Redes de Computadores e a Internet

Evandro Cantú
Instituto Federal de Santa Catarina
Campus São José
cantu@ifsc.edu.br

Julho de 2009.

Apresentação

O objetivo desde texto é servir de material de apoio para um curso introdutório de redes de computadores, voltado para a disciplina de Redes de Computadores I do CST em Sistemas de Telecomunicações e para a disciplina de Redes de Computadores do Curso Técnico de Telecomunicações do Instituto Federal de Santa Catarina, Campus São José.

As redes de computadores são estudadas tomando como referência a arquitetura Internet, estudada em torno de seus principais protocolos, o TCP/IP.

A principal referência bibliográfica utilizada na construção deste material foi o livro de KUROSE e ROSS, **Redes de Computadores e a Internet: Uma abordagem *top-down***, no qual os autores abordam os conceitos de redes de computadores, iniciando pelas aplicações e depois descendo pelas demais camadas que formam a arquitetura Internet. As **RFCs** (*Request for comments*), que constituem os padrões para a Internet, também foram consultadas e sempre que possível foram incluídos ilustrações e desenhos constantes nestes documentos.

O texto está organizado em seis capítulos. O primeiro faz uma introdução às redes de computadores e a arquitetura Internet, dando uma visão ampla das redes e dos principais conceitos envolvidos. O segundo capítulo apresenta os protocolos e as aplicações de rede, apresentando em particular a aplicação WWW, o correio eletrônico e a transferência de arquivos e a aplicação DNS. O terceiro capítulo apresenta os protocolos de transporte da Internet, UDP e TCP. O quarto capítulo aparesenta os protocolos de rede da Internet, cujo principal componente é o protocolo IP. Finalmente, o quinto capítulo apresenta as redes locais e os protocolos de enlace, com destaque para a tecnologia Ethernet.

Evandro Cantú, Julho de 2009.

Sumário

Capítulo 1

Introdução as redes de computadores e a Internet

1.1 A Rede Internet

A rede Internet é hoje a principal redes de computadores no mundo e será utilizada como exemplo principal no nosso estudo sobre redes de computadores.

A Internet é na verdade uma "rede de redes", interligando milhões de dispositivos computacionais espalhados ao redor do mundo, cuja topologia básica está ilustrada na figura 1.1.

A maioria dos dispositivos que compõem a Internet é formada por computadores pessoais, estações de trabalho, ou servidores, que armazenam e transmitem informações, como por exemplo, páginas Web, arquivos de texto ou mensagens eletrônicas. Todos estes dispositivos são chamados **hospedeiros** (***hosts***) ou **sistemas terminais**.

As **aplicações de rede**, como por exemplo, paginação na Web, transferência de arquivos ou correio eletrônico, rodam nos sistemas terminais.

Os sistemas terminais, assim como os principais componentes da Internet, precisam de **protocolos de comunicação**, que servem para controlar o envio e a recepção das informações na Internet. O **TCP** (*Transmission Control Protocol*) e o **IP** (*Internet Protocol*) são os principais protocolos da Internet, daí o fato de a Internet ser também conhecida como rede **TCP/IP**.

Os sistemas terminais são conectados entre si por meio de **enlaces de comunicação**, que por sua vez podem ser de diferentes tipos, como por exemplo, um enlace ponto-a-ponto (tipo o PPP) ou multiponto (como uma rede local Ethernet). Os enlaces de comunicação, por sua vez, são suportados por um meio físico, os quais podem ser cabos coaxiais, fios de cobre, fibras ópticas ou o ar a partir do uso do espectro de freqüência de rádio.

Na Internet, nem todos os computadores são diretamente conectados, neste caso, utilizam-se dispositivos de chaveamento intermediário, chamados **roteadores** (*routers* ou ainda *gateways*).

Em cada roteador da Internet as mensagens que chegam nos enlaces de entrada são **armazenadas e encaminhadas** (*store-and-forward*) aos enlaces de saída, seguindo de

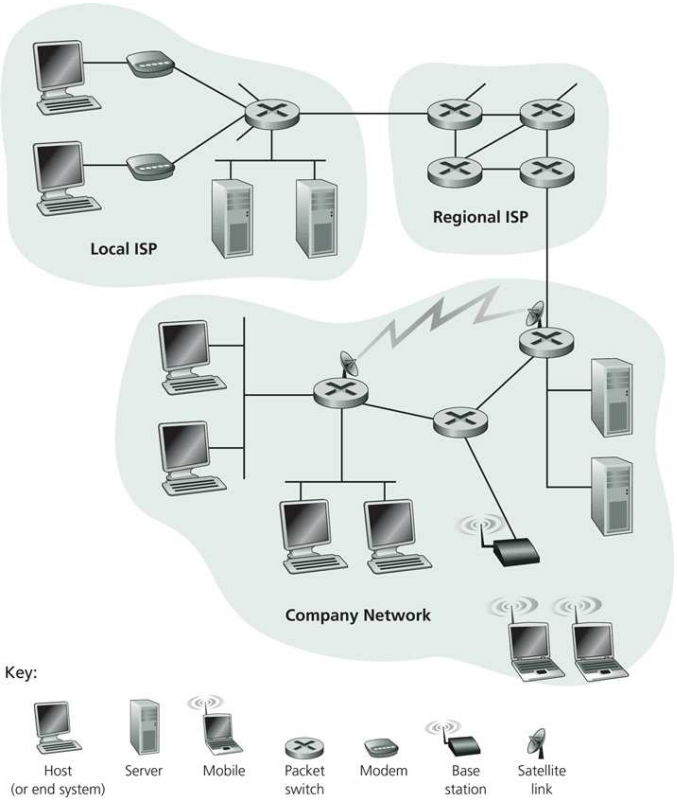


Figura 1.1: Rede Internet (KUROSE; ROSS, 2006a).

roteador em roteador até seu destino. Neste processo, a técnica de comutação utilizada é conhecida como **comutação de pacotes**, em contraste com a **comutação de circuitos** que é comumente utilizada nos sistemas telefônicos.

Na comutação de pacotes, as **mensagens** que serão transmitidas são fragmentadas em **pacotes** menores, os quais viajaram na Internet de forma independente uns dos outros.

O **protocolo IP** é o responsável por estabelecer a rota pela qual seguirá cada pacote na malha de roteadores da Internet. Esta rota é construída tendo como base o endereço de destino de cada pacote, conhecido como **endereço IP**.

Além de um endereço IP, um nome também pode ser associado a um sistema terminal a fim de facilitar sua identificação por nós humanos. Por exemplo, 200.18.10.1 é o endereço IP e www.ifsc.edu.br é o nome do servidor Web do Instituto Federal de Santa Catarina. A aplicação **DNS** (*domain name system*) associa dinamicamente nomes a endereços IP.

Em outras palavras, pode-se dizer que a Internet é uma rede de redes, interconectando redes de computadores públicas e privadas, as quais devem rodar o protocolo IP em conformidade com a convenção de endereços IP e nomes da Internet.

A topologia da Internet é hierárquica, onde os sistemas terminais são conectados a **provedores locais** (ou ISP – *Internet Service Provider*), que por sua vez são conectados a **provedores regionais**, e estes últimos a **provedores nacionais** ou **internacionais**.

O provedor local do IF-SC em São José está conectado ao provedor nacional da RNP (Rede Nacional de Ensino e Pesquisa – `www.rnp.br`) (veja mapa da rede da RNP no endereço `www.rnp.br/backbone`).

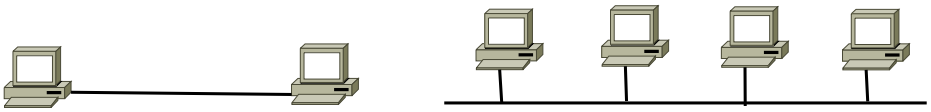
A conexão de um computador a um provedor local é feita por meio de uma **rede de acesso**, a qual pode ser um **acesso residencial** (por exemplo, via modem e linha discada) ou **acesso corporativo** via rede local.

Todos os conceitos a pouco citados serão nossos objetos de estudo ao longo deste curso de redes de computadores. Iniciaremos a partir de alguns conceitos gerais sobre redes e depois avançaremos em direção aos conceitos específicos que compõe a arquitetura Internet.

1.2 Formas de conectividade em rede

Uma rede de computadores é conexão de dois ou mais computadores para permitir o compartilhamento de recursos e a troca de informações entre as máquinas.

A conectividade dos computadores em rede pode ocorrer em diferentes escalas. A rede mais simples consiste em dois ou mais computadores conectados por um meio físico, tal como um par metálico ou um cabo coaxial. O meio físico que conecta dois computadores costuma ser chamado de **enlace de comunicação** e os computadores são chamados de **hospedeiros** (ou ***hosts***). Um enlace de comunicação limitado a um par de nós é chamado de **enlace ponto-a-ponto**. Um enlace pode também envolver mais de dois nós, neste caso, podemos chamá-lo de **enlace multiponto**. Um enlace multiponto, formando um barramento de múltiplo acesso, é um exemplo de enlace utilizado nas **redes locais**, como uma rede local **Ethernet**. A figura 1.2 ilustra estas formas de conectividade em rede.



(a) Ponto a ponto. (b) Multiponto.

Figura 1.2: Conectividade em rede.

Se as redes de computadores fossem limitadas a situações onde todos os nós fossem diretamente conectados a um meio físico comum, o número de computadores que poderiam ser interligados seria também muito limitado. Na verdade, numa rede geograficamente distribuída, como a **Internet**, nem todos os computadores precisam estar diretamente conectados. Uma conectividade indireta pode ser obtida usando uma **rede comutada**. Nesta rede comutada podemos diferenciar os nós da rede que estão na sua periferia, como computadores terminais conectados ao núcleo da rede via enlaces ponto-a-ponto ou multiponto, daqueles que estão no núcleo da rede, formado por comutadores ou roteadores, como mostra a figura 1.3. Note que esta última forma de conectividade é equivalente a mostrada na figura 1.1 que mostrou a topologia básica da Internet.

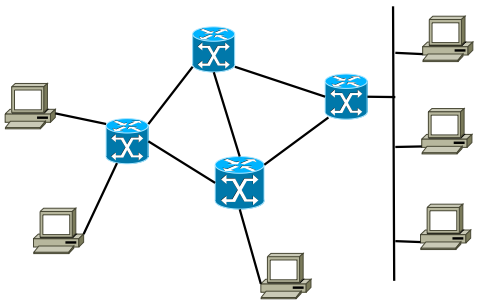


Figura 1.3: Rede chaveada.

1.3 Classificação das redes quanto a abrangência geográfica

Uma forma de classificar as redes de computadores é quanto a sua abrangência geográfica. Dois exemplos bem conhecidos são as **rede locais** (ou **LANs** – *local area networks*) e as **redes geograficamente distribuídas** (ou **WANs** – *wide area networks*). As LANs geralmente estão limitadas ao espaço de uma instituição. As WANs, por sua vez, têm alcance global, como a Internet.

Além da questão do tamanho, este tipo de classificação também tem relação com a tecnologia utilizada para interconexão em rede. Por exemplo, uma rede local geralmente conecta diretamente seus hospedeiros por um enlace multiponto, como descrito na sessão anterior. As WANs, por outro lado, geralmente são formadas pela interconexão de várias redes por meio de elementos de chaveamento, como roteadores.

Outras redes são ainda classificadas como **redes metropolitanas** (**MANs** – *metropolitan area networks*), as quais tem abrangência de uma região metropolitana. Quanto as tecnologias empregadas nas redes MANs, podem ser tanto a interconexão de redes por

meio de elementos de chaveamento e utilizando um *backbone* comum, como também as novas tecnologias redes sem fio, como as redes *WiMax*.

Outra denominação utilizada são as **redes pessoais** (**PAMs** – *personal area network*), as quais tem a abrangência de uma sala, como as redes *Bluetooth* (IEEE802.15) ou as redes de sensores sem fio (IEEE802.15.4), as quais permitem interconectar diversos disposistivos por meio de enlaces sem fio.

1.4 Comutação de pacotes X comutação de circuitos

Nas redes de computadores se utiliza a **comutação de pacotes** como tecnologia de comunicação no núcleo da rede, em contraste com as redes telefônicas tradicionais que usam a **comutação de circuitos**.

Na **comutação de circuitos**, quando dois sistemas terminais desejam se comunicar a rede estabelece um circuito dedicado fim-a-fim entre os dois sistemas. É por exemplo o que acontece numa ligação telefônica. A partir do número discado, a rede estabelece um caminho entre os dois interlocutores e reserva um circuito para possibilitar a conversação. O circuito ficará reservado durante todo o tempo em que durar a comunicação.

A comutação de circuitos se mostrou ineficiente para ser aplicada nas redes de computadores devido as características do tráfego nestas redes. Nas primeiras redes de computadores as aplicações típicas eram o acesso remoto e a transferência de arquivos, nas quais o tráfego consistia de um fluxo esporádico de dados, com curtos intervalos de atividade espaçados no tempo, diferente do fluxo contínuo do tráfego telefônico.

Na **comutação de pacotes**, os recursos da rede não são reservados. As mensagens usam os recursos da rede na medida da necessidade, compartilhando os recursos na forma de uma "multiplexação estatística", como mostra a figura 1.4. Considere, por exemplo, o que acontece quando um computador deseja enviar um pacote de dados a outro computador da rede. Como na comutação de circuitos, o pacote será transmitido sobre uma série de diferentes enlaces de comunicação, todavia, não haverá uma reserva de um circuito fim-a-fim. O pacote será encaminhado de roteador em roteador, e caso o enlace de saída de um roteador de sua rota esteja ocupado, o pacote deverá ser armazenado e aguardar a liberação do enlace em uma fila, sofrendo um atraso.

Os defensores da comutação de pacotes sempre argumentam que a comutação de circuitos é ineficiente, pois reserva o circuito mesmo durante os períodos de silêncio na comunicação. Por exemplo, durante uma conversa telefônica, os silêncios da conversação, ou as esperas para chamar uma outra pessoa, não podem ser utilizados para outras conexões. Além disto, os tempos necessários para o estabelecimento de circuitos fim-a-fim são grandes, além de ser uma tarefa complicada e requerer esquemas complexos de sinalização ao longo de todo o caminho da comunicação.

Por outro lado, os opositores da comutação de pacotes argumentam que a mesma não seria apropriada para aplicações tempo real, como por exemplo conversas telefônicas, devido os atrasos variáveis em filas de espera, difíceis de serem previstos. Todavia, com o desenvolvimento de técnicas específicas e o aumento da velocidade dos enlaces, observa-se uma tendência em direção à migração dos serviços telefônicos também para a tecnologia de comutação de pacotes.

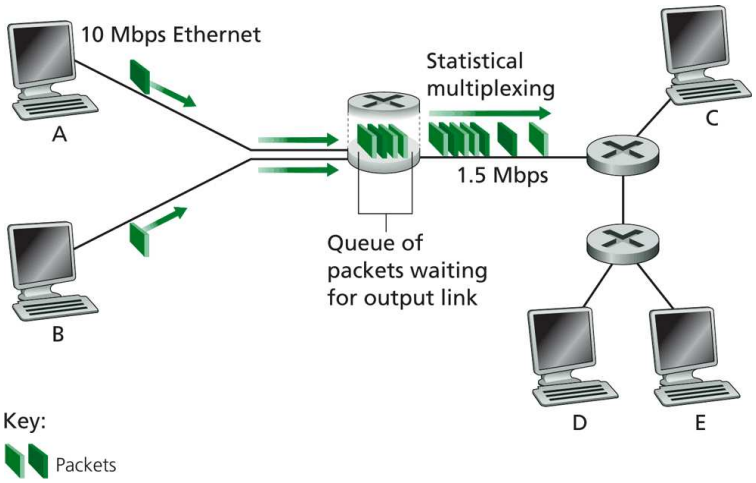


Figura 1.4: Comutacao de Pacotes (KUROSE; ROSS, 2006a).

Comutação de pacotes X comutação de mensagens

Na **comutação de pacotes** moderna, o computador que vai transmitir uma mensagem longa, fragmenta esta mensagem em pacotes menores antes de enviá-la pela rede. O receptor, por sua vez, reagrupa os pacotes recebidos para formar a mensagem original. Este processo é realizado visando melhorar a performance da rede.

Uma comutação de pacotes na qual o emissor não fragmenta a mensagem é chamada de **comutação de mensagens**.

Considere o exemplo apresentado por Kurose e Ross (2006b), ilustrado na figura 1.5, na qual uma mensagem de 7,5 Mbits é enviada entre um computador origem e um destino, passando por dois roteadores e três enlaces. Considere ainda que cada enlace tem taxa de transmissão de 1,5 Mbps. Assumindo que não há congestionamento na rede, a mensagem leva 5 segundos para ser transmitida da origem até o primeiro roteador. O atraso de transmissão é dado pelo quociente entre o tamanho do pacote (bits) pela transmissão do enlace (bits/seg). Como em cada roteador a mensagem deve ser armazenada e transmitida, a mesma só pode ser reencaminhada após ser recebida completamente pelo roteador. Desta forma, o tempo total para a mensagem percorrer os três enlaces e ser transmitida da origem até o destino é de 15 segundos.

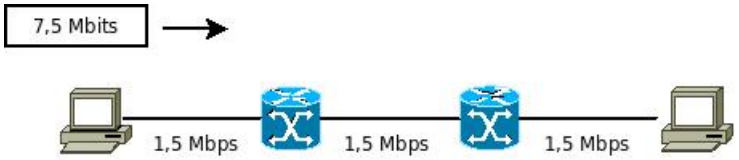


Figura 1.5: Comutação de pacotes x comutação de mensagens.

Suponha agora que a mesma mensagem foi fragmentada em 5000 pacotes, cada um com 1,5 Kbits. Neste caso, cada pacote leva 1 mseg para ser transmitido da origem até o primeiro roteador. Todavia, enquanto este pacote é movido do primeiro roteador para

o segundo, outro pacote pode já ser enviado da origem até o primeiro roteador. Assim o segundo pacote chega ao primeiro roteador no tempo 2 mseg. Seguindo neste lógica, o último pacote chega ao primeiro roteador no tempo 5000 mseg = 5 segundos. Como este último pacote deve ainda percorrer os dois últimos enlaces, ele chegará ao destino no tempo 5,002 segundos.

No segundo caso o tempo total para transmitir os 7,5 Mbits foi reduzido praticamente a 1/3 do tempo anterior! Porque isto ocorre? Observe que na comutação de mensagem, enquanto um nó está transmitindo, os demais nós permanecem ociosos. No caso da comutação de pacotes, uma vez que o primeiro pacote chega ao último roteador, teremos três nós transmitindo ao mesmo tempo, otimizando o uso dos enlaces.

Outra vantagem da comutação de pacotes será na recuperação de possíveis perdas ou erros de bits nos pacotes. Quando uma perda de pacotes ou erro ocorre, muitas vezes o pacote precisa ser retransmitido. No caso de uma mensagem segmentada em vários pacotes, apenas os pacotes com erro precisam ser retransmitidos e não a mensagem inteira.

A desvantagem da comutação de pacotes será as informações adicionais que precisam ser adicionadas ao **cabeçalho** (*header*) de cada pacote ou mensagem.

1.5 Tipos de roteamento em redes de comutação de pacotes

Há duas classes de redes de comutação de pacotes, as redes baseadas em **datagramas**, como a Internet, e as redes baseadas em **circuito virtual**. A diferença básica destas duas redes está na forma como os pacotes são roteados em direção ao destino.

Roteamento em redes baseadas em circuito virtual

Nas redes baseadas em **circuito virtual**, a rota para os pacotes é estabelecida a priori, numa fase de estabelecimento do circuito virtual. Uma vez estabelecido o circuito virtual, todos os pacotes seguem pela mesma rota, cada um deles carregando a informação de qual circuito virtual o mesmo deve tomar em cada roteador. Os exemplos de redes que utilizam esta técnica incluem as redes X.25, as redes *frame-relay* e as redes ATM (*asynchronous transfer mode*).

O processo de estabelecimento de um circuito virtual é similar ao estabelecimento de conexão nas redes de comutação de circuitos, entretanto, os enlaces individuais não ficam reservados de forma exclusiva para uma única conexão, podendo, durante uma transmissão, serem compartilhados por outras transmissões.

Roteamento em redes baseadas em datagrama

Nas redes baseadas em **datagramas**, não há estabelecimento de conexão ou circuito virtual. Os pacotes são encaminhados em função do endereço do destino.

Em muitos aspectos as redes baseadas em datagramas são análogas aos serviços postais. Quando alguém vai enviar uma carta a um destinatário, o mesmo coloca a carta em um envelope e escreve o endereço do destino sobre o envelope. O endereço tem uma estrutura hierárquica, incluindo, no caso do Brasil, o país, o estado, a cidade, a rua e o número da casa. Por exemplo, se alguém enviar uma carta da França para sua casa, o correio da França primeiro vai direcionar a carta para o centro postal do Brasil (por exemplo, situado em São Paulo). O centro postal do Brasil vai então direcionar a carta para Santa Catarina, estado destino da carta (na agência central de Florianópolis, por exemplo). A agência de Florianópolis vai então direcioná-la a agência de sua cidade, que por sua vez vai repassar ao carteiro para entregar a carta emsua casa.

Na rede baseada em datagrama, cada pacote atravessa a rede contendo no cabeçalho o endereço do nó destino, que como o endereço postal, tem uma estrutura hierárquica. Quando o pacote chega a um roteador, o mesmo examina uma parte do endereço e o encaminha ao roteador adjacente. No próximo roteador o processo se repete.

Na Internet a comutação de datagramas é implementada pelo protocolo IP, o qual é o responsável por estabelecer a rota pela qual seguirá cada pacote na malha de roteadores. Esta rota é construída tendo como base o endereço de destino de cada pacote, conhecido como endereço IP.

O serviço de comunicação por datagramas oferecido pelo protocolo IP é conhecido como serviço "melhor esforço" (*best effort*), uma vez que não há garantia que o dado chegue ao destino. Este serviço envolve cada roteador no caminho entre o computador origem e o destino da comunicação.

A figura 1.6 ilustra, na forma de um mapa conceitual, uma taxonomia das redes de telecomunicações, mostrando as diferentes formas de comutação e multiplexação utilizadas.

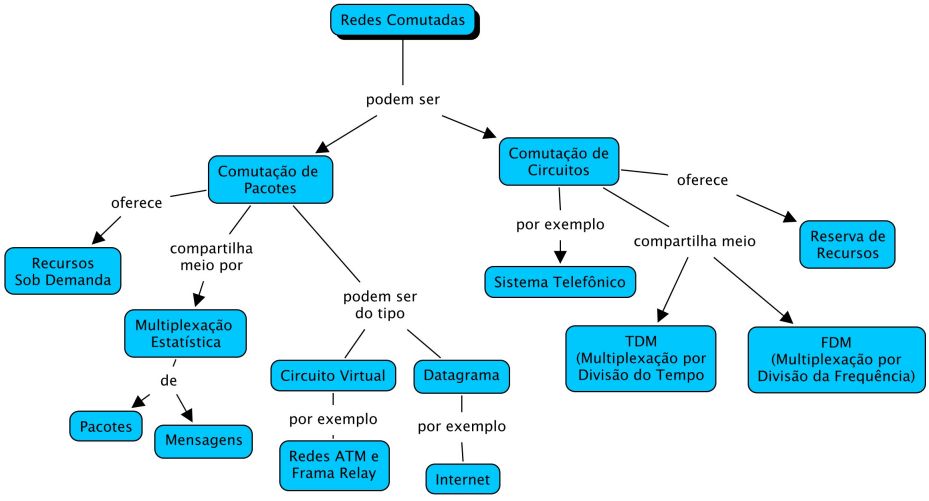


Figura 1.6: Taxonomia das redes de telecomunicações.

1.6 Vazão, atraso e perda de pacotes

Segundo Peterson e Davie (2004) a performance das redes de pacotes pode mensurada de dois modos principais:

- **Vazão** (*throughput*), também chamada de largura de banda (*bandwith*);
- **Atraso** (delay).

A **vazão** é definida como o número de bits que podem ser transmitidos sobre a rede num dado tempo, sendo expressa em bits/segundo (bps). Por exemplo, numa rede Ethernet podemos ter como vazão 10Mbps. Muitas vezes usa-se o termo taxa de transmissão para se referir a vazão.

O **atraso** é o tempo que um pacote leva para atravessar uma rede desde a origem até o destino, passando pelos roteadores e enlaces intermediários. O atraso é medido em termos de tempo. Por exemplo, uma rede transcontinental pode ter um atraso de 20 milisegundos, isto é, um pacote leva 20 ms para atravessar a rede.

Ao percorrer uma rede, um pacote sofre uma série de atrasos em cada um dos nós do caminho. Este atraso em cada nó da rede tem quatro componentes principais: o **atraso de processamento**, o **atraso de fila**, o **atraso de transmissão** e o **atraso de propagação** (KUROSE; ROSS, 2006b).

- Atraso de processamento

  É o tempo necessário para examinar o cabeçalho do pacote, checar a integridade dos dados recebidos e determinar para onde o mesmo deverá ser encaminhado. Estes atrasos geralmente estão na ordem de micro-segundos.

- Atraso de fila

  É o tempo que cada pacote espera em fila nos roteadores antes de ser encaminhado ao próximo enlace. Este atraso é variável e depende do tráfego na rede. Se a fila estiver vazia e não houver outro pacote sendo transmitido o atraso é zero. Se há pacotes na fila o tempo de espera pode ser longo e pode chegar a ordem de milisegundos.

- Atraso de transmissão

  É o tempo necessário para que cada bit do pacote seja empurrado para dentro do enlace. Se o tamanho do pacote for L bits e a taxa de transmissão do enlace for R bits/seg. Então o atraso de transmissão será L/R segundos.

- Atraso de propagação

  Uma vez que o pocote foi empurrado para dentro do enlace, o atraso de propagação é o tempo de viagem até o próximo nó. Este tempo é função da distância entre os nós e da velocidade de propagação do sinal no meio físico. A velocidade de propagação depende do meio físico (par trançado, fibra óptica, espectro eletromagnético, etc) e é um pouco menor ou igual a velocidade da luz. Para distância pequenas, como num enlace entre dois computadores de um campus, este atraso pode ser desprezível,

ficando na ordem de poucos micro-segundos. Para grandes distâncias, como num enlace de satélite, pode levar um tempo importante, da órdem de centenas de mili-segundos.

O atraso total fim a fim será a soma dos atrasos em cada um dos nós do caminho.

$d_{nó} = d_{proc} + d_{fila} + d_{trans} + d_{prop}$

Perda de pacotes

A perda de pacotes ocorre quando a capacidade de armazenamento da fila de um roteador se esgota. Neste caso os novos pacotes que chegam são descartados (*dropped*) e são considerados perdidos. A fração dos pacotes perdidos aumenta a medida que a intensidade de tráfego aumenta.

Pacotes corrompidos totalmente também podem ser considerados como perdidos.

1.7 Suporte a serviços comuns para as aplicações

O objetivo das redes de pacotes é oferecer um suporte de comunicação para que as aplicações rodando em dois computadores remotos possam trocar informações.

Intuitivamente, podemos ver a rede como provendo um **canal lógico** de comunicação para que os processos de aplicação cliente e servidor possam se comunicar. Para usar este canal de comunicação, os programas de aplicação cliente enviam seus pedidos através de uma "porta", que conecta o cliente ao servidor, e através da qual ele espera a resposta do serviço é requisitado.

Dois tipos comuns de serviço solicitado pelas aplicações à rede são:

- Serviço tipo pedido/resposta (*request/reply*);
- Serviço tipo fluxo de dados tempo real (*audio/video streaming*).

A paginação na Web é um exemplo de serviço tipo pedido/resposta, onde um processo cliente solicita uma informação e um processo servidor fornece a informação solicitada. Não há restrições de tempo entre o pedido e a resposta, entretanto, é necessário que a informação transmitida seja livre de erros.

Uma conversa telefônica via Internet é um exemplo de fluxo de dados em tempo real, neste caso há restrições temporais na transmissão, por outro lado, um pequeno silêncio ocasionado por um erro ou ruído pode não ser um problema grave para o entendimento geral da conversa.

Para estes dois tipos de requisições de serviços, muitas redes de computadores dispõem de dois tipos de serviços de transporte:

- Serviço orientado a conexão, com transmissão de dados garantida;

- Serviço não orientado a conexão, com transmissão de dados tipo "melhor esforço" (*best effort*).

Quando uma aplicação usa o serviço orientado a conexão o cliente e o servidor trocam pacotes de controle entre si antes de enviarem os pacotes de dados. Isto é chamado de procedimento de estabelecimento de conexão (*handshaking*), onde se estabelecem os parâmetros para a comunicação. Uma vez concluído o *handshaking* a conexão é dita estabelecida e os dois sistemas terminais podem trocar dados. O serviço de transferência garantida, que assegura que os dados trocados são livres de erro, o que é conseguido a partir de temporizações, mensagens de reconhecimento e retransmissão de pacotes. Por exemplo, quando um sistema terminal B recebe um pacote de A, ele envia um reconhecimento; quando o sistema terminal A recebe o reconhecimento ele sabe que o pacote que ele enviou foi corretamente recebido; caso A não recebe confirmação, ele assume que o pacote não foi recebido por B e retransmite o pacote.

No serviço não orientado a conexão não há *handshaking*. Quando um lado de uma aplicação quer enviar pacotes ao outro lado ele simplesmente envia os pacotes. Como o serviço é não garantido, também não há reconhecimento, de forma que a fonte nunca tem certeza que o pacote foi recebido pelo destinatário. Como o serviço é mais simples, os dados podem ser enviados mais rapidamente.

Na Internet, o serviço orientado a conexão é implementado pelo protocolo TCP (*Transmission Control Protocol*) e o serviço não orientado a conexão é implementado pelo protocolo UDP (*User Datagram Protocol*). As aplicações conhecidas como o telnet, correio eletrônico, transferência de arquivos e WWW usam o TCP. Outras aplicações usam o UDP, como o DNS (*domain name system*) e aplicações multimídia como voz sobre Internet e aplicações de áudio e vídeo.

1.8 Padronização na área de redes de computadores

Durante os primeiros tempos das redes de computadores os diversos fabricantes trabalharam de forma separada no desenvolvimento de suas tecnologias, muitas delas incompatíveis entre si. Com o intuito de estabelecer uma padronização e permitir uma integração entre as diversas tecnologias, a ISO (*International Standard Organization* – `www.iso.org`), juntamente com o ITU (*International Telecommunication Union* – `www.itu.int`), organismos responsáveis pelo estabelecimento de normas e padrões em telecomunicações, definiram um **modelo de referência** com sete camadas de protocolos. Este modelo ficou conhecido como **modelo OSI** (*open system interconnection*), no qual cada camada deveria criar um nível de abstração diferente, devendo realizar uma função bem definida a camada superior, omitindo detalhes quanto a sua implementação.

As **sete camadas do modelo OSI**, apresentadas na figura 1.7, tiveram muito sucesso na literatura de redes de computadores, sendo uma referência para a sistematização e descrição protocolos de rede. Todavia, o modelo OSI não teve sucesso comercial, uma vez que poucos produtos seguiram a risca as recomendações do modelo.

Outro importante organismo de padronização na área de redes de computadores é o IETF (*Internet Engineering Task Force* – `www.ietf.org`), o qual coordena a pradonização

Figura 1.7: As sete camadas do modelo OSI.

dos protocolos para a Internet. Cada padrão é publicado através de documentos conhecidos como RFCs (*Request For Comments* – `www.ietf.org/rfc.html`), os quais contém a descrição de cada protocolo padrão utilizado na Internet.

Na área de redes locais de computadores os esforços de padronização são coordenados pelo IEEE (*Institute of Electrical and Electronics Engineers* – `www.ieee.org`), através dos padrões IEEE802. Dentre as tecnologias conhecidas para redes locais em uso atualmente, destaca-se a redes Ethernet, padronizadas como IEEE802-3, as redes locais sem fio IEEE802-11, as redes metropolitanas sem fio IEEE802-16, entre outras.

A figura 1.8 ilustra, na forma de um mapa conceitual, os principais organismos de padronização na área de redes de computadores.

1.9 Arquitetura de redes

As camadas de protocolos facilitam o projeto e a implementação das redes de computadores, e no nosso caso, também o estudo das redes. Através das camadas de protocolos, o problema de construir uma rede fica decomposto em diversas partes, onde cada camada pode ser implementada separadamente, sem afetar as demais.

A idéia geral da divisão em camadas de protocolos é começar com os serviços oferecidos pelo hardware, e ir adicionando uma sequência de camadas, cada uma delas provendo um serviço com maior grau de abstração a camada superior.

A figura 1.9 mostra um exemplo de arquitetura de rede com quatro camadas de protocolos (PETERSON; DAVIE, 2004). Nesta arquitetura poderíamos agrupar na camada inferior os protocolos relacionados relacionados à comunicação entre nós vizinhos e à transmissão de bits sobre os enlaces físicos. Na camada imediatamente superior poderíamos agrupar os protocolos para tratar os problemas relativos ao encaminhamento de pacotes entre dois computadores remotos, passando pelos roteadores intermediários, permitindo a conectividade computador a computador, abstraindo o fato de que existe uma topologia complexa entre quaisquer dois computadores. A próxima camada poderia implementar um canal de comunicação lógico fim a fim entre os processos de aplicação, oferecendo um serviço apropriado para que os processos de aplicação troquem mensagens. Por fim,

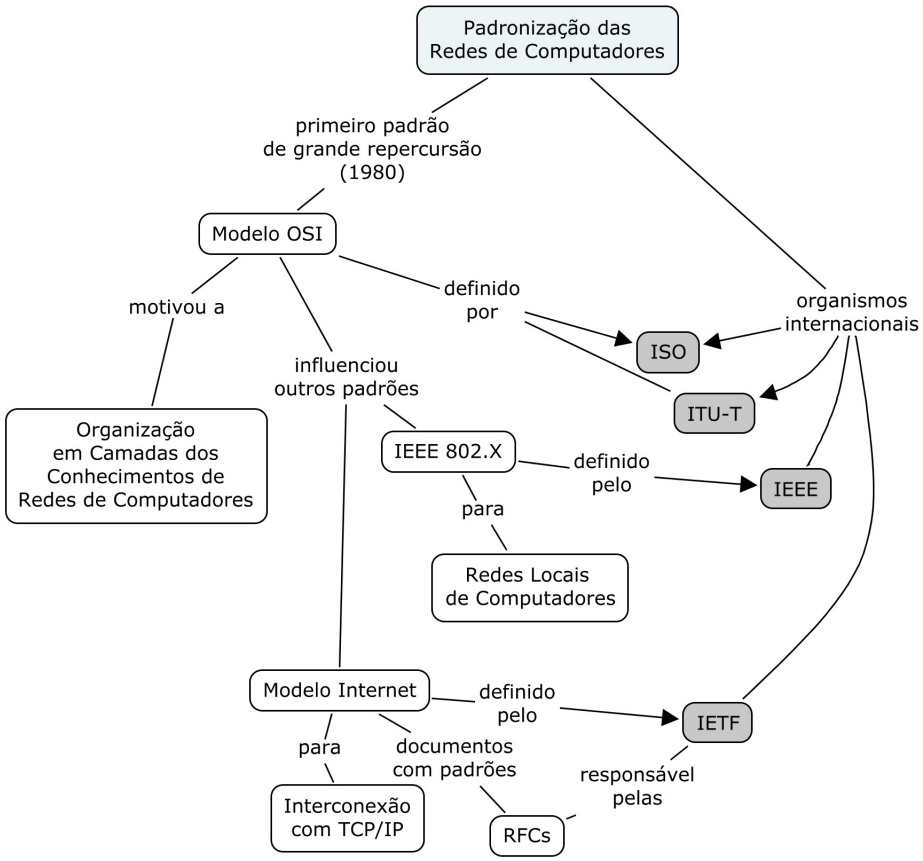


Figura 1.8: Padronização na área de redes de telecomunicações.

camada superior poderia definir as regras para a troca de mensagens entre os processos de aplicação específicos.

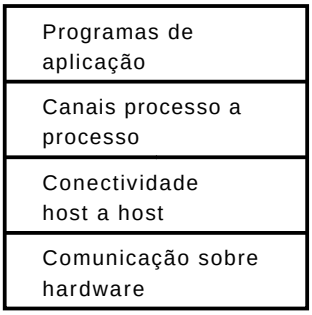


Figura 1.9: Exemplo de arquitetura em camadas.

No caso dos canais de comunicação lógicos fim a fim entre os processos de aplicação, para atender aos dois tipos de aplicações descritos na seção 1.7 (aplicações tipo pedido/-resposta e aplicações tipo fluxo de dados tempo real), poder-se-ia ter dois canais distintos, como mostrado na figura 1.10, um oferecendo um serviço orientado a conexão e garantido, e outro oferecendo um serviço não orientado a conexão. Esta arquitetura de redes, com quatro camadas de protocolos, é muito próxima do modelo utilizado pela Internet.

Na arquitetura Internet os diversos protocolos estão organizados em quatro camadas

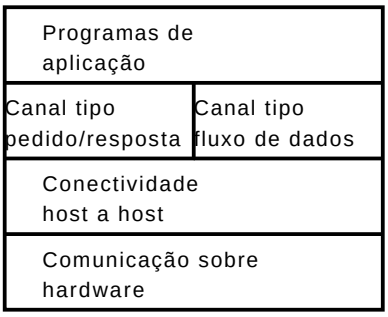


Figura 1.10: Arquitetura em camadas com alternativas de serviços em uma dada camada.

chamadas: a **camada de aplicação**, a **camada de transporte**, a **camada rede**[1] e a **camada enlace/física**[2]. A figura 1.11 mostra uma ilustração da pilha de protocolos Internet.

Figura 1.11: Camadas da arquitetura Internet.

Camada de Aplicação

Os protocolos da camada de aplicação definem as regras e o formato das mensagens que são trocadas entre as aplicações de rede, por exemplo, a aplicação WWW (*world wide web*) é governada pelas regras do protocolo de aplicação HTTP (*hiper text transfer protocol*); o correio eletrônico envia as mensagens usando o protocolo de aplicação SMTP (*simple mail transfer protocol*); a transferência de arquivos usa o protocolo de aplicação FTP (*file transfer protocol*). As mensagens trocadas entre as entidades da camada aplicação utilizam os canais disponibilizados pelos protocolos da camada inferior.

Camada de Transporte

Os protocolos da camada de transporte garantem um canal de comunicação lógico fim-a-fim entre os processos rodando no lado do cliente e no lado do servidor, para que as aplicações possam trocar mensagens entre si. Esta camada oferece dois tipos de serviços às aplicações:

[1] O modelo de referência da arquitetura Internet define esta camada como **camada inter-rede**, entretanto, muitos autores se referem à ela simplesmente como **camada rede**.

[2] O modelo Internet não especifica claramente o que acontece abaixo da camada rede; alguns autores, como Tanenbaum (2003), se referem a esta camada como **camada host/rede**; outros autores, como Kurose e Ross (2006a), a dividem em **camada enlace** e **camada física**, como no modelo OSI.

- Um serviço orientado a conexão, fornecido pelo protocolo TCP (*Transmission Control Protocol*), utilizado por aplicações tipo pedido/resposta;
- Um serviço de datagrama, não orientado a conexão, através do protocolo UDP (*User Datagram Protocol*), utilizado por aplicações tipo fluxo de dados em tempo real.

Como em cada computador da rede podemos ter diferentes processos de aplicação rodando, por exemplo, várias seções de navegadores Web, um dos serviços oferecidos pela camada de transporte é a multiplexação/demultiplexação de aplicações, entregando as mensagens em "portas" específicas para cada processo.

Camada rede

A camada rede é fundamentada num serviço não-orientado a conexão, no qual as mensagens são fragmentadas em pacotes, chamados datagramas, os quais atravessam a rede de roteador em roteador, desde o computador origem até o computador destino usando a técnica de comutação de pacotes. Nesta viagem, uma das tarefas dos protocolos da camada rede é definir a rota que seguirão os datagramas. Na Internet o principal protocolo da camada rede é o protocolo IP (*Internet Protocol*).

Os componentes principais da camada rede são:

- O protocolo IP, que define o formato do datagrama e a forma de endereçamento;
- Os protocolos e algoritmos de roteamento.

A camada rede envolve cada computador e roteador do caminho entre o computador origem e o destino, diferentemente das camadas de aplicação e transporte que somente precisam implementadas nas duas pontas da comunicação.

A camada rede se situa logo abaixo da camada de transporte na pilha de protocolos. Enquanto os protocolos de transporte oferecem comunicação lógica entre processos rodando em diferentes computadores, a camada rede oferece comunicação lógica entre os computadores.

Camada Enlace/Física

Para mover um pacote de um nó até o nó adjacente, dentro de uma determinada rota, a camada rede necessita dos serviços dos protocolos da camada de enlace. Por exemplo, para transferir dados entre dois computadores conectados em uma rede local Ethernet, é utilizado um protocolo de enlace de múltiplo acesso, como o CSMA-CD (*carrier sense multiple acssess protocol with colision detection*). Já no caso de dois computadores conectados por um enlace ponto a ponto, como uma linha discada, é utilizado um protocolo de enlace ponto-a-ponto, como o PPP (*point to point protocol*).

Vinculado à camada enlace está a camada física, que é responsável por mover os bits que compõe os dados entre um nó e outro utilizando um meio físico específico. Os meios físicos podem ser cabos coaxiais, fios de cobre, fibras ópticas ou o ar a partir do uso do espectro de freqüência de rádio.

A figura 1.12 ilustra, na forma de um mapa conceitual, uma relação da arquitetura Internet com os principais conceitos e tecnologias da área de redes de computadores.

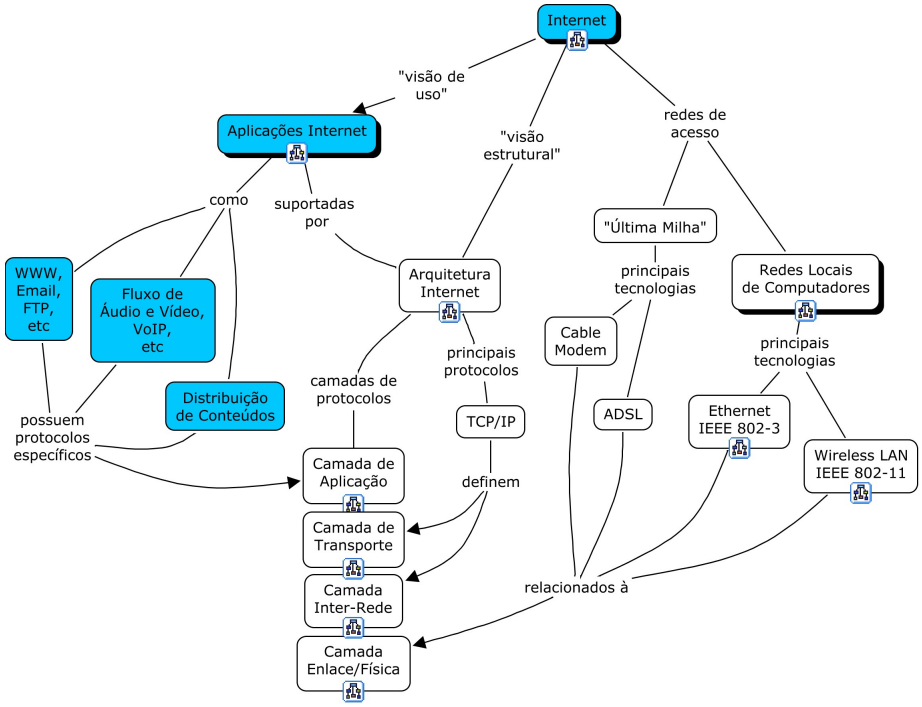


Figura 1.12: Visão da arquitetura Internet.

Capítulo 2

Aplicações Internet e Protocolos de Aplicação

As aplicações são a "razão de ser" da Internet, permitindo que os usuários possam fazer coisas úteis e interessantes na rede. Sem as aplicações, a Internet não teria sentido.

As aplicações de rede são programas que rodam nos sistemas terminais ou *hosts* e se comunicam entre si através da rede. São programas de aplicação típicos da Internet: o login remoto a sistemas (Telnet ou SSH), a transferência de arquivos (FTP), o correio eletrônico (e-mail), a paginação na Web ou WWW (*world wide web*), o bate-papo em rede (chat), telefonia na Internet (VoIP), a vídeo conferência, a execução de áudio e vídeo, etc.

As aplicações de rede são programas ou, como dizem no jargão dos sistemas operacionais, processos que se comunicam entre si pela da troca de mensagens através da rede.

A Internet oferece o suporte para a troca de mensagens entre as aplicações através de canais de comunicação lógicos, que são oferecidos pelos protocolos de transporte da Internet, o TCP e o UDP.

2.1 Protocolos de aplicação

Cada aplicação utiliza protocolos específicos, chamados protocolos de aplicação, que definem como os processos de aplicação, rodando em diferentes computadores, trocam mensagens entre si. Em particular, os protocolos de aplicação definem:

- Os tipos de mensagens trocadas, por exemplo, uma mensagem de solicitação ou resposta;
- A sintaxe e a semântica das mensagens, definindo os campos de cada mensagem e seu significado;
- As regras definindo quando e como um processo envia ou responde uma mensagem.

Os protocolos de aplicação, apesar de importantes, são apenas uma pequena parte de uma aplicação de rede. Por exemplo, a aplicação WWW é uma aplicação de rede que

permite aos usuários obterem documentos da Web sob demanda. Os componentes da aplicação WWW incluem documentos em formato HTML (*hypertext markup language*), navegadores Web (como o Firefox ou o Internet Explorer), servidores de páginas Web (como o Apache do Linux) e o protocolo de aplicação HTTP (*hiper text transfer protocol*).

2.2 Clientes e servidores

Uma aplicação de rede tem tipicamente duas partes, um lado cliente e um lado servidor que se comunicam entre si. Por exemplo, um navegador Web implementa o lado cliente do HTTP, e um servidor Web implementa o lado servidor HTTP.

Para algumas aplicações, um computador pode implementar ora o lado cliente ora o lado servidor. Por exemplo, considere um de acesso remoto via Telnet entre um computador A e um computador B. Se o computador A inicia a seção Telnet, então A é o cliente e B é o servidor. Por outro lado, se o computador B inicia a seção, ele é que será o cliente e A o servidor.

2.3 Portas e comunicação através da rede

Uma aplicação envolve a comunicação de dois processos através da rede. Os dois processos se comunicam através do envio e recebimento de mensagens através mecanismos chamados portas (*sockets*). Os processos assumem que há uma infraestrutura de transporte no outro lado da porta do processo emissor que transportará as mensagens até a porta do processo destino.

O conceito de portas faz parte da implementação dos protocolos de transporte da Internet TCP e UDP (ver página **??**). Em resumo, pode-se dizer que os protocolos de transporte estabelecem um canal de comunicação lógico para a transferência de mensagens porta-a-porta entre os processos de aplicação rodando em dois computadores remotos.

2.4 Endereçamento das aplicações

Para que um processo em um computador possa enviar uma mensagem a um computador remoto, ele deve endereçar quem vai receber a mensagem. O endereço envolve duas peças de informação: (1) o nome ou o endereço IP da máquina destino; (2) o número da porta do processo do lado do receptor.

Por exemplo, para endereçar o servidor Web do IF-SC devemos fornecer o endereço IP ou o nome do servidor, por exemplo, 200.18.10.1 ou www.ifsc.edu.br, respectivamente. Quanto ao número da porta, como algumas aplicações tem o número padronizado, e em geral o agente usuário escolhe o número de porta automaticamente em função da aplicação em uso.

2.5 Agente usuário

O agente usuário é a interface entre o usuário e a aplicação de rede. Mais especificamente, o agente usuário é um programa de computador, comercial ou de domínio público, que implementa a interface do lado cliente de uma aplicação de rede. Por exemplo, o agente usuário para um cliente WWW pode ser, por exemplo, o navegador Firefox.

2.6 Serviços de transporte utilizados pelas aplicações

A Internet dispõe de dois serviços de transporte para os protocolos de aplicação, o UDP e o TCP. Quando uma aplicação é projetada para a Internet, a primeira decisão do projetista deve ser definir qual protocolo de transporte será utilizado.

A escolha dependerá do tipo serviço que a aplicação vai necessitar. Quanto aos tipos de serviços requisitados pelas aplicações, podemos classificá-los em três dimensões:

- Quanto à perda de dados

  Algumas aplicações, como por exemplo, transmissão de áudio, podem tolerar algumas perdas; outras aplicações, como por exemplo, uma transferência de arquivos, requerem transferência confiável.

- Quanto aos requisitos temporais

  Algumas aplicações, como por exemplo, telefonia na Internet, requerem baixo retardo para serem viáveis, outras, como uma mensagem de correio eletrônico, não tem restrições temporais.

- Quanto à largura de banda

  Algumas aplicações, como por exemplo, multimídia, requerem quantia mínima de banda para serem "viáveis"; outras aplicações são mais "elásticas" e conseguem usar qualquer quantia de banda disponível, como por exemplo, a paginação na Web.

Para atender a estes requisitos, os protocolos de transporte da Internet TCP e UDP oferecem as seguintes facilidades:

Serviço TCP

- Serviço orientado a conexão: uma abertura de conexão é requerida entre cliente e servidor;
- Transporte confiável: garante comunicação livre de erros entre o processo emissor e receptor;
- Controle de fluxo: evita que o emissor possa "afogar" com dados um receptor mais lento;
- Controle de congestionamento: permite "estrangular" o emissor quando a rede está sobrecarregada.

- Não provê: garantias temporais ou de banda mínima.

Serviço UDP

- Transferência de dados não confiável: não há garantia de entrega de dados livre de erros;
- Não Provê: abertura de conexão, confiabilidade, controle de fluxo, controle de congestionamento, garantias temporais ou de banda mínima.

2.7 Protocolo HTTP

O protocolo HTTP define como os navegadores Web (clientes) requisitam páginas de servidores Web. Para requisitar uma página Web, o cliente HTTP primeiramente abre uma conexão TCP na porta 80 do servidor. Uma vez aberta a conexão TCP o cliente envia mensagens de requisição HTTP para o servidor Web, o qual responde com uma mensagem de resposta HTTP que contém os objetos solicitados.

Há duas versões do protocolo HTTP implementadas pelos navegadores, o HTTP/1.0 (RFC1945) e o HTTP/1.1 (RFC2068) e ambas as versões usam como protocolo de transporte o TCP. O HTTP/1.0 usa o que se chama conexões não persistentes, onde após a requisição de cada objeto, o servidor responde e encerra a conexão TCP. O HTTP/1.1 permitiu melhorar o desempenho dos navegadores Web através do uso de conexões persistentes, onde o servidor mantém a conexão TCP aberta após o envio da resposta. Desta forma, as requisições e as respostas subsequentes entre o mesmo par cliente/servidor podem utilizar a mesma conexão já aberta, eliminando o tempo de abertura de conexão. Caso a conexão deixe de ser utilizada por um certo tempo o servidor se encarrega de liberar a conexão.

Formato das mensagens HTTP

O protocolo HTTP é baseado no paradigma pedido/resposta, havendo dois tipos de mensagens: **mensagens de requisição** e **mensagens de resposta**.

A **mensagens de requisição** (*request*) tem a seguinte estrutura:

```
GET /diretorio/pagina.html
Host: www.ifsc.edu.br
Connection: close
User-agent: Mozilla/4.0
Accept-language:pt
(extra carriage return, line feed)
```

A primeira linha apresenta o comando básico para requisição de uma página Web, seguido pela parte do URL que indica o caminho e o nome do objeto que se deseja (GET /diretorio/pagina.html). As linhas seguintes, chamadas de cabeçalho, são opcionais. A segunda linha (Host: www.ifsc.edu.br) indica o nome computador onde reside o objeto; a

terceira linha (Connection: close) informa para fechar a conexão após envio da resposta; a quarta linha (User-agent: Mozilla/4.0) indica o tipo do agente usuário utilizado e a linha (Accept-language:pt) indica que o português é a língua preferencial. Do ponto de vista do usuário o mesmo só enxerga o endereço URL que digitou e o navegador monta e envia as mensagens HTTP de forma transparente.

A **mensagens de resposta** (*response*) tem a seguinte estrutura:

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 09:23:24 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data . . .)
```

A resposta tem três partes, uma primeira linha informando o estado (status) da solicitação, seis linhas de cabeçalho e os dados que compõe objeto solicitado. A primeira linha indica a versão do protocolo, o código e estado da mensagem (HTTP/1.1 200 OK). A segunda linha (Connection: close) indica que a conexão será encerrada; a terceira linha (Date: Thu, 06 Aug 1998 12:00:15 GMT) informa a data da última modificação no objeto solicitado, utilizada por servidores proxy; a quarta linha (Server: Apache/1.3.0 (Unix)) indica o tipo do servidor, a quinta linha (Content-Length: 6821) indica o tamanho do objeto em bytes e a última linha (Content-Type: text/html) informa o conteúdo da mensagem. Os dados vem em seguida. Os códigos de estado (status) mais comuns são:

```
200 OK
301 Moved Permanently
400 Bad Request
404 Not Found
505 HTTP Version Not Supported
```

Exercício

É possível ver as mensagens trocadas pelo protocolo HTTP, executando manualmente os comandos em uma conexão TCP, na porta 80, com um servidor Web. Para tal, pode-se fazer primeiramente um Telnet (acesso remoto), na porta 80 de um servidor Web. Em seguida, uma vez estabelecida a conexão, pode-se trocar comandos HTTP manualmente.

Por exemplo:

```
$ telnet www.das.ufsc.br 80
  Trying 150.162.12.10 ...
  Connected to www.das.ufsc.br.
  Escape character is '^]'.
GET /~cantu/index.html
```

permite estabelecer um canal TCP na porta 80 com servidor www.das.ufsc.br e receber o arquivo HTML com a página do usuário cantu.

2.8 Protocolo FTP

O protocolo FTP (*file transfer protocol*) (RFC959) é o protocolo que suporta a aplicação de transferência de arquivos entre computadores.

Numa sessão FTP um usuário pode transferir arquivos de um computador remoto para um computador local e vice-versa (*download* e *upload*, respectivamente). Uma maneira típica de realizar um FTP é utilizar um terminal de texto do Linux (ou Windows), iniciando a aplicação e executando os comandos apropriados. O primeiro comando do usuário (`open`) deve fornecer endereço do computador remoto, estabelecendo com isto uma conexão TCP entre o processo FTP cliente e servidor. Depois o usuário deve fornecer sua identificação e sua senha. Outros comandos possíveis são: mudar de diretório (`cd`), solicitar arquivos (`get`), enviar arquivos (`put`), etc.

O protocolo FTP usa duas conexões paralelas TCP para transferir arquivos: uma para controle da conexão e outra para a transferência de dados. O controle de conexão é usado para trocar informações como a identificação do usuário e senha e para transferir os comandos FTP. A conexão de dados é usada para transferir os arquivos propriamente ditos. Cada uma destas duas conexões TCP usa uma porta específica: a conexão de controle de conexão usa a porta 21 e a conexão de dados usa a porta 20.

A figura 2.1 ilustra o serviço FTP, conforme descrito na RFC959. Na figura os blocos *Server/User PI* identificam o interpretador de protocolo e *Server/User DTP* identificam o processo de transfereência de dados, tanto do lado do servidor quanto do lado do usuário.

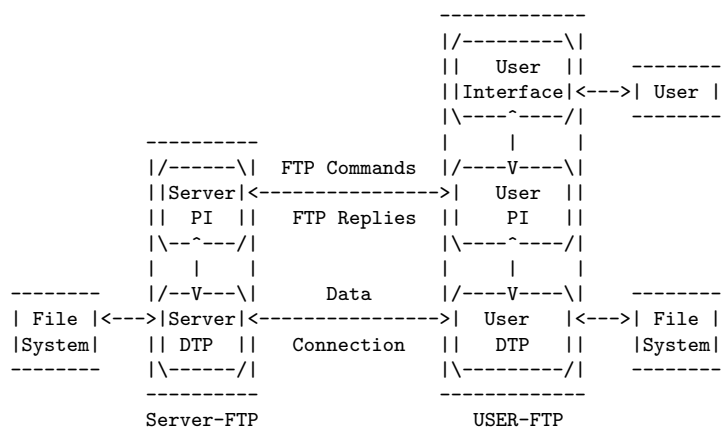


Figura 2.1: Serviço FTP (RFC959).

As mensagens de controle FTP são codificadas em formato ASCII, com caracteres maiúsculos, como nos exemplos abaixo.

```
USER NAME (USER)
PASSWORD (PASS)
CHANGE WORKING DIRECTORY (CWD)
LOGOUT (QUIT)
RETRIEVE (RETR)
STORE (STOR)
```

As respostas são sempre de três dígitos, com uma mensagem opcional seguindo o número, como nos exemplos abaixo..

```
331 User name OK, password required
125 Data conection already open; transfer starting
425 Can´t open data conection
452 Error writing file.
```

2.9 Protocolo SMTP

O protocolo SMTP (*simple mail transfer protocol*) (RFC821) é o protocolo utilizado pelo correio eletrônico, ou email (*electronic mail*). Ele usa o serviço de transferência de dados confiável do TCP para transferir uma mensagem desde o remetente até a caixa postal do destinatário. O SMTP, como outros protocolos de aplicação, tem dois lados, o lado cliente e o lado servidor. Quem envia a mensagem faz o papel do cliente e quem recebe de servidor, todavia, ambos os lados do SMTP devem ser implementados em cada servidor de email.

Através do protocolo SMTP, o servidor de email envia as mensagens que estão na sua fila de saída em direção ao servidor destino. Caso o servidor destino não esteja acessível, o servidor de email tentará enviá-la mais tarde, persistindo nestas tentativas por alguns dias, quando então remove a mensagem e notifica quem a tinha enviado.

Para ler uma mensagem em sua caixa postal o destinatário da mensagem deve, a partir de seu leitor de email, requisitá-la de seu servidor. O servidor de email então requisita uma autenticação do usuário, através de uma identificação e uma senha, para depois repassar as mensagens que porventura chegaram a esta pessoa.

A figura 2.2 ilustra o modelo de uso do SMTP.

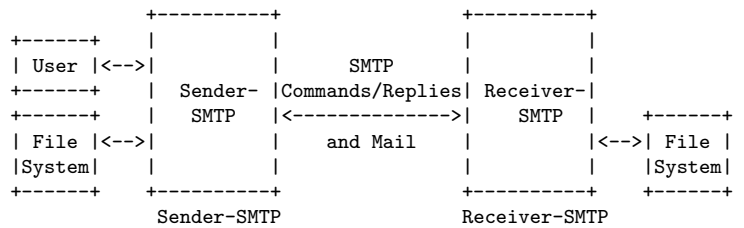


Figura 2.2: Modelo de uso do SMTP (RFC821).

As mensagens trocadas pelo protocolo SMTP são mensagens em caracteres ASCII. Para enviar uma mensagem, o cliente SMTP estabelece uma conexão TCP, na porta 25, com o servidor SMTP. Uma vez estabelecida à conexão TCP, cliente e servidores de email entram em uma fase de apresentação mútua (*handshaking*), trocando algumas informações (como, o cliente indica o endereço de email do emissor e do destinatário), antes de enviarem a mensagem eletrônica em si.

Veja um exemplo de uma seqüência de mensagens SMTP, constantes na RFC821, trocadas entre Smith no host Alpha.ARPA e Jones, Green, e Brown no host Beta.ARPA. Assume-se que o host Alpha contata o host Beta diretamente.

```
S: MAIL FROM:<Smith@Alpha.ARPA>
R: 250 OK

S: RCPT TO:<Jones@Beta.ARPA>
R: 250 OK

S: RCPT TO:<Green@Beta.ARPA>
R: 550 No such user here

S: RCPT TO:<Brown@Beta.ARPA>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: <CRLF>.<CRLF>
R: 250 OK
```

O email foi aceito por Jones e Brown. O usuário Green não tem caixa postal no host Beta.

Protocolo para leitura de email POP3

Uma vez enviada uma mensagem eletrônica, ela é colocada na caixa postal do destinatário. Uma maneira natural para o destinatário de ler as mensagens de sua caixa postal, seria acessar diretamente o seu servidor de email através de um web-mail.

Caso o usuário deseje utilizar um leitor de email diretamente em seu computador pessoal (como Outlook), vai haver a necessidade de transferir as mensagens do usuário do seu servidor de email para seu computador. Para realizar esta tarefa normalmente utiliza-se um protocolo POP3 (RFC1939). O processo inicia quando o cliente abre uma conexão TCP na porta 110 do servidor de email. Com a conexão TCP estabelecida, o POP3 processa três fases: autorização (quando o usuário envia seu nome e senha e recebe suas mensagens), transação (quando o usuário requisita ações sobre as mensagens, como por exemplo marcando algumas para serem apagadas) e atualização (quando o usuário encerra a sessão e o servidor apaga as mensagens marcadas para serem removidas).

2.10 DNS

O **DNS** (*domain name system*) (RFC1034 e RFC 1035) é um sistema de nomes de domínio que permite traduzir um nome de domínio (como www.ifsc.edu.br) em um endereço IP (como 200.18.10.1). Isto facilita na medida em que não precisamos mais memorizar endereços IP, mas sim nomes de domínio, muito mais fáceis de serem lembrados e ao mesmo tempo identificados com o proprietário do domínio.

Para resolver os nomes, o DNS realiza uma busca em um banco de dados distribuído,

armazenado em sistemas cooperativos independentes, chamados **resolvedores de nomes**, que fazem a tradução do nome de domínio em endereço IP.

Uma solicitação de tradução de um nome a um resolvedor de nomes pode usar um ou mais resolvedores para obter o endereço desejado. Há dois modos possíveis para um servidor resolver um nome: resolução interativa ou resolução recursiva. Em ambos os casos, o servidor consultado verifica se o nome solicitado pertence a seu sub-domínio. Se for o caso, traduz o nome ao endereço de acordo com sua base de dados. Se não puder resolver o nome completamente, verifica o tipo de solicitação feita pelo cliente. Se o cliente solicitou busca recursiva o servidor contata um DNS que possa resolver o nome e devolve a resposta ao cliente. Caso a solicitação foi do tipo interativa, ele fornece o nome de um DNS ao cliente e não a resposta da resolução completa do nome.

Uma configuração típica de para um programa usuário socilitar consultas ao serviço DNS é mostrada na figura 2.3. O programa usuário interage com o DNS através de um resolvedor de nomes local, enviando solicitações (*queries*) e recebendo respostas (*responses*). O resolvedor de nomes pode ter as informações solicitadas pelo usuário em memória *cache*, ou, caso não as tenha, ele faz uma consulta a um resolvedor de nomes externo. A resposta obtida do servidor externo é então passada ao usuário solicitante e também armazenada na *cache* do resolver local visando atender possíveis consultas futuras ao mesmo endereço. O resolvedor de nomes externo, por sua vez, pode realizar socilitações a outros servidores para obter a resposta que lhe foi solicitada.

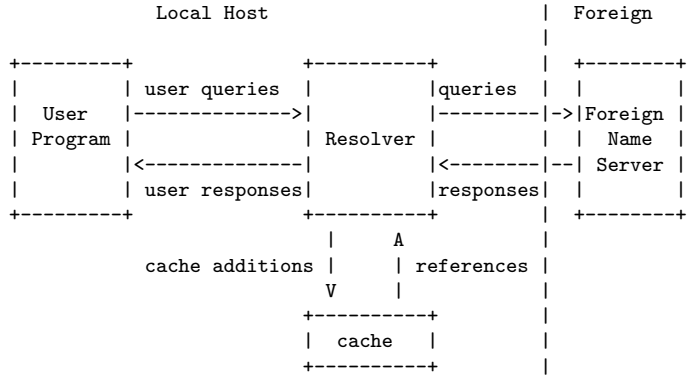


Figura 2.3: DNS (RFC1035).

Todo resolvedor de nomes precisa saber como contatar pelo menos um servidor de nomes raiz. Em adição, um resolvedor de nomes deve saber o endereço de um resolvedor de nomes de domínio imediatamente superior (servidor pai). Todos os nomes recentemente usados são armazenados na na memória *cache* local do resolvedor de nomes, bem como a informação de como foram obtidos. Como a informação em memória pode estar desatualizada, o resolvedor de nomes marca como não autoritativa (*non authoritative*), podendo o cliente contatar a autoridade para ver se o nome ainda é válido.

Mensagem DNS

Toda comunicação do serviço DNS é realizada por meio de uma mensagem cujos campos são mostrados na figura 2.4.

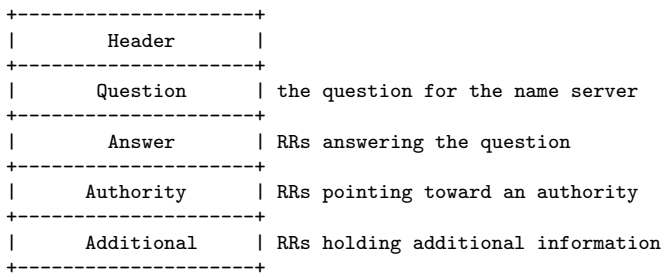


Figura 2.4: Formato da mensagem DNS (RFC1035).

O cabeçalho (*Header*) é sempre presente e especifica a quantidade campos presentes na sequência da mensagem e também se a mensagem é uma solicitação ou resposta. O campo questão (*Question*) contém a solicitação ao resolvedor de nomes. O campo resposta (*Answer*) contém a resposta, chamada de RR (*resource record*). O campo autoridade (*Authority*) contém RRs que apontam para um resolverdor de nomes com "autoridade" para responder a solicitação. O campo adicional (*Additional*) contém RRs outras indormações relacionadas a solicitação.

As mensagens DNS são encaminhadas como mensagens de aplicação utilizando como protocolo de transporte o UDP e a porta 53.

O serviço DNS está situado no nível da camada aplicação, contudo é um serviço auxiliar utilizado pelas aplicações usuários, as quais, antes iniciar uma comunicação na Internet, precisam obter o endereço IP do seu correspondente.

Capítulo 3

Camada de Transporte da Internet

Situada entre a camada de aplicação e a camada rede, a camada de transporte tem a função de prover um **canal de comunicação lógico fim-a-fim entre os processos de aplicação** rodando em diferentes computadores, sem se preocupar com os detalhes da infra-estrutura física usada para carregar as mensagens entre eles.

Os protocolos de transporte são implementados nos sistemas terminais, não necessitando serem implementados nos roteadores da rede, os quais atuam somente até a camada rede.

No lado do emissor, as **mensagens** recebidas das aplicações são fragmentadas e encapsuladas em **unidades de dados de protocolos**, ou **PDUs** (*protocol data unit*), chamadas **segmentos**, aos quais adiciona-se um cabeçalho. Cada segmento é então repassado a camada rede que por sua vez encapsula em unidades de dados de protocolos da camada rede, ou **datagramas**.

3.1 O serviço de multiplexação e demultiplexação de aplicações

O protocolo IP entrega dados entre dois sistemas terminais (*hosts*), cada qual identificado por seu endereço IP. A responsabilidade dos protocolos de transporte é entregar estes dados (segmentos) a aplicação apropriada rodando em cada *host*.

Cada um dos segmentos da camada transporte tem em seu cabeçalho um campo que indica a qual processo o mesmo deve ser entregue. Estes campos são conhecidos como números de porta. O cabeçalho inclui um campo com o número de porta do emissor e o número de porta do receptor, como mostra a figura 3.1.

Os números de porta variam de 0 a 65535, sendo que até a porta 1023 são números reservados para aplicações específicas. Na Internet cada uma das aplicações mais conhecidas utilizam portas padronizadas, como exemplificado abaixo.

```
------------------------------------------------------------------
13  TCP daytime  Daytime
13  UDP daytime  Daytime
```

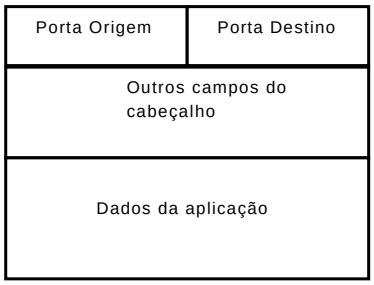


Figura 3.1: Portas no cabeçalho dos protocolos de transporte.

```
20  TCP ftp-data File Transfer [Default Data]
21  TCP ftp File Transfer [Control]
22  TCP ssh      SSH Remote Login Protocol
23  TCP telnet   Telnet
25  TCP smtp     Simple Mail Transfer
53  UDP domain   Domain Name Server
80  TCP www-http World Wide Web HTTP
110 TCP pop3     Post Office Protocol - Ver 3
443 TCP https    http protocol over TLS/SSL
---------------------------------------------------------------------
```

Um servidor de aplicações espera conexões em portas bem conhecidas. Por exemplo, a aplicação Telnet utiliza a porta 23 para aceitar conexões. Quando um cliente Telnet inicia uma seção, ele envia ao servidor um segmento TCP com porta destino 23 e coloca como número de porta origem uma porta que não esteja sendo utilizada no host cliente, por exemplo, a porta X. A porta X será onde o cliente vai esperar a resposta do servidor. Quando o servidor recebe o segmento, ele verifica que o mesmo é endereçado a porta 23 e então sabe que se trata da aplicação Telnet. No envio da resposta o servidor inverte as portas origem e destino. Enviando ao cliente um segmento com porta destino X e origem 23.

3.2 UDP (*User Datagram Protocol*)

O **protocolo UDP** (RFC768) oferece às aplicações um **serviço de datagramas**, estendendo o serviço oferecido pelo IP com a **multiplexação e demultiplexação** de aplicações e um mecanismo de **detecção de erros**.

Características do UDP:

- Não orientado a conexão, não introduzindo, portanto, atrasos para esta tarefa.
- Tem pequeno *overhead* (informações de controle) no cabeçalho.
- Não provê mecanismos para controle de fluxo, controle de congestionamento, garantias temporais ou de banda mínima.

Por estas características é apropriado para aplicações tempo real, como telefonia e transferência de áudio e vídeo sobre a Internet.

O formato do "segmento" UDP (alguns autores chamam de datagrama UDP, pois pouco acrescenta ao datagrama IP) é bastante simples, além dos campos reservados para as portas de origem e destino, há um campo que indica o comprimento do segmento (*length*) e o *checksum*, o qual é utilizado para o detecção de erros no segmento, como mostra a figura 3.2. O campo de dados da aplicação é preenchido com os dados da aplicação, por exemplo, para aplicações de aúdio tempo real o campo é preenchido com as amostras de aúdio.

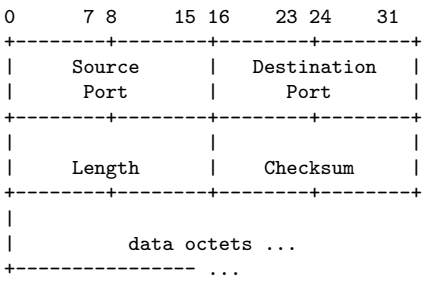


Figura 3.2: Formato do datagrama UDP (RFC768).

3.2.1 *Checksum*

O ***checksum*** do UDP permite a **detecção de erros** nos dados transmitidos. Para isto, o emissor UDP faz o complemento 1 da soma de todos as palavras de 16 bits do segmento e coloca o resultado no campo cheksum. Por exemplo, suponha que temos três palavras de 16 bits:

```
 0110011001100110
 0101010101010101
 0000111100001111
```

A soma será

```
 0110011001100110
 0101010101010101
+----------------
 1011101110111011
```

Adicionando a terceira palavra a esta soma

```
 1011101110111011
 0000111100001111
+----------------
 1100101011001010
```

O complemento 1 é obtido invertendo cada bit 1 por 0 e vice-versa. Desta forma o complemento da soma será `0011010100110101`, o qual será o *cheksum*. No lado do receptor UDP, todas as palavras de 16 bits recebidas são adicionadas, incluindo o cheksum.Se não houve erros na transmissão, a soma será `1111111111111111`. Se um dos bits for 0, então é sabido que houve erros.

3.3 TCP (*Transmission Control Protocol*)

O **protocolo TCP** (RFC793), como o UDP, também oferece a **multiplexação/demultiplexação de aplicações** e o mecanismo de **detecção de erros**. A grande diferença é que o TCP é um protocolo **orientado a conexão** e com **transferência garantida**, onde os dois processos devem acordar entre eles uma abertura de conexão para que os dados possam ser transferidos. Além destas características, o TCP integra ainda um serviço de **controle de fluxo**, que assegura que nenhum dos lados da comunicação envie pacotes rápido demais, pois uma aplicação em um lado pode não conseguir processar a informação na velocidade que está recebendo, e um serviço de **controle de congestionamento** ajuda a prevenir congestionamentos na rede.

Uma conexão TCP é uma conexão full-duplex (isto é, em ambos os sentidos e simultânea) e é sempre fim-a-fim, entre o host emissor e o host receptor. Uma vez estabelecida à conexão os dois processos podem trocar informações. O processo cliente passa o bloco de dados através da porta apropriada. O TCP então manipula estes dados, dirigindo para o buffer de envio. Os dados são então fragmentados e encapsulados na forma de segmentos. Os segmentos, por sua vez, são passados a camada rede onde eles são separadamente encapsulados em datagramas IP, que são enviados através da rede. Quando o TCP do receptor recebe os dados, os mesmos são recebidos no buffer de recepção. A aplicação no lado do receptor então lê os dados a partir deste buffer.

A figura 3.3 ilustra, na forma de um mapa conceitual, uma visão dos principais conceitos da camada de transporte da Internet, mostrando explicitamente a relação entre os mesmos.

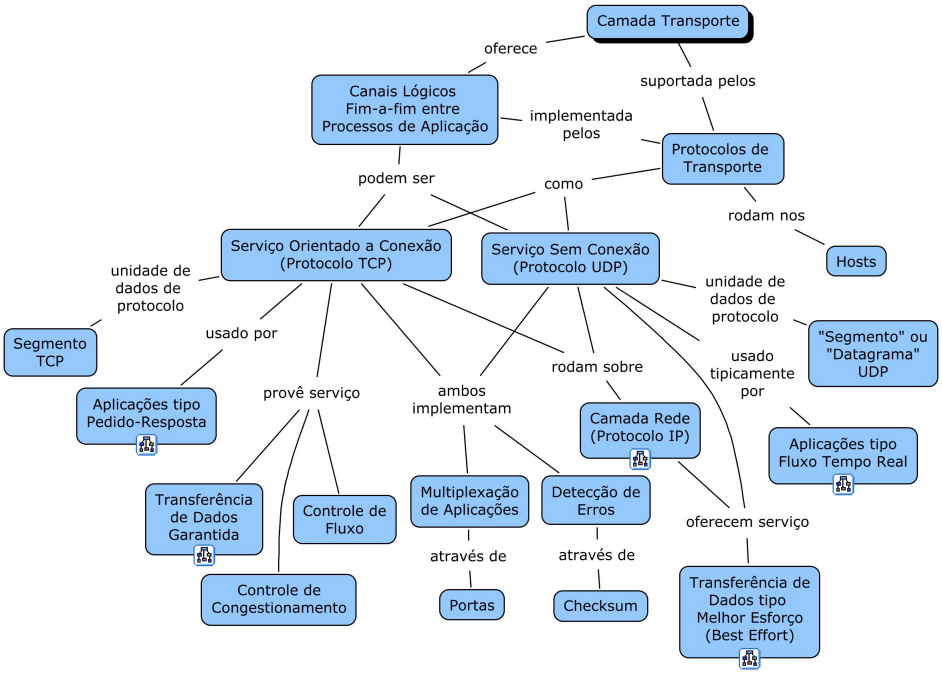


Figura 3.3: Visão dos principais conceitos da camada de transporte da Internet.

3.3.1 Mecanismo de transmissão Garantida

Para garantir uma entrega de dados livre de erros, muitos protocolos com transmissão garantida utilizam uma técnica conhecida como confirmação positiva com retransmissão. A técnica exige que um receptor comunique-se com a origem, retornando uma mensagem de reconhecimento (*acknoledge*) a medida que recebe os dados. O transmissor, por sua vez, inicia um temporizador para cada pacote que envia e retransmite o pacote caso este temporizador se complete antes que chegue uma confirmação de recebimento.

O protocolo Bit-alternado é um protocolo tipo pára e espera (*stop-and-wait*), isto é, para cada pacote que envia aguarda um reconhecimento. Este protocolo utiliza números de sequência 0 e 1 para identificar os pacotes enviados. A figura 3.4 (a) ilustra um caso do funcionamento do protocolo Bit-alternado onde não ocorreram erros no envio dos pacotes e nem na recepção dos reconhecimentos. A figura 3.4 (b) ilustra um caso no qual houve uma perda de pacotes. A figura 3.4 (c) ilustra um caso no qual houve uma perda de um reconhecimento e a 3.4 figura (d) um caso de estouro do tempozador prematuramente.

O problema de um protocolo como o Bit-alternado é que o emissor deve esperar o reconhecimento de cada pacote antes que um novo pacote possa ser enviado, o que torna a transmissão bastante ineficiente. Protocolos mais elaborados, como o TCP, permitem que o emissor transmita múltiplos pacotes antes de esperar uma confirmação. Este processo é chamado de *pipeline* e é geralmente implementado através de um mecanismo conhecido como "janelas deslizantes".

No mecanismo de janelas deslizantes o emissor pode enviar uma sequência de pacotes, contidos dentro de uma "janela" de tamanho fixo, antes de esperar uma confirmação. Neste caso, os pacotes devem ser identificados com uma faixa maior de números de sequência, uma vez que vários pacotes podem ser enviados antes de esperar um reconhecimento. Na figura 3.5 (a) , os pacotes contidos dentro da janela (numerados de 1 a 4) podem ser enviados em sequência. Quando o transmissor redece a confirmação do primeiro pacote da janela, a janela desliza, figura 3.5 (b), permitindo que um novo pacote seja enviado.

A figura 3.6 mostra uma comparação entre um protocolo pára e espera (*stop-and-wait*) e um protocolo que utiliza o mecanismo de janelas deslizantes (*pipeline*).

3.3.2 Segmento TCP

A figura 3.7 mostra a estrutura do segmento TCP. No cabeçalho, além dos números de porta e *checksum*, que também existem no UDP, há outros campos com informações necessárias a implementação do serviço de transferência garantida, controle de fluxo e controle de congestionamento.

Os campos fundamentais do segmento TCP são os seguintes:

- Porta origem e porta destino (*Source Port* e *Destination Port*);
- Número de seqüência e reconhecimento, utilizado para o emissor e receptor implementarem o serviço de transferência garantida (*Sequence Number* e *Acknowledgment Number*).

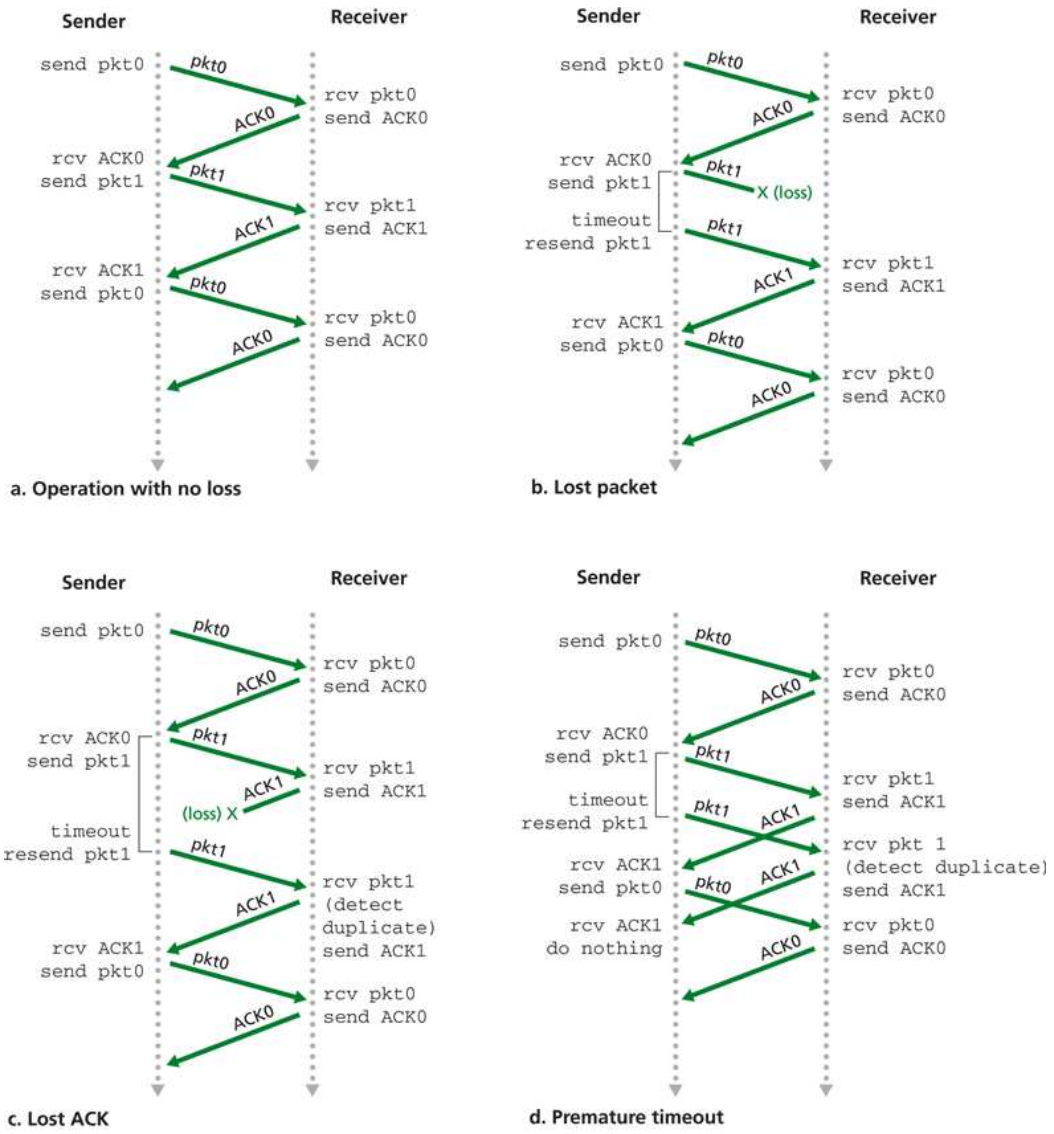


Figura 3.4: Envio de pacotes e reconhecimentos TCP (KUROSE; ROSS, 2006a).

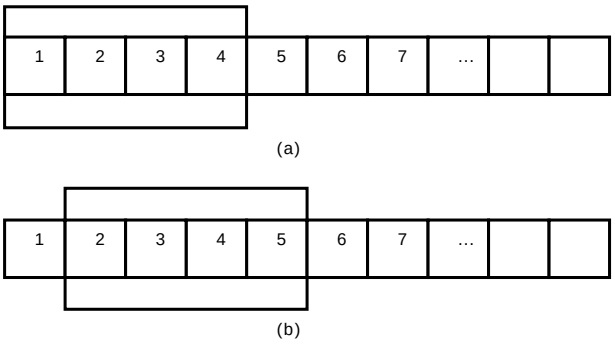


Figura 3.5: Mecanismo de janelas deslizantes.

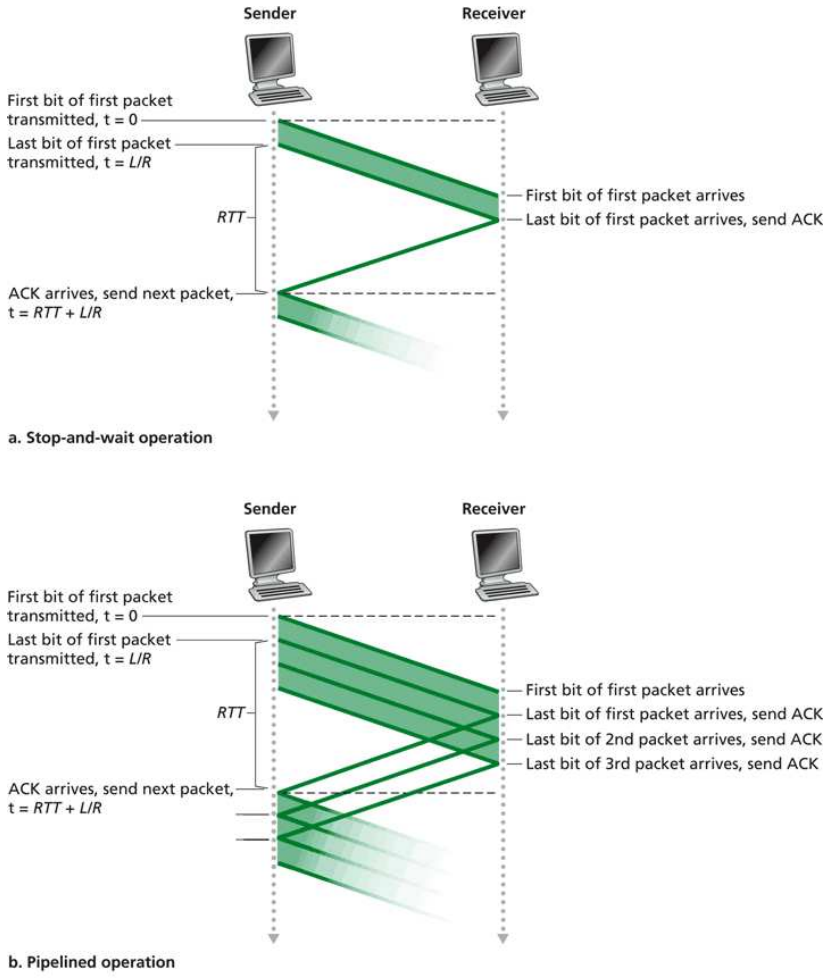


Figura 3.6: Comparação de um protocolo *stop-and-wait* e um *pipeline* (KUROSE; ROSS, 2006a).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figura 3.7: Formato do segmento TCP (RFC793).

- Tamanho da janela do receptor, usado para o controle de fluxo, e indica o número de bytes que o receptor é capaz de receber (*Window*).
- Seis bits de *flags*: o **Ack** é usado para indicar que o campo de reconhecimento é válido; o **Rst**, **Syn** e **Fin** são usados para abertura e encerramento de conexão, o **Psh** indica que o receptor deve passar imediatamente os dados para a camada superior e o **Urg** indica urgente (pouco usado).
- Tamanho do cabeçalho (*Data Offset*), especifica o tamanho da cabeçalho, que pode variar em funções do campo de opções; tipicamente o cabeçalho contém as 5 primeiras linhas, totalizando 20 bytes.
- O *checksum* é usado para detecção de erros.
- O campo de opções (*Options*) é usado quando o emissor e receptor precisam negociar o tamanho máximo de segmento (MSS).
- O *Padding* é utilizado para completar o campo opções (se necessário).

O campo de dados da aplicação (*data*) do segmento TCP, contém um fragmento ou pedaço dos dados da aplicação, cujo tamanho máximo, chamado de MSS (*maximum segment size*), depende da implementação do TCP. Os valores típicos são 1.500 bytes ou 512 bytes, não incluindo o cabeçalho. (Em geral o valor de MSS é escolhido para evitar a fragmentação do datagrama IP na camada inferior. Este valor em algumas implementações pode ser configurado manualmente ou estabelecido automaticamente pelo protocolo).

3.3.3 Números de seqüência e reconhecimento no TCP

Dois campos importantes do segmento TCP são os números de seqüência e reconhecimento, os quais fazem o controle da transferência de dados confiável.

Como vimos, os dados das aplicações são transportados pelos segmentos TCP. Caso as mensagens forem maior que o valor de MSS, tamanho máximo do segmento, as mesmas são fragmentadas para poderem ser acomodadas na parte de dados do segmento. Por exemplo, um arquivo GIF de 500K bytes trocado pelo HTTP será fragmentado em vários pedaços para ser transmitido pelo TCP. Os números de seqüência servem, portanto, para que o lado receptor TCP possa reordenar corretamente os dados recebidos.

Os números de seqüência não correspondem a uma série de segmentos transmitidos, mas refletem a quantidade de bytes que o TCP está transmitindo. Por exemplo, suponha que o bloco total de dados que será transmitido tenha 500.000 bytes, que o valor de MSS é de 1.000 bytes, e que o primeiro byte dos dados é numerado como zero. Para transmitir esta quantidade de bytes o TCP formará 500 segmentos. Ao primeiro segmento atribui-se o número de seqüência zero, ao segundo 1000, ao terceiro 2000 e assim por diante, como mostra a figura 3.8.

Os reconhecimentos servem para o receptor informar o emissor quais blocos que foram recebidos corretamente. Todavia, lembre-se que uma comunicação TCP é sempre *full-duplex*, o que significa que o host A pode estar recebendo dados do host B ao mesmo

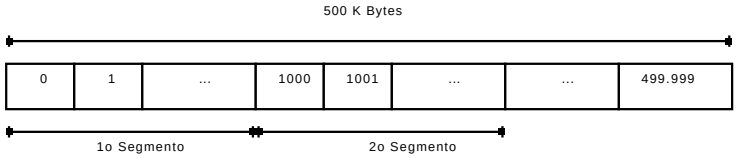


Figura 3.8: Números de sequência TCP.

tempo em que está enviando dados ao host B (como parte da mesma conexão TCP). Desta forma, haverá números de reconhecimentos para dados seguindo de A para B e outros para dados seguindo de B para A.

O número de reconhecimento que o host A coloca no seu segmento é o número de seqüência do próximo byte que o host A espera receber do host B.Por exemplo, suponha que o host A recebeu todos os bytes numerados de 0 a 535 de B e que está prestes a enviar um segmento a B. Neste caso, o host A coloca como número de reconhecimento 536, o que vai indicar a B que o mesmo recebeu todos os bytes até este número.

Em outro exemplo, suponha que o host A recebeu todos os bytes numerados de 0 a 535 de B e em seguida recebeu de B um segmento contendo bytes de 900 a 1000. Note que A não recebeu os bytes que vão de 536 a 899. Como A ainda está esperando bytes a partir de 536, ele reenvia a B um segmento com número de reconhecimento 536. Continuando este exemplo, suponha agora que A receba o segmento que faltava, com os bytes que vão de 536 a 899. Neste caso, como ele já recebeu inclusive os dados contendo os bytes de 900 a 1000, ele envia um reconhecimento com número 1001. Isto é chamado de reconhecimento cumulativo, que indica que recebeu todos os bytes até este número.

Telnet: Caso de estudo para números de seqüência e reconhecimento (KUROSE; ROSS, 2006b)

O Telnet é uma aplicação interativa usada para acesso remoto a sistemas e roda sobre o protocolo de transporte TCP.

O Telnet permite que um usuário utilize uma máquina A e estabeleça uma seção interativa em uma máquina B, como se estivesse utilizando um terminal. Quem solicita o Telnet assume o papel de cliente. Cada caractere digitado pelo usuário cliente será enviado ao computador remoto; o computador remoto então enviará uma cópia de cada caractere para ser mostrado na tela do cliente. Desta forma, cada caractere atravessa a rede duas vezes entre o tempo em que o usuário digita uma tecla e a visualização da mesma na tela.

Vamos examinar os segmentos TCP trocados durante uma seção Telnet. Suponha que o usuário tecla a letra **C**. Suponha ainda que os números de seqüência iniciais usados pelo cliente e pelo servidor sejam 42 e 79, respectivamente. Isto indica que o primeiro byte a ser enviado pelo cliente ao servidor terá o número de seqüência 42 e o primeiro byte a ser enviado pelo servidor ao cliente terá o número de seqüência 79. Lembre também que o número de reconhecimento indica o número de seqüência do próximo byte esperado. Desta forma, depois de estabelecia a conexão TCP, e antes do envio de quaisquer dados, o cliente esta esperando pelo byte 79 e o servidor está esperando pelo byte 42.

A figura 3.9 mostra três segmentos trocados entre o cliente e o servidor. O primeiro segmento é enviado pelo cliente, contendo um caractere ASCII com a letra **C** (número de

seqüência 42). O segundo segmento é enviado pelo servidor ao cliente e serve para dois propósitos: provê um reconhecimento do caractere recebido (número de reconhecimento 43) e envia o caractere **C** de volta para ser apresentado na tela do cliente (número de seqüência 79). No terceiro segmento trocado, o cliente reconhece o caractere recebido (número de reconhecimento 80).

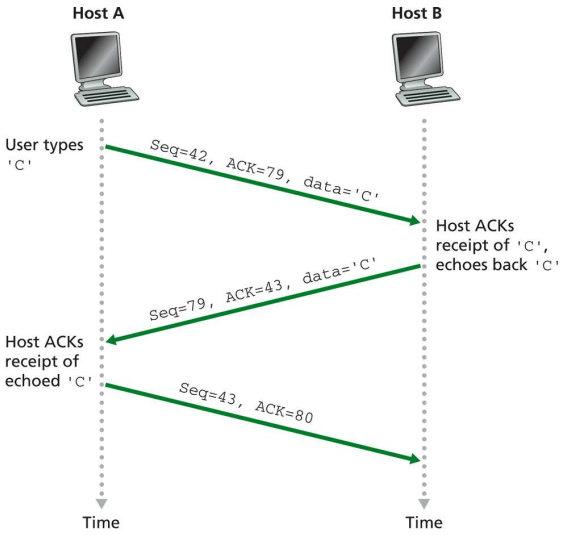


Figura 3.9: Números de sequência TCP (KUROSE; ROSS, 2006a).

3.3.4 O serviço transferência de dados garantida no TCP

Para criar o serviço de transferência de dados garantida o TCP manipula três grandes eventos relacionados à transmissão/retransmissão de dados.

1. Quando recebe dados da camada aplicação o TCP cria segmentos com números de seqüência, correspondentes aos próximos número de seqüência a serem transmitidos, e inicia um temporizador para cada segmento criado.

2. Caso o temporizador de um segmento enviado estoure o tempo (*time-out*), o TCP retransmite este segmento.

3. Caso o TCP receba um reconhecimento um segmento enviado (ou de um conjunto de segmentos), ele cancela os temporizadores remanescentes a estes segmentos; ou ainda, caso receba reconhecimentos de segmentos que já haviam sido reconhecidos (reconhecimentos cumulativos), ele retransmite os segmentos cujos números de seqüência são superiores ao reconhecimento cumulativo.

Vamos explicar como estes eventos que são tratados pelo TCP analisando alguns cenários descritos em (KUROSE; ROSS, 2006b). No primeiro cenário, mostrado na figura 3.10, o host A envia 8 bytes de dados ao host B (com número de seqüência 92). O host B envia reconhecimento dos 8 bytes recebidos (reconhecimento 100), o qual é perdido.

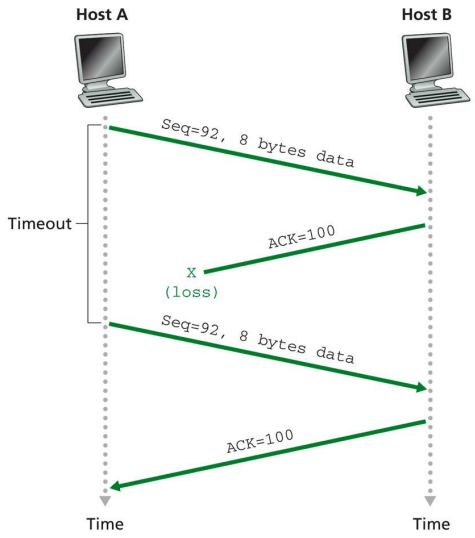


Figura 3.10: Números de sequência TCP: Cenário 1 (KUROSE; ROSS, 2006a).

Depois do estouro do temporizador do segmento 92, o mesmo é reenviado pelo host A. Quando o host B recebe o segmento duplicado, ele reenvia o reconhecimento.

No segundo cenário, mostrado na figura 3.11, o host A transmitiu ao host B um segmento com 8 bytes (número de seqüência 92) e em seguida mais um segmento com 20 bytes (número de seqüência 100). O host B recebeu estes segmentos e enviou números de reconhecimento (100 e 120 respectivamente). Todavia, o reconhecimento do segmento 92 (reconhecimento 100) chegou depois do time-out. Logo o host A retransmitiu o segmento com número de seqüência 92. Como o host B já havia recebido este segmento e também o seguinte (com número de seqüência 100), ele reenviou o reconhecimento cumulativo deste último segmento (reconhecimento 120).

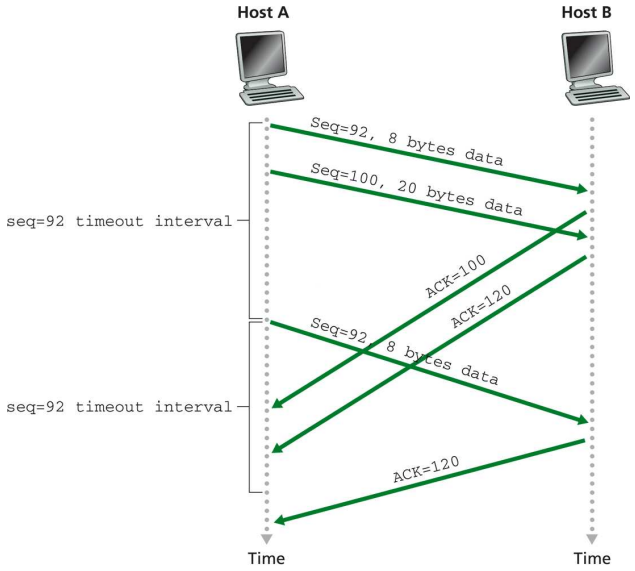


Figura 3.11: Números de sequência TCP (KUROSE; ROSS, 2006a).

No próximo cenário, mostrado na figura 3.12, como no caso anterior, o host A transmitiu ao host B um segmento com 8 bytes (número de seqüência 92) e em seguida mais um segmento com 20 bytes (número de seqüência 100). O host B recebeu estes segmentos e enviou números de reconhecimento (100 e 120 respectivamente). Todavia, o reconhecimento do segmento 92 (número de reconhecimento 100) se perdeu, o que não aconteceu com o segmento 100 (número de reconhecimento 120). Como o host A recebeu este último reconhecimento, ele sabe que o host B recebeu todos os segmentos por ele enviados, e está esperando agora segmentos com número de seqüência 120.

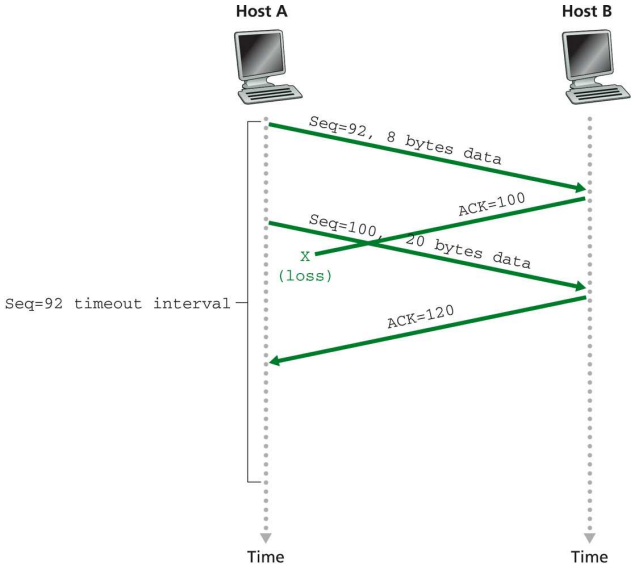


Figura 3.12: Números de sequência TCP (KUROSE; ROSS, 2006a).

3.3.5 Gerenciamento de conexões no TCP

Para trocarem segmentos de dados utilizando o TCP o emissor e o receptor devem estabelecer uma conexão TCP através da troca de pacotes de controle entre si. Isto é chamado de procedimento de estabelecimento de conexão (*handshaking*), onde se estabelecem os parâmetros para a comunicação. Uma vez concluído o *handshaking* a conexão é dita estabelecida e os dois sistemas terminais podem trocar dados.

Abertura de conexão

Na fase de estabelecimento de conexão, são inicializadas as variáveis do protocolo TCP, como os números de seqüência inicial e o tamanho de *buffers*. O processo cliente é o que inicia o estabelecimento da conexão sendo o servidor contatado pelo cliente.

O estabelecimento da conexão se dá em três passos, conforme mostrado na figura 3.13:

1. O lado cliente do TCP envia um segmento de sincronização, chamado SYN (com o *flag* Syn setado em 1), ao lado servidor do TCP, especificando um número inicial de seqüência escolhido aleatóriamente (Seq=X).

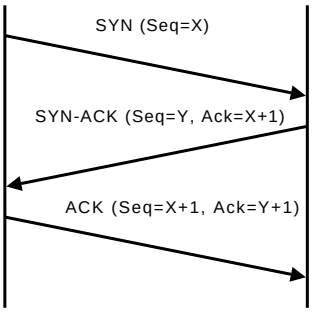


Figura 3.13: Abertura de conexão TCP.

2. O servidor recebe o SYN, aloca buffers e inicializa variáveis, e envia uma mensagem de reconhecimento da conexão, chamada SYNACK (com o *flag* Syn e *flag* Ack setados em 1), onde reconhece o pedido de conexão e especifica seu número inicial de seqüência (Seq=Y, Ack=X+1).

3. Uma vez recebido o reconhecimento da conexão pelo servidor, o cliente confirma o recebimento com um segmento chamado ACK (*flag* Syn agora em 0 e *flag* Ack setado em 1 indicando um reconhecimento válido) e também aloca buffers e inicializa variáveis da conexão (Seq=X+1, Ack=Y+1).

Uma vez que os três passos do estabelecimento da conexão forem completados, os hosts cliente e servidor podem trocar segmentos contendo dados entre eles.

Recusa de conexão

No caso de uma recusa de conexão, por exemplo quando um cliente solicita uma conexão numa porta inativa, o servidor contatado envia um segmento de controle chamado RST (*reset*) (com o *flag* Rst setado em 1), conforme ilustra a figura 3.14:

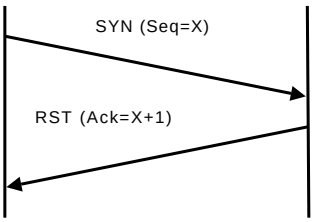


Figura 3.14: Recusa de conexão TCP.

Encerramento de conexão

Para o enceramento da conexão quatro segmentos são trocados, conforme mostrado na figura 3.15. Quem inicia a desconexão envia de um segmento especial, chamado FIN (com *flag* Fin setado em 1). Quem recebe o segmento solicitando o fim da conexão, primeiro reconhece o segmento recebido e depois envia ele também um segmento FIN (Estas duas mensagens podem estar concatenadas em uma única mensagem FIN-ACK). O encerramento definitivo da conexão se dá quando o que iniciou a desconexão recebe e reconhece o segundo segmento FIN.

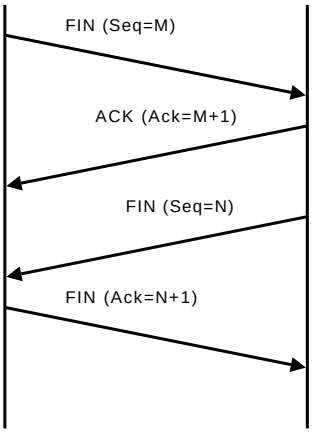


Figura 3.15: Enceramento de conexão TCP.

Comando `tcpdump`

O comando `tcpdump` permite colocar uma interface de rede em modo "promíscuo", capturando cada pacote circulando no meio físico junto a interface. A partir dos pacotes capturados é possível analizar pacotes trocados durante uma conexão TCP.

O exemplo abaixo mostra uma saída do comando `tcpdump`, no qual estão listados quatro segmentos TCP trocados entre um cliente e um servidor Web. Cada linha da captura mostra na sequência: IP.porta origem > IP.porta destino: flags setados, número de sequência inicial:final (número de bytes do segmento), janela do receptor, etc. O primeiro segmento é a solicitação de abertura de conexão (SYN), com flag S setado e número de sequência inicial 2035538770 (X). O segundo segmento é a resposta a solicitação do cliente (SYN-ACK) com flag S setado e número de sequência inicial 2981782888 (Y) e número de reconhecimento 2035538771 (X + 1). O terceiro segmento finaliza a abertura de conexão (ACK), com reconhecimento 2981782889 (Y + 1). O quarto segmento é uma requisição do cliente ao servidor, com flag P setado, com 711 Bytes de dados.

```
root@Campeche:/home/evandro# tcpdump -ntS tcp and port 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
IP 192.168.0.10.33643 > 200.18.10.1.80: S 2035538770:2035538770(0)
 win 5840 <mss 1460,sackOK,timestamp 4007676 0,nop,wscale 7>
IP 200.18.10.1.80 > 192.168.0.10.33643: S 2981782888:2981782888(0) ack 2035538771
 win 5792 <mss 1412,sackOK,timestamp 93122504 4007676,nop,wscale 6>
IP 192.168.0.10.33643 > 200.18.10.1.80: . ack 2981782889
 win 46 <nop,nop,timestamp 4007685 93122504>
IP 192.168.0.10.33643 > 200.18.10.1.80: P 2035538771:2035539482(711) ack 2981782889
 win 46 <nop,nop,timestamp 4007685 93122504>
...
```

Máquina de Estados do TCP

A figura 3.16 ilustra o diagrama de estados possíveis que o protocolo TCP pode assumir durante o processo e abertura e encerramento de conexões.

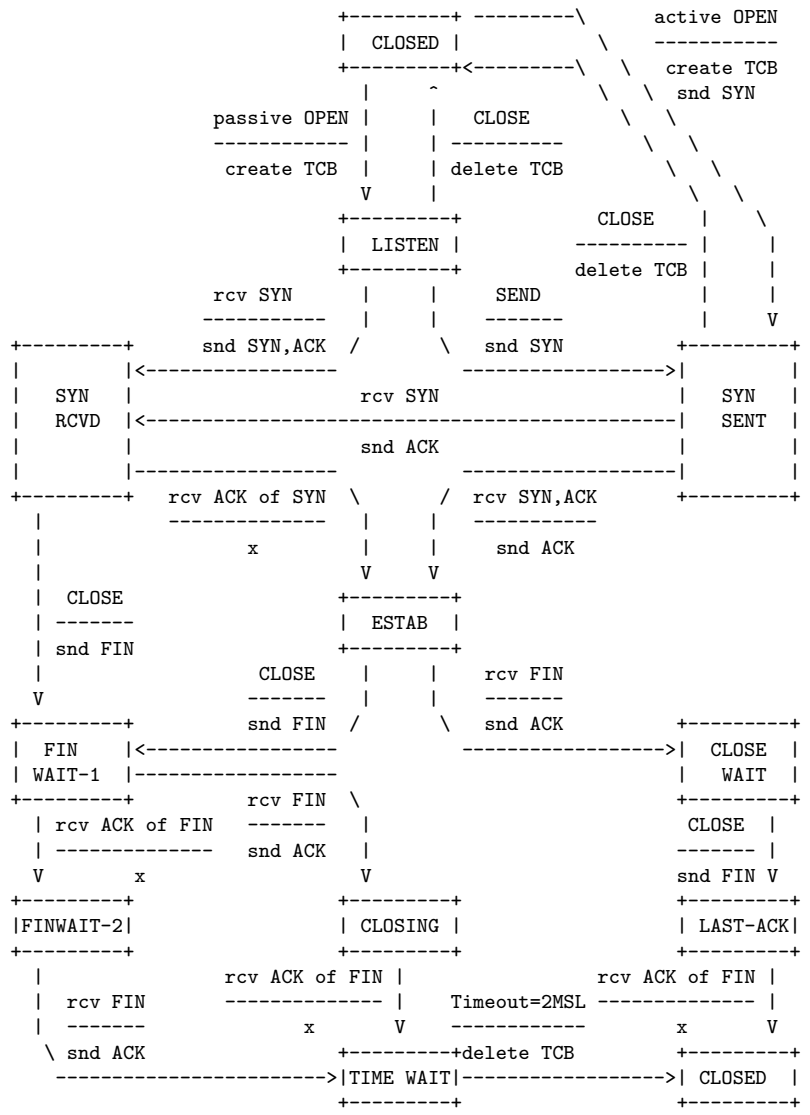


Figura 3.16: Diagrama de estados das conexões TCP (RFC793).

Durante o processo de estabelecimento de conexão o TCP passa por uma série de estados. Os estados são: LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, e o estado CLOSED. CLOSED é um estado fictício, porque representa um estado onde não há TCB (*Transmission Control Block*) (Bloco de controle de transmissão), pois não há mais conexão.

O significado dos estados são:

- LISTEN - representa a espera de uma conexão de algum TCP remoto e porta.

- SYN-SENT - representa a espera de uma confirmação de conexão após o envio de uma solicitação de conexão.
- SYN-RECEIVED - representa a espera de uma confirmação de conexão após ter recebido uma solicitação de conexão e enviado uma confirmação .
- ESTABLISHED - representa uma conexão aberta, na qual dados recebidos podem ser entregues ao usuário.
- FIN-WAIT-1 - representa a espera de um encerramento de conexão após o envio de uma solicitação de encerramento.
- FIN-WAIT-2 - representa a espera de uma solicitação de encerramento de conexão de um TCP remoto.
- CLOSE-WAIT - representa a espera de uma solicitação de encerramento de conexão do usuário local.
- CLOSING - representa a espera de uma confirmação de encerramento de conexão de um TCP remoto.
- LAST-ACK - representa a espera de uma confirmação de encerramento de conexão previamente enviada ao TCP remoto (a qual inclui um reconhecimento de sua solicitação de encerramento de conexão).
- TIME-WAIT - representa a espera de um tempo suficiente para garantir que o TCP remoto receba a confirmação de encerramento de conexão.
- CLOSED - representa a coenxão fechada.

Comando `netstat`

O comando `netstat`, com o parâmetro `-t`, pode ser utilizado para visualizar o estado das conexões TCP ativas em um host, como mostrado no exemplo a seguir:

```
evandro@Campeche:~$ netstat -t
Conexões Internet Ativas (sem os servidores)
Proto Recv-Q Send-Q Endereço Local          Endereço Remoto         Estado
tcp        0      0 Campeche.local:54203    helmet.uol.com.br:www   ESTABELECIDA
tcp        0      0 Campeche.local:36536    i.uol.com.br:www        ESTABELECIDA
tcp        0      0 Campeche.local:38409    200-98-211-131.inte:www ESTABELECIDA
tcp        0      0 Campeche.local:48448    200.221.7.75:www        TIME_WAIT
tcp        0      1 Campeche.local:54565    200.221.6.19:www        ÚLTIMO_ACK
tcp        0      0 Campeche.local:38408    200-98-211-131.inte:www ESTABELECIDA
tcp        1      0 Campeche.local:38408    200-98-211-131.inte:www ESPERANDO_FECHAR
```

O comando `netstat` com o parâmetro `-n` imprime endereços IP e portas em formato numérico.

3.3.6 Controle de fluxo

O **controle de fluxo** no TCP visa evitar que um emissor não "afogue" um receptor, enviando mais dados que ele possa processar. Quando uma conexão TCP recebe bytes que estão corretos e em sequência, ela os armazena em um *buffer* de recepção, antes de entregá-los a aplicação. Como o processo de aplicação pode não ler estes dados imediatamente, pode ocorrer um estouro da capacidade do *buffer*. Assim, o mecanismo de controle de fluxo implementa um tipo de casamento de velocidades entre o emissor e o receptor.

O controle de fluxo é implementado por meio de uma variável, chamada **janela do receptor** (*receive window*), que mantém o valor atual da capacidade de armazenamento do *buffer* de recepção. Esta janela limita a quantidade de bytes que pode ser enviada antes de esperar por um reconhecimento. O tamanho desta janela é dinâmico, podendo variar durante uma conexão. Como a conexão é *full-duplex*, cada um dos receptores mantém a informação sobre sua janela de recepção.

A figura 3.17 ilustra o mecanismo de controle de fluxo. Novos dados chegando da camada IP ocupam o espaço livre do buffer. Dados consumidos pela aplicação liberam espaço do buffer.

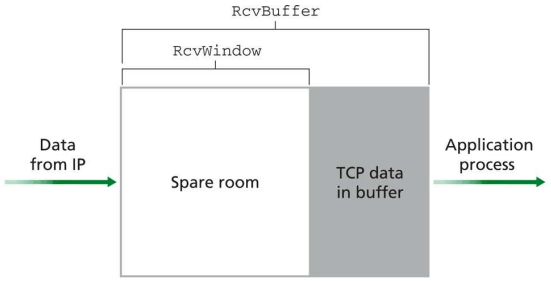


Figura 3.17: Janela do receptor (*RcvWindow*) e *buffer* de recepção (*RcvBuffer*) do mecanismo de controle de fluxo do TCP (KUROSE; ROSS, 2006a).

3.3.7 Temporização no TCP

A cada segmento enviado pelo TCP ele inicializa um **temporizador** visando aguardar um reconhecimento em um tempo limite. Caso o temporizador se esgote (*timeout*) o TCP retransmite o segmento.

O valor do temporizador deve ser maior que o tempo de ida e volta dos pacotes. Contudo, não pode ser muito pequeno, pois pode gerar estouros desnecessários do temporizador, nem muito longo, pois aumenta o tempo de reação a possíveis erros.

Para definir o valor do temporizador o TCP faz uma estimativa dinâmica do tempo de ida e volta dos pacotes, chamado RTT (*roud trip time*). A partir desta estimativa, calcula o valor médio e desvio, e aplica uma margem de segurança para definir o valor do temporizador.

A figura 3.18 ilustra o processo de estimativa do tempo de ida e volta (RTT) dos pacotes no TCP .

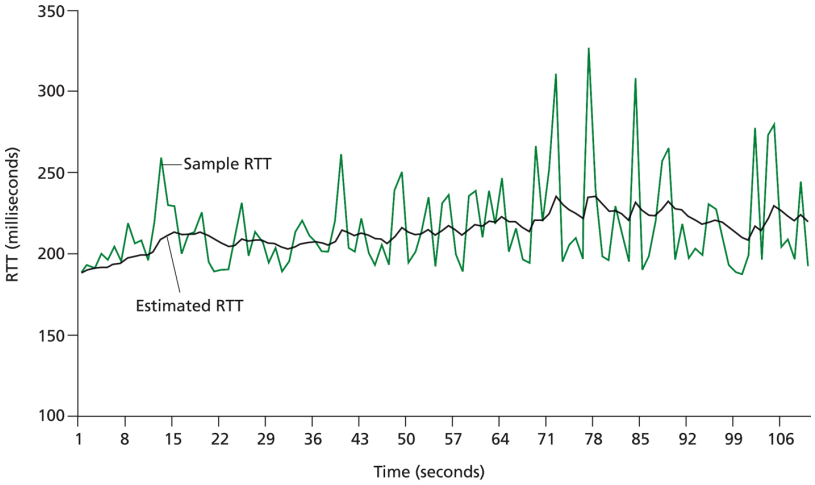


Figura 3.18: Estimativa do tempo de ida e volta (RTT) no TCP (KUROSE; ROSS, 2006a).

3.3.8 Controle de congestionamento no TCP

O controle de congestionamento visa minimizar o congestionamento no núcleo da rede.

O TCP implementa um **mecanismo de controle de congestionamento fim a fim** inferido a partir da observação dos pacotes perdidos ou recebidos com atraso. A idéia é relativamente simples. Caso o TCP detecte perda de pacotes ele supõe que há congestionamento da rede. A partir dai o TCP diminui o fluxo de pacotes enviado, visando colaborar com a diminuição do possível congestionamento no núcleo da rede. Lembre que a perda de pacotes ocorre principalmente quando estoura a capacidade de armazenamento nos roteadores devido ao congestionamento e os novos pacotes que chegam são descartados (*dropped*).

O TCP controla sua taxa de transmissão limitando o número de segmentos transmitidos e ainda não reconhecidos. Este controle depende do tamanho da **janela** do receptor, como vimos na seção 3.3.6 no estudo do controle de fluxo. Idealmente, deve-se permitir que o TCP transmita o mais rápido possível, desde que não perca segmentos. Para isto, a conexão TCP começa com uma janela de tamanho pequeno, visando sondar a existência de banda, e vai aumentando até que ocorra uma perda. Quando isto acontece, o TCP reduz o tamanho da janela para um nível seguro, e volta a aumentá-la, sondando novamente a existência de banda.

Uma das medidas de performance do TCP é a vazão (*throughput*) na qual o emissor transmite dados ao receptor. Esta taxa depende do tamanho da **janela** do receptor. Se o TCP transmite um conjunto de segmentos, totalizando a quantidade de bytes disponíveis na janela, então ele deve esperar um tempo de ida e volta (RTT) até receber um reconhecimento, a partir dai ele pode voltar a enviar dados. Se uma conexão transmite uma quantidade de segmentos do tamanho da janela do receptor a cada RTT, então a taxa será:

$Vazão = \frac{Janela}{RTT}$ (KUROSE; ROSS, 2006b).

Vimos que para o **controle de fluxo**, o fluxo de pacotes é controlado pela variável

janela do receptor (*RevWindow*), a qual é informada constantemente pelo receptor ao remetente por meio do campo `window`, presente no cabeçalho do segmento TCP (ver figura 3.7). Para o **controle de congestionamento** é utilizado duas variáveis a mais: a **janela de congestionamento** (*CongWin*) e o **limiar**. A janela de congestionamento vai impor uma limitação adicional a quantidade de tráfego que um emissor pode gerar. Especificamente a quantidade de dados enviados e não reconhecidos dentro de uma conexão TCP não deve exceder o mínimo de *CongWin* e *RevWindow*.

Vamos examinar como a **janela de congestionamento** (*CongWin*) evolui durante uma conexão TCP. Assim que a conexão é estabelecida, o processo de aplicação remetente escreve bytes no *buffer* de envio do TCP emissor. O TCP agrupa os bytes em segmentos de tamanho MSS e envia para a camada rede para transmissão. Inicialmente a janela de congestionamento é igual a MSS. Isto significa que o TCP envia um segmento e espera por um reconhecimento. Se este segmento foi reconhecido antes do esgotamento do temporizador, o TCP dobra o tamanho da janela de congestionamento, passando a enviar dois segmentos. Se receber reconhecimentos novamente, volta a dobrar o tamanho da janela para quatro. O TCP continua este procedimento enquanto: a) a janela de congestionamento estiver abaixo do valor de limiar; b) os reconhecimentos chegarem antes do esgotamento do temporizador.

Esta fase do procecimento de controle de congestionamento é chamada de **partida lenta**, na qual a janela inicia com o valor de MSS e aumenta de forma exponencial.

A fase de partida lenta termina quando o tamanho da janela atinge o valor de limiar (*threshold*). A partir deste momento a janela continua a crescer linearmente, um MSS para cada RTT. Esta fase é chamada de **prevenção de congestionamento**.

Se houver um **esgotamento de temporização**, o valor do limiar é ajustado para a metade do valor corrente da janela, e a janela é ajustada para um MSS, reiniciando o processo de partida lenta.

Como refinamento deste mecanismo, o TCP cancela o processo de partida lenta caso receba **três reconhecimentos duplicados** antes de um estouro de temporizador. Neste caso, ele reduz o tamanho da janela para a metade de seu valor corrente e volta a crescer linearmente. Este algoritmo é chamado de **aumento aditivo, dimuinuição multiplicativa** (AIMD – *additive-increase, multiplicative-decrease*).

A filosofia do AIMD é que três reconhecimentos duplicados indica que a rede é capaz de entregar alguns segmentos. Entretanto, um estouro de temporizador antes de três reconhecimentos duplicados é mais "alarmante".

A evolução do mecanismo de controle de congestionamento TCP é ilustrada na figura 3.19.

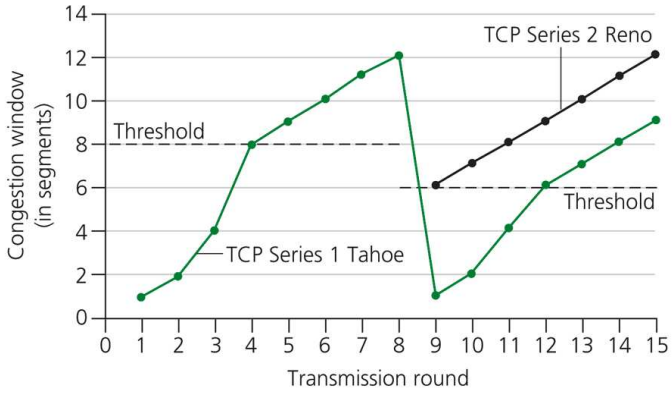


Figura 3.19: Evolução da janela de congestionamento TCP (KUROSE; ROSS, 2006a).

Capítulo 4

Camada Rede

A camada de transporte provê um canal lógico processo a processo para as aplicações rodando em diferentes hosts. Para prover este serviço, a camada de transporte usa a camada rede, a qual provê um serviço de comunicação de computador a computador na rede.

Papéis da camada rede:

- Determinação da rota que tomarão os datagramas desde o computador origem até o destino, a partir de algorítmos de roteamento.
- Chaveamento dos datagramas chegando nos enlaces de entrada de cada roteador para a saída apropriada.

4.1 Protocolo IP (*Internet protocol*)

Na Internet a camada rede é implementada pelo **protocolo IP**, o qual oferece um serviço de datagramas, onde cada datagrama é tratado como uma unidade independente e não recebe nenhum tratamento de erros ou reconhecimento fim a fim. O protocolo IP em uso atualmente na Internet é o IPv4, definido na RFC791. Outra versão do protocolo, conhecida como IPv6, será apresentada na seção 4.7.

Quando a camada rede do lado de um emissor recebe um segmento da camada de transporte ela o encapsula em um datagrama IP, escreve o endereço do destino e outros campos do cabeçalho e envia ao primeiro roteador em direção ao computador destino. Para que o datagrama atinga o destino, a camada rede envolve cada computador e cada roteador no caminho entre a origem e o destino dos segmentos.

Os três principais componentes da camada rede da Internet são:

- **Protocolo IP**, que provê uma forma de endereçamento, formato do datagrama e convenções de empacotamento.
- **Protocolos de roteamento**, que permitem a determinação de rotas e elaboração de tabelas de roteamento. Os protocolos de roteamento mais conhecidos são o RIP, o OSPF e o BGP.

- **Protocolo ICMP**, utilizado para reportagem de erros e sinalização entre os roteadores e entre os computadores.

A figura 4.1 ilustra, na forma de um mapa conceitual, uma visão dos principais conceitos da camada rede da Internet, mostrando explicitamente a relação entre os mesmos.

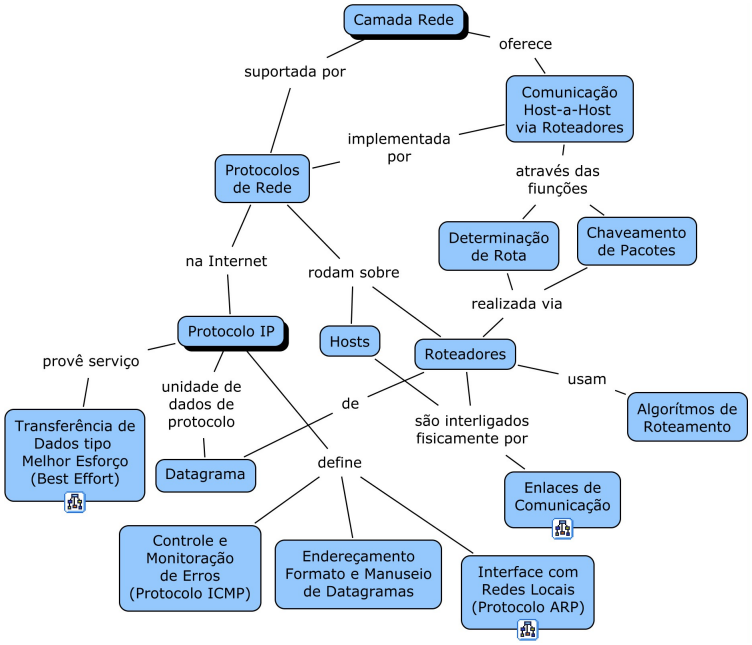


Figura 4.1: Visão dos principais conceitos da camada rede da Internet.

4.1.1 Datagrama IP

Um datagrama IP é a unidade básica de transferência na Internet. O formato do datagrama apresenta um cabeçalho, que contém os endereços IP da fonte (*Source Address*) e do destino (*Destination Address*), além de outros campos, e uma área de dados.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figura 4.2: Formato do datagrama IP (RFC791).

- O campo versão (*Version*) indica a versão do protocolo, no caso a versão 4, também conhecida como IPv4.

- O campo IHL (*internet header length*) indica o comprimento do cabeçalho, em função dos campos opcionais, tipicamente o cabeçalho contém as 5 primeiras linhas, totalizando 20 bytes..

- O tipo de serviço (*Type of Service*) permite diferenciar datagramas de diferentes tipos de serviço, sendo utilizado para implementar mecanismos de QoS (Qualidade de Serviço).

- O campo *Total Length* indica o comprimento total do datagrama em bytes.

- Os parâmetros identificação (Identification), *flags* e fragmentação (*Fragment Offset*) são usados em caso de fragmentação do datagrama IP.

- O tempo de sobrevida, TTL (*Time to Live*), indica o tempo de vida do datagrama, após o qual o mesmo é descartado. Na prática é medido em número de saltos (*hops*), sendo decrementado de uma unidade em cada roteador.

- O protocolo (*Protocol*) indica o protocolo da camada superior que está sendo transportado, como por exemplo TCP ou UDP. Para cada protocolo há um número padronizado pela RFC1700, como por exemplo:

```
Assigned Internet Protocol Numbers
Decimal    Keyword     Protocol
     1     ICMP        Internet Control Message
     2     IGMP        Internet Group Management
     4     IPv4        IP in IP (encasulation)
     6     TCP         Transmission Control
    17     UDP         User Datagram
```

- O *Header Checksum* é utilizado para detecção de erros no cabeçalho.

- O campo de opções (*Options*) é raramente usado.

- Os dados sucedem o cabeçalho, podendo ser, por exemplo, segmentos TCP ou UDP, mensagens do protocolo ICMP, mensagens de roteamento, etc.

O comprimento total do datagrama teoricamente poderia ser de 64K bytes (em função dos 16 bits do campo: $2^{16} = 65535$), todavia, na prática, nunca é maior que 1.500 bytes e freqüentemente é limitado em 576 bytes. Isto é feito para evitar a fragmentação do datagrama na rede física, já que o mesmo é encapsulado em um quadro da camada enlace e nem todas tem quadros de mesmo tamanho. No caso das redes locais Ethernet o tamanho do quadro é de 1.500 bytes e em outros enlaces é de 576 bytes. O tamanho máximo dos pacotes que podem ser transportados pela camada enlace é chamado de MTU (*maximum transfer unit*), como ilustrado na figura 4.3:

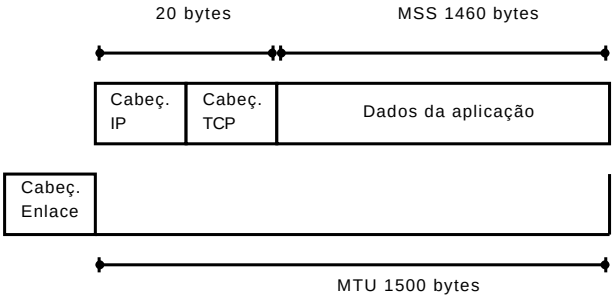


Figura 4.3: Valor típico de MSS e MTU.

4.1.2 Fragmentação do IP

A fragmentação de um datagrama IP pode ocorrer quando um datagrama chega em um enlace de entrada de um roteador e precisa ser enviado através de um enlace de saída cuja MTU (*maximum transfer unit*) não permite acomodar todos os bytes do datagrama. Neste caso, o datagrama deverá ser fragmentado em dois ou mais fragmentos (datagramas de menor tamanho) antes de ser enviado através do enlace de saída.

Os fragmentos precisam ser remontados antes de serem entregues a camada transporte no destino, uma vez que tanto o TCP quanto o UDP aguardam um datagrama completo. A solução adotada pelos projetistas do IPv4 foi fragmentar os datagramas nos roteadores intermediários e somente remontá-los no destino final. Para permitir a remontagem do datagrama são utilizados três campos do cabeçalho IP: **identificação** (*Identification*), ***flags*** e **fragmentação** (*Fragment Offset*). Quando um datagrama é criado o emissor o identifica com um número de identificação (*Identification*), assim como um endereço IP de origem e destino. Para cada novo datagrama criado o emissor gera uma nova identificação incrementando o valor deste campo. Quando um datagrama é fragmentado cada fragmento recebe o número de identificação, endereço IP de origem e destino iguais aos do datagrama original. Uma vez fragmentado o datagrama, cada fragmento será transmitido de forma independente, podendo inclusive chegarem no destino fora de ordem. Para a remontagem, a fim de identificar a posição de cada fragmento dentro do datagrama original, o campo *Fragment Offset* é utilizado. O último fragmento é reconhecido através do *flag*, que é definido como 0 (zero), ficando para os demais fragmentos o *flag* definido como 1 (um).

A figura 4.4 ilustra um exemplo, apresentado por Kurose e Ross (2006a), no qual um datagrama de 4000 bytes (20 bytes de cabeçalho IP mais 3980 bytes de dados) chega a um roteador e deve ser reenviado por um enlace com MTU de 1500 bytes. Para isto, os 3980 bytes de dados devem ser alocados em três fragmentos, todos identificados com o mesmo número de identificação.

A tabela abaixo ilustra os valores dos campos *Identification*, *Offset* e *Flag* para os três fragmentos. O terceiro fragmento carrega somente 1020 bytes de dados (=3980-1480-1480). O campo *Offset* do primeiro fragmento é 0, significando que os dados devem ser inseridos a partir da posição 0 byte do datagrama a ser remontado. Para o segundo fragmento o *Offset* é 185, significando deve ser inserido na posição 1480 bytes (note que 185 . 8=1480). Para o terceiro fragmento o *Offset* é 370, significando deve ser inserido na posição 2960 bytes (note que 370 . 8 =2960).

Fragmento	Bytes	*Identification*	*Offset*	*Flag*
1o	1480	777	0	1
2o	1480	777	185	1
3o	1020	777	370	0

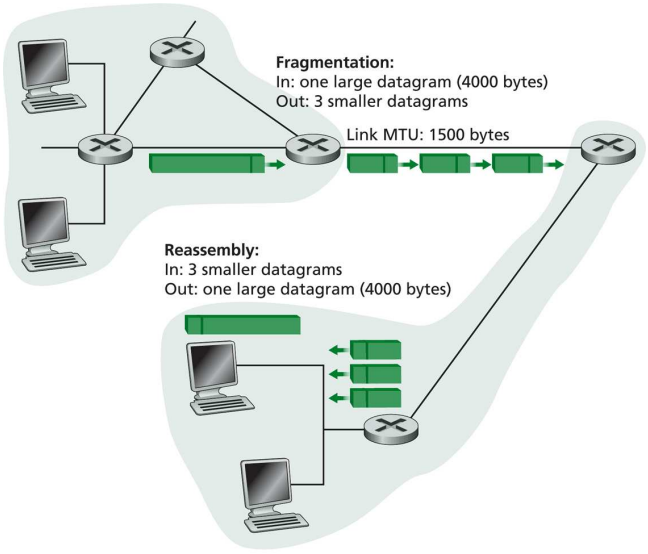


Figura 4.4: Fragmentação do datagrama IP (KUROSE; ROSS, 2006a).

4.1.3 Protocolo ICMP

Conforme já mencionado, o protocolo IP fornece um serviço de datagramas não confiável e não orientado a conexão, onde um datagrama segue de roteador em roteador até alcançar seu destino final. Se um roteador não consegue encontrar uma rota ou entregar um datagrama, ou se uma condição anormal é detectada, o roteador precisa informar a fonte original dos dados para que esta tome alguma ação ou corrija o problema. O **protocolo ICMP** (*Internet Control amd Message Protocol*), definido na RFC792, permite que os roteadores enviem mensagens de erro e controle a outros roteadores ou hosts; oferecendo uma comunicação entre a camada rede de uma máquina e a camada rede de outra máquina.

Tecnicamente o ICMP é um mecanismo de reportagem de erros. Ou seja, quando um datagrama causa um erro, o ICMP pode reportar a condição de erro de volta a fonte original do datagrama; a fonte então relata o erro para a aplicação ou realiza uma ação com vistas a corrigir o erro. Por exemplo, quando rodando uma aplicação Telnet ou HTTP, podemos encontrar mensagens como *Destination Network Unreachable* (rede destino não encontrada), que tem origem no protocolo ICMP.

O ICMP é normalmente considerado como parte do IP, todavia as mensagens ICMP são carregadas na porção de dados de um datagrama IP, que as identifica como tipo ICMP (*Type* 1). Os datagramas contendo as mensagens ICMP seguem de volta, seguindo exatamente o caminho que tomaram os dados do usuário, podendo elas também serem perdidas ou corrompidas.

Formato das Mensagens ICMP

Cada mensagem ICMP tem um campo de tipo (*Type*) e um campo de código (*Code*), e também contém os primeiros 8 bytes do datagrama que causou o erro (com isto o emissor pode determinar o pacote que causou o erro). Por exemplo, a mensagem ICMP Destino Inalcançável (Fig. 4.5).

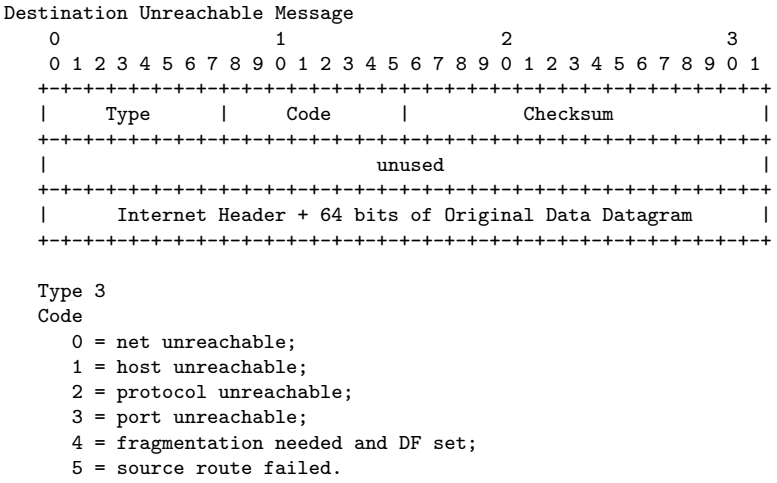


Figura 4.5: Mensages ICMP Destino Inalcançável (RFC792).

Na sequência estão listados alguns tipos de mensagens ICMP.

```
Summary of Message Types
   0  Echo Reply
   3  Destination Unreachable
   4  Source Quench
   5  Redirect
   8  Echo
  11  Time Exceeded
  12  Parameter Problem
  13  Timestamp
  14  Timestamp Reply
  15  Information Request
  16  Information Reply
```

`Ping`

Nem todas as mensagens ICMP são de reportagem de erros. A aplicação `ping`, por exemplo, envia mensagens ICMP *Echo Request* a um host e espera como retorno mensagens *Echo Reply* para verificar se o host está disponível

A origem do nome "ping" vem do sonar para localizar objetos (STEVENS, 2008) e na Internet serve para testar se um host é alcançável pela rede IP.

A figura 4.6 mostra o formato da mensagem ICMP de requisição e resposta de eco.

São identificadas com código (0) e tipo (8 – *echo request*; 0 – *echo* reply). O campo identificador (*identifier*) permite ao `ping` identificar se a resposta recebida quando há múltiplas

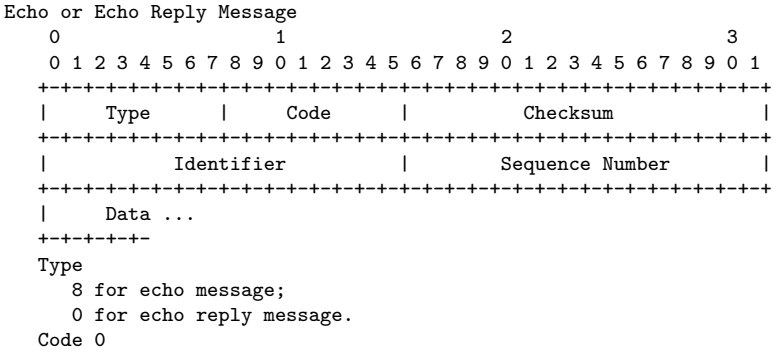


Figura 4.6: Mensages ICMP Requisição e Resposta de Eco (RFC792).

instâncias do `ping` rodando. O campo número de sequência (*sequence number*) inicia com 0 e é incrementado a cada nova requisição de eco enviada. O *ping* imprime os números de sequência de cada pacote retornado, permitindo ao usuário acompanhar quais pacotes foram perdidos, reordenados ou ducplicados.

A seguir está o exemplo de uma saída típica do `ping`:

```
evandro@Campeche:~$ ping 200.135.37.65
PING 200.135.37.65 (200.135.37.65) 56(84) bytes of data.
64 bytes from 200.135.37.65: icmp_seq=1 ttl=54 time=22.1 ms
64 bytes from 200.135.37.65: icmp_seq=2 ttl=54 time=20.3 ms
64 bytes from 200.135.37.65: icmp_seq=3 ttl=54 time=21.8 ms
64 bytes from 200.135.37.65: icmp_seq=4 ttl=54 time=21.0 ms
64 bytes from 200.135.37.65: icmp_seq=5 ttl=54 time=20.1 ms

--- 200.135.37.65 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 20.156/21.105/22.100/0.799 ms
```

A saída do `ping` mostra o retorno de cada datagrama enviado, identificado pelo número de sequência. Outras informações são o tempo de ida e volta do datagrama e o campo TTL (*time to live*) do datagrama IP. O valor inicial recomendado para o TTL é 64 (servidores Linux usam este valor). Outros sistemas inicializam com outros valores (servidores Windows usam o valor máximo 255). No exemplo acima, portanto, pode-se concluir que o datagrama com a resposta de eco passou por 10 roteadores intermediários.

`Traceroute`

Outra aplicação que utiliza mensagens ICMP é o `traceroute`.

O `traceroute`, que é capaz de traçar a rota que liga um computador a outro computador. Para determinar o nome e o endereço dos roteadores entre a fonte e o destino, o `traceroute` na fonte envia uma série de datagrama IP ordinários ao destino. O primeiro datagrama tem o campo TTL (*time to live*) igual a 1, o segundo 2, o terceiro 3, e assim por diante, e inicia temporizadores para cada datagrama. Quando o enésimo datagrama chega ao enésimo roteador, este verifica que o tempo de sobrevida do datagrama acaba de terminar. Pelas regras do IP, o datagrama é então descartado e uma mensagem ICMP de advertência **tempo excedido** (*Time*

Exceeded) é enviada a fonte (tipo 11 código 0), com o nome do roteador e seu endereço IP. Quando a resposta chega de volta a fonte, a mesma calcula o tempo de viagem em função dos temporizadores.

O `traceroute` envia datagramas IP encapsulados em segmentos **UDP** a um host destino. Todavia escolhe um número de `porta destino` com um valor desconhecido (maior que 30000), tornando improvável que o host destino esteja usando esta porta. Quando o datagrama chega ao destino uma mensagem ICMP **porta inalcançável** (*port enreachable*) (tipo 3 código 3) é gerada e enviada a origem. O programa `traceroute` precisa saber diferenciar as mensagens ICMP recebidas – tempo excedido e porta inalcançável – para saber quando a rota foi concluída.

A seguir está o exemplo de uma saída típica do `traceroute`:

4.2 Endereçamento IP

Endereço IP é um endereço lógico de 32 bits, escrito em quatro octetos representados em decimal, cada um variando de O a 255. Os números são separados por pontos. Por exemplo, **193.32.216.9** é um endereço válido e sua notação em binário é:

`11000001 00100000 11011000 00001001`.

Cada computador que esteja rodando o TCP/IP exige um endereço IP exclusivo.

Cada **endereço IP** engloba duas partes: o **identificador da rede** e o **identificador do computador**. O identificador da rede identifica a rede onde se encontram todos os computadores da mesma rede local. O identificador do computador identifica um dispositivo em uma rede local, como um computador ou roteador. Por exemplo, a figura 4.7 ilustra três redes locais interconectadas por um roteador com três interfaces. Olhando para os endereços IP atribuídos a cada computador e a cada interface do roteador, podemos notar, por exemplo, que os dispositivos conectados a rede local da direita e acima tem os endereços IP da forma 200.1.1.X. Isto é, compartilham os 24 bits mais à esquerda do endereço IP. No jargão IP, esta parte do endereço forma o identificador da rede. Os 8 bits restantes permitem identificar cada computador da rede local. O endereço da rede local seria 200.1.1.0/24, onde a notação /24 é também conhecida como **máscara de rede**, e indica que os 24 bits mais à esquerda dos 32 bits do IP identificam a rede.

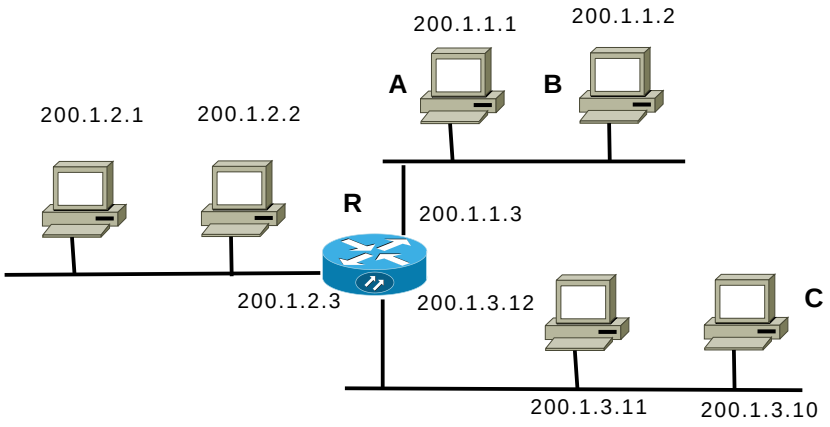


Figura 4.7: Endereçamento IP e sub-redes.

O valor da **máscara de rede** pode também ser representado em decimal com pontos. Por exemplo, a máscara /24 pode ser representada como 255.255.255.0, o que significa que os 24 primeiros bits são um e os 8 restantes zero.

A máscara de rede é utilizada para mascarar uma parte do endereço IP para que se possa distinguir o identificador da rede do identificador do computador. Para se extrair o identificador da rede a partir do endereço IP completo uma operação lógica AND é realizada com a máscara de rede.

Por exemplo, para descobrir o identificador de rede do computador cujo endereço IP é 200.135.233.4 e cuja máscara de rede é 255.255.255.0, devemos fazer uma operação AND desdes dois valores:

```
    11001000 10000111 11101001 00000100
    11111111 11111111 11111111 00000000
AND _____________________________________
    11001000 10000111 11101001 00000000
```

que resulta no endereço de rede 200.135.233.0.

4.2.1 Classes de endereçamento de IP

Para garantir endereços exclusivos em âmbito mundial, os endereços IP são licenciados a partir do organismo internacional chamado IANA (*Internet Assigned Numbers Authority* – `www.iana.org`).

Quando o protocolo IP foi criado previa três classes de endereçamento: Classe A (bits mais significativos do endereço `00`), com 128 redes possíveis e 16777216 sistemas finais (menos os valores especiais "tudo zero", reservados para o endereçamento da rede, e "tudo um", reservado para *broadcast*); Classe B (MSB `10`), com 16384 redes possíveis e 65536 sistemas finais (menos os dois valores reservados); e a Classe C (MSB `110`) com 2097152 redes possíveis, cada qual com 254 sistemas finais (256 combinações de bit combinations menos os valores reservados "tudo zero" e "tudo um"). O conjunto de endereços com MSB `1110` ficou reservado para *multicast* e o restante para uso futuro. A figura 4.8 ilustra as classes de endereçamento IP.

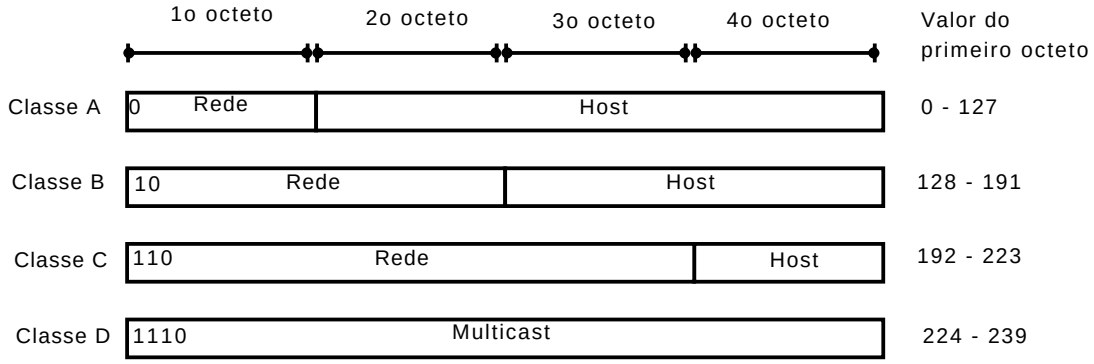


Figura 4.8: Classes de endereçamento IP.

Estas classes de endereçamento não são mais utilizadas como parte formal dos da arquitetura de endereçamento IP, pois, com o crescimento do número de organizações de pequeno e médio porte o espaço de endereçamento ficou limitado. Por exemplo, uma rede classe C pode acomodar

até 2^8 endereços, ou seja 256 hosts, o que pode ser muito pouco para muitas organizações. Já uma classe B, poderia acomodar 2^{16} endereços, ou 64.634 endereços, o que seria demais para uma organização com, por exemplo, 2000 computadores.

O endereçamento CIDR (*classless interdomain routing*) (RFC4632), apresenta estratégias para atribuições de endereços IPv4 de 32 bits, tendo em vista a preservação do espaço de endereçamento e a limitação do crescimento das tabelas de roteamento.

Na notação CIDR um prefixo é representado por de 4-octetos, como um endereço tradicional IPv4 ou um endereço de rede, seguido pelo caracter / (barra) e um decimal entre 0 e 32, o qual descreve um número significativo de bits.

Por exemplo, a rede Classe B de número 172.16.0.0 é definida no CIDR com o prefixo 172.16.0.0/16, na qual o /16 indica que a máscara de rede utilizada para extrair a porção da rede do número de 32 bits.

A tabela a seguir (RFC4632), mostra um resumo de todos os prefixos CIDR possíveis.

notation	addrs/block	# blocks	
n.n.n.n/32	1	4294967296	host route
n.n.n.x/31	2	2147483648	p2p link
n.n.n.x/30	4	1073741824	
n.n.n.x/29	8	536870912	
n.n.n.x/28	16	268435456	
n.n.n.x/27	32	134217728	
n.n.n.x/26	64	67108864	
n.n.n.x/25	128	33554432	
n.n.n.0/24	256	16777216	legacy Class C
n.n.x.0/23	512	8388608	
n.n.x.0/22	1024	4194304	
n.n.x.0/21	2048	2097152	
n.n.x.0/20	4096	1048576	
n.n.x.0/19	8192	524288	
n.n.x.0/18	16384	262144	
n.n.x.0/17	32768	131072	
n.n.0.0/16	65536	65536	legacy Class B
n.x.0.0/15	131072	32768	
n.x.0.0/14	262144	16384	
n.x.0.0/13	524288	8192	
n.x.0.0/12	1048576	4096	
n.x.0.0/11	2097152	2048	
n.x.0.0/10	4194304	1024	
n.x.0.0/9	8388608	512	
n.0.0.0/8	16777216	256	legacy Class A
x.0.0.0/7	33554432	128	
x.0.0.0/6	67108864	64	
x.0.0.0/5	134217728	32	
x.0.0.0/4	268435456	16	
x.0.0.0/3	536870912	8	
x.0.0.0/2	1073741824	4	
x.0.0.0/1	2147483648	2	
0.0.0.0/0	4294967296	1	default route

n é um decimal de 8-bit.
x é um valor de 1 a 7 bits, dependendo do comprimento do prefixo.

Um enlace ponto a ponto (`p2p link`) é definido na RFC3021, permitindo que um enlace ponto-a-ponto, entre dois roteadores por exemplo, possa utilizar apenas dois endereços IP, já que não tem sentido falar em endereço de rede ou *broadcast* em um enlace deste tipo.

Endereços IP especiais

Alguns endereços IP têm utilização especial:

- 0.0.0.0 – Endereço com todos os 32 bits iguais a 0: Pode aparecer em tabelas de roteamento identificando o destino como a prória rede e também identificar rotas default;
- 255.255.255.255 – Endereço com todos os 32 bits iguais a 1: É considerado um endereço de difusão limitado a rede do host origem do datagrama (não encaminhado);
- 127.0.0.0 – Endereço reservado para teste de (*loopback*) e comunicação entre processos da mesma máquina.
- Os endereços com o primeiro octeto entre 240 e 255 são reservados para uso futuro.

4.2.2 Divisão em sub-redes

Além de poder solicitar um bloco de endereços IP de qualquer tamanho, com o CIDR uma organização pode ainda dividir seu bloco de endereço em sub-redes, criando suas próprias redes internas.

Por exemplo, imagine que uma organização tenha recebido o bloco de endereços 200.23.16.0/20. A máscara de rede /20 indica que os 20 primeiros bits do endereço identificam a rede, sobrando, portando, 12 bits para identificar hosts dentro desta rede. Assim, com este bloco de endereços, esta organização pode endereçar 2^{12} hosts (4096 hosts). Esta organização pode também dividir este bloco em blocos menores, formando sub-redes. No exemplo abaixo, o bloco 200.23.16.0/20 foi sub-dividido em oito blocos, cada um podendo alocar 2^9 hosts (512 hosts), um para cada departamento da organização.

```
Organização     11001000.00010111.00010000.00000000   200.23.16.0/20

Departamento1   11001000.00010111.00010010.00000000   200.23.18.0/23
Departamento2   11001000.00010111.00010100.00000000   200.23.20.0/23
...
Departamento7   11001000.00010111.00011110.00000000   200.23.30.0/23
```

Comando `ifconfig`

O comando `ifconfig` pode ser utilizado para visualizar a configuração ou configurar uma interface de host.

A seguir é mostrada uma saída do comando `ifconfig`:

```
evandro@Campeche:~$ ifconfig
eth0      Link encap:Ethernet  Endereço de HW 00:19:21:55:61:d0
          inet end.: 192.168.0.10  Bcast:192.168.0.255  Masc:255.255.255.0
          endereço inet6: fe80::219:21ff:fe55:61d0/64 Escopo:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Métrica:1
          pacotes RX:5522 erros:0 descartados:0 excesso:0 quadro:0
          Pacotes TX:4625 erros:0 descartados:0 excesso:0 portadora:0
          colisões:0 txqueuelen:1000
```

```
          RX bytes:5995540 (5.7 MB) TX bytes:735163 (717.9 KB)
          IRQ:18 Endereço de E/S:0x800

lo        Link encap:Loopback Local
          inet end.: 127.0.0.1  Masc:255.0.0.0
          endereço inet6: ::1/128 Escopo:Máquina
          UP LOOPBACK RUNNING  MTU:16436  Métrica:1
          pacotes RX:620 erros:0 descartados:0 excesso:0 quadro:0
          Pacotes TX:620 erros:0 descartados:0 excesso:0 portadora:0
          colisões:0 txqueuelen:0
          RX bytes:31000 (30.2 KB) TX bytes:31000 (30.2 KB)
```

Na primeira parte da saída do comando `ifconfig` é mostrada a configuração da interface `eth0` relativa a placa de rede Ethernet informando o endereço físico, endereço IP, endereço de *broadcast*, máscara de rede, MTU (*maximum transfer unit*) do enlace, entre outras informações. Na segunda parte é mostrada a configuração da interface lógica para testes de *loopback*.

4.2.3 Endereços IP privados

A RFC1918 reservou os seguintes três blocos de endereços IP para uso privado na Internet:

```
10.0.0.0        -   10.255.255.255  (10/8 prefix)
172.16.0.0      -   172.31.255.255  (172.16/12 prefix)
192.168.0.0     -   192.168.255.255 (192.168/16 prefix)
```

Nos referimos ao primeiro bloco como bloco de 24-bit, o segundo bloco como bloco de 20-bit, e o terceiro bloco como bloco de 16-bit . Note que (na notação anterior ao CIDR) o primeiro bloco consiste de uma rede classe A, enquanto que o segundo bloco consiste de um conjunto contíguo de 16 redes classe B, e o terceiro bloco consiste de um conjunto contíguo de 256 classe C.

Os endereços acima são ditos endereços IP **privados** e tem validade somente no escopo da rede interno de uma organização. Para conexão a Internet pública, precisa-se de endereços IP **públicos**.

Utilidade dos endereços privados

Os hosts dentro de uma organização podem ser particionados em três categorias:

1. Hosts que não requerem acesso a outras organizações ou a Internet global; hosts nesta categoria podem utilizar endereços IP privados.

2. Hosts que requerem acesso a um conjunto limitado de serviços externos à organização (como E-mail, FTP, login remoto), os quais podem ser mediados por *gateways* (como um roteador NAT). Para muitos hosts nesta categoria um acesso externo irrestrito pode não ser necessário, ou até mesmo não desejável para fins de segurança. Hosts nesta categoria também podem utilizar endereços IP privados.

3. Hosts que requerem acesso a serviços externos à organização; hosts nesta categoria requerem endereços IP públicos (como servidores web).

4.3 NAT (*Network Address Translation*)

O **NAT** (*network address translation*) é um mecanismo que possibilita a uma rede local usar apenas um endereço **IP público**, no que concerne ao mundo exterior, e utilizar endereços **IP privados** no que concerne a rede interna. Apenas um endereço IP público é usado para todos os dispositivos, o que possibilita modificar endereços de dispositivos na rede local sem notificar o mundo exterior. Com o NAT os dispositivos dentro da rede local não são explicitamente endereçáveis, ou visíveis do mundo exterior, o que pode ser considerado uma forma de segurança para a rede interna.

Outra importante vantagem do uso do NAT é a economia de IP públicos.

A implementação do NAT faz uso das **portas** dos protocolos da camada de transporte. Todos os datagramas deixando a rede local têm o mesmo e único endereço IP NAT de origem e diferentes números de porta origem. Para isto, um roteador NAT deve:

- Datagramas saindo: trocar (IP origem, porta) de cada datagrama saindo para (IP NAT, nova-porta). Os clientes/servidores remotos vão responder usando (IP NAT, nova-porta) como endereço destino;

- Lembrar, utilizando uma tabela de tradução NAT, de cada par de tradução (IP origem, porta) para (IP NAT, nova-porta);

- Datagramas entrando: trocar (IP NAT, nova-porta) nos campos de destino de cada datagrama entrando para o (IP origem, porta) correspondente armazenado na tabela NAT.

A figura 4.9. ilustra um exemplo apresentado por Kurose e Ross (2006a). Neste exemplo uma estação da rede local contata um servidor web remoto no endereço 128.119.40.186, porta 80, e informa seu endereço e porta origem 10.0.0.1, 3345. O roteador NAT traduz o endereço IP e porta da estação interna pelo IP NAT, nova-porta, 138.76.29.7, 5001. O servidor responde ao IP NAT, nova-porta. O roteador NAT, uma vez que recebe a resposta do servidor, traduz novamente para o endereço IP e porta origem da estação da rede local.

4.4 protocolo DHCP

O **protocolo DHCP** (*dynamic computador configuration protocol*) e permite a alocação dinâmica de endereços IP dentro de uma rede local. Com o DHCP, um servidor DHCP recebe uma solicitação de um cliente e aloca dinamicamente um endereço IP em resposta ao pedido do cliente. Com o DHCP um computador cliente pode adquirir toda a configuração necessária em uma única mensagem (por exemplo, o endereço IP, máscara de rede, roteador padrão, servidor DNS, etc).

O servidor DHCP deve ser configurado com a faixa de endereços IP disponíveis para oferecer. Quando um computador se conecta na rede, ele solicita um endereço IP se apresentando com seu endereço físico. O servidor então escolhe um endereço IP dentro da faixa disponível e aloca ao solicitante.

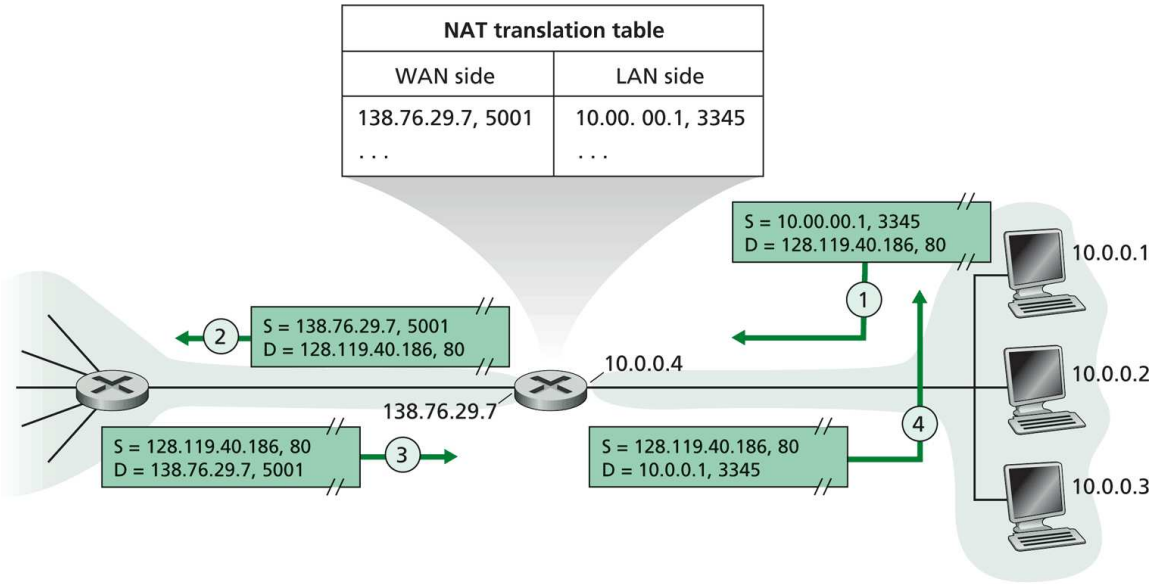


Figura 4.9: NAT (KUROSE; ROSS, 2006a).

4.5 Protocolo ARP

Quando um computador deseja enviar um datagrama a um destinatário conectado à sua rede local, ele entrega o datagrama a interface de rede para ela mapear o endereço IP no **endereço físico** do computador destino.

O endereço físico também é conhecido como **endereço de MAC** e corresponde ao endereço Ethernet de 48 bits, representado na forma de seis bytes hexadecimais, por exemplo, `00:19:21:55:61:d0`.

O protocolo ARP, definido na RF826, permite encontrar o endereço físico a partir do endereço IP da máquina alvo. Para tal, o protocolo usa um mecanismo de difusão (*broadcast*), enviando uma solicitação a todas as máquinas da rede, sendo que a máquina alvo responde indicando o par **endereço IP/endereço físico**. A figura 4.10 ilustra o funcionamento do protocolo ARP.

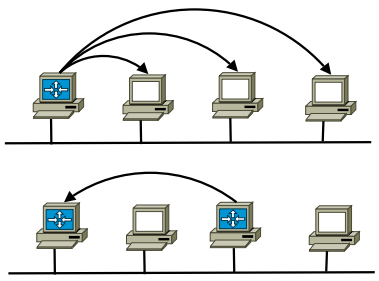


Figura 4.10: Protocolo ARP.

Para melhorar a performance do protocolo, cada máquina possui uma memória (*cache*) com as últimas consultas realizadas, evitando múltiplos *broadcasts*. Ainda como refinamento, junto com o *broadcast*, a estação solicitante envia seu par endereço IP/endereço físico, permitindo que todas as máquinas da rede incluam este par em suas *caches* locais.

Quando um hardware é trocado, a máquina que sofreu a mudança se anuncia na rede com o novo par endereço IP/endereço físico, logo após sua entrada em operação.

Comando `arp`

O comando `arp`, com parâmetro `-a`, mostra todas as entradas ARP armazenadas na memória *cache*, como no exemplo abaixo, relacionando endereços IP e endereços físicos.

```
evandro@Campeche:~$ arp -a
? (192.168.0.1) em 00:19:5B:00:4B:43 [ether] em eth0
```

Protocolo RARP

O protocolo RARP realiza a operação inversa do ARP, isto é, a partir de um endereço físico permite encontrar o endereço IP da máquina.

É geralmente utilizado por máquinas sem disco rígido (*disk-less*) para obter um endereço IP de um servidor. Para tal, um computador RARP envia um *broadcast* com o seu endereço físico solicitando um endereço IP. A máquina autorizada a responder o pedido RARP envia a resposta.

4.6 Roteamento

O roteamento de datagramas na Internet é uma das principais funções da camada rede. O protocolo IP assume que um computador é capaz de enviar datagramas a qualquer outro computador conectado á mesma rede local. Caso o destinatário não esteja na mesma rede, parte da função de roteamento é transferida para os roteadores (*gateways*).

Os roteadores podem ser equipamentos específicos ou computadores normais que possuem mais de uma interface de rede. O roteamento no IP baseia-se exclusivamente no identificador de rede do endereço destino. Cada roteador possui uma tabela, chamada tabela de roteamento, cujas entradas são pares: endereço de rede/endereço de roteador. Por exemplo, quando um computador deseja enviar um datagrama, inicialmente ele verifica se o destinatário está conectado a rede local. Se for o caso, ele entrega o datagrama a interface de rede que se encarrega de mapear o IP no endereço físico do computador destino, encapsular o datagrama IP em um quadro da rede e transmiti-lo. Caso o computador destino não se encontre na rede local, ele envia o datagrama ao roteador padrão (*gateway default*) da rede local. O roteador procura na sua tabela de roteamento o endereço do roteador que deve ser usado para alcançar a rede onde está conectado o destinatário do datagrama. O roteador encontrado pode não fazer parte da rede destino, mas, deve fazer parte do caminho a ser percorrido para alcançá-la.

Veja um exemplo de como funcionam as tabelas de roteamento, considerando a rede apresentada na figura 4.11.

Suponha que o computador A (IP `200.1.1.1`) tenha a tabela de roteamento dada na tabela a seguir e deseja enviar um datagrama IP ao computador B (IP `200.1.1.2`). Neste caso, o computador consulta sua tabela de roteamento e descobre que a rede 200.1.1.0/24 casa com o identificador da rede do computador B. A tabela indica que o próximo roteador é 0.0.0.0, o que quer dizer que está na mesma rede local. Então o computador A passa o datagrama diretamente a camada enlace através da interface `eth0` para proceder à entrega ao computador B.

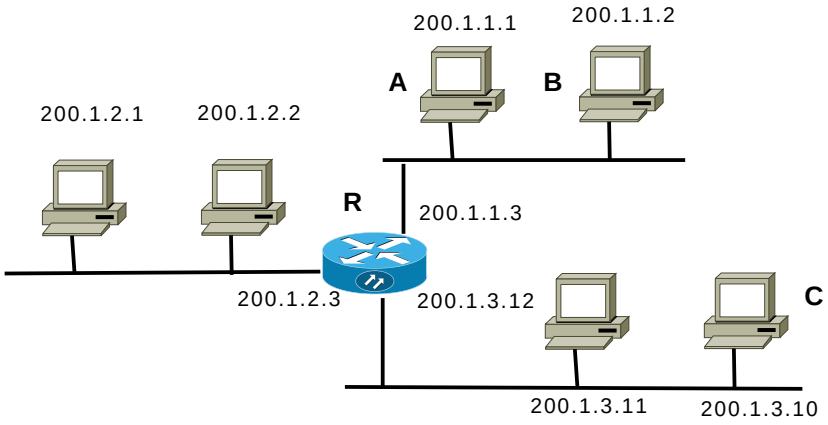


Figura 4.11: Roteamento na Internet.

```
Tabela de Roteamento do computador A
Destino         Roteador        Máscara           Interface
200.1.1.0       0.0.0.0         255.255.255.0     0 eth0
200.1.2.0       200.1.1.3       255.255.255.0     0 eth0
200.1.3.0       200.1.1.3       255.255.255.0     0 eth0
```

Suponha agora o caso em que o computador A queira enviar um datagrama ao computador C (IP `200.1.3.10`), situado em outra rede, no caso a rede 200.1.3.0/24. Consultando sua tabela de roteamento ele verifica que esta rede é acessível a partir do roteador `200.1.1.3`. Então ele passa o datagrama ao roteador para dar prosseguimento a entrega, o qual também é acessível a partir da interface `eth0`.

O roteador então consulta sua tabela de roteamento (veja tabela abaixo) e verifica que a rede 200.1.3.0/24 é acessível diretamente através da sua interface `eth2`. Sendo assim, ele entrega o datagrama a camada de enlace da rede 200.1.3.0/24 para fazer a entrega ao computador C.

```
Tabela de Roteamento do roteador R
Destino         Roteador       Máscara           Interface
200.1.1.0       0.0.0.0        255.255.255.0     0 eth0
200.1.2.0       0.0.0.0        255.255.255.0     0 eth1
200.1.3.0       0.0.0.0        255.255.255.0     0 eth2
```

Comando `netstat`

O comando `netstat` pode ser utilizado para visualizar as tabelas de roteamento. O parâmetro `-r` serve para listar a tabela de roteamento e o parâmetro `-n` imprime os endereços IP no formato numérico.

A seguir é mostrada uma saída do comando `netstat`:

```
evandro@Campeche:~$ netstat -rn
Tabela de Roteamento IP do Kernel
Destino         Roteador        MáscaraGen.     Opções   MSS Janela  irtt Iface
192.168.0.0     0.0.0.0         255.255.255.0    U         0 0          0 eth0
169.254.0.0     0.0.0.0         255.255.0.0      U         0 0          0 eth0
0.0.0.0         192.168.0.1     0.0.0.0          UG        0 0          0 eth0
```

4.6.1 Roteamento estático e protocolos de roteamento

A construção das tabelas de roteamento em cada roteador pode ser feita manualmente por um administrador de rede ou através do uso de protocolos de roteamento. A construção manual das tabelas de roteamento chama-se **roteamento estático**.

O uso de **protocolos de roteamento** permite que as tabelas sejam construídas e atualizadas dinamicamente. Devido a problemas de escala e de autonomia administrativa, os roteadores que vão participar de um roteamento dinâmico devem ser agrupados em regiões, ou **sistemas autônomos (SAs)** e devem rodar o mesmo protocolo em cada região.

Há duas classes de protocolos de rotemento dinâmico na Internet: os **protocolos intra–SA** e **inter-SA**.

Os protocolos de rotemento intra–SA atuam internamente ao sistema autônomo, definindo as melhores rotas em função de um custo atribuido a cada enlace a percorrer. Dois protocolos de rotemento intra–SA bastante utilizados na Internet são o RIP e o OSPF.

O **RIP** (*routing information protocol*) constrói as tabelas dinamicamente utilizando um **algoritmo de roteamento** chamado **vetor de distâncias**, calculando as melhores rotas tendo como base o número de enlaces a percorrer.

O **OSPF** (*Open Shortest Path First*) é um protocolo de roteamento mais elaborado, baseado no algoritmo **estado do enlace**, desenhado para convergir rapidamente ao ser informado de qualquer alteração de estado do enlace dentro da rede.

O roteamento **inter-SA** permite interconectar diferentes sistemas autônomos. Na Internet o protocolo mais utilizado neste domínio é o **BGP** (*border gateway protocol*), o qual permite implementar políticas específicas dependendo dos sistemas autônomos que serão integrados.

4.7 Protocolo IPv6

O protocolo IPv6 é a nova versão do IP, especificado pela RFC2460, e deve substituir o IPv4. Além disto, o IPv6 é incrementado por várias outras RFCs, por exemplo: RFC2463 (*ICMP for the Internet Protocol Version 6*) e RFC3775 (*Mobility Support in IPv6*).

Apesar de ter sido definido a quase 10 anos o IPv6 ainda não tem data para ser implementado difinitivamente na Internet. Este atraso se justifica pelo sucesso do IPv4, em particular por sua simplicidade de administração. Sua resiliência, que é a capacidade de voltar ao seu estado normal depois de ter sofrido uma pressão, como por exemplo, congestão ou falta de memória. Sua escalabilidade, na qual partes do sistema podem ser administrados como sistemas autônomos independentes. Sua flexibilidade e extensibilidade, a qual tem possibilitado soluções de alguns de seus problemas, como o CIDR que acabou com as classes de endereçamento, o NAT que possibilitou o uso de IP privados em uma rede interna e a autoconfiguração dos endereços em uma rede local com DHCP.

Um dos motivos que levaram ao desenvolvimento do IPv6 foi a escassez de endereços IP que se avizinhava no início dos anos 1990. Em partes isto foi devido ao uso de classes no IPv4, a qual atribuiu faixas gigantescas de endereços para algumas poucas empresas e instituições. Posteriormente, este problema acabou sendo minimizado com a proposição do endereçamento IP sem classes (CIDR) e também com o uso do NAT. Além da limitação do espaço de endereçamento o IPv4 não incluia em seu projeto original suporte para mobilidade, segurança e qualidade de serviço. Na questão da qualidade de serviço (QoS), o campo TOS (*type of service*) acabou

sendo utilizado posteriormente, como especificado na RFC1349 (*Type of Service in the Internet Protocol Suite*).

As principais melhorias introduzidas pelo IPv6 são:

- Ampliação do espaço de endereçamento de 32 bits para 128 bits;
- Melhoria no formato do datagrama IP;
- Melhoria no processo de autoconfiguração;
- Inserção de mecanismos para tratamento de mobilidade (MIPv6);
- Inserção de mecanismos de segurança (IPsec);
- Inserção de mecanismos para facilitar o gerenciamento de QoS.

4.7.1 Estrutura do datagrama IPv6

O datagrama IPv6 possui seu cabeçalho com tamanho fixo de 40 bytes, como mostra a figura 4.12.

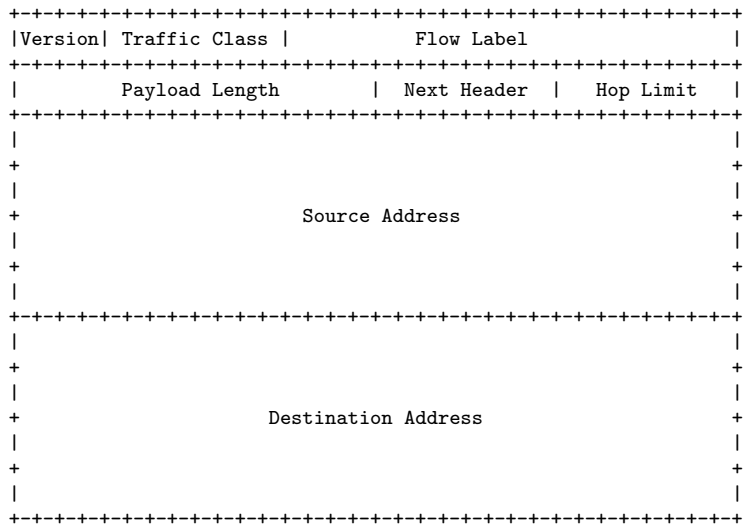


Figura 4.12: Formato do datagrama IPv6 (RFC2460).

- O campo versão (*Version*), de 4 bits, indica a versão do protocolo, no caso = 6.
- Classe de tráfego (*Traffic Class*), com 8 bits, é utilizado por nós de origem e roteadores intermediários para diferenciar diferentes classes e prioridade de tráfego entre os pacotes IPv6; tem funcionalidade similar a utilização do campo TOS do IPv4 permitindo de implementar "serviços diferenciados" de QoS (*DiffServ*) para os pacotes IP.
- Rótulo de fluxo (*Flow Label*), com 20 bits, pode ser utilizado por nós de origem para rotular sequências de pacotes que requerem tratamento especial em roteadores, como serviços tempo real.
- Comprimento dos dados (*Payload Length*), com 16 bits, especifica o tamanho dos dados após o cabeçalho, em bytes.

- Próximo cabeçalho (*Next Header*), com 8 bits, especifica o tipo do cabeçalho imediatamente após o cabeçalho IPv6; pode utilizar números similares aos do campo *Protocol* do IPv4, ou números específicos para o IPv6 (RFC3232), os quais são atualmente estabelecidos por uma base *on-line* no endereço `www.iana.org`, como por exemplo:

```
Next Header IPv6 - Header types
    00 = Hop-by-Hop Options
    41 = ipv6
    43 = Routing
    44 = Fragment
    51 = Authentication
    60 = Destination Options
    50 = Encapsulating Security Payload
    xx = Upper Layer Header
    58 = Internet Control Message Protocol (ICMP)
    59 - no next header
```

- Limite de saltos (*Hop Limit*), com 8 bits, é decrementado de 1 em cada nó de encaminhamento do pacote; o pacote é descartado quando o *Hop Limit* chega a zero.
- Endereço fonte (Source Address), com 128 bits.
- Endereço destino (Destination Address), com 128 bits.

Foram removidos do cabeçalho IPv6 os seguintes campos que haviam no IPv4:

- Tamanho do cabeçalho (*Header Length*), uma vez que o cabelalho IPv6 possui tamanho fixo;
- *Identification*, *Flags*, *Fragment Offset*, uma vez que no IPv6 não há mais fragmentação do datagrama, sendo utilizada uma técnica conhecida como descoberta da MTU do caminho (*Path MTU Discovery*);
- *Header Checksum*, o que tornou o processamento mais leve, evitando recalcular o valor do *checksum* em cada nó.

Extensões do cabeçalho IPv6

As extensões do cabeçalho IPv6, definidas na RFC2460 são as seguintes:

- Hop-by-Hop Options header
- Routing header
- Fragment header
- Destination Options header
- Authentication header
- Encrypted Security Payload header

As extensões são identificados pelo campo *Next Header* e são colocados entre o cabeçalho IPv6 e o pacote de nível superior. Elas são examinados unicamente pelo nó identificado no endereço de destino, a não ser o cabeçalho *hop-by-hop*.

O cabeçalho de opções *Hop-by-Hop* carrega informações que devem ser processadas por todos nós ao longo do caminho. As informações poderão ser utilizadas para reserva de recursos (exemplo RSVP), para encontrar destinos de *multicast*, entre outras, e deve seguir necessariamente o cabeçalho IPv6. A opção *Jumbogram* da extensão *hop-by-hop* permite que pacotes maiores que 64K sejam transmitidos. O campo *Option Data Lenght*, de 32 bits informa o tamanho dos dados, que devem se seguir ao mesmo.

A extensão cabeçalho de roteamento (*Routing header*) permite informar um conjunto de roteadores que devem ser visitados até o seu destino final e pode ser utilizada por aplicações que exigem qualidade de serviço.

A extensão cabeçalho de fragmentação (*Fragment header*) é usada para possibilitar a fragmentação e remontagem de datagramas. O IPv6 usa a técnica "*PATH MTU Discovery*" para determinar a máxima MTU na trajetória do pacote. Se o pacote for maior que esta MTU então ele é fragmentado na fonte. Os roteadores ao longo da rota não fragmentam o pacote. O destino remonta o pacote fragmentado.

Endereçamento IPv6 (RFC 3513)

O IPv6 tem endereçamento de 128 bits: espaço de 2^{128} endereços!!

É utilizada uma notação hexadecimal com agrupamento de 16 bits separados por "dois pontos", como por exemplo:

`2001:0DB8:5002:2019:1111:76ff:FEAC:E8A6` .

São utilizadas regras para simplificar a representação de zeros, por exemplo o endereço:

`0237:0000:ABCD:0000:0000:0000:0000:0010` , pode ser transformado em:

`237::ABCD:0:0:0:0:10` , ou, melhor ainda,

`237:0:ABCD::10`.

Tipos de endereçamento IPv6

- *Unicast*: identifica um hospedeiro (interface);
- *Multicast*: identifica um conjunto de hospedeiros. O Ipv6 eliminou o *broadcast* e enriqueceu o processo de multicast;
- *Anycast*: uma novidade no Ipv6. É um endereço que pode ser atribuído a múltiplos hospedeiros. O roteamento da Internet permite encaminhar pacotes ao endereço *anycast* mais próximo.

Endereço *Unicast*

Os endereços *unicast* são **endereços públicos**, roteáveis na Internet. Apresentam uma estrutura hierárquica, cujas interfaces são configuradas com máscaras de 64 bits.

O endereço *link-local* traz a noção de **endereços privados** dentro de uma rede. Não são roteáveis e começam com `1111111010`. Por exemplo: `FE80:A0000::1` .

Outros endereços notáveis:

- Endereço *Loopback*: `::1`
- Endereço não especificado: `::`
- Endereços *Multicast*: `FF02::1` – todos os nós do enlace; `FF02::2` – todos os roteadores do enlace.

4.7.2 ICMPv6

O ICMPv6 (RFC2463) reporta erros se pacotes não podem ser processados apropriadamente e envia informações sobre o status da rede. Apresenta várias melhorias em relação ao ICMPv4. O IGMP, que trata *multicast* no IPv4, foi incorporado no ICMPv6. O ARP/RARP foi incorporado através da utilização da técnica *Neighbour Discovery* (descoberta de informações da vizinhança). Incorporou mensagens de registro de localização para IP Móvel.

As mensagens ICMPv6 são transportadas por datagramas IPv6 com número *Next Header* igual a 58H e tem formato geral mostrado na figuraa 4.13.

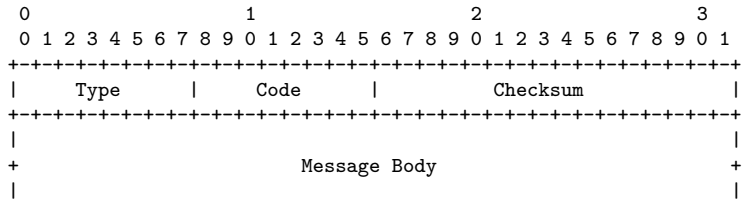


Figura 4.13: Formato geral da mensagem ICMPv6 (RFC2463).

As mensagem com bit mais significativo (MSB) do campo *type* = 0 indicam **erro** (valores de 0–127) e com o MSB = 1 são de **informação** (valores de 128–255).

São exemplos de mensagens de erro:

```
Type
   1    Destination Unreachable
   2    Packet Too Big
   3    Time Exceeded
   4    Parameter Problem
```

São exemplos de mensagens de informação:

```
Type
 128      Echo Request
 129      Echo Reply
 133-136  Neighbor Discovery
```

O campo *Code* indica o código da mensagem, por exemplo, para mensagens de erro *Destination Unreachable* poderemos ter:

```
Type            1
Code            0 - No route to destination
                1 - Communication with destination
                      administratively prohibited
                2 - Beyond scope of source address
                3 - Address unreachable
                4 - Port unreachable
                5 - Source address failed ingress/egress policy
                6 - Reject route to destination
```

A mensagems de erro *Packet Too Big* (*Type* = 2, *Code* = 0) é utilizada pela técnica de descoberta da MTU do enlace (*Path MTU Discovery*), uma vez que o IPv6 não fragmenta do datagrama nos roteadores intermediários. Nesta técnica o nó emissor assume a MTU do seu enlace de saída e envia o pacote para destino. Caso algum roteador da rota detecte que o pacote é muito grande para o MTU então ele avisa o nó fonte com ICMPv6 *Packet Too Big* (que inclui o tamanho da MTU do enlace problema). O nó usa esta nova MTU para encaminhar o pacote novamente ao seu destino, podendo o processo todo pode se repetir em outros roteadores.

A mensagems de erro *Time Exceeded* (*Type* = 3, *Code* = 0) é enviada quando o limite de saltos (*hop limit*) é excedido em um dado roteador.

4.7.3 Autoconfiguração

A autoconfiguração (RFC2462) é um dos pontos fortes do IPv6 permitindo gerar endereços IP automaticamente para as interfaces, sem nenhum tipo de configuração manual. Para isto o IPv6 combina prefixos divulgados pelos roteadores com o endereço de MAC (ou número randômico) para. Na ausência de roteadores usa-se o prefixo FE80 para gerar um endereço privado "link-local".

A autoconfiguração é usada quando um sítio não necessita estabelecer endereços particulares as estações. Caso haja necessidade de se atribuir endereços específicos, pode-se utilizar o DHCPv6, como no IPv4.

Na autoconfiguração são utilizadas mensagens ICMPv6 de descoberta de vizinhos (*Neighbor Discovery*) (RFC2461), as quais são equivalentes as mensagens ARP. As mensagens ICMPv6 *Neighbor Discovery* podem ser do tipo (entre outras):

```
Neighbor Discovery:
Type Code
133  0    Router Solicitation
134  0    Router Advertisement
135  0    Neighbor Solicitation
136  0    Neighbor Advertisement
```

As mensagens *Neighbor Solicitation* e *Neighbor Advertisement* são utilizadas para a descoberta de vizinhos e detecção de endereços duplicados, como ilustra a figura 4.14.

A mensagem *Router Solicitation* é utilizada para a descoberta de roteadores na rede local e *Router Advertisement* são utilizadas pelos roteadores, periodicamente, para anunciar sua presença e divulgar seu prefixo, como ilustra a figura 4.15.

.

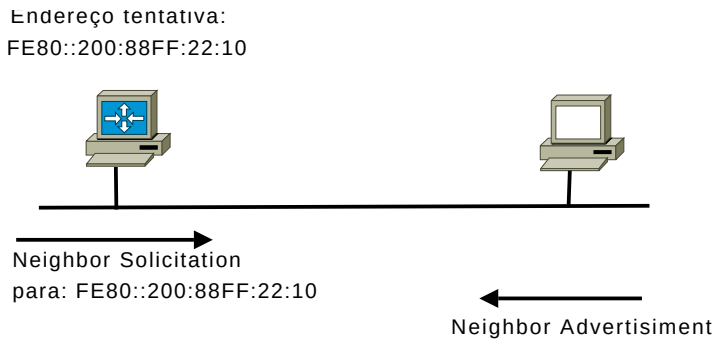


Figura 4.14: *Neighbor Solicitation* e *Neighbor Advertisement*.

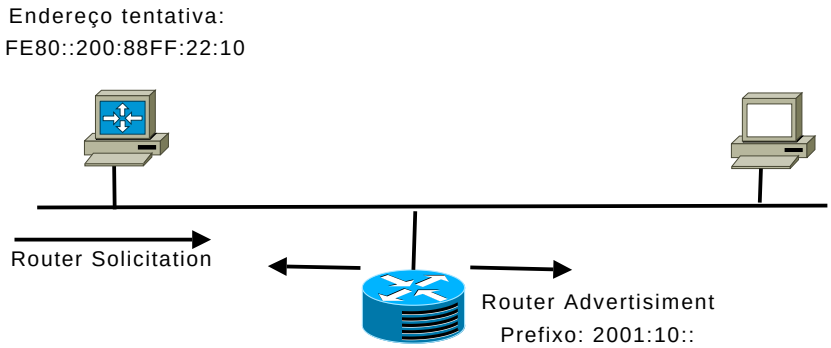


Figura 4.15: *Router Solicitation* e *Router Advertisement*.

Passos para autoconfiguração

1. Um endereço tentativa é formado com o prefixo FE80 (*link-local address*) e associado a interface;

2. O nó se junta aos grupos multicasting;

3. Uma mensagem *Neighbor Solicitation* é enviado com o endereço tentativa como *target address*. Se detectado IP duplicado o nó deverá ser configurado manualmente;

4. O nó realiza um *Router Solicitation* para o endereço `FF02::2`. (grupo *multicast* dos roteadores);

5. Todos roteadores do grupo *multicast* respondem e o nó se autoconfigura para cada um deles;

Formação do Endereço a partir do MAC

Os 64 bits do identificador de hospedeiro são formados pela inserção de FFFE entre o terceiro e quarto byte do MAC e complementando-se o segundo bit menos significativo do primeiro. Supondo que o hospedeiro possui endereço MAC de `11:AA:BB:2A:EF:35` e o prefixo recebido do Roteador (via Advertência) é `2001:10:/64` então o endereço formado é: `2001:10::11A8:BBFF:FE2A:EF35` .

4.7.4 Transição de IPv4 para IPv6

Nem todos os roteadores podem ser atualizados simultaneamente, uma vez que "dias de mudança geral" são inviáveis. Os projetistas do IPv6 pensaram a transição de forma gradativa, com roteadores IPv4 e IPv6 convivendo simultaneamente. Para isto é utilizada uma técnica conhecida como "tunelamento", na qual datagramas IPv6 são carregados em datagramas IPv4 entre roteadores IPv4, como ilustra a figura 4.16.

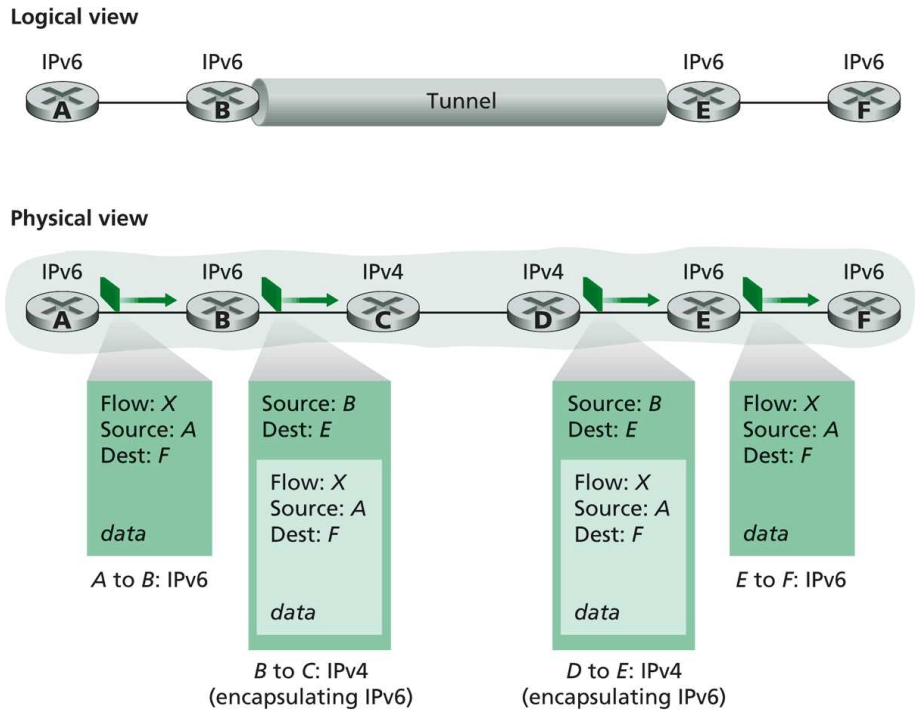


Figura 4.16: Tunelamento de um datagrama IPv6 em um IPv4 (KUROSE; ROSS, 2006a).

Capítulo 5

Protocolos de Enlace e Redes Locais

As redes locais são redes de computadores concentradas em uma área geográfica restrita, por exemplo no âmbito de uma empresa ou universidade, e permitem aos computadores e usuários da rede compartilharem recursos. Com a possibilidade de as redes locais serem conectadas entre si, formando a Internet, cresceu de forma extraordinária as possibilidades de acesso a recursos e serviços, de forma que hoje praticamente não se pensa mais uma rede local isolada.

5.1 Protocolos de enlace

A camada inter-rede da Internet oferece um serviço de comunicação de datagramas entre dois sistemas terminais. Esta comunicação passa por caminhos que iniciam no host de origem, passando por uma série de roteadores e termina no host destino. Cada equipamento, como hosts e roteadores, é chamado de nó e o canal de comunicação entre dois nós adjacentes ao longo de uma rota é chamado de enlace. Desta forma, para mover um datagrama desde sua origem até seu destino, ele precisa percorrer cada um dos enlaces individuais entre os diversos nós.

Os enlaces entre nós vizinhos podem ser suportados por diferentes tecnologias, utilizando protocolos específicos, os quais são chamados de **protocolos de enlace**. As unidades de dados de protocolos trocadas pelos protocolos de enlace são chamadas **quadros** (*frames*) e tipicamente encapsulam um datagrama da camada rede.

Assim como os protocolos inter-rede são protocolos fim a fim que movem datagramas de um host a outro, os protocolos de enlace são protocolos nó a nó, movendo quadros sobre um simples enlace entre nós vizinhos.

5.1.1 Serviços oferecidos pelos protocolos de enlace

Dentre os possíveis serviços oferecidos pelos protocolos de enlace está o **acesso ao meio físico** e o **encapsulamento** (*framing*). No caso da Internet, os datagramas da camada inter-rede são encapsulados em quadros e o acesso ao meio vai depender do tipo de protocolo utilizado. Grosso modo podemos dividir os protocolos de enlace em dois grandes grupos: os **protocolos de enlace ponto a ponto** e os **protocolos de enlace multiponto**, que caracterizam as redes locais. No caso de um protocolo ponto a ponto o acesso ao meio é bastante simples, aceitando o envio de um quadro caso o meio estiver livre. Já no caso de protocolos multiponto, como o protocolo de rede local Ethernet, há necessidades de mecanismos especiais para acesso ao meio.

Além destes serviços básicos, dependendo do protocolo, outras ações podem ser executadas sobre os quadros, como: comunicação *full-duplex* ou *half-duplex*, detecção e correção de erros, entrega de dados garantida e controle de fluxo.

Técnicas de detecção e correção de erros

Em qualquer transmissão de informação existe o risco do erro sob o efeito de perturbações aleatórias ou de ruídos. De um modo geral, os erros nos dados transmitidos através da rede podem ser:

- Erros de bit introduzidos nos dados;
- Perda de pacotes;
- Falha nos enlaces de comunicação.

Os erros de bits são bastante raros, havendo técnicas para detectá-los e mesmo corrigi-los. Se o erro for muito grave, o pacote pode ser descartado e terá que ser retransmitido. No caso da perda de pacotes, a retransmissão é a solução. Já no caso de falha de um enlace, algumas vezes é possível utilizar uma rota alternativa, evitando a ligação com defeito. A detecção e correção de erros no nível de bits dos quadros enviados de um nó a outro nó fisicamente conectado são geralmente serviços oferecidos pelos protocolos da camada de enlace.

Três técnicas simples de detecção e correção de erros no nível de bits são a checagem de paridade, os métodos de *checksum* e os métodos de redundância cíclica (CRC).

Checagem de paridade

Talvez a forma mais simples de detecção de erros de bits seja utilizar um simples bit de paridade. Por exemplo, suponha que um dado D a ser transmitida tenha d bits. Usando um esquema de paridade, o emissor acrescenta ao dado um bit adicional e escolhe seu valor como o total de bits em 1 de d + 1 bits (o total de bits em D mais o bit de paridade), de forma que seja par, como mostra o exemplo abaixo.

```
       Dado      Paridade
+----------------+-+
 0111000110101011 1
```

Quando o receptor recebe o dado, ele computa os bits em 1, incluindo o bit de paridade, e verifica se o resultado é par. Caso não seja, o receptor sabe que algum bit teve seu valor alterado.

Algumas técnicas permitem, além de detectar erros em bits, de corrigi-los. Estas técnicas são conhecidas como FEC (*forward error correction*). Elas são úteis, pois permitem diminuir a necessidade de retransmissões pelo emissor.

Checksum

Na técnica de *checksum* o dado D é tratado como uma seqüência de palavras binárias. O método consiste em somar a seqüência de palavras e usar a soma para detectar erros nos bits. Este é método utilizado pelos protocolos Internet e foi descrito na seção 3.2.1.

Checagem de redundância cíclica

Os códigos de redundância cíclica, ou códigos CRC (**cyclic redundancy check**), estão entre os métodos mais utilizados nas redes de computadores para detecção de erros, pois podem descobrir mais erros que um checksum.

Os códigos CRC são também conhecidos como códigos polinomiais, já que podem ser vistos como um polinômio onde os coeficientes são 0 e 1. Por exemplo, o número binário de 4 bits, 1011, corresponde ao polinômio

$M(x) = 1.x^3 + 0.x^2 + 1.x^1 + 1.x^0 = x^3 + x^1 + 1$

cujo grau é 3.

Os códigos CRC operam como segue. Para uma peça de dados D a ser transmitida, o emissor e o receptor devem acordar primeiramente sobre um polinômio gerador, G, de grau r. Assim, o emissor adiciona ao dado D mais r bits, de forma que o resultado da soma d + r seja exatamente divisível por G usando aritmética módulo 2.

Quando o receptor recebe os dados, ele divide d + r por G; caso a divisão não seja exata, ele sabe que há erros nos dados; caso contrário, o dado é considerado correto.

5.1.2 Protocolos de enlace ponto-a-ponto

Um protocolo de enlace ponto a ponto consiste de um simples emissor em uma extremidade de um enlace e um simples receptor na outra ponta. Muitos protocolos tem sido desenvolvido para este tipo de comunicação, como por exemplo, o protocolo PPP (*point-to-point protocol*) e o HDLC (*high-level data link control*).

O PPP é um dos protocolos ponto a ponto mais utilizados atualmente. É tipicamente o protocolo escolhido para conectar um computador pessoal residencial a um provedor de acesso a Internet usando uma linha telefônica. Também é utilizado em enlaces ponto a ponto entre roteadores.

Protocolo PPP

O protocolo PPP pode operar sobre uma linha telefônica (usando por exemplo uma conexão via modem de 54K bps), sobre um enlace SONET/SDH (*syncrhonous optical network/syncrhonous digital hierarchy*), sobre uma conexão X.25 ou sobre um circuito digital RDSI (rede digital de serviços integrados).

O protocolo PPP recebe um pacote da camada inter-rede (por exemplo, um datagrama IP) e o encapsula em um quadro da camada enlace PPP, de forma que o receptor será capaz de identificar o início e o fim do quadro, bem como o pacote da camada rede que ele contém.

O formato do quadro PPP, mostrado na figura 5.1, sempre inicia e termina com 01111110 (chamado de *flag*), o segundo byte é sempre 11111111 (chamado de endereço) e o terceiro byte é sempre 00000011 (chamado de controle). Os demais campos são os seguintes:

- *Protocol* (1 ou 2 bytes), indica ao receptor qual o protocolo da camada de rede que está sendo encapsulado no quadro. No caso de um datagrama IP, este campa tem o valor hexadecimal 21.
- *Information* (tamanho variável, podendo ter no máximo 1500 bytes), contém o pacote encapsulado (dado), por exemplo um datagrama IP.

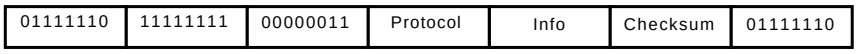


Figura 5.1: Formato do quadro PPP.

- *Checksum* (2 a 4 bytes), usado para detectar erros nos bits do quadro transmitido.

O protocolo SLIP (*Serial Line Internet Protocol*) é outro protocolo similar ao protocolo PPP.

5.1.3 Protocolos de enlace de múltiplo acesso

O problema central nos enlaces de múltiplo acesso é determinar quem deve transmitir e quando. Como vários podem transmitir quadros ao mesmo tempo, estes poderão colidir e serão perdidos. Os protocolos de acesso múltiplo ao meio permitem coordenar o acesso ao meio, evitando ou detectando estas colisões.

Nas redes de computadores as técnicas mais utilizadas para acesso a um meio comum são conhecidas como **protocolos de acesso randômico**. Vários protocolos deste tipo foram desenvolvidos e os mais conhecidos derivam do protocolo ALOHA, desenvolvido no final dos anos sessenta para permitir interligar via rádio os computadores espalhados pelo campus da universidade do Hawaii (USA), situados em diferentes ilhas do Pacífico. O protocolo utilizado nas redes Ethernet e nas redes locais sem derivam do ALOHA e são baseados na técnica de escuta da portadora antes de enviar uma informação (CSMA – *carrier sense multiple access*).

Protocolo ALOHA

Na primeira versão do protocolo ALOHA, quando um nó tinha um quadro a ser transmitido, ele o transmitia imediatamente. Se após um tempo de atraso o emissor ouvisse sua transmissão (reflexão do sinal de rádio transmitido), ele assumia que não havia ocorrido conflito. Caso contrário, assumia que havia ocorrido que uma colisão e retransmitia o quadro com uma probabilidade p, senão esperava um tempo correspondente ao tempo de transmissão e tentava enviar novamente com probabilidade p.

Protocolo CSMA

O protocolo CSMA foi projetado para funcionar com computadores conectados em barramento e introduziu dois novos princípios em relação ao ALOHA:

- Escutar a portadora antes de enviar um quadro (*carrier sense*) (o que não era possível no ALHOA devido ao tempo de propagação do sinal de rádio). Neste processo, o nó escuta o canal: caso o canal estiver livre transmite o quadro imediatamente; caso o canal estiver ocupado, volta a escutá-lo depois de decorrido um tempo randômico para tentar nova transmissão.

- Se alguém começar a transmitir no mesmo tempo, pára a transmissão. Este procedimento é chamado de detecção de colisões (*colision detection*), onde os nós continuam ouvindo o canal enquanto transmitem: caso detectem uma sobreposição de transmissões (colisões), param imediatamente a transmissão.

Estas duas regras são as características principais do protocolo (CSMA/CD – *carrier sense multiple access/colision detection*), utilizado nas redes locais baseadas no protocolo Ethernet.

5.2 Redes Locais

A Ethernet (padronizadas como IEEE 802.3) talvez seja a tecnologia mais utilizada atualmente em redes locais (LAN – *local area networks*) e utiliza difusão (*broadcast*) sobre um barramento lógico para a comunicação entre seus pares. As redes locais sem fio (padronizadas como IEEE802.11) também utilizam *broadcast* sobre um meio comum, no caso o ar e o espectro eletromegnético.

Numa rede local, todos os computadores e demais dispositivos de rede são diretamente conectados e usam o mesmo tipo de protocolo de enlace. Um roteador conectando a rede local a Internet provê uma forma de acesso a Internet a todos os equipamentos da rede local, como mostra a figura 5.2.

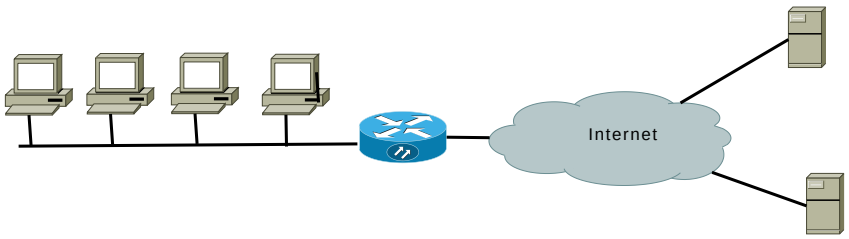


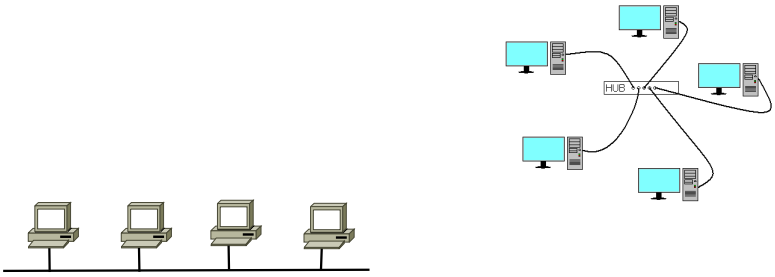
Figura 5.2: Rede local conectada a Internet por meio de roteador.

5.2.1 Topologia de redes locais

Uma maneira de classificar as redes locais é quanto a topologia de conexão de seus nós. As topologias mais conhecidas são o **barramento**, a **estrela** e o **anel**.

A topologia em barramento foi popularizada pelas redes locais Ethernet. As primeiras redes Ethernet conectavam diretamente seus nós por meio de um barramento de cabos coaxiais, utilizando um protocolo de múltiplo acesso para controlar o acesso ao meio. As redes Ethernet atuais utilizam dispositivos contentradores, como *hubs* e *switches*, formando uma **topologia física** do tipo **estrela**. Todavia a **topologia lógica** continua sendo do tipo **barramento**, utilizando o protocolo CSMA-CD para controlar o acesso ao meio. A figura 5.3 ilustra estas formas de conectividade em rede.

A topologia em anel foi popularizada nos anos 1980 pelas redes *token-ring* (padronizadas como IEEE 802.5), as quais utilizam uma conexão física em anel de seus nós e controlam o acesso ao anel por meio de um protocolo baseado na passagem de "ficha" (*token*) entre as estações. Outras tecnologias também utilizaram a topologia em anel ou variantes desta topologia, como as redes ópticas FDDI (*fiber digital distributed interface*) e DQDB (*distributed queue dual bus*)(padronizadas como IEEE 802.6). A figura 5.4 ilustra computadores conectados em uma topologia em anel.



(a) Barramento. (b) Estrela.

Figura 5.3: Topologias em barramento e estrela.

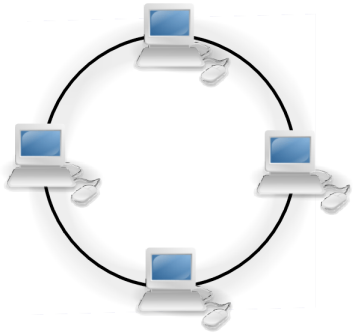


Figura 5.4: Topologia em anel.

Com a popularização das redes Ethernet e o avanço tecnológico que tem proporcionado taxas de transmissão cada vez mais altas, o mercado atual é dominado por esta tecnologia, estando as topologias em anel praticamente em desuso.

5.2.2 Placas adaptadoras

Os protocolos de enlace são em geral implementados sobre placas adaptadoras que fazem a interface entre o host, seja ele um computador terminal ou um roteador, e o meio físico. Desta forma, os principais componentes de um adaptador de rede são sua interface com o barramento do host e sua interface com o enlace físico. Por exemplo, uma placa de rede Ethernet de 100 Mbps possui uma interface para conexão da mesma diretamente no barramento do computador, e uma interface de rede, que pode ser tipo RJ45 (conexão com par trançado) ou BNC (conexão com cabo coaxial) (em desuso).

5.2.3 Endereços físicos

Os nós das redes locais trocam quadros (*frames*) entre si através de um canal comum (*broadcast*). Isto significa que, quando um nó transmite um quadro, todos os demais nós vão receber este quadro. Todavia, em geral, um nó não quer enviar quadros a todos nós, mas sim a um nó particular. Para prover esta funcionalidade, os nós de uma rede local devem ser capaz de endereçar os demais nós quando enviam um quadro. Desta maneira, quando um nó recebe um quadro, ele pode determinar se o quadro está endereçado a ele ou a outro nó da rede:

- Se o endereço do quadro recebido casa com o endereço físico do nó que o recebeu, então

o nó extrai o datagrama (da camada inter-rede) do quadro recebido (camada de enlace) e repassa para cima na sua pilha de protocolos;

- Se o endereço do quadro recebido não casa com o endereço físico do nó o recebeu, o nó simplesmente ignora o quadro.

Na verdade não é o nó da rede que tem um endereço físico, mas sim, cada adaptador de rede da rede local. Nas redes locais Ethernet, o endereço físico é também chamado de endereço Ethernet ou ainda **endereço MAC** (*media access control*). Um endereço Ethernet é um número expresso na notação hexadecimal, de seis bytes, dando 248 possíveis endereços, por exemplo, o endereço `00:19:21:55:61:d0`. Este endereço é permanente, sendo gravado pelo fabricante do adaptador de rede em uma memória ROM (*read only memory*).

Resolução de endereço físico

Quando um datagrama da camada inter-rede, endereçado a um computador de uma rede local chega ao roteador de borda, a partir da Internet, o roteador deverá encapsular este datagrama em um quadro da camada enlace para poder entregá-lo ao computador destino. Para que isto seja feito, o roteador deverá mapear o endereço IP no endereço físico do computador destino. Como descrito na seção 4.5 esta tarefa é realizada pelo protocolo ARP.

5.2.4 Ethernet

Ethernet é a tecnologia de redes locais mais difundida atualmente. Pode-se dizer que a Ethernet está para as redes locais, assim como a Internet está para as redes geograficamente distribuídas de alcance global.

A Ethernet usa o protocolo de acesso randômico CSMA-CD, que é completamente descentralizado, e o hardware (placa de rede, *hubs* e *switches* Ethernet) tem um custo bastante atrativo.

Existem várias tecnologias de rede local Ethernet, que operam em velocidades de 10/100 Mbps até 1/10 Gbps. Podem rodar sobre cabo coaxial, par trançado de cobre ou ainda fibra óptica, sendo que, ao nível lógico, todas as máquinas compartilham um barramento comum, sendo portando a velocidade de acesso também compartilhada entre as estações.

A figura 5.5 ilustra, na forma de um mapa conceitual, uma visão dos principais conceitos envolvidos nas redes Ethernet.

Quadro Ethernet

O quadro (frame) Ethernet tem as seguintes características, mostradas na figura 5.6:

- Preâmbulo (8 bytes), cada um dos primeiros sete bytes do preâmbulo tem o valor `10101010` e o oitavo byte tem o valor `10101011`.
- Endereço Destino e Origem (6 + 6 bytes), contém o endereço físico da origem e destino do quadro, nomeados `AA-AA-AA-AA-AA-AA` e `BB-BB-BB-BB-BB-BB`, respectivamente.
- Tipo (2 bytes), permite identificar o tipo do protocolo da camada superior, por exemplo, o protocolo IP (ou outro como Novell IPX).

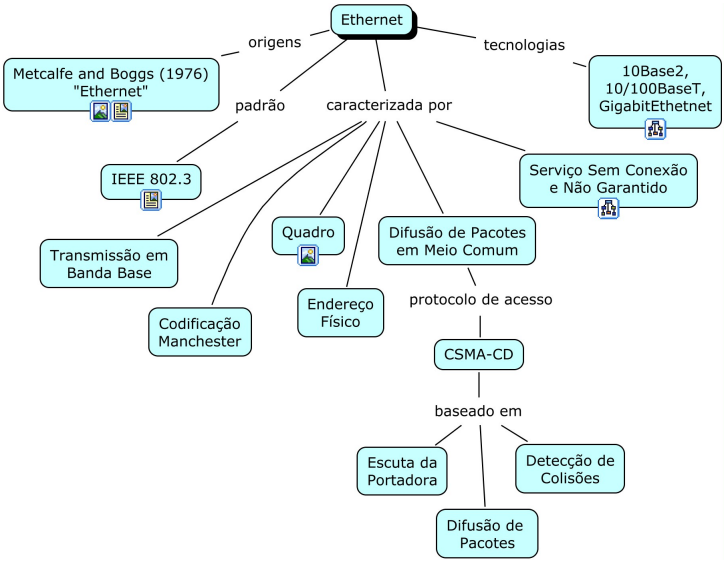


Figura 5.5: Visão dos principais conceitos das redes locais Ethernet.

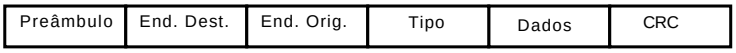


Figura 5.6: Formato do quadro Ethernet.

- Dados (46 a 1500 bytes), carrega o datagrama IP, sendo o MTU (*maximum tranfer unit*) o quadro Ethernet 1.500 bytes.

- CRC (*cyclic redundancy check*) (4 bytes), permite ao receptor detectar quaisquer erros introduzidos nos bits do quadro recebido.

5.3 *Hubs*, pontes e *switches*

5.3.1 Hubs

O modo mais simples para interconectar computadores numa rede local é através de um ***hub***. Um *hub*, ou concentrador, é um dispositivo que simplesmente pega os bits dos quadros de uma porta de entrada e retransmite às portas de saída. *Hubs* são essencialmente repetidores e operam sobre os bits, atuando portanto ao nível da camada física (camada 1 do modelo OSI).

Apesar de no nível físico a topologia dos equipamentos conectados a um *hub* ser do tipo estrela, no nível lógico um *hub* implementa um barramento de múltiplo acesso entre os host que são conectados em suas interfaces.

5.3.2 Pontes

Uma **ponte** (*bridge*) é um dispositivo eletrônico que permite que várias redes locais sejam concatenadas. Cada ponte conecta dois segmentos de rede e faz com que uma cópia de cada quadro que chega a um segmento seja transmitida ao outro segmento. Deste modo, os dois segmentos da rede local operam como se fosse uma rede única. Diferentemente do hub, uma ponte manipula quadros completos, atuando portanto ao nível da camada enlace (camada 2 do modelo OSI).

5.3.3 *Switches*

Um ***switch*** (ou comutador) é um dispositivo eletrônico capaz de comutar o tráfego de uma LAN, atuando no nível da camada 2, diminuindo o espaço de conflitos no acesso ao meio comum. Pelo fato de reduzir o espaço de conflitos no acesso ao meio os *switches* aumentam a eficiência da rede local.

Os *switches* de rede local geralmente substituem os *hubs* compartilhados e são concebidos para trabalhar com as infra-estruturas de cabeamento já existentes.

A seguir estão duas operações básicas realizadas pelos switches:

- Comutar quadros de dados – Os *switches* recebem quadros em uma interface, selecionam a porta correta para encaminhar os quadros com base no endereço físico destino do quadro e, em seguida, encaminham os quadros com base na escolha do caminho.
- Manter as operações do *switch* – Os *switches* criam e mantêm tabelas de encaminhamento.

Um switch cria uma tabela de encaminhamento aprendendo os endereços MAC dos hosts que estão conectados a cada uma de suas portas. Quando dois hosts conectados querem se comunicar, o switch consulta a tabela de encaminhamento e estabelece uma conexão virtual entre as portas.

Referências Bibliográficas

COMER, D. E. *Interligação em Redes com TCP/IP*. Trad. 5. Rio de Janeiro: Campus, 2006.

KUROSE, J. F.; ROSS, K. W. *Computer Networking: A Top-Down Approach Featuring the Internet*. 4. ed. USA: Addison Wesley, 2006.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet: Uma abordagem top-down*. Trad. 3 ed. São Paulo: Addison Wesley, 2006.

PETERSON, L. L.; DAVIE, B. S. *Redes de Computadores: Uma abordagem de sistemas*. Trad. 3 ed. Rio de Janeiro: Campus, 2004.

RFC. *Request for Comments*. Disponível em: <www.ietf.org; www.rfc-editor.org>.

STEVENS, W. R. *TCP/IP Illustrated*. New York: Assidon-Wesley, 2008.

TANENBAUM, A. S. *Redes de Computadores*. trad. 4 ed. Rio de Janeiro: Elsevier, 2003.