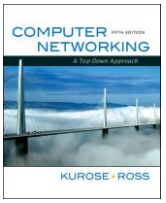
Chapter 2
Application Layer



Computer Networking: A Top Down Approach, 5th edition. Jim Kurose, Keith Ross Addison-Wesley, April 2009.

A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

Chapter 2: Tầng ứng dụng

- 2.1 Các nguyên lý của tầng ứng dụng
- 2.2 Web và HTTP
- 2.3 FTP
- 2.4 Thư điện tử
  - SMTP, POP3, IMAP
- 2.5 Hệ thống tên miền - DNS
- 2.6 Các ứng dụng dạng ngang hàng
- 2.7 Lập trình Socket cho TCP
- 2.8 Lập trình Socket cho UDP

Chapter 2: Tầng ứng dụng

Những vấn đề chính:

- Khái niệm, sự thi hành của các giao thức tầng ứng dụng
  - Các mô hình dịch vụ tầng vận chuyển
  - Mô hình client-server
  - Mô hình peer-to-peer
- Nghiên cứu các giao thức tầng ứng dụng
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
- Lập trình ứng dụng mạng
  - socket API

Một số ứng dụng tầng mạng

- e-mail
- web
- instant messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video clips
- voice over IP
- real-time video conferencing
- grid computing
-
-
-

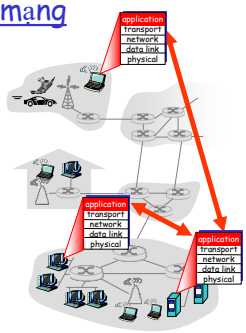
Tạo lập một ứng dụng mạng

Viết một chương trình mà

- Chạy trên các hệ thống cuối (khác nhau)
- Truyền tin thông qua mạng
- Ví dụ: phần mềm web server truyền tin với phần mềm browser

Không cần viết phần mềm cho các thiết bị lõi của mạng

- Thiết bị lõi mạng không chạy ứng dụng người sử dụng
- Ứng dụng trên hệ thống cuối cho phép xử lý nhanh và truyền



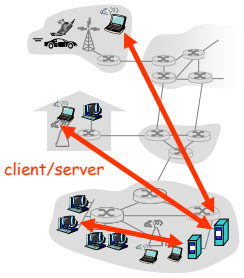
Chapter 2: Tầng ứng dụng

- 2.1 Các nguyên lý của tầng ứng dụng
- 2.2 Web và HTTP
- 2.3 FTP
- 2.4 Thư điện tử
  - SMTP, POP3, IMAP
- 2.5 Hệ thống tên miền - DNS
- 2.6 Các ứng dụng dạng ngang hàng
- 2.7 Lập trình Socket cho TCP
- 2.8 Lập trình Socket cho UDP
- 2.9 Xây dựng Web server

Kiến trúc của các ứng dụng

- Khách/chủ (Client-server)
- Ngang hàng (Peer-to-peer (P2P))
- Lai giữa client-server và P2P

Kiến trúc Khách-Chủ

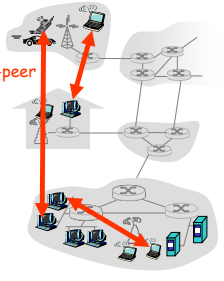


Chủ:
- Luôn luôn vai trò chủ
- Địa chỉ IP cố định
- Có khả năng mở rộng

Khách:
- Kết nối với chủ
- Có thể là kết nối không liên tục
- Có thể là địa chỉ IP động
- Không kết nối trực tiếp với nhau

Kiến trúc ngang hàng thuần túy

- *Không bao giờ đóng vai trò là chủ*
- Các hệ thống cuối kết nối với nhau một cách tùy ý
- Các đối tượng ngang hàng được kết nối không liên tục với nhau và thay đổi địa chỉ IP



Khả năng mở rộng cao, nhưng khó quản lý

Lai giữa client-server và P2P

Skype
- Ứng dụng ngang hàng voice-over-IP
- Server trung tâm: tìm kiếm địa chỉ của các thành viên ở xa của nhóm
- Kết nối client-client: trực tiếp (không thông qua server)

Instant messaging
- Chat giữa hai người là ngang hàng
- Dịch vụ trung tâm: phát hiện/định vị sự có mặt của client
  - user đăng ký địa chỉ IP của mình với server trung tâm khi vào mạng
  - user liên lạc với trung tâm dịch vụ để tìm địa chỉ IP của bạn cùng nhóm

Tiến trình kết nối

Process: Chương trình chạy bên trong một trạm.

- Trong cùng một trạm, hai tiến trình kết nối với nhau thông qua inter-process communication (định nghĩa bởi OS).
- Tiến trình của các trạm khác nhau kết nối thông qua trao đổi messages

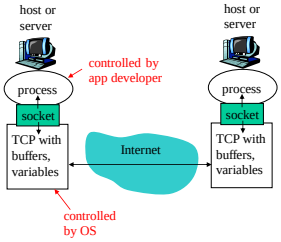
Client process: Tiến trình khởi tạo kết nối

Server process: Tiến trình chờ kết nối

- Lưu ý: Các ứng dụng cùng với kiến trúc ngang hàng có các tiến trình khách & chủ

Sockets

- Tiến trình gửi/nhận thông báo tới/từ bản thân nó socket
- socket tương tự như cánh cửa
  - Tiến trình gửi đẩy thông báo ra khỏi cửa
  - Tiến trình gửi dựa trên hạ tầng của vận chuyển, phía ngược lại của cửa sẽ đem thông báo đến socket



- API: (1) Sự lựa chọn của giao thức vận chuyển; (2) Khả năng sửa đổi một số tham số

Tiến trình xác định địa chỉ

- Để nhận các thông báo, tiến trình phải có định danh (*identifier)*
- Thiết bị trạm có duy nhất một địa chỉ IP 32-bit
- *Q:* địa chỉ IP của trạm có đủ để định danh tiến trình?

Tiến trình xác định địa chỉ

- Để nhận các thông báo, tiến trình phải có định danh (*identifier)*
- Thiết bị trạm có duy nhất một địa chỉ IP 32-bit
- *Q:* địa chỉ IP của trạm có đủ để định danh tiến trình?
  - *A:* Không, *nhiều* tiến trình cùng chạy trên một trạm
- *identifier* bao gồm cả IP address và port numbers kết hợp với tiến trình trên trạm.
- Ví dụ số hiệu cổng:
  - HTTP server: 80
  - Mail server: 25
- Gửi HTTP message tới gaia.cs.umass.edu web server:
  - IP address: 128.119.245.12
  - Port number: 80
- more shortly…

Định nghĩa giao thức tầng ứng dụng

- Các kiểu trao đổi messages,
  - e.g., request, response
- Cú pháp Message:
  - Trường nào trong messages & bao nhiêu trường được mô tả
- Ngữ nghĩa của Message
  - Ý nghĩa của thông tin trong trường
- Tập các quy tắc để gửi và hồi đáp messages

Các giao thức chung:

- Định nghĩa trong RFCs
- Cho phép thực hiện tương kết
- e.g., HTTP, SMTP

Giao thức riêng:

- e.g., Skype

Những gì mà dịch vụ tầng ứng dụng cần?

Mất dữ liệu
- Một số ứng dụng (e.g., audio) cho phép mất một ít dữ liệu
- Các ứng dụng khác (e.g., file transfer, telnet) yêu cầu 100% dữ liệu được truyền

Thời gian
- Một số ứng dụng (e.g., Internet telephony, interactive games) độ trễ thấp để đạt được hiệu quả

Thông lượng
- Một số ứng dụng (e.g., multimedia) yêu cầu thông lượng tối thiểu để đạt được hiệu quả
- Các ứng dụng khác ("các ứng dụng mềm dẻo") thích nghi với thông lượng

An toàn
- Mã hóa, toàn vẹn dữ liệu, ...

Các yêu cầu đối với dịch vụ vận chuyển của các ứng dụng thông dụng

Ứng dụng	Mất dữ liệu	Thông lượng	Thời gian
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
instant messaging	no loss	elastic	yes and no

Các giao thức vận chuyển liên mạng

Giao thức TCP:
- *connection-oriented:* yêu cầu thiết lập các tiến trình giữa client và server
- *reliable transport* đảm bảo an toàn đối với các tiến trình gửi và nhận
- *flow control:* sender không lấn át receiver
- *congestion control:* điều chỉnh sender khi mạng quá tải
- *does not provide:* thời gian, đảm bảo thông lượng tối thiểu, an toàn

Giao thức UDP:
- Chuyển dữ liệu không tin cậy giữa các tiến trình gửi/nhận
- Không cung cấp: thiết lập kết nối, độ tin cậy, điều khiển luồng, điều khiển tắc nghẽn, thời gian, đảm bảo thông lượng, hoặc an toàn

Các ứng dụng Internet: các giao thức tầng ứng dụng, vận chuyển

Ứng dụng	Giao thức tầng ứng dụng	Giao thức tầng vận chuyển
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 (eg Youtube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	typically UDP

Chapter 2: Tầng ứng dụng

- 2.1 Principles of network applications
  - app architectures
  - app requirements
- 2.2 Web and HTTP
- 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

Web và HTTP

Một số thuật ngữ

- Trang web bao gồm các đối tượng
- Đối tượng có thể là HTML file, JPEG image, Java applet, audio file,...
- Trang web chứa các tệp dựa trên cấu trúc HTML trong đó bao gồm các đối tượng truy vấn
- Mỗi đối tượng được địa chỉ hóa bởi một URL
- Ví dụ URL:

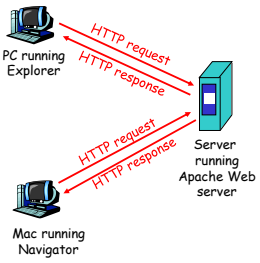
`www.someschool.edu/someDept/pic.gif`

host name | path name

Tổng quan về HTTP

HTTP: hypertext transfer protocol

- Giao thức ứng dụng web
- Mô hình client/server
  - *client:* trình duyệt yêu cầu, nhận, "hiển thị" các đối tượng web
  - *server:* Web server gửi các đối tượng hồi đáp các yêu cầu



Tổng quan về HTTP (continued)

Uses TCP:

- client khởi tạo kết nối TCP (tạo socket) tới server, cổng 80
- server chấp nhận kết nối TCP từ client
- HTTP messages trao đổi giữa (HTTP client) và Web server (HTTP server)
- Đóng kết nối TCP

HTTP: giao thức "không trạng thái"

- server không lưu thông tin các yêu cầu cũ của client

Kết nối HTTP

HTTP không xác thực

- Nhiều nhất 1 đối tượng được gửi qua một kết nối TCP.

HTTP xác thực

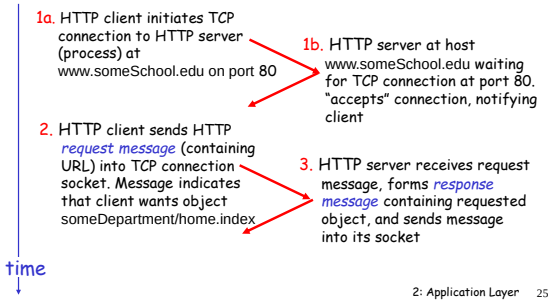
- Nhiều đối tượng có thể được gửi kết nối TCP đơn giữa client và server.

HTTP không xác thực

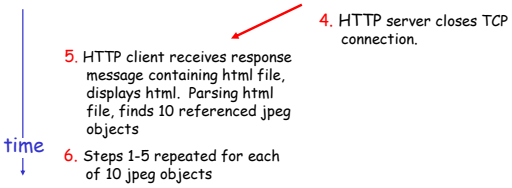
Giả sử nhập địa chỉ URL

`www.someSchool.edu/someDepartment/home.index`

(chứa text, truy vấn tới 10 jpeg)



HTTP không xác thực(cont.)



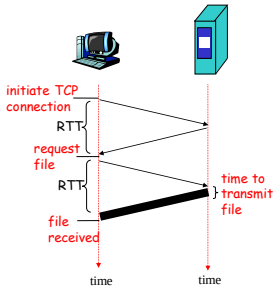
HTTP không xác thực: thời gian hồi đáp

Định nghĩa RTT: thời gian để một gói tin di chuyển từ client tới server và trở về.

Thời gian hồi đáp:

- Một RTT để khởi tạo kết nối TCP
- Một RTT cho yêu cầu HTTP và một số bytes đầu của HTTP trả về
- Thời gian truyền tệp

total = 2RTT+transmit time



HTTP xác thực

Hạn chế của HTTP không xác thực:

- Yêu cầu 2 RTTs/ 1 object
- OS chi phí phụ cho mỗi kết nối TCP
- Trình duyệt thường mở song song các kết nối TCP để tải/nạp các đối tượng truy vấn

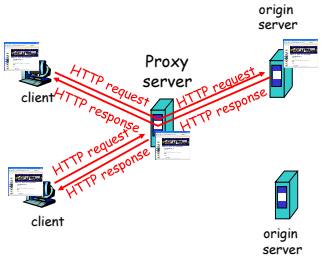
HTTP xác thực

- server để lại kết nối mở sau khi gửi đi hồi đáp
- Các HTTP messages tương tự giữa client/server được gửi qua kết nối mở
- client gửi yêu cầu sớm nhất có thể kết từ lúc bắt gặp đối tượng truy vấn
- Giống như một RTT cho tất cả các đối tượng truy vấn

Web caches (proxy server)

Mục đích: đáp ứng yêu cầu của client mà không cần tới server gốc

- Người sử dụng thiết lập trình duyệt: truy cập Web thông qua cache
- Trình duyệt gửi tất cả các yêu cầu HTTP tới cache
  - Đối tượng trong cache: cache trả về client
  - Ngược lại cache yêu cầu đối tượng từ server gốc, sau đó trả về client



Chapter 2: Application layer

- 2.1 Principles of network applications
- 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

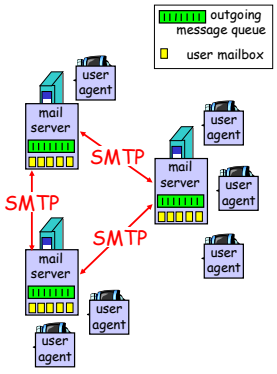
Thư điện tử

Ba thành phần cơ bản:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent

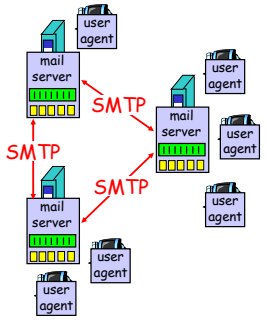
- a.k.a. "mail reader"
- composing, editing, reading mail messages
- e.g., Eudora, Outlook, elm, Mozilla Thunderbird
- outgoing, incoming messages stored on server



Thư điện tử: mail servers

Mail Servers

- mailbox chứa thư tới
- message queue các thư đã gửi đi
- SMTP protocol để gửi thư giữa các mail servers
  - client: server gửi thư
  - "server": server nhận thư

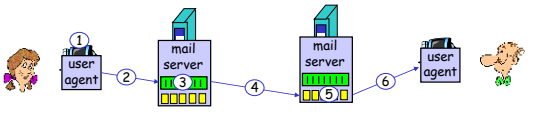


Thư điện tử: SMTP [RFC 2821]

- Dùng TCP để gửi thư từ client tới server, cổng 25
- Truyền trực tiếp: từ server gửi tới server nhận
- Ba giai đoạn của việc truyền
  - Thủ tục bắt tay (handshaking)
  - Truyền messages
  - Kết thúc
- Tương tác command/response
  - commands: ký tự ASCII
  - response: mã trạng thái và mệnh đề
- messages phải ở dạng 7-bit ASCII

Scenario: Alice gửi message tới Bob

1) Alice sử dụng UA soạn thư và gửi tới `bob@someschool.edu`
2) Alice's UA gửi thư tới mail server của cô ta; thư được lưu trong hàng đợi
3) SMTP phía client mở kết nối TCP với mail server của Bob
4) SMTP client gửi thư của Alice thông qua kết nối TCP
5) Mail server của Bob sẽ lưu thư trong mailbox
6) Bob nhờ UA đọc thư



Ví dụ về tương tác SMTP

```
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
```


SMTP: lời kết

- SMTP sử dụng liên kết ổn định
- SMTP yêu cầu message (header & body) ở dạng 7-bit ASCII
- SMTP server sử dụng `CRLF.CRLF` xác định kết thúc message

So sánh với HTTP:

- HTTP: pull
- SMTP: push
- Cả hai đều có tương tác ASCII command/response, mã trạng thái
- HTTP: mỗi đối tượng được đóng gói trong message hồi đáp
- SMTP: nhiều đối tượng gửi trong nhiều phần của message

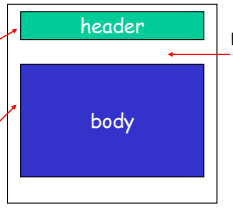
Khuôn dạng Mail message

SMTP: giao thức để trao đổi email msgs

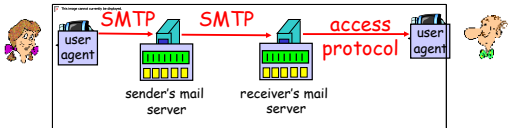
RFC 822: chuẩn định dạng text của message:

- header lines, e.g.,
  - To:
  - From:
  - Subject:

  different from SMTP commands!
- body
  - the "message", ASCII characters only



Các giao thức truy nhập Mail



- SMTP: delivery/storage tới server của người nhận
- Giao thức truy nhập Mail: nhận về từ server
  - POP: Post Office Protocol [RFC 1939]
    - Xác thực (agent <-->server) và download
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Nhiều đặc trưng hơn (phức tạp hơn)
    - Chọn lựa các msgs lưu trữ trên server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

Giao thức POP3

Giai đoạn xác thực

- client commands:
  - `user:` khai báo tên
  - `pass:` mật khẩu
- server responses
  - `+OK`
  - `-ERR`

Giai đoạn giao dịch, client:

- `list:` danh sách số hiệu message
- `retr:` nhận về message theo số hiệu
- `dele:` xóa
- `quit`

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 và IMAP

POP3
- Ví dụ trên sử dụng chế độ "download và delete".
- Bob không thể đọc lại e-mail nếu anh ta thay đổi client
- "Download-and-keep": bản sao của messages trên các clients khác nhau
- POP3 không lưu trạng thái người sử dụng suốt phiên truyền thông

IMAP
- Lưu giữ tất cả messages trong cùng một vị trí: server
- Người sử dụng có thể tổ chức các messages trong thư mục
- IMAP lưu giữ trạng thái người sử dụng trong suốt phiên truyền:
  - Tên thư mục và ánh xạ giữa message IDs và folder name

Chapter 2: Application layer

- 2.1 Principles of network applications
- 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
- 2.9 Building a Web server

DNS: Hệ thống tên miền

People: có nhiều định danh:
- CMND, họ tên, passport

Internet hosts, routers:
- Địa chỉ IP
- "tên", e.g., ww.yahoo.com

Q: ánh xạ giữa địa chỉ IP và tên?

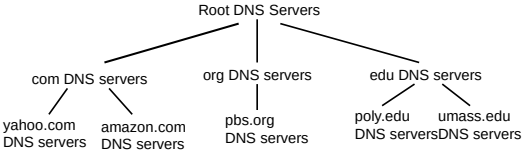
Domain Name System:
- *Cơ sở dữ liệu phân tán* triển khai theo lược đồ của nhiều *name servers*
- *application-layer protocol* host, routers, name servers liên lạc để thực hiện (address/name translation)
  - Chú ý: chức năng lõi của Internet thực hiện như giao thức của application-layer
  - Phức tạp tại sự ghép nối các mạng

DNS

DNS services
- Tên trạm để chuyển đổi với địa chỉ IP
- Bí danh trạm
  - Chính tắc, bí danh
- mail server bí danh
- Tải phân tán
  - Tạo bản sao của Web servers: thiết lập địa chỉ IP addresses cho một tên chính tắc

Tại sao DNS không tập trung?
- Sự cố tại một điểm đơn
- Lưu lượng luồng
- Khoảng cách tới CSDL tập trung
- Sự duy trì

Lược đồ phân tán CSDL



Client tìm IP cho www.amazon.com:
- client truy vấn a root server tìm com DNS server
- client truy vấn com DNS server tìm amazon.com DNS server
- client truy vấn amazon.com DNS server nhận địa chỉ IP cho www.amazon.com

DNS: Root name servers

- Trao đổi với local name server không tìm ra ánh xạ tên và địa chỉ
- root name server:
  - Kết nối với name server có thẩm quyền nếu ánh xạ tên và địa chỉ không được tìm thấy
  - Nhận thông tin ánh xạ
  - Trả thông tin về cho local name server

13 root name servers worldwide

TLD và Authoritative Servers

- Top-level domain (TLD) servers:
  - đảm đương cho com, org, net, edu, etc, và tất cả tên miền các quốc gia uk, fr, ca, jp.
  - Giải pháp mạng duy trì các servers cho com TLD
  - Đảm bảo cho edu TLD
- Các DNS servers có thẩm quyền:
  - DNS servers của các tổ chức cung cấp tên trạm ánh xạ tới địa chỉ IP cho các trạm servers (e.g., Web, mail).
  - Có thể được duy trì bởi các tổ chức hoặc nhà cung cấp dịch vụ

Server tên cục bộ

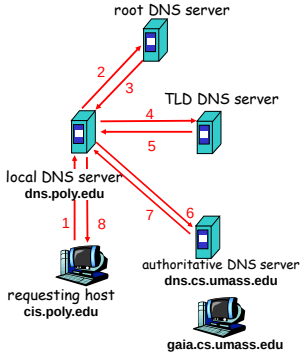
- Không tuân theo một lược đồ chặt chẽ
- Mỗi ISP (công ty, trường học) có một cái riêng
  - Còn được gọi là "server tên ngầm định"
- Lúc một trạm tạo truy vấn DNS, truy vấn được gửi tới DNS server cục bộ
  - Hoạt động như một proxy, chuyển tiếp truy vấn vào lược đồ địa chỉ.

Ví dụ về DNS

- Trạm tại cis.poly.edu cần địa chỉ IP của gaia.cs.umass.edu

Tạo truy vấn:

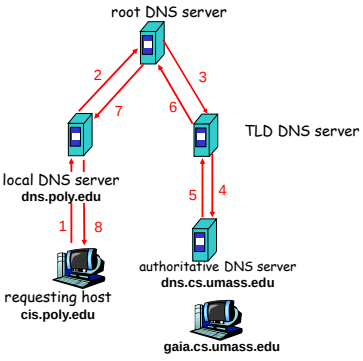
- Tạo kết nối và chuyển truy vấn tới server cục bộ
- "Ta không biết tên này, nhưng hỏi server này"



Ví dụ về DNS

Truy vấn lặp:

- Chuyển gánh nặng lên các server tên được kết nối
- Quá tải luồng?



DNS: bộ đệm và cập nhật bản ghi

- Mỗi server đều thực hiện việc tạo bộ đệm ánh xạ
  - Đầu vào cache tính thời gian chờ sau một khoảng thời gian nhất định
  - TLD servers là bản ghi chache gặp nhiều nhất trong servers tên cục bộ
    - Vì vậy root name servers không cần truy vấn thường xuyên
- Cơ chế cập nhật/thông báo (update/notify) được thiết kế bởi IETF
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

Bản ghi DNS

DNS: cơ sở DL phân tán lưu trữ bản ghi nguồn (RR)

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

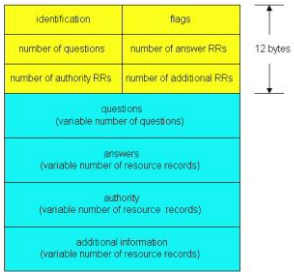
- Type=A
  - **name** là tên trạm
  - **value** là địa chỉ IP
- Type=NS
  - **name** là tên miền (e.g. foo.com)
  - **value** là tên trạm của server có thẩm quyền cho miền này
- Type=CNAME
  - **name** là tên bí danh cho một số tên chính tắc
    `www.ibm.com` là tên thật `servereast.backup2.ibm.com`
  - **value** là tên chính tắc
- Type=MX
  - **value** là tên của mailserver liên kết với **name**

DNS giao thức, thông báo

DNS giao thức: *query* và *reply* messages, cả hai đều cùng định dạng *message*

msg header

- identification: 16 bit # cho truy vấn, trả lời truy vấn sử dụng tương tự #
- flags:
  - query hoặc reply
  - Yêu cầu đệ quy
  - Hiệu lực đệ quy
  - reply là dạng xác thực



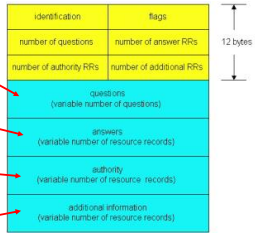
DNS giao thức, thông báo

Name, type fields for a query

RRs in response to query

records for authoritative servers

additional "helpful" info that may be used



Chèn bản ghi vào DNS

- Ví dụ: khởi tạo mới "Network Utopia"
- Đăng ký tên miền networkuptopia.com tại *DNS "hộ tịch viên"* (e.g., Network Solutions)
  - Cung cấp tên, địa chỉ IP của server xác thực tên (primary and secondary)
  - "Hộ tịch viên" chèn hai RRs vào com TLD server:

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

- Tạo server xác thực dạng bản ghi A cho www.networkuptopia.com; dạng bản ghi MX cho networkutopia.com

Chapter 2: Application layer

- 2.1 Principles of network applications
- 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

Lập trình Socket

Mục đích: học cách xây dựng ứng dụng client/server truyền thông lẫn nhau sử dụng sockets

Socket API

- Giới thiệu trong BSD4.1 UNIX, 1981
- Được tạo lập, sử dụng, giải phóng bởi ứng dụng
- Lược đồ client/server
- Hai kiểu của dịch vụ vận chuyển qua socket API:
  - Không xác thực
  - Xác thực, hướng luồng các byte

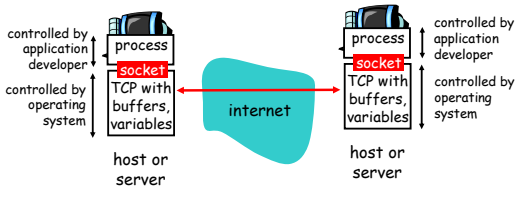
socket

Một *trạm cục bộ*, *Ứng dụng được tạo lập*, *OS-điều khiển* giao diện (một "cửa") trong đó tiến trình ứng dụng có thể *vừa gửi và nhận* messages tới từ tiến trình ứng dụng khác

Socket-lập trình sử dụng TCP

Socket: là cánh cửa giữa tiến trình ứng dụng và giao thức vận chuyển giữa các đầu cuối (UCP or TCP)

TCP service: vận chuyển một cách tin cậy các **bytes** từ một tiến trình đến tiến trình khác



Lập trình Socket *với TCP*

Client liên lạc với server
- Tiến trình server phải được chạy trước
- server phải tạo sẵn socket (cửa) chờ liên lạc từ client

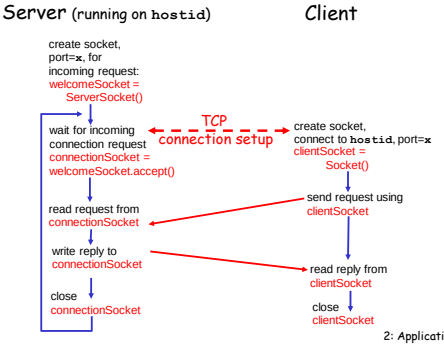
Client liên lạc với server bằng:
- Tạo client-local TCP socket
- Chỉ rõ IP address, port number của tiến trình server
- Khi client tạo socket: client TCP tạo lập kết nối tới server TCP

- Khi được liên lạc bởi client, server TCP tạo socket mới cho tiến trình server để truyền thông với client
  - Cho phép server hội thoại với nhiều clients
  - Số hiệu cổng nguồn dùng để phân biệt các clients

application viewpoint

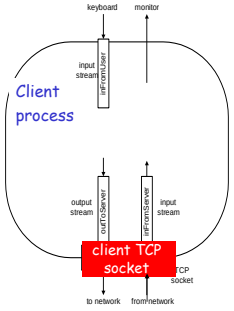
TCP cung cấp dịch vụ tin cậy để truyền các bytes ("ống dẫn") Giữa client và server

Tương tác Client/server socket: TCP



Luồng (Stream)

- Một luồng là một chuỗi các ký tự mà có thể đi vào hoặc ra khỏi một tiến trình.
- Một luồng vào được gắn một số nguồn vào cho tiến trình, e.g., keyboard or socket.
- Một luồng ra được gắn với nguồn ra, e.g., monitor or socket.



Lập trình Socket với TCP

Một liên kết được tạo lập thông qua TCP/IP sockets là một liên kết hướng kết nối. Điều này có nghĩa là liên kết giữa server và client sẽ được mở và duy trì trong suốt quá trình hội thoại giữa hai bên, và chỉ ngắt kết nối khi một trong hai bên chính thức ngắt quá trình trao đổi. Từ đó phân tách thành 2 dạng tiến trình (client và server), chúng ta sẽ xem xét một cách độc lập từng loại tiến trình. Trước hết chúng ta nghiên cứu đối với server, để thiết lập một tiến trình yêu cầu 5 bước.

1. Tạo một đối tượng ServerSocket.

ServerSocket yêu cầu tham số cổng (port), số hiệu cổng từ 1024 đến 65535. Ví dụ:

ServerSocket servSock = new ServerSocket(1234);

Trong ví dụ này Server sẽ đợi ('listen for') kết nối từ client ở cổng 1234.

Ví dụ: Java server (TCP)

2. Chuyển server vào trạng thái chờ

Server sẽ đợi vô hạn định ('blocks') để client kết nối.

Điều đó được thực hiện bởi việc gọi phương thức accept của lớp ServerSocket, nó sẽ trả về một đối tượng Socket khi mà kết nối được thiết lập. Ví dụ:

Socket link = servSock.accept();

3. Thiết lập luồng vào và ra.

Các phương thức getInputStream và getOutputStream của lớp Socket được sử dụng để truy vấn đến luồng liên kết với socket trả về trong bước 2. Các luồng đó được dùng để truyền thông với client và dùng thiết lập kết nối. Đối với một ứng dụng không phải dạng GUI, chúng ta có thể bọc một đối tượng Scanner bao bọc đối tượng InputStream được trả về bởi phương thức getInputStream để thu được đầu vào định hướng xâu. Ví dụ

Scanner input = new
Scanner(link.getInputStream());getOutputStream

Ví dụ: Java server (TCP), cont.

Tương tự chúng ta có thể bọc một đối tượng PrintWriter xung quanh đối tượng OutputStream được trả về bởi phương thức getOutputStream. Cung cấp cho cấu từ của PrintWriter là true dẫn đến bộ đệm ra luôn được xóa sạch sau mỗi lệnh gọi println. Ví dụ:

PrintWriter output = new
PrintWriter(link.getOutputStream(),true);

4. Gửi và nhận dữ liệu.

Nhờ thiết lập các đối tượng Scanner và PrintWriter nên việc truyền và nhận dữ liệu rất đơn giản. Chúng ta dễ dàng sử dụng phương thức nextLine cho nhận dữ liệu và phương thức println cho gửi dữ liệu, chẳng hạn chúng ta có thể dùng cho bàn phím Vào/Ra. Ví dụ:

String input = input.nextLine();

Ví dụ: Java server (TCP), cont.

5. Đóng kết nối (sau khi hoàn thành hội thoại).

Điều này được thực hiện thông qua phương thức close của lớp Socket. Ví dụ:

```
link.close();
```

Ví dụ sau đây minh họa cho các bước trên.

Trong ví dụ đơn giản này server sẽ chấp nhận các messages từ client và sẽ đếm số lượng messages, gửi trả lại thứ tự của các messages cho client. Thao tác chính cho dịch vụ này là client và server luân phiên nhau gửi và nhận dữ liệu. Phần chi tiết còn lại là xác định khi nào thì ngừng hội thoại và cái gì là dữ liệu kết thúc. Ở đây ta dùng chuỗi "***CLOSE***" sẽ được gửi bởi client khi nó muốn đóng liên kết. Khi server nhận được tín hiệu này nó sẽ gửi xác nhận số hiệu message trước đó và đóng liên kết với client. Phía client tất nhiên là sẽ chờ nhận dữ liệ cuối cùng từ server và đóng kết nối. Từ lúc server đưa ra lời gọi, dịch vụ sẽ thực thi vô thời hạn, lời gọi phương thức handleClient nằm trong vòng lặp vô hạn.

Ví dụ: Java client (TCP).

Thiết lập tương ứng với client là 4 bước

1. Tạo kết nối tới server

Chúng ta tạo một đối tượng Socket cung cấp cấu tử với 2 tham số sau:

- Địa chỉ IP của server
- Số hiệu cổng giành riêng cho dịch vụ

(Tất nhiên là số hiệu cổng của server và client phải như nhau)

Để đơn giản hóa, chúng ta thực hiện tiến trình server và client trên cùng một trạm, điều đó cho phép chúng ta lấy ra địa chỉ IP bằng việc gọi phương thức tĩnh getLocalHost của lớp InetAddress. Ví dụ:

```
Socket link = new Socket(InetAddress.getLocalHost(),1234);
```


Ví dụ: Java client (TCP) cont.

2. Thiết lập luồng vào và ra.

Dùng các phương thức getInputStream và getOutStream của đối tượng Socket để thiết lập chính xác luồng giống như ở bước 2 của server.

3. Gửi và nhận dữ liệu.

Đối tượng Scanner ở đầu cuối client sẽ nhận messages được gửi bởi đối tượng PrintWriter tại đầu cuối server, trong khi đối tượng PrintWriter tại đầu cuối client sẽ gửi messages và các đối tượng Scanner ở đầu cuối server sẽ nhận (sử dụng các phương thức nextLine và println tương ứng).

4. Đóng kết nối.

Ở đây thực hiện tương tự tiến trình ở server, sử dụng phương thức close của lớp Socket).

Chapter 2: Summary

Những nội dung chính của chương:

- Kiến trúc của ứng dụng
  - client-server
  - Ngang hàng
  - hybrid
- Yêu cầu của dịch vụ ứng dụng:
  - độ tin cậy, bang thông, độ trễ
- Mô hình vận chuyển liên mạng
  - Hướng kết nối, xác thực: TCP
  - Không xác thực, gói dữ liệu: UDP
- Các giao thức:
  - HTTP
  - FTP
  - SMTP, POP, IMAP
  - DNS
  - P2P: BitTorrent, Skype
- Lập trình socket

Chapter 2: Summary

Những nội dung chính của chương:

- Phương thức chính trao đổi request/reply message:
  - client yêu cầu thông tin hoặc dịch vụ
  - server hồi đáp bằng dữ liệu, mã trạng thái
- Định dạng message:
  - headers: trường chứa thông tin về dữ liệu
  - data: thông tin được truyền thông