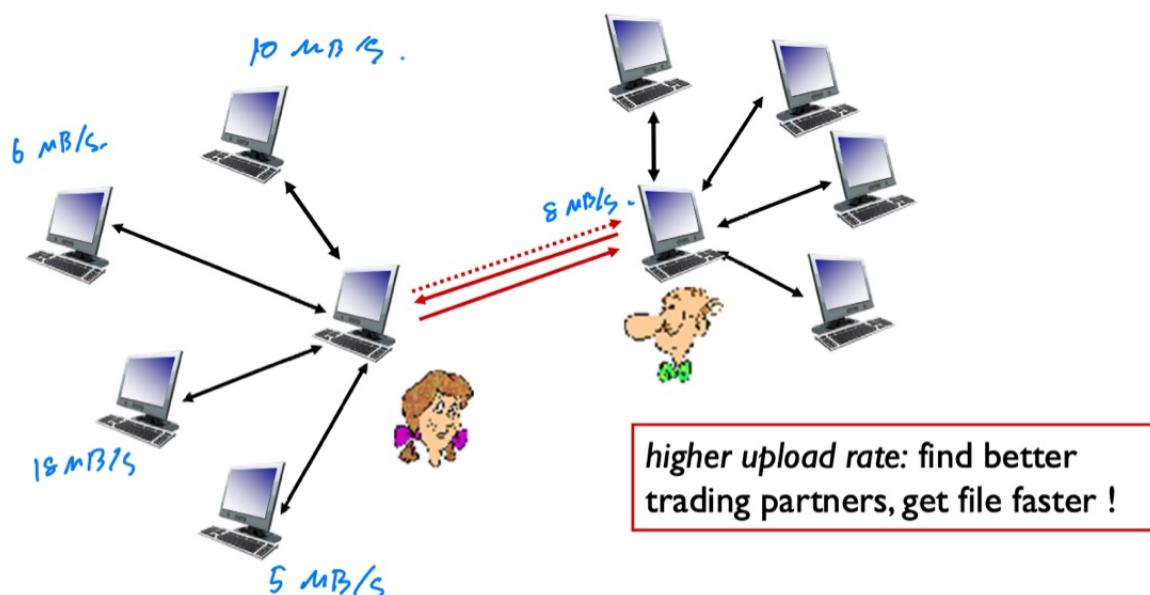
tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



BitTorrent

- Requesting Chunks
 - ในแต่ละเวลา แต่ละ peers มีไฟล์ที่แตกต่างกัน (ไม่ร่องงานทำ Done)
 - โดยปกติมักจะมีการเช็คใน Trackers เพื่อหา peers ที่กระจายไฟล์อยู่บ่อยๆ เพื่อให้ได้ความเร็วรวมกันทุกๆ peers เ酵ะที่สุดที่จะทำได้
- Tit-for-Tat
 - กระจายไฟล์ให้ Top 4 (Peers ที่ดีที่สุดจำนวน 4 คน) ที่ความเร็วสูงสุด
 - ประเมิน Top 4 ทุกๆ 10 วินาที
 - ทุกๆ 30 วินาที ทำการ random เลือก Top 4 ขึ้นมาใหม่ ถ้าเจอ Better Peers (ความเร็วในการ Upload สูงกว่า) จะเลือกมาเป็น Top 4



Video Streaming and CDNs: Context

Internet Bandwidth ส่วนมากถูกใช้ใน Video Traffic

- **Netflix:** 37% ของจำนวน Downstream (Residential ISP traffic) ~ ประมาณ 75 ล้าน
- **YouTube:** 16% ของจำนวน Downstream (Residential ISP traffic) ~ ประมาณ 1 ล้านล้านคน

Multimedia: Video

- Video: รูปที่มารวมกันหลายๆ รูป (e.g. 24 รูปภาพ ต่อวินาที)
- การส่งแบบ Spatial coding (within images): แทนที่จะส่งค่าสีขนาด N ไปที่เดียว จำทำ การส่งสีและจำนวนที่มีการใช้ช้าแบบไปด้วย ทำให้ลดจำนวน Bits ที่ส่งข้อมูล length Encode
- การส่งแบบ Temporal Coding (from one image to next) : แทนที่จะส่งแบบ completed frames จำส่งค่าที่ $i + 1$ และส่วนที่เปลี่ยนไปแค่นั้น
- CBR (Constant Bit Rate): Video encoding rate fixed
- VBR (Variable Bit Rate): Video encoding rate changes, as amount of spatial, temporal coding changes (เรทที่ใช้ในการเข้ารหัสวิดีโอเปลี่ยนไปตามค่าของ Spatial, Temporal Coding)

Streaming multimedia: DASH

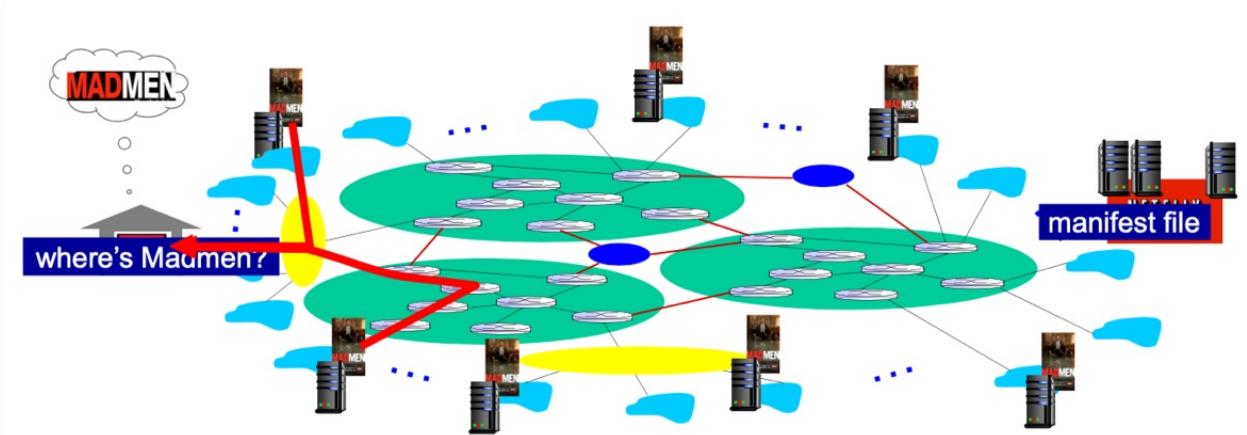
DASH: Dynamic, Adaptive Streaming over HTTP

- **Server**
 - แบ่งไฟล์ออกเป็นหลายๆ ส่วน (chunks)
 - แต่ละส่วน chunks เก็บและส่งที่ความเร็วต่างกัน
 - มี manifest file ที่ใช้บอก URL (ที่อยู่) ของแต่ละ chunks
- **Client**
 - มีการทำสอดความเร็วในการส่งข้อมูล Server-to-Client อยู่เป็นระยะๆ
 - ส่ง request ไปขอไฟล์ตามที่อยู่ใน Manifest ในแต่ละครั้ง
 - เลือกอัตราการ coding ที่ดีที่สุดตามความเร็วในการส่ง (bandwidth) ณ ขณะนั้น
 - สามารถเลือกอัตราการเข้ารหัส (spatial และ temporal) ที่ต่างกันได้ ในแต่ละช่วงเวลา ขึ้นอยู่กับความเร็วในการเชื่อมต่อขณะนั้น, Temporal \rightarrow ใช้ Bandwidth ที่น้อยกว่า
 - Intelligence ที่ผ่าน Client เป็นคนตัดสินใจว่า
 - When: เมื่อไหร่ที่จะมีการขอ chunks ใหม่
 - What encoding rate: ตัวเข้ารหัสแบบไหนที่ควรเลือกใช้
 - Where: จะร้องขอ chunks ที่ไหน (โดยปกติจะร้องขอไปที่ URL Server ที่อยู่ใกล้ที่สุด หรือมี Bandwidth ที่สำรองให้บริการสูงที่สุด)

Content Distribution Networks (CDN)

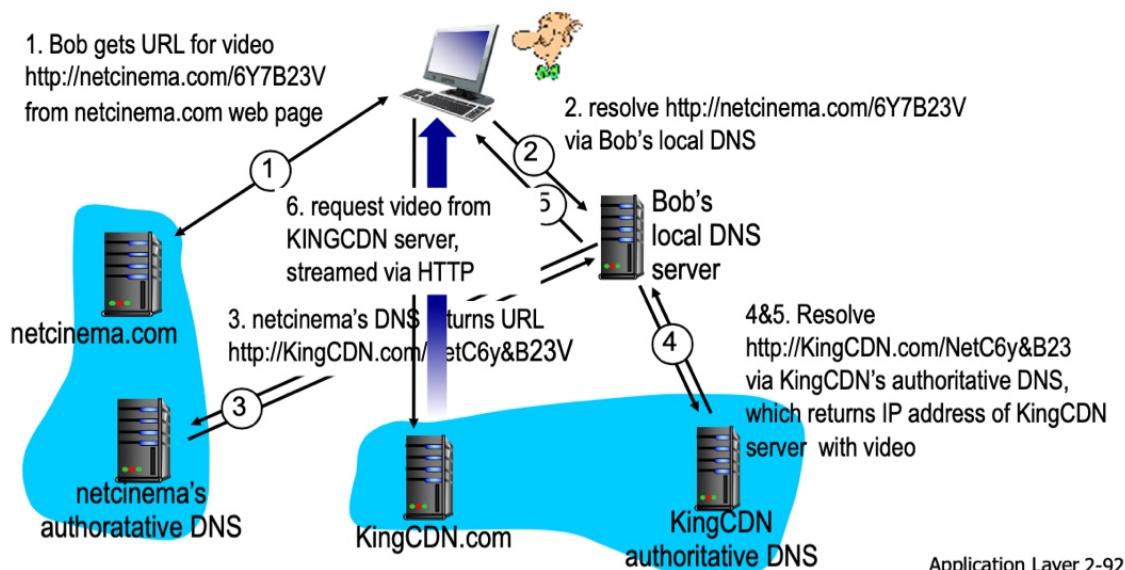
การให้บริการคอนเทนต์ไปยังเครื่อง Client หลายๆ เครื่องที่อยู่ต่างกันตามภูมิภาค (ในระยะที่ไกลจาก Server มาๆ)

- **CDN:** ทำสำเนาของคอนเทนต์ไว้ที่ CDN Node จากนั้นผู้ใช้ไม่จำเป็นต้องวิ่งไปไกล สามารถไปเอา Content ที่ CDN ที่ใกล้ที่สุดได้ทันที



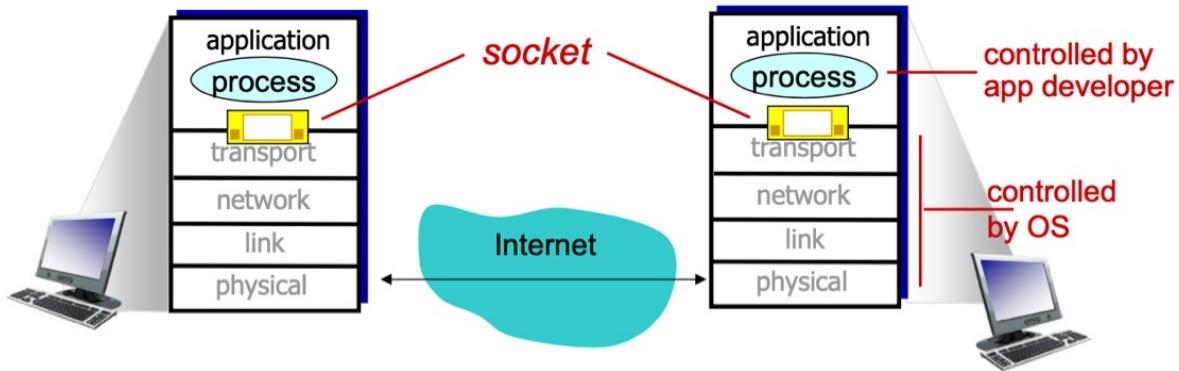
CDN Content Access: a closer look

1. Bob ทำการร้องขอวิดีโอที่ URL ปลายทาง
2. Local DNS ของ Bob ทำการ Resolve hostname-to-IP Address
3. DNS ทำการส่งค่า IP Address ของ Authoritative Server (netcinema.com → KingCDN.com) กลับมา
4. Authoritative ของ KingCDN ทำการส่งค่า IP Address ของ KingCDN กลับมา
5. วิดีโอลงไปหา Bob โดยผ่าน HTTP



Socket Programming (IP Address, Port Number)

อินเทอร์เฟซที่ใช้ในการสื่อสารระหว่างโปรแกรมคอมพิวเตอร์ต่างๆ ผ่านเครือข่าย (network) หรือการเชื่อมต่อระหว่างโปรแกรมบนเครื่องเดียวกัน ในคำอธิบายง่ายๆ คือ "ช่องทาง" ที่อนุญาตให้โปรแกรมสื่อสารกันได้ผ่านเครือข่ายหรือการเชื่อมต่อต่างๆ ซึ่งอยู่ในรูปของหมายเลข IP และพอร์ต (port number) เพื่อระบุเครื่องและบริการที่โปรแกรมต้องการเชื่อมต่อไป



2 ประเภทของ Sockets

- **TCP:** Reliable, byte-stream-oriented
- **UDP:** Unreliable datagram

TCP



UDP



Socket Programming with UDP

UDP (User Datagram Protocol) เป็นโปรโตคอลที่ใช้สำหรับการส่งข้อมูลระหว่างเครือข่าย แต่ไม่ได้มีความน่าเชื่อถือเทียบกับ TCP (Transmission Control Protocol) ที่มีการตรวจสอบและจัดการข้อผิดพลาดในการสื่อสาร สัญญาณ UDP มากถูกส่งไปทางเครือข่ายโดยไม่มีการตรวจสอบและแก้ไขข้อผิดพลาดหรือการเรียงลำดับข้อมูล เพื่อเพิ่มความเร็วในการส่งข้อมูลและลดการเชื่อมต่อที่ช้าช้อน

- UDP ไม่มีการสร้าง connection ระหว่าง client และ server
- ไม่มีการทำ handshaking (การส่งคำขอแล้วรอ ACK เพื่อเริ่มการสื่อสาร) ก่อนส่งข้อมูล

- การส่งข้อมูลอาจเกิด out-of-order ได้ (หมายถึงสถานการณ์ที่ข้อมูลที่ถูกส่งผ่านโปรโตคอล UDP มีการเรียงลำดับไม่ถูกต้องเมื่อถึงที่ปลายปลายของผู้รับ นั่นหมายความว่าข้อมูลไม่ถึงมาในลำดับที่ถูกส่งไป การเกิดเหตุการณ์นี้อาจเกิดขึ้นเมื่อมีการส่งข้อมูลผ่านทางเครือข่ายที่มีการเข้ามาแก้ไขเส้นทางการส่งข้อมูล หรือเมื่อมีการแทรกข้อมูลอื่นเข้ามาระหว่างการส่งข้อมูลที่ถูกส่งไป ทำให้ข้อมูลถูกส่งมาที่ผู้รับไม่ตามลำดับที่ส่งไป เรื่องนี้เกิดจากลักษณะของ UDP ที่ไม่มีการรักษาลำดับการส่งข้อมูลและไม่ตรวจสอบข้อผิดพลาด)

server (running on serverIP)

```
create socket, port= x:  
serverSocket =  
socket(AF_INET,SOCK_DGRAM)  
  
read datagram from  
serverSocket  
  
write reply to  
serverSocket  
specifying  
client address,  
port number
```

client

```
create socket:  
clientSocket =  
socket(AF_INET,SOCK_DGRAM)  
  
Create datagram with server IP and  
port=x; send datagram via  
clientSocket  
  
read datagram from  
clientSocket  
  
close  
clientSocket
```

Socket Programming with TCP

TCP (Transmission Control Protocol) เป็นโปรโตคอลในระบบคอมพิวเตอร์แบบเครือข่าย (network protocol) ที่ใช้สำหรับการส่งข้อมูลระหว่างอุปกรณ์หรือโปรแกรมที่ทำงานบนเครือข่ายต่างๆ โดย TCP เป็นหนึ่งในคอมโพเนนต์สำคัญของโปรโตคอลระดับแอปพลิเคชันในโครงสร้าง TCP/IP ที่ใช้สำหรับการสื่อสารข้อมูลระหว่างเครือข่ายคอมพิวเตอร์ โดย TCP มีลักษณะเป็น "connection-oriented" หรือต้องการการเชื่อมต่อก่อนการส่งข้อมูล

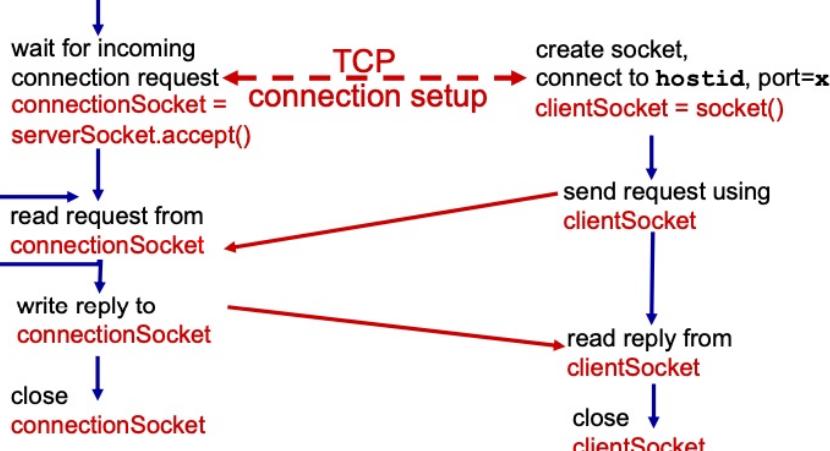
- ผู้ส่งต้องมีการติดต่อและสร้างการเชื่อมต่อ กับ Server ก่อนการส่ง (การทำ handshaking)
- Server ต้องพร้อมให้บริการและสร้าง Socket (door) ให้กับผู้ใช้เข้ามาเชื่อมต่อ
- ผู้ใช้ทำการติดต่อ กับ Server ปลายทางโดยการ
 - สร้าง TCP Socket โดยการใช้ IP Address และ Port Number
 - จากนั้น Server ทำการ established connection กับ client

server (running on hostid)

```

create socket,
port=x, for incoming
request:
serverSocket = socket()

```



client

ลักษณะสำคัญของ TCP ประกอบด้วย:

- **Connection-Oriented:** การสื่อสารด้วย TCP จะเริ่มต้นด้วยการเปิดการเชื่อมต่อ (connection) ระหว่างอุปกรณ์สองเครื่อง การเชื่อมต่อนี้จะมีการกำหนดเรียงลำดับข้อมูลและจัดการข้อผิดพลาดในการสื่อสาร
- **Reliable Data Transfer:** TCP ใช้การรับรองความถูกต้องและการเรียงลำดับข้อมูล ดังนั้นข้อมูลที่ส่งไปจะถูกตรวจสอบให้แน่ใจว่าถูกส่งถึงและถูกส่งไปตามลำดับ
- **Flow Control:** TCP มีการจัดการความเร็วในการส่งข้อมูล เพื่อไม่ให้ข้อมูลถูกส่งไปเร็วเกินไปและทำให้ผู้รับไม่สามารถประมวลผลข้อมูลที่เข้ามาได้อย่างมีประสิทธิภาพ
- **Congestion Control:** TCP มีการจัดการการจราจรที่เกิดขึ้นในเครือข่าย หากมีการรับข้อมูลเข้ามากเกินไป TCP จะลดความเร็วในการส่งข้อมูลเพื่อหลีกเลี่ยงปัญหาการแบ่งปันทรัพยากรเครือข่าย
- **Full Duplex Communication:** TCP สนับสนุนการสื่อสารแบบ full-duplex ซึ่งหมายความว่าข้อมูลสามารถถูกส่งและรับพร้อมกันในเวลาเดียวกัน
- **Segmentation and Reassembly:** TCP แบ่งข้อมูลที่จะส่งออกเป็นช่วงๆ และมีการประกอบข้อมูลใหม่เมื่อถึงที่ผู้รับ

Lecture 03: Introduction to Transport Layer

หน้าที่สำคัญของ Transport Layer

Layer 4 หรือที่เรียกว่า Transport Layer เป็นหนึ่งในชั้นของโครงสร้างโครงข่ายคอมพิวเตอร์ที่รวมถึงการสื่อสารและการจัดการข้อมูลระหว่างอุปกรณ์ที่เชื่อมต่อกันในเครือข่าย หน้าที่หลักของ Transport Layer คือการให้บริการสื่อสารระหว่างโปรแกรมและอุปกรณ์บนเครือข่าย โดยในโครงสร้าง TCP/IP หรืออินเทอร์เน็ต ชั้นนี้จะมีโปรโตคอลต่าง ๆ ซึ่งรวมถึง TCP (Transmission Control Protocol) และ UDP (User Datagram Protocol) เป็นต้น

หน้าที่หลักของ Transport Layer ประกอบด้วย:

- การจัดการการเชื่อมต่อ (**Connection Management**): ในกรณีของโปรโตคอลที่ต้องการการเชื่อมต่อ ก่อนการสื่อสาร เช่น TCP ชั้นนี้จะรับผิดชอบในการเปิดและปิดการเชื่อมต่อระหว่างอุปกรณ์ เพื่อให้สามารถสื่อสารระหว่างกันได้
- การจัดการการส่งข้อมูล (**Data Transmission**): Transport Layer รับผิดชอบในการแบ่งและรวมข้อมูลเพื่อส่งผ่านเครือข่าย โปรโตคอล ในชั้นนี้จะแบ่งข้อมูลให้เป็นเซกเมนต์ย่อย ๆ (segment) หรือ\data\ต้าแกรมย่อย ๆ ที่จะถูกส่งผ่านเครือข่าย และในกรณีของ TCP จะรับผิดชอบในการเรียงลำดับและการควบคุมความเร็วในการส่งข้อมูล
- การจัดการความเสถียรและความแม่น้ำเชื่อถือ (**Reliability and Integrity**): โปรโตคอล ในชั้นนี้มักจะมีการตรวจสอบและรับรองความถูกต้องของข้อมูลที่ถูกส่งไป โดยใช้การส่งคำสั่งสำหรับการยืนยันการรับข้อมูลหรือการแก้ไขข้อผิดพลาดที่เกิดขึ้นในการส่งข้อมูล
- การจัดการพอร์ต (**Port Management**): แต่ละโปรแกรมที่ทำงานบนอุปกรณ์เครือข่ายจะมีหมายเลขพอร์ตที่กำหนด เพื่อระบุว่าข้อมูลที่ถูกส่งไปเป็นของโปรแกรมใด ชั้น Transport ใช้พอร์ตในการจัดการปลายทางของข้อมูลเพื่อให้แต่ละโปรแกรมรับรู้ข้อมูลที่เป็นของตนเอง
- การจัดการการบันทึกข้อผิดพลาด (**Error Handling**): การจัดการข้อผิดพลาดในการส่งข้อมูล เช่น การสูญหายข้อมูล หรือข้อมูลที่ถูกส่งมาไม่ถูกต้อง เป็นส่วนหนึ่งของหน้าที่ของ Transport Layer โปรโตคอล ในชั้นนี้อาจใช้การส่งคำสั่งเพื่อขอให้ข้อมูลถูกส่งซ้ำหรือส่งใหม่ในกรณีที่เกิดข้อผิดพลาด

Multiplexing and Demultiplexing

Multiplexing และ Demultiplexing เป็นกระบวนการที่เกิดขึ้นในชั้น Transport Layer เพื่อให้หลาย ๆ แอปพลิเคชันหรือโปรแกรมสามารถใช้งานบนเครือข่ายเดียวกันได้พร้อมๆ กัน โดยใช้รายการพอร์ต (Port) ที่เป็นตัวระบุการสื่อสารระหว่างแอปพลิเคชันหรือโปรแกรมต่างๆ ในเครือข่ายเดียวกัน ดังนั้นเมื่อข้อมูลถูกส่งผ่านเครือข่าย หนึ่งในกระบวนการที่สำคัญคือ การแยกและแจกแจงข้อมูลไปยังแอปพลิเคชันที่ถูกเรียกใช้งาน

- Multiplexing:

กระบวนการ Multiplexing เป็นการรวมรวมข้อมูลจากหลายแหล่ง (source) แล้วส่งผ่านเครือข่ายไปยังปลายทางเดียว โดยใช้รายการพอร์ตเพื่อแยกและรับข้อมูลที่ถูกส่งไปเป็นของแอปพลิเคชันหรือโปรแกรมใด ดังนั้นหลาย ๆ แอปพลิเคชันสามารถใช้งานบนเครือข่ายเดียวกันได้โดยไม่มีความขัดแย้ง

- Demultiplexing:

กระบวนการ Demultiplexing เป็นการแยกและรับข้อมูลที่ถูกส่งมาผ่านเครือข่ายในรูปแบบของพอร์ตที่ถูกกำหนด โดย Transport Layer ที่รับข้อมูลจะทำหน้าที่ดูว่าข้อมูลเหล่านี้เป็นของแอปพลิเคชันหรือโปรแกรมใด แล้วนำข้อมูลนั้นไปส่งต่อไปยังแอปพลิเคชันที่เหมาะสมตามพอร์ตที่ระบุ

โดยสรุป การ Multiplexing จะรวมข้อมูลจากแหล่งต่างๆ เข้ามาในช่องทางเดียวกันเพื่อส่งผ่านเครือข่าย และการ Demultiplexing จะแยกและส่งข้อมูลต่อไปยังแอปพลิเคชันที่ถูกต้องตามพอร์ตที่ระบุ ดังนั้น กระบวนการนี้ช่วยให้แอปพลิเคชันหลาย ๆ รายการสามารถที่จะใช้งานบนเครือข่ายเดียวกันได้โดยมีการแยกและรับข้อมูลอย่างมีประสิทธิภาพและสามารถรับรู้ถึงแหล่งข้อมูลแต่ละตัวได้.

Port Number

- เป็นเลข 16-bit
- ใช้คู่กับหมายเลข IP Address ในการระบุ Process
- Range 3 ประเภทของ Port Number
 - System or Well-known port** → ใช้บ่อยสุด เป็นที่รู้จัก และใช้โดย OS (0 - 1023)
 - User or Registered ports** → ผู้ใช้เป็นคนกำหนดให้ Process ต่างๆ (1024 - 49151)
 - Dynamic ports** → OS Random ในการใช้งาน (49152 - 65535)

หน้าที่ของ Port Number คือ:

- การระบุแอปพลิเคชันหรือโปรแกรม: หมายเลขพอร์ตจะช่วยในการระบุแอปพลิเคชันหรือโปรแกรมที่ทำงานบนเครือข่าย เมื่อมีการสื่อสารระหว่างอุปกรณ์ พอร์ตจะระบุว่าข้อมูลนั้นเป็นของแอปพลิเคชันหรือโปรแกรมใด
- การแยกและรับข้อมูล: เมื่อข้อมูลถูกส่งผ่านเครือข่าย พอร์ตจะช่วยในการแยกและรับข้อมูลนั้นเป็นของแอปพลิเคชันหรือโปรแกรมใด และนำข้อมูลไปส่งให้แอปพลิเคชันที่ถูกต้อง
- การเชื่อมต่อแบบพอร์ต: พอร์ตช่วยในการเชื่อมต่อและการสื่อสารระหว่างแอปพลิเคชันหรือโปรแกรมบนอุปกรณ์ต่างๆ ในเครือข่าย แต่ละแอปพลิเคชันหรือโปรแกรมจะใช้พอร์ตที่แตกต่างกันเพื่อทำการเชื่อมต่อและสื่อสาร
- การบริหารจัดการทรัพยากร: ในการใช้พอร์ตที่มีหลาย แอปพลิเคชันหรือโปรแกรมที่ทำงานบนอุปกรณ์เดียวกัน การใช้พอร์ตช่วยในการจัดการทรัพยากรเครือข่าย เพื่อให้แต่ละแอปพลิเคชันได้รับสิทธิ์ในการใช้ทรัพยากรตามความจำเป็น

Lecture 04: Principle of Reliable Data Transfer

Reliable Data Transfer

"**Sender know that the transmission is error**" Reliable Data Transfer (RDT) เป็นกระบวนการในการส่งข้อมูลระหว่างอุปกรณ์หรือโปรแกรมบนเครือข่ายให้มั่นคงและนำเข้าถูกต้อง โดย RDT มุ่งเน้นในการรับรองความถูกต้องและความเรียบลื่นของข้อมูลที่ถูกส่งผ่านเครือข่าย เมื่อเครือข่ายอาจมีข้อผิดพลาดหรือการสูญหายข้อมูลก็ตาม

คุณสมบัติหลักของ Reliable Data Transfer ได้แก่:

- **Acknowledgment and Retransmission (ARQ):** ในกรณีที่ข้อมูลถูกส่งออกไปแล้วไม่ได้รับการยืนยันจากอุปกรณ์ปลายทางหรือเกิดข้อผิดพลาดในการรับข้อมูล ระบบ ARQ จะร้องขอให้ข้อมูลถูกส่งซ้ำเพื่อให้แน่ใจว่าข้อมูลถูกส่งถูกต้อง
- **เลขลำดับของข้อมูล อุปกรณ์ปลายทางจะใช้เลขลำดับของข้อมูลที่ถูกส่งเข้ามาให้ถูกต้อง**
- **Flow Control:** การควบคุมการไหลของข้อมูลเพื่อให้ไม่เกิดการรับข้อมูลเข้ามาเร็วเกินไปที่อุปกรณ์ปลายทาง แบบที่ไม่สามารถประมวลผลข้อมูลได้เร็วพอ
- **การตรวจสอบความถูกต้องและการแก้ไขข้อผิดพลาด (Error Detection and Correction):** กระบวนการตรวจสอบและแก้ไขข้อผิดพลาดเบื้องต้นเพื่อรับรองว่าข้อมูลถูกส่งถูกต้อง ในบางกรณีอาจมีการแนบพิลต์เช็คชั้ม (Checksum) หรือการใช้ข้อความไปพร้อมกับเลขอย่างเพื่อตรวจสอบความถูกต้องของข้อมูล

เพื่อให้การสื่อสารระหว่างอุปกรณ์หรือโปรแกรมบนเครือข่ายเป็นไปอย่างมั่นคงและนำเข้าถูกต้อง กระบวนการ Reliable Data Transfer จะช่วยแก้ไขข้อผิดพลาดและให้การส่งข้อมูลที่มีประสิทธิภาพในสภาวะที่เครือข่ายมีการสูญหายข้อมูลหรือข้อผิดพลาดในการสื่อสาร โดยแนวคิดของ RDT สามารถปรับใช้กับหลายๆ โปรโตคอลและรูปแบบการสื่อสารต่างๆ อาทิเช่น TCP และ UDP ในโครงสร้างของโปรโตคอล TCP/IP หรืออินเทอร์เน็ต

Automatic Repeat Request (ARQ)

เป็นเทคนิคในการจัดการแก้ไขข้อผิดพลาดในการสื่อสารระหว่างอุปกรณ์หรือโปรแกรมบนเครือข่าย โดย ARQ ใช้หลักการของการส่งข้อมูลจนกว่าจะได้รับการยืนยันจากอุปกรณ์ปลายทางว่าข้อมูลถูกส่งถูกต้อง หากไม่ได้รับการยืนยันหรือพบข้อผิดพลาดในการส่งข้อมูล ARQ จะใช้เทคนิคในการทำให้ข้อมูลถูกส่งซ้ำอีกครั้งเพื่อให้แน่ใจว่าข้อมูลถูกส่งถูกต้อง อุปกรณ์ปลายทางเรียบร้อยและถูกต้อง

ARQ มีหลายโหมดหรือแบบที่ใช้ในการทำงาน เช่น

Stop-and-Wait ARQ

โหมดนี้ทุกครั้งที่ส่งข้อมูลแล้ว อุปกรณ์ส่งจะรอรับการยืนยันการรับข้อมูลจากอุปกรณ์ปลายทางก่อนที่จะส่งข้อมูลครั้งถัดไป เป็นหนึ่งในเทคนิคของการจัดการความเรียงลำดับและความถูกต้องของข้อมูลในการสื่อสารระหว่างอุปกรณ์หรือโปรแกรมบนเครือข่าย โดยเฉพาะในระบบที่มีการส่งข้อมูลผ่านเครือข่ายที่อาจมีข้อผิดพลาดหรือการสูญหายข้อมูลเกิดขึ้น

คุณสมบัติหลักของ Stop-and-Wait ARQ คือ:

- **Acknowledgment and Retransmission:** เมื่ออุปกรณ์ส่งข้อมูลไปยังอุปกรณ์ปลายทางแล้ว อุปกรณ์ปลายทางจะส่งคำสั่ง ACK (Acknowledgment) กลับมายืนยันว่าข้อมูลถูกรับและถูกต้อง หากข้อมูลถูกส่งมาไม่ถูกต้องหรือไม่ถูกรับ อุปกรณ์ปลายทางจะไม่ส่ง ACK กลับ ซึ่งจะทำให้อุปกรณ์ส่งรู้ว่ามีข้อผิดพลาดเกิดขึ้น
- **การรอเวลา (Timeout):** เมื่ออุปกรณ์ส่งข้อมูลแล้ว จะกำหนดเวลาในการรอรับ ACK กลับจากอุปกรณ์ปลายทาง หากเวลาผ่านไปแล้วยังไม่ได้รับ ACK กลับมา อุปกรณ์ส่งจะถือว่ามีข้อผิดพลาดเกิดขึ้นและจะทำการส่งข้อมูลซ้ำ
- **การส่งข้อมูลชิ้นถัดไปหลังจาก ACK:** เมื่อได้รับ ACK ยืนยันว่าข้อมูลถูกส่งถึงและถูกต้อง อุปกรณ์ส่งจะกำหนดให้ส่งข้อมูลชิ้นถัดไป และกระบวนการนี้จะทำซ้ำไปเรื่อยๆ

เทคนิค Stop-and-Wait ARQ มีข้อดีคือง่ายต่อการออกแบบและการปรับใช้ แต่ก็มีข้อเสียคือมีการใช้เวลาในการรอ ACK และการทำงานของระบบจะช้าลงเนื่องจากต้องรอ ACK ก่อนที่จะส่งข้อมูลชิ้นถัดไป ดังนั้น Stop-and-Wait ARQ มักถูกนำไปใช้ในเครือข่ายที่มีความเสถียรและความน่าเชื่อถือสูง เช่น ในการสื่อสารผ่านเชื่อมต่อและเส้นทางที่มีประสิทธิภาพสูง

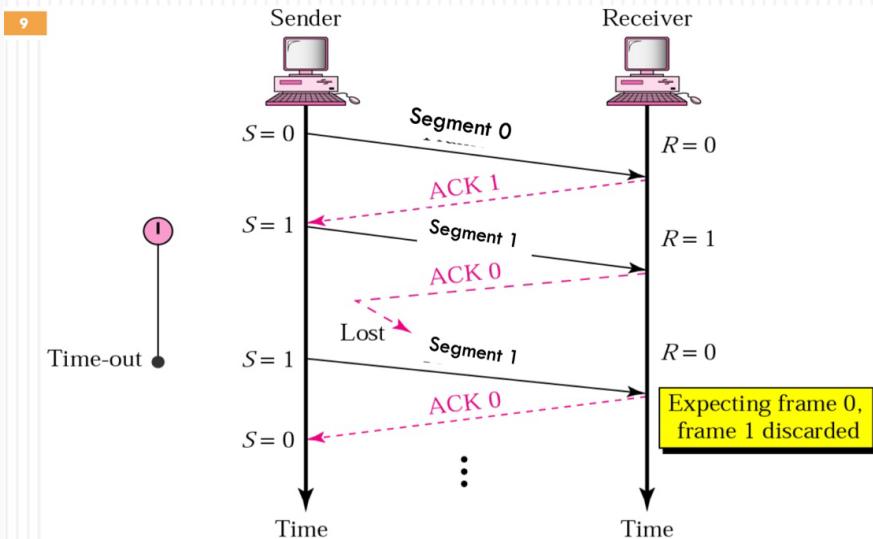
ข้อดีของ Stop-and-Wait ARQ

- ง่ายต่อการออกแบบและการปรับใช้: Stop-and-Wait ARQ เป็นเทคนิคที่ง่ายต่อการออกแบบและการปรับใช้อุปกรณ์ส่งสามารถส่งข้อมูลไปแล้วรอรับ ACK กลับมาได้โดยง่าย
- การจัดการเหตุการณ์แบบเรียกเก็บ: อุปกรณ์ส่งจะรอรับ ACK กลับมาก่อนที่จะส่งข้อมูลชุดถัดไป ทำให้ง่ายต่อการจัดการเหตุการณ์เมื่อมีการสูญหายข้อมูลหรือข้อผิดพลาดในการรับ
- ไม่มีการนำทางแบบเลื่อนหน้าต่าง: ใน Stop-and-Wait ARQ, ไม่ต้องจัดการการนำทางแบบเลื่อนหน้าต่างหรือเลขอยู่เพื่อควบคุมการส่งข้อมูล ทำให้ง่ายต่อการจัดการ

ข้อเสียของ Stop-and-Wait ARQ

- ประสิทธิภาพในการใช้ทรัพยากรของเครือข่าย: เนื่องจากอุปกรณ์ส่งต้องรอรับ ACK กลับมาก่อนที่จะส่งข้อมูลชุดถัดไป จะทำให้ใช้ทรัพยากรของเครือข่ายอย่างไม่มีประสิทธิภาพ โดยเฉพาะในระบบเครือข่ายที่มีความเร็วสูง
- ความหน่วงในการสื่อสาร: การรับ ACK กลับมาก่อนที่จะส่งข้อมูลชุดถัดไปจะทำให้เกิดความหน่วงในการสื่อสาร ทำให้ประสิทธิภาพลดลงเมื่อเทียบกับเทคนิคอื่น ๆ
- ความช้าในการส่งข้อมูล: ในการสื่อสารที่เครือข่ายมีการสูญหายข้อมูลหรือเกิดข้อผิดพลาดในการรับข้อมูล อุปกรณ์ส่งจะต้องรอรับ ACK กลับมาก่อนที่จะส่งข้อมูลชุดถัดไป ทำให้ความเร็วในการส่งข้อมูลลดลง
- ไม่เหมาะสมสำหรับเครือข่ายที่มีความไม่เสถียร: หากมีการสูญหายข้อมูลบ่อยครั้งหรือการสื่อสารในเครือข่ายที่ไม่เสถียร การใช้ Stop-and-Wait ARQ อาจทำให้การสื่อสารช้าลงและเสียเวลาเนื่องจากต้องรอรับ ACK กลับมาก่อน

Stop-and-Wait ARQ: ACK Lost



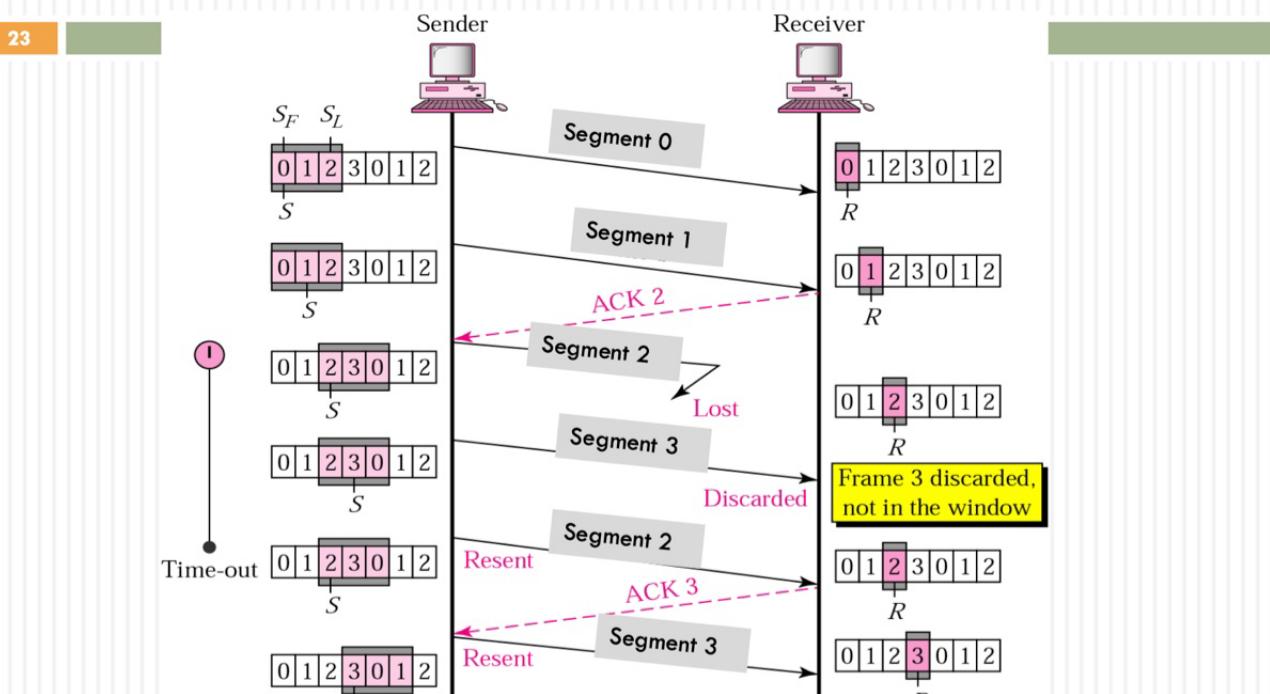
Go-Back-N ARQ

เป็นเทคนิคการจัดการความเรียงลำดับและความถูกต้องของข้อมูลในการสื่อสารระหว่างอุปกรณ์หรือโปรแกรมบนเครือข่าย โดยใช้การส่งข้อมูลเป็นชุดและการร้องขอให้ส่งข้อมูลข้ามในกรณีที่ข้อมูลถูกสูญหายหรือเกิดข้อผิดพลาดในการรับข้อมูล

คุณสมบัติหลักของ Go-Back-N ARQ คือ

- Sliding Window:** Go-Back-N ARQ ใช้ระบบ Sliding Window ในการจัดการข้อมูล ทึ้งอุปกรณ์ส่งและอุปกรณ์ปลายทางมีหน้าต่าง (Window) ที่กำหนดขนาดไว้ โดยอุปกรณ์ส่งสามารถส่งข้อมูลในหน้าต่างนั้นได้เท่านั้น
- Acknowledgment and Retransmission:** เมื่ออุปกรณ์ปลายทางรับข้อมูลถูกต้อง จะส่งคำสั่ง ACK (Acknowledgment) กลับมายืนยันว่าข้อมูลถูกรับ แต่หากเกิดข้อผิดพลาดในการรับข้อมูล อุปกรณ์ปลายทางจะไม่ส่ง ACK กลับมาและทำการร้องขอให้ส่งข้อมูลในหน้าต่างที่สูญหายกลับมา
- การรับข้อมูลต่อเนื่อง: หากข้อมูลถูกส่งมาแบบต่อเนื่องในหน้าต่าง (Window) เมื่ออุปกรณ์ปลายทางรับข้อมูลไปแล้ว เลี้ยว มันจะต้องรับข้อมูลในหน้าต่างถัดไปที่ยังไม่ได้รับข้อมูลมาก่อน ซึ่งทำให้เพิ่มประสิทธิภาพในการสื่อสาร
- การส่งข้อมูลข้าม: เมื่ออุปกรณ์ส่งข้อมูลแล้วไม่ได้รับ ACK ยืนยันหรือเกิดข้อผิดพลาดในการรับข้อมูล อุปกรณ์ส่งจะทำการส่งข้อมูลในหน้าต่างนั้นทั้งหมดใหม่อีกครั้ง (Go-Back) เพื่อให้แน่ใจว่าข้อมูลถูกส่งถึงและถูกต้อง

Go-Back-N ARQ มีข้อดีคือสามารถใช้ร่วมกับ Sliding Window เพื่อเพิ่มประสิทธิภาพการสื่อสาร แต่ข้อเสียคือหากมีข้อผิดพลาดเกิดขึ้นในการรับข้อมูล อุปกรณ์ปลายทางจะต้องทำการรับข้อมูลทั้งหน้าต่างนั้นใหม่ ทำให้การทำงานชัดขึ้นในกรณีที่เกิดข้อผิดพลาดติดต่อกันหลาย ๆ ครั้ง ดังนั้น Go-Back-N ARQ นักจะถูกใช้ในเครือข่ายที่มีการสูญหายข้อมูลน้อย และเสถียรสูง



ข้อดี-ข้อเสียของ Go-Back-N ARQ

ข้อดีของ Go-Back-N ARQ

- ประสิทธิภาพในการสื่อสาร: เนื่องจาก Go-Back-N ARQ ส่งข้อมูลเป็นชุดและรอ ACK ตามหน้าต่างการส่งข้อมูล การใช้ Sliding Window ช่วยเพิ่มประสิทธิภาพในการสื่อสาร และอุปกรณ์ส่งสามารถส่งข้อมูลต่อเนื่องได้โดยไม่ต้องรอสั่งส่งข้อมูลชิ้นถัดไปทีละชุด
- ง่ายต่อการออกแบบและการปรับใช้: การออกแบบและการปรับใช้ Go-Back-N ARQ จะง่ายกว่าเทคนิคอื่น ๆ เช่น Selective Repeat ARQ เนื่องจากมีหน้าต่างการส่งข้อมูลและการรับข้อมูลที่คงที่
- การเพิ่มประสิทธิภาพในการสื่อสารเมื่อเครือข่ายมีความเสถียรสูง: ในเครือข่ายที่มีการสูญหายข้อมูลน้อยและเสถียรสูง การใช้ Go-Back-N ARQ สามารถช่วยเพิ่มประสิทธิภาพในการสื่อสารได้

ข้อเสียของ Go-Back-N ARQ

- ข้อผิดพลาดแบบ Burst: หากมีการสูญหายข้อมูลในระยะเวลาสั้น ๆ (burst) จำนวนมาก เช่น ข้อผิดพลาดที่เกิดขึ้นในช่วงเวลาเดียวกัน อุปกรณ์ปลายทางจะต้องรับข้อมูลในหน้าต่างถัดไปใหม่ทั้งหมด ซึ่งอาจทำให้เสียเวลาและลดประสิทธิภาพ
- การใช้ทรัพยากรของเครือข่าย: 在การส่งข้อมูลใหม่หลังจากเกิดข้อผิดพลาด อุปกรณ์ส่งจะต้องส่งข้อมูลในหน้าต่างนั้นทั้งหมด ทำให้ใช้ทรัพยากรของเครือข่ายมากขึ้น
- ความเร็วในการตอบสนอง: หากอุปกรณ์ส่งต้องรอรับ ACK จากอุปกรณ์ปลายทางก่อนที่จะส่งข้อมูลชิ้นถัดไป อาจทำให้ความเร็วในการสื่อสารลดลง โดยเฉพาะในเครือข่ายที่มีการสูญหายข้อมูลบ่อยครั้ง
- การใช้พื้นที่เก็บข้อมูล: เนื่องจาก Go-Back-N ARQ จะต้องเก็บข้อมูลที่ส่งไปแล้วในหน้าต่างการส่งข้อมูล หากมีขนาดของหน้าต่างการส่งข้อมูลใหญ่ อุปกรณ์ส่งจะต้องใช้พื้นที่เก็บข้อมูลมากขึ้น

Selective Repeat ARQ

Selective Repeat ARQ (Automatic Repeat reQuest) เป็นเทคนิคการจัดการความเรียงลำดับและความถูกต้องของข้อมูลในการสื่อสารระหว่างอุปกรณ์หรือโปรแกรมบนเครือข่าย ในทางปฏิบัติ มีความคล้ายคลึงกับ Go-Back-N ARQ แต่มีความยืดหยุ่นกว่าในการรับข้อมูลที่ถูกส่งมาผิดลำดับหรือสูญหาย

คุณสมบัติหลักของ Selective Repeat ARQ คือ

- Sliding Window: เมื่อเทียบกับ Go-Back-N ARQ, Selective Repeat ใช้ระบบ Sliding Window ในการจัดการข้อมูลทั้งอุปกรณ์ส่งและอุปกรณ์ปลายทางมีหน้าต่าง (Window) ที่กำหนดขนาดไว้ แต่ใน Selective Repeat อุปกรณ์ปลายทางสามารถรับข้อมูลที่ถูกส่งมากกว่าหนึ่งชุด และสามารถรับชุดข้อมูลเฉพาะที่สูญหายหรือผิดลำดับได้
- Acknowledgment and Retransmission: อุปกรณ์ปลายทางจะส่งคำสั่ง ACK (Acknowledgment) กลับมาเมื่อได้รับข้อมูลที่ถูกต้อง แต่หากเกิดข้อผิดพลาดในการรับข้อมูล เช่น ข้อมูลสูญหายหรือผิดลำดับ อุปกรณ์ปลายทางจะส่งคำสั่ง NAK (Negative Acknowledgment) หรือเลขอย่างเพื่อร้องขอให้ส่งข้อมูลชุดนั้นใหม่
- การรับข้อมูลต่อเนื่อง: อุปกรณ์ปลายทางสามารถรับข้อมูลต่อเนื่องได้โดยไม่จำเป็นต้องรอรับข้อมูลชุดก่อนหน้าก่อน
- เลขอย่างและการจัดเรียงลำดับ: ใน Selective Repeat ARQ, เลขอย่างถูกใช้เพื่อระบุลำดับของข้อมูลที่ถูกส่ง และอุปกรณ์ปลายทางจะเก็บข้อมูลที่ถูกส่งมาและรับเพื่อจัดเรียงลำดับให้ถูกต้อง
- ใน Selective Repeat ARQ, การร้องขอให้ส่งข้อมูลข้าจจะเป็นเฉพาะชุดข้อมูลที่สูญหายหรือผิดลำดับเท่านั้น ไม่จำเป็นต้องส่งข้อมูลทั้งหน้าต่างหรือชุดข้อมูลทั้งหมดเหมือนใน Go-Back-N ARQ

ข้อดีของ Selective Repeat ARQ

- ประสิทธิภาพในการใช้ทรัพยากร: จะไม่ต้องทำการส่งข้อมูลทั้งหน้าต่างใหม่หากเกิดข้อผิดพลาด ทำให้ไม่ต้องใช้ทรัพยากรเครือข่ายมากนัก
- การใช้เวลาไม่น้อยกว่าเมื่อเทียบกับ Go-Back-N: เนื่องจากไม่ต้องส่งข้อมูลทั้งหน้าต่างใหม่เมื่อเกิดข้อผิดพลาด จึงทำให้การใช้เวลาไม่น้อยกว่าในการรับข้อมูลข้า

ข้อเสียของ Selective Repeat ARQ

- ความซับซ้อนในการออกแบบและการปรับใช้: การทำงานที่มีการจัดการเลขอย่างและการเก็บข้อมูลที่ถูกส่งมาให้ถูกต้องตามลำดับ ทำให้การออกแบบและการปรับใช้มีความซับซ้อนมากกว่า Go-Back-N ARQ
- การใช้พื้นที่เก็บข้อมูล: เนื่องจากต้องเก็บข้อมูลที่ถูกส่งมาให้ถูกต้องตามลำดับ อุปกรณ์ปลายทางอาจจำเป็นต้องใช้พื้นที่เก็บข้อมูลมากขึ้น

Selective Repeat ARQ จึงจะถูกใช้ในเครือข่ายที่มีการสูญหายข้อมูลน้อยและมีความเสถียรสูง และมีการจัดการความถูกต้องแบบที่ละเอียดมากขึ้น

