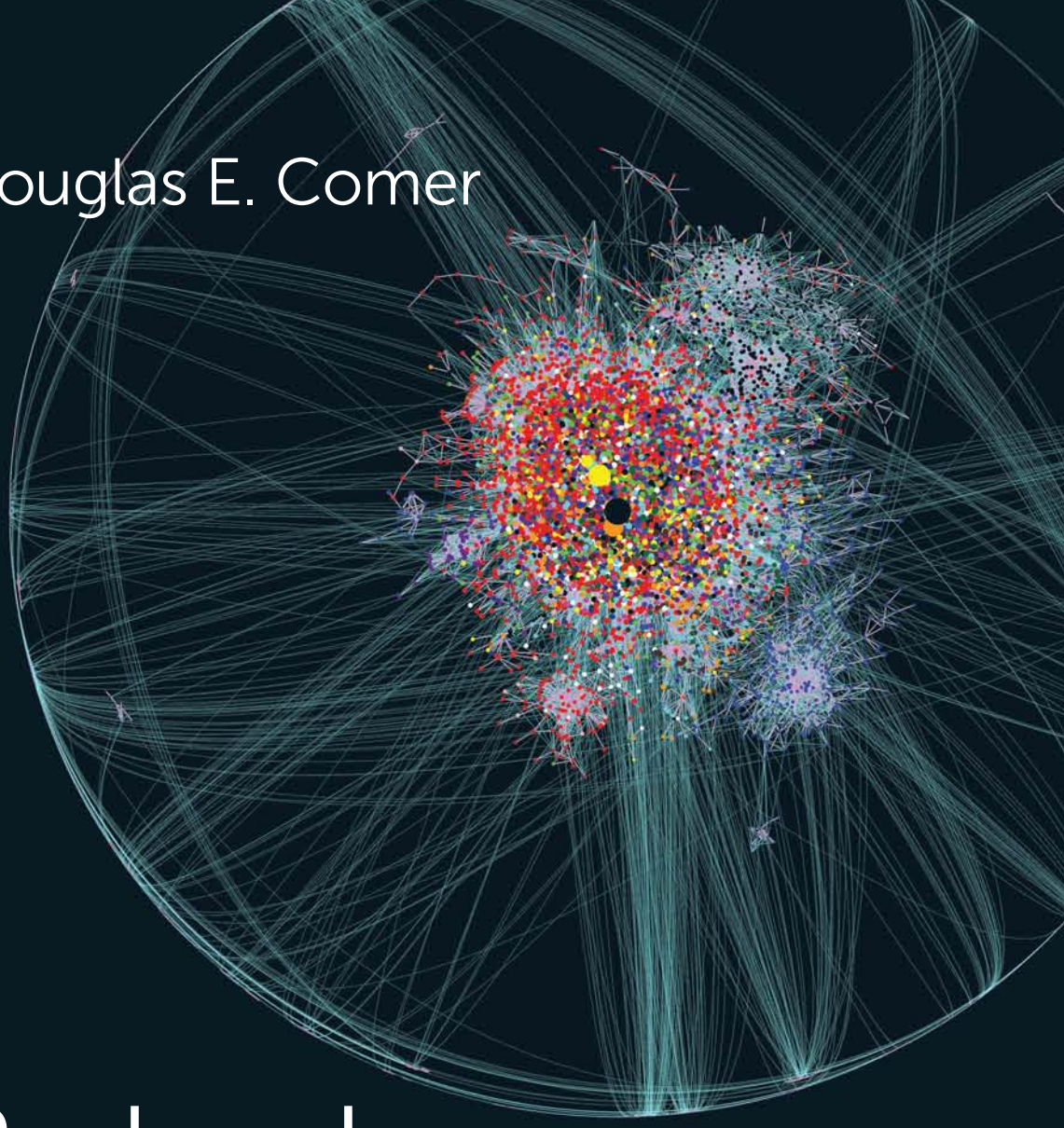
Douglas E. Comer



Redes de Computadores e Internet

bookman

6ª EDIÇÃO

O autor

Dr. Douglas Comer é um internacionalmente reconhecido especialista em redes de computadores, protocolos TCP/IP e Internet. Foi um dos pesquisadores que contribuíram com a formação da Internet no fim dos anos 1970 e nos anos 1980, sendo membro do *Internet Architecture Board*, o grupo responsável por guiar o desenvolvimento da Internet. Também foi presidente do comitê técnico CSNET, membro do comitê executivo CSNET e presidente do Distributed Systems Architecture Board da DARPA (*Defense Advanced Research Projects Agency*). Foi ainda Vice-Presidente de Pesquisa na Cisco Systems.

Comer é consultor de projeto de redes de computadores para empresas e palestrante frequente em ambientes acadêmicos e profissionais ao redor do mundo. Seu sistema operacional, Xinu, e a implementação de protocolos TCP/IP (ambos documentados em seus livros) são utilizados em produtos comerciais. É professor honorário de Ciências da Computação na Purdue University, onde leciona redes de computadores, redes de internet, arquitetura de computadores e sistemas operacionais. Lá desenvolveu laboratórios de informática inovadores que dão aos alunos a oportunidade de ter experiências práticas na operação de sistemas, redes de computadores e protocolos.

Além de escrever livros técnicos *best-sellers*, já traduzidos para 16 idiomas, atuou como editor norte-americano do periódico *Software – Practice and Experience* por 20 anos. Comer é membro da ACM. Informações adicionais podem ser encontradas em: *www.cs.purdue.edu/homes/comer.*

C732r Comer, Douglas E.
Redes de computadores e internet [recurso eletrônico] / Douglas E. Comer ; tradução: José Valdeni de Lima, Valter Roesler. – 6. ed. – Porto Alegre : Bookman, 2016.

Editado como livro impresso em 2016.
ISBN 978-85-8260-373-4

1. Redes de computadores. 2. Internet. I. Título.

CDU 004.7

Catalogação na publicação: Poliana Sanchez de Araujo – CRB 10/2094

CAPÍTULO 4

Aplicações tradicionais da Internet

4.1 Introdução

O capítulo anterior introduz os tópicos de aplicativos da Internet e programação em redes. Este capítulo explica que os serviços da Internet são executados por aplicativos que interagem por meio do modelo cliente-servidor. Ele também aborda a API de sockets.

Além disso, continua o exame dos aplicativos da Internet. Também define o conceito de um protocolo de transferência e explica como aplicativos implementam protocolos de transferência. Por fim, considera exemplos de aplicativos da Internet que foram padronizados e descreve o protocolo de transferência para cada uso.

4.2 Protocolos da camada de aplicação

Sempre que um programador cria dois aplicativos que se comunicam, ele especifica detalhes como:

- A sintaxe e a semântica da mensagem que pode ser trocada.
- O iniciador da interação, que é ou o cliente ou o servidor.
- As ações a serem realizadas em caso de erro.
- Como os dois lados sabem quando a comunicação termina.

Ao especificar os detalhes de comunicação, um programador define um protocolo da camada de aplicação. Existem dois tipos genéricos de protocolos da camada de aplicação que dependem do uso pretendido:

- *Serviço privado*. Um programador ou uma empresa cria um par de aplicativos que se comunicam por meio da Internet com a intenção de que nenhum outro crie software cliente ou servidor para esse serviço. Não existe necessidade de publicar e distribuir uma especificação formal para definir a interação, porque nenhuma aplicação externa precisaria entender os detalhes de protocolo. De fato, se a interação entre os dois aplicativos é suficientemente simples, pode até não haver um documento de protocolo interno.
- *Serviço padronizado*. Um serviço da Internet é definido com a expectativa de que muitos programadores criem o software servidor que oferece o serviço ou o software cliente para acessar o serviço. Em tais casos, o protocolo da camada de aplicação deve ser documentado independentemente de qualquer implementação. Além disso, a especificação deve ser precisa, e não ambígua, para que aplicativos cliente-servidor possam ser construídos de tal modo que se *interoperem* corretamente.

O tamanho da especificação do protocolo depende da complexidade do serviço; a especificação para um serviço trivial pode caber em uma única página de texto. Por exemplo, os protocolos da Internet incluem um serviço padronizado de aplicação conhecido como *DAYTIME*, que permite ao cliente procurar a data, o local e o horário na localização do servidor. O protocolo é simples: um cliente faz a conexão com o servidor, que envia uma representação ASCII de data e horário e desliga a conexão. Por exemplo, um servidor pode enviar um string como este:

Mon Sep 9 20:18:37 2013

O cliente lê os dados da conexão até que o caractere "*fim-de-arquivo*" seja encontrado.

Em síntese:

> *Para permitir que aplicativos de serviços padronizados se interoperem, um protocolo padrão da camada de aplicação é criado independentemente de qualquer implementação.*

4.3 Representação e transferência

Os protocolos da camada de aplicação especificam dois aspectos da interação: representação e transferência. A Figura 4.1 explica a diferença.

Aspecto	Descrição
Representação dos dados	Sintaxe dos itens de dados que são trocados, forma específica usada durante a transferência, tradução de inteiros, caracteres e arquivos enviados entre os computadores
Transferência dos dados	Interação entre cliente e servidor, sintaxe e semântica da mensagem, tratamento de erros de troca válida e inválida e fim da interação

Figura 4.1 Dois aspectos-chave de um protocolo da camada de aplicação.

Para um serviço básico, um simples protocolo padrão pode especificar ambos aspectos; para serviços mais complexos, usa-se protocolos padrão separados para especificar cada aspecto. Por exemplo, o protocolo DAYTIME descrito anteriormente usa um simples padrão para especificar que uma data e um horário são representados como um string ASCII e que a transferência é feita por um servidor que envia um string e fecha a conexão. A próxima seção explica que serviços mais complexos definem protocolos separados para descrever a sintaxe dos objetos e a transferência deles. Os projetistas de protocolos fazem a distinção clara entre os dois aspectos:

> *Por convenção, a palavra* transfer *no título de um protocolo da camada de aplicação significa que o protocolo especifica o aspecto de transferência dos dados da comunicação.*

4.4 Protocolos Web

A WWW (*World Wide Web)* é um dos serviços mais utilizados na Internet. Devido à complexidade da rede, muitos padrões de protocolo têm sido criados para especificar vários aspectos e detalhes. A Figura 4.2 ilustra os três padrões-chave.

Padrão	Propósito
HyperText Markup Language (HTML)	Uma representação padrão usada para especificar os conteúdos e o formato de uma página Web
Uniform Resource Locator (URL)	Uma representação padrão que especifica o formato e o significado dos identificadores da página Web
HyperText Transfer Protocol (HTTP)	Um protocolo de transferência que especifica como um browser interage com o servidor Web para transferir dados

Figura 4.2 Três padrões-chave usados pelo serviço WWW.

4.5 Representação de documento com HTML

A *HyperText Markup Language (*HTML) é um padrão de representação que especifica a sintaxe para a página Web. O HTML tem as seguintes características gerais:

- Usa uma representação textual.
- Descreve páginas Web que contém multimídia.
- Segue o paradigma da linguagem declarativa em vez do da imperativa.

- Fornece as especificações de marcação (*markup specifications*) em vez de formatação.
- Permite que um hiperlink seja aninhado em um objeto qualquer.
- Permite ao documento incluir metadados.

Embora um documento HTML seja um arquivo de texto, a linguagem permite ao programador especificar uma página Web complexa que contem gráficos, áudio e vídeo. De fato, para serem precisos, os projetistas deveriam ter usado o termo hipermídia em vez de *hipertexto*, porque a HTML permite que um objeto qualquer, como uma imagem, contenha um link para outra página (algumas vezes chamado de hyperlink).

A HTML é classificada como linguagem *declarativa* porque permite apenas especificar o que deve ser feito, não como fazer. Ela é classificada como uma *linguagem de marcação* (*markup language*) porque fornece apenas as instruções gerais de visualização e não inclui instruções de formatação detalhadas. Por exemplo, a HTML permite que uma página declare o nível de importância de um cabeçalho, mas não exige que o autor especifique os detalhes da fonte de caracteres, como o tipo exato da fonte, o projeto de uma nova fonte, o tamanho e o espacejamento usado para materializar o cabeçalho[1]. Efetivamente, um navegador tem a liberdade de escolher a maioria dos detalhes de materialização de uma página. O uso de uma linguagem de marcação é importante porque permite ao navegador adaptar a página ao hardware de materialização. Por exemplo, uma página pode ser formatada em alta ou baixa resolução, para um quadro (*frame*) com uma resolução particular, uma tela grande ou uma tela pequena de um aparelho de mão, tais como smartphone ou tablet.

Para resumir:

> HyperText Markup Language *(HTML) é um padrão de representação para páginas Web. A fim de permitir que uma página seja materializada em um equipamento qualquer, a HTML fornece instruções gerais para materializar e permitir ao navegador escolher os detalhes para a referida materialização.*

Para especificar as marcações, a HTML usa as *marcas* (*tags*) aninhadas no documento. Essas marcas consistem em um termo entre os símbolos "*menor do que*" e "*maior do que*"; elas fornecem a estrutura do documento tão bem quanto as dicas de formatação. As marcas controlam toda a materialização; espaço em branco (isto é, linhas extras e caracteres em branco) pode ser inserido em qualquer ponto do documento HTML sem qualquer efeito na versão formatada que o navegador materializa.

Um documento HTML inicia com a marca `<HTML>` e termina com a `</HTML>`. O par de marcas `<HEAD>` e `</HEAD>` delimita o cabeçalho, enquanto o par de marcas `<BODY>` e `</BODY>` delimita o corpo. No cabeçalho, as marcas `<TITLE>` e `</TITLE>` delimitam o título do documento. A Figura 4.3 ilustra o formato geral de um documento HTML[2].

A HTML usa a marca *IMG* para codificar a referência para uma imagem externa. Por exemplo, a marca

```
<IMG SRC="house_icon.jpg">
```

[1] A HTML inclui extensões que permitem especificar as fontes de caracteres, mas elas não são obrigatórias.

[2] A HTML não faz distinção entre letras maiúsculas e minúsculas nas marcas; letras maiúsculas são usadas para enfatizar.

```
<HTML>

  <HEAD>
     <TITLE>
           text that forms the document title
     </TITLE>
  </HEAD>

  <BODY>
    body of the document appears here
  </BODY>

</HTML>
```

Figura 4.3 O formato geral do documento HTML.

especifica que o arquivo *house_icon.jpg* contém uma imagem que o navegador deveria inserir no documento. Parâmetros adicionais podem ser especificados na marca IMG para definir o alinhamento da figura com textos ao redor. Por exemplo, a Figura 4.4 ilustra a saída para o seguinte HTML com a figura alinhada com o meio da linha.

```
Here is an icon of a house. <IMG SRC="house_icon.jpg" ALIGN=middle>
```

Um navegador posiciona a imagem verticalmente, assim o texto é alinhado com o meio da imagem.

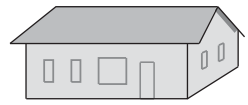
Here is an icon of a house. 

Figura 4.4 Ilustra o alinhamento da figura em HTML.

4.6 Uniform Resource Locators e hiperlinks

A Web usa uma forma sintática conhecida como *Uniform Resource Locator* (*URL*) para localizar uma página Web. A forma geral da URL é:

protocol : / / computer_name : port / document_name ? parameters

onde *protocol* é o nome do protocolo usado para acessar o documento, *computer_name* é o nome do domínio do computador no qual o document reside, *port* é um número de porta de protocolo facultativo em que o servidor está atendendo, *document_name* é o nome opcional do documento no computador especificado e *? parameters* são os parâmetros opcionais para a página.

Por exemplo, a URL

http://www.netbook.cs.purdue.edu/example.html

especifica o protocolo *http*, um computador chamado *www.netbook.cs.purdue.edu* e o nome do arquivo *example.html*.

As URLs típicas que os usuários utilizam têm muitas de suas partes omitidas. Por exemplo, a URL

www.netbook.cs.purdue.edu

omite o protocolo (http é assumido), a porta (80 é assumida), o nome do documento (index.html é assumido) e os parâmetros (nenhum é assumido).

A URL contém a informação necessária para que o navegador recupere a página. O navegador usa os caracteres separadores vírgula, barra e interrogação para dividir a URL em cinco componentes: um protocolo, um nome de computador, um número de porta do protocolo, um nome de documento e os parâmetros. O navegador usa o nome do computador e o número da porta do protocolo para fazer a conexão com o servidor no qual a página reside e usa o nome do documento e os parâmetros para requerer a página específica.

Em HTML, uma marca *âncora* usa URLs para ter a capacidade hiperlink (isto é, a habilidade de ligar uma página Web em outra). O exemplo seguinte mostra um documento HTML com uma âncora na página da *Prentice Hall*:

```
This book is published by
<A HREF="http://www.prenhall.com">
Prentice Hall</A>, one of
the larger publishers of Computer Science textbooks.
```

A âncora referencia *http://www.prenhall.com*. Quando mostrada na tela, a entrada HTML produz:

This book is published by Prentice Hall, one of the larger publishers of Computer Science textbooks.

4.7 A transferência de documento Web com HTTP

O *protocolo de transferência de hipertexto* (HTTP, *HyperText Transfer Protocol*), é o protocolo de transferência principal que um navegador usa para interagir com um servidor Web. Em termos de modelo cliente-servidor, o navegador é um cliente que extrai o nome do servidor de uma URL e contata-o. A maioria das URLs contém uma referência explícita ao protocolo *HTTP*; caso ela seja omitida, o HTTP é assumido.

O HTTP pode ser caraterizado da seguinte maneira:

- Usa mensagem de controle textual
- Transfere arquivos de dados binários
- Pode baixar ou carregar dados
- Incorpora caching

Uma vez que estabeleça uma conexão, um navegador envia uma *requisição* HTTP para o servidor. A Figura 4.5 lista os quatro tipos principais de requisições:

Requisição	Descrição
GET	Requere um documento; o servidor responde com envio da informação do status seguido por uma cópia do documento
HEAD	Requere a informação do status; o servidor responde com a informação do status, mas não envia uma cópia do documento
POST	Envia dados para o servidor; o navegador anexa os dados a um item especificado (isto é, uma mensagem é anexada a uma lista)
PUT	Envia dados para o servidor; o servidor usa os dados para substituir completamente o item especificado (isto é, subescreve os dados anteriores)

Figura 4.5 Os quatro tipos principais de requisições.

A forma mais comum de interação começa quando um navegador requere uma página de um servidor. O navegador envia uma requisição *GET* na conexão e o servidor responde enviando um cabeçalho, uma linha em branco e o documento requisitado. No HTTP, uma requisição e um cabeçalho usados em resposta são informações textuais. Por exemplo, uma requisição *GET* tem a seguinte forma:

GET */item version CRLF*

onde *item* fornece a URL para o item que está sendo requisitado, *version* especifica uma versão do protocolo (usualmente HTTP/1.0 ou HTTP/1.1) e *CRLF* denota dois caracteres ASCII, *carriage return* e *linefeed*, que são usados para indicar o fim de uma linha de texto.

A informação de versão é importante no HTTP porque ela permite que o protocolo mude e permaneça compatível com a versão anterior. Por exemplo, quando um navegador usa a versão 1.0 do protocolo e interage com um servidor que usa uma versão mais atual, o servidor retorna para a versão anterior do protocolo e formula uma resposta de acordo. Para resumir:

> *Um navegador, quando usa HTTP, envia a informação de versão que permite ao servidor escolher a versão mais atualizada do protocolo, que o navegador e o servidor possam entender.*

A primeira linha de um cabeçalho-resposta contém um código status que diz ao navegador se tem o servidor manipulou a requisição. Se a requisição foi formulada incorretamente ou se o item requisitado não estava disponível, o código do status aponta o problema. Por exemplo, um servidor retorna o conhecido status código *404* se o item requisitado não pôde ser encontrado. Quando a requisição é atendida, um servidor retorna o status código *200*; linhas adicionais do cabeçalho fonecem mais informações sobre o item, como seu comprimento, quando foi modificado pela última vez e o tipo de conteúdo. A Figura 4.6 mostra o formato geral de linhas em um cabeçalho básico de resposta.

O campo *status_code é* um valor numérico representado como uma cadeia de caracteres de dígitos decimais que informa um *status* e um *status_string*, que é uma explicação correspondente para um humano ler. A Figura 4.7 lista exemplos de strings e códigos de status comumente usados. O campo *server_identification* contém uma cadeia de caracteres que fornece uma descrição do servidor legível pelo ser humano, possi-

```
HTTP/1.0 status_codestatus_stringCRLF
Server: server_identificationCRLF
Last-Modified: date_document_was_changedCRLF
Content-Length: datasizeCRLF
Content-Type: document_typeCRLF
CRLF
```

Figura 4.6 Formato geral de linhas em um cabeçalho básico de resposta.

velmente incluindo o nome do domínio do servidor. O campo *datasize* no cabeçalho *Content-Length* especifica o tamanho do item de dados que segue, medido em bytes. O campo *document_type* contém uma cadeia de caracteres que informa ao navegador sobre o conteúdo do documento. A cadeia de caracteres contém dois itens separados por uma barra: o tipo do documento e sua representação. Por exemplo, quando um servidor retorna um documento HTML, o *document_type* é *text/html* e, quando ele retorna um arquivo jpeg, o tipo é *image/jpeg*.

Código do status	Cadeia de caracteres correspondente do status
200	OK
400	*Bad Request*
404	*Not Found*

Figura 4.7 Exemplos de códigos de status usados no HTTP.

A Figura 4.8 mostra um exemplo de saída de um servidor Web Apache. O item requisitado é um arquivo de texto de 16 caracteres (isto é, o texto *This is a test.* mais o caractere *NEWLINE*). Embora a requisição GET especifique a versão HTTP 1.0, o servidor executa a versão 1.1. O servidor retorna nove linhas de cabeçalho, uma linha em branco e o conteúdo do arquivo.

```
HTTP/1.1 200 OK
Date: Sat, 1 Aug 2013 10:30:17 GMT
Server: Apache/1.3.37 (Unix)
Last-Modified: Thu, 15 Mar 2012 07:35:25 GMT
ETag: "78595-81-3883bbe9"
Accept-Ranges: bytes
Content-Length: 16
Connection: close
Content-Type: text/plain

This is a test.
```

Figura 4.8 Exemplo de resposta HTTP de um servidor Web Apache.

4.8 Caching nos navegadores

O caching fornece uma otimização importante para o acesso Web, porque usuários tendem a visitar o mesmo site Web repetidamente. Grande parte do conteúdo de um dado

site consiste em grandes imagens que usam padrões tais como *Graphics Image Format* (*GIF*) ou *Joint Photographic Experts Group* (*JPEG*). Tais imagens geralmente contêm fundos ou banners que não mudam com frequência. A ideia-chave é:

> *Um navegador pode reduzir o tempo de download significativamente salvando uma cópia de cada imagem em um cache no disco do usuário e usando essa cópia.*

Uma questão pertinente: o que acontece se o documento no servidor Web muda depois que um navegador armazena uma cópia da imagem no seu cache? Isto é, como um navegador pode saber se sua cópia no cache está *desatualizada*? A Figura 4.8 contém uma pista: o cabeçalho *Last-Modified*. Sempre que um navegador obtém um documento a partir de um servidor Web, o cabeçalho especifica a última vez que o documento foi alterado. Um navegador salva a informação da data Last-Modified da cópia armazenada no cache. Antes de usar um documento do cache local, o navegador faz uma requisição do *HEAD* para o servidor e compara a data *Last-Modified* do servidor com a data *Last-Modified* da cópia do cache. Se a versão do cache está desatualizada, ele carrega a nova versão. O Algoritmo 4.1 resume o caching.

Algoritimo 4.1

```
Dada:
    Uma URL para um item em uma página Web
Obter:
    Uma cópia da página
Método:
    if (o item não está no cache local) {
            Requisição GET solicita e armazena uma cópia em cache
    } else {
        Requisição HEAD solicita ao servidor;
        if (item em cache está atualizado) {
            usar item em cache;
        } else {
            Requisição GET solicita e armazena uma cópia em cache
        }
    }
```

Algoritmo 4.1 O caching no navegador é usado pra reduzir o tempo de download.

O algoritmo omite vários detalhes menores. Por exemplo, o HTTP permite a um site Web incluir um cabeçalho *sem cache* que especifica que um dado item não deveria ser armazenado no cache. Além disso, os navegadores não armazenam no cache pequenos itens, porque mantê-los pode aumentar o tempo de pesquisa no cache e o tempo de carga do item pequeno com uma requisição GET é aproximadamente o mesmo necessário para fazer uma requisição HEAD.

4.9 Arquitetura do navegador

Um navegador é complexo porque fornece uma grande quantidade de serviços gerais e suporta uma interface gráfica complexa capaz de atender aos referidos serviços. É claro que um navegador deve entender o HTTP, mas, também, fornecer suporte para outros protocolos. Em particular, como uma URL pode especificar um determinado protocolo, um navegador deve conter o código-cliente capaz de tratar cada um dos protocolos utilizados. Para cada serviço, o navegador deve saber como interagir com um servidor e como interpretar as respostas. Por exemplo, um navegador deve saber como acessar o serviço FTP discutido na próxima seção. A Figura 4.9 ilustra os componentes que compõem um navegador.

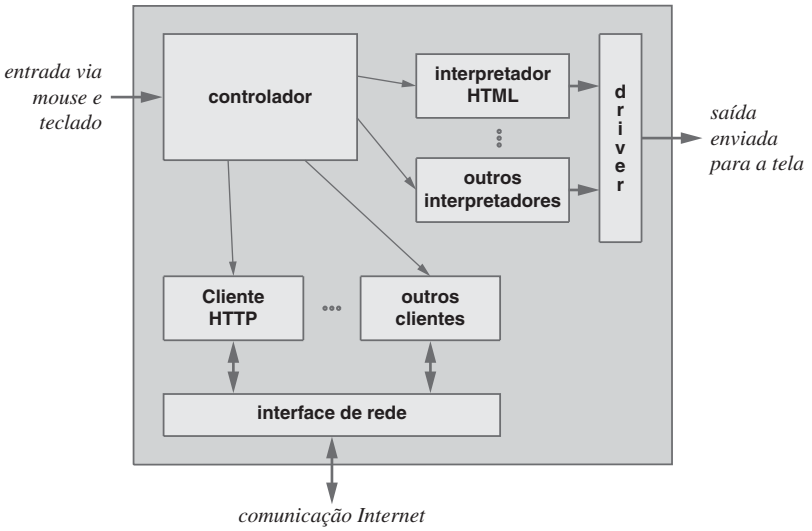


Figura 4.9 Arquitetura de um navegador que pode ter acesso a múltiplos serviços.

4.10 File Transfer Protocol (FTP)

Um *arquivo* é uma abstração fundamental de armazenamento. Como um arquivo pode conter um objeto arbitrário (isto é, um documento, um programa de computador, uma imagem gráfica ou um videoclipe), a facilidade de enviá-lo de um computador para outro é um mecanismo muito poderoso para a troca de dados. O termo *transferência de arquivo* é usado para tal serviço.

A *transferência de arquivo* através da Internet é complicada porque os computadores são heterogêneos, o que significa que cada sistema de computador tem que possuir as representações dos arquivos, a informação de tipo, a nomeação e os mecanismos de acesso ao arquivo.

Em alguns sistemas de computador, a extensão .jpg é usada para uma imagem JPEG e, em outros, é utilizada a extensão *.jpeg*. Em alguns sistemas, cada linha em um arquivo de texto é encerrada por um único caractere *LINEFEED*, enquanto outros sistemas exigem o caractere *CARRIAGE RETURN* seguido do *LINEFEED*. Alguns sistemas utilizam uma barra (/) como um separador em nomes de arquivos, e outros usam uma

barra invertida (\). Além disso, um sistema operacional pode definir um conjunto de contas de usuário que especificam o direito de acesso a arquivos determinados. No entanto, a informação de conta é diferente entre os computadores, de modo que o usuário X em um computador não é o mesmo que o usuário *X* no outro.

O serviço padrão de *transferência de arquivos* na Internet usa o *File Transfer Protocol* (FTP). O FTP pode ser caracterizado a partir destes aspectos:

- *Conteúdos arbitrários de arquivo*. O FTP pode transferir qualquer tipo de dados, incluindo documentos, imagens, músicas ou vídeos armazenados.
- *Transferência bidirecional*. O FTP pode ser usado para baixar (download) arquivos (transferir do servidor para o cliente) ou carregar (upload) arquivos (transferir do cliente para o servidor).
- *Suporte para autenticação e propriedade*. O FTP permite que cada arquivo tenha propriedade e restrições de acesso.
- *Habilidade para navegar em pastas.* O FTP permite que um cliente navegue nos conteúdos de um diretório (ou seja, uma pasta).
- *Mensagens de controle textual.* Como muitos outros serviços aplicativos da Internet, as mensagens de controle trocadas entre um cliente e um servidor FTP são enviadas como texto ASCII.
- *Acomodação de heterogeneidade.* O FTP esconde os detalhes dos sistemas operacionais dos computadores individuais e pode transferir uma cópia de um arquivo entre dois computadores quaisquer.

Como poucos usuários lançam uma aplicação FTP, o protocolo é geralmente invisível. No entanto, ele costuma ser chamado automaticamente pelo navegador quando um usuário solicita uma *transferência* de arquivo.

4.11 O paradigma de comunicação FTP

Um dos aspectos mais interessantes do FTP decorre da maneira com que um cliente interage com o servidor. No geral, a abordagem parece simples: um cliente estabelece uma conexão com um servidor de FTP e envia uma série de pedidos para que o servidor responda. Ao contrário do HTTP, um servidor FTP não envia respostas pela mesma conexão em que o cliente envia solicitações. Em vez disso, a conexão que o cliente cria, chamada de *conexão de controle*, é reservada para os comandos. Cada vez que o servidor precisa baixar ou carregar um arquivo, ele (não o cliente) abre uma nova conexão. Para distingui-las da conexão de controle, as conexões usadas para transferir arquivos são chamadas de *conexões de dados*.

Surpreendentemente, o FTP inverte o relacionamento cliente-servidor para conexões de dados. Isto é, quando abre uma conexão de dados, o cliente age como se fosse um servidor (ou seja, espera pela conexão de dados) e o servidor age como se fosse um cliente (ou seja, inicia a conexão de dados). Depois de ter sido utilizada para uma transferência, a conexão de dados é fechada. Se o cliente envia outro pedido, o servidor abre uma nova conexão de dados. A Figura 4.10 ilustra a interação, mas omite vários detalhes importantes. Por exemplo, depois de criar a conexão de controle, um cliente deve se conectar ao servidor por meio do envio de um login e de uma senha; um *login anônimo* com senha de *convidado* é usado para obter os arquivos que

são públicos. Um servidor envia um status numérico através da conexão de controle como uma resposta a cada pedido, inclusive o do login; a resposta permite que o cliente saiba se o pedido era válido.

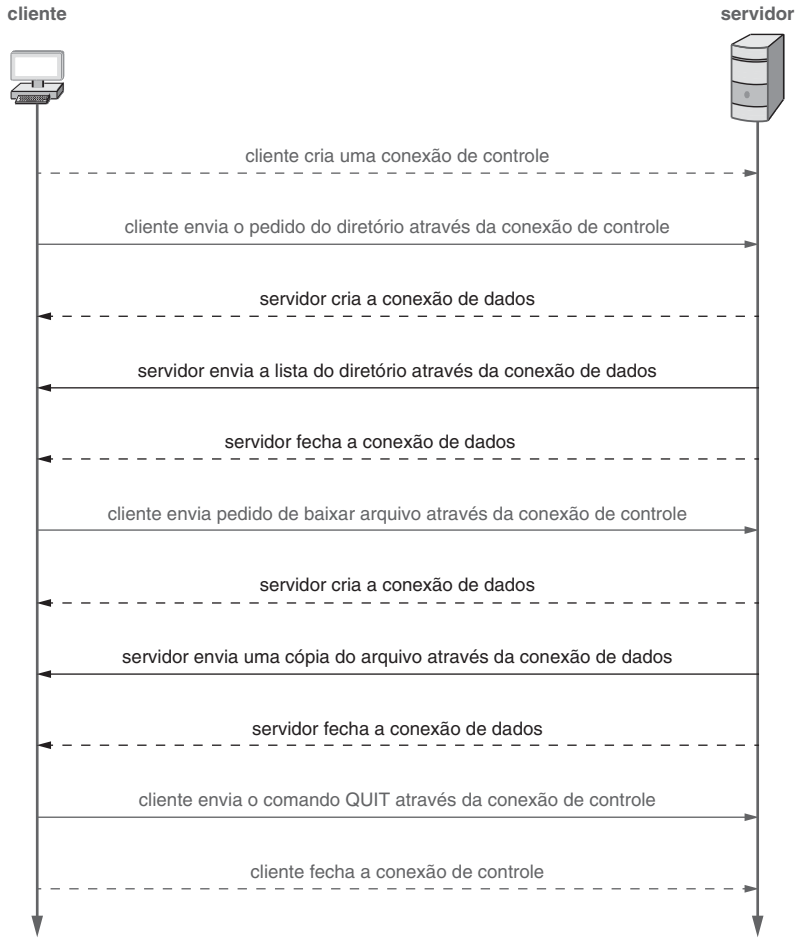


Figura 4.10 Ilustração da conexão FTP durante uma sessão típica.

Outro detalhe interessante diz respeito aos números das portas de protocolo usadas. Nesse sentido, surge a pergunta: o que o número de porta de protocolo deve especificar para que um servidor se conecte ao cliente? O FTP permite que o cliente decida: antes de fazer um pedido para o servidor, um cliente aloca uma porta de protocolo em seu sistema operacional local e envia o número da porta para o servidor. Ou seja, o cliente amarra a porta para aguardar uma conexão e depois transmite o número da porta através da conexão de controle como uma sequência de dígitos decimais. O servidor lê o número e segue os passos que o Algoritmo 4.2 especifica.

A transmissão de informações de porta entre um par de aplicações pode parecer inócua, mas não é, e a técnica não funciona bem em todas as situações. Em particular,

Algoritmo 4.2

Dada:
  Uma conexão de controle FTP
Conseguir:
  Uma transferência de um arquivo através de uma conexão TCP
Método:
  Cliente envia uma solicitação para um arquivo específico através de uma conexão de controle;
  Cliente atribui uma porta de protocolo local, chama-a de X e se liga a ela;
  Cliente envia "PORT X" para o servidor através de uma conexão de controle;
  Cliente espera para aceitar uma conexão de dados na "PORT X";
  Servidor recebe comando "PORT X" e extrai o número X;
  Temporariamente o servidor assume o papel de cliente e o servidor cria
    uma conexão TCP para a porta X no computador do cliente;
  Temporariamente assumindo o papel de um servidor,
    o cliente aceita a conexão TCP (chamada de "conexão de dados");
  Servidor envia o arquivo solicitado pela conexão de dados;
  Servidor fecha a conexão de dados;

Algoritmo 4.2 Passos realizados pelo cliente e pelo servidor de FTP para transferência de um arquivo.

a transmissão de um número de porta de protocolo irá falhar se um dos dois terminais estiver por trás de um dispositivo *Network Address Translation* (NAT), como um roteador sem fio usado em uma residência ou em um pequeno escritório. O Capítulo 23 explica que o FTP é uma exceção – para suportar um FTP, um dispositivo NAT reconhece uma conexão de controle FTP, inspeciona o conteúdo da conexão e reescreve os valores em um comando PORT.

4.12 Mensagem eletrônica

Embora os serviços de *mensagens instantâneas* tenham se tornado populares, o e-mail continua a ser um dos aplicativos mais usados na Internet. Como esse serviço foi concebido antes dos computadores pessoais e dos PDAs estarem disponíveis, ele foi projetado para permitir que um usuário em um computador envie uma mensagem diretamente para outro usuário em outro computador. A Figura 4.11 ilustra a arquitetura original e o Algoritmo 4.3 lista os passos realizados.

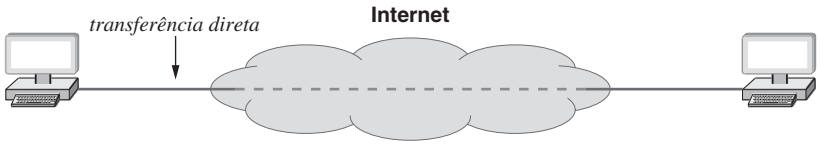


Figura 4.11 A configuração original do e-mail com transferência direta de um computador remetente para um computador receptor.

Algoritmo 4.3

Dada:
Comunicação de e-mail de um usuário para outro
Fornecer:
A transmissão de uma mensagem para o destinatário pretendido
Método:
Usuário invoca aplicação de interface e gera uma mensagem de e-mail para o *user x@destination.com*;
Interface de e-mail do usuário passa a mensagem de pedido de transferência de mensagem;
Aplicação de transferência de mensagem se torna um cliente e abre uma conexão TCP para o *destination.com*;
Aplicação de transferência de mensagem utiliza o protocolo SMTP para transferir a mensagem e em seguida fecha a conexão;
O servidor de mensagens no *destination.com* recebe a mensagem e coloca uma cópia na caixa de entrada do *user x*;
O *user x* no *destination.com* roda a aplicação da interface de mensagens para exibir a mensagem;

Algoritmo 4.3 Passos para enviar e-mail no paradigma inicial.

Como o algoritmo 4.3 indica, mesmo o software original de e-mail foi dividido conceitualmente em duas partes separadas:

- Uma aplicação de interface de mensagem
- Uma aplicação de transferência de mensagem

Um usuário chama um *aplicativo de mensagem* diretamente. A interface desse aplicativo fornece mecanismos que permitem que um usuário componha e edite mensagens de saída, bem como leia e processe mensagens recebidas. Um aplicativo de mensagens não age como um cliente ou um servidor e não transfere mensagens para outros usuários. Em vez disso, ele lê mensagens da *caixa postal* do usuário (ou seja, um arquivo no computador do usuário) e passa mensagens adiante para um *aplicativo de transferência de mensagens*. O aplicativo de transferência de mensagens age como um cliente para enviar cada mensagem de e-mail para o seu destino. Além disso, ele também atua como um servidor para aceitar as mensagens recebidas e armazenar cada uma delas na caixa de correio do usuário apropriado.

As normas de protocolo usadas para mensagens na Internet podem ser divididas em três tipos amplos, como a Figura 4.12 descreve

Tipo	Descrição
Transferência	Um protocolo usado para mover a cópia de uma mensagem de um computador para outro
Acesso	Um protocolo que permite ao usuário ter acesso à sua caixa de correio para ler ou enviar mensagens
Representação	Um protocolo que especifica o formato da mensagem quando armazenada no disco

Figura 4.12 Os três tipos de protocolos usados com mensagens.

4.13 O Simple Mail Transfer Protocol (SMTP)

O *Simple Mail Transfer Protocol* (SMTP) é o protocolo padrão que o programa de transferência de mensagens usa para transferir uma mensagem para um servidor por meio da Internet. O SMTP pode ser caracterizado assim:

- Segue o paradigma stream
- Usa mensagem de controle textual
- Transfere somente mensagem de texto
- Permite a um remetente especificar os nomes dos destinatários e checar cada um deles
- Envia uma cópia de uma dada mensagem

O aspecto mais inesperado do SMTP decorre da sua restrição às mensagens textuais. Uma seção posterior explica o padrão MIME dos anexos permitidos na mensagem, tais como imagens gráficas ou arquivos binários, mas o mecanismo do SMTP é restrito ao texto.

O segundo aspecto do SMTP é sua capacidade de enviar uma única mensagem para vários destinatários em um determinado computador. O protocolo permite que um cliente liste os usuários um a um e em seguida envie uma única cópia de uma mensagem para todos os usuários da lista. Ou seja, um cliente envia "Eu tenho uma mensagem de correio para o usuário A" e o servidor responde "OK" ou "O usuário não existe aqui". Na verdade, cada mensagem do servidor SMTP começa com um código numérico; assim, respostas são escritas da forma "250 OK" ou "550 O usuário não existe aqui". A Figura 4.13 mostra um exemplo de uma sessão SMTP que ocorre quando uma mensagem de correio é transferida do usuário *John_Q_Smith* no computador *example.edu* para dois usuários no computador *somewhere.com*.

Na figura, cada linha é marcada como *Cliente:* ou *Servidor:* para indicar se é o servidor ou o cliente que a envia; o protocolo não inclui os rótulos em itálico. O comando *HELO* permite ao cliente se autoautenticar por meio do envio de seu nome de domínio. Por fim, a notação <*CR*> <*LF*> denota um CARRIAGE RETURN seguido de um LINEFEED (ou seja, um fim-de-linha). Assim, o corpo de uma mensagem de e-mail é finalizado por uma linha que consiste em um período com nenhum outro texto ou espacejamento.

O termo *simple* no nome indica que o SMTP é simplificado. Como o antecessor do SMTP era muito complexo, os projetistas eliminaram os recursos desnecessários e concentraram-se no básico.

```
Server:   220 somewhere.com Simple Mail Transfer Service Ready

Client:   HELO example.edu
Server:   250 OK

Client:   MAIL FROM:<John_Q_Smith@example.edu>
Server:   250 OK

Client:   RCPT TO:<Matthew_Doe@somewhere.com>
Server:   550 No such user here

Client:   RCPT TO:<Paul_Jones@somewhere.com>
Server:   250 OK

Client:   DATA
Server:   354 Start mail input; end with <CR><LF>.<CR><LF>
Client:   ...sends body of mail message, which can contain
Client:   ...arbitrarily many lines of text
Client:   <CR><LF>.<CR><LF>
Server:   250 OK

Client:   QUIT
Server:   221 somewhere.com closing transmission channel
```

Figura 4.13 Um exemplo de sessão SMTP.

4.14 ISPs, servidores de mensagens e acesso às mensagens

À medida que a Internet se expandiu para incluir novos consumidores, um novo paradigma surgiu por meio do e-mail. Como a maioria dos usuários não deixa o seu computador funcionando de forma contínua e não sabe como configurar e gerenciar um servidor de e-mail, os ISPs começaram a oferecer serviços de e-mail. Em essência, um ISP executa um servidor de e-mail e fornece uma caixa de correio para cada assinante. Em vez de fornecer um software de e-mail tradicional, cada ISP oferece um software de interface que permite ao usuário acessar sua caixa de correio. A Figura 4.14 ilustra o arranjo.

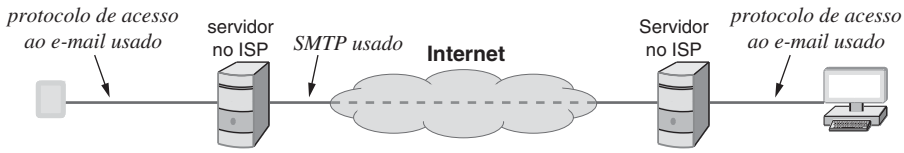


Figura 4.14 Uma configuração de e-mail onde um ISP roda um servidor de e-mail e fornece acesso à caixa de correio do usuário.

O acesso ao e-mail ocorre de uma das duas formas:

- Um aplicativo de interface de e-mail de propósito especial
- Um navegador Web que acessa uma página Web de e-mail

Aplicativos de interface de propósito especial são usados tipicamente em dispositivos móveis, como tablets ou smartphones. Devido ao fato de eles conhecerem o tamanho da tela e a capacidade do dispositivo, esses aplicativos podem mostrar o e-mail no formato apropriado para o referido dispositivo. Outra vantagem de usar um aplicativo de e-mail especial está relacionada com a habilidade de baixar toda a caixa de correio no dispositivo local. Baixá-la toda é particularmente importante se o usuário móvel

pretende ficar desconectado da rede, pois isso permite a ele processar as mensagens mesmo que esteja sem conexão com a Internet (por exemplo, durante um vôo). Uma vez que a conectividade com a Internet seja estabelecida, o aplicativo se encarrega da comunicação com o servidor no ISP do usuário para carregar as mensagens redigidas e para baixar novas mensagens que podem ter chegado na caixa de correio dele.

Usar um navegador Web como uma interface de e-mail é natural: um ISP fornece uma página Web especial que mostra as mensagens da caixa de correio do usuário. Dessa forma, o usuário lança um navegador Web padrão e acessa o servidor de e-mail no ISP. A página Web inicial chama um mecanismo de autenticação que pergunta ao usuário login e senha; o servidor Web usa o login do usuário para selecionar sua caixa de correio. O servidor Web recupera as mensagens da caixa de correio, gera uma página HTML que lista as mensagens e retorna a página para o navegador do usuário. A principal vantagem de usar uma página Web para e-mail é a facilidade de ler mensagens de qualquer computador ou dispositivo – o usuário não necessita de um dispositivo particular nem precisa rodar um aplicativo de interface de e-mail especial. Assim, um usuário em viagem pode ter acesso às suas mensagens por meio do computador na recepção de um hotel.

4.15 Protocolos de acesso ao e-mail (POP, IMAP)

Diferentes protocolos têm sido criados para fornecer *acesso* ao e-mail. Um protocolo de acesso é distinto de um protocolo de transferência porque um acesso envolve somente um usuário interagindo com sua caixa de correio, enquanto uma transferência envolve o envio de mensagens de um usuário qualquer em um computador para uma caixa de correio qualquer em outro computador. Protocolos de acesso têm as seguintes características:

- Fornecem acesso à caixa de correio do usuário
- Permitem ao usuário ver cabeçalhos, baixar, excluir ou enviar mensagens individuais
- Rodam o cliente no computador ou no dispositivo do usuário
- Rodam o servidor no computador onde a caixa de correio está armazenada

A possibilidade de ver a lista de mensagens sem baixar os conteúdos delas é especialmente útil nos casos em que a conexão entre o usuário e o servidor de e-mail é lenta. Por exemplo, um usuário navegando através de um telefone celular pode olhar os cabeçalhos e deletar os spams sem esperar para baixar os conteúdos dessas mensagens.

Diversos mecanismos têm sido propostos para acesso ao e-mail. Alguns ISPs fornecem software de acesso livre aos seus assinantes. Além disso, dois padrões de protocolos de acesso ao e-mail foram criados; a Figura 4.15 lista esses padrões:

Acrônimo	Expansão
POP3	*Post Office Protocol version 3*
IMAP	*Internet Mail Access Protocol*

Figura 4.15 Os dois protocolos padrão de acesso ao e-mail.

Embora ofereçam os mesmos serviços básicos, os dois protocolos diferem em muitos detalhes. Cada um fornece seu próprio mecanismo de autenticação que um usuário segue para ele mesmo. A autenticação é necessária para garantir que um usuário não acesse a caixa de correio de outro usuário.

4.16 Padrões de representação de e-mail (RFC2822, MIME)

Duas representações de e-mail foram padronizadas:

- RFC2822 *Mail Message Format*
- *Multi-purpose Internet Mail Extensions* (MIME)

RFC2822 Mail Message Format. O formato padrão de mensagem de e-mail retira seu nome do documento padrão do IETF (Internet Engineering Task Force), *Request For Comments 2822*. O formato é direto: uma mensagem de e-mail é representada como um arquivo de texto e consiste em uma seção *cabeçalho*, uma linha em branco e um *body*. As linhas do cabeçalho têm a forma:

Keyword: information

onde o conjunto de *keywords* é definido para incluir *From:*, *To:*, *Subject:*, *Cc:* e assim por diante. Além disso, linhas do cabeçalho que iniciam com a letra maiúscula X podem ser adicionadas sem afetar o processamento de e-mail. Assim, um e-mail pode incluir uma linha randômica no cabeçalho, tal como:

```
X-Worst-TV-Shows: any reality show
```

*Multi-purpose Internet Mail Extensions (*MIME*).* Lembre-se de que o SMTP pode transferir somente mensagens de texto. O padrão MIME estende a funcionalidade de e-mail para permitir a transferência de mensagens de dados não textuais. O MIME especifica como um arquivo binário pode ser codificado dentro de caracteres imprimíveis, incluídos na mensagem, e decodificado pelo destinatário.

Embora isso tenha introduzido o padrão de codificação *Base64* que tem se tornado popular, o MIME não restringe a codificação para uma forma específica. Em vez disso, permite ao remetente e ao destinatário escolher uma codificação que seja mais conveniente. Para especificar o uso de uma codificação, o remetente inclui linhas adicionais no cabeçalho da mensagem. Além disso, o MIME permite que um remetente divida a mensagem em várias partes e especifique uma codificação para cada parte independentemente. Assim, com o MIME, um usuário pode enviar uma mensagem textual e anexar imagens gráficas, uma planilha e um clipe de áudio, cada um com sua própria codificação. O sistema de e-mail destinatário pode decidir como processar os anexos (ou seja, se salva uma cópia no disco ou mostra uma cópia).

De fato, o MIME adiciona duas linhas no cabeçalho: uma para declarar que o MIME foi utilizado para criar a mensagem e outra para especificar como a informação MIME está incluída no *body*. Por exemplo, as linhas do cabeçalho:

```
MIME-Version: 1.0
Content-Type: Multipart/Mixed; Boundary=Mime_separator
```

especificam que a mensagem foi composta usando a versão *1.0* do MIME e que a linha contendo *Mime_separator* aparecerá no corpo antes de cada parte da mensagem.

Quando o MIME é usado para enviar uma mensagem de texto padrão, a segunda linha muda para:

```
Content-Type: text/plain
```

O MIME é compatível com os sistemas de e-mail que não entendem o padrão MIME ou a codificação. É claro que tais sistemas não têm nenhuma maneira de extrair anexos não textuais – eles tratam o corpo como um único bloco de texto. Para resumir:

> *O padrão MIME insere linhas extras de cabeçalho para permitir que anexos não textuais possam ser enviados dentro de uma mensagem de e-mail. Um anexo é codificado como letras imprimíveis e uma linha separadora aparece antes de cada anexo.*

4.17 Domain Name System (DNS)

O *Domain Name System* (DNS) fornece um serviço que mapeia nomes simbólicos legíveis por seres humanos para endereços de computadores. Navegadores, softwares de e-mail e a maioria dos outros aplicativos da Internet usam o DNS. O sistema fornece um interessante exemplo de interação cliente-servidor, pois o mapeamento não é executado por um simples servidor. Em vez disso, a informação de nomeação é distribuída, por meio da Internet, entre vários servidores localizados em sites. Sempre que um aplicativo precisa traduzir um nome, ele torna-se cliente do sistema de nomeação. O cliente envia uma mensagem de requisição para um servidor de nome, que encontra o endereço correspondente e envia uma mensagem de resposta. Se ele não consegue responder à requisição, um servidor de nome torna-se temporariamente cliente de um outro servidor de nome, até que seja encontrado um servidor que pode responder à requisição.

Sintaticamente, cada nome consiste em uma sequência de segmentos alfanuméricos separados por pontos. Por exemplo, um computador na Purdue University tem o nome de domínio:

mymail.purdue.edu

e um computador na Google tem o nome de domínio:

```
gmail.google.com
```

Os nomes de domínio são hierárquicos, com a parte mais significativa do nome na direita. O segmento mais à esquerda do nome (*mymail* e *gmail* nos exemplos) é o nome do computador individual. Outros segmentos no domínio de nome identificam o grupo que possue o nome. Por exemplo, o segmento *purdue* dá o nome da universidade e *google* dá o nome da companhia. O DNS não especifica o número de segmentos no nome. Em vez disso, cada organização pode escolher quantos segmentos usar nos seus computadores e o que cada segmento representa.

O *domain name system* especifica valores para o segmento mais significativo, que é chamado de *domínio de nível superior* (TLD, *Top-Level Domain*). Os domínios de nível superior são controlados pela *Internet Corporation for Assigned Names and Numbers* (ICANN), que designa um ou mais registros de domínio para administrar um determinado domínio de nível superior e aprovar nomes específicos. Alguns TLDs são

genéricos, o que significa que estão geralmente disponíveis. Outros TLDs são restritos a grupos específicos ou a agências governamentais. A Figura 4.16 lista exemplos de domínios DNS de nível superior.

Uma organização escolhe um nome de acordo com um dos domínios de nível superior existentes. Por exemplo, a maioria das corporações dos EUA opta por se registrar sob o domínio *com*. Assim, uma empresa chamada *Foobar* pode solicitar o domínio *foobar* sob o domínio de nível superior *com*. Uma vez que o pedido for aprovado, será atribuído à empresa Foobar o domínio:

foobar.com

Se o nome já tiver sido atribuído a outra organização chamada Foobar, é possível utilizar *foobar.biz* ou *foobar.org*, mas não *foobar.com*. Além disso, uma vez que *foobar.com* tenha sido atribuído, a empresa Foobar pode escolher quantos níveis adicionais sejam necessários para adicionar significado aos seus computadores. Assim, se a Foobar tem filiais na costa Leste e Oeste, pode utilizar nomes como:

computer1.east-coast.foobar.com

Nome domínio	Atribuído
aero	Setor de transporte aéreo
arpa	Domínio de infraestrutura
asia	Para ou sobre a Ásia
biz	Negócios
com	Organizações comerciais
coop	Associações cooperativas
edu	Instituições educacionais
gov	Governos
info	Informação
int	Organizações internacionais
jobs	Gerentes de recursos humanos
mil	Domínio militar
mobi	Provedores de conteúdo mobile
museum	Museus
name	Indivíduos
net	Grandes centros de suporte de rede
org	Organizações não comerciais
pro	Profissionais credenciados
travel	Viagem e turismo
country code	Uma nação soberana

Figura 4.16 Exemplos de domínios de nível superior e o grupo ao qual cada um é atribuído.

ou pode escolher uma hierarquia de nomeação plana com todos os computadores identificados pelo nome seguido do nome do domínio da empresa:

computer1.foobar.com

Além da estrutura organizacional familiar, o DNS permite que as organizações usem um registro geográfico. Por exemplo, a organização Corporation For National Research Initiatives registrou o domínio:

cnri.reston.va.us

devido ao fato de a organização estar localizada na torre de Reston, Virgínia, nos Estados Unidos. Assim, os nomes dos computadores na organização terminam com *.us* em vez de *.com*.

Alguns países estrangeiros têm adotado uma combinação de nomes de domínios geográficos e organizacionais. Por exemplo, universidades na Inglaterra registram sob o domínio:

ac.uk

onde *ac* é uma abreviação para *academic* e *uk* é o código oficial para *United Kingdom*.

4.18 Nomes de domínios que começam com um nome de serviço

Muitas organizações utilizam nomes de domínio que refletem o serviço que um computador fornece. Por exemplo, um computador que executa um servidor para o *file transfer protocol* pode ser chamado:

ftp.foobar.com

De forma similar, um computador que roda um servidor Web pode ser chamado:

www.foobar.com

Tais nomes são mnemônicos, mas não são obrigatórios. Em particular, o uso de www para nomear computadores que rodam um servidor Web é meramente uma convenção – um computador qualquer pode rodar um servidor Web, mesmo que o nome do domínio do computador não contenha www. Além dissso, um computador que tem um nome de domínio começando com *www* não precisa necessariamente rodar um servidor Web. Em síntese:

> *Usar o primeiro segmento no nome do domínio para indicar um serviço (ou seja, www) é meramente uma convenção para ajudar os humanos.*

4.19 A hierarquia do DNS e o modelo servidor

Uma das principais características do *domain name system* é a autonomia – o sistema é projetado para permitir que cada organização possa atribuir nomes a computadores ou alterar esses nomes sem informar a uma autoridade central. Para colocar

em prática a autonomia, cada organização é autorizada a operar servidores de DNS para a sua parte da hierarquia. Assim, a universidade Purdue opera um servidor de nomes que terminam em *purdue.edu* e a empresa IBM opera um servidor de nomes que terminam em *ibm.com*. Cada servidor DNS contém informações que o ligam a outros servidores de nomes de domínio para cima e para baixo na hierarquia. Além disso, um dado servidor pode ser *replicado*, de modo que existam várias cópias físicas dele. A replicação é especialmente útil para servidores muito utilizados, como os *servidores raiz* que fornecem informações sobre domínios de nível superior, porque um único servidor não poderia lidar com a carga. Nesses casos, os administradores devem garantir que todas as cópias sejam coordenadas para fornecer exatamente a mesma informação.

Cada organização é livre para escolher os detalhes de seus servidores. Uma pequena organização que tem apenas alguns computadores pode contratar um provedor para executar um servidor DNS em seu nome. Uma grande organização que gere o seu próprio servidor pode optar por colocar todos os nomes para a organização em um único servidor físico, ou pode optar por dividir os seus nomes entre vários servidores. A divisão pode coincidir com a estrutura organizacional (por exemplo, nomes para uma subsidiária podem estar em um servidor separado) ou com a estrutura geográfica (por exemplo, um servidor separado para cada site da empresa). A Figura 4.17 ilustra como a hipotética empresa Foobar poderia organizar sua estrutura de servidores se possuísse uma divisão de doces e uma divisão de sabões.

4.20 Resolução de nome

A tradução de um nome de domínio em um endereço é chamada de resolução de nome, e dizemos que o nome é *resolvido* para um endereço. O software usado para executar a tradução é conhecido como *resolvedor de nome* (ou simplesmente *resolvedor*). Na API socket, por exemplo, o resolvedor é invocado chamando a função *gethostbyname*. O resolvedor se torna um cliente, contata um servidor de DNS e retorna uma resposta para o chamador.

Cada resolvedor é configurado com o endereço de um ou mais *locais* de servidores de nome de domínio[3]. O resolvedor formula uma mensagem de *solicitação de DNS*, envia a mensagem para o servidor local e aguarda que ele envie uma mensagem de *DNS com a resposta*. Um resolvedor pode optar por utilizar ou uma cadeia de caracteres ou um paradigma de mensagem ao se comunicar com um servidor DNS; a maioria dos resolvedores são configurados para usar um paradigma de mensagem porque ele impõe menos sobrecarga para uma pequena requisição.

Como um exemplo de resolução de nome, considere a hierarquia de DNS ilustrada na Figura 4.17 (a) e assuma que um computador na divisão *soap* solicita o nome *chocolate.candy.foobar.com*. O resolvedor será configurado para enviar o pedido para o servidor de DNS local (ou seja, o servidor para *foobar.com*). Embora ele não possa responder ao pedido, sabe entrar em contato com o servidor para *candy.foobar.com*, o que pode gerar uma resposta.

[3] A estratégia de contatar primeiro o servidor local ficará clara quando for discutido caching.

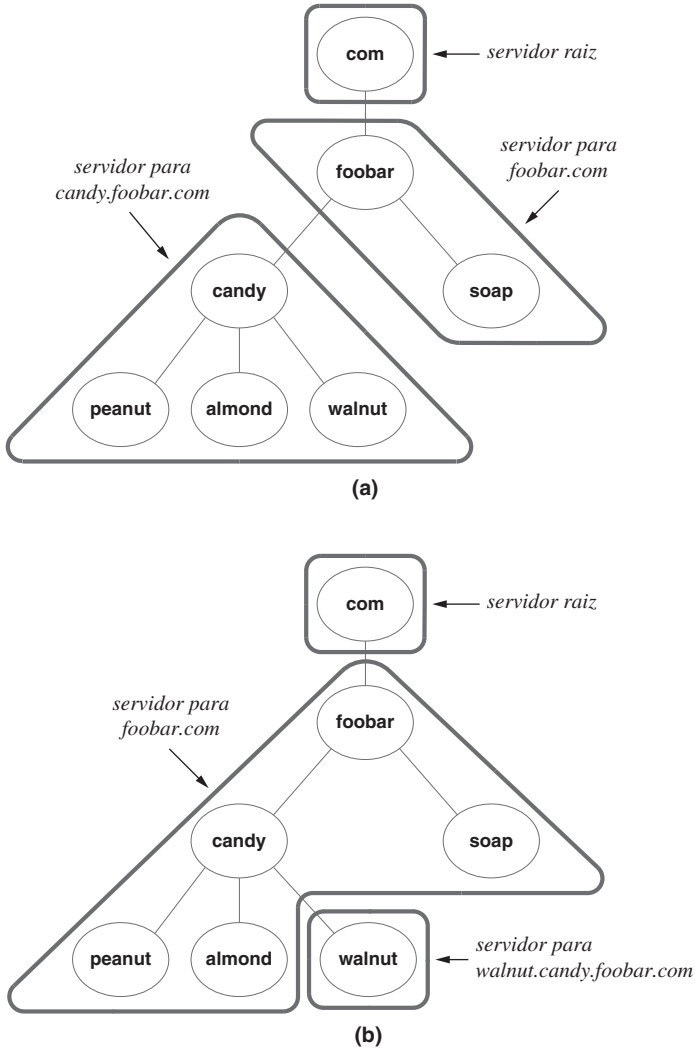


Figura 4.17 Uma hipotética hierarquia de DNS e duas possíveis atribuições de nomes aos servidores.

4.21 Caching nos servidores de DNS

O princípio da *referência de localidade* que é a base para fazer caching aplica-se ao *domain name system* de duas maneiras:

- Espacial: um usuário tende a procurar mais frequentemente os nomes dos computadores locais do que os nomes dos computadores remotos.
- Temporal: um usuário tende a procurar repetidamente o mesmo conjunto de nomes de domínio.

Como já foi visto anteriormente, o DNS explora a localidade espacial da seguinte maneira: um resolvedor de nomes primeiro contata um servidor. Para explorar a locali-

dade temporal, um servidor DNS armazena em cache todas as pesquisas. O Algoritmo 4.4 resume o processo.

De acordo com o algoritmo, quando chega um pedido para um nome de fora do conjunto para o qual o servidor é uma autoridade, ocorrem mais interações cliente-servidor. O servidor torna-se temporariamente um cliente de outro servidor de nomes. Quando o outro servidor retorna uma resposta, o servidor original armazena em cache a resposta e envia uma cópia dela de volta para o resolvedor que enviou o pedido. Assim, além de saber o endereço de todos os servidores que estão abaixo na hierarquia, cada servidor DNS deve saber o endereço de um servidor raiz.

A questão fundamental relacionada ao armazenamento em cache refere-se à duração de tempo que os itens devem ser armazenados – se um item é armazenado em cache por muito tempo, ele vai se tornar *obsoleto*. O DNS resolve o problema tomando medidas que permitem que um servidor com autoridade especifique um tempo limite de cache para cada item. Assim, quando um servidor local procura um nome, a resposta é composta de um *registro de recursos* (*resource record*) que especifica um tempo limite de cache tão bem quanto uma resposta. Sempre que um servidor armazena em cache uma resposta, ele respeita o tempo limite especificado no registro de recurso. Resumindo:

> *Como cada registro de recursos DNS gerado por um servidor com autoridade especifica um tempo limite de cache, um servidor DNS nunca retorna uma resposta obsoleta.*

Algoritmo 4.4

```
Dada:
   Uma mensagem de pedido de um resolvedor de nome DNS
Fornecer:
   Uma mensagem de resposta que contém o endereço
Método:
   Extrai o nome, N, do pedido;
   if (servidor é uma autoridade para N) {
        Forma e envia uma resposta para o solicitante;
   else if (resposta para N está no cache) {
        Forma e envia uma resposta para o solicitante;
   else {/ * Precisa procurar uma resposta * /
        if (servidor de autoridade para N é conhecido) {
             Envia pedido para servidor de autoridade;
        } else {
             Envia pedido para servidor raiz;
        }
        Recebe resposta e coloca no cache;
        Forma e envia uma resposta para o solicitante;
   }
```

Algoritmo 4.4 Passos realizados por um servidor de DNS para resolver um nome.

Armazenar DNS em cache não para com os servidores: um resolvedor também pode armazenar itens em cache. Na verdade, o software resolver na maioria dos sistemas de computador armazena em cache as respostas de pesquisas de DNS, o que significa que os pedidos sucessivos para o mesmo nome não necessitam usar a rede, porque o resolvedor pode satisfazer o pedido do cache no disco local do computador.

4.22 Tipos de entradas de DNS

Cada entrada em um banco de dados de DNS consiste em três itens: um nome de domínio, um *tipo* de registro e um valor. O tipo de registro especifica como o valor deve ser interpretado (ou seja, que o valor é um endreço IPv4). Mais importante, uma consulta enviada para um servidor DNS especifica um nome de domínio e um tipo; o servidor só retorna uma ligação que corresponde ao tipo de consulta.

Quando um aplicativo precisa de um endereço IP, o navegador especifica o tipo A (IPv4) ou o tipo *AAAA* (IPv6). Um aplicativo de e-mail usando SMTP procura um nome específico de domínio do tipo de *MX*, que solicita um *servidor de mensagens*. A resposta que um servidor retorna corresponde ao tipo solicitado. Assim, um sistema de e-mail irá receber uma resposta que corresponde ao tipo *MX*, e um navegador receberá uma resposta que corresponde ao tipo A ou *AAAA*. O ponto importante é:

> *Cada entrada em um servidor DNS tem um tipo. Quando um resolvedor procura um nome, ele especifica o tipo desejado, e o servidor DNS retorna apenas as entradas que correspondem ao tipo especificado.*

O sistema tipo DNS pode produzir resultados inesperados porque o endereço retornado pode depender do tipo. Por exemplo, uma corporação pode decidir usar o nome *corporation.com* para os serviços de e-mail e Web. Com o DNS, a corporação consegue dividir a carga entre computadores separados por meio do mapeamento do tipo *A* para um computador e do tipo MX para um outro. A desvantagem desse esquema é que ele não é intuitivo para o ser humano – ele pode enviar e-mail para *corporation.com* mesmo que não seja possível ter acesso ao servidor Web nem fazer *ping* para o computador.

4.23 Registros dos recursos aliases e CNAME

O DNS oferece um tipo *CNAME* que é análogo a um link simbólico em um sistema de arquivo – a entrada fornece um *alias* para outra entrada. Para entender como *aliases* podem ser úteis, suponha que a corporação Foobar tem dois computadores chamados *charlie.foobar.com* e *lucy.foobar.com*. Suponha que no futuro ela decida rodar o servidor Web no computador *Lucy* e deseje seguir a convenção de usar o nome *www* para o computador que roda o servidor www da organização. Embora a organização possa renomear o computador *lucy*, existe uma solução muito mais fácil: criar uma entrada *CNAME* para *www.foobar.com* que aponta para *lucy*. Sempre que o resolvedor envia a requisição para *www.foobar.com*, o servidor retorna o endereço do computador *lucy*.

O uso de *aliases* é especialmente conveniente porque permite que uma organização substitua o computador usado para um determinado serviço sem alterar os nomes

ou os endereços dos computadores. Por exemplo, a organização Foobar pode mover seu serviço Web do computador *lucy* para o computador *charlie* alterando o servidor e o registro *CNAME* no servidor DNS – os dois computadores mantêm seus nomes originais e seus endereços IP. O uso de *aliases* também permite que uma organização associe vários *aliases* com um único computador. Assim, a organização Foobar pode rodar um servidor FTP e um servidor Web no mesmo computador e pode criar registros CNAME:

www.foobar.com

ftp.foobar.com

4.24 As abreviaturas e o DNS

O DNS não adota abreviaturas – um servidor só responde a um nome completo. No entanto, a maioria dos resolvedores pode ser configurado com um conjunto de sufixos que permite que um utilizador abrevie nomes. Por exemplo, cada resolvedor da organização Foobar pode ser programado para procurar um nome duas vezes: uma vez com nenhuma mudança e outra vez com o sufixo *foobar.com* anexado. Se o usuário digitar um nome de domínio completo, o servidor local irá retornar o endereço e o processamento continuará. Se o usuário digitar um nome abreviado, o resolvedor primeiro tentará resolver o nome e receberá um erro porque não existe tal denominação. O resolvedor tentará então adicionar um sufixo e procurar o nome resultante. Como o resolvedor é executado no computador pessoal de um usuário, este pode escolher a ordem em que os sufixos são tentados.

Naturalmente, permitir que cada usuário configure seu resolvedor para lidar com abreviaturas tem uma desvantagem: o nome que um determinado usuário digita pode diferir do nome que o outro usuário escreve. Assim, se um usuário comunica nomes para outro (por exemplo, envia um nome de domínio por meio de uma mensagem de e-mail), deve ter o cuidado de especificar os nomes completos e não as abreviaturas.

4.25 Nomes de domínio internacionais

Como usa o conjunto de caracteres ASCII, o DNS não pode armazenar nomes em alfabetos que não são representados em ASCII. Idiomas como russo, grego, chinês e japonês contêm, cada um deles, caracteres para os quais não existem representações em ASCII. Muitas línguas europeias utilizam sinais diacríticos que não podem ser representados em ASCII.

Durante anos, o IETF debateu modificações e extensões do DNS para acomodar nomes de domínios internacionais. Depois de considerar muitas propostas, ele escolheu uma abordagem conhecida como *Internationalizing Domain Names in Applications* (*IDNA*). Em vez de modificar o DNS, o IDNA usa ASCII para armazenar todos os nomes. Ou seja, quando é dado um nome de domínio que contém caracteres não representados em ASCII, o IDNA traduz o nome em uma seqüência de caracteres ASCII e armazena o resultado no DNS. Quando um usuário procura o nome, a mesma tradução é aplicada para converter o nome em uma string ASCII, e a cadeia ASCII resultante é colocada em uma consulta DNS. Essencialmente, o IDNA depende de aplicativos para

traduzir entre o conjunto de caracteres internacionais que o usuário vê e a forma ASCII interna usada no DNS.

As regras para a tradução internacional de nomes de domínios são complexas e utilizam o *Unicode*[4]. A tradução é aplicada a cada rótulo no nome do domínio e resulta em rótulos com este aspecto:

$$\texttt{xn--}\alpha\texttt{-}\beta$$

onde *xn--* é uma seqüência de quatro caracteres reservados que indica que o rótulo é um nome internacional, α é o subconjunto de caracteres da etiqueta original, que pode ser representado em ASCII, e β é uma seqüência de caracteres ASCII adicionais que diz a um aplicativo IDNA como inserir caracteres não ASCII em α para compor a versão para impressão da etiqueta.

As versões recentes dos navegadores mais utilizados, Firefox e Internet Explorer, podem aceitar e exibir nomes de domínio não ASCII porque eles implementam o IDNA. Se um aplicativo não implementa o IDNA, a saída pode parecer estranha para o usuário. Ou seja, quando um aplicativo que não implementa o IDNA exibe um nome de domínio internacional, o usuário vê a forma interna ilustrada acima, incluindo a seqüência inicial *xn--* e as partes seguintes α e β.

Para resumir:

> *O padrão IDNA para nomes de domínio internacionais codifica cada rótulo como uma seqüência de caracteres ASCII e conta com aplicativos para traduzir entre o conjunto de caracteres que um usuário espera e a versão codificada armazenada no DNS.*

4.26 Representações extensíveis (XML)

Os protocolos de aplicativos tradicionais abordados neste capítulo empregam uma representação fixa. Ou seja, o protocolo do aplicativo especifica um conjunto exato de mensagens que um cliente e um servidor podem trocar, bem como a forma exata de dados que acompanha a mensagem. A principal desvantagem de uma abordagem fixa é a dificuldade em fazer mudanças. Por exemplo, como os padrões de e-mail restringem o conteúdo da mensagem a textos, foi necessária uma grande mudança para adicionar extensões MIME.

A alternativa para uma representação fixa é um sistema extensível que permite que um remetente especifique o formato de dados. Um padrão para a representação extensível tem se tornado amplamente aceito: a *Extensible Markup Language* (XML). O XML é semelhante ao HTML no sentido de que ambas as linguagens encaixam tags num documento de texto. Ao contrário das do HTML, as tags em XML não são especificados *a priori* e não correspondem aos comandos de formatação. Em vez disso, o XML descreve a estrutura de dados e fornece nomes para cada campo. Tags em XML são bem equilibradas – cada ocorrência de uma tag <*X*> deve ser seguida por uma ocorrência de </*X*>. Além disso, como o XML não atribui qualquer significado para marcas, nomes

[4] O algoritmo de tradução usado para codificar uma etiqueta com caracteres não ASCII é conhecido como algoritmo *Puny* e a cadeia de caracteres resultante é conhecida como *Punycode*.

de marcas podem ser criados conforme a necessidade. Em particular, eles podem ser selecionados para facilitar a análise e o acesso aos dados. Por exemplo, se duas empresas concordam em trocar listas telefônicas corporativas, elas podem definir um formato XML que tem itens de dados, como o nome de um funcionário, o seu número de telefone e a referência ao seu escritório. As empresas podem optar por dividir ainda mais um nome em um sobrenome e um primeiro nome. A Figura 4.18 mostra um exemplo.

```
<ADDRESS>
    <NAME>
        <FIRST>  John    </FIRST>
        <LAST>   Public  </LAST>
    </NAME>
    <OFFICE> Room320      </OFFICE>
    <PHONE>  765-555-1234 </PHONE>
</ADDRESS>
```

Figura 4.18 Um exemplo de XML para a lista telefônica de uma empresa.

4.27 Resumo

Os protocolos da camada de aplicação, necessários para serviços padronizados, definem aspectos de representação e transferência de dados da comunicação. Os protocolos de representação utilizados com a World Wide Web incluem a *HyperText Markup Language* (HTML) e o padrão URL. O protocolo de transferência de Web, que é conhecido como *HyperText Transfer Protocol* (HTTP), especifica como um navegador se comunica com um servidor Web para baixar ou carregar o conteúdo. Para acelerar a carga, um navegador armazena em cache o conteúdo da página e usa um comando *HEAD* para solicitar informações sobre o status dela. Se a versão em cache permanece atualizada, o navegador a utiliza; caso contrário, ele dá proseguimento com um pedido GET para baixar uma nova cópia.

O HTTP utiliza mensagens textuais. Cada resposta a partir de um servidor começa com um cabeçalho que a descreve. As linhas no cabeçalho começam com um valor numérico, representado como dígitos ASCII, que informa o status (ou seja, se um pedido tem erro). Os dados que seguem o cabeçalho podem conter valores binários quaisquer.

O *File Transfer Protocol* (FTP) fornece grande carga de arquivo. Ele requer um cliente para entrar no sistema do servidor; o FTP suporta um login de convidado anônimo e uma senha para acesso ao arquivo público. O aspecto mais interessante do FTP provém de seu uso incomum de conexões. Um cliente estabelece um controle de conexão que é utilizado para enviar uma série de comandos. Sempre que um servidor necessita enviar dados (ou seja, uma carga de arquivo ou uma listagem de um diretório), o servidor age como um cliente e o cliente atua como um servidor. Isto é, o servidor inicia uma nova conexão de dados para o cliente. Uma vez que um único arquivo foi enviado, a conexão de dados é finalizada.

Três tipos de protocolos da camada aplicação são utilizados com correio eletrônico: transferência, representação e acesso. O *Simple Mail Transfer Protocol* (SMTP) serve como o padrão-chave de transferência; o SMTP só pode transferir uma mensagem textual. Existem dois padrões de representação para e-mail: o RFC2822 define o formato da mensagem com um cabeçalho e um corpo separados por uma linha em branco.

O padrão *Multi-purpose Internet Mail Extensions* (MIME) define um mecanismo para enviar arquivos binários como anexos em uma mensagem. O MIME insere linhas de cabeçalho extras que informam ao destinatário como interpretar a mensagem. O MIME requer um remetente para codificar um arquivo como texto para impressão.

Os protocolos de acesso a e-mail, como POP3 e IMAP, permitem que um usuário par acesse uma caixa postal. O acesso tornou-se popular porque um assinante pode permitir que um ISP execute um servidor de e-mail e mantenha a caixa postal do usuário.

O *Domain Name System* (DNS) fornece mapeamento automático dos nomes legíveis pelo ser humano para endereços de computadores. O DNS é composto de muitos servidores que controlam uma parte do espaço de nomes. Os servidores são dispostos em uma hierarquia e cada um deles conhece a localização dos outros na hierarquia.

O DNS usa o cache para manter a eficiência; quando um servidor com autoridade fornece uma resposta, cada servidor que transfere a resposta também armazena uma cópia em seu cache. Para evitar que as cópias se tornem obsoletas, o servidor especifica por quanto tempo o nome pode ser armazenado em cache.

Exercícios

4.1 Quais detalhes um protocolo de aplicação especifica?

4.2 Por que um protocolo para um serviço padronizado documentado independe de uma implementação?

4.3 Quais são os dois aspectos fundamentais de protocolos de aplicação e o que cada um inclue?

4.4 Cite exemplos de protocolos Web que ilustram cada um dos dois aspectos de um protocolo de aplicação.

4.5 Resuma as características do HTML.

4.6 Quais são as quatro partes de um URL e qual pontuação é usada para separá-las?

4.7 Quais são os quatro tipos de pedidos HTTP e quando cada um deles é usado?

4.8 Como um navegador sabe se uma solicitação HTTP é sintaticamente incorreta ou se o item referenciado não existe?

4.9 Quais objetos de dados um navegador armazena em cache e por que esse armazenamento é feito?

4.10 Liste os passos que um navegador necessita para determinar se deve usar o item de seu cache.

4.11 Um navegador pode usar outros protocolos de transferência além do HTTP? Explique.

4.12 Quando um usuário solicita uma lista de diretório FTP, quantas conexões TCP são estabelecidas? Explique.

4.13 Determine se a seguinte afirmação é verdadeira ou falsa: quando um usuário executa um aplicativo FTP, o aplicativo funciona como um cliente e como um servidor. Explique sua resposta.

4.14 Como um servidor FTP conhece o número da porta a ser usada em uma conexão de dados?

4.15 De acordo com o paradigma original de e-mail, um usuário pode receber um e-mail se o seu computador não executa um servidor de e-mail? Explique.

4.16 Liste os três tipos de protocolos usados com e-mail e descreva cada um deles.

4.17 Quais são as características do SMTP?

4.18 Um SMTP pode transferir uma mensagem que contenha uma pontuação (.) na própria linha? Por quê?

4.19 Onde fica um protocolo de acesso de e-mail usado?

4.20 Quais são os dois principais protocolos de acesso de e-mail?

4.21 Por que foi criado o MIME?

4.22 Qual o objetivo geral do *domain name system*?

4.23 Supondo que o ISO atribuiu N códigos de países, quantos domínios de nível superior existem?

4.24 Determine se a seguinte afirmação é verdadeira ou falsa: um servidor Web deve ter um nome de domínio que começa com www. Explique.

4.25 Determine se a seguinte afirmação é verdadeira ou falsa: uma companhia multinacional pode escolher dividir sua hierarquia de nomes de domínio de tal maneira que tenha um servidor de nomes na Europa, um na Ásia e outro na América do Norte.

4.26 Quando um servidor de nomes de domínio envia um pedido para um servidor de autoridade e quando ele responde sem enviar o pedido para um servidor de autoridade?

4.27 Determine se a seguinte afirmação é verdadeira ou falsa: se uma empresa muda seu servidor Web do computador *x* para o computador *y*, os nomes dos dois computadores devem mudar. Explique.

4.28 Determine se a seguinte afirmação é verdadeira ou falsa: um servidor DNS pode retornar um endereço IP diferente para um determinado nome, dependendo se a pesquisa especifica e-mail ou serviço Web. Explique.

4.29 O padrão IDNA exige mudanças em servidores de DNS? E em clientes DNS? Explique.

4.30 Pesquise na Web para obter informações sobre "pesquisa de DNS iterativa". Em que circunstâncias a pesquisa iterativa é usada?

4.31 Como o XML permite que um aplicativo especifique campos como um nome e um endereço?

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.