Redes de computadores e a Internet

Prof. Heitor Soares Ramos Filho
Colaboração: Bruno Lopes Vieira

20 de outubro de 2017

Sumário

Capítulo 1

Redes de computadores

Uma rede de computadores consiste de dois ou mais computadores interligados de forma a prover transferência de dados e compartilhamento de recursos. Assim, utilizando as tecnologias de rede, torna-se possível compartilhar uma impressora numa empresa, acessar páginas da *web*, acessar arquivos armazenados num único servidor disponível a todas as estações interligadas, entre outros serviços.

As redes de computadores permitem que o acesso a informação seja distribuído entre vários computadores, ao invés de concentrar todos os serviços e informações em um único computador. Diversas vantagens podem ser extraídas de um serviço distribuído. A possibilidade de trocar informações à longas distâncias possibilitou, por exemplo, a implementação de redes de caixas bancários eletrônicos, a troca de mensagens de texto pela internet (*e-mail* e *instant messengers*), utilização de jogos com outros usuários independente de onde estejam, acesso a diversas informações a partir de um aparelho celular.

Como forma de melhor compreender o funcionamento das redes de computadores, é imprescindível o estudo de sua taxonomia. A primeira etapa, então, seria classificá-las de acordo com algum critério lógico. De acordo com Tanenbaum [2003], as redes de computadores podem ser classificadas de acordo com o seu tamanho da seguinte forma:

LAN – Local Area Network: rede de pequeno porte limitada a um único prédio.

CAN – Campus Area Network: abrange um conjunto de prédios.

MAN – Meetropolitan Area Network: trata de uma rede espalhada por um centro metropolitano (por exemplo, uma loja em que

se conecta todas as filiais à matriz para troca de informações diversas).

WAN – Wide Area Network: integra várias localizações geograficamente separadas, como, por exemplo, cidades, estados, países e até mesmo continentes.

Uma outra abordagem de redes mais contemporânea inclui a categoria **PAN (Personal Area Network)**, que trata de redes de curto alcance em geral composta por dispositivos de acesso móvel com tecnologias como o `Bluetooth`, muito empregada em dispositivos de comunicação móvel como os aparelhos de celular (com seu uso pode-se trocar arquivos dentre aparelhos diversos, conectar o aparelho a um fone de ouvido sem fio etc.).

A interligação de computadores em rede é bastante complexa e diversos aspectos devem ser levados em consideração; desde os aspectos físicos relacionados às interfaces de redes e seus componentes elétricos e eletrônicos até às aplicações como transferências de arquivos e compartilhamento de impressoras, por exemplo. A taxonomia apresentada é bastante útil para facilitar organização das técnicas empregadas em redes de computadores. Por exemplo, uma rede local apresenta diversos elementos, tipicamente computadores, compartilhando um meio físico (cabos e ar, por exemplo); em uma rede de longa distância temos, tipicamente, uma conexão entre dois elementos, sendo que conexão pode ser utilizada para interligar duas redes locais. Nas redes locais as conexões são confiáveis e nas redes de longa distância frequentemente encontra-se erros de transmissão. Fica claro que as técnicas empregadas nas redes locais devem ser diferentes das técnicas empregadas em uma rede de longa distância.

Uma das técnicas utilizadas nos estudos de sistemas complexos é subdividir o problema, geralmente um problema grande e complexo, em problemas mais simples. Dessa forma, uma outra maneira de lidar com a complexidade intrínseca a interligação de computadores em rede é a subdivisão do tema em partes menores que somadas representam o problema. Para isto, é frequente a subdivisão em camadas apresentada a seguir.

1.1 Camadas

Para que se possa melhor compreender o funcionamento de uma rede de computadores, utiliza-se o exemplo proposto por Kurose and Ross [2005], transpondo a uma analogia humana.

Para se descrever um sistema de viagens aéreas, compreendendo sua estrutura, uma das maneiras é relacionar suas etapas: compra da passagem, *check-in*, acesso ao portão de embarque, embarque. A partir daí o avião toma sua rota para atingir o destino da viagem. Após a viagem, as etapas se invertem: desembarque, saída pelo portão de desembarque, *check-out* e, caso houvera algum problema durante a viagem, reclamação na empresa a qual foi efetuada a compra do bilhete (ver tabela 1.1).

Tabela 1.1: Analogia das passagens aéreas

↑	↓
Passagem (compra)	Passagem (reclamação)
Bagagem (*check-in*)	Bagagem (*check-out*)
Portão (entrada)	Portão (saída)
Avião (embarque)	Avião (desembarque)
Vôo	Vôo

Seguindo o exemplo, agora utilizando uma rede de computadores, temos que a ação de comprar uma passagem de avião para viajar até um destino seria equivalente a enviar um pacote de um computador a outro (em redes define-se "pacote" como uma pequena unidade de dados a ser transmitida pela rede; em vez de se transmitir toda a informação de uma só vez, essa é dividida em pequenas partes, sendo cada uma delas é denominada "pacote"). Ainda na analogia, observa-se que a cada etapa da ida, há uma correspondente na volta (pode-se observar na tabela 1.1 as correspondentes lado a lado).

Assim sendo, pode-se deduzir que há um mesmo "nível" a cada etapa, correpondedo, de acordo com o tipo do percursos (ida ou volta), uma função (ver tabela 1.2).

Tabela 1.2: Analogia em termos de função

↑		↓
compra	**Passagem**	reclamação
check-in)	**Bagagem**	*check-out*
entrada	**Portão**	saída
embarque	**Embarque**	desembarque
rota	**Vôo**	rota

Nesse exemplo, as etapas foram divididas em camadas, criando um esqueleto partir do qual se pode discutir as etapas de uma viagem aérea. Observa-se que cada uma dessas camadas possui uma função específica.

É dessa forma que um pacote trafega em uma rede. Ele desce de uma camada superior até a mais inferior e depois sobe novemente até a camada de origem, tendo cada uma sua função própria na transmissão. Numa rede de computadores como a Internet (`TCP/IP`), as camadas se organizam de acordo com a tabela 1.3.

Tabela 1.3: Camadas da Internet – rede `TCP/IP`

Aplicação	Comporta as aplicações de rede (`HTTP`, `FTP` etc.).
Transporte	Transferência de dados dentre dois nós (`TCP` e `UDP`).
Rede	Roteamento dos dados (`IP`).
Enlace	Transferência de dados dentre elementos vizinhos.
Física	Bits diretamente nos canais físicos.

Exemplificando, agora não mais em analogia e sim num exemplo prático, ao se abrir uma página em um navegador *web* (ex.: Mozilla Firefox, Internet Explorer etc.), a mensagem de solicitação fornecida à camada de Aplicação será será codificada em pacotes, os quais serão endereçados ao destinatário (computador que hospeda a página em questão) na camada de Transporte. Em seguida a camada de Rede traçará o caminho a ser percorrido pelos dados, que navegarão dentre os pontos da rede pela camada de Enlace utilizando-se dos meios físicos (cabos, ondas de rádio etc.) da camada Física.

Cada uma das camadas oferece um determinado serviço importante para a transmissão dos dados na rede. Para implementar esses serviços, os programas responsáveis por cada camada adicionam informações chamadas de cabeçalhos. Cada camada adiciona seu cabeçalho aos dados a serem transmitidos (carga útil) como ilustrado na Figura 1.1. No lado direito da figura é apresentado o nome que o pacote recebe em cada camada.

1.2 Protocolos

Uma questão pertinente é: como os aplicativos dentro de uma rede podem operar? Haja visto que uma rede de computadores pode ser totalmente heterogênea, não necessariamente utilizando as mesmas aplicações.

Para resolver esse problema, surgiu a idéia de criar os chamados protocolos. Eles são nada mais que um conjunto de formalidades para a troca de informações com a finalidade de melhorar o entendimento entre as partes que estão se comunicando. Imagine o procedimento efetuado ao perguntar que horas são a algum desconhecido na rua (ver tabela 1.4).

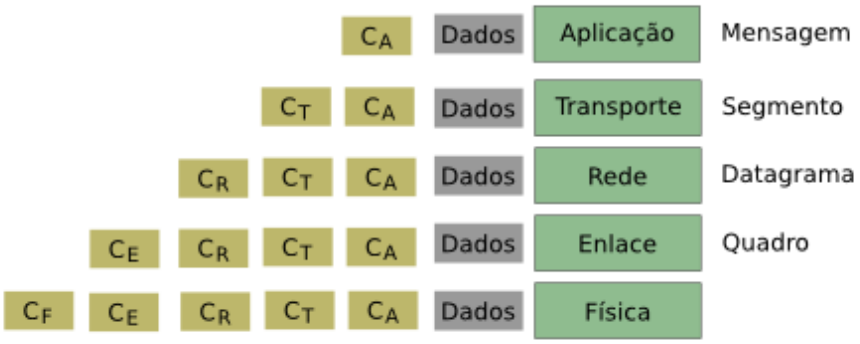

Figura 1.1: Encapsulamento em redes de pacotes

Tabela 1.4: "Regras" para solicitar informação de horário

Pessoa 1	**Pessoa 2**
Olá?	Olá!
Por favor, que horas são?	São 10 horas.
Obrigado!	Não há de que.

Da mesma forma, mas de maneira mais rígida, um computador segue um conjunto de normas predefinidas para efetuar alguma solicitação. Nos protocolos computacionais não pode haver mudança no formato das mensagens e nem na sequência em que elas aparecem. Por exemplo, para se utilizar o protocolo `FTP` (*File Transfer Protocol*), um protocolo para transferência de arquivos, as mensagens seriam conforme a tabela 1.5.

Tabela 1.5: Transação `FTP`

Cliente	**Servidor**
open server.ftp.br	Conectado a server.ftp.br
user foo	Senha solicitadapara foo.
minhasenha	usuário foo logado!
get arquivo.bin	Abrindo conexão de dados: <arquivo.bin>
bye	Goodbye!

re

Dessa forma o usuário inicia a seção, usando autenticação por *login* e senha, solicita a transferência do arquivo "arquivo1.bin" e o servidor o retorna o conteúdo do arquivo. Em seguida o usuário informa que não efetuará mais operações, encerrando a conexão

com o servidor.

Para padronizar e melhor divulgar os protocolos, criou-se as chamadas RFCs (*Request for Comments*), que são documentos onde o criador de um protocolo o define e o publica para que a comunidade tome conhecimento e possa desenvolver aplicações que sigam esse padrão (aplicações que podem "conversar" de acordo com esse protocolo). Elas podem ser encontradas no site da IETF (*Iternet Engineering Task Force* – http://www.ietf.org), uma comunidade dedicada a manter padrões na Internet.

As RFCs recebem esse nome por, no princípio, serem documentos onde desenvolvedores solicitavam comentários a seus padrões, sendo hoje definidos como os documentos que regem-os.

Há vários protocolos de comunicação utilizados em redes de computadores para prover os mais diversos serviços. A tabela 1.6 apresenta alguns desses e suas respectivas funções.

Tabela 1.6: Protocolos de comunicação

Protocolo	**Função**
HTTP	(HyperText Transfer Protocol) – Troca de páginas *web*.
FTP	(File Transfer Protocol) – Troca de arquivos dentre *hosts*.
Telnet	Acesso remoto a *hosts*.
SSH	(Secure Shell) – Acesso remoto de forma segura a *hosts*.
SMTP	(Simple Mail Transfer Protocol) – Envio de mensagens de correio eletrônico.
POP	(Post Office Protocol)– Leitura de mensagens de correio eletrônico.
IMAP	(Internet Message Access Protocol)– Outro protocolo de leitura de mensagens de correio eletrônico.

1.3 Estrutura da rede

Uma forma de melhor compreender as rede de computadores é estudar sua estrutura (como os computadores são interligados, qual a sua hierarquia etc.). De acordo com Kurose and Ross [2005], a estrutura básica de uma rede é:

Bordas da rede: são as aplicações e seus hospedeiros (*hosts* ou nós – dentre eles, os computadores de uso comum, como os que es usam para ler esse texto). Representam os computadores que são interligados de forma a permitir o uso de aplicações

ou o acesso a alguma a partir de um servidor (visualização de páginas *web*, assistir vídeos sob-demanda, ouvir música em alguma rádio *on-line*); contempla também os servidores (computadores responsáveis por "servir" os demais com dados; seria, por exemplo, o computador onde uma página estava armazenada, aguardando que alguém a solicite) da rede, ou seja, ela é composta pelo servidores e estações de trabalho.

Núcleo da rede: são os roteadores, equipamentos destinados a prover a ligação entre duas ou mais redes, criando redes de redes. Sua função é interligar as redes, assim sendo, um equipamento (roteador) será o responsável por gerenciar o tráfego de informação, sabendo distinguir a que rede ele deve ser transmitido, garantindo, então, que a informação chegue ao seu destino; no núcleo da rede é que se dão as etapas da transmissão.

Redes de acesso: é o meio fisico para conexão das redes. Apresenta-se como cabos, ondas eletromagnéticas etc.

1.3.1 Bordas da rede

Visto que os computadores em geral e os servidores compõem as bordas da rede, pode-se facilmente deduzir que nelas se concentram os principais serviços. Estes, por sua vez, são os chamados sistemas finais. Como exemplos de sistemas final, têm-se os servidores *web*, servidores de *e-mail* e servidores de arquivos.

Há duas formas de se acessar esses serviços. Através do modelo **cliente/servidor** (ver figura 1.2), onde um *host* solicita algo a um servidor, o qual responde a solicitação. Isso é perceptível quando um navegador *web* solicita uma página a um servidor e este responde com todos os dados necessários a visualização desta.

O outro modelo seria o ***peer-to-peer*** (ver figura 1.3), que tende a minimizar (em alguns casos até mesmo a exterminar) o uso de servidores. Como exemplo imagine-se um aplicativo de compartilhamento de arquivos (como GNUtella, KaZaA e Napster) onde o usuário efetua uma busca na rede pelo arquivo desejado e, ao encotrá-lo, recebe-o de um outro usuário diretamente, sem contato com servidores (ou com um contato mínimo).

Esses serviços ainda podem ser divididos em outras duas categorias:

Serviços orientados à conexão: mantem-se o controle da troca de mensagens dentre os sistemas finais, de forma a prover funcionalidades como controle de congestionamento, verificação de

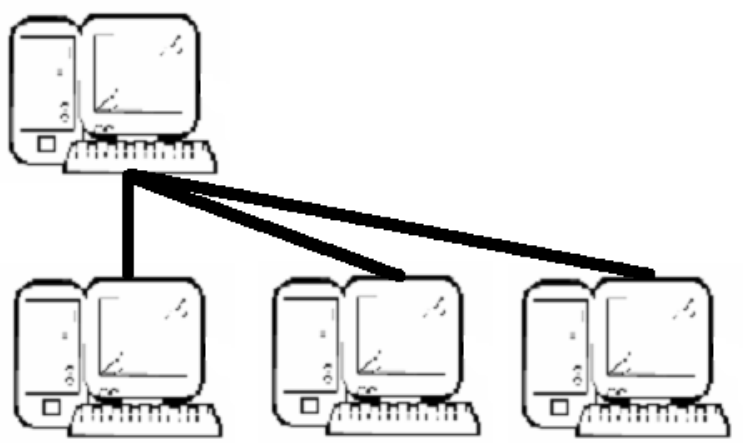

Figura 1.2: Exemplo de estrutura Cliente/Servidor

integridade das mensagens trocadas etc. (ex.: transferência de arquivos via protocolo `FTP`, onde é interessante ao usuário que, caso haja um erro na transmissão de algum pacote, este seja reenviado, de forma que o arquivo resultante não esteja corrompido).

Serviços não-orientados à conexão: provê a transferência de dados não-confiável entre sistemas finais. Este tipo de transferência é bastante utilizada pelas aplicações que não permitem atraso (ex.: teleconferência, onde, caso alguma informação seja perdida, não é interessante que ela seja recuperada. Neste caso, vídeo seguiria enquanto o dado seria retransmitido e, quando este chegasse, não faria mais sentido à transmissão dado que sua ordem cronológica já fora superada. Seria como assistir um vídeo com cenas embaralhadas; o mesmo ocorre em rádios `on-line`).

1.3.2 Núcleo da rede

É no núcleo da rede que se concentram os dispositivos que realizam as interconexões dentre as redes. Para efetuar essas conexões, há duas abordagens principais:

Comutação de circuitos: o canal de comunicação é reservado para dedicação exclusiva a conexão, mesmo que seja subutilizado (ex.: chamada telefônica, mesmo que se permaneça em silêncio, o canal está reservado à chamada).

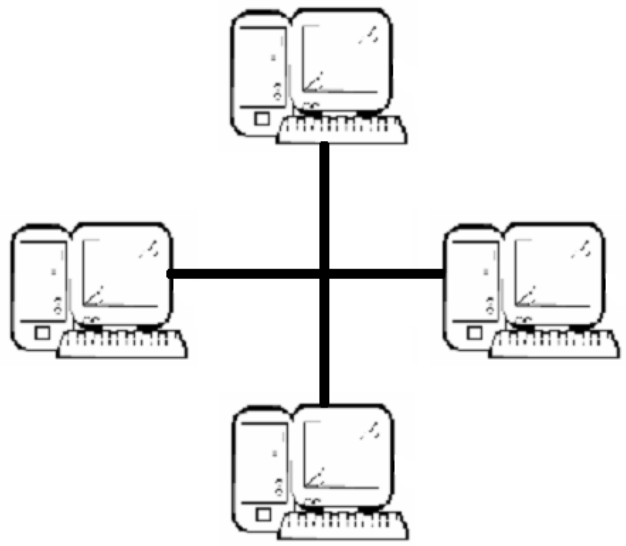

Figura 1.3: Exemplo de estrutura *Peer to Peer*

Comutação de pacotes: o canal de comunicação é dividido dentre seus integrantes, sendo as informações transmitidas em blocos pequenos denominados "pacotes", os quais trafegarão de acordo com uma fila de revezamento, aproveitando melhor a largura de banda.

Comutação por mensagens: é o precursor da comutação por pacotes, diferindo apenas por enviar a mensagem num bloco inteiro, sem dividí-lo em partes menores.

1.4 Comutação por pacotes

A grande maioria das redes atuais atuam em comutação por pacotes. Dessa forma, este tópico será focado no que concerne ao núcleo da rede. Cada mensagem a ser transmitida é dividida em pacotes e estes trafegarão compartilhando o mesmo canal. Cada um dos pacotes usará toda a banda disponível, de forma a prover o melhor aproveitamento possível do canal. Além do que, dessa forma, se provê o melhor aproveitamento do canal, visto que ele tende a ficar sem uso o menor tempo o possível. Caso ele não esteja em uso, estará liberado para que um outro *host* possa transmitir

e/ou receber mensagens.

Entretanto, a comutação por pacotes também apresenta alguns problemas. O primeiro e mais simples seria o atraso fim-a-fim, que nada mais é que o tempo necessário para a transferência de uma mensagem. Esse atraso é a soma do atraso de transmissão (tempo necessário para um *host* transmitir a mensagem) acrescido do atraso de propagação (tempo necessário para que o sinal se propague no meio físico e seja recebido pelo destinatário), além do atraso de processamento em cada elemento da rede (processamento nodal) e atraso de enfileiramento. Isso, a princípio pode parecer irrelevante, especialmente se se trata de uma transferência de arquivos, por exemplo, algo que não seja reproduzido de forma *ad-hoc* (em tempo de execução, assim que se recebe o pacote, o mesmo é reproduzido). Porém, caso se utilize uma transferência de vídeo esse atraso pode ser extremamente danoso ao resultado da transmissão. O vídeo pode ficar com efeito de serrilhamento nas imagens e ou apresentar pausas indesejadas.

Isso pode se agravar caso a demanda de pacotes seja muito grande, o que pode fazer com que esse atraso seja aumentado devido ao congestionamento do canal. A medida que as mensagens são transmitidas, os roteadores as enfileiraram para poder transmití-las até seu destino. Caso a fila do roteador (que é limitada, já que é armazenada na memória) fique cheia e continuem a chegar pacotes, estes serão descartados. Isso pode ser exemplificado através da visualização do aplicativo desenvolvido por Kurose [2005] e na figura 1.4, onde pode-se ver três *hosts* ligados a um roteador; cada um dos *hosts* envia pacotes (ver as cores) que são transmitidos ao roteador, que por sua vez os remete aos destinatários; onde acima do roteador está representada a fila de roteamento, que, caso fique cheia e cheguem outros pacotes, estes serão descartados. Dependendo de como a aplicação trata esse fluxo de pacotes, os pacotes descartados podem ser retransmitidos (serviço orientado a conexão) ou não (serviço não orientado a conexão). O descarte por exaustão de enfileiramento é o principal motivo de perdas de pacotes em redes mas não é o único. Os pacotes também podem chegar corrompidos devido a alterações e deformações impostas pelos meios físicos, principalmente os meios físicos menos confiáveis como as redes sem fio ou as redes de longa distância.

1.5 Meios físicos

Diversos meios físicos são utilizados para conectar computadores em rede. Os mais comuns são:

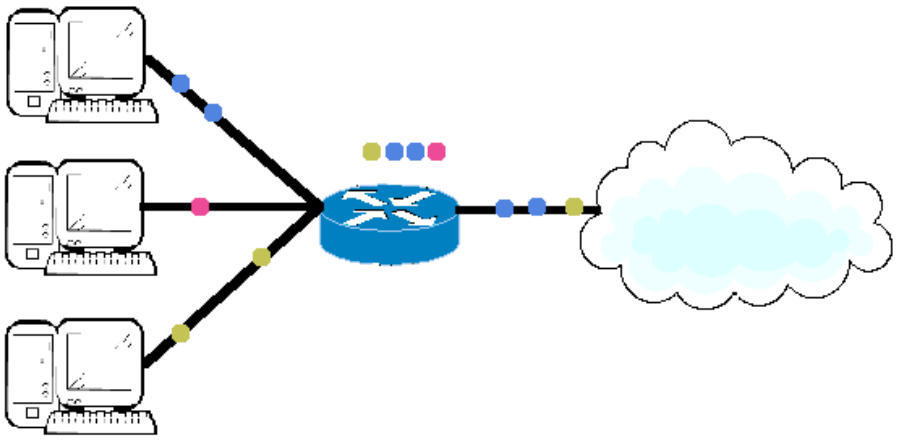

Figura 1.4: Roteamento de pacotes numa rede com enfileiramento

Twisted pair (par trançado): pares de fios de cobre trançados com transmissão de até:

- 10Mbps (categoria 3);
- 100Mbps (categoria 5);
- 1Gbps (categoria 6).

Esses cabos são trançados para minimizar o efeito de interferências eletromagnéticas, na medida do possível.

Cabo coaxial: dois condutores de cobre concêntricos (muito utilizado em conexões de TV UHF). Esses cabos apresentam baixos níveis de interferência pois apresentam uma blindagem eletromagnética.

Fibra óptica: fibra de vidro que transporta pulsos de luz (cada sinal luminoso corresponde a um bit), com baixa taxa de erros e alta velocidade, sendo imune a ruídos eletromagnéticos.

Ondas de rádio: sinal transportado por ondas eletromagnéticas sem fios físicos. Esse tipo de meio físico apresenta alta taxa de interferência e é bastante susceptível a ruídos.

1.6 A Internet

Uma internet consta da interligação de duas ou mais redes de computadores. Quando se fala da `Internet` (note-se o "I" maiúsculo, de forma que se trata de substantivo próprio), refere-se a uma rede

de computadores mundial, que nada mais é que a interligação de várias redes menores.

1.6.1 Tipos de acesso comerciais

Há vários modos de se conectar à Internet. Dentre eles, destacam-se os subcitados:

Modem discado: Conexão direta ao roteador de até 56kbps através de linha telefônica, onde a linha permanece indisponível para chamadas telefônicas. Foi o método mais comum no surgimento da internet, sendo, ainda hoje, largamente utilizado no Brasil.

ADSL (*Asymmetric Digital Subscriber Line*): Conexão através de linha telefônica com taxa de upload de até 1Mbps e 8Mbps de download, não interferindo na linha telefônica por operar numa freqüência diferente (ex.: Velox, Speedy).

HFC (Híbrido Fibra e Coaxial): Assim como o ADSL, funciona de forma assimétrica, com até 30Mbps de *upload* e 2Mbps de *download*. Junto ao ADSL tem sido os grandes substitutos do acesso discado (em Maceió temos a Big TV).

Wi-Fi: é uma tecnologia de acesso a redes sem fio, onde o usuário deve estar no raio de alcance de um ponto de acesso (*hotspot*), sendo muito utilizado em aeroportos e *shopping centers* para prover acesso aos freqüentadores.

Wi-Max (Worldwide Interoperability for Microwave Access): similar ao Wi-Fi, porém agrega novas tecnologias de forma a prover melhor desempenho à conexão. Essas redes são a grande promessa para as conexões sem fio de longa distância.

PLC (Power Line Communication): trata da transmissão de dados através dos cabos da rede elétrica, funcionando, analogamente a relação do ADSL com a rede telefônica, numa freqüência diferente da utilizada na transmissão elétrica, sendo ainda pouco utilizada.

1.6.2 Histórico da Internet

Devido a necessidades militares de transferência rápida de informações e da necessidade de se construir uma infra-estrutura de comunicações que não tenha um ponto único de falha (que possa estar sujeito a ser destruído em uma guerra), a ARPA (*Advanced*

Research Projects Agency, órgão do Pentágono) construiu uma rede capaz de transferir informações da sede do Pentágono a outros lugares, na universidade de Berkeley, denominada ARPANet no fim da década de 60. As informações era divididas em unidades menores denominadas "pacotes", as quais eram chaveadas dentre os computadores interligados à rede.

Numa primeira experiência, interligaram-se a Universidades da Califórnia, a de Utah e o Instituto de pesquisas de Stanford. A primeira transmissão constou do envio do termo LOG da Califórnia para Utah, que seria respondida com o termo IN. Entretanto, houve uma queda no sistema durante a primeira transmissão, sendo enviada apenas as letras LO. Numa segunda tentativa, a transmissão correu conforme o esperado, tornando o termo *log-in* num sinônimo de autenticação para a primeira internet.

Os conceitos que deram origem a Internet surgiram do trabalho de Leonard Kleinrock [ver Kleinrock, 1966] onde foram introduzidas as idéias de comutação de pacotes. Durante o período de 1961 – 1972 se deu a evolução desta teoria. Até então trabalhava-se apenas na perspectiva de um canal dedicado dentre os *hosts*, o que gera, além de alto custo, desperdício de recursos. A partir desse advento, pôde-se otimizar a transferência de dados, permitindo que vários computadores transmitissem dados simultaneamente.

A partir daí que se desenvolveu a ARPANet, ligando militares e pesquisadores através de canais de comunicação por baixo da terra, sem a necessidade de um ponto central de intercomunicação. A partir da década de 70 foi concedida permissão de acesso à ARPANet a universidades e demais instituições com trabalhos relacionados à defesa. Antes do fim de 1975 a ARPANet já contava com 100 *hosts*.

A partir de seu crescimento, pesquisadores já notavam a mudança de comportamento na ARPANet. As aplicações e serviços utilizados começavam a tomar nova forma, permitindo melhores meios de comunicação. Tal grande foi essa revolução que o protocolo de comutação de pacotes original da ARPANet (o NCP – *Network Control Protocol*) se tornara obsoleto, sendo substituído pelo TCP.

Entretanto a Internet ainda se matinha como algo pouco atrativo ao público em geral, até que, na década de 90, Tim Bernes-Lee desenvolveu a primeira faceta multimídia da Internet: a *World Wide Web* (WWW). A partir começaram a crescer os serviços responsáveis por popularizar a Internet, como o serviço de correio eletrônico (*e-mail*), mensageiros instantâneos (Jabber, MSN Messenger), compartilhadores de arquivos (GNUTela, Napster, KaZaA), dentre muitos outros.

Com esses novos serviços, a partir de 1992 houve a grande explosão da Internet, com o acréscimo exponencial da quantidade de

usuários, popularizando a grande rede de computadores. Atualmente a Internet conta com mais de 400 milhões de *hosts* permanentes, além de muitos outros que ficam conectados apenas por alguns momentos, sendo desconectados depois.

Mais informações sobre o histórico da Internet podem ser encontradas em Wikimedia Foundation [2007], Internet System Consortium [2004].

1.7 Internet 2

A Internet 2 é uma rede mundial de alto desempenho que está sendo montada para superar os atuais problemas da Internet, surgindo num consórcio de 150 universidades e outras instituições dos Estados Unidos. Para tal, utilizam-se *links* de acesso de pelo menos 155Mbps.

Além da velocidade, foca-se também os serviços que a infraestrutura atual da Internet não permite que sejam executados, como a tele-educação, transmitindo aulas e palestras pela rede em tempo real. Os dados que trafegam na Internet atual são tratados de forma indiscriminada pelos roteadores, ou seja, tanto faz um pacote de uma aplicação qualquer ou de uma aplicação multimídia ou de tempo-real estar na fila do roteador. Os roteadores utilizam o esquema (FIFO - *first in first out*) de enfileiramento, que não prioriza nenhum tipo de tráfego. Espera-se da nova estrutura da Internet a priorização do tráfego. Por exemplo a priorização do tráfego multimídia, visto que este não tolera atrasos, porém com o cuidado de não inviabilizar o tráfego de outros pacotes. Seu objetivo inicial é acadêmico, porém, acredita-se que, como ocorreu com a Internet, venha a se popularizar rapidamente.

No Brasil, a Internet 2 começará operando a 155Mps (velocidade 15 vezes maior), podendo alcançar 2,5 Gbs (250 vezes maior) [ver Rede Nacional de Ensino e Pesquisa, 2007].

No capítulo anterior discutiu-se o que são redes de computadores e como elas funcionam. Entendendo de uma forma geral seu funcionamento, foi apresentada a Internet como uma rede de redes; a grande rede mundial que reúne diversas redes em uma só.

Faz-se agora necessário compreender a Internet mais a fundo. A partir daqui inicia-se a discussão sobre os serviços disponíveis na Internet.

Capítulo 2

A *Web* e outros serviços da Internet

2.1 Princípios de aplicações de rede

Daqui por diante discutir-se-á os serviços que circundam a Internet. As funcionalidades são diversas: desde servidores de *e-mail* a serviços de videoconferência.

É preciso agora compreender melhor como essas aplicações funcionam. No contexto da Internet é necessário que as aplicações sejam executadas nos sistemas finais e se comuniquem através de uma rede. Vale salientar que não há aplicações desenvolvidas para o núcleo da rede, ele não opera na camada de aplicação. Todas as aplicações da Internet funcionam nos *hosts* finais.

Agora se dará início ao estudo mais aprofundado da arquitetura dos sistemas da Internet. Dessa forma, espera-se atingir melhor compreensão do funcionamento da rede.

2.1.1 Arquitetura cliente-servidor

Conforme visto anteriormente, uma aplicação que opera na arquitetura cliente/servidor pura trata de um cliente que efetua alguma solicitação a um servidor e este a responde. Exemplos desse tipo de situação são muito recorrentes no dia-a-dia, como, por exemplo, os serviços de correio eletrônico (`SMTP, POP e IMAP`), navegação em páginas da *Web* (`HTTP`) e transferência de arquivos (`FTP`).

É importante destacar que, na arquitetura cliente/servidor, um cliente não poderá se comunicar diretamente com outro. Todas as operações são intermediadas pelo servidor. Isso gera melhor controle por parte do servidor, já que é possível monitorar todas as transmissões efetuadas, porém pode gerar tráfego desnecessário e

até mesmo atrasos indesejados.

Basta imaginar uma aplicação de transmissão de mensagens instantâneas (*chat online*), que se estivesse totalmente implementada no paradigma cliente/servidor apresentaria diversas características indesejadas. Todas as mensagens seriam primeiro transmitidas ao servidor e este as redirecionaria ao(s) destinatário(s). Ou seja, todo o tráfego seria primeiro deslocado a um agente intermediador que seria responsável por enviá-lo ao interlocutor. Em se tratando de mensagens de texto, o atraso não seria muito grande, apesar do tráfego na rede ser duas vezes o necessário e o servidor pode ficar sobrecarregado caso haja muito cliente utilizando esse serviço. Mas se estivermos utilizando chamadas de voz, além do tráfego ser muito maior, a possibilidade dos atrasos atrapalharem a comunicação aumentam muito. Ao se passar para aplicações de videoconferência, o atraso seria inevitável, haja vista que o tráfego para transmissão de vídeo é extremamente maior.

2.1.2 Arquitetura *peer to peer* pura

Já foi dito anteriormente que na arquitetura *peer to peer* ou não se utiliza servidores ou eles são pouco utilizados, em geral apenas para controle, por exemplo a autenticação. Uma aplicação conecta-se diretamente com o *host* que lhe é de interesse. Dessa forma cada *host* tem acesso direto a qualquer outro. Seu uso é muito difundido nos serviços de compartilhamento de arquivos (como na rede Gnutella).

Com esse conceito, os sistemas podem interagir diretamente, sem intermediários, reduzindo o tráfego de dados na rede e evitando atrasos indesejados. Porém uma rede desse tipo se torna muito difícil de gerenciar. Como não há um servidor central, não se pode ter controle dos *hosts* conectados. Não há como controlar a troca de mensagens e nem sequer ter certeza dos dados da rede.

2.1.3 Arquitetura híbrida (cliente/servidor – *peer to peer*)

Utilizando-se as características de cada uma das arquiteturas supracitadas quando mais for conveniente, pode-se resolver os problemas anteriormente citados. No exemplo dado na seção 2.1.1, o bate-papo *online*, o problema pode ser facilmente transponível efetuando a conexão cliente/servidor para obter os dados dos contatos e a troca de mensagens ocorrer diretamente dentre os *hosts* (*peer to peer*). Dessa forma se minimizará os atrasos e o tráfego indesejado será suprimido. Exemplos desse tipo de aplicação são os serviços de mensagens Jabber e MSN. Neles, para conectar-se o usuário efetua

log-in num servidor, o qual retornará a lista de contatos do usuário. Ao se iniciar uma conversa com um outro usuário, a transmissão se dá diretamente com o outro *host*.

Já a situação exposta na seção 2.1.2, o compartilhamento de arquivos, pode ser resolvida apenas implantando o controle dos *hosts* por um servidor. Manter-se-ia a troca de mensagens diretamente *host* a *host*. O servidor apenas manteria o controle dos arquivos fornecidos por cada estação facilitando o sistema de busca (em vez de solicitar a cada *host* que informe seus arquivos disponíveis, cada um o faria ao servidor após a conexão e a busca seria diretamente no servidor, que encaminharia o usuário ao *host* que possui o arquivo de interesse do solicitante) e a gerência efetiva da rede. Isso pode ser observado nas redes Napster, KaZaA e EDonkey.

A escrita de aplicações que se comunicam em rede é realizada definindo protocolos em camada de aplicação ou utilizando as definições já feitas. Por exemplo, para escrever um navegador *Web* não se faz necessário definir novos protocolos de comunicação dado que o `HTTP` já está bem definido e documentado. Basta seguir a `RFC-2616` e implementar o navegador *Web*. Caso deseje-se escrever um serviço que ainda não está especificado, deve-se especificar um novo protocolo de camada de aplicação e apenas escolher qual serviço é mais apropriado, o serviço orientado a conexão ou o serviço não-orientado a conexão. Não se faz necessário que o desenvolvedor da aplicação conheça, com detalhes, toda a infra-estrutura da rede, como por exemplo, como os dados serão roteados entre os *hosts*. Essa estrutura é possível por causa da divisão em camadas e o acoplamento entre essas camadas.

Em qualquer uma dessas arquiteturas apresentadas anteriormente, as aplicações precisam se comunicar. Detalhes dessa comunicação está descrito na próxima seção.

2.2 Comunicação na rede

Para que uma aplicação possa interagir com outras é necessário que estas troquem informações. Antes de prosseguir, defina-se um processo como um programa qualquer que está sendo executado num sistema hospedeiro.

Se essa comunicação for efetuada no mesmo *host*, a troca de mensagens é estabelecida pelo Sistema Operacional numa comunicação interprocessos. No caso dos *hosts* serem sistemas diferentes, essa comunicação é efetuada através da troca de mensagens. Assim sendo, ter-se-á dois tipos de processos:

Processo cliente: processo que inicia a comunicação;

Processo servidor: processo que aguarda ser contactado (o processo fica em espera até que receba alguma mensagem).

Observa-se que em sistemas *peer to peer* há processos cliente e servidor na mesma aplicação, já que hora ela se comporta como cliente, hora como servidor.

Essa troca de mensagens ocorre através dos chamados *sockets*. Os processos enviam uma mensagem através de seu *socket* e também recebem mensagens através deles. Os *Sockets* definem e controlam os métodos de entrada e saída das mensagens.

Um *socket* pode ser visto como uma porta onde um processo de envio empurra a mensagem para fora da porta e um processo de recebimento a deixa entrar. Os processos de envio e de recebimento confiam na infra-estrutura de transporte oferecida pela rede. Diversas linguagens de programação oferecem bibliotecas para facilitar a implementação de *sockets*, ou seja, para facilitar o desenvolvimento de aplicações em rede.

Os *sockets* são identificados na Internet através do endereçamento. Na próxima seção o endereçamento utilizado na Internet é discutido com mais detalhes.

2.3 Endereçamento

Para que um *host* possa ser localizado na rede, é preciso que ele possua um identificador único, um endereço. Dessa forma, foi criado o conceito de endereçamento `IP`. Um conjunto de quatro *bytes*, ou seja, 32 bits, que identificará cada sistema, gerando numa representação decimal, um bloco de quatro números de 0 a 255 (ex: 201.4.93.96).

É importante frisar que essa estrutura de endereçamento é definida pelo protocolo `IPv4` [ver Information Sciences Institute – University of Southern California, 1981], sendo que já está em fase de estudo (a transição entre essas versões ocorrerá de forma muito lenta) o `IPv6` [ver Deering and Hinden, 1998].

Entretanto sabe-se bem que é muito comum usar várias aplicações num mesmo sistema (ex: navegar em páginas da *Web* enquanto se utiliza algum mensageiro instantâneo e ouvir música por uma rádio na Internet). Dessa forma apenas o endereço do *host* não será suficiente para efetuar a troca de mensagens, as aplicações dentro de um *host* também devem ser identificadas de forma única. Afinal, ao chegar uma mensagem, como o sistema poderá encaminhá-la a aplicação correspondente se todas possuem o mesmo endereço, sem ter de enviá-la a todas? Para resolver essa questão utiliza-se o conceito de **portas**. Cada mensagem é encaminhada a um

endereço associada a uma porta, havendo convenções de portas comuns, conforme pode ser visto na tabela 2.1.

Tabela 2.1: Tabela de convenção de algumas portas

Porta	**Aplicação**
21	Serviços `FTP`
23	Acesso remoto Telnet
25	Envio de *e-mail* `SMTP`
80	Serviço de navegação *Web*

Dessa forma, um *socket* é unicamente identificado na Internet através da tupxla $\text{IP}_{\text{origem}}$:$\text{Porta}_{\text{origem}}$ $\longleftrightarrow$$\text{IP}_{\text{destino}}$:$\text{Porta}_{\text{destino}}$, como ilustrado na Figura 2.1. Na figura, temos que o *host* A está rodando uma aplicação que é identificada como processo 1. No *host* B também está rodando uma aplicação e ela também é identificada como processo 1. Note que a identificação das aplicações deve ser única dentro de um mesmo *host* mas as aplicações podem ter a mesma identificação desde que estejam executando em *hosts* diferentes. O sistema operacional de cada *host* irá controlar a identificação dessas aplicações. Dado que essas duas aplicações queiram se comunicar, um *socket* é estabelecido entre elas. Esse *socket* pode utilizar o serviço orientado a conexão ou o serviço não-orientado a conexão oferecidos pela camada de transporte.

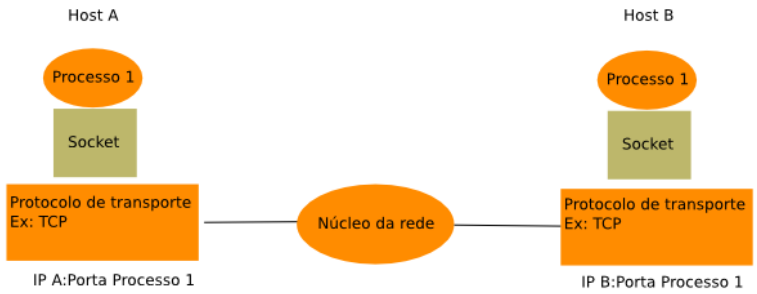

Figura 2.1: Elementos de um Socket

Na próxima seção são discutidos, com mais detalhes, os serviços oferecidos pela camada de transporte.

2.4 Serviços de transporte

Há vários serviços de transporte disponíveis numa rede de computadores. Antes de se discutir esses serviços, apresenta-se um breve resumo das necessidades de controle no transporte de mensagens.

Aplicações de transferência de arquivos (como FTP) e acesso remoto a sistemas (Telnet) não admitem perda de dados. Se um arquivo não for transmitido por completo, o mesmo em geral não pode ser utilizado, assim como o acesso remoto com perda de dados pode trazer vários problemas ao sistema. Entretanto aplicações de *streaming* multimídia como transmissão de áudio *online* podem admitir a perda de uma certa quantidade de dados, não havendo perdas consideráveis.

Essa última também exige um limite mínimo de banda para transferência (velocidade de transmissão), assim como as de teleconferência. Há ainda aplicações que não admitem atrasos na transmissão de dados, como as de telefonia IP e jogos *online*.

A partir daí, definem-se os serviços de transporte disponibilizados pelos dois principais protocolos de controle de transmissão de dados. Na Internet temos:

TCP: apresenta a transmissão de dados orientada à conexão (requer conexão entre processos cliente e servidor), com transporte confiável (caso algum pacote não chegue ao destino o mesmo será retransmitido), controle de fluxo (evita que o receptor seja sobrecarregado) e controle de congestionamento (evita que o canal de comunicação fique sobrecarregado); porém não oferece garantia de tempo de transmissão e controle de banda passante.

UDP: oferece transmissão de dados não-confiável sem nenhuma garantia de como dar-se-á o processo de transmissão (fluxo, congestionamento, temporização e banda).

Dado o supracitado, surge a questão: que protocolo de transferência utilizar? Em se tratando de aplicações que requerem confiabilidade, o TCP oferece todo o aparato necessário para esse controle. Porém todos esses recursos exigem tempo de processamento. Além do que, não interessa a uma *streaming* multimídia *online* a retransmissão de um pacote perdido, visto que o tempo de execução desse já passou. Para esse tipo de aplicação, que requer velocidade na transmissão, o UDP apresenta-se como uma melhor alternativa.

Vale salientar que pode-se utilizar o UDP com confiabilidade, desde que seja implementado o serviço de confiabilidade diretamente

na aplicação. Esta opção raramente é utilizada pois, como já foi dito anteriormente, o desenvolvedor, em geral, deve escolher o serviço que melhor se adeque à sua aplicação. No caso de uma aplicação que não se adeque nem ao TCP e nem ao UDP, essa opção pode ser útil mas o seu desenvolvimento será, certamente, mais complexo.

O UDP também não oferece garantias de tempo de entrega e nem de taxa de transmissão. Esses serviços ainda estão sendo desenvolvidos na Internet. Atualmente, as aplicações que necessitam tempos de resposta mais baixo e não necessitam de confiabilidade, optam por UDP pois os controles de fluxo e congestionamento presentes no TCP podem diminuir a taxa de transmissão de uma conexão.

Nas próximas seções serão discutidos alguns protocolos de aplicação frequentemente encontrados na Internet.

2.5 Serviço de resolução de nomes – DNS

Para acessar um sistema remoto, é necessário que se saiba o endereço do servidor. Ou seja, é necessário que o endereço IP seja conhecido.

Entretanto torna-se muito difícil a memorização de endereços de computadores em sua forma numérica. Na tentativa de facilitar a utilização de endereços, criou-se o serviço de resolução de nomes, o DNS (*Domain Name System*), onde um servidor é responsável por traduzir um endereço alfanumérico num endereço IP. Assim sendo, ao invés do usuário acessar um sítio através de um endereço IP como 200.17.114.42, utiliza-se www.ufal.br. O serviço DNS é realizado através de um banco de dados distribuídos de nomes. Isso significa que não há um único servidor que contenha todas as traduções de nomes para endereços IPs. Essa informação está espalhada em diversos servidores onde cada entidade mantém os seus registros para o seu domínio e algumas entidades mantém os registros raiz. Por exemplo, existem os servidores responsáveis pelo início da hierarquia do DNS, ou seja, o domínio ".". A partir daí esses servidores apontam para os servidores responsáveis pelos domínios abaixo desse domínio, por exemplo, o domínio .com., .br., .ar., .net. e .tv.. Assim, sucessivamente, outros servidores são responsáveis pelos domínios em níveis abaixo desses como por exemplo os domínios .com.br., .net.br. e assim por diante. O nome DNS completo contempla o "." no final mas este é suprimido pelas aplicações.

Cada país elegeu uma instituição para ser responsável pela distribuição de nomes, por exemplo, no Brasil, uma instituição fica responsável por todos os domínios abaixo do .br. Caso uma instituição deseje ter um nome abaixo da hierarquia do Brasil, basta

realizar uma solicitação através do sítio http://registro.br, que o seu nome seja incluído nesses servidores e aponte o seu domínio para o seu próprio servidor. Por exemplo, a UFAL tem no registro.br um cadastro que aponta o domínio da UFAL para os seus próprios servidores DNS, de forma que o cadastro dos computadores da UFAL fica nos servidores de DNS da própria UFAL. Para o registro.br basta apenas saber quem é o servidor de DNS da UFAL. Dessa maneira, a UFAL administra quem é o servidor que responde pelo nome ftp.ufal.br e www.ufal.br, por exemplo, e também a todos os domínios abaixo de ufal.br. Por exemplo temos o domínio tci.ufal.br do Instituto de Computação da UFAL. Por sua vez, recursivamente, os servidores que respondem por www.tci.ufal.br, ftp.tci.ufal.br, por exemplo, ficam nos servidores DNS do TCI que estão cadastrados no servidor de DNS da UFAL.

Inicialmente o controle do DNS dos registros brasileiros ficou com a Fapesp (Fundação de Amparo à Pesquisa do Estado de São Paulo), mas, recentemente, esse controle passou para o comitê gestor da Internet brasileira (CGI BR).

No site http://registro.br pode-se inclusive verificar quem são os proprietários de determinado domínio e/ou efetuar o registro de um domínio que esteja disponível.

A questão agora é: como funciona a resolução de nomes? Primeiro é importante saber que a resolução de nomes é feita da direita para a esquerda, separando-se por pontos (".") as subredes. Ou seja, o endereço www.ufal.br é resolvido da seguinte forma:

br: acessam-se os "servidores raiz" (treze servidores espalhados pelo planeta que respondem pelas primeiras chamadas da resolução de nomes) identificando-se ao servidor de qual país será redirecionado para o restante do processo, no caso br indica que deve ser redirecionado ao servidor do Brasil.

ufal: no servidor do Brasil, identifica-se quem é responsável pela tradução do domínio ufal: kharma.ufal.br – 200.17.114.40.

www: acessando o servidor DNS do domínio ufal, verifica-se que www corresponde a 200.17.114.38, retornando ao *host* que o solicitou para que o acesso desejado possa ser efetuado.

Com a utilização do DNS, defini-se uma forma universal para acessar objetos na Internet, a URL (*Uniform Resource Locator*). A URL é utilizada para acesso a qualquer objeto depositado na Internet através de qualquer serviço. A URL tem o seguinte formato:

```
protocol://server-name.domain-name.top-level-domain:
                port/directory/filename
```

ex:

```
http://www.healthyway.com:8080/exercise/mtbike.html
```

```
ftp://ftp.company.com/freeware/progra.exe
```

2.6 A *Web* e o protocolo HTTP

A *Web* pode ser definida como um conjunto de hipermídias (hipertextos, imagens, vídeos etc.) disponíveis para acesso através da Internet. Para que fique mais claro, define-se hipertexto de uma forma simplista como textos que possuem ligações com outros elementos. Assim, torna-se possível navegar entre páginas da Internet através dos *hiperlinks* (ligações de referência a um outro "texto" – sendo esse texto uma outra página – ou até mesmo recursos multimídia como vídeos e animações).

Para que se possa ter melhor idéia da importância da *Web*, basta imaginar como seria a Internet sem esse recurso. Não havendo a disponibilidade dessas páginas não há como acessar *sites*, nem *blogs*, nem qualquer outro tipo de sistema que use essas idéias, como redes de relacionamento ou portais de notícias. Era assim que tudo funcionava até 1990, quando Tim Berners-Lee concebeu a *Web*. Em 1 de agosto de 1991 ele disponibilizou a primeira página *Web* da Internet, que explicava como a *Web* seria e suas vantagens, como criar um navegador etc. Foi a partir daí que houve a grande explosão de uso da Internet, que a partir de 1992 começou a se popularizar, passando a ser largamente utilizada. No Brasil, a Internet iniciou sua operação comercial em 1994 mas já estava operando nas universidades.

Para que a idéia de hipertexto funcionasse, houve a necessidade de criação de uma linguagem capaz de transcrever as ligações de hipertexto. Para suprir essa necessidade, surgiu o HTML, uma linguagem de hipertextos [ver World Wide Web Consortium, 1999a].

2.6.1 O HTML e o protocolo HTTP

Só que ainda seria necessário uma maneira prática de transferir as páginas HTML, um meio de acesso rápido e prático. Foi com essa idéia que se desenvolveram os servidores *Web*. Como já foi visto anteriormente, para que duas aplicações possam conversar na Internet (servidor *Web* e navegador *Web*) é necessário um meio formal de comunicação dentre elas. Esse meio seria o protocolo HTTP [ver Fielding et al., 1999].

O protocolo HTTP funciona originalmente de maneira muito simples, apenas com três métodos:

GET: solicita um objeto (página HTML, imagem etc.) ao servidor *Web*.

POST: envia informações ao servidor.

HEAD: solicita ao servidor que algum elemento seja ignorado à resposta.

Um exemplo de utilização do protocolo HTTP está ilustrado na figura 2.2. Nesta figura, um cliente abre uma conexão com o servidor *Web* da UFAL (em geral na porta 80) e solicita a página `index.html`. O servidor responde com um código informando que a página existe e transfere a página para o cliente.

O protocolo HTTP é conhecido por ser *stateless*, ou seja, não mantém as conexões. A cada pedido por um objeto, o servidor o transfere e encerra a conexão. Com isso, cada vez que uma ligação leva a um servidor *Web* é aberta uma nova conexão com o este servidor. Essa característica é utilizada no serviço *Web* pois, em geral, baixamos uma página, lemos o que nos interessa, e depois navegamos por páginas no mesmo servidor ou em outros servidores, dependendo de para onde a ligação (*link*) nos leva. O fato de não manter a conexão faz com que os servidores possam manter mais clientes pois a manutenção de conexão por tempos longos gera uma sobrecarga no servidor. Perceba que para manter uma conexão o servidor deve ter o registro de todos os clientes ativos. Se essa conexão demorasse, a memória do servidor poderia ficar cheia caso vários clientes conectassem simultaneamente. Não manter a conexão diminui as chances de muitos clientes estarem conectados simultaneamente nos servidores e os sobrecarregarem.

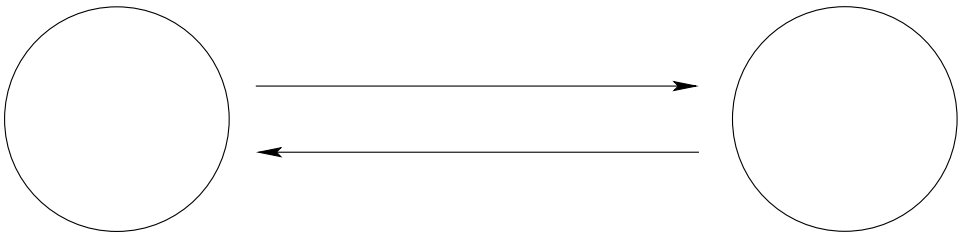

Figura 2.2: Exemplo de acesso HTTP

Com sua evolução, surgiu o HTTP 1.1 (versão atualmente em uso), que acrescenta os seguintes comandos:

PUT: envia um arquivo ao servidor *Web*.

DELETE: apaga um arquivo no servidor *Web*.

Além disto, o HTTP 1.1 melhora o desempenho, modificando a forma como as conexões são realizadas. Por exemplo, a transferência de uma página com 10 figuras necessitaria a abertura de 11 conexões do cliente com o servidor ao utilizar o HTTP 1.0. Uma conexão para trazer a página e mais dez conexões para trazer cada uma das imagens. Utilizando HTTP 1.1 seria necessário apenas uma conexões para trazer a página e as outras dez imagens. Esse fato aumenta o desempenho do protocolo HTTP além de alivar a sobrecarga para a abertura dessas conexões no servidor. Essa característica é conhecida como persistência, pois a conexão persiste até que todos os arquivos sejam transferidos. O HTTP continua *stateless* porém, com o uso da persistência, apresenta melhor desempenho no caso de páginas com vários objetos (muito comum nas páginas atuais onde encontramos várias imagens, animações e aplicações embutidas).

Observe-se ainda na figura 2.2 que junto com o arquivo de resposta, o servidor envia anteriormente um código. É através desse código que o navegador identifica o tipo de resposta enviada. Na tabela 2.2 há alguns dos principais códigos de resposta HTTP.

Tabela 2.2: Códigos de resposta HTTP

Código	**Significado**
200	Requisição efetuada com sucesso; arquivo anexo.
301	Objeto movido permanentemente para <destino>.
400	Servidor não entende a mensagem enviada.
404	Objeto não encontrado.
505	Versão do HTTP não suportada pelo servidor.

Note-se que os códigos iniciados em 2 significam que o processo correu normalmente, em 3 uma modificação no servidor, em 4 um erro na solicitação e em 5 erro no servidor. Sendo ainda que os iniciados em 1 são apenas mensagens informativas.

2.6.2 Os *Cookies*

Uma outra questão interessante é: como personalizar páginas para um usuário? Em outras palavras, como os *sites*, por exemplo, mecanismos de sempre identificar o último usuário a efetuar *login*, já que a resposta do HTTP seria sempre o mesmo *site*, sem ser necessário que ele efetue *login* novamente?

O fato do HTTP ser *statless* faz com que seja mais difícil identificar o usuário. Por exemplo, no FTP, a conexão é mantida até que

o usuário desista dela. Esse fato facilita a identificação do usuário pois a seção é mantida. Como já foi explicado, o HTTP não mantém essa conexão e para cada requisição nova o HTTP entende como se fosse um novo cliente.

Para isso se usam os chamados *cookies*, que são pequenos arquivos de texto que guardam informações para personalização de páginas. Através deles, via HTML a página lê seus conteúdos e redefine-se para prover maior interatividade e customização. Além disso, os *cookies* podem ser utilizados para manter informações de estado em vários processos, como, por exemplo, um carrinho de compras de uma loja virtual. Supondo que no meio de uma seleção para compras falte energia, caso sejam implementados *cookies* o usuário, ao retornar ao *site* terá sua seleção de acordo com o último estado.

Ainda outras funcionalidades são atribuídas aos *cookies*, como obtenção de informações para tentar melhorar mecanismos de buscas.

2.6.3 Servidor *Proxy*

Com o desenvolvimento da *Web*, as páginas foram ficando cada vez mais cheias de objetos como imagens, animações e aplicações embutidas e o carregamento dessas páginas foi se tornando lento. Antes do surgimento da banda larga, era inviável fazer determinadas páginas pois seria muito lento para carregar na casa dos usuários. Para minimizar esse efeito, foi criado um serviço que evitasse que se caso de mais de um usuário da mesma rede desejasse visualizar uma página cada um iniciasse uma conexão com o servidor e efetuasse o *download* do mesmo arquivo. Ou seja, páginas comumente acessadas podem gerar tráfego desnecessário, podendo até mesmo congestionar o canal de comunicação. Como essa página já passou por um canal de comunicação, se ela fosse armazenada localmente, os próximos clientes que desejassem visualizar a mesma página poderiam baixar essa página de um servidor local, evitando o congestionamento. Esse serviço é conhecido como *proxy* com *cache*.

Um servidor *proxy* tem por função responder por acessos à *Web*, em vez dos usuário conectarem-se diretamente. Ou seja, se um usuário deseja acessar um determinado *site*, ele o solicita ao *proxy* que fará a solicitação ao servidor *Web*, retornando ao *proxy* que por sua vez responderá ao cliente solicitante (ver figura 2.3). Assim, pode-se implementar políticas de acesso, determinando qual o limite de banda a cada usuário, que sítios não devem ser acessados e reduzir o tempo de resposta à uma requisição. Um servidor *proxy*

serve como um ponto de gerenciamento das conexões *web* de uma rede. Para isto, basta exigir que todos os clientes dessa rede sempre naveguem utilizando o servidor *proxy*.

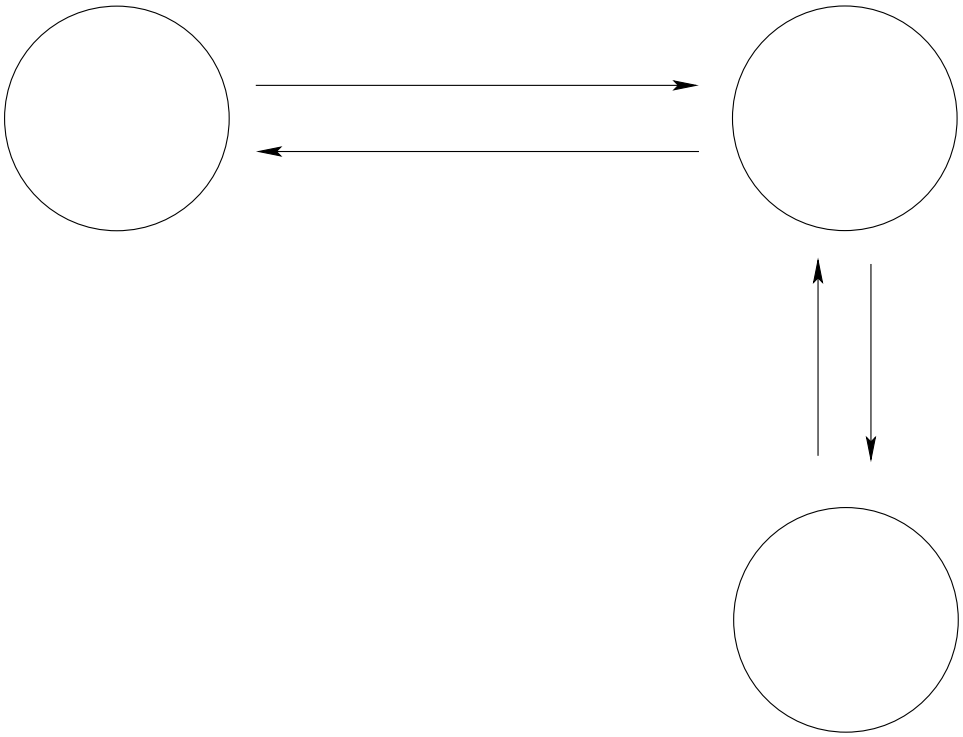

Figura 2.3: Acesso à Internet através de um *proxy*

O *cache* é um repositório local, onde antes de acessar à Internet o servidor verificará se já não possui o objeto solicitado. Com isto ele consegue economizar a quantidade de dados que serão baixados no caso de várias requisições iguais feitas pelos clientes de uma rede. Por exemplo se dois usuários acessam a página da UFAL, o servidor *proxy* só precisa trazê-la uma única vez e entregar aos dois usuários.

Caso o *proxy* não contenha o objeto solicitado no seu *cache*, ele acessará o servidor *Web* externo.

Para evitar problemas caso o objeto seja atualizado (evitar que o usuário não consiga acesso a objetos mais recentes), o servidor *proxy* sempre verifica se a data de atualização do objeto requisitado está mais nova do que a data do objeto que ele tem armazenado. Caso isto ocorra, ele ignora o objeto que está armazenado no repositório local e baixa o objeto mais atualizado, substituindo o objeto obsoleto. O servidor *proxy* consegue identificar quando o objeto está obsoleto pois ele realiza o método conhecido por `GET` condicional,

ou seja, baixa o arquivo apenas se a data de atualização daquele arquivo for mais recente do que uma data especificada (a data de atualização do arquivo armazenado no repositório local) e não baixa caso a data do arquivo localizado no servidor externo seja menor ou igual a do repositório local.

Dessa forma a figura 2.4 ilustra o acesso através de um *proxy* com *cache* onde este possui o objeto solicitado (*cache-hit*) e a figura 2.5 o caso similar, só que o servidor não possui o objeto e terá de efetuar a solicitação ao servidor *Web* (*cache-miss*).

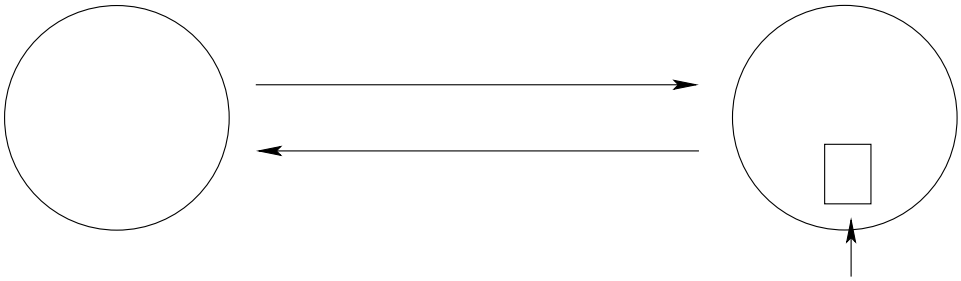

Figura 2.4: *Cache-hit*

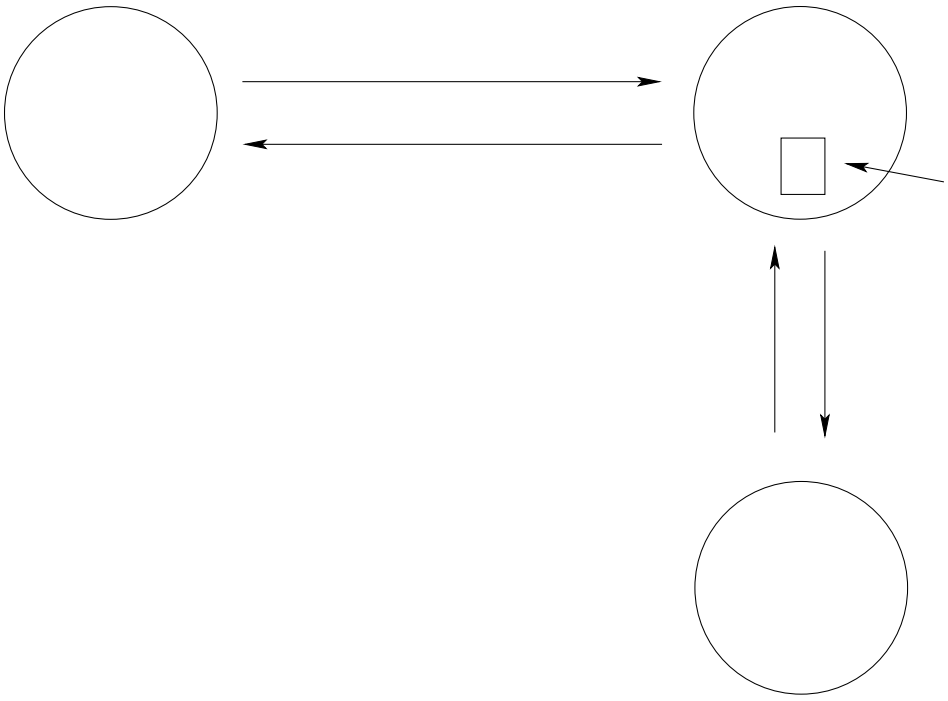

Figura 2.5: *Cache-miss*

2.7 O protocolo FTP

Como forma de melhorar a troca de arquivos numa rede de computadores, criou-se um protocolo destinado exclusivamente à transferência de arquivos, o FTP (*File Transfer Protocol*). Ele permite o *download* e *upload* de arquivos utilizando comando textuais simples e intuitivos.

Por se tratar da transferência de arquivos, é necessário que a transferência seja confiável. Em outras palavras, é necessário que haja garantia de que o arquivo recebido não esteja corrompido e que este não seja acessado por usuários indevidos. Por isso o protocolo FTP é utilizado sobre o TCP. A porta padrão do protocolo FTP é a 21. A partir da conexão ao servidor, o usuário informa seu nome de usuário e senha e que arquivo deseja efetuar *download* ou *upload*, iniciando-se uma conexão para transferência de dados na porta 20. Com isto o FTP é tido como um protocolo onde o controle é fora da banda, ou seja, os comandos são realizados em uma conexão TCP diferente da conexão que transfere os arquivos.

Um exemplo de conexão FTP, onde o comando `get` solicita um arquivo e o comando `put` envia um arquivo ao servidor, seria:

```
open ftp.ufal.br
220 ProFTPD 1.3.0 Server (UFAL) [192.168.0.4]
Name (ftp.ufal.br:heitor):  foo
331 Password required for foo.
Password:
230 User foo logged in.
Remote system type is UNIX.
Using binary mode to transfer files.

ftp>  get meu_arquivo.txt
200 PORT command successful
150 Opening BINARY mode data connection for meu_arquivo.txt (79 bytes)
226 Transfer complete.
79 bytes received in 0.42 secs (0.2 kB/s)

ftp> put outro_arquivo.txt
local: outro_arquivo.txt remote: outro_arquivo.txt
200 PORT command successful
150 Opening BINARY mode data connection for outro_arquivo.txt
226 Transfer complete.
79 bytes sent in 0.00 secs (2204.2 kB/s)

ftp> bye
```

```
221 Goodbye.
```

2.8 Serviço de correio eletrônico

Uma das formas mais populares de utilizar a Internet é a partir do serviço de correio eletrônico (*e-mail*). Ele consiste basicamente na transferência de mensagens de texto dentre usuários especificados.

Para prover este serviço é utilizado o protocolo `SMTP` (*Simple Mail Transfer Protocol* – porta 25), que é responsável pela transferência dessas mensagens do usuário ao servidor. Ou seja, um usuário envia uma mensagem de correio eletrônico através de um programa local em seu computador ou de um *Webmail* (interface de acesso ao à caixa de correio eletrônico através de uma página *Web*) e esta é encaminhada ao servidor de correio eletrônico do usuário. Este, por sua vez, transmite a mensagem ao servidor do usuário de destino, estando então disponível para consulta pelo usuário (ver figura 2.6).

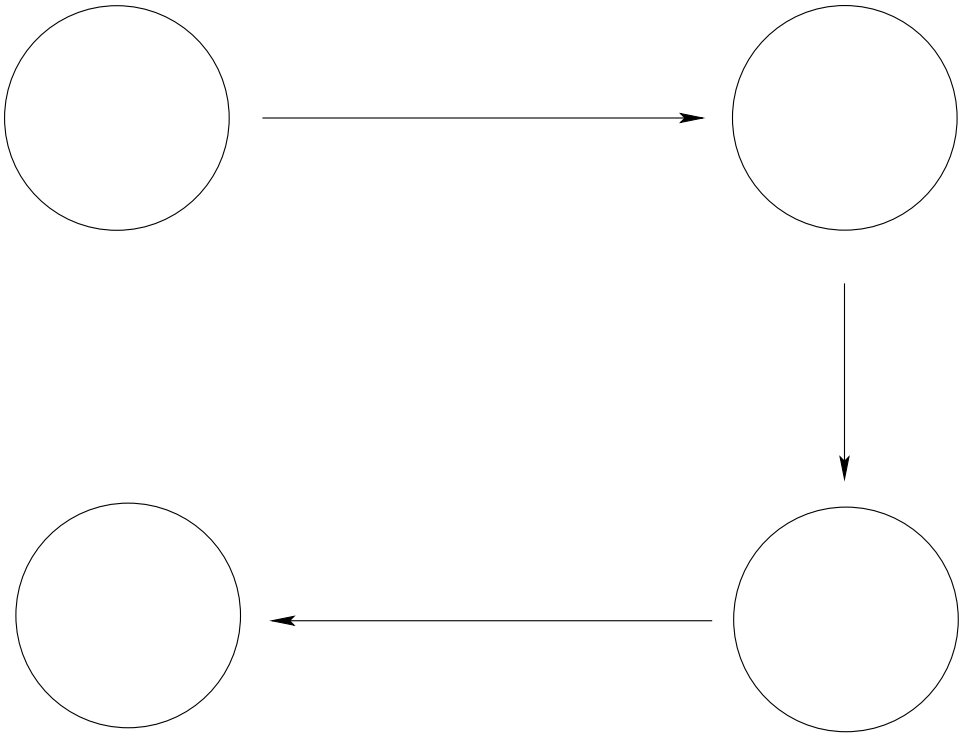

Figura 2.6: Transferência de mensagem de correio eletrônico dentre um usuário remetente a um destinatário

O detalhe é que o protocolo `SMTP` não é utilizado para receber as mensagens de correio eletrônico, apenas para transmití-las. Então,

como um usuário pode ter acesso a suas mensagens? Observe-se ainda na figura 2.6 que o destinatário usa o protocolo POP (*Post Office Protocol* – porta 110) para receber suas mensagens. Ou seja, para transmitir mensagens de correio eletrônico, é utilizado o protocolo SMTP. Para solicita-las ao servidor, utiliza-se o protocolo POP. Vale ressaltar que há outros protocolo de recebimento de mensagens de correio eletrônico, o IMAP (*Internet Message Access Protocol* – porta 465). O IMAP oferece mais recursos ao usuário como a possibilidade de manutenção de pastas dentro do servidor. O protocolo POP é mais simples e mais leve para o servidor enquanto que o protocolo IMAP tem mais recursos sendo que consome mais recursos do servidor.

Capítulo 3

Aplicações *Web*

Não é difícil imaginar quão incerto foi o futuro da *Web* em seus primeiros anos. O responsável pela criação das páginas (*Web designer*) tinha como obrigação prover todas as etapas do processo, desde a redação do texto até mesmo a codificação `HTML`.

Com o passar do tempo e a evolução natural da *Web*, surgiram diversas funções para a construção de páginas. Desde redatores especializados, profissionais responsáveis pela usabilidade, *design*, *marketing*, programação etc. A *Web* agora possuía padrões de grande importância à sua compreensão.

3.1 Arquitetura da Informação para a *Web*

Como forma de facilitar e melhorar o processo de desenvolvimentos de páginas *Web*, surgiu o conceito de Arquitetura da Informação. Não haveria mais apenas um *webdesigner* responsável por todo o processo. As etapas seriam divididas em profissionais de habilidades distintas. Pode-se definir que esta é a fusão da tecnologia com o *design* e a produção de textos [ver Rosenfeld and Morville, 2002, Dijck, 2003].

Assim, a Arquitetura da Informação para a *Web* se firmou como a estruturação e organização da informação em sistemas computacionais de forma a prover soluções para sua organização visual. A informação deve ser útil e inteligível, com o fluxo de navegação bem estruturado. O usuário deve facilmente poder obter todos os recursos disponíveis, o texto deve se apresentar de forma clara, os recursos devem funcionar conforme o esperado, a informação deve ser facilmente localizável e o *design* da página deve ser tal que, além da estética, não torne a navegação cansativa [ver Agner and Silva, 2003].

3.2 Características das aplicações

Para que se possa compreender as características em comum das aplicações *web*, faz-se necessário conhecer seus tipos. Seguem esses conceitos, de acordo com Souza [2005].

Informativo: trata de apresentações de notícias, catálogos de produtos, manuais, livros eletrônicos etc. Tem por objetivo a informação direta, clara e objetiva.

Interativo: tipo de aplicação destinada a interagir com o usuário, tendo como principais representantes os jogos de computador, incluindo também aplicações mais simples, como formulários de registros em sítios.

Transacional: destina-se a realização de negócios na *web*, como sistemas para compras *on-line* e sistemas bancários.

Workflow: engloba os sistemas destinados a prover o fluxo da informação para a melhoria do sistema de trabalho, como sistemas de planejamento.

Ambientes de trabalho cooperativo: são ferramentas onde vários usuários interagem para a construção da apresentação final, como sistemas de edição de texto colaborativos.

Comunidades *on-line*: sistemas destinados a prover a interação dentre usuários, como *chats*, fóruns e sistemas de relacionamento.

Portais *Web*: grandes agregadores de conteúdo que redirecionam o usuário a seus interesses.

3.3 HyperText Markup Language – HTML

Conforme dito anteriormente, Tim Berners-Lee, precisava de uma linguagem para criação de páginas capaz de implementar ligações internas e externas de conteúdo. Essa linguagem seria o HTML [ver World Wide Web Consortium, 2007a].

O HTML é composto de *tags*. Assim, todos os comandos da linguagem devem estar dentro de marcadores "< "e ">". O documento é dividido em duas seções:

<**head**>**:** apresenta o cabaçalho do documento, com suas definições básicas, como codificação do texto, título etc.

<**body**>**:** contém o conteúdo do documento em si, com suas estruturas.

Então, um documento básico em `HTML` seria:

```
<HTML>
  <head>
    <title>Minha primeira página HTML</title>
  </head>
  <body>
    Minha primeira página HTML!
  </body>
</HTML>
```

O que resultaria na página, interpretada por um navegador *Web*, da figura 3.1.

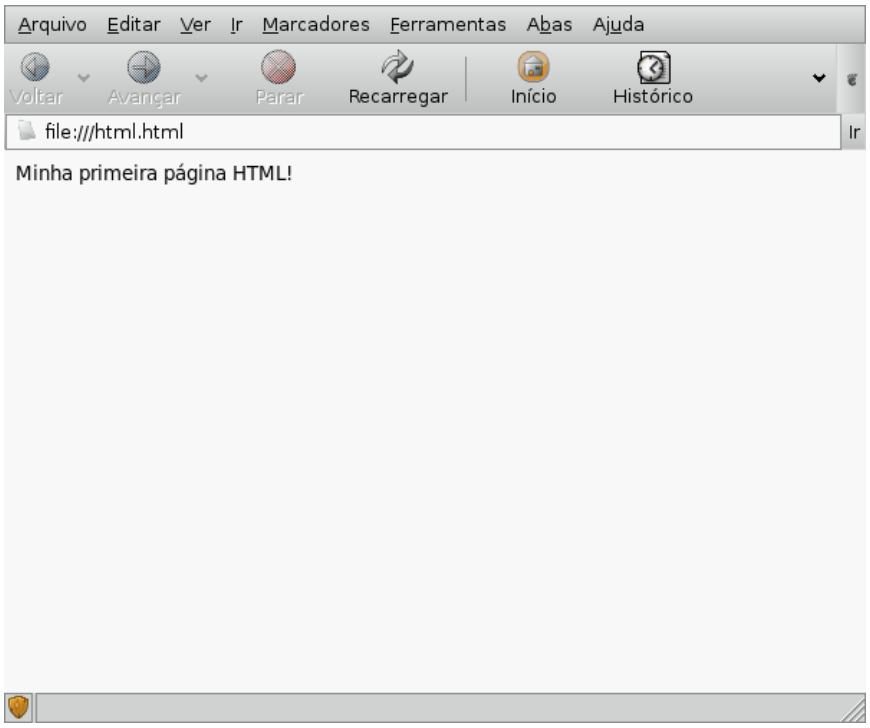

Figura 3.1: Página HTML básica interpretada por um navegador *Web*

O `HTML` possui uma imeensa gama de recursos, como tabelas e, obviamente, a possibilidade de criar ligações dentre páginas, os chamados *hyperlinks*. Para criar um hyperlink, basta usar a seguinte *tag*:

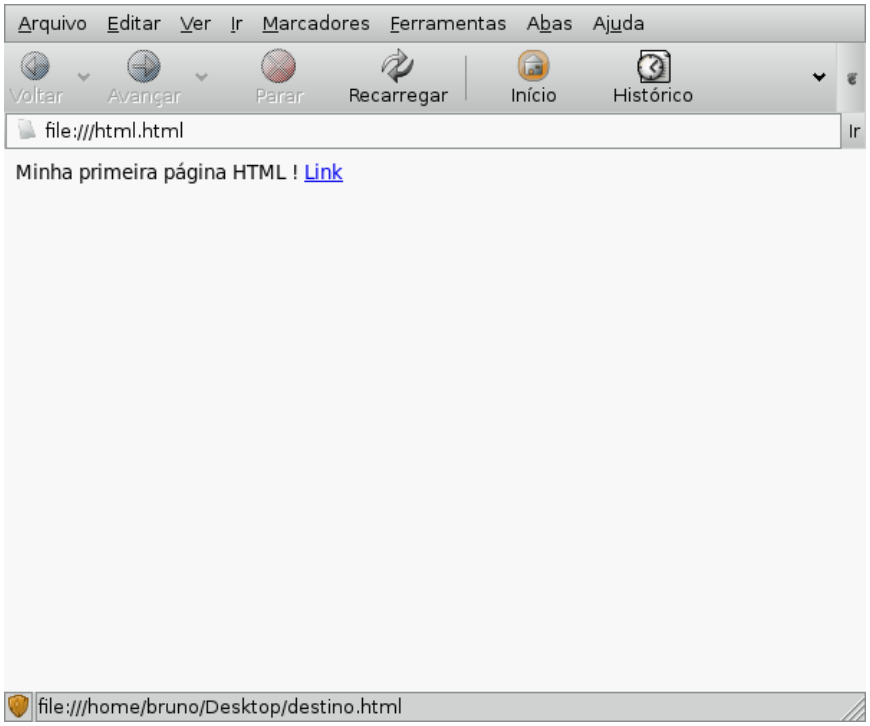

Figura 3.2: *Hyperlink*

`<a href=destino.html> Link </a>` (ver figura 3.2). Assim, o texto "Link" torna-se uma ligação de referência à página "destino.html".

Assim, combinando *tags*, pode-se criar páginas com vários recursos. A especificação do HTML válido pode ser encontrada em World Wide Web Consortium [1999b]. Essa é a especificação da World Wide Web Consortium (W3C) – entidade responsável pela regulamentação dos padrões da *Web*. Nessa definição consta o que é necessário para tornar uma página HTML válida, que é a forma de garantir a compatilidade de uma página independente de plataforma de *hardware/software*.

Um bom material para estudos mais aprofundados de HTML pode ser encontrado em de Castro [1995].

3.4 Cascading Style Sheets – CSS

Para que um mesmo texto possa ser utilizado em mais uma página, utilizando apenas HTML, eles teriam a mesma formatação. Da mesma forma, para modificar simplesmente a aparência geral de um texto, há a necessidade de alterar a página HTML propriamente dita. Por exemplo, se for desejável mudar a forma como o

link aparece em uma página (o padrão é em letras azuis e sublinhado) temos que alterar cada um dos *links* com *tags* específicas que modificam a aparência desses *links*. Relacionados a aparência estão as cores, tipos e tamanhos de fontes e as cores e imagens de fundo, por exemplo.

Como forma de simplificar esses processos, desenvolveu-se uma linguagem de estilo, o CSS [ver World Wide Web Consortium, 2007b]. Através de seu uso, é possível alterar a aparência de um documento escrito em uma linguagem de marcação (no caso, HTML), permitindo a separação entre aparência e formato de apresentação.

Ou seja, ao invés de se definir a aparência do documento diretamente no código HTML, referencia-se uma outro documento, o qual conterá apenas o detalhamento dos estilos a serem utilizados.

Com isto, podemos utilizar o HTML para formatar o documento, ou seja, definir onde e como os objetos irão aparecer no documento e utilizar o CSS para definir a aparência, ou seja, cores, tipologia de fonte, imagens e cores de fundo. Os esquemas mais sofisticados também separam o conteúdo, os amarzenando em banco de dados. Dessa forma, o profissional conteudista pode editar o conteúdo de uma página sem se preocupar como essa conteúdo irá aparecer, deixando essa tarefa para o profissional que irá definir como a informação deve aparecer e outro profissional, o designer, irá definir a aparência geral. Com isso temos uma arquitetura mais apropriada para os grandes sítios presentes na *Web* atual.

Como forma de exemplicar, segue o código abaixo:

```
body
  {
     font-family: Times, Verdana, sans-serif;
     background-color: #FAF;
     margin: 5px 10px;
  }
```

Este código define alguns parâmetro de aparência (fonte, cor de fundo e margem) para a seção <body> de um documento qualquer que o referencie.

Há duas maneiras de referenciar CSS, ambos na seção <head>. Pode-se transcrevê-lo diretamente dentre as *tags*:
`<style type="text/css">...</style>`.
Ou referenciá-lo num arquivo separado:
`<link rel="stylesheet" href="folha_de_estilo.css">`.

Mais informações para aprofundamento podem ser encontradas também no material de de Castro [1995].

3.5 eXtensible Markup Language – XML

Com as necessidades de armazenar grande volume de informação de temas variados, a IBM criou uma linguagem específica a esse propósito: a GML (*General Markup Language*). Para que se possa melhor conceber as vantagens de se utilizar uma linguagem específica ao armazenamento de dados, imagine-se a quantidade de dados que seriam oriundos da documentação dos vários formatos de armazenamento de dados em arquivos específicos a cada sistema, tendo em vista a quantidade de áreas pesquisadas pela IBM até hoje.

Assim, a ISO (organização internacional destinada a padronizações) resolveu padronizar a linguagem, de forma que ela pudesse ser largamente utilizada, criando então o SGML (*Standard* GML). Sua principal característica é a facilidade em se adaptar aos mais variados tipos de dados. Surgiram a partir daí vários sistemas de armazenamento de informação.

Com o surgimento do HTML, utilizaram-se conceitos seus e a base do SGML para desenvolver uma nova linguagem, capaz de armazenar diversos tipos de dados de forma portável. Assim, desenvolveu-se o XML.

O XML [ver World Wide Web Consortium, 1996]é uma linguagem de marcação (assim como o HTML) extremamente flexível, utilizada para definição de dados. A grande diferença ao HTML é que a interpretação de cada uma das *tags* é exclusiva da aplicação. Ou seja, uma *tag* <body> não necessariamente está relacionada ao corpo do documento. Ela pode ter qualquer outro significado. Com o XML o projetista desse conteúdo poderá criar *tags* de acordo com a necessidade da aplicação. Isso torna o XML bastante flexível e diferente do HTML.

Um exemplo de arquivo na estrutura XML seria:

```
<?xml version="1.0" encoding="UTF-8"?>
<Funcionario>
  <Pessoal>
    <Nome>Foo da Silva</Nome>
    <Titulacao>Doutor</Titulacao>
    <Estado_Civil>Casado</Estado_Civil>
    <Sexo>m</Sexo>
  </Pessoal>
  <Profissional>
    <Cargo>Professor</Cargo>
    <Locacao>Instituto de Computação</Lotacao>
  </Profissional>
</Funcionario>
```

Ou seja, pode-se criar *tags* conforme a necessidade. Não há limitações de campos.

Um dos principais modos de usar XML na *Web* é a separação de camadas em documentos. Assim, utiliza-se o HTML para definir as estruturas da página, o CSS na aparência da página e o XML na gestão do conteúdo. Assim, caso se necessite modificar o conteúdo de uma página, não há necessidade de alterar o arquivo que contém as estruturas.

porém, ainda resta uma questão: como expressar a estrutura de um documento XML a uma aplicação? Para isso existe o chamado *XML Schema*. De acordo com Dicas-L [2005], um *XML Schema*:

- define elementos que podem aparecer em um documento;
- define atributos que podem aparecer em um documento;
- define que elementos são elementos filhos;
- define a ordem dos elementos filhos;
- define o número de elementos filhos;
- define se um elemento é vazio ou pode incluir texto;
- define tipos de dados para elementos e atributos;
- define valores padrão e fixos para elementos e atributos.

Podemos, então, definir como seria a estrutura de uma entrada <Pessoal>, que será compreendida por qualquer aplicação. Assim, a definição com XML *Schema* seria:

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.w3schools.com"
  xmlns="http://www.w3schools.com"
  elementFormDefault="qualified">

  <xs:element name="Pessoal">
     <xs:complexType>
       <xs:sequence>
         <xs:element name="Nome" type="xs:string"/>
         <xs:element name="Titulacao" type="xs:string"/>
         <xs:element name="Estado_Civil" type="xs:string"/>
         <xs:element name="Sexo" type="xs:string"/>
       </xs:sequence>
     </xs:complexType>
```

```
</xs:element>

</xs:schema>
```

Para referenciar o XML *Schema*, bastaria então:

```
<?xml version="1.0"?>

<Pessoal
xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com pessoal.xsd">
```

Onde “pessoal.xsd” é o arquivo que contém o código do *Schema*.

Para informações mais aprofundadas que complementem os estudos de XML, há: World Wide Web Consortium [1996], Dicas-L [2005], XML Tutorial [1999].

Referências Bibliográficas

L. Agner and F. Silva. Uma introdução à disciplina de arquitetura de informação: Conceitos e discussões. In *Anais do 2º Congresso Internacional de Pesquisa em Design*, 2003.

Maria Alice Soares de Castro. Tutorial HTML do ICMC-USP, 1995. URL http://www.icmc.usp.br/ensino/material/html/. última consulta em novembro de 2007.

S. Deering and R. Hinden. Internet protocol: Darpa internet program, 1998. URL http://www.ietf.org/rfc/rfc0791.txt. RFC 2460, última consulta em novembro de 2007.

Dicas-L. Tutorial XML schema, 2005. URL http://www.dicas-l.com.br/dicas-l/20050326.php. última consulta em novembro de 2007.

P. V. Dijck. *Information Architecture for Designers: structuring websites for business success*. Rotovision, 2003.

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1, 1999. URL http://www.ietf.org/rfc/rfc0791.txt. RFC 2616, última consulta em novembro de 2007.

Information Sciences Institute – University of Southern California. Internet protocol: Darpa internet program, 1981. URL http://www.ietf.org/rfc/rfc0791.txt. RFC 791, última consulta em novembro de 2007.

Internet System Consortium. Isc internet domain survey, 2004. URL http://www.isc.org/index.pl?/ops/ds/. última consulta em outubro de 2007.

L. Kleinrock. Theory of queues applied to time-shared computer systems. In IEEE press, editor, *Proceedings of the IEEE Region Six Conference Record*, pages 491–500, Tucson, Arizona, April 1966.

J. F. Kurose. Queuing and loss applet, 2005. URL http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/queuing/queuing.html. última consulta em outubro de 2007.

J. F. Kurose and K.W. Ross. *Computer Networking*. Addison Wesley, 3 edition, 2005.

Rede Nacional de Ensino e Pesquisa. Internet 2, 2007. URL http://www.rnp.br/noticias/imprensa/2001/not-imp-010810.html. última consulta em outubro de 2007.

L. Rosenfeld and P. Morville. *Information Architecture For The World Wide Web*. OŔeilly, 2 edition, 2002.

O. R. Souza. Processos de apoio ao desenvolvimento de aplicações web, março 2005.

A. S. Tanenbaum. *Redes de computadores*. Elsevier, 2003.

Wikimedia Foundation. Histria da internet, 2007. URL http://pt.wikipedia.org/wiki/Hist%C3%B3ria_da_Internet. última consulta em outubro de 2007.

World Wide Web Consortium. Extensible markup language (XML), 1996. URL http://www.w3.org/XML. última consulta em novembro de 2007.

World Wide Web Consortium. HTML 4.01 specification, 1999a. URL http://www.w3.org/TR/REC-html40/. última consulta em novembro de 2007.

World Wide Web Consortium. Html 4.01 strict dtd, 1999b. URL http://www.w3.org/TR/html4/strict.dtd. última consulta em novembro de 2007.

World Wide Web Consortium. W3C HTML, 2007a. URL http://www.w3.org/html. última consulta em novembro de 2007.

World Wide Web Consortium. Cascading style sheets, 2007b. URL http://www.w3.org/Style/CSS. última consulta em novembro de 2007.

XML Tutorial. Tutorial XML schema, 1999. URL http://www.w3schools.com/xml. última consulta em novembro de 2007.