

Lớp ứng dụng

Các ứng dụng mạng là *lý do tồn tại* của mạng máy tính - nếu chúng ta không thể hình dung ra bất kỳ ứng dụng hữu ích nào, sẽ không cần bất kỳ cấu trúc cơ sở hạ tầng và giao thức mạng nào để hỗ trợ chúng. Kể từ khi Internet ra đời, nhiều ứng dụng hữu ích và giải trí đã thực sự được tạo ra. Những ứng dụng này đã là động lực đằng sau sự thành công của Internet, thúc đẩy mọi người ở nhà, trường học, chính phủ và doanh nghiệp biến Internet thành một phần không thể thiếu trong các hoạt động hàng ngày của họ.

Các ứng dụng Internet bao gồm các ứng dụng dựa trên văn bản cổ điển đã trở nên phổ biến trong những năm 1970 và 1980: e-mail văn bản, truy cập từ xa vào máy tính, truyền tệp và nhóm tin. Chúng bao gồm *ứng dụng* sát thủ vào giữa những năm 1990, World Wide Web, bao gồm lướt web, tìm kiếm và thương mại điện tử. Kể từ khi bắt đầu thiên niên kỷ mới, các ứng dụng mới và hấp dẫn tiếp tục xuất hiện, bao gồm thoại qua IP và hội nghị truyền hình như Skype, Facetime và Google Hangouts; video do người dùng tạo như YouTube và phim theo yêu cầu như Netflix; và các trò chơi trực tuyến nhiều người chơi như Second Life và World of Warcraft. Trong cùng thời gian này, chúng ta đã thấy sự xuất hiện của một thể hệ ứng dụng mạng xã hội mới như Facebook, Instagram và Twitter đã tạo ra mạng người trên mạng Internet hoặc các bộ định tuyến và liên kết truyền thông. Và gần đây nhất, cùng với sự xuất hiện của điện thoại thông minh và sự phổ biến của truy cập Internet không dây 4G / 5G, đã có rất nhiều ứng dụng di động dựa trên vị trí, bao gồm các ứng dụng đăng ký, hẹn hò và dự báo giao thông đường bộ phổ biến (như Yelp, Tinder và Waz), ứng dụng thanh toán di động (như WeChat và Apple Pay) và ứng dụng nhắn tin (như WeChat và WhatsApp). Rõ ràng, không có sự chậm lại của các ứng dụng Internet mới và thú vị. Có lẽ một số độc giả của văn bản này sẽ tạo ra thể hệ tiếp theo của các ứng dụng Internet sát thủ!

Trong chương này, chúng tôi nghiên cứu các khía cạnh khái niệm và triển khai của các ứng dụng mạng. Chúng tôi bắt đầu bằng cách xác định các khái niệm lớp ứng dụng chính, bao gồm các dịch vụ net-work theo yêu cầu của các ứng dụng, máy khách và máy chủ, quy trình và giao diện lớp xuyên công. Chúng tôi kiểm tra chi tiết một số ứng dụng mạng, bao gồm Web, e-mail, DNS, phân phối tệp ngang hàng (P2P) và phát trực tuyến video. Sau đó, chúng tôi đề cập đến phát triển ứng dụng mạng, trên cả TCP và UDP. Cụ thể, chúng tôi nghiên cứu giao diện socket và đi qua một số ứng dụng client-server đơn giản trong Python. Chúng tôi cũng cung cấp một số bài tập lập trình socket thú vị và thú vị ở cuối chương.

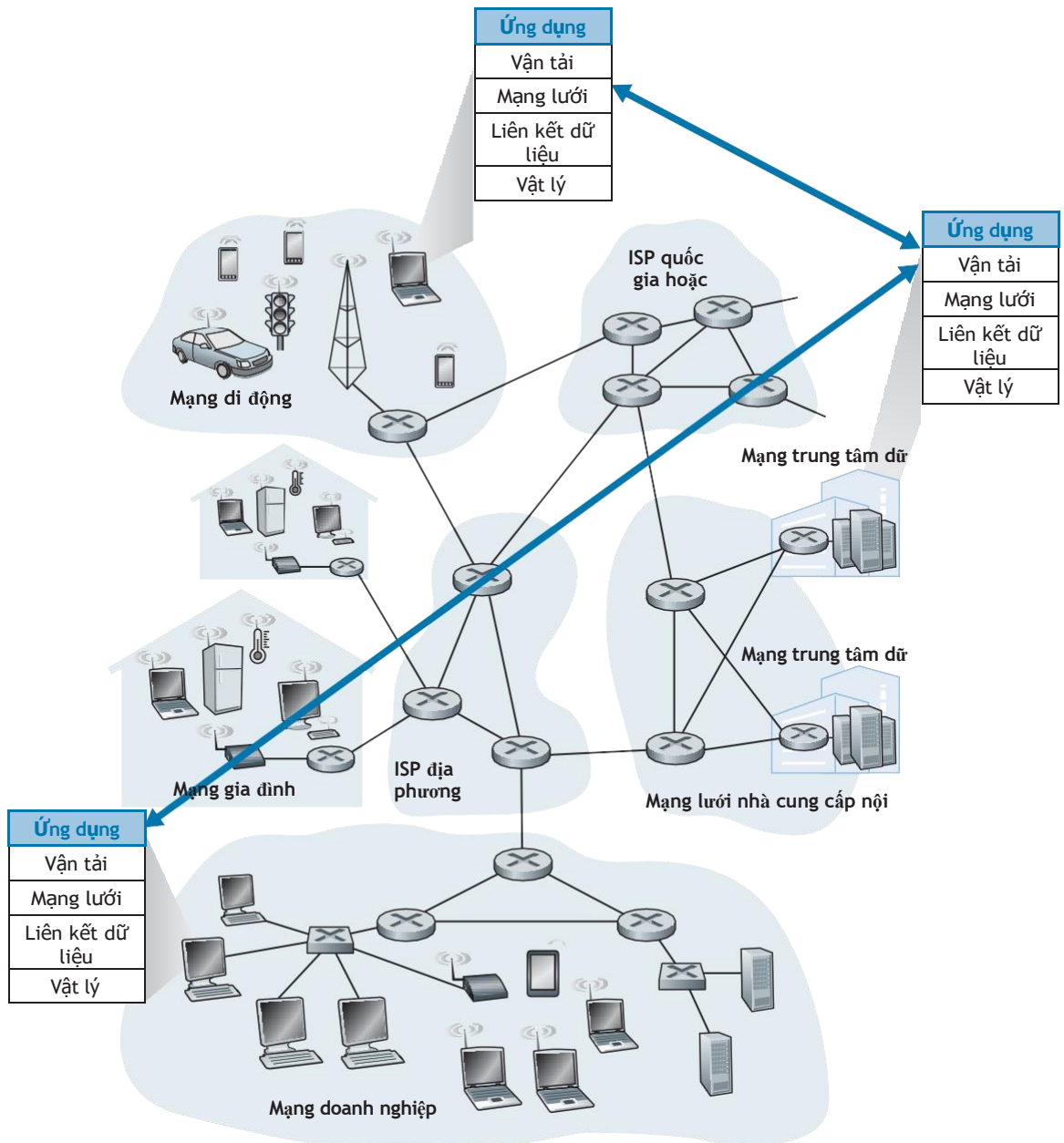
Lớp ứng dụng là một nơi đặc biệt tốt để bắt đầu nghiên cứu các giao thức của chúng tôi. Đó là mặt đất quen thuộc. Chúng tôi đã làm quen với nhiều ứng dụng dựa trên các giao thức mà chúng tôi sẽ nghiên cứu. Nó sẽ cho chúng ta cảm giác tốt về tất cả các giao thức và sẽ giới thiệu cho chúng ta nhiều vấn đề tương tự mà chúng ta sẽ thấy lại khi chúng ta nghiên cứu các giao thức lớp vận chuyển, mạng và liên kết.

2.1 Nguyên tắc ứng dụng mạng

Giả sử bạn có một ý tưởng cho một ứng dụng mạng mới. Có lẽ ứng dụng này sẽ là một dịch vụ tuyệt vời cho nhân loại, hoặc sẽ làm hài lòng giáo sư của bạn, hoặc sẽ mang lại cho bạn sự giàu có to lớn, hoặc đơn giản là sẽ rất thú vị để phát triển. Dù động lực có thể là gì, bây giờ chúng ta hãy kiểm tra cách bạn biến ý tưởng thành một ứng dụng mạng trong thế giới thực.

Cốt lõi của phát triển ứng dụng mạng là viết các chương trình chạy trên các hệ thống đầu cuối khác nhau và giao tiếp với nhau qua mạng. Trong ứng dụng Web có hai chương trình riêng biệt giao tiếp với nhau: chương trình trình duyệt chạy trong máy chủ của người dùng (máy tính để bàn, máy tính xách tay, máy tính bảng, điện thoại thông minh, v.v.); và chương trình máy chủ Web đang chạy trong máy chủ Web. Một ví dụ khác, trong ứng dụng Video theo yêu cầu như Netflix (xem Phần 2.6), có một chương trình do Netflix cung cấp chạy trên điện thoại thông minh, máy tính bảng hoặc máy tính của người dùng; và một chương trình máy chủ Netflix chạy trên máy chủ lưu trữ Netflix. Các máy chủ thường (nhưng chắc chắn không phải luôn luôn) được đặt trong một trung tâm dữ liệu, như thể hiện trong Hình 2.1.

Do đó, khi phát triển ứng dụng mới của bạn, bạn cần viết phần mềm sẽ chạy trên nhiều hệ thống đầu cuối. Phần mềm này có thể được viết, ví dụ, bằng C, Java hoặc Python. Điều quan trọng, bạn không cần phải viết phần mềm chạy trên các thiết bị lỗi làm việc mạng, chẳng hạn như bộ định tuyến hoặc thiết bị chuyển mạch lớp liên kết. Ngay cả khi bạn muốn viết phần mềm ứng dụng cho các thiết bị lỗi mạng này, bạn sẽ không thể làm như vậy. Như chúng ta đã học trong Chương 1 và như thể hiện trước đó trong Hình 1.24, các thiết bị lỗi mạng không hoạt động ở lớp ứng dụng mà thay vào đó hoạt động ở các lớp thấp hơn - cụ thể là ở lớp mạng trở xuống. Thiết kế cơ bản này - cụ thể là giới hạn phần mềm ứng dụng vào các hệ thống cuối - như thể hiện trong Hình 2.1, đã tạo điều kiện cho sự phát triển và triển khai nhanh chóng của một loạt các ứng dụng mạng.



Hình 2.1 ♦ Giao tiếp cho một ứng dụng mạng diễn ra giữa các hệ thống đầu cuối ở lớp ứng dụng

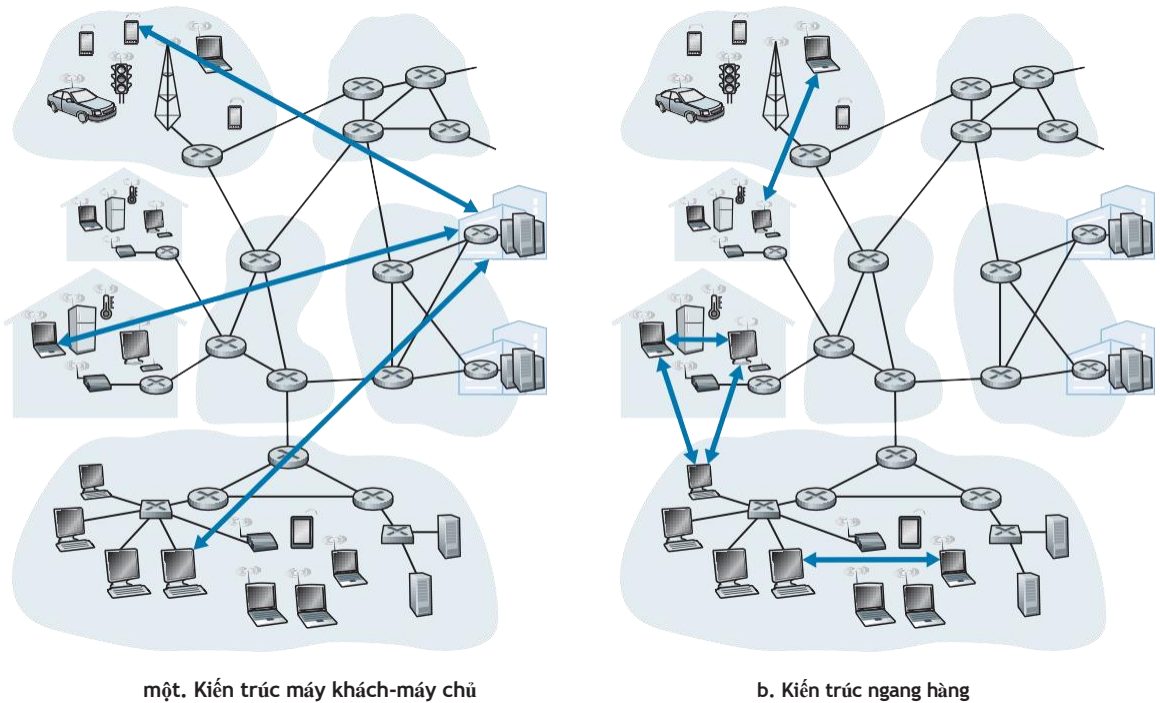
2.1.1 Kiến trúc ứng dụng mạng

Trước khi đi sâu vào mã hóa phần mềm, bạn nên có một kế hoạch kiến trúc rộng cho ứng dụng của mình. Hãy nhớ rằng kiến trúc của ứng dụng khác biệt rõ rệt với kiến trúc mạng (ví dụ: kiến trúc Internet năm lớp được thảo luận trong Chương 1). Từ quan điểm của nhà phát triển ứng dụng, kiến trúc mạng được cố định và cung cấp một bộ dịch vụ cụ thể cho các ứng dụng. Mặt khác, **kiến trúc ứng dụng** được thiết kế bởi nhà phát triển ứng dụng và chỉ ra cách ứng dụng được cấu trúc trên các hệ thống đầu cuối khác nhau. Khi chọn kiến trúc ứng dụng, một nhà phát triển ứng dụng có thể sẽ dựa trên một trong hai mô hình kiến trúc chiếm ưu thế được sử dụng trong các ứng dụng mạng hiện đại: kiến trúc máy khách-máy chủ hoặc kiến trúc ngang hàng (P2P).

Trong **kiến trúc máy khách-máy chủ**, có một máy chủ luôn bật, được gọi là *máy chủ*, phục vụ các yêu cầu từ nhiều máy chủ khác, được gọi là *máy khách*. Một ví dụ kinh điển là ứng dụng Web mà một dịch vụ máy chủ Web luôn bật yêu cầu từ các trình duyệt chạy trên máy chủ máy khách. Khi một máy chủ Web nhận được một yêu cầu cho một đối tượng từ một máy chủ khách hàng, nó đáp ứng bằng cách gửi các đối tượng được yêu cầu đến máy chủ khách hàng. Lưu ý rằng với kiến trúc máy khách-máy chủ, các máy khách không giao tiếp trực tiếp với nhau; ví dụ, trong ứng dụng Web, hai trình duyệt không trực tiếp giao tiếp. Một đặc điểm khác của kiến trúc máy khách-máy chủ là máy chủ có một địa chỉ cố định, nổi tiếng, được gọi là địa chỉ IP (mà chúng ta sẽ thảo luận sớm). Bởi vì máy chủ có một địa chỉ cố định, nổi tiếng và vì máy chủ luôn bật, một cli-ent luôn có thể liên hệ với máy chủ bằng cách gửi một gói tin đến địa chỉ IP của máy chủ. Một số ứng dụng nổi tiếng hơn với kiến trúc máy khách-máy chủ bao gồm Web, FTP, Telnet và e-mail. Kiến trúc client-server được thể hiện trong Hình 2.2(a).

Thông thường trong một ứng dụng máy khách-máy chủ, một máy chủ lưu trữ đơn không có khả năng theo kịp tất cả các yêu cầu từ máy khách. Ví dụ: một trang web mạng xã hội phổ biến có thể nhanh chóng bị choáng ngợp nếu chỉ có một máy chủ xử lý tất cả các yêu cầu của nó. Vì lý do này, một **trung tâm dữ liệu**, chứa một số lượng lớn máy chủ, thường được sử dụng để tạo ra một máy chủ ảo mạnh mẽ. Các dịch vụ Internet phổ biến nhất — chẳng hạn như công cụ tìm kiếm (ví dụ: Google, Bing, Baidu), thương mại Internet (ví dụ: Amazon, eBay, Alibaba), e-mail dựa trên web (ví dụ: Gmail và Yahoo Mail), phương tiện truyền thông xã hội (ví dụ: Facebook, Instagram, Twitter và WeChat) — chạy trong một hoặc nhiều trung tâm dữ liệu. Như đã thảo luận trong Phần 1.3.3, Google có 19 trung tâm dữ liệu được phân phối trên toàn thế giới, xử lý chung tìm kiếm, YouTube, Gmail và các dịch vụ khác. Một trung tâm dữ liệu có thể có hàng trăm ngàn máy chủ, phải được cung cấp và duy trì. Ngoài ra, các nhà cung cấp dịch vụ phải trả chi phí kết nối và băng thông định kỳ để gửi dữ liệu từ trung tâm dữ liệu của họ.

Trong **kiến trúc P2P**, có sự phụ thuộc tối thiểu (hoặc không) vào các máy chủ chuyên dụng trong các trung tâm dữ liệu. Thay vào đó, ứng dụng khai thác giao tiếp trực tiếp giữa các cặp máy chủ được kết nối không liên tục, được gọi là *peers*. Các đồng nghiệp không thuộc sở hữu của nhà cung cấp dịch vụ, mà thay vào đó là máy tính để bàn và máy tính xách tay do người dùng kiểm soát, với hầu hết các đồng nghiệp cư trú tại nhà, trường đại học và văn phòng. Bởi vì các đồng nghiệp giao tiếp



Hình 2.2 ♦ (a) Kiến trúc máy khách-máy chủ; (b) Kiến trúc P2P

Nếu không đi qua một máy chủ chuyên dụng, kiến trúc được gọi là peer-to-peer. Một ví dụ về ứng dụng P2P phổ biến là ứng dụng chia sẻ tệp BitTorrent.

Một trong những tính năng hấp dẫn nhất của kiến trúc P2P là **khả năng tự mở rộng của chúng**. Ví dụ: trong ứng dụng chia sẻ tệp P2P, mặc dù mỗi peer tạo ra khối lượng công việc bằng cách yêu cầu tệp, mỗi peer cũng bổ sung dung lượng dịch vụ cho hệ thống bằng cách phân phối tệp cho các đồng nghiệp khác. Kiến trúc P2P cũng hiệu quả về chi phí, vì chúng thường không yêu cầu cơ sở hạ tầng máy chủ và băng thông máy chủ đáng kể (trái ngược với thiết kế máy khách-máy chủ với trung tâm dữ liệu). Tuy nhiên, các ứng dụng P2P phải đối mặt với những thách thức về bảo mật, hiệu suất và độ tin cậy do cấu trúc phi tập trung cao của chúng.

2.1.2 Quy trình giao tiếp

Trước khi xây dựng ứng dụng mạng của bạn, bạn cũng cần một sự hiểu biết cơ bản về cách các chương trình, chạy trong nhiều hệ thống cuối, giao tiếp với nhau. Trong thuật ngữ của hệ điều hành, nó không thực sự là các chương trình mà là **các quy trình**

giao tiếp. Một quá trình có thể được coi là một chương trình đang chạy trong một hệ thống cuối. Khi các tiến trình đang chạy trên cùng một hệ thống cuối, chúng có thể giao tiếp với nhau bằng giao tiếp giữa các tiến trình, sử dụng các quy tắc được điều chỉnh bởi hệ điều hành của hệ thống cuối. Nhưng trong cuốn sách này, chúng tôi không đặc biệt quan tâm đến cách các tiến trình trong cùng một máy chủ giao tiếp, mà thay vào đó là cách các quy trình chạy trên *các máy chủ khác nhau* (với các hệ điều hành có khả năng khác nhau) giao tiếp.

Các quy trình trên hai hệ thống đầu cuối khác nhau giao tiếp với nhau bằng cách trao đổi **tin nhắn** trên mạng máy tính. Một quá trình gửi tạo và gửi tin nhắn vào mạng; Quá trình nhận nhận các tin nhắn này và có thể trả lời bằng cách gửi lại tin nhắn. Hình 2.1 minh họa rằng các quá trình giao tiếp với nhau nằm trong lớp ứng dụng của ngăn xếp pro-tocol năm lớp.

Quy trình máy khách và máy chủ

Một ứng dụng mạng bao gồm các cặp quy trình gửi tin nhắn cho nhau qua mạng. Ví dụ, trong ứng dụng Web, một quá trình trình duyệt máy khách trao đổi thư với một quá trình máy chủ Web. Trong hệ thống chia sẻ tệp P2P, một tệp được chuyển từ một quy trình trong một ngang hàng sang một quy trình trong một ngang hàng khác. Đối với mỗi cặp quy trình giao tiếp, chúng tôi thường gắn nhãn một trong hai quy trình là **máy khách** và quy trình còn lại là **máy chủ**. Với Web, trình duyệt là một quá trình máy khách và một máy chủ Web là một quá trình máy chủ. Với chia sẻ tệp P2P, đồng nghiệp đang tải xuống tệp được gắn nhãn là máy khách và ngang hàng đang tải lên tệp được gắn nhãn là máy chủ.

Bạn có thể đã quan sát thấy rằng trong một số ứng dụng, chẳng hạn như trong chia sẻ tệp P2P, một quy trình có thể vừa là máy khách vừa là máy chủ. Thật vậy, một quá trình trong hệ thống chia sẻ tệp P2P có thể tải lên và tải xuống tệp. Tuy nhiên, trong bối cảnh của bất kỳ phiên giao tiếp nhất định nào giữa một cặp quy trình, chúng ta vẫn có thể gắn nhãn một quy trình là máy khách và quy trình khác là máy chủ. Chúng tôi định nghĩa pro-cesses máy khách và máy chủ như sau:

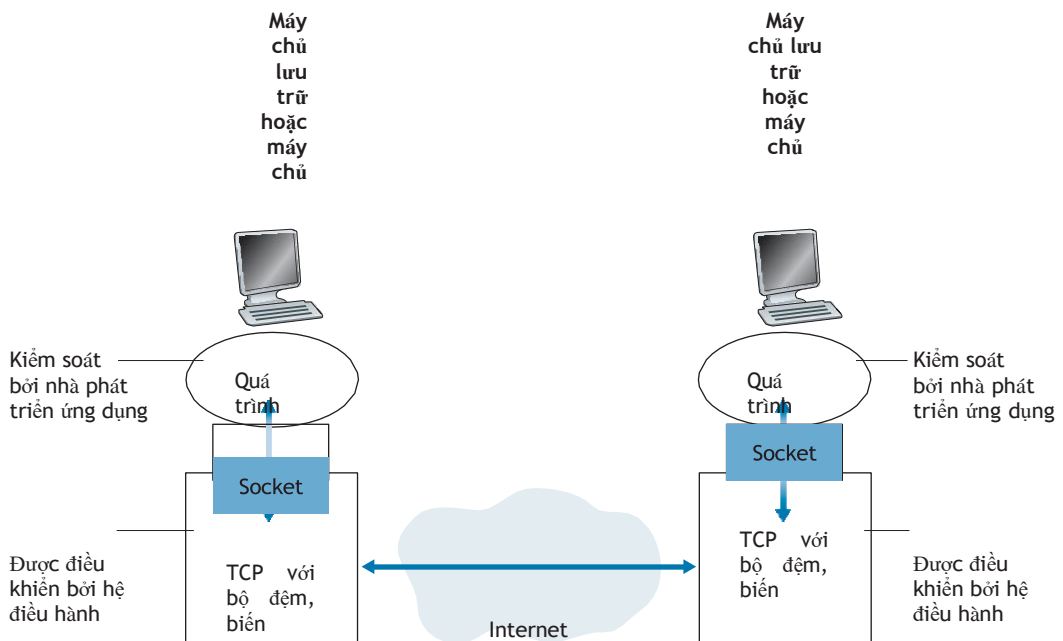
Trong bối cảnh phiên giao tiếp giữa một cặp quy trình, pro-cess khởi tạo giao tiếp (nghĩa là ban đầu liên hệ với quy trình khác vào đầu phiên) được gắn nhãn là máy khách. Quá trình chờ đợi để được liên hệ để bắt đầu phiên là máy chủ.

Trong Web, một quá trình trình duyệt khởi tạo liên hệ với một quá trình máy chủ Web; do đó quá trình trình duyệt là máy khách và quá trình máy chủ Web là máy chủ. Trong chia sẻ tệp P2P, khi Peer A yêu cầu Peer B gửi một tệp cụ thể, Peer A là cli- ent và Peer B là máy chủ trong ngữ cảnh của phiên giao tiếp cụ thể này. Khi không có sự nhầm lẫn, đôi khi chúng ta cũng sẽ sử dụng thuật ngữ "phía máy khách và phía máy chủ của một ứng dụng". Ở cuối chương này, chúng ta sẽ bước qua mã sim-ple cho cả phía máy khách và máy chủ của các ứng dụng mạng.

Giao diện giữa quá trình và mạng máy tính

Như đã lưu ý ở trên, hầu hết các ứng dụng bao gồm các cặp quy trình giao tiếp, với hai quy trình trong mỗi cặp gửi tin nhắn cho nhau. Bất kỳ thông điệp nào được gửi từ quy trình này sang quy trình khác phải đi qua mạng cơ bản. Một quá trình gửi tin nhắn vào và nhận tin nhắn từ mạng thông qua giao diện phần mềm được gọi là **ổ cắm**. Hãy xem xét một sự tương tự để giúp chúng ta hiểu các quy trình và ổ cắm. Một quá trình tương tự như một ngôi nhà và ổ cắm của nó tương tự như cửa của nó. Khi một tiến trình muốn gửi một tin nhắn đến một tiến trình khác trên một máy chủ khác, nó sẽ đẩy thông điệp ra khỏi cửa (ổ cắm). Quá trình gửi này giả định rằng có một cơ sở hạ tầng chuyển cảng ở phía bên kia cánh cửa của nó sẽ vận chuyển thông điệp đến cửa của quy trình đích. Khi tin nhắn đến máy chủ đích, tin nhắn đi qua cửa (ổ cắm) của quá trình nhận và quá trình nhận sau đó tác động lên tin nhắn.

Hình 2.3 minh họa giao tiếp ổ cắm giữa hai quá trình giao tiếp qua Internet. (Hình 2.3 giả định rằng giao thức truyền tải cơ bản được sử dụng bởi các quy trình là giao thức TCP của Internet.) Như thể hiện trong hình này, socket là giao diện giữa lớp ứng dụng và lớp vận chuyển trong máy chủ. Nó còn được gọi là **Giao diện lập trình ứng dụng (API)** giữa ứng dụng và mạng, vì socket là giao diện lập trình mà các ứng dụng mạng được xây dựng. Nhà phát triển ứng dụng có quyền kiểm soát mọi thứ ở phía lớp ứng dụng của ổ cắm nhưng có ít quyền kiểm soát phía lớp vận chuyển của ổ cắm. Điều khiển duy nhất mà nhà phát triển ứng dụng có ở phía lớp vận chuyển là (1) lựa chọn giao thức vận chuyển và (2) có lẽ khả năng sửa một vài



Hình 2.3 ♦ Các quy trình ứng dụng, socket và giao thức truyền tải cơ bản

các tham số lớp vận chuyển như bộ đệm tối đa và kích thước phân đoạn tối đa (sẽ được đề cập trong Chương 3). Khi nhà phát triển ứng dụng chọn giao thức truyền tải (nếu có lựa chọn), ứng dụng được xây dựng bằng cách sử dụng các dịch vụ lớp vận chuyển được cung cấp bởi giao thức đó. Chúng ta sẽ khám phá các ổ cắm một số chi tiết trong Phần 2.7.

Giải quyết các quy trình

Để gửi thư bưu điện đến một điểm đến cụ thể, điểm đến cần phải có địa chỉ. Tương tự, để một tiến trình chạy trên một máy chủ gửi các gói đến một tiến trình chạy trên một máy chủ khác, quá trình nhận cần phải có một địa chỉ. Để xác định quá trình nhận, hai phần thông tin cần được chỉ định:

(1) địa chỉ của máy chủ lưu trữ và (2) mã định danh chỉ định quy trình nhận trong máy chủ đích.

Trên Internet, máy chủ được xác định bằng **địa chỉ IP** của nó. Chúng ta sẽ thảo luận về địa chỉ IP rất chi tiết trong Chương 4. Hiện tại, tất cả những gì chúng ta cần biết là địa chỉ IP là số lượng 32 bit mà chúng ta có thể coi là nhận dạng duy nhất máy chủ. Ngoài việc biết địa chỉ của máy chủ mà tin nhắn được gửi đến, quá trình gửi cũng phải xác định quá trình nhận (cụ thể hơn là ổ cắm nhận) đang chạy trong máy chủ. Thông tin này là cần thiết vì nói chung một máy chủ có thể chạy nhiều ứng dụng mạng. Số **cổng đích** phục vụ mục đích này. Các ứng dụng phổ biến đã được gán số cổng cụ thể. Ví dụ: máy chủ Web được xác định bằng cổng số 80. Một quá trình máy chủ thư (sử dụng giao thức SMTP) được xác định bằng cổng số 25. Một danh sách các số cổng nổi tiếng cho tất cả các giao thức tiêu chuẩn Internet có thể được tìm thấy tại www.iana.org. Chúng ta sẽ kiểm tra số cổng một cách chi tiết trong Chương 3.

2.1.3 Dịch vụ vận tải có sẵn cho các ứng dụng

Hãy nhớ lại rằng một socket là giao diện giữa quá trình ứng dụng và giao thức transport-layer. Ứng dụng ở phía gửi đẩy tin nhắn qua ổ cắm. Ở phía bên kia của ổ cắm, giao thức lớp vận chuyển có trách nhiệm đưa tin nhắn đến ổ cắm của quá trình nhận.

Nhiều mạng, bao gồm cả Internet, cung cấp nhiều hơn một giao thức lớp vận chuyển. Khi bạn phát triển một ứng dụng, bạn phải chọn một trong các giao thức lớp vận chuyển có sẵn. Làm thế nào để bạn thực hiện sự lựa chọn này? Rất có thể, bạn sẽ nghiên cứu các dịch vụ được cung cấp bởi các giao thức lớp vận chuyển có sẵn và sau đó chọn giao thức với các dịch vụ phù hợp nhất với nhu cầu của ứng dụng của bạn. Tình hình tương tự như việc chọn phương tiện di chuyển bằng tàu hỏa hoặc máy bay để đi lại giữa hai thành phố. Bạn phải chọn cái này hay cái kia, và mỗi phương thức vận chuyển cung cấp các dịch vụ khác nhau. (Ví dụ: tàu cung cấp dịch vụ đón và trả khách ở trung tâm thành phố, trong khi máy bay cung cấp thời gian di chuyển ngắn hơn.)

Các dịch vụ mà một giao thức lớp vận chuyển có thể cung cấp cho các ứng dụng gọi nó là gì? Chúng tôi có thể phân loại rộng rãi các dịch vụ có thể theo bốn chiều: truyền dữ liệu đáng tin cậy, thông lượng, thời gian và bảo mật.

Truyền dữ liệu đáng tin cậy

Như đã thảo luận trong Chương 1, các gói tin có thể bị mất trong mạng máy tính. Đối với exam-ple, một gói có thể tràn bộ đệm trong bộ định tuyến hoặc có thể bị loại bỏ bởi máy chủ hoặc bộ định tuyến sau khi một số bit của nó bị hỏng. Đối với nhiều ứng dụng — chẳng hạn như thư điện tử, truyền tệp, truy cập máy chủ từ xa, chuyển tải liệu Web và ứng dụng tài chính — mất dữ liệu có thể gây ra hậu quả tàn phá (trong trường hợp sau, đối với ngân hàng hoặc khách hàng!). Do đó, để hỗ trợ các ứng dụng này, phải làm gì đó để đảm bảo rằng dữ liệu được gửi bởi một đầu của ứng dụng được phân phối tương ứng và hoàn toàn đến đầu kia của ứng dụng. Nếu một giao thức cung cấp dịch vụ phân phối dữ liệu được đảm bảo như vậy, nó được cho là cung cấp **truyền dữ liệu đáng tin cậy**. Một dịch vụ quan trọng mà giao thức lớp vận chuyển có khả năng cung cấp cho ứng dụng là truyền dữ liệu đáng tin cậy từ quy trình này sang quy trình khác. Khi một giao thức truyền tải cung cấp dịch vụ này, quá trình gửi chỉ có thể truyền dữ liệu của nó vào socket và biết hoàn toàn tự tin rằng dữ liệu sẽ đến mà không có lỗi trong quá trình nhận.

Khi giao thức lớp vận chuyển không cung cấp truyền dữ liệu đáng tin cậy, một số dữ liệu được gửi bởi quá trình gửi có thể không bao giờ đến quy trình nhận. Điều này có thể được chấp nhận đối với **các ứng dụng chịu tổn thất**, đáng chú ý nhất là các ứng dụng đa phương tiện như âm thanh / video đàm thoại có thể chịu được một số lượng mất dữ liệu. Trong các ứng dụng đa phương tiện này, dữ liệu bị mất có thể dẫn đến một trục trặc nhỏ trong âm thanh / video — không phải là một khiếm khuyết nghiêm trọng.

Thông qua

Trong Chương 1, chúng tôi đã giới thiệu khái niệm thông lượng khả dụng, trong bối cảnh phiên giao tiếp giữa hai quy trình dọc theo đường dẫn mạng, là tốc độ mà quá trình gửi có thể cung cấp các bit đến quy trình nhận. Bởi vì các phiên khác sẽ chia sẻ băng thông dọc theo đường dẫn mạng và vì các phiên khác này sẽ đến và đi, thông lượng khả dụng có thể dao động theo thời gian. Những quan sát này dẫn đến một dịch vụ tự nhiên khác mà giao thức lớp vận chuyển có thể cung cấp, cụ thể là đảm bảo thông lượng có sẵn ở một số tốc độ được chỉ định. Với dịch vụ như vậy, ứng dụng có thể yêu cầu thông lượng được đảm bảo là r bit / giây và giao thức truyền tải sau đó sẽ đảm bảo rằng thông lượng có thể sẵn sàng luôn ít nhất là r bit / giây. Một dịch vụ thông lượng được đảm bảo như vậy sẽ thu hút nhiều ứng dụng. Ví dụ: nếu ứng dụng điện thoại Internet mã hóa giọng nói ở tốc độ 32 kbps, nó cần gửi dữ liệu vào mạng và có dữ liệu được gửi đến ứng dụng nhận ở tốc độ này. Nếu giao thức truyền tải không thể cung cấp thông lượng này, ứng dụng sẽ cần mã hóa ở tốc độ thấp hơn (và nhận đủ thông lượng để duy trì tốc độ mã hóa thấp hơn này) hoặc có thể phải từ bỏ, vì việc nhận, giả sử, một nửa thông lượng cần thiết ít hoặc không sử dụng cho ứng dụng điện thoại Internet này. Các ứng dụng có yêu cầu thông lượng được cho là **các ứng dụng nhạy cảm với băng thông**. Nhiều ứng dụng đa phương tiện hiện tại nhạy cảm với băng thông, mặc dù một số ứng dụng đa phương tiện có thể sử dụng thích ứng

Các kỹ thuật mã hóa để mã hóa giọng nói hoặc video được số hóa với tốc độ phù hợp với thông lượng có sẵn.

Mặc dù các ứng dụng nhảy cảm với băng thông có các yêu cầu thông lượng cụ thể, **các ứng dụng đàn hồi** có thể sử dụng nhiều hoặc ít thông lượng nếu có. Thư điện tử, truyền tệp và chuyển Web đều là các ứng dụng đàn hồi. Tất nhiên, thông lượng càng nhiều thì càng tốt. Có một câu ngạn ngữ nói rằng một người không thể quá giàu, quá gầy hoặc có quá nhiều thông lượng!

Thời gian

Một giao thức lớp vận chuyển cũng có thể cung cấp đảm bảo thời gian. Cũng như đảm bảo thông lượng, đảm bảo thời gian có thể có nhiều hình dạng và hình thức. Một đảm bảo ví dụ có thể là mỗi bit mà người gửi bơm vào ổ cắm sẽ đến ổ cắm của người nhận không quá 100 mili giây sau đó. Một dịch vụ như vậy sẽ hấp dẫn đối với các ứng dụng thời gian thực tương tác, chẳng hạn như điện thoại Internet, môi trường ảo, hội nghị từ xa và trò chơi nhiều người chơi, tất cả đều yêu cầu các ràng buộc về thời gian chặt chẽ đối với việc phân phối dữ liệu để có hiệu quả, xem [Gauthier 1999; Ramjee 1994]. Ví dụ, sự chậm trễ lâu trong điện thoại Internet có xu hướng dẫn đến tạm dừng không tự nhiên trong cuộc trò chuyện; Trong trò chơi nhiều người chơi hoặc môi trường tương tác ảo, độ trễ lâu giữa việc thực hiện một hành động và xem phản hồi từ môi trường (ví dụ: từ một người chơi khác ở cuối cuộc trò chuyện từ đầu đến cuối) làm cho ứng dụng cảm thấy kém thực tế hơn. Đối với các ứng dụng không theo thời gian thực, độ trễ thấp hơn luôn được ưu tiên hơn độ trễ cao hơn, nhưng không có ràng buộc chặt chẽ nào được đặt ra đối với độ trễ từ đầu đến cuối.

An ninh

Cuối cùng, một giao thức truyền tải có thể cung cấp một ứng dụng với một hoặc nhiều dịch vụ bảo mật. Ví dụ, trong máy chủ gửi, một giao thức vận chuyển có thể mã hóa tất cả dữ liệu được truyền bởi quá trình gửi và trong máy chủ nhận, pro-tocol lớp vận chuyển có thể giải mã dữ liệu trước khi cung cấp dữ liệu cho quá trình nhận. Một dịch vụ như vậy sẽ cung cấp tính bảo mật giữa hai quy trình, ngay cả khi dữ liệu được quan sát bằng cách nào đó giữa các quy trình gửi và nhận. Giao thức truyền tải cũng có thể cung cấp các dịch vụ bảo mật khác ngoài tính bảo mật, bao gồm tính toàn vẹn dữ liệu và xác thực điểm cuối, các chủ đề mà chúng tôi sẽ đề cập chi tiết trong Chương 8.

2.1.4 Dịch vụ vận tải được cung cấp bởi Internet

Cho đến thời điểm này, chúng tôi đã xem xét các dịch vụ vận tải mà một mạng máy tính *có thể* cung cấp nói chung. Bây giờ chúng ta hãy cụ thể hơn và kiểm tra loại dịch vụ vận tải được cung cấp bởi Internet. Internet (và, nói chung hơn, mạng TCP / IP) làm cho hai giao thức truyền tải có sẵn cho các ứng dụng, UDP và TCP. Khi bạn (với tư cách là nhà phát triển ứng dụng) tạo một ứng dụng mạng mới cho

Ứng dụng	Data	Thông qua	Nhảy cảm về
Truyền / tải xuống tệp	Không mất mát	Thun	Không
Thư điện tử	Không mất mát	Thun	Không
Tài liệu web	Không mất mát	Đàn hồi (vài kbps)	Không
Điện thoại Internet/ Hội nghị truyền hình	Khả năng chịu tổn thất	Âm thanh: vài kbps–1 Mbps Video: 10 kbps–5 Mbps	Có: 100 mili giây
Phát trực tuyến âm thanh / video được lưu trữ	Khả năng chịu tổn thất	Tương tự như trên	Có: vài giây
Trò chơi tương tác	Khả năng chịu tổn thất	Vài kbps–10 kbps	Có: 100 mili giây
Nhắn tin trên điện thoại thông minh	Không mất mát	Thun	Có và không

Hình 2.4 ♦ Yêu cầu của các ứng dụng mạng được chọn

Internet, một trong những quyết định đầu tiên bạn phải đưa ra là nên sử dụng UDP hay TCP. Mỗi giao thức này cung cấp một bộ dịch vụ khác nhau cho các ứng dụng gọi. Hình 2.4 cho thấy các yêu cầu dịch vụ cho một số ứng dụng được chọn.

Dịch vụ TCP

Mô hình dịch vụ TCP bao gồm dịch vụ định hướng kết nối và dịch vụ truyền dữ liệu đáng tin cậy. Khi một ứng dụng gọi TCP làm giao thức truyền tải của nó, ứng dụng sẽ nhận được cả hai dịch vụ này từ TCP.

- *Dịch vụ định hướng kết nối.* TCP có thông tin điều khiển lớp vận chuyển trao đổi máy khách và máy chủ với nhau *trước khi* các nhà hiền triết cấp ứng dụng bắt đầu chạy. Cái gọi là thủ tục bắt tay này cảnh báo máy khách và máy chủ, cho phép họ chuẩn bị cho sự tấn công dữ dội của các gói. Sau giai đoạn bắt tay, một **kết nối TCP** được cho là tồn tại giữa các socket của hai quá trình. Kết nối là một kết nối song công đầy đủ trong đó hai quá trình có thể gửi tin nhắn cho nhau qua kết nối cùng một lúc. Khi ứng dụng gửi tin nhắn xong, nó phải xé bỏ connection. Trong Chương 3, chúng ta sẽ thảo luận chi tiết về dịch vụ hướng kết nối và kiểm tra cách nó được triển khai.
- *Dịch vụ truyền dữ liệu đáng tin cậy.* Các quy trình giao tiếp có thể dựa vào TCP để cung cấp tất cả dữ liệu được gửi mà không có lỗi và theo đúng thứ tự. Khi một bên của ứng dụng truyền một luồng byte vào một socket, nó có thể tin tưởng vào TCP để cung cấp cùng một luồng byte đến socket nhận, không có byte bị thiếu hoặc trùng lặp.

TCP cũng bao gồm một cơ chế kiểm soát tắc nghẽn, một dịch vụ vì phúc lợi chung của Internet chứ không phải vì lợi ích trực tiếp của các chuyên gia truyền thông. Cơ chế kiểm soát tắc nghẽn TCP điều tiết quá trình gửi (máy khách hoặc máy chủ) khi mạng bị tắc nghẽn giữa người gửi và người nhận. Như chúng ta sẽ thấy trong Chương 3, kiểm soát tắc nghẽn TCP cũng cố gắng giới hạn mỗi kết nối TCP trong phần băng thông mạng công bằng của nó.

Dịch vụ UDP

UDP là một giao thức vận chuyển nhẹ, không rườm rà, cung cấp các dịch vụ tối thiểu. UDP không kết nối, vì vậy không có bắt tay trước khi hai quá trình bắt đầu giao tiếp. UDP cung cấp dịch vụ truyền dữ liệu không đáng tin cậy — nghĩa là, khi một quá trình gửi tin nhắn vào ổ cắm UDP, UDP *không* đảm bảo rằng tin nhắn sẽ đến được quy trình nhận. Hơn nữa, các tin nhắn đến quá trình nhận có thể đến không theo thứ tự.



TẬP TRUNG VÀO BẢO

BẢO MẬT TCP

Cả TCP và UDP đều không cung cấp bất kỳ mã hóa nào — dữ liệu mà quá trình gửi đi vào socket của nó là cùng một dữ liệu truyền qua mạng đến quá trình định mệnh. Vì vậy, ví dụ: nếu quá trình gửi gửi mật khẩu bằng văn bản rõ ràng (tức là không được mã hóa) vào ổ cắm của nó, mật khẩu văn bản rõ ràng sẽ di chuyển qua tất cả các liên kết giữa người gửi và người nhận, có khả năng bị đánh hơi và phát hiện tại bất kỳ liên kết can thiệp nào. Vì quyền riêng tư và các vấn đề bảo mật khác đã trở nên quan trọng đối với nhiều ứng dụng, cộng đồng Internet đã phát triển một cải tiến cho TCP, được gọi là **Bảo mật lớp vận chuyển** (TLS) [RFC 5246]. TCP-enhanced-with-TLS không chỉ làm mọi thứ mà TCP truyền thống làm mà còn cung cấp các dịch vụ bảo mật từ quy trình đến quy trình quan trọng, bao gồm mã hóa, toàn vẹn dữ liệu và xác thực điểm cuối. Chúng tôi nhấn mạnh rằng TLS không phải là giao thức truyền tải Internet thứ ba, cùng cấp độ với TCP và UDP, mà thay vào đó là một cải tiến của TCP, với các cải tiến đang được triển khai trong lớp ứng dụng. Đặc biệt, nếu một ứng dụng muốn sử dụng các dịch vụ của TLS, nó cần bao gồm mã TLS (các thư viện và lớp hiện có, được tối ưu hóa cao) ở cả phía máy khách và máy chủ của ứng dụng. TLS có API socket riêng tương tự như TCP socket API truyền thống. Khi một ứng dụng sử dụng TLS, quá trình gửi sẽ truyền dữ liệu văn bản rõ ràng đến ổ cắm TLS; TLS trong máy chủ gửi sau đó mã hóa dữ liệu và chuyển dữ liệu được mã hóa đến ổ cắm TCP. Dữ liệu được mã hóa truyền qua Internet đến ổ cắm TCP trong quá trình nhận.

Ổ cắm nhận chuyển dữ liệu được mã hóa đến TLS, giải mã dữ liệu. Cuối cùng, TLS chuyển dữ liệu văn bản rõ ràng qua ổ cắm TLS của nó đến quá trình nhận. Chúng tôi sẽ đề cập đến TLS một số chi tiết trong Chương 8.

UDP không bao gồm cơ chế kiểm soát tắc nghẽn, vì vậy phía gửi của UDP có thể bơm dữ liệu vào lớp bên dưới (lớp mạng) ở bất kỳ tốc độ nào nó muốn. (Tuy nhiên, lưu ý rằng thông lượng end-to-end thực tế có thể thấp hơn tốc độ này do khả năng truyền tải hạn chế của các liên kết can thiệp hoặc do tắc nghẽn).

Các dịch vụ không được cung cấp bởi các giao thức truyền tải Internet

Chúng tôi đã tổ chức các dịch vụ giao thức vận chuyển theo bốn chiều: truyền dữ liệu đáng tin cậy, thông lượng, thời gian và bảo mật. Dịch vụ nào trong số này được cung cấp bởi TCP và UDP? Chúng tôi đã lưu ý rằng TCP cung cấp truyền dữ liệu đầu cuối đáng tin cậy. Và chúng ta cũng biết rằng TCP có thể dễ dàng tăng cường ở lớp ứng dụng với TLS để cung cấp các dịch vụ bảo mật. Nhưng trong mô tả ngắn gọn của chúng tôi về TCP và UDP, thiếu rõ ràng là bất kỳ đề cập nào đến đảm bảo thông lượng hoặc thời gian — các dịch vụ *không* được cung cấp bởi các giao thức truyền tải Internet ngày nay. Điều này có nghĩa là các ứng dụng nhạy cảm với thời gian như điện thoại Internet không thể chạy trong Internet ngày nay? Câu trả lời rõ ràng là không, Internet đã lưu trữ các ứng dụng nhạy cảm về thời gian trong nhiều năm. Các ứng dụng này thường hoạt động khá tốt vì chúng được thiết kế để đối phó, ở mức độ lớn nhất có thể, với sự thiếu đảm bảo này. Tuy nhiên, thiết kế thông minh có những hạn chế của nó khi độ trễ là quá mức, hoặc thông lượng end-to-end bị hạn chế. Tóm lại, Internet ngày nay thường có thể cung cấp dịch vụ thỏa đáng cho các ứng dụng nhạy cảm với thời gian, nhưng nó không thể cung cấp bất kỳ đảm bảo thời gian hoặc thông lượng nào.

Hình 2.5 chỉ ra các giao thức truyền tải được sử dụng bởi một số ứng dụng Internet phổ biến. Chúng tôi thấy rằng e-mail, truy cập thiết bị đầu cuối từ xa, Web và truyền tệp đều sử dụng TCP. Các ứng dụng này đã chọn TCP chủ yếu vì TCP cung cấp truyền dữ liệu đáng tin cậy, đảm bảo rằng tất cả dữ liệu cuối cùng sẽ đến đích. Bởi vì các ứng dụng điện thoại Internet (như Skype) thường có thể chịu được một số mất mát nhưng yêu cầu một tỷ lệ tối thiểu để có hiệu quả, các nhà phát triển ứng dụng điện thoại Internet

Ứng dụng	Giao thức lớp ứng dụng	Giao thức vận chuyển cơ bản
Thư điện tử	SMTP [RFC 5321]	TCP
Truy cập thiết bị đầu cuối từ xa	Điện thoại [RFC 854]	TCP
Web	HTTP 1.1 [RFC 7230]	TCP
Truyền tệp tin	FTP [RFC 959]	TCP
Truyền phát đa phương tiện	HTTP (ví dụ: YouTube), DTRO	TCP
Điện thoại Internet	SIP [RFC 3261], RTP [RFC 3550] hoặc độc quyền (ví dụ: Skype)	UDP hoặc TCP

Hình 2.5 ♦ Các ứng dụng Internet phổ biến, các giao thức lớp ứng dụng và các giao thức truyền tải cơ bản của chúng

thường thích chạy các ứng dụng của họ qua UDP, do đó phá vỡ cơ chế kiểm soát tắc nghẽn và chi phí gói tin của TCP. Nhưng vì nhiều tường lửa được cấu hình để chặn (hầu hết các loại) lưu lượng UDP, các ứng dụng điện thoại Internet thường được thiết kế để sử dụng TCP làm bản sao lưu nếu giao tiếp UDP không thành công.

2.1.5 Giao thức lớp ứng dụng

Chúng ta vừa biết rằng các quy trình mạng giao tiếp với nhau bằng cách gửi tin nhắn vào ô cắm. Nhưng những thông điệp này được cấu trúc như thế nào? Ý nghĩa của các trường khác nhau trong các thông điệp là gì? Khi nào các quy trình gửi tin nhắn? Những câu hỏi này đưa chúng ta vào lĩnh vực của các giao thức lớp ứng dụng. **Giao thức lớp ứng dụng** xác định cách các quy trình của ứng dụng, chạy trên các hệ thống đầu cuối khác nhau, truyền thông điệp cho nhau. Cụ thể, một giao thức lớp ứng dụng định nghĩa:

- Các loại tin nhắn được trao đổi, ví dụ, tin nhắn yêu cầu và tin nhắn phản hồi
- Cú pháp của các loại thư khác nhau, chẳng hạn như các trường trong thư và cách các trường được mô tả
- Ngữ nghĩa của các trường, nghĩa là ý nghĩa của thông tin trong các trường
- Quy tắc xác định thời điểm và cách thức một quy trình gửi tin nhắn và trả lời tin nhắn

Một số giao thức lớp ứng dụng được chỉ định trong RFC và do đó thuộc phạm vi công cộng. Ví dụ: giao thức lớp ứng dụng của Web, HTTP (Giao thức truyền siêu văn bản [RFC 7230]), có sẵn dưới dạng RFC. Nếu một nhà phát triển trình duyệt tuân theo các quy tắc của HTTP RFC, trình duyệt sẽ có thể truy xuất các trang Web từ bất kỳ máy chủ Web nào cũng đã tuân theo các quy tắc của HTTP RFC. Nhiều giao thức lớp ứng dụng khác là độc quyền và cố tình không có sẵn trong phạm vi công cộng. Ví dụ: Skype sử dụng các giao thức lớp ứng dụng độc quyền.

Điều quan trọng là phải phân biệt giữa các ứng dụng mạng và các giao thức lớp ứng dụng. Giao thức lớp ứng dụng chỉ là một phần của ứng dụng mạng (mặc dù, một phần rất quan trọng của ứng dụng theo quan điểm của chúng tôi!). Hãy xem xét một vài ví dụ. Web là một ứng dụng máy khách-máy chủ cho phép người dùng lấy tài liệu từ các máy chủ Web theo yêu cầu. Ứng dụng Web bao gồm nhiều thành phần, bao gồm một tiêu chuẩn cho các định dạng tài liệu (nghĩa là HTML), trình duyệt Web (ví dụ: Chrome và Microsoft Internet Explorer), máy chủ Web (ví dụ: máy chủ Apache và Microsoft) và giao thức lớp ứng dụng. Giao thức lớp ứng dụng của Web, HTTP, xác định định dạng và chuỗi thông điệp được trao đổi giữa trình duyệt và máy chủ Web. Do đó, HTTP chỉ là một phần (mặc dù, một phần quan trọng) của ứng dụng Web. Một ví dụ khác, chúng ta sẽ thấy trong Phần 2.6 rằng dịch vụ video của Netflix cũng có nhiều thành phần,

bao gồm các máy chủ lưu trữ và truyền video, các máy chủ khác quản lý thanh toán và các chức năng máy khách khác, máy khách (ví dụ: ứng dụng Netflix trên điện thoại thông minh, máy tính bảng hoặc máy tính) và giao thức DASH cấp ứng dụng xác định định dạng và chuỗi tin nhắn được trao đổi giữa máy chủ và máy khách Netflix. Do đó, DASH chỉ là một phần (mặc dù, một phần quan trọng) của ứng dụng Netflix.

2.1.6 Các ứng dụng mạng được đề cập trong cuốn sách này

Các ứng dụng mới đang được phát triển mỗi ngày. Thay vì bao gồm một số lượng lớn các ứng dụng Internet theo cách bách khoa, chúng tôi đã chọn tập trung vào một số lượng nhỏ các ứng dụng vừa phổ biến vừa quan trọng. Trong chương này, chúng ta thảo luận về năm ứng dụng quan trọng: Web, thư điện tử, dịch vụ thư mục, truyền phát video và ứng dụng P2P. Đầu tiên chúng ta thảo luận về Web, không chỉ vì nó là một ứng dụng cực kỳ phổ biến, mà còn vì giao thức lớp ứng dụng của nó, HTTP, rất đơn giản và dễ hiểu. Sau đó chúng ta thảo luận về thư điện tử, ứng dụng giết người đầu tiên của Internet. E-mail phức tạp hơn Web theo nghĩa là nó sử dụng không chỉ một mà là các giao thức lớp ứng dụng *several*. Sau e-mail, chúng tôi bao gồm DNS, cung cấp dịch vụ thư mục cho Internet. Hầu hết người dùng không tương tác trực tiếp với DNS; thay vào đó, người dùng gọi DNS gián tiếp thông qua các ứng dụng khác (bao gồm Web, truyền tệp và thư điện tử). DNS minh họa độc đáo cách một phần chức năng mạng cốt lõi (dịch tên mạng sang địa chỉ mạng) có thể được triển khai ở lớp ứng dụng trên Internet. Sau đó, chúng tôi thảo luận về các ứng dụng chia sẻ tệp P2P và hoàn thành nghiên cứu ứng dụng của mình bằng cách thảo luận về phát trực tuyến video theo yêu cầu, bao gồm phân phối video được lưu trữ qua mạng phân phối nội dung.

2.2 Web và HTTP

Cho đến đầu những năm 1990, Internet được sử dụng chủ yếu bởi các nhà nghiên cứu, học giả và sinh viên đại học để đăng nhập vào các máy chủ từ xa, để chuyển các tệp từ máy chủ cục bộ sang máy chủ từ xa và ngược lại, để nhận và gửi tin tức, và để nhận và gửi thư *elec-tronic*. Mặc dù các ứng dụng này đã (và tiếp tục) cực kỳ hữu ích, Internet về cơ bản không được biết đến bên ngoài các cộng đồng học thuật và nghiên cứu. Sau đó, vào đầu những năm 1990, một ứng dụng mới lớn đã xuất hiện — World Wide Web [Berners-Lee 1994]. Web là ứng dụng Internet đầu tiên thu hút sự chú ý của công chúng. Nó đã thay đổi đáng kể cách mọi người tương tác bên trong và bên ngoài môi trường làm việc của họ. Nó đã nâng cấp Internet từ chỉ một trong nhiều mạng dữ liệu lên cơ bản là mạng dữ liệu duy nhất.

Có lẽ điều hấp dẫn nhất đối với người dùng là Web hoạt động *theo yêu cầu*. Người dùng nhận được những gì họ muốn, khi họ muốn. Điều này không giống như phát sóng truyền thống

đài phát thanh và truyền hình, buộc người dùng phải điều chỉnh khi nhà cung cấp nội dung cung cấp nội dung. Ngoài việc có sẵn theo yêu cầu, Web còn có nhiều tính năng tuyệt vời khác mà mọi người yêu thích và trân trọng. Rất dễ dàng cho bất kỳ cá nhân nào để cung cấp thông tin qua Web — mọi người đều có thể trở thành nhà xuất bản với chi phí cực kỳ thấp. Các siêu liên kết và công cụ tìm kiếm giúp chúng ta điều hướng qua một đại dương thông tin. Hình ảnh và video kích thích các giác quan của chúng ta. Biểu mẫu, JavaScript, video và nhiều thiết bị khác cho phép chúng tôi tương tác với các trang và trang web. Và Web và các giao thức của nó đóng vai trò là nền tảng cho YouTube, e-mail dựa trên web (như Gmail) và hầu hết các ứng dụng Internet di động, bao gồm Instagram và Google Maps.

2.2.1 Tổng quan về HTTP

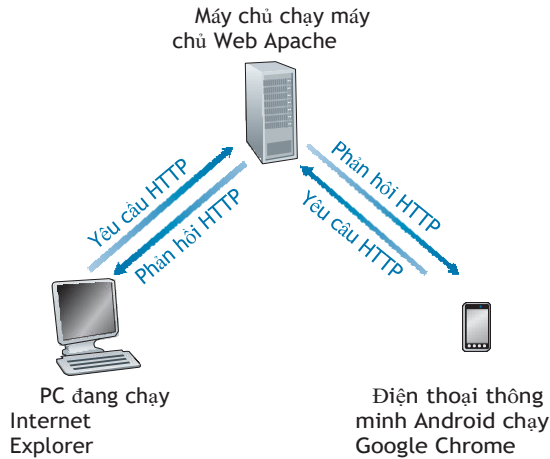
HyperText Transfer Protocol (HTTP), giao thức lớp ứng dụng của Web, là trung tâm của Web. Nó được định nghĩa trong [RFC 1945], [RFC 7230] và [RFC 7540]. HTTP được triển khai trong hai chương trình: chương trình máy khách và chương trình máy chủ. Chương trình máy khách và chương trình máy chủ, thực thi trên các hệ thống đầu cuối khác nhau, nói chuyện với nhau bằng cách trao đổi thông điệp HTTP. HTTP xác định cấu trúc của các thông điệp này và cách máy khách và máy chủ trao đổi thông điệp. Trước khi giải thích chi tiết HTTP, chúng ta nên xem xét một số thuật ngữ Web.

Một **trang Web** (còn được gọi là tài liệu) bao gồm các đối tượng. **Đối tượng** chi đơn giản là một tệp — chẳng hạn như tệp HTML, hình ảnh JPEG, tệp Javascript, tệp biểu định kiểu CCS hoặc video clip — có thể định địa chỉ bằng một URL duy nhất. Hầu hết các trang Web bao gồm một **tệp HTML cơ sở** và một số đối tượng được tham chiếu. Ví dụ: nếu một trang Web chứa văn bản HTML và năm hình ảnh JPEG, thì trang Web có sáu đối tượng: tệp HTML cơ sở cộng với năm hình ảnh. Tệp HTML cơ sở tham chiếu đến các đối tượng khác trong trang với URL của đối tượng. Mỗi URL có hai thành phần: tên máy chủ của máy chủ chứa đối tượng và tên đường dẫn của đối tượng. Ví dụ: URL

`http://www.someSchool.edu/someDepartment/picture.gif`

có `www.someSchool.edu` cho tên máy chủ và `/someDepartment/picture.gif` cho tên đường dẫn. Bởi vì **các trình duyệt Web** (như Internet Explorer và Chrome) triển khai phía máy khách của HTTP, trong ngữ cảnh của Web, chúng tôi sẽ sử dụng các từ *trình duyệt* và *máy khách* thay thế cho nhau. **Các máy chủ web**, thực hiện phía máy chủ của HTTP, chứa các đối tượng Web, mỗi địa chỉ bằng một URL. Các máy chủ Web phổ biến bao gồm Apache và Microsoft Internet Information Server.

HTTP định nghĩa cách các máy khách Web yêu cầu các trang Web từ các máy chủ Web và cách các máy chủ chuyển các trang Web sang máy khách. Chúng ta thảo luận chi tiết về sự tương tác giữa máy khách và máy chủ sau, nhưng ý tưởng chung được minh họa trong Hình 2.6. Khi người dùng yêu cầu một trang Web (ví dụ, bấm vào một siêu kết nối), trình duyệt sẽ gửi



Hình 2.6 ♦ Hành vi yêu cầu-phản hồi HTTP

Thông báo yêu cầu HTTP cho các đối tượng trong trang đến máy chủ. Máy chủ nhận các yêu cầu và phản hồi bằng các thông báo phản hồi HTTP có chứa các đối tượng.

HTTP sử dụng TCP làm giao thức vận chuyển cơ bản của nó (thay vì chạy trên UDP). Máy khách HTTP trước tiên bắt đầu kết nối TCP với máy chủ. Khi kết nối được thiết lập, trình duyệt và máy chủ xử lý truy cập TCP thông qua giao diện socket của chúng. Như được mô tả trong Phần 2.1, ở phía máy khách, giao diện ổ cắm là cửa giữa quá trình máy khách và kết nối TCP; ở phía máy chủ, nó là cánh cửa giữa quá trình máy chủ và kết nối TCP. Máy khách gửi thông báo yêu cầu HTTP vào giao diện socket của nó và nhận phản hồi HTTP messages từ giao diện socket của nó. Tương tự, máy chủ HTTP nhận thông báo yêu cầu từ giao diện socket của nó và gửi tin nhắn phản hồi vào giao diện socket của nó. Khi máy khách gửi tin nhắn vào giao diện socket của nó, tin nhắn sẽ nằm ngoài tay khách hàng và "nằm trong tay" TCP. Hãy nhớ lại từ Phần 2.1 rằng TCP cung cấp dịch vụ truyền dữ liệu đáng tin cậy đến HTTP. Điều này ngụ ý rằng mỗi thông điệp yêu cầu HTTP được gửi bởi một quy trình máy khách cuối cùng sẽ đến máy chủ nguyên vẹn; tương tự, mỗi thông điệp phản hồi HTTP được gửi bởi quá trình máy chủ cuối cùng sẽ đến nguyên vẹn tại máy khách. Ở đây chúng ta thấy một trong những lợi thế lớn của kiến trúc phân lớp - HTTP không cần phải lo lắng về dữ liệu bị mất hoặc chi tiết về cách TCP phục hồi sau khi mất hoặc sắp xếp lại dữ liệu trong mạng. Đó là công việc của TCP và các giao thức ở các lớp dưới của ngăn xếp giao thức.

Điều quan trọng cần lưu ý là máy chủ gửi các tệp được yêu cầu đến máy khách mà không lưu trữ bất kỳ thông tin trạng thái nào về máy khách. Nếu một máy khách cụ thể yêu cầu cùng một đối tượng hai lần trong khoảng thời gian vài giây, máy chủ sẽ không phản hồi bằng cách nói rằng nó chỉ phục vụ đối tượng cho máy khách; Thay vào đó, máy chủ gửi lại đối tượng, vì nó đã hoàn toàn quên những gì nó đã làm trước đó. Bởi vì một máy chủ HTTP duy trì

không có thông tin về các máy khách, HTTP được cho là một **giao thức không trạng thái**. Chúng tôi cũng lưu ý rằng Web sử dụng kiến trúc ứng dụng máy khách-máy chủ, như được mô tả trong Phần 2.1. Một máy chủ Web luôn bật, với một địa chỉ IP cố định và nó phục vụ các yêu cầu từ hàng triệu trình duyệt khác nhau.

Phiên bản gốc của HTTP được gọi là HTTP / 1.0 và có từ đầu những năm 1990 [RFC 1945]. Tính đến năm 2020, phần lớn các giao dịch HTTP diễn ra qua HTTP / 1.1 [RFC 7230]. Tuy nhiên, ngày càng nhiều trình duyệt và máy chủ Web cũng cung cấp một phiên bản HTTP mới được gọi là HTTP / 2 [RFC 7540]. Ở cuối phần này, chúng tôi sẽ giới thiệu về HTTP / 2.

2.2.2 Kết nối không liên tục và liên tục

Trong nhiều ứng dụng Internet, máy khách và máy chủ giao tiếp trong một khoảng thời gian dài, với máy khách thực hiện một loạt các yêu cầu và máy chủ phản hồi từng yêu cầu. Tùy thuộc vào ứng dụng và cách ứng dụng đang được sử dụng, một loạt các yêu cầu có thể được thực hiện liên tục, *periodically* theo định kỳ hoặc không liên tục. Khi tương tác máy khách-máy chủ này diễn ra qua TCP, nhà phát triển ứng dụng cần đưa ra quyết định quan trọng — mỗi cặp yêu cầu / phản hồi nên được gửi qua một kết nối TCP *riêng biệt* hay tất cả các yêu cầu và phản hồi tương ứng của chúng nên được gửi qua *cùng một* kết nối TCP? Trong cách tiếp cận trước, ứng dụng được cho là sử dụng **các kết nối không liên tục**; và trong cách tiếp cận sau, **các kết nối liên tục**. Để hiểu sâu sắc về vấn đề thiết kế này, chúng ta hãy xem xét những ưu điểm và nhược điểm của các kết nối liên tục trong bối cảnh của một ứng dụng cụ thể, cụ thể là HTTP, có thể sử dụng cả kết nối không liên tục và kết nối liên tục. Mặc dù HTTP sử dụng các kết nối liên tục ở chế độ mặc định, các máy khách và máy chủ HTTP có thể được cấu hình để sử dụng các kết nối không liên tục thay thế.

HTTP với các kết nối không liên tục

Chúng ta hãy đi qua các bước chuyển một trang Web từ máy chủ sang máy khách cho trường hợp kết nối không liên tục. Giả sử trang bao gồm một tệp HTML cơ sở và 10 hình ảnh JPEG và tất cả 11 đối tượng này đều nằm trên cùng một máy chủ. Giả sử thêm URL cho tệp HTML cơ sở là

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

Đây là những gì xảy ra:

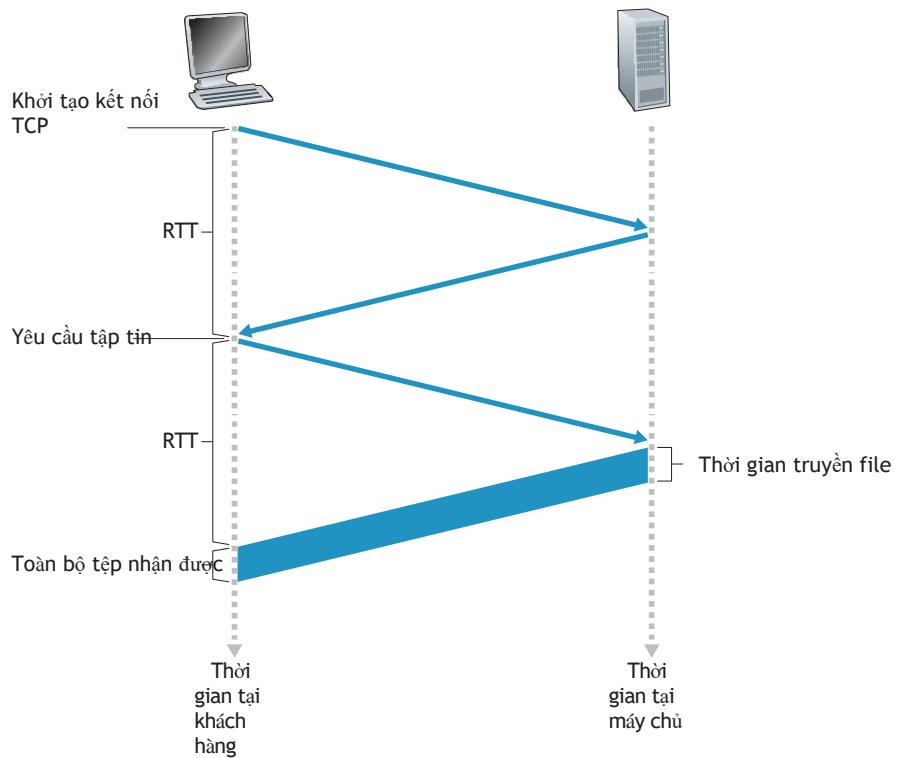
1. Quá trình máy khách HTTP khởi tạo kết nối TCP với máy chủ `www.someSchool.edu` trên cổng số 80, là số cổng mặc định cho HTTP. Liên kết với kết nối TCP, sẽ có một ổ cắm ở máy khách và một ổ cắm tại máy chủ.

2. Máy khách HTTP gửi thông báo yêu cầu HTTP đến máy chủ thông qua socket của nó. Thông báo yêu cầu bao gồm tên đường dẫn `/someDepartment/home.index`. (Chúng ta sẽ thảo luận về các thông điệp HTTP một số chi tiết bên dưới.)
3. Quá trình máy chủ HTTP nhận thông báo yêu cầu thông qua socket của nó, truy xuất đối tượng `/someDepartment/home.index` từ bộ nhớ của nó (RAM hoặc đĩa), đóng gói đối tượng trong thông báo phản hồi HTTP và gửi thông báo phản hồi đến máy khách thông qua socket của nó.
4. Quá trình máy chủ HTTP yêu cầu TCP đóng kết nối TCP. (Nhưng TCP không thực sự chấm dứt kết nối cho đến khi nó biết chắc chắn rằng máy khách đã nhận được thông báo phản hồi còn nguyên vẹn.)
5. Máy khách HTTP nhận được thông báo phản hồi. Kết nối TCP chấm dứt. Thông báo chỉ ra rằng đối tượng đóng gói là một tệp HTML. Máy khách trích xuất tệp từ thông báo phản hồi, kiểm tra tệp HTML và tìm tham chiếu đến 10 đối tượng JPEG.
6. Bốn bước đầu tiên sau đó được lặp lại cho mỗi đối tượng JPEG được tham chiếu.

Khi trình duyệt nhận được trang Web, nó sẽ hiển thị trang cho người dùng. Hai trình duyệt khác nhau có thể diễn giải (nghĩa là hiển thị cho người dùng) một trang Web theo một số cách khác nhau. HTTP không liên quan gì đến cách một trang web được giải thích bởi một khách hàng. Thông số kỹ thuật HTTP ([RFC 1945] và [RFC 7540]) chỉ xác định giao thức truyền thông giữa chương trình HTTP máy khách và chương trình HTTP máy chủ.

Các bước trên minh họa việc sử dụng các kết nối không liên tục, trong đó mỗi kết nối TCP được đóng sau khi máy chủ gửi đối tượng—kết nối không tồn tại cho các đối tượng khác. HTTP / 1.0 sử dụng các kết nối TCP không liên tục. Lưu ý rằng mỗi kết nối TCP không liên tục vận chuyển chính xác một thông báo yêu cầu và một thông báo phản hồi. Do đó, trong ví dụ này, khi người dùng yêu cầu trang Web, 11 kết nối TCP được tạo ra.

Trong các bước được mô tả ở trên, chúng tôi đã cố tình mơ hồ về việc liệu máy khách có nhận được 10 JPEG trên 10 kết nối TCP nối tiếp hay liệu một số JPEG có thu được qua các kết nối TCP song song hay không. Thật vậy, người dùng có thể cấu hình một số trình duyệt để kiểm soát mức độ song song. Trình duyệt có thể mở nhiều TCP con-nections và yêu cầu các phần khác nhau của trang Web qua nhiều kết nối. Như chúng ta sẽ thấy trong chương tiếp theo, việc sử dụng các kết nối song song rút ngắn thời gian phản hồi. Trước khi tiếp tục, hãy thực hiện phép tính sau phong bì để ước tính lượng thời gian trôi qua từ khi máy khách yêu cầu tệp HTML cơ sở cho đến khi máy khách nhận được toàn bộ tệp. Để kết thúc này, chúng tôi xác định **thời gian khứ hồi (RTT)**, là thời gian cần thiết để một gói nhỏ di chuyển từ máy khách đến máy chủ và sau đó quay lại máy khách. RTT bao gồm độ trễ lan truyền gói, độ trễ xếp hàng gói trong các bộ định tuyến và thiết bị chuyển mạch trung gian và độ trễ xử lý gói. (Những sự chậm trễ này đã được thảo luận trong Phần 1.4.) Bây giờ hãy xem xét điều gì sẽ xảy ra khi người dùng nhấp vào siêu liên kết. Như thể hiện trong Hình 2.7, điều này khiến trình duyệt bắt đầu kết nối TCP giữa trình duyệt và máy chủ Web; Điều này liên quan đến



Hình 2.7 ♦ Tính toán mặt sau phong bì cho thời gian cần thiết để yêu cầu và nhận tệp HTML

"bắt tay ba chiều" — máy khách gửi một phân đoạn TCP nhỏ đến máy chủ, máy chủ thừa nhận và phản hồi bằng một phân đoạn TCP nhỏ và cuối cùng, cli-ent thừa nhận trở lại máy chủ. Hai phần đầu tiên của cái bắt tay ba chiều lấy một RTT. Sau khi hoàn thành hai phần đầu tiên của bắt tay, máy khách sẽ gửi thông báo yêu cầu HTTP kết hợp với phần thứ ba của bắt tay ba chiều (xác nhận) vào kết nối TCP. Khi thông báo yêu cầu đến máy chủ, máy chủ sẽ gửi tệp HTML vào kết nối TCP. Yêu cầu/phản hồi HTTP này ăn hết một RTT khác. Do đó, đại khái, tổng thời gian phản hồi là hai RTT cộng với thời gian truyền tại máy chủ của tệp HTML.

HTTP với kết nối liên tục

Kết nối không liên tục có một số thiếu sót. Đầu tiên, một kết nối hoàn toàn mới phải được thiết lập và duy trì cho *mỗi đối tượng được yêu cầu*. Đối với mỗi kết nối này, bộ đệm TCP phải được phân bổ và các biến TCP phải được giữ trong cả máy khách và máy chủ. Điều này có thể đặt một gánh nặng đáng kể lên máy chủ Web, có thể phục vụ các yêu cầu từ hàng trăm máy khách khác nhau cùng một lúc. Thứ hai

như chúng tôi vừa mô tả, mỗi đối tượng bị chậm trễ phân phối hai RTT — một RTT để thiết lập kết nối TCP và một RTT để yêu cầu và nhận một đối tượng.

Với kết nối liên tục HTTP / 1.1, máy chủ để kết nối TCP mở sau khi gửi phản hồi. Các yêu cầu và phản hồi tiếp theo giữa cùng một máy khách và máy chủ có thể được gửi qua cùng một kết nối. Cụ thể, toàn bộ trang Web (trong ví dụ trên, tệp HTML cơ sở và 10 hình ảnh) có thể được gửi qua một kết nối TCP liên tục duy nhất. Hơn nữa, nhiều trang Web nằm trên cùng một máy chủ có thể được gửi từ máy chủ đến cùng một máy khách qua một kết nối TCP liên tục duy nhất. Các yêu cầu này cho các đối tượng có thể được thực hiện liên tục, mà không cần chờ trả lời các yêu cầu đang chờ xử lý (pipelining). Thông thường, máy chủ HTTP đóng kết nối khi nó không được sử dụng trong một thời gian nhất định (khoảng thời gian chờ có thể định cấu hình). Khi máy chủ nhận được các yêu cầu back-to-back, nó sẽ gửi các đối tượng trở lại. Chế độ mặc định của HTTP sử dụng các kết nối liên tục với pipelining. Chúng tôi sẽ so sánh định lượng hiệu suất của các kết nối không liên tục và liên tục trong các vấn đề bài tập về nhà của Chương 2 và 3. Bạn cũng được khuyến khích xem [Heidemann 1997; Nielsen 1997; RFC 7540].

2.2.3 Định dạng thông điệp HTTP

Các thông số kỹ thuật HTTP [RFC 1945; RFC 7230; RFC 7540] bao gồm các định nghĩa của các định dạng thông điệp HTTP. Có hai loại thông điệp HTTP, yêu cầu message và thông điệp phản hồi, cả hai đều được thảo luận dưới đây.

Thông báo yêu cầu HTTP

Dưới đây chúng tôi cung cấp một thông báo yêu cầu HTTP điển hình:

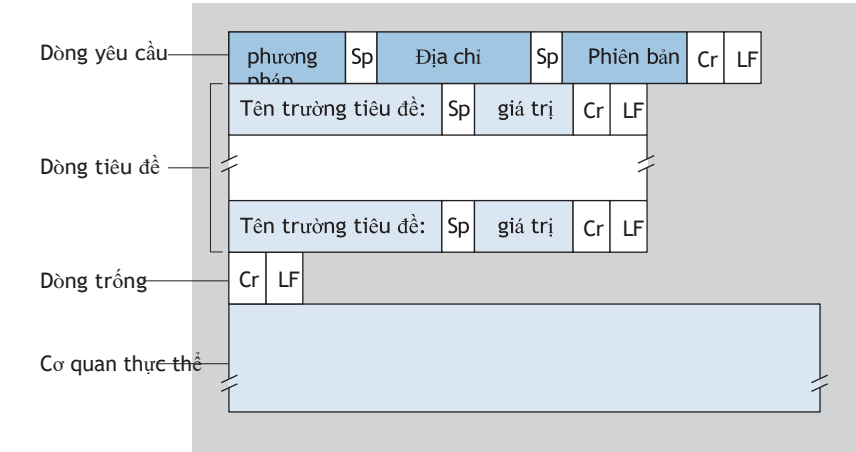
```
GET / somedir/page.html HTTP /
1.1 Máy chủ: www.someschool.edu
Kết nối: đóng
Tác nhân người dùng:
Mozilla/5.0 Ngôn ngữ
chấp nhận: fr
```

Chúng ta có thể học được rất nhiều bằng cách xem xét kỹ thông báo yêu cầu đơn giản này. Trước hết, chúng ta thấy rằng thông điệp được viết bằng văn bản ASCII thông thường, để con người biết chữ máy tính bình thường của bạn có thể đọc nó. Thứ hai, chúng ta thấy rằng thông điệp bao gồm năm dòng, mỗi dòng theo sau là một dòng trả về vận chuyển và một nguồn cấp dữ liệu dòng. Dòng cuối cùng là fol- thấp bởi một trở lại vận chuyển bổ sung và nguồn cấp dữ liệu dòng. Mặc dù thông báo yêu cầu cụ thể này có năm dòng, một thông báo yêu cầu có thể có nhiều dòng hơn hoặc ít nhất là một dòng. Dòng đầu tiên của thông báo yêu cầu HTTP được gọi là **dòng yêu cầu**; các dòng tiếp theo được gọi là **dòng tiêu đề**. Dòng yêu cầu có ba trường: trường phương thức, trường URL và trường phiên bản HTTP. Trường phương thức có thể nhận một số giá trị khác nhau, bao gồm GET, POST, HEAD, PUT và DELETE.

Phần lớn các thông báo yêu cầu HTTP sử dụng phương thức GET. Phương thức GET được sử dụng khi trình duyệt yêu cầu một đối tượng, với đối tượng được yêu cầu được xác định trong trường URL. Trong ví dụ này, trình duyệt đang yêu cầu đối tượng /somedir/ page.html. Phiên bản này là tự giải thích; trong ví dụ này, trình duyệt ngụ ý phiên bản HTTP / 1.1.

Bây giờ chúng ta hãy nhìn vào các dòng tiêu đề trong ví dụ. Dòng tiêu đề Host : www.someschool.edu chỉ định máy chủ lưu trữ mà đối tượng cư trú. Bạn có thể nghĩ rằng dòng tiêu đề này là không cần thiết, vì đã có kết nối TCP tại chỗ với máy chủ. Tuy nhiên, như chúng ta sẽ thấy trong Phần 2.2.5, thông tin được cung cấp bởi dòng tiêu đề máy chủ được yêu cầu bởi bộ đệm proxy Web. Bằng cách bao gồm dòng tiêu đề Connection: close, trình duyệt đang nói với máy chủ rằng nó không muốn bận tâm đến các kết nối liên tục; nó muốn máy chủ đóng kết nối sau khi gửi đối tượng được yêu cầu. Dòng tiêu đề User-agent : chỉ định tác nhân người dùng, nghĩa là loại trình duyệt đang thực hiện yêu cầu đến máy chủ. Ở đây tác nhân người dùng là Mozilla / 5.0, một trình duyệt Firefox. Dòng tiêu đề này rất hữu ích vì máy chủ thực sự có thể gửi các phiên bản khác nhau của cùng một đối tượng đến các loại tác nhân người dùng khác nhau. (Mỗi phiên bản được giải quyết bằng cùng một URL.) Cuối cùng, tiêu đề Accept-language : chỉ ra rằng người dùng thích nhận phiên bản tiếng Pháp của đối tượng, nếu một đối tượng như vậy tồn tại trên máy chủ; nếu không, máy chủ sẽ gửi phiên bản mặc định của nó. Tiêu đề Ngôn ngữ chấp nhận : chỉ là một trong nhiều tiêu đề đàm phán nội dung có sẵn trong HTTP.

Sau khi xem xét một ví dụ, bây giờ chúng ta hãy xem định dạng chung của một thông báo yêu cầu, như thể hiện trong Hình 2.8. Chúng tôi thấy rằng định dạng chung bám sát ví dụ trước đó của chúng tôi. Tuy nhiên, bạn có thể nhận thấy rằng sau các dòng tiêu đề (và trả về vận chuyển bổ sung và nguồn cấp dữ liệu dòng) có một "cơ thể thực thể". Cơ quan thực thể



Hình 2.8 ♦ Định dạng chung của một thông điệp yêu cầu HTTP

trống với phương thức GET, nhưng được sử dụng với phương thức POST. Máy khách HTTP thường sử dụng phương thức POST khi người dùng điền vào biểu mẫu—ví dụ: khi người dùng cung cấp từ tìm kiếm cho công cụ tìm kiếm. Với thông báo POST, người dùng vẫn yêu cầu một trang Web từ máy chủ, nhưng nội dung cụ thể của trang Web phụ thuộc vào những gì người dùng đã nhập vào các trường biểu mẫu. Nếu giá trị của trường phương thức là POST, thì phần thân thực thể chứa những gì người dùng đã nhập vào trường biểu mẫu.

Chúng tôi sẽ hồi hận nếu chúng tôi không đề cập rằng một yêu cầu được tạo bằng biểu mẫu không nhất thiết phải sử dụng phương thức POST. Thay vào đó, biểu mẫu HTML thường sử dụng phương thức GET và bao gồm dữ liệu đã nhập (trong các trường biểu mẫu) trong URL được yêu cầu. Ví dụ: nếu một biểu mẫu sử dụng phương thức GET, có hai trường và đầu vào cho hai trường là *khí* và *chuối*, thì URL sẽ có cấu trúc `www.somesite.com/animalsearch?monkeys&bananas`. Trong quá trình lướt web hàng ngày của bạn, bạn có thể đã nhận thấy các URL mở rộng thuộc loại này.

Phương thức HEAD tương tự như phương thức GET. Khi một máy chủ nhận được một yêu cầu với phương thức HEAD, nó sẽ phản hồi bằng một thông báo HTTP nhưng nó bỏ qua đối tượng được yêu cầu. Các nhà phát triển ứng dụng thường sử dụng phương pháp HEAD để gỡ lỗi. Phương pháp PUT thường được sử dụng kết hợp với các công cụ xuất bản Web. Nó cho phép người dùng tải một đối tượng lên một đường dẫn (thư mục) cụ thể trên một máy chủ Web cụ thể. Phương thức PUT cũng được sử dụng bởi các ứng dụng cần tải đối tượng lên máy chủ Web. Phương pháp DELETE cho phép người dùng hoặc ứng dụng xóa một đối tượng trên máy chủ Web.

Thông báo phản hồi HTTP

Dưới đây chúng tôi cung cấp một thông báo phản hồi HTTP điển hình. Thông báo phản hồi này có thể là phản hồi cho thông báo yêu cầu mẫu vừa được thảo luận.

```
HTTP / 1.1 200 OK
Kết nối: đóng
Ngày: Tue, 18 Aug 2015 15:44:04 GMT
Máy chủ: Apache / 2.2.3 (CentOS)
Sửa đổi lần cuối: Tue, 18 Aug 2015 15:11:03 GMT
Độ dài nội dung: 6821
Loại nội dung: văn bản /
html
(dữ liệu, dữ liệu, dữ liệu, dữ liệu, dữ liệu...)
```

Chúng ta hãy xem xét cẩn thận thông báo phản hồi này. Nó có ba phần: một **dòng trạng thái** ban đầu, sáu **dòng tiêu đề** và sau đó là **phần thân thực thể**. Phần thân thực thể là phần thịt của thông điệp — nó chứa chính đối tượng được yêu cầu (được biểu thị bằng dữ liệu dữ liệu, dữ liệu, dữ liệu, dữ liệu, dữ liệu . . .). Dòng trạng thái có ba trường: trường phiên bản giao thức, mã trạng thái và thông báo trạng thái tương ứng. Trong ví dụ này, dòng trạng thái cho biết rằng máy chủ đang sử dụng HTTP / 1.1 và mọi thứ đều ổn (nghĩa là máy chủ đã tìm thấy và

Bây giờ chúng ta hãy nhìn vào các dòng tiêu đề. Máy chủ sử dụng dòng tiêu đề `Connection: close` để cho máy khách biết rằng nó sẽ đóng kết nối TCP sau khi gửi tin nhắn. Dòng tiêu đề `Date:` cho biết thời gian và ngày khi phản hồi HTTP được tạo và gửi bởi máy chủ. Lưu ý rằng đây không phải là thời điểm đối tượng được tạo hoặc sửa đổi lần cuối; Đó là thời gian máy chủ truy xuất đối tượng từ hệ thống tệp của nó, chèn đối tượng vào thông báo phản hồi và gửi thông báo phản hồi. Dòng tiêu đề `Server:` chỉ ra rằng thông điệp được tạo bởi một máy chủ Web Apache; nó tương tự như dòng tiêu đề `User-agent:` trong thông báo yêu cầu HTTP. Dòng tiêu đề `Sửa đổi lần cuối:` cho biết thời gian và ngày khi đối tượng được tạo hoặc sửa đổi lần cuối. Tiêu đề `Sửa đổi lần cuối:` mà chúng tôi sẽ sớm đề cập chi tiết hơn, rất quan trọng đối với bộ nhớ đệm đối tượng, cả trong máy khách cục bộ và trong máy chủ bộ nhớ cache mạng (còn được gọi là máy chủ proxy). Dòng tiêu đề `Content-Length:` cho biết số byte trong đối tượng được gửi. Dòng tiêu đề `Content-Type:` chỉ ra rằng đối tượng trong nội dung thực thể là văn bản HTML. (Loại đối tượng được chỉ định chính thức bởi `Content-Type:` header chứ không phải bởi phần mở rộng tệp.)

Sau khi xem xét một ví dụ, bây giờ chúng ta hãy kiểm tra định dạng chung của một tin nhắn phản hồi, được hiển thị trong Hình 2.9. Định dạng chung này của tin nhắn phản hồi khớp với ví dụ trước về tin nhắn phản hồi. Hãy nói thêm một vài từ về mã trạng thái và cụm từ của chúng. Mã trạng thái và cụm từ liên quan cho biết kết quả của yêu cầu. Một số mã trạng thái phổ biến và cụm từ liên quan bao gồm:

- 200 OK: Yêu cầu thành công và thông tin được trả về trong phản hồi.
- 301 Di chuyển vĩnh viễn: Đối tượng được yêu cầu đã được di chuyển vĩnh viễn; URL mới được chỉ định trong Vị trí: tiêu đề của thông báo phản hồi. Phần mềm máy khách sẽ tự động truy xuất URL mới.



Hình 2.9 ♦ Định dạng chung của một thông điệp phản hồi HTTP

- 400 Bad Request: Đây là mã lỗi chung cho biết rằng máy chủ không thể hiểu được yêu cầu.
- 404 Không tìm thấy: Tài liệu được yêu cầu không tồn tại trên máy chủ này.
- Phiên bản HTTP 505 không được hỗ trợ: Phiên bản giao thức HTTP được yêu cầu không được máy chủ hỗ trợ.

Bạn muốn thấy một thông báo phản hồi HTTP thực sự như thế nào? Điều này rất dễ thực hiện và rất dễ thực hiện! Đầu tiên Telnet vào máy chủ Web yêu thích của bạn. Sau đó nhập thông báo yêu cầu một dòng cho một số đối tượng được đặt trên máy chủ. Ví dụ: nếu bạn có quyền truy cập vào dấu nhắc lệnh, hãy nhập:

```
Telnet gaia.cs.umass.edu 80
```

```
GET /kurose_ross/interactive/index.php HTTP/1.1 Máy
chủ: gaia.cs.umass.edu
```

(Nhấn trả về vận chuyển hai lần sau khi nhập dòng cuối cùng.) Điều này mở một TCP connection cổng 80 của máy chủ gaia.cs.umass.edu và sau đó gửi thông báo yêu cầu HTTP. Bạn sẽ thấy một thông báo phản hồi bao gồm tệp HTML cơ sở cho các bài tập về nhà tương tác cho sách giáo khoa này. Nếu bạn chỉ muốn nhìn thấy các dòng thông báo HTTP và không nhận được chính đối tượng, hãy thay thế GET bằng HEAD.

Trong phần này, chúng tôi đã thảo luận về một số dòng tiêu đề có thể được sử dụng trong

Thông báo yêu cầu và phản hồi HTTP. Đặc tả HTTP xác định nhiều, nhiều dòng tiêu đề hơn có thể được chèn bởi các trình duyệt, máy chủ Web và máy chủ bộ nhớ cache công việc mạng. Chúng tôi chỉ bao gồm một số lượng nhỏ trong tổng số các dòng tiêu đề. Chúng tôi sẽ đề cập đến một vài điều dưới đây và một số nhỏ khác khi chúng tôi thảo luận về bộ nhớ đệm Web mạng trong Phần 2.2.5. Một phân tích rất dễ đọc và toàn diện của giao thức HTTP, bao gồm các tiêu đề và mã trạng thái của nó, được đưa ra trong [Krishnamurthy 2001].

Làm thế nào để trình duyệt quyết định dòng tiêu đề nào sẽ bao gồm trong thông báo yêu cầu? Làm thế nào để một máy chủ Web quyết định dòng tiêu đề nào sẽ bao gồm trong một message phản hồi? Trình duyệt sẽ tạo các dòng tiêu đề như một chức năng của loại và phiên bản trình duyệt, cấu hình người dùng của trình duyệt và liệu trình duyệt hiện có phiên bản đối tượng được lưu trong bộ nhớ cache nhưng có thể lỗi thời hay không. Máy chủ web hoạt động tương tự: Có các sản phẩm, phiên bản và cấu hình khác nhau, tất cả đều ảnh hưởng đến dòng tiêu đề nào được bao gồm trong tin nhắn phản hồi.

2.2.4 Tương tác giữa người dùng và máy chủ: Cookie

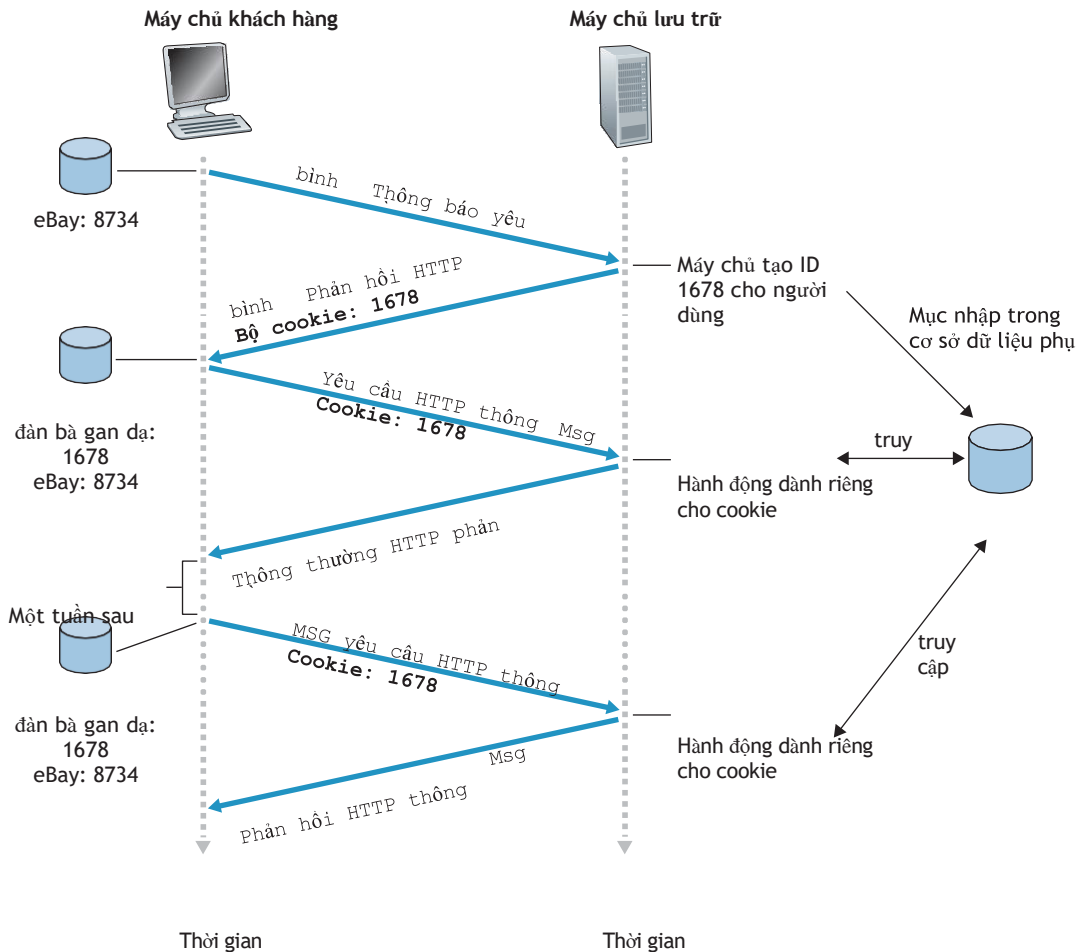
Chúng tôi đã đề cập ở trên rằng một máy chủ HTTP là không trạng thái. Điều này giúp đơn giản hóa thiết kế máy chủ và đã cho phép các kỹ sư phát triển các máy chủ Web hiệu suất cao có thể sử dụng hàng ngàn kết nối TCP đồng thời. Tuy nhiên, nó thường là mong muốn cho một trang web để xác định người dùng, hoặc bởi vì máy chủ muốn hạn chế truy cập của người dùng



Ghi chú video
Sử dụng Wireshark
để điều tra giao thức
HTTP

hoặc vì nó muốn phục vụ nội dung như một chức năng của danh tính người dùng. Đối với những mục đích này, HTTP sử dụng cookie. Cookie, được định nghĩa trong [RFC 6265], cho phép các trang web theo dõi người dùng. Hầu hết các trang web thương mại lớn đều sử dụng cookie ngày nay.

Như thể hiện trong Hình 2.10, công nghệ cookie có bốn thành phần: (1) dòng tiêu đề cookie trong thông báo phản hồi HTTP; (2) dòng tiêu đề cookie trong thông báo yêu cầu HTTP; (3) tệp cookie được lưu giữ trên hệ thống cuối của người dùng và được quản lý bởi trình duyệt của người dùng; và (4) một cơ sở dữ liệu back-end tại trang web. Sử dụng Hình 2.10, chúng ta hãy đi qua một ví dụ về cách cookie hoạt động. Giả sử Susan, người luôn luôn



Hình 2.10 ♦

Giữ trạng thái
người dùng
với cookie

truy cập Web bằng Internet Explorer từ PC tại nhà của cô, danh bạ Amazon.com lần đầu tiên. Chúng ta hãy giả sử rằng trong quá khứ cô ấy đã truy cập trang eBay. Khi yêu cầu đến máy chủ Amazon Web, máy chủ sẽ tạo một số nhận dạng duy nhất và tạo một mục nhập trong cơ sở dữ liệu phụ trợ được lập chỉ mục theo số nhận dạng. Máy chủ web của Amazon sau đó phản hồi trình duyệt của Susan, bao gồm trong phản hồi HTTP tiêu đề `Set-cookie:` chứa số nhận dạng. Ví dụ: dòng tiêu đề có thể là:

```
Bộ cookie: 1678
```

Khi trình duyệt của Susan nhận được thông báo phản hồi HTTP, nó sẽ thấy tiêu đề `Set-cookie:`. Sau đó, trình duyệt sẽ thêm một dòng vào tệp cookie đặc biệt mà nó quản lý. Dòng này bao gồm tên máy chủ của máy chủ và số nhận dạng trong tiêu đề `Set-cookie:`. Lưu ý rằng tệp cookie đã có mục nhập cho eBay, vì Susan đã truy cập trang web đó trong quá khứ. Khi Susan tiếp tục duyệt trang Amazon, mỗi khi cô yêu cầu một trang web, trình duyệt của cô sẽ tham khảo tệp cookie của cô, trích xuất số nhận dạng của cô cho trang web này và đặt dòng tiêu đề cookie bao gồm số nhận dạng trong yêu cầu HTTP. Cụ thể, mỗi yêu cầu HTTP của cô đến máy chủ Amazon bao gồm dòng tiêu đề:

```
Cookie: 1678
```

Bằng cách này, máy chủ Amazon có thể theo dõi hoạt động của Susan tại trang Amazon. Mặc dù trang web Amazon không nhất thiết phải biết tên của Susan, nhưng nó biết chính xác người dùng 1678 đã truy cập trang nào, theo thứ tự nào và vào thời gian nào! Amazon sử dụng cookie để cung cấp dịch vụ giỏ hàng của mình — Amazon có thể duy trì danh sách tất cả các giao dịch mua dự định của Susan để cô có thể thanh toán chung vào cuối phiên.

Nếu Susan quay trở lại trang web của Amazon, giả sử, một tuần sau, trình duyệt của cô sẽ tiếp tục đặt dòng tiêu đề `Cookie: 1678` trong các thông báo yêu cầu. Amazon cũng giới thiệu sản phẩm cho Susan dựa trên các trang web mà cô đã truy cập tại Amazon trong quá khứ. Nếu Susan cũng tự đăng ký với Amazon — cung cấp tên đầy đủ, địa chỉ e-mail, địa chỉ bưu chính và thông tin thẻ tín dụng — Amazon sau đó có thể đưa thông tin này vào cơ sở dữ liệu của mình, do đó liên kết tên của Susan với số nhận dạng của cô ấy (và tất cả các trang cô ấy đã truy cập tại trang web trong quá khứ!). Đây là cách Amazon và các trang web thương mại điện tử khác cung cấp "mua sắm bằng một cú nhấp chuột" — khi Susan chọn mua một mặt hàng trong lần truy cập tiếp theo, cô ấy không cần phải nhập lại tên, số thẻ tín dụng hoặc địa chỉ của mình.

Từ cuộc thảo luận này, chúng tôi thấy rằng cookie có thể được sử dụng để xác định người dùng. Lần đầu tiên người dùng truy cập một trang web, người dùng có thể cung cấp nhận dạng người dùng (có thể là tên của họ). Trong các phiên tiếp theo, trình duyệt chuyển tiêu đề cookie đến máy chủ, do đó xác định người dùng đến máy chủ. Do đó, cookie có thể được sử dụng để tạo lớp phiên người dùng trên HTTP không trạng thái. Ví dụ: khi người dùng đăng nhập vào

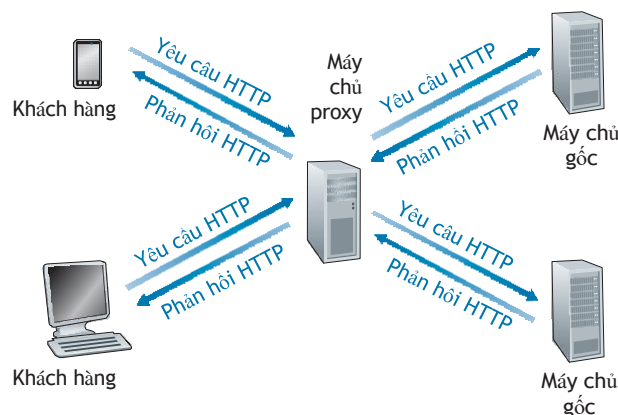
một ứng dụng e-mail dựa trên Web (như Hotmail), trình duyệt gửi thông tin cookie đến máy chủ, cho phép máy chủ xác định người dùng trong suốt phiên của người dùng với ứng dụng.

Mặc dù cookie thường đơn giản hóa trải nghiệm mua sắm trên Internet cho người dùng, nhưng chúng gây tranh cãi vì chúng cũng có thể được coi là xâm phạm quyền riêng tư. Như chúng ta vừa thấy, sử dụng kết hợp cookie và thông tin tài khoản do người dùng cung cấp, một trang web có thể tìm hiểu rất nhiều về người dùng và có khả năng bán thông tin này cho bên thứ ba.

2.2.5 Bộ nhớ đệm web

Bộ đệm ẩn Web — còn được gọi là **máy chủ proxy** — là một thực thể mạng đáp ứng các yêu cầu HTTP thay mặt cho máy chủ Web gốc. Bộ đệm ẩn Web có bộ nhớ đĩa riêng và giữ các bản sao của các đối tượng được yêu cầu gần đây trong bộ nhớ này. Như thể hiện trong Hình 2.11, trình duyệt của người dùng có thể được cấu hình sao cho tất cả các yêu cầu HTTP của người dùng trước tiên được chuyển đến bộ đệm Web [RFC 7234]. Khi trình duyệt được cấu hình, mỗi yêu cầu trình duyệt cho một đối tượng trước tiên được chuyển đến bộ đệm ẩn Web. Ví dụ, giả sử một trình duyệt đang yêu cầu đối tượng `http://www.someschool.edu/campus.gif`. Đây là những gì xảy ra:

1. Trình duyệt thiết lập kết nối TCP với bộ đệm ẩn Web và gửi yêu cầu HTTP cho đối tượng vào bộ đệm ẩn Web.
2. Bộ nhớ cache Web kiểm tra xem nó có bản sao của đối tượng được lưu trữ cục bộ hay không. Nếu có, bộ đệm Web trả về đối tượng trong thông báo phản hồi HTTP cho trình duyệt máy khách.
3. Nếu bộ đệm Web không có đối tượng, bộ đệm ẩn Web sẽ mở một kết nối TCP với máy chủ gốc, nghĩa là `www.someschool.edu`. Bộ nhớ cache Web



Hình 2.11 ♦ Client request objects thông qua Web cache

sau đó gửi một yêu cầu HTTP cho đối tượng vào bộ nhớ cache đến máy chủ TCP connection. Sau khi nhận được yêu cầu này, máy chủ gốc sẽ gửi đối tượng trong phản hồi HTTP đến bộ đệm ẩn Web.

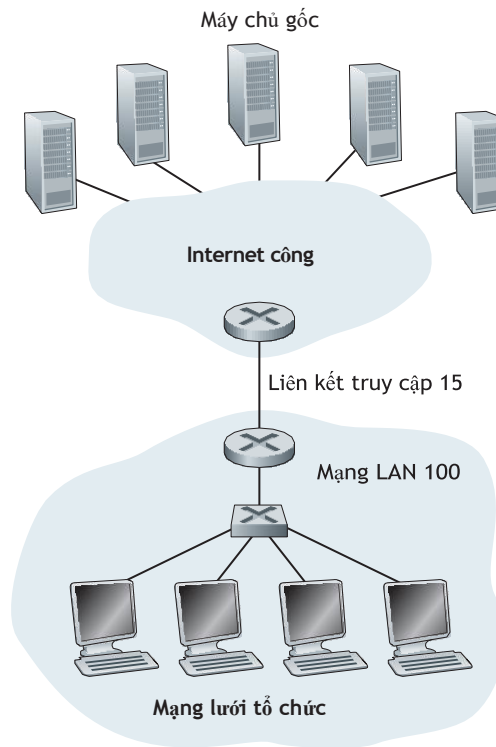
4. Khi bộ đệm ẩn Web nhận được đối tượng, nó lưu trữ một bản sao trong bộ nhớ cục bộ của nó và gửi một bản sao, trong một thông báo phản hồi HTTP, đến trình duyệt máy khách (qua kết nối TCP hiện có giữa trình duyệt máy khách và bộ đệm ẩn Web).

Lưu ý rằng bộ nhớ cache vừa là máy chủ vừa là máy khách cùng một lúc. Khi nó nhận được yêu cầu từ và gửi phản hồi đến trình duyệt, nó là một máy chủ. Khi nó gửi yêu cầu đến và nhận phản hồi từ một máy chủ gốc, nó là một máy khách.

Thông thường, một bộ nhớ cache Web được mua và cài đặt bởi ISP. Ví dụ: một uni-versity có thể cài đặt bộ nhớ cache trên mạng campus của nó và định cấu hình tất cả các trình duyệt trong khuôn viên trường để trở về bộ nhớ cache. Hoặc một ISP dân cư lớn (chẳng hạn như Comcast) có thể cài đặt một hoặc nhiều bộ nhớ cache trong mạng của mình và định cấu hình trước các trình duyệt được vận chuyển để trở về bộ nhớ cache đã cài đặt.

Web caching đã được triển khai trên Internet vì hai lý do. Đầu tiên, bộ đệm Web có thể làm giảm đáng kể thời gian phản hồi cho yêu cầu của máy khách, đặc biệt nếu băng thông cổ chai giữa máy khách và máy chủ gốc nhỏ hơn nhiều so với băng thông tắc nghẽn giữa máy khách và bộ đệm. Nếu có kết nối tốc độ cao giữa máy khách và bộ đệm, như thường có và nếu bộ đệm có đối tượng được yêu cầu, thì bộ đệm sẽ có thể phân phối đối tượng nhanh chóng đến máy khách. Thứ hai, như chúng tôi sẽ sớm minh họa bằng một ví dụ, bộ đệm Web có thể làm giảm đáng kể lưu lượng truy cập vào liên kết truy cập Internet của một tổ chức. Bằng cách giảm lưu lượng truy cập, tổ chức (ví dụ: công ty hoặc trường đại học) không phải nâng cấp băng thông nhanh chóng, do đó giảm chi phí. Hơn nữa, bộ đệm Web có thể làm giảm đáng kể lưu lượng truy cập Web trên Internet nói chung, do đó cải thiện hiệu suất cho tất cả các ứng dụng.

Để hiểu sâu hơn về lợi ích của bộ nhớ cache, chúng ta hãy xem xét một bài kiểm tra trong bối cảnh của Hình 2.12. Hình này cho thấy hai mạng - mạng thể chế và phần còn lại của Internet công cộng. Mạng thể chế là một mạng LAN tốc độ cao. Một bộ định tuyến trong mạng tổ chức và một bộ định tuyến trên Internet được kết nối bằng liên kết 15 Mbps. Các máy chủ gốc được gắn vào Internet nhưng được đặt trên toàn cầu. Giả sử rằng kích thước đối tượng trung bình là 1 Mbits và tốc độ yêu cầu trung bình từ trình duyệt của tổ chức đến máy chủ gốc là 15 yêu cầu mỗi giây. Giả sử rằng các thông điệp yêu cầu HTTP nhỏ không đáng kể và do đó không tạo ra lưu lượng truy cập trong mạng hoặc trong liên kết truy cập (từ bộ định tuyến tổ chức đến bộ định tuyến Internet). Cũng giả sử rằng khoảng thời gian cần thiết từ khi bộ định tuyến ở phía Internet của liên kết truy cập trong Hình 2.12 chuyển tiếp một yêu cầu HTTP (trong sơ đồ dữ liệu IP) cho đến khi nó nhận được phản hồi (thường là trong nhiều sơ đồ dữ liệu IP) trung bình là hai giây. Một cách không chính thức, chúng tôi gọi sự chậm trễ cuối cùng này là "độ trễ Internet". Tổng thời gian phản hồi — nghĩa là thời gian từ yêu cầu của trình duyệt đối tượng cho đến khi nhận được đối tượng — là tổng độ trễ mạng LAN, độ trễ truy cập (nghĩa là độ trễ giữa hai bộ định tuyến) và độ trễ Internet. Bây giờ chúng ta hãy làm



Hình 2.12 ♦ Nút thắt cổ chai giữa mạng thể chế và Internet

Một tính toán rất thô thiển để ước tính sự chậm trễ này. Cường độ lưu lượng trên mạng LAN (xem Phần 1.4.2) là

$$(15 \text{ yêu cầu/giây}) (1 \text{ Mbit/yêu cầu}) / (100 \text{ Mbps}) = 0,15$$

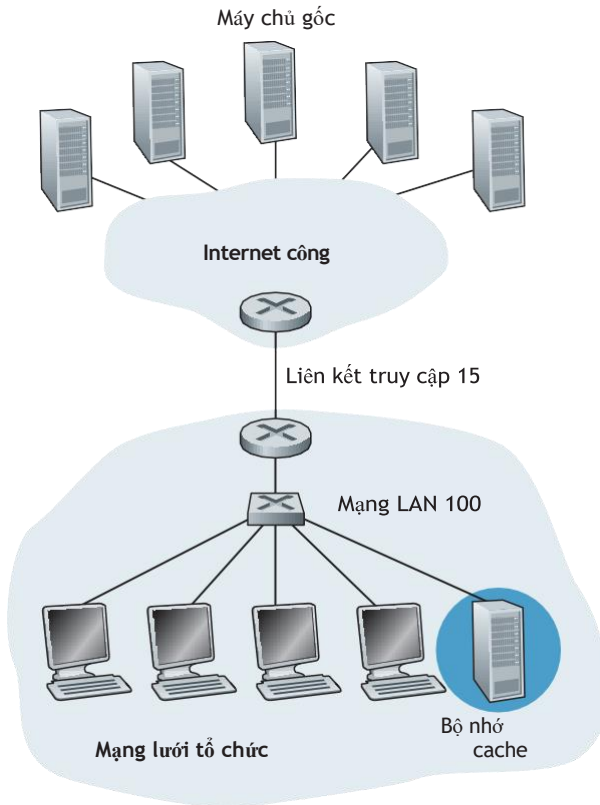
trong khi cường độ lưu lượng truy cập trên liên kết truy cập (từ bộ định tuyến Internet đến bộ định tuyến tổ chức) là

$$(15 \text{ yêu cầu/giây}) (1 \text{ Mbit/yêu cầu}) / (15 \text{ Mbps}) = 1$$

Cường độ lưu lượng 0,15 trên mạng LAN thường dẫn đến, nhiều nhất, hàng chục mili giây-onds độ trễ; do đó, chúng ta có thể bỏ qua độ trễ mạng LAN. Tuy nhiên, như đã thảo luận trong Phần 1.4.2, khi cường độ lưu lượng tiếp cận 1 (như trường hợp của liên kết truy cập trong Hình 2.12), độ trễ trên một liên kết trở nên rất lớn và phát triển không bị ràng buộc. Do đó, thời gian phản hồi trung bình để đáp ứng các yêu cầu sẽ theo thứ tự phút, nếu không muốn nói là nhiều hơn, điều này không thể chấp nhận được đối với người dùng của tổ chức. Rõ ràng một cái gì đó phải được thực hiện.

Một giải pháp khả thi là tăng tốc độ truy cập từ 15 Mbps lên 100 Mbps. Điều này sẽ làm giảm cường độ lưu lượng truy cập trên liên kết truy cập xuống 0,15, điều này dẫn đến độ trễ không đáng kể giữa hai bộ định tuyến. Trong trường hợp này, tổng thời gian phản hồi sẽ khoảng hai giây, nghĩa là độ trễ Internet. Nhưng giải pháp này cũng có nghĩa là viện phải nâng cấp liên kết truy cập từ 15 Mbps lên 100 Mbps, một đề xuất tốn kém.

Bây giờ hãy xem xét giải pháp thay thế là không nâng cấp liên kết truy cập mà thay vào đó cài đặt bộ đệm Web trong mạng tổ chức. Giải pháp này được minh họa trong Hình 2.13. Tỷ lệ truy cập — phần nhỏ các yêu cầu được bộ nhớ đệm thỏa mãn — thường nằm trong khoảng từ 0,2 đến 0,7 trong thực tế. Đối với mục đích minh họa, giả sử rằng bộ nhớ cache cung cấp tỷ lệ truy cập là 0,4 cho tổ chức này. Bởi vì các máy khách và bộ nhớ cache được kết nối với cùng một mạng LAN tốc độ cao, 40 phần trăm các yêu cầu sẽ được thỏa mãn gần như ngay lập tức, giả sử, trong vòng 10 mili giây, bởi bộ nhớ cache. Không bao giờ, 60% yêu cầu còn lại vẫn cần được thỏa mãn bởi các máy chủ gốc. Nhưng chỉ với 60 phần trăm các đối tượng được yêu cầu đi qua liên kết truy cập, cường độ lưu lượng truy cập trên liên kết truy cập giảm từ 1,0 xuống 0,6. Thông thường, một



Hình 2.13 ♦ Thêm bộ nhớ cache vào mạng tổ chức

Cường độ lưu lượng nhỏ hơn 0,8 tương ứng với một độ trễ nhỏ, giả sử, hàng chục mili giây, trên liên kết 15 Mbps. Độ trễ này là không đáng kể so với độ trễ Internet hai giây. Với những cân nhắc này, độ trễ trung bình do đó là

$$0,4 \text{ (0,01 giây)} + 0,6 \text{ (2,01 giây)}$$

chỉ lớn hơn 1,2 giây một chút. Do đó, giải pháp thứ hai này cung cấp thời gian phản hồi thậm chí còn thấp hơn giải pháp đầu tiên và nó không yêu cầu tổ chức nâng cấp liên kết của mình với Internet. Tất nhiên, tổ chức này phải mua và cài đặt bộ nhớ cache Web. Nhưng chi phí này thấp—nhiều bộ nhớ cache sử dụng phần mềm miễn công cộng chạy trên các PC rẻ tiền.

Thông qua việc sử dụng **Mạng phân phối nội dung (CDN)**, bộ đệm web đang ngày càng đóng một vai trò quan trọng trong Internet. Một công ty CDN cài đặt nhiều bộ đệm phân tán theo địa lý trên Internet, do đó bản địa hóa phần lớn lưu lượng truy cập. Có các CDN được chia sẻ (chẳng hạn như Akamai và Limelight) và CDN chuyên dụng (chẳng hạn như Google và Netflix). Chúng ta sẽ thảo luận chi tiết hơn về CDN trong Phần 2.6.

GET có điều kiện

Mặc dù bộ nhớ đệm có thể làm giảm thời gian phản hồi do người dùng nhận thức, nhưng nó đưa ra một vấn đề mới — bản sao của một đối tượng nằm trong bộ nhớ cache có thể đã cũ. Nói cách khác, đối tượng nằm trong máy chủ Web có thể đã được sửa đổi kể từ khi bản sao được lưu trữ tại máy khách. May mắn thay, HTTP có một cơ chế cho phép bộ nhớ cache xác minh rằng các đối tượng của nó được cập nhật. Cơ chế này được gọi là **GET có điều kiện** [RFC 7232]. Thông báo yêu cầu HTTP được gọi là thông báo GET có điều kiện nếu:

(1) thông báo yêu cầu sử dụng phương thức GET và (2) thông báo yêu cầu bao gồm một Nền-Sửa đổi-Kể từ: dòng tiêu đề.

Để minh họa cách GET có điều kiện hoạt động, chúng ta hãy đi qua một ví dụ. Đầu tiên, thay mặt cho trình duyệt yêu cầu, bộ đệm proxy sẽ gửi thông báo yêu cầu đến máy chủ Web:

```
GET / fruit / kiwi.gif HTTP /
1.1 Máy chủ:
www.exotiquecuisine.com
```

Thứ hai, máy chủ Web gửi một thông báo phản hồi với đối tượng được yêu cầu đến bộ nhớ cache:

```
HTTP / 1.1 200 OK
Ngày: Sat, 3 Oct 2015 15:39:29
Máy chủ: Apache/1.3.0 (Unix)
Sửa đổi lần cuối: Wed, 9 Sep 2015
09:23:24 Loại nội dung: hình ảnh / gif

(dữ liệu, dữ liệu, dữ liệu, dữ liệu, dữ liệu...)
```


Bộ nhớ cache chuyển tiếp đối tượng đến trình duyệt yêu cầu nhưng cũng lưu trữ đối tượng cục bộ. Điều quan trọng, bộ nhớ cache cũng lưu trữ ngày sửa đổi cuối cùng cùng với đối tượng. Thứ ba, một tuần sau, một trình duyệt khác yêu cầu cùng một đối tượng thông qua bộ nhớ cache và đối tượng vẫn còn trong bộ nhớ cache. Vì đối tượng này có thể đã được sửa đổi tại máy chủ Web trong tuần qua, bộ nhớ cache thực hiện kiểm tra cập nhật bằng cách phát hành GET có điều kiện. Cụ thể, bộ nhớ cache gửi:

```
GET / fruit / kiwi.gif HTTP /
1.1 Máy chủ:
www.exotiquecuisine.com
Nếu-sửa đổi-từ: Wed, 9 Sep 2015 09:23:24
```

Lưu ý rằng giá trị của dòng tiêu đề `If-modified-since:` chính xác bằng giá trị của dòng tiêu đề `Last-Modified:` được gửi bởi máy chủ một tuần trước. GET có điều kiện này yêu cầu máy chủ chỉ gửi đối tượng nếu đối tượng đã được sửa đổi kể từ ngày được chỉ định. Giả sử đối tượng đã không được sửa đổi kể từ 9 Sep 2015 09:23:24. Sau đó, thứ tư, máy chủ Web gửi một thông báo phản hồi đến bộ nhớ cache:

```
HTTP / 1.1 304 Không sửa đổi
Ngày: Sat, 10 Oct 2015 15:39:29
Máy chủ: Apache/1.3.0 (Unix)
(phần thân thực thể trống)
```

Chúng ta thấy rằng để đáp ứng với GET có điều kiện, máy chủ Web vẫn gửi thông báo phản hồi nhưng không bao gồm đối tượng được yêu cầu trong thông báo phản hồi. Bao gồm đối tượng được yêu cầu sẽ chỉ lãng phí băng thông và tăng thời gian phản hồi nhận thức của người dùng, đặc biệt nếu đối tượng lớn. Lưu ý rằng thông báo phản hồi cuối cùng này có 304 Not Modified trong dòng trạng thái, cho bộ nhớ cache biết rằng nó có thể tiếp tục và chuyển tiếp bản sao được lưu trong bộ nhớ cache (bộ đệm proxy) của đối tượng đến trình duyệt yêu cầu.

2.2.6 HTTP / 2

HTTP / 2 [RFC 7540], được chuẩn hóa vào năm 2015, là phiên bản mới đầu tiên của HTTP kể từ HTTP / 1.1, được chuẩn hóa vào năm 1997. Kể từ khi tiêu chuẩn hóa, HTTP / 2 đã cất cánh, với hơn 40% trong số 10 triệu trang web hàng đầu hỗ trợ HTTP / 2 vào năm 2020 [W3Techs]. Hầu hết các trình duyệt—bao gồm Google Chrome, Internet Explorer, Safari, Opera và Firefox—cũng hỗ trợ HTTP/2.

Mục tiêu chính của HTTP / 2 là giảm độ trễ nhận thức bằng cách cho phép ghép kênh yêu cầu và phản hồi trên một kết nối TCP duy nhất, cung cấp ưu tiên yêu cầu và đẩy máy chủ và cung cấp nén hiệu quả các trường tiêu đề HTTP. HTTP / 2 không thay đổi phương thức HTTP, mã trạng thái, URL hoặc trường tiêu đề. Thay vào đó, HTTP / 2 thay đổi cách dữ liệu được định dạng và vận chuyển giữa máy khách và máy chủ.

Để thúc đẩy nhu cầu về HTTP / 2, hãy nhớ lại rằng HTTP / 1.1 sử dụng các kết nối TCP liên tục, cho phép một trang Web được gửi từ máy chủ đến máy khách qua một kết nối TCP duy nhất. Bằng cách chỉ có một kết nối TCP trên mỗi trang web, số lượng sock-et tại máy chủ được giảm và mỗi trang web được vận chuyển nhận được một phần bằng thông mạng hợp lý (như được thảo luận dưới đây). Nhưng các nhà phát triển trình duyệt Web nhanh chóng phát hiện ra rằng việc gửi tất cả các đối tượng trong một trang web qua một kết nối TCP duy nhất có **vấn đề chặn Head of Line (HOL)**. Để hiểu chặn HOL, hãy xem xét một trang Web bao gồm trang cơ sở HTML, một video clip lớn gần đầu trang Web và nhiều đối tượng nhỏ bên dưới video. Giả sử thêm có một liên kết tắc nghẽn tốc độ thấp đến trung bình (ví dụ: liên kết không dây tốc độ thấp) trên đường dẫn giữa máy chủ và máy khách. Sử dụng một kết nối TCP duy nhất, video clip sẽ mất nhiều thời gian để đi qua liên kết nút cổ chai, trong khi các đối tượng nhỏ bị trì hoãn khi chúng chờ phía sau video clip; Đó là, video clip ở đầu dòng chặn các vật thể nhỏ phía sau nó. Trình duyệt HTTP / 1.1 thường làm việc xung quanh vấn đề này bằng cách mở nhiều kết nối TCP song song, do đó có các đối tượng trong cùng một trang web được gửi song song với trình duyệt. Bằng cách này, các đối tượng nhỏ có thể đến và được hiển thị trong trình duyệt nhanh hơn nhiều, do đó giảm độ trễ do người dùng nhận thức. Kiểm soát tắc nghẽn TCP, được thảo luận chi tiết trong Chương 3, cũng cung cấp lòng

mày-
 Đây là một động lực ngoài ý muốn để sử dụng nhiều kết nối TCP song song thay vì một kết nối liên tục duy nhất. Nói một cách đơn giản, kiểm soát tắc nghẽn TCP nhằm mục đích cung cấp cho mỗi kết nối TCP chia sẻ một liên kết nút cổ chai một phần bằng nhau của băng thông có sẵn của liên kết đó; vì vậy nếu có n kết nối TCP hoạt động trên một liên kết nút cổ chai, thì mỗi kết nối xấp xỉ nhận được $1/n$ băng thông. *Bằng cách mở nhiều kết nối TCP song song để vận chuyển một trang Web duy nhất, trình duyệt có thể "gian lận" và lấy một phần lớn băng thông liên kết. Nhiều trình duyệt HTTP / 1.1 mở tối đa sáu kết nối TCP song song không chỉ để phá vỡ chặn HOL mà còn để có được nhiều băng thông hơn.*

Một trong những mục tiêu chính của HTTP / 2 là loại bỏ (hoặc ít nhất là giảm số lượng) kết nối TCP song song để vận chuyển một trang Web duy nhất. Điều này không chỉ làm giảm số lượng ổ cắm cần được mở và duy trì tại các máy chủ mà còn cho phép kiểm soát tắc nghẽn TCP hoạt động như dự định. Nhưng chỉ với một kết nối TCP để vận chuyển một trang Web, HTTP / 2 yêu cầu cơ chế được thiết kế cẩn thận để tránh chặn HOL.

Khung HTTP / 2

Giải pháp HTTP / 2 để chặn HOL là chia mỗi tin nhắn thành các khung nhỏ và xen kẽ các thông báo yêu cầu và phản hồi trên cùng một kết nối TCP. Để hiểu điều này, hãy xem xét lại ví dụ về một trang web bao gồm một video clip lớn và, giả sử, 8 đối tượng nhỏ hơn. Do đó, máy chủ sẽ nhận được 9 yêu cầu đồng thời từ bất kỳ trình duyệt nào muốn xem trang Web này. Đối với mỗi yêu cầu này, máy chủ cần gửi 9 thông báo phản hồi HTTP cạnh tranh đến trình duyệt. Giả sử tất cả các khung hình là của

Độ dài cố định, video clip bao gồm 1000 khung hình và mỗi đối tượng nhỏ hơn bao gồm hai khung hình. Với khung xen kẽ, sau khi gửi một khung hình từ video clip, các khung hình đầu tiên của mỗi đối tượng nhỏ được gửi. Sau đó, sau khi gửi khung thứ hai của video clip, các khung hình cuối cùng của mỗi đối tượng nhỏ được gửi. Do đó, tất cả các đối tượng nhỏ hơn được gửi sau khi gửi tổng cộng 18 khung. Nếu interleaving không được sử dụng, các đối tượng nhỏ hơn sẽ chỉ được gửi sau khi gửi 1016 khung. Do đó, cơ chế đóng khung HTTP / 2 có thể làm giảm đáng kể độ trễ nhận thức của người dùng. Khả năng chia nhỏ một thông điệp HTTP thành các khung độc lập, xen kẽ chúng và sau đó lắp ráp lại chúng ở đầu kia là cải tiến quan trọng nhất của HTTP / 2. Việc đóng khung được thực hiện bởi lớp con đóng khung của giao thức HTTP / 2. Khi một máy chủ muốn gửi phản hồi HTTP, phản hồi được xử lý bởi lớp con đóng khung, nơi nó được chia thành các khung. Trường tiêu đề của phản hồi trở thành một khung và nội dung của thư được chia thành một cho các khung bổ sung hơn. Các khung của phản hồi sau đó được xen kẽ bởi lớp con đóng khung trong máy chủ với các khung của các phản hồi khác và được gửi qua kết nối TCP liên tục duy nhất. Khi các khung đến máy khách, trước tiên chúng được lắp ráp lại thành các thông điệp phản hồi ban đầu ở lớp phụ đóng khung và sau đó được trình duyệt xử lý như bình thường. Tương tự, của khách hàng Các yêu cầu HTTP được chia thành các khung và xen kẽ.

Ngoài việc chia nhỏ mỗi thông điệp HTTP thành các khung độc lập, lớp con đóng khung cũng mã hóa nhị phân các khung. Các giao thức nhị phân hiệu quả hơn để phân tích cú pháp, dẫn đến các khung nhỏ hơn một chút và ít bị lỗi hơn.

Ưu tiên tin nhắn phản hồi và đẩy máy chủ

Ưu tiên tin nhắn cho phép các nhà phát triển tùy chỉnh mức độ ưu tiên tương đối của các yêu cầu để tối ưu hóa hiệu suất ứng dụng tốt hơn. Như chúng ta vừa biết, lớp phụ đóng khung tổ chức các thông điệp thành các luồng dữ liệu song song dành cho cùng một người yêu cầu. Khi một máy khách gửi các yêu cầu đồng thời đến một máy chủ, nó có thể ưu tiên các phản hồi mà nó yêu cầu bằng cách gán trọng số từ 1 đến 256 cho mỗi thư. Con số cao hơn cho thấy mức độ ưu tiên cao hơn. Sử dụng các trọng số này, máy chủ có thể gửi các khung đầu tiên cho các phản hồi với mức độ ưu tiên cao nhất. Ngoài ra, khách hàng cũng nêu rõ sự phụ thuộc của mỗi tin nhắn vào các tin nhắn khác bằng cách chỉ định ID của tin nhắn mà nó phụ thuộc.

Một tính năng khác của HTTP / 2 là khả năng cho một máy chủ gửi nhiều phản hồi cho một yêu cầu máy khách. Nghĩa là, ngoài phản hồi cho yêu cầu ban đầu, máy chủ có thể *đẩy* các đối tượng bổ sung vào máy khách mà không cần máy khách phải yêu cầu từng đối tượng. Điều này là có thể vì trang cơ sở HTML chỉ ra các đối tượng sẽ cần thiết để hiển thị đầy đủ trang Web. Vì vậy, thay vì chờ đợi các yêu cầu HTTP cho các đối tượng này, máy chủ có thể phân tích trang HTML, xác định các đối tượng cần thiết và gửi chúng đến máy khách *trước khi nhận được các yêu cầu rõ ràng cho các đối tượng này*. Server push giúp loại bỏ độ trễ thêm do chờ đợi các yêu cầu.

HTTP / 3

QUIC, được thảo luận trong Chương 3, là một giao thức "vận chuyển" mới được triển khai trong lớp ứng dụng trên giao thức UDP xương trăn. QUIC có một số tính năng mong muốn cho HTTP, chẳng hạn như ghép kênh tin nhắn (xen kẽ), kiểm soát luồng trên mỗi luồng và thiết lập kết nối có độ trễ thấp. HTTP / 3 là một giao thức HTTP mới được thiết kế để hoạt động trên QUIC. Kể từ năm 2020, HTTP / 3 được mô tả trong các bản nháp Internet và vẫn chưa được chuẩn hóa đầy đủ. Nhiều tính năng HTTP / 2 (chẳng hạn như tin nhắn xen kẽ) được gộp bởi QUIC, cho phép thiết kế đơn giản hơn, hợp lý hơn cho HTTP / 3.

2.3 Thư điện tử trên Internet

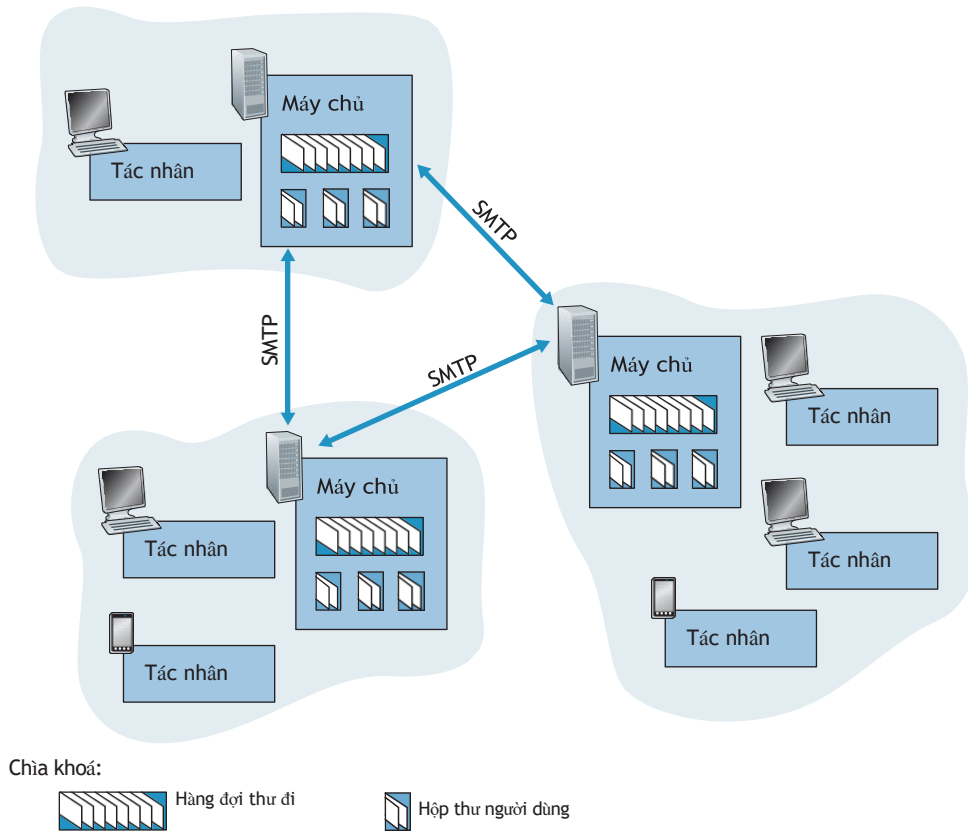
Thư điện tử đã xuất hiện từ khi bắt đầu Internet. Đây là ứng dụng phổ biến nhất khi Internet còn sơ khai [Segaller 1998], và đã trở nên phức tạp và mạnh mẽ hơn trong những năm qua. Nó vẫn là một trong những ứng dụng quan trọng nhất và được sử dụng nhiều nhất của Internet.

Như với thư bưu chính thông thường, e-mail là một phương tiện liên lạc không đồng bộ — mọi người gửi và đọc tin nhắn khi thuận tiện cho họ mà không cần phải phối hợp với lịch trình của người khác. Trái ngược với thư bưu chính, thư điện tử nhanh, dễ phân phối và không tốn kém. E-mail hiện đại có nhiều tính năng mạnh mẽ, bao gồm thư có tệp đính kèm, siêu liên kết, văn bản định dạng HTML và ảnh nhúng.

Trong phần này, chúng tôi xem xét các giao thức lớp ứng dụng là trung tâm của e-mail Internet. Nhưng trước khi chúng ta đi sâu vào thảo luận về các giao thức này, chúng ta hãy có cái nhìn cấp cao về hệ thống thư Internet và các thành phần chính của nó.

Hình 2.14 trình bày một cái nhìn cấp cao về hệ thống thư Internet. Chúng ta thấy từ sơ đồ này rằng nó có ba thành phần chính: **tác nhân người dùng**, **máy chủ thư** và **Giao thức truyền thư đơn giản (SMTP)**. Bây giờ chúng tôi mô tả từng tổng hợp này trong ngữ cảnh của người gửi, Alice, gửi thư e-mail cho người nhận, Bob. Tác nhân người dùng cho phép người dùng đọc, trả lời, chuyển tiếp, lưu và soạn thư. Ví dụ về tác nhân người dùng cho e-mail bao gồm Microsoft Outlook, Apple Mail, Gmail dựa trên web, Ứng dụng Gmail chạy trong điện thoại thông minh, v.v. Khi Alice soạn xong thư, tác nhân người dùng của cô ấy sẽ gửi thư đến máy chủ thư của cô ấy, nơi thư được đặt trong hàng đợi thư đi của máy chủ thư. Khi Bob muốn đọc thư, tác nhân người dùng của anh ấy sẽ truy xuất thư từ hộp thư của anh ấy trong máy chủ thư của anh ấy.

Máy chủ thư tạo thành cốt lõi của cơ sở hạ tầng e-mail. Mỗi người nhận, chẳng hạn như Bob, có một **hộp thư** nằm trong một trong các máy chủ thư. Hộp thư của Bob quản lý và duy trì các thư đã được gửi cho anh ta. Một thư điển hình bắt đầu hành trình của nó trong tác nhân người dùng của người gửi, sau đó chuyển đến máy chủ thư của người gửi, sau đó



Hình 2.14 ♦ Một cái nhìn cấp cao của hệ thống e-mail Internet

đi đến máy chủ thư của người nhận, nơi nó được gửi vào hộp thư của người nhận. Khi Bob muốn truy cập các thư trong hộp thư của mình, máy chủ thư chứa hộp thư của anh ấy sẽ xác thực Bob (bằng tên người dùng và mật khẩu của anh ấy). Máy chủ thư của Alice cũng phải xử lý các lỗi trong máy chủ thư của Bob. Nếu máy chủ của Alice không thể gửi thư đến máy chủ của Bob, máy chủ của Alice sẽ giữ thư trong hàng **đợi tin nhắn** và cố gắng chuyển thư sau. Thư lại thường được thực hiện cứ sau 30 phút hoặc lâu hơn; nếu không thành công sau vài ngày, máy chủ sẽ xóa thư và thông báo cho người gửi (Alice) bằng thông điệp email.

SMTP là giao thức lớp ứng dụng chính cho thư điện tử Internet. Nó sử dụng dịch vụ truyền dữ liệu đáng tin cậy của TCP để chuyển thư từ máy chủ thư của người gửi đến máy chủ thư của người nhận. Như với hầu hết các giao thức lớp ứng dụng, SMTP có hai mặt: phía máy khách, thực thi trên máy chủ thư của người gửi và phía máy chủ, thực thi trên máy chủ thư của người nhận. Cả phía máy khách và máy chủ của

SMTP chạy trên mọi máy chủ thư. Khi một máy chủ thư gửi thư đến các máy chủ thư khác, nó hoạt động như một máy khách SMTP. Khi một máy chủ thư nhận được thư từ các máy chủ thư khác, nó hoạt động như một máy chủ SMTP.

2.3.1 SMTP

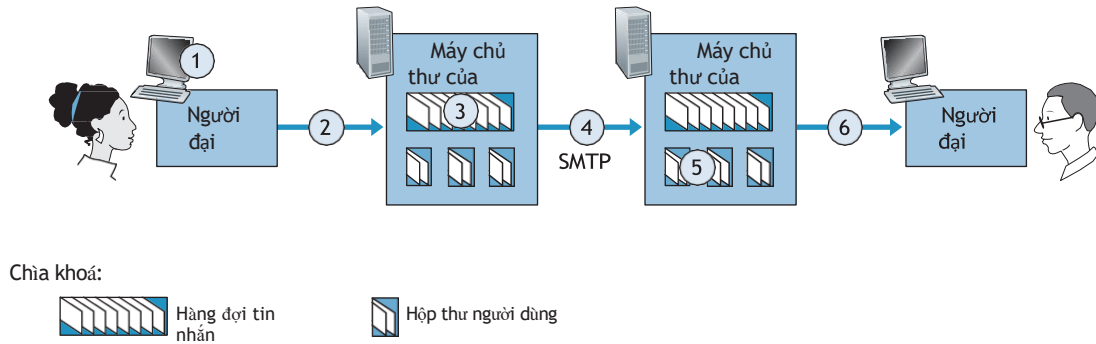
SMTP, được định nghĩa trong RFC 5321, là trung tâm của thư điện tử Internet. Như đã đề cập ở trên, SMTP chuyển thư từ máy chủ thư của người gửi đến máy chủ thư của người nhận. SMTP cũ hơn nhiều so với HTTP. (SMTP RFC ban đầu có từ năm 1982 và SMTP đã xuất hiện từ rất lâu trước đó.) Mặc dù SMTP có nhiều phẩm chất tuyệt vời, bằng chứng là sự phổ biến của nó trên Internet, tuy nhiên nó vẫn là một công nghệ kế thừa sở hữu những đặc điểm cổ xưa nhất định. Ví dụ: nó hạn chế nội dung (không chỉ tiêu đề) của tất cả các thư thành ASCII 7 bit đơn giản. Hạn chế này có ý nghĩa vào đầu những năm 1980 khi khả năng truyền tải khan hiếm và không ai gửi e-mail các tệp đính kèm lớn hoặc các tệp hình ảnh, âm thanh hoặc video lớn. Nhưng ngày nay, trong kỷ nguyên đa phương tiện, hạn chế ASCII 7 bit là một chút khó khăn — nó yêu cầu dữ liệu đa phương tiện nhị phân phải được mã hóa thành ASCII trước khi được gửi qua SMTP; và nó yêu cầu thông điệp ASCII tương ứng được giải mã trở lại nhị phân sau khi truyền tải SMTP. Hãy nhớ lại từ Phần 2.2 rằng HTTP không yêu cầu dữ liệu đa phương tiện phải được mã hóa ASCII trước khi chuyển.

Để minh họa hoạt động cơ bản của SMTP, chúng ta hãy đi qua một scenario chung. Giả sử Alice muốn gửi cho Bob một tin nhắn ASCII đơn giản.

1. Alice gọi tác nhân người dùng của mình cho e-mail, cung cấp địa chỉ e-mail của Bob (ví dụ: `bob@someschool.edu`), soạn thư và hướng dẫn tác nhân người dùng gửi thư.
2. Tác nhân người dùng của Alice gửi thư đến máy chủ thư của cô ấy, nơi nó được đặt trong hàng đợi thư.
3. Phía máy khách của SMTP, chạy trên máy chủ thư của Alice, sẽ thấy thư trong hàng đợi thư. Nó mở một kết nối TCP đến một máy chủ SMTP, chạy trên máy chủ thư của Bob.
4. Sau một số bắt tay SMTP ban đầu, máy khách SMTP gửi tin nhắn của Alice vào kết nối TCP.
5. Tại máy chủ thư của Bob, phía máy chủ của SMTP nhận được tin nhắn. Máy chủ thư của Bob sau đó đặt thư vào hộp thư của Bob.
6. Bob gọi đại lý người dùng của mình để đọc tin nhắn một cách thuận tiện.

Kịch bản được tóm tắt trong Hình 2.15.

Điều quan trọng cần quan sát là SMTP thường không sử dụng các dịch vụ thư trung gian để gửi thư, ngay cả khi hai máy chủ thư được đặt ở hai đầu đối diện của thế giới. Nếu máy chủ của Alice ở Hồng Kông và máy chủ của Bob ở St. Louis, kết nối TCP là kết nối trực tiếp giữa máy chủ Hồng Kông và St. Louis. Trong



Hình 2.15 ♦ Alice gửi tin nhắn cho Bob

đặc biệt, nếu máy chủ thư của Bob ngừng hoạt động, thư vẫn còn trong máy chủ thư của Alice và chờ một lần thử mới — thư không được đặt trong một số máy chủ thư trung gian.

Bây giờ chúng ta hãy xem xét kỹ hơn cách SMTP chuyển thư từ máy chủ thư gửi đến máy chủ nhận thư. Chúng ta sẽ thấy rằng SMTP proto-col có nhiều điểm tương đồng với các giao thức được sử dụng để tương tác trực tiếp với con người. Đầu tiên, máy khách SMTP (chạy trên máy chủ gửi mail) có TCP thiết lập kết nối với cổng 25 tại máy chủ SMTP (chạy trên máy chủ máy chủ thư nhận). Nếu máy chủ không hoạt động, máy khách sẽ thử lại sau. Khi kết nối này được thiết lập, máy chủ và máy khách thực hiện một số bắt tay lớp ứng dụng — giống như con người thường tự giới thiệu trước khi chuyển thông tin từ người này sang người khác, máy khách và máy chủ SMTP tự giới thiệu trước khi truyền thông tin. Trong giai đoạn bắt tay SMTP này, ứng dụng khách SMTP cho biết địa chỉ e-mail của người gửi (người tạo thư) và địa chỉ e-mail của người nhận. Khi máy khách SMTP và máy chủ đã tự giới thiệu với nhau, máy khách sẽ gửi tin nhắn. SMTP có thể tin tưởng vào dịch vụ truyền dữ liệu đáng tin cậy của TCP để đưa tin nhắn đến máy chủ mà không gặp lỗi. Máy khách sau đó lặp lại quá trình này trên cùng một kết nối TCP nếu nó có các thông điệp khác để gửi đến máy chủ; nếu không, nó sẽ hướng dẫn TCP đóng kết nối.

Tiếp theo chúng ta hãy xem bản ghi ví dụ về các tin nhắn được trao đổi giữa máy khách SMTP (C) và máy chủ SMTP (S). Tên máy chủ của máy khách là `crepes.fr` và tên máy chủ của máy chủ là `hamburger.edu`. Các dòng văn bản ASCII được mở đầu bằng `C:` chính xác là các dòng mà máy khách gửi vào ổ cắm TCP của nó và các dòng văn bản ASCII được mở đầu bằng `S:` chính xác là các dòng mà máy chủ gửi vào ổ cắm TCP của nó. Bảng điểm sau đây bắt đầu ngay sau khi kết nối TCP được thiết lập.

```
S: 220 hamburger.edu
C: HELO crepes.fr
```



```

S: 250 Xin chào crepes.fr, hân hạnh được gặp
bạn C: MAIL TỪ: <alice@crepes.fr>
S: 250 alice@crepes.fr ... Người gửi
ok C: RCPT ĐẾN: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Người nhận ok
C: DATA
S: 354 Nhập thư, kết thúc bằng "." trên một dòng của
chính nó C: Bạn có thích sốt cà chua không?
C: Còn dưa chua thì sao?
C: .
S: 250 Tin nhắn được chấp nhận để gửi
C: THOÁT
S: 221 hamburger.edu kết nối đóng

```

Trong ví dụ trên, khách hàng gửi tin nhắn ("Bạn có thích sốt cà chua không? Còn dưa chua thì sao?") từ crepes.fr máy chủ thư đến hamburger.edu máy chủ thư. Là một phần của cuộc đối thoại, khách hàng đã đưa ra năm lệnh: HELO (viết tắt của HELLO), MAIL FROM, RCPT TO, DATA và QUIT. Những lệnh này là tự giải thích. Máy khách cũng gửi một dòng bao gồm một dấu chấm duy nhất, cho biết kết thúc của tin nhắn đến máy chủ. (Trong ASCII jar-gon, mỗi tin nhắn kết thúc bằng CRLF. CRLF, trong đó CR và LF lần lượt là viết tắt của trả về vận chuyển và cấp dữ liệu dòng.) Máy chủ đưa ra các câu trả lời cho mỗi lệnh, với mỗi câu trả lời có mã trả lời và một số quốc gia mở rộng bằng tiếng Anh (tùy chọn). Chúng tôi đề cập ở đây rằng SMTP sử dụng các kết nối liên tục: Nếu máy chủ gửi thư có một số thư để gửi đến cùng một máy chủ nhận thư, nó có thể gửi tất cả các thư qua cùng một kết nối TCP. Đối với mỗi thư, khách hàng bắt đầu quá trình với một MAIL FROM: crepes.fr mới, chỉ định kết thúc thư với một khoảng thời gian riêng biệt và chỉ phát hành QUIT sau khi tất cả các thư đã được gửi.

Chúng tôi khuyên bạn nên sử dụng Telnet để thực hiện một cuộc đối thoại trực tiếp với một máy chủ SMTP. Để làm điều này, vấn đề

máy chủ telnetTên 25

trong đó serverName là tên của máy chủ thư cục bộ. Khi bạn làm điều này, bạn chỉ cần thiết lập kết nối TCP giữa máy chủ cục bộ của bạn và máy chủ thư. Sau khi gõ dòng này, bạn sẽ ngay lập tức nhận được trả lời 220 từ máy chủ. Sau đó phát hành các lệnh SMTP HELO, MAIL FROM, RCPT TO, DATA, CRLF.CRLF và QUIT vào những thời điểm thích hợp. Bạn cũng nên làm Programming Assignment 3 ở cuối chương này. Trong phân công đó, bạn sẽ xây dựng một tác nhân người dùng đơn giản triển khai phía máy khách của SMTP. Nó sẽ cho phép bạn gửi một tin nhắn e-mail đến một người nhận tùy ý thông qua một máy chủ thư cục bộ.

2.3.2 Định dạng thư

Khi Alice viết một lá thư ốc sên thông thường cho Bob, cô ấy có thể bao gồm tất cả các loại thông tin tiêu đề ngoại vi ở đầu thư, chẳng hạn như địa chỉ của Bob, địa chỉ trả lại của chính cô ấy và ngày. Tương tự, khi một thông điệp email được gửi từ người này sang người khác, một tiêu đề chứa thông tin ngoại vi đứng trước nội dung của chính thư. Thông tin ngoại vi này được chứa trong một loạt các dòng tiêu đề, được xác định trong RFC 5322. Các dòng tiêu đề và nội dung của thư được phân tách bằng một dòng trống (nghĩa là bằng CRLF). RFC 5322 chỉ định định dạng chính xác cho các dòng tiêu đề thư cũng như các diễn giải ngữ nghĩa của chúng. Như với HTTP, mỗi dòng tiêu đề chứa văn bản có thể đọc được, bao gồm một từ khóa theo sau là dấu hai chấm theo sau là một giá trị. Một số từ khóa là bắt buộc và những từ khóa khác là tùy chọn. Mỗi tiêu đề phải có dòng tiêu đề Từ: và dòng tiêu đề Đến:; tiêu đề có thể bao gồm dòng tiêu đề Chủ đề: cũng như các dòng tiêu đề tùy chọn khác. Điều quan trọng cần lưu ý là các dòng tiêu đề này *khác với* các lệnh SMTP mà chúng tôi đã nghiên cứu trong Phần 2.3.1 (mặc dù chúng chứa một số từ phổ biến như "từ" và "đến"). Các lệnh trong phần đó là một phần của giao thức bắt tay SMTP; Các dòng tiêu đề được kiểm tra trong phần này là một phần của chính thư.

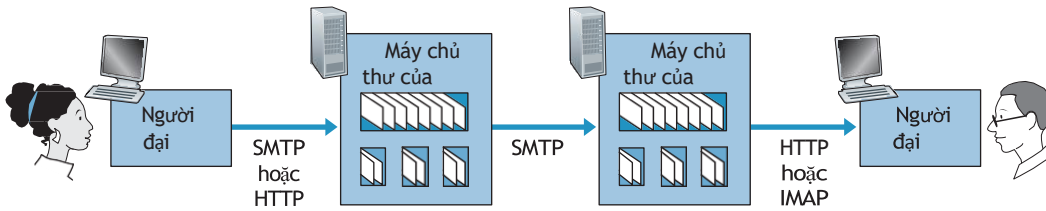
Một tiêu đề thư điển hình trông như thế này:

```
Từ: alice@crepes.fr
Đến: bob@hamburger.edu
Chủ đề: Tìm kiếm ý nghĩa của cuộc sống.
```

Sau tiêu đề thư, một dòng trống theo sau; sau đó nội dung thư (trong ASCII) theo sau. Bạn nên sử dụng Telnet để gửi thư đến máy chủ thư có chứa một số dòng tiêu đề, bao gồm dòng tiêu đề Chủ đề:. Để làm điều này, phát hành `telnet serverName 25`, như đã thảo luận trong Phần 2.3.1.

2.3.3 Giao thức truy cập thư điện tử

Khi SMTP gửi thư từ máy chủ thư của Alice đến máy chủ thư của Bob, thư sẽ được đặt trong hộp thư của Bob. Cho rằng Bob (người nhận) thực thi tác nhân người dùng của mình trên máy chủ cục bộ của mình (ví dụ: điện thoại thông minh hoặc PC), việc xem xét đặt máy chủ thư trên máy chủ cục bộ của anh ấy là điều tự nhiên. Với cách tiếp cận này, máy chủ thư của Alice sẽ dialog trực tiếp với PC của Bob. Tuy nhiên, có một vấn đề với cách tiếp cận này. Nhớ lại rằng máy chủ thư quản lý hộp thư và chạy phía máy khách và máy chủ của SMTP. Nếu máy chủ thư của Bob nằm trên máy chủ cục bộ của anh ấy, thì máy chủ của Bob sẽ phải luôn bật và kết nối với Internet, để nhận thư mới, có thể đến bất cứ lúc nào. Điều này là không thực tế đối với nhiều người dùng Internet. Thay vào đó, một người dùng thông thường chạy một tác nhân người dùng trên máy chủ cục bộ nhưng truy cập hộp thư của nó được lưu trữ trên máy chủ thư luôn được chia sẻ. Máy chủ thư này được chia sẻ với những người dùng khác.



Hình 2.16 ♦ Giao thức email và các thực thể giao tiếp của chúng

Bây giờ chúng ta hãy xem xét đường dẫn mà một tin nhắn e-mail đi khi nó được gửi từ Alice đến Bob. Chúng tôi vừa học được rằng tại một số điểm dọc theo đường dẫn, thông điệp e-mail cần được gửi vào máy chủ thư của Bob. Điều này có thể được thực hiện đơn giản bằng cách yêu cầu tác nhân người dùng của Alice gửi tin nhắn trực tiếp đến máy chủ thư của Bob. Tuy nhiên, thông thường tác nhân người dùng của người gửi không đối thoại trực tiếp với máy chủ thư của người nhận. Thay vào đó, như thể hiện trong Hình 2.16, tác nhân người dùng của Alice sử dụng SMTP hoặc HTTP để gửi thông điệp e-mail vào máy chủ thư của cô ấy, sau đó máy chủ thư của Alice sử dụng SMTP (như một SMTP cli-ent) để chuyển tiếp thông điệp e-mail đến máy chủ thư của Bob. Tại sao thủ tục hai bước? Chủ yếu là vì không chuyển tiếp qua máy chủ thư của Alice, tác nhân người dùng của Alice không có bất kỳ truy đòi nào đến máy chủ thư đích không thể truy cập được. Bằng cách yêu cầu Alice gửi e-mail trước tiên vào máy chủ thư của riêng mình, máy chủ thư của Alice có thể liên tục cố gắng gửi thư đến máy chủ thư của Bob, cứ sau 30 phút, cho đến khi máy chủ thư của Bob hoạt động. (Và nếu máy chủ thư của Alice bị hỏng, thì cô ấy có quyền khiếu nại với quản trị viên hệ thống của mình!)

Nhưng vẫn còn một mảnh ghép còn thiếu cho câu đố! Làm thế nào để một người nhận như Bob, chạy một tác nhân người dùng trên máy chủ cục bộ của anh ta, có được thư của anh ta, đang ngồi trong một máy chủ thư? Lưu ý rằng tác nhân người dùng của Bob không thể sử dụng SMTP để lấy tin nhắn vì lấy tin nhắn là một thao tác kéo, trong khi SMTP là một giao thức đẩy.

Ngày nay, có hai cách phổ biến để Bob lấy e-mail của mình từ máy chủ thư. Nếu Bob đang sử dụng e-mail dựa trên web hoặc ứng dụng điện thoại thông minh (như Gmail), thì tác nhân người dùng sẽ sử dụng HTTP để truy xuất e-mail của Bob. Trường hợp này yêu cầu máy chủ thư của Bob phải có giao diện HTTP cũng như giao diện SMTP (để giao tiếp với máy chủ thư của Alice). Phương pháp thay thế, thường được sử dụng với các ứng dụng thư khách như Microsoft Outlook, là sử dụng **Giao thức Truy cập Thư Internet (IMAP)** được xác định trong RFC 3501. Cả hai phương pháp HTTP và IMAP đều cho phép Bob quản lý các thư mục, được duy trì trong máy chủ thư của Bob. Bob có thể di chuyển tin nhắn vào các thư mục anh ấy tạo, xóa thư, đánh dấu thư là quan trọng, v.v.

2.4 DNS—Dịch vụ Thư mục của Internet

Con người chúng ta có thể được xác định theo nhiều cách. Ví dụ, chúng ta có thể được xác định bằng những cái tên xuất hiện trên giấy khai sinh của chúng ta. Chúng tôi có thể được xác định bằng số an sinh xã hội của chúng tôi. Chúng tôi có thể được

xác định bằng số giấy phép lái xe của chúng tôi.

Mặc dù mỗi định danh có thể được sử dụng để xác định mọi người, nhưng trong một ngữ cảnh nhất định, một định danh có thể phù hợp hơn một định danh khác. Ví dụ, các máy tính tại IRS (cơ quan thu thuế khét tiếng ở Hoa Kỳ) thích sử dụng số an sinh xã hội có độ dài cố định hơn là tên giấy khai sinh. Mặt khác, những người bình thường thích tên giấy khai sinh ghi nhớ hơn là số an sinh xã hội. (Thật vậy, bạn có thể tưởng tượng nói, "Xin chào. Tên tôi là 132-67-9875. Xin hãy gặp chồng tôi, 178-87-1146.")

Giống như con người có thể được xác định theo nhiều cách, các máy chủ Internet cũng vậy. Một định danh cho một máy chủ lưu trữ là **tên máy chủ của nó**. Tên máy chủ — chẳng hạn như `www.facebook.com`, `www.google.com`, `gaia.cs.umass.edu` — là ghi nhớ và do đó được con người đánh giá cao. Tuy nhiên, tên máy chủ cung cấp rất ít, nếu có, thông tin về vị trí trong Internet của máy chủ. (Tên máy chủ như `www.eurecom.fr`, kết thúc bằng mã quốc gia `.fr`, cho chúng ta biết rằng máy chủ có thể ở Pháp, nhưng không nói nhiều hơn.) Hơn nữa, vì tên máy chủ có thể bao gồm các ký tự chữ và số có độ dài thay đổi, chúng sẽ khó xử lý bởi các routers. Vì những lý do này, máy chủ cũng được xác định bởi cái gọi là **địa chỉ IP**.

Chúng tôi thảo luận về địa chỉ IP một số chi tiết trong Chương 4, nhưng sẽ rất hữu ích khi nói một vài từ ngắn gọn về chúng ngay bây giờ. Một địa chỉ IP bao gồm bốn byte và có cấu trúc phân cấp cứng nhắc. Địa chỉ IP trông giống như `121.7.106.83`, trong đó mỗi dấu chấm phân tách một trong các byte được biểu thị bằng ký hiệu thập phân từ 0 đến 255. Địa chỉ IP có thứ bậc vì khi chúng tôi quét địa chỉ từ trái sang phải, chúng tôi thu được ngày càng nhiều thông tin cụ thể hơn về vị trí của máy chủ lưu trữ trên Internet (nghĩa là trong mạng nào, trong mạng lưới). Tương tự, khi chúng tôi quét địa chỉ bưu chính từ dưới lên trên, chúng tôi ngày càng thu được thông tin cụ thể hơn về vị trí của người nhận.

2.4.1 Dịch vụ do DNS cung cấp

Chúng ta vừa thấy rằng có hai cách để xác định máy chủ lưu trữ — bằng tên máy chủ và địa chỉ IP. Mọi người thích định danh tên máy chủ ghi nhớ nhiều hơn, trong khi các bộ định tuyến thích các địa chỉ IP có độ dài cố định, có cấu trúc phân cấp. Để khắc phục các tùy chọn này, chúng ta cần một dịch vụ thư mục dịch tên máy chủ thành địa chỉ IP. Đây là nhiệm vụ chính của **hệ thống tên miền (DNS) của Internet**. DNS là (1) cơ sở dữ liệu phân tán được triển khai trong hệ thống phân cấp máy chủ **DNS** và (2) giao thức lớp ứng dụng cho phép máy chủ truy vấn cơ sở dữ liệu phân tán. Các máy chủ DNS thường là các máy UNIX chạy phần mềm Berkeley Internet Name Domain (BIND) [BIND 2020]. Giao thức DNS chạy qua UDP và sử dụng cổng 53.

DNS thường được sử dụng bởi các giao thức lớp ứng dụng khác, bao gồm HTTP và SMTP, để dịch tên máy chủ do người dùng cung cấp sang địa chỉ IP. Để kiểm tra, hãy xem xét điều gì sẽ xảy ra khi một trình duyệt (nghĩa là máy khách HTTP), chạy trên máy chủ của một số người dùng, yêu cầu URL `www.someschool.edu/index.html`. Để máy chủ của người dùng có thể gửi thông báo yêu cầu HTTP lên Web

`www.someschool.edu` máy chủ, máy chủ của người dùng trước tiên phải lấy địa chỉ IP của `www.someschool.edu`. Điều này được thực hiện như sau.

1. Cùng một máy người dùng chạy phía máy khách của ứng dụng DNS.
2. Trình duyệt trích xuất tên máy chủ, `www.someschool.edu`, từ URL và chuyển tên máy chủ đến phía máy khách của ứng dụng DNS.
3. Máy khách DNS gửi truy vấn có chứa tên máy chủ đến máy chủ DNS.
4. Máy khách DNS cuối cùng sẽ nhận được trả lời, bao gồm địa chỉ IP cho tên máy chủ.
5. Khi trình duyệt nhận được địa chỉ IP từ DNS, nó có thể bắt đầu kết nối TCP với quy trình máy chủ HTTP nằm ở cổng 80 tại địa chỉ IP đó.

Chúng tôi thấy từ ví dụ này rằng DNS thêm một độ trễ bổ sung — đôi khi đáng kể — cho các ứng dụng Internet sử dụng nó. May mắn thay, như chúng ta thảo luận bên dưới, địa chỉ IP mong muốn thường được lưu vào bộ nhớ cache trong máy chủ DNS "gần đó", giúp giảm lưu lượng mạng DNS cũng như độ trễ DNS trung bình.

DNS cung cấp một vài dịch vụ quan trọng khác ngoài việc dịch tên máy chủ sang địa chỉ IP:

- **Răng cưa máy chủ.** Máy chủ lưu trữ có tên máy chủ phức tạp có thể có một hoặc nhiều tên bí danh. Ví dụ: tên máy chủ như `relay1.west-coast.enterprise.com` có thể có hai bí danh như `enterprise.com` và `www.enterprise.com`. Trong trường hợp này, tên máy chủ `relay1.west-coast.enterprise.com` được cho là một **tên máy chủ chính tắc**. Tên máy chủ bí danh, khi có, thường ghi nhớ nhiều hơn tên máy chủ chính tắc. DNS có thể được gọi bởi một ứng dụng để lấy tên máy chủ chuẩn cho tên máy chủ bí danh được cung cấp cũng như địa chỉ IP của máy chủ.
- **Răng cưa máy chủ thư.** Vì những lý do rõ ràng, rất mong muốn rằng địa chỉ e-mail là ghi nhớ. Ví dụ: nếu Bob có tài khoản với Yahoo Mail, địa chỉ e-mail của Bob có thể đơn giản như `bob@yahoo.com`. Tuy nhiên, tên máy chủ của máy chủ thư Yahoo phức tạp hơn và ít ghi nhớ hơn nhiều so với `yahoo.com` đơn giản (ví dụ: tên máy chủ chính tắc có thể là một số thứ giống như `relay1.west-coast.yahoo.com`). DNS có thể được gọi bởi một ứng dụng thư để lấy tên máy chủ chuẩn cho tên máy chủ bí danh được cung cấp cũng như địa chỉ IP của máy chủ. Trên thực tế, bản ghi MX (xem bên dưới) cho phép máy chủ thư và máy chủ Web của công ty có tên máy chủ (bí danh) giống hệt nhau; ví dụ: máy chủ Web và máy chủ thư của công ty đều có thể được gọi là `enter-prise.com`.
- **Phân phối tải.** DNS cũng được sử dụng để thực hiện phân phối tải giữa các máy chủ được trả lời, chẳng hạn như các máy chủ Web được sao chép. Các trang web bận rộn, chẳng hạn như `cnn.com`, được sao chép trên nhiều máy chủ, với mỗi máy chủ chạy trên một hệ thống đầu cuối khác nhau và mỗi máy chủ có một địa chỉ IP khác nhau. Đối với các máy chủ Web được sao chép, một *tập hợp IP*



NGUYÊN TẮC TRONG

DNS: CÁC CHỨC NĂNG MẠNG QUAN TRỌNG THÔNG QUA MÔ HÌNH MÁY KHÁCH-MÁY CHỦ

Giống như HTTP, FTP và SMTP, giao thức DNS là một giao thức lớp ứng dụng vì nó (1) chạy giữa các hệ thống đầu cuối giao tiếp bằng cách sử dụng mô hình máy khách-máy chủ và (2) dựa trên giao thức truyền tải end-to-end cơ bản để truyền thông điệp DNS giữa các hệ thống đầu cuối giao tiếp. Tuy nhiên, theo một nghĩa khác, vai trò của DNS khá khác biệt so với các ứng dụng Web, truyền tệp và e-mail. Không giống như các ứng dụng này, DNS không phải là một ứng dụng mà người dùng tương tác trực tiếp. Thay vào đó, DNS cung cấp một chức năng Internet cốt lõi, cụ thể là dịch tên máy chủ sang địa chỉ IP cơ bản của chúng, cho các ứng dụng người dùng và phần mềm khác trên Internet. Chúng tôi đã lưu ý trong Phần 1.2 rằng phần lớn sự phức tạp trong kiến trúc Internet nằm ở "các cạnh" của mạng. DNS, thực hiện quy trình dịch từ tên đến địa chỉ quan trọng bằng cách sử dụng máy khách và máy chủ nằm ở rìa mạng, là một ví dụ khác về triết lý thiết kế đó.

Do đó, địa chỉ được liên kết với một tên máy chủ bí danh. Cơ sở dữ liệu DNS chứa tập hợp địa chỉ IP này. Khi máy khách thực hiện truy vấn DNS cho tên được ánh xạ tới một tập hợp địa chỉ, máy chủ sẽ phản hồi với toàn bộ bộ địa chỉ IP, nhưng xoay thứ tự các địa chỉ trong mỗi câu trả lời. Bởi vì một máy khách thường gửi thông báo yêu cầu HTTP của nó đến địa chỉ IP được liệt kê đầu tiên trong bộ, DNS rotation phân phối lưu lượng giữa các máy chủ được sao chép. Xoay vòng DNS cũng được sử dụng cho e-mail để nhiều máy chủ thư có thể có cùng tên bí danh. Ngoài ra, các công ty phân phối lều như Akamai đã sử dụng DNS theo những cách tinh vi hơn [Dilley 2002] để cung cấp phân phối nội dung Web (xem Phần 2.6.3).

DNS được chỉ định trong RFC 1034 và RFC 1035, và được cập nhật trong một số RFC bổ sung. Đây là một hệ thống phức tạp và chúng tôi chỉ đề cập đến các khía cạnh chính của hoạt động của nó ở đây. Độc giả quan tâm được giới thiệu đến các RFC này và cuốn sách của Albitz và Liu [Albitz 1993]; xem thêm bài báo hồi cứu [Mockapetris 1988], cung cấp một mô tả hay về những gì và tại sao của DNS, và [Mockapetris 2005].

2.4.2 Tổng quan về cách DNS hoạt động

Bây giờ chúng tôi trình bày tổng quan cấp cao về cách DNS hoạt động. Cuộc thảo luận của chúng tôi sẽ tập trung vào dịch vụ dịch từ tên máy chủ sang địa chỉ IP.

Giả sử rằng một số ứng dụng (chẳng hạn như trình duyệt Web hoặc ứng dụng thư) chạy trong máy chủ của người dùng cần dịch tên máy chủ sang địa chỉ IP. Ứng dụng sẽ gọi phía máy khách của DNS, chỉ định tên máy chủ cần dịch. (Trên nhiều máy dựa trên UNIX, `gethostbyname()` là lệnh gọi hàm mà một ứng dụng gọi để thực hiện bản dịch.) DNS trong máy chủ lưu trữ của người dùng sau đó

tiếp quản, gửi một tin nhắn truy vấn vào mạng. Tất cả các truy vấn DNS và trả lời message được gửi trong các biểu đồ dữ liệu UDP đến cổng 53. Sau một sự chậm trễ, từ mili giây đến vài giây, DNS trong máy chủ của người dùng sẽ nhận được thông báo trả lời DNS cung cấp ánh xạ mong muốn. Ánh xạ này sau đó được chuyển đến ứng dụng gọi. Do đó, từ góc độ của ứng dụng gọi trong máy chủ của người dùng, DNS là một hộp đen cung cấp dịch vụ dịch thuật đơn giản, dễ hiểu. Nhưng trên thực tế, hộp đen triển khai dịch vụ rất phức tạp, bao gồm một số lượng lớn các máy chủ DNS được phân phối trên toàn cầu, cũng như giao thức lớp ứng dụng chỉ định cách các máy chủ DNS và máy chủ truy vấn giao tiếp.

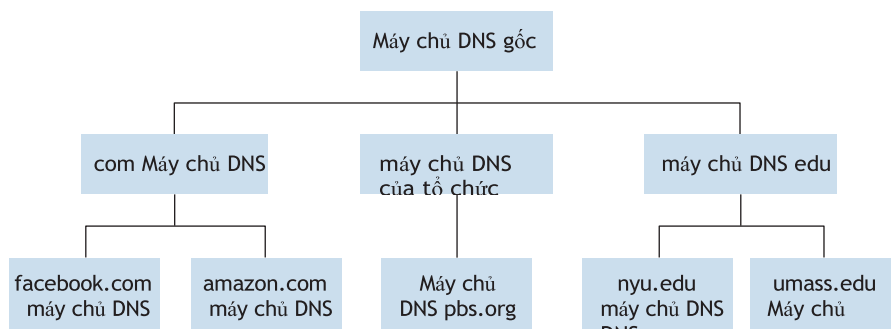
Một thiết kế đơn giản cho DNS sẽ có một máy chủ DNS chứa tất cả các bản đồ. Trong thiết kế tập trung này, máy khách chỉ cần hướng tất cả các truy vấn đến máy chủ DNS duy nhất và máy chủ DNS phản hồi trực tiếp với các máy khách truy vấn. Mặc dù sự đơn giản của thiết kế này rất hấp dẫn, nhưng nó không phù hợp với Internet ngày nay, với số lượng máy chủ khổng lồ (và ngày càng tăng). Các vấn đề với một thiết kế tập trung bao gồm:

- **Một điểm thất bại duy nhất.** Nếu máy chủ DNS gặp sự cố, toàn bộ Internet cũng vậy!
- **Lưu lượng giao thông.** Một máy chủ DNS duy nhất sẽ phải xử lý tất cả các truy vấn DNS (cho tất cả các yêu cầu HTTP và thông điệp e-mail được tạo từ hàng trăm triệu máy chủ).
- **Cơ sở dữ liệu tập trung từ xa.** Một máy chủ DNS duy nhất không thể "gần" tất cả các máy khách truy vấn. Nếu chúng ta đặt máy chủ DNS duy nhất ở thành phố New York, thì tất cả các truy vấn từ Úc phải đi đến phía bên kia địa cầu, có lẽ qua các liên kết chậm và tắc nghẽn. Điều này có thể dẫn đến sự chậm trễ đáng kể.
- **Bảo trì.** Máy chủ DNS duy nhất sẽ phải lưu giữ hồ sơ cho tất cả các máy chủ Internet. Cơ sở dữ liệu tập trung này không chỉ rất lớn mà còn phải được cập nhật thường xuyên để tính đến mọi máy chủ mới.

Tóm lại, một cơ sở dữ liệu tập trung trong một máy chủ DNS duy nhất đơn giản là *không mở rộng quy mô*. Do đó, DNS được phân phối theo thiết kế. Trên thực tế, DNS là một ví dụ tuyệt vời về cách cơ sở dữ liệu phân tán có thể được triển khai trên Internet.

Cơ sở dữ liệu phân tán, phân cấp

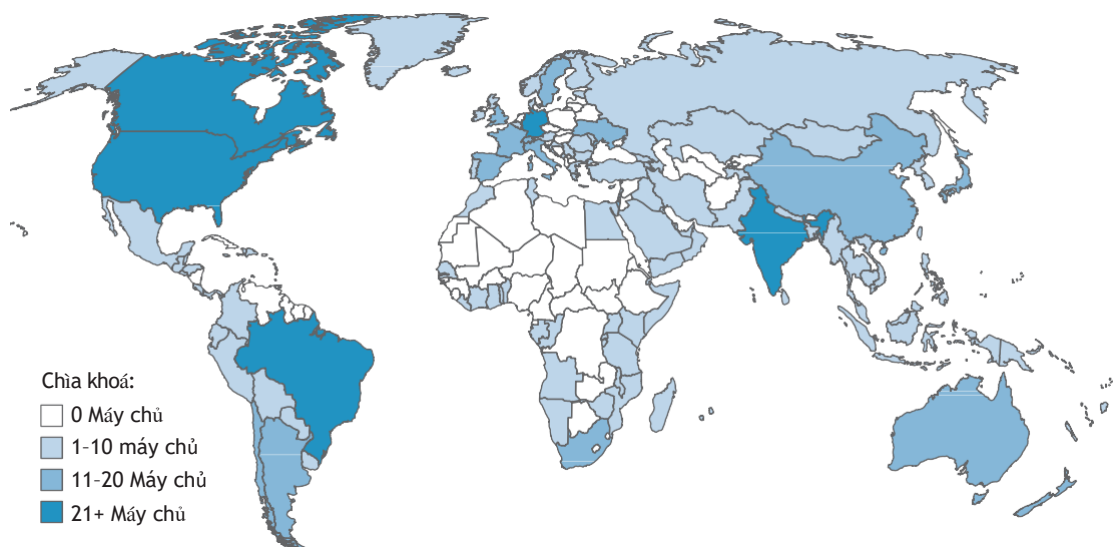
Để giải quyết vấn đề quy mô, DNS sử dụng một số lượng lớn máy chủ, được tổ chức theo kiểu phân cấp và phân phối trên toàn thế giới. Không có máy chủ DNS duy nhất nào có tất cả ánh xạ cho tất cả các máy chủ trên Internet. Thay vào đó, các map-ping được phân phối trên các máy chủ DNS. Để xấp xỉ đầu tiên, có ba lớp máy chủ DNS — máy chủ DNS gốc, máy chủ DNS tên miền cấp cao nhất (TLD) và máy chủ DNS có thẩm quyền — được tổ chức theo thứ bậc như thể hiện trong Hình 2.17. Để hiểu cách ba lớp máy chủ này tương tác, giả sử một máy khách DNS muốn xác định địa chỉ IP cho www.amazon.com tên máy chủ. Để xấp xỉ đầu tiên, các sự kiện sau sẽ diễn ra. Trước tiên, khách hàng liên hệ với một trong các



Hình 2.17 ♦ Một phần của hệ thống phân cấp máy chủ DNS

các máy chủ gốc, trả về địa chỉ IP cho các máy chủ TLD cho com miền cấp cao nhất. Sau đó, máy khách liên hệ với một trong các máy chủ TLD này, trả về địa chỉ IP của máy chủ có thẩm quyền cho `amazon.com`. Cuối cùng, máy khách liên hệ với một trong những máy chủ có thẩm quyền để `amazon.com`, trả về địa chỉ IP cho tên máy chủ `www.amazon.com`. Chúng tôi sẽ sớm kiểm tra quá trình tra cứu DNS này chi tiết hơn. Nhưng trước tiên chúng ta hãy xem xét kỹ hơn ba lớp máy chủ DNS này:

- **Máy chủ DNS gốc.** Có hơn 1000 phiên bản máy chủ gốc nằm rải rác trên toàn thế giới, như thể hiện trong Hình 2.18. Các máy chủ gốc này là bản sao của 13 máy chủ gốc different, được quản lý bởi 12 tổ chức khác nhau và được điều phối thông qua Cơ quan cấp số Internet [IANA 2020]. Danh sách đầy đủ các máy chủ tên gốc, cùng với các tổ chức quản lý chúng và địa chỉ IP của chúng có thể được tìm thấy tại [Máy chủ gốc 2020]. Máy chủ tên gốc cung cấp địa chỉ IP của máy chủ TLD.
- **Máy chủ tên miền cấp cao nhất (TLD).** Đối với mỗi tên miền cấp cao nhất — các tên miền cấp cao nhất như `com`, `org`, `net`, `edu` và `gov` và tất cả các tên miền cấp cao nhất của quốc gia như `uk`, `fr`, `ca` và `jp` — có máy chủ TLD (hoặc cụm máy chủ). Công ty Verisign Global Registry Services duy trì các máy chủ TLD cho tên miền cấp cao nhất và công ty Educause duy trì các máy chủ TLD cho tên miền cấp cao nhất `edu`. Cơ sở hạ tầng mạng hỗ trợ TLD có thể lớn và phức tạp; xem [Osterweil 2012] để biết tổng quan tốt đẹp về công việc mạng Verisign. Xem [Danh sách TLD 2020] để biết danh sách tất cả các miền cấp cao nhất. Máy chủ TLD cung cấp địa chỉ IP cho các máy chủ DNS có thẩm quyền.
- **Máy chủ DNS có thẩm quyền.** Mọi tổ chức có máy chủ có thể truy cập công khai (chẳng hạn như máy chủ Web và máy chủ thư) trên Internet phải cung cấp các bản ghi DNS có thể truy cập công khai ánh xạ tên của các máy chủ đó đến địa chỉ IP. Máy chủ DNS có thẩm quyền của tổ chức chứa các bản ghi DNS này. Một tổ chức có thể chọn triển khai máy chủ DNS có thẩm quyền của riêng mình để giữ các bản ghi này; Ngoài ra, tổ chức có thể trả tiền để các hồ sơ này được lưu trữ trong một

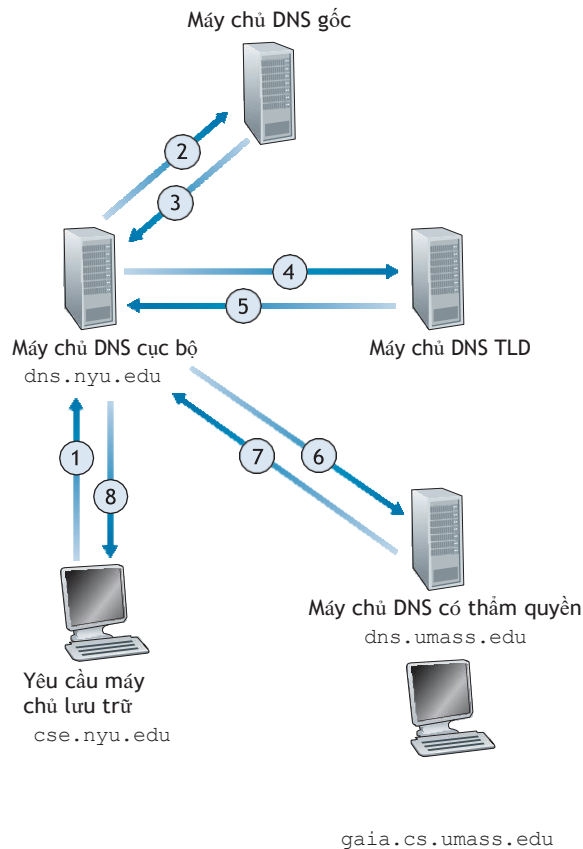


Biểu đồ 2.18 ♦ Máy chủ gốc DNS năm 2020

máy chủ DNS có thẩm quyền của một số nhà cung cấp dịch vụ. Hầu hết các trường đại học và các công ty lớn triển khai và duy trì máy chủ DNS có thẩm quyền chính và phụ (sao lưu) của riêng họ.

Các máy chủ gốc, TLD và DNS có thẩm quyền đều thuộc về hệ thống phân cấp của các máy chủ DNS, như thể hiện trong Hình 2.17. Có một loại máy chủ DNS quan trọng khác được gọi là **máy chủ DNS cục bộ**. Một máy chủ DNS cục bộ không hoàn toàn thuộc về hệ thống phân cấp của các máy chủ nhưng vẫn là trung tâm của kiến trúc DNS. Mỗi ISP—chẳng hạn như ISP dân cư hoặc ISP tổ chức—có một máy chủ DNS cục bộ (còn được gọi là máy chủ định danh mặc định). Khi một máy chủ kết nối với ISP, ISP cung cấp cho máy chủ địa chỉ IP của một hoặc nhiều máy chủ DNS cục bộ của nó (thường thông qua DHCP, được thảo luận trong Chương 4). Bạn có thể dễ dàng xác định địa chỉ IP của máy chủ DNS cục bộ của mình bằng cách truy cập các cửa sổ trạng thái mạng trong Windows hoặc UNIX. Máy chủ DNS cục bộ của máy chủ lưu trữ thường "gắn" máy chủ. Đối với ISP tổ chức, máy chủ DNS cục bộ có thể nằm trên cùng một mạng LAN với máy chủ; đối với một ISP dân cư, nó thường được tách ra khỏi máy chủ lưu trữ bởi không quá một vài routers. Khi một máy chủ thực hiện truy vấn DNS, truy vấn được gửi đến máy chủ DNS cục bộ, hoạt động như một proxy, chuyển tiếp truy vấn vào hệ thống phân cấp máy chủ DNS, như chúng ta sẽ thảo luận chi tiết hơn bên dưới.

Chúng ta hãy xem một ví dụ đơn giản. Giả sử `cse.nyu.edu` máy chủ mong muốn địa chỉ IP của `gaia.cs.umass.edu`. Cũng giả sử rằng máy chủ DNS cục bộ của NYU cho `cse.nyu.edu` được gọi là `dns.nyu.edu` và một máy chủ DNS có thẩm quyền cho `gaia.cs.umass.edu` được gọi là `dns.umass.edu`. Như thể hiện trong



Hình 2.19 ♦ Tương tác của các máy chủ DNS khác nhau

Hình 2.19, đầu tiên `cse.nyu.edu` máy chủ gửi một thông báo truy vấn DNS đến máy chủ DNS cục bộ của nó, `dns.nyu.edu`. Thông báo truy vấn chứa tên máy chủ sẽ được dịch, cụ thể là `gaia.cs.umass.edu`. Máy chủ DNS cục bộ chuyển tiếp thông báo truy vấn đến máy chủ DNS gốc. Máy chủ DNS gốc lưu ý hậu tố `edu` và trả về máy chủ DNS cục bộ một danh sách các địa chỉ IP cho các máy chủ TLD chịu trách nhiệm về `edu`. Máy chủ DNS cục bộ sau đó gửi lại thông báo truy vấn đến một trong các máy chủ TLD này. Máy chủ TLD lưu ý hậu tố `umass.edu` và phản hồi bằng địa chỉ IP của máy chủ DNS có thẩm quyền cho Đại học Massachusetts, cụ thể là `dns.umass.edu`. Cuối cùng, máy chủ DNS cục bộ gửi lại truy vấn message trực tiếp đến `dns.umass.edu`, phản hồi với địa chỉ IP của `gaia`

`.cs.umass.edu`. Lưu ý rằng trong ví dụ này, để có được ánh xạ cho một tên máy chủ, tám tin nhắn DNS đã được gửi: bốn tin nhắn truy vấn và bốn mes trả lời-

hiền nhân! Chúng ta sẽ sớm thấy bộ nhớ đệm DNS làm giảm lưu lượng truy vấn này như thế nào.

Ví dụ trước của chúng tôi giả định rằng máy chủ TLD biết máy chủ DNS có thẩm quyền cho tên máy chủ. Nói chung, điều này không phải lúc nào cũng đúng. Thay vào đó, máy chủ TLD

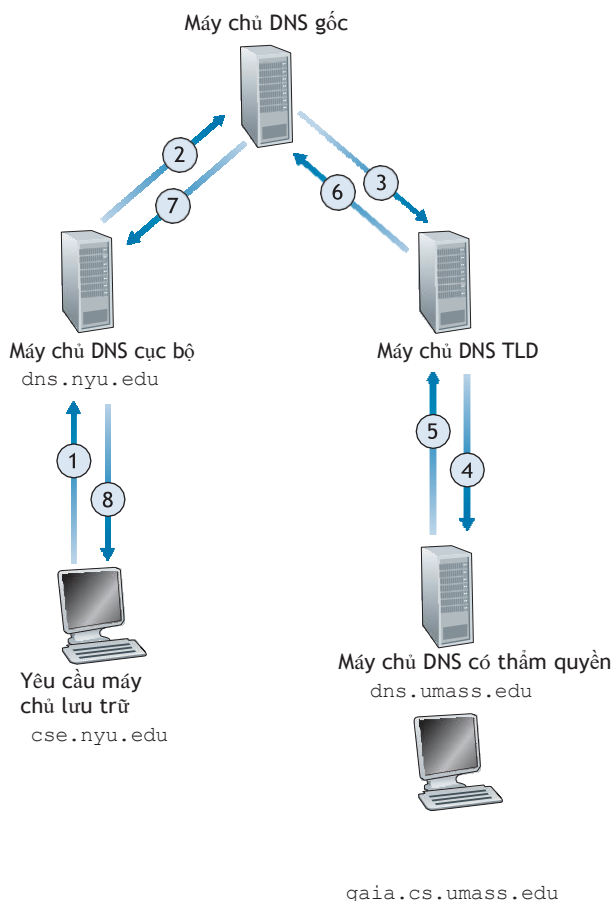
có thể chỉ biết về một máy chủ DNS trung gian, từ đó biết máy chủ DNS có thẩm quyền cho tên máy chủ. Ví dụ: giả sử một lần nữa rằng Đại học Massachusetts có một máy chủ DNS cho trường đại học, được gọi là `dns.umass.edu`. Cũng giả sử rằng mỗi khoa tại Đại học Massachusetts có máy chủ DNS riêng và mỗi máy chủ DNS của bộ phận có thẩm quyền cho tất cả các máy chủ trong khoa. Trong trường hợp này, khi máy chủ DNS trung gian, `dns.umass.edu`, nhận được truy vấn cho máy chủ có tên máy chủ kết thúc bằng `cs.umass.edu`, nó sẽ trở về `dns.nyu.edu` địa chỉ IP của `dns.cs.umass.edu`, có thẩm quyền cho tất cả các tên máy chủ kết thúc bằng `cs.umass.edu`. Máy chủ DNS cục bộ `dns.nyu.edu` sau đó gửi truy vấn đến máy chủ DNS có thẩm quyền, trả về ánh xạ mong muốn đến máy chủ DNS cục bộ, từ đó trả về ánh xạ cho yêu cầu chủ nhà. Trong trường hợp này, tổng cộng 10 tin nhắn DNS được gửi!

Ví dụ thể hiện trong Hình 2.19 sử dụng cả **truy vấn đệ quy** và **truy vấn lặp**. Truy vấn được gửi từ `cse.nyu.edu` đến `dns.nyu.edu` là một truy vấn đệ quy, vì truy vấn yêu cầu `dns.nyu.edu` lấy ánh xạ thay mặt cho nó. Tuy nhiên, ba truy vấn tiếp theo là lặp đi lặp lại vì tất cả các câu trả lời được trả về `dns.nyu.edu` trực tiếp. Về lý thuyết, bất kỳ truy vấn DNS nào cũng có thể lặp đi lặp lại hoặc đệ quy. Ví dụ, Hình 2.20 cho thấy một chuỗi truy vấn DNS mà tất cả các truy vấn là đệ quy. Trong thực tế, các truy vấn thường tuân theo mô hình trong Hình 2.19: Truy vấn từ máy chủ yêu cầu đến máy chủ DNS cục bộ là đệ quy và các truy vấn còn lại là lặp lại.

Bộ nhớ đệm DNS

Cuộc thảo luận của chúng tôi cho đến nay đã bỏ qua **bộ nhớ đệm DNS**, một tính năng cực kỳ quan trọng của hệ thống DNS. Trên thực tế, DNS khai thác rộng rãi bộ nhớ đệm DNS để cải thiện hiệu suất trễ và giảm số lượng tin nhắn DNS lan truyền trên Internet. Ý tưởng đằng sau bộ nhớ đệm DNS rất đơn giản. Trong chuỗi truy vấn, khi máy chủ DNS nhận được trả lời DNS (ví dụ: ánh xạ từ tên máy chủ đến địa chỉ IP), nó có thể lưu ánh xạ vào bộ nhớ cache trong bộ nhớ cục bộ của nó. Ví dụ, trong Hình 2.19, mỗi khi máy chủ DNS cục bộ `dns.nyu.edu` nhận được trả lời từ một số máy chủ DNS, nó có thể lưu trữ bất kỳ thông tin nào có trong thư trả lời. Nếu một cặp tên máy chủ / địa chỉ IP được lưu trữ trong máy chủ DNS và một truy vấn khác đến máy chủ DNS cho cùng một tên máy chủ, máy chủ DNS có thể cung cấp địa chỉ IP mong muốn, ngay cả khi nó không có thẩm quyền cho tên máy chủ. Bởi vì máy chủ và ánh xạ giữa tên máy chủ và địa chỉ IP không có nghĩa là vĩnh viễn, máy chủ DNS loại bỏ thông tin được lưu trong bộ nhớ cache sau một khoảng thời gian (thường được đặt thành hai ngày).

Ví dụ: giả sử rằng một máy chủ lưu trữ `apricot.nyu.edu` truy vấn `dns.nyu.edu` địa chỉ IP cho tên máy chủ `cnn.com`. Hơn nữa, giả sử rằng một vài giờ sau, một máy chủ NYU khác, giả sử, `kiwi.nyu.edu`, cũng truy vấn `dns.nyu.edu` có cùng tên máy chủ. Do bộ nhớ đệm, máy chủ DNS cục bộ sẽ có thể trả lại ngay địa chỉ IP của `cnn.com` cho máy chủ yêu cầu thứ hai này mà không phải truy vấn bất kỳ máy chủ DNS nào khác. Một máy chủ DNS cục bộ có thể



Hình 2.20 ♦ Truy vấn đệ quy trong DNS

cũng lưu trữ địa chỉ IP của máy chủ TLD, do đó cho phép máy chủ DNS cục bộ bỏ qua các máy chủ DNS gốc trong chuỗi truy vấn. Trên thực tế, do bộ nhớ đệm, các máy chủ gốc bị bỏ qua cho tất cả trừ một phần rất nhỏ các truy vấn DNS.

2.4.3 Bản ghi DNS và tin nhắn

Các máy chủ DNS cùng nhau triển khai các bản ghi tài nguyên lưu trữ cơ sở dữ liệu phân tán DNS (**RR**), bao gồm các RR cung cấp ánh xạ địa chỉ IP từ tên máy chủ đến IP. Mỗi tin nhắn trả lời DNS mang một hoặc nhiều bản ghi tài nguyên. Trong tiểu mục này và phần sau, chúng tôi cung cấp tổng quan ngắn gọn về các bản ghi và thông báo tài nguyên DNS; chi tiết hơn có thể được tìm thấy trong [Albitz 1993] hoặc trong DNS RFC [RFC 1034; RFC 1035].

Bản ghi tài nguyên là một bộ bốn chứa các trường sau:

(Tên, Giá trị, Loại, TTL)

TTL là thời gian tồn tại của bản ghi tài nguyên; nó xác định khi nào một tài nguyên nên được xóa khỏi bộ nhớ cache. Trong các bản ghi ví dụ được đưa ra bên dưới, chúng tôi bỏ qua trường TTL. Ý nghĩa của Tên và Giá trị phụ thuộc vào Loại:

- Nếu Type = A, thì Tên là tên máy chủ và Giá trị là địa chỉ IP cho host-name. Do đó, bản ghi Loại A cung cấp ánh xạ tên máy chủ thành địa chỉ IP tiêu chuẩn. Ví dụ: (relay1.bar.foo.com, 145.37.93.126, A) là bản ghi Loại A.
- Nếu Type = NS, thì Tên là một miền (chẳng hạn như foo.com) và Giá trị là tên máy chủ của một máy chủ DNS có thẩm quyền biết cách lấy địa chỉ IP cho các máy chủ trong miền. Bản ghi này được sử dụng để định tuyến các truy vấn DNS xa hơn trong chuỗi truy vấn. Ví dụ: (foo.com, dns.foo.com, NS) là bản ghi Type NS.
- Nếu Type = CNAME, thì Giá trị là tên máy chủ chuẩn cho Tên tên máy chủ bí danh. Bản ghi này có thể cung cấp cho máy chủ truy vấn tên chuẩn cho tên máy chủ. Ví dụ: (foo.com, relay1.bar.foo.com, CNAME) là bản ghi CNAME.
- Nếu Type = MX, thì Giá trị là tên chuẩn của máy chủ thư có tên máy chủ bí danh Tên. Ví dụ: (foo.com, mail.bar.foo.com, MX) là một bản ghi MX. Bản ghi MX cho phép tên máy chủ của máy chủ thư có bí danh đơn giản. Lưu ý rằng bằng cách sử dụng bản ghi MX, một công ty có thể có cùng tên bí danh cho máy chủ thư của mình và cho một trong các máy chủ khác của công ty (chẳng hạn như máy chủ Web). Để lấy tên chuẩn cho máy chủ thư, máy khách DNS sẽ truy vấn bản ghi MX; để lấy tên chuẩn cho máy chủ khác, máy khách DNS sẽ truy vấn bản ghi CNAME.

Nếu máy chủ DNS có thẩm quyền cho một tên máy chủ cụ thể, thì máy chủ DNS sẽ chứa bản ghi Loại A cho tên máy chủ. (Ngay cả khi máy chủ DNS không có tác giả, nó có thể chứa bản ghi Loại A trong bộ nhớ cache của nó.) Nếu máy chủ không có thẩm quyền cho tên máy chủ, thì máy chủ sẽ chứa bản ghi Loại NS cho miền bao gồm tên máy chủ; nó cũng sẽ chứa một bản ghi Loại A cung cấp địa chỉ IP của máy chủ DNS trong trường Giá trị của bản ghi NS. Ví dụ: giả sử một máy chủ TLD edu không có thẩm quyền cho gaia.cs.umass.edu máy chủ. Sau đó, máy chủ này sẽ chứa một bản ghi cho một tên miền bao gồm máy chủ gaia.cs.umass.edu, ví dụ: (umass.edu, dns.umass.edu, NS). Máy chủ TLD edu cũng sẽ chứa bản ghi Loại A, ánh xạ dns.umass.edu máy chủ DNS đến địa chỉ IP, ví dụ: (dns.umass.edu, 128.119.40.111, A).

Tin nhắn DNS

Trước đó trong phần này, chúng tôi đã đề cập đến truy vấn DNS và tin nhắn trả lời. Đây là hai loại tin nhắn DNS duy nhất. Hơn nữa, cả tin nhắn truy vấn và trả lời đều có cùng định dạng, như thể hiện trong Hình 2.21. Ngữ nghĩa của các trường khác nhau trong thông điệp DNS như sau:

- 12 byte đầu tiên là *phần tiêu đề*, có một số trường. Trường đầu tiên là số 16 bit xác định truy vấn. Mã định danh này được sao chép vào thư trả lời cho một truy vấn, cho phép khách hàng khớp các câu trả lời đã nhận với các truy vấn đã gửi. Có một số cờ trong trường cờ. Cờ truy vấn/trả lời 1 bit cho biết thư là truy vấn (0) hay trả lời (1). Cờ có thẩm quyền 1 bit được đặt trong thư trả lời khi máy chủ DNS là máy chủ có thẩm quyền cho tên được truy vấn. Cờ mong muốn đệ quy 1 bit được đặt khi máy khách (máy chủ lưu trữ hoặc máy chủ DNS) mong muốn máy chủ DNS thực hiện đệ quy khi nó không có bản ghi. Trường có sẵn đệ quy 1 bit được đặt trong thư trả lời nếu máy chủ DNS hỗ trợ đệ quy. Trong tiêu đề, cũng có bốn trường số. Các trường này cho biết số lần xuất hiện của bốn loại phần dữ liệu theo sau tiêu đề.
- Phần *câu hỏi* chứa thông tin về truy vấn đang được thực hiện. Phần này bao gồm (1) trường tên chứa tên đang được truy vấn và (2) trường kiểu cho biết loại câu hỏi đang được hỏi về tên—ví dụ: địa chỉ máy chủ được liên kết với tên (Loại A) hoặc máy chủ thư cho tên (Loại MX).

Xác định	Cờ	— 12 byte
Số lượng câu hỏi	Số lượng RR trả lời	
Số lượng RR có thẩm	Số lượng RR bổ sung	
Câu hỏi (số lượng câu hỏi thay đổi)		— Tên, trường nhập cho truy vấn
Câu trả lời (số lượng bản ghi tài nguyên thay đổi)		— RR để trả lời truy vấn
Thẩm quyền (số lượng bản ghi tài nguyên thay đổi)		— Hồ sơ cho các máy chủ có thẩm quyền
Thông tin bổ sung (số lượng bản ghi tài nguyên thay đổi)		— Thông tin "hữu ích" bổ sung có thể được sử dụng

Hình 2.21 ♦ Định dạng tin nhắn DNS

- Trong thư trả lời từ máy chủ DNS, *phần câu trả lời* chứa các bản ghi tài nguyên cho tên được truy vấn ban đầu. Hãy nhớ lại rằng trong mỗi bản ghi tài nguyên có Loại (ví dụ: A, NS, CNAME và MX), Giá trị và TTL. Một câu trả lời có thể trả về nhiều RR trong câu trả lời, vì một tên máy chủ có thể có nhiều địa chỉ IP (ví dụ, đối với các máy chủ Web được sao chép, như đã thảo luận trước đó trong phần này).
- Phần *thẩm quyền* chứa hồ sơ của các máy chủ có thẩm quyền khác.
- Phần *bổ sung* chứa các bản ghi hữu ích khác. Ví dụ: trường trả lời trong trả lời truy vấn MX chứa bản ghi tài nguyên cung cấp tên máy chủ canonical của máy chủ thư. Phần bổ sung chứa bản ghi Loại A cung cấp địa chỉ IP cho tên máy chủ chuẩn của máy chủ thư.

Bạn muốn gửi thông báo truy vấn DNS trực tiếp từ máy chủ bạn đang làm việc đến một số máy chủ DNS như thế nào? Điều này có thể dễ dàng được thực hiện với **chương trình nslookup**, có sẵn từ hầu hết các nền tảng Windows và UNIX. Ví dụ: từ máy chủ Windows, mở Command Prompt và gọi chương trình nslookup bằng cách chỉ cần gõ "nslookup". Sau khi gọi nslookup, bạn có thể gửi truy vấn DNS đến bất kỳ máy chủ DNS nào (gốc, TLD hoặc có thẩm quyền). Sau khi nhận được tin nhắn trả lời từ máy chủ DNS, nslookup sẽ hiển thị các bản ghi có trong thư trả lời (ở định dạng con người có thể đọc được). Để thay thế cho việc chạy nslookup từ máy chủ của riêng bạn, bạn có thể truy cập một trong nhiều trang Web cho phép bạn sử dụng nslookup từ xa. (Chỉ cần gõ "nslookup" vào công cụ tìm kiếm và bạn sẽ được đưa đến một trong những trang web này.) Phòng thí nghiệm DNS Wireshark ở cuối chương này sẽ cho phép bạn khám phá DNS chi tiết hơn nhiều.

Chèn bản ghi vào cơ sở dữ liệu DNS

Cuộc thảo luận ở trên tập trung vào cách các bản ghi được truy xuất từ cơ sở dữ liệu DNS. Bạn có thể tự hỏi làm thế nào các bản ghi đi vào cơ sở dữ liệu ngay từ đầu. Hãy xem cách điều này được thực hiện trong bối cảnh của một ví dụ cụ thể. Giả sử bạn vừa tạo ra một công ty khởi nghiệp mới thú vị có tên Network Utopia. Điều đầu tiên bạn chắc chắn sẽ muốn làm là đăng ký networkutopia.com tên miền tại một công ty đăng ký. **Reg-istrar** là một thực thể thương mại xác minh tính duy nhất của tên miền, nhập tên miền vào cơ sở dữ liệu DNS (như được thảo luận bên dưới) và thu một khoản phí nhỏ từ bạn cho các dịch vụ của nó. Trước năm 1999, một công ty đăng ký duy nhất, Network Solutions, đã độc quyền đăng ký tên miền cho các tên miền com, net và org. Nhưng bây giờ có rất nhiều nhà đăng ký cạnh tranh cho khách hàng và Tập đoàn Internet về Tên và Số được gán (ICANN) công nhận các nhà đăng ký khác nhau. Danh sách đầy đủ các nhà đăng ký được công nhận có sẵn tại <http://www.internic.net>.

Khi bạn đăng ký tên miền networkutopia.com với một số reg-istrar, bạn cũng cần cung cấp cho nhà đăng ký tên và địa chỉ IP của máy chủ DNS có thẩm quyền chính và phụ của bạn. Giả sử tên và địa chỉ IP là dns1.networkutopia.com, dns2.networkutopia.com, 212.2.212.1 và 212.212.212.2. Đối với mỗi DNS trong số hai DNS có thẩm quyền này



TẬP TRUNG VÀO BẢO

LỖ HỔNG DNS

Chúng tôi đã thấy rằng DNS là một thành phần quan trọng của cơ sở hạ tầng Internet, với nhiều dịch vụ quan trọng — bao gồm Web và e-mail — đơn giản là không có khả năng hoạt động mà không có nó. Do đó, chúng tôi tự nhiên hỏi, DNS có thể bị tấn công như thế nào? DNS có phải là một con vịt đang ngồi, chờ đợi để bị loại khỏi dịch vụ, trong khi gỡ bỏ hầu hết các ứng dụng Internet với nó?

Loại tấn công đầu tiên xuất hiện trong đầu là tấn công DDoS bandwidth-flooding (xem Phần 1.6) chống lại các máy chủ DNS. Ví dụ: kẻ tấn công có thể cố gắng gửi đến mỗi máy chủ gốc DNS một loạt các gói, nhiều đến mức phần lớn các truy vấn DNS hợp pháp không bao giờ được trả lời. Một cuộc tấn công DDoS quy mô lớn như vậy chống lại các máy chủ gốc DNS thực sự đã diễn ra vào ngày 21 tháng 10 năm 2002. Trong cuộc tấn công này, những kẻ tấn công đã tận dụng một mạng botnet để gửi tải xe tải tin nhắn ping ICMP đến mỗi

13 địa chỉ IP gốc DNS. (Thông điệp ICMP được thảo luận trong Phần 5.6. Hiện tại, chỉ cần biết rằng các gói ICMP là các loại sơ đồ IP đặc biệt.) May mắn thay, cuộc tấn công quy mô lớn này gây ra thiệt hại tối thiểu, có ít hoặc không ảnh hưởng đến trải nghiệm Internet của người dùng. Những kẻ tấn công đã thành công trong việc hướng một loạt các gói tin vào các máy chủ gốc. Nhưng nhiều máy chủ gốc DNS được bảo vệ bởi các bộ lọc gói, được cấu hình để luôn chặn tất cả các thông điệp ping ICMP hướng vào các máy chủ gốc. Những

Do đó, các máy chủ được bảo vệ đã được tha và hoạt động như bình thường. Hơn nữa, hầu hết các máy chủ DNS cục bộ lưu trữ địa chỉ IP của các máy chủ tên miền cấp cao nhất, cho phép quá trình truy vấn thường bỏ qua các máy chủ gốc DNS.

Một cuộc tấn công DDoS có khả năng hiệu quả hơn chống lại DNS là gửi một loạt các truy vấn DNS đến các máy chủ tên miền cấp cao nhất, ví dụ, đến các máy chủ tên miền cấp cao nhất xử lý tên miền .com. Việc lọc các truy vấn DNS được chuyển hướng đến máy chủ DNS khó hơn; Và các máy chủ tên miền cấp cao nhất không dễ dàng bị bỏ qua như các máy chủ gốc. Một cuộc tấn công như vậy đã diễn ra chống lại nhà cung cấp dịch vụ tên miền cấp cao nhất Dyn vào ngày 21 tháng 10 năm 2016. Cuộc tấn công DDoS này được thực hiện thông qua một số lượng lớn các yêu cầu tra cứu DNS từ một mạng botnet bao gồm khoảng một trăm nghìn thiết bị IoT như máy in, camera IP, công dân cư và màn hình trẻ em đã bị nhiễm phần mềm độc hại Mirai. Trong gần một ngày, Amazon, Twitter, Netflix, Github và Spotify đã bị xáo trộn.

DNS có khả năng bị tấn công theo những cách khác. Trong một cuộc tấn công trung gian, kẻ tấn công chặn các truy vấn từ máy chủ và trả về các câu trả lời không có thật. Trong cuộc tấn công DNS poisoning, kẻ tấn công gửi trả lời không có thật đến máy chủ DNS, lừa máy chủ chấp nhận các bản ghi không có thật vào bộ nhớ cache của nó. Một trong hai cuộc tấn công này có thể được sử dụng, ví dụ,

để chuyển hướng một người dùng Web không nghi ngờ đến trang web của kẻ tấn công. Phần mở rộng bảo mật DNS (DNSSEC [Gieben 2004; RFC 4033] đã được thiết kế và triển khai để bảo vệ chống lại việc khai thác như vậy. DNSSEC, một phiên bản bảo mật của DNS, giải quyết nhiều cuộc tấn công có thể xảy ra này và

đang trở nên phổ biến trên Internet.

máy chủ, nhà đăng ký sau đó sẽ đảm bảo rằng bản ghi Loại NS và Loại A được nhập vào máy chủ TLD com. Cụ thể, đối với máy chủ có thẩm quyền chính cho `networkutopia.com`, nhà đăng ký sẽ chèn hai bản ghi tài nguyên sau vào hệ thống DNS:

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

Bạn cũng sẽ phải đảm bảo rằng bản ghi tài nguyên Loại A cho máy chủ Web của bạn `www.networkutopia.com` và bản ghi tài nguyên Loại MX cho máy chủ thư của bạn `mail.networkutopia.com` được nhập vào máy chủ DNS có thẩm quyền của bạn. (Cho đến gần đây, nội dung của mỗi máy chủ DNS được định cấu hình tĩnh, ví dụ, từ tệp cấu hình được tạo bởi trình quản lý hệ thống. Gần đây, một tùy chọn UPDATE đã được thêm vào giao thức DNS để cho phép dữ liệu được thêm hoặc xóa khỏi cơ sở dữ liệu thông qua các tin nhắn DNS. [RFC 2136] và [RFC 3007] chỉ định bản Cập Nhật động DNS.)

Khi tất cả các bước này được hoàn thành, mọi người sẽ có thể truy cập trang web của bạn và gửi e-mail cho nhân viên tại công ty của bạn. Hãy kết thúc cuộc thảo luận của chúng ta về DNS bằng cách xác minh rằng tuyên bố này là đúng. Việc xác minh này cũng giúp củng cố những gì chúng ta đã tìm hiểu về DNS. Giả sử Alice ở Úc muốn xem trang web `www.networkutopia.com`. Như đã thảo luận trước đó, trước tiên máy chủ của cô ấy sẽ gửi truy vấn DNS đến máy chủ DNS cục bộ của cô ấy. Máy chủ DNS cục bộ sau đó sẽ liên hệ với máy chủ TLD com. (Máy chủ DNS cục bộ cũng sẽ phải liên hệ với máy chủ DNS gốc nếu địa chỉ của TLD com server không được lưu trữ.) Máy chủ TLD này chứa các bản ghi tài nguyên Loại NS và Loại A được liệt kê ở trên, vì nhà đăng ký đã chèn các bản ghi tài nguyên này vào tất cả các máy chủ TLD com. Máy chủ TLD com gửi trả lời đến máy chủ DNS cục bộ của Alice, với câu trả lời chứa hai bản ghi tài nguyên. Máy chủ DNS cục bộ sau đó gửi truy vấn DNS đến `212.212.212.1`, yêu cầu bản ghi Loại A tương ứng với `www.networkutopia.com`. Bản ghi này cung cấp địa chỉ IP của máy chủ Web mong muốn, giả sử, `212.212.71.4`, mà máy chủ DNS cục bộ chuyển trở lại máy chủ của Alice. Trình duyệt của Alice hiện có thể bắt đầu kết nối TCP với máy chủ `212.212.71.4` và gửi yêu cầu HTTP qua kết nối. Whew! Có rất nhiều điều đang diễn ra hơn những gì bắt mắt khi một người lướt web!

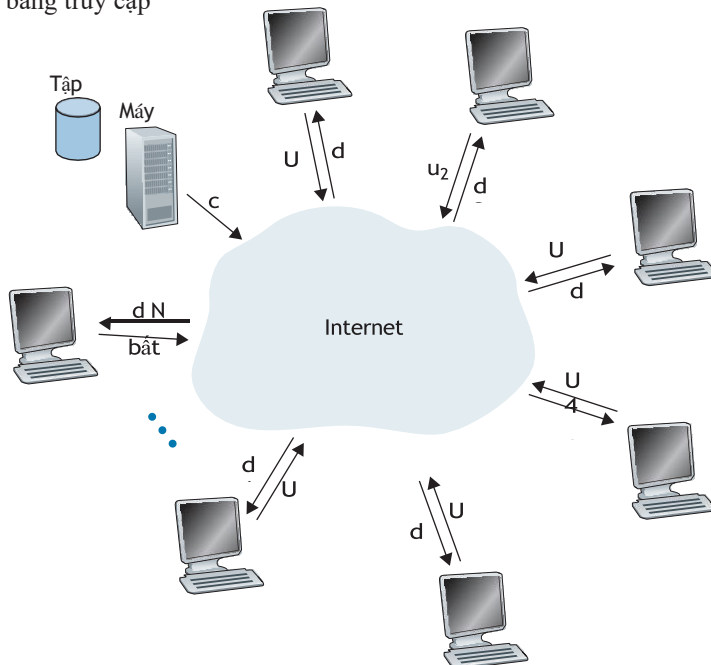
2.5 Phân phối tập ngang hàng

Các ứng dụng được mô tả trong chương này cho đến nay — bao gồm Web, e-mail và DNS — tất cả đều sử dụng kiến trúc máy khách-máy chủ với sự phụ thuộc đáng kể vào các máy chủ cơ sở hạ tầng luôn bật. Hãy nhớ lại từ Phần 2.1.1 rằng với kiến trúc P2P, có sự phụ thuộc tối thiểu (hoặc không) vào các máy chủ cơ sở hạ tầng luôn bật. Thay vào đó, các cặp máy chủ được kết nối không liên tục, được gọi là đồng nghiệp, giao tiếp trực tiếp với nhau. Các đồng nghiệp không thuộc sở hữu của nhà cung cấp dịch vụ, mà thay vào đó là PC, máy tính xách tay và smartphones do người dùng kiểm soát.

Trong phần này, chúng tôi xem xét một ứng dụng P2P rất tự nhiên, cụ thể là phân phối một tập lớn từ một máy chủ duy nhất đến một số lượng lớn máy chủ (được gọi là đồng nghiệp). Tập có thể là phiên bản mới của hệ điều hành Linux, bản vá phần mềm cho hệ điều hành hiện có hoặc tập video MPEG. Trong phân phối tập máy khách-máy chủ, máy chủ phải gửi một bản sao của tập cho mỗi đồng nghiệp — đặt gánh nặng rất lớn lên máy chủ và tiêu tốn một lượng lớn băng thông máy chủ. Trong phân phối tập P2P, mỗi peer có thể phân phối lại bất kỳ phần nào của tập mà nó đã nhận được cho bất kỳ đồng nghiệp nào khác, do đó hỗ trợ máy chủ trong quá trình phân phối. Tính đến năm 2020, giao thức phân phối tập P2P phổ biến nhất là BitTorrent. Ban đầu được phát triển bởi Bram Cohen, hiện nay có nhiều máy khách BitTorrent độc lập khác nhau phù hợp với giao thức Bit-Torrent, giống như có một số máy khách trình duyệt Web phù hợp với giao thức HTTP. Trong tiểu mục này, trước tiên chúng ta kiểm tra khả năng tự mở rộng của kiến trúc P2P trong bối cảnh phân phối tập. Sau đó, chúng tôi mô tả BitTorrent một số chi tiết, làm nổi bật các đặc điểm và tính năng quan trọng nhất của nó.

Khả năng mở rộng của kiến trúc P2P

Để so sánh kiến trúc máy khách-máy chủ với kiến trúc ngang hàng và minh họa khả năng tự mở rộng vốn có của P2P, bây giờ chúng ta xem xét một mô hình định lượng đơn giản để phân phối tập đến một tập hợp ngang hàng cố định cho cả hai loại kiến trúc. Như thể hiện trong Hình 2.22, máy chủ và các đồng nghiệp được kết nối với Internet bằng truy cập



Hình 2.22 ♦ Một vấn đề phân phối tập tin minh họa

Liên kết. Biểu thị tốc độ tải lên liên kết truy cập của máy chủ bởi *chúng tôi*, tốc độ tải lên của liên kết truy cập ngang hàng thứ i theo giao diện người dùng và tốc độ tải xuống của liên kết truy cập ngang hàng thứ i bằng d_i . Cũng biểu thị kích thước của tệp được phân phối (tính bằng bit) bởi F và số lượng đồng nghiệp muốn lấy bản sao của tệp bằng N . Thời **gian phân phối** là thời gian cần thiết để có được một bản sao của tệp cho tất cả các đồng nghiệp N . Trong phân tích của chúng tôi về thời gian phân phối bên dưới, cho cả kiến trúc máy khách-máy chủ và P2P, chúng tôi đưa ra giả định đơn giản hóa (và nói chung là chính xác [Akella 2003]) rằng lõi Internet có băng thông dồi dào, ngụ ý rằng tất cả các tắc nghẽn đều nằm trong mạng truy cập. Chúng tôi cũng giả định rằng máy chủ và máy khách không tham gia vào bất kỳ ứng dụng mạng nào khác, để tất cả băng thông truy cập tải lên và tải xuống của họ có thể được dành hoàn toàn để phân phối tệp này.

Trước tiên, hãy xác định thời gian phân phối cho kiến trúc máy khách-máy chủ, mà chúng tôi biểu thị bằng D_{cs} . Trong kiến trúc máy khách-máy chủ, không có đồng nghiệp nào hỗ trợ phân phối tệp. Chúng tôi thực hiện các quan sát sau:

- Máy chủ phải truyền một bản sao của tệp đến mỗi đồng nghiệp N . Do đó, máy chủ phải truyền *các bit* NF . Vì tốc độ tải lên của máy chủ là *chúng tôi*, thời gian để phân phối tệp phải ít nhất là NF / u_s .
- Hãy để d_{min} biểu thị tốc độ tải xuống của peer có tỷ lệ tải xuống thấp nhất, tức là $d_{min} = \min \{d_1, d_2, \dots, d_N\}$. Đồng nghiệp có tốc độ tải xuống thấp nhất không thể nhận được tất cả các bit F của tệp trong vòng chưa đầy F / d_{min} giây. Do đó, thời gian phân phối tối thiểu ít nhất là F / d_{min} .

Đặt hai quan sát này lại với nhau, chúng ta thu được

$$D_{cs} \geq \max \left\{ \frac{NF}{u_s}, \frac{F}{d_{min}} \right\}.$$

Điều này cung cấp giới hạn thấp hơn về thời gian phân phối tối thiểu cho kiến trúc máy khách-máy chủ. Trong các bài tập về nhà, bạn sẽ được yêu cầu chứng minh rằng máy chủ có thể lên lịch truyền để thực sự đạt được giới hạn dưới. Vì vậy, hãy lấy giới hạn dưới được cung cấp ở trên làm thời gian phân phối thực tế, nghĩa là,

$$D_{cs} = \max \left\{ \frac{NF}{u_s}, \frac{F}{d_{min}} \right\} \quad (2.1)$$

Chúng ta thấy từ Phương trình 2.1 rằng đối với N đủ lớn, thời gian phân phối máy khách-máy chủ được cho bởi NF / u_s . Do đó, thời gian phân phối tăng tuyến tính với số lượng đồng nghiệp N . Vì vậy, ví dụ: nếu số lượng đồng nghiệp từ tuần này sang tuần khác tăng gấp nghìn lần từ một nghìn lên một triệu, thời gian cần thiết để phân phối tệp cho tất cả các đồng nghiệp tăng thêm 1.000.

Bây giờ chúng ta hãy đi qua một phân tích tương tự cho kiến trúc P2P, nơi mỗi peer có thể hỗ trợ máy chủ phân phối tệp. Đặc biệt, khi một peer nhận được một số dữ liệu tệp, nó có thể sử dụng khả năng tải lên của riêng mình để phân phối lại dữ liệu cho các đồng nghiệp khác. Tính toán thời gian phân phối cho kiến trúc P2P có phần phức tạp hơn so với kiến trúc máy khách-máy chủ, vì thời gian phân phối phụ thuộc vào cách mỗi peer phân phối các phần của tệp cho các đồng nghiệp khác. Tuy nhiên, có thể thu được một biểu thức đơn giản cho thời gian phân phối tối thiểu [Kumar 2006]. Để kết thúc này, trước tiên chúng tôi thực hiện các quan sát sau:

- Khi bắt đầu phân phối, chỉ có máy chủ có tệp. Để đưa tệp này vào cộng đồng đồng nghiệp, máy chủ phải gửi từng bit của tệp ít nhất một lần vào liên kết truy cập của nó. Do đó, thời gian phân phối tối thiểu ít nhất là F / us . (Không giống như sơ đồ máy khách-máy chủ, một bit được gửi một lần bởi máy chủ có thể không phải được gửi lại bởi máy chủ, vì các đồng nghiệp có thể phân phối lại bit giữa họ.)
- Như với kiến trúc máy khách-máy chủ, peer có tốc độ tải xuống thấp nhất không thể có được tất cả các bit F của tệp trong vòng chưa đầy $F / dmin$ giây. Do đó, thời gian phân phối tối thiểu ít nhất là $F / dmin$.
- Cuối cùng, hãy quan sát rằng tổng dung lượng tải lên của toàn bộ hệ thống bằng với tốc độ tải lên của máy chủ cộng với tỷ lệ tải lên của từng đồng nghiệp riêng lẻ, nghĩa là $utotal = us + ul + \sum_{i=1}^N ul_i$. Hệ thống phải cung cấp (tải lên) các bit F cho mỗi N peers, do đó cung cấp tổng số bit NF . Điều này không thể được thực hiện với tốc độ nhanh hơn $utotal$. Như vậy, thời gian phân phối tối thiểu cũng ít nhất $NF / (us + ul + \sum_{i=1}^N ul_i)$.

Đặt ba quan sát này lại với nhau, chúng ta có được thời gian phân phối tối thiểu cho P2P, ký hiệu là $DP2P$.

$$DP2P \text{ tối đa c' } \frac{F}{dmin} \frac{NF}{N} \text{ s} \quad (2.2)$$

chúng tôi n \searrow $\begin{matrix} \text{Hoa Kỳ} + \\ \text{Giao} \\ \text{diện người} \\ \text{dùng} \\ i = 1 \end{matrix}$

Phương trình 2.2 cung cấp giới hạn dưới cho thời gian phân phối tối thiểu cho kiến trúc P2P. Nó chỉ ra rằng nếu chúng ta tưởng tượng rằng mỗi peer có thể phân phối lại một chút ngay khi nó nhận được bit, thì có một sơ đồ phân phối lại thực sự đạt được giới hạn dưới này [Kumar 2006]. (Chúng tôi sẽ chứng minh một trường hợp đặc biệt của kết quả này trong bài tập về nhà.) Trong thực tế, khi các khối của tệp được phân phối lại thay vì các bit riêng, Phương trình 2.2 đóng vai trò là một xấp xỉ tốt về thời gian phân phối tối thiểu thực tế. Do đó, hãy lấy cận dưới được cung cấp bởi Phương trình 2.2 làm thời gian phân phối tối thiểu thực tế, nghĩa là,

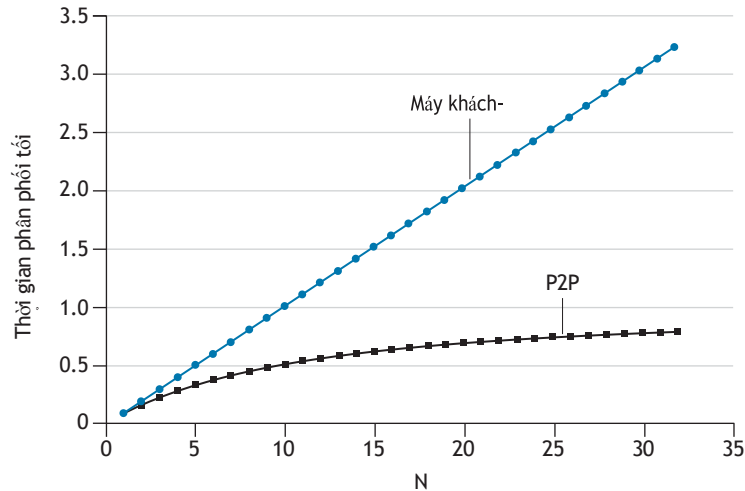
$$DP2P = \text{tối đa c' } \frac{F}{dmin} \frac{NF}{N} \text{ s} \quad \text{c} \quad \text{h} \quad \text{úng tôi}$$

$dmin$

$$Ho$$
$$a$$
$$Kỳ$$
$$+$$
$$\backslash$$
$$Gia$$
$$o$$
$$diệ$$
$$n$$
$$ngư$$
$$ời$$
$$dùn$$
$$g$$

$$S^N$$
$$(2.3)$$

$$\begin{matrix} i \\ = \\ 1 \end{matrix}$$



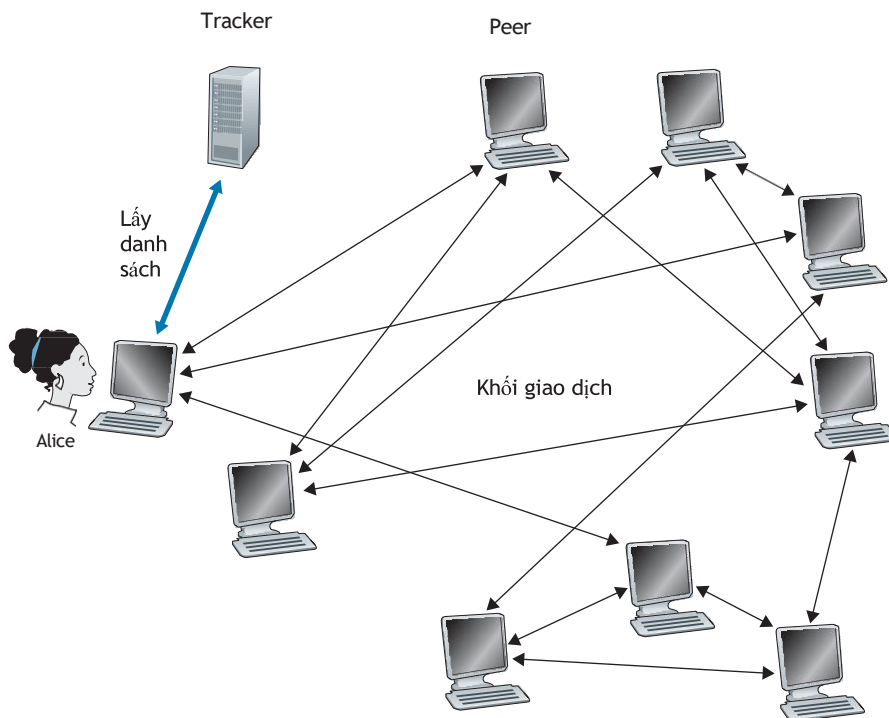
Hình 2.23 ♦ Thời gian phân phối cho kiến trúc P2P và máy khách-máy chủ

Hình 2.23 so sánh thời gian phân phối tối thiểu cho máy khách-máy chủ và kiến trúc P2P giả định rằng tất cả các công ty ngang hàng có cùng tốc độ tải lên u . Trong Hình 2.23, chúng ta đã đặt $F/u = 1$ giờ, $u_s = 10u$ và $d_{\min} \leq u_s$. Do đó, một peer có thể truyền toàn bộ tệp trong một giờ, tốc độ truyền của máy chủ gấp 10 lần tốc độ tải lên ngang hàng và (để đơn giản) tốc độ tải xuống ngang hàng được đặt đủ lớn để không có

một hiệu ứng. Chúng ta thấy từ Hình 2.23 rằng đối với kiến trúc máy khách-máy chủ, thời gian phân phối tăng tuyến tính và không bị ràng buộc khi số lượng đồng nghiệp tăng lên. Tuy nhiên, đối với kiến trúc P2P, thời gian phân phối tối thiểu không những luôn ít hơn thời gian phân phối của kiến trúc máy khách-máy chủ; nó cũng ít hơn một giờ cho *bất kỳ* số lượng đồng nghiệp N . Do đó, các ứng dụng với kiến trúc P2P có thể tự mở rộng quy mô. Khả năng mở rộng này là hậu quả trực tiếp của các đồng nghiệp là nhà phân phối lại cũng như người tiêu dùng bit.

BitTorrent

BitTorrent là một giao thức P2P phổ biến để phân phối tệp [Chao 2011]. Trong biệt ngữ BitTorrent, bộ sưu tập của tất cả các đồng nghiệp tham gia phân phối một tệp cụ thể được gọi là *torrent*. Các đồng nghiệp trong torrent tải xuống các khối có kích thước bằng nhau của tệp với nhau, với kích thước đoạn điển hình là 256 KBytes. Khi một peer lần đầu tiên tham gia một torrent, nó không có khối. Theo thời gian, nó tích lũy ngày càng nhiều khối. Trong khi nó tải xuống các khối, nó cũng tải các đoạn lên các đồng nghiệp khác. Khi một peer đã có được toàn bộ tệp, nó có thể (ích kỷ) rời khỏi torrent hoặc (vị tha) vẫn ở trong torrent và tiếp tục tải các đoạn lên các đồng nghiệp khác. Ngoài ra, bất kỳ người ngang hàng nào cũng có thể rời khỏi torrent bất cứ lúc nào chỉ với một tập hợp con các khối và sau đó tham gia lại torrent.



Hình 2.24 ♦ Phân phối file với BitTorrent

Bây giờ chúng ta hãy xem xét kỹ hơn cách BitTorrent hoạt động. Vì BitTorrent là một giao thức và hệ thống khá phức tạp, chúng tôi sẽ chỉ mô tả các cơ chế quan trọng nhất của nó, quét một số chi tiết dưới tấm thảm; Điều này sẽ cho phép chúng ta nhìn thấy khu rừng qua những tán cây. Mỗi torrent có một nút cơ sở hạ tầng được gọi là *trình theo dõi*. Khi một peer tham gia một torrent, nó sẽ tự đăng ký với trình theo dõi và định kỳ thông báo cho trình theo dõi rằng nó vẫn còn trong torrent. Theo cách này, trình theo dõi theo dõi các đồng nghiệp đang tham gia vào torrent. Một torrent nhất định có thể có ít hơn mười hoặc hơn một nghìn đồng nghiệp tham gia vào bất kỳ thời điểm nào.

Như thể hiện trong Hình 2.24, khi một đồng nghiệp mới, Alice, tham gia torrent, trình theo dõi chọn ngẫu nhiên một tập hợp con các đồng nghiệp (để cụ thể, giả sử 50) từ tập hợp các đồng nghiệp tham gia và gửi địa chỉ IP của 50 đồng nghiệp này cho Alice. Sở hữu danh sách các đồng nghiệp này, Alice cố gắng thiết lập các kết nối TCP đồng thời với tất cả các đồng nghiệp trong danh sách này. Hãy gọi tất cả các đồng nghiệp mà Alice thành công trong việc thiết lập kết nối TCP là "đồng nghiệp lân cận". (Trong Hình 2.24, Alice được chứng minh là chỉ có ba đồng nghiệp lân cận. Thông thường, cô ấy sẽ có nhiều hơn nữa.) Khi thời gian phát triển, một số đồng nghiệp này có thể rời đi và các đồng nghiệp khác (ngoài 50 ban đầu) có thể cố gắng thiết lập kết nối TCP với Alice. Vì vậy, các đồng nghiệp lân cận của một đồng nghiệp ngang hàng sẽ dao động theo thời gian.

Tại bất kỳ thời điểm nào, mỗi peer sẽ có một tập hợp con các khối từ tệp, với các đồng nghiệp different có các tập con khác nhau. Theo định kỳ, Alice sẽ hỏi từng đồng nghiệp nhằm chán của mình (qua các kết nối TCP) cho danh sách các phần mà họ có. Nếu Alice có L hàng xóm khác nhau, cô ấy sẽ nhận được danh sách L của khối. Với kiến thức này, Alice sẽ đưa ra yêu cầu (một lần nữa qua các kết nối TCP) cho các đoạn mà cô ấy hiện không có.

Vì vậy, tại bất kỳ thời điểm nào, Alice sẽ có một tập hợp con các khối và sẽ biết hàng xóm của cô ấy có khối nào. Với thông tin này, Alice sẽ có hai quyết định không cần thiết phải đưa ra. Đầu tiên, cô ấy nên yêu cầu phần nào đầu tiên từ neighbors của mình? Và thứ hai, cô ấy nên gửi cho những người hàng xóm nào của cô ấy những phần được yêu cầu? Khi quyết định yêu cầu đoạn nào, Alice sử dụng một kỹ thuật gọi là **hiếm nhất trước**. Ý tưởng là xác định, trong số những khối mà cô ấy không có, những khối hiếm nhất trong số những người hàng xóm của cô ấy (nghĩa là những khối có ít bản sao lặp đi lặp lại nhất trong số những người hàng xóm của cô ấy) và sau đó yêu cầu những khối hiếm nhất đó trước. Theo cách này, các khối hiếm nhất được phân phối lại nhanh hơn, nhằm mục đích (đại khái) cân bằng số lượng bản sao của mỗi đoạn trong torrent.

Để xác định yêu cầu nào cô ấy phản hồi, BitTorrent sử dụng một thuật toán giao dịch thông minh. Ý tưởng cơ bản là Alice ưu tiên cho những người hàng xóm đang cung cấp dữ liệu của cô ấy với *tỷ lệ cao nhất*. Cụ thể, đối với mỗi người hàng xóm của mình, Alice liên tục đo tốc độ cô nhận được bit và xác định bốn đồng nghiệp đang cho cô ăn bit với tốc độ cao nhất. Sau đó, cô ấy đáp lại bằng cách gửi khối cho bốn đồng nghiệp này. Cứ sau 10 giây, cô lại tính toán lại tỷ lệ và có thể sửa đổi bộ bốn đồng nghiệp. Trong biệt ngữ BitTorrent, bốn đồng nghiệp này được cho là **không bị nghẹt thở**. Điều quan trọng, cứ sau 30 giây, cô ấy cũng chọn ngẫu nhiên thêm một người hàng xóm và gửi nó đi. Hãy gọi đồng nghiệp được chọn ngẫu nhiên thêm một người hàng xóm và gửi nó đi. Hãy gọi đồng nghiệp được chọn ngẫu nhiên Bob. Trong biệt ngữ BitTorrent, Bob được cho là **lạc quan không bị nghẹt thở**. Bởi vì Alice đang gửi dữ liệu cho Bob, cô ấy có thể trở thành một trong bốn người tải lên hàng đầu của Bob, trong trường hợp đó Bob sẽ bắt đầu gửi dữ liệu cho Alice. Nếu tốc độ Bob gửi dữ liệu cho Alice đủ cao, Bob có thể trở thành một trong bốn người tải lên hàng đầu của Alice. Nói cách khác, cứ sau 30 giây, Alice sẽ chọn ngẫu nhiên một đối tác giao dịch mới và bắt đầu giao dịch với đối tác đó. Nếu hai đồng nghiệp hài lòng với giao dịch, họ sẽ đưa nhau vào bốn danh sách hàng đầu của họ và tiếp tục giao dịch với nhau cho đến khi một trong những đồng nghiệp tìm thấy một đối tác tốt hơn. Hiệu quả là các đồng nghiệp có khả năng tải lên ở mức tương thích có xu hướng tìm thấy nhau. Việc lựa chọn hàng xóm ngẫu nhiên cũng cho phép các đồng nghiệp mới có được các khối, để họ có thể có thứ gì đó để giao dịch. Tất cả các đồng nghiệp lân cận khác ngoài năm đồng nghiệp này (bốn đồng nghiệp "hàng đầu" và một đồng nghiệp thăm dò) đều bị "nghẹt thở", nghĩa là họ không nhận được bất kỳ phần nào từ Alice. BitTorrent có một số cơ chế thú vị không được thảo luận ở đây, bao gồm các phần (khối nhỏ), pipelining, lựa chọn đầu tiên ngẫu nhiên, chế độ endgame và chống snubbing [Cohen 2003].

Cơ chế khuyến khích giao dịch vừa được mô tả thường được gọi là ăn miếng trả miếng [Cohen 2003]. Nó đã được chứng minh rằng chương trình khuyến khích này có thể bị phá vỡ [Liogkas 2006; Locher 2006; Piatek 2008]. Tuy nhiên, hệ sinh thái BitTorrent cực kỳ thành công, với hàng triệu đồng nghiệp tích cực chia sẻ tệp trong

hàng trăm ngàn torrent. Nếu BitTorrent được thiết kế mà không ăn miếng trả miếng (hoặc một biến thể), nhưng nếu không thì hoàn toàn giống nhau, BitTorrent thậm chí có thể sẽ không tồn tại bây giờ, vì phần lớn người dùng sẽ là freeriders [Saroiu 2002].

Chúng tôi kết thúc cuộc thảo luận về P2P bằng cách đề cập ngắn gọn đến một ứng dụng khác của P2P, cụ thể là Bảng Hast phân tán (DHT). Bảng băm phân tán là một cơ sở dữ liệu đơn giản, với các bản ghi cơ sở dữ liệu được phân phối trên các đồng nghiệp trong hệ thống P2P. DHT đã được triển khai rộng rãi (ví dụ: trong BitTorrent) và là chủ đề của nghiên cứu sâu rộng. Tổng quan được cung cấp trong Ghi chú Video trong Trang web Đồng hành.



Ghi chú video
Đi bộ qua các bảng băm
phân tán

2.6 Mạng phân phối nội dung và phát trực tuyến video

Theo nhiều ước tính, phát trực tuyến video — bao gồm Netflix, YouTube và Amazon Prime — chiếm khoảng 80% lưu lượng truy cập Internet vào năm 2020 [Cisco 2020]. Phần này chúng tôi sẽ cung cấp một cái nhìn tổng quan về cách các dịch vụ phát trực tuyến video phổ biến được đề cập trong Internet ngày nay. Chúng ta sẽ thấy chúng được triển khai bằng cách sử dụng các giao thức và máy chủ cấp ứng dụng hoạt động theo một số cách như bộ đệm.

2.6.1 Internet Video

Trong phát trực tuyến các ứng dụng video được lưu trữ, phương tiện cơ bản là video được ghi sẵn, chẳng hạn như phim, chương trình truyền hình, sự kiện thể thao được ghi sẵn hoặc video do người dùng tạo được ghi sẵn (chẳng hạn như những video thường thấy trên YouTube). Các video có dây trước này được đặt trên máy chủ và người dùng gửi yêu cầu đến máy chủ để xem video *theo yêu cầu*. Nhiều công ty Internet ngày nay cung cấp video phát trực tuyến, bao gồm Netflix, YouTube (Google), Amazon và TikTok.

Nhưng trước khi bắt đầu thảo luận về phát trực tuyến video, trước tiên chúng ta nên cảm nhận nhanh về chính phương tiện video. Video là một chuỗi các hình ảnh, thường được hiển thị ở tốc độ không đổi, ví dụ: ở tốc độ 24 hoặc 30 hình ảnh mỗi giây. Một hình ảnh không nén, được mã hóa kỹ thuật số bao gồm một mảng các pixel, với mỗi pixel được mã hóa thành một số bit để đại diện cho độ chói và màu sắc. Một đặc điểm quan trọng của video là nó có thể được nén, do đó đánh đổi chất lượng video với tốc độ bit. Các thuật toán nén có sẵn ngày nay có thể nén video về cơ bản bất kỳ tốc độ bit nào mong muốn. Tất nhiên, tốc độ bit càng cao, chất lượng hình ảnh càng tốt và trải nghiệm xem tổng thể của người dùng càng tốt.

Từ góc độ mạng, có lẽ đặc điểm nổi bật nhất của video là tốc độ bit cao. Video Internet nén thường dao động từ 100 kbps đối với video chất lượng thấp đến hơn 4 Mbps để phát trực tuyến phim độ nét cao; Phát trực tuyến 4K hình dung tốc độ bit hơn 10 Mbps. Điều này có thể chuyển thành lượng lưu lượng truy cập và dung lượng lưu trữ khổng lồ, đặc biệt là đối với video cao cấp. Ví dụ: một 2 Mbps duy nhất

Video có thời lượng 67 phút sẽ tiêu tốn 1 gigabyte dung lượng lưu trữ và lưu lượng truy cập. Cho đến nay, thước đo hiệu suất quan trọng nhất để phát trực tuyến video là thông lượng trung bình từ đầu đến cuối. Để cung cấp bố cục liên tục, mạng phải cung cấp thông lượng trung bình cho ứng dụng phát trực tuyến ít nhất bằng tốc độ bit của video nén.

Chúng tôi cũng có thể sử dụng tính năng nén để tạo nhiều phiên bản của cùng một video, mỗi phiên bản ở một mức chất lượng khác nhau. Ví dụ: chúng ta có thể sử dụng nén để tạo, giả sử, ba phiên bản của cùng một video, với tốc độ 300 kbps, 1 Mbps và 3 Mbps. Sau đó, người dùng có thể quyết định phiên bản nào họ muốn xem như một chức năng của băng thông có sẵn hiện tại của họ. Người dùng có kết nối Internet tốc độ cao có thể chọn phiên bản 3 Mbps; Người dùng xem video qua 3G bằng điện thoại thông minh có thể chọn phiên bản 300 kbps.

2.6.2 Truyền phát HTTP và DASH

Trong phát trực tuyến HTTP, video được lưu trữ đơn giản tại máy chủ HTTP dưới dạng tệp thông thường với một URL cụ thể. Khi người dùng muốn xem video, máy khách thiết lập kết nối TCP với máy chủ và đưa ra yêu cầu HTTP GET cho URL đó. Sau đó, máy chủ sẽ gửi tệp video, trong thông báo phản hồi HTTP, nhanh nhất khi các giao thức mạng cơ bản và điều kiện lưu lượng cho phép. Về phía máy khách, các byte được thu thập trong bộ đệm ứng dụng khách. Khi số byte trong bộ đệm này vượt quá ngưỡng xác định trước, ứng dụng khách bắt đầu phát lại — cụ thể, ứng dụng video trực tuyến định kỳ lấy khung hình video từ bộ đệm ứng dụng khách, giải nén khung hình và hiển thị chúng trên màn hình của người dùng. Do đó, ứng dụng phát trực tuyến video đang hiển thị video khi nó đang nhận và đệm các khung hình tương ứng với các phần sau của video.

Mặc dù phát trực tuyến HTTP, như được mô tả trong đoạn trước, đã được triển khai rộng rãi trong thực tế (ví dụ: YouTube kể từ khi thành lập), nhưng nó có một thiếu sót lớn: Tất cả các máy khách đều nhận được cùng một mã hóa video, mặc dù có sự thay đổi lớn về lượng băng thông có sẵn cho máy khách, cả trên các máy khách khác nhau và cũng theo thời gian cho cùng một máy khách. Điều này đã dẫn đến sự phát triển của một loại phát trực tuyến dựa trên HTTP mới, thường được gọi là **Truyền phát thích ứng động qua HTTP (DASH)**. Trong DASH, video được mã hóa thành nhiều phiên bản khác nhau, với mỗi phiên bản có tốc độ bit khác nhau và tương ứng, một mức chất lượng khác nhau. Khách hàng tự động yêu cầu các đoạn video có độ dài vài giây. Khi lượng băng thông có sẵn cao, khách hàng tự nhiên chọn các đoạn từ phiên bản tốc độ cao; Và khi băng thông có sẵn thấp, nó tự nhiên chọn từ phiên bản tốc độ thấp. Máy khách chọn các đoạn khác nhau cùng một lúc với thông báo yêu cầu HTTP GET [Akhshabi 2011].

DASH cho phép khách hàng có tốc độ truy cập Internet khác nhau phát trực tuyến video ở các tốc độ mã hóa khác nhau. Máy khách có kết nối 3G tốc độ thấp có thể nhận được phiên bản tốc độ bit thấp (và chất lượng thấp) và máy khách có kết nối cáp quang có thể nhận được phiên bản chất lượng cao. DASH cũng cho phép khách hàng thích ứng với băng thông có sẵn nếu băng thông end-to-end có sẵn thay đổi trong phiên. Tính năng này là

Đặc biệt quan trọng đối với người dùng di động, những người thường thấy băng thông của họ dao động khi họ di chuyển đối với các trạm gốc.

Với DASH, mỗi phiên bản video được lưu trữ trong máy chủ HTTP, mỗi phiên bản có một URL khác nhau. Máy chủ HTTP cũng có một **tệp kê khai**, cung cấp URL cho mỗi phiên bản cùng với tốc độ bit của nó. Trước tiên, khách hàng yêu cầu tệp kê khai và tìm hiểu về các phiên bản khác nhau. Sau đó, máy khách chọn một đoạn tại một thời điểm bằng cách chỉ định URL và phạm vi byte trong thông báo yêu cầu HTTP GET cho mỗi đoạn. Trong khi tải xuống các khối, máy khách cũng đo băng thông nhận được và chạy thuật toán ngăn chặn tốc độ để chọn đoạn để yêu cầu tiếp theo. Đương nhiên, nếu máy khách có nhiều bộ đệm video và nếu băng thông nhận được đo được cao, nó sẽ chọn một đoạn từ phiên bản tốc độ bit cao. Và đương nhiên, nếu máy khách có ít bộ đệm video và băng thông nhận được đo được thấp, nó sẽ chọn một đoạn từ phiên bản tốc độ bit thấp. Do đó, DASH cho phép khách hàng tự do chuyển đổi giữa các mức chất lượng khác nhau.

2.6.3 Mạng lưới phân phối nội dung

Ngày nay, nhiều công ty video Internet đang phân phối các luồng đa Mbps theo yêu cầu cho hàng triệu người dùng hàng ngày. YouTube, ví dụ, với thư viện hàng trăm triệu video, phân phối hàng trăm triệu luồng video cho người dùng trên khắp thế giới mỗi ngày. Truyền phát tất cả lưu lượng truy cập này đến các địa điểm trên toàn thế giới trong khi cung cấp khả năng phát liên tục và tính tương tác cao rõ ràng là một nhiệm vụ khó khăn.

Đối với một công ty video Internet, có lẽ cách tiếp cận đơn giản nhất để cung cấp dịch vụ video trực tuyến là xây dựng một trung tâm dữ liệu khổng lồ duy nhất, lưu trữ tất cả các video của mình trong trung tâm dữ liệu và truyền phát video trực tiếp từ trung tâm dữ liệu đến khách hàng trên toàn thế giới. Nhưng có ba vấn đề lớn với phương pháp này. Đầu tiên, nếu máy khách ở xa trung tâm dữ liệu, các gói từ máy chủ đến máy khách sẽ đi qua nhiều liên kết giao tiếp và có khả năng đi qua nhiều ISP, với một số ISP có thể nằm ở các lục địa khác nhau. Nếu một trong các liên kết này cung cấp thông lượng thấp hơn tốc độ tiêu thụ video, thông lượng từ đầu đến cuối cũng sẽ thấp hơn tốc độ tiêu thụ, dẫn đến sự chậm trễ đáng kể khó chịu cho người dùng. (Hãy nhớ lại từ Chương 1 rằng thông lượng từ đầu đến cuối của luồng bị chi phối bởi thông lượng tại liên kết nút cổ chai.) Khả năng điều này xảy ra tăng lên khi số lượng liên kết trong đường dẫn từ đầu đến cuối tăng lên. Hạn chế thứ hai là một video phổ biến có thể sẽ được gửi nhiều lần qua cùng một liên kết truyền thông. Điều này không chỉ lãng phí băng thông mạng, mà chính công ty video Internet sẽ trả tiền cho nhà cung cấp ISP (được kết nối với trung tâm dữ liệu) để gửi *cùng một byte* vào Inter-net nhiều lần. Vấn đề thứ ba với giải pháp này là một trung tâm dữ liệu duy nhất đại diện cho một điểm thất bại duy nhất — nếu trung tâm dữ liệu hoặc các liên kết của nó với Internet bị hỏng, nó sẽ không thể phân phối *bất kỳ* luồng video nào.

Để đáp ứng thách thức phân phối lưu lượng dữ liệu video khổng lồ cho người dùng được phân phối trên toàn thế giới, hầu hết tất cả các công ty phát trực tuyến video lớn đều sử dụng **Mạng phân phối nội dung (CDN)**. CDN quản lý các máy chủ trong

nhiều vị trí phân bố theo địa lý, lưu trữ các bản sao của video (và các loại nội dung Web khác, bao gồm tài liệu, hình ảnh và âm thanh) trong các máy chủ của nó và cố gắng hướng mỗi yêu cầu của người dùng đến một vị trí CDN sẽ cung cấp trải nghiệm người dùng tốt nhất. CDN có thể là CDN **riêng**, nghĩa là thuộc sở hữu của chính nhà cung cấp nội dung; ví dụ: CDN của Google phân phối video YouTube và các loại nội dung khác. CDN có thể là CDN **của bên thứ ba** phân phối nội dung thay mặt cho nhiều nhà cung cấp nội dung; Akamai, Limelight và Level-3 đều vận hành CDN của bên thứ ba. Một cái nhìn tổng quan rất dễ đọc về CDN hiện đại là [Leighton 2009; Nygren 2010].


CDN thường áp dụng một trong hai triết lý vị trí máy chủ khác nhau [Huang 2008]:

- **Nhập sâu.** Một triết lý, được tiên phong bởi Akamai, là *thâm nhập sâu* vào mạng truy cập của các nhà cung cấp dịch vụ Internet, bằng cách triển khai các cụm máy chủ trong các ISP truy cập trên toàn thế giới. (Mạng truy cập được mô tả trong Phần 1.3.) Akamai áp dụng cách tiếp cận này với các cụm ở hàng nghìn địa điểm. Mục tiêu là đến gần người dùng cuối, do đó cải thiện độ trễ và thông lượng nhận thức của người dùng bằng cách giảm số lượng liên kết và bộ định tuyến giữa người dùng cuối và máy chủ CDN mà từ đó nó nhận được nội dung. Do thiết kế phân tán cao này, nhiệm vụ duy trì và quản lý các cụm trở nên khó khăn.
- **Mang về nhà.** Triết lý thiết kế thứ hai, được thực hiện bởi Limelight và nhiều công ty CDN khác, là *đưa các ISP về nhà* bằng cách xây dựng các cụm lớn với số lượng nhỏ hơn (ví dụ: hàng chục) trang web. Thay vì vào bên trong các ISP truy cập, các CDN này thường đặt các cụm của chúng trong Điểm trao đổi Internet (IXP) (xem Phần 1.3). So với phi-losophy thiết kế chuyên sâu, thiết kế mang về nhà thường dẫn đến chi phí bảo trì và quản lý thấp hơn, có thể phải trả giá bằng độ trễ cao hơn và thông lượng thấp hơn cho người dùng cuối.

Khi các cụm của nó được đặt đúng chỗ, CDN sẽ sao chép nội dung trên các cụm của nó. CDN có thể không muốn đặt một bản sao của mọi video trong mỗi cụm, vì một số video hiếm khi được xem hoặc chỉ phổ biến ở một số quốc gia. Trên thực tế, nhiều CDN không đẩy video vào cụm của chúng mà thay vào đó sử dụng chiến lược kéo đơn giản: Nếu máy khách yêu cầu video từ cụm không lưu trữ video, thì cụm sẽ truy xuất video (từ kho lưu trữ trung tâm hoặc từ cụm khác) và lưu trữ một bản sao cục bộ trong khi truyền video đến máy khách cùng một lúc. Bộ nhớ đệm Web tương tự (xem Phần 2.2.5), khi bộ nhớ của cụm đầy, nó sẽ xóa các video không được yêu cầu thường xuyên.

Hoạt động CDN

Sau khi xác định hai cách tiếp cận chính để triển khai CDN, bây giờ chúng ta hãy đi sâu vào các vấn đề cơ bản về cách CDN hoạt động. Khi trình duyệt trong trình duyệt của người dùng



CƠ SỞ HẠ TẦNG MẠNG CỦA GOOGLE

Để hỗ trợ một loạt các dịch vụ của mình - bao gồm tìm kiếm, Gmail, lịch, video YouTube, bản đồ, tài liệu và mạng xã hội - Google đã triển khai một mạng riêng và cơ sở hạ tầng CDN rộng lớn. Cơ sở hạ tầng CDN của Google có ba tầng cụm máy chủ:

- Mười chín "trung tâm dữ liệu lớn" ở Bắc Mỹ, Châu Âu và Châu Á [Vị trí của Google 2020], với mỗi trung tâm dữ liệu có đơn đặt hàng 100.000 máy chủ. Các trung tâm dữ liệu lớn này chịu trách nhiệm phục vụ nội dung động (và thường được cá nhân hóa), bao gồm kết quả tìm kiếm và thư Gmail.
- Với khoảng 90 cụm trong IXP nằm rải rác trên khắp thế giới, với mỗi cụm bao gồm hàng trăm máy chủ máy chủ [Adhikari 2011a] [Google CDN 2020]. Các cụm này chịu trách nhiệm phục vụ nội dung tĩnh, bao gồm cả video YouTube.
- Hàng trăm cụm "enter-deep" nằm trong một ISP truy cập. Ở đây một cụm thường bao gồm hàng chục máy chủ trong một rack duy nhất. Các máy chủ enter-deep này thực hiện tách TCP (xem Phần 3.7) và phục vụ nội dung tĩnh [Chen 2011], bao gồm các phần tĩnh của các trang Web thể hiện kết quả tìm kiếm.

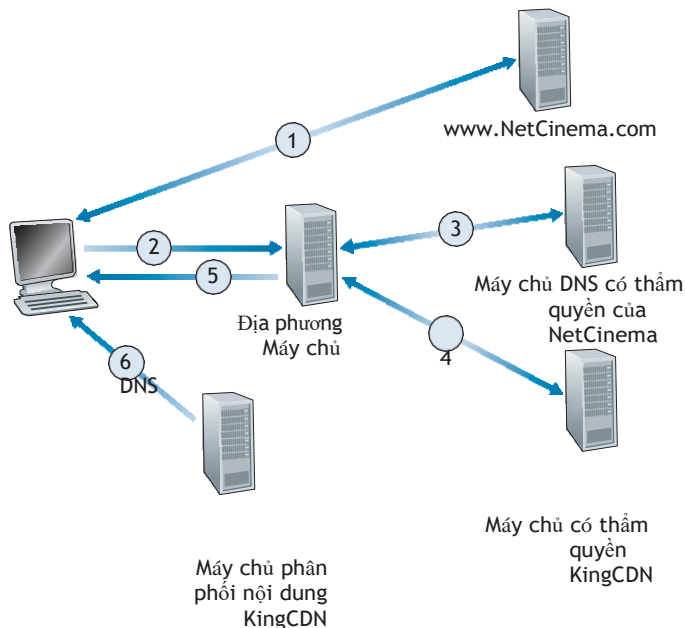
Tất cả các trung tâm dữ liệu và vị trí cụm này được nối mạng cùng với mạng riêng của Google. Khi người dùng thực hiện truy vấn tìm kiếm, thường truy vấn đầu tiên được gửi qua ISP cục bộ đến bộ đệm enter deep gần đó, từ đó nội dung tĩnh được truy xuất; trong khi cung cấp nội dung tĩnh cho máy khách, bộ nhớ cache gần đó cũng chuyển tiếp truy vấn qua mạng riêng của Google đến một trong những trung tâm dữ liệu lớn, từ đó kết quả tìm kiếm được cá nhân hóa được truy xuất. Đối với video YouTube, bản thân video có thể đến từ một trong những bộ nhớ cache mang về nhà, trong khi các phần của trang web xung quanh video có thể đến từ bộ nhớ cache sâu gần đó và các quảng cáo xung quanh video đến từ các trung tâm dữ liệu. Tóm lại, ngoại trừ các ISP địa phương, các dịch vụ đám mây của Google phần lớn được cung cấp bởi cơ sở hạ tầng mạng độc lập với Internet công cộng.

host được hướng dẫn truy xuất một video cụ thể (được xác định bằng URL), CDN phải chặn request để có thể (1) xác định cụm máy chủ CDN phù hợp cho client đó tại thời điểm đó, và (2) redirect request của client đến server trong cluster đó. Chúng ta sẽ sớm thảo luận về cách CDN có thể xác định một cụm phù hợp. Nhưng trước tiên hãy kiểm tra cơ chế đằng sau việc chặn và chuyển hướng một yêu cầu.

Hầu hết các CDN tận dụng DNS để chặn và chuyển hướng các yêu cầu; một cuộc thảo luận xen kẽ về việc sử dụng DNS như vậy là [Vixie 2009]. Hãy xem xét một cách đơn giản

ví dụ để minh họa cách DNS thường tham gia. Giả sử một nhà cung cấp nội dung, NetCinema, sử dụng công ty CDN bên thứ ba, KingCDN, để phân phối vid-eos của mình cho khách hàng của mình. Trên các trang web NetCinema, mỗi video của nó được gán một URL bao gồm chuỗi "video" và một mã định danh duy nhất cho chính video; ví dụ: *Transformers 7* có thể được gán `http://video.netcinema.com/6Y7B23V`. Sáu bước sau đó xảy ra, như thể hiện trong Hình 2.25:

1. Người dùng truy cập trang Web tại NetCinema.
2. Khi người dùng nhấp vào `http://video.netcinema.com/6Y7B23V` liên kết, máy chủ lưu trữ của người dùng sẽ gửi truy vấn DNS cho `video.netcinema.com`.
3. Máy chủ DNS cục bộ của người dùng (LDNS) chuyển tiếp truy vấn DNS đến máy chủ DNS có thẩm quyền cho NetCinema, máy chủ này quan sát chuỗi "video" trong `video.netcinema.com` tên máy chủ. Để "bàn giao" truy vấn DNS cho KingCDN, thay vì trả về địa chỉ IP, máy chủ DNS có thẩm quyền của NetCinema trả về LDNS một tên máy chủ trong miền của KingCDN, ví dụ: `a1105.kingcdn.com`.
4. Từ thời điểm này, truy vấn DNS đi vào cấu trúc cơ sở hạ tầng DNS riêng của KingCDN. LDNS của người dùng sau đó gửi truy vấn thứ hai, bây giờ cho `a1105.kingcdn.com` và hệ thống DNS của KingCDN cuối cùng trả về địa chỉ IP của máy chủ nội dung KingCDN cho LDNS. Do đó, ở đây, trong hệ thống DNS của KingCDN, máy chủ CDN mà từ đó máy khách sẽ nhận được nội dung của nó được chỉ định.



Hình 2.25 ♦ DNS chuyển hướng yêu cầu của người dùng đến máy chủ CDN

- LDNS chuyển tiếp địa chỉ IP của nút CDN phục vụ nội dung đến máy chủ của người dùng.
- Khi máy khách nhận được địa chỉ IP cho máy chủ nội dung KingCDN, nó sẽ thiết lập kết nối TCP trực tiếp với máy chủ tại địa chỉ IP đó và đưa ra yêu cầu HTTP GET cho video. Nếu DASH được sử dụng, trước tiên máy chủ sẽ gửi cho máy khách một tệp kê khai với danh sách các URL, một cho mỗi phiên bản của video và máy khách sẽ tự động chọn các đoạn từ các phiên bản khác nhau.

Chiến lược lựa chọn cụm

Cốt lõi của bất kỳ triển khai CDN nào là **chiến lược lựa chọn cụm**, nghĩa là cơ chế để hướng động các máy khách đến cụm máy chủ hoặc trung tâm dữ liệu trong CDN. Như chúng ta vừa thấy, CDN tìm hiểu địa chỉ IP của máy chủ LDNS của máy khách thông qua tra cứu DNS của máy khách. Sau khi tìm hiểu địa chỉ IP này, CDN cần chọn một cụm thích hợp dựa trên địa chỉ IP này. CDN thường sử dụng các chiến lược lựa chọn cụm độc quyền. Bây giờ chúng tôi khảo sát ngắn gọn một vài cách tiếp cận, mỗi cách tiếp cận đều có những ưu điểm và nhược điểm riêng.

Một chiến lược đơn giản là gán máy khách cho cụm gần nhất về mặt địa lý. Sử dụng cơ sở dữ liệu vị trí địa lý thương mại (chẳng hạn như Quova [Quova 2020] và Max-Mind [MaxMind 2020]), mỗi địa chỉ IP LDNS được ánh xạ đến một vị trí địa lý. Khi nhận được yêu cầu DNS từ một LDNS cụ thể, CDN chọn cụm gần nhất về mặt địa lý, nghĩa là cụm cách LDNS ít km nhất "khi chim bay". Một giải pháp như vậy có thể hoạt động khá tốt cho một phần lớn các clients [Agarwal 2009]. Tuy nhiên, đối với một số máy khách, giải pháp có thể hoạt động kém, vì cụm gần nhất về mặt địa lý có thể không phải là cụm gần nhất về độ dài hoặc số bước nhảy của đường dẫn mạng. Hơn nữa, một vấn đề cố hữu với tất cả các phương pháp tiếp cận dựa trên

DNS là một số người dùng cuối được cấu hình để sử dụng LDNS định vị từ xa [Shaikh 2001; Mao 2002], trong trường hợp đó vị trí LDNS có thể cách xa vị trí của khách hàng. Hơn nữa, chiến lược đơn giản này bỏ qua sự thay đổi về độ trễ và độ rộng băng thông có sẵn theo thời gian của các đường dẫn Internet, luôn gán cùng một cụm cho một máy khách cụ thể. Để xác định cụm tốt nhất cho máy khách dựa trên điều kiện lưu lượng hiện tại, CDN có thể thực hiện **các phép đo thời gian thực** định kỳ về hiệu suất trễ và mất mát giữa cụm và máy khách. Ví dụ, một CDN có thể yêu cầu mỗi cụm của nó định kỳ gửi đầu dò (ví dụ: tin nhắn ping hoặc truy vấn DNS) đến tất cả các LDNS trên toàn thế giới. Một nhược điểm của phương pháp này

là nhiều LDNS được cấu hình để không đáp ứng với các đầu dò như vậy.

2.6.4 Nghiên cứu điển hình: Netflix và YouTube

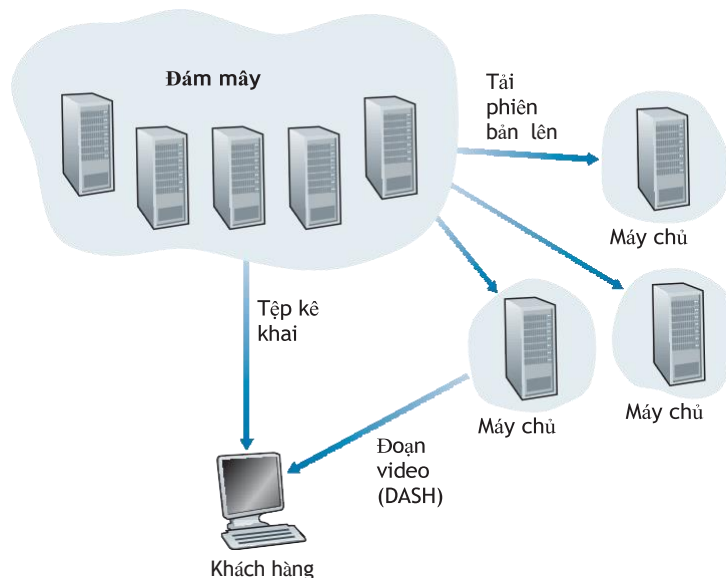
Chúng tôi kết thúc cuộc thảo luận về phát trực tuyến video được lưu trữ bằng cách xem xét hai triển khai quy mô lớn rất thành công: Netflix và YouTube. Chúng ta sẽ thấy rằng mỗi hệ thống này có một cách tiếp cận rất khác nhau, nhưng sử dụng nhiều nguyên tắc cơ bản được thảo luận trong phần này.

Netflix

Tính đến năm 2020, Netflix là nhà cung cấp dịch vụ hàng đầu cho các bộ phim và phim truyền hình trực tuyến ở Bắc Mỹ. Như chúng ta thảo luận bên dưới, phân phối video Netflix có hai cấu trúc chính: đám mây Amazon và cơ sở hạ tầng CDN riêng của nó.

Netflix có một trang web xử lý nhiều chức năng, bao gồm đăng ký và đăng nhập người dùng, thanh toán, danh mục phim để duyệt và tìm kiếm và hệ thống đề xuất phim. Như thể hiện trong Hình 2.26, trang web này (và các cơ sở dữ liệu phụ trợ liên quan của nó) chạy hoàn toàn trên các máy chủ Amazon trong đám mây Amazon. Ngoài ra, đám mây Amazon xử lý các chức năng quan trọng sau:

- **Nhập nội dung.** Trước khi Netflix có thể phân phối phim cho khách hàng của mình, trước tiên họ phải nhập và xử lý phim. Netflix nhận các phiên bản phim chính của studio và tải chúng lên các máy chủ lưu trữ trên đám mây Amazon.
- **Xử lý nội dung.** Các máy trong đám mây Amazon tạo ra nhiều định dạng khác nhau cho mỗi bộ phim, phù hợp với một loạt các trình phát video khách hàng chạy trên máy tính để bàn, điện thoại thông minh và bảng điều khiển trò chơi được kết nối với televi-sion. Một phiên bản khác nhau được tạo cho mỗi định dạng này và ở nhiều tốc độ bit, cho phép phát trực tuyến thích ứng qua HTTP bằng DASH.
- **Tải các phiên bản lên CDN của nó.** Khi tất cả các phiên bản của phim đã được tạo, các máy chủ lưu trữ trên đám mây Amazon tải các phiên bản lên CDN của nó.



Hình 2.26 ♦ Nền tảng phát trực tuyến video Netflix

Khi Netflix lần đầu tiên tung ra dịch vụ phát trực tuyến video vào năm 2007, họ đã thuê ba công ty CDN bên thứ ba để phân phối nội dung video của mình. Netflix kể từ đó đã tạo CDN riêng của mình, từ đó nó hiện phát trực tuyến tất cả các video của mình. Để tạo CDN của riêng mình, Netflix đã cài đặt giá đỡ máy chủ cả trong IXP và trong chính ISP cư trú. Netflix hiện có giá đỡ máy chủ tại hơn 200 địa điểm IXP; xem [Bottger 2018] [Netflix Open Connect 2020] để biết danh sách IXP hiện tại chứa giá đỡ Netflix. Ngoài ra còn có hàng trăm địa điểm ISP chứa giá đỡ Netflix; cũng xem [Netflix Open Connect 2020], nơi Netflix cung cấp cho các đối tác ISP tiềm năng hướng dẫn về cách cài đặt giá đỡ Netflix (miễn phí) cho mạng của họ. Mỗi máy chủ trong rack có một số cổng Ethernet 10 Gbps và hơn 100 terabyte dung lượng lưu trữ. Số lượng máy chủ trong một rack khác nhau: cài đặt IXP thường có hàng chục máy chủ và chứa toàn bộ thư viện video phát trực tuyến Netflix, bao gồm nhiều phiên bản video để hỗ trợ DASH. Netflix không sử dụng pull-caching (Mục 2.2.5) để điền các máy chủ CDN của mình vào IXP và ISP. Thay vào đó, Netflix phân phối bằng cách đẩy video đến các máy chủ CDN của mình trong giờ thấp điểm. Đối với những địa điểm không thể chứa toàn bộ thư viện, Netflix chỉ đẩy các video phổ biến nhất, được xác định hàng ngày. Thiết kế CDN Netflix được mô tả chi tiết trong các video YouTube [Netflix Video 1] và [Netflix Video 2]; xem thêm [Bottger 2018].

Sau khi mô tả các thành phần của kiến trúc Netflix, chúng ta hãy xem xét kỹ hơn sự tương tác giữa máy khách và các máy chủ khác nhau có liên quan đến việc phân phối phim. Như đã chỉ ra trước đó, các trang Web để duyệt thư viện video Netflix được phục vụ từ các máy chủ trong đám mây Amazon. Khi người dùng chọn một bộ phim để phát, phần mềm Netflix, chạy trên đám mây Amazon, trước tiên sẽ xác định máy chủ CDN nào có bản sao của phim. Trong số các máy chủ có phim, phần mềm sau đó xác định máy chủ "tốt nhất" cho yêu cầu của khách hàng đó. Nếu máy khách đang sử dụng ISP dân dụng có lắp giá đỡ máy chủ CDN Netflix trong ISP đó và giá đỡ này có bản sao phim được yêu cầu, thì máy chủ trong giá đỡ này thường được chọn. Nếu không, một máy chủ tại IXP gần đó thường được chọn.

Khi Netflix xác định máy chủ CDN sẽ phân phối nội dung, nó sẽ gửi cho máy khách địa chỉ IP của máy chủ cụ thể cũng như tệp kê khai, có URL cho các phiên bản khác nhau của phim được yêu cầu. Máy khách và máy chủ CDN đó sau đó tương tác trực tiếp bằng cách sử dụng phiên bản độc quyền của DASH. Cụ thể, như được mô tả trong Phần 2.6.2, máy khách sử dụng tiêu đề phạm vi byte trong thông báo yêu cầu HTTP GET, để yêu cầu các đoạn từ các phiên bản khác nhau của phim. Netflix sử dụng các đoạn dài khoảng bốn giây [Adhikari 2012]. Trong khi các đoạn đang được tải xuống, máy khách đo thông lượng nhận được và chạy thuật toán xác định tốc độ để xác định chất lượng của đoạn tiếp theo cần yêu cầu. Netflix thể hiện nhiều nguyên tắc chính được thảo luận trước đó trong phần này, bao gồm phát trực tuyến thích ứng và phân phối CDN. Tuy nhiên, vì Netflix sử dụng CDN riêng của mình, chỉ phân phối video (chứ không phải các trang web), Netflix đã có thể đơn giản hóa và điều chỉnh thiết kế CDN của mình. Cụ thể, Netflix không cần sử dụng chuyển hướng DNS, như đã thảo luận trong Phần 2.6.3, để kết nối một máy khách cụ thể với máy chủ CDN; thay vào đó, phần mềm Netflix (chạy trên đám mây Amazon) trực tiếp thông báo

máy khách để sử dụng một máy chủ CDN cụ thể. Hơn nữa, Netflix CDN sử dụng bộ nhớ đệm đẩy thay vì kéo bộ nhớ đệm (Phần 2.2.5): nội dung được đẩy vào máy chủ vào thời gian đã lên lịch vào giờ thấp điểm, thay vì động trong thời gian bộ nhớ cache bị nhớ.

YouTube

Với hàng trăm giờ video được tải lên YouTube mỗi phút và vài tỷ lượt xem video mỗi ngày, YouTube không thể chối cãi là trang web chia sẻ video lớn nhất thế giới. YouTube bắt đầu dịch vụ vào tháng 4 năm 2005 và được Google mua lại vào tháng 11 năm 2006. Mặc dù thiết kế và giao thức của Google / YouTube là ưu tiên, thông qua một số nỗ lực đo lường độc lập, chúng tôi có thể hiểu cơ bản về cách YouTube hoạt động [Zink 2009; Torres 2011; Adhikari 2011a]. Cũng như Netflix, YouTube sử dụng rộng rãi công nghệ CDN để phân phối video của mình [Torres 2011]. Tương tự như Netflix, Google sử dụng CDN riêng để phân phối video YouTube và đã cài đặt các cụm máy chủ ở hàng trăm vị trí IXP và ISP khác nhau. Từ những địa điểm này và trực tiếp từ các trung tâm dữ liệu khổng lồ của mình, Google phân phối video YouTube [Adhikari 2011a]. Tuy nhiên, không giống như Netflix, Google sử dụng bộ nhớ đệm kéo, như được mô tả trong Phần 2.2.5 và chuyển hướng DNS, như được mô tả trong Phần 2.6.3. Hầu hết thời gian, chiến lược lựa chọn cụm của Google hướng máy khách đến cụm mà RTT giữa máy khách và cụm là thấp nhất; tuy nhiên, để cân bằng tải giữa các cụm, đôi khi máy khách được hướng (thông qua DNS) đến một cụm xa hơn [Torres 2011].

YouTube sử dụng phát trực tuyến HTTP, thường cung cấp một số lượng nhỏ các phiên bản khác nhau cho video, mỗi phiên bản có tốc độ bit khác nhau và mức chất lượng tương ứng. YouTube không sử dụng tính năng phát trực tuyến thích ứng (chẳng hạn như DASH), mà thay vào đó yêu cầu người dùng chọn phiên bản theo cách thủ công. Để tiết kiệm băng thông và tài nguyên máy chủ có thể bị lãng phí khi định vị lại hoặc chấm dứt sớm, YouTube sử dụng yêu cầu phạm vi byte HTTP để giới hạn luồng dữ liệu được truyền sau khi tìm nạp trước lượng video mục tiêu.

Vài triệu video được tải lên YouTube mỗi ngày. Không chỉ các video YouTube được truyền từ máy chủ này sang máy khách khác qua HTTP mà người tải lên YouTube còn tải video của họ từ máy khách này sang máy chủ khác qua HTTP. YouTube xử lý từng video mà nó nhận được, chuyển đổi nó sang định dạng video YouTube và tạo nhiều phiên bản ở các tốc độ bit khác nhau. Quá trình xử lý này diễn ra hoàn toàn trong các trung tâm dữ liệu của Google. (Xem nghiên cứu điển hình về cơ sở hạ tầng mạng của Google trong Phần 2.6.3.)

2.7 Lập trình ổ cắm: Tạo ứng dụng mạng

Bây giờ chúng ta đã xem xét một số ứng dụng mạng quan trọng, hãy khám phá cách các chương trình ứng dụng mạng thực sự được tạo ra. Hãy nhớ lại từ Phần 2.1 rằng một ứng dụng mạng điển hình bao gồm một cặp chương trình — một chương trình máy khách và

Một chương trình máy chủ — nằm trong hai hệ thống đầu cuối khác nhau. Khi hai chương trình này được thực thi, một tiến trình máy khách và một quy trình máy chủ được tạo ra và các quy trình này giao tiếp với nhau bằng cách đọc từ và ghi vào ổ cắm. Do đó, khi tạo một ứng dụng mạng, nhiệm vụ chính của nhà phát triển là viết mã cho cả chương trình máy khách và máy chủ.

Có hai loại ứng dụng mạng. Một loại là một triển khai có hoạt động được chỉ định trong một tiêu chuẩn giao thức, chẳng hạn như RFC hoặc một số tài liệu tiêu chuẩn khác; Một ứng dụng như vậy đôi khi được gọi là "mở", vì các quy tắc chỉ định hoạt động của nó được biết đến với tất cả mọi người. Để thực hiện như vậy, các chương trình máy khách và máy chủ phải tuân thủ các quy tắc do RFC quy định. Đối với exam-ple, chương trình máy khách có thể là một triển khai phía máy khách của giao thức HTTP, được mô tả trong Phần 2.2 và được xác định chính xác trong RFC 2616; tương tự, chương trình máy chủ có thể là một triển khai của giao thức máy chủ HTTP, cũng được xác định chính xác trong RFC 2616. Nếu một nhà phát triển viết mã cho chương trình máy khách và một nhà phát triển khác viết mã cho chương trình máy chủ và cả hai nhà phát triển đều tuân thủ đầy đủ các quy tắc của RFC, thì hai chương trình sẽ có thể tương tác. Thật vậy, nhiều ứng dụng mạng ngày nay liên quan đến giao tiếp giữa các chương trình máy khách và máy chủ được tạo bởi các nhà phát triển độc lập, ví dụ: trình duyệt Google Chrome giao tiếp với máy chủ Web Apache hoặc máy khách BitTorrent giao tiếp với trình theo dõi BitTorrent.

Loại ứng dụng mạng khác là ứng dụng mạng độc quyền. Trong trường hợp này, các chương trình máy khách và máy chủ sử dụng giao thức lớp ứng dụng chưa được xuất bản công khai trong RFC hoặc ở nơi khác. Một nhà phát triển duy nhất (hoặc nhóm phát triển) tạo ra cả chương trình máy khách và máy chủ và nhà phát triển có toàn quyền kiểm soát những gì có trong mã. Nhưng vì mã không thực hiện giao thức mở, các nhà phát triển độc lập khác sẽ không thể phát triển mã tương tác với ứng dụng.

Trong phần này, chúng ta sẽ xem xét các vấn đề chính trong việc phát triển ứng dụng máy khách-máy chủ và chúng ta sẽ "làm bản tay" bằng cách xem xét mã triển khai một ứng dụng máy khách-máy chủ rất đơn giản. Trong giai đoạn phát triển, một trong những quyết định đầu tiên mà nhà phát triển phải đưa ra là liệu ứng dụng sẽ chạy qua TCP hay qua UDP. Hãy nhớ lại rằng TCP là định hướng kết nối và cung cấp một kênh luồng byte đáng tin cậy thông qua đó dữ liệu chảy giữa hai hệ thống cuối. UDP không có kết nối và gửi các gói dữ liệu độc lập từ hệ thống đầu này sang hệ thống đầu kia mà không có bất kỳ đảm bảo nào về việc phân phối. Cũng nên nhớ rằng khi một chương trình máy khách hoặc máy chủ triển khai một proto-col được xác định bởi RFC, nó nên sử dụng số cổng nổi tiếng được liên kết với giao thức; Ngược lại, khi phát triển một ứng dụng độc quyền, nhà phát triển phải cẩn thận để tránh sử dụng các số cổng nổi tiếng như vậy. (Số cổng đã được thảo luận ngắn gọn trong Phần 2.1. Chúng được đề cập chi tiết hơn trong Chương 3.)

Chúng tôi giới thiệu lập trình ổ cắm UDP và TCP bằng ứng dụng UDP đơn giản và ứng dụng TCP đơn giản. Chúng tôi trình bày các ứng dụng UDP và TCP đơn giản trong Python 3. Chúng tôi có thể đã viết mã bằng Java, C hoặc C++, nhưng chúng tôi chọn Python chủ yếu vì Python thể hiện rõ ràng các khái niệm ổ cắm khóa. Với

Python có ít dòng mã hơn và mỗi dòng có thể được giải thích cho lập trình viên mới làm quen mà không gặp khó khăn. Nhưng không cần phải sợ hãi nếu bạn không quen thuộc với Python. Bạn sẽ có thể dễ dàng theo dõi mã nếu bạn có lập trình experience trong Java, C hoặc C++.

Nếu bạn quan tâm đến lập trình máy khách-máy chủ với Java, bạn được khuyến khích xem Trang web đồng hành cho sách giáo khoa này; trên thực tế, bạn có thể tìm thấy ở đó tất cả các ví dụ trong phần này (và các phòng thí nghiệm liên quan) trong Java. Đối với những độc giả quan tâm đến lập trình máy khách-máy chủ trong C, có một số tài liệu tham khảo tốt có sẵn [Donahoo 2001; Stevens 1997; Frost 1994]; các ví dụ Python của chúng tôi dưới đây có giao diện tương tự như C.

2.7.1 Lập trình ổ cắm với UDP

Trong tiểu mục này, chúng ta sẽ viết các chương trình client-server đơn giản sử dụng UDP; trong phần sau, chúng ta sẽ viết các chương trình tương tự sử dụng TCP.

Nhớ lại từ Phần 2.1 rằng các quy trình chạy trên các máy khác nhau giao tiếp với nhau bằng cách gửi tin nhắn vào ổ cắm. Chúng tôi đã nói rằng mỗi quá trình tương tự như một ngôi nhà và ổ cắm của quy trình tương tự như một cánh cửa. Ứng dụng nằm ở một bên cửa trong nhà; Giao thức lớp vận chuyển nằm ở phía bên kia của cánh cửa ở thế giới bên ngoài. Nhà phát triển ứng dụng có quyền kiểm soát mọi thứ ở phía lớp ứng dụng của ổ cắm; Tuy nhiên, nó có ít quyền kiểm soát phía lớp vận chuyển.

Bây giờ chúng ta hãy xem xét kỹ hơn sự tương tác giữa hai pro-cesses giao tiếp sử dụng ổ cắm UDP. Trước khi quá trình gửi có thể đẩy một gói dữ liệu ra khỏi cửa ổ cắm, khi sử dụng UDP, trước tiên nó phải đính kèm địa chỉ đích vào gói. Sau khi gói tin đi qua socket của người gửi, Internet sẽ sử dụng địa chỉ đích này để định tuyến gói tin qua Internet đến socket trong quá trình nhận. Khi gói tin đến ổ cắm nhận, quá trình nhận sẽ truy xuất gói thông qua ổ cắm, sau đó kiểm tra nội dung của gói và thực hiện hành động thích hợp.

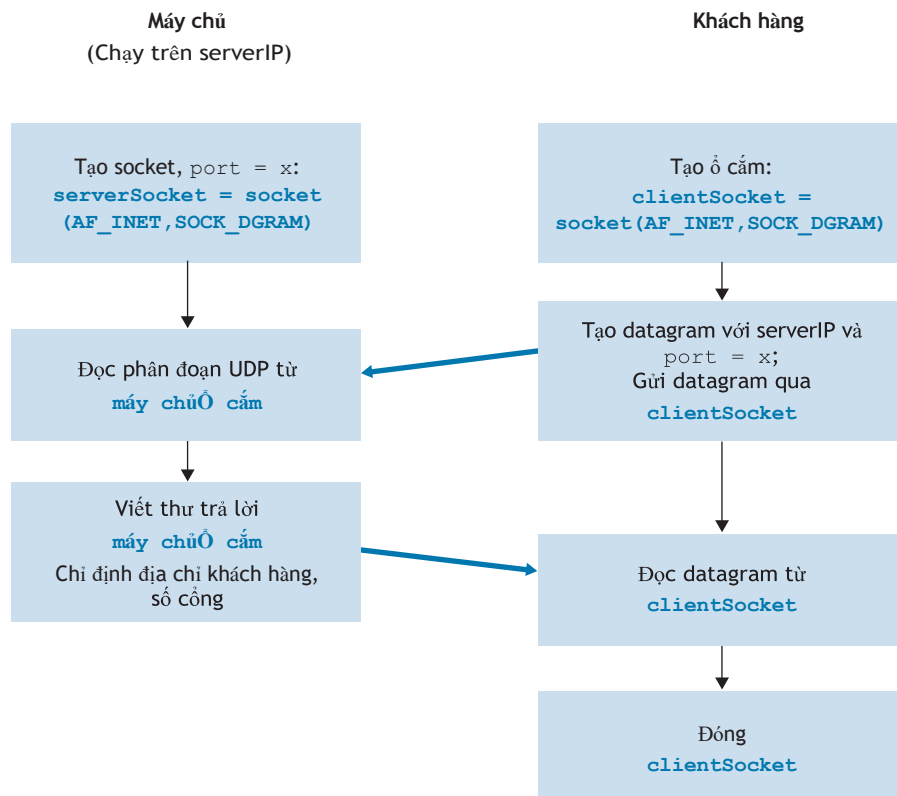
Vì vậy, bây giờ bạn có thể tự hỏi, những gì đi vào địa chỉ đích được đính kèm với gói? Như bạn có thể mong đợi, địa chỉ IP của máy chủ đích là một phần của địa chỉ đích. Bằng cách bao gồm địa chỉ IP đích trong gói, các bộ định tuyến trên Internet sẽ có thể định tuyến gói qua Internet đến máy chủ đích. Nhưng vì một máy chủ có thể đang chạy nhiều quy trình ứng dụng làm việc mạng, mỗi quy trình có một hoặc nhiều ổ cắm, nên cũng cần phải xác định ổ cắm cụ thể trong máy chủ đích. Khi một ổ cắm được tạo, một mã định danh, được gọi là **số cổng**, được gán cho nó. Vì vậy, như bạn có thể mong đợi, địa chỉ đích của gói cũng bao gồm số cổng của ổ cắm. Tóm lại, quá trình gửi gắn vào gói một địa chỉ đích, bao gồm địa chỉ IP của máy chủ đích và số cổng của ổ cắm đích. Hơn nữa, như chúng ta sẽ sớm thấy, địa chỉ nguồn của người gửi — bao gồm

Địa chỉ IP của máy chủ nguồn và số cổng của ổ cắm nguồn — cũng được đính kèm với gói. Tuy nhiên, việc đính kèm địa chỉ nguồn vào gói tin thường không được thực hiện bởi mã ứng dụng UDP; thay vào đó nó được tự động thực hiện bởi hệ điều hành cơ bản.

Chúng ta sẽ sử dụng ứng dụng client-server đơn giản sau đây để trình diễn lập trình socket cho cả UDP và TCP:

1. Máy khách đọc một dòng ký tự (dữ liệu) từ bàn phím của nó và gửi dữ liệu đến máy chủ.
2. Máy chủ nhận dữ liệu và chuyển đổi các ký tự thành chữ hoa.
3. Máy chủ gửi dữ liệu đã sửa đổi đến máy khách.
4. Máy khách nhận dữ liệu đã sửa đổi và hiển thị dòng trên màn hình của nó.

Hình 2.27 nêu bật hoạt động liên quan đến socket chính của máy khách và máy chủ giao tiếp qua dịch vụ truyền tải UDP.



Hình 2.27 ♦ Ứng dụng client-server sử dụng UDP

Bây giờ chúng ta hãy làm bản tay và xem xét cặp chương trình máy khách-máy chủ để triển khai UDP của ứng dụng đơn giản này. Chúng tôi cũng cung cấp một phân tích chi tiết, từng dòng sau mỗi chương trình. Chúng ta sẽ bắt đầu với UDP client, nó sẽ gửi một thông báo cấp ứng dụng đơn giản đến máy chủ. Để máy chủ có thể nhận và trả lời tin nhắn của máy khách, nó phải sẵn sàng và chạy—nghĩa là, nó phải chạy như một tiến trình trước khi máy khách gửi tin nhắn của nó.

Chương trình máy khách được gọi là `UDPClient.py` và chương trình máy chủ được gọi là `UDPServer.py`. Để nhấn mạnh các vấn đề chính, chúng tôi cố ý cung cấp mã tối thiểu. "Mã tốt" chắc chắn sẽ có thêm một vài dòng phụ trợ, đặc biệt là để xử lý các trường hợp lỗi. Đối với ứng dụng này, chúng tôi đã tùy ý chọn 12000 cho số cổng máy chủ.

UDPClient.py

Đây là code cho phía client của ứng dụng:

```
từ socket import *
serverName = 'hostname'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = input('Nhập câu viết thường:')
clientSocket.sendto(message.encode(), (serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(modifiedMessage.decode())
clientSocket.close()
```

Bây giờ chúng ta hãy xem xét các dòng mã khác nhau trong `UDPClient.py`.

từ nhập ổ cắm *

Mô-đun ổ cắm tạo thành cơ sở của tất cả các giao tiếp mạng bằng Python. Bằng cách bao gồm dòng này, chúng tôi sẽ có thể tạo các ổ cắm trong chương trình của chúng tôi.

```
serverName = 'tên máy chủ'
serverPort = 12000
```

Dòng đầu tiên đặt biến `serverName` thành chuỗi 'hostname'. Ở đây, chúng tôi cung cấp một chuỗi chứa địa chỉ IP của máy chủ (ví dụ: "128.138.32.126") hoặc tên máy chủ của máy chủ (ví dụ: "cis.poly.edu"). Nếu chúng tôi sử dụng tên máy chủ, thì tra cứu DNS sẽ tự động được thực hiện để lấy địa chỉ IP.) Dòng thứ hai đặt biến số nguyên `serverPort` thành 12000.

```
clientSocket = socket(AF_INET, SOCK_DGRAM)
```

Dòng này tạo socket của client, được gọi là `clientSocket`. Parameter đầu tiên chỉ ra họ địa chỉ; đặc biệt, `AF_INET` chỉ ra rằng mạng cơ sở đang sử dụng IPv4. (Đừng lo lắng về điều này ngay bây giờ — chúng tôi sẽ thảo luận về IPv4 trong Chương 4.) Tham số thứ hai chỉ ra rằng ổ cắm thuộc loại `SOCK_DGRAM`, có nghĩa là nó là ổ cắm UDP (chứ không phải ổ cắm TCP). Lưu ý rằng chúng tôi không chỉ định số cổng của ổ cắm máy khách khi chúng tôi tạo nó; Thay vào đó, chúng tôi để hệ điều hành làm điều này cho chúng tôi. Bây giờ cánh cửa của quy trình khách hàng đã được tạo, chúng tôi sẽ muốn tạo một tin nhắn để gửi qua cửa.

```
message = input('Nhập câu viết thường:')
```

`input()` là một hàm tích hợp sẵn trong Python. Khi lệnh này được thực thi, người dùng tại máy khách được nhắc với các từ "Nhập câu viết thường:" Sau đó, người dùng sử dụng bàn phím của mình để nhập một dòng, được đưa vào thông báo biến. Bây giờ chúng ta có một socket và một tin nhắn, chúng ta sẽ muốn gửi tin nhắn qua socket đến máy chủ đích.

```
clientSocket.sendto(message.encode(), (serverName, serverPort))
```

Trong dòng trên, trước tiên chúng ta chuyển đổi thông điệp từ kiểu chuỗi sang kiểu byte, vì chúng ta cần gửi byte vào một socket; điều này được thực hiện với phương thức `encode()`. Phương thức `sendto()` đính kèm địa chỉ đích (`serverName, serverPort`) vào tin nhắn và gửi gói kết quả vào socket của tiến trình, `clientSocket`. (Như đã đề cập trước đó, địa chỉ nguồn cũng được đính kèm với gói, mặc dù điều này được thực hiện tự động thay vì rõ ràng bằng mã.) Gửi tin nhắn từ máy khách đến máy chủ thông qua ổ cắm UDP thật đơn giản! Sau khi gửi gói, máy khách chờ nhận dữ liệu từ máy chủ.

```
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
```

Với dòng trên, khi một gói tin đến từ Internet tại socket của máy khách, dữ liệu của gói được đưa vào biến `modifiedMessage` và địa chỉ nguồn của gói được đưa vào biến `serverAddress`. Biến `serverAddress` chứa cả địa chỉ IP của máy chủ và số cổng của máy chủ. Chương trình `UDPCliet` không thực sự cần thông tin địa chỉ máy chủ này, vì nó đã biết địa chỉ máy chủ ngay từ đầu; nhưng dòng Python này vẫn cung cấp địa chỉ máy chủ. Phương pháp `recvfrom` cũng lấy kích thước bộ đệm 2048 làm đầu vào. (Kích thước bộ đệm này hoạt động cho hầu hết các mục đích.)

```
print(modifiedMessage.decode())
```

Dòng này in ra `modifiedMessage` trên màn hình của người dùng, sau khi chuyển đổi `message` từ byte sang chuỗi. Nó phải là dòng ban đầu mà người dùng đã nhập, nhưng bây giờ được viết hoa.

```
clientSocket.close()
```

Dòng này đóng ổ cắm. Quá trình sau đó chấm dứt.

UDPServer.py

Bây giờ chúng ta hãy xem xét phía máy chủ của ứng dụng:

```
từ socket import *
serverPort = 12000
serverSocket = socket (AF_INET,
SOCK_DGRAM) serverSocket.bind (('',
serverPort)) print ("Máy chủ đã sẵn sàng
để nhận")
while True:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(),
địa chỉ khách hàng)
```

Lưu ý rằng phần đầu của `UDPServer` tương tự như `UDPClient`. Nó cũng nhập mô-đun ổ cắm, cũng đặt biến số nguyên `serverPort` thành 12000 và cũng tạo một ổ cắm loại `SOCK_DGRAM` (ổ cắm UDP). Dòng mã đầu tiên khác biệt đáng kể so với `UDPClient` là:

```
serverSocket.bind(('', serverPort))
```

Dòng trên liên kết (nghĩa là gán) số cổng 12000 với ổ cắm của máy chủ. Do đó, trong `UDPServer`, mã (được viết bởi nhà phát triển ứng dụng) được gán rõ ràng một số cổng cho ổ cắm. Theo cách này, khi bất kỳ ai gửi một gói tin đến cổng 12000 tại địa chỉ IP của máy chủ, gói đó sẽ được chuyển đến ổ cắm này. `UDPServer` sau đó đi vào một vòng lặp `while`; vòng lặp `while` sẽ cho phép `UDPServer` nhận và xử lý các gói tin từ máy khách vô thời hạn. Trong vòng lặp `while`, `UDPServer` chờ một gói tin đến.

```
message, clientAddress = serverSocket.recvfrom(2048)
```

Dòng mã này tương tự như những gì chúng ta đã thấy trong `UDPClient`. Khi một gói đến socket của máy chủ, dữ liệu của gói được đưa vào thông báo biến và

Địa chỉ nguồn của gói được đưa vào biến `clientAddress`. Biến `clientAddress` chứa cả địa chỉ IP của máy khách và số cổng của máy khách. Tại đây, `UDPServer` sẽ sử dụng thông tin địa chỉ này, vì nó cung cấp địa chỉ trả lại, tương tự như địa chỉ trả lại với thư bưu chính thông thường. Với thông tin địa chỉ nguồn này, máy chủ bây giờ biết nơi nó nên hướng trả lời.

```
modifiedMessage = message.decode().upper()
```

Dòng này là trái tim của ứng dụng đơn giản của chúng tôi. Nó lấy dòng được gửi bởi máy khách và sau khi chuyển đổi thông điệp thành một chuỗi, sử dụng phương thức `upper()` để viết hoa nó.

```
serverSocket.sendto(modifiedMessage.encode(), clientAddress)
```

Dòng cuối cùng này gắn địa chỉ của máy khách (địa chỉ IP và số cổng) vào tin nhắn viết hoa (sau khi chuyển đổi chuỗi thành byte) và gửi gói kết quả vào ổ cắm của máy chủ. (Như đã đề cập trước đó, địa chỉ máy chủ cũng được đính kèm với gói, mặc dù điều này được thực hiện tự động thay vì rõ ràng bằng mã.) Internet sau đó sẽ gửi gói tin đến địa chỉ máy khách này. Sau khi máy chủ gửi gói, nó vẫn ở trong vòng lặp `while`, chờ một gói UDP khác đến (từ bất kỳ máy khách nào chạy trên bất kỳ máy chủ nào).

Để kiểm tra cập chương trình, bạn chạy `UDPClient.py` trên một máy chủ và `UDPS-erver.py` trên một máy chủ khác. Đảm bảo bao gồm tên máy chủ hoặc địa chỉ IP thích hợp của máy chủ trong `UDPClient.py`. Tiếp theo, bạn thực thi `UDPServer.py`, chương trình máy chủ được biên dịch, trong máy chủ lưu trữ. Điều này tạo ra một quá trình trong máy chủ không hoạt động cho đến khi nó được liên lạc bởi một số khách hàng. Sau đó, bạn thực thi `UDPClient.py`, chương trình máy khách được biên dịch, trong máy khách. Điều này tạo ra một quá trình trong máy khách. Cuối cùng, để sử dụng appli- cation tại client, bạn gõ một câu theo sau là một carriage return.

Để phát triển ứng dụng máy khách-máy chủ UDP của riêng bạn, bạn có thể bắt đầu bằng cách sửa đổi một chút các chương trình máy khách hoặc máy chủ. Ví dụ: thay vì chuyển đổi tất cả các chữ cái thành chữ hoa, máy chủ có thể đếm số lần các chữ cái `s` xuất hiện và trả về số này. Hoặc bạn có thể sửa đổi máy khách để sau khi nhận được câu viết hoa, người dùng có thể tiếp tục gửi thêm câu đến máy chủ.

2.7.2 Lập trình ổ cắm với TCP

Không giống như UDP, TCP là một giao thức hướng kết nối. Điều này có nghĩa là trước khi cli-ent và server có thể bắt đầu gửi dữ liệu cho nhau, trước tiên chúng cần bắt tay và thiết lập kết nối TCP. Một đầu của kết nối TCP được gắn vào ổ cắm máy khách và đầu kia được gắn vào ổ cắm máy chủ. Khi tạo TCP con- nection, chúng tôi liên kết với nó địa chỉ socket máy khách (địa chỉ IP và số cổng) và địa chỉ socket máy chủ (địa chỉ IP và số cổng). Với sự kết nối TCP được thiết lập, khi một bên muốn gửi dữ liệu sang phía bên kia, nó chỉ cần bỏ

dữ liệu vào kết nối TCP thông qua socket của nó. Điều này khác với UDP, trong đó máy chủ phải đính kèm địa chỉ đích vào gói trước khi thả nó vào ổ cắm.

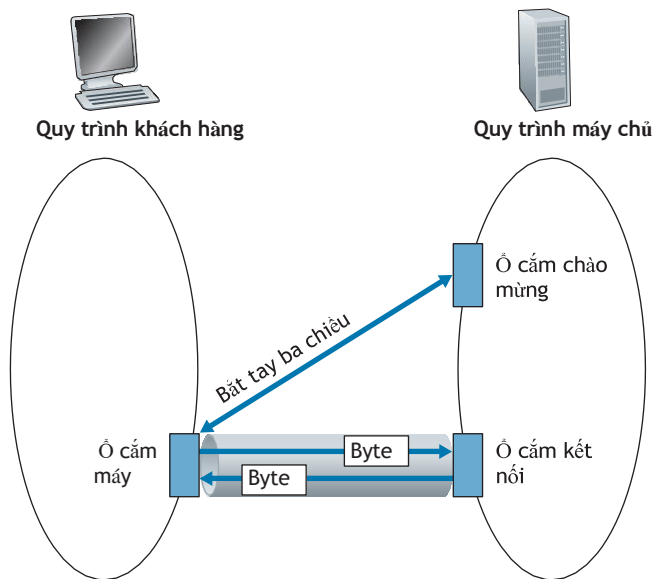
Bây giờ chúng ta hãy xem xét kỹ hơn về sự tương tác của các chương trình máy khách và máy chủ trong TCP. Máy khách có công việc bắt đầu liên lạc với máy chủ. Để máy chủ có thể phản ứng với liên hệ ban đầu của khách hàng, máy chủ phải sẵn sàng. Điều này ngụ ý hai điều. Đầu tiên, như trong trường hợp UDP, máy chủ TCP phải chạy như một tiến trình trước khi máy khách cố gắng bắt đầu liên hệ. Thứ hai, chương trình máy chủ phải có một cánh cửa đặc biệt - chính xác hơn là một ổ cắm đặc biệt - chào đón một số liên hệ ban đầu từ một quy trình máy khách chạy trên một máy chủ tùy ý. Sử dụng sự tương tự nhà / cửa của chúng tôi cho một quy trình / ổ cắm, đôi khi chúng tôi sẽ đề cập đến chiến thuật ban đầu của khách hàng là "gõ cửa chào đón".

Với quá trình máy chủ đang chạy, tiến trình máy khách có thể bắt đầu kết nối TCP với máy chủ. Điều này được thực hiện trong chương trình máy khách bằng cách tạo ổ cắm TCP. Khi máy khách tạo ổ cắm TCP của nó, nó chỉ định địa chỉ của ổ cắm chào mừng trong máy chủ, cụ thể là địa chỉ IP của máy chủ lưu trữ và số cổng của ổ cắm. Sau khi tạo socket, máy khách bắt đầu bắt tay ba chiều và thiết lập kết nối TCP với máy chủ. Bắt tay ba chiều, diễn ra trong lớp vận chuyển, hoàn toàn vô hình đối với các chương trình máy khách và máy chủ.

Trong quá trình bắt tay ba chiều, quy trình khách hàng gõ vào cánh cửa chào đón của quá trình máy chủ. Khi máy chủ "nghe thấy" tiếng gõ, nó sẽ tạo ra một cánh cửa mới — chính xác hơn là một ổ cắm mới dành riêng cho máy khách cụ thể đó. Trong ví dụ dưới đây của chúng tôi, cửa chào đón là một đối tượng TCP socket mà chúng tôi gọi là `serverSocket`; socket mới được tạo dành riêng cho máy khách thực hiện con-nection được gọi là `connectionSocket`. Sinh viên lần đầu tiên gặp phải với TCP đôi khi nhầm lẫn giữa ổ cắm chào mừng (là điểm liên lạc ban đầu cho tất cả các máy khách muốn giao tiếp với máy chủ) và mỗi ổ cắm kết nối phía máy chủ mới được tạo sau đó được tạo để giao tiếp với từng máy khách.

Từ quan điểm của ứng dụng, socket của client và con-nection socket của máy chủ được kết nối trực tiếp bằng một pipe. Như thể hiện trong Hình 2.28, quá trình cli-ent có thể gửi các byte tùy ý vào socket của nó và TCP đảm bảo rằng quá trình máy chủ sẽ nhận (thông qua socket kết nối) mỗi byte theo thứ tự được gửi. Do đó, TCP cung cấp một dịch vụ đáng tin cậy giữa các quy trình máy khách và máy chủ. Hơn nữa, giống như mọi người có thể ra vào cùng một cửa, quá trình khách hàng không chỉ gửi byte vào mà còn nhận byte từ ổ cắm của nó; Tương tự, quá trình máy chủ không chỉ nhận byte từ mà còn gửi byte vào ổ cắm kết nối của nó.

Chúng tôi sử dụng cùng một ứng dụng client-server đơn giản để chứng minh socket programming với TCP: Máy khách gửi một dòng dữ liệu đến máy chủ, máy chủ viết hoa dòng và gửi lại cho máy khách. Hình 2.29 nêu bật hoạt động liên quan đến socket chính của client và server giao tiếp qua TCP transport service.



Hình 2.28 ♦ Quá trình TCPServer có hai socket

TCPClient.py

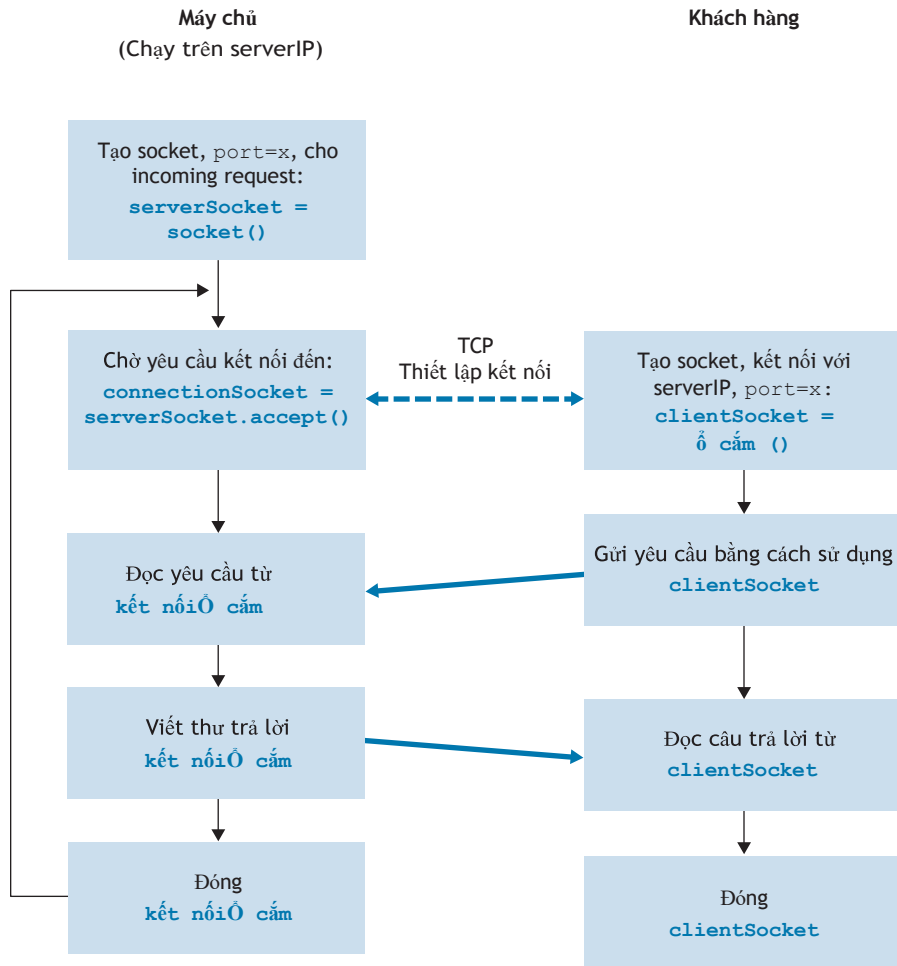
Đây là code cho phía client của ứng dụng:

```
từ socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
câu = input('Đầu vào câu viết thường:')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print('From Server: ', modifiedSentence.decode())
clientSocket.close()
```

Bây giờ chúng ta hãy xem xét các dòng khác nhau trong mã khác biệt đáng kể so với việc triển khai UDP. Dòng đầu tiên như vậy là việc tạo ra ổ cắm máy khách.

```
clientSocket = socket(AF_INET, SOCK_STREAM)
```

Dòng này tạo socket của client, được gọi là `clientSocket`. Tham số đầu tiên một lần nữa chỉ ra rằng mạng cơ bản đang sử dụng IPv4. Tham số thứ hai

**Hình 2.29** ♦ Ứng dụng client-server sử dụng TCP

chỉ ra rằng ổ cắm thuộc loại `SOCK_STREAM`, có nghĩa là nó là ổ cắm TCP (chứ không phải ổ cắm UDP). Lưu ý rằng chúng tôi một lần nữa không chỉ định số cổng của ổ cắm máy khách khi chúng tôi tạo nó; Thay vào đó, chúng tôi để hệ điều hành làm điều này cho chúng tôi. Bây giờ dòng mã tiếp theo rất khác so với những gì chúng ta đã thấy trong `UDPClient`:

```
clientSocket.connect ((serverName, serverPort))
```

Hãy nhớ lại rằng trước khi máy khách có thể gửi dữ liệu đến máy chủ (hoặc ngược lại) bằng ổ cắm TCP, trước tiên kết nối TCP phải được thiết lập giữa máy khách và máy chủ. Các

dòng trên khởi tạo kết nối TCP giữa máy khách và máy chủ. Tham số của phương thức `connect()` là địa chỉ của phía máy chủ của kết nối. Sau khi dòng mã này được thực thi, bắt tay ba chiều được thực hiện và sự kết hợp TCP được thiết lập giữa máy khách và máy chủ.

```
sentence = input('Nhập câu viết thường:')
```

Như với `UDPClient`, ở trên có được một câu từ người dùng. Câu chuỗi tiếp tục thu thập các ký tự cho đến khi người dùng kết thúc dòng bằng cách nhập ký tự xuống dòng. Dòng mã tiếp theo cũng rất khác với `UDPClient`:

```
clientSocket.send(sentence.encode())
```

Dòng trên gửi câu qua socket của client và vào kết nối TCP. Lưu ý rằng chương trình *không* tạo rõ ràng một gói và đính kèm địa chỉ đích vào gói, như trường hợp của ổ cắm UDP. Thay vào đó, chương trình cli-ent chỉ đơn giản là thả các byte trong câu chuỗi vào liên kết TCP. Máy khách sau đó chờ đợi để nhận byte từ máy chủ.

```
modifiedSentence = clientSocket.recv(2048)
```

Khi các ký tự đến từ máy chủ, chúng được đặt vào chuỗi `modifiedSentence`. Các nhân vật tiếp tục tích lũy trong sửa đổi `modifiedSentence` cho đến khi dòng kết thúc bằng một nhân vật trả về xe ngựa. Sau khi in câu viết hoa, chúng tôi đóng ổ cắm của khách hàng:

```
clientSocket.close()
```

Dòng cuối cùng này đóng ổ cắm và do đó, đóng kết nối TCP giữa máy khách và máy chủ. Nó khiến TCP trong máy khách gửi thông báo TCP đến TCP trong máy chủ (xem Phần 3.5).

TCPServer.py

Bây giờ chúng ta hãy xem xét chương trình máy chủ.

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
print('Máy chủ đã sẵn sàng để nhận')
```

trong khi True:

```
connectionSocket, addr = serverSocket.accept()
câu = connectionSocket.recv(1024).decode()
capitalizedSentence = sentence.upper()
connectionSocket.send(capitalizedSentence.encode(
)) connectionSocket.close()
```

Bây giờ chúng ta hãy xem xét các dòng khác biệt đáng kể so với UDPServer và TCP-Client. Như với TCPClient, máy chủ tạo một TCP socket với:

```
serverSocket = socket (AF_INET, SOCK_STREAM)
```

Tương tự như UDPServer, chúng ta liên kết số cổng máy chủ, serverPort, với socket này:

```
serverSocket.bind(('', serverPort))
```

Nhưng với TCP, serverSocket sẽ là socket chào đón của chúng tôi. Sau khi thiết lập cánh cửa chào đón này, chúng tôi sẽ chờ đợi và lắng nghe một số khách hàng gõ cửa:

```
máy chủSocket.listen(1)
```

Dòng này có máy chủ lắng nghe các yêu cầu kết nối TCP từ máy khách. Tham số chỉ định số lượng kết nối hàng đợi tối đa (ít nhất là 1).

```
connectionSocket, addr = serverSocket.accept()
```

Khi một máy khách gõ vào cánh cửa này, chương trình sẽ gọi phương thức accept() cho serverSocket, tạo ra một socket mới trong máy chủ, được gọi là connectionSocket, dành riêng cho máy khách cụ thể này. Máy khách và máy chủ sau đó hoàn thành việc bắt tay, tạo kết nối TCP giữa clientSocket của client và connectionSocket của máy chủ. Với kết nối TCP được thiết lập, máy khách và máy chủ giờ đây có thể gửi byte cho nhau qua kết nối. Với TCP, tất cả các byte được gửi từ một phía chỉ được đảm bảo đến phía bên kia nhưng cũng được đảm bảo đến theo thứ tự.

```
connectionSocket.close()
```

Trong chương trình này, sau khi gửi câu đã sửa đổi cho máy khách, chúng tôi đóng ổ cắm con-nection. Nhưng vì serverSocket vẫn mở, một máy khách khác bây giờ có thể gõ cửa và gửi cho máy chủ một câu để sửa đổi.

Điều này hoàn thành cuộc thảo luận của chúng tôi về lập trình socket trong TCP. Bạn được khuyến khích chạy hai chương trình trong hai máy chủ riêng biệt và cũng để sửa đổi chúng để đạt được các mục tiêu hơi khác nhau. Bạn nên so sánh cặp chương trình UDP với cặp chương trình TCP và xem chúng khác nhau như thế nào. Bạn cũng nên thực hiện nhiều bài tập lập trình socket được mô tả ở cuối Chương 2 và 5. Cuối cùng, chúng tôi hy vọng một ngày nào đó, sau khi thành thạo các chương trình ở cấp này và nâng cao hơn, bạn sẽ viết ứng dụng mạng phổ biến của riêng mình, trở nên rất giàu có và nổi tiếng, và nhớ đến các tác giả của cuốn sách giáo khoa này!

2.8 Tóm tắt

Trong chương này, chúng ta đã nghiên cứu các khía cạnh khái niệm và triển khai của các ứng dụng mạng. Chúng tôi đã tìm hiểu về kiến trúc máy khách-máy chủ phổ biến được nhiều ứng dụng Internet áp dụng và thấy việc sử dụng nó trong các giao thức HTTP, SMTP và DNS. Chúng tôi đã nghiên cứu các giao thức cấp ứng dụng quan trọng này và các ứng dụng liên quan tương ứng của chúng (Web, truyền tệp, e-mail và DNS) một cách chi tiết. Chúng ta đã tìm hiểu về kiến trúc P2P và đối chiếu nó với kiến trúc client-server. Chúng tôi cũng đã tìm hiểu về phát trực tuyến video và cách các hệ thống phân phối video hiện đại tận dụng CDN. Chúng tôi đã kiểm tra cách API socket có thể được sử dụng để xây dựng các ứng dụng mạng. Chúng tôi đã hướng dẫn việc sử dụng ở cấp cho các dịch vụ vận chuyển end-to-end end-to-end (TCP) và end-less (UDP). Bước đầu tiên trong hành trình của chúng tôi xuống kiến trúc mạng nhiều lớp đã hoàn tất!

Ngay từ đầu cuốn sách này, trong Phần 1.1, chúng tôi đã đưa ra một định nghĩa khá mơ hồ, trần trụi về một giao thức: "định dạng và thứ tự của các thông điệp được trao đổi giữa hai hoặc nhiều thực thể liên lạc, cũng như các hành động được thực hiện khi truyền và / hoặc nhận tin nhắn hoặc sự kiện khác." Tài liệu trong chương này, và đặc biệt là nghiên cứu chi tiết của chúng tôi về các giao thức HTTP, SMTP và DNS, hiện đã thêm chất đáng kể vào định nghĩa này. Giao thức là một khái niệm quan trọng trong mạng; Nghiên cứu của chúng tôi về các giao thức ứng dụng hiện đã cho chúng tôi cơ hội phát triển cảm giác trực quan hơn về tất cả các giao thức.

Trong Phần 2.1, chúng tôi đã mô tả các mô hình dịch vụ mà TCP và UDP cung cấp cho các ứng dụng gọi chúng. Chúng tôi đã xem xét kỹ hơn các mô hình dịch vụ này khi chúng tôi phát triển các ứng dụng đơn giản chạy trên TCP và UDP trong Phần 2.7. Tuy nhiên, chúng tôi đã nói rất ít về cách TCP và UDP cung cấp các mô hình dịch vụ này. Ví dụ: chúng tôi biết rằng TCP cung cấp một dịch vụ dữ liệu đáng tin cậy, nhưng chúng tôi chưa cho biết làm thế nào nó làm như vậy. Trong chương tiếp theo, chúng ta sẽ xem xét cẩn thận không chỉ những gì, mà còn cả cách thức và lý do của các giao thức vận tải.

Được trang bị kiến thức về cấu trúc ứng dụng Internet và các giao thức cấp ứng dụng, giờ đây chúng ta đã sẵn sàng đi sâu hơn vào ngăn xếp giao thức và kiểm tra lớp vận chuyển trong Chương 3.

Vấn đề bài tập về nhà và câu hỏi

Chương 2: Câu hỏi ôn tập

MỤC 2.1

- R1. Liệt kê năm ứng dụng Internet không độc quyền và các proto-cols lớp ứng dụng mà chúng sử dụng.
- R2. Sự khác biệt giữa kiến trúc mạng và kiến trúc ứng dụng là gì?
- Câu 3. Đối với một phiên giao tiếp giữa một cặp quy trình, tiến trình nào là máy khách và máy chủ là gì?
- R4. Tại sao các thuật ngữ máy khách và máy chủ vẫn được sử dụng trong các ứng dụng ngang hàng?
- R5. Thông tin nào được sử dụng bởi một tiến trình chạy trên một máy chủ để xác định một pro-cess đang chạy trên một máy chủ khác?
- Câu 6. Vai trò của HTTP trong một ứng dụng mạng là gì? Những thành phần nào khác là cần thiết để hoàn thành một ứng dụng Web?
- Câu 7. Đề cập đến Hình 2.4, chúng ta thấy rằng không có ứng dụng nào được liệt kê trong Hình 2.4 yêu cầu cả không mất dữ liệu và thời gian. Bạn có thể hình dung về một ứng dụng không yêu cầu mất dữ liệu và điều đó cũng rất nhạy cảm với thời gian không?
- R8. Liệt kê bốn loại dịch vụ lớn mà một giao thức vận tải có thể cung cấp. Đối với mỗi lớp dịch vụ, hãy cho biết liệu UDP hoặc TCP (hoặc cả hai) có cung cấp dịch vụ như vậy hay không.
- Câu 9. Hãy nhớ lại rằng TCP có thể được tăng cường với TLS để cung cấp các dịch vụ bảo mật từ quy trình đến xử lý, bao gồm cả mã hóa. TLS hoạt động ở lớp vận chuyển hay lớp ứng dụng? Nếu nhà phát triển ứng dụng muốn TCP được tăng cường với TLS, nhà phát triển phải làm gì?

PHẦN 2.2–2.5

- Câu 10. Giao thức bắt tay có nghĩa là gì?
- Câu 11. Giao thức không quốc tịch có nghĩa là gì? IMAP có phải là người không quốc tịch không? Còn SMTP thì sao? Câu 12. Làm thế nào các trang web có thể theo dõi người dùng? Họ có luôn cần sử dụng cookie không?
- Câu 13. Mô tả cách bộ nhớ đệm Web có thể làm giảm độ trễ trong việc nhận một đối tượng được yêu cầu. Bộ nhớ đệm Web sẽ làm giảm độ trễ cho tất cả các đối tượng được yêu cầu bởi người dùng hay chỉ cho một số đối tượng? Tại sao?
- Câu 14. Telnet vào một máy chủ Web và gửi một tin nhắn yêu cầu nhiều dòng. Bao gồm trong thông báo yêu cầu dòng tiêu đề If-modified-since: để buộc thông báo phản hồi với mã trạng thái 304 Not Modified .
- Câu 15. Có bất kỳ ràng buộc nào đối với định dạng của nội dung HTTP không?

Còn nội dung email được gửi bằng SMTP thì sao? Làm thế nào dữ liệu tùy ý có thể được truyền qua SMTP?

- Câu 16. Giả sử Alice, với một tài khoản e-mail dựa trên web (chẳng hạn như Hotmail hoặc Gmail), gửi một tin nhắn cho Bob, người truy cập thư của mình từ máy chủ thư của mình bằng IMAP. Thảo luận về cách thông điệp được chuyển từ chủ nhà của Alice đến chủ nhà của Bob. Đảm bảo liệt kê một loạt các giao thức lớp ứng dụng được sử dụng để di chuyển thư giữa hai máy chủ.
- Câu 17. In tiêu đề của thông điệp email bạn đã nhận được gần đây. Có bao nhiêu dòng tiêu đề `Received:` không? Phân tích từng dòng tiêu đề trong thư.
- Câu 18. Vấn đề chặn HOL trong HTTP / 1.1 là gì? HTTP / 2 cố gắng giải quyết nó như thế nào?
- R19. Tại sao cần bản ghi MX? Sử dụng bản ghi CNAME sẽ không đủ? (Giả sử ứng dụng email tra cứu địa chỉ email thông qua truy vấn Loại A và máy chủ đích chỉ chạy máy chủ email.)
- R20. Sự khác biệt giữa truy vấn DNS đệ quy và lặp lại là gì?

MỤC 2.5

- Câu 21. Trong trường hợp nào tải xuống tệp thông qua P2P nhanh hơn nhiều so với cách tiếp cận máy khách-máy chủ tập trung? Biện minh cho câu trả lời của bạn bằng Phương trình 2.2.
- Câu 22. Hãy xem xét một Alice ngang hàng mới tham gia BitTorrent mà không sở hữu bất kỳ khối nào. Nếu không có bất kỳ phần nào, cô ấy không thể trở thành người tải lên top bốn cho bất kỳ dòng nghiệp nào khác, vì cô ấy không có gì để tải lên. Làm thế nào sau đó Alice sẽ có được khối đầu tiên của mình?
- Câu 23. Giả sử trình theo dõi BitTorrent đột nhiên không khả dụng. Hậu quả của nó là gì? Các tệp vẫn có thể được tải xuống?

MỤC 2.6

- Câu 24. CDN thường áp dụng một trong hai triết lý vị trí máy chủ khác nhau. Đặt tên và mô tả ngắn gọn chúng.
- Câu 25. Bên cạnh những cân nhắc liên quan đến mạng như độ trễ, mất mát và hiệu suất băng thông, có những yếu tố quan trọng khác để thiết kế chiến lược lựa chọn máy chủ CDN. Chúng là gì?

MỤC 2.7

- Câu 26. Trong Phần 2.7, máy chủ UDP được mô tả chỉ cần một socket, trong khi máy chủ TCP cần hai socket. Tại sao? Nếu máy chủ TCP hỗ trợ n kết nối đồng thời, mỗi kết nối từ một máy chủ khách hàng khác nhau, máy chủ TCP sẽ cần bao nhiêu ổ cắm?

Câu 27. Đối với ứng dụng máy khách-máy chủ qua TCP được mô tả trong Phần 2.7, tại sao chương trình máy chủ phải được thực thi trước chương trình máy khách? Đối với ứng dụng máy khách-máy chủ qua UDP, tại sao chương trình máy khách có thể được thực thi trước chương trình máy chủ?

Vấn đề

P1. Đúng hay sai?

- Người dùng yêu cầu một trang Web bao gồm một số văn bản và ba hình ảnh. Đối với trang này, khách hàng sẽ gửi một tin nhắn yêu cầu và nhận được bốn tin nhắn phản hồi.
- Hai trang Web riêng biệt (ví dụ: `www.mit.edu/research.html` và `www.mit.edu/students.html`) có thể được gửi qua cùng một kết nối liên tục.
- Với các kết nối không liên tục giữa trình duyệt và máy chủ gốc, một phân đoạn TCP duy nhất có thể mang hai thông báo yêu cầu HTTP riêng biệt.
- Tiêu đề Ngày: trong thông báo phản hồi HTTP cho biết khi nào đối tượng trong phản hồi được sửa đổi lần cuối.
- Thông điệp phản hồi HTTP không bao giờ có nội dung thư trống.

P2. SMS, iMessage, Wechat và WhatsApp đều là các hệ thống mes-saging thời gian thực của điện thoại thông minh. Sau khi thực hiện một số nghiên cứu trên Internet, đối với mỗi hệ thống này, hãy viết một đoạn về các giao thức mà chúng sử dụng. Sau đó viết một đoạn văn giải thích chúng khác nhau như thế nào.

Trang 3. Giả sử bạn mở trình duyệt và nhập `http://yourbusiness.com/about.html` vào thanh địa chỉ. Điều gì xảy ra cho đến khi trang web được hiển thị? Cung cấp chi tiết về (các) giao thức được sử dụng và mô tả cấp cao về các tin nhắn được trao đổi.

Trang 4. Hãy xem xét chuỗi ký tự ASCII sau đây đã được Wireshark chụp khi trình duyệt gửi tin nhắn HTTP GET (nghĩa là đây là nội dung thực tế của thông báo HTTP GET). Các ký tự `<cr>` `<lf>` là các ký tự trả về vận chuyển và nguồn cấp dữ liệu dòng (nghĩa là chuỗi ký tự in nghiêng `<cr>` trong văn bản dưới đây đại diện cho ký tự trả về dòng duy nhất được chứa tại thời điểm đó trong tiêu đề HTTP). Trả lời các câu hỏi sau, cho biết nơi trong thông báo HTTP GET bên dưới bạn tìm thấy câu trả lời.

```
GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai
a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0
(Windows; U; Windows NT 5.1; en-US; rv: 1.7.2) Gec
ko / 20040804 Netscape / 7.2 (ax) <cr><lf>Chấp
nhận: ex t / xml, ứng dụng / xml, ứng dụng / xhtml
+ xml, văn bản
/html; q = 0,9, văn bản / đơn giản; q = 0,8, hình ảnh / png,
*/*; q = 0,5
```



```
<cr><lf>Accept-Language: en-us,en; q = 0,5 < cr >< lf > Chấp
nhận - Mã hóa: zip, deflate<cr><lf>Accept-Charset: ISO
-8859-1,UTF-8; q = 0,7,*; q = 0,7<cr><lf>Keep-Alive: 300<cr>
<lf>Kết nối: keep-alive<cr><lf><cr><lf>
```

- URL của tài liệu được trình duyệt yêu cầu là gì?
- Trình duyệt đang chạy phiên bản HTTP nào?
- Trình duyệt có yêu cầu kết nối không liên tục hoặc liên tục không?
- Địa chỉ IP của máy chủ lưu trữ mà trình duyệt đang chạy là gì?
- Loại trình duyệt nào khởi tạo thông báo này? Tại sao loại trình duyệt cần thiết trong thông báo yêu cầu HTTP?

P5. Văn bản dưới đây cho thấy câu trả lời được gửi từ máy chủ để phản hồi thông báo HTTP GET trong câu hỏi trên. Trả lời các câu hỏi sau đây, chỉ ra nơi trong thông báo dưới đây bạn tìm thấy câu trả lời.

```
HTTP / 1.1 200 OK<cr><lf>Ngày: Tue, 07 Mar 2008
12:39:45GMT<cr><lf>Máy chủ: Apache / 2.0.52
(Fedora)
<cr><lf>Sửa đổi lần cuối: Sat, 10 Dec2005 18:27:46
GMT<cr><lf>ETag: "526c3-f22-
a88a4c80"<cr><lf>Accept- Phạm vi:
byte<cr><lf>Content-Length: 3874<cr><lf> Keep-
Alive: timeout = max = 100<cr><lf>Kết nối:
Keep-Alive<cr><lf>Content-Type: văn bản / html;
charset = ISO-8859-1<cr><lf><cr><lf><!doctype html
public "-//
w3c//dtd html 4.0transitional//en"><lf><html><lf>
<head><lf> <meta http-equiv="Content-Type"
content="text/html; charset = iso-8859-1 "><lf> <meta
name = "GENERATOR" content = "Mozilla / 4.79 [en]
(Windows NT 5.0; U) Netscape]"><lf> <title>CMPSCI 453 /
591 /
NTU-ST550ASpring 2005 trang chủ</title><lf></head><lf>
<nhiều văn bản tài liệu hơn sau đây (không được hiển thị)>
```

- Máy chủ có thể tìm thấy tài liệu thành công hay không? Trả lời tài liệu được cung cấp lúc mấy giờ?
- Tài liệu được sửa đổi lần cuối khi nào?
- Có bao nhiêu byte trong tài liệu được trả về?
- 5 byte đầu tiên của tài liệu được trả về là gì? Máy chủ có đồng ý với kết nối liên tục không?

Trang 6. Lấy đặc tả HTTP / 1.1 (RFC 2616). Trả lời các câu hỏi sau:

- Giải thích cơ chế được sử dụng để báo hiệu giữa máy khách và máy chủ để chỉ ra rằng kết nối liên tục đang bị đóng. Máy khách, máy chủ hoặc cả hai có thể báo hiệu sự đóng kết nối không?

- b. HTTP cung cấp những dịch vụ mã hóa nào?
- c. Máy khách có thể mở ba hoặc nhiều kết nối đồng thời với một máy chủ nhất định không?
- d. Máy chủ hoặc máy khách có thể đóng kết nối truyền tải giữa chúng nếu một trong hai phát hiện kết nối không hoạt động trong một thời gian. Có thể một bên bắt đầu đóng kết nối trong khi bên kia đang truyền dữ liệu qua kết nối này không? Giải thích.

Trang 7. Giả sử trong trình duyệt Web của bạn, bạn nhấp vào một liên kết để có được một trang Web.

Địa chỉ IP cho URL được liên kết không được lưu vào bộ nhớ cache trong máy chủ cục bộ của bạn, do đó, cần phải tra cứu DNS để lấy địa chỉ IP.

Giả sử rằng n máy chủ DNS được truy cập trước khi máy chủ của bạn nhận được địa chỉ IP từ DNS; các lần truy cập liên tiếp phải chịu RTT RTT_1, \dots, RTT_n . Giả sử thêm rằng các

Trang web được liên kết với liên kết chứa chính xác một đối tượng, bao gồm một lượng lớn văn bản HTML. Cho RTT_0 biểu thị RTT giữa máy chủ cục bộ và máy chủ chứa đối tượng. Giả sử thời gian truyền là $0,002 * RTT_0$ của đối tượng, bao nhiêu thời gian trôi qua kể từ khi máy khách nhấp vào liên kết cho đến khi máy khách nhận được đối tượng?

Trang 8. Xem xét lại Bài toán P7 và giả sử $RTT_0 = RTT_1 = RTT_2 = \dots$

$RTT_n = RTT$, Hơn nữa, giả sử một tệp HTML mới, đủ nhỏ để có thời gian truyền không đáng kể, tham chiếu đến chín đối tượng nhỏ bằng nhau trên cùng một máy chủ. Bao nhiêu thời gian trôi qua với

- a. HTTP không liên tục không có kết nối TCP song song?
- b. HTTP không liên tục với trình duyệt được cấu hình cho 6 kết nối song song?
- c. HTTP dài đẳng?

Trang 9. Hãy xem xét Hình 2.12, trong đó có một mạng lưới thể chế được kết nối với Internet. Hơn nữa, giả sử liên kết truy cập đã được nâng cấp lên 54 Mbps và mạng LAN tổ chức được nâng cấp lên 10 Gbps. Giả sử rằng kích thước đối tượng trung bình là 1.600.000 bit và tốc độ yêu cầu trung bình từ các trình duyệt của viện đến các máy chủ gốc là 24 yêu cầu mỗi giây. Cũng giả sử rằng khoảng thời gian cần thiết từ khi bộ định tuyến ở phía Internet của liên kết truy cập chuyển tiếp yêu cầu HTTP cho đến khi nhận được phản hồi trung bình là ba giây (xem Phần 2.2.5). Lập mô hình tổng thời gian phản hồi trung bình

là tổng của độ trễ truy cập trung bình (nghĩa là độ trễ từ bộ định tuyến Internet đến bộ định tuyến của tổ chức) và độ trễ Internet trung bình. Đối với độ trễ truy cập trung bình, hãy sử dụng $\Delta/(1 - \Delta b)$, trong đó Δ là thời gian trung bình cần thiết để gửi một đối tượng qua

Liên kết truy cập và B là tỷ lệ đến của các đối tượng đến liên kết truy cập.

- a. Tìm tổng thời gian phản hồi trung bình.
- b. Bây giờ giả sử một bộ nhớ cache được cài đặt trong mạng LAN tổ chức. Giả sử tỷ lệ bỏ lỡ là 0,3. Tìm tổng thời gian phản hồi.

Trang 10. Hãy xem xét một liên kết dài 30 mét, qua đó người gửi có thể truyền

với tốc độ 300 bit / giây theo cả hai hướng. Giả sử rằng các gói chứa dữ liệu dài 100.000 bit và các gói chỉ chứa điều khiển (ví dụ: ACK hoặc

bắt tay) dài 200 bit. Giả sử rằng N kết nối song song mỗi kết nối nhận được $1/N$ băng thông liên kết. Bây giờ, hãy xem xét giao thức HTTP và giả sử rằng mỗi đối tượng được tải xuống dài 100 Kbit và đối tượng được tải xuống ban đầu chứa 10 đối tượng được tham chiếu từ cùng một người gửi. Tải xuống song song thông qua các phiên bản song song của HTTP không liên tục có ý nghĩa trong trường hợp này không? Bây giờ hãy xem xét HTTP liên tục. Bạn có mong đợi lợi ích đáng kể so với trường hợp không dai dẳng? Biện minh và giải thích câu trả lời của bạn.

Trang 11. Hãy xem xét kịch bản được giới thiệu trong vấn đề trước. Bây giờ, giả sử rằng liên kết được Alice chia sẻ với Bob. Alice không sử dụng các phiên bản song song của HTTP không liên tục trong khi Bob sử dụng HTTP không liên tục với năm lần tải xuống song song.

- Alice có lợi thế gì so với Bob không? Tại sao hoặc tại sao không?
- Nếu Alice mở năm trường hợp HTTP không liên tục song song, thì các kết nối song song của cô ấy có lợi không? Tại sao hoặc tại sao không?

Trang 12. Viết một chương trình TCP đơn giản cho một máy chủ chấp nhận các dòng đầu vào từ một cli-ent và in các dòng lên đầu ra tiêu chuẩn của máy chủ. (Bạn có thể làm điều này bằng cách sửa đổi chương trình TCPServer.py trong văn bản.) Biên dịch và thực hiện chương trình của bạn. Trên bất kỳ máy nào khác có chứa trình duyệt Web, hãy đặt máy chủ proxy trong trình duyệt thành máy chủ đang chạy chương trình máy chủ của bạn; Đồng thời tính toán số cổng một cách thích hợp. Trình duyệt của bạn bây giờ sẽ gửi thông báo yêu cầu GET đến máy chủ của bạn và máy chủ của bạn sẽ hiển thị các thông báo trên đầu ra tiêu chuẩn của nó. Sử dụng nền tảng này để xác định xem trình duyệt của bạn có tạo thông báo GET có điều kiện cho các đối tượng được lưu trữ cục bộ hay không.

Trang 13. Cân nhắc gửi qua HTTP / 2 một trang Web bao gồm một tệp video và ba hình ảnh. Giả sử rằng video clip được vận chuyển dưới dạng 5000 khung hình và mỗi hình ảnh chụp bốn khung hình.

- Nếu tất cả các khung hình video được gửi trước mà không cần xen kẽ, cần bao nhiêu "thời gian khung hình" cho đến khi tất cả các hình ảnh được gửi?
- Nếu các khung được xen kẽ, cần bao nhiêu lần khung hình cho đến khi cả ba hình ảnh được gửi?

Trang 14. Hãy xem xét trang Web trong vấn đề 13. Bây giờ ưu tiên HTTP / 2 được sử dụng. Giả sử tất cả các hình ảnh được ưu tiên hơn video clip và hình ảnh đầu tiên được ưu tiên hơn hình ảnh thứ hai, hình ảnh thứ hai so với hình ảnh thứ ba, v.v. Cần bao nhiêu lần khung hình cho đến khi hình ảnh thứ hai được gửi?

Trang 15. Sự khác biệt giữa MAIL FROM: in SMTP và From: trong chính thư là gì?

Trang 16. SMTP đánh dấu sự kết thúc của nội dung thư như thế nào? Làm thế nào về HTTP? HTTP có thể sử dụng phương pháp tương tự như SMTP để đánh dấu sự kết thúc của nội dung thư không? Giải thích.

Trang 17. Đọc RFC 5321 cho SMTP. MTA là viết tắt của gì? Hãy xem xét e-mail spam nhận được sau đây (được sửa đổi từ một e-mail spam thực sự). Giả sử chỉ có người khởi tạo e-mail spam này là độc hại và tất cả các máy chủ khác là trung thực, hãy xác định máy chủ độc hại đã tạo ra e-mail spam này.

Từ - Fri Nov 07 13:41:30 2008

Đường dẫn trở về: <tennis5@pp33head.com>

Đã nhận: từ barmail.cs.umass.edu (barmail.cs.umass.edu

[128.119.240.3]) bởi cs.umass.edu (8.13.1/8.12.6) cho <hg@cs.umass.edu>; Thứ sáu, 7 Nov 2008 13:27:10 -0500

Nhận: từ asusus-4b96 (localhost [127.0.0.1]) bởi barmail.cs.umass.edu (Spam Firewall) cho <hg@cs.umass.giao dục>; Thứ sáu, 7

Tháng Mười Một 2008 13:27:07 -0500 (EST)

Nhận được: từ asusus-4b96 ([58.88.21.177]) bằng đường bưu điện. cs.umass.edu

cho <hg@cs.umass.edu>; Thứ sáu, 07 Nov 2008 13:27:07 -0500 (EST)

Đã nhận: từ [58.88.21.177] bởi inbnd55.exchangeddd.Com; Thứ Bảy 8

Tháng mười một 2008 01:27:07 +0700

Từ: "Jonny" <tennis5@pp33head.com>

Đến: <hg@cs.umass.edu>

Chủ đề: Làm thế nào để đảm bảo tiền tiết kiệm của bạn

Trang 18. a. *Cơ sở dữ liệu whois là gì?*

- b. Sử dụng cơ sở dữ liệu whois khác nhau trên Internet để lấy tên của hai máy chủ DNS. Cho biết bạn đã sử dụng cơ sở dữ liệu whois nào.
- c. Sử dụng nslookup trên máy chủ lưu trữ cục bộ của bạn để gửi truy vấn DNS đến ba máy chủ DNS: máy chủ DNS cục bộ của bạn và hai máy chủ DNS bạn tìm thấy trong phần (b). Hãy thử truy vấn báo cáo Loại A, NS, và MX. Tóm tắt những phát hiện của bạn.
- d. Sử dụng nslookup để tìm một máy chủ Web có nhiều địa chỉ IP. Máy chủ Web của tổ chức của bạn (trường học hoặc công ty) có nhiều địa chỉ IP không?
- e. Sử dụng cơ sở dữ liệu ARIN whois để xác định dải địa chỉ IP được sử dụng bởi trường đại học của bạn.
- f. Mô tả cách kẻ tấn công có thể sử dụng cơ sở dữ liệu whois và công cụ nslookup để thực hiện trinh sát trên một tổ chức trước khi khởi động một cuộc tấn công.
- g. Thảo luận tại sao cơ sở dữ liệu whois nên được công bố công khai.

Trang 19. Trong vấn đề này, chúng tôi sử dụng công cụ đào hữu ích có sẵn trên các máy chủ Unix và Linux để khám phá hệ thống phân cấp của máy chủ DNS. Hãy nhớ lại rằng trong Hình 2.19, một máy chủ DNS trong hệ thống phân cấp DNS ủy quyền một truy vấn DNS cho một máy chủ DNS thấp hơn trong hệ thống phân cấp, bằng cách gửi lại cho máy khách DNS tên của máy chủ DNS cấp thấp hơn đó. Đầu tiên đọc trang người đàn ông để đào, và sau đó trả lời các câu hỏi sau.

- a. Bắt đầu với máy chủ DNS gốc (từ một trong các máy chủ gốc [a-m]. root-servers.net), bắt đầu một chuỗi các truy vấn cho địa chỉ IP cho máy chủ Web của bộ phận của bạn bằng cách sử dụng *dig*. Hiện thị danh sách tên của máy chủ DNS trong chuỗi ủy quyền khi trả lời truy vấn của bạn.
- b. Lập lại phần (a) cho một số trang web phổ biến, chẳng hạn như google.com, yahoo.com, hoặc amazon.com.

Trang 20. Xem xét các kịch bản được minh họa trong Hình 2.12 và 2.13. Giả sử tỷ lệ của mạng lưới thể chế là Rl và của liên kết nút cổ chai là Rb . Giả sử có N máy khách yêu cầu một tệp có kích thước L với HTTP cùng một lúc. Đối với những giá trị nào của Rl , việc truyền tệp sẽ mất ít thời gian hơn khi proxy được cài đặt tại mạng tổ chức? (Giả sử RTT giữa máy khách và bất kỳ máy chủ lưu trữ nào khác trong mạng tổ chức là không đáng kể.)

Trang 21. Giả sử rằng bộ phận của bạn có một máy chủ DNS cục bộ cho tất cả các máy tính trong bộ phận. Bạn là người dùng bình thường (tức là không phải là quản trị viên mạng / hệ thống). Bạn có thể xác định xem một trang web bên ngoài có khả năng được truy cập từ một máy tính trong bộ phận của bạn vài giây trước không? Giải thích.

Trang 22. Xem xét phân phối một tệp $F = 10$ Gbit cho các đồng nghiệp N .

Máy chủ có tốc độ tải lên của chúng tôi = 1 Gbps và mỗi peer có tỷ lệ tải xuống là

$di = 200$ Mbps và tốc độ tải lên là u . Với $N = 10, 100$ và 1.000 và

$u = 2$ Mbps, 10 Mbps và 100 Mbps, chuẩn bị một bảng cho thời gian phân phối tối thiểu tính bằng giây cho mỗi kết hợp N và u cho cả phân phối máy khách-máy chủ và phân phối P2P.

Trang 23. Xem xét phân phối một tệp bit F cho các đồng nghiệp N bằng cách sử dụng kiến trúc máy khách-máy chủ. Giả sử một mô hình chất lỏng trong đó máy chủ có thể đồng thời truyền đến nhiều đồng nghiệp, truyền đến từng đồng nghiệp ở các tốc độ khác nhau, miễn là tốc độ kết hợp không vượt quá chúng ta.

- a. Giả sử rằng chúng tôi / $N \dots$ TỐI THIỂU. Chỉ định sơ đồ phân phối có thời gian phân phối là NF / us .
- b. Giả sử rằng chúng tôi / $N \geq dmin$. Chỉ định sơ đồ phân phối có thời gian phân phối là $F / dmin$.
- c. Kết luận rằng thời gian phân phối tối thiểu nói chung được đưa ra bởi tối đa $5 NF / us, F / dmin$ 6.

Trang 24. Cân nhắc phân phối tệp bit F cho các đồng nghiệp N bằng kiến trúc P2P.

Giả sử một mô hình chất lỏng. Để đơn giản, giả sử rằng $dmin$ rất lớn, do đó băng thông tải xuống ngang hàng không bao giờ là nút cổ chai.

- a. Giả sử rằng *chúng tôi* ... $(M\tilde{y} + U1 + \dots + uN)/N$. Chỉ định sơ đồ phân phối có thời gian phân phối là F / us .

- b. Giả sử rằng chúng ta \bar{U} (chúng ta $+ u1 + \dots + uN)/N$. Chỉ định số đồ phân phối có thời gian phân phối là $NF / (us + u1 + \dots + uN)$.
- c. Kết luận rằng thời gian phân phối tối thiểu nói chung được đưa ra bởi tối đa $5 F / us, NF / (us + u1 + \dots + uN) 6$.

Trang 25. Hãy xem xét một mạng lớp phủ với N đồng nghiệp hoạt động, với mỗi cặp ngang hàng có một kết nối TCP đang hoạt động. Ngoài ra, giả sử rằng các kết nối TCP đi qua tổng cộng các bộ định tuyến M . Có bao nhiêu nút và cạnh trong mạng lớp phủ tương ứng?

Trang 26. Giả sử Bob tham gia một torrent BitTorrent, nhưng anh ta không muốn tải bất kỳ dữ liệu nào lên bất kỳ đồng nghiệp nào khác (anh ta muốn trở thành người được gọi là người lái tự do).

- a. Alice, người đã sử dụng BitTorrent, nói với Bob rằng anh ta không thể nhận được một bản sao hoàn chỉnh của tệp được chia sẻ bởi bầy đàn. Alice có đúng hay không? Tại sao?
- b. Charlie tuyên bố rằng Alice đã sai và anh ta thậm chí đã sử dụng một tập hợp nhiều máy tính (với các địa chỉ IP riêng biệt) trong phòng thí nghiệm máy tính trong bộ phận của mình để làm cho việc tải xuống của anh ta nhanh hơn, sử dụng một số kịch bản phối hợp bổ sung. Kịch bản của anh ấy có thể làm gì?

Trang 27. Hãy xem xét một hệ thống DASH có N phiên bản video (ở N tốc độ và chất lượng khác nhau) và phiên bản âm thanh N (ở N tốc độ và chất lượng khác nhau). Giả sử chúng tôi muốn cho phép trình phát chọn bất kỳ lúc nào bất kỳ phiên bản video N nào và bất kỳ phiên bản âm thanh N nào.

- a. Nếu chúng tôi tạo tệp để âm thanh được trộn lẫn với video, vì vậy máy chủ chỉ gửi một luồng phương tiện tại một thời điểm nhất định, máy chủ sẽ cần lưu trữ bao nhiêu tệp (mỗi URL khác nhau)?
- b. Thay vào đó, nếu máy chủ gửi các luồng âm thanh và video riêng biệt và có máy khách đồng bộ hóa các luồng, máy chủ sẽ cần lưu trữ bao nhiêu tệp?

Trang 28. Cài đặt các chương trình Python TCPClient và UDPClient trên một máy chủ và TCPServer và UDPServer trên một máy chủ khác.

- a. Giả sử bạn chạy TCPServer và bạn cố gắng kết nối bằng UDPClient. Chuyện gì xảy ra? Tại sao?
- b. Giả sử bạn chạy UDPClient trước khi chạy UDPServer. Chuyện gì xảy ra? Tại sao?
- c. Điều gì xảy ra nếu bạn hardwire trong các chương trình máy khách và máy chủ python các số cổng khác nhau cho phía máy khách và máy chủ trong cặp máy khách-máy chủ TCP hoặc UDP?

Trang 29. Giả sử rằng trong UDPClient.py, sau khi chúng ta tạo ổ cắm, chúng ta thêm dòng:

```
clientSocket.bind(('', 5432))
```

Nó sẽ trở nên cần thiết để thay đổi UDPServer.py? Số cổng cho các ổ cắm trong UDPClient và UDPServer là gì? Họ đã làm gì trước khi thực hiện thay đổi này?

Trang 30. Bạn có thể cấu hình trình duyệt của mình để mở nhiều kết nối đồng thời đến một trang Web không? Ưu điểm và nhược điểm của việc có một số lượng lớn các kết nối TCP đồng thời là gì?

Trang 31. Chúng ta đã thấy rằng các ổ cắm TCP Internet coi dữ liệu được gửi dưới dạng luồng byte nhưng các ổ cắm UDP nhận ra ranh giới tin nhắn. Một lợi thế và một nhược điểm của API hướng byte so với việc API nhận dạng và duy trì rõ ràng các ranh giới thông điệp do ứng dụng xác định là gì?

Trang 32. Máy chủ Web Apache là gì? Giá bao nhiêu? Nó hiện có chức năng gì? Bạn có thể muốn xem Wikipedia để trả lời câu hỏi này.

Bài tập lập trình ổ cắm

Trang web đồng hành bao gồm sáu bài tập lập trình ổ cắm. Bốn bài tập đầu tiên được tóm tắt dưới đây. Nhiệm vụ thứ năm sử dụng giao thức ICMP và được tóm tắt ở cuối Chương 5. Chúng tôi rất khuyến khích học sinh hoàn thành một số, nếu không phải tất cả, các bài tập này. Học sinh có thể tìm thấy chi tiết đầy đủ về các bài tập này, cũng như các đoạn mã quan trọng của mã Python, tại trang web www.pearsonglobaleditions.com.

Gán 1: Máy chủ Web

Trong bài tập này, bạn sẽ phát triển một máy chủ Web đơn giản bằng Python có khả năng xử lý chỉ một yêu cầu. Cụ thể, máy chủ Web của bạn sẽ (i) tạo một ổ cắm kết nối khi được liên hệ bởi một máy khách (trình duyệt); (ii) nhận yêu cầu HTTP từ kết nối này; (iii) phân tích cú pháp yêu cầu để xác định tệp cụ thể đang được yêu cầu; (iv) lấy tệp được yêu cầu từ hệ thống tệp của máy chủ; (v) tạo thông báo phản hồi HTTP bao gồm tệp được yêu cầu đứng trước các dòng tiêu đề; và (vi) gửi phản hồi qua kết nối TCP đến trình duyệt yêu cầu. Nếu trình duyệt yêu cầu tệp không có trong máy chủ của bạn, máy chủ của bạn sẽ trả về thông báo lỗi "404 Not Found".

Trong Trang web Đồng hành, chúng tôi cung cấp mã khung xương cho máy chủ của bạn. Công việc của bạn là hoàn thành mã, chạy máy chủ của bạn và sau đó kiểm tra máy chủ của bạn bằng cách gửi yêu cầu từ các trình duyệt chạy trên các máy chủ khác nhau. Nếu bạn chạy máy chủ của bạn trên một máy chủ đã có một máy chủ Web chạy trên đó, sau đó bạn nên sử dụng một cổng khác với cổng 80 cho máy chủ Web của bạn.

Bài tập 2: UDP Pinger

Trong bài tập lập trình này, bạn sẽ viết một chương trình ping máy khách bằng Python. Khách hàng của bạn sẽ gửi một tin nhắn ping đơn giản đến máy chủ, nhận lại tin nhắn pong tương ứng từ máy chủ và xác định độ trễ giữa thời điểm máy khách

đã gửi tin nhắn ping và nhận được tin nhắn Pong. Sự chậm trễ này được gọi là Thời gian khứ hồi (RTT). Chức năng được cung cấp bởi máy khách và máy chủ tương tự như chức năng được cung cấp bởi chương trình ping tiêu chuẩn có sẵn trong các hệ thống điều hành hiện đại. Tuy nhiên, các chương trình ping tiêu chuẩn sử dụng Giao thức tin nhắn điều khiển Internet (ICMP) (mà chúng ta sẽ nghiên cứu trong Chương 5). Ở đây chúng tôi sẽ tạo ra một không chuẩn (nhưng đơn giản!) Chương trình ping dựa trên UDP.

Chương trình ping của bạn là gửi 10 tin nhắn ping đến máy chủ mục tiêu qua UDP. Đối với mỗi tin nhắn, khách hàng của bạn sẽ xác định và in RTT khi tin nhắn pong tương quan được trả về. Vì UDP là một giao thức không đáng tin cậy, một gói được gửi bởi máy khách hoặc máy chủ có thể bị mất. Vì lý do này, khách hàng không thể chờ đợi một cách bất chấp để trả lời tin nhắn ping. Bạn nên yêu cầu khách hàng đợi tối đa một giây để trả lời từ máy chủ; Nếu không nhận được trả lời, khách hàng nên cho rằng gói tin đã bị mất và in tin nhắn tương ứng.

Trong bài tập này, bạn sẽ được cung cấp mã hoàn chỉnh cho máy chủ (có sẵn trong Trang web đồng hành). Công việc của bạn là viết mã máy khách, mã này sẽ rất giống với mã máy chủ. Trước tiên, bạn nên nghiên cứu kỹ mã máy chủ. Sau đó, bạn có thể viết mã máy khách của mình, tự do cắt và dán các dòng từ mã máy chủ.

Nhiệm vụ 3: Ứng dụng thư khách

Mục tiêu của nhiệm vụ lập trình này là tạo ra một ứng dụng thư khách đơn giản gửi e-mail đến bất kỳ người nhận nào. Khách hàng của bạn sẽ cần thiết lập kết nối TCP với máy chủ thư (ví dụ: máy chủ thư Google), đối thoại với máy chủ thư bằng giao thức SMTP, gửi thư e-mail đến người nhận (ví dụ: bạn bè của bạn) qua máy chủ thư và cuối cùng đóng kết nối TCP với máy chủ thư.

Đối với nhiệm vụ này, Trang web đồng hành cung cấp mã khung xương cho khách hàng của bạn. Công việc của bạn là hoàn thành mã và kiểm tra khách hàng của bạn bằng cách gửi e-mail đến các tài khoản người dùng khác nhau. Bạn cũng có thể thử gửi qua các máy chủ khác nhau (ví dụ: thông qua máy chủ thư của Google và thông qua máy chủ thư của trường đại học của bạn).

Nhiệm vụ 4: Web Proxy

Trong nhiệm vụ này, bạn sẽ phát triển một proxy Web. Khi proxy của bạn nhận được yêu cầu HTTP cho một đối tượng từ trình duyệt, nó sẽ tạo một yêu cầu HTTP mới cho cùng một đối tượng và gửi nó đến máy chủ gốc. Khi proxy nhận được phản hồi HTTP tương ứng với đối tượng từ máy chủ gốc, nó sẽ tạo ra một phản hồi HTTP mới, bao gồm cả đối tượng và gửi nó đến máy khách.

Đối với nhiệm vụ này, Trang web đồng hành cung cấp mã bộ xương cho máy chủ proxy. Công việc của bạn là hoàn thành mã, và sau đó kiểm tra nó bằng cách yêu cầu các trình duyệt khác nhau yêu cầu các đối tượng Web thông qua proxy của bạn.

Phòng thí nghiệm Wireshark: HTTP

Sau khi chân ướt chân ráo với trình đánh hơi gói Wireshark trong Phòng thí nghiệm 1, giờ đây chúng tôi đã sẵn sàng sử dụng Wireshark để điều tra các giao thức đang hoạt động. Trong phòng thực hành này, chúng ta sẽ khám phá một số khía cạnh của giao thức HTTP: tương tác GET / reply cơ bản, định dạng tin nhắn HTTP, truy xuất các tệp HTML lớn, truy xuất các tệp HTML có URL nhúng, kết nối liên tục và không liên tục, xác thực và bảo mật HTTP.

Như trường hợp của tất cả các phòng thí nghiệm Wireshark, mô tả đầy đủ về phòng thí nghiệm này có sẵn tại trang web của cuốn sách này, www.pearsonglobaleditions.com.

Phòng thí nghiệm Wireshark: DNS

Trong phòng thực hành này, chúng ta hãy xem xét kỹ hơn phía máy khách của DNS, giao thức dịch tên máy chủ Internet sang địa chỉ IP. Hãy nhớ lại từ Phần 2.5 rằng vai trò của cli-ent trong DNS tương đối đơn giản — một máy khách gửi một truy vấn đến máy chủ DNS cục bộ của nó và nhận được phản hồi trở lại. Phần lớn có thể tiếp tục dưới vỏ bọc, vô hình đối với các máy khách DNS, vì các máy chủ DNS phân cấp giao tiếp với nhau để giải quyết đệ quy hoặc lặp đi lặp lại truy vấn DNS của máy khách. Tuy nhiên, từ quan điểm của DNS cli-ent, giao thức khá đơn giản — một truy vấn được xây dựng cho máy chủ DNS cục bộ và nhận được phản hồi từ máy chủ đó. Chúng tôi quan sát thấy DNS hoạt động trong phòng thực hành này.

Như trường hợp của tất cả các phòng thí nghiệm Wireshark, mô tả đầy đủ về phòng thí nghiệm này có sẵn tại trang web của cuốn sách này, www.pearsonglobaleditions.com.

MỘT CUỘC PHÒNG VÁN VỚI...

Tim Berners-Lee

Ngài Tim Berners-Lee được biết đến như là người phát minh ra World Wide Web. Năm 1989, trong khi làm việc như một thành viên tại CERN, ông đã đề xuất một hệ thống quản lý thông tin phân tán dựa trên Internet bao gồm phiên bản gốc của giao thức HTTP. Trong cùng năm đó, ông đã thực hiện thành công thiết kế của mình trên máy khách và máy chủ. Ông đã nhận được giải thưởng Turing năm 2016 cho "phát minh ra World Wide Web, trình duyệt Web đầu tiên và các giao thức và thuật toán cơ bản cho phép Web mở rộng quy mô". Ông là người đồng sáng lập World Wide Web Foundation, và hiện là thành viên giáo sư về khoa học máy tính tại Đại học Oxford và giáo sư tại CSAIL tại MIT.



Được phép của Tim Berners-

Ban đầu bạn học vật lý. Mạng tương tự như vật lý như thế nào?

Khi bạn nghiên cứu vật lý, bạn tưởng tượng những quy tắc ứng xử nào ở quy mô rất nhỏ có thể tạo ra thế giới quy mô lớn như chúng ta thấy. Khi bạn thiết kế một hệ thống toàn cầu như Web, bạn cố gắng phát minh ra các quy tắc ứng xử của các trang web và liên kết và những thứ có thể tạo ra một thế giới quy mô lớn như chúng ta muốn. Một là phân tích và tổng hợp khác, nhưng chúng rất giống nhau.

Điều gì đã ảnh hưởng đến bạn đề chuyên về mạng?

Sau khi tốt nghiệp ngành vật lý, các công ty nghiên cứu viễn thông dường như là những nơi thú vị nhất. Bộ vi xử lý vừa xuất hiện và viễn thông đang chuyển đổi rất nhanh từ logic cứng sang các hệ thống dựa trên bộ vi xử lý. Nó rất thú vị.

Phần thử thách nhất trong công việc của bạn là gì?

Khi hai nhóm bắt đầu mạnh mẽ về điều gì đó, nhưng cuối cùng muốn đạt được mục tiêu chung, việc tìm ra chính xác ý nghĩa của mỗi nhóm và hiểu làm ở đâu có thể rất khó khăn. Chủ tịch của bất kỳ nhóm làm việc nào cũng biết điều đó. Tuy nhiên, đây là những gì cần thiết để đạt được tiến bộ hướng tới sự đồng thuận trên quy mô lớn.

Những người nào đã truyền cảm hứng cho bạn một cách chuyên nghiệp?

Cha mẹ tôi, những người đã tham gia vào những ngày đầu của máy tính, đã cho tôi một niềm đam mê với toàn bộ chủ đề. Mike Sendall và Peggie Rimmer, những người mà tôi đã làm việc nhiều lần tại CERN là một trong những người đã dạy tôi và khuyến khích tôi. Sau đó tôi đã học được cách ngưỡng mộ mọi người, bao gồm Vannevar Bush, Doug Englebart và Ted Nelson, những người đã có những giấc mơ tương tự trong thời đại của họ nhưng không có lợi ích của sự tồn tại để PC và Internet có thể thực hiện nó.