Unidade II

Camadas de alto nível de protocolos e seus modelos de serviços

Começaremos o estudo das camadas do modelo OSI com uma abordagem que se inicia pela Camada de Aplicação e, de cima para baixo, desce até a Camada Física. Tal abordagem facilita o entendimento, já que as aplicações estão muito próximas dos usuários. Elas são muitas vezes conhecidas por eles e, portanto, de fácil aprendizado.

Entendendo as aplicações, torna-se fácil entender os serviços necessários para suportar tais aplicações e identificar as diversas maneiras como esses serviços são fornecidos ou implementados nas camadas mais baixas.

3 CAMADA DE APLICAÇÃO

Em primeiro lugar, é importante sabermos que a Camada de Aplicação faz a interface entre o protocolo de comunicação e o aplicativo que pediu ou receberá a informação através da rede. Dizemos que ela é o nível que possui o maior número de protocolos existentes, porque está mais próxima do usuário e este possui as mais diferentes necessidades. Com a evolução do *software*, as diversas aplicações evoluem e geram novas necessidades, aumentando, assim, cada vez mais, a quantidade de diferentes protocolos existentes nessa camada.

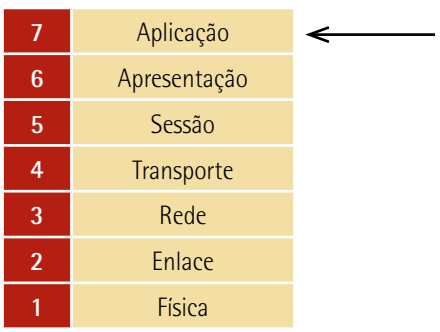

Figura 29 - Camada de Aplicação do Modelo OSI (KOVACH, 2009)

3.1 Princípios de aplicações

Dizemos que as aplicações de rede são "a razão de ser" de uma rede de computadores. Isso quer dizer que, para baixar seus *e-mails*, o seu aplicativo de *e-mail* entrará em contato com a Camada de Aplicação do protocolo de rede, efetuando esse pedido.

Transferir um arquivo entre os dois sistemas finais requer uma forma de trabalhar com as incompatibilidades deles, e essa é a função da Camada de Aplicação. O dado entregue pelo usuário

à Camada de Aplicação do sistema recebe a denominação de SDU (Service Data Unit). A Camada de Aplicação junta à SDU (no caso, os dados do usuário) um cabeçalho chamado PCI (Protocol Control Information). O objeto resultante dessa junção é chamado de PDU (Protocol Data Unit), que corresponde à unidade de dados especificada de certo protocolo da Camada de Aplicação.

Lembrete

O processo que descrevemos agora é o mesmo mostrado no exemplo de encapsulamento, apresentado na Unidade I deste livro-texto.

3.2 Funções específicas da Camada de Aplicação

A Camada de Aplicação é responsável por diversas funções importantes no processo de transmissão de informação através das redes de computadores. São elas:

- categorização dos processos de aplicação;
- processamento de transações;
- acesso a bancos de dados;
- gerência de rede, entre outras.

Exemplos de aplicações nessa camada incluem: transferência de arquivos, serviços de diretório, terminal virtual, *world wide web*, entre outros.

Nesta camada, as aplicações são consideradas processos distribuídos em comunicação. Elas são executadas nos sistemas finais (hospedeiros), nas interfaces de usuário e trocam mensagens para implementar a aplicação.

As mensagens essenciais trocadas entre as aplicações são as de pedido e resposta, que são conhecidas como processos cliente-servidor e serão detalhadas no próximo item.

Dizemos que os protocolos da Camada de Aplicação são uma "parte" da aplicação de rede que define mensagens trocadas por aplicações e ações tomadas. Exemplos: *web* e HTTP, correio eletrônico e SMTP. Um protocolo da Camada de Aplicação define:

- os tipos de mensagens trocadas (requisição e resposta);
- a sintaxe dos vários tipos de mensagens (campos e suas delimitações);
- a semântica dos campos (significado da informação);
- as regras para determinar quando e como um processo envia e responde às mensagens.

3.3 Comunicação entre processos e processos cliente-servidor

Quando falamos da Camada de Aplicação e de como ela pode ser construída, é necessário ter um entendimento básico de como os programas que rodam em vários sistemas finais comunicam-se entre si, os quais chamamos de **processos** que se comunicam. Podemos imaginar um processo como um programa que está rodando dentro de um sistema final. Dois processos se comunicam enviando e recebendo mensagens através de suas **portas**, que é uma via de acesso ao processo.

Um processo é um programa que é executado em um hospedeiro; dois processos no mesmo hospedeiro se comunicam usando o que chamamos de comunicação interprocessos, definida pelo sistema operacional. Quando falamos de comunicação através das redes de computadores – que é no que de fato estamos interessados neste livro-texto –, dois processos em hospedeiros distintos se comunicam usando um protocolo da Camada de Aplicação, ou seja, eles se comunicam pela troca de mensagens por meio da rede de computadores. Dessa forma, um processo originador cria e envia mensagens para a rede; um processo destinatário as recebe e possivelmente responde, devolvendo outras. Os processos se comunicam utilizando a Camada de Aplicação através da pilha de cinco camadas da arquitetura TCP/IP.

O que chamamos de porta é a interface de programação entre a Camada de Aplicação e a Camada de Transporte (*socket*), também chamada de API (Application Programming Interface), pela qual as aplicações em rede são inseridas na internet. Dizemos que, na internet, dois processos se comunicam enviando dados para um *socket* ou lendo dados de um *socket*.

Para identificar na origem e no destino os processos com os quais se quer comunicar, é necessário que eles se identifiquem por meio de endereços.

Observação

Endereçamento dos processos é como chamamos um processo que identifica o outro com o qual quer se comunicar.

Duas informações são essenciais nessa identificação: endereço IP do hospedeiro do outro processo e o "número de porta". Isso permite que o hospedeiro receptor determine a qual processo deve ser entregue a mensagem.

Quando falamos de processos, é importante ainda mencionar que existem mecanismos que permitem a comunicação transparente entre a aplicação de rede e o usuário. Um agente de usuário (User Agent – UA) é uma interface entre o usuário e a aplicação de rede que implementa o protocolo da Camada de Aplicação. Por exemplo, www: navegador (*browser)*; correio: leitor ou compositor de mensagens; *streaming* de áudio e vídeo: tocador de mídia.

Lembrete

Por que se diz "agente de usuário" em vez de "*browser*"?

Embora os *browsers* sejam importantes interfaces dos documentos HTML e XHTML, existem outros programas e sistemas que também interpretam esses documentos. Aplicativos de *e-mail* são um exemplo e não são *browsers*. Ao preferir o termo "agente de usuário", queremos destacar um sentido mais amplo e diferenciá-lo de *browser* simplesmente.

Uma aplicação de rede típica possui duas partes: o lado cliente em um hospedeiro que se comunica com o lado servidor de outro hospedeiro. Para muitas aplicações, um hospedeiro implementa ambos os lados, cliente e servidor de uma aplicação, e pode inclusive implementá-los ao mesmo tempo para um dado serviço.

Um exemplo de aplicação que implementa ao mesmo tempo o lado cliente e o lado servidor é o **correio eletrônico**. Quando o servidor de correio envia é considerado **cliente,** e quando o servidor recebe é chamado **servidor**.

É intitulado **cliente** aquele que inicia contato com o servidor (quem "fala primeiro"), ou seja, quem tipicamente solicita serviço do servidor. Por exemplo, para a *web*, o navegador, e para o correio eletrônico, o leitor de mensagens. Um **servidor** provê ao cliente o serviço requisitado. Como exemplos de servidores, podemos considerar: o servidor *web* envia a página solicitada e o servidor de correio entrega as mensagens.

A figura 30 mostra que a Camada de Aplicação que originou a mensagem, o chamado cliente, é aquela que faz a solicitação do pedido à outra aplicação na mesma camada, no destino. Este, por sua vez, devolve uma mensagem de resposta.

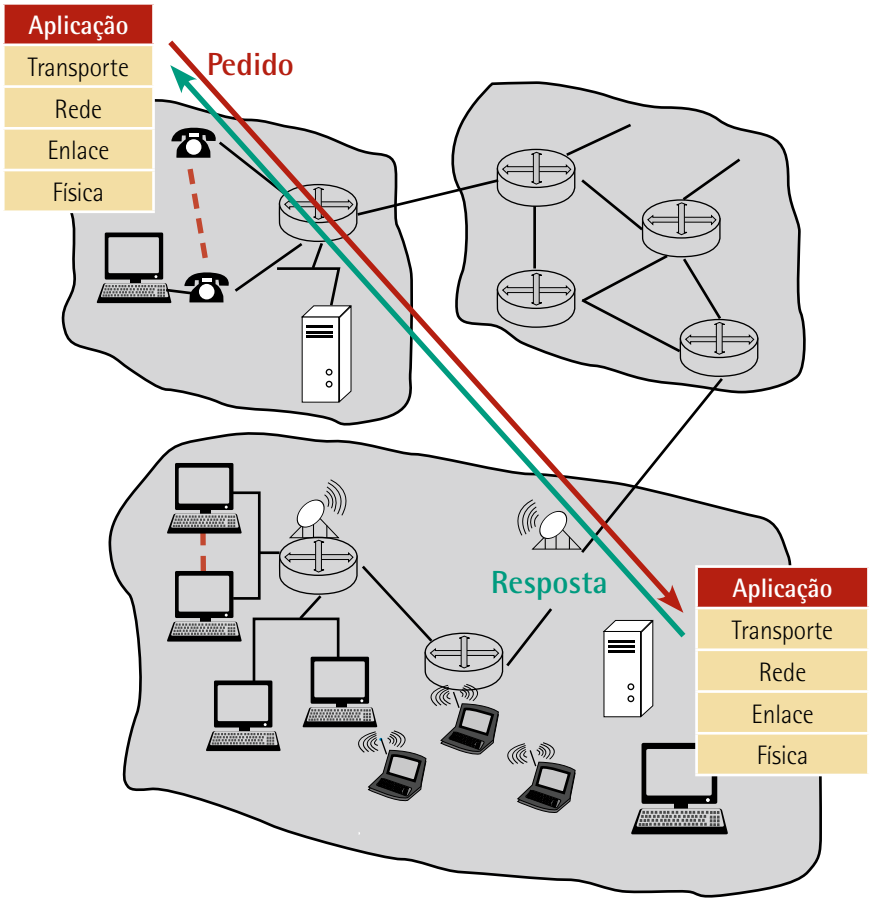

Figura 30 - Aplicações do tipo cliente e servidor (KUROSE; ROSS, 2010)

Vejamos agora para quais serviços uma aplicação necessitaria de um protocolo de transporte. O quadro 4 nos dá exemplos de aplicações que consideram tais parâmetros como requisitos de transmissão através das redes.

Perda de dados

Algumas aplicações, como o áudio, podem tolerar algum tipo de perda de informação. Outras, como transferência de arquivos e acesso remoto, requerem transferência 100% confiável. A alteração ou perda de um único *bit* pode fazer com que a leitura de um arquivo transferido não seja mais possível.

Largura de banda

Algumas aplicações, como multimídia, requerem certa quantia mínima de banda para serem consideradas viáveis. Elas são sensíveis à largura de banda. Outras aplicações chamadas de aplicações elásticas conseguem usar qualquer quantia de banda disponível e se moldar às condições da rede de acordo com a banda.

Sensibilidade temporal

Para as aplicações em tempo real, como telefonia sob internet (VoIP) e jogos interativos, é preferível um atraso pequeno, tolerável, para que sejam consideradas viáveis. As aplicações do tipo assíncronas, como o correio eletrônico, já possuem tolerância a atrasos, porque não são executadas em tempo real.

Lembrete

Lembre-se da diferença entre aplicações **síncronas** e aplicações **assíncronas**: as síncronas são as que possuem interação ou comunicação em tempo real, como é o caso dos comunicadores de mensagens instantâneas; as assíncronas são aquelas em que a interação acontece sem a intervenção em tempo real, como o *e-mail*, que você envia e aguarda a resposta e que pode ocorrer a qualquer momento.

No quadro 4 vemos diversos exemplos de aplicações e como elas são consideradas em cada um dos parâmetros mencionados acima.

Quadro 1 – Exemplos de aplicações e seus requisitos na rede

Aplicação	Perda de dados	Largura de banda	Sensibilidade temporal
Transferência de arquivos	Não tolera	Elástica	Não é sensível
Correio eletrônico	Não tolera	Elástica	Não é sensível
Objetos da www (HTTP)	Não tolera	Elástica	Não é sensível
Áudio ou vídeo em tempo real	Tolerante	Áudio: 5 Kb - 1 Mb Vídeo: 10 Kb - 5 Mb	Sensível, acima de 100 mseg
Áudio ou vídeo gravado	Tolerante	Áudio: 5 Kb - 1 Mb Vídeo: 10 Kb - 5 Mb	Sensível, acima de alguns seg
Jogos interativos *on-line*	Tolerante	Maior que alguns Kbps	Sensível, acima de 100 mseg
Aplicações bancárias	Não tolera	Elástica	Pode ou não ser sensível

As aplicações de transferência de arquivos não toleram qualquer tipo de perda nos dados, porque ao final da transferência um *bit* faltante sequer impediria a abertura do arquivo no destino. O mesmo se aplica ao correio eletrônico e aos objetos carregados em páginas da *web*. Aplicações bancárias também não podem sofrer perdas ou alterações nos dados, pois uma simples mudança pode gerar erros catastróficos para o usuário da aplicação. Entretanto, aplicações em tempo real, como o áudio, podem superar a perda de uma palavra durante uma frase falada, por exemplo. Dependendo da quantidade de perda, torna-se tolerável também a transmissão de vídeos gravados, que passam a ser exibidos com alguns *pixels* ou quadros faltantes, mas o usuário é capaz de interpretar o contexto final. Jogos interativos *on-line* podem ser os mais delicados para se considerar perdas, mas também são toleráveis, dependendo da quantidade de perda de dados.

Se na transferência de arquivos a largura de banda de transmissão for pequena, fará com que a transmissão se torne lenta, ao passo que se a largura de banda for grande, a transferência se tornará mais rápida, ou seja, a aplicação se molda às condições da rede – é elástica. O mesmo acontece para as aplicações de correio eletrônico, objetos carregados em páginas da www e aplicações bancárias. Já para as aplicações em tempo real, um certo limite de largura de banda mínima é necessário para serem executadas. Abaixo do mínimo considerado podem se tornar inviáveis.

As aplicações de transferência de arquivos, correio eletrônico e objetos da www não são sensíveis a atrasos, justamente por serem aplicações assíncronas. Já as aplicações síncronas, ou de tempo real, como áudio, vídeo e jogos, são sensíveis a atrasos na transmissão e podem ser prejudicadas, dependendo do atraso. Um limite máximo de atraso deve ser considerado para garantir a boa experiência do usuário na utilização do serviço. Aplicações bancárias, dependendo da transação, podem sofrer erros críticos originados do atraso, já que trabalham com curtos períodos de sessão devido a questões de segurança.

Observação

Atraso é considerado em geral um problema para muitas aplicações. Entretanto, a variação do atraso, também chamado de *jitter*, em geral é pior, pois reduz a qualidade do serviço (QoS) na transmissão.

Qualidade de Serviço – QoS

O termo QoS (acrônimo de "*Quality of Service*") ou "Qualidade de Serviço", em português, refere-se à capacidade de fornecer um serviço com qualidade que atenda às exigências em matéria de perdas, tempos de resposta e de banda concorrida.

Associado ao QoS temos o termo **nível de serviço** (*service level*), que define o nível de exigência para a capacidade de uma rede de fornecer um serviço fim a fim, ou seja, de um hospedeiro a outro na rede com velocidade e nível de perdas controlados. Geralmente são considerados três níveis de QoS:

- **Melhor esforço** (em inglês *best effort*), que não fornece nenhuma diferenciação entre as várias redes e, por consequência, não permite nenhuma garantia, já que não se sabe qual será o seu caminho na nuvem. Este nível de serviço é também chamado *lack of QoS*.

- **Serviço diferenciado** (*differenciated service* ou *soft QoS*), que permite definir níveis de prioridade num caminho predeterminado na nuvem, sem contudo fornecer uma garantia estrita, já que não há reserva de recursos e sim priorização.

- **Serviço garantido** (em inglês *guaranteed service* ou *hard QoS*), que consiste em reservar recursos na rede para certos tipos de fluxos. O principal mecanismo utilizado para obter tal nível de serviço

é o RSVP (Resource reSerVation Protocol, que se pode traduzir como Protocolo de Reserva de Recursos).

Problemas relacionados às transmissões

Existem alguns problemas que podem acontecer durante as transmissões na *web* e podem ter as mais variadas causas. Vejamos a seguir.

*Perda de Pacote (*Dropped Packets*)*: os roteadores lidam com um número gigantesco de *bits* a cada segundo, são centenas de milhares de pacotes que chegam e que eventualmente podem ultrapassar a capacidade de processamento do equipamento.

Nesses casos, os pacotes são armazenados em *buffers* que formam um fila aguardando para serem processados. Algumas vezes essa fila no *buffer* ultrapassa a capacidade de armazenagem, a partir desse momento os novos pacotes serão descartados.

Atraso: o mesmo vale para atraso, mesmo que o pacote não seja descartado, o tempo em que esteve aguardando para ser processado gerou demora no seu envio e consequente atraso. São muitos os motivos para gerar atrasos na recepção, alguns deles são inerentes à própria rede. A esse atraso damos o nome de "atraso fim a fim", que engloba o tempo de processamento dos pacotes nos roteadores, o tempo de transmissão (uma série de *bits* que formam um pacote só poderá ser transmitida até o próximo roteador quando todos tiverem sido recebidos) e o tempo de propagação (pulso elétrico percorre o meio físico).

O atraso é imprevisível e variável de acordo com o momento da rede.

Jitter: acontece quando os pacotes de uma determinada fonte chegam ao seu destino com diferentes atrasos. Este é o inimigo número um da qualidade do *streaming* de vídeo ou do serviço VoIP.

Out-of-Order Delivery: quando é roteado através da internet, um conjunto de pacotes pode tomar diversas rotas de acordo com o momento da rede, já que os pacotes são sempre direcionados para o caminho com menos tráfego. Isso pode fazer com que a entrega seja feita fora da sequência. Este problema obrigou que fossem criados protocolos especiais complementares para rearranjar os pacotes "desordenados".

Erro: muitas vezes os pacotes perdem seu caminho para o destino ou simplesmente sofrem interferências do meio de transmissão e acabam corrompidos. O receptor detecta esse erro e descarta o pacote, enviando à origem uma solicitação de reenvio.

Saiba mais

Leia o artigo "Afinal, o que é Qualidade de Serviço?", de José Mauricio Santos Pinheiro (2004) para entender um pouco mais sobre o tema. Ele está disponível no seguinte endereço:

<http://www.projetoderedes.com.br/artigos/artigo_qualidade_servico.php>.

Você pode ainda ler o livro *Redes convergentes - entenda a evolução das redes de telecomunicações a caminho da convergência*, de José Umberto Sverzut.

As seções a seguir detalharão as principais aplicações consideradas na Camada de Aplicação.

3.4 WWW: a *World Wide Web* e o protocolo HTTP

No início da década de 1990, entrou em cena a aplicação-chave da internet: a World Wide Web ou www. Ela foi considerada a terceira grande tecnologia de comunicação que transformou drasticamente a maneira como as pessoas interagem dentro e fora de seus ambientes de trabalho. O rádio e a televisão foram as duas primeiras, respectivamente. Essas, sem dúvida, trouxeram grandes mudanças às pessoas, além de inovações na forma de pensar e agir, mas têm o inconveniente de forçar o usuário a sintonizar quando o provedor disponibiliza o conteúdo.

A principal característica da *web* é a sua capacidade de prover conteúdo sob demanda, o que revolucionou nossa maneira de interagir com os diversos conteúdos oferecidos por ela. À medida que o usuário deseja, ele acessa as informações que quiser, quando quiser, e elas estarão sempre disponíveis, diferentemente da transmissão de rádio e televisão, que o obriga a receber o conteúdo transmitido naquele canal e horário. A *web* é formada por diversos *hiperlinks* e dispositivos de busca e possui interação com páginas e *sites* a um baixo custo.

Como grande parte das aplicações, dizemos que esta também está implementada em dois programas: um cliente e um servidor, executados em diferentes sistemas finais, e eles "conversam" pela troca de mensagens **HTTP** (Hypertext Transfer Protocol). Esse é o protocolo das aplicações *web*.

A www é formada por diversas páginas *web*. Quase todas as páginas consistem de pelo menos uma página-base HTML e vários objetos referenciados nela. Um objeto é simplesmente um arquivo que pode ser acessado com uma única URL (Uniform Resource Locator). Por exemplo, se uma página *web* contiver um texto HTML e três imagens, então ela terá quatro objetos: o arquivo-base HTML e mais três imagens. O arquivo-base referencia os outros objetos na página por meio dos URLs dos objetos.

Dizemos que cada URL possui dois componentes: o nome de hospedeiro do servidor, que abriga o objeto, e o nome de caminho do objeto. Por exemplo, na URL *www.unip.br/algum-depto/pic.gif*, *www.unip.br* é o nome do hospedeiro e */algum-depto/pic.gif* é o nome do caminho.

O agente de usuário para www é chamado navegador (ou *browser*). Alguns exemplos de navegadores comuns são: MS Internet Explorer, Mozilla Firefox, Chrome, Safari, entre outros. Servidor para www se chama "servidor *web*". São servidores *web* populares o Apache, de domínio público, e o Microsoft Internet Information Server (IIS), da gigante Microsoft.

Você sabia?

Participação de cada um dos navegadores no mercado

Segundo a Net Market Share (Usage Share Statistics for Internet Technologies), um levantamento feito em maio de 2011 ainda aponta o Internet Explorer, da Microsoft, como o navegador mais utilizado. Veja no gráfico abaixo como o mercado de navegadores está dividido:

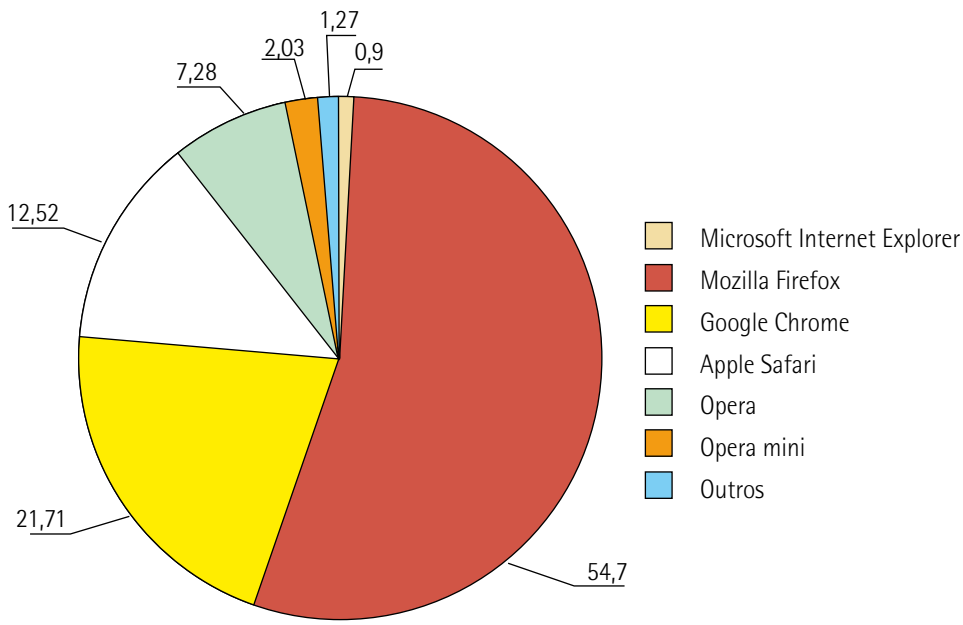

Figura 31 - Participação de cada um dos navegadores no mercado

Como já mencionamos, o protocolo HTTP da Camada de Aplicação para *web* define a estrutura de troca de mensagens. Ele identifica como os clientes requisitam páginas aos servidores e como eles as transferem aos clientes. Na troca de mensagens cliente-servidor, dizemos que o cliente é o navegador que pede, recebe e exibe objetos www, enquanto o servidor *web* envia objetos em resposta aos pedidos recebidos dos clientes, como mostrado na figura 32:

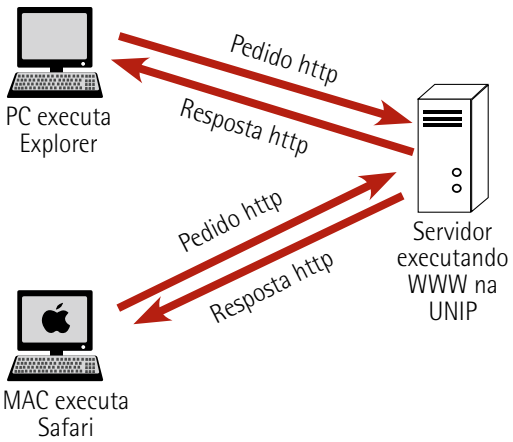

Figura 32 - Pedido e resposta HTTP (KUROSE; ROSS, 2010)

O protocolo HTTP usa o TCP como protocolo de transporte e segue a seguinte rotina para troca de mensagens:

1. O cliente inicia conexão TCP (cria *socket*) com o servidor, na porta 80 (porta padrão do HTTP).

2. O servidor aceita a conexão TCP do cliente.

3. As mensagens HTTP (mensagens do protocolo da Camada de Aplicação) são trocadas entre os navegadores (clientes HTTP) e servidores *web* (servidores HTTP).

4. A conexão TCP é encerrada.

Dizemos que o protocolo HTTP é "sem estado", ou seja, o servidor não mantém nenhuma informação sobre pedidos anteriores do cliente. Assim, a cada vez que um objeto é solicitado, ele o reenvia, pois é como se tivesse esquecido completamente o que fez antes. Protocolos que mantêm "estado" são complexos, pois o histórico (estado) tem que ser guardado. Caso a conexão cliente-servidor seja interrompida, suas visões do "estado" podem ser inconsistentes e devem ser reconciliadas.

Dependendo da aplicação que está sendo utilizada e de que forma as requisições HTTP em uma aplicação *web* serão executadas em cima do protocolo TCP, o criador da aplicação pode decidir se os pares de requisição/resposta devem utilizar apenas uma conexão TCP para transferir todas as páginas *web* (conexão persistente) ou se as requisições devem ser enviadas por conexões TCP distintas (conexões não persistentes). Embora o HTTP utilize conexões persistentes em seu modo padrão, os clientes e os servidores podem utilizar conexões não persistentes. Cada uma tem suas vantagens e desvantagens, que entenderemos a seguir.

Percorrendo as etapas de uma página *web* na sua transferência de um servidor para um cliente, vejamos o que acontece com as conexões não persistentes. Suponhamos que usuário digite a URL: www.unip.br/arquitetura/inicial.index, que contém texto, referências e 10 imagens jpeg:

1a. O cliente http inicia conexão TCP com o servidor http (processo), www.unip.br. A porta 80 é padrão para o servidor http.

1b. O servidor http no hospedeiro www.unip.br espera por conexão TCP na porta 80. "Aceita" a conexão, avisando ao cliente.

2. O cliente http envia mensagem de pedido de http (contendo URL) através do *socket* da conexão TCP.

3. O servidor http recebe mensagem de pedido, formula mensagem de resposta contendo objeto solicitado (arquitetura/inicial.index) e envia mensagem via *socket*.

4. O servidor http encerra conexão TCP.

5. O cliente http recebe mensagem de resposta contendo arquivo html, visualiza html. Analisando arquivo html, encontra 10 objetos jpeg referenciados.

6. Os passos 1 a 5 são repetidos para cada um dos 10 objetos jpeg.

Assim, foi possível perceber que as conexões TCP foram encerradas logo após o servidor enviar o objeto, ou seja, a conexão não persiste para outros objetos. Além disso, vimos que a cada conexão são transferidas exatamente uma mensagem de requisição e uma mensagem de resposta. As conexões não persistentes utilizam a versão 1.0 do protocolo HTTP. Uma desvantagem é que cada objeto sofre partida lenta para ser transportado, já que depende de uma nova conexão TCP.

Observação

Embora a porta 80 seja a porta padrão dos servidores *web*, é possível configurar um servidor *web* para usar qualquer outra porta TCP. Nestes casos, torna-se necessário especificar a porta ao acessar o site.

Nas conexões persistentes, o servidor mantém a conexão TCP aberta após enviar a resposta para que as próximas conexões possam ser enviadas por meio da mesma conexão. O servidor HTTP se encarrega de fechar uma conexão TCP, que não é utilizada a certo intervalo de tempo (pausa) configurável. Ao receber requisições sem interrupções, os objetos são enviados em sequência ou até mesmo em paralelo. Uma desvantagem das conexões persistentes, utilizadas a partir da versão 1.1 do protocolo HTTP, é que as conexões estabelecidas, ao serem mantidas para cada objeto solicitado, podem sobrecarregar o servidor *web*. Devem ser alocados *buffers* e conservadas algumas informações da conexão TCP tanto no lado cliente como no lado servidor.

Proxy/Cache Web

Em linhas gerais, toda vez que você acessa uma página *web* é estabelecida uma conexão TCP com o servidor hospedeiro, que envia os arquivos solicitados para seu *browser* e este monta a página visitada.

Como você viu no texto anterior, essas conexões são do tipo persistente, que consomem recursos do servidor em questão. Agora, por exemplo, imagine uma empresa que tenha 1.000 funcionários e que todos os dias esses funcionários acessem o Google cinco vezes ao dia. Você deve conhecer a página inicial do Google e sabe que ela tem uma caixa de texto, um botão e o logotipo da empresa. Portanto, a cada nova visita de um funcionário, os servidores do Google vão estabelecer uma conexão TCP persistente e enviar esses três objetos. Até aí nenhum problema, certo? Certo.

Os servidores do Google estão dimensionados para atender quantos usuários queiram acessá-los. O problema está no tráfego de dados na internet dessa empresa. Vamos calcular: cada um dos 1.000 funcionários acessando cinco vezes ao dia o *site* do Google, requisitando os mesmos três objetos que têm 49 Kb, dá um total, por dia, de incríveis 245.000 Kb ou 245 Mb de dados trafegados somente com a página inicial do Google. Assustador, não?

Veja o volume de ocupação da banda com a mesma informação, aqueles mesmos três objetos representariam por volta de 10% do consumo de uma empresa que tivesse uma banda de 20 Mbps.

Um desperdício!

Para contornar esse desperdício foi criado o servidor Proxy, também conhecido por Cache Web. Em resumo, toda vez que um usuário solicita uma página *web*, antes do Proxy solicitar ao servidor de destino os arquivos, ele estabelece a conexão TCP e envia um código solicitando somente os nomes e datas de alteração dos arquivos da página *web* em questão. Depois disso, verifica se tem esses arquivos armazenados em seus registros e suas datas de modificação. Caso sejam válidos, encaminha-os aos usuários sem necessidade de tráfego na *web*.

Lembrete

Cache é mais um exemplo de aplicação cliente e servidor. Quando recebe requisições de um navegador e lhe envia respostas, é um **servidor**. Quando envia requisições para um servidor de origem e recebe respostas dele, é um **cliente** (KUROSE, 2010).

3.5 Transferência de arquivos e o FTP

A transferência de arquivos é uma aplicação que existe desde 1971, quando a internet ainda era uma experiência. FTP (File Transfer Protocol) é o protocolo utilizado para transferir um arquivo de um hospedeiro a outro.

Veja na figura 33 como funciona uma sessão FTP: para o usuário transferir arquivos, ao digitar no navegador o endereço FTP, ele deve fornecer uma identificação e uma senha. Assim que o servidor autorizar o usuário, ele copiará um ou mais arquivos armazenados no seu sistema de arquivos local para o sistema de arquivos remoto (ou vice-versa).

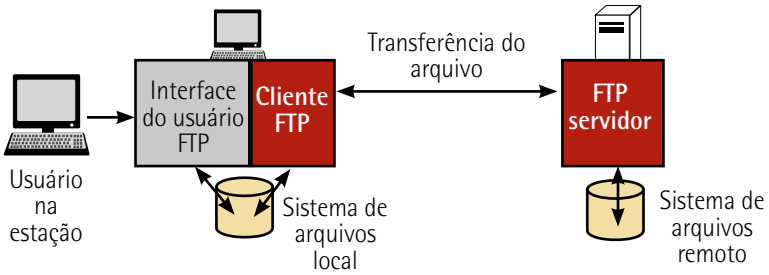

Figura 33 - Exemplo de uma sessão FTP (KUROSE; ROSS, 2010)

O FTP se assemelha muito ao HTTP. A diferença notável é que o FTP utiliza duas conexões TCP paralelas, enquanto o HTTP utiliza apenas uma, como apresentado na figura 34. As duas conexões paralelas fazem uso de duas portas diferentes e servem para transporte de informações distintas: uma delas transporta apenas informações de controle, como os dados de autenticação (usuário e senha) e o diretório que se deseja manusear, enquanto a outra transporta os arquivos que serão trocados. No modelo cliente-servidor, é considerado cliente o lado que inicia a transferência (de ou para um sistema remoto) e considerado servidor o hospedeiro remoto.

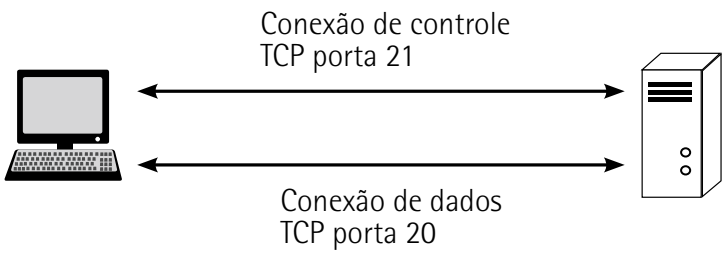

Figura 34 - Duas conexões paralelas FTP (KOVACH, 2009)

Resumindo, o processo de transferência de dados FTP pode ser realizado da seguinte maneira:

1. O cliente FTP contata o servidor FTP, na porta 21, especificando o TCP como protocolo de transporte.

2. São abertas duas conexões TCP paralelas:

- *controle*: para troca comandos, respostas entre cliente e servidor (porta 21);
- *dados*: dados de arquivo de/para servidor (porta 20).

3. O servidor FTP mantém o "estado" em relação ao usuário: ele associa a conexão de controle à autenticação.

4. Caso o navegador seja fechado e reaberto em um intervalo curto de tempo, os dados de controle estarão mantidos, não sendo necessária nova autenticação.

Lembrete

O FTP utiliza o protocolo de transporte TCP nas suas transmissões. Ele utiliza as portas de número 20 e 21 para executar as conexões TCP na transferência de arquivos.

3.6 Correio eletrônico e seus protocolos

A aplicação de correio eletrônico foi a primeira que usou as redes de computadores, fazendo com que as pessoas passassem a utilizá-las intensamente. Ela existe desde o início da internet e é composta de três grandes elementos: agentes de usuário, servidores de correio e protocolos.

Saiba mais

Assista a uma das primeiras reportagens sobre correio eletrônico que mostra que, no início dos anos 1990, mandar mensagens pela internet era coisa de aficionado de computador. Acesse <http://vimeo.com/32171111>

A figura 35 mostra os elementos presentes em uma aplicação de correio eletrônico. Como falamos anteriormente, o agente de usuário desta aplicação é o leitor de correio, que serve para compor, editar e ler mensagens de correio. São exemplos de leitores de correio: Microsoft Outlook, Netscape Comunicator, Gmail, entre outros.

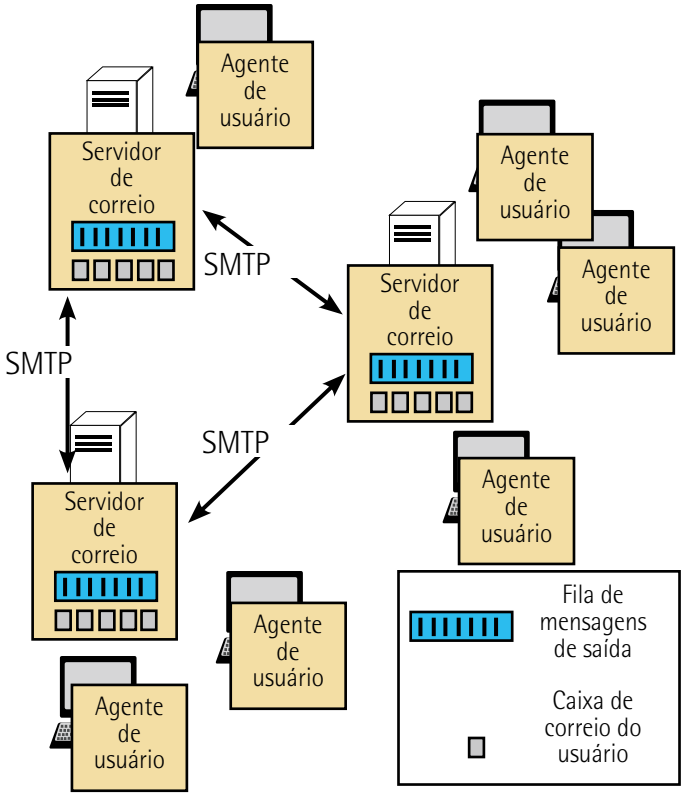

Figura 35 - Elementos que compõem os correios eletrônicos (KUROSE; ROSS, 2010)

As mensagens de saída e chegada são armazenadas no servidor. A caixa de correio contém mensagens de chegada (ainda não lidas) para o usuário. A fila de mensagens contém mensagens de saída (a serem enviadas), e o protocolo SMTP é utilizado na comunicação entre servidores de correio para transferir mensagens de correio.

Conheça!

Clientes de *e-mail* no mercado

A empresa Campaign Monitor desenvolveu um método para descobrir qual cliente de *e-mail* está sendo mais utilizado pelas pessoas. Para isso envia *e-mails* diários com imagens e, quando o cliente de *e-mail* solicita ao servidor da empresa a imagem, é possível detectar, via protocolo TCP, qual é esse cliente de *e-mail*.

Essa solução é bastante inteligente, mas, como a empresa mesmo admite, gera uma distorção, inflando os clientes de *e-mail* que exibem imagens automaticamente por padrão, como o Outlook 2000 e o iPhone. E também reduz o número para aqueles que bloqueiam as imagens por padrão, como o Gmail e Outlook 2007. Os clientes de *e-mail* que não são capazes de exibir imagens, como os modelos mais antigos do Blackberry e outros dispositivos móveis, não foram considerados neste estudo.

De qualquer maneira, esse é o estudo mais abrangente já feito até hoje e pode dar pistas do que está acontecendo no mercado de *e-mails* atual.

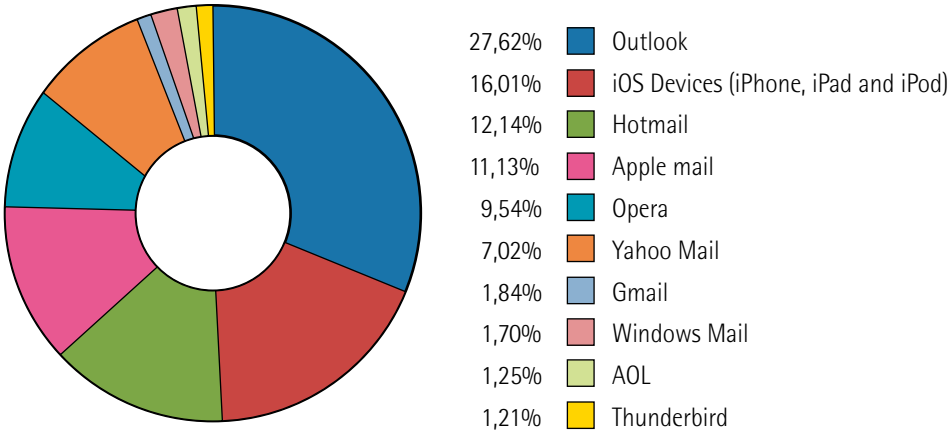

Figura 36

Saiba mais

Conheça a empresa Campaign Monitor e veja outros resultados de estudos estatísticos sobre a área levantados por ela. Acesse <http://www.campaignmonitor.com>.

A aplicação de correio eletrônico também é uma aplicação do tipo cliente-servidor. Nesse caso, o **cliente** é o servidor de correio que envia mensagens e o **servidor** é o servidor de correio que recebe mensagens. Os protocolos dessa aplicação utilizam o protocolo TCP para a transferência confiável de mensagens do correio do cliente ao servidor, na porta 25, por meio das três fases da transferência: *handshaking* (cumprimento), transferência das mensagens e encerramento.

Talvez você não saiba, mas o correio eletrônico surgiu antes mesmo da internet. A aplicação de *e-mail* foi o primeiro passo para a criação e o desenvolvimento da rede internacional de computadores.

Em 1965 surgiu o primeiro sistema de troca de mensagens eletrônicas de que se tem notícia e permitia a comunicação de computadores do tipo *mainframe*.

Segundo um relato existente, em 1969 houve a primeira troca de mensagens de correio eletrônico na rede ARPANET. A mensagem seguiu do computador do laboratório de Kleinrock na UCLA para o de Douglas Engelbart no *Stanford Research Institute*, passando pela rede da ARPA (Advanced Research Projects Agency).

Foi o programador Ray Tomlinson que em 1971 iniciou o uso do sinal @ para separar os nomes do usuário e da máquina no endereço de correio eletrônico.

Os protocolos de correio eletrônico mais importantes são os citados a seguir:

- SMTP (Simple Mail Transfer Protocol): utilizado na entrega e no armazenamento de mensagens no servidor do receptor.

- Protocolos de acesso ao correio: recuperam mensagens do servidor:

  – POP (Post Office Protocol);

  – IMAP (Internet Mail Access Protocol): mais comandos que o POP, porém mais complexo com relação ao manuseio de mensagens armazenadas no servidor;

  – HTTP (Hypertext Transfer Protocol): apresenta mensagens recuperadas do servidor através de páginas *web*. Exemplos: Gmail, Hotmail, Yahoo! Mail, Webmail etc.

Veja agora o que acontece quando Maria envia um *e-mail* para João e em que momento cada um desses protocolos são utilizados durante a transmissão:

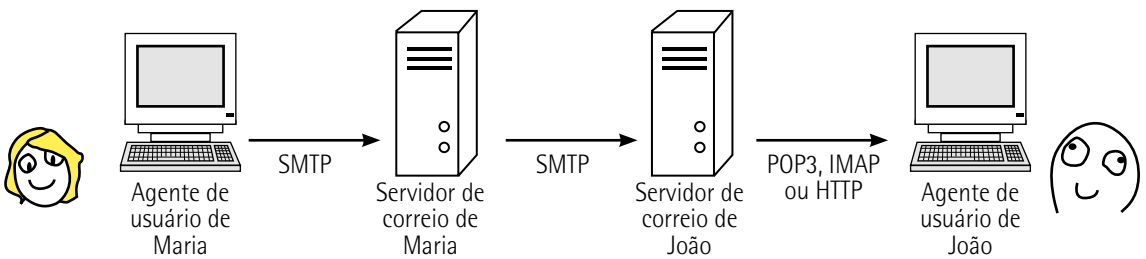

Figura 37 - Protocolos de correio eletrônico utilizados em cada etapa da interação (KUROSE; ROSS, 2010)

1. Maria utiliza sua interface de correio eletrônico ou agente de usuário para escrever sua mensagem. Assim, ela fornece o endereço do João, compõe uma mensagem e dá instruções ao agente de usuário para que ele possa enviá-la. Ao enviar a mensagem, o agente de usuário o faz utilizando o protocolo SMTP.

2. O agente de usuário de Maria envia a mensagem para seu servidor de correio, também através do protocolo SMTP, onde ela é colocada na fila de mensagens.

3. O lado cliente do SMTP, ou seja, que roda no servidor de correio de Maria, vê a mensagem na fila e abre uma conexão TCP para um servidor SMTP, que roda no servidor de correio de João.

4. O cliente SMTP envia a mensagem de Maria por meio da conexão TCP.

5. No servidor de correio de João, o servidor SMTP recebe a mensagem e a coloca na caixa postal de João.

6. João utiliza seu agente de usuário para ler as mensagens recebidas.

Observação

O correio eletrônico utiliza o protocolo TCP da Camada de Transporte para o envio de mensagens. A conexão TCP para esta aplicação utiliza a porta de número 25.

Primeira mensagem de correio eletrônico

Está publicada em uma reportagem no Diário Digital de Portugal a primeira mensagem de correio eletrônico enviada há mais de 40 anos:

Primeira mensagem de correio electrónico enviada há quarenta anos

A primeira mensagem de correio electrónico entre dois computadores (*e-mail* em rede) situados em locais distantes foi enviada em 29 de outubro de 1969, quase dois meses depois do primeiro nó que deu origem à Internet.

O texto dessa primeira mensagem continha apenas duas letras e um ponto - "LO.". O investigador da Universidade da Califórnia em Los Angeles (UCLA) Leonard Kleinrock queria escrever "LOGIN", mas o sistema foi abaixo a meio da transmissão.

A mensagem seguiu do computador do laboratório de Kleinrock na UCLA para o de Douglas Engelbart no Stanford Research Institute, utilizando como suporte a recém-criada rede da ARPA (Advanced Research Projects Agency), agência financiada pelo governo norte-americano.

O primeiro nó de ligação entre dois computadores da Arpanet tinha sido estabelecido pouco tempo antes, em 02 de setembro de 1969, pelo que a história da Internet e do *e-mail* em rede se confundem.

No início da década de 1960, surgiram alguns sistemas de troca de mensagens entre terminais de um mesmo computador, em tempo diferido (1961) e em tempo real (1965), mas o primeiro *e-mail* em rede foi transmitido apenas em 1969.

Dois anos depois, em 1971, Ray Tomlinson inventou os primeiros programas para envio de *e-mail*s em rede através da Arpanet e criou a arroba ("at", em Inglês - @) para separar o login do utilizador do domínio do servidor.

Em 1976, a rainha de Inglaterra, Isabel II, enviou o seu primeiro *e-mail*, e em 1978 surgiu o primeiro *spam*, entendido como mensagem de correio electrónico enviada para múltiplos destinatários sem consentimento destes.

> Quarenta anos depois, 70% dos *e-mail*s enviados diariamente são *spam*, uma "praga" que acompanha o crescimento dos vírus e do *marketing* na internet, mas que tem sido combatida, com relativo sucesso, por diversos sistemas de filtragem entretanto desenvolvidos.
>
> [...]
>
> "Pela sua formalidade, o *e-mail* é algo pouco apelativo para os utilizadores mais jovens. Os *blogs* e o Twitter ocupam um espaço menos informal", disse à agência Lusa Libório Silva, autor do livro sobre correio electrónico mais vendido em Portugal, "*e-mail*", editado em 2008.
>
> Libório Silva afirmou que o *e-mail* continua a ser uma ferramenta em expansão em todo o mundo, pela facilidade de utilização e pela capacidade de envio de ficheiros associados a mensagens.
>
> Libório Silva destacou como principal ruptura na história do *e-mail* o surgimento dos serviços de webmail, que são actualmente líderes de mercado entre utilizadores individuais, mas não nas empresas, que continuam a preferir servidores internos.[1]

3.7 Serviços de diretório de nomes – DNS

Podemos iniciar esta seção fazendo uma nova analogia, dizendo que as pessoas possuem muitos identificadores, por exemplo, nome, número de CPF, número de identidade, número identificador nas escolas, entre outros. Um identificador pode ser mais adequado que outro, dependendo do uso, ou seja, para seu banco, você é identificado por seu número de CPF, mas, na sua escola, você é identificado por um número que o identifica como aluno.

Na internet, dizemos que hospedeiros e roteadores devem ser identificados pelo nome do hospedeiro, como www.unip.com.br, www.google.com.br e www.yahoo.com.br. Esses nomes são fáceis de lembrar, por isso nós gostamos deles. Por meio desses endereços, não é possível identificar a localização desses hospedeiros na internet. Pelo final **br** sabemos que provavelmente esse servidor está no Brasil, mas essa é a única informação que temos.

Esses nomes de hospedeiros são os mais variados, podendo ter um tamanho grande ou, inclusive, conter caracteres alfanuméricos, que seriam difíceis para os roteadores que devem encaminhar as mensagens aos hospedeiros e identificá-los através de tais nomes. Assim, roteadores não identificam hospedeiros por meio de nomes, mas sim por meio de endereços IP. Nesse momento, a única informação que devemos ter em mente é que os endereços IP possuem um formato padrão em sua versão – 4, por ser formado de um conjunto de 4 *bytes*. Falaremos mais detalhadamente sobre endereços IP na Unidade III.

[1] Disponível em: <http://diariodigital.sapo.pt/news.asp?section_id=18&id_news=417591>. Acesso em: 27 abr. 2012.

Assim, há duas maneiras de identificar os hospedeiros na internet: por meio de seu **nome**, como nós preferimos, e por meio de **endereços IP**, que é como os roteadores preferem. A fim de conciliar essas preferências surge a aplicação de serviço de diretório de nomes ou sistema de nomes de domínio (DNS – Domain Name System), que tem por objetivo traduzir os nomes dados aos hospedeiros para endereços IP.

Dizemos que o DNS é uma base de dados distribuída e implementada na hierarquia de muitos servidores de nomes, como mostra o exemplo da figura 38:

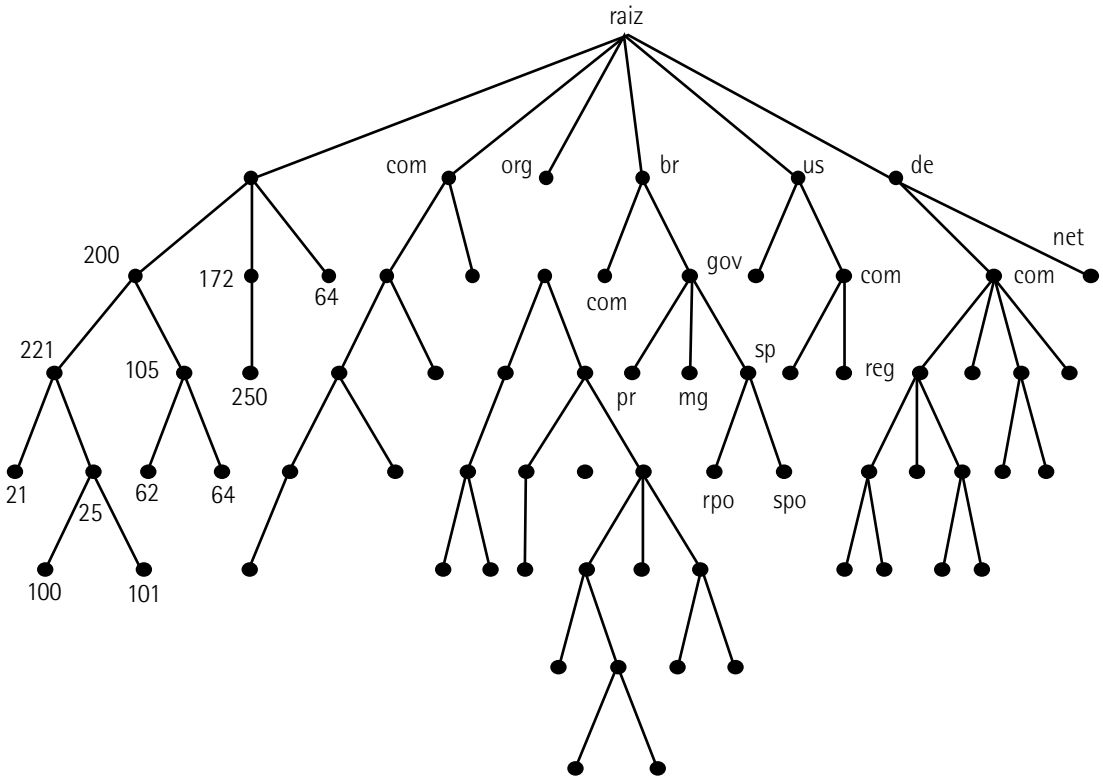

Figura 38 - Hierarquia do serviço de nomes (elaborada pela autora)

O protocolo da Camada de Aplicação permite que hospedeiros, roteadores e servidores de nomes se comuniquem para resolver nomes, ou seja, fazer o serviço de **tradução** de nome para endereço IP. Embora muitos não tenham clara essa informação, é importante salientar que o DNS é função imprescindível da internet e é implementado como protocolo de Camada de Aplicação. O DNS é um serviço que roda sobre UDP e TCP e utiliza a porta 53 (KUROSE, 2010).

O DNS provê outros serviços importantes, além da tradução de nomes de hospedeiros para endereços IP. Vejamos:

- **Apelidos de hospedeiros**: também chamado de *alias* (pseudônimo, em inglês), o nome do hospedeiro (também conhecido como nome canônico) pode ter mais de um apelido. Os apelidos existem para facilitar a lembrança pelo nome canônico, por serem mais fáceis de lembrar. Por exemplo, se digitarmos http://www.folha.uol.com.br ou www.folha.com.br, ambas as formas devem retornar à mesma página *web*.

- **Apelidos de servidor de correio eletrônico**: pode-se obter o nome canônico a partir do apelido, ou seja, como é adequado que endereços de *e-mail* sejam fáceis de ser lembrados. Assim, o DNS pode ser chamado por uma aplicação de correio para obter o nome canônico a partir de um apelido fornecido, assim posso enviar uma mensagem para maria@hotmail.com, mas na verdade o nome canônico é maria@relay1.hotmail.com. Diversos registros de nomes podem ter o mesmo apelido.

- **Distribuição de carga**: realizada entre servidores replicados, trata-se de um conjunto de endereços IP associado a um único nome canônico. As respostas do DNS contêm o conjunto de endereços (é feito um rodízio na ordem dos endereços) para *sites* movimentados que necessitam distribuir as requisições dos clientes aos servidores replicados, diminuindo o tráfego de informações.

O DNS é um sistema complexo, mas neste livro-texto mencionaremos apenas os aspectos fundamentais de sua operação.

Observação

TCP e UDP são protocolos de Camada de Transporte e serão detalhados mais adiante. Algumas aplicações, como é o caso do DNS, utilizam ambos para executar seus serviços.

Um exemplo pode ser visualizado na figura 39:

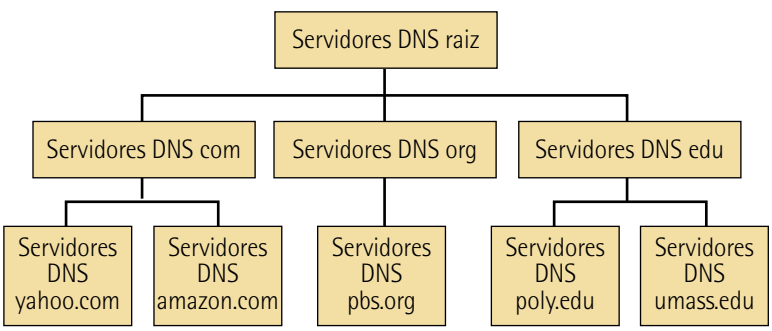

Figura 39 - Servidores de DNS (KUROSE; ROSS, 2010)

Antigamente, os hospedeiros e seus endereços eram armazenados somente em um servidor e em um único arquivo hosts.txt, que centralizava todas as informações para que as traduções fossem executadas. Com as informações concentradas em um único lugar, percebeu-se que era um único ponto de falha e que, se houvesse algum problema com o servidor de nomes, toda a internet ficaria sem comunicação. Além disso, esse formato fazia com que o volume de tráfego se tornasse muito grande, pois era necessário manipular todas as consultas DNS de milhões de hospedeiros. Com uma única base de dados, ela jamais estaria perto de todos os clientes que fizessem consultas e eventualmente precisariam viajar desnecessariamente até o outro lado do globo para que a tradução fosse executada.

A manutenção dessa base de dados centralizada também não era nada simples, já que atualizações eram necessárias para novas entradas de dados na base. Com a descrição desse cenário, torna-se óbvio que a centralização do DNS em uma única base de dados não é escalável.

Saiba mais

Veja uma animação sobre o funcionamento desta aplicação no DNS Explained – CENTR. Acesse diretamente o *link*:

<http://www.centr.org/main/6200-CTR/5418-CTR.html>.

Atualmente, nenhum servidor mantém todos os mapeamentos de nomes para um endereço IP. Existe uma base de dados distribuída e hierárquica ao redor de todo o mundo. Veja os tipos de servidor-padrão considerados:

- **Servidor de nomes local**: cada provedor ou empresa tem um servidor de nomes local (padrão). O pedido DNS de hospedeiro vai primeiro ao servidor de nomes local.
- **Servidor de nomes oficial**: para hospedeiros, guarda o nome e o endereço IP deles e pode realizar tradução nome/endereço para esse nome.
- **Servidores de domínio de alto nível** (Top-Level Domain – TLD): responsáveis por domínios de alto nível, genéricos e de países, como com, org, net, edu, gov, br, uk, ca etc.
- **Servidores de nomes com autoridade**: responsáveis por domínios das organizações e domínios de segundo nível.

Kurose (2010) mostra um exemplo claro das interações entre os servidores DNS, no momento de uma requisição:

Suponha que a Universidade de Massachusetts tenha um servidor de nomes para a universidade, denominado dns.umass.edu. Imagine também que cada um dos departamentos da universidade tenha seu próprio servidor de nomes e que cada servidor de nomes departamental seja um servidor de nomes com autoridade para todos os hospedeiros do departamento. Nesse caso, quando o servidor de nomes intermediário dns.umass.edu receber uma consulta para um hospedeiro cujo nome termina com cs.umass.edu, ele retornará a dns.poly.edu o endereço IP de dns.cs.umass.edu, que tem autoridade para todos os nomes de hospedeiro que terminam com cs.umass.edu. Então, o servidor de nomes local dns.poly.edu enviará a consulta ao servidor de nomes com autoridade, que retornará o mapeamento desejado para o servidor de nomes local e que, por sua vez, o repassará ao hospedeiro requisitante. Nesse caso, serão enviadas 10 mensagens DNS no total. A figura 40 ilustra esse exemplo:

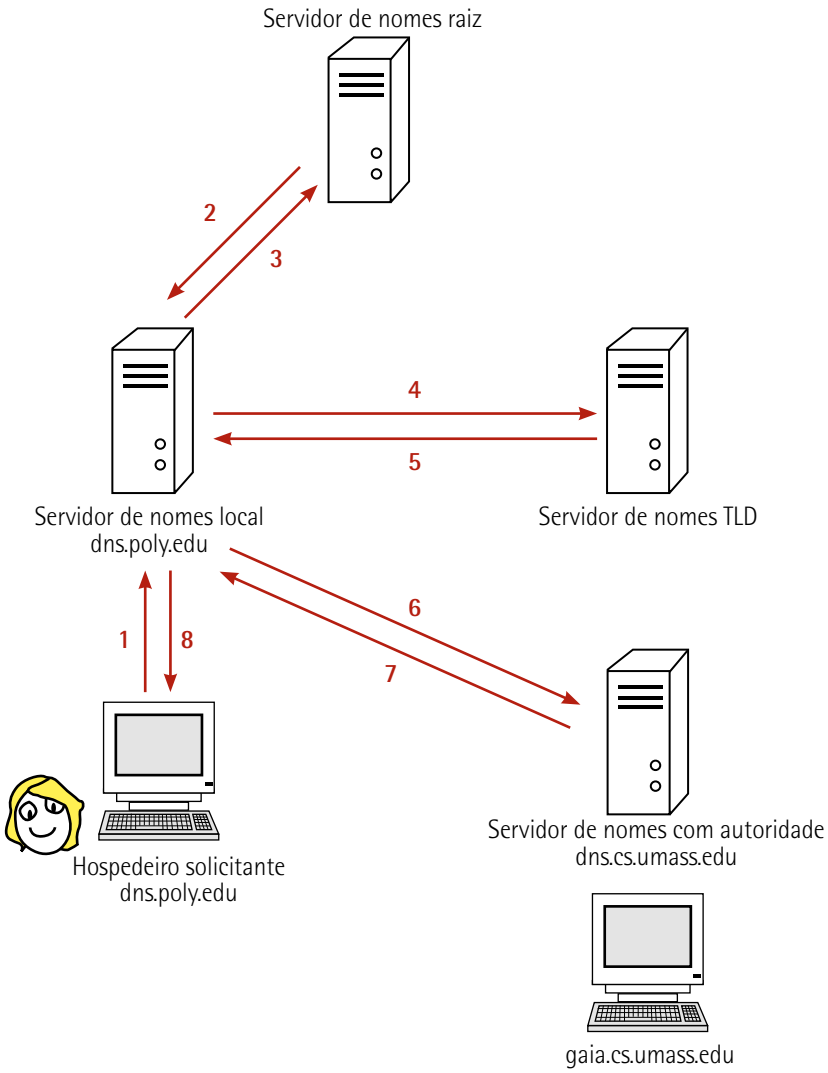

Figura 40 - Interação de vários servidores de DNS (KUROSE; ROSS, 2010)

O espaço de nomes do DNS é dividido em zonas não superpostas (KUROSE, 2010):

- zona que inclui os hospedeiros administrados diretamente por um servidor;
- zona que contém uma parte da árvore e servidores de nomes que armazenam informações referentes à zona;
- zona que contém um servidor principal (obtém suas informações a partir do disco) e servidores secundários (obtém suas informações a partir do servidor principal).

Lembrete

Os servidores de diretórios responsáveis por prover informações como nomes e endereços das máquinas são normalmente chamados servidores de nomes. Na internet, o serviço de nomes usados é o DNS, que apresenta

uma arquitetura cliente/servidor, podendo envolver vários servidores DNS na resposta a uma consulta.

Em 2009, foram computados 13 servidores DNS raiz no mundo todo e sem eles a internet não funcionaria. Veja como estão distribuídos na figura 39. Destes 13, repare na concentração geográfica dos servidores nos Estados Unidos. Dez estão localizados por lá, um na Ásia e dois na Europa.

Figura 41 - Servidores DNS raiz em 2009 - nome, organização e localização (KUROSE; ROSS, 2010)

Para aumentar a base desses servidores, foram criadas réplicas localizadas por todo o mundo, inclusive no Brasil. O site oficial dos servidores DNS raiz disponibiliza um mapa com os servidores DNS distribuídos pelo mundo. A figura 42 foi retirada do site <http://www.root-servers.org> e teve sua última atualização em 2009.

Figura 42 - Servidores DNS em 2009

DNS – um pouco de história

O sistema de distribuição de nomes de domínio teve início em 1984 e com ele os nomes de hospedeiros residentes em um banco de dados puderam ser distribuídos entre servidores múltiplos, diminuindo assim a carga em qualquer servidor que provê administração no sistema de nomeação de domínios.

A implementação do DNS-Berkeley foi desenvolvida originalmente para o sistema operacional BSD UNIX 4.3.

A implementação do Servidor de DNS Microsoft se tornou parte do sistema operacional Windows NT na versão Server 4.0. O DNS passou a ser o serviço de resolução de nomes padrão a partir do Windows 2000 Server, como a maioria das implementações de DNS teve suas raízes nas RFCs 882 e 883, e foi atualizado nas RFCs 1034 e 1035.[2]

4 CAMADA DE APRESENTAÇÃO E SESSÃO

4.1 Camada de Apresentação

Também chamada de Camada de Tradução, essa camada é a segunda camada mais alta da pilha de protocolos, destacada na figura 43. Ela converte o formato do dado recebido pela Camada de Aplicação em um formato comum a ser usado na transmissão desse dado, ou seja, um formato entendido pelo protocolo usado. Ela é responsável pela transformação e formatação de dados e pela sintaxe de seleção.

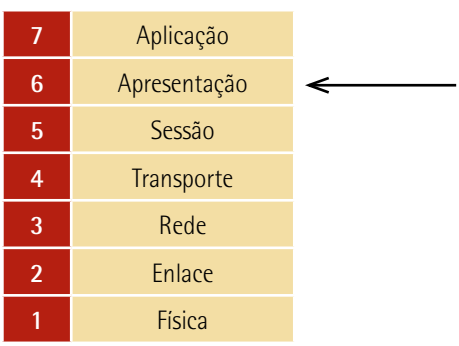

Figura 43 - Camada de Apresentação do modelo OSI (KOVACH, 2009)

4.1.1 Principais funções

Para facilitar o entendimento, um exemplo comum da Camada de Apresentação é a compressão de dados e criptografia.

[2] Fonte: Disponível em: <http://www.inteligensis.pt/bc/bc.htm>. Acesso em: 24 abr. 2012.

> A compressão de dados pega os dados recebidos da camada 7 e os comprime (como se fosse um compactador comumente encontrado em PCs, como o Zip ou o Arj), e a camada 6 do dispositivo receptor fica responsável por descompactar esses dados. Dessa forma, a transmissão dos dados torna-se mais rápida, já que haverá menos dados a serem transmitidos: os dados recebidos da camada 7 foram "encolhidos" e enviados à camada 5.[3]

A Camada de Apresentação pode ainda ter outros usos, como a conversão do padrão de caracteres (código de página), quando, por exemplo, o dispositivo transmissor usa um padrão diferente do ASCII. Exemplos de diferenças entre formatos de dados incluem ordem de *bytes* (poderia ser lido da esquerda para a direita ou vice-versa) e conjunto de caracteres (caracteres ASCII ou conjunto de caracteres EBCDIC, da IBM), bem como diferenças na representação numérica.

Para aumentar a segurança, é possível utilizar algum esquema de criptografia nesse nível, sendo que os dados só serão decodificados na camada 6 do dispositivo receptor. Assim, no dispositivo de origem, a mensagem é enviada criptografada, ou seja, os dados da informação original são modificados em um formato para enviar. A formatação de dados serve para que o nó receptor entenda o que o nó emissor envia. A figura 44 ilustra o momento em que os dados são encriptados na origem, ou seja, codificados para serem transportados através da nuvem sem correr o risco de serem identificados caso haja uma interceptação no meio. Apenas no nível da apresentação do destino é que a informação é decriptada.

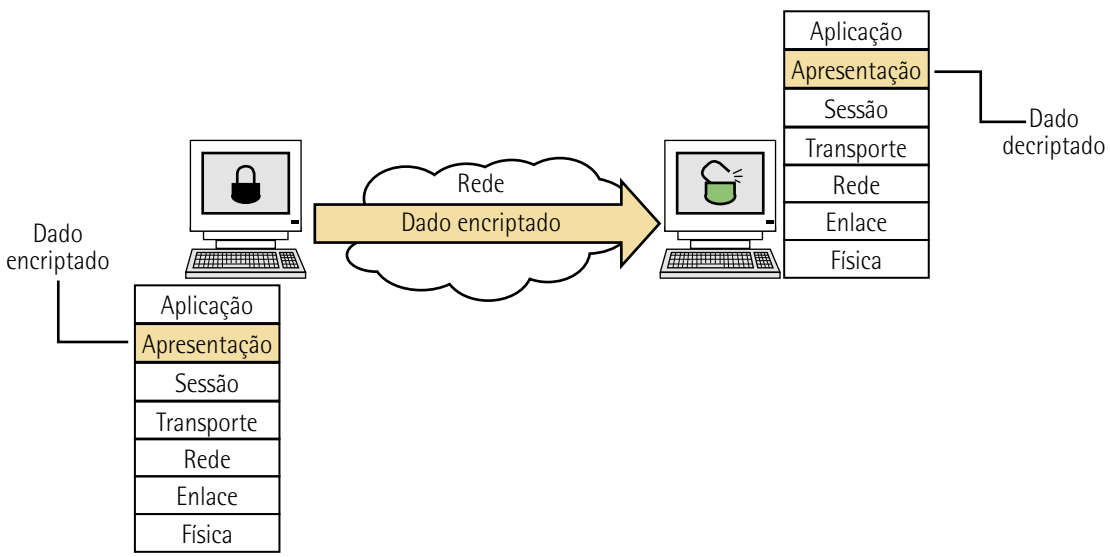

Figura 44 - Exemplo de criptografia na Camada de Apresentação (elaborada pela autora)

4.2 Camada de Sessão

A Camada de Sessão é uma das camadas superiores do modelo OSI e está situada logo abaixo da Camada de Apresentação, destacada na figura 45. Essa camada foi criada pela ISO, não sendo encontrada em redes de computadores que antecedem esse modelo. O principal objetivo da Camada de Sessão é

[3] Disponível em: <http://www.lucalm.hpg.ig.com.br/osi.htm>. Acesso em: 27 abr. 2012.

oferecer às camadas de apresentação cooperantes meios de organizar e sincronizar sua comunicação. Ela permite que duas aplicações em computadores diferentes estabeleçam uma sessão de comunicação.

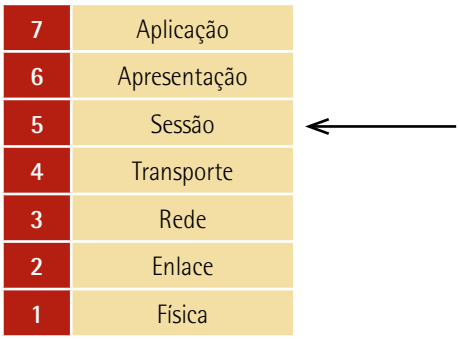

Figura 45 - Camada de Sessão do modelo OSI (KOVACH, 2009)

4.2.1 Visão geral

Como toda estrutura de camadas, a Camada de Sessão oferece seus serviços à Camada de Transporte, que fará a quebra do datagrama, incluindo uma marcação lógica ao longo da transmissão para identificar os blocos já recebidos, e solicitará retransmissão somente das partes necessárias.

> É na Camada de Sessão que as aplicações definem como será feita a transmissão dos dados e adiciona a eles marcações durante a transmissão. Isso porque, se houver em algum momento falha na rede, os computadores que estão no processo de comunicação reiniciam a transmissão dos dados considerando a última marcação recebida pelo computador receptor.
>
> Os protocolos de Camada de Sessão tratam de sincronizações (*checkpoints*) na transferência de arquivos.[4]

4.2.2 Principais serviços

Os principais serviços oferecidos por essa camada consideram:

- *Intercâmbio de dados:* atividade responsável pelo estabelecimento da conexão, troca de dados e fechamento da conexão com a outra ponta;
- *Gerenciamento de diálogos:* através dos chamados *tokens* é possível negociar a troca de dados, sincronização dos dados e a liberação da conexão durante a sessão. Os *tokens* são os responsáveis por "ter a vez de falar". Assim, a máquina que estiver com o *token* é que poderá transmitir naquele momento;
- *Sincronização:* quando há interrupção na rede, uma forma de retornar ao ponto onde parou é através dos chamados "pontos de sincronização" nos diálogos. Os pontos

[4] Disponível em: <http://www.projetoderedes.com.br/artigos/artigo_modelo_osi.php>. Acesso em: 27 abr. 2012.

de sincronização são marcações em dois níveis, para permitir retornar com maior precisão;

- *Gerenciamento de atividades:* divide as mensagens no nível da aplicação em unidades lógicas menores e independentes, chamadas atividades;

- *Relatório de exceções:* permite retomar as ações executadas no nível de sessão por meio de relatórios que detalham os problemas acontecidos com mensagens que retornaram.

Como cada camada oferece seus serviços para a camada diretamente acima, a Camada de Sessão considera o chamado *Ponto de Acesso aos Serviços da Sessão* (PASS), que permite a utilização dos seus serviços pela Camada de Apresentação.

Intercâmbio de dados

Os dados no nível da sessão devem ser organizados para o estabelecimento da comunicação e transmissão adequada. Assim, utiliza o chamado Intercâmbio de Dados nessa ação, que envolve três fases: estabelecimento, utilização e liberação.

O estabelecimento de sessão é feito por meio de um pedido de conexão com a camada de transporte. Envolve a negociação entre os usuários e os diversos parâmetros da conexão. Alguns desses parâmetros são pertinentes à conexão de transporte e são simplesmente passados para essa conexão sem qualquer modificação.[5]

A liberação pode ser feita de duas formas na Camada de Sessão:

- *De forma abrupta*: análoga à desconexão na Camada de Transporte e, uma vez emitida, a conexão não recebe mais nenhum dado. É utilizada para abortar conexões.

- *Disciplinada*: utiliza um *handshake* completo: pedido, indicação, resposta e confirmação. Essa forma de liberação pode aceitar mensagens até que uma confirmação seja enviada.

Gerenciamento de diálogos

Existem muitas situações em que o *software* da camada superior está estruturado de forma a esperar que os usuários se revezem (comunicação *half-duplex*). Para tal, foram introduzidos controles para determinar de quem é a vez de transmitir. O gerenciamento de diálogos foi implementado por meio do uso de *tokens* de dados. Assim, ao se estabelecer uma sessão, pode ser utilizado um parâmetro que indique o modo (*half-duplex*) e outro parâmetro que diga qual dos lados recebe inicialmente o *token*. Somente o usuário que está com o *token* pode transmitir, enviando o *token* para o outro usuário assim que encerrar sua transmissão.

[5] Disponível em: <http://penta2.ufrgs.br/rc952/trab2/sessao2.html>. Acesso em: 27 abr. 2012.

Sincronização

A sincronização é utilizada para devolver às entidades na Camada de Sessão um estado conhecido. Isso pode ser necessário no caso de ocorrerem erros ou divergências. Pode parecer desnecessário, uma vez que a Camada de Transporte cuida dos erros de comunicação, porém podem ocorrer erros na camada superior.[6]

A informação na Camada de Sessão pode ser dividida em páginas. Essas páginas podem ser separadas por pontos de sincronização. Se ocorrer algum problema, é possível reiniciar a partir de um ponto de sincronização anterior (ressincronização). Quando essa ressincronização ocorre, o salvamento de mensagens e a retransmissão subsequente ocorrem acima da Camada de Sessão.

Existem dois tipos de pontos de sincronização: os principais, que delimitam partes logicamente significativas da aplicação, chamadas **unidades de diálogo**, e outros pontos de sincronização secundários. Os pontos de sincronização principais são subdivididos em vários pontos de sincronização secundários. Quando ocorre a interrupção e a necessidade de ressincronização, retorna-se até o ponto de sincronização principal mais recente ou a um ponto de sincronização secundário, desde que este não tenha sido precedido de um ponto principal.

Para a fixação de pontos de sincronização, são utilizados *tokens*. Existem dois *tokens* independentes para o ponto principal e o secundário. Esses *tokens* são distintos entre si e diferentes também dos utilizados para controle de dados na comunicação *half-duplex*.

Gerenciamento de atividades

É utilizado para permitir que o usuário divida o fluxo de mensagens em unidades lógicas (atividades), que é completamente independente de outra subsequente ou anterior. O usuário determina o que deve constituir cada atividade (e não a Camada de Sessão).

Tudo o que a Camada de Sessão faz é transmitir para o receptor as indicações de início, finalização, retomada, interrupção ou descarte de uma atividade. Porém, a Camada de Sessão não sabe quando as solicitações de atividades são feitas e como são as reações do receptor. O gerenciamento de atividades é a forma principal de se estruturar uma sessão. Assim, para que não ocorram pedidos simultâneos de início de atividades, todo gerenciamento é controlado por um *token* (o mesmo utilizado para pontos de sincronização principal), que pode ser passado e solicitado de maneira independente de dados e de *tokens* de sincronização secundários.

A ISO concluiu que, se um usuário iniciar uma atividade enquanto o outro estiver fazendo uma sincronização secundária, podem ocorrer problemas. Para solucionar isso,

[6] Disponível em <http://penta2.ufrgs.br/rc952/trab2/sessao2.html>. Acesso em: 27 abr. 2012.

antes que uma atividade ou operação de sincronização seja iniciada, o usuário deve reter os *tokens* de atividade, de sincronização secundária e de dados.

Outra questão importante é a relação entre as atividades e os pontos de sincronização. Cada vez que é iniciada uma nova atividade, os números de séries dos pontos de sincronização são reinicializados e é criado um ponto de sincronização principal. Podem ser criados pontos de sincronizações adicionais, secundários ou não, dentro de atividades. Uma vez que uma atividade é iniciada, se ocorrer uma ressincronização, não é possível retornar para uma atividade anterior.

Relatório de exceções

Esse serviço é utilizado para que sejam relatados erros inesperados. Se o usuário tiver algum problema, ele pode relatá-lo ao seu parceiro, explicando o que aconteceu.

O relatório de exceções não se aplica apenas a erros detectados pelo usuário, mas também para problemas internos na Camada de Sessão ou problemas relatados pelas camadas inferiores. Porém, a decisão da ação que deve ser tomada é sempre feita pelo usuário.[7]

Resumo

Essa unidade teve foco nas camadas mais altas do Modelo OSI.

A **Camada de Aplicação** é responsável pela interface entre o protocolo de comunicação e o aplicativo que está sendo executado pelo usuário final. Devido às diversas necessidades de aplicações existentes, são inúmeros os protocolos que residem nesta camada. Embora os protocolos sejam muito importantes neste nível, é importante lembrar que eles são apenas uma parte da aplicação de rede.

Dizemos que a comunicação entre duas aplicações que estão distribuídas são possíveis graças aos **processos** que rodam na aplicação de origem e na aplicação de destino, através de suas **portas**, ou APIs.

Diversas aplicações são do tipo **cliente-servidor**. Intitulamos **cliente** a aplicação que "fala primeiro" e **servidor** aquele que provê serviço ao cliente que faz a solicitação. Algumas aplicações podem, inclusive, implementar ao mesmo tempo o lado cliente e o lado servidor, como é o caso do correio eletrônico.

[7] Disponível em: <http://penta2.ufrgs.br/rc952/trab2/sessao2.html>. Acesso em: 27 abr. 2012.

Alguns parâmetros são importantes para analisar qual protocolo de transporte será utilizado durante e implementação da aplicação. São eles: **perda de dados**, **largura de banda** e **sensibilidade temporal**.

Conhecemos algumas das aplicações típicas de rede e seus respectivos protocolos: a World Wide Web, com o protocolo http; transferência de arquivos, com o FTP; correio eletrônico, com o SMTP, POP, IMAP e http, e o serviço de diretório de nomes, com o DNS.

A **www** tem sua grande importância por ter revolucionado a forma de pensar das pessoas quando entrega conteúdo sob demanda. O protocolo responsável pelas aplicações *web* é o **HTTP** e o agente de usuário aqui é o **navegador**. O HTTP utiliza conexões do tipo cliente-servidor através do protocolo TCP de transporte na porta 80. Vimos que a versão mais atual o protocolo HTTP é mais robusta que sua versão inicial, que utilizava uma conexão TCP a cada vez que precisava carregar um objeto nas páginas *web*.

A aplicação de **transferência de arquivos** tem sua importância por ser uma necessidade antiga e que nos acompanha até hoje. O protocolo que rege esta aplicação é o **FTP**, que também usa o TCP no estabelecimento das conexões com o servidor. A aplicação utiliza duas portas distintas para a transmissão das informações: as portas 20 e 21, pois assim reserva uma exclusiva para transmitir os dados enquanto a outra lida com as operações de controle. Aqui consideramos o lado cliente o que inicia a transferência e o lado servidor o que hospeda os arquivos, remotamente.

A aplicação de **correio eletrônico** é a mais antiga das aplicações das redes de computadores. Ela utiliza protocolos diferentes, dependendo da ação que é executada. Os protocolos mais conhecidos são o **SMTP**, **POP3**, **IMAP** e o próprio **HTTP**, quando se trata de correio *web*. Os elementos importantes que não podem ser esquecidos são: **fila de mensagens de saída**, **servidores de correio**, **caixa de correio do usuário** e os **agentes de usuário** que, neste caso, são os leitores de correio. Esta aplicação, com o protocolo SMTP, utiliza a porta de número 25.

O **serviço de diretório de nomes** serve para que os roteadores possam identificar os nomes que são digitados pelos usuários quando acessam páginas da internet. O protocolo utilizado é o **DNS**, que é o responsável por converter nomes por endereços IP. Vimos que o DNS é formado por uma grande estrutura **hierárquica e distribuída** para ser eficiente no processo de tradução. Para isso, alguns tipos de servidores são considerados: **servidores de nomes local**, **servidores de nome oficial**, **servidores de domínio de alto nível** e **servidores de nomes com autoridade**.

Dentre os serviços que o DNS provê estão incluídos: **apelidos de hospedeiros**, **apelidos de servidor de correio eletrônico** e **distribuição de carga**. O DNS se apoia tanto no TCP como no UDP, como protocolos de Camada de Transporte, e utiliza a porta 53.

A **Camada de Apresentação** é responsável pela **tradução** e conversão do formato recebido pela Camada de Aplicação. Dentre os exemplos de aplicações que fazem uso da Camada de Apresentação estão **criptografia**, **compressão** dos dados e **formato** do padrão de caracteres.

A **Camada de Sessão** é responsável especialmente por oferecer às camadas de apresentação, ou diretamente à aplicação, formas de **organizar** e **sincronizar** a comunicação, estabelecendo sessões. Para isso ela trabalha com **marcações** e sincronismos que permite garantir o retorno da aplicação do ponto onde parou, caso alguma interrupção na rede tenha acontecido. Dentre os principais serviços que ela executa, destacam-se**: intercâmbio de dados**, **gerenciamento de diálogos**, **sincronização**, **gerenciamento de atividades** e **relatório de exceções**.

Exercícios

Questão 1. O que é o *jitter* na comunicação?

A) Variação do atraso.
B) Teste de ruído.
C) Nível de serviço.
D) Serviço diferenciado.
E) Melhor esforço.

Resposta correta: alternativa A.

Análise das alternativas:

A) Alternativa correta.

Justificativa:

Jitter é o nome dado à variação do atraso na comunicação. Atraso é considerado em geral um problema para muitas aplicações. Entretanto, a variação do atraso, ou *jitter*, em geral é pior, pois reduz a qualidade do serviço (QoS) na transmissão.

B) Alternativa incorreta.

Justificativa:

Teste de ruído é a verificação do perfil de inconsistência do sinal.

C) Alternativa incorreta.

Justificativa:

Nível de serviço (*service level*) é o que define o nível de exigência para a capacidade de uma rede.

D) Alternativa incorreta.

Justificativa:

Serviço diferenciado (diferencia *ted service* ou *soft QoS*) que permite definir níveis de prioridade num caminho predeterminado na nuvem, sem contudo fornecer uma garantia estrita, já que não há reserva de recursos, e sim priorização.

E) Alternativa incorreta.

Justificativa:

Melhor esforço (*best effort*, em inglês) não fornece nenhuma diferenciação entre as várias redes e, por consequência, não permite nenhuma garantia, já que não se sabe qual será o seu caminho na nuvem. Este nível de serviço é também chamado *lack of QoS*.

Questão 2 (Forouzan, Behrouz A. *Comunicação de dados e redes de computadores*). Qual é protocolo projetado para fornecer serviços de segurança e de compressão de dados gerados na Camada de Aplicação?

A) SSL.
B) AS.
C) SADB.
D) SPI.
E) SAR.

Resolução desta questão na Plataforma.