

Comunicação de Dados

e

Redes de Computadores

Índice

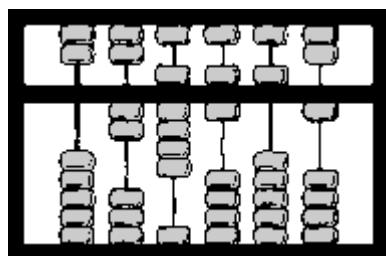
Índice	2
1 Histórico das Redes de Computadores e Telecomunicações	3
2 Transmissão de Dados	11
3 Sistemas de Comunicação.....	24
4 Erros.....	30
5 Interface de Comunicação de Dados	35
6 Protocolos de Enlace	44
7 Modelo Hierárquico de Protocolos e Padronização	53
8 Projeto e Desenvolvimento de Protocolos	66
9 Redes Locais, Ethernet e Internet.....	84
10 Camada de Rede.....	120
11 Camada de Transporte	149
12 Camada de Sessão	160
13 Implementação da Camada de Sessão.....	184
14 Camada de Apresentação	193
15 Camada de Aplicação.....	208
16 Redes Sem Fio e QoS.....	223

Capítulo

1

Histórico das Redes de Computadores e Telecomunicações

1.1 Histórico da Computação



adições que ficou conhecida como Pascalina.

Em 1670 Gottfried von Leibniz inventou uma calculadora que efetuava as quatro operações fundamentais e extraía a raiz quadrada.

Joseph Marie Jacquard introduziu, em 1801, nos teares de sua fábrica um sistema de cartões perfurados que representavam justamente os desenhos pretendidos, inventando assim, o primeiro tear programável.

A invenção da Máquina Analítica (também mecânica) por Charles Babbage, professor de matemática, por volta de 1833, abriu caminho para a construção do que hoje seria descrito como um computador digital mecânico automático totalmente controlado por um programa.

Alguns anos depois George Boole iniciava um processo que implicaria em importantes aplicações tecnológicas, publicando em 1847 e em 1854 os livros “A Análise Matemática da Lógica” e “Uma Investigação das Leis do Pensamento” respectivamente, dando a ele o título de inventor da lógica matemática. Sua proposta era de que qualquer coisa podia ser representada por símbolos e regras. O desenvolvimento de suas idéias deu origem à chamada álgebra de Boole.

Em 1890, Hermann Hollerith desenvolveu um equipamento que visava minimizar o imenso trabalho dispensado no censo dos Estados Unidos. Seu invento baseava-se nos cartões perfurados idealizados por Jacquard e o sucesso foi tão grande que ele fundou, em 1896, a Tabulation Machine Company. A TMC veio a fundir-se com mais duas empresas formando a Computing Tabulation Recording Company. A mesma CTRC, anos depois da morte de Hollerith, mudava de nome e nascia a IBM - International Business Machine.

Já no século XX, nos anos 30, Alan Turing inventou um dispositivo de uso geral, capaz de receber instruções para trabalhar com qualquer tipo de informação, chamado de Máquina de Turing. A máquina funcionaria usando mecanismos relacionados com conceitos de cálculo de entrada, saída e com um programa.

Na mesma época, Konrad Zuse criou o primeiro computador eletromecânico chamado Versuchmodell 1 ou Z-1. Foi a primeira calculadora universal binária controlada por um programa. Zuse ainda produziu mais três máquinas (Z2, Z3 e Z4), que aperfeiçoaram a Z1.

Anos mais tarde, o matemático húngaro John von Neumann, por volta de 1945, formalizou o projeto lógico de um computador. Ele sugeriu que as instruções fossem armazenadas na memória do computador. Até então elas eram lidas de cartões perfurados e executadas, uma a uma. Participou do projeto de construção do ENIAC – Electronic Numerical Integrator and Calculator (University of Pennsylvania), construído durante a Segunda Guerra Mundial, dentro de um programa do exército americano que procurava automatizar o cálculo de tabelas balísticas. Foi inaugurado em fevereiro de 1946, era uma calculadora universal programável e eletrônica, pesava cerca de 30 toneladas, com 1500 relés, 17 mil válvulas, e 150 kW de potência. Tanto as operações aritméticas quanto as de armazenamento de dados eram conduzidas eletronicamente.

No final da década de 40, Claude Shannon, um estudante do MIT, em sua tese de mestrado criou as operações lógicas usando código binário.

Já no início da década de 50, várias máquinas foram construídas. Elas eram todas diferentes, mas todas seguiam a chamada arquitetura de Von Neumann, delineada nos primeiros trabalhos sobre a construção de computadores digitais. Nesta década também surgem as quatro primeiras máquinas a transistores SEAC, TRANSAC S100, Atlas Guidance Model I e CDC 1604, todas de construção americana.

Os circuitos integrados foram desenvolvidos e aperfeiçoados nos anos 60 sob influência do programa espacial americano, possibilitando o surgimento de minicomputadores que eram mais poderosos e bem menores.

Em 1971, a Intel lança o primeiro microprocessador. Durante esta década também foram desenvolvidos grandes computadores chamados de mainframes e surgiu o primeiro computador pessoal, o Apple II, feito em 1976 pelos americanos Steven Jobs e Stephan Wozniak.

A IBM lançou o PC/XT com um disco rígido de 110Mbits em 1983.

DOS, Windows, MAC, Linux, Internet, Ethernet, SUN, AIX, SNMP, ADSL, Wireless, Access point, iPhone, Cloud Computing...

1.2 Histórico da Comunicação de Dados

A comunicação de dados começou com a invenção do telégrafo por Samuel F. Morse em 1838. As mensagens eram codificadas em cadeias de símbolos binários e então transmitidas manualmente por um operador através de um dispositivo gerador de pulsos elétricos. Foram implantadas cerca de 40 milhas de linha para telégrafo em 1844.

Usando as linhas de telégrafo, em 1860, realizou-se a transmissão de 15bits/s.

Dados de radar, codificados em binário, foram transmitidos via facilidades de telégrafo para computadores na década de 1940. Usavam a *Bell System* (linhas e troncos) “*Common carriers*”.

“*Teletypewriter*” – As “*common carriers*” tornaram disponíveis dispositivos de entrada/saída que poderiam ser usados para enviar informação escrita ou codificada sobre linhas telefônicas.

No final da década de 50 ocorreu a explosão de desenvolvimentos para facilitar o uso de computadores remotamente.

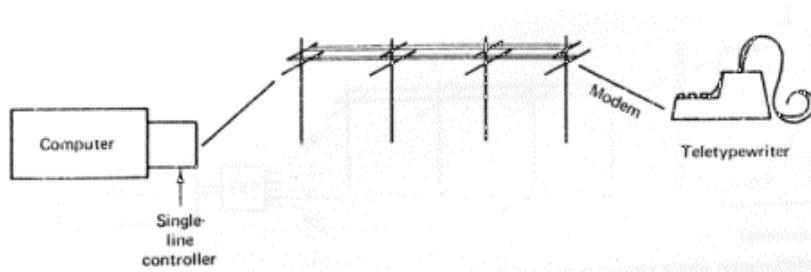


Figura 1.1 - Teletypewriter

Os primeiros terminais interativos foram desenvolvidos na década de 60 e permitiam aos usuários acessar o computador central através de linhas de comunicação. Nesta época surgiram os sistemas de Time Sharing (tempo compartilhado).

Fornecimento de computadores com interfaces (terminais, modems e linhas analógicas) para “batch processing”. O computador roda no modo “batch” em uma parte do dia. Na outra parte do dia o computador coleta informações de locais remotos.

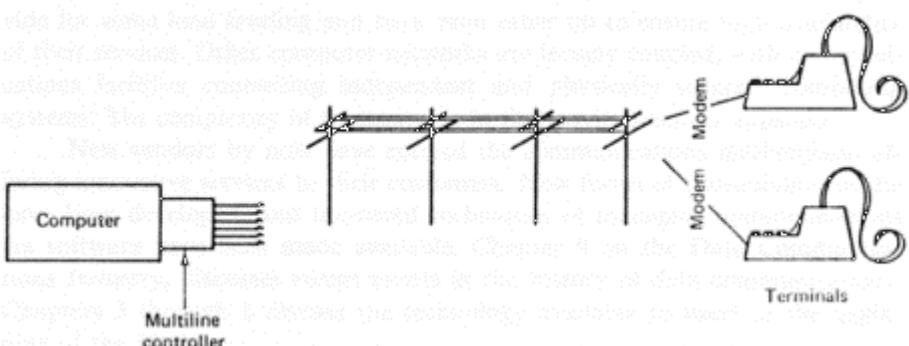


Figura 1.2 - Computador com controlador para multilinhas (multiponto)

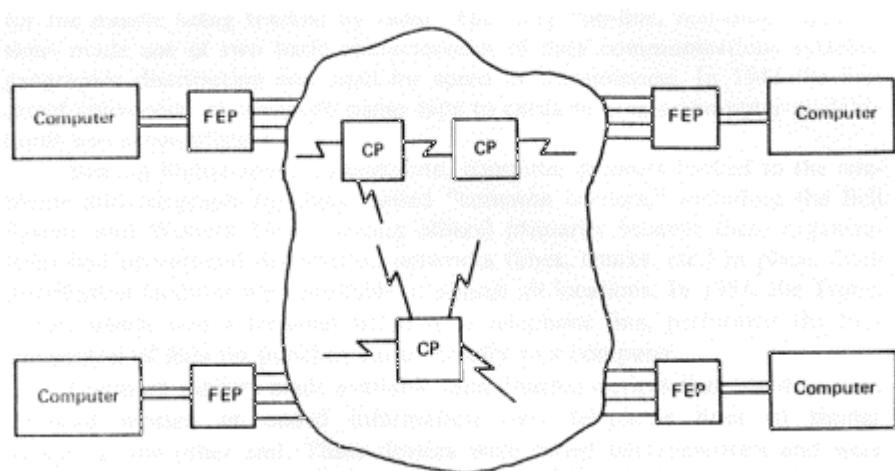


Figura 1.3 - Redes em tamanho, número de terminais e complexidade

Uma das primeiras redes de computadores, experimentais, entrou em operação entre 1969 e 1970, através do projeto ARPANET estimulado pela ARPA (Advanced Research Projects Agency) [Agência para Projetos de Pesquisa Avançados] que

passou a se chamar DARPA e faz parte do Departamento de Defesa dos Estados Unidos. Quatro Universidades americanas foram interligadas por esta rede que utilizava a tecnologia de comutação de pacotes. A primeira demonstração pública do sistema ocorreu em 1972 e a primeira conexão internacional da ARPANET ocorreu em 1973 com a University College of London na Inglaterra.

A idéia do projeto ARPANET era a construção de um sistema de comunicações que não pudesse ser interrompido por avarias locais. Essa preocupação era devido à guerra fria que estava no seu auge. Os militares americanos queriam uma rede de telecomunicações que não possuísse uma central e que não pudesse ser destruída por nenhum ataque localizado.

Anos mais tarde, o DCA (Defense Communication Agency) dividiu a ARPANET, formando duas redes, sendo que a parte maior ficou conhecida como MILNET e a outra parte ficou reservada para futuras pesquisas.

1.3 Histórico e Escopo da Internet

Em meados de 1979, a ARPA criou o ICCB (Internet Control and Configuration Board) que reunia pesquisadores envolvidos no desenvolvimento do TCP/IP (Transmission Control Protocol)/(Internet Protocol). No início dos anos 80, surgia a Internet, a partir do momento em que a ARPA passou a adotar os novos protocolos TCP/IP nas máquinas de sua rede de pesquisa. A partir disso a ARPANET se tornou o “backbone” da Internet.

Para incentivar pesquisadores das universidades a adotar e usar os novos protocolos, o DARPA tornou disponível uma implementação de baixo custo. Nesta época, muitas universidades usavam o sistema operacional UNIX disponível na University of California: *Berkeley Software Distribution*, também chamado Berkeley UNIX ou BSD UNIX. A ARPA conseguiu atingir cerca de 90% dos departamentos de ciência da computação das universidades com a integração do TCP/IP ao BSD UNIX.

Um usuário experiente de UNIX pode facilmente aprender a fazer uma cópia remota de arquivo (rpc), idêntico a cópia do UNIX.

O BSD UNIX forneceu uma nova abstração do sistema operacional conhecida como *socket*, que permitem aos programas de aplicação acessar os protocolos de comunicação.

Em 1986, o NSF (National Science Foundation) financiou várias redes regionais para se conectarem com as principais instituições voltadas para pesquisa científica e integrarem a Internet.

Inicialmente os nomes e endereços de todos os computadores ligados a Internet eram mantidos em um arquivo e editados mensalmente. Já em 1985 um banco de dados central já não seria o suficiente.

Um novo mecanismo foi desenvolvido, o *Domain Name System* (com máquinas chamadas name servers → servidores de nomes).

1986 – 20.000 computadores ligados a Internet;

1987 - taxa de crescimento de 15% ao mês;

1990 – 200.000 computadores;

1994 – 3.000.000 de computadores conectados à Internet em 61 países.

No Brasil, as redes começaram a surgir em 1988, ligando universidades e centros de pesquisa do Rio de Janeiro, São Paulo e Porto Alegre a instituições nos Estados Unidos. No mesmo ano, o Ibase (Instituto Brasileiro de Análises Econômicas e Sociais), começou a testar o AlterNex, o primeiro serviço brasileiro de Internet não-acadêmica e não-governamental que, em 1992, seria aberto ao acesso público.

A RNP (Rede Nacional de Pesquisas) surgiu em 1989 para unir as redes que ligavam as universidades e centros de pesquisas e formar um backbone de alcance nacional.

As diretrizes técnicas, a coordenação das pesquisas e desenvolvimento dos protocolos TCP/IP foi realizado pelo IAB (Internet Architecture Board) que surgiu em 1983 quando a ARPA reorganizou o ICCB.

A partir de 1989 a IAB passou a se encarregar dos aspectos políticos e comerciais do binômio TCP/IP - Internet. Os dois principais grupos do IAB são:

IRTF - *Internet Research Task Force* (grupos de pesquisa), coordena as atividades de pesquisa do TCP/IP; e

IETF - *Internet Engineering Task Force* (grupos de trabalho), concentra problemas de engenharia.

Os relatórios técnicos da documentação de trabalhos na Internet, proposição de protocolos novos ou revisados e padrões dos protocolos TCP/IP são chamados de RFCs (Requests for Comments). Anteriormente foram publicados os IENs (*Internet Engineering Notes*). INTERNIC (Internet Network Information Center) trata de muitos detalhes administrativos para a Internet e distribui RFCs e IENs.

1.4 Histórico de Telecomunicações

O histórico das telecomunicações pode ter começado com os sinais de fogo e fumaça na pré-história, batidas em troncos e tambores, pombos correio... No link <http://www.telephonetribute.com/timeline.html> aparece uma lista dos principais fatos com as datas de ocorrência ao longo da história da humanidade.

As redes de telecomunicações sofreram uma grande evolução desde os tempos de Alexander Graham Bell até os nossos dias. Passamos de redes analógicas comutadas manualmente às modernas centrais digitais com transmissão através de cabos de fibra ótica.

Para cada tipo de serviço especializado (telefonia, telex, comunicação de dados, etc.) criaram-se redes dedicadas, onde em geral, apenas os meios de transmissão de longa distância são compartilhados. Deste modo, chegamos ao cenário da figura abaixo onde um usuário corporativo necessita contratar diversos serviços a, possivelmente, fornecedores diferentes para atender às diversas necessidades de comunicação de sua empresa.

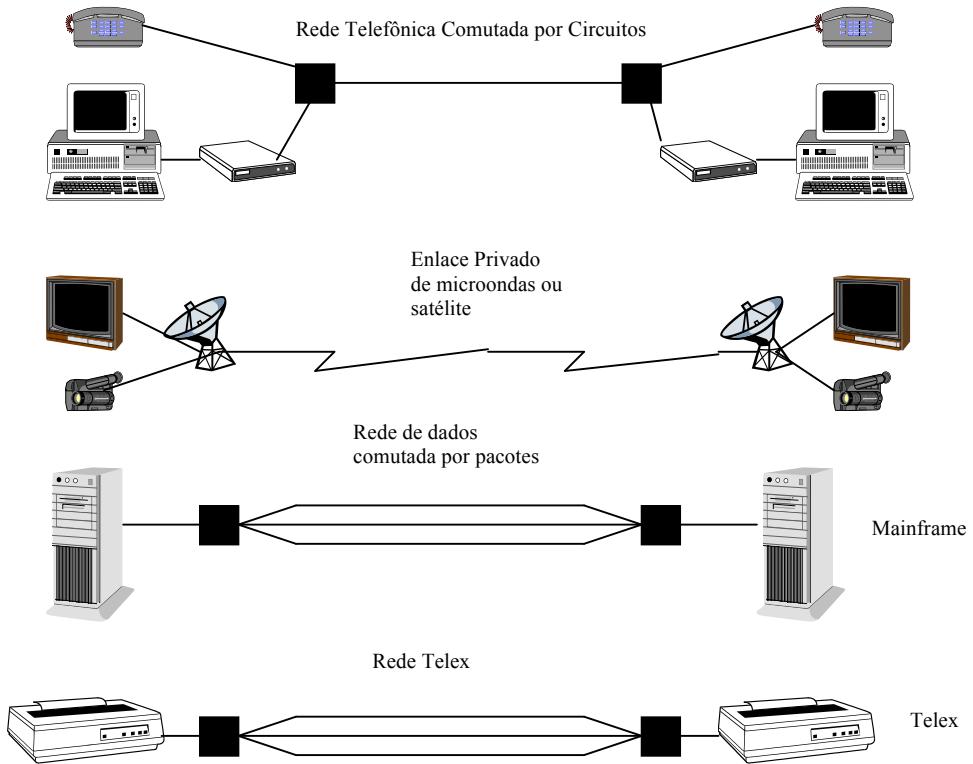


Figura 1.4 - Redes de comunicação antes da RDSI (Redes Digitais de Serviços Integrados)

A rede telefônica utiliza uma técnica conhecida como comutação de circuitos onde canais de voz, com largura de faixa de 4kHz, são alocados de forma dedicada ao longo do percurso entre os terminais chamador e o chamado, enquanto durar a conexão (chamada telefônica). Apesar de boa parte dos canais de comunicação entre as centrais, assim como a própria central de comutação, serem digitais, os acessos aos usuários são ainda na sua maioria analógicos. Deste modo, equipamentos como computadores necessitam transmitir os seus dados digitais analogicamente, através de modems. Posteriormente este sinal analógico será codificado digitalmente nas centrais para transmissão na rede telefônica digital. Na central digital destino, ele é decodificado para analógico para ser entregue ao usuário remoto, onde é demodulado para digital (pelo modem) para que o computador destino possa processar a informação recebida. Veja figura abaixo.

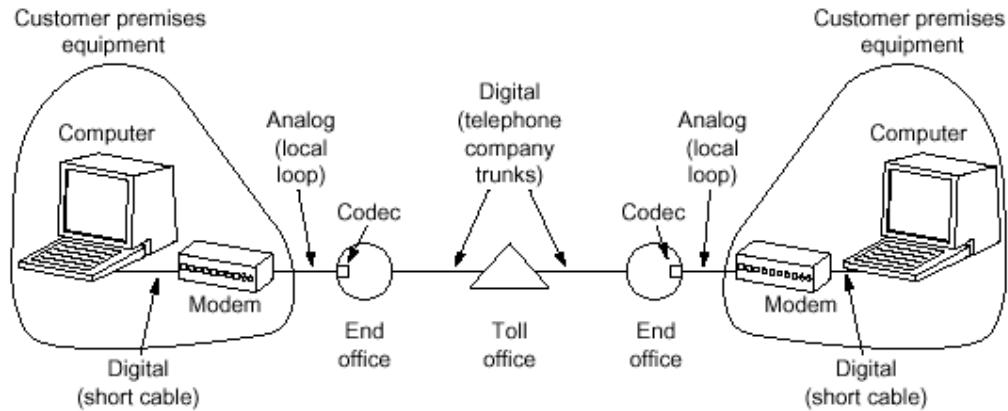


Figura 1.5 - Uma chamada entre dois computadores

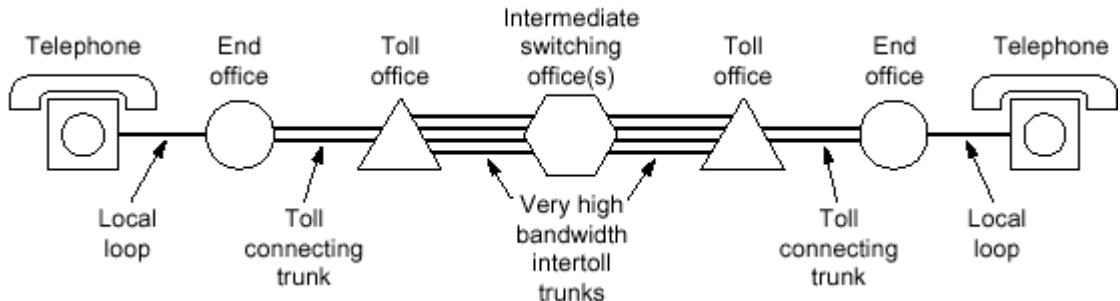


Figura 1.6 - Rota de um circuito típico de uma chamada a média distância

É realizada uma multiplexação por divisão no tempo (TDM - Time Division Multiplexing) entre as estações final (End office), interurbana (Toll office) e de comutação intermediária (Intermediate Switching Office), sendo que todo o tráfego é realizado através de fibra ótica e a largura de banda sempre aumenta no sentido da estação de comutação intermediária.

Nos primórdios da fibra ótica, cada companhia telefônica tinha seu próprio sistema ótico TDM. Assim em 1989 a CCITT divulgou um conjunto de recomendações chamado SDH (Synchronous Digital Hierarchy - Hierarquia Digital Síncrona) juntamente com o padrão SONET (Synchronous Optical NETwork- Rede Óptica Síncrona). SDH e SONET se diferem em poucos aspectos.

Em 1865 representantes de diversos governos europeus se reuniram para formar o CCITT (Comitê Consultivo Internacional de Telegrafia e Telefonia), predecessor do atual ITU (International Telecommunication Union). A missão do CCITT era padronizar as telecomunicações internacionais, até então dominadas pelo telégrafo. Em 1947 o ITU tornou-se um órgão das Nações Unidas. A tarefa do ITU-T (T – Telecommunications) é definir recomendações técnicas para interfaces de telefonia, telégrafos e comunicação de dados (recomendações que se transformam em padrões internacionais).

1.5 Histórico de Redes Sem Fio

As redes *ad hoc* tiveram seu início com as pesquisas realizadas na década de 70 pela United States Defense Advanced Research Projects Agency (U.S DARPA) com o projeto Packet radio network (PRNET).

Tal projeto tinha como intuito explorar e ampliar os usos e aplicações de redes de pacote de rádio em um ambiente tático militar com o objetivo principal de melhorar a comunicação de dados.

Em 1983, a agência americana (U.S Darpa) fez o lançamento do programa Survivable Adaptive Network, conhecido como SURAN, o qual foi desenvolvido para ampliar a base tecnológica que havia sido desenvolvida no projeto PRNET.

O SURAN foi desenvolvido com o intuito de suportar grandes redes e também desenvolver protocolos de rede adaptativos os quais tivessem a capacidade de adaptar-se às rápidas e voláteis mudanças nas condições de ambientes táticos.

O GloMo (Global Mobile Information Systems) foi o ultimo programa iniciado pela U.S DARPA em 1994, e foi desenvolvido para satisfazer os requisitos de defesa para sistemas de informações consistentes e com capacidade ampla de expansão.

Com o passar do tempo as aplicações das redes *ad hoc* ultrapassaram o uso meramente militar e passaram a servir a outros fins como busca e salvamento de pessoas, conferências entre outras aplicações.

Em meados da década de 1990 a IEEE (Institute of Electrical and Electronics Engineers) recebeu a tarefa de elaborar um padrão de LANs sem Fios. Ao invés do caro espectro licenciado, os sistemas IEEE 802.11 operam nas bandas não licenciadas, como as bandas ISM (Industrial, Scientific, and Medical) definidas pelo ITU-R. O padrão inicial (de 1997) definia uma LAN sem fios que funcionava a 1 ou 2 Mbps. Em 1999 o 802.11b funcionava a 11 Mbps. Em 2003 aumentou a velocidade do 802.11a/g para 54 Mbps. Em 2009 o 802.11n pode chegar até 450 Mbps.

Capítulo

2

Transmissão de Dados

2.1 Transmissão Analógica e Digital

Na transmissão analógica os sinais elétricos variam continuamente entre todos os valores possíveis, permitidos pelo meio de transmissão.

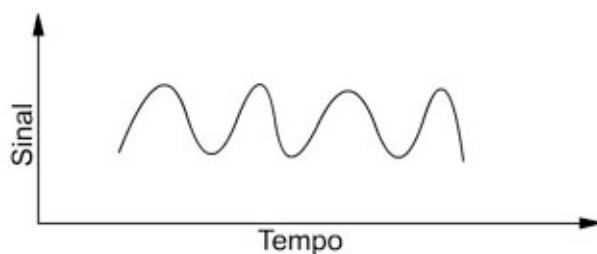


Figura 2.1 – Sinal de microfone de telefone

Na transmissão digital uma série de sinais que tem apenas dois valores elétricos (ou gama discreta de valores) é transmitida.

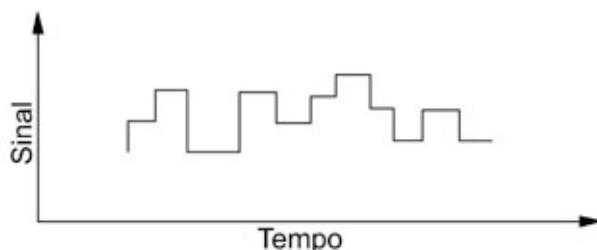


Figura 2.2 – Sinal de telegrafia (Morse)

Este último tipo de sinal é obtido pela rápida inversão do estado corrente.

	0	1	
Por emissão →	Ausência	presença	
Por interrupção →	Presença	ausência	
Por dupla corrente →	“-1”	“+1”	

CCITT – Comitê Consultivo Internacional de Telefonia e Telegrafia

EIA – Associação das Indústrias Eletrônicas

A transmissão digital é a que mais se adapta à forma binária de codificação usada em processamento eletrônico de dados.

2.2 Modulação

As limitações que meios de transmissão impõem aos sinais transmitidos podem ser contornadas através do processo de modulação, que consiste em se imprimir uma informação em uma onda portadora pela variação de um ou mais dos seus parâmetros:

- Amplitude

Varia a amplitude da portadora conforme a amplitude do sinal modulante. A principal vantagem dessa técnica é a facilidade para realizar a modulação e demodulação, entretanto tem duas grandes desvantagens: a velocidade da troca de amplitude é limitada pela largura de banda da linha e a outra é que pequenas mudanças da amplitude sofrem detecção não confiável. As desvantagens da modulação em amplitude fizeram com que esta técnica não fosse mais utilizada pelos modems, a não ser em conjunção com outras técnicas.

- Freqüência

A freqüência da onda portadora é alterada de acordo com o sinal modulante. Essa técnica também é conhecida como FSK (Frequency Shift Keying). As suas desvantagens são a limitação da variação da freqüência pela largura de banda da linha e a distorção causada pelas linhas torna a detecção mais difícil do que na modulação de amplitude.

- Fase

A fase da onda portadora varia de acordo com o sinal modulante. Essa técnica, para detectar a fase de cada símbolo, requer sincronização de fase entre receptor e transmissor, complicando o projeto do receptor.

A escolha do tipo de modulação de acordo com os requisitos e as necessidades do projeto é uma decisão fundamental para garantir em grande parte o êxito de um sistema de comunicação.

2.3 Demodulação

É o processo de recuperação da informação (extração do sinal modulante) de uma onda portadora, sendo necessário que o processo de modulação seja reversível.

2.4 MODEM

O MODEM (**M**ODulador-**D**EModulador) é o dispositivo que realiza a adequação dos sinais binários ao canal de transmissão, servindo de interface entre este canal e o terminal de dados e permitindo a transmissão de sinais a longa distância.

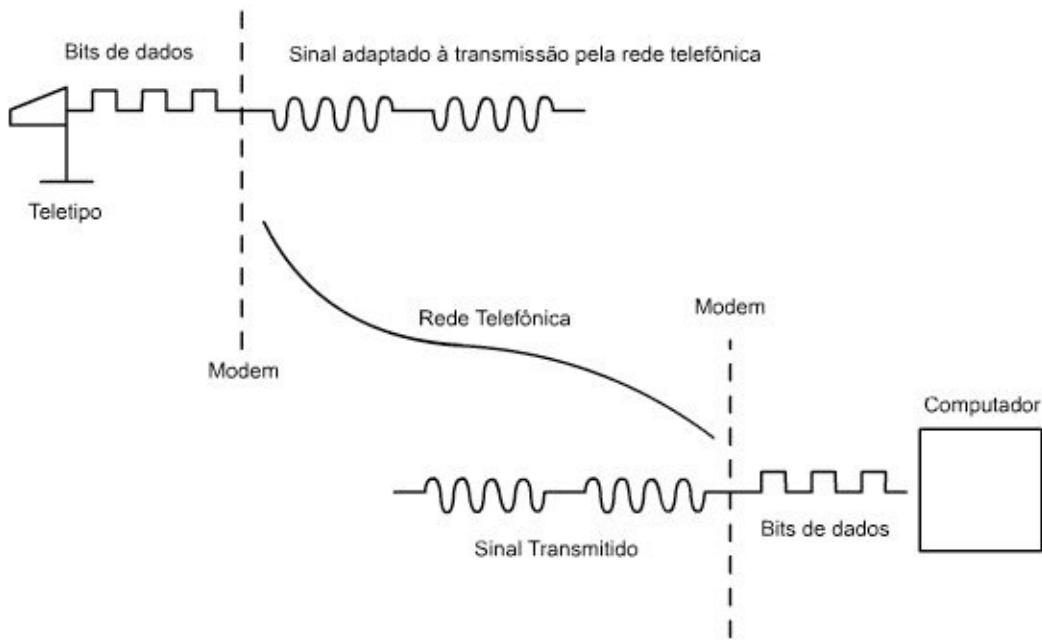


Figura 2.3 – Ligação ponto a ponto com Modem

Como a banda passante do sistema telefônico é de 300 Hz a 3300 Hz o espectro de freqüência do pulso deve ser deslocado para a parte do espectro disponível para transmissão.

A técnica de modulação permitiu diversificação de aplicações:

- Transmissão de várias informações no mesmo meio, utilizando freqüências diferentes;
- Fonia e telegrafia no mesmo canal; e
- Vídeo, voz e dados no mesmo canal.

A maioria dos modems em operação transmite uma onda portadora senoidal contínua que é modificada de acordo com os dados que devem ser enviados.

$$a = A \operatorname{seno} (2\pi f T + \theta)$$

a = Amplitude instantânea da portadora no instante T .

A = Amplitude máxima da portadora.

f = Freqüência da portadora.

θ = Fase da portadora.

Os valores A , f ou θ podem ser variados, para fazer a onda portar informação, como pode ser visto na figura abaixo.

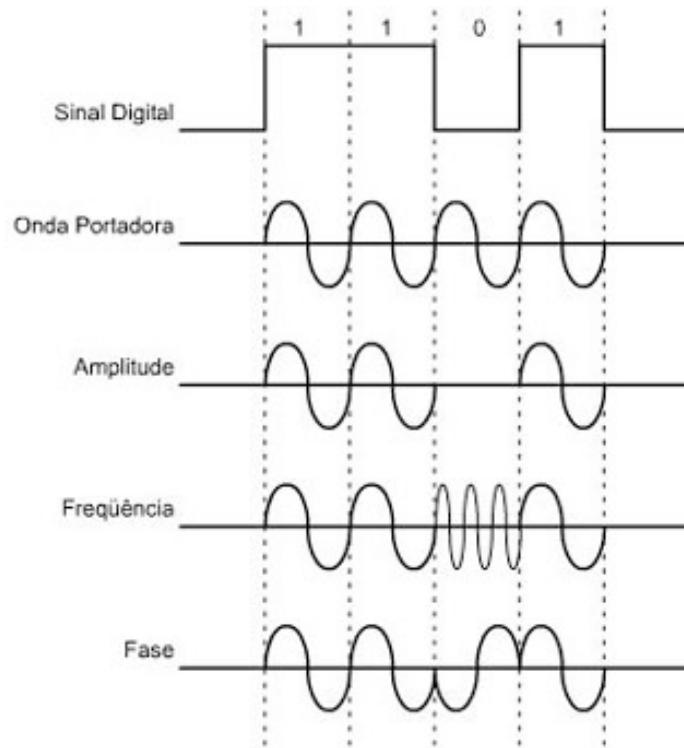


Figura 2.4 – Modulação em amplitude, freqüência e fase

O sinal modulador pode ser de natureza analógica ou digital.

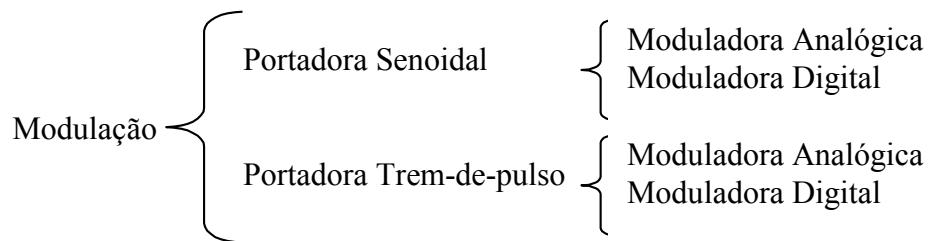


Figura 2.5 – Tipos de portadora e moduladora

2.5 Modulação em Portadora Senoidal

Quando a moduladora é analógica:

- AM (amplitude modulation) ou modulação em amplitude.
- FM (frequency modulation) ou modulação em freqüência.
- PM (phase modulation) ou modulação em fase.

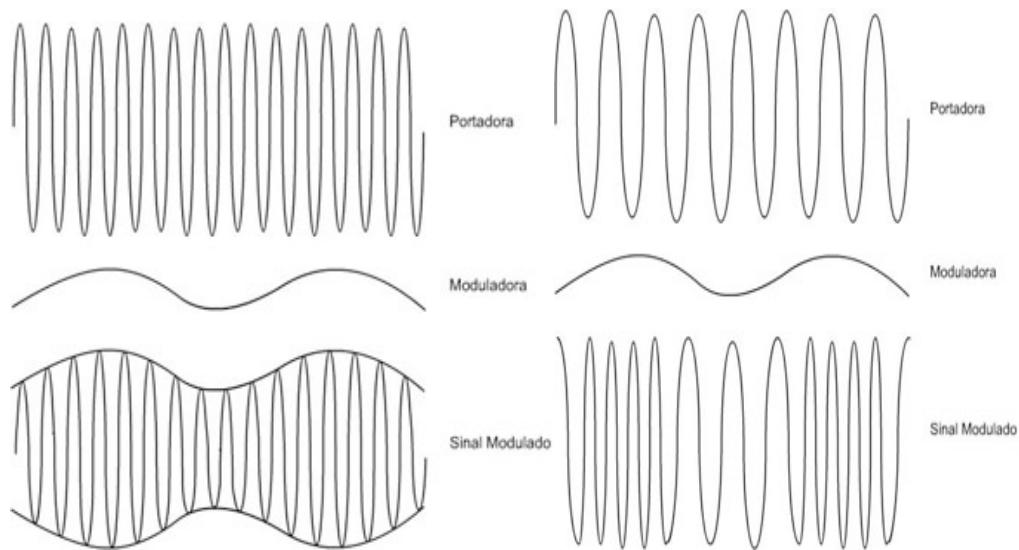


Figura 2.6 – AM e FM

Quando a moduladora é digital (a característica variada da portadora senoidal assume alguns valores descritos possíveis).

- a) ASK (amplitude shift keying) ou modulação por salto de amplitudes.
- b) FSK (frequency shift keying) ou modulação por salto de freqüência.
- c) PSK (phase shift keying) ou modulação por saltos de fases.

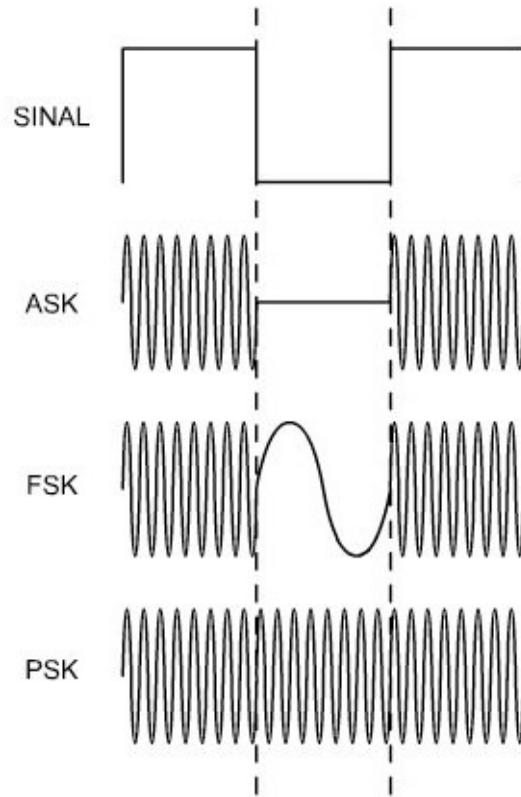


Figura 2.7 – ASK, FSK e PSK

2.6 Modulação em Portadora Trem-de-Pulsos

O trem de pulsos periódico é um sinal de natureza discreta, tem como características a amplitude, a duração e o período.

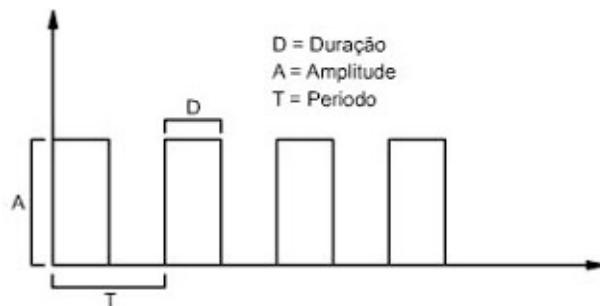


Figura 2.8 – Trem de Pulsos

Moduladora Analógica:

- PAM (pulse amplitude modulation): modulação de pulso em amplitude [informação impressa na característica de amplitude].
- PWM (pulse width modulation): modulação de pulsos em largura [informação impressa na característica de duração de pulsos].
- PPM (pulse position modulation): modulação em posição de pulso [informação representada pelos deslocamentos relativos dos pulsos em relação a referências de tempo igualmente espaçadas].
- PFM (pulse frequency modulation): modulação de pulsos em freqüência [informação representada pela quantidade de pulsos que ocorrem em cada intervalo de tempo].

Na modulação digital com portadora de trem-de-pulsos a informação é impressa indiretamente através de códigos.

- PCM (pulse code modulation): modulação em código de pulsos.
- DPCM: PCM diferencial. Informação é enviada por código representando a diferença dos valores da moduladora em instantes consecutivos de amostragem.
- DM (delta modulation): modulação delta.

Todos os três tipos põe no meio de transmissão um sinal pulsado e codificado.

PCM e DPCM → código diretamente relacionado com o valor absoluto da portadora.

O sinal de saída do DM é um sinal bit relacionado basicamente com a correção a ser imposta ao receptor distante para que siga o sinal e apenas indiretamente com a informação da moduladora.

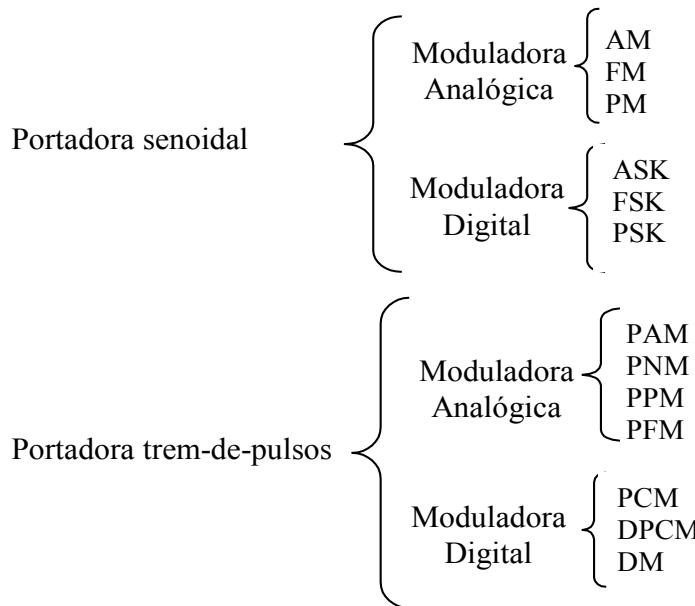


Figura 2.9 – Tipos de modulação

2.7 Linhas de Transmissão

Os três principais obstáculos que uma linha de transmissão oferecem ao envio de um sinal são: atenuação, distorção de retardo e ruído.

Uma fonte de energia num extremo e uma carga no extremo oposto, conectados pela linha de transmissão.

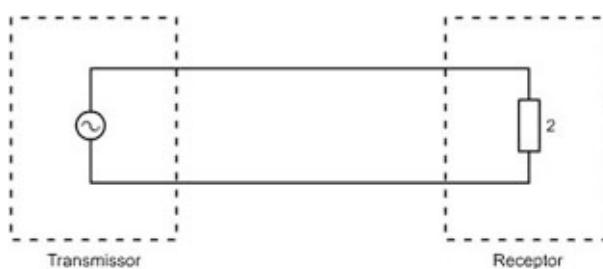


Figura 2.10 - Linha de Transmissão

Uma linha telefônica típica de 0,1 Km de comprimento pode ser simulada através do circuito abaixo, representando um filtro passa-baixa (circuito elétrico que permite apenas a passagem de sinais elétricos com freqüências abaixo de um determinado valor limite), no nosso caso com freqüência de corte de 3,3kHz.

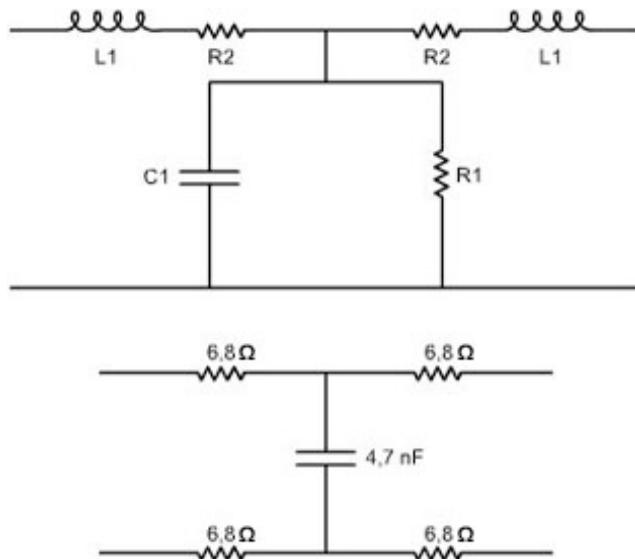


Figura 2.11 – Filtro passa-baixa representando a linha telefônica de 0,1 km

Distorção é uma mudança indesejada na forma de onda.

2.8 Distorção por Atenuação

O antigo CCITT definiu as características mínimas do canal de voz internacional, com qualidades especiais para transmissão de dados, através da recomendação M-102.

A distorção por atenuação (distorção de amplitude) consiste numa perda de energia a medida que o sinal se propaga e diminuição da relação sinal/ruído, dificultando a recuperação da informação transmitida na sua chegada.

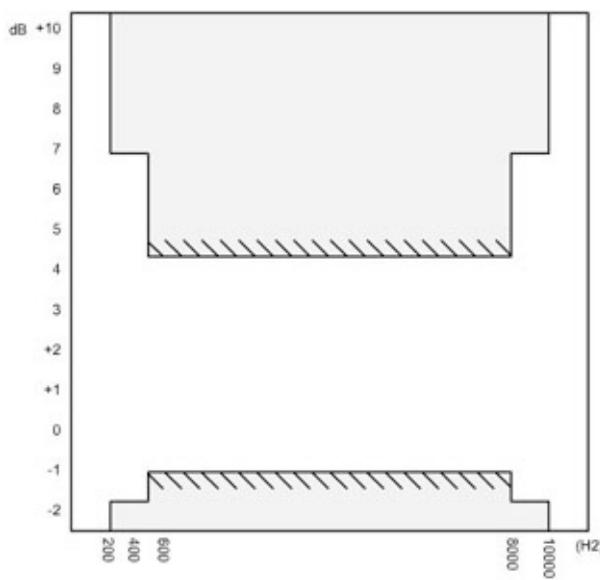


Figura 2.12 - Variação permitível da atenuação

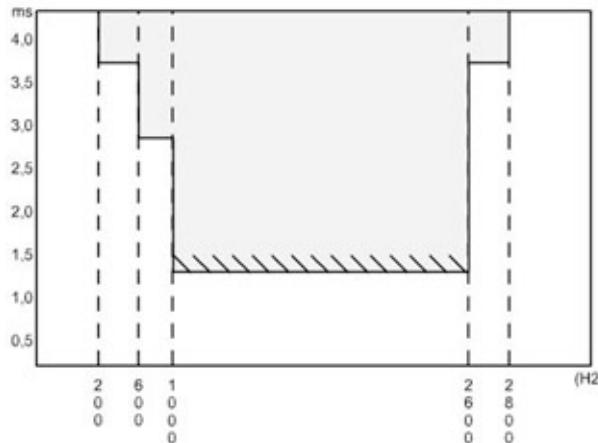
2.9 Distorção por ruído

Consiste na interferência provocada por uma energia indesejada proveniente de outras fontes, podendo ser ruído térmico, alterando as propriedades do meio de transmissão e, consequentemente, a qualidade do sinal; acoplamento indutivo podendo provocar linha cruzada em linha telefônica.

Quando os ruídos são adicionados a um sinal que contém informação, esta pode ser parcialmente mascarada ou totalmente eliminada. Dependendo do ruído pode ser difícil eliminá-lo, constituindo um dos problemas básicos da comunicação elétrica.

2.10 Distorção por retardo

Também conhecida como distorção de fase, onde o sinal é retardado mais em algumas freqüências do que em outras.



2.13 - Variação permissível do retardo

Caso a distorção do canal não se enquadre entre esses limites, deverão ser utilizados equalizadores que compensem a variação do tempo de retardo.

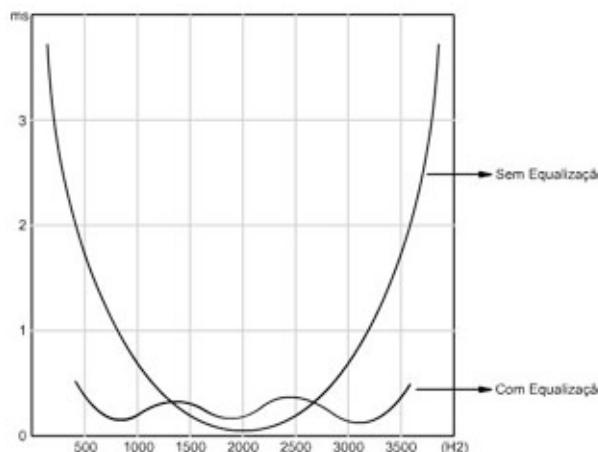


Figura 2.14 – Sinal com e sem equalização

Exemplo de uma distorção por retardo, supondo uma transmissão a 2000 bits/s (usando modulação por freqüência).

- “0” = a 1000 Hz
- “1” = a 3000 Hz
- retorno relativo $\cong 1$ ms
- bit “0” transmitido com mais rapidez acaba chegando ao receptor antes do bit “1” que o precede.

Para transmitir os sinais de dados que se apresentam na forma de pulsos retangulares necessitamos de uma largura de banda infinita. Como os canais possuem uma largura de banda finita, os sinais devem ser adaptados através de processos de codificação ou modulação, descritos acima.

Codificação \Rightarrow Banda Base (Modem banda base)

Modulação \Rightarrow Modem Analógico

Os sinais são normalmente definidos por suas características em função do tempo. Por outro lado, os sistemas físicos têm suas características dependentes das freqüências que os excitam (o sistema apresenta comportamento diferente e seletivo em relação à freqüência).

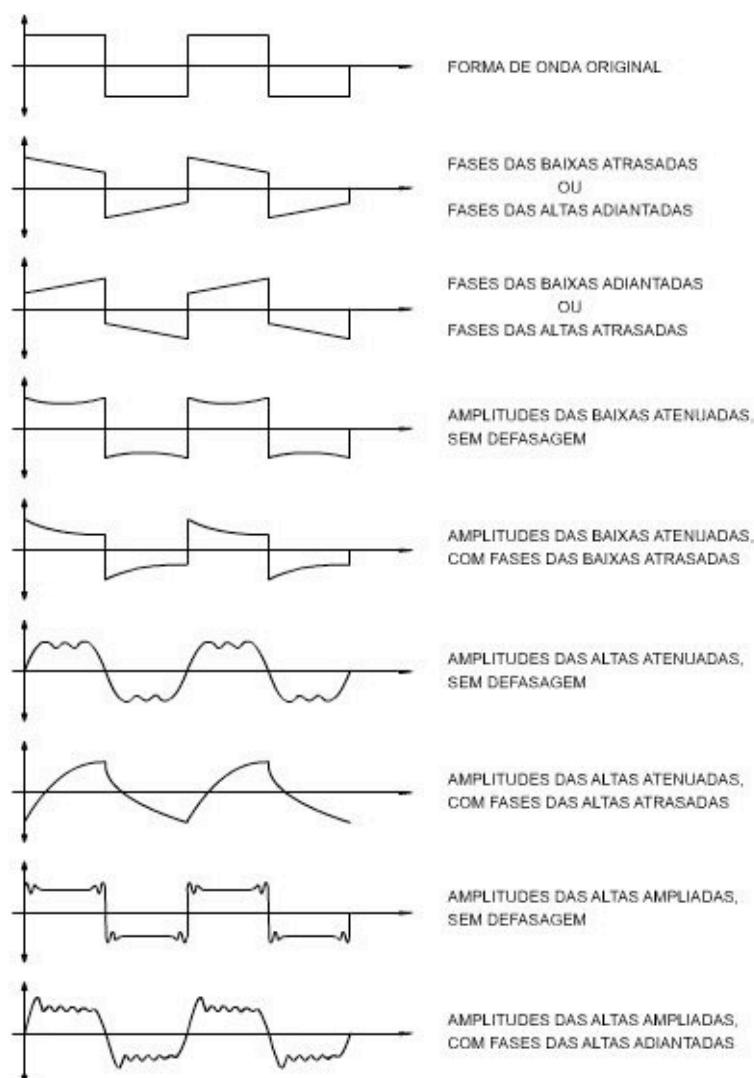


Figura 2.15 – Interpretação das distorções na onda quadrada

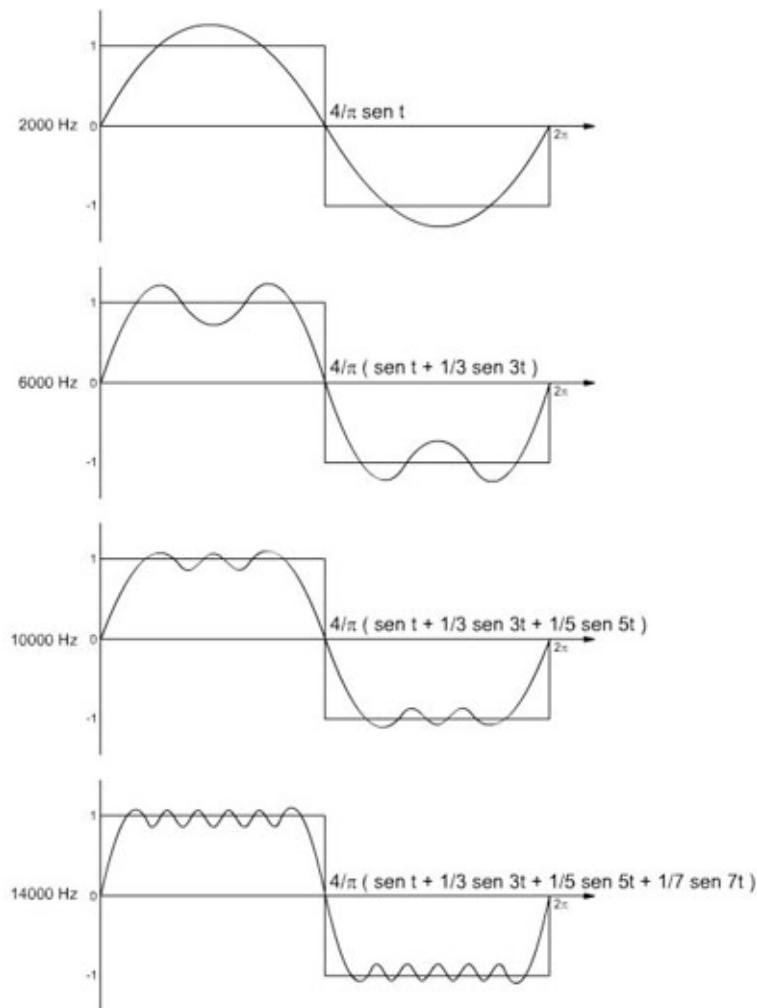


Figura 2.16 - Aproximação de uma função retangular por funções ortogonais

Considerando a passagem de um pulso retangular por um filtro passa-baixas ideal com freqüência de corte ($f_1 = 1/2T$). Lóbulo central de largura $2T$ e lóbulos laterais de largura T .

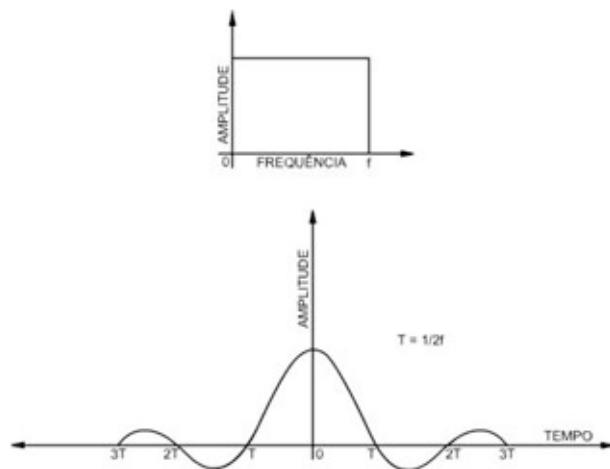


Figura 2.17 - Resposta de um filtro passa-baixa ideal a um pulso regular de duração T segundos

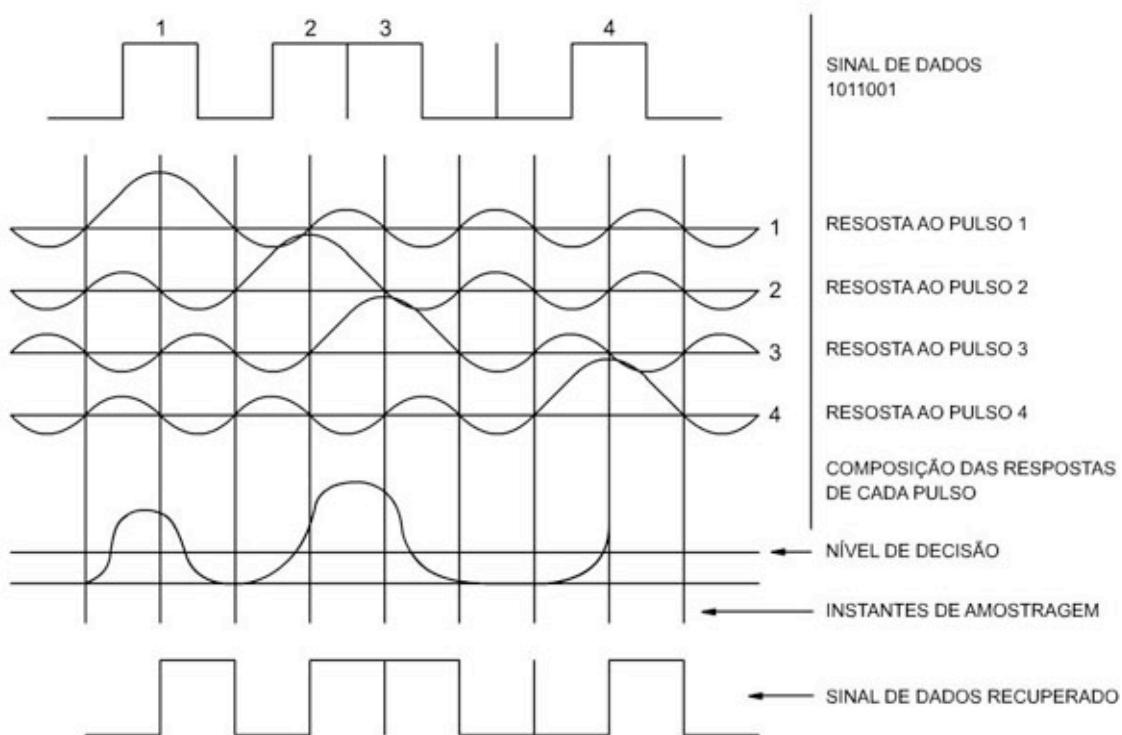
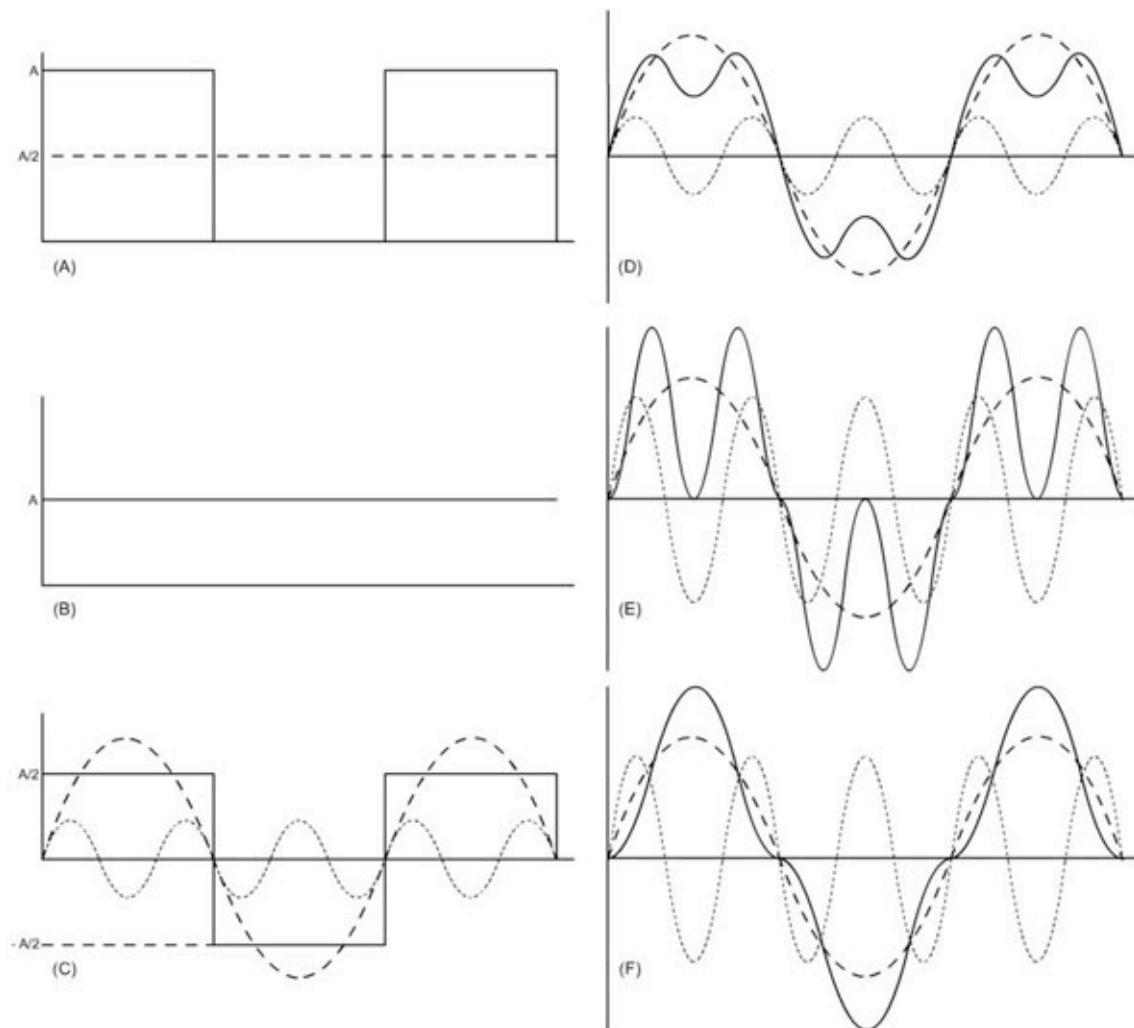


Figura 2.18 - Passagem do sinal de dados através de um filtro passa-baixa ideal e recuperação do sinal

Reconhecimento do sinal de saída:

- Analisando a cada T segundos a amplitude do sinal em comparação com o nível de decisão estabelecido. Caso a amplitude do sinal no instante de amostragem seja superior ao do sinal de decisão é reconhecido em pulso correspondente a 1 bit.
- Fixando-se um nível de decisão e verificando-se a ocorrência de transições neste nível. Estas transições são definidas quando o sinal ultrapassa nos sentidos positivos e negativos a amplitude no nível de decisão.



- (A) Trem-de-pulsos original (onda quadrada)
- (B) Componente constante
- (C) Componente simétrica do trem-de-pulsos e suas principais componentes senoidais
- (D) Composição da fundamental com a 3^a harmônica
- (E) Composição da fundamental com a 3^a harmônica, tomada com a mesma amplitude
- (F) Composição da fundamental com a 3^a harmônica defasada de π

Figura 2.19 – Diversas formas de onda

Capítulo

3

Sistemas de Comunicação

Arte do transporte de informação de um ponto a outro.

3.1 Modelo do Sistema de Comunicação

Para o tratamento conceitual seguro dos fenômenos em um sistema de comunicações é necessário se estabelecer um modelo simplificado que consiga descrever as suas características essenciais de funcionamento.

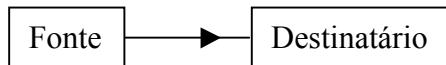
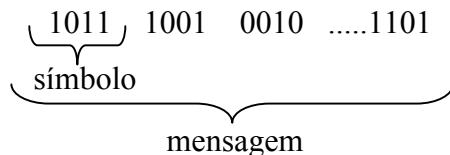


Figura 3.1 – Fonte e destinatário

A fonte produz informação, dispondo de elementos simples e símbolos. Destinatário é a quem a informação é dirigida.



Alfabeto de elementos: $a = \{0, 1\}$ $b = \{A, B, C\dots\}$. Símbolo: conjunto ordenado de elementos. O conjunto completo de símbolos forma o alfabeto de símbolos. Do alfabeto b pode-se compor os símbolos AA, AB, ... ou AAA, BBA... A saída da fonte será sempre de símbolos. A mensagem é o que a fonte produz, consistindo em um conjunto ordenado de símbolos que a mesma seleciona de seu alfabeto, conforme critérios próprios. A informação é considerada relacionada com a aleatoriedade no aparecimento dos símbolos. A cada símbolo é associado uma quantidade de informação que é função de sua probabilidade de ocorrência.

O problema básico consiste em estudar a maneira como serão transmitidos estes símbolos de modo que a informação associada não seja perdida nem alterada.

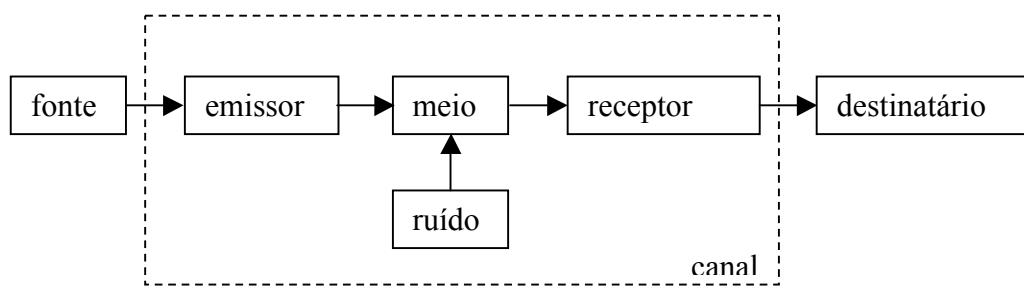


Figura 3.2 – Fonte, canal e destinatário

Canal: transporta os símbolos e a informação associada da fonte ao destino. O Emissor: entrega um sinal de energia adequado (modulador). O Meio: propaga a energia entregue pelo emissor até o receptor. O Receptor: retira a energia do meio e recupera os símbolos (demodulador). Problemas: ruídos e distorção.

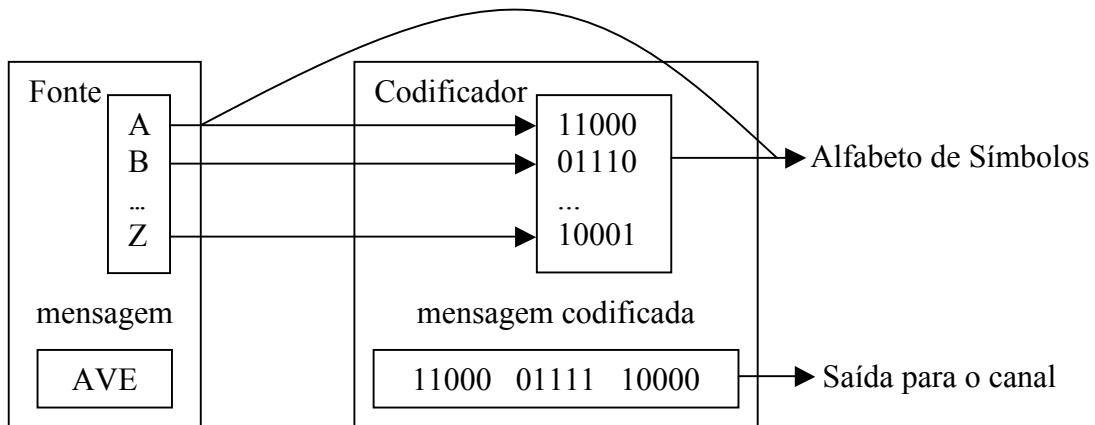


Figura 3.3 – Codificador e decodificador

Frequentemente a natureza dos símbolos gerados pela fonte não é adequada para acionar o canal de transmissão. Devemos então alterar a natureza dos elementos que é feita pela codificação. Decodificador: acoplador de informação entre o canal e o destinatário.

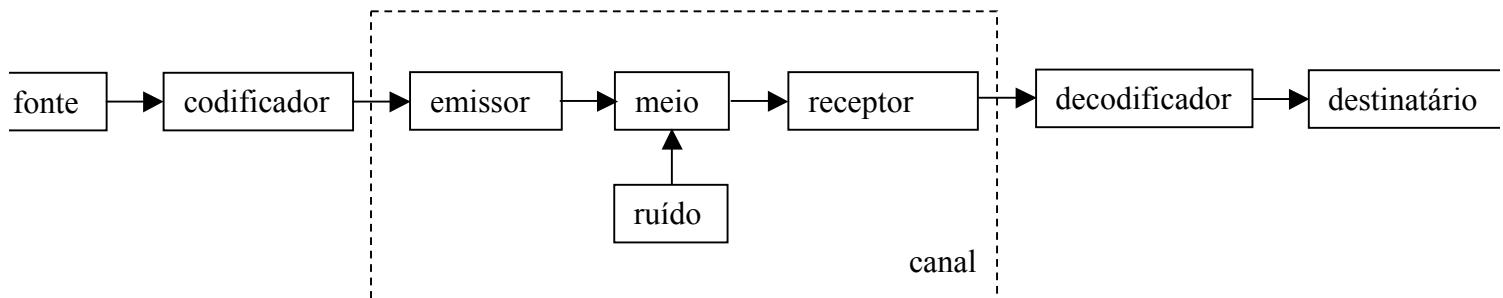


Figura 3.4 – Modelo completo de um sistema de comunicações

Toda fonte de informação dispõe de um conjunto de símbolos. No mecanismo de seleção sucessiva de símbolos, para a composição de mensagem, se produz a informação.

$$N = n^m$$

Alfabeto de “n” elementos. Símbolos compostos de “m” elementos dentre os “n” citados. Número “n” de configurações possíveis arrumando-se “n” elementos tomados “m” a “m”.

Exemplo: {1, 0}, n = 2, m = 5 e N = 32.

Variedade de um símbolo é número total de elementos que compõe as configurações possíveis que pode assumir este símbolo.

$$V = N \quad ?$$

N varia exponencialmente (manejo incômodo). Empregando logaritmo passo de lei exponencial para linear.

$$V = \log_2 N = m \log_2 n$$

Base dois expressa a variedade na unidade “bit”.

3.2 Bit, Dibit e Tribit

O pulso binário apresenta a variedade de 1 bit. O bit é um pulso que apresenta 2 níveis distintos {0, 1}.

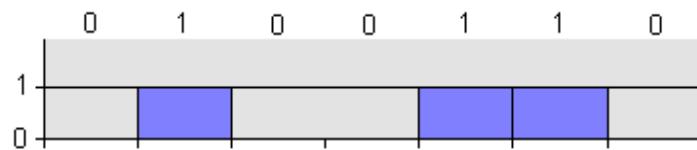


Figura 3.5 - Codificação Bit

$$V = \log_2 2 = 1 \text{ bit}$$

O dibit é um pulso que apresenta 4 níveis distintos, portanto a cada pulso podem ser transmitidos dois bits.

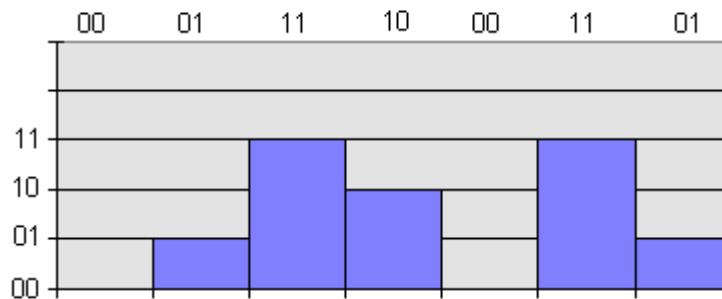


Figura 3.6 - Codificação Dibit

$$V = \log_2 4 = 2 \text{ bits}$$

O tribit é um pulso que apresenta 8 níveis distintos.

$$V = \log_2 8 = 3 \text{ bits}$$

3.3 Baud

Se um sinal binário é transmitido na velocidade de sinal V_s (bit/s), exigindo uma faixa B do sistema, poderá ser transmitido com a mesma velocidade na faixa $B/2$ se modificado para dibit e na $B/3$ se transformado em tribit. O número de amostras por segundo é medido em baud. Durante cada baud é modulado um símbolo. Desse

modo, uma linha de n bauds transmite n símbolos/s. A faixa de passagem mínima B (ou frequência da fundamental) que o sistema deve possuir para permitir o reconhecimento de uma seqüência de pulsos entrantes de duração T, deve deixar passar a fundamental do padrão.

$$B = \frac{1}{2T}$$

$$Vm = \frac{1}{T} \text{ baud}$$

 o inverso do menor tempo presente no sinal.

$$Vm = 2B$$

Para bit \rightarrow 2 níveis:

$$Vs = \frac{1}{T} \text{ bit/s}$$

$$\frac{Vs}{Vm} = 1 \frac{\text{bit/s}}{\text{baud}}$$

Para dabit \rightarrow 4 níveis:

$$Vs = \frac{2}{T} \text{ bit/s}$$

$$\frac{Vs}{Vm} = 2 \frac{\text{bit/s}}{\text{baud}}$$

Para tribit \rightarrow 8 níveis:

$$Vs = \frac{3}{T} \text{ bit/s}$$

$$\frac{Vs}{Vm} = 3 \frac{\text{bit/s}}{\text{baud}}$$

Como B relaciona com Vm, podemos concluir que usando a mesma largura de faixa B, ao se transmitir o sinal dabit se consegue o dobro de Vs e com o tribit o triplo de Vs. O canal telefônico tem uma faixa de passagem de 3.100Hz e na prática é usado para transmissão de sinais de dados até 2.400 baud. Por isso quando se quer transmitir à velocidade de sinalização de 4.800 bit/s neste canal, se usa um sinal dabit ao qual corresponde a velocidade de modulação de 2.400 baud. De modo similar, para se transmitir em sinal com Vs = 7.200 bit/s se emprega o sinal tribit com a Vm de 2.400 baud. É necessário então, codificar o sinal de modo a casar as necessidades de transmissão de informação com as necessidades de transmissão de sinal no meio.

Exercício:

Um modem converte um sinal de 9.600 bit/s num sinal quadribit. Quantos bauds tem a saída do modem? Qual a freqüência da fundamental desta saída? Este sinal é apropriado para transmissão em canal telefônico?

Resposta:

$$V_s = 9.600 \text{ bit/s} - 16 \text{ níveis}$$

$$V_s = 4V_m$$

$$V_m = 2.400 \text{ bauds}$$

$$V_m = 2B$$

$$2.400 = 2B \rightarrow B = 1.200 \text{ Hz}$$

Sim. Por que o valor da frequência da fundamental está entre 300Hz e 3300Hz que é a faixa de passagem do canal telefônico.

3.4 Informação e sua Medida

Quanto maior a probabilidade menor a dúvida e menor a informação adquirida. Quantidade de informação própria de um símbolo x_i .

$$I(x_i) = f[P(x_i)]$$

Informação → Conjunto. Encontrar uma função f adequada. Propriedades da função:

a) Se $p(x_i) = 1$, então $I(x_i) = 0$

b) Se $p(x_i) = 0$, então $I(x_i) = \infty$

c) $I(x_i)$ é decrescente com $p(x_i)$

A função $I(x_i) = -\log_2 p(x_i)$ satisfaz a todos os requisitos.

$$I(x_i) = -\log_2 p(x_i) \text{ shannon}$$

A probabilidade de uma condição:

$$p = \frac{n^o \text{ de eventos favoráveis}}{n^o \text{ de eventos possíveis}} = \frac{nf}{np}$$

Exercício:

Seja o experimento “lançar um dado para tirar a face 6”. São eventos possíveis as faces 1, 2, 3, 4, 5 e 6. O único evento favorável é a face 6. A probabilidade de sair 6 é:

$$p(6) = \frac{1}{6} = 0,16666\dots$$

Para a língua inglesa foi feito um estudo de probabilidade de ocorrência das letras, a mais freqüente sendo a letra “e” com a probabilidade de 0,131 e a menos freqüente, a letra “z” com a probabilidade de ocorrência de 0,00077.

Quando num texto aparece a letra “e”, ela carrega uma informação própria

$$(e) = -\log_2^{0,131} = 2,95 \text{ shannon}$$

Já quando aparece a letra “z” a informação própria do símbolo é:

$$(z) = -\log_2^{0,00077} = 10,36 \text{ shannon}$$

Por ser mais rara, isto é, menos provável, a letra “z” traz mais informação própria.

3.5 Sistemas de Telecomunicações e Teleprocessamento

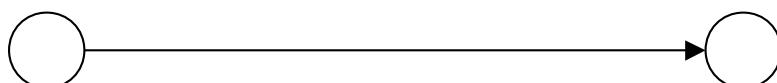
Palavra	<u>Sinal</u>	<u>Tecnologia</u>
	escrita	Telegrafia
	Falada	Telefonia
Imagen	estática	Facsímile
	Dinâmica	Televisão
Dados		Comunicação de Dados

Teleprocessamento é o processamento eletrônico de dados executado remotamente. É implementado por hardware e software voltados para a comunicação e por todo um conjunto de regras que disciplinam esta relação.

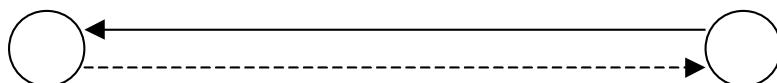
3.6 Características do Canal

- a)Operação simplex – quando a informação precisa ser transmitida apenas do ponto ao ponto B.
- b)Operação semi-duplex – quando a informação precisa ser transmitida em ambos os sentidos, de modo alternado.
- c)Operação duplex – quando a informação precisa ser transmitida em ambos os sentidos, de modo simultâneo.

Simplex



Half-Duplex



Full-Duplex

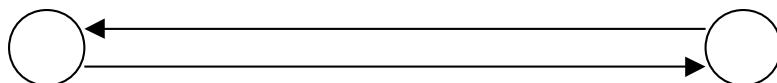


Figura 3.7 – Operações simplex, half-duplex e full-duplex

Pergunta: Você pode ter operação full-duplex a dois fios?

Capítulo

4

Erros

Um circuito será considerado satisfatório para transmissão de dados se, em cada 10^5 bits transmitidos apenas 1 bit errado ocorrer, segundo as recomendações do CCITT.

Considerações sobre erros

- São inevitáveis em qualquer sistema de comunicação real. Os erros são raros na parte digital da infraestrutura de transmissão. A parte analógica, de pares de fios trançados é, porém, predominante e deve continuar assim por muito tempo ainda devido ao alto custo envolvido na sua substituição.
- Sua ocorrência não é homogeneamente distribuída ao longo da transmissão – ocorre em rajadas, normalmente de comprimento inferior a 8 bits errados. Atualmente, à velocidade maior de transmissão, o *ruido* pode atingir um número maior de bits.
- É importante, na escolha de um código de transmissão de dados, considerar as características dos canais utilizados para incluir maior ou menor redundância, a fim de assegurar a confiabilidade da informação recebida.

$$\text{Eficiência} = \frac{\text{bits de informação transmitidos}}{\text{total de bits transmitidos}} * 100\%$$

- Esta relação mostra um compromisso entre eficiência e confiabilidade
- Quanto mais confiável o esquema de detecção/correção de erros, mais cara será sua implementação

Abordagens possíveis em tratamento de erros

Incluir informações redundantes suficientes em cada bloco de dado enviado, de forma que o receptor seja capaz de deduzir qual deveria ser o caractere transmitido é uma das duas estratégias básicas para tratar erros. Emprega *códigos de correção de erros*.

Outra estratégia, implementada via *códigos de detecção de erros*, consiste na inclusão de redundância suficiente para permitir que o receptor deduza que houve um erro e peça a retransmissão do bloco, sem compromisso com a identificação do erro.

Ex: Cada *bit* poderia ser transmitido duas vezes permitindo detectar a presença de um erro na desigualdade do par. Desta forma, no entanto, havendo 2 *bits* sucessivos errados, o erro poderia não ser detectado. Além disso a quantidade de bits úteis transmitidos, neste caso cai para a metade do total.

Detecção e retransmissão

Sobre os blocos de dados enviados são aplicados algoritmos que geram *bits* redundantes. Estes bits são também transmitidos, permitindo verificar, no destino, se os *bits* de informação estão chegando sem alteração.

Se o canal for muito suscetível a ruídos, será necessário usar maior redundância e algoritmos mais complexos para detectar e neutralizar os erros.

Estes algoritmos, na sua forma mais simples, constituem os testes de paridade. Veremos a seguir outras técnicas utilizadas nesta modalidade de tratamento de erros.

Paridade Vertical (Vertical Redundancy Checking) – VRC

Esta técnica consiste na adição de um bit de controle para cada caracter transmitido.

Tomando o código ASCII como exemplo, os sete primeiros *bits* de cada conjunto de oito se referem ao caractere alfanumérico transmitido, enquanto o oitavo é um *bit*:

- Um, se há um número ímpar de *bits* 1 nos 7 *bits* precedentes.
- Zero, se há um número par de *bits* 1 nos 7 *bits* precedentes.

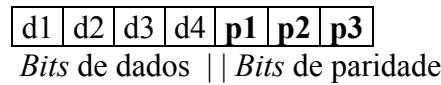
Após a transmissão, o receptor pode recalcular esse *bit* e comparar com o recebido, para verificar se o conjunto está correto.

A eficiência de transmissão nesse caso é de 87,5%

Os códigos são usualmente descritos usando-se a notação (n,k) para indicar um bloco contendo **n** *bits* totais em cada bloco, dentre os quais **k** são *bits* de informação.

Um outro exemplo é o código (7,3).

Em cada bloco de 7 *bits*, 3 são de paridade, cada um testando a paridade de um subconjunto de 3 dos 4 *bits* de informação do bloco, conforme mostrado abaixo:



$$\begin{aligned}
 p1 &= d1 \oplus d2 \oplus d3 \\
 p2 &= d2 \oplus d3 \oplus d4 \\
 p3 &= d1 \oplus d2 \oplus d4
 \end{aligned}
 \quad \oplus \rightarrow \text{Ou exclusivo (xor).}$$

Para compensar a falha mostrada na paridade vertical, se em um caractere houver erro em dois *bits*, aplica-se redundância adicional.

Eficiência 57,14%

Paridade de bloco (Horizontal Redundancy Checking) – HRC

Aqui é adicionado um caractere ao fim de um bloco de comprimento fixo de caracteres, onde cada *bit* é o resultado da aplicação da função \oplus sobre os *bits* de posição correspondente dos caracteres do bloco.

Se houver uma inversão de dois *bits* em um caractere não detectada pela VRC, ela será pela HRC.

Para bloco de 80 caracteres e o código ASCII a eficiência do canal é 87,3%
 $((7*80/8*81) * 100\%)$

SRC (Spiral Redundancy Checking)

Versão de HRC modificada, na qual os *bits* para o cálculo da paridade horizontal são tomados diagonalmente, como mostrado na próxima figura:

CARÁCTER \ POSIÇÃO DO BIT	1	2	3	4	5	6	7	8	PARIDADE DO BLOCO
1	0	1	0	0	1	0	1	0	0
2	1	0	0	0	0	1	0	1	0
3	0	0	1	1	0	0	0	1	0
4	0	1	1	1	1	0	1	1	1
5	0	0	0	0	1	0	1	0	1
6	0	0	0	0	0	0	1	0	1
7	1	1	1	1	1	1	1	1	1
PARIDADE DO CARÁCTER	1	0	0	0	1	1	0	1	0

Figura 4.1

Entrelaçamento

Técnica que permite detectar erros múltiplos, median um esquema de simples VRC, que consiste em enviar em blocos de 7 caracteres todos os *bits* de posição 1, seguidos de todos os *bits* de posição 2, e assim até os *bits* de posição 7, finalizando

com os de paridade. Esse sistema resiste até mesmo a rajadas de erros que alterem 7 bits consecutivos.

Sua eficiência é de 87,5%, a mesma do VRC.

CRC (Cyclic Redundancy Checking)

Elaborada para operar sobre dados a serem transferidos independentemente da sua configuração de caractere. Consiste em aplicar um polinômio sobre grupos de *bits* (12 ou 16) de um bloco de dados, cujo resultado cumulativo é transmitido ao final do bloco. Na recepção são efetuadas as mesmas operações sobre o bloco, e o resultado obtido pode ser comparado com o resultado recebido.

Considerando um bloco de 80 caracteres de 8 *bits*, mais 16 de teste, a eficiência será de 97,5% ((640/656) * 100%)

As técnicas de detecção de erros descritas têm por objetivo sinalizar apenas a existência do erro. Uma vez detectado, o emissor dos dados deve ser de algum modo, avisado de que deve retransmitir o conjunto de dados sinalizado. Isso implica que os dados sejam mantidos na fonte, em um *buffer* do tamanho do bloco, até que seu correto recebimento tenha sido assinalado pelo receptor. Essa sinalização é geralmente efetuada por dois caracteres denominados *ACK* e *NAK*, que significam respectivamente, recepção correta e recepção incorreta (erro no bloco).

Detecção e correção de erros

Essa modalidade de recuperação de erros, em oposição à detecção de erros com solicitação de retransmissão, consiste na adição de *bits* redundantes à mensagem, permitindo sua sinalização e também e restauração do conteúdo original.

O código (7,4) é um exemplo de esquema que consiste na adição de *bits* redundantes que permitem detectar e corrigir um erro ocorrido no bloco. R. H. Hamming desenvolveu vários desses esquemas, que receberam o nome de Hamming codes.

A quantidade de erros recuperáveis depende da quantidade adequada de *bits* redundantes, que cresce bastante quando se quer corrigir dois ou mais erros.

O Hamming pode funcionar da seguinte maneira: para um bloco de informações de tamanho n , onde $n < 2^m - 1$, sendo m um número inteiro, teremos m Hamming *bits*. Os Hamming *bits* são inseridos nas posições 1, 2, 4, 8, e assim sucessivamente, nas posições que são potências de 2, da direita para a esquerda. No exemplo descrito a seguir, o tamanho do bloco escolhido foi de 12 *bits*:

Posição dos <i>bits</i>	8	7	6	5	4	3	2	1
Informação a ser enviada	1	1	0	0	0	0	1	0

O bloco é constituído a partir dos *bits* de informação e dos *bits* de Hamming:

Posição dos bits	12	11	10	9	8	7	6	5	4	3	2	1
Bits de informação no bloco	1	1	0	0	H	0	0	1	H	0	H	H

H → Posição dos Hamming bits.

Os valores dos Hamming bits são o resultado da operação ou exclusivo (xor) sobre o código binário da posição dos bits 1 ocorridos nos bits do bloco:

$$05_{(10)} = 0101$$

$$11_{(10)} = \underline{1011}$$

$$\oplus = 1110$$

$$12_{(10)} = \underline{1100}$$

$$H = 0010$$

Posição dos bits	12	11	10	9	8	7	6	5	4	3	2	1
Bloco a ser transmitido	1	1	0	0	0	0	0	1	0	0	1	0
	0	0	0	0	1	0	0	1	1	0	1	0
Bloco recebido	1	0	0	0	0	0	0	0	1	0	0	1

^

O receptor executa a operação de ou exclusivo com os Hamming bits e os números que indicam a localização dos bits 1 recebidos:

$$H = 0010$$

$$12_{(10)} = \underline{1100}$$

$$\oplus = 1110$$

$$05_{(10)} = \underline{0101}$$

$$\oplus = 1011 = 11_{(10)} \rightarrow \text{posição do bit errado}$$

Se o resultado, como no exemplo, não for zero, isso indicará que houve erro na transmissão. O valor desse resultado é a posição de bit que deve ser alterada para corrigir o erro.

Se dois erros ocorrerem, isso será detectado, mas o resultado do xor será sem sentido. Três erros podem ocorrer e escapar do sistema de detecção.

$$H = 0010_{(2)}$$

$$12_{(10)} = \underline{1100}_{(2)}$$

$$\oplus = \underline{1110}_{(2)}$$

$$5_{(10)} = \underline{0101}_{(2)}$$

$$\oplus = 1011_{(2)} = 11_{(10)} \rightarrow \text{posição do bit errado}$$

Se o resultado como no exemplo não for zero, isso indicará que houve erro na transmissão. O valor decimal desse resultado é a posição de bit que deve ser alterada para corrigir o erro.

Se dois erros ocorrerem, isso será detectado, mas o resultado do OR-EXCLUSIVO será sem sentido. Três erros podem ocorrer e escapar ao esquema de detecção.

Capítulo

5

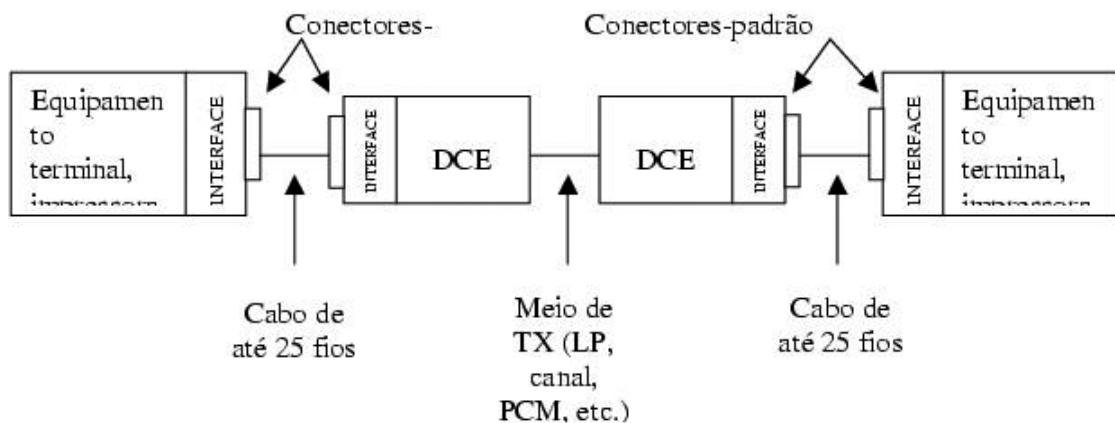
Interface de Comunicação de Dados

Introdução

Pode-se dividir os equipamentos utilizados em um *Link de Comunicação de Dados* em dois grupos:

- Equipamentos Terminais de dados (DTE) : Originam/recebem a informação digital.
- Equipamentos de Comunicação de Dados (DCE) : Têm a função de
 - Na transmissão, tratar o sinal digital gerado pelo terminal, de forma que o mesmo possa ser transmitido ao longo de um meio (modulação)
 - Na recepção, recuperar a forma original (digital) do sinal que foi gerado pelo terminal remoto e entrega-lo ao terminal receptor (demodulação).

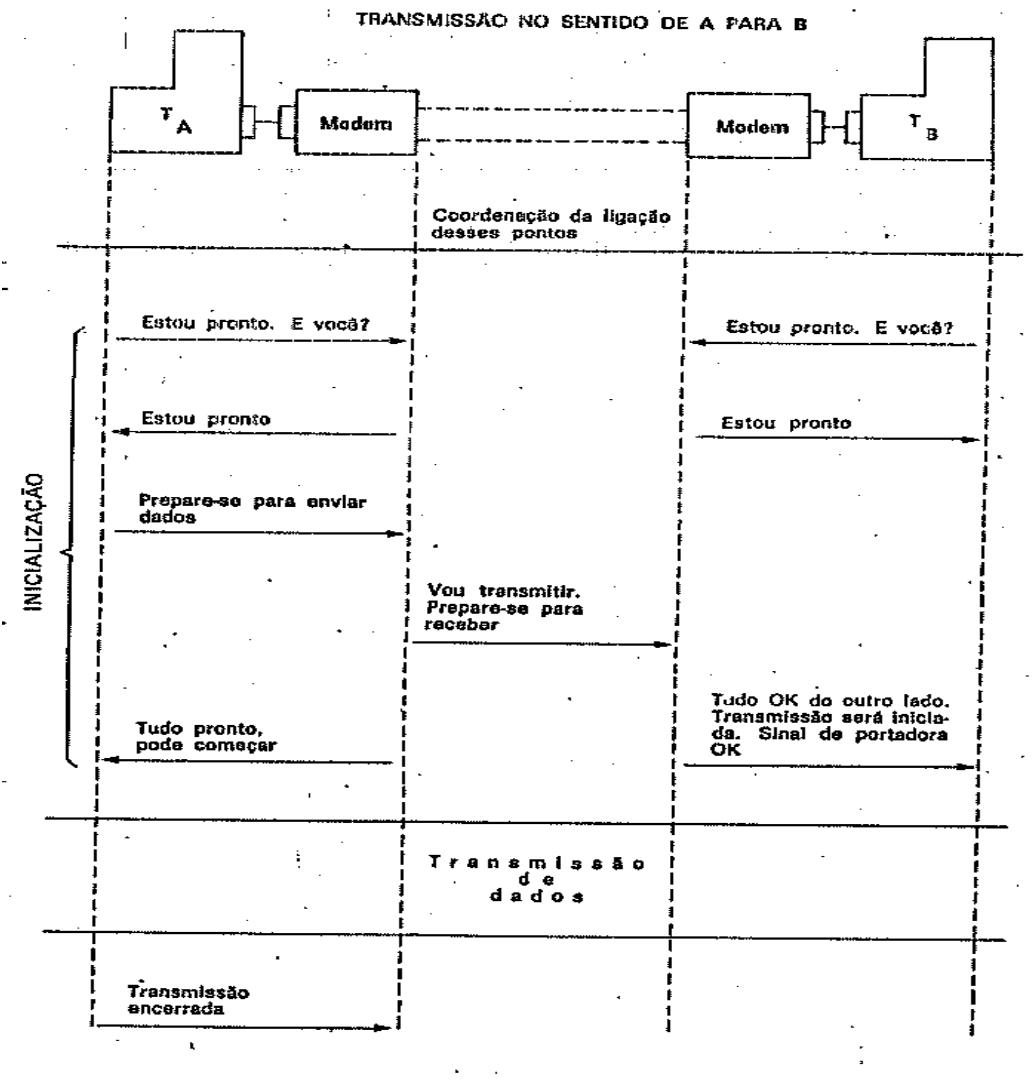
Ambos os grupos de equipamentos apresentam uma interface de entrada/saída, normalizada pela CCITT, como mostrado no exemplo da figura abaixo, que representa a Interface padronizada RS232:



Uma ligação de dados pode ser dividida em 4 fases distintas:

1. Estabelecimento : representa a obtenção de continuidade entre os dois pontos. Esse evento só ocorre para um circuito comutado.
2. Inicialização : representa um diálogo preliminar entre terminais e modems com o objetivo de preparar, coordenar e sincronizar as partes envolvidas na transmissão/recepção.

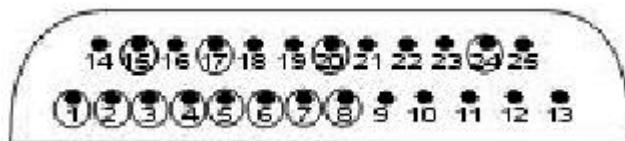
3. Transmissão: representa efetivamente o tráfego de dados depois de estabelecida a fase de inicialização.
4. Corte: representa o desligamento do circuito, por iniciativa de um dos lados para uma ligação *full duplex*, ou por inversão do sentido para ligação *half duplex*.



A Interface Normalizada V.24

Nas fases de uma ligação de comunicação há troca de sinais que não representam a transmissão efetiva de dados entre os terminais e os modems. Esses sinais, assim como os de transmissão efetiva, são normalizados pelo CCITT, através de recomendação V.24.

A interface mecânica é normalizada pela ISO, através da norma ISO 2593, que define um conector padrão DB 25P, como mostrado na figura abaixo:



Círcuito normalmente utilizado.

Para equipamentos produzidos nos EUA, a norma seguida é a RS232 -- C – EIA (Electronic Industries Association), compatível com a CCITT V.24.

Cada circuito de interface pode ser definido por características :

- Função;
- Direção (modem → terminal ou terminal → modem);
- Características elétricas.

Com relação às características elétricas, as recomendações V.28, V.10, V.35, V.36 e V.11 tratam do problema em função da tecnologia, aplicação, etc.

Os circuitos podem também ser destinados a:

- . Controle;
- . Informação propriamente dita;
- . Sincronismo;
- . Terra.

Existe a diferença entre circuitos padronizados e o número de pinos dos conectores porque V.24 procura cobrir todas as utilizações possíveis dos modems, nas diversas faixas de velocidade. Portanto, para cada tipo específico de modem, apenas um subconjunto desses circuitos será necessário.

Interface Padrão RS232-C

Essa padronização é aplicável para interconexão entre o equipamento terminal de dados (DTE) e o equipamento de comunicação de dados (DCE), empregando transferência de dados binários tipo serial.

É aplicável para taxa de sinalização de dados na faixa de 0 a 56 Kbps na transferência de sinais de dados, timing e controle, quando usado em conjunto com equipamentos eletrônicos - cada um deles tendo um retorno simples comum (signal ground) – que podem ser interconectados no ponto de interface. Não se aplica para interfaces em que são usados contatos de fechamento e corrente.

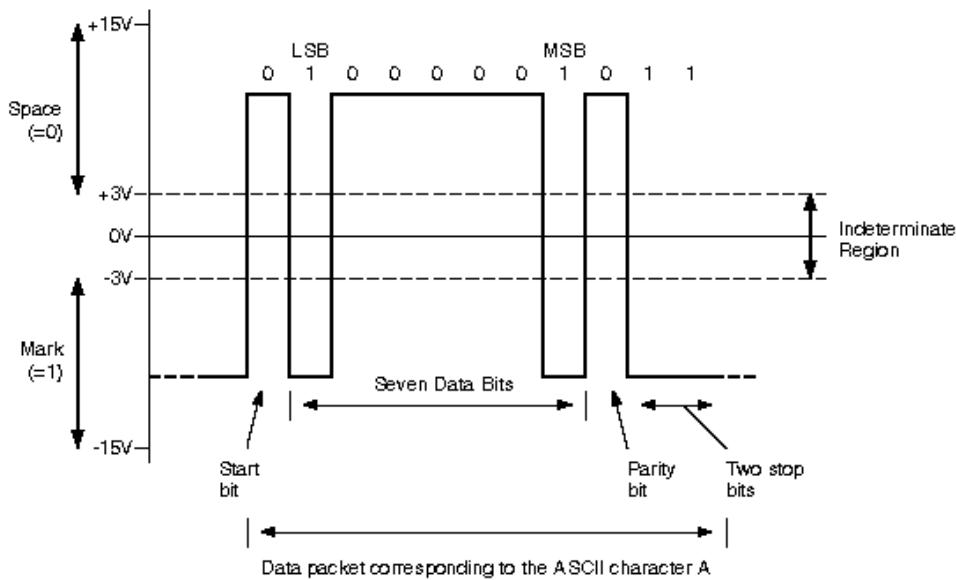
Considera respostas para chamadas automáticas. Contudo, esta padronização não inclui todos os circuitos necessários para originar automaticamente uma conexão.

É compatível com a CCITT V.24 e V.28.

Características elétricas

Para circuitos de transferências de dados, o sinal é considerado em condição de marca quando a tensão no circuito de transferência, medida no ponto de interface, é mais negativa que -3V com relação ao circuito AB (signal ground). O sinal é considerado na condição de espaço quando a tensão for maior que +3V com relação ao circuito

AB. A região compreendida entre -3V e +3V é definida como a região de transição. O estado do sinal não necessariamente será identificado de forma única quando a tensão está na região de transição. Durante a transmissão de dados, a condição de *marca* é usada para discriminar o estado 1 binário, enquanto a condição de *espaço* é usada para discriminar o estado 0 binário.



Para circuitos de transferências de timing e controle, a função é considerada ON quando a tensão no circuito de transferência é positiva em relação ao circuito AB, e é considerada OFF quando a tensão é considerada negativa com respeito ao circuito AB, conforme o quadro abaixo:

Notação	Tensão de transferência	
	Negativo	Positivo
Estado binário	1	0
Condição de sinal	Marca	Espaço
Função	Off	On

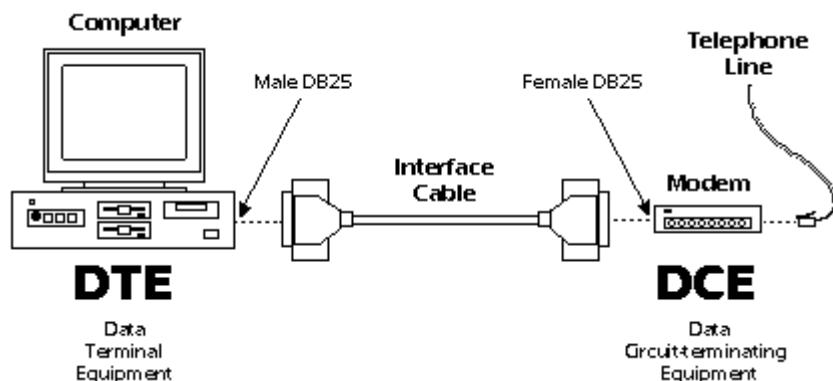
Figura 6.4.

Características mecânicas

Nas interfaces entre o equipamento terminal de dados e o equipamento de comunicação de dados estão localizados conectores de sinais de interface.

É recomendada a utilização de cabos curtos, aproximadamente 15 metros, mas é possível o uso de cabos maiores desde que a carga de capacidade não exceda a 2500pF (medida no ponto de interface e incluindo a terminação do gerador do sinal). É importante observar que a recomendação não normaliza todos os circuitos possíveis da interface.

É interessante que na utilização seja dada preferência aos sinais e pinos normalizados, e que na necessidade do uso de pinos/circuito adicionais sejam tomados cuidados extremos especiais.



A figura abaixo exemplifica, para um caso elementar, a troca de sinais entre as interfaces digitais dos modems e terminais, bem como a relação dos mesmos com a interface analógica:

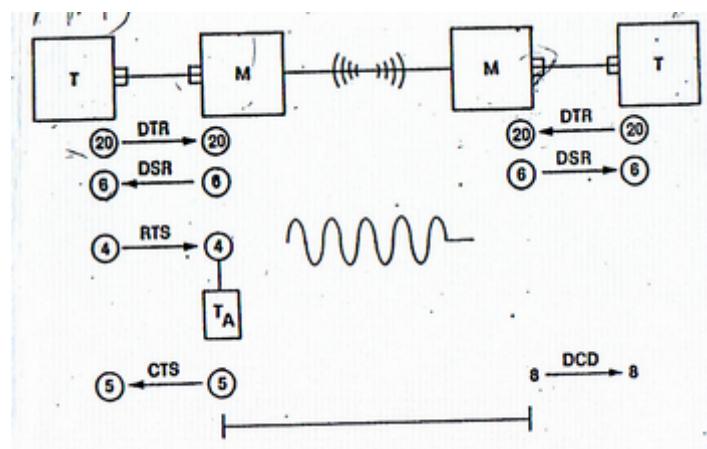


Figura 6.5.

Observa-se que:

- Imediatamente após energizados, os terminais e modems trocam os sinais DTR e DSR pelos pinos correspondentes para indicarem que estão aptos a iniciarem *processo de transmissão*.
- O terminal que deseja transmitir dados *solicita autorização* ao modem, enviando o sinal RTS.
- O modem que recebe RTS libera a portadora e prepara o temporizador TA que condicionará a autorização pelo envio do sinal CTS. O tempo de TA deverá ser pelo menos igual à somatória do tempo necessário para propagação da portadora, do tempo necessário para detecção pelo modem distante e do tempo para informação ao terminal distante.
-

Códigos – Alfabetos

CCITT V – 3 Alfabeto Internacional nº 5

Caracteres de 7 bits, mais 1 bit de paridade. Os caracteres (totalizando $2^7 = 128$) incluem letras maiúsculas, sinais de controle, sinais de pontuação, sinais gráficos e algarismos.

ASCII (American Standard Code for Information Interchange)

Pequenas diferenças em relação ao CCITT, relacionadas com as posições de alguns caracteres de controle.

BCD (Binary Coded Decimal)

Possui 6 bits mais 1 de paridade. Extensão do BCD – 4 bits (quatro dígitos binários usados para representar os dígitos decimais de 0 a 9) com finalidade de incluir letras.

EBCDIC (Extended Binary Coded Decimal Interchange Code)

Utilizado pela IBM nos sistemas 360/370, possui 8 bits de dados mais 1 de paridade, somando um total de 256 caracteres válidos.

Em regra geral o Brasil procura seguir a recomendação V-3 (CCITT).

Fatores que influenciam na escolha do código:

- Número de caracteres usados
- Tipo de informação a ser representada
- Segurança requerida

Transmissão Paralela:

Dados → bits → caracteres

O sistema deve transmitir os bits do caracter de uma só vez.

Exemplo: 6 trajetórias separadas

+1 para fins de controle

Desvantagens:

1. Alto custo das linhas
2. Não se faz a grandes distâncias

Vantagem:

1. Terminal mais barato (não tem necessidade de decidir quais os bits que aparecem primeiro em cada caracter).

Transmissão Serial:

Os “bits” que compõem o caracter são transmitidos um de cada vez. Neste caso o receptor deve saber qual “bit” é o primeiro do caracter, para decodificar a informação (sincronismo).

Transmissão Serial Assíncrona:

A sincronização é conseguida por um elemento de “start” que precede cada caracter, e por um elemento de “stop” após os bits de informação.

A informação transmitida serialmente é composta por:

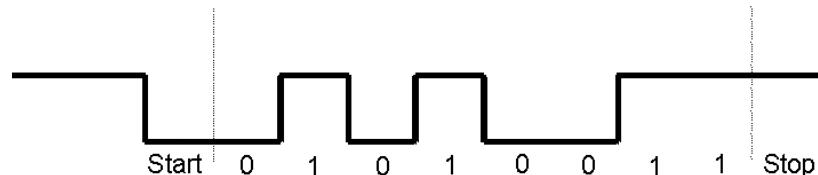
Start , informação e stop

Start : transmitido no tempo de 1 bit

Stop : transmitido no tempo de 1, 1½ ou 2 bits

Vantagem: os caracteres podem ser transmitidos e regularmente espaçados no tempo

Desvantagem: boa parte transmitida não transporta informação útil



Cod : ASCII

Exemplo: Teletipo

- não tem armazenadores temporários
- envia caracteres em

Transmissão Serial Síncrona:

Os bits de um caracter são seguidos imediatamente pelo do próximo, não havendo elemento entre eles.

Os conjuntos de caracteres são divididos em blocos de comprimento variável.

Neste tipo de comunicação para haver sincronismo, os relógios do receptor e do transmissor devem estar em fase. Que é conseguido pelo envio de bits de sincronização no início do bloco.

Observação: Se o bloco é muito extenso, coloca-se caracteres (diferente de informação) de ressincronização no meio do bloco.

- Vantagens:**
1. Utilização mais eficiente da linha
 2. Melhor proteção contra erros. (FIM do bloco, bits de verificação de erros)

- Desvantagens:**
1. Se há erro de sincronização todo bloco é perdido.
 2. Caracteres devem ser enviados em blocos (obrigando as máquinas a terem buffers).

Sincronismo de Bits

Na recepção é feita a recuperação do clock, em função do trem de pulsos recebidos. Este clock recuperado possibilita a leitura dos bits recebidos.

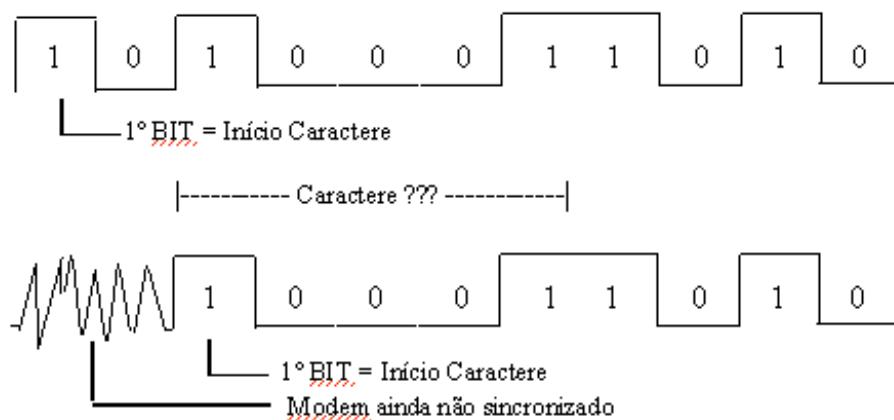
Observação: haverá necessidade de transições no trem de dados (não existindo seqüências longas de 0's e 1's. Uso de SCRAMBLER para misturar os bits).

Sincronismo de Caracter

Após a sincronização de bit, há necessidade de saber onde começa e termina um caractere. A sincronização de caractere é feita por caracteres especiais do tipo SYN.

Observação: dois ou mais caracteres especiais para sincronização.

Figura - Sincronismo de Caracteres



Sincronismo de Texto

Após a sincronização de caractere, precisamos saber onde começa e termina um texto. Isto é feito pela inserção de novos caracteres de controle, como, por exemplo:

STX (Start of Text): Início do texto
ETX (End of Text): Fim do texto

O bloco terá então aproximadamente a seguinte configuração:

SYN	SYN	STX	MENSAGEM	ETX	SYN	SYN
-----	-----	-----	----------	-----	-----	-----

Existe ainda uma série de caracteres especiais, que serão introduzidos nos blocos. Damos a seguir um exemplo, destacando o caractere BCC (*Block Check Character*), que se destina ao controle de erros da transmissão.

SYN	SYN	STX	MENSAGEM	ETX	BCC
-----	-----	-----	----------	-----	-----

Capítulo

6

Protocolos de Enlace

6.1. Definições

1. Conjunto de regras estabelecido para a transmissão ordenada e automática de dados
2. Regras seqüenciais de requisições e respostas pelas quais unidades de uma rede coordenam e controlam operações de transferência de dados
3. Conjunto de regras que governam a troca de informações entre dois ou mais processos.

Exemplos de protocolos: BSC (Binary Synchronous Communication), SDLC (Synchronous Data Link Control), HDLC (High Data Link Control) e X25.

6.2. Protocolos de Linha

A sistematização das 4 fases de uma ligação de dados - estabelecimento, inicialização, transmissão e corte - bem como as características especiais da comunicação é que define os chamados *protocolos de linha*.

Eles têm como funções básicas:

- Endereçamento;
- Estabelecimento de conexão;
- Confirmação de recebimento;
- Controle de erro;
- Retransmissão;
- Controle de fluxo.

6.3. Protocolos orientados a caractere

Protocolos orientados a caractere são aqueles em que as regras são baseadas em caracteres especiais com funções bem definidas.

6.3.1. Protocolo BSC (Binary Synchronous Communication)

O protocolo BSC é um protocolo-padrão da IBM para operar no modo *half-duplex*. O BSC admite basicamente três códigos: EBCDIC, ASCII e Transcode de 6 bits. Formato básico de um bloco de mensagem em BSC:

SYN	SYN	SOH	CABEÇALHO	STX	TEXTO	ETX/ETB	BCC
-----	-----	-----	-----------	-----	-------	---------	-----

No exemplo apresentado, que representa uma conexão com endereço de terminal, o bloco é enviado com caracteres de sincronismo SYN seguidos do caractere SOH (Start of Header – envio de cabeçalho) que pode indicar o envio de endereço terminal. Após SOH é colocado o endereço propriamente dito do terminal de destino.

Em seguida ao endereço inicia-se o *campo de mensagem*.

Aparece, primeiro, o caractere STX (Start of Text), que indica início da área de dados propriamente dita.

Após o preenchimento do campo de dados, podem seguir os caracteres ETB (End of Transmission Block), significando que apesar de todo o campo ter sido preenchido, o texto ainda não acabou; ou ETX (End of Text), representando o fim do texto.

Ao final destes caracteres é colocado mais um, o BCC (Block Check Character), que é montado a partir dos métodos de detecção e correção escolhidos.

Outros caracteres significativos são, por exemplo:

- EOT (End of Transmission): Representa o fim da transmissão de um ou mais blocos e/ou cabeçalho, ou a resposta a uma chamada, quando o terminal não tem nada a transmitir.
- ENQ (Enquiry): Usado para o transmissor *exigir* a resposta a uma transmissão.
- ACK (Acknowledgement): Usado para indicar que um bloco foi bem recebido.
- NACK (Negative Acknowledgement): Usado para indicar que um bloco foi recebido com erro (a partir da análise do BCC) e que deve ser retransmitido.

Qualquer que seja o protocolo de linha adotado pode-se ter ligação ponto a ponto ou ligação multiponto, de forma que o computador possa transmitir para um dos terminais e um dos terminais possa transmitir para o computador, sempre com iniciativa de quem controla a rede.

A figura abaixo ilustra as fases de conversação do protocolo BSC, que serão detalhadas a seguir.

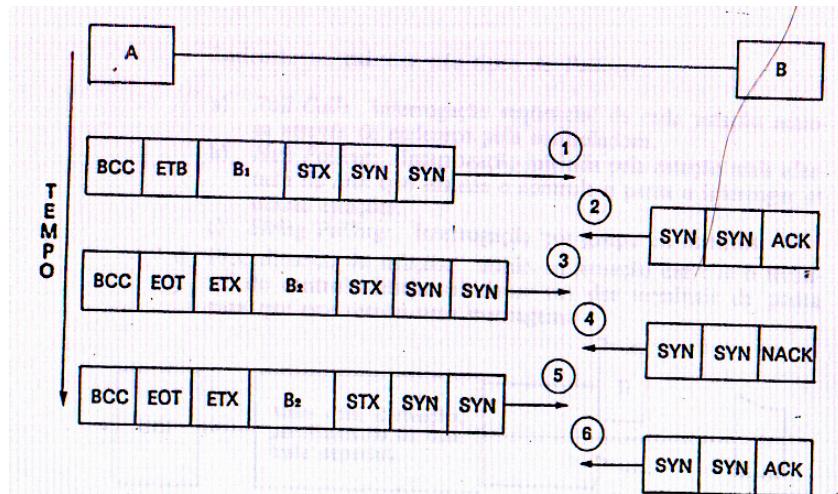


Figura 7.1

6.3.2. Fases de conversação do protocolo BSC

1. A informação do bloco é acrescida de caracteres de sincronismo (SYN), seguidos de caractere que indica início de texto (STX). Em seguida ao bloco B1 aparece o caractere ETB, indicando o fim desse bloco, mas o texto continuará, depois de inserido o caractere de controle de erro BCC.
2. Como no exemplo o terminal recebeu sem erros o bloco B1, envia para A o caractere *bloco bem recebido* (ACK), precedido dos caracteres de sincronismo (SYN).
3. O terminal A envia então o bloco B2, de modo similar ao envio de B1, com a introdução do caractere ETX no lugar do ETB, por se tratar do último bloco. Como a transmissão será terminada, é introduzido o caractere EOT.
4. O terminal B não recebe bem o bloco B2 e informa ao terminal A, pelo caractere NACK, precedido dos caracteres de sincronismo (SYN). Isto também significa pedido de retransmissão do bloco B2.
5. O terminal A retransmite todo o bloco B2.
6. O terminal B recebe bem e confirma.

6.3.3. Ligação multiponto

- *Procedimento de controle terminal:* nas configurações do tipo multiponto (onde há derivação de terminação) há a necessidade de disciplinar o partilhamento temporário dos recursos.
- *Procedimento de chamada (polling):* procedimento no qual o terminal de controle pergunta seqüencialmente aos terminais de ponta se querem transmitir. Estes, por sua vez, devem aceitar ou rejeitar convites; são expressos através de combinações dos caracteres especiais do protocolo.



Figura 7.2

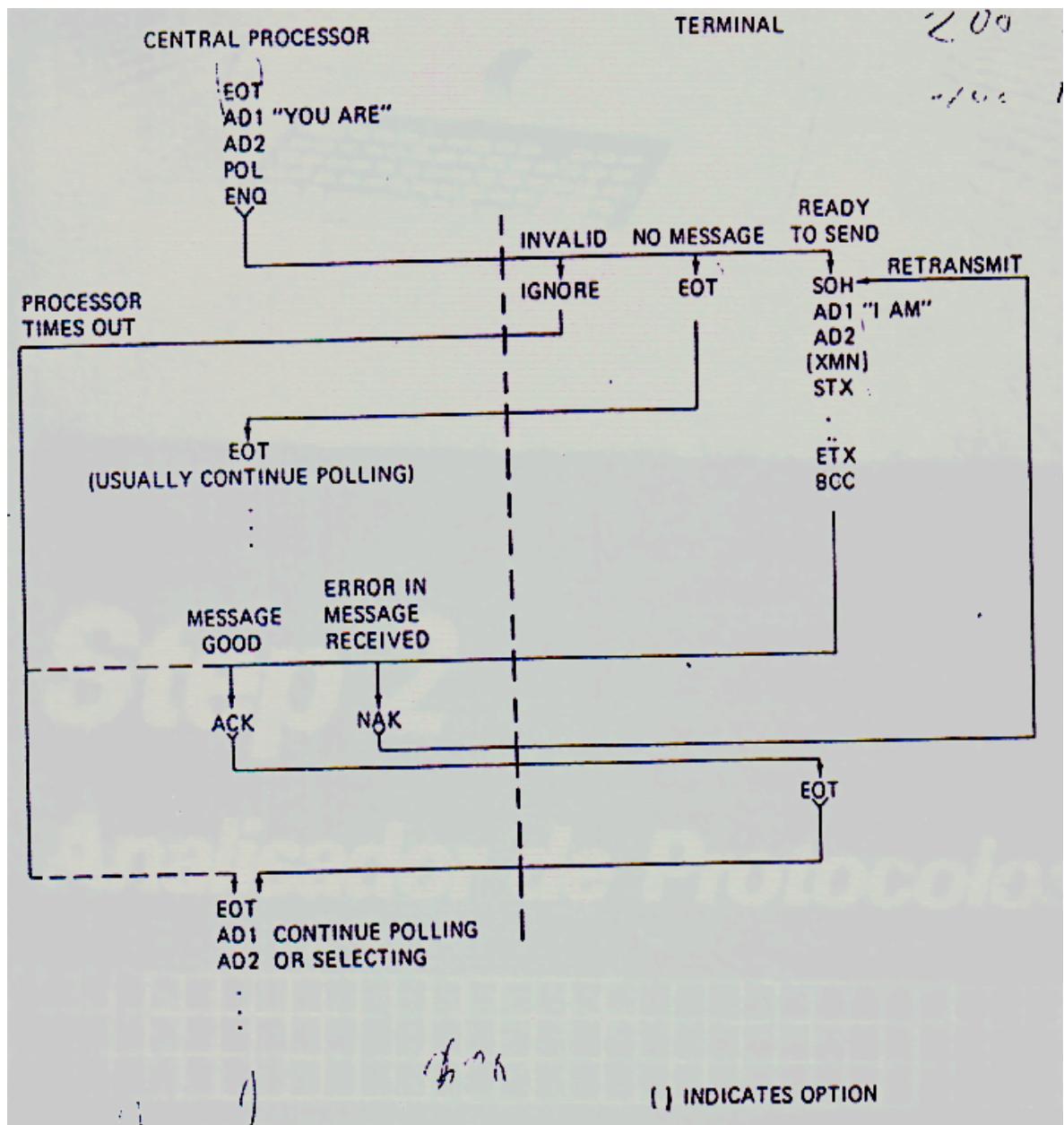


Figura 7.3

Procedimento de seleção: usado na situação em que o terminal de controle deseja selecionar um dos terminais de ponta para que este receba uma mensagem.

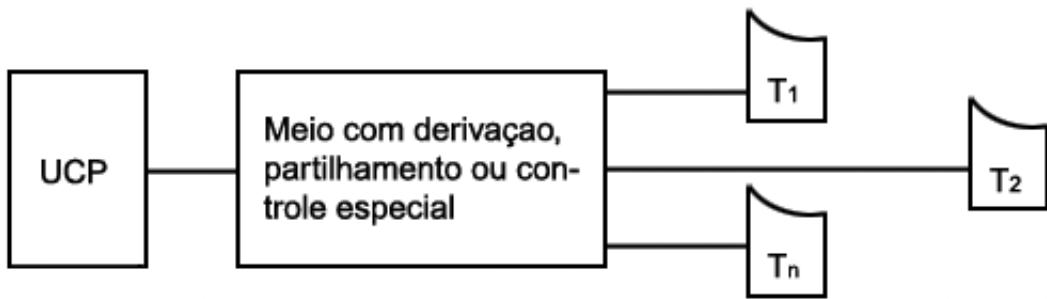


Figura 7.4

Exemplo de um Procedimento de chamada:

ENQ	POL	T1	SOH	SYN	SYN
-----	-----	----	-----	-----	-----

T1, quer transmitir?

SYN	SYN	T1	EOT	SYN
-----	-----	----	-----	-----

Não

ENQ	POL	T2	SOH	SYN	SYN
-----	-----	----	-----	-----	-----

T2, quer transmitir?

SYN	SYN	SOH	T2	STX	TEXTO	ETB	BCC
-----	-----	-----	----	-----	-------	-----	-----

Sim, eis a mensagem.

ENQ	POL	T2	SOH	SYN	SYN
-----	-----	----	-----	-----	-----

OK T2, recebi bem, mande outro bloco.

.....

ENQ	POL	T1	SOH	SYN	SYN
-----	-----	----	-----	-----	-----

T1, quer transmitir?

Basicamente existem três tipos de Polling:

- Roll Call*: interrogação seqüencial de cada estação remota através de endereço pela controladora.
- Hub Polling*: interrogação iniciada pela estação mais afastada do link que assume o controle e passa a interrogar as outras estações.
- String Polling*: interrogação por grupo de estações.

Existem também três tipos básicos de procedimentos de seleção:

- Normal Selection*: quando a seleção é enviada a cada estação, aguardando resposta individual para dar procedimento à seqüência.
- Sequence Selection*: quando a seleção é enviada seguida de mensagem.
- Broadcasting Selection*: quando a seleção é enviada a todas as estações ao mesmo tempo.

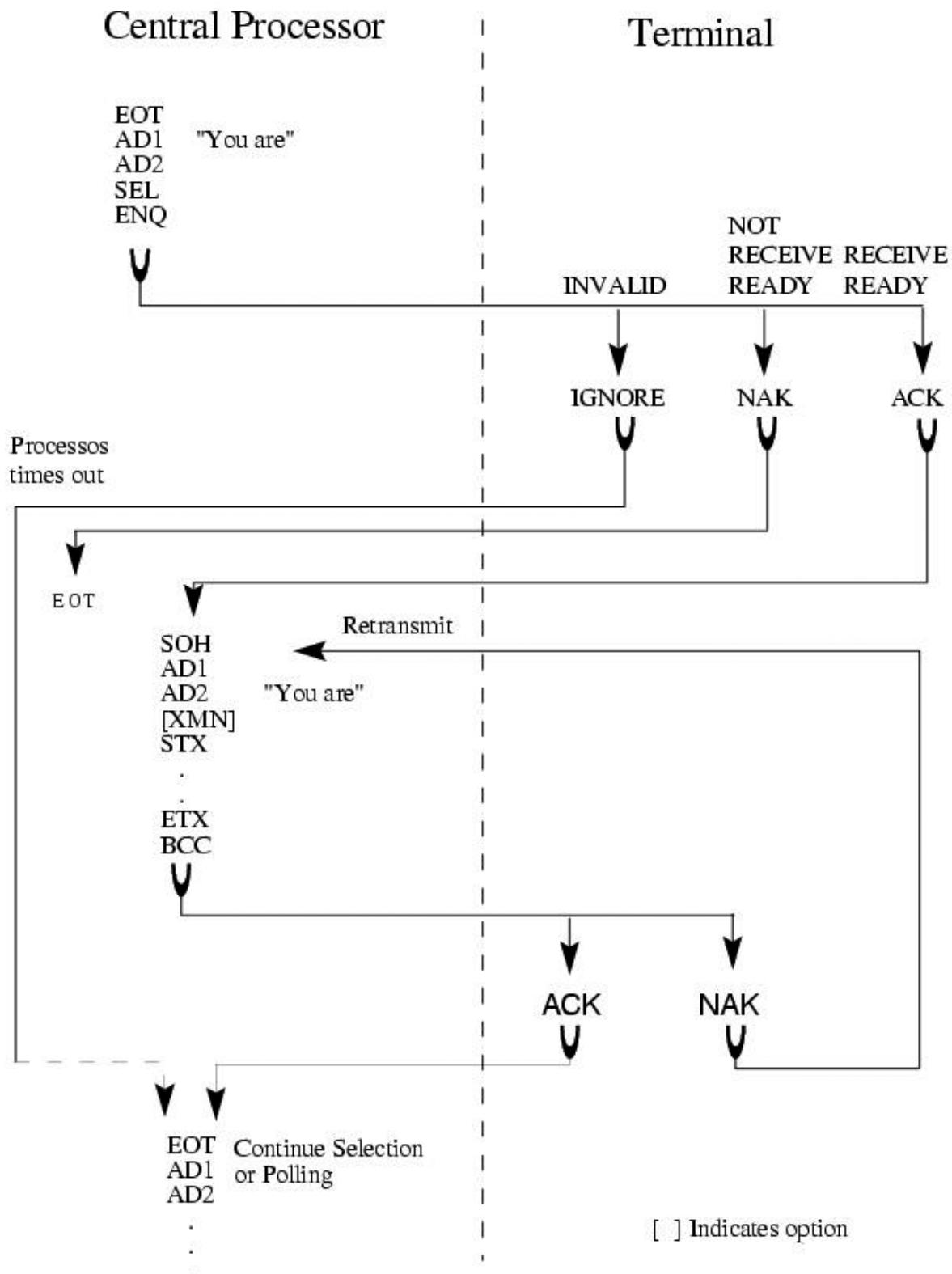
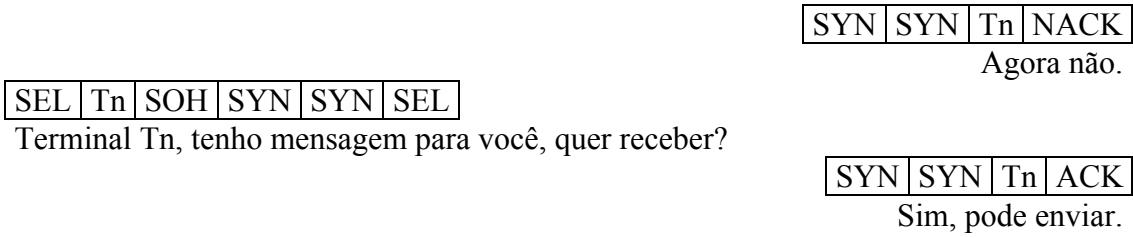


Figura 7.5

Exemplo de um Procedimento de seleção:

SEL	Tn	SOH	SYN	SYN
-----	----	-----	-----	-----

Terminal Tn, tenho mensagem para você, quer receber?



Existem três tipos básicos de procedimento de seleção:

1. **Normal Selection**: quando a seleção é enviada a cada estação, aguardando resposta individual para dar procedimento à seqüência.
2. **Sequence Selection**: quando a seleção é enviada seguida de mensagem.
3. **Broadcasting selection**: quando a seleção é enviada a todas as estações ao mesmo tempo.

Existem aplicações em que, para otimizar o tempo de resposta, se dispensa o envio de ACK para confirmação, assumindo que, quando não for recebido NACK, significa que tudo está bem. A essa prática dá-se o nome de *Dispensa do ACK positivo*.

6.4 - Protocolos orientados a bit

Os protocolos orientados a bit usam campos com tamanho e posições definidas no cabeçalho para realização das funções de enlace. O cabeçalho e cauda têm tamanhos fixo e o campo de dados pode ter um número qualquer de bits. Foram desenvolvidos para superar as deficiências dos protocolos orientados a caracter. Os requisitos essenciais para estes novos protocolos eram:

- independência do código utilizado;
- adaptação a diferentes aplicações e configurações;
- operação em half-duplex e full-duplex;
- alta eficiência;
- elevada confiabilidade;

Final de 1969 a **ANSI** (*American National Standards Institute*) e a **ISO** (*International Standards Organization*) iniciaram os trabalhos formais para o estabelecimento de padrões de protocolos orientados a bit. Em 1973, a IBM anuncia o **SDLC** (*Synchronous Data Link Control*). Em 1979, a ISO divulgou um conjunto de padrões que define o **HDLC** (*High Level Data Link Control Procedure*). O HDLC abriga, na verdade, uma série de outros protocolos como, por exemplo, o **LAPB** (*Link Access Procedure Balanced*) adotado pelo **CCITT** (hoje **ITU-T**) no nível 2 do **X.25**, e o **LAPD** (*Link Access Procedure D-Channel*), definido como protocolo de enlace a **RDSI** (Rede Digital de Serviços Integrados).

6.4.1 - HDLC

O **HDLC** é um dos derivados do protocolo de enlace de dados utilizado na SNA da IBM, o SDLC (protocolo *Synchronous Data Link Control*). Depois de desenvolver o SDLC, a IBM o submeteu ao ANSI e à ISO para sua aceitação como padrão nos EUA e no mundo, respectivamente. O ANSI o modificou tornando-o conhecido como ADCCP (*Advanced Data Communication Control Procedure*), e a ISO o alterou transformando-o no HDLC (*High-level Data Link Control*).

Para satisfazer a grande variedade de aplicações, HDLC define três tipos de estações, duas configurações de link e três modos de transferência. Os três modos estação :

1. **Estação Primária:** Fica responsável pelo controle da operação de acesso. Frames emitidos por ela são chamados *commands* (comandos).
2. **Estação Secundária:** Opera sob controle da estação primária. Os frames emitidos são chamados *responses* (respostas). A primária mantém uma separação de acesso lógico com cada estação secundária da linha.
3. **Estação Combinada:** Combina as características da estação primária e da secundária. Ela pode emitir tanto *commands* quanto *responses*.

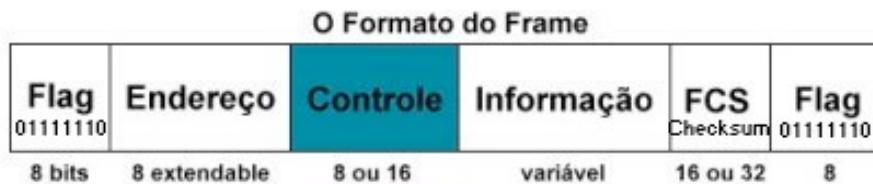
As duas configurações de link :

1. **Não-Balanceada:** Consiste de uma estação e uma ou mais secundárias. Suporta transmissão tanto *half-duplex* quanto *full-duplex*.
2. **Balanceada:** Consiste das duas estações combinadas e suporta transmissão tanto *half-duplex* quanto *full-duplex*.

Os três modos de transferência:

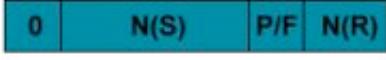
1. **Modo Normal de Resposta (NRM):** Usado com uma configuração não balanceada. A estação primária pode iniciar a transmissão de dados para a estação secundária, mas a secundária não pode somente transmitir dados em resposta de um comando da primária.
2. **Modo Assíncrono Balanceado (ABM):** Usado com uma configuração balanceada. A estação balanceada pode iniciar a transmissão receber permissão de outra estação combinada.
3. **Modo de Resposta Assíncrona (ARM):** Usada com uma configuração não-balanceada. A estação secundária pode iniciar transmissão sem permissão explícita da estação primária. A primária ainda tem a responsabilidade da linha, incluindo inicialização, recuperação de erro, e desconexão lógica.

O HDLC usa transmissão síncrona. Todas as transmissões são em forma de frames, e um simples formato de frame é suficiente para todos os tipos de trocas de dados e controles.



1. **Campo Flag:** delimita o frame nas duas extremidades com um único padrão 01111110. Um simples flag pode ser usado como flag de fechamento por um frame e como flag de abertura pelo próximo.
2. **Campo de Endereço:** identifica a estação secundária transmitida ou recebida do frame. Este campo não é necessitado por acessos ponto-a-ponto, mas é sempre incluído a fim de uniformizar. É usado apenas para distinguir comandos e respostas.
3. **Campo de dados:** pode conter informações arbitrárias. Ele pode ser arbitrariamente longo, embora a eficiência do *checksum* diminua com o aumento do comprimento do quadro devido à maior probabilidade de vários erros em rajada. Este campo está presente somente nos I-frames e U-frames.
4. **Campo sequência de checagem (FCS):** é um código de detecção de erro calculado com o resto dos bits do frame, exclusivo dos flags. O checksum é uma variação menos importante do código de redundância cílica, que utiliza CRC-CCITT como polinômio gerador.
5. **Campo de controle:** é usado para números de sequência, confirmações e outras finalidades. HDLC define três tipos de frames, com formato diferente do campo de controle.
 - Frame de Informação (*I-frames*): leva os dados a ser transferidos pelo usuário. Adicionalmente, fluxo de dados e controle de erro, usando o mecanismo ARQ, pegam carona (*piggypacked*) em um frame de informação.
 - Frame Superviso (*S-frame*): prove o mecanismo ARQ quando *piggybacking* não é usado.
 - Frame Não-numerado(*U-frame*): prove funções suplementares de controle de acesso.

Formato do Campo Contole 8-bit

I: Informação		Legenda
		N(S) = N° sequencia de envio N(R) = N° sequencia de chegada (próximo)
S: Supervisor		S = Bit supervisor M = Bit Não-enumerado (Modificador pagina) P/F = Poll/bit final
U: Não-numerado		

Capítulo

7

Modelo Hierárquico de Protocolos e Padronização

7.1 Introdução

As Redes de Computadores - locais ou a longa distância - surgiram para viabilizar o compartilhamento eficiente de recursos computacionais (hardware, software e dados) dos usuários. Em geral, esses recursos são sistemas heterogêneos: equipamentos de fabricantes diferentes têm características diferentes, utilizam e rodam softwares com características específicas distintas para as aplicações desejadas pelos usuários, e manipulam e produzem dados com formatos incompatíveis. Tal heterogeneidade dificulta consideravelmente a interconexão de equipamentos de fabricantes diferentes. A interconexão de redes, por sua vez, contribui para dificultar o problema ainda mais. A incompatibilidade entre equipamentos e/ou redes foi inicialmente resolvida através do uso de Conversores.

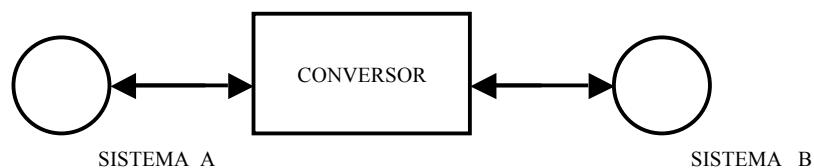


Figura 7.1: Interconexão dos sistemas A e B via conversor.

O conversor interpreta a informação originária de um dos sistemas e passa a informação para o sistema destinatário, após traduzi-la para uma forma compatível ao destino. O uso de conversores, entretanto, é deficiente. São lentos e inadequados para solucionarem incompatibilidades em nível de aplicações.

O termo Aberto se aplica a qualquer sistema adotando os padrões do SC16(subcomitê criado para estudar o problema de padronização) pode interagir com qualquer outro sistema que obedeça aos mesmos padrões. O termo OSI refere-se, pois, aos padrões para a troca de informações entre terminais, computadores, pessoas, redes, processos etc. que estão abertos para os demais, com o propósito de troca de informação através do uso dos padrões aplicáveis.

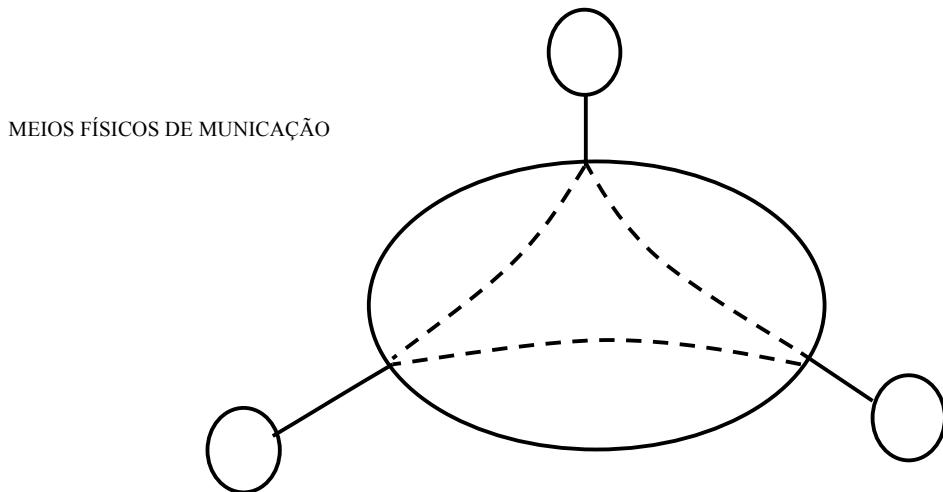


Figura 7.2: Sistemas conectados por meios físicos.

No ambiente OSI, um sistema é um conjunto de computadores com o software associado, periféricos, terminais, operadores humanos, processos físicos, meios para transferência de informação etc., capaz de processar a informação. A transferência de informação entre sistemas é feita por meios físicos, o "OSI" não se refere apenas aos aspectos de comunicação entre sistemas, mas também como estes sistemas cooperam para executar uma tarefa distribuída de um processo de aplicação.

7.2 Estruturas em Camadas

Há um conjunto implícito de regras que regulamenta a comunicação. Na terminologia de Redes de Computadores, este conjunto de regras recebe o nome de Protocolo.

O projeto da "arquitetura" de uma Rede de Computadores que permita seus usuários a se comunicarem eficientemente não é uma tarefa simples. Tudo tem de ser cuidadosa e corretamente especificado para que os vários processos de aplicação possam cooperar de forma harmônica e compatível no processamento de tarefas distribuídas pela rede. O problema é complexo, quer o projeto seja de uma Rede a Longa Distância ou de uma Rede Local. Para reduzir a complexidade no projeto de Redes de Computadores: o princípio é projetar uma rede com um conjunto hierárquico de camadas, cada camada baseada na camada inferior. Reduzindo o projeto global da rede ao projeto de cada uma das camadas, simplifica-se consideravelmente o trabalho de desenvolvimento e de manutenção.

A arquitetura do RM - OSI (Modelo de Referência para a interconexão de Sistemas Abertos) é construída segundo um processo hierárquico onde cada camada utiliza os serviços providos pela camada imediatamente inferior, para fornecer um serviço de "melhor qualidade" à camada superior.

Vantagens:

- a complexidade do esforço global de desenvolvimento é reduzida através de abstrações, isto é, não interessa para uma determinada camada como as demais

implementam o fornecimento de seus serviços; só importa o que a camada oferece como serviço.

- a independência entre as camadas, ou seja, a camada (N) preocupa-se apenas em utilizar os serviços da camada (N-1) e fornecer os seus serviços à camada (N+1), independentemente do seu protocolo. É assim que uma camada pode ser alterada sem mudar as demais.

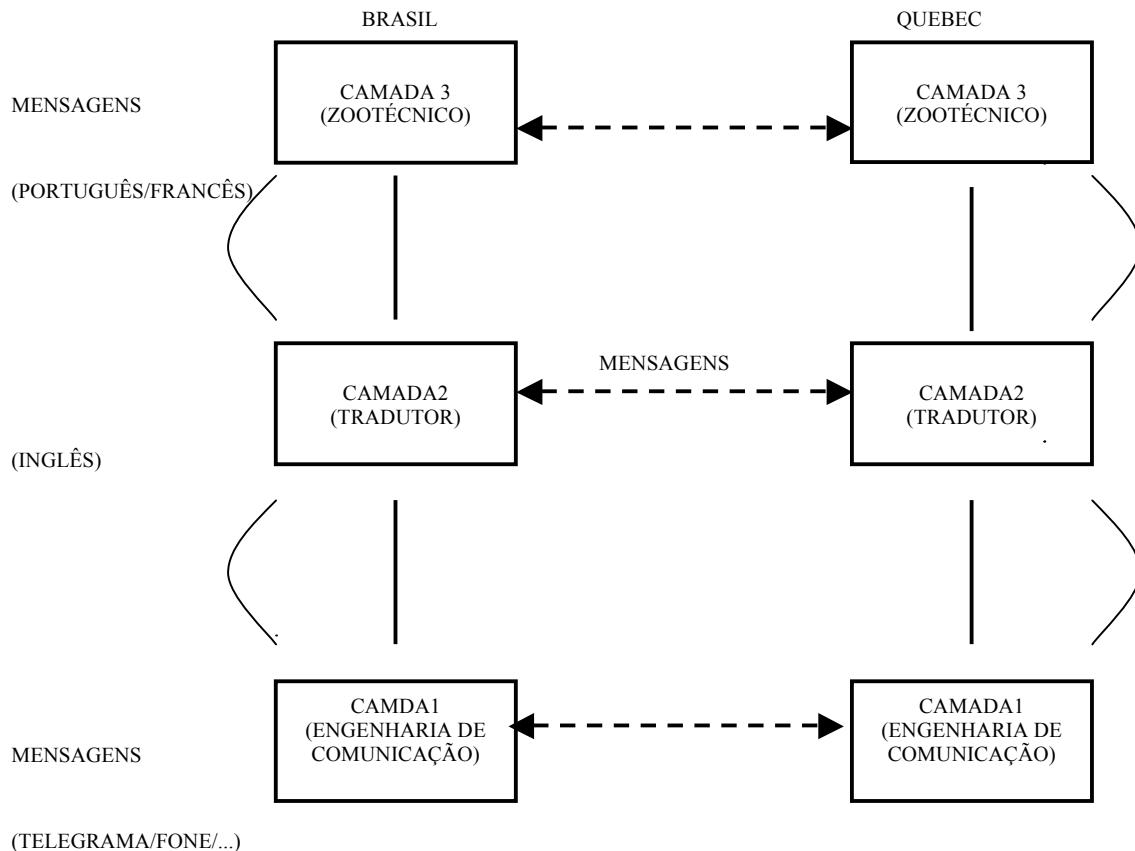


Figura 7.3: Arquitetura hierárquica com três camadas para a comunicação entre os zootécnicos.

Uma ilustração abaixo pode ajudar a sedimentar os conceitos apresentados da estruturação de uma arquitetura em camadas. Imagine dois zootécnicos, um no Brasil e outro no Québec, que desejam se comunicar. Como eles não falam uma língua em comum, cada um contrata um tradutor. O zootécnico brasileiro contrata um tradutor Português-Inglês-Português, e o québécois contrata um tradutor Francês-Inglês-Francês. Cada tradutora contrata, então, um engenheiro de comunicações. Tudo arranjado, o zootécnico brasileiro deseja enviar ao québécois a mensagem "Eu gosto de coelhos". O procedimento utilizado para viabilizar a comunicação desejada é: primeiro, o zootécnico brasileiro passa a mensagem "Eu gosto de coelhos" a seu tradutor que a transforma para "I like rabbits". Em seguida, o tradutor passa esta última mensagem a seu engenheiro que a transmite via telegrama, telefone, rede de computador ou por outro meio, dependendo do que os dois engenheiros acertaram de

antemão. Quando chega a mensagem em Québec, ela é traduzida para "J'aime les lapins" e entregue ao zootécnico quебécois. O processo descrito pode ser esquematizado como mostra a figura acima. A camada (3) corresponde à aplicação propriamente dita (no caso a conversa o entre os dois zoot nicos); a camada (2) oferece o servi o de tradu o de mensagens e a camada (1) oferece o servi o de transmiss o de mensagens. Observe que o protocolo em cada uma das camadas  e totalmente independente dos outros, desde que as "interfaces" entre as camadas n o sejam alteradas.

A arquitetura do RM-OSI foi desenvolvida a partir de tr s elementos b sicos:

- Os processos de aplic o existentes no ambiente OSI.
- As conex es que ligam os processos de aplic o e que lhes permitem trocar informa es.
- Os sistemas.

Os servi os (N) s o oferecidos  s entidades (N+1) nos Pontos de Acesso de Servi os ("Service Access Points" - SAP) ou SAPs (N), que representam as interfaces l gicas entre as entidades (N) e (N+1). Assim, uma entidade (N) oferece servi o (N) na SAP (N), e uma entidade (N+1) procura o mesmo servi o (N) na SAP (N). Um SAP (N)  e servido e usado apenas por uma entidade (N) e uma entidade (N+1) respectivamente, por m uma entidade (N) pode servir v rios SAPs (N) e uma entidade (N+1) usar v rios SAPs (N).

Informa es entre entidades (N+1) s o trocadas atrav s de uma associa o chamadas conex o (N), estabelecida na camada (N) usando um protocolo (N). A camada (N) oferece conex es (N) como parte dos servi os (N). As conex es podem ser do tipo ponto-a-ponto ou de pontos-terminais-m ltiplos; estas  ltimas correspondem a associa es m ltiplas entre entidades. O terminal de uma conex o (N) se d  num SAP (N) e  e chamado Ponto Terminal de Conex o (N) (CEP (N), do termo em ingl s "Connection End Point".

ENTIDADES (N+1)

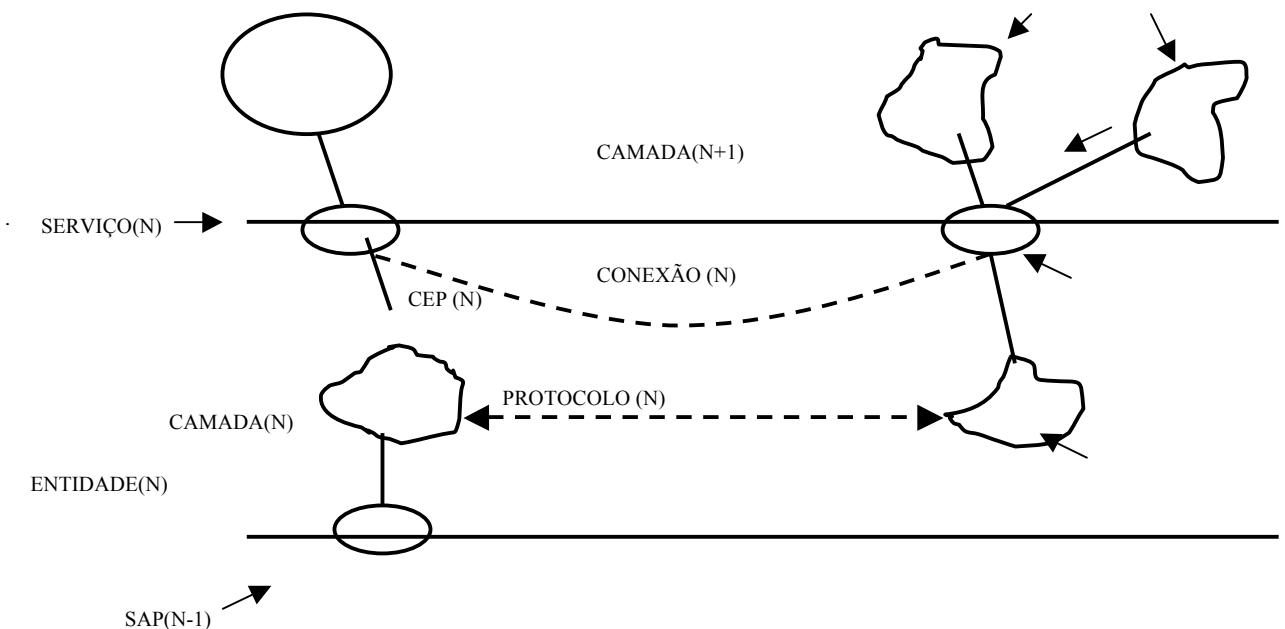


Figura 7.4: Ilustração de alguns termos do RM-OSI da ISO.

7.3 As Sete Camadas do RM-OSI da ISO

A própria ISO afirma que seria difícil provar que sete camadas formam a "melhor" arquitetura para a interconexão de Sistemas Abertos. Alguns princípios aplicados para chegar às sete camadas para a arquitetura do RM-OSI:

- Não criar um número muito grande de camadas a fim de não dificultar o trabalho de integração dessas camadas;
- Criar camadas separadas para tratar de funções que sejam claramente diferentes;
- Funções similares devem ser agrupadas numa mesma camada;

A camada mais alta, camada (7), é a Camada de Aplicação; a camada (6) é a Camada de Apresentação; a (5), a Camada de Sessão; a (4), a Camada de Transporte; a (3) a Camada de Rede a (2), a Camada de Enlace de Dados; e a camada (1), a mais baixa, é a Camada de Meios Físicos. Seguem detalhes sobre cada uma destas camadas.

Camada de Meios Físicos (1) – A camada física permite a flexibilidade de uso de vários meios físicos para interconexão com procedimentos de controle diferentes. Define as características mecânicas, elétricas, funcionais e de procedimentos para ativar, manter e desativar conexões físicas para a transmissão de bits. Os meios físicos podem ser: pares de fios, cabo coaxial, fibra ótica e/ou sem fio. Os hubs também

fazem parte da camada física, assim como certas partes dos “switches”, interfaces e placas de redes.

Camada de Enlace de Dados (2) – A camada de enlace detecta e, possivelmente, corrige erros na camada de meios físicos, fornece à camada de rede a capacidade de pedir estabelecimento de circuitos de dados na camada física (isto é, a capacidade de controlar o chaveamento de circuitos). É responsável pelo envio da confirmação de recebimento (acknowledgement) de quadro. Cabe a esta camada resolver problemas causados por quadros perdidos, duplicados ou danificados. Outra função desta camada é o controle de fluxo de informações, impedindo, por exemplo, que um transmissor rápido afogue um receptor lento com informações.

Camada de Rede (3) – A camada de rede agrupa protocolos de operação da rede, tais como algoritmos de roteamento e de controle de congestionamento. Cabe a ela levar os pacotes da origem ao destino, optando pelo caminho apropriado. A camada de rede deve conhecer a topologia da sub-rede, proporcionando uma rota que evite congestionamentos, podendo inclusive exigir muitos saltos em nós intermediários. Quando a origem e o destino estão em redes diferentes, cabe a camada de redes resolver os problemas de compatibilidade como: diferenças de endereçamento, tamanho do pacote e protocolos diferentes.

Camada de Transporte (4) - A camada de transporte controla o transporte de dados do sistema fonte ao sistema destino, isto é, fim-a-fim, para que o serviço de transporte atinja sua totalidade. O propósito da camada de transporte é, pois, fornecer o serviço de transferência transparente de dados (fim-a-fim) entre entidades da camada de sessão. A complexidade das funções na camada de transporte, que são responsáveis pela qualidade do serviço oferecido, depende da qualidade do serviço de rede disponível. Se a conexão oferecida pela camada da rede for confiável e econômica, as funções necessárias na camada de transporte serão proporcionalmente reduzidas.

Camada de Sessão (5) – A camada de sessão organiza e sincroniza o diálogo, e gerencia a troca de dados entre entidades da camada de apresentação comunicantes. Os serviços são classificados em duas categorias: "Serviço de Administração de Sessão" que une duas entidades, e mais tarde as desune (login/logoff); e "Serviço de Diálogo de Sessão" que controla a troca de dados, delimita e sincroniza operações de dados entre duas entidades de apresentação. Como por exemplo, trocar informações em modo half-duplex ou full-duplex.

Camada de Apresentação (6) – A Camada de apresentação, ao invés de se preocupar com a movimentação ordenada de bits, se relaciona com a preservação do significado das informações transportadas, resolvendo problemas de diferença de sintaxe entre sistemas abertos comunicantes. Computadores podem ter diferentes formas de representação interna dos dados, havendo a necessidade de acordos e convenções para assegurar compreensão mútua. É tarefa da camada de apresentação codificar (podendo haver compressão e criptografia) dados estruturados desde o formato interno usado no transmissor para um fluxo de bits adequado à transmissão, e depois decodificá-los na representação do destino.

Camada de Aplicação (7) – A camada de Aplicação é a mais alta do RM-OSI. Todas as outras camadas existem para dar suporte a esta. É a "janela", entre usuários

comunicantes no ambiente OSI, através da qual ocorre toda troca de informação significativa para esses usuários. Exemplos de protocolos da camada de aplicação na internet são: Telnet, SSH, FTP, SMTP, SNMP e HTTP.

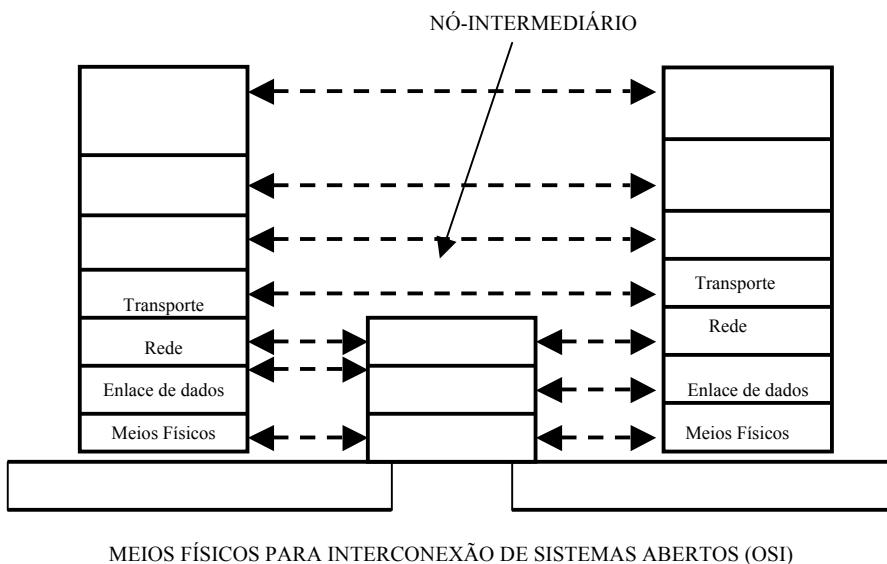


Figura 7.5: As setes camadas para a arquitetura OSI.

7.3 Identificação e Rotas

O RM OSI foi proposto para possibilitar a comunicação entre processos em sistemas abertos. Um sistema aberto é constituído por entidades. Para que entidades em sistemas distintos possam se comunicar, elas devem ser capazes de identificar umas as outras. A identificação de entidades, em sistemas distribuídos conectados por rede(s) é um problema difícil. A dificuldade aumenta quando se consideram várias redes interconectadas, ou mesmo redes isoladas, com capacidade de alterar suas configurações dinamicamente.

7.4 Nomes, Endereços e Rotas

Schoch [SCHO 78] deu as seguintes definições para esses termos que embora sejam informais, são úteis para diferenciar os conceitos envolvidos:

- o Nome indica o que se procura;
- o Endereço indica onde ele se encontra; e
- a Rota indica como se chega lá.

Nomes são identificadores usados para distinguir uma entidade de outra. Um nome deve ser único. Um nome pode incorporar informação estruturada na forma de atributos que caracterizam a entidade. Uma classe de atributos pode ser usada para indicar a localização de uma entidade numa rede. Endereços são nomes que incorporam atributos de localização, mas um nome não é necessariamente um

endereço. Numa rede, o endereço de uma máquina de um usuário dá a sua localização relativa aos outros usuários, no espaço delimitado pela(s) rede(s). Endereço específico de rede e endereço único. No primeiro caso, um usuário tem um endereço que deve ser único na sua rede, mas um outro usuário, em uma outra rede, pode ter o mesmo endereço. No segundo caso, cada usuário tem um endereço que é único em todo o espaço delimitado por todas as redes interligadas. Uma rota caracteriza a localização de uma entidade-destino pela indicação de um caminho para se chegar até ela. Uma rota é um endereço, enquanto que um endereço pode ser ou não uma rota.

7.5 Sistemas de Identificação

Um sistema de identificação consiste de:

- Um alfabeto para a construção de identificadores de várias camadas.
- Regras para a formação de identificadores.
- Funções de mapeamento para transformar identificadores de uma camada em identificadores de camadas inferiores.
- Mecanismos de atualização de contexto, para que as entidades identificadas ou objetos por elas manipuladas possam ser relocados, compartilhados, criados ou destruídos.

Para que um sistema de identificação seja adequado, num ambiente aberto, várias metas devem ser atingidas, das quais destacamos as seguintes:

- Adequação para aplicações de transações e aplicações usando conexões.
- Suporte para identificadores adequados a humanos e identificadores adequados ao processamento por máquinas.
- Suporte para identificadores com contexto global e identificadores com contexto local.
- Suporte para relocação de entidades.
- Suporte para cópias múltiplas de objetos.
- Suporte para que vários objetos ou entidades tenham o mesmo identificador.

Deve ser possível identificar unicamente cada entidade e objeto do ambiente aberto global. O problema é como criar um espaço global de identificadores únicos num ambiente heterogêneo onde cada domínio de identificação local usa um método diferente de criar identificadores únicos neste domínio. Uma solução é a de usar concatenação hierárquica onde cada domínio local tenha um identificador único e concatene o identificador local a este. Uma outra alternativa é a de definir um espaço global de identificação e dividir este espaço entre os domínios locais, os quais devem mapear seus identificadores locais nos identificadores globais.

7.5.1. IDENTIFICADORES NO RM-OSI

A primeira observação que se faz quanto ao RM-OSI (Reference Model – Open Systems Interconnection - International Organization for Standardization) é: a ISO utiliza o termo Título no lugar do nosso termo Nome. Assim, um Título-Global identifica uma entidade (N), independentemente de sua localização, e nunca é alterado. Um Título-Local é um nome que identifica univocamente entidades distintas, mas só num contexto ou espaço limitado, denominado Domínio de Título.

Um Título-Global consiste de duas partes:

1. Um Nome de Domínio de Título para identificar o Domínio de Título no ambiente OSI.
2. Um Sufixo de Título que é único na extensão do Domínio de Título identificado (por a).

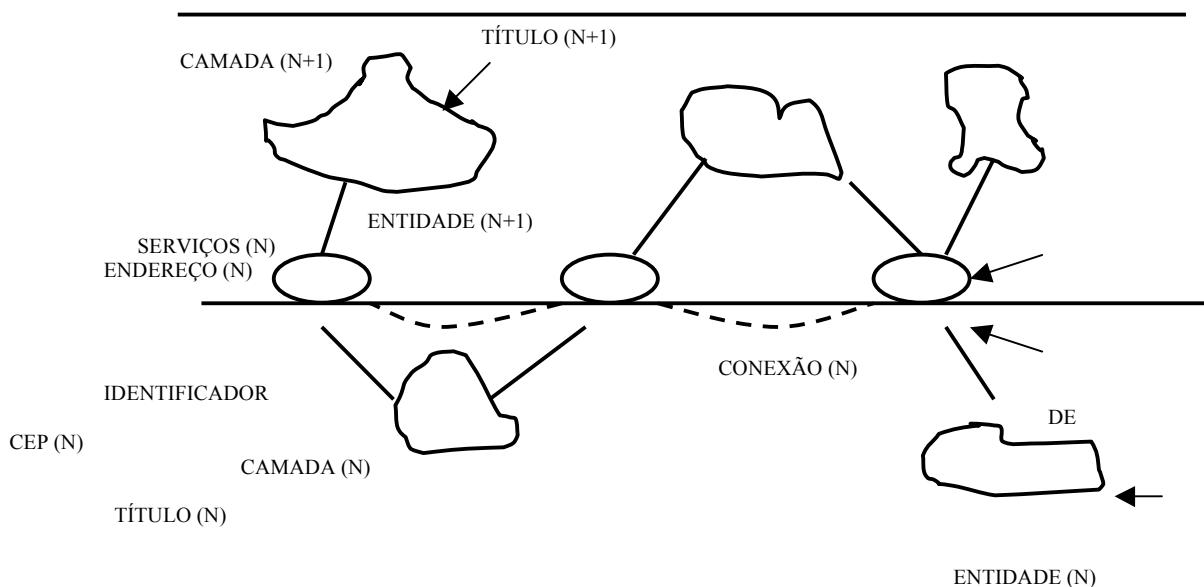


Figura 7.6: Ilustração dos termos (RM-OSI da ISO): Título (N), Endereço(N) e Identificador de CEP(N) (CEP – Connection end Point).

Os Domínios de Título importantes são as próprias camadas. Neste caso, e de forma genérica, o Nome de Domínio de Título identifica a camada (N), enquanto um Sufixo de Título é um Título-Global (N) que identifica uma entidade (N) (na camada (N)).

Um Ponto de Acesso a Serviço, SAP (N), é referenciado através de um Endereço (N). Um Endereço (N) identifica univocamente um SAP (N) ao qual se liga uma entidade, na separação entre a camada (N) e a camada (N+1). Se a entidade não se liga mais ao SAP (N), o endereço não dará mais acesso à entidade. Caso o SAP (N) seja remanejado para uma outra entidade, o seu endereço dá acesso a esta nova entidade e não à antiga, que porém, continua a ser identificada pelo seu Título-Global.

A ISO também define o conceito de Rota na arquitetura do RM-OSI. Uma função de roteamento serve para "traduzir" o endereço de uma entidade em um caminho ou rota pela qual a entidade pode ser alcançada.

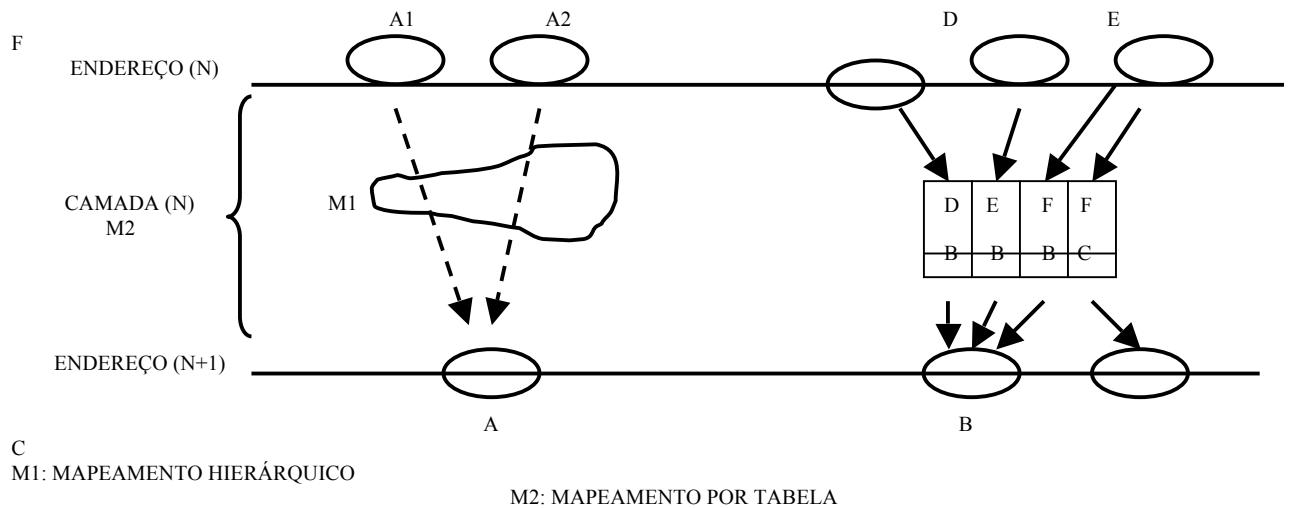


Figura 7.7: Tipos de mapeamento entre Endereços (N) e Endereços (N+1).

Dentro de um SAP(N), o Identificador(N) é usado pela entidade(N) e pela entidade (N+1), em ambos os lados do SAP(N), para identificar a conexão(N). Cada identificador de CEP(N) consiste de duas partes:

- um endereço(N-1) do SAP(N-1) a ser usado conjuntamente com a conexão(N-1);
- um sufixo de CEP(N) que é único no contexto ou espaço do SAP(N-1).

7.5.2. CONEXÕES

Um dos serviços que a camada (N) oferece à camada (N+1) é a transferência de informação entre entidades (N + 1) cooperantes. Essa transferência se dá numa conexão (N). Como vimos, uma conexão (N) é uma associação estabelecida para comunicação entre entidades (N + 1). Cada uma das entidades (N+ 1) é identificada pelo seu endereço (N). Uma dada entidade (N) pode ter uma ou mais conexões estabelecidas com outras entidades (N), com uma só outra entidade(N), ou então com ela própria. Uma entidade (N + 1) tem acesso a uma conexão (N) através de um SAP (N) (Ponto de Acesso de Serviço). Uma conexão (N) poderá ter dois ou mais pontos terminais. Uma entidade (N + 1) faz referência à conexão (N) usando o identificador CEP (N) apropriado.

Estabelecimento de uma conexão

Para que a conexão (N) seja estabelecida é preciso que haja uma conexão (N-1) disponível. Se a conexão(N-1) não estiver disponível, será necessário estabelecer-la. O estabelecimento de uma conexão(N) implica consequentemente que estejam estabelecidas conexões em todas as camadas inferiores.

Liberação de uma Conexão

A liberação de uma conexão(N-1) não causa necessariamente a liberação da conexão ou conexões(N) que a usam. A liberação da conexão(N) entre entidades(N+1) requer ou a disponibilidade de uma conexão (N+1) ou, um acordo entre as entidades (N+1) para desfazer a conexão(N).

Multiplexação

As conexões(N) são suportadas por conexões(N-1); uma das funções da camada(N) faz a correspondência das conexões (N) com as conexões (N-1). A correspondência pode ser de três tipos:

1. correspondência uma a uma na qual cada conexão(N) é construída em cima de apenas uma conexão(N-1);
2. multiplexação para cima, na qual várias conexões (N) são multiplexadas em apenas uma conexão (N-1);
3. multiplexação para baixo, na qual uma conexão (N) é construída em cima de várias conexões (N-1).

O tipo(1), multiplexação para cima, é a única maneira de estabelecer várias conexões (N) num ambiente onde apenas existe uma conexão (N-1). O tipo (2) torna mais econômico o uso do serviço (N-1).

O tipo (3), multiplexação para baixo, é indicado nos casos em que se deseja uma melhor vazão da camada (N-1)- obtida através da utilização de conexões (N-1) múltiplas.

Transferência de Dados

Informação de controle e dados de usuários são trocados entre entidades(N) através de unidades de dados do protocolo (N). Uma unidade de dados de protocolo (N) é uma unidade de dados especificada num protocolo (N) que contém informação de protocolo (N) e, possivelmente, dados de usuário (N).

1. Informação de Controle de Protocolo(N) - PCI (N) - ((N) "Protocol Control Information") é qualquer informação que suporta a operação conjunta de entidades (N). PCI (N) é trocada entre duas entidades (N) cooperantes através de uma conexão (N-1).
2. Dados de Usuário (N)- são dados trocados entre quaisquer duas entidades(N) como um serviço para as entidades (N+1) sendo atendidas pelas, entidades (N).
3. Unidade de Dados de Protocolos (N) - PDU(N) - ((N) Protocol Data Unit") é uma unidade contendo PCI(N) e possivelmente dados de usuário.
4. Unidade de Dados de Serviço (N)- SDU (N) - ((N) - "Service Data Unit") é uma unidade transferida entre uma entidade (N+1) e uma entidade (N).
5. Unidade de Dados de Interface (N) - IDU (N) - ((N) "Interface Data Unit") - é a unidade, de informação transferida através de um Ponto de Acesso a Serviço entre uma entidade (N+1) e uma entidade (N).

6. Informação de Controle de Interface (N) é informação de controle trocada entre uma entidade (N-1) e uma entidade (N) para coordenar a operação conjunta destas entidades.
7. Dados de Interface (N) é informação recebida por uma entidade (N) ou ainda de uma entidade (N+1), para ser transmitida para outra entidade (N+1) numa conexão (N).

É de importância salientar que a transferência de dados pede ser efetuada durante o estabelecimento de uma conexão, isto é, Dados de Usuário (N), podem ser levados pelo pedido de estabelecimento de conexão (N) e pela resposta de estabelecimento de conexão (N). Além disso, Unidades de Dados nas várias camadas podem não ter tamanhos compatíveis.

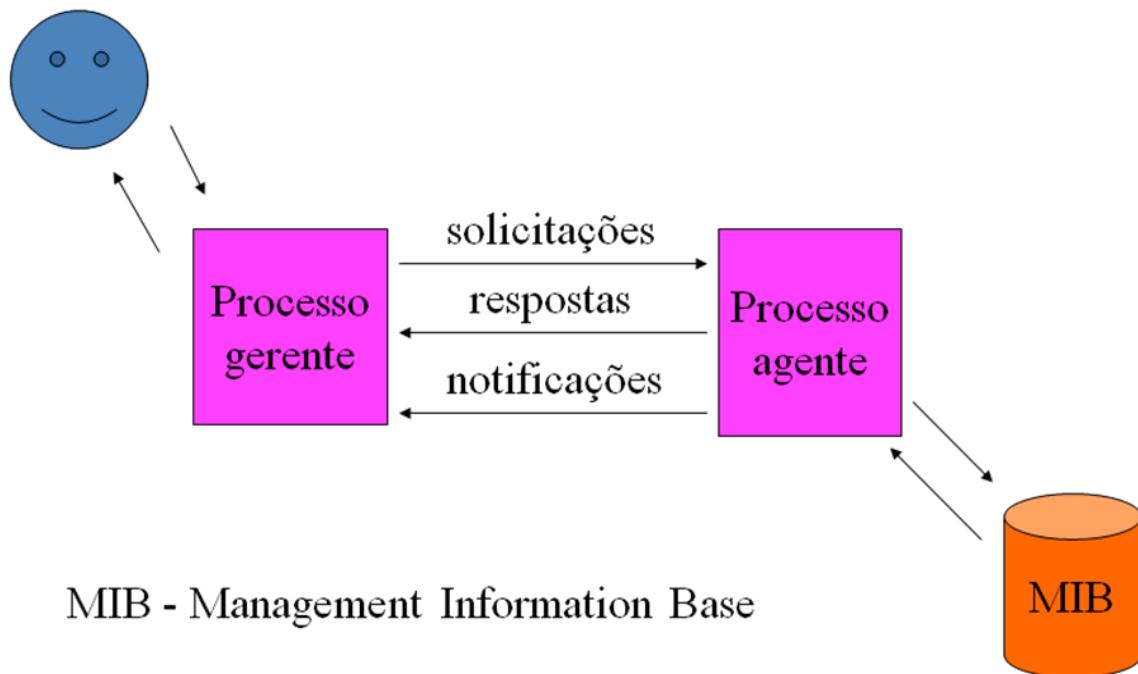
A camada (N) pode ainda prover:

- Seqüenciamento de fluxo de dados para garantir que os dados serão entregues na mesma ordem que foram submetidos pela camada (N+1);
- Detecção e Notificação de Erro para melhorar a qualidade da conexão;
- Controle de Fluxo entre entidades (N) pares, e/ou Controle de Fluxo de Interface entre camadas adjacentes,
- Função Reset para recuperação de perdas de sincronização entre entidades (N) pares.

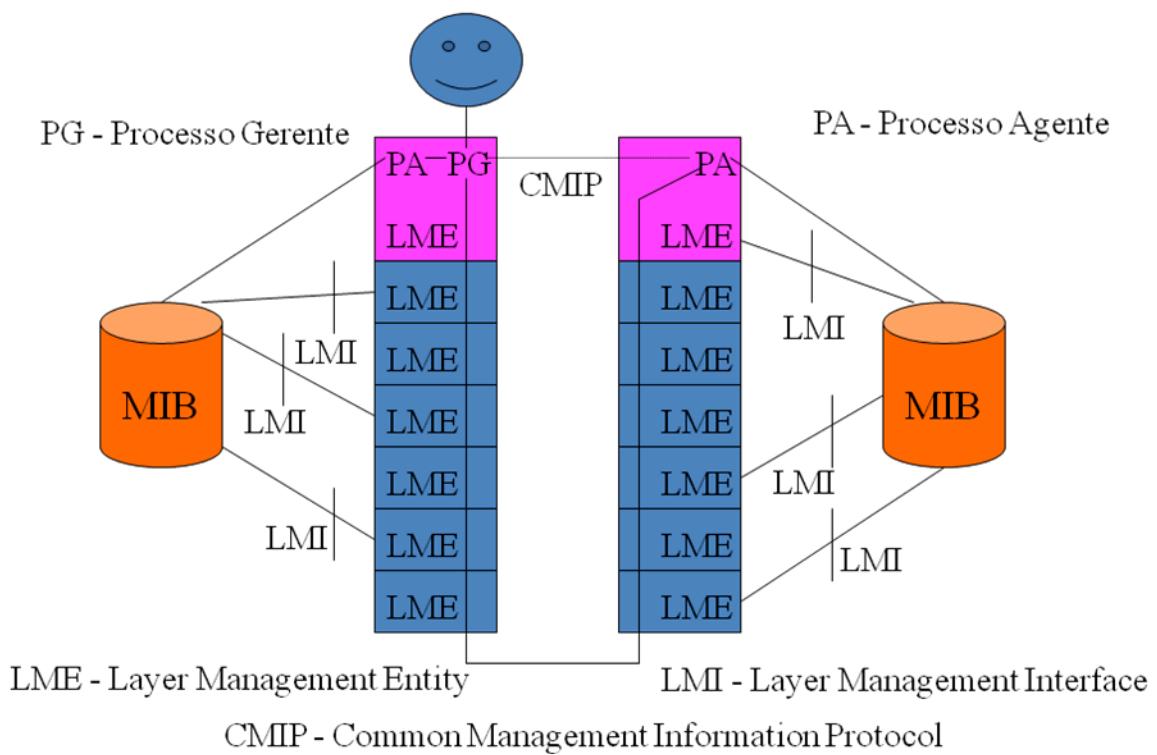
7.6 Arquitetura de Gerência OSI

Gerência de Redes é uma aplicação distribuída, onde processos de gerência (agente-gerente) trocam informações entre si com o objetivo de monitorar e controlar a rede. O Processo Gerente envia solicitações ao Processo Agente, que por sua vez responde as solicitações e envia notificações ao processo gerente. Para responder as solicitações o Processo Agente consulta uma MIB, ou seja, uma Base de Informações Administrativas, onde ficam armazenadas as informações da rede que são estruturadas em forma de árvore, seguindo o paradigma de orientação a objetos, onde objetos gerenciados representam os recursos da rede.

Arquitetura de Gerência OSI



Arquitetura de Gerência OSI



Capítulo

8

Projeto e Desenvolvimento de Protocolos

8.1. Introdução

No capítulo anterior, dizíamos que um protocolo é um conjunto de regras que viabilizam e ordenam a comunicação entre entidades cooperantes em sistemas abertos, possivelmente heterogêneos. Em capítulos subsequentes, discutiremos protocolos para as camadas de transporte, sessão, apresentação e aplicação do RM-OSI. Em nossa discussão, freqüentemente lançaremos mão da língua portuguesa - linguagem natural para os brasileiros - para apresentarmos tais protocolos. O uso apenas da linguagem natural pode, porém, resultar em descrições informais as quais geralmente têm os atributos de serem incompletas e longas, não-consistentes, de compreensão dúbia ou de interpretações variadas. É óbvio que um protocolo não pode ser descrito informalmente sob o risco de não cobrir todas as possíveis situações. O resultado, de qualquer forma, é um protocolo que, quanto mais complexo, menos chance terá de viabilizar coisa alguma, e muito menos a comunicação entre sistemas heterogêneos.

É imprescindível que a especificação ou descrição de um protocolo seja concisa e precisa, totalmente ausente de ambigüidades é crença geral, que isto só se verifica através de uma especificação formal do protocolo. Não que a descrição informal (até mesmo causal) do protocolo deixe de ter seus méritos. Como já dissemos, talvez seja esta a maneira mais indicada para os humanos entenderem alguns protocolos - principalmente os mais complexos, que não são incomuns no ambiente de processamento distribuído de uma rede local. Em adição, o desenvolvimento inicial de protocolos utilizou a descrição informal, provavelmente como resultado da inexistência de técnicas formais para a especificação de protocolos conciliadas. Entretanto, assim que os protocolos começaram a se tornar mais sofisticados e complicados, formou-se um consenso sobre a necessidade de especificações através de técnicas formais. Na segunda seção deste capítulo, revisaremos os principais tipos de técnicas para especificação de protocolos.

Mesmo quando a descrição informal de um protocolo é fornecida para benefício de compreensão de sua operação, é de boa prática fornecer também a sua especificação formal, isto porque esta serve de base para a validação, verificação, testes e implementação do protocolo.

A validação e verificação de um protocolo são atividades importantes durante o seu projeto. Essa importância é reforçada quando o protocolo está para se tornar um produto comercial e, mais ainda, quando o protocolo sendo especificado está para ser sugerido como um padrão. Por validação de um protocolo entende-se: as atividades para mostrar ou assegurar que a especificação e implementação do protocolo satisfarão as necessidades da comunicação para a qual ele está sendo projetado (ex: Os tempos de atraso na entrega das mensagens estão

dentro do esperado? O controle de fluxo empregado é adequado?). As atividades de validação, desempenhadas durante todo o projeto de protocolo, podem incluir estudos de simulação, modelagem analítica, etc. Aqui estamos interessados num único aspecto da validação, a verificação de protocolos, isto é, na determinação de certas características lógicas da especificação do protocolo que indicam se ele tem defeitos ou não (ex: possibilidades de *deadlock*). As operações para verificação de um protocolo são, portanto, realizadas em cima de sua especificação. O uso de técnicas formais de especificação contribui inquestionavelmente para facilitar quando não para possibilitar, estas operações.

Após verificada a especificação de um protocolo, o próximo passo é implementá-lo. Uma tarefa importante é o teste de consistência da implementação (ou de partes da implementação) com a especificação do protocolo. Em outras palavras, a implementação age de acordo com a especificação. Dois bônus valiosos, consequentes da especificação formal de um protocolo, que tem sido alvos de investigações são:

- 1) a possibilidade de implementação automática
- 2) a geração automática de testes a partir da especificação.

Em 1, a idéia é gerar programas executáveis diretamente, por meios automáticos, a partir da especificação de um protocolo. Talvez não se tenha chegado ainda ao estágio em que a implementação do protocolo como um todo, seja obtida automaticamente. Boa parte ainda terá que ser feita à mão. Entretanto, a implementação parcial automática traz subsídios para facilitar esta tarefa e certamente contribuiu para reduzir o custo e a introdução de erros, características de implementações manuais. Em 2, alguns “cenários” de testes são automaticamente obtidos da especificação. Mais uma vez, isto não elimina por completo o trabalho de desenvolvimento do testador da implementação, mas é uma contribuição para amenizar os esforços deste desenvolvimento.

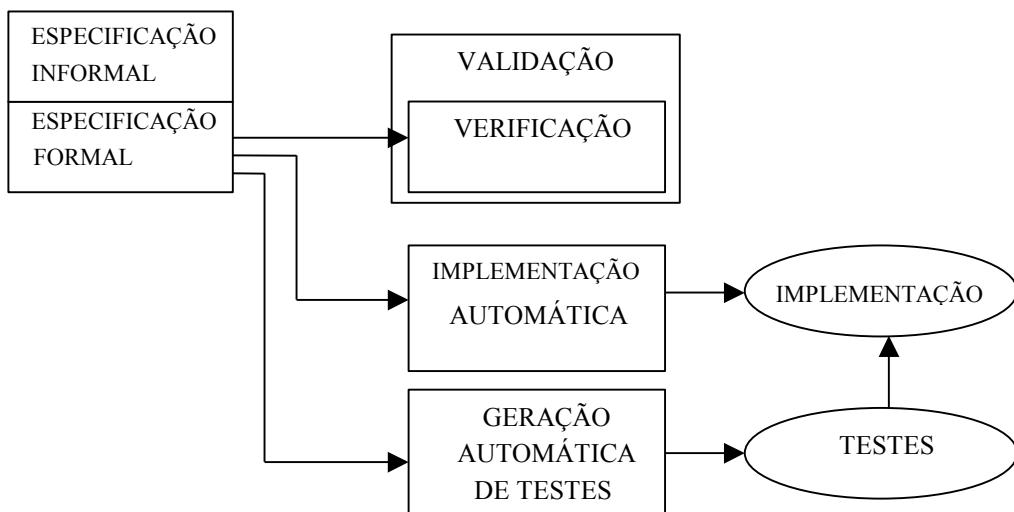


Figura 8.1- Projeto de Protocolo

Tradicionalmente, os protocolos têm sido implementados inteiramente em software. Entretanto, para que os requisitos de velocidade das redes atuais sejam

atendidos, pode ser necessário implementar parte de um protocolo em hardware, o que implica em promover a partição hardware/software da sua especificação. Atualmente o projetista tem poucos recursos para auxiliá-lo no refinamento da especificação e na partição do protocolo. Neste caso, é útil fornecer ao projetista medidas de desempenho de maneira a automatizar o processo. Combinam-se os parâmetros utilizados numa função de custo única, denominada função objetivo, que fornece uma medida da qualidade da partição, auxiliando o projetista a selecionar aquela que melhor satisfaz as especificações.

Utiliza-se uma ferramenta de análise de desempenho, denominada Tangram-II, para auxiliar na identificação das operações críticas em termos de tempo e outros parâmetros tais como, por exemplo, vazão. Com base nessas medidas o projetista pode analisar a sensibilidade das especificações de projeto aos parâmetros do modelo e definir a partição.

O protocolo é inicialmente modelado nas redes de Petri estocásticas. Posteriormente, é gerado um novo modelo, markoviano, utilizando o Tangram-II. Os valores dos parâmetros deste novo modelo são fornecidos por um método de otimização baseado em algoritmos genéticos. Para cada conjunto de valores fornecidos pelo algoritmo genético, o Tangram-II calcula medidas de desempenho que indicam o seu grau de aptidão em relação a uma função objetivo especificada pelo projetista. Desta forma, realiza-se uma seleção dos melhores conjuntos de parâmetros, gera-se uma nova população e realimenta-se o Tangram-II. Este processo se repete até que a função objetivo esteja dentro de uma precisão especificada. Por exemplo, se a especificação for uma determinada curva de vazão para o protocolo, são fornecidos os parâmetros do modelo que melhor aproximam a curva.

Os avanços tecnológicos que ocorreram nos últimos anos nas áreas de comunicação celular, redes locais sem fio (wireless LANs) e comunicação via satélite criaram novas linhas de pesquisa em Ciência da Computação que começaram a ser investigadas recentemente. Uma delas é a área de computação móvel que tem recebido muita atenção por parte da comunidade científica. O termo móvel significa "ser capaz de mover mas retendo a conexão com a rede". Computação móvel representa um novo paradigma computacional que tem como objetivo prover aos usuários acesso permanente a rede independente de sua localização física. Esse acesso pode ser feito utilizando um dispositivo computacional portátil como computadores laptops ou palmtops, ou telefones celulares. Com a diminuição dos custos desses dispositivos, a computação móvel se tornará viável não somente para o segmento empresarial mas para as pessoas de uma forma geral. É importante observar que computação móvel expande a área de computação distribuída para incluir serviços oferecidos a um usuário independente de sua localização e, mais importante, de mudanças na localização. A combinação de comunicação sem fio com a mobilidade de computadores criou problemas novos nas áreas de redes de computadores, sistemas operacionais, sistemas de informação, otimização, dentre outras.

Os protocolos de comunicação sempre foram projetados levando-se em consideração características do ambiente onde são executados. No caso da pilha de protocolos para um ambiente móvel, as características principais do ambiente, que são bandwidth baixa e alta taxa de erro, afetam os protocolos de todas as camadas. Logo, através do uso de técnicas de compressão pode-se obter uma melhor eficiência do canal de comunicação. Além disso, vários princípios de projeto de protocolos foram propostos dependendo da função de cada camada.

Ao se projetar protocolos de comunicação deve-se seguir certos princípios como os definidos em Holzmann (*Design and Validation of Computer Protocols*) que identifica cinco partes distintas, a saber:

1. O serviço a ser provido pelo protocolo
2. As suposições sobre o ambiente no qual o protocolo será executado
3. O vocabulário de mensagens usado para implementar o protocolo
4. A codificação (formato) de cada mensagem no vocabulário
5. As regras de procedimento usadas para garantir a consistência das trocas de mensagens e, em última análise, executar o serviço especificado

Normalmente, por um abuso de linguagem, a quinta parte é chamada de protocolo e é a mais difícil de projetar e verificar.

É interessante observar que cada parte da especificação do protocolo pode definir uma hierarquia. Por exemplo, o vocabulário do protocolo pode ser formado por uma hierarquia de classes de mensagens.

De uma forma geral, o projeto do protocolo deve ser estruturado buscando sempre:

1. Simplicidade: protocolo deve ser construído a partir de um pequeno número de funções bem projetadas e bem entendidas.
2. Modularidade: um protocolo complexo pode ser construído a partir de módulos mais simples que interagem de forma bem definida e simples.
3. Bem-formado: um protocolo, como qualquer outro sistema, não deve conter funções que nunca serão executadas ou que não foram definidas; deve possuir limites conhecidos como tamanho de fila de mensagens; deve ser auto-estabilizante e poder ser adaptado.
4. Robustez: idealmente, o protocolo deve fazer suposições mínimas sobre o ambiente onde será executado. Na prática, isso é difícil de obter pois o ambiente influencia diretamente a forma como o protocolo deve trabalhar.
5. Consistência: protocolos, como outros algoritmos distribuídos, devem possuir certas propriedades como não possuírem deadlocks ou livelocks, terminações erradas.

8.2. Especificação formal de protocolos

O projeto de um protocolo baseia-se, sobretudo, na especificação formal do protocolo. Segundo P. R. F. [CUNH 83], a especificação de um protocolo deve explicar exatamente todas as condições que ele deve satisfazer e nada mais. Ela deve expressar o essencial e omitir o não essencial; deve ser, como já dito, clara e precisa. É aí onde reside a dificuldade, baseando-se que estes dois predicados são, às vezes, antagônicos. As técnicas formais para especificação de protocolos existem para que se façam especificações com esses dois predicados. Esta seção trata das técnicas formais mais comumente empregadas na especificação de protocolos de comunicação para redes de computadores.

A arquitetura da rede que adotamos é a arquitetura para interconexão de sistemas abertos da ISO, o RM-OSI. O RM-OSI é estruturado em sete camadas hierárquicas de protocolos onde cada camada oferece serviços à camada imediatamente superior (ex.: a camada (7) oferece serviços de comunicação aos processos sendo executados nos vários sistemas conectados a rede). Os serviços de cada camada são providos pelo Protocolo da camada. Falamos então de Especificação

de serviços e Especificação de protocolos. Consideraremos primeiro a especificação de serviços.

8.2.1. Especificação de serviços

Especificação de serviços de um protocolo consiste na descrição do comportamento de entrada e saída da camada de protocolo correspondente [BOCH 80], isto é, a especificação de serviços do protocolo (N) descreve o serviço de comunicação fornecido pela camada (N).

A especificação de serviços do protocolo (N) deve definir as primitivas de serviço (N), a ordem de execução dessas primitivas e os efeitos da execução de cada uma delas. Ela, porém, não se deve ocupar em definir detalhes de implementação das primitivas, esses detalhes são pertinentes à interface e as camadas (N) e (N+1) e, como já visto, podem variar em função do sistema operacional sob o qual irá rodar a implementação da interface e do próprio protocolo ou da linguagem de codificação empregada, etc.

Um exemplo dos pontos importantes numa especificação de serviço nos é fornecida em [BOCH 80] com o serviço da camada de transporte (camada 4 do RM-OSI). Neste caso, algumas primitivas básicas são os pedidos de conexão, desconexão, enviar e receber mensagens. A execução destas primitivas envolve uma entidade na camada (4), aquela que provê o serviço, e uma entidade na camada (5), aquela que recebe o serviço. Durante a execução, estas duas entidades comunicam-se através de parâmetros. A especificação do serviço de transporte deve definir os possíveis valores para estes parâmetros e a direção de transferência para cada parâmetro. A ordem de execução das primitivas também deve ser explicitada (ex: enviar não pode ser executado antes de conexão). Num dado instante, as primitivas com possibilidade de execução e os valores permitidos para os parâmetros dependem do passado das interações entre as entidades. A especificação deve indicar todas as situações, os estados possíveis de acontecer, enquanto o serviço está sendo prestado e como estas situações ou estados são alterados, em função de alterações passadas e de uma nova interação. A alteração do estado pode ainda ser em função de interações passadas no sistema local (restrições locais de interações anteriores em sistemas remotos e/ou restrições globais). No serviço de transporte, o fato de enviar e receber só poderem ser executados após a conexão ter sido efetuada com sucesso, é uma restrição local; já o fato do valor do parâmetro que indica o número de mensagens do primeiro receber, executado num sistema, ser igual ao valor do mesmo parâmetro do primeiro enviar, executando em outro sistema remoto comunicante, é um exemplo de uma restrição global.

A utilização de técnicas para a especificação formal de serviços é ainda muito escassa. Dentre os trabalhos iniciais na área, destacam-se os empregos de métodos de engenharia de software em geral e métodos descritivos do histórico de entradas e saídas da camada de protocolo; histórico este fornecido em termos das seqüências de entrada e saída permitidas e de suas interdependências. Uma outra alternativa para a especificação das seqüências permitidas de interações é através de métodos de especificação algébrica [GUTT 78]; mais detalhes acerca dos métodos mencionados, incluindo comparações entre eles podem ser encontrados em [SUNS 79]; a maioria dos trabalhos em especificação formal de protocolos, concentram-se mais nos protocolos do que nos serviços que eles fornecem.

8.2.2. Especificação de Protocolos

A especificação dos serviços da camada N não define como o protocolo N fornece os serviços. A especificação do protocolo N define as entidades cooperantes da camada N e como estas entidades fornecem os serviços, não define toda entidade N, mas apenas na extensão necessária para assegurar compatibilidade com as outras entidades da camada. Mais uma vez, a questão de como a entidade será implementada é deixada em aberto para propiciar liberdade na escolha dos métodos de implementação.

Na especificação do protocolo N, a camada N deve ser descrita. A própria ISO apresenta diretrizes para uma descrição de camada (ISO 81). De acordo com ela, os seguintes pontos devem constar na descrição da camada N:

- 1.Uma exposição geral do propósito da camada e de seus serviços.
- 2.Uma especificação exata do serviço fornecido pela camada.
- 3.Uma especificação exata do serviço necessário a ser fornecido pela camada N-1.
- 4.A estrutura interna da camada N, em termos das entidades N e de suas relações.
- 5.Uma descrição do(s) protocolo(s) entre as entidades N incluindo:
 - Uma descrição geral e informal da operação das entidades.
 - Detalhes adicionais, tais como considerações para melhorar o desempenho, sugestões para implementação, ou uma descrição pormenorizada que se aproxime de uma implementação.
 - Uma especificação do protocolo que inclua:
 - Uma lista dos tipos e formatos de mensagens trocadas entre as entidades N.
 - As regras que governam a reação de cada entidade N a comandos nas interfaces, mensagens de outras entidades e eventos internos.

Observe que o ponto 3 é redundante com a descrição da camada N-1; ele é incluindo para tornar a especificação do protocolo N independente das demais.

Uma especificação formal do protocolo N pode ser feita através do uso de várias técnicas. Estas técnicas são classificadas em três categorias: modelos de transição, linguagens de programação e modelos mistos.

8.2.2.1. Modelos de Transição

Nesta categoria classificam-se os modelos de Máquina de estados Finita – (MEF) [DANT 80], Redes de Petri [MERL 76], Linguagens Formais [TENG 78], os chamados Gráficos da UCLA [POST 74] e Colóquios [MEZZ 73], dentre os mais conhecidos. Destes, os mais populares são MEF e Redes de Petri.

André Danthine, afirma que as redes de petri são provavelmente mais indicadas no estágio inicial da especificação de protocolo [DANT 80]. Isso porque elas são mais próximas de uma linguagem natural e facilitam a identificação de uma nova situação ou “estado” para onde o protocolo migra, em resposta a ocorrência de eventos ou “transições” (simbolizando a emissão de comandos nas interfaces, chegadas de

mensagens ou estouros de temporização). Em estágios subseqüentes, porém, quando a especificação do protocolo é retocada e finalizada, a realização de MEF é mais vantajosa, pois propicia uma especificação mais compacta. Na realidade, uma MEF e redes de Petri são equivalentes e uma pode ser transformada na outra sem muita dificuldade. Aqui, escolhemos o modelo de MEF para ilustração dos modelos de transição. A razão para o nome desta categoria, como também o sentido em que são utilizados os termos transição e estado de um protocolo, ficarão melhor esclarecidos ao longo da nossa apresentação.

Uma MEF é formalmente definida como uma quíntupla (X, E, S, FPE, PS) onde X é um conjunto finito de estados, E um conjunto finito de entradas, S um conjunto finito de saídas ; FPE a função de próximo estado que mapeia os pares correntes (estado, entrada) no próximo estado, isto é, a função $(FPE: exX \rightarrow X)$, e PS é uma função de saída que mapeia os pares correntes (estado, entrada) na saída corrente, isto é, a função $(FS: ExX \rightarrow S)$. FPE é também chamada função de transição de estado.

Uma MEF ou se encontra num estado (estado presente) ou se encontra em transição para um próximo estado. Uma transição ocorre em resposta a uma nova entrada. O próximo estado para onde migra a MEF é determinada pela função FPE , que utiliza, na sua decisão, informações sobre o estado presente e da entrada recebida. A saída, fornecida pela MEF, no próximo estado é indicada pela função FS . A operação de uma MEF pode emular o funcionamento de um protocolo.

Como em uma MEF, o protocolo N permanece num certo estado até que seja excitado por alguma entrada (ex: comandos na camada $N+1$, chegada de mensagens da camada $N-1$, *timeouts*, etc). O protocolo responde à entrada fazendo algum processamento (modelado por uma transição MEF) e em seguida muda para um novo estado, produzindo uma saída, na forma de uma requisição de serviço da camada $N-1$ ou envio de uma mensagem para a camada $N+1$, inicialização de temporizadores, etc. Portanto, a construção de uma MEF correspondente a um protocolo N é uma maneira de especificar formalmente o protocolo.

Consideremos como exemplo do uso de MEFs o seguinte protocolo de enlace de dados que podemos imaginar como sendo utilizado entre duas interfaces de uma rede local. Cada interface é composta de uma parte emissora e de uma parte receptora. A emissora de uma interface conversa com a parte receptora da outra. Considerando-se um dos pares emissora - receptora, a emissora envia somente quadros de dados para a receptora que, por sua vez, manda apenas reconhecimentos de volta para a emissora, para sinalizar quadros de dados recebidos corretamente. O meio de transmissão não é livre de ruído e pode perder quadros ou entregá-los com erro. Um quadro recebido com erro é desprezado e equivale, portanto, a um quadro perdido. O controle de fluxo é do tipo envia-espera, isto é, após enviar um quadro de dados, a emissora aguarda a chegada de seu reconhecimento. Se decorrido um certo intervalo de tempo (chamado de intervalo de temporização), o reconhecimento ainda não tiver chegado, a emissora retransmite o quadro de dados. Após receber o reconhecimento, a emissora pode receber um outro quadro do nível superior e transmiti-lo. Para manter seqüenciamento de quadros de dados e de seus respectivos reconhecimentos, os valores 0 e 1 são utilizados. Assim, o quadro número 1 só é enviado após recebimento do reconhecimento para o quadro 0, e vice-versa. Os quadros recebidos corretamente do meio de transmissão pelas interfaces são passados para o nível superior (chamado "usuário", no que segue). Acabamos de especificar (parcialmente) o protocolo informalmente. A seguir apresentamos a especificação com o formalismo do modelo MEF.

Uma MEF, quando usada para modelar um protocolo, é denominada Máquina de Protocolo. Geralmente, um protocolo divide-se logicamente em duas partes, uma para cada entidade, implementando o protocolo. Assim, existem duas máquinas que modelam o protocolo do exemplo. Como neste exemplo, as duas entidades são simétricas (cada uma tem comportamento idêntico à outra), basta especificarmos uma única máquina de protocolo e lembrarmos que existe uma tal máquina para cada lado.

A máquina de protocolo está sempre num determinado estado, em qualquer instante de tempo. O estado consiste de todas as variáveis da parte receptora e do “estado principal” da máquina. Para a emissora, temos dois estados principais: “aguardando um quadro do usuário” e “aguardando um reconhecimento”. Além disso, temos na parte emissora uma variável binária indicando o próximo quadro a transmitir ou aguardando reconhecimento. Temos, então, para a parte emissora, quatro estados:

- Espera quadro do usuário que será transmitido como quadro 0.
- Espera quadro do usuário que será transmitido como quadro 1.
- Espera reconhecimento do quadro 0.
- Espera reconhecimento do quadro 1.

Para a receptora temos apenas dois estados:

- Aguardando quadro 0.
- Aguardando quadro 1.

O estado total da máquina de protocolo é o produto cartesiano dos estados da emissora e da receptora, já que cada entidade tem uma emissora e uma receptora. Formalizando, temos que X, o conjunto finito de estados é:

$$X = \{(EU, 0,0), (EU,0,1), (EU,1,0), (EU,1,1), (ER,0,0), (ER,0,1), (ER,1,0), (ER,1,1)\}$$

Onde EU significa “Esperando por um quadro de usuário” e ER significa “Esperando por um reconhecimento”. Os dois números no estado são variáveis binárias associadas à emissora e à receptora, respectivamente. Continuando com o formalismo da MEF, precisamos de E, o conjunto de entrada. Uma entrada, aqui, significa um evento passível de acontecer enquanto a máquina estiver num determinado estado, temos:

$$E = \{QU, ACK0IN, ACK1IN, DADO0IN, DADO1IN, TEMPORIZAÇÃO\}$$

Onde:

- QU = Quadro do Usuário;
- ACK0IN = recebimento de um reconhecimento para o quadro 0;
- ACK1IN = recebimento de um reconhecimento para o quadro 1;
- DADO0IN = recebimento do quadro de dados 0;
- DADO1IN = recebimento do quadro de dados 1;
- TEMPORIZAÇÃO = estouro do temporizador.

Observe que QU representa um evento ocorrendo na interface superior; ACK0IN; ACK1IN; DADO0IN; DADO1IN representam eventos ocorrendo na interface inferior; e TEMPORIZAÇÃO representa um evento interno. O conjunto

finito de saídas, S , representa as ações que devem ser feitas pelo protocolo em resposta a uma entrada. Neste caso, temos:

$$S = \{\text{ENVU}, \text{RECU}, \text{ACK0OUT}, \text{ACK1OUT}, \text{DADO0OUT}, \text{DADO1OUT}, \text{LT}, \text{DT}\}$$

Onde:

- ENVU = envia quadro ao usuário;
- RECU = recebe quadro do usuário;
- ACK0OUT = envia reconhecimento para o quadro 0;
- ACK1OUT = envia reconhecimento para o quadro 1;
- DADO0OUT = envia quadro de dados com numeração 0;
- DADO1OUT = envia quadro de dados com numeração 1;
- LT = liga temporizador;
- DT = desliga temporizador.

Utilizaremos um método gráfico para completar a especificação da máquina de protocolo. A seguinte convenção será adotada:

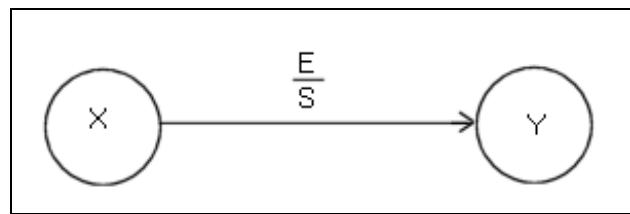


Figura 8.2 – Convenção para Método Gráfico

Onde a entrada E causa uma transição do estado X para o estado Y gerando uma saída S. O diagrama de estados para o protocolo considerado é fornecido na figura a seguir:

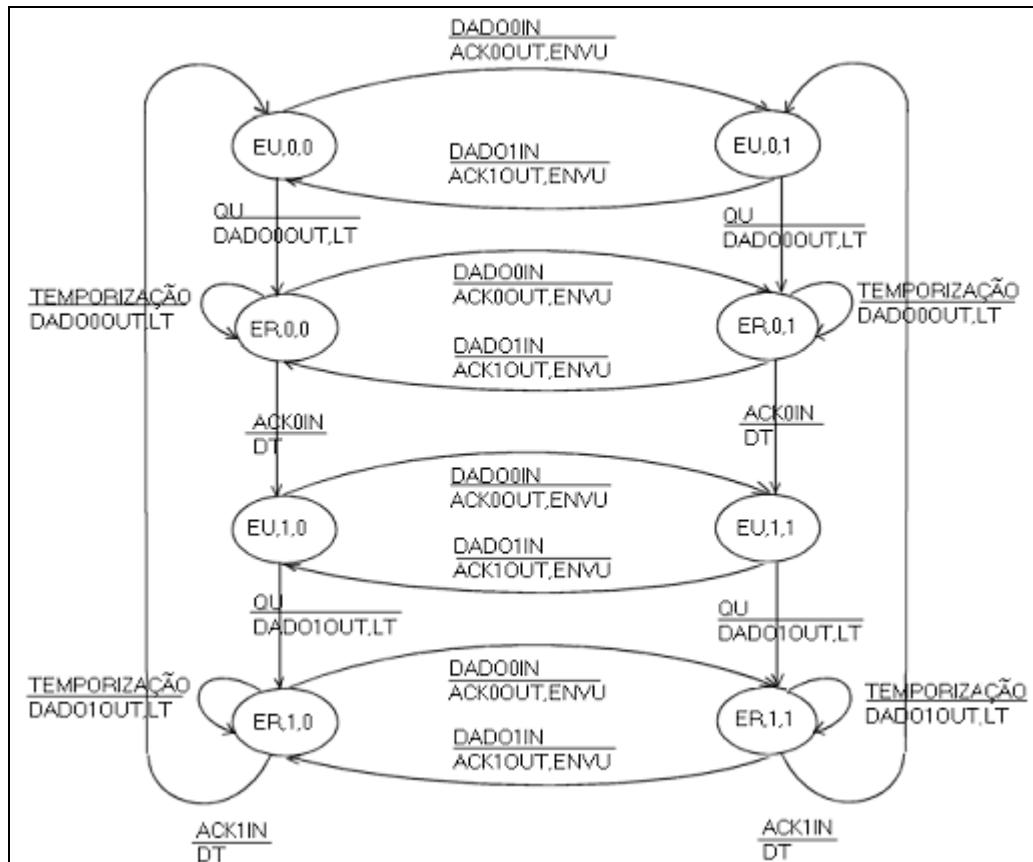
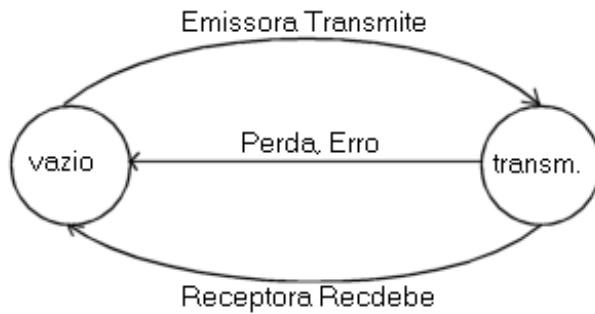


Figura 8.3 – Diagrama de Estados

Tomemos como exemplo o estado (EU, 0, 0): a emissora está esperando um quadro do usuário que, quando chegar, será enviado com numeração 0, e a parte receptora esta aguardando um quadro de dados com numeração 0. Há duas transições partir deste estado. Associada à parte emissora, a recepção de um quadro do usuário (QU) fará com que o quadro seja enviado e o temporizador será ligado; a máquina passa então ao estado (ER,0,0) para esperar o reconhecimento do quadro. No estado (EU,0,0), a parte emissora/receptora da entidade está aguardando o quadro de dados com numeração 0 e, quando este chegar um reconhecimento será enviado, o quadro recebido será passado para o usuário e a máquina passa para o estado (EU,0,1), onde a parte emissora não mudou, mas a parte receptora esta aguardando o quadro de dados 1.

Lembre-se agora que existe uma máquina de protocolo tal como mostrada para cada entidade comunicante (os dois lados da conexão). Num determinado instante, cada máquina pode estar num estado distinto, e há naturalmente uma dependência natural entre as duas máquinas. Por exemplo, a entrada DADO0IN numa máquina, depende da saída DADO0OUT na outra. Porém o fato do canal perder quadros implica que DADO0IN e DADO0OUT não tem dependência direta. A comunicação entre as duas máquinas se faz entre as duas MEFs adicionais, representando as duas direções do canal. Uma direção do canal pode estar em um de dois estados “vazio” ou “transferindo”. A MEF modelando cada direção do canal e mostrada na figura a seguir:



Porém nem tudo são flores. O método MEF apresenta uma dificuldade considerável quando o protocolo é mais sofisticado do que aquele que consideramos no exemplo. Esta dificuldade é causada pela “explosão do espaço de estados”. Para o protocolo que consideramos, o número total de estados era 8. Agora se permitíssemos um controle de fluxo em que o emissor pudesse continuar a enviar quadro de dados, até uma máxima W quadros, antes de parar para esperar pelo reconhecimento, então o número de estados realmente cresceria. Qual seria o número de estados para $W = 8$? Fica claro que mesmo para protocolos simples, desenhar o diagrama de estados e identificar todas as possíveis transições entre eles não é trivial, quando o número de estados é grande. Existem métodos para restringir a dificuldade proveniente da explosão do espaço de estados. Alguns deles são:

- Especificação parcial: a idéia é apenas especificar o que é principal, as transições principais, ignorando variáveis e parâmetros secundários.
- Junção de transições: a idéia aqui é combinar duas ou mais transições em uma única transição conjunta na especificação global do protocolo.
- Decomposição em subcamadas: a idéia é criar várias subcamadas para serviços específicos e desta forma diminuir a complexidade do protocolo de uma maneira global.
- Classificação de estados por asserções: este método identifica asserções que são predicados em um conjunto de estados. Na especificação global do protocolo, considera-se coletivamente os estados em cada um dos conjuntos formados, ao invés de individualmente. Isso reduz, assim, o número total de estados a serem considerados.
- Automação: a geração automática de estados e de transições entre eles reduz o esforço humano na especificação de um protocolo com muitos estados. A idéia neste método não é diminuir o número de estados mas, sim, simplificar o processo de geração do diagrama de estados.

Linguagens de Programação

À medida que um protocolo se torna mais complexo, a dificuldade de especificá-lo a través dos modelos de transição aumenta. Alguns métodos existem para amenizar esta dificuldade, mas a partir de um certo grau de complexidade do protocolo, a explosão de estados torna-se tão severa que a especificação através dos modelos de transição torna-se inviável. A especificação formal do protocolo tem então que ser feita através de outras técnicas; uma delas é por meio do uso de linguagens de programação.

Implementação de protocolos

Concluída a verificação da especificação de um protocolo, é a vez de implementá-lo nos vários sistemas da rede que irão utilizá-lo em suas comunicações. A implementação de um protocolo, em princípio, pode ser feita em hardware ou em software. No contexto de uma rede local, a implementação dos protocolos (1), (2) e (3) é freqüentemente realizada em hardware em torno de um microprocessador. Muitas das funções desses protocolos já são inclusive implementadas por pastilhas de circuito integrado. Antes que as questões pertinentes à implementação de um protocolo sejam endereçadas, é necessário decidir sobre certos pontos do protocolo deixados indefinidos em sua especificação. Isso não quer dizer que a especificação seja defeituosa ou incompleta. Nesse estágio ela já deve ter sido, inclusive, verificada. Tais pontos foram propositadamente deixados em aberto na especificação, para que as decisões tomadas pelo implementador reflitam peculiaridades e facilidades do sistema em que rodará a implementação e/ou para que as decisões possibilitem níveis desejados de desempenho para a operação do protocolo. Geralmente esses pontos tomam a forma de constantes ou primitivas na especificação. Por exemplo, a especificação de um protocolo de transporte para retransmissões e controle de fluxos não especifica o valor do intervalo do temporizador de retransmissão (apenas indica que o intervalo é uma constante, mas não seu valor). Tampouco define que estratégias específicas devem ser utilizadas para o controle de fluxos, gerenciamento de buffers, retransmissões etc. Isso toma a forma de nomes de primitivas, na especificação, sem maior detalhamento. O implementador, porém, deve decidir que valores e estratégias usar. O objetivo de primitivas é justamente o de agrupar as partes do protocolo que são dependentes da implementação. É importante observar que as decisões do implementador não afetam a operação correta do protocolo, isto é, não o impedem de prestar o serviço para o qual foi projetado em conjunto com outra entidade implementada de forma diferente, mas podem influir no seu desempenho. De qualquer forma, antes de implementar um protocolo, deve-se decidir sobre os pontos em aberto na sua especificação. É como se a especificação fosse refinada para incluir as decisões. Assim, passaremos a considerar os aspectos genéricos relacionados com as indefinições de um protocolo e a discutir opções para eliminá-las. Os aspectos de implementação propriamente dita são abordados em seguida.

Opções de implementação

As opções disponíveis ao implementador são limitadas basicamente por dois aspectos: as facilidades do sistema em que a implementação rodará e o desempenho almejado do protocolo. Conseqüentemente, uma discussão rigorosa e minuciosa do assunto requer a capacidade de avaliar o desempenho do protocolo em função de opções contempladas. Do ponto de vista apenas do desempenho, a opção selecionada seria a que oferece um melhor nível de desempenho.

Dentre os pontos não totalmente definidos na especificação de um protocolo, escolhemos os seguintes: valores de temporização, estratégias de retransmissão, de reconhecimento e de controle de fluxos, localização de usuários e alocação de recursos. A lista não é completa nem se aplica na sua totalidade a qualquer protocolo.

Ela serve, entretanto, para introduzir o assunto e ilustrar seus aspectos principais. Discutimos agora algumas possíveis opções para definição de cada um desses pontos.

Valores de temporização

Uma temporização de retransmissão, num protocolo N, indica o intervalo de tempo que uma entidade N deve esperar para transmitir uma PDU N não confirmada pela entidade N – par receptora. Um alto valor de temporização causará atrasos desnecessários na comunicação entre as entidades N. Por outro lado, um baixo valor de temporização pode causar retransmissões freqüentes, reduzindo-se a vazão útil da conexão usada na comunicação. É desejável, pois, que o valor da temporização de retransmissão seja escolhido de tal maneira a evitar esses problemas, forçando uma retransmissão apenas quando o reconhecimento da PDU N enviada “já devia ter chegado”. Para tanto, o valor escolhido deveria ser a soma dos tempos de trânsito da PDU N e de seu reconhecimento através das camadas inferiores de um protocolo no sistema emitente, do tempo de transmissão nos meios físicos, dos tempos de trânsito nas camadas inferiores mais atrasos de consumo da PDU N no sistema receptor, e do processamento necessário para gerar o reconhecimento de volta. Esses tempos são, em geral, variáveis aleatórias sujeitas a flutuações. Uma possibilidade é utilizar as medidas dos tempos acima, talvez conjugadas a um fator que considere as flutuações.

Estratégia de retransmissão

O número de retransmissões a serem feitas numa conexão antes que essa seja julgada defeituosa, é a tentativa de usá-la, abortá-la, nem sempre vem definido na especificação. Neste caso, cabe ao implementador escolher um valor. Observe-se que também neste caso a decisão deve considerar as implementações nos outros sistemas, pois o número de retransmissões pode influenciar a definição da temporização de desistência dos outros sistemas. A estratégia de retransmissões pode também incluir mecanismos para ajuste dinâmico do intervalo de retransmissão em função do tráfego na rede. Por exemplo, a função de retirada de protocolo de acesso aleatório ao meio de transmissão numa rede local é um de seus mecanismos. A escolha da função de retirada afeta o desempenho do protocolo, mas se observadas certas restrições, pode ser particularizada para cada sistema, sem afetar a operação correta do protocolo.

Estratégia de reconhecimento

O uso de reconhecimento é feito pelos mecanismos de controle de erro e de controle de fluxo. Em ambos os casos, a chegada de um reconhecimento íntegro inibe a retransmissão de PDUs e, consequentemente, reduz a utilização de recursos de comunicação, de processamento nos sistemas e de recursos de armazenamento usados pelos mecanismos de retransmissão. A estratégia de reconhecimento é executada ativamente pela parte receptora do protocolo; a parte emissora apenas consome passivamente, quando chegam, liberando os recursos de armazenamento utilizados pelas PDUs reconhecidas. Vejamos o modo como reconhecimentos podem ser gerados e enviados pela parte receptora do protocolo.

Uma opção de implementação da parte receptora é gerar um reconhecimento para cada PDU de dados recebida corretamente. Essa estratégia pode desperdiçar recursos de comunicação da rede e forçar a parte emissora do sistema remoto comunicante e dedicar muito esforço para o recebimento e processamento de reconhecimentos. Três alternativas para evitar estes problemas seriam:

- Σ concatenar reconhecimentos em outras PDUs, voltando para o sistema remoto;
- Σ reduzir o número de reconhecimentos de um por PDU, recebida para um por “X” PDUs – a emissão de um reconhecimento com número de seqüência N confirma o recebimento correto das PDUs números N, N-1, N-2, N-X + 1;
- Σ emitir um reconhecimento para todas as PDUs recebidas, dentro de um determinado intervalo de tempo – neste caso, uma temporização controla a emissão do reconhecimento e o intervalo de temporização é claramente dependente da temporização de retransmissão da entidade emissora.

A parte receptora tem ainda que definir como proceder para “emitir” o reconhecimento na chegada de uma PDU. Aqui temos basicamente duas opções: 1 emitir o reconhecimento somente após passar a PDU para a camada superior de protocolo (ou usuário) ou, então, 2 emitir imediatamente o reconhecimento. A opção 1 dá margem para a implementação descartar PDUs ainda não reconhecidas, liberando recursos quando necessitados. A opção 2 responsabiliza a implementação pela guarda da PDU, impedindo-a de jogar com os buffers de armazenamento.

Implementação como Processo do Usuário

Um processo de usuário, na maioria dos sistemas operacionais, propicia um “ambiente de execução” para os programas dos usuários do sistema. Uma maneira “natural” de implementar um protocolo é, pois, codificar sua especificação na forma de um programa, requisitar um processo do sistema operacional e rodar o programa no processo obtido. Esta solução pode, entretanto, ser lenta e não atender ao objetivo 2, pois o usuário, em geral, tem pouco controle sobre a prioridade com a qual ele será escalonado pelo sistema operacional. Por outro lado, ela não exige do programador conhecimentos aprofundados do núcleo do sistema operacional.

Implementação no Núcleo do Sistema Operacional

A rede, do ponto de vista do sistema local, pode ser considerada como um dispositivo periférico (de comunicação) tal qual terminais, unidades de disco, impressora etc. “Drivers” para esses dispositivos geralmente residem no núcleo do sistema operacional e os processos sendo executados (processos de usuário) manipulam tais dispositivos através de chamadas ao sistema operacional. A implementação do protocolo, sob esta visão da rede, poderia ser feita como um driver residindo no núcleo. A interação com as outras camadas de protocolo poderia ser então realizada através de operações de leitura (“read”) e escrita (“write”), via um conjunto especial de chamadas do sistema operacional. Esta solução, evita a dependência do escalonamento de processos para execução da implementação, facilitando o alcance do objetivo 2, e simplifica o projeto de aplicação que usa a rede. Por outro lado, o implementador deve ser um “expert” no sistema operacional; a

execução do “driver” do protocolo pode degradar sensivelmente a resposta do sistema local ou o driver pode ainda não caber no espaço de memória reservado para o núcleo.

Implementação num Processador “Front-End”

Se as limitações das soluções anteriores descartam sua adoção, pode-se optar por implementar o protocolo num processador “front-end” dedicado. Esta solução isola a implementação do núcleo do sistema operacional da máquina hospedeira, evita o consumo excessivo de recursos do hospedeiro, e o processador front-end pode ainda ser compartilhado entre vários hospedeiros. O custo do processador pode, entretanto, não ser suportável, e ainda resta o problema de interconecta-lo à máquina hospedeira. Esta interconexão pode inclusive tomar-se em gargalo, degradando o desempenho das interações entre as várias camadas de protocolo.

Um último ponto a ser considerado, independentemente da forma de integração adotada, é o controle de fluxo na interface com o usuário (ou a camada superior) da implementação. Dois são os aspectos desse controle de fluxo: o controle da taxa em que o usuário inicia as chamadas de serviço para a implementação, e o controle da passagem de informações entre a implementação e seu usuário.

A taxa na qual o usuário inicia chamadas de serviço deve ser controlada para evitar que as entidades no protocolo implementado sejam congestionadas com muitos pedidos. Tal controle pode ser efetuado automaticamente, em decorrência dos limites do mecanismo para comunicação interprocessos utilizado para implementar as primitivas de serviço, ou mecanismos especiais podem ser introduzidos na implementação para este fim. A implementação deve também prover meios de controlar o tamanho de unidades de dados que ela manipulará e a taxa na qual tais unidades são submetidas a ela e entregues ao usuário. Um exemplo de como construir um mecanismo de controle de fluxo na interface de uma implementação é apresentado em [QUEI 84]. Neste caso, filas de “entrada” e “saída” da implementação são definidas onde são depositados os pedidos e prestações de serviço do usuário e da implementação respectivamente. O comprimento finito destas filas limita automaticamente o número das interações através da interface e tanto o usuário quanto a implementação podem consumir os itens nas filas, nas taxas convenientes. O tamanho dos dados pode também ser facilmente controlado para conveniência, tanto do usuário, quanto da implementação, através da passagem de um descritor; (ex.: um apontador de arquivo contendo os dados). De posse do descritor, a implementação e o usuário podem ler e escrever os dados no arquivo em “pedaços” de tamanho adequado. Três maneiras de como implementar esta solução são apresentadas em [NBS 83].

Neste estágio, todos os pontos em aberto na especificação foram definidos, e decidiu-se sobre o modo de integrar a implementação no sistema local. Resta fazê-la.

MODELO GERAL DE IMPLEMENTAÇÃO

Discutiremos aqui um modelo sugerido em [NBS 81], para a implementação de protocolos. O modelo é geral, sendo válido para qualquer implementação realizada em software, para execução em qualquer sistema. A estruturação do modelo

assemelha-se à estrutura da especificação formal do protocolo e tem a vantagem de possibilitar a implementação e as suas interações.

Cada um dos módulos do programa-implementação corresponde univocamente a uma transição, ou a um procedimento, primitiva ou evento das interfaces definidas na especificação. Outros módulos do modelo refletem as escolhas feitas pelo implementador para “refinar” a especificação, como, por exemplo, controle de fluxo na interface, escalonamento das várias atividades do protocolo, multiplexação de atividades concorrentes nas diversas entidades da camada etc. Caso o protocolo envolva o controle de várias conexões, existirá uma entidade para controlar cada uma delas. A informação sobre o estado de cada entidade é guardada numa estrutura de dados chamada “Estrutura de Controle de Entidades”.

A implementação é excitada pela chegada de eventos das camadas adjacentes. Ao chegar um evento, ele é identificado pelo “Processador de Interface” para determinar qual a entidade de destino que deverá processá-lo. Em seguida, o processador de interface escalona o evento para atendimento pelas “Rotinas de Eventos de Interface”. A rotina de eventos de interface apropriada então constrói uma Estrutura de Dados com informações sobre o evento de interface particular e chama o “escalonador de Funções de Transição”.

De posse do tipo de evento de interface e da identificação da entidade-destino, o escalonador examina o estado dessa entidade na estrutura de controle de entidades. O estado presente da entidade e o tipo de evento servem de índice na “tabela de Transições” para obter uma lista de transições possíveis, em ordem de prioridade. O escalonador então chama o módulo responsável para processar o tipo de transição com a prioridade mais alta. Se nenhuma transição é aplicável (a lista se encontra vazia), uma indicação de erro é enviada de volta e o evento descartado.

O módulo chamado pode, por sua vez – em função das ações especificadas para a transição a ser processada – chamar procedimentos, invocar primitivas ou gerar eventos para as camadas adjacentes (cada um desses últimos representados por um módulo no modelo).

A geração desses eventos é feita pelas rotinas de eventos de interfaces que constroem as estruturas de dados associadas e chamam o processador de interface para despachar os eventos para a camada apropriada.

IMPLEMENTAÇÃO AUTOMÁTICA

A partir do modelo da especificação do protocolo, a sua implementação pode ser feita, escrevendo-se o código para cada módulo do programa-implementação. A atividade nesta fase consiste simplesmente do esforço de programação. Idealmente, a especificação formal do protocolo – principalmente, se feita por uma linguagem de programação – deve facilitar razoavelmente a sua “tradução” para uma implementação executável. Melhor seria se pudéssemos ter a produção automática da implementação, ou pelo menos de alguns de seus módulos. Isto é possível.

Naturalmente que a totalidade da implementação não pode ser gerada automaticamente. Os módulos que dependem das escolhas do implementador para os pontos em aberto na especificação não podem ter sua codificação automatizada. A “forma” destes módulos é definida em função das inclinações do implementador, das características da máquina hospedeira e de seu sistema operacional. Isto força o implementador a escrever manualmente o código para esses módulos. De nossas discussões anteriores, podemos identificar os módulos com codificação manual como

sendo o processador de interface, as rotinas de eventos de interface e os módulos de primitivas de serviço. Em geral, tudo que depende das interfaces com a camada deve ser implementado manualmente.

Os módulos de processamento de transições, procedimentos, escalonador de funções de transição e a lista de transições podem ser gerados automaticamente a partir dos segmentos da especificação formal referentes a estes módulos. Convém observar que para tanto, deve-se separar cuidadosamente as escolhas a serem feitas pelo implementador da especificação desses segmentos e o modelo de implementação deve ser modularizado para isolar claramente os módulos com geração automática. Métodos para produzir implementações a partir de especificações formais de protocolos são discutidos nas referências fornecidas, relativas ao assunto. Em [NBS 81], discuti-se um tal método, na forma de um compilador de especificação formal, que lê a especificação com declarações de dados e produz uma função na linguagem de programação C, para cada transição e procedimento. Para se ter uma idéia da aplicabilidade do método, as classes 2 e 4 do protocolo de transporte do NBS (praticamente idêntico ao da ISO) foram implementadas usando a organização do modelo geral de implementação; a implementação contém aproximadamente 10000 linhas de código na linguagem de programação C, 60% quais geradas automaticamente ou aproveitáveis para qualquer protocolo.

TESTES DE IMPLEMENTAÇÃO DE PROTOCOLOS

A implementação de um protocolo, como de qualquer outro algoritmo, pode conter erros de lógica. Uma implementação logicamente errada pode até ser executável, mas, em geral, ela produzirá resultados inesperados e, pior, indesejáveis. Para que isto não aconteça, é necessário que a implementação seja testada para, senão garantir, aumentar nossa confiança de que sua execução alcançará os objetivos do protocolo. Esta seção trata de testes de implementações de protocolos.

Como visto na última seção, a implementação de um protocolo pode ser feita em hardware ou em software. A implementação em hardware é de menos interesse aqui porque ela é geralmente restrita à sub-rede de comunicação (camada (1), (2) e (3) do RM-OSI) cujos componentes e/ou serviços já são disponíveis comercialmente, tornando os seus fornecedores responsáveis pelos testes. As implementações em software dos protocolos (4) a (7) nos são mais interessantes. Isto porque estes protocolos, sendo de responsabilidade dos usuários da rede, provavelmente serão implementados como programas executáveis nos vários sistemas na rede. Assim, supomos implicitamente implementações em software de protocolos de alto nível. Esta suposição, porém, não é restritiva desde que o conteúdo da seção pode ser aplicável sem maiores dificuldades às implementações de outras camadas do RM-OSI, mesmo quando realizadas em hardware. O teste da implementação de um protocolo é feito para determinar se ela conforma com a especificação (já verificada) do protocolo. O teste de conformidade é positivo se o software implementando o protocolo executa as funções e provê os serviços do protocolo de acordo com a sua especificação.

GERAÇÃO DE SEQÜÊNCIAS DE TESTES

As seqüências de testes devem ser elaboradas de tal forma a endereçar aspectos salientes dos serviços e dos mecanismos do protocolo para provar se eles são

providos corretamente pela implementação em teste. Os critérios essenciais a serem atendidos, na elaboração de cada teste, na seqüência, são:

- (1) O teste tem um objetivo claro, podendo-se declarar precisamente o que será alcançado com sua realização.
- (2) O resultado do teste é conhecido “a priori”.
- (3) O teste é peculiar e não uma variante trivial de outro teste na seqüência. Isto não implica que testes distintos não possam usar a mesma seqüência de testes para alcançar seu objetivo. Na verdade, testes de invariância de comportamento requerem a sua repetição.

Capítulo

9

Redes Locais, Ethernet e Internet

Definição:

Conjunto de sistemas de computação independentes, autônomos, interconectados, para permitir a cada sistema utilizar todos os recursos dos outros sistemas, chamando suas próprias sub-rotinas.

Como foi dada grande atenção no decorrer das aulas a interface de comunicação serial RS232C. Vamos iniciar nosso estudo de redes de computadores partindo do uso desta interface em redes locais de computadores pessoais.

Todo computador pessoal tem a disposição uma placa de comunicação serial RS-232C que tem o objetivo de possibilitar ligação ponto a ponto e que através de um circuito adaptador pode possibilitar uma rede.

O circuito adaptador para possibilitar uma rede pode ser simples como o mostrado na figura A. Onde é apresentado um diodo por porta, dois resistores de terminação e uma fonte de -12V.

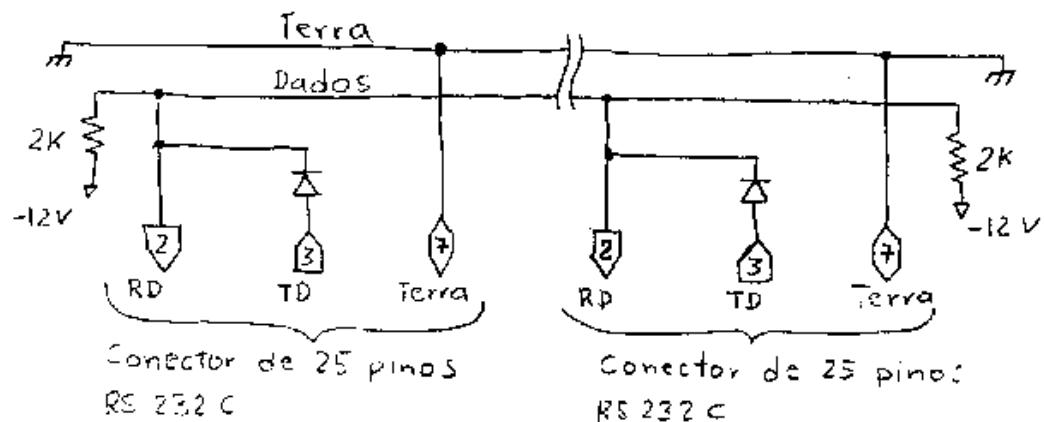


Figura A - Circuito adaptador simples.

Para melhorar o circuito apresentado na figura A basta usar um cabo blindado (recordar o circuito do RS 232 entendido).

Pelo circuito da figura A quando caracteres são enviados de um terminal, eles são recebidos em todos os terminais da rede, incluindo o terminal de onde foram enviados. Entretanto, se dois ou mais terminais enviam diferentes mensagens ao mesmo tempo. Cada terminal que transmite receberá outra mensagem que é diferente da enviada; esta mensagem será a lógica OR das mensagens enviadas. Isto permite uma importantíssima propriedade que é a detecção de colisão.

Aqui usa-se o fato de que um terminal RS-232 mantém sua linha de transmissão em voltagem negativa quando não transmite, e então pulsa a linha de transmissão para voltagem positiva no início de um caractere (bit de start).

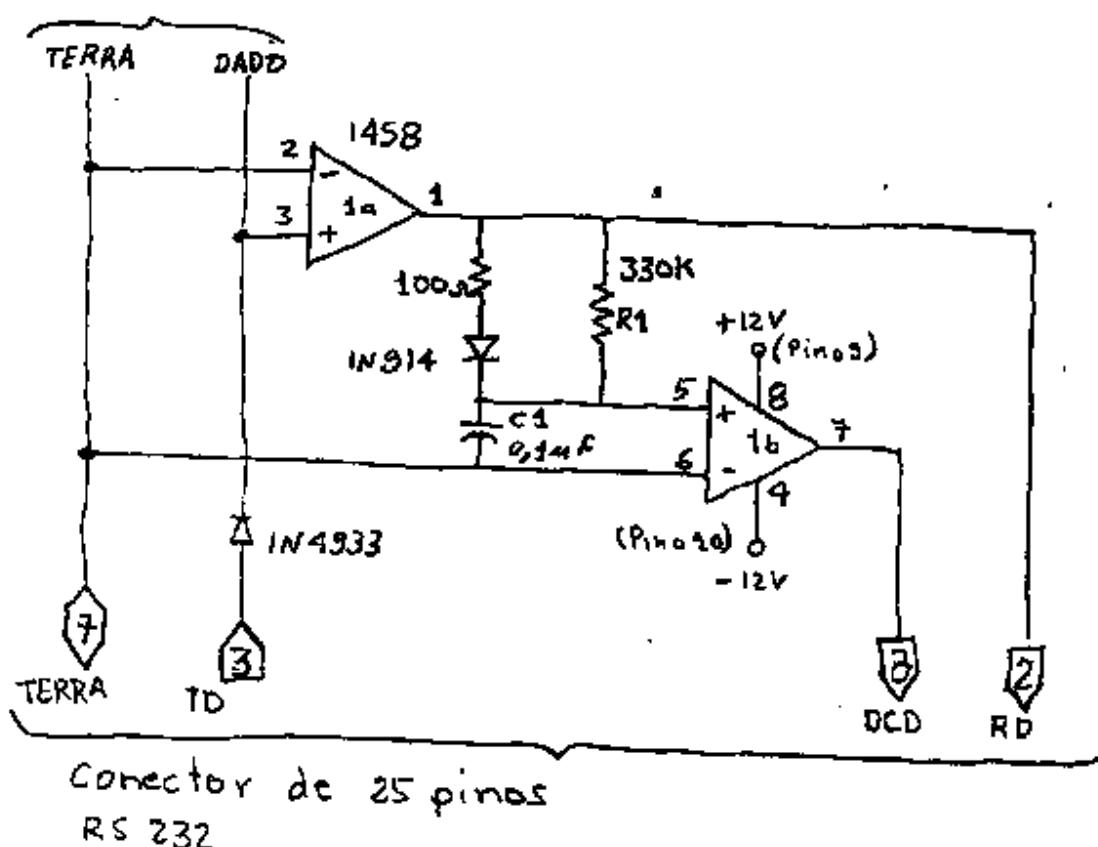
O padrão RS-232 define o nível positivo como um “0” transmitido e o nível negativo como um binário “1”. Em outras palavras, o caractere começa com um “0”, seguido pelo byte do código transmitido. Pelo menos um binário “1” é inserido depois de cada byte, e ele é chamado de stop bit.

O resistor de terminação fornece o nível negativo e cada porta RS232 pode levar a linha para o nível positivo pelo pulso de início de um caractere. Em termos de bits, o resistor fornece os “1s” e a porta fornece os “0s”.

Os limites de velocidade e distância surgem da combinação das limitações do “driver” de corrente e da carga da saída receptora da rede.

Limits: Distância, velocidade e número de receptores.

Por exemplo: Com a figura A pode-se usar a rede em 19.200 BPS para seis dispositivos separados de 20 pés, ou pode-se conectar três dispositivos com duas milhas de fio e fluir 300 BPS.



Algumas modificações simples podem ser realizadas para expandir a capacidade da rede. Na figura B é mostrado um circuito que possibilita maior número de saídas receptoras.

Na figura B um amplificador operacional é usado para “bufferizar” um sinal que esta chegando. Isto reduz a carga localizada de cada “nó” na rede, permitindo um virtualmente ilimitado número de “nós” na rede.

O sinal que indica se a rede está ocupada é fornecido pelo resistor R1 e capacitor C1. A finalidade deste sinal é de desobrigar o software de trocar a condição da rede. O circuito trabalha pela carga de C1 durante o bit de “start”

do caracter. O capacitor então se descarregará por R1. A escolha destes dois valores é feita pela mais baixa taxa de dados usada na rede. A figura B tem valores de R1 e C1 para 1200 BPS.

Tabela com valores de R1 relacionados com a taxa de transferência:

Taxa de transferência (BPS)	Valor do R1 (k Ohms)
1200	330
2400	160
4800	82
9600	39
19.2k	22
38.4k	10
76.8k	5.1
153.6k	2.2

Com este exemplo podemos começar a perceber alguns conceitos básicos tratados em redes de computadores.

Por Exemplo:

Antes de uma estação transmitir ela sente o estado da rede pelo pino 8 verificando se a mesma está ocupada ou não. Isto faz parte do protocolo de acesso ao meio.

Outro fato a observar é que é possível detectar colisão de mensagens caso duas estações transmitirem ao mesmo tempo e então fazer com que uma estação transmita entes da outra.

O protocolo de acesso a meio é dividido em dois grupos:

Não Determinísticos:

- CSMA (Carrier Sense Multiple Access) ouvir antes de falar.
- CSMA/CD (CSMA com Colision Detection) ouvir enquanto fala.
- CSMA/CD-OB (CSMA/CD com Ordely Backoff) retransmissão escalonada.

Determinístico:

- Marca de controle (token).

TCP/IP

9.1 Introdução e visão geral

9.1.1 A motivação para a interligação em redes

A comunicação de dados tornou-se parte fundamental da computação. As redes de abrangência mundial reúnem dados sobre assuntos diversificados como condições atmosféricas, produção de safra e tráfego aéreo. Os grupos estabelecem listas de correio eletrônico para que possam compartilhar informações de interesse comum. As pessoas que cultivam hobbies promovem intercâmbio de programas para seus computadores de uso pessoal. No meio científico, as redes de dados são essenciais porque permitem que os cientistas enviem programas e dados a supercomputadores remotos para processamento, com o intuito de recuperar os resultados e trocar informações com seus colaboradores.

Lamentavelmente, a maioria das redes constitui entidades independentes estabelecidas para atender às necessidades de um grupo isolado. Os usuários selecionaram uma tecnologia de hardware que seja adequada aos seus problemas de comunicação. É impossível a estruturação de uma rede universal com base em uma única tecnologia de hardware, já que nenhuma rede única atende a todas as aplicações. Alguns usuários precisam de uma rede de alta velocidade para conectar-se a máquinas, mas essas redes não podem ser expandidas para alcançar grandes distâncias. Outros preferem uma rede de velocidade inferior que faça conexão com máquinas a centenas de milhas de distância.

Nos últimos 15 anos, foi desenvolvida uma nova tecnologia para possibilitar a interconexão de muitas redes físicas diferentes e fazê-las operar como uma unidade coordenada. Essa tecnologia, denominada interligação em redes, acomoda distintas tecnologias básicas e hardware, proporcionando uma forma de interconectar redes heterogêneas e um conjunto de convenções que possibilitam as comunicações. A tecnologia de interligação em redes esconde os detalhes de hardware de rede, e permite que os computadores se comuniquem independentemente de suas conexões físicas.

A tecnologia de interligação em redes que será descrita neste livro é um exemplo de interconexão de sistema aberto porque, ao contrário dos sistemas de comunicação patenteados, disponíveis por determinado fornecedor, as especificações estão disponíveis publicamente. Por esse motivo, qualquer pessoa está apta a desenvolver o software necessário para estabelecer comunicação de interligação em redes. O mais importante é que toda a tecnologia foi projetada para estimular a comunicação entre máquinas de arquitetura de hardware distintas, para utilizar qualquer hardware de comutação de pacotes, e para aceitar diversos sistemas operacionais.

Para avaliar a tecnologia de interligação em redes, pense no quanto ela influencia uma atividade profissional. Considere, por exemplo, o que resulta da

interconexão dos computadores utilizados pelos cientistas. Qualquer cientista poderá promover o intercâmbio dos dados resultantes de um experimento com outro cientista. Os centros nacionais podem coletar dados de um fenômeno natural e torná-los disponíveis para todos os cientistas. Os programas e serviços de computadores disponíveis em um determinado local podem ser utilizados por cientistas de outras localidades. Em consequência, aumenta a velocidade com que se desenvolvem as investigações científicas. E as mudanças são drásticas.

9.1.2 A interligação em redes TCP/IP

Ao longo dos anos, as agências governamentais norte-americanas perceberam a importância e o potencial da tecnologia de interligação em redes e vêm financiando as pesquisas que possibilitaram a interconexão global de redes. A tecnologia da ARPA inclui um conjunto de padrões de rede que especificam os detalhes de sistema pelo qual os computadores se comunicam, bem como um conjunto de convenções para interconexão em rede e para roteamento. Denominado oficialmente Pilha de Protocolos de interligação em redes TCP/IP, e geralmente citado como TCP/IP, essa pilha pode ser utilizada para comunicação em qualquer conjunto de redes interconectadas. Algumas empresas, por exemplo, utilizam o TCP/IP para interconectar todas as redes de sua organização, ainda que a empresa não se comunique com redes externas. Outros grupos utilizam o TCP/IP para estabelecer comunicações entre sites geograficamente distantes.

Embora a tecnologia TCP/IP seja, por si só, notável, ela é especialmente interessante porque sua viabilidade foi demonstrada em larga escala. Constitui a tecnologia de base para uma interligação em redes global que conecta domicílios, campus universitário e outras escolas, organismos e laboratórios do governo em 61 países. Nos Estados Unidos, a National Science Foundation (NSF), o Department of Energy (DOE), o Department of Defense (DOD), a Health and Human Service Agency (HHS) e a National Aeronautics and Space Administration (NASA) participaram do financiamento da internet e utilizaram o TCP/IP para conectar muitas de suas instalações de pesquisa. Conhecida como internet ARPA/NSF, internet TCP/IP, internet global, ou simplesmente como internet, a interligação em redes resultante permite que os pesquisadores de instituições conectadas compartilhem informações com seus colegas de todo o mundo com a mesma facilidade com que compartilham informações com pesquisadores da sala ao lado. Um sucesso extraordinário, a internet demonstra a viabilidade da tecnologia TCP/IP e mostra como se pode lidar com uma diversidade de tecnologia de redes.

9.1.3 Serviços de interligação em redes

É impossível avaliar os detalhes globais que existem por trás do TCP/IP sem compreender os serviços que ele fornece. Esta seção faz uma ligeira análise dos serviços da interligação em redes, evidenciando os serviços que a maioria dos usuários acessa, deixando para capítulos posteriores a abordagem de como computadores se interconectam via TCP/IP, e de como a funcionalidade dos serviços é implementada.

Grande parte de nossa abordagem de serviços terá como foco padrões denominados protocolos. Protocolos como TCP e IP fornecem as regras para a comunicação. Eles contêm os detalhes de formatos de mensagens, descrevem o que um computador faz ao receber uma mensagem e especificam como um computador trata os erros ou outras condições anormais. O mais importante é que esses protocolos permitem que tratemos a comunicação através do computador, independente do hardware da rede de qualquer fornecedor em particular. De certa forma, os protocolos estão para a comunicação assim como os algoritmos estão para a computação. Um algoritmo permite que uma pessoa especifique ou compreenda a computação sem conhecer os detalhes de um conjunto de instruções de uma CPU em especial. Da mesma forma, um protocolo de comunicação permite que alguém especifique ou entenda uma comunicação de dados sem depender de conhecimentos minuciosos do hardware da rede de um fornecedor específico.

Ocultar os detalhes de baixo nível de comunicação facilita a melhoria da produtividade de diversas formas. Primeiro, os programadores lidam com abstrações de alto nível, não precisando aprender ou lembrar-se de todos os detalhes sobre determinada configuração de hardware. Assim sendo podem-se criar novos programas rapidamente. Segundo, como os programas desenvolvidos com abstrações de alto nível não estão restritos a uma arquitetura de máquina específica ou a um hardware de rede específico, eles não precisam ser mudados quando as máquinas ou as redes forem reconfiguradas. Terceiro, como os programas aplicativos desenvolvidos com protocolos de alto nível são independentes do hardware, eles podem promover uma comunicação direta entre um par de máquinas arbitrário. Os programadores não precisam elaborar versões especiais de software aplicativo para mover e traduzir dados entre cada par de máquinas possíveis.

Veremos que todos os serviços de rede são descritos por protocolos. As próximas seções referem-se a protocolos utilizados para especificar serviços em nível de aplicativos, bem como aqueles utilizados para definir serviços em nível de rede. Os últimos capítulos explicam com mais detalhes cada um desses protocolos.

9.1.3.1 Serviços de interligação em redes em nível de aplicativos

Do ponto de vista do usuário, uma interligação em redes TCP/IP aparenta ser um conjunto de programas aplicativos que utilizam a rede para desempenhar tarefas de comunicação consideradas necessárias. Usamos a expressão capacidade de interoperabilidade para definir a capacidade de sistemas computadorizados diferentes participarem da solução de problemas computacionais. Os programas aplicativos de interligação em redes demonstram alto grau de capacidade de interoperabilidade. A maioria dos usuários que acessa a internet o faz simplesmente executando programas aplicativos sem compreender a tecnologia do TCP/IP, a estrutura da interligação em redes principal, ou mesmo o caminho por onde os dados trafegam para seu destino; eles lançam mão de programas aplicativos e do software da rede principal para tratar desses detalhes. Apenas os programadores que desenvolvem programas aplicativos de rede precisam considerar a interligação em redes e ter certo entendimento da tecnologia.

Os serviços aplicativos da internet mais comuns e difundidos incluem:

- *Correio eletrônico.* O correio eletrônico permite que um usuário elabore memorandos e os envie a indivíduos ou grupos. Uma outra parte do aplicativo do correio eletrônico permite que os usuários leiam os memorandos que receberem. O correio eletrônico tem sido tão bem-sucedido que muitos usuários da internet dependem dele para correspondência comercial normal. Embora existam muitos sistemas de correio eletrônico, a utilização do TCP/IP faz com que a entrega de correio seja mais confiável, já que não depende de computadores para processamentos intermediários na transmissão de mensagens. Um sistema de entrega de correio TCP/IP opera através do contato direto entre a máquina do transmissor e a máquina do receptor. Assim, o transmissor sabe que quando a mensagem deixa a máquina local, ela foi recebida com êxito no destino.
- *Transferência de arquivos.* Embora alguns usuários às vezes transfiram arquivos através do correio eletrônico, ele se destina, sobretudo, a mensagens de pouco texto. Os protocolos TCP/IP incluem um programa aplicativo que permite que os usuários enviem ou recebam arbitrariamente arquivos externos de programas de dados. Ao utilizar, por exemplo, um programa de transferência de arquivos, a pessoa pode copiar de uma máquina para outra uma base de dados extensa contendo imagens de satélite, um programa escrito em Pascal ou C++, ou um dicionário de inglês. O sistema indica uma maneira de checar os usuários autorizados, ou até de evitar acessos. Tal como ocorre com o correio eletrônico, a transferência de arquivos na interligação em redes TCP/IP é confiável porque as duas máquinas envolvidas comunicam-se diretamente, sem depender de máquinas intermediárias que façam cópias do arquivo ao longo do processo.
- *Login remoto.* O login remoto permite que, de seu computador, um usuário entre em conexão com uma máquina remota e estabeleça uma sessão interativa de login. O login remoto faz com que uma janela na tela do usuário pareça conectar-se diretamente com a máquina remota, enviando cada toque no teclado a uma máquina remota e exibindo cada caractere que o computador remoto imprimiu na janela do usuário. Quando a sessão de login remoto termina, o aplicativo retorna o usuário ao sistema local.

Voltaremos a falar sobre esses e outros aplicativos nos próximos capítulos, a fim de analisá-los com maiores detalhes. Veremos exatamente como eles utilizam os protocolos básicos TCP/IP, e por que a existência de padrões para protocolos de aplicativos contribuiu para assegurar sua disseminação.

9.1.3.2 Serviços de interconexão no nível da camada de rede

Um programador que escreve programas aplicativos que utilizam protocolos TCP/IP possui uma visão totalmente diferente de uma interligação em redes, se comparado a um usuário que simplesmente executa aplicativos como correio eletrônico. No nível da camada de rede, uma interconexão proporciona dois extensos tipos de serviços que todos os programas aplicativos utilizam. Embora não seja importante, neste momento, compreender os detalhes desses serviços eles não podem ser omitidos de qualquer visão geral do TCP/IP:

- *Serviço de entrega de pacotes sem conexão.* Este serviço, explicado com detalhes ao longo do texto, forma a base para todos os serviços de interligação em redes. A entrega sem conexão constitui uma preocupação do serviço

oferecido pela maioria das redes distribuidoras de encomendas. Isso simplesmente significa que a interligação em redes TCP/IP promove o roteamento de pequenas mensagens de uma máquina para outra, com base nas informações do endereço contidas na mensagem. Como o serviço sem conexão promove o roteamento de cada pacote separadamente, não há garantia de entrega, e nem de entrega na mesma ordem na qual os pacotes foram transmitidos. Já que quase sempre há um mapeamento direto para o hardware, o serviço sem conexão é extremamente eficiente. O mais importante é que a entrega de pacotes, sem conexão, com base para todos os serviços de interligação em redes, torna os protocolos TCP/IP adaptáveis a uma ampla gama de hardware de redes.

- *Serviço de transporte de streams confiáveis.* A maioria dos aplicativos precisa de muito mais do que uma entrega de pacotes, porque eles exigem que o software de comunicação corrija automaticamente erros de transmissão, pacotes perdidos, ou falhas de comutações ao longo do caminho entre o transmissor e receptor. O serviço de transporte confiável trata desses problemas. Ele permite que um aplicativo de um computador estabeleça uma “conexão” com um aplicativo de outro computador, e a seguir envie um grande volume de dados através da conexão, como se fosse uma conexão de hardware direta e permanente. Naturalmente, em um nível mais baixo, os protocolos de comunicação dividem a cadeia de dados em mensagens curtas e as envia, uma de cada vez, esperando que o receptor confirme a recepção.

Muitas redes oferecem serviços básicos semelhantes aos mencionados acima, de modo que alguém poderia questionar o que diferencia os serviços TCP/IP de outros. As principais características diferenciadoras são:

- *Independência da tecnologia de redes.* Embora o TCP/IP seja baseado em tecnologia convencional de comutação de pacotes, ele é independente do hardware de qualquer fornecedor específico. A internet global inclui diversas tecnologias de rede, desde as redes projetadas para operar em um prédio até as projetadas para cobrir grandes distâncias. Os protocolos TCP/IP definem a unidade de transmissão de dados denominada *datagrama*, e especifica como transmitir datagramas em uma rede específica.
- *Interconexão universal.* Uma interligação em redes TCP/IP permite a comunicação de que qualquer par de computadores ao qual ela é conectada. A cada computador é atribuído um endereço universalmente reconhecido por toda a interligação em redes. Cada datagrama traz os endereços de sua origem e de seu destino. Os computadores de comutação intermediária utilizam o endereço de destino para tomar decisões sobre roteamento.
- *Confirmações fim-a-fim.* Os protocolos de interligação em redes TCP/IP fornecem uma confirmação entre a origem e o destino final, e não entre máquinas sucessivas ao longo do caminho, mesmo quando as duas máquinas não se conectam a uma mesma rede física.
- *Padrões de protocolo de aplicativos.* Além dos serviços básicos no nível de transporte (como conexões de streams confiáveis), os protocolos TCP/IP incluem padrões para muitos aplicativos comuns, inclusive o correio eletrônico, a transferência de arquivos e o login remoto. Assim, quando estão

desenvolvendo programas aplicativos que utilizam TCP/IP, os programadores sempre descobrem que o software existente oferece os serviços de comunicação de que eles precisam.

9.1.4 História e escopo da Internet

O que faz com que a tecnologia TCP/IP seja tão notável deve-se, em parte, à sua utilização quase universal, e também à dimensão e à taxa de crescimento da internet. A ARPA iniciou suas atividades voltando-se para uma tecnologia de interligação em redes em meados da década de 1970, e a arquitetura e os protocolos adquiriram sua forma atual por volta de 1977-79. Nessa ocasião, a ARPA era conhecida como a primeira agência a financiar a pesquisa de redes de comutação de pacotes e como a pioneira de muitas idéias sobre comutação de pacotes por meio de sua famosa ARPANET. A ARPANET utilizava a interconexão convencional de linha ponto a ponto pelo sistema de leasing, mas a ARPA também financiou a exploração de comutadores de pacotes em redes de rádio e canais de comunicação via satélite. Na realidade, a crescente diversidade de tecnologias de hardware de rede contribuiu para forçar a ARPA a estudar a interligação em redes e impulsionar a efetivação de uma interligação em redes.

A disponibilidade de recursos financeiros para pesquisa, pela ARPA, despertou a atenção e aguçou a imaginação de vários grupos de pesquisa, especialmente daqueles cujos membros tiveram experiência anterior com a comutação de pacotes na ARPANET. A ARPA programou reuniões informais com pesquisadores a fim de trocar idéias e discutir os resultados das experiências. Por volta de 1979, tantos eram os pesquisadores envolvidos no desenvolvimento do TCP/IP que a ARPA constituiu um comitê informal para coordenar e orientar o projeto dos protocolos e da arquitetura da emergente internet. Denominado Internet Control and Configuration Board (ICCB), o grupo reuniu-se regularmente até 1983 quando passou por uma reorganização.

A internet global teve seu início mais ou menos em 1980, quando a ARPA passou a adotar os novos protocolos TCP-IP nas máquinas ligadas às suas redes de pesquisa. A ARPANET, já instalada, rapidamente tornou-se o backbone da nova internet, e foi utilizada para muitas das primeiras experiências com TCP/IP. A transição para a tecnologia da internet foi completada em janeiro de 1983, quando o Office of the Secretary of Defense ordenou que todos os computadores conectados a redes de longa distância utilizassem o TCP/IP. Ao mesmo tempo, a Defense Communication Agency (DCA) dividiu a ARPANET em duas redes distintas, uma para futuras pesquisas e outra para comunicação de caráter militar. A parte relacionada à pesquisa conservou o nome ARPANET; a parte militar, maior, tornou-se conhecida como rede militar, MILNET.

Para incentivar os pesquisadores das universidades a adotarem e utilizarem os novos protocolos, a RPA possibilitou uma implementação disponível a baixo custo. Nessa ocasião, a maioria dos departamentos de ciência da computação das universidades executava uma versão do sistema operacional UNIX; esse sistema disponível na Berkley Software Distribution, da Universidade da Califórnia, era conhecido como Berkeley UNIX ou BSD UNIX. Ao financiar a Bolt Beranek and Newman, Inc. (BBN), para implementar seus protocolos TCP/IP para uso com o

UNIX, e ao financiar a universidade da Califórnia, Berkeley, para integrar os protocolos TCP/IP ao BSD UNIX, a ARPA conseguiu atingir até 90% dos departamentos de ciência da computação das universidades. O novo software de protocolo surgiu numa ocasião especialmente importante, porque muitos departamentos acabavam de adquirir o segundo ou o terceiro computador, e os estavam conectando as redes locais. Os departamentos precisavam de protocolos de comunicação e geralmente nenhum outro estava disponível.

O software distribuído pela universidade da Califórnia, Berkeley, tornou-se popular porque oferecia mais do que protocolos TCP/IP básicos. Além de programas padrão, de aplicativos TCP/IP, Berkeley oferecia um conjunto de utilitários para serviços de rede que lembravam os serviços UNIX utilizados em uma máquina individual. A principal vantagem dos utilitários Berkeley reside em sua semelhança com o padrão UNIX. Um usuário experiente da UNIX, por exemplo, pode facilmente aprender a usar o utilitário de cópia de arquivo remoto de Berkeley (rcp), porque ele se comporta exatamente como o utilitário de cópia de arquivo da UNIX, exceto quando permite que os usuários copiem arquivos para máquinas remotas ou a partir delas.

Além de um conjunto de programas utilitários, o Berkeley UNIX apresentou uma nova abstração no sistema operacional conhecido como socket, que facilita aos programas de aplicação acessar protocolos de comunicação. Por se tratar de uma popularização do mecanismo UNIX para I/O, o socket possuiu opções para vários tipos de protocolos de rede além do TCP/IP. Seu projeto foi debatido desde a apresentação, e muitos pesquisadores de sistemas operacionais propuseram alternativas. Entretanto, independente de seus méritos como um todo, a introdução do socket foi importante porque permitiu que, sem muito esforço, os programadores utilizassem os protocolos TCP/IP. Assim, os pesquisadores foram incentivados a experimentar o TCP/IP.

O sucesso alcançado pela tecnologia TCP/IP e a internet entre pesquisadores da ciência da computação, levou outros grupos a adotá-lo. Ao perceber que a comunicação de redes logo seria parte decisiva da pesquisa científica, a National Science Foundation assumiu um papel importante na expansão da internet TCP/IP visando a envolver o maior número possível de cientistas. Em 1985, essa organização iniciou um programa para estabelecer redes de acesso centralizadas em torno de seus seis centros de supercomputadores. Em 1986, ela expandiu o desenvolvimento de redes ao financiar um novo backbone denominado NSFNET, que praticamente alcançava todos os seus centros de supercomputadores e conectou-os a ARPANET. Finalmente, em 1986, a NSF forneceu o capital inicial para muitas redes regionais que agora se conectam às principais instituições voltadas para a pesquisa científica de determinada área. Todas as redes financiadas pela NSF utilizam protocolos TCP/IP, e todas são parte da internet global.

Sete anos após sua criação, a internet cresceu, abrangendo centenas de redes individuais localizadas nos Estados Unidos e na Europa. Conectou aproximadamente 20000 computadores de universidades, órgãos públicos e laboratórios de pesquisa organizacional. O tamanho e a utilização da internet continuou em ascensão muito mais acelerada do que o previsto. No final de 1987, estimou-se que o crescimento alcançara 15 % ao mês. Em torno de 1994, a internet global alcançava mais de 3 milhões de computadores em 61 países.

A utilização de protocolos TCP/IP e o crescimento da internet não se limitaram a projetos financiados pelo governo. Grandes companhias voltadas para o setor de computadores conectaram-se à internet, bem como muitas outras organizações de grande porte como companhias de petróleo, indústria automobilística, empresas de eletrônica, companhias farmacêuticas e portadoras de telecomunicações. As empresas de pequeno porte e médio porte começaram a conectar-se na década de 1990. Além disso muitas outras utilizavam os protocolos TCP/IP em suas interligações em redes corporativas, mesmo tendo optado por não participar da internet global.

Uma expansão acelerada trouxe problemas de escala não previstos no projeto original e motivou os pesquisadores a encontrar técnicas para gerenciar numerosos recursos distribuídos. No projeto original, por exemplo, os nomes e endereços de todos os computadores conectados à internet eram mantidos em um único arquivo que era editado manualmente e, a seguir, distribuído a todos os sites da internet. Em meados da década de 1980, tornou-se óbvio que um banco de dados de origem não seria suficiente. Primeiro, os pedidos para atualização de arquivos rapidamente provocaria excesso do pessoal disponível para processá-los. Segundo, ainda que existisse um arquivo-fonte correto, a capacidade da rede seria insuficiente para permitir a distribuição freqüente para todos os sites ou o acesso on-line a cada site.

Novos protocolos foram desenvolvidos e um sistema de atribuição de nome foi colocado em vigor através da internet global para permitir que qualquer usuário automaticamente determinasse o nome de uma máquina remota. Conhecido como Domain Name System, o mecanismo conta com máquinas denominadas servidores de nome para responder a consultas sobre nomes. Nenhuma máquina contém todo o banco de dados de nomes de domínios. Em vez de uma máquina, os dados são distribuídos por um conjunto de máquinas que utilizam protocolos TCP/IP para se comunicarem entre si quando estiverem respondendo a uma consulta.

9.1.5 O conselho de arquitetura da internet

Já que a série de protocolos de interligação em redes TCP/IP não surgiu de um fornecedor específico, ou de uma sociedade profissional reconhecida, é natural perguntar “quem estabelece as diretrizes técnicas e decide quando os protocolos devem-se tornar um padrão?” A resposta é um grupo conhecido como Internet Architecture Board (IAB). O IAB proporciona a base e a coordenação para muitas pesquisas e desenvolvimentos, como a base de protocolos TCP/IP e orienta a evolução da internet. Ele decide quais protocolos são uma parte necessária da pilha TCP/IP e estabelece políticas oficiais.

Criado em 1983, quando a ARPA reorganizou o Internet Control and Configuration Board, o IAB herdou muitos de seus atributos de um grupo anterior. Seus objetivos iniciais eram estimular o intercâmbio de idéias entre os principais envolvidos em pesquisas relacionadas ao TCP/IP e à internet, e manter os pesquisadores concentrados em objetivos comuns. Ao longo dos seis primeiros anos, o IAB passou de um grupo de pesquisa específico da ARPA para uma organização autônoma. Durante esses anos, cada membro do IAB presidia uma Internet Task Force, com a atribuição de investigar um problema ou conjuntos de questões consideradas importantes. O IAB consistia em aproximadamente dez equipes de

trabalho, e as funções variavam da investigação de como a carga de tráfego de vários aplicativos afeta a internet, até o tratamento de problemas técnicos de curto prazo da internet. O IAB reunia-se várias vezes por ano para receber relatórios de status de cada equipe de trabalho, analisar e revisar diretrizes técnicas, discutir políticas e trocar informações com representantes de instituições como ARPA e NFS, as quais financiavam as operações e pesquisas da internet.

O presidente do IAB ocupava o cargo de Arquiteto da Internet, e era responsável pela sugestão de diretrizes técnicas e coordenação de atividades das várias equipes de trabalho.

Cabia a ele novas equipes de trabalho, por recomendação do IAB, e também representar o IAB perante outros.

Os iniciantes em TCP/IP às vezes se surpreendem ao saber que o IAB não gerenciava um grande orçamento: embora estabelecesse diretrizes, não financiava a maior parte da pesquisa e da tecnologia previstas. Em seu lugar, voluntários executavam grande parte do trabalho. Cada membro do IAB era responsável pelo recrutamento de voluntários para trabalhar em suas equipes, convocar e realizar reuniões de equipes de trabalho e informar o andamento ao IAB. Normalmente, os voluntários originavam-se da comunidade acadêmica ou de organizações comerciais que produziam ou utilizavam TCP/IP. Pesquisadores atuantes participavam das atividades das equipes de trabalho da internet por duas razões. Por um lado, servir em uma equipe de trabalho lhes oferecia oportunidade de adquirir mais conhecimentos sobre novas questões a serem pesquisadas. Por outro lado, novas idéias e soluções de problemas projetados e testados por equipes de trabalho freqüentemente tornavam-se parte da tecnologia TCP/IP da internet. Os componentes percebiam que seu trabalho exercia direta influência positiva na área.

9.1.6 A reorganização do IAB

Em meados de 1989, a tecnologia TCP/IP e a internet haviam ultrapassado o projeto de pesquisa inicial, se transformando em recursos das quais milhares de pessoas dependiam para exercer suas atividades comerciais diárias. Não era mais possível apresentar novas idéias mudando algumas instalações da noite para o dia. Em grande parte, todas as centenas de companhias comerciais que ofereciam produtos TCP/IP determinavam quais produtos iriam interoperar, decidindo quando incorporar mudanças a seu software. Os pesquisadores que planejavam especificações e testavam novas idéias em laboratório já não poderiam esperar uma aceitação imediata e prática das idéias. O irônico é que os pesquisadores que projetaram e monitoraram o desenvolvimento do TCP/IP sentiram-se vencidos pelo sucesso de seu plano inicial. Em resumo, o TCP/IP tornou-se uma tecnologia de produção bem sucedida e o mercado começou a dominar sua evolução.

Para refletir as realidades políticas e comerciais do TCP/IP e da internet, o IAB foi reorganizado em meados de 1989. A presidência mudou, os pesquisadores foram removidos do próprio IAB para um grupo subsidiário e um novo conselho do IAB foi constituído para incluir representantes da comunidade.

A figura 1.1 ilustra a nova organização do IAB e o relacionamento de subgrupos.

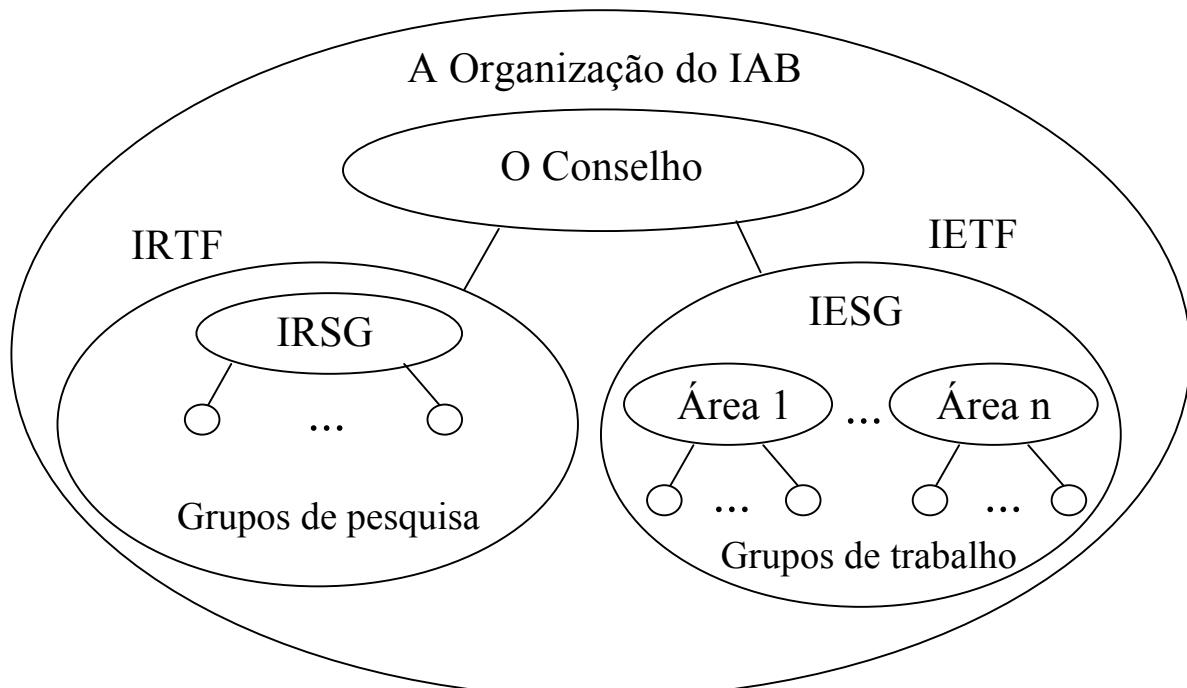


Figura 9.1 A estrutura do IAB após a reorganização de 1989

De acordo com a Figura 9.1 além do próprio conselho, a organização do IAB abriga dois grupos principais: a Internet Research Task Force (IRTF) e a Internet Engeneering Task Force (IETF).

A IETF concentra problemas de engenharia, a curto ou médio prazo. A IETF existia na estrutura original do IAB, e seu sucesso contribuiu, em parte, para motivar a reorganização. Ao contrário da maioria das equipes de trabalho do IAB, que limitava-se a alguns indivíduos que se concentravam em uma tarefa específica, a IETF cresceu para incluir dezenas de membros atuantes que trabalhavam em muitos problemas simultaneamente. Antes da reorganização, a IETF era dividida em mais de 20 grupos de trabalho, cada um concentrando-se em um problema específico. Esses grupos realizavam reuniões isoladas para formular soluções para problemas. Além disso, toda a IETF reunia-se regularmente para analisar os relatórios dos grupos de trabalho e discutir as mudanças propostas ou as inovações à tecnologia TCP/IP. Normalmente, essas reuniões eram realizadas três vezes por ano, todas elas atraindo centenas de participantes e espectadores. A IETF tornou-se muito grande para ser administrada por um presidente.

Como a IETF era conhecida em toda a internet, e porque suas reuniões eram amplamente reconhecidas e assistidas, a estrutura reorganizada do IAB mantém a IETF, mas divide-a em aproximadamente um dezena de áreas, cada uma com seu próprio administrador. O presidente da IETF e os administradores de área formam o Internet Engeneering Steering Group (IESG) responsável pela coordenação da produção dos grupos de trabalho da IETF. O nome “IETF” agora refere-se a toda a estrutura, incluindo o presidente, administradores de área e demais membros dos grupos de trabalho.

Criada durante a reorganização, a Internet Research Task Force é o complemento para pesquisa da IETF. A IRTF coordena as atividades de pesquisa relacionadas aos protocolos TCP/IP ou à arquitetura de interligação em redes em geral. A exemplo da IETF, a IRTF dispõe de um pequeno grupo denominado Internet Research Steering Group, ou IRSG, que estabelece prioridades e coordena as atividades de pesquisa. Ao contrário da IETF, a IRTF é atualmente uma organização muito menor e menos atuante. Cada membro do IRSG administra um internet research group voluntário, análogo aos grupos de trabalho da IETF. A IRTF não é composta por áreas.

9.1.7 A Internet Society

Quando, em 1992, a internet desvinculou-se da tutela governamental nos Estados Unidos, foi constituída uma sociedade, visando estimular a participação naquela rede. Com o nome de The Internet Society, o grupo é uma organização internacional inspirada na National Geographic Society. Na posição de host do IAB, a Internet Society continua a estimular as adesões à internet e sua utilização em todo o mundo.

9.1.8 RFCs (Request for Comments) da Internet

Já mencionamos que nenhum fornecedor detém o direito de propriedade sobre a tecnologia TCP/IP, bem como nenhuma sociedade profissional ou instituições responsáveis pelos padrões. Desse modo, a documentação de protocolos, padrões e políticas não pode ser obtida de um fornecedor. Em vez disso, a National Science Foundation financia um grupo da AT&T para manter e distribuir informações sobre TCP/IP e internet global. Conhecido como Internet Network Information Center (INTERNIC), o grupo trata de muitos detalhes administrativos para a internet, além de distribuir a documentação.

Documentação para o trabalho da internet, as propostas para protocolos novos ou revisados e os padrões de protocolos TCP/IP constam todos de uma série de relatórios técnicos denominados Request for Comments da internet, ou RFCs. Os RFCs podem se curtos ou longos, podem abranger conceitos amplos ou detalhados e podem ser padrões ou simplesmente propostas para novos protocolos. O editor do RFC é um membro do IAB. Embora os RFCs sejam editados, a eles não é dispensada a mesma atenção que aos jornais de pesquisa acadêmica. Além disso, alguns relatórios relativos à internet foram publicados anteriormente, em séries paralelas de relatórios denominados Internet Engineering Notes, ou IENs. Embora a série IEN já não vigore mais, nem todos os IENs aparecem na série RFC. Há referências a RFCs e a uns poucos IENs ao longo do texto.

A série RFC é numerada seqüencialmente, na ordem cronológica em que os RFCs são preparados. A cada RFC novo ou revisado é atribuído um novo número; assim, os leitores precisam estar atentos para obter a versão de documento cujo número seja mais elevado. Está disponível um índice para facilitar a identificação da versão correta.

Para colaborar com o INTERNIC e agilizar a recuperação de documentos, em todo o mundo muitos sites armazenam cópias de RFCs e as tornam disponíveis para a comunidade. É possível obter RFCs via correio postal, correio eletrônico, ou diretamente através da internet, utilizando um programa de transferência de arquivos. Além disso, o INTERNIC e outras organizações colocam à disposição versões preliminares de documentos RFC conhecidos como Internet Drafts. Informe-se com pessoas experientes em rede local sobre como obter RFCs ou internet drafts no seu site, ou consulte o Apêndice 1 para obter instruções sobre sua recuperação.

9.1.9 Protocolos e padronização da Internet

Os leitores familiarizados com redes de comunicação de dados sabem que existem vários padrões de protocolo de comunicação. Muitos deles são anteriores à internet, remetendo à pergunta: “por que os planejadores da internet inventaram novos protocolos, quando tantos padrões internacionais já existiam?” A resposta é complexa, mas segue uma máxima simples:

Utilize os padrões dos protocolos existentes sempre que esses padrões se aplicarem; crie novos protocolos apenas quando os padrões existentes forem insuficientes e estejam aptos para utilizar novos padrões se eles se tornarem disponíveis e proporcionarem uma funcionalidade equivalente.

Assim, embora aparente ser o contrário, a Pilha de Protocolos TCP/IP da internet não foi criada com o objetivo de ignorar ou evitar padrões existentes. Surgiu simplesmente porque nenhum dos protocolos existentes satisfaziam à necessidade de um sistema de comunicação de interligação em redes interoperáveis.

9.1.10 Expansão e tecnologia futuras

A tecnologia TCP/IP e a internet continuam a se desenvolver. Novos protocolos estão sendo propostos; os antigos estão sendo revisados. A NSF acrescentou considerável complexidade ao sistema ao introduzir uma rede backbone, redes regionais e centenas de redes universitárias. Em todo o mundo, outros grupos também continuam a conectar-se à internet. Entretanto, a mudança mais significativa surge não das conexões adicionais de rede, mas de tráfego adicional. À medida que novos usuários conectam-se à internet e surgem novos aplicativos, muda a configuração do tráfego. Quando físicos, químicos e biólogos começam a utilizar a internet, eles trocavam arquivos de dados coletados de seus experimentos. Esses arquivos pareciam grandes se comparados às mensagens de correio eletrônico. Tão logo a internet tornou-se popular e os usuários passaram a buscar informações através de serviços como gopher e a World Wide Web, novamente o tráfego aumentou.

O crescimento da demanda por conexão de redes não deve estar fora de cogitação. Durante vários anos, a indústria computacional experimentou incessante demanda do aumento da capacidade de processamento e de maior armazenamento de dados. Os usuários ainda estão começando a aprender como utilizar redes. No futuro,

podemos esperar aumentos contínuos na demanda de comunicação. As tecnologias de comunicação de maior capacidade serão, portanto, necessárias para a adaptação ao crescimento.

A figura 1.2 resume a expansão da internet e ilustra um importante produto em evolução: a mudança de complexidade surge porque numerosos grupos autônomos gerenciam partes da internet global. Os projetos iniciais para muitos subsistemas dependiam de uma gerência centralizada. É necessário maior esforço para expandir esses projetos a fim de adaptá-los a uma gerência descentralizada.

	Número de redes	Número de computadores	Número de administradores
1980	10	10^2	10^0
1990	10^3	10^5	10^1
1997	10^6	10^8	10^2

Figura 1.2

Expansão da Internet conectada. Além do crescimento em tráfego, resultante do tamanho cada vez maior, a Internet enfrenta uma complexidade que tem sua origem numa gerência descentralizada, não só do desenvolvimento, como das operações

9.1.11 Organização do texto

O material sobre TCP/IP está contido em três volumes. O presente volume apresenta a tecnologia TCP/IP, os aplicativos que utilizam e a arquitetura da internet global em maiores detalhes. Aborda os fundamentos de protocolos como o TCP/IP e o IP, e mostra como eles se ajustam na interligação em redes. Além de fornecer detalhes, o texto destaca os princípios gerais que sustentam os protocolos de rede, explica por que os protocolos TCP/IP adaptam-se facilmente a muitas tecnologias de rede físicas essenciais. O Volume II aborda, em profundidade os detalhes internos dos protocolos TCP/IP, mostrando como eles são implementados. Ele apresenta o código de um sistema de trabalho para ilustrar como os protocolos isolados funcionam juntos e contém detalhes úteis às pessoas cuja responsabilidade é criar interligações em redes empresariais. O Volume III mostra como os aplicativos distribuídos utilizam o TCP/IP na comunicação. Aí está em foco o paradigma cliente/servidor, base de toda a programação distribuída. A interface entre programas e protocolos é discutida, e é mostrada como os programas clientes/servidor são organizados. Além disso, o Volume III descreve o conceito do procedimento remoto e indica como os programadores utilizam as ferramentas necessárias para desenvolver softwares do tipo cliente/servidor.

Até aqui, falamos sobre a tecnologia TCP/IP e a internet, em termos gerais, resumindo os serviços oferecidos e a história de seu desenvolvimento. O próximo capítulo oferece um resumo do tipo de hardware de rede utilizado em toda a internet. Sua finalidade não é realçar nuances do hardware de um fornecedor específico, mas concentra-se naquelas características de cada tecnologia, consideradas fundamentais para uma arquitetura de interligação em redes. Os capítulos posteriores pesquisam os protocolos e a internet, atingindo três objetivos: exploram os conceitos gerais e analisam o modelo arquitetônico da internet, examinam os detalhes dos protocolos TCP/IP e abordam os padrões para serviços de alto nível, como correio eletrônico e

transferência de arquivos. Os capítulos 3 até 12 analisam os princípios fundamentais e descrevem o software de protocolo de rede existente em quaisquer máquinas que utilizem TCP/IP. Os últimos capítulos descrevem os serviços que abrangem múltiplas máquinas, incluindo a difusão de informações de roteamento, resolução de nome e aplicativos como o correio eletrônico.

Dois apêndices seguem-se ao texto principal. O primeiro contém um guia dos RFCs. Ele continua desenvolvendo a descrição dos RFCs encontrados neste capítulo e oferece exemplos de informações que podem ser encontradas em RFCs. Descreve, em detalhes, como obter RFCs através de correio eletrônico, do correio postal e da transferência de arquivos. Finalmente, como o índice do RFC padrão é apresentado em ordem cronológica, o apêndice apresenta uma lista de RFCs, organizada por tópicos, para que os iniciantes localizem facilmente os RFCs relacionados com determinado assunto.

O segundo apêndice contém uma lista, em ordem alfabética, de termos e abreviações utilizadas na literatura e no texto. Já que os iniciantes muitas vezes consideram a nova terminologia confusa e difícil de ser lembrada, eles são estimulados a utilizar a lista em ordem alfabética, em vez de pesquisar ao longo do texto.

como obter RFCs através de correio eletrônico, do correio postal e da transferência de arquivos. Finalmente, como o índice do RFC padrão é apresentado em ordem cronológica, o apêndice apresenta uma lista de RFCs, organizada por tópicos, para que os iniciantes localizem facilmente os RFCs relacionados com determinado assunto.

O segundo apêndice contém uma lista, em ordem alfabética, de termos e abreviações utilizadas na literatura e no texto. Já que os iniciantes muitas vezes consideram a nova terminologia confusa e difícil de ser lembrada, eles são estimulados a utilizar a lista em ordem alfabética, em vez de pesquisar ao longo do texto.

9.1.11.1 Resumo

Uma interligação em redes consiste em um conjunto de redes conectadas que agem como um todo coordenado. A principal vantagem é que ela proporciona uma interconexão universal, ao mesmo tempo em que permite que grupos de indivíduos utilizem qualquer hardware de rede que melhor atenda às suas necessidades. Examinaremos os princípios que orientam a comunicação de interligação em redes em geral e os detalhes de uma pilha de protocolos dessa interconexão, em particular. Também discutiremos como esses protocolos são utilizados em uma interconexão desse tipo. A tecnologia que utilizaremos em nosso exemplo, denominada TCP/IP em virtude de seus dois protocolos principais, foi desenvolvida pela ARPA – Advanced Research Projects Agency. Ela fornece a base para a internet global, uma interligação em redes ampla e operacional que conecta universidades, organizações e departamentos do governo em muitos países em todo o mundo. A internet global está passando por uma rápida expansão.

9.2 Análise das tecnologias básicas de rede

9.2.1 Introdução

É importante compreender que a internet não é um novo modelo de rede física. Ao contrário, é um conjunto de redes físicas ligadas entre si e de convenções sobre o uso de redes que estabelecem a conexão entre computadores. Já que o hardware de rede desempenha um papel apenas secundário no contexto geral, torna-se necessário, para a compreensão da tecnologia da internet, fazer a distinção entre os mecanismos de baixo nível oferecidos pelo hardware e os recursos de alto nível que os protocolos TCP/IP oferecem. É importante compreender também a influência que os recursos fornecidos pela tecnologia de comutação de pacotes exerce na escolha de projetos de alto nível.

O objetivo deste capítulo é introduzir os conceitos e terminologias básicas sobre a comutação de pacotes e analisar algumas tecnologias básicas de hardware de rede que vêm sendo usadas nas ligações entre redes que utilizam o TCP/IP. Os capítulos posteriores descrevem como tais redes são conectadas entre si e como os protocolos TCP/IP conciliam hardwares completamente diferentes. A lista contida neste capítulo certamente não será abrangente, mas servirá para demonstrar claramente a variedade de redes físicas com as quais os TCP/IP operam. Você pode, tranquilamente, ignorar a maior parte dos detalhes técnicos, mas deve tentar compreender a idéia da comutação de pacotes e imaginar o desenvolvimento de um sistema de comunicação homogêneo com hardwares heterogêneos. O mais importante é observar atentamente as particularidades dos mecanismos de endereçamento físico utilizados pelas várias tecnologias; os capítulos posteriores explicarão detalhadamente como os protocolos de alto nível utilizam os endereços físicos.

9.2.2 Duas abordagens sobre a comunicação de rede

Independente do tipo de conexão que façam, seja entre computadores ou entre terminais e computadores, as redes de comunicação dividem-se em dois tipos básicos: de comutação de circuitos (também conhecidas como redes baseadas em conexões) e de comutação de pacotes (conhecidas, ainda, como redes sem conexão). As primeiras operam formando uma conexão dedicada (circuito) entre duas pontas. O sistema telefônico dos Estados Unidos utiliza uma tecnologia de comutação de circuitos – uma chamada telefônica estabelece um circuito da linha de quem telefona, através de uma central de comutação local, passando por linhas do tronco, até uma central de comutação remota e, finalmente, ao destinatário da chamada. Enquanto um circuito estiver aberto, o equipamento telefônico testa o microfone várias vezes, converte os sinais para o formato digital e os transmite através do circuito para o receptor. O transmissor tem a garantia de que os sinais serão distribuídos e reproduzidos, pois o circuito oferece um percurso de dados seguro, de 64 kbps, o mínimo necessário para o envio de voz digitalizada. A vantagem da comutação de circuitos reside na sua capacidade segura: uma vez que um circuito é estabelecido, nenhuma outra atividade

de rede poderá reduzir a capacidade do circuito. A desvantagem da comutação de circuitos é o alto custo: o preço é fixo, independente do tráfego. Por exemplo, o preço de uma ligação telefônica é o mesmo, ainda quando as duas pontas não se comunicam.

As redes de comutação de pacotes, utilizadas normalmente para fazer a conexão entre computadores, fazem uma abordagem completamente diferente. Em uma rede daquele tipo, as mensagens a serem transmitidas através das estações da rede são divididas em pequenas unidades chamadas pacotes que são multiplexados por meio de conexões entre máquinas de alta capacidade. Um pacote que geralmente contém apenas pequenas unidades de informações transporta uma identificação que capacita o hardware da rede a enviar as informações a determinado destino. Por exemplo, a transmissão de um arquivo extenso entre dois equipamentos deve ser feita a partir da divisão do arquivo em vários pacotes antes de encaminhá-los à rede. O hardware da rede envia os pacotes aos seus respectivos destinos onde o software os reúne novamente em um único arquivo. A grande vantagem da comutação de pacotes é a possibilidade de realizar simultaneamente várias comunicações entre computadores, com conexão entre equipamentos compartilhados por todos os pares de equipamentos que estão se comunicando. A desvantagem é que, à medida que a atividade se intensifica, um determinado par de computadores conectados entre si recebe uma capacidade menor da rede. Ou seja, toda vez que uma rede de comutação de pacotes estiver sobrecarregada, os computadores conectados a ela terão que esperar até poderem enviar pacotes adicionais.

Apesar da grande desvantagem de não assegurar a capacidade da rede, as redes de comutação de pacotes tornaram-se muito populares. Os motivos para a adoção desse tipo de rede são o preço e a eficiência. Já que várias podem compartilhar o hardware da rede, o número de conexões diminui e o custo se mantém baixo. O fato de os engenheiros terem conseguido montar um hardware de rede de alta velocidade fez com que a capacidade deixasse de ser um problema. A comutação de pacotes é tão utilizada pelas interconexões de computadores que, durante todo o restante do capítulo, o termo rede vai se referir somente às redes de comutação de pacotes.

9.2.3 Redes de longas distâncias e redes locais

As redes de comutação de pacotes que operam em grandes áreas geográficas são basicamente diferentes daquelas que operam em áreas pequenas (p ex. um único compartimento). Com o intuito de ajudar a caracterizar as diferenças de capacidade e de finalidade, as tecnologias de comutação de pacotes são divididas em duas categorias, de acordo com a extensão: redes de longas distâncias (WANs) e redes locais (LANs). As duas categorias não possuem definições formais. Ao contrário, os fornecedores utilizam os termos de forma coloquial para que os consumidores saibam diferenciar as duas tecnologias.

As tecnologias de rede remota ou de redes de longas distâncias possibilitam a comunicação entre grandes distâncias. A maioria das tecnologias de rede de longas distâncias não impõe um limite na extensão da distância; permite que os dois extremos se comuniquem a uma distância arbitrária. Uma rede de longas distâncias, por exemplo, pode operar em um continente ou conectar computadores de continentes diferentes. Geralmente, as redes de longas distâncias operam em velocidades mais lentas do que as redes locais, e necessitam de um retardo de transmissão bem maior

entre as conexões. A velocidade de uma rede de longas distâncias varia de 56 kbps a 155 Mbps. O retardo de transmissão em uma rede de longas distâncias pode variar desde alguns milissegundos até vários décimos de segundo.

As tecnologias de rede local possuem uma velocidade de conexão entre computadores bem mais rápida, mas deixam a desejar na capacidade de operar em longas distâncias. Por exemplo, uma rede local típica abrange uma área pequena, como um único edifício ou um campus, e funciona entre 10 Mbps e 2 Gbps. Já que essas tecnologias operam em pequenas áreas, o retardo de transmissão é bem menor do que o das tecnologias de rede de longas distâncias, o qual pode durar desde alguns décimos de um milissegundo, até no máximo dez milissegundos.

Já mencionamos a escolha entre velocidade e distância: tecnologias que oferecem uma velocidade maior de comunicação abrangem pequenas áreas geográficas. Existem também outras diferenças entre as categorias de tecnologias. Na tecnologia de rede local, cada computador geralmente contém um dispositivo de interface de rede que conecta a máquina diretamente ao ambiente de rede. Normalmente, a rede em si é passiva, dependendo dos dispositivos eletrônicos dos computadores ligados a ela para gerar e receber os sinais elétricos necessários. Na tecnologia de rede de longas distâncias, uma rede geralmente consiste em uma série de computadores denominados comutadores de pacotes, interligados por linhas e modems de comunicação. É possível aumentar o tamanho da rede com a adição de um novo comutador e de uma nova linha de comunicações. Ligar o computador de um usuário a uma rede de longas distâncias significa conectá-lo a uma dos comutadores de pacotes. Cada comutador, ao longo de um percurso na rede de longas distâncias, inicia um retardo de transmissão ao receber um pacote e o transfere ao comutador seguinte. Desse modo, quanto maior a rede de longas distâncias, mais tempo ela levará para desobstruir o tráfego.

Este livro trata do software que concentra as diferenças tecnológicas entre as redes e que faz a interconexão independente do hardware básico. Para saber optar entre os diferentes modelos de software, é necessário compreender como eles se relacionam com o hardware da rede. As seções contêm exemplos de tecnologias de rede que foram usadas na internet, mostrando algumas das diferenças existentes entre elas. Os capítulos posteriores explicam como o software TCP/IP identifica tais diferenças, tornando o sistema de comunicação independente da tecnologia básica do hardware.

9.2.3.1 Endereços de hardware da rede

Cada tecnologia do hardware da rede define um mecanismo de endereçamento utilizado pelos computadores para definir o destino de cada pacote. A todos os computadores conectados a uma rede é atribuído um único endereço, geralmente um número inteiro. Um pacote enviado através de uma rede possui um campo de endereço de destino que contém o endereço do destinatário. O endereço de destino aparece na mesma localização em todos os pacotes, possibilitando ao hardware da rede identificar o endereço de destino facilmente. Um transmissor deve saber o endereço do destinatário e o deve inserir no campo de endereço de destino de um pacote antes de encaminhá-lo.

Todas as tecnologias do hardware especificam como os endereços são atribuídos aos computadores. O hardware especifica, por exemplo o número de bits do

endereço, assim como a localização do campo do endereço de destino de um pacote. Apesar de algumas tecnologias utilizarem esquemas de endereçamento compatíveis, a maioria não o faz. Este capítulo contém alguns exemplos de esquemas de endereçamento de hardware; os capítulos posteriores explicam como os TCP/IP conciliam esquemas de endereçamento de hardware diferentes.

9.2.4 A tecnologia Ethernet

Ethernet é o nome dado a uma tecnologia de rede local popular, de comutação de pacotes, criada pela Xerox PARC no início da década de 1970. As empresas Xerox, Intel e Digital Equipment padronizaram a Ethernet em 1978; o IEEE criou uma versão compatível do padrão, utilizando o número 802.3. A Ethernet tornou-se uma tecnologia de rede local popular; a maioria das empresas de médio e grande porte a utiliza. Este pioneirismo gerou os seus frutos; a grande maioria dos fabricantes de hardware usa a topologia Ethernet para LAN, além de ser considerada a melhor topologia em confiabilidade e produtividade operacional para LAN. Já que se tornou tão popular, a Ethernet possui muitas variantes; falaremos primeiro sobre o modelo original, e depois sobre as variantes.

Segundo o modelo ISO/OSI, o Ethernet é um padrão que define os níveis 1 e 2 especificados respectivamente pelas normas 802.3 e 802.2 da IEEE.

Com relação ao nível físico, o Ethernet é uma rede de topologia de barramento, com CSMA/CD “(carrier sense multiple access with collision detection)”: cada estação que quiser acessar a linha verifica sua ocupação; se estiver livre, transmite; se não, espera o fim da transmissão atual para transmitir. Se duas estações iniciarem a transmissão ao mesmo tempo, a colisão é detectada por ambas estações, que abortam a transmissão, gerando uma retransmissão. Esta retransmissão segue algoritmo que visa reduzir a chance de nova colisão.

O Protocolo de Controle de Acesso aos Meios Físicos

O principal trecho da especificação de camada de link de dados para redes Ethernet descreve a forma como as estações deverão compartilhar o acesso a cabos coaxiais através de um processo denominado CSMA/CD (Carrier Sense Multiple Access with Collision Detection). O CSMA/CD é um protocolo de controle de acesso aos meios físicos (MAC) que determina a forma como os nós da rede compartilham o acesso ao cabo. O meio físico é o cabo coaxial que conecta os nós da rede, e o protocolo de controle de acesso é o esquema de compartilhamento. Para que pacotes de informação possam percorrer o cabo da rede Ethernet, eles devem lidar com o CSMA/CD.

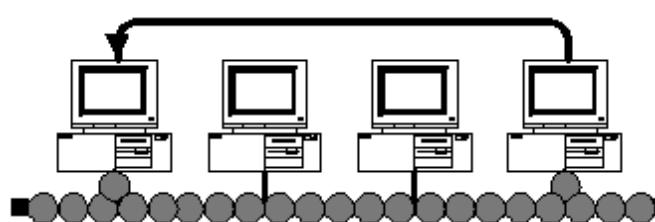
O CSMA/CD verifica a rede antes de transmitir. Se receber dados de um software de alto nível, antes de enviá-los, a placa adaptadora de rede verifica se outra estação está transmitindo no cabo. A placa só transmitirá a mensagem quando o cabo estiver livre.

O CSMA/CD também funciona como mediador quando o inevitável acontece: dois ou mais nós começam a transmitir simultaneamente em um cabo livre, e as transmissões colidem. As placas adaptadoras podem detectar essas colisões por causa do nível de sinal elétrico mais alto que as transmissões simultâneas produzem. Quando detectam uma colisão, as placas adaptadoras de rede começam a transmitir o

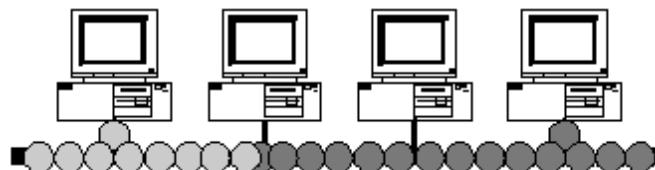
que denominamos "sinal de congestionamento", garantindo que todos os nós conflitantes percebam a colisão. Em seguida, todas as placas param de transmitir e acessam suas programações internas, a fim de determinar um tempo aleatoriamente selecionado para retransmissão. Esse período de "retração" assegura que as estações não continuem a enviar sinais que possam entrar em colisão quando o tráfego no cabo diminuir.



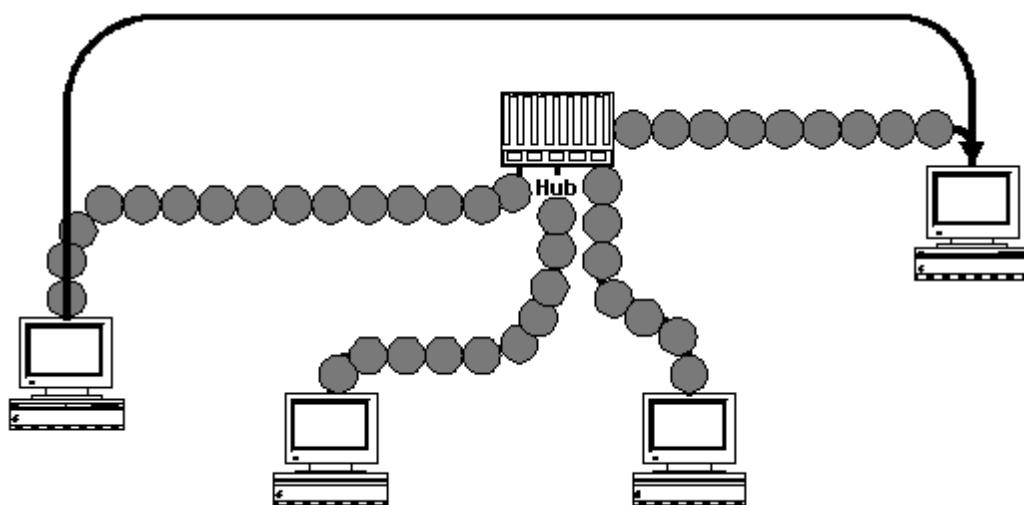
O sinal em um cabo coaxial desbalanceado é anulado devido à própria reflexão do mesmo no final do circuito.



Com a carga nas extremidades o sinal é distribuído pelo barramento, sendo que apenas o micro de destino receberá a informação.



Se um ou mais requisitarem o barramento, haverá uma colisão entre os sinais e o primeiro a solicitar ficará com o barramento



A informação está disponível em todos os micros no padrão ethernet.

Cada cabo da Ethernet possui aproximadamente 0,5 polegadas de diâmetro e até 500 metros de comprimento. Um resistor é anexado entre o fio central e a proteção, em cada uma das extremidades, para evitar o reflexo de sinais elétricos.

O modelo original da Ethernet possuía um cabo coaxial, como o ilustrado na figura 2.1. Também chamado de ether, o cabo em si é completamente passivo; todos os componentes eletrônicos ativos que fazem a rede funcionar são associados aos computadores conectados à rede.



Figura 2.1 Um corte transversal no cabo coaxial usado na Ethernet original.

Para se fazer a conexão entre um computador e um cabo coaxial Ethernet é necessário um dispositivo de hardware chamado transceptor. Fisicamente, a conexão entre um transceptor e a Ethernet necessita de um pequeno orifício nas camadas exteriores do cabo, como ilustra a Figura 2.2. Em tese, o termo usado para descrever a conexão entre um transceptor de Ethernet e um cabo é derivação (TAP). Geralmente, pequenos pinos de metal montados no transceptor passam pelo orifício e oferecem contatos elétricos ao fio central e à proteção metálica. Alguns conectores fabricados necessitam que o cabo seja cortado, e um “T”, inserido.

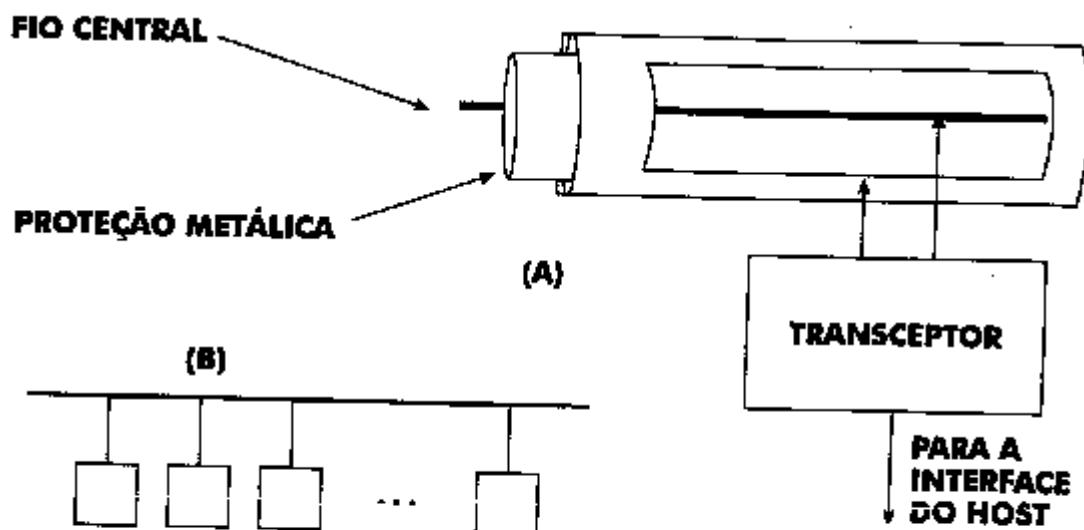


Figura 2.2 (a) Um corte transversal em um cabo Ethernet, mostrando os detalhes dos componentes elétricos entre o transceptor e o cabo, e (b) o diagrama esquemático de uma Ethernet com vários computadores conectados

Cada conexão a uma Ethernet necessita de dois componentes eletrônicos principais. Um transceptor que conecta-se ao fio central e à proteção metálica do

cabo, captando e enviando sinais no ether. Uma interface do host ou um adaptador de host encaixa-se no barramento do computador (p. ex. na placa mãe) e conecta-se ao transceptor.

Um transceptor é um pequeno componente do hardware, geralmente localizado fisicamente ao lado do ether. Além do hardware analógico que recebe e controla os sinais elétricos no ether, um transceptor contém um circuito digital que possibilita sua comunicação com um computador digital. O transceptor pode detectar quando o ether está em uso e converter os sinais elétricos analógicos do ether para a forma digital (e a partir dela). Um cabo chamado AUI (Attachment Unit Interface) conecta o transceptor a uma placa adaptadora de um computador host. Chamado informalmente de cabo transceptor, o cabo AUI possui muitos fios. Tais fios transportam a energia elétrica necessária para a operação do transceptor, os sinais que controlam a operação do transceptor e o conteúdo dos pacotes que estão sendo enviados ou recebidos. A figura 2.3 ilustra como os componentes formam a conexão entre um barramento num sistema de computador e um cabo Ethernet.

Cada uma das interfaces do host controla a operação de um transceptor, de acordo com as instruções que recebe do software do computador. Para o software do sistema operacional, a interface parece ser um dispositivo de entrada e saída que recebe instruções básicas de transferência de informações a partir do computador, controla o transceptor para executá-las, interrompe quando a operação chega ao fim e reporta as informações de status. Apesar de o transceptor ser um simples dispositivo do hardware, a interface do host pode ser complexa (p. ex. pode conter um microprocessador utilizado para controlar transferências entre a memória do computador e o ether).

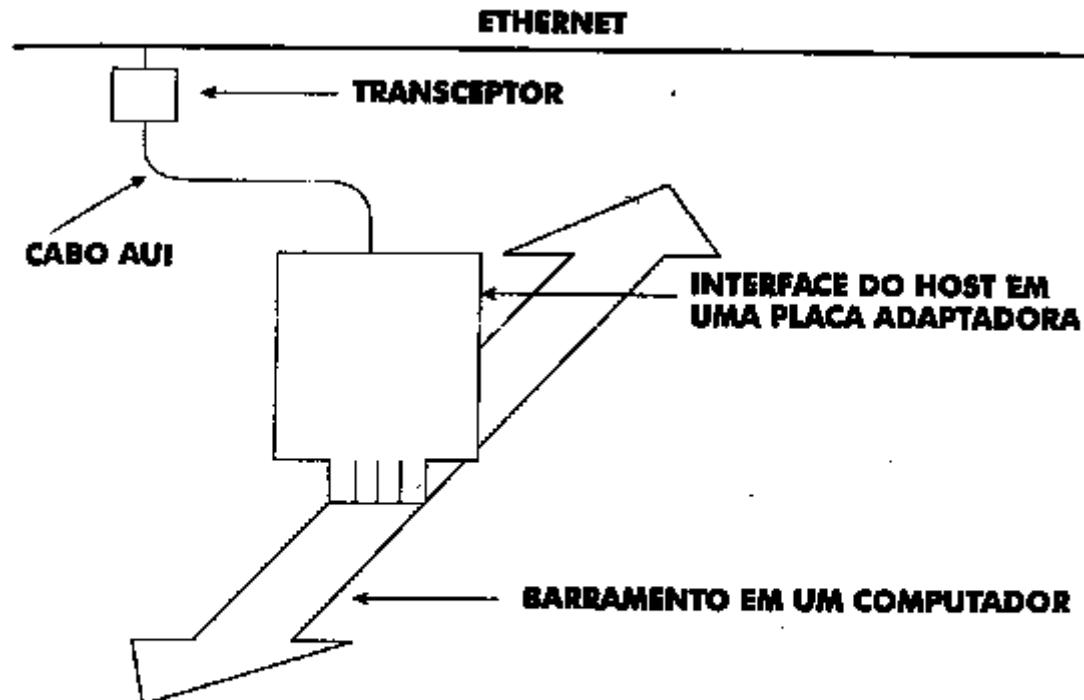


Figura 2.3 Os dois principais componentes eletrônicos que formam uma conexão entre um barramento de computador e uma Ethernet. O cabo AUI que conecta a interface do host ao transceptor transporta a energia elétrica e os sinais para controlar a operação do transceptor, assim como os pacotes que estão sendo transmitidos ou recebidos.

Na prática, as instruções que utilizam a Ethernet original em um ambiente de trabalho convencional instalam o cabo da Ethernet ao longo do teto de cada sala e conectam todas as salas ao cabo. A Figura 2.4 ilustra o esquema de fiação física resultante.

9.2.4.1 Ethernet de fio fino

Vários componentes da tecnologia original de Ethernet possuem propriedades indesejáveis. Por exemplo, em virtude de o transceptor possuir componentes eletrônicos, seu custo não é baixo. Além disso, já que os transceptor são localizados no cabo, e não nos computadores, o acesso a eles torna-se difícil, bem como sua substituição. O cabo coaxial que forma o ether também pode ser de difícil instalação. Para oferecer o máximo de proteção contra interferência elétrica de dispositivos como motores elétricos, o cabo possui uma forte proteção que o torna difícil de ser dobrado. Um cabo AUI também é forte e difícil de ser dobrado.

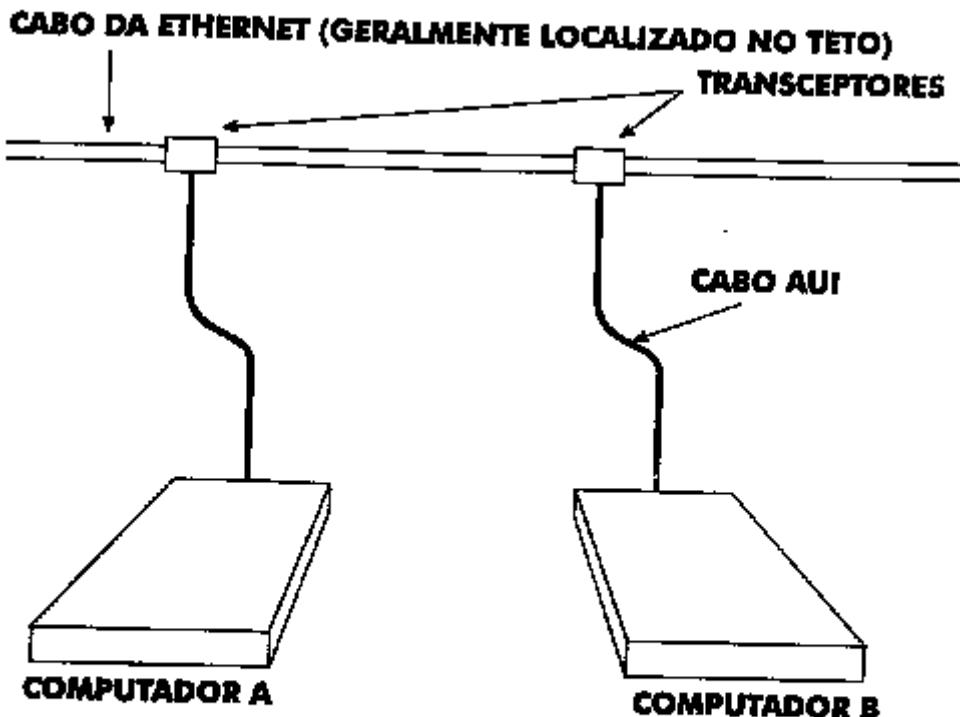


Figura 2.4 A conexão fiaça entre dois computadores a uma Ethernet com o esquema original de fiação. Em um ambiente de trabalho, o cabo da Ethernet é normalmente localizado junto ao teto do corredor; cada sala possui um cabo AUI que conecta um computador do escritório a um transceptor ligado ao cabo da Ethernet.

Para reduzir custos em ambientes como escritórios, que não possuem muita interferência elétrica, os engenheiros criaram um esquema de fiação alternativo para a Ethernet. Conhecido como Ethernet de fio fino (Thin-wire ethernet ou thinnet), o cabo coaxial alternativo é mais fino, mais barato e mais flexível. Entretanto, uma Ethernet de fio fino possui algumas desvantagens. Já que não oferece muita proteção contra interferência elétrica, ela não pode ficar próxima a equipamentos elétricos muito potentes, como os de uma fábrica. Além disso, a Ethernet de fio fino opera em distâncias um pouco menores e comporta uma quantidade de conexões menor, pra rede, do que a ethernet de fio grosso.

Para reduzir ainda mais o custo, os engenheiros substituíram o transceptor de alto custo com circuitos especiais de alta velocidade e fizeram uma conexão direta entre um computador e o ether. Assim, em um esquema de fio fino, um computador possui tanto a interface do host como o circuito que é conectado ao cabo. Os fabricantes de microcomputadores e de estações de trabalho consideram a Ethernet de fio fino um esquema particularmente atrativo, em virtude da possibilidade da conexão entre o hardware da Ethernet e computadores de placa única, e de instalar diretamente os conectores na parte traseira do computador.

Como o Ethernet de fio fino conecta diretamente um computador a outro, o esquema de fiação funciona perfeitamente quando vários computadores ocupam um mesmo compartimento. O cabo de fio fino percorre o trajeto diretamente de um computador a outro. Para incluir um outro computador, basta conectá-lo à cadeia. A Figura 2.5 ilustra as conexões utilizadas pela Ethernet de fio fino.

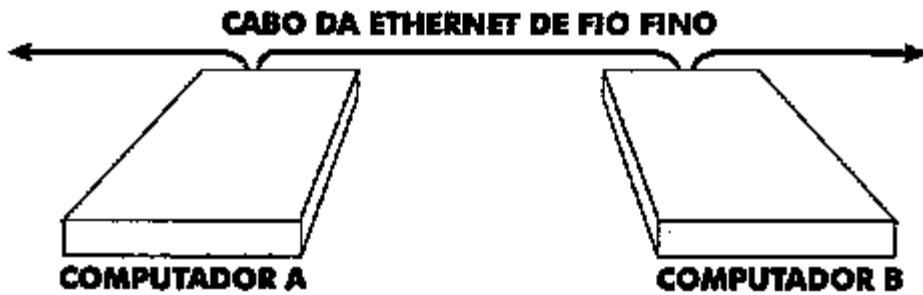


Figura 2.5 Conexão física entre dois computadores com o esquema de fiação thinnet. O ether percorre diretamente o trajeto de um computador a outro; não é necessário nenhum hardware de transceptor externo

As redes Ethernet de fio fino são projetadas para serem facilmente conectadas e desconectadas. Essa tecnologia utiliza conectores BNC, os quais não necessitam de ferramentas para conectar um computador ao cabo. Assim, é possível que o próprio usuário conecte um computador a uma rede Ethernet de fio fino sem a ajuda de um técnico. É claro que o fato de os usuários poderem manipular o ether sozinhos traz algumas desvantagens; se um usuário desconectar o ether, impedirá a comunicação de todas as outras máquinas no ether. Muitas vezes, entretanto, as vantagens são bem maiores do que as desvantagens.

9.2.4.2 Ethernet de pares trançados

Graças ao avanço da tecnologia foi possível construir redes Ethernet que não necessitam da proteção elétrica de um cabo coaxial. Chamada de Ethernet de pares trançados, essa tecnologia permite que um computador acesse uma rede Ethernet utilizando um par de fios de cobre normais sem proteção, semelhantes aos utilizados para fazer a conexão entre equipamentos telefônicos. A vantagem desse tipo de tecnologia é que, além de reduzir os custos, oferece proteção a outros computadores da rede no caso de um usuário desconectar um único computador. Em alguns casos, uma tecnologia de pares trançados possibilita que uma instituição utilize a Ethernet com a fiação telefônica já existente, sem a adição de novos cabos. Conhecido tecnicamente como 10Base-T, o esquema de fiação de pares trançados conecta cada computador a um HUB da Ethernet, como ilustra a Figura 2.6.

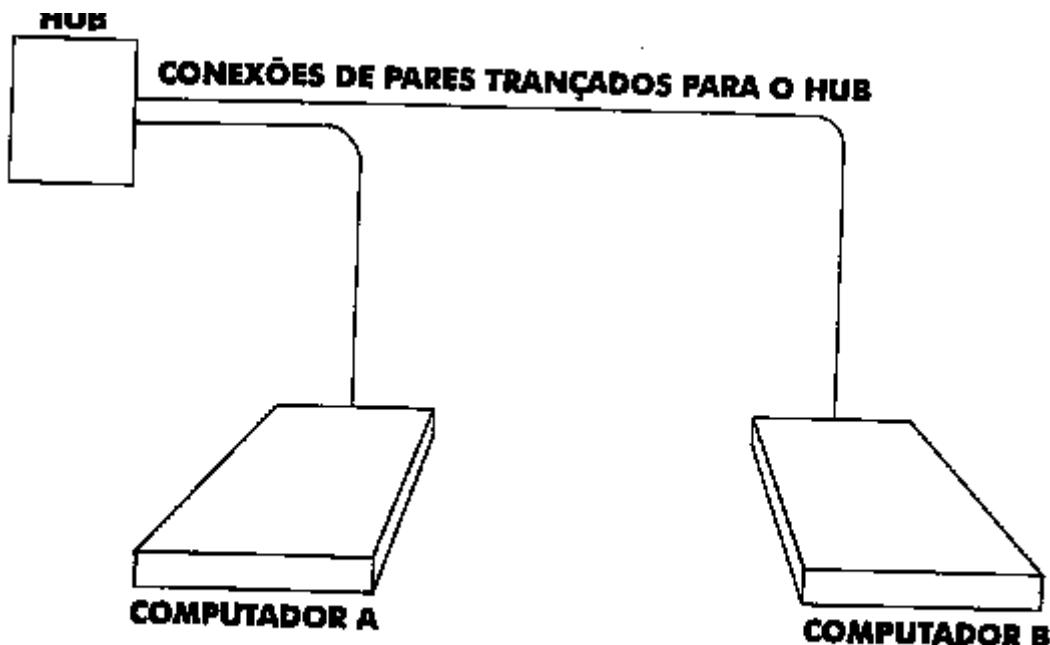


Figura 2.6 Ilustração de uma fiação de pares trançados utilizada pela Ethernet. Cada computador conecta-se a um HUB por cima de um par de fios

O HUB é um dispositivo eletrônico que estimula os sinais num cabo Ethernet. Fisicamente, o HUB é formado por uma pequena caixa que geralmente é alojada em um gabinete de fiação; uma conexão entre um HUB e um computador deve ter menos de cem metros de extensão. Um HUB necessita de energia elétrica e, talvez, de pessoal qualificado para fazer o monitoramento e o controle de sua operação na rede. Para a interface com um computador, a conexão a um HUB parece funcionar do mesmo modo que a conexão a um transceptor. Ou seja, um HUB da Ethernet possui a mesma capacidade de comunicação que uma Ethernet de fio fino ou de fio grosso; os conectores apenas oferecem um esquema de fiação alternativo.

9.2.4.3 Esquemas de fiação múltiplos e com adaptadores

Para se fazer uma conexão à Ethernet de fio grosso é necessário um conector AUI; já para uma conexão à Ethernet de fio fino é necessário um conector RJ45, similar aos conectores modulares utilizados pelos aparelhos telefônicos. A maioria dos produtos da Ethernet permite que o cliente escolha o esquema de fiação que deseja. As placas adaptadoras para os microcomputadores, por exemplo, geralmente vêm acompanhadas de três conectores, como ilustra a Figura 2.7. Apesar de só um conector poder ser utilizado de cada vez, um computador que possua tal adaptador pode facilmente ser transferido de um esquema de fiação a outro.

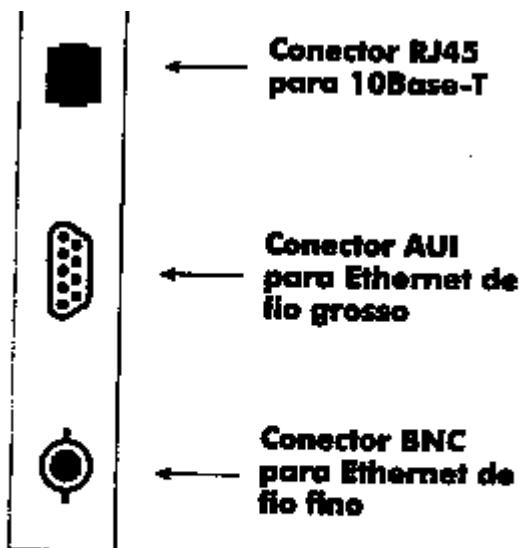


Figura 2.7 Um típico cartão adaptador da Ethernet com três conectores para os três esquemas de fiação da Ethernet. Apesar de o adaptador possuir três conectores, só um esquema de fiação pode ser utilizado por vez

9.2.4.4 Características de uma rede Ethernet

A Ethernet é uma tecnologia de barramento de difusão de 10 Mbps, 100 Mbits e até 1 Gbits, com método de entrega sem garantia e controle de acesso distribuído. É um barramento porque todas as estações compartilham um único canal de comunicação; é de difusão (broadcast) porque todos os transceptores recebem cada uma das transmissões. O método adotado, de encaminhar pacotes somente de uma estação a outra ou a um subconjunto de todas as estações, será discutido mais adiante. Por enquanto, é suficiente compreender que os transceptores não diferenciam as transmissões – um transceptor transporta todos os pacotes do cabo para a interface do host que, por sua vez, seleciona os pacotes que o computador deve receber e retém os demais. A Ethernet é também chamada de mecanismos Best-Effort porque o hardware não fornece qualquer informação ao transmissor sobre a entrega do pacote. Por exemplo, se a máquina destinatária estiver desligada, os pacotes que estavam destinados a ela se perderão e o transmissor não será notificado. Posteriormente, veremos como os protocolos TCP/IP lidam com hardware para transmissão Best-Effort.

O controle de acesso à Ethernet é distribuído porque, ao contrário de algumas tecnologias de rede, a Ethernet não possui nenhuma autoridade central para permitir o acesso. O esquema de acesso à Ethernet é chamado CSMA/CD (Carrier Sense Multiple Access with Collision Detect). É um CSMA porque vários equipamentos podem acessar a Ethernet simultaneamente e cada um deles estabelece se o meio está ou não livre, detectando a presença ou não de sinal. Quando uma interface do host tem um pacote para transmitir, ela verifica o meio para saber se há alguma mensagem sendo transmitida (ou seja, detecta o transceptor). Se nenhuma transmissão for detectada, a interface do host inicializa a transmissão. Cada uma das transmissões possui um limite de duração (porque há um tamanho máximo de pacote). Além disso, o hardware deve observar um intervalo mínimo de tempo entre as transmissões, o que significa que nenhum par de equipamentos comunicantes pode utilizar a rede sem oferecer aos demais equipamentos uma oportunidade de acesso.

9.2.4.5 Detecção de colisão e recuperação

Quando um transceptor inicia uma transmissão, o sinal não consegue alcançar todos os locais da rede, simultaneamente. Ao contrário, ele percorre o cabo a aproximadamente 80% da velocidade da luz. Desse modo, é possível que dois transceptores detectem que a rede está disponível e inicializem uma transmissão ao mesmo tempo. Quando os dois sinais elétricos se sobrepõem, ambos tornam-se inválidos. Tais incidentes são chamados colisões.

A Ethernet lida com as colisões de uma forma simples. Cada transceptor monitora o cabo, enquanto ele está transmitindo e verifica se algum sinal incomum interfere na transmissão. Em tese, o monitoramento recebe o nome de CD (collision detect), tornando a ethernet uma rede de CSMA/CD. Quando uma colisão é detectada, a interface do host interrompe a transmissão, espera que a atividade cesse e tenta inicializá-la novamente. É preciso ter muito cuidado, caso contrário, a rede pode ser desativada com todos os transceptores que estejam tentando transmitir e as transmissões podem resultar em colisões.

Para evitar tais situações, a Ethernet utiliza uma política chamada binary exponencial backoff, na qual um transmissor espera um tempo aleatório após a primeira colisão. A duração máxima deste tempo aleatório é duas vezes a da primeira tentativa, caso a segunda tentativa também resulte em colisões; quatro vezes caso a terceira tentativa também resulte em colisão, e assim por diante. O motivo para o uso da política binary exponencial backoff é que, em uma situação improvável na qual muitas estações tentam transmitir simultaneamente, pode ocorrer um grande congestionamento. Em tal situação, há grande probabilidade de duas estações escolherem intervalos aleatórios que estejam próximos uns dos outros. Por isso, a probabilidade de uma outra colisão ocorre é muito grande. Dobrando-se o tamanho máximo do retardo aleatório de transmissão, a estratégia binary exponencial backoff dispersa rapidamente as tentativas de retransmissão das estações em um período de tempo razoavelmente longo, diminuindo, assim, a probabilidade de ocorrência de futuras colisões.

9.2.4.6 A capacidade da Ethernet

A ethernet padrão possui 10 Mbps, o que significa que as informações podem ser transmitidas no cabo a uma velocidade de 10 milhões de bits por segundo. Apesar de o computador poder gerar informações na velocidade da Ethernet, não se deve considerar a velocidade da rede, por si só, como a taxa na qual dois computadores fazem o intercâmbio de informações. Ao contrário, a velocidade da rede deve ser considerada como uma medida da capacidade total do tráfego da rede. Compare uma rede a uma auto-estrada que conecta várias cidades, e os pacotes aos carros dessa auto-estrada. Largas bandas passantes possibilitam a condução de carga (tráfego) pesada, ao passo que bandas passantes estreitas significam que a auto-estrada não pode conduzir a mesma quantidade de tráfego. Uma Ethernet de 10 Mbps, por exemplo, pode controlar poucos computadores que geram carga pesada, ou muitos gerando carga leve. Atualmente, as redes Ethernet tem atingido maiores velocidades, sendo possível o tráfego de informações a 100 Mbits e a 1 Gbits.

9.2.4.7 Endereços físicos da Ethernet

A Ethernet possui um sistema de endereçamento de 48 bits. A cada computador conectado a uma Ethernet é projetado um único número de 48 bits conhecido como o endereço da Ethernet. Para projetar um endereço, os fabricantes do hardware da Ethernet compram blocos de endereços de Ethernet e os designam em seqüência, à medida que fabricam o hardware da interface da Ethernet. Assim, duas interfaces de hardware não possuem, teoricamente, o mesmo endereço físico da Ethernet.

Normalmente, o endereço da Ethernet pode ser lido pela máquina na interface de hardware do host. Já que os endereços de Ethernet pertencem aos dispositivos do hardware, às vezes são chamados endereços do hardware ou endereços físicos. Observe a seguinte característica dos endereços físicos da Ethernet:

Os endereços físicos são associados ao hardware da interface da Ethernet; transferir a interface do hardware para uma nova máquina ou substituir uma interface do hardware que apresentou problemas irá alterar o endereço físico da máquina.

O fato de saber que os endereços físicos da Ethernet podem ser alterados esclarece e porquê de níveis mais elevados de software de rede serem projetados para acomodar tais alterações.

A interface de hardware do host analisa os pacotes e decide quais serão repassados do host. Vale lembrar que cada interface recebe uma cópia pacote – mesmo os enviado a outros equipamentos. A interface do host utiliza como um filtro o campo do endereço de destino de um pacote. A interface desconsidera os pacotes endereçados aos outros equipamentos e deixa passar para o host somente os pacotes endereçados a ele. O mecanismo de endereçamento e o filtro do hardware são necessários para evitar que um computador fique sobrecarregado com informações desnecessárias. Apesar de a CPU da máquina (estaçao de trabalho) poder fazer a verificação, quando ela é realizada pela interface do host o tráfego na Ethernet não diminui a velocidade de processamento de todos os outros computadores.

Um endereço da Ethernet, de 48 bits, pode ter mais utilidade do que simplesmente especificar um único computador de destino. Um endereço pode ser de três tipos:

- O endereço físico de uma interface de rede (um endereço unicast)
- O endereço de difusão (broadcast) da rede
- Um endereço multicast

Por convenção, o endereço de difusão (broadcast) (todos 1s) é reservado para o envio simultâneo a todas as estações. Os endereços de multicast provêm uma forma limitada de difusão, na qual um subconjunto de computadores de uma rede aceita monitorar determinado endereço de multicast. O conjunto dos computadores integrantes é conhecido como grupo de multicast. Para unir um grupo de multicast, um computador deve programar sua interface do host para aceitar o endereço de multicast do grupo. A vantagem do multicast está na possibilidade de limitar a difusão: cada computador de um grupo de multicast pode ser acessado com uma simples transmissão de pacotes, mas os computadores que optam por não participar de determinado grupo recebem os pacotes enviados ao grupo.

Para conciliar transmissão e endereçamento de multicast, o hardware da interface da Ethernet deve reconhecer mais do que seu endereço físico. Uma interface do host geralmente recebe pelo menos dois tipos de pacotes: os endereçados ao endereço físico da interface (ou seja, unicast) e os endereçados aos endereços de transmissão da rede. Algumas interfaces podem ser programadas para identificar endereços de multicast ou até mesmo endereços físicos alternados. Quando o sistema operacional é inicializado, ele inicializa também a interface da Ethernet, oferecendo um conjunto de endereços a serem reconhecidos. A interface, então, analisa o campo de endereço de destino em cada pacote, deixando passar para a máquina (host) somente as transmissões designadas a um dos endereços especificados.

9.2.4.8 O formato do quadro da Ethernet

A ethernet deve ser considerada uma conexão no nível de enlace de dados entre máquinas. Desse modo, faz sentido observar as informações transmitidas como um quadro. Os quadros da Ethernet possuem vários comprimentos, sendo que não há nenhum inferior a 64 octetos ou superior a 1518 octetos (cabecalho, informações e CRC). Como em todas as redes de comutação de pacotes, cada quadro da Ethernet contém um campo com o endereço de seu destinatário. A Figura 2.8 ilustra o formato do quadro da Ethernet que possui o endereço físico de origem, assim como o endereço de destino.

Preâmbulo	Endereço destino	Endereço origem	Tipo de quadro	Dados do Usuário	CRC
8 octetos	6 octetos	6 octetos	2 octetos	64 - 1.500 octetos	4 octetos

Fig. 2.8 O formato de um quadro (pacote) à medida que percorre a Ethernet precedido por um preâmbulo. Os campos não estão desenhados em escala.

Além de identificar a origem e o destino, cada quadro transmitido através da Ethernet contém um preâmbulo, um campo de tipo, um campo de informações, e um CRC. O preâmbulo consiste em 64 bits de 0s e 1s alternados com o propósito de ajudar no recebimento de nós sincronizados. O CRC de 32 bits ajuda a interface na detecção dos erros de transmissão: o transmissor considera o CRC como uma função da informação no quadro, e o receptor reconsidera o CRC para verificar se o pacote foi ou não recebido intacto.

O campo de tipo do quadro contém um número inteiro de 16 bits que identifica o tipo de informação que está sendo transportada no quadro. Para a Internet, o campo de tipo do quadro é fundamental porque demonstra que os quadros da Ethernet são auto-identificáveis. Quando um quadro chega a determinado equipamento, o sistema operacional utiliza o tipo do quadro para determinar qual módulo do software do protocolo deve processar o quadro. As maiores vantagens dos quadros auto-identificáveis estão na possibilidade de utilizar vários protocolos em conjunto em um só equipamento e permitem que vários protocolos sejam mesclados na mesma rede.

física, sem interferência. Uma pessoa pode ter um programa de aplicação, por exemplo, utilizando os protocolos da Internet, ao passo que outra utilizou um protocolo local experimental. O sistema operacional utiliza o campo de tipo de um quadro que chega a fim de decidir como processar o conteúdo. Veremos mais adiante que os protocolos TCP/IP utilizam quadros de Ethernet auto-identificáveis para fazer a distinção entre os vários protocolos.

9.2.4.9 Como aumentar o comprimento da Ethernet com repetidores

Apesar de um cabo da Ethernet possuir um comprimento máximo, a rede pode ser aumentada de duas maneiras: com o auxílio de repetidores e de pontes. Um dispositivo de hardware chamado repetidor pode ser utilizado para transmitir sinais elétricos de um cabo a outro. Entretanto, no máximo dois repetidores podem ser colocados entre duas máquinas, de modo que o comprimento total de uma única Ethernet continua muito pequeno (três segmentos de 500 metros cada). A Figura 2.9 ilustra um uso típico de repetidores no edifício de uma empresa. Um único cabo percorre verticalmente o edifício, e um repetidor une o backbone a um cabo adicional em cada um dos andares. Os computadores conectam-se aos cabos em cada andar.

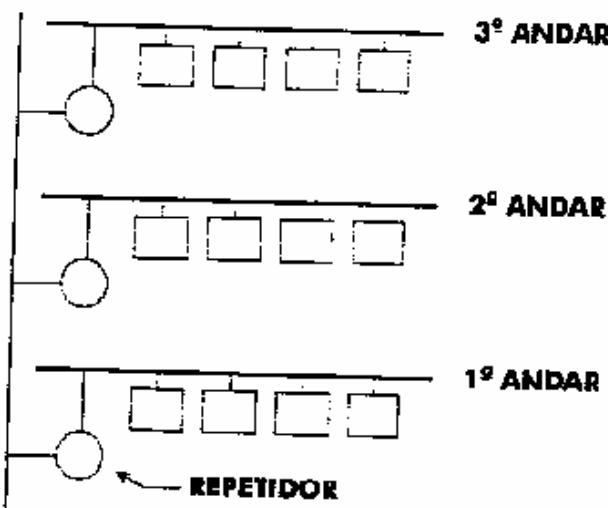


Fig 2.9 Repetidores utilizados para unir os cabos da Ethernet num edifício. No máximo dois repetidores podem ser posicionados entre um par de comunicantes entre si.

9.2.4.10 Como Aumentar o Comprimento da Ethernet com pontes

As pontes são melhores do que os repetidores porque não repercutem os ruídos, as falhas ou os quadros com má formação; um quadro inteiramente válido deve ser recebido antes que a ponte o aceite e o transmita para o outro segmento. Além disso, as interfaces das pontes estão de acordo com as regras do CSMA/CD da Ethernet, de tal modo que as colisões e as demoras na difusão em um único fio permanecem isoladas do outro fio. Desse modo um número (quase) arbitrário de Ethernets pode ser conectado juntamente com as pontes.

O importante é que:

As pontes concentram os detalhes da interconexão: um conjunto de segmentos em ponte age como uma simples Ethernet.

Um computador utiliza exatamente o mesmo hardware para se comunicar com outro computador através da ponte e no segmento local.

A maioria das pontes faz muito mais do que duplicar quadros de um fio a outro: tomam decisões importantes, como a escolha de quais quadros passar adiante. Tais pontes são chamadas de adaptadoras ou de aprendizagem. Uma ponte adaptadora possui um computador com duas interfaces de Ethernet. O software em uma ponte adaptadora possui duas listas de endereços, uma para cada interface. Quando um quadro chega de uma Ethernet E₁, a ponte adaptadora inclui o endereço de origem da Ethernet de 48 bits à lista associada com E₁. Do mesmo modo, quando um quadro chega de uma Ethernet E₂ a ponte inclui o endereço de origem à lista associada com E₂. Assim, com o passar do tempo, a ponte adaptadora saberá que equipamentos encontram-se na E₁ e na E₂.

Após gravar o endereço de origem de um quadro, a ponte adaptadora utiliza o endereço de destino para decidir se passa adiante, ou não, o próximo quadro. Caso a lista de endereços indique que o destino encontra-se na Ethernet da qual o quadro se originou, a ponte não dará continuidade ao quadro. Se o destino não se encontrar na lista de endereços (ou seja, o destino é um endereço de difusão, ou é um endereço de multicast, ou a ponte ainda não encontrou a instalação do destinatário), a ponte passará o quadro adiante, a outra Ethernet.

As vantagens das pontes adaptadoras devem estar óbvias. Já que a ponte utiliza os endereços encontrados no tráfego normal, ela é completamente automática – não é necessário configurar a ponte com endereços específicos. Já que não passa adiante o tráfego desnecessário, uma ponte ajuda a aumentar a eficiência de uma rede congestionada, isolando o tráfego em segmentos específicos. As pontes operam excepcionalmente bem se uma rede puder ser dividida fisicamente em dois segmentos que contenham um conjunto de computadores com comunicações freqüentes (p. ex., cada segmento contém um conjunto de estações de trabalho juntamente com um servidor). Ou seja:

Uma ponte Ethernet adaptadora conecta dois segmentos de Ethernet, passando quadros de uma para outro. Através dos endereços de origem, identifica os equipamentos que se encontram nos segmentos da Ethernet e agrupa informações adquiridas pelos endereços de destino para não realizar uma transmissão desnecessária.

De acordo com a visão do TCP/IP, as Ethernets ligadas por pontes são simplesmente uma outra forma de conexão física de rede. O importante é que:

Em virtude da conexão entre cabos físicos, fornecida pelas pontes e pelos repetidores, ser transparente para os equipamentos conectados à Ethernet,

consideramos os segmentos múltiplos da Ethernet, conectados por pontes e repetidores, um sistema único de rede física.

A maioria das pontes comercializadas no mercado é muito mais sofisticada e robusta do que as que foram descritas aqui. Na primeira vez em que são colocadas em operação, verificam a existência de outras pontes e fazem o reconhecimento da topologia da rede. Utilizam um algoritmo spanning-tree distribuído para decidir qual o melhor modo de transmitir os quadros. As pontes decidem como distribuir os pacotes de difusão, de modo que somente uma cópia de cada quadro de difusão seja encaminhado a cada cabo. Sem tal algoritmo, as Ethernets e as pontes conectadas em círculo produziriam resultados catastróficos, pois iriam transmitir pacotes em ambas as direções, simultaneamente.

9.2.5 ATM (Asynchronous Transfer Mode)

O ATM (Asynchronous Transfer Mode) é o nome dado a uma tecnologia de rede de alta velocidade, baseada em conexão, que vem sendo usada tanto nas redes que operam em pequenas como em grandes áreas geográficas. Pelos padrões correntes, as redes de alta velocidade são aquelas que operam a uma velocidade de, no mínimo, 100 Mbps; o ATM pode intercambiar informações com velocidades de até gigabit/segundo. É claro que para se obter velocidades tão altas é necessário um equipamento complexo de última geração. Conseqüentemente, as redes ATM possuem um custo mais alto do que as demais tecnologias.

Para alcançar altas velocidades de transferência, uma rede ATM utiliza técnicas de hardware e software especiais. Em primeiro lugar, uma rede ATM é formada por um ou mais comutadores de alta velocidade que são conectados aos computadores host e a outros comutadores ATM. Em segundo lugar, o ATM utiliza fibras óticas para fazer conexões, inclusive conexões entre um computador host e um comutador ATM. As fibras óticas possuem uma velocidade de transferência maior do que a dos fios de cobre; normalmente, a conexão entre um host e um comutador ATM opera a uma velocidade de 100 ou 155 Mbps. E, em terceiro, as camadas mais baixas de uma rede ATM utilizam quadros de tamanhos fixos chamados células. Como as células possuem exatamente o mesmo tamanho, o hardware do comutador ATM pode processá-las rapidamente.

9.2.6.1 O tamanho da célula do ATM

Por incrível que pareça, cada célula do ATM possui somente 53 octetos. A célula possui cinco octetos de cabeçalho, seguidos por 48 octetos de informação. Os capítulos posteriores mostrarão que ao utilizar uma tecnologia ATM para enviar um tráfego IP, o tamanho de 53 octetos é irrelevante – uma rede ATM recebe e transmite pacotes muito maiores.

9.2.6.2 Rede baseada em conexão

A rede ATM difere das redes de comutação de pacotes porque oferece um serviço em conexão. Antes de um computador host conectado a um ATM enviar células, ele deve primeiramente interagir com o comutador para especificar o endereço do destinatário. A interação é análoga a uma ligação telefônica. O host especifica o endereço do computador remoto e espera o comutador ATM entrar em contato com o sistema remoto e estabelecer um caminho (rota fixa). Se o computador remoto não aceitar o pedido, não responder ou se o comutador ATM não puder alcançar o computador remoto, o pedido para estabelecer a comunicação falha.

Quando uma conexão é feita, o comutador ATM local escolhe um identificador da conexão e passa-o para o host, juntamente com uma mensagem informando o sucesso da conexão. O host utiliza o identificador da conexão ao enviar ou receber células.

Ao terminar uma conexão, o host comunica-se novamente com o comutador ATM para que a conexão seja desfeita. O comutador desconecta os dois computadores. A desconexão equivale a tirar um telefone do gancho no final de uma ligação telefônica; após a desconexão, o comutador pode novamente utilizar o identificador da conexão.

Capítulo

10

Camada de Rede

A camada de rede se ocupa de levar pacotes desde a origem por todo o caminho até o destino. Chegar ao destino pode exigir muitos saltos em nós intermediários ao longo do percurso. Essa função contrasta nitidamente com a da camada de enlace de dados, que tem o objetivo mais modesto de simplesmente mover quadros de uma extremidade para outra de um fio. Portanto, a camada de rede é a camada mais baixa que lida com transmissão fim a fim.

Para alcançar seus objetivos, a camada de rede deve conhecer a topologia da sub-rede de comunicações e escolher caminhos apropriados através dela. Também deve cuidar de escolher rotas que evitem a sobrecarga de algumas linhas de comunicação enquanto outras ficam ociosas. Finalmente, quando a origem e o destino estão em redes diferentes, é tarefa da camada de rede lidar com essas diferenças e solucionar os problemas que resultam do fato.

10.1 Temas de Projeto da Camada de Rede

Nesta seção forneceremos uma introdução a alguns dos temas com que devem se envolver os projetistas da camada de rede. Esses temas incluem o serviço fornecido à camada de transporte, o roteamento de pacotes através da sub-rede, controle de congestionamento e a conexão de múltiplas redes (interconexão de redes). Examinaremos os três últimos assuntos com maiores detalhes, mais adiante, neste capítulo.

10.1.1 Serviços Fornecidos à Camada de Transporte

A camada de rede fornece serviços à camada de transporte. Como em algumas redes (p.ex, as redes ARPANET X.25), a camada de rede funciona nos IMPs e a camada de transporte nos hosts, a fronteira entre as camadas de rede e de transporte nessas redes também é a fronteira entre a sub-rede e os hosts. Isso significa que os serviços fornecidos pela camada de rede definem os serviços proporcionados pela sub-rede.

Quando a sub-rede é executada por uma concessionária comum ou PTT e os hosts pertencem aos usuários, o serviço da camada de rede se torna a interface entre a concessionária e os usuários. Dessa forma, ela define os direitos e responsabilidades da concessionária, e por esse motivo é de grande importância tanto para a concessionária como para os usuários. Como se poderia esperar sob essas circunstâncias, tem havido grande discussão e controvérsia em relação aos serviços que devem ser oferecidos. Vamos examinar brevemente essa controvérsia.

Os serviços da camada de rede foram projetados tendo em vista os seguintes objetivos:

1. Os serviços devem ser independentes da tecnologia da sub-rede.
2. A camada de transporte deve estar isolada do número, tipo e topologia das sub-redes presentes.
3. Os endereços da rede que se tornam disponíveis para a camada de transporte devem usar um plano de numeração uniforme, mesmo através de lance e WANs.

Dados esses objetivos, os projetistas da camada de rede tiveram bastante liberdade para escrever especificações detalhadas dos serviços a serem oferecidos à camada de transporte. Essa liberdade rapidamente degenerou em uma violenta batalha entre duas facções opostas. O ponto central da discussão era a dúvida se a camada de rede deveria fornecer serviço baseado em conexões ou serviço sem conexões.

Esse mesmo tema ocorre na camada de enlace de dados, como vimos no capítulo anterior, mas não é tão sério naquele caso. A definição do LLC fornece ambos, e como as LANs são sempre de propriedade dos usuários e operadas por eles, os usuários podem utilizar o que preferirem. Com a camada de rede, a situação é diferente. Se a concessionária só oferece um tipo de serviço e os usuários desejam o outro, em geral não existe nenhuma alternativa.

Um grupo (representado pela comunidade Internet ARPA) argumenta que o trabalho da sub-rede é mover bits de um lado para outro e nada mais. No seu ponto de vista (baseado em praticamente 20 anos de experiência real com uma rede em funcionamento), a sub-rede é inherentemente não-confiável, não importa como é projetada. Por conseguinte os hosts devem aceitar o fato de que ela é não-confiável e fazer eles mesmos o controle de erros (i.e, a detecção e detecção de erros) e o controle de fluxo.

Esse ponto de vista conduz rapidamente à conclusão de que o serviço da rede deve ser sem conexões, com primitivas *ENVIAR PACOTE* e *RECEBER PACOTE*. Em particular, nenhum controle de fluxo e verificação de erros deve ser feito, porque os hosts vão fazer isso de qualquer forma, e há provavelmente pouca vantagem em fazê-lo duas vezes. Além disso, cada pacote tem de transportar o endereço de destino completo, porque cada pacote enviado expedido independentemente dos seus predecessores, se houver.

O outro grupo (representado pelas concessionárias) argumenta que cada camada de rede (e a sub-rede) deve fornecer um serviço confiável, baseado em conexões que têm as seguintes propriedades:

1. Antes de enviar dados, a entidade de transporte de origem deve configurar uma conexão para a entidade de transporte de destino. Essa conexão, que recebe um identificador especial, é então utilizada até que não existam mais dados a enviar, e ao fim desse tempo a conexão é explicitamente liberada.
2. Quando uma conexão é definida, as duas entidades de transporte e a camada de rede que fornece o serviço podem entrar em uma negociação sobre os parâmetros do serviço e sobre a qualidade e o custo do serviço a ser fornecido.
3. A comunicação se dá em ambos os sentidos, e os pacotes são entregues em seqüência, sem erros. O modelo conceitual por trás disso é uma fila bem-comportada com a propriedade de primeiro a entrar, primeiro a sair.
4. O fluxo de controle é fornecido automaticamente, para evitar que um transmissor rápido despeje pacotes na fila a uma velocidade maior do que aquela em que o receptor podem tirá-los de lá, o que levaria a um estouro.

Outras propriedades, tais como uma confirmação explícita de entrega e pacotes de alta prioridade, são opcionais.

Uma analogia entre o serviço baseado em conexões e o serviço sem conexões pode esclarecer a situação. A rede telefônica pública oferece um serviço baseado em conexões. Primeiro, o cliente disca um número para estabelecer uma conexão. Depois, as partes conversam (trocam dados). Finalmente, a conexão é desfeita. Embora o que acontece no interior do sistema telefônico seja sem dúvida muito complicado, os dois usuários são presenteados com a ilusão de um canal dedicado ponto a ponto que sempre entrega as informações na ordem em que são enviadas.

Em contraste, o sistema postal (ou sistema telegráfico) é sem conexões. Cada carta leva o endereço completo do destino e é transportada independentemente de todas as outras cartas. As cartas não chegam necessariamente na ordem em que foram postadas. Se um carteiro perde accidentalmente uma carta, o correio não se interrompe para enviar uma duplicata. Em resumo, o controle de erros e o controle de fluxo são tratados pelos próprios usuários, fora do sistema postal (sub-rede). A Figura 5-1 resume as diferenças entre o serviço baseado em conexões e o serviço sem conexões.

A argumentação sobre serviços baseados em conexões e sem conexões envolve realmente a questão de onde inserir a complexidade. No serviço baseado em conexões, ela está na camada de rede (sub-rede); no serviço sem conexões ela está na camada e transporte (*hosts*). Os partidários do serviço sem conexões dizem que a capacidade de computação do usuário está se tornando barata, e assim não há razão para inserir a complexidade nos *hosts* (freqüentemente em chip de co-processador especial de rede). Além disso, argumenta-se que a sub-rede é um investimento nacional importante que irá subsistir por décadas, de modo que não deve ser atravancado com recursos que podem se tornar rapidamente obsoletos, mas terão de ser calculados na estrutura de preços por muitos anos. Ademais, algumas aplicações, tais como voz digitalizada e a coleta de dados em tempo real, podem considerar a entrega *veloz* como muito mais importante do que a entrega *precisa*.

Por outro lado, os defensores do serviço baseado em conexões dizem que a maior parte dos usuários não está interessada em executar complexos protocolos da camada de transporte em suas máquinas. O que eles querem é um serviço confiável e livre de problemas, e esse serviço pode ser mais bem suprido com conexões. Finalmente, existe o problema real da credibilidade. Se os usuários não estão dando crédito os reclames da concessionária de que a sub-rede só perde um pacote uma vez em cada lua cheia, eles irão implementar toda a verificação de erros de qualquer forma na camada de transporte para se protegerem. Entretanto, se um número suficiente de pessoas fizer isso, torna-se um desperdício gastar um bocado de esforço e dinheiro na camada de rede para conseguir

Tema	Baseados em conexão	Sem conexão
Endereço de destino	Necessário apenas durante a configuração	Necessário em cada pacote
Seqüência de pacotes	Garantido	Não garantido
Controle de erros	Feito pela camada de rede (p. ex, sub-rede).	Feito pela camada de transporte (p. ex, hosts).
Controle de fluxo	Fornecido pela camada de rede	Não fornecido pela camada de rede

É possível negociação de opções?	Sim	Não
São usados identificadores de conexões?	Sim	Não

Figura 10-1 Resumo das principais diferenças entre o serviço baseado em conexões e o serviço sem conexões

alta confiabilidade. A sub-rede poderia simplesmente fornecer um serviço barato e simples sem conexões e dizer aos usuários para fazer todo o serviço, visto que muitos deles vão fazê-lo de qualquer maneira, não importa o que a concessionária lhes diga.

Para encurtar a história, os defensores do serviço baseado em conexões superaram em número os defensores do serviço sem conexões no comitê que escreveu a definição do serviço da camada de rede, e assim o serviço de rede OSI foi originalmente um serviço baseado em conexões. (Foi dito que os PTTs preferiam o serviço baseado em conexões porque não poderiam cobrar pelo tempo de conexão se não houvesse conexão). Contudo, as pessoas favoráveis ao serviço sem conexões mantiveram um lobby pela causa e a ISSO modificou posteriormente a definição do serviço para incluir as duas classes de serviços. Ambos os tipos são agora permitidos, e os protocolos para suporte a eles foram incorporados à estrutura do modelo OSI.

A camada de rede não é o único campo de batalha entre os dois grupos. Os mesmos temas surgem em todas as camadas. A luta foi finalmente resolvida em definitivo por uma revisão no modelo OSI de modo a permitir que os dois tipos de serviços sejam oferecidos daí em diante. Na figura 5-2, vemos como os dois tipos de serviço se relacionam um com o outro. Na parte superior de cada camada (exceto na camada de aplicação) estão os PASs (pontos de acesso ao serviço), através dos quais a camada acima acessa os serviços. Cada PAS tem um endereço único que o identifica. A partir da camada de enlace de dados, esses serviços podem ser baseados em conexões ou sem conexões (o tema é discutível na camada física visto que o controle de erros e o controle de fluxo não são relevantes nessa camada).

Dois caminhos óbvios são o serviço completamente baseado em conexões de cima para baixo e o serviço sem conexões. Entretanto, também é possível o serviço baseado em conexões fornecido pelas camadas de rede ou de transporte mesmo que as camadas inferiores sejam sem conexões. Nesse caso, as camadas de rede ou de transporte devem tratar a conversão. Por exemplo, um serviço de transporte baseado em conexões poderia ser construído em uma LAN com um serviço de enlace de dados sem conexões colocando-se o controle de erros, o controle de fluxo e funcionalidades relacionadas na camada de rede ou na camada de transporte.

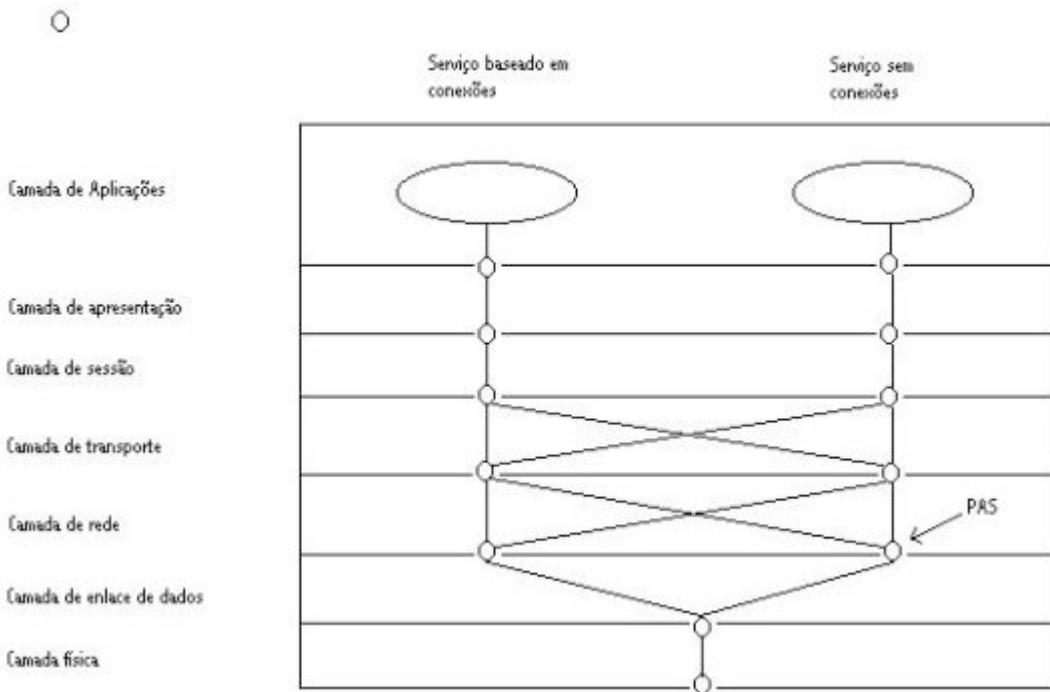


Figura 10-2 Misturas de serviços baseados em conexões e serviços sem conexões no modelo OSI

O mecanismo inverso, implementando um serviço sem conexões para as camadas superiores na parte alta de um serviço baseado em conexões também é possível. Embora isso pareça desperdício, poderia ser útil na conexão de duas LANS completamente sem conexões com uma WAN que oferece somente o serviço de rede baseado em conexões. Em qualquer evento, a conversão entre os dois tipos de serviço pode ser feita pela camada de rede ou pela de transporte, mas não em camadas mais altas.

As Primitivas de Serviço da Rede OSI

O Padrão Internacional 8348 define o serviço de rede pela especificação das primitivas que se aplicam à fronteira entre a camada de rede e a camada de transporte. São fornecidas tanto as primitivas baseadas em conexões quanto as sem conexões. As primitivas baseadas em conexões utilizam o modelo da Figura 5-3. Nesse modelo, uma conexão é um par de filas conceituais entre dois PASRs (endereços de rede), uma fila para o tráfego em cada sentido. Antes de estabelecer uma conexão, temos a situação da Figura 5-3(a). Depois de um estabelecimento bem-sucedido, temos a situação da Figura 5-3(b). Finalmente, depois que três pacotes são enviados, o estado das conexões poderia ser semelhante à Figura 5-3(c).

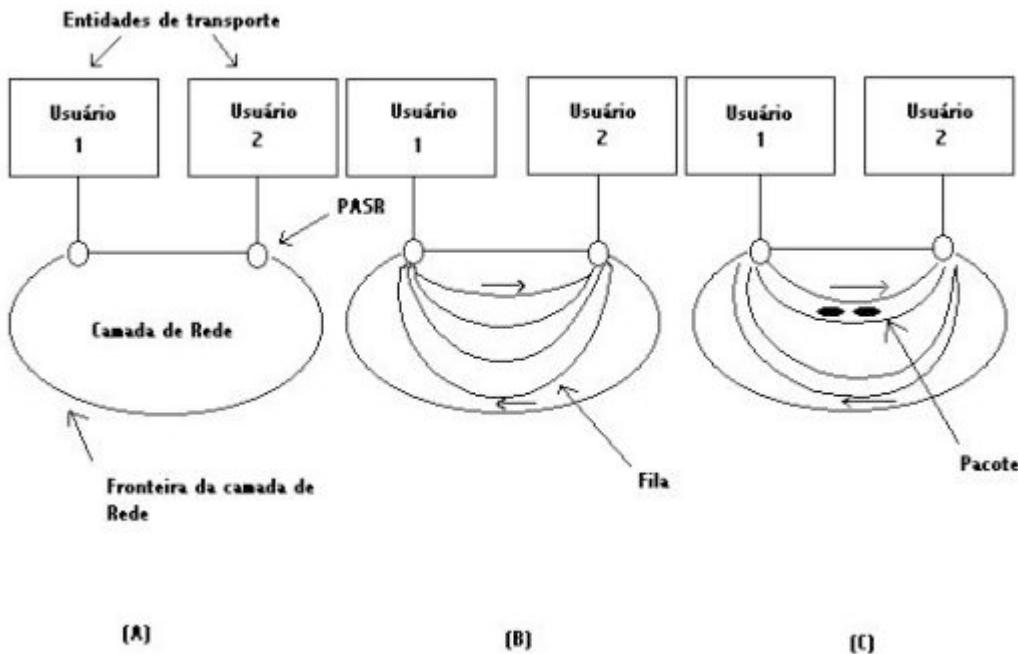


Figura 10-3 (A) Antes de estabelecer uma conexão (B) Depois de estabelecer uma conexão (C) Depois que dois pacotes são enviados, mas ainda não recebidos.

As primitivas de serviço de rede baseada em conexões do modelo OSI estão listadas na Figura 5-4(a). Elas podem ser agrupadas em quatro categorias, para estabelecer, liberar usar e reinicializar conexões, respectivamente. A maior parte das primitivas tem parâmetros. A maneira exata como os parâmetros são passados para as primitivas é dependente da implementação. O efeito de cada primitiva pode ser descrito pelo modo como ela altera o estado das filas da Figura 5-3.

A primitiva *N_CONNECT.pedido*, usada para estabelecer uma conexão, especifica o endereço de rede conectar e o endereço de rede do chamador. E também contém duas variáveis booleanas usadas para solicitar a confirmação de cada pacote enviado. Se a camada de rede não fornecer confirmações a variável é fixada em *falso* quando é entregue ao destino da primitiva *N-CONNECT.indicação*. Se a camada de rede fornece confirmações mas o destino não deseja usa-las, então ele fixa o flag em *falso* na sua *N-CONNECT.resposta*. Somente se as duas entidades de transporte e o fornecedor do serviço de rede quiserem usa-las elas são utilizadas. Esse recurso é um exemplo de negociação de opções.

O flag *exp_procurados* é um segundo exemplo de negociação de opções. Se aceito pelas três partes, permite o uso de dados expressos, essencialmente pacotes que podem violar a ordenação normal da fila saltar para o início dela. O fato de fazerem realmente isso ou não, é dependente da implementação. Um exemplo típico de dados expressos é um usuário em um terminal pressionando a tecla DEL para interromper um programa em execução. O pacote DEL irá como dados expressos.

O parâmetro *qds* é formado na realidade de duas listas de valores que determinam a qualidade do serviço fornecido pela conexão. A primeira lista determina o objetivo-o que o chamador deseja realmente. A segunda lista determina os valores mínimos considerados aceitáveis. Se o serviço de rede é incapaz de fornecer pelo menos o valor mínimo de qualquer parâmetro especificado pelo chamador (parte que faz a chamada) ou pelo chamado (parte que recebe a chamada), o estabelecimento da

conexão falha. Os valores que podem ser especificados incluem throughput, retardo, taxa de erros, sigilo e custo, entre outros.

N-CONNECT.request (chamado, chamador, confs_procuradas, gds, dados_do_usuario).
N-CONNECT.indication (chamado, chamador, confs_procuradas, gds, dados_do_usuario).
N-CONNECT.response (respondedor, confs_procuradas, gds, dados_do_usuario).
N-CONNECT.confirmation (respondedor, confs_procuradas, gds, dados_do_usuario).
N-DISCONNECT.request (originador, motivo, dados_do_usuario, endereço_de_resposta).
N-DISCONNECT.indication (originador, motivo, dados_do_usuario, endereço_de_resposta).
N-DATA.request (dados_do_usuario)
N-DATA.indication (dados_do_usuario)
N-DATA.ACNOWLEDGE.request ()
N-DATA.ACNOWLEDGE.indication ()
N-EXPEDITED-DATA.request (dados_do_usuario)
N-EXPEDITED-DATA.indication (dados_do_usuario)
N-RESET.request (originador, motivo).
N-RESET.indication (originador, motivo).
N-RESET.response ()
N-RESET.confirmation ()

(a)

N-UNITDATA.request (endereço_de_origem, endereço_de_destino, qds, dados_do_usuario).
N-UNITDATA.indication (endereço_de_origem, endereço_de_destino, qds, dados_do_usuario).
N-FACILITY.request (qds)
N-FACILITY.indication (endereço_de_destino, qds, motivo).
N-REPORT.indication (endereço_de_destino, qds, motivo).

(b)

Figura 10-4 (a) Primitivas de serviço da rede baseada em conexões OSI (b) Primitivas de serviço da rede sem conexões OSI

Notas sobre a terminologia:

- Chamado: Endereço da rede (PASR) a ser chamado
- Chamador: Endereço da rede (PASR) usado pela entidade de transporte que chama
- Conf_procuradas: Flag booleano e especificando se confirmações são desejadas
- Exp_procurados: Flag booleano especificando se os dados expedidos serão enviados
- Qds: Qualidade do serviço desejado

- Dados _ do _ usuário: 0 ou mais bytes de dados transmitidos mas não examinados
- Respondedor: Endereço da rede (PASR) conectado ao destino
- Originador: Especificação da razão para que o evento tenha acontecido

O chamador pode incluir alguns dados do usuário no pedido de conexão. O chamado pode inspecionar esses dados antes de decidir se aceita ou rejeita o pedido. Os pedidos de conexões são aceitos com a primitiva *N-CREATE*.resposta e rejeitados com a primitiva *N-DISCONNECT*.pedido. Quando um pedido é rejeitado, o campo *motivo* permite que o chamado informe por que não aceitou a conexão e se a condição é permanente ou transitória. A própria camada de rede também pode rejeitar tentativas de estabelecer conexões, por exemplo, se a qualidade do serviço desejado não estiver disponível (condição permanente) ou a sub-rede estiver sobrecarregada atualmente (condição transitória).

As primitivas *N-CREATE* restantes e as primitivas *N-DISCONNECT* são objetivas e necessitam de poucos comentários adicionais. Depois que uma conexão é estabelecida, qualquer das partes pode transmitir dados usando *N-SEND*.pedido. Quando esses pacotes chegam, uma primitiva *N-SEND.indicação* é invocada na extremidade de recepção. Os dados expressos usam primitivas análogas a essas últimas para dados regulares.

Se as confirmações tiverem sido admitidas, quando um pacote é recebido, espera-se que o recipiente emita uma *N-SEND-ACKNOWLEDGE*.pedido. Essa primitiva não contém qualquer número de seqüência, e assim a parte que envia os dados originais deve simplesmente contar confirmações. Se a qualidade do serviço é baixa e os dados e confirmações podem ser perdidos, esse esquema não é muito satisfatório. Por outro lado, não é tarefa da camada de rede fornecer um serviço isento de erros; esse trabalho é realizado pela camada de transporte. As confirmações da camada de rede são meramente uma tentativa de melhorar a qualidade do serviço, e não de torná-lo perfeito.

As primitivas *N-RESET* são usadas para relatar catástrofes, tais como quedas de qualquer entidade de transporte ou do próprio fornecedor do serviço de rede. Depois que uma *N-RESET* é pedida, indicada, respondida e confirmada, as filas deverão ao seu estado original vazio. Os dados presentes nas filas no momento do *N-RESET* são perdidos. Novamente nesse caso, é trabalho da camada de transporte recuperar-se de *N-RESETs*.

As primitivas OSI sem conexões são dadas na Figura 5-4(b). As primitivas *N-UNITDATA* são usadas para enviar até 64.512 bytes de dados (1k menor que 2^{16} , para proporcionar bastante espaço para diversos cabeçalhos e ainda manter a unidade final menor que 2¹⁶ bytes). As primitivas *N-UNITDATA* não fornecem nenhum controle de erros, nenhum controle de fluxo e nenhum outro controle. O transmissor apenas descarrega o pacote na sub-rede e espera pelo melhor.

A primitiva *N-FACILITY*.pedido é projetada de modo a permitir que um usuário de serviço de rede pergunte sobre características médias de entrega ao destino especificado, tais como a porcentagem de pacotes que estão sendo entregues. A

primitiva *N-FACILITY.indicação* vem da própria camada de rede, não de uma entidade de transporte remota. A primitiva N-REPORT.indicação permite à camada de rede relatar problemas ao usuário de serviço de rede. Por exemplo, se um determinado destino está disponível, esse fato poderia ser relatado com essa primitiva. Os detalhes de como a primitiva é usada são dependentes da rede e não definidos no padrão.

Uma das funções da camada de rede é proporcionar uma nomenclatura uniforme para uso da camada de transporte. No caso ideal, todos os operadores da rede pelo mundo deveriam se juntar e concordar sobre um espaço de nomes único, de modo que cada fio que se estendesse a partir de uma sub-rede em um escritório ou lar teria um endereço único em nível mundial. Infelizmente, isso está longe de acontecer.

Para fazer o melhor uso possível da situação existente, o endereçamento da camada de rede do modelo OSI foi projetado para incorporar a diversidade de endereçamento de rede de hoje em dia. Todas as primitivas de serviço da rede utilizam endereços PASR para identificar a origem e o destino. O formato desses endereços PASR é mostrado na Figura 5-5. Cada endereço PASR tem três campos. O primeiro é o campo IAF (Identificador de Autoridade e Formato), que identifica o tipo de endereço presente no terceiro campo, o PED (Parte Específica do Domínio). Os códigos foram alocados a fim de permitir que o campo PED contenha endereços de pacotes da rede, números de telefones, números ISDN, números de telex e esquemas de numeração similares existentes, tanto em binário como em decimal compactado. O campo IAF pode conter valores de 10 até 99, deixando grande quantidade de espaço para planos de numeração futuros.

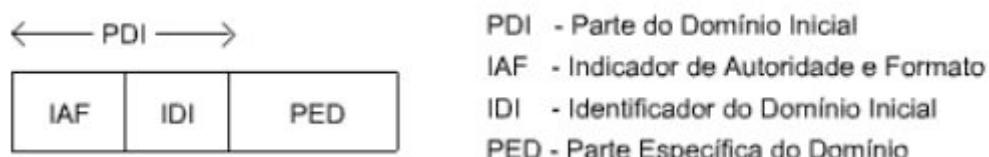


Figura 10-5 O formato dos endereços da rede OSI (PASRs)

O segundo, ou campo IDI (Identificador do domínio inicial), especifica o domínio ao qual pertence o número na parte PED. Por exemplo, se o PED é um número de telefone, o IDI poderia ser o código do país para esse número. O endereço PASR tem comprimento variado, de até 40 dígitos decimais ou 20 bytes de extensão.

A fim de compreender para que servem os endereços PASR, pode ser útil fazer uma analogia ou com o sistema telefônico. A maioria dos escritórios e lares modernos são equipados com uma ou mais tomadas às quais os telefones podem ser conectados. Cada tomada tem um fio que se estende até uma central de comutação telefônica (ou PBX). A essa tomada é atribuído em número único em nível mundial, consistindo em um código de país, um de área e um número de assinante.

Sempre que um telefone é conectado a uma tomada, esse telefone pode ser alcançado pela discagem do número da tomada. Note que o número realmente se aplica à *tomada*, não ao telefone atualmente conectado a ela. Quando é deslocado de um escritório para outro, o telefone adquire o número do novo escritório; não leva o número antigo com ele. Portanto, as tomadas dão ao sistema telefônico um espaço de

nomes uniforme, independente de qual telefone em particular possa estar conectado no momento, ou de qual pessoa estará respondendo quando ele tocar.

Nessa analogia, as tomadas telefônicas são os PASRs e os números de telefones são os endereços PASR. Quando uma entidade de transporte pede à rede que faça uma chamada para uma máquina remota, ela especifica o endereço PASR (i.e., o número do telefone) a ser chamado. A camada de rede não cuida de qual entidade de transporte (i.e., telefone) está atualmente associada (conectada) a esse PASR, e certamente não sabe a que usuário pertence a entidade de transporte (i.e., que pessoa irá responder ao telefone). O modo como as entidades de transporte se conectam aos PASRs é um assunto da camada de transporte, e não da camada de rede.

10.1.2 Organização Interna da Camada de Rede

Tendo examinado as duas classes de serviços que a camada de rede pode fornecer aos seus usuários, é hora de ver o seu funcionamento interno. Infelizmente, o modelo OSI não fornece uma especificação dos algoritmos básicos, tais como o roteamento e o controle do congestionamento. Estes são dependentes da implementação. Apesar disso, eles são importantes, e definitivamente vale a pena estuda-los em detalhes.

Como uma consequência, este capítulo irá abordar muitos temas de projetos de sub-redes que não são estritamente parte da camada de rede, embora estejam relacionados com ela. Também discutiremos, onde for apropriado, a camada de rede do OSI. Para tornar claro onde estaremos discutindo o projeto de sub-redes (em oposição ao OSI), usaremos os termos de sub-redes “IMP” e “host” em lugar de “camada de rede” e “camada de transporte”, embora os últimos sejam equivalentes em algumas redes.

Existem basicamente duas filosofias diferentes para organização da sub-rede, uma usando conexões e a outra funcionando sem conexões.

No contexto da operação **interna** da sub-rede, uma conexão é usualmente chamada de circuito virtual, em analogia aos circuitos físicos estabelecidos pelo sistema telefônico. Os pacotes independentes da organização sem conexões são chamados datagramas, em analogia aos telegramas.

Os circuitos virtuais são usados geralmente em sub-redes cujo serviço principal é baseado em conexões, e assim os descreveremos nesse contexto. A idéia básica dos circuitos virtuais é evitar que se tenha de tomar decisões sobre roteamento para cada pacote enviado. Em vez disso, quando uma conexão é estabelecida, escolhe-se uma rota desde a máquina de origem até a de destino, como parte da definição da conexão. Essa rota é usada para todo o tráfego que flui pela conexão, exatamente da mesma forma como funciona o sistema telefônico. Quando a conexão é desfeita, o circuito virtual é descartado.

Em contraposição, com uma sub-rede de datagramas nenhuma rota é estabelecida antecipadamente, mesmo que o serviço seja baseado em conexões. Cada pacote enviado é roteado independentemente dos seus predecessores. Pacotes sucessivos podem seguir diferentes rotas. Embora tenham de realizar mais trabalho, as sub-redes de datagramas são também mais resistentes e se adaptam a falhas e congestionamentos com mais facilidade do que as sub-redes de circuitos virtuais. Discutiremos mais adiante os prós e os contras das duas abordagens. Se os pacotes que fluem por um determinado circuito virtual usarem sempre a mesma rota através da sub-rede, cada IMP deve lembrar para onde seguem os pacotes de cada um dos circuitos virtuais atualmente abertos que passam através dele.

Todos os IMPs devem manter uma tabela com uma entrada para cada circuito virtual aberto. Evidentemente, os circuitos virtuais que não passam através do IMP X não têm entradas na tabela de X. Cada pacote que viaja através da sub-rede deve conter em seu cabeçalho um campo de número de circuito virtual, além dos números de seqüência, somas de verificação e outros. Quando um pacote chega a um IMP, o IMP sabe sobre qual linha ele chegou e qual é o número do circuito virtual. Baseado apenas nessas informações, o pacote tem de ser direcionado para o IMP correto.

Quando uma conexão de rede é definida, um número de circuito virtual que ainda não esteja em uso nessa máquina é escolhido como identificador da conexão. Como cada máquina escolhe independentemente os números de circuitos virtuais, o mesmo número de circuito virtual tem probabilidade de estar em uso em dois diferentes caminhos através de algum IMP intermediário levando a ambigüidades.

Considere a sub-rede da Figura 5-6(a). Suponha que um processo no host de A deseja se comunicar com um processo no host de D. A escolhe o circuito virtual 0. Vamos imaginar que seja escolhida a rota *ABCD*. Simultaneamente, um processo em B decide que deseja se comunicar com um processo em D (não o mesmo que o processo de A escolheu). Se não houver nenhum circuito virtual aberto começando em B nesse ponto, o host B também escolherá o circuito virtual 0. Suponha ainda que a rota *BCD* seja selecionada como a melhor. Depois que ambos os circuitos virtuais foram definidos, o processo em A envia sua primeira mensagem para D, no circuito virtual 0. Quando o pacote chega a D, o host não sabe qual processo do usuário lhe deu o pacote.

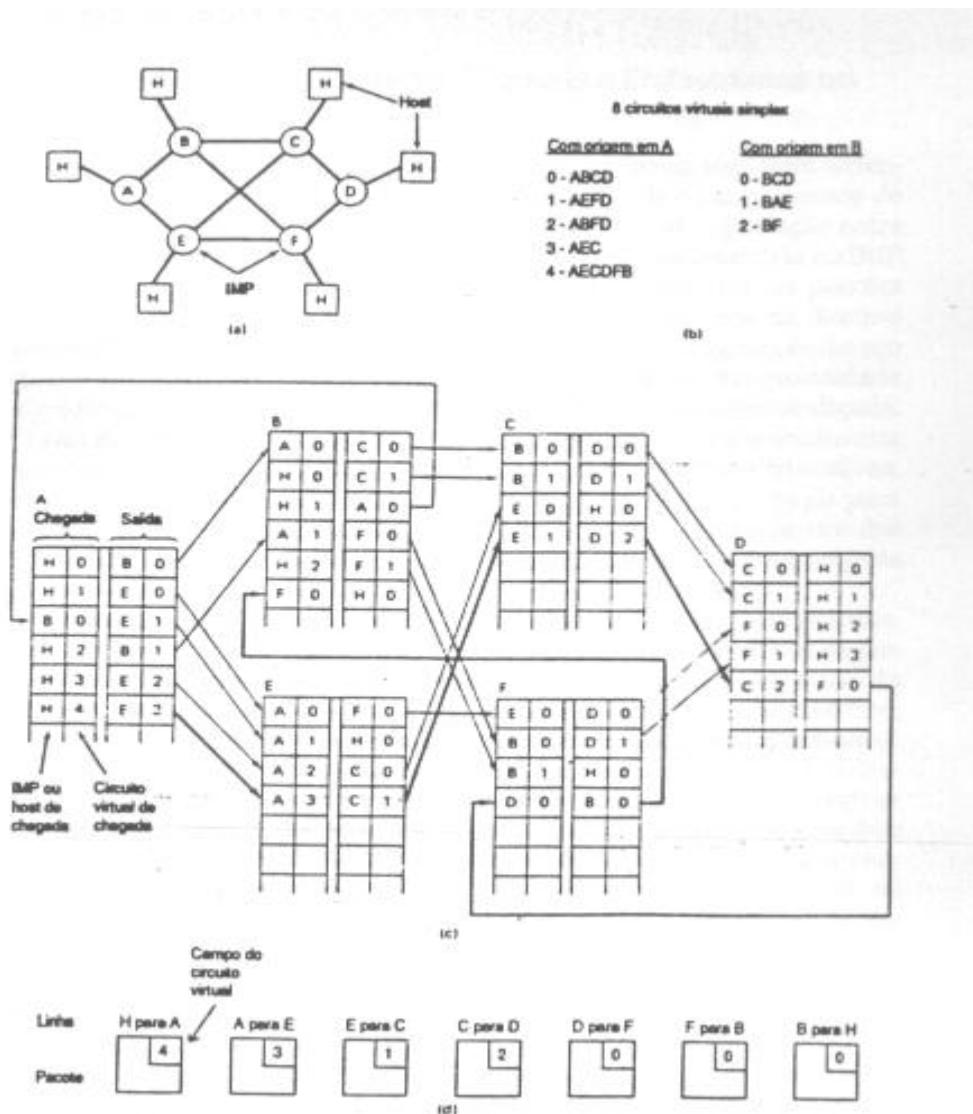


Figura 10-6 (a) Sub-rede de exemplo (b) Oito circuitos virtuais através da sub-rede
(c) Tabelas do IMP para os circuitos virtuais em (b) (d) As mudanças no circuito virtual como uma progressão de pacote

Para solucionar esse problema, sempre que deseja criar um novo circuito virtual de saída, o host escolhe o número de circuito mais baixo que não esteja em uso no momento. O IMP (digamos X) não dirige esse pacote definido para o IMP seguinte (digamos Y) ao longo da rota em que está. Em vez disso, X verifica em sua tabela todos os números de circuitos que estão sendo usados atualmente para o tráfego que chega a Y. Depois, escolhe o numero livre mais baixo e substitui o número que está no pacote por esse número, sobrescrevendo o número escolhido pelo host. De modo semelhante, o IMP Y escolhe o número de circuito mais baixo que esteja livre entre ele e o IMP seguinte.

Quando esse pacote de configuração chega finalmente ao destino, o IMP de lá escolhe o número de circuito de entrada mais baixo disponível, sobrescreve o número de circuito encontrado no pacote e passa o pacote para o host. Dessa forma, o host de destino sempre vê o mesmo número de circuito em todo o tráfego que chega por um determinado circuito virtual, não importando se o host de origem está usando coerentemente um número diferente.

A Figura 5-6(b) mostra oito exemplos de circuitos virtuais pertinentes a sub-rede da parte (a). A parte (c) da figura mostra as tabelas dos IMPs, supondo que os circuitos foram criados na ordem: *ABCD, BCD, AEFD, BAE, ABFD, BF, AEC e AECDFB*. O último circuito (*AECDFB*) pode parecer um pouco cheio de rodeios, mas, se as linhas *AB*, *BC* e *EF* fossem muito sobrecarregadas quando o algoritmo de roteamento fosse executado, essa poderia muito bem ser a melhor opção.

Cada entrada consiste em uma parte de entrada e uma de saída. Cada uma das duas partes tem um nome de IMP (usado para indicar uma linha) e um número de circuito virtual. Quando chega um pacote, a tabela é examinada na parte esquerda (de chegada), usando a linha de chegada e o número de circuito virtual encontrado no pacote como chave. Quando é encontrada uma coincidência, a parte de saída naquela posição da tabela informa em que número de circuito virtual deve ser inserido o pacote e para qual IMP ele deve ser enviado. *H* representa host, tanto no lado de entrada quanto no de saída.

Como exemplo, considere um pacote viajando do host A para o host B sobre o circuito virtual (i.e., rota *AECDFB*). Quando o IMP A recebe um pacote do seu próprio host, com o circuito 4, ele pesquisa em sua tabela, encontrando uma correspondência para H 4 na entrada 5 (a entrada superior é 0). A parte de saída dessa entrada é E 3, o que significa substituir o circuito 4 pelo circuito 3 e enviá-lo para o IMP E. Então, o IMP E recebe um pacote de A com o circuito 3 e assim procura por A 3 e encontra uma correspondência com a terceira entrada. Agora o pacote vai para C com o circuito 1. A seqüência de entradas usadas está assinalada pela linha em negrito. A Figura 5-6(d) mostra essa seqüência de números do pacote.

Pelo fato de que os circuitos virtuais podem ser inicializados a partir de qualquer extremidade, ocorre um problema quando definições de chamadas estão se propagando em ambos os sentidos a um só tempo ao longo de uma cadeia de IMPs. Em algum ponto eles chegarão a IMPs adjacentes. Cada IMP tem de extrair agora um número de circuito virtual para usar no circuito (full-duplex) que ele está tentando estabelecer. Se foram programados para escolher um número mais baixo que ainda não esteja em uso no elo, eles usarão o mesmo número, fazendo com que dois circuitos virtuais não relacionados sobre a mesma linha física tenham o mesmo número. Quando um pacote de dados chegar mais tarde, o IMP de recepção não tem meios para dizer se ele é um pacote no sentido direto em um circuito ou um pacote no sentido inverso em outro. Se os circuitos forem simplex, não existe nenhuma ambigüidade.

Observe que cada processo deve ser obrigado a indicar quando está fazendo uso de um circuito virtual, de modo que este possa ser purgado das tabelas do IMP, a fim de recuperar o espaço. Nas redes públicas, a motivação para isso é outra: os usuários são invariavelmente cobrados pelo tempo de conexão, bem como pelos dados transportados. Isso vale para o uso de circuitos virtuais internos à sub-rede. A outra possibilidade é usar datagramas internamente, em cujo caso os IMPs não têm uma tabela com uma entrada para cada circuito virtual aberto. Em vez disso, eles têm uma tabela informando qual linha de saída utilizar para cada IMP de destino possível. Essas tabelas também são necessárias quando os circuitos virtuais são usados internamente, a fim de especificar a rota para um determinado pacote. Cada datagrama deve conter o endereço completo de destino (a máquina e o endereço P ARP ao qual o processo de destino está associado). Quando um pacote chega, o IMP procura a linha de saída a usar e o envia através desse caminho. Nada no pacote é modificado. Além disso, o escabelecimento e a liberação das conexões da camada de rede ou de transporte não requer qualquer trabalho especial da parte dos IMPs.

Comparação entre Circuitos Virtuais e Datagramas no Interior da Sub-rede

Tanto os circuitos virtuais quanto os datagramas têm seus defensores e seus detratores. Tentaremos agora resumir os argumentos de ambas as partes. No interior da sub-rede, a principal negociação entre circuitos virtuais e datagramas está entre o espaço de memória no IMP e a banda passante. Os circuitos virtuais permitem que os pacotes contenham números de circuitos em lugar de endereços de destino completos. Se os pacotes tendem a ser claramente curtos, um endereço de destino completo em cada pacote pode representar uma quantidade significativa de overhead, e, portanto de banda passante desperdiçada. O uso de circuitos virtuais intermos à sub-rede se torna especialmente atrativo quando muitos dos hosts são realmente terminais interativos, com apenas uns poucos caracteres por pacote. O preço pago para utilizar internamente circuitos virtuais é o espaço da tabela dentro dos IMPS. Dependendo do custo relativo dos circuitos de comunicação versus memória do IMP, um ou outro pode ser mais barato.

Para sistemas de processamento de transações (p.ex., lojas, chamando para verificar compras com cartões de crédito), o overhead requerido para definir e limpar um circuito virtual pode diminuir o uso do circuito. Se a parte principal do tráfego que se espera ter é desse tipo, o uso de circuitos virtuais no interior da sub-rede faz pouco sentido. Os circuitos virtuais também apresentam um problema de vulnerabilidade. Se um IMP cair e perder sua memória, mesmo que retome um segundo depois, todos os circuitos virtuais que passam através dele terão de ser abortados. Em contraste, se um IMP de datagramas cair, somente os usuários cujos pacotes estavam enfileirados no IMP no momento da queda irão sofrer, e talvez nem mesmo todos eles, dependendo de já *terem* sido confirmados ou não. A perda de uma linha de comunicações é fatal para circuitos virtuais que a utilizam, mas pode ser facilmente compensada se forem usados datagramas. Os datagramas também permitem aos IMPs balancear o tráfego por toda a subrede, visto que as rotas podem ser alteradas no meio de uma conexão. Vale a pena explicitar que o serviço oferecido (baseado em conexões ou sem conexões) é um assunto separado da estrutura da sub-rede (circuito virtual ou datagrama). Em teoria, todas as quatro combinações são possíveis. Obviamente, uma implementação em circuito virtual de um serviço baseado em conexões e uma implementação de datagrama para um serviço sem conexões são razoáveis. A implementação de conexões usando datagramas também faz sentido quando a sub-rede está tentando fornecer um serviço altamente resistente.

A quarta possibilidade, um serviço sem conexões sobre uma subrede de circuito virtual, parece estranha, mas poderia acontecer em uma sub-rede projetada originalmente para serviço baseado em conexões, com o serviço sem conexões aplicado como uma reflexão posterior. Em tal arranjo, a sub-rede poderia ter de definir, utilizar e liberar um circuito virtual para cada pacote enviado, o que não seria muito agradável.

A figura 5-7 resume algumas diferenças entre as sub-redes que usam internamente datagramas e as que usam circuitos virtuais.

Tema	<i>Sub-rede de datagrama</i>	<i>Sub-rede circuito virtual</i>
Definição do Circuito	Não possível	Obrigatório
Obrigatório	Cada pacote contém os endereços completos de origem e destino	Cada pacote contém um número certo do circuito virtual

Informações sobre o estado	A sub-rede não contém informações sobre o estado	Cada circuito virtual estabelecido exige espaço na tabela da sub-rede
Roteamento	Cada pacote é roteado de forma independente	A rota é escolhida quando o circuito virutal é definido e todos os pacotes seguem esta rota
Efeito de falhas em nós	Nenhum, exceto para pacotes perdidos durante a queda	Todos os CVs que passaram através do equipamento que falhou estão encerrados
Controle de congestionamento	Difícil	Fácil se buffers em número suficiente podem ser colocados antecipadamente para cada CV definido
Complexidade	Na camada de transporte	Na camada de rede
Adequado a	Serviços baseados em conexões e serviço sem conexões	Serviço baseado em conexões

Figura 10-7 Comparação entre as sub-redes de datagramas e de circuito virtual. Note a semelhança com os serviços sem conexões da Figura 5-1.

10.1.3 Roteamento

A função real da camada de rede é rotear pacotes desde a máquina de origem até a máquina de destino. Na maioria das sub-redes, os pacotes irão exigir muitos saltos para realizar a viagem. A única exceção notável é o caso das redes de difusão, mas mesmo aqui o roteamento tem importância, se a origem e o destino não estão na mesma rede. Os algoritmos que selecionam as rotas e as estruturas de dados que utilizam são uma área importante do projeto da camada de rede. Nesta seção iremos descrever exatamente o problema. Mais adiante neste capítulo, examinaremos em detalhes muitos dos algoritmos que forem propostos.

O algoritmo de roteamento é parte do software da camada de rede responsável por decidir sobre que linha de saída um pacote que chega deve ser transmitido. Se a sub-rede usa internamente datagramas, essa decisão deve ser tomada novamente para cada pacote de dados que chega. Entretanto, se a sub-rede utiliza internamente circuitos virtuais, as decisões sobre o roteamento são feitas somente quando um novo circuito virtual está sendo definido. Depois disso, os pacotes de dados seguem exatamente a rota que foi estabelecida previamente. O último caso é chamado às vezes de roteamento de sessão, porque uma rota permanece em vigor para uma sessão inteira (por exemplo, uma sessão de registro em um terminal ou uma transferência de arquivo).

Independente do fato de as rotas serem escolhidas separadamente para cada pacote ou apenas quando novas conexões são estabelecidas, há certas propriedades que são desejáveis em um algoritmo de roteamento: correção, simplicidade, resistência, estabilidade, equanimidade e favorabilidade. A correção e a simplicidade dificilmente precisam de comentários, mas a necessidade de resistência pode ser menos óbvia a princípio. Uma vez que uma rede importante entra no ar, ela pode ter a perspectiva de funcionar continuamente durante anos sem falhas totais do

sistema. Durante esse período haverá falhas de hardware e software de todos os tipos. Hosts, IMPs e linhas irão parar e voltar a funcionar repetidamente, e a topologia irá se modificar muitas vezes. O algoritmo de roteamento tem de estar apto a conviver com essas mudanças na topologia e no tráfego sem exigir que todos os jobs em todos os hosts sejam abortados e a rede seja reinicializada toda vez que algum IMP se quebra.

A estabilidade também é um objetivo importante para o algoritmo de roteamento. Existem algoritmos de roteamento que nunca convergem para o equilíbrio, não importa por quanto tempo funcionem. A equanimidade e a favorabilidade podem soar como características óbvias-seguramente ninguém se oporia a elas, mas, da forma como se apresentam, elas são em geral objetivos contraditórios. Como um exemplo simples desse conflito, veja a Figura 5-8. Suponha que exista bastante tráfego entre A e A', entre B e B' e entre C e C' para saturar os enlaces horizontais. A fim de maximizar o fluxo total, o tráfego de X para X' seria completamente interrompido. Infelizmente, X e X' podem não ver isso dessa maneira. Evidentemente, é necessário algum compromisso entre a eficiência global e a equanimidade para conexões individuais.

Antes que possamos tentar até mesmo encontrar compromissos entre equanimidade e favorabilidade, devemos decidir o que buscamos otimizar. A redução do retardo médio de um pacote é uma candidata evidente, mas isso iria tornar máximo o throughput total da rede. Além disso, esses dois objetivos também são conflitantes, visto que a operação de qualquer sistema de enfileiramento próximo a sua capacidade máxima ($p \rightarrow 1$) implica um longo retardo no enfileiramento. Como um compromisso, muitas redes tentam minimizar o número de saltos que um pacote tem de fazer, porque a redução do número de saltos tende a melhorar o retardo e ainda reduzir a quantidade de banda passante consumida, o que tende a melhorar também o throughput.

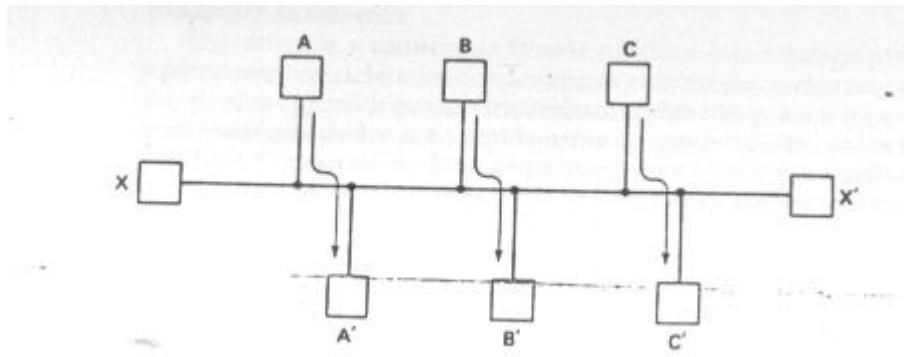


Figura 10-8 Conflito entre equanimidade e favorabilidade

Os algoritmos de roteamento podem ser agrupados em duas classes principais: os não-adaptativos e os adaptativos. Os algoritmos não-adaptativos, ao contrário dos adaptativos, não baseiam suas decisões de roteamento em medidas ou estimativas do tráfego e da topologia atuais. Se um algoritmo adaptativo gerencia a boa adaptação ao tráfego, ele irá naturalmente realizar a saída de um algoritmo que ignore o que acontece na rede, mas se adaptar bem ao tráfego é mais fácil de dizer do que fazer. Os algoritmos adaptativos podem ainda ser subdivididos em centralizados, isolados e distribuídos (McQuillan, 1974); estes serão discutidos em detalhes a seguir.

10.1.4 Congestionamento

Quando estão presentes pacotes em excesso em uma (parte de) sub-rede, o desempenho se degrada. Essa situação é chamada de congestionamento. A Figura 5-9 retrata o sintoma. Quando o número de pacotes descarregados na sub-rede pelos hosts está dentro da sua capacidade de transporte, todos são entregues (com exceção de uns poucos que são afetados por erros de transmissão), e o número entregue é proporcional ao número enviado. Entretanto, conforme o tráfego aumenta muito, os IMPs não são mais capazes de cuidar dele, e começam a perder pacotes. Isso tende a tornar as coisas piores. No caso de tráfego muito intenso, o desempenho entra em colapso total e praticamente nenhum pacote é entregue.

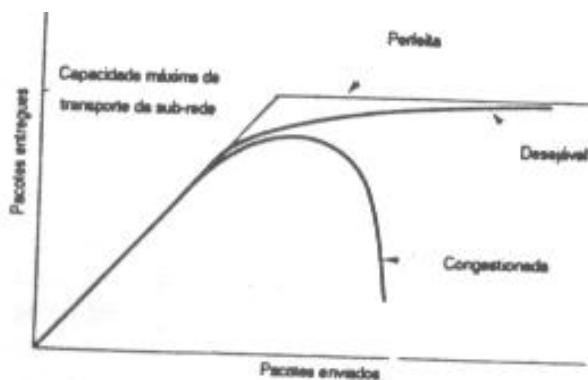


Figura 10-9 Quando é oferecido tráfego em demasia, vem o congestionamento, e o desempenho se degrada nitidamente.

O congestionamento pode ser provocado por diversos fatores. Se os IMPs são excessivamente lentos para realizar as várias tarefas de contabilidade que são exigidas deles (enfileirar buffers, atualizar tabelas, etc.), as filas podem crescer, ainda que exista um excesso de capacidade de linha. Por outro lado, mesmo que a CPU do IMP seja infinitamente rápida, as filas crescerão sempre que a velocidade do tráfego de entrada exceder a capacidade das linhas de saída. Isso pode acontecer, por exemplo, se três linhas de entrada estão entregando pacotes na sua velocidade máxima, e todos precisam ser direcionados ao longo da mesma linha de saída. Dessa forma, o problema se resume a não haver buffers no IMP em número suficiente. Recebendo um suprimento infinito de buffers, o IMP pode sempre reduzir qualquer engarrafamento temporário simplesmente conservando em seu poder todos os pacotes por todo o tempo necessário. É evidente que, para uma operação estável, os hosts não podem bombar indefinidamente pacotes para a sub-rede a uma taxa mais alta do que ela pode absorver.

O congestionamento tende a se nutrir de si mesmo e tornar-se pior. Se um IMP não tiver nenhum buffer livre ele tem de ignorar os pacotes recém-chegados.

Quando um pacote é descartado, o IMP que enviou o pacote se interrompe e o retransmite, talvez muitas vezes, em última análise. Como o IMP que transmite não pode descartar o pacote até que seja confirmado, o congestionamento no extremo do receptor força o transmissor a se abster de liberar um buffer que ele normalmente teria

liberado. Dessa maneira, o congestionamento se propaga, como carros aproximando-se de um posto de pedágio.

Definitivamente, vale a pena destacar explicitamente a diferença entre o controle do congestionamento e o controle de fluxo, o que os autores de muitos livros e trabalhos sobre o assunto não parecem compreender.

O controle do congestionamento tem relação com garantir que a sub-rede seja capaz de transportar o tráfego oferecido. É um tema global, envolvendo o comportamento de todos os hosts, todos os IMPs, o processamento de armazenamento e expedição dentro dos IMPs, e todos os outros fatores que tendem a diminuir a capacidade de transporte da sub-rede.

Em contraste, o controle de fluxo se relaciona com o tráfego ponto a ponto entre um determinado transmissor e um determinado receptor. Seu serviço é garantir que um transmissor rápido não possa transmitir continuamente dados mais rapidamente do que o receptor possa absorvê-los. O controle de fluxo quase sempre envolve algum feedback do receptor para o transmissor, a fim de informar ao transmissor como as coisas estão indo à outra extremidade.

Para ver a diferença entre esses dois concertos, considere uma rede de fibra ótica com uma capacidade de 1000 Gbps, sobre a qual um supercomputador estava tentando transferir um arquivo para um microcomputador a 100 Mbps. Embora não existisse na verdade qualquer congestionamento (a rede propriamente dita não está com problemas), o controle de fluxo seria necessário, a fim de forçar o supercomputador a parar freqüentemente, para dar ao microcomputador uma chance de se recuperar do atraso.

No outro extremo, considere uma rede "armazena e transmite" com linhas de 1 Mbps e 1000 grandes minicomputadores, metade deles tentando transferir arquivos à taxa de 100 Kbps para a outra metade. Aqui o problema não seria o de transmissores rápidos sobrecarregando recipientes lentos, mas simplesmente que o tráfego total oferecido poderia exceder facilmente aquele que a rede poderia manipular.

Mais adiante neste capítulo, examinaremos em detalhes o controle do congestionamento e discutiremos diversos algoritmos para lidar com ele. Deve ficar claro que o controle de congestionamento e o roteamento estão intimamente relacionados, com decisões pobres sobre roteamento, sendo uma causa importante de congestionamento.

10.1.5 Interconexão de Redes

Exatamente como o controle do congestionamento está estreitamente relacionado com a função principal da camada de rede, o roteamento está relacionado com a interconexão de redes. Quando as máquinas de origem e de destino estão em diferentes redes, todos os problemas usuais do roteamento estão presentes, e piores. Por exemplo, se as redes que contêm as máquinas de origem e de destino não estiverem diretamente conectadas, o algoritmo de roteamento tem de encontrar um caminho através de uma ou mais redes intermediárias. Podem existir muitas opções possíveis, todas com diferentes características, vantagens e desvantagens.

Roteamento à parte, outro problema com a interconexão de redes é que nem todas as redes usam os mesmos protocolos. Protocolos diferentes implicam diferentes formatos de pacotes, cabeçalhos, procedimentos de controle de fluxo, regras de

confirmação e outros detalhes. Em consequência disso, quando são deslocados pacotes de uma rede para outra, as conversões são necessárias. Algumas vezes não diretas, mas geralmente todas são. Simplesmente pense sobre o que acontece quando um pacote tem de atravessar sub-redes de circuitos virtuais e datagramas no seu caminho até o destino. Mais adiante neste capítulo, estudaremos em detalhes a interconexão de redes, seus problemas e suas soluções.

10.2 Algoritmo de Roteamento

Os algoritmos de roteamento podem ser agrupados em duas classes principais: os não-adaptativos e os adaptativos. Os algoritmos não-adaptativos não baseiam suas decisões sobre roteamento em medidas ou estimativas do tráfego e da topologia correntes. Em vez disso, a escolha da rota a utilizar para i de i para j (para todo i e todo J) é computada com antecedência, separadamente, e transferida para os IMPs quando a rede é inicializada. Esse procedimento é chamado algumas vezes de roteamento estático.

Os algoritmos adaptativos, por outro lado, tentam alterar suas decisões sobre o roteamento de modo a refletirem mudanças na topologia e no tráfego correntes. Existem três famílias diferentes de algoritmos adaptativos, que se distinguem pelo tipo de informações que utilizam. Os algoritmos globais usam informações coletadas de toda a sub-rede, em uma tentativa de tomar decisões ótimas. Essa abordagem é chamada de roteamento centralizado.

Os algoritmos locais rodam separadamente em cada IMP e só utilizam as informações ali disponíveis, tais como comprimentos de filas. Esses algoritmos são conhecidos como algoritmos isolados. Finalmente, a terceira classe de algoritmos utiliza uma mistura de informações globais e locais. Esses são os chamados algoritmos distribuídos. Todas as três classes serão discutidas detalhadamente a seguir. Informações adicionais sobre o roteamento podem ser encontradas em Bell e Jabbour (1986).

10.2.1. Roteamento pelo Caminho Mais Curto

Vamos iniciar nosso estudo dos algoritmos de roteamento com uma técnica amplamente utilizada em várias formas, porque é simples e fácil de entender. A idéia é construir um grafo da sub-rede, com cada nó do grafo representando um IMP e cada arco representando uma linha de comunicação. Para escolher uma rota entre um determinado par de IMPs, o algoritmo simplesmente encontra o caminho mais curto entre eles.

O conceito de caminho mais curto merece algumas explicações. Uma forma de medir o comprimento de um caminho é o número de saltos. Usando essa unidade de medida, os caminhos ABC e ABE na Figura 5.10 são igualmente longos. Outra unidade de medida é a distância geográfica em quilômetros, em cujo caso ABC é claramente muito mais longo do que ABE (considerando que a figura esteja desenhada em escala).

Contudo, muitas outras unidades de medida também são possíveis. Por exemplo, cada arco poderia ser rotulado com o enfileiramento médio e o retardo de

transmissão para um pacote de teste padrão conforme determinado por execuções de testes horários ou diários. Com essa rotulação do grafo, o caminho mais curto é o caminho mais rápido, em vez de ser o caminho com menos arcos ou quilômetros.

No caso mais geral, os rótulos nos arcos poderiam ser computados em função da distância, da banda passante, do tráfego médio, do custo da comunicação, do comprimento médio da fila, do retardo medido e de outros fatores. Pela modificação da função ponderada, o algoritmo deveria computar então o caminho "mais curto" de acordo com qualquer um de uma série de critérios.

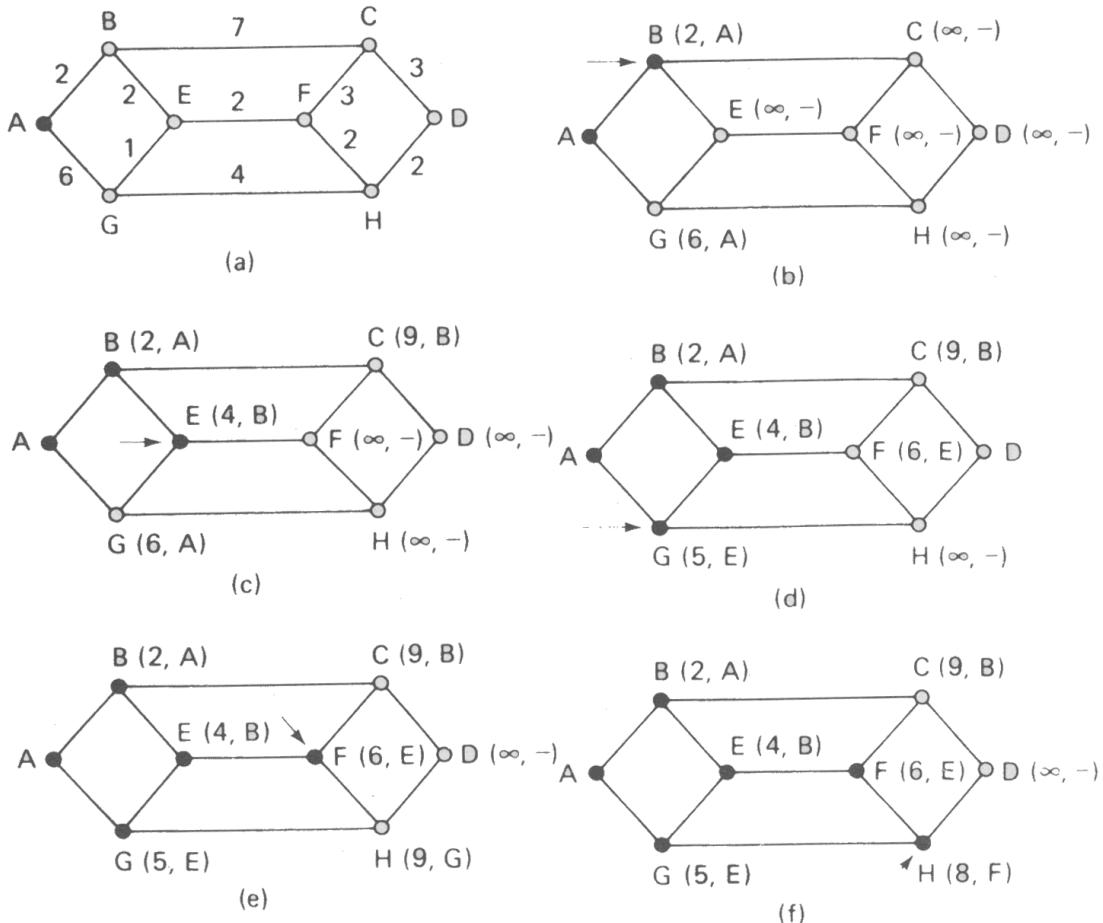


Figura 10.10. As cinco primeiras etapas usadas na computação do caminho mais curto de A para D. As setas indicam o nó de trabalho.

São conhecidos diversos algoritmos para a computação do caminho mais curto entre dois nós de um grafo. O algoritmo mostrado é de Dijkstra (1959). Cada nó é rotulado (dentro de parênteses) com sua distância a partir do nó de origem ao longo do melhor caminho conhecido. Inicialmente, nenhum caminho é conhecido e, portanto, todos os nós estão rotulados com infinito. Conforme o algoritmo continua e os caminhos são encontrados, os rótulos podem se alterar, refletindo os melhores caminhos. Um rótulo pode ser experimental ou permanente. Inicialmente, todos os rótulos são experimentais. Quando se descobre que um rótulo representa o caminho mais curto possível desde a origem até esse nó, ele é transformado em permanente; nunca mais será alterado.

Para ilustrar como funciona o algoritmo de rotulação, observe o grafo ponderado não-direcionado da Figura 5.10(a), onde os valores representam, por exemplo, a distância. Queremos descobrir o caminho mais curto de A para D. Vamos começar assinalando o nó A como permanente, indicando-o com um círculo

preenchido. Depois examinamos, um de cada vez, os nós adjacentes ao nó A (o nó de trabalho), rotulando novamente cada nó com a distância até A. Sempre que um nó é rotulado outra vez, também o rotulamos com o nó a partir do qual o teste foi feito, de modo que possamos reconstruir mais tarde o caminho final. Tendo examinado cada um dos nós adjacentes ao nó A, examinamos todos os nós rotulados experimentalmente em todo o grafo, e transformamos aquele que tiver o menor rótulo como permanente, como mostra a Figura 5.10(b). Esse nó passa a ser o novo nó de trabalho.

Agora partimos de B e examinamos todos os nós adjacentes a ele. Se a soma do rótulo sobre B com a distância de B até o nó que está sendo considerado for menor que o rótulo nesse nó, temos um caminho mais curto e, portanto, o nó é novamente rotulado.

Depois que todos os nós adjacentes ao nó de trabalho forem inspecionados e os rótulos experimentais, se possível, alterados, todo o grafo é pesquisado em busca do nó rotulado experimentalmente com o menor valor. Esse nó é transformado em permanente, e passa a ser o nó de trabalho para a etapa seguinte. A Figura 5.10 mostra os cinco primeiros passos do algoritmo.

Para ver por que o algoritmo funciona, observe a Figura 5.10(c). Nesse ponto, acabamos de tornar E permanente. Suponha que houvesse um caminho mais curto do que ABE, digamos AXYZE. Existem duas possibilidades: ou o nó Z já foi transformado em permanente, ou ainda não. Se ele já foi transformado, então E já foi testado (na etapa seguinte àquela em que Z foi transformado em permanente), de modo que AXYZE não teria escapado à nossa atenção.

Agora, considere o caso em que Z ainda está rotulado experimentalmente. Ou o rótulo em Z é maior ou igual ao que está em E, e nesse caso AXYZE não pode ser um caminho mais curto do que ABE, ou ele é menor do que o de E, e então Z, e não E, irá se tornar permanente primeiro, permitindo que E seja testado a partir de Z.

Esse algoritmo é dado em Pascal na Figura 5.12. A única diferença entre o programa e o algoritmo descrito anteriormente é que, na Figura 5.12, computamos o menor caminho a partir do nó terminal, t, em vez de a partir do nó de origem, s. Como o caminho mais curto de t para s em um grafo não-direcionado é igual ao caminho mais curto de s para t, não importa em qual extremidade iniciamos (a menos que existam vários caminhos mais curtos, em cujo caso a inversão da pesquisa poderia descobrir um caminho diferente). O motivo para pesquisar em sentido inverso é que cada nó está rotulado com seu predecessor em vez de com seu sucessor. Quando se copia o caminho final na variável de saída, caminho, o caminho é invertido. Pela inversão da pesquisa, os dois efeitos se cancelam, e a resposta é produzida na ordem correta.

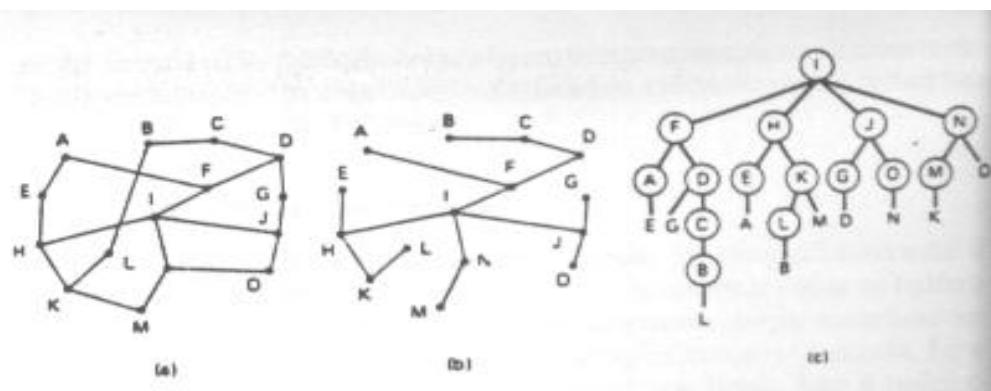


Figura 11-11 Progressão pelo caminho inverso

A principal vantagem da progressão pelo caminho inverso é que ela é razoavelmente eficiente e também fácil de implementar. Não requer que os IMPs saibam sobre árvores geradoras, nem tem o overhead de uma lista de destinos ou mapa de bits em cada pacote de difusão como ocorre na abordagem de múltiplos destinos. Nem exige qualquer mecanismo especial a fim de parar o processo, como ocorre com a enchente (ou um contador de saltos em cada pacote e um conhecimento a priori do diâmetro da sub-rede, ou uma lista de pacotes já vistos por origem).

```

{Encontra o caminho mais curto da origem até o final de um determinado grafo.}

const n = ...;           {numero de nos}
    infinito = ...;       {um numero maior que qualquer comprimento de caminho possivel}

type no = 0 .. n;
    listadenos = array [1 .. n] of no;
    matriz = array [1 .. n] of integer;

procedure Caminho Curto (a: matriz; s,t: no; var caminho: listadenos);
{Encontra o caminho mais curto de s para t na matriz a e o retorna em caminho.}

type lab = (perm, tent);      {label experimental ou permanente?}
    LabelNo = record predecessor: no; comprimento: integer; labl: lab end;
    EstadoGrafo = array [1 .. n] of LabelNo;

var estado: EstadoGrafo; i, k: no; mim: integer;
begin
    {inicializar}
    for i := 1 to n do
        with estado [i] do
            begin predecessor := 0; comprimento := infinito; labl := tent end;
    estado [t].comprimento := 0; estado [t].labl := perm;
    k := t;                      {k e o no inicial em funcionamento}

repeat   {existe um caminho melhor a partir de k?}
    for i := 1 to n do
        if (a [k,i] < 0) and (estado [i].labl = tent) then      {i e adjacente a tent.}
            if estado [k].comprimento + a [k,i] < estado [i].comprimento then
                begin
                    estado [i].predecessor := k;
                    estado [i].comprimento := estado [k].comprimento + a [k,i]
                end;
    {Encontra o no rotulado como tentativa com o menor label.}
    mim := infinito; k := 0;
    for i := 1 to n do
        if (estado [i].labl = exp) and (estado [i].comprimento < mim) then
            begin
                mim := estado [i].comprimento;
                k := i                     {k sera o proximo no de trabalho, a menos que seja substituido}
            end;
    estado [k].labl := perm
until k = s;                  {repetir ate encontrarmos a origem}
{Copia o caminho no array de saida.}
k := s; i := 0;
repeat
    i := i + 1;
    caminho [i] := k;
    k := estado [k].predecessor;
until k = 0
end; {CaminhoMaisCurto}

```

Figura 10.11 Um procedimento para computar caminho mais curto através de um grafo

10.2.2. Roteamento de Caminhos Múltiplos

Até agora, estamos pressupondo que existe um único "melhor" caminho entre qualquer par de nós e que todo o tráfego entre eles usa esse caminho. Em muitas redes, existem diversos caminhos entre pares de nós que são quase igualmente bons. Freqüentemente, pode ser obtido melhor desempenho pela divisão do tráfego sobre diversos caminhos, a fim de reduzir a carga em cada uma das linhas de comunicação. A técnica de utilizar múltiplas rotas entre um único par de nós é chamada roteamento de caminhos múltiplos ou algumas vezes roteamento bifurcado.

O roteamento de caminhos múltiplos é aplicado tanto a sub-redes de datagramas quanto a sub-redes de circuitos virtuais. Para sub-redes de datagrama, quando um pacote chega a um IMP para seguir em frente, é feita uma escolha entre as diversas alternativas para esse pacote, independente das escolhas feitas para outros pacotes para o mesmo destino no passado. Para sub-redes de circuitos virtuais, sempre que um circuito virtual é definido, uma rota é definida, mas diferentes circuitos virtuais (da parte de diferentes usuários) são roteados independentemente.

O roteamento de caminhos múltiplos pode ser implementado como a seguir. Cada IMP mantém uma tabela com uma linha para cada IMP de destino possível. Uma linha determina a melhor, a segunda melhor, a terceira melhor, etc, linha de saída para aquele destino, juntamente com um peso relativo. Antes de direcionar um pacote, um IMP gera um número aleatório e depois escolhe entre as alternativas, usando os pesos como probabilidades. As tabelas são elaboradas manualmente pelos operadores da rede, carregadas nos IMPs antes de a rede ser estabelecida e não mais modificadas depois disso.

Como um exemplo, considere a sub-rede da Figura 5.12(a). A tabela a de roteamento do IMP J é dada na Figura 5.12(b). Se J recebe um pacote cujo destino é A, ele utiliza a linha rotulada A. Aqui, três opções são apresentadas. A linha para A é a primeira opção, seguida pelas linhas para I e H, respectivamente. Para se decidir, J gera um número aleatório entre 0,00 e 0,99. Se o número estiver abaixo de 0,63, a linha A é usada; se ele estiver entre 0,63 e 0,83, I é usada; em caso contrário, H é usada. Portanto, os três pesos são as probabilidades respectivas para A, I ou H serem usadas.

10.3 ALGORITMOS PARA CONTROLE DO CONGESTIONAMENTO

Nos próximos subitens serão averiguadas as cinco estratégias para o controle do congestionamento.

10.3.1 Pré-alocação de Buffers

Quando um circuito virtual é definido, o pacote com o pedido de chamada, segue o seu caminho através da sub-rede, realizando entradas em tabelas, conforme avança. Quando ele chega ao destino, a rota a ser seguida por todo o tráfego de dados subsequente, fica determinado, e as entradas são feitas nas tabelas de roteamento para todos os IMPs intermediários.

Normalmente, o pacote que solicita a chamada não reserva qualquer espaço em buffer nos IMPs intermediários, apenas entradas na tabela. Entretanto, uma modificação simples no algoritmo de configuração poderia fazer cada pacote com pedido de chamada reservar também um ou mais buffers de dados. Se um pacote com pedido de chamada chegar a um IMP e todos os buffers estiverem reservados, outra rota deve ser procurada ou então um "sinal de ocupado" deve ser retornado ao chamador. Mesmo se os buffers não estiverem reservados, alguns circuitos virtuais podem ter de ser novamente roteados ou rejeitados por falta de espaço na tabela, e assim a reserva de buffers não acrescentaria nenhum problema que já não estivesse lá.

Pela alocação permanente de buffers para cada circuito virtual em cada IMP, sempre existe um lugar para armazenar qualquer pacote que chega, até que possa ser encaminhado. Primeiro, considere o caso de um protocolo IMP-IMP de pára-espera. Um buffer por circuito virtual por IMP é suficiente para circuitos simplex, e um para cada sentido é suficiente para circuitos-full-duplex. Quando chega um pacote, a confirmação não é enviada de volta ao IMP transmissor até que o pacote tenha sido encaminhado. Efetivamente, uma confirmação significa que o receptor ainda não recebeu o pacote corretamente, mas também que ele possui um buffer livre e está inclinado a aceitar outro pacote. Se o protocolo IMP-IMP permitir muitos pacotes pendentes, cada IMP terá de dedicar o equivalente em buffers de toda uma janela a cada circuito virtual, para eliminar completamente a possibilidade de congestionamento.

Quando cada circuito virtual que passa através de um IMP possui uma quantidade suficiente de espaço em buffer dedicado a ele, a comutação de pacotes se torna bastante similar a comutação de circuitos. Em ambos os casos é obrigatório um procedimento envolvendo a configuração. E são permanentemente alocados recursos para conexões específicas, existindo ou não algum tráfego. Em ambos os casos o congestionamento é impossível, porque todos os recursos necessários para processar o tráfego já estavam reservados. Nos dois casos há um uso potencialmente ineficiente dos recursos, porque os recursos que não estão sendo usados pela conexão para a qual são alocados estão indisponíveis para qualquer outra.

Por ser caro dedicar um conjunto completo de buffers a um circuito virtual ocioso, algumas sub-redes podem usá-lo somente onde são essenciais baixo retardo e grande banda passante como, por exemplo, em circuitos virtuais que transportam voz digitalizada. Para circuitos virtuais em que o baixo retardo não é absolutamente essencial o tempo todo, uma estratégia razoável é associar um timer a cada buffer. Se o buffer permanecer ocioso por tempo excessivo, ele é liberado, para ser readquirido quando chegar o próximo pacote. Evidentemente, a aquisição de um buffer poderia levar algum tempo e, assim, os pacotes terão de ser encaminhados sem buffers dedicados, até que a cadeia de buffers possa novamente ser definida.

10.3.2 Descarte de Pacotes

Nosso segundo mecanismo de controle do congestionamento é exatamente o oposto do primeiro. Em vez de reservar todos os buffers antecipadamente, nada é reservado. Ou seja:

Se um pacote chegar e não houver nenhum lugar onde colocá-lo, o IMP simplesmente o descarta.

Se a sub-rede oferecer o serviço de datagrama aos hosts, isso é tudo: o congestionamento é resolvido descartando-se pacotes à vontade.

Se a sub-rede oferece o serviço de circuitos virtuais, uma cópia do pacote deve ser mantida em algum lugar, de modo que ele possa ser retransmitido mais tarde.

Uma possibilidade é que o IMP que envia o pacote descartado continue se interrompendo e retransmitido o pacote até que ele seja recebido. Outra possibilidade é a de que o IMP de transmissão desista após um certo número de tentativas e solicite ao IMP de origem para se interromper e começar tudo de novo.

O descarte de pacotes à vontade pode ser levado longe demais. É evidentemente uma estupidez extrema ignorar um pacote que chega contendo uma confirmação de um IMP vizinho. Essa confirmação permitiria ao IMP abandonar um pacote recebido nesse momento e portanto liberar um buffer. Entretanto, se o IMP não tem buffers de reserva, ele não pode adquirir qualquer outro pacote que chega para ver se ele contém uma confirmação. A solução é reservar permanentemente um buffer por linha de entrada, a fim de permitir a todos os pacotes que chegam serem inspecionados. É bastante lógico para um IMP examinar um pacote recém-chegado, fazer uso de qualquer confirmação transportada por ele e depois descartar de alguma maneira o pacote. Alternativamente, o portador de boas notícias poderia ser recompensado por mantê-las usando o buffer recentemente liberado como novo buffer de entrada.

Se o congestionamento vai ser evitado, é necessária uma regra para informar quando manter um pacote quando descartá-lo. Irland (1978) estudou esse problema e chegou a uma heurística simples, ainda que eficiente, para descartar pacotes. Na ausência de qualquer regra explícita em contrário, uma única linha de saída poderia tomar para si todos os buffers disponíveis em um IMP, visto que são simplesmente atribuídos ao primeiro que chegar.

A Figura 5-22(a) mostra um IMP com um total de 10 buffers. Três desses buffers estão permanentemente atribuídos às linhas de entrada. Os sete restantes contêm pacotes enfileirados para transmissão em uma das linhas de saída.

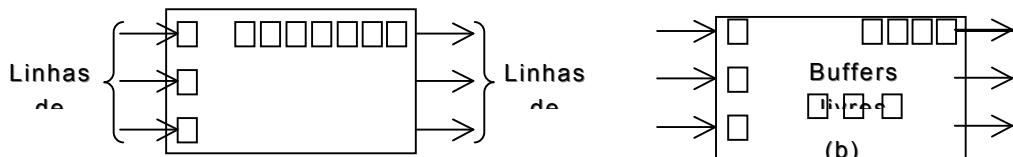


Figura 10-22 O congestionamento pode ser reduzido estabelecendo-se um limite superior para o número de buffers enfileirados em uma linha de saída

Muito embora duas linhas de saída estejam ociosas, quaisquer pacotes que cheguem destinados a essas linhas devem ser descartados porque não existem buffers de reserva. Isso é obviamente um desperdício. A idéia de Irland é limitar o número de buffers que podem estar associados a uma fila de saída qualquer. Por exemplo, se o limite fosse fixado em quatro, a situação da Figura 5-22(b) iria prevalecer: três buffers atribuídos. Um pacote recentemente chegado que quisesse continuar sobre a primeira linha de saída seria descartado, em vez de ter permissão para aumentar o comprimento da fila para cinco.

Essa estratégia não é realmente tão drástica quanto aparenta. Afinal de contas, essa linha de saída já está funcionando em sua capacidade máxima. Ter sete pacotes enfileirados em vez de quatro não irá empurrar os bits para fora nem um pouco mais rápido, mas permitirá que o tráfego para as outras linhas seja encaminhado imediatamente, possivelmente duplicando ou triplicando a velocidade de saída do IMP. Em qualquer caso, o pacote descartado será retransmitido em pouco tempo. Se

estiver bem ajustado, o sistema ainda será retransmitido antes que a fila se esvazie, e assim a sua rejeição inicial nem mesmo será notada.

Irland estudou diversos algoritmos diferentes para determinar o comprimento Máximo da fila, m , para um IMP com k buffers (os buffers dedicados de modo permanente a entrada não contam). O caso sem controle é $m = k$. Se existirem s linhas de saída, o caso $m = k/s$ significa efetivamente que cada buffer está dedicado a uma determinada linha de saída. Nenhuma linha pode tomar emprestado nem mesmo um buffer de uma linha ociosa. Intuitivamente isso não é eficiente, e o estudo admite essa saída.

O valor ótimo de m vem a ser uma função complicada do tráfego médio. Embora o IMP pudesse tentar medir seu tráfego e ajustar continuamente m , se o tráfego viesse em rajadas, essa probabilidade não funcionaria bem. Entretanto, Irland descobriu um método experimental simples que geralmente dá um desempenho bom, embora não ótimo: $m=k/s$. Por exemplo, para sete buffers em pool e três linhas, $m=7/3$, e assim 4 buffers seriam alocados.

Uma idéia afim, devida a Kamoun (1976), evita diretamente que qualquer linha ou linhas fiquem com fome: um número mínimo de buffers é dedicado a cada linha. Se não existir nenhum tráfego, os buffers vazios são reservados. O método de Irland pode ser combinado com o de Kamoun para se ter um número mínimo e um máximo de buffers para cada linha. A ARPANET utiliza esse método.

Embora seja fácil, o descarte de pacotes tem algumas desvantagens. A principal, é a banda passante extra necessária para as duplicatas. Se, a probabilidade de um pacote ser descartado é p , o número esperado de transmissões antes que ele seja aceito é $1/(1-p)$. Um tempo relacionado com esse é a duração que poderia ter o intervalo de temporização. Se for muito curto, as duplicatas serão geradas quando não forem necessárias, tornando pior o congestionamento. Se for longo demais, o tempo de retardo irá aumentar.

Uma forma de minimizar a quantidade de banda passante desperdiçada na retransmissão de pacotes descartados é descartar sistematicamente pacotes que ainda não tenham viajado para longe e, portanto não representem um grande investimento em recursos. O caso limite dessa estratégia é o descarte de pacotes recém-chegados de hosts em detrimento do descarte do tráfego em trânsito. Por exemplo, os IMPs poderiam recusar ou descartar novos pacotes de hosts associados sempre que o número de buffers ocupados por novos pacotes (ou pelo total de pacotes) exceder algum limite.

10.3.3 Controle Isarrítmico do Congestionamento

O congestionamento ocorre quando existem pacotes em excesso na sub-rede. Uma abordagem direta para controlá-lo é limitar o número de pacotes na sub-rede. Davies (1972) propôs um método que reforça precisamente essa limitação.

Nesse método, chamado isarrítmico porque mantém constante o número de pacotes, existem permissões, que circulam dentro da subrede. Sempre que um IMP deseja enviar um pacote entregue a ele pelo seu host, deve primeiro capturar uma permissão e destruí-la. Quando o IMP de destino finalmente remove o pacote da sub-rede, ele

regenera a permissão. Essas regras simples asseguram que o número de pacotes na sub-rede nunca irá ultrapassar o número de permissões inicialmente presentes.

Entretanto, esse método tem alguns problemas. Primeiro, embora garanta que a sub-rede, como um todo, nunca ficará congestionada, não garante que um determinado IMP não será repentinamente entulhado com pacotes.

Em segundo lugar, a forma de distribuir as permissões está longe de ser óbvia. Para evitar que um pacote recentemente gerado venha a sofrer um longo retardo enquanto o IMP local tenta descobrir uma permissão, as permissões devem ser distribuídas de maneira uniforme, para que cada IMP tenha algumas. Por outro lado, a fim de permitir a transferência de arquivos em grande banda passante, é indesejável para o IMP transmissor ter de sair caçando por todas os lados até encontrar permissões em número suficiente. Seria bem melhor se todas estivessem centralizadas, de modo que os pedidos de quantidades substanciais pudessem ser honrados rapidamente. Deve ser encontrado um ponto de compromisso, tal como um número máximo de permissões que pudessem estar presentes em qualquer IMP, com as permissões em excesso sendo obrigadas a procurar um IMP com espaço para elas. Observe que o passeio aleatório das permissões em excesso, por si mesmo, já introduz uma carga na sub-rede.

Em terceiro lugar, mas não menos importante, se as permissões jamais são destruídas por qualquer motivo (por exemplo erro de transmissão, mau funcionamento de IMPs, o descarte por um IMP congestionado), a capacidade de transporte de rede será reduzida para sempre. Não há nenhum modo fiel de descobrir quantas permissões ainda existem enquanto a rede estiver funcionando.

10.3.4 Controle de Fluxo

Algumas redes (notadamente a ARPANET) tentaram usar mecanismos de controle de fluxo para eliminar o congestionamento. Embora os esquemas de controle de fluxo possam ser utilizados pela camada de transporte para evitar que um host sature outro, e embora os esquemas de controle de fluxos possam ser usados para evitar que um IMP sature seus vizinhos, é difícil controlar a quantidade total de tráfego na rede usando regras de controle de fluxo fim a fim. Mais ainda, se os hosts são forçados a parar de transmitir devido a regras estritas de controle de fluxo, a sub-rede não será carregada pesadamente.

O controle de fluxo não pode realmente solucionar problemas de congestionamento por um bom motivo: o tráfego no computador é em rajadas. Na maior parte do tempo um usuário interativo ocupa seu terminal e fica diante dele coçando a cabeça, mas de vez em quando pode desejar examinar um arquivo extenso. O pico de tráfego potencial é imensamente mais alto do que a taxa média. Qualquer esquema de controle de fluxo que esteja ajustado de modo a limitar cada usuário à taxa média fornecerá um serviço ruim quando o usuário quiser uma rajada de tráfego. Por outro lado, se o limite do controle de fluxo estiver ajustado em um valor alto o suficiente para permitir que o pico de tráfego seja alcançado, ele tem pouco valor como controle de congestionamento quando diversos usuários demandarem o pico de uma só vez. (Se metade das pessoas no mundo repentinamente pegasse seu telefone para chamar a outra metade, haveria uma grande quantidade de sinais de ocupado; o sistema telefônico também está projetado para o tráfego médio, e não para o pior caso.).

Quando usado em uma tentativa de dominar o congestionamento, o controle de fluxo pode se aplicar ao tráfego entre pares de:

Processos do usuário (p.ex, uma mensagem pendente por circuito virtual).

Processos do usuário (p.ex, uma mensagem pendente por sircito virtual).

Hosts, sem levar em consideração o número de circuitos virtuais abertos.

IMPS de origem e de destino, sem se importar com os hosts.

Além disso, o número de circuitos virtuais abertos pode ser limitado.

10.3.5 Pacotes Limitadores

Embora a limitação do volume de tráfego entre cada par de IMPs ou hosts possa indiretamente aliviar o congestionamento, ela o faz ao preço de uma redução potencial no throughput, mesmo quando não existe nenhum prenúncio de congestionamento. O que é realmente necessário é um mecanismo que seja disparado somente quando o sistema está congestionado.

Uma forma é fazer com que cada IMP monitore a utilização percentual de cada uma das suas linhas de saída. Associado a cada linha está uma variável real, u , cujo valor, entre 0,0 e 1,0, reflete a utilização recente dessa linha. Para manter uma boa estimativa de u , uma amostra da utilização instantânea da linha, f (0 ou 1), pode ser feita periodicamente, atualizando-se u de acordo com

$$u_{\text{novo}} = u_{\text{velho}} + (1 - a)f$$

onde a constante a determina a velocidade com que o IMP esquece seu histórico recente.

Sempre que u se desloca acima do limiar, a linha de saída entra em estado de "atenção". Cada pacote recém-chegado é verificado para se saber se a sua Linha de saída está em estado de atenção. Se estiver, o IMP envia um pacote limitador de volta ao host de origem, dando-lhe o destino encontrado no pacote. O próprio pacote é marcado (um bit de cabeçalho é ativado) de forma que não venha a gerar mais tarde qualquer outro pacote limitador, e é encaminhado da forma usual.

Quando recebe o pacote limitador, o host de origem é obrigado a reduzir o tráfego enviado ao destino especificado em X%. Como outros pacotes visando o mesmo destino já estão provavelmente a caminho e irão gerar ainda mais pacotes-limitadores, o host deve ignorar os pacotes limitadores que foram referentes a esse destino por um intervalo de tempo fixado. Depois que esse período tiver expirado, o host fica à escuta de mais pacotes limitadores durante um outro intervalo. Se chegar algum, a linha ainda está congestionada, e assim o host reduz ainda mais o fluxo e começa novamente a ignorar os pacotes limitadores. Se nenhum pacote limitador chegar durante o período de escuta, o host pode aumentar de novo o fluxo. O feedback implícito nesse protocolo deve evitar o congestionamento, ainda que não controle qualquer fluxo, a menos que ocorram problemas.

Foram propostas algumas variações sobre esse algoritmo de controle de congestionamento. Para uma deles, os IMPs poderiam manter dois níveis críticos. Acima do primeiro nível, os pacotes limitadores são enviados de volta. Acima do segundo, o tráfego que chega é simplesmente descartado, seguindo a teoria de que o host provavelmente já foi avisado. Sem tabelas extensivas, é difícil para o IMP saber quais hosts foram avisados recentemente sobre quais destinos, e deles não o foram.

Outra variação é utilizar os comprimentos das filas e não a utilização da linha como sinal de disparo. A mesma ponderação exponencial pode ser usada tanto com essa unidade de medida quanto com u, evidentemente. Ainda outra possibilidade é fazer com que os IMPs propaguem as informações sobre o congestionamento juntamente com informações sobre o roteamento, de modo que o disparo não esteja baseado apenas nas observações dos IMPs, mas no fato de que em algum lugar ao longo do caminho existe um engarrafamento. Pela propagação das informações sobre o congestionamento pela sub-rede, os pacotes limitadores podem ser enviados mais cedo, antes que muitos pacotes estejam no caminho, evitando assim que o congestionamento aumente.

10.3.6 Impasses

O máximo em congestionamento é um impasse (deadlock), também chamado de travamento. O primeiro IMP não pode prosseguir até que o segundo IMP faça alguma coisa, e o segundo IMP não pode continuar porque está esperando que o primeiro IMP faça algo. Ambos os IMPs chegaram a um estado de suspensão completa e permanecerão nele para sempre. Os impasses não são considerados uma propriedade desejável em sua rede.

O travamento mais simples pode acontecer com dois IMPs.

Capítulo

11

Camada de Transporte

Os programas da camada de transporte rodam somente nos hosts e não nos IMPs, em contraste com as três camadas inferiores que estão presentes em ambos.

Objetivos da Camada de Transporte

- O objetivo primário da camada de transporte é propiciar o transporte de dados entre processos de usuários que rodam em sistemas interconectados;
- O transporte de dados deve ser confiável e eficiente. Pois esta é a camada mais alta com esta responsabilidade, liberando as camadas superiores desta função;
- A camada de transporte deve melhorar a qualidade dos serviços de rede, a fim de atender às necessidades e os requisitos da camada de sessão. Para isto, a camada de transporte deve ter funções de estabelecimento de conexão, endereçamento, seqüencialização, recuperação de erros e falhas, multiplexação, controle de fluxo, gerência de “buffer” e sincronização.

Serviços da Camada de Transporte

A implementação de um protocolo de transição é denominado ET (Estação de Transporte). Cada sistema hospedeiro da rede para se comunicar com outro deve ter uma ET, por exemplo, como um processo independente ao qual o usuário tem acesso via “primitivas de comunicação”.

A interação entre a ET e os demais processos, seus usuários, se dá através da interface com processo-usuário (figura 4.2), Pontos de Acesso a Serviço de Transporte (TSAP), e é efetuada pela “requisição de serviços de transporte” pelos usuários e a consequente “prestação de serviços de transporte” pela ET.

PRIMITIVAS DE SERVIÇO

*Serão ilustrados os serviços típicos da camada de transporte examinando as primitivas de serviço da proposta de padronização do protocolo de transporte feita pela ISO em Janeiro de 1984.

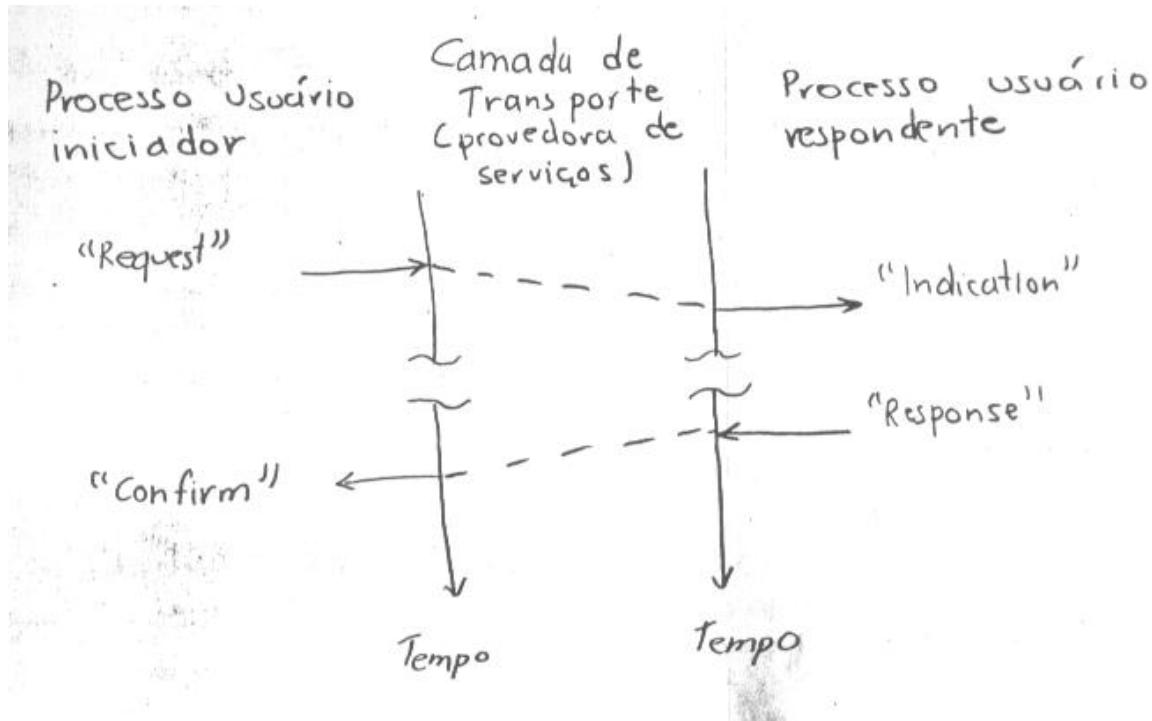
*As interações entre entidades implementadas nas várias Ets são realizadas através da troca de Unidades de Dados do Protocolo de Transporte (TPDUs) As várias TPDUs utilizadas recebem nomes relacionados aos que prestam, exemplos:

Pedido de Conexão (Connection Request TPDU)	CR-TPDU
Confirma Conexão (Connection Confirm TPDU)	CC-TPDU
Pedido de Desconexão (Disconnect Request TPDU)	DR-TPDU
Confirma Desconexão (Disconnect Confirm TPDU)	DC-TPDU
Dados (Data TPDU)	DT-TPDU
Dados Urgentes (Expedited Data TPDU)	ED-TPDU
Reconhecimento de Dados (Data Acknolegde TPDU)	AK-TPDU
Reconhecimento Urgente (Expedited Acknolegde TPDU)	EA-TPDU
Rejeição (Reject TPDU)	RJ-TPDU
Erro (Error TPDU)	ER-TPDU

*Cada TPDU tem associado um número de campos, utilizados para o controle das interações entre as entidades de transporte e o processamento correto das informações, exemplos:

Indicador de Comprimento (Length Indicator)	LI
Crédito	CDT
Identificador de TSAP	TSAP-ID
Referência de Destino	DST-REF
Referência de Fonte (Source Reference)	SRC-REF
Número de DT TPDU	TPDU-NR
Número de ED TPDU	ED-TPDU-NR
Número de Seqüência de Resposta	YR-TU-NR
Número de ED TPDU de Resposta	YR-EDTU-NR

Uma entidade que emite uma CR-TPDU é chamada INICIADORA e a entidade com quem deseja estabelecer conexão é chamada. REPONDENTE. Se uma entidade se envolver com múltiplas conexões simultaneamente ela pode ser iniciadora e respondente.



As interações entre uma ET e um processo usuário de transporte são realizados através da troca de primitivas de serviços de transporte. Uma primitiva tem associada uma lista de parâmetros e, uma vez invocada, desencadeia os eventos de primitivas de serviços. Em resposta a estes eventos, as ET envolvidas emitem TPDUs ou invocam as primitivas adequadas.

Par complementar de primitivas

"REQUEST" -----→ "Indication"

"CONFIRM" ←----- "Response"

Existem dois grupos de primitivas, divididos de acordo com os serviços envolvidos (ver tabela 4.1):

- 1 – Primitivas para estabelecimento e encerramento de conexão.
- 2 – Primitivas para transporte de dados.

Estabelecimento de uma Conexão (Ler e observar figura 4.4.a)

A camada de sessão passa a primitiva T-CREATE REQUEST com parâmetros para a camada de transporte, por exemplo, o endereço de transporte chamado e funções opcionais requisitadas pelo processo iniciador, como por exemplo:

- ★ Tamanho de TPDU.
- ★ Mapeamento de endereços de transporte para endereços de rede.
- ★ Funções a serem empregadas na fase de transferência de dados (qualidade de serviços).

O exame destes parâmetros pela ET iniciadora leva a escolher o serviço de rede mais apropriado, como por exemplo, multiplexação em uma ou mais conexões de rede.

A ET, então, prepara o TPDU que no caso é o CR-TPDU carregando seus campos de acordo com os valores dos parâmetros da Primitiva T-CREATE REQUEST.

A TPDU preparada é passada, então, como uma Unidades de Dados de Serviços de Rede (NSDU), para a camada de rede para ser entregue a ET respondente.

A ET respondente recebe a TPDU e invoca a primitiva T-CREATE INDICATION para informar ao processo usuário respondente o pedido de conexão.

Para estabelecer a conexão o processo usuário respondente passa a primitiva T-CREATE RESPONSE e a ET envia o TPDU CC-TPDU com campos definidos pelos parâmetros da primitiva invocada (ver figura 4.4.a).

Objetivo da Camada de Transporte

O objetivo da camada de transporte é propiciar o transporte de dados entre os processos de usuários que rodam em sistemas interconectados.

Os programas da camada de transporte rodam somente nos **hosts** e não nos equipamentos utilizados para interconexão de redes, como hubs, pontes e roteadores, pois a camada de transporte encontra-se acima das três camadas de interconexão de redes, especificamente acima da camada de rede.

O transporte de dados deve ser confiável e eficiente. A camada é responsável pela qualidade dos serviços de rede, a fim de atender às necessidades e requisitos da camada acima, sessão (OSI) ou aplicação (INTERNET), dependendo da arquitetura usada. Para tal a camada de transporte deve apresentar as seguintes funções:

Estabelecimento de conexão

Endereçamento

Sequencialização

Recuperação de erros e falhas
Multiplexação
Controle de fluxo
Gerência de “buffer”
Sincronização

Serviços da camada de Transporte

A camada de transporte divide-se em (1) **Entidade ou Estação de Transporte (ET)** e (2) **Primitivas de Acesso**.

Entidade de Transporte (ET) - é a implementação do protocolo de transporte.

Primitivas de Acesso – são os comandos utilizados pelas camadas superiores para acessar os serviços oferecidos pela camada de trasporte.

Entidade de Transporte (ET)

Considerações:

- Cada host da rede deve ter sua ET para se comunicar
- O processo de usuário tem acesso as Ets através das primitivas de acesso

SAP – Service Access Points

As operações entre entidades de camadas adjacentes dentro de um mesmo sistema aberto ocorrem nos pontos de acesso de serviço (SAP - Service Access Point), que estão localizados na interface entre duas camadas. No caso da camada transporte esses pontos são denominados TSAP (Transport Service Access Point).

USUÁRIO: requisita serviços de transporte.

ET: presta serviços de transporte.

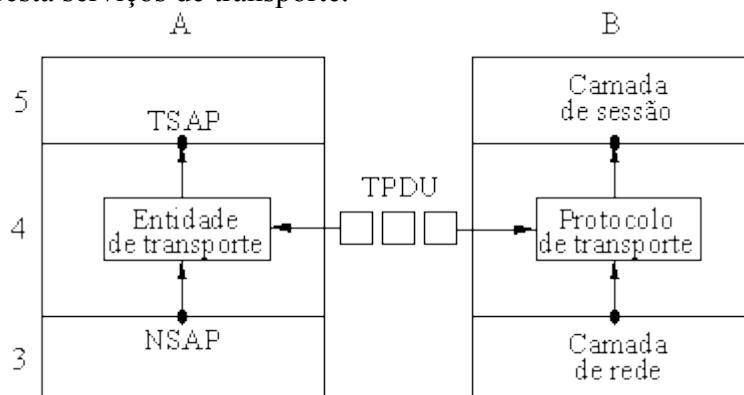


Figura 13.1 - Comunicação entre camadas

Primitivas de Serviço

As camadas se comunicam pelos TSAP utilizando-se de **primitivas de serviço**. Estas primitivas estão definidas em quatro tipos:

- **request** (pedido): iniciada por uma camada (N+1) para pedir algum serviço à camada (N);
- **indication** (indicação): emitida pela camada (N) à camada (N+1) para indicar a ocorrência de algum evento;
- **response** (resposta): enviada à camada (N) - pela camada (N+1), em resposta à indicação recebida da camada (N);
- **confirmation** (confirmação): emitida pela camada (N) para indicar ao originador do pedido do serviço que o serviço foi completado.

Um serviço que utiliza as quatro primitivas chama-se **serviço confirmado**. Um **serviço não-confirmado** utiliza-se apenas das duas primeiras: pedido e indicação.

O serviço de transporte recebe do nível superior a ele os dados a serem enviados, na forma de **unidade de dados de serviço de transporte** - TSDU (*Transport Service Data Unit*). Estas TSDUs são passadas ao serviço de transporte como parâmetros de primitivas de serviço usadas para transferir dados. As entidades

pares (entidades de mesma camada residentes em sistemas abertos diferentes) trocam elementos de protocolo denominados **unidades de dados do protocolo de transporte** - TPDU (*Transport Protocol Date Unit*).

Normalmente o tamanho de uma TSDU não é fixo, podendo exceder o tamanho que a TPDU pode transportar. Quando isso ocorre, o protocolo de transporte **segmenta** a TSDU e a transporta em várias TPDU, **remontando-a** novamente no destino.

Para minimizar as interações entre camadas, um usuário da camada de transporte entrega de uma só vez à essa várias TSDUs que serão transportadas em uma única TPDU. Este procedimento é chamado de **concatenação**. Na recepção, o usuário da camada de transporte deve ser capaz de identificar e separar as diversas TSDUs entregues em uma única interação por essa. Este procedimento é chamado de **separação**.

Uma **conexão de transporte** é definida como uma associação estabelecida entre entidades de sessão com o objetivo de transferir dados. Cada conexão de transporte é associada a uma conexão de rede - no caso das duas camadas estarem operando em modo orientado à conexão. Nessa situação, se a conexão de rede falhar, a conexão de transporte pode ser associada à outra conexão de rede; e as TPDU perdidas podem ser retransmitidas. Tal procedimento é conhecido como **reassociação após falha**.

Nem sempre o mapeamento das conexões é de **um-para-um**, podendo ocorrer no transmissor uma **multiplexação** de mais de uma conexão de transporte em uma conexão de rede. O inverso ocorre no receptor, a **demultiplexação** da conexão de rede nas várias conexões de transporte. Além disso, pode acontecer de termos uma conexão de transporte associada a mais de uma conexão de rede. Este processo é denominado, no transmissor, de **splitting**. No receptor, acontece o processo inverso, chamado de **recombinação**.

Quando tem-se uma conexão estabelecida, pode-se transmitir dados. A camada de transporte transmissora faz uma enumeração das TPDU enviadas, para que a camada de transporte receptora possa fazer a ressequência das mesmas, garantindo que o receptor receba as TPDU na mesma ordem que o transmissor as enviou. Existem dois tipos de enumerações para a camada de transporte: normal e extendida. A numeração extendida é utilizada em redes de alto *throughput*, exigindo poucos reconhecimentos de recebimentos de dados. Até que tal reconhecimento chegue, ocorre a **retenção das TPDU** correspondentes pela a entidade de transporte transmissora, permitindo assim uma retransmissão se necessário. A retransmissão feita pela camada de transporte por ocorrência de erros na camada de rede é conhecida como **ressincronização** da conexão de transporte.

A camada de transporte pode realizar um controle de erros através de uma **checksum** inserida como um parâmetro das TPDU. Ela também realiza um controle de fluxo denominado **janela deslizante** com alocação de crédito. Levando em consideração que estes artifícios podem provocar um retardo no fluxo de dados; então, para transmitir dados com urgência, foi definido prioridades para os chamados **dados expressos** sobre os dados normais.

Terminada a transmissão de dados, a conexão é liberada normalmente. Quando a conexão é liberada por ocasião de uma reinicialização ou por falha da conexão de rede, diz-se que houve uma **liberação com erro**.

A camada de transporte provê um mecanismo sutil de prevenção a erros após a liberação da conexão. O mecanismo trata-se de não permitir que sejam reutilizadas referências de transporte durante um período de tempo grande o bastante para que não tenham mais TPDU com tais referências circulando pela rede. Tais referências são ditas **referências congeladas**.

Tipos de TPDUs

FASE DE OPERAÇÃO	TPDU	FUNÇÃO
Estabelecimento de conexão	CR	Pedido de Conexão
	CC	Confirmação de conexão
Transferência de dados normais	DT	Dados normais
	AK	Reconhecimento de dados normais
	RJ	Rejeição de dados normais
	ED	Dados expressos
	EA	Reconhecimento de dados expressos
Liberação da Conexão	DR	Pedido de liberação de conexão
	DC	Confirmação de liberação de conexão
	ER	Indicação de erro de protocolo

Figura 13.2 - Tipos de TPDUs

Formato das TPDU

Todas as TPDU devem conter um número inteiro de octetos que são numerados a partir de 1 e na ordem crescente em que são colocados na NSDU. Os *bits* no octeto são numerados de 1 até 8, onde o *bit* 1 é o de menor ordem. Na representação de números binários que usam octetos consecutivos, o octeto de menor ordem tem o valor mais significativo.

Podemos representar a constituição de uma TPDU, genericamente, por:

- um cabeçalho constituído de:
 - um campo indicador de tamanho **LI**;
 - uma **parte fixa** e uma **parte variável** (se existir);
- um campo de dados (se existir).

A figura abaixo esclarece com perfeição o que foi dito acima.

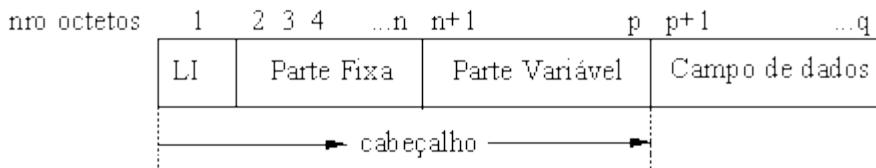


Figura 13.3

O campo indicador de tamanho **LI** está contido no primeiro octeto (8 bits) de todas as TPDU. Ele é um número binário e seu valor máximo é 254 (11111110). LI corresponde ao tamanho do cabeçalho, excluindo a ele mesmo.

A **parte fixa do cabeçalho** contém parâmetros de uso freqüente. O primeiro parâmetro da parte fixa é o código da TPDU, e a partir dele que são definidos o tamanho e a estrutura da TPDU. Em certos casos, a classe do protocolo e o formato (normal ou estendido) também influenciam no tamanho e na estrutura da TPDU.

Em contradição à parte fixa, a **parte variável do cabeçalho** contém os parâmetros menos freqüentemente usados. A parte variável, se existir, deve conter um ou mais parâmetros. Seu tamanho é definido pelo LI menos o tamanho da parte fixa.

O campo **indicação do tamanho** do parâmetro contém, em octetos, o tamanho do campo **valor do parâmetro**. Por fim o campo de dados, contendo os dados do usuário - que são tratados de forma transparente pelo protocolo de transporte. Pode-se representar a estrutura de cada TPDU da seguinte forma:

LI	CB	CDT	DST_REF	SRC_REF	CLASSE/ OPÇÃO	TSAP-ID	TAMANHO TPDU	VERSÃO	SEGURANÇA	CHECKSUM	ACK TIME	DADOS
1110	XXXX											

Campos da TPDU:

Tabela 13.1

LI	indicador de tamanho
TDU	unidade de dados do protocolo de transporte
CDT	crédito
TSAP-ID	ponto de acesso do serviço de transporte
YR-TU-NR	número da seqüência
AKTIME	tempo de acknowledgement
DST-REF	destino
SRC-REF	fonte

Camada de Transporte da INTERNET

A camada de transporte tem o objetivo de prover uma comunicação confiável entre dois processos, estando eles ocorrendo dentro da mesma rede ou não. Ela deve garantir que os dados sejam entregues livres de erros, em seqüência e sem perdas ou duplicação.

A Arquitetura Internet especifica dois tipos de protocolos na camada de transporte: o UDP (User Datagram Protocol) e o TCP (Transmission Control Protocol). O UDP é um protocolo não orientado à conexão que pode ser considerado como uma extensão do protocolo IP, e não oferece nenhuma garantia em relação à entrega dos dados ao destino.

Já o protocolo TCP oferece aos seus usuários um serviço de transferência confiável de dados, através da implementação de mecanismos de recuperação de dados perdidos, danificados ou recebidos fora de seqüência, minimizando o atraso na sua transmissão.

A cada fragmento transmitido é incorporado um número de seqüência, de forma a não se perder a ordem dos segmentos a serem juntados para formar o datagrama. Existe um mecanismo de reconhecimento para executar essa função que funciona da seguinte forma: o reconhecimento transmitido pelo receptor ao receber o segmento X é o número do próximo segmento que o receptor espera receber ($X+1$), indicando que já recebeu todos os segmentos anteriores a este. Através da análise dos números de segmento, o receptor pode ordenar os segmentos que chegaram fora de ordem e eliminar os segmentos duplicados. Com base no checksum que é adicionado a cada segmento transmitido, os erros de transmissão são tratados e os segmentos danificados são descartados. Existe ainda um controle de fluxo baseado no envio da capacidade de recebimento do receptor, contado a partir do último byte recebido, ao transmissor. Desta forma o transmissor consegue controlar a quantidade de dados que são enviados ao receptor para não haver descarte de segmentos nem necessidade de retransmissão, que ocasionam a queda do desempenho da rede.

Para permitir que vários usuários (processos de aplicação) possam utilizar simultaneamente os serviços do protocolo TCP, foi criado o conceito de porta. Para não haver problemas de identificação de usuários, o identificador da porta é associado ao endereço IP onde a entidade TCP está sendo realizada, definindo assim um socket. A associação de portas a processos de aplicação (usuários) é tratada de forma independente por cada entidade TCP. No entanto, processos servidores que são muito utilizados, como FTP, Telnet, etc, são associados a portas fixas, divulgadas aos usuários. Uma conexão é identificada pelo par de sockets ligados em suas extremidades. Um socket local pode participar de várias conexões diferentes com sockets remotos. Uma conexão pode ser utilizada para transportar dados em ambas as direções simultaneamente, ou seja, as conexões TCP são full-duplex.

É importante observar aqui que quando se fala que o TCP é orientado à conexão, não se fala em conexão a nível físico, mas sim a nível lógico. Este conceito pode ser compreendido através da figura abaixo.

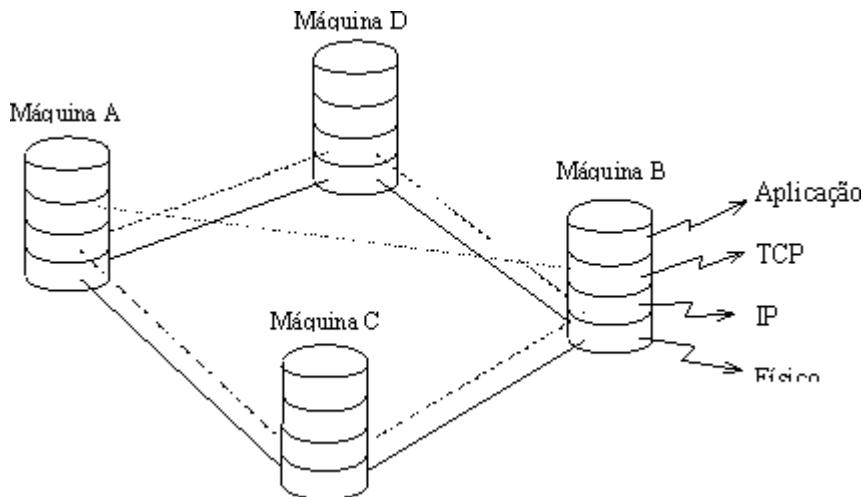


Figura 13.4 - Serviços orientados e não orientados à conexão

No caso da figura acima, a máquina A quer se comunicar com a máquina B através de uma rede em anel utilizando TCP/IP. A única conexão física que existe entre A e B é através do anel, passando pelas máquinas C e D. Em nível de IP, a comunicação não é orientada à conexão, portanto é muito simples enxergar que os dados possuem apenas dois caminhos para ir de A até B: através de C ou através de D. Em nível de TCP, porém, a comunicação entre os computadores A e B ocorre como se houvesse uma conexão direta entre eles. Isso implica que, se em nível de IP os dados podem chegar fora de ordem, o TCP tem que garantir a ordenação destes dados, de forma que eles sempre cheguem na ordem correta, como aconteceria se houvesse uma conexão física direta entre A e B.

Capítulo

12

Camada de Sessão

As camadas de sessão, apresentação e aplicações formam o conjunto das camadas superiores do Modelo de Referência OSI. Em contraste com as quatro camadas inferiores, que visam suprir a comunicação confiável fim a fim, as camadas superiores estão relacionadas com o fornecimento de serviços voltadas ao usuário. Elas usam canais principais, livres de erros, proporcionados pela camada de transporte, e acrescentam recursos adicionais úteis para uma ampla variedade de aplicações, de forma que as pessoas que escrevem essas aplicações não tenham de reimplementar repetidamente esses recursos, inúmeras vezes, como parte de cada programa diferente.

A camada de sessão é basicamente uma invenção da ISO. Antes do Modelo OSI, nenhuma rede existente possuía uma camada de sessão (embora alguns serviços de sessão da OSI estejam presentes na SNA, ainda que espalhados por diversas camadas). Durante o desenvolvimento do OSI, houve considerável polemica sobre a necessidade de uma camada de sessão. Por exemplo, a proposta britânica à ISO tinha apenas cinco camadas e não incluía a camada de sessão.

Muito embora a grande maioria do comitê da ISO decidisse posteriormente incluir uma camada de sessão, fica evidente que a partir da brevidade deste capítulo que a camada de sessão é uma camada “magra”, com relativamente poucos recursos, em relação às camadas mais baixas. Além disso, quando se estabelece uma conexão com a camada de sessão podem ser selecionadas opções que desativam maior parte dos recursos disponíveis. Ela não é tão importante quanto, digamos, a camada de transporte, e muitas aplicações nem se quer necessitam dos poucos recursos que ela possa ter. Apesar de tudo, ela é parte do Modelo de Referência do OSI; assim, passamos ao estudo dos serviços que a camada de sessão oferece e do modo como eles funcionam.

12.1. TEMAS DE PROJETO DA CAMADA DE SESSÃO

Nesta seção, iremos discutir algumas das questões relevantes para o projeto da camada de sessão. Estas questões incluem o gerenciamento de diálogos, a sincronização e o gerenciamento das atividades, entre outras. Todas podem ser consideradas como serviço “de valor agregado” inseridos no topo da conexão de transporte pura.

12.1.1. SERVIÇOS FORNECIDOS À CAMADA DE APRESENTAÇÃO

A camada de sessão fornece serviços à camada de apresentação. Sua posição hierárquica é mostrada na figura 16.1.

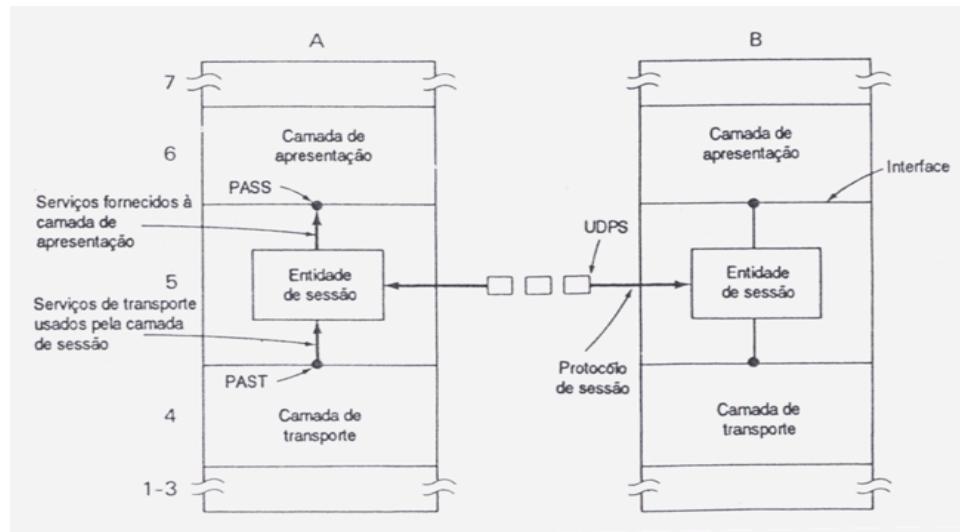


Figura 12.1. As camadas de Transporte, Seção e Apresentação.

Essa figura é semelhante a uma outra mostrada anteriormente, exceto pelo fato de os pontos de acesso de serviço serem chamados de pontos de acesso de aos serviços de sessão (PASSs) e de as unidades de dados serem denominadas unidades de dados do protocolo de sessão (UDPSs). De modo análogo ao ocorre na camada de transporte, usaremos o termo fornecedor do serviço de sessão e entidade de sessão de forma intercambiável.

A principal função da camada de sessão é oferecer meios para os usuários da camada de sessão (exemplos: entidades de apresentação ou, às vezes, simplesmente processos comuns do usuário) estabeleçam conexões, chamadas sessões e transfiram dados por intermédio delas de forma ordenada. A sessão poderia ser usada para um login remoto para um terminal distante, ou ainda para uma transferência de arquivos, ou ainda para qualquer entre muitos propósitos. Embora as primitivas sem conexões estejam disponíveis na camada de sessão, uma sessão sem conexões não pode fazer qualquer uso dos recursos voltados ao usuário para os quais a camada de sessão foi projetada. Por essa razão focalizaremos inicialmente o modelo baseado em conexões. Contudo, mais adiante, discutiremos neste capítulo, uma aplicação interessante das sessões sem conexões.

Uma sessão guarda uma estreita semelhança com uma conexão de transporte, mas as duas não são idênticas. Em geral, quando chega uma solicitação para que a camada de sessão estabeleça uma sessão, deve ser estabelecida uma conexão de transporte a fim de levar a cabo a comunicação. Quando a sessão é encerrada, também é liberada a conexão de transporte. Nesse exemplo, há um mapeamento de um para um entre a sessão e a conexão de transporte. Essa situação está ilustrada na figura 16.2.

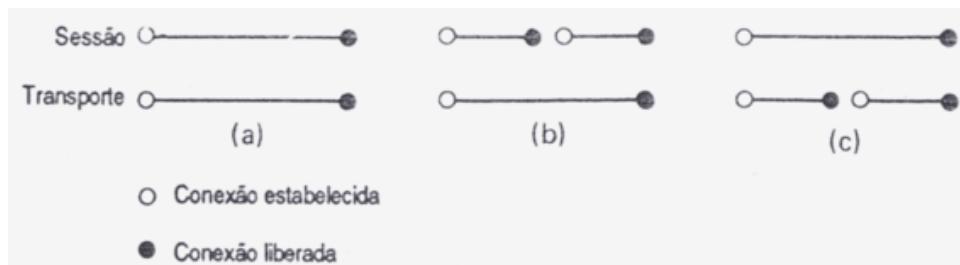


Figura 12.2. Três formas de mapear sessões sobre conexões de transporte

- d) Mapeamento de um para um
- e) Sessões consecutivas utilizam a mesma conexão de transporte
- f) Uma sessão abrange varias conexões de transporte. O eixo horizontal corresponde ao tempo.

Entretanto, também são possíveis outros mapeamentos. Considere o caso de uma empresa aérea com escritórios de reserva em muitas cidades. Cada escritório tem agentes com terminais conectados a um minicomputador do escritório local. Os minicomputadores estão conectados por uma rede remota a um computador principal que contém o banco de dados reserva. Sempre que um agente responde a uma chamada, se estabelece uma sessão para o computador principal. Quando a chamada é processada a sessão se encerra, mas não há necessidade de se dar ao trabalho de liberar a conexão de transporte subjacente, porque ela será seguramente necessária dentro de alguns segundos.

Um terceiro mapeamento possível entre as sessões e as conexões de transporte é dado na figura 7-2(c). Neste caso temos uma sessão que abrange varias conexões de transporte. Se, por exemplo, falhar (seja qual for o motivo) uma conexão de transporte, a camada de sessão pode estabelecer uma nova conexão de transporte e continuar a sessão através dela. Se as entidades de transporte residirem nos hosts, essa situação não deve ocorrer, por as entidades de transporte devem recuperar elas próprias de falhas na camada de rede (i.e., sub-rede). Contudo, se as entidades de transporte forem externas aos hosts, o problema da recuperação de falhas externas é deslocado para a camada de sessão, porque esta se torna então a camada de software mais baixa que pode sobreviver a desastres na sub-rede.

Para completar, não é permitido multiplexar diversas sessões simultaneamente sobre uma conexão de transporte da forma como a camada de transporte pode multiplexar varias conexões de transporte sobre uma conexão de rede. Em um instante qualquer, cada conexão de transporte conduz no máximo uma sessão. A multiplexação é feita para reduzir os custos ou melhorar o desempenho, funções da camada de transporte.

12.1.2. INTERCÂMBIO DE DADOS

O recurso mais importante da camada de sessão é o intercâmbio de dados. Uma sessão, como uma conexão de transporte passa por três fases: estabelecimento, utilização e liberação. As primitivas fornecidas à camada de apresentação para estabelecer, usar e liberar são bastante semelhantes às primitivas fornecidas a camada de sessão para estabelecer, usar e liberar conexões de transporte. Em muitos casos tudo que a entidade de sessão tem de fazer quando uma primitiva é invocada pelo

usuário de sessão é chamar uma primitiva de transporte correspondente para que o trabalho seja realizado.

Por exemplo, quando um usuário se sessão emite uma primitiva S – CONNECT_pedido para estabelecer uma sessão, o fornecedor da sessão simplesmente emite um T – CONNECT_pedido para estabelecer uma conexão de transporte (supondo que nenhuma conexão de transporte esteja disponível). De modo semelhante, o estabelecimento da sessão, como o estabelecimento da conexão de transporte, envolve negociação entre parceiros (usuários) para definir os valores dos diversos parâmetros. Na realidade, alguns desses parâmetros são pertinentes à conexão de transporte, tais como a qualidade do serviço e o flag que informa se são ou não permitidos os dados expedidos, que são simplesmente passados para conexão de transporte sem qualquer modificação. Outros parâmetros estão relacionados especificamente com à camada de sessão. Por exemplo, se uma sessão está sendo estabelecida entre dois computadores com a finalidade de intercambiar correio eletrônico em ambos os sentidos, um parâmetro de sessão poderia especificar qual dos dois lados será o primeiro a transmitir.

A diferença entre uma liberação disciplinada e uma liberação abrupta pode ser vista na figura 7-3. Na liberação abrupta tão logo um usuário emite a primitiva adequada para a desconexão, nenhum dado pode ser entregue a ele. Na figura 7.3(a), uma vez que B tenha invocado T – DISCONNECT_pedido, sua extremidade da conexão é imediatamente liberada. Ele não pode receber a mensagem que está atualmente em transito para ela, mesmo se essa mensagem chegar antes que a UDPT DP chegue a extremidade A. Além disso, A pode não recusar o pedido de liberação da conexão. O fornecimento do serviço de transporte apenas emite uma T-disconnect.indicação e pronto. É como uma conexão telefônica: quando uma parte desliga o telefone, a conexão é encerrada, não importa o desejo da outra parte.

A liberação disciplinada funciona de forma diferente. Ela utiliza um handshake completo com as primitivas *pedido*, *indicação*, *resposta* e *confirmação*. Na figura 7-3(b), mesmo depois de ter emitido uma primitiva S – RELEASE_pedido, B ainda pode aceitar a mensagem até que A tenha aceitado a liberação. Simplesmente por ter emitido um S – RELEASE_pedido não encerra por si mesmo a sessão. O usuário remoto tem que concordar. Se o usuário remoto quiser continuar a sessão ele pode rejeitar a tentativa de liberação (fixando um parâmetro na primitiva S – RELEASE.resposta), e a sessão continua como se nada tivesse acontecido. Uma sessão só é encerrada quando ambas as partes concordam em faze-lo.

Se você acha estranho que a liberação disciplinada esteja presente na camada de sessão mas não na camada de transporte pode ter certeza de que você não está sozinho. Quando delineava o padrão da camada de transporte para o governo dos E.U.A, o National Bureau of Standards (NBS) também considerou este detalhe bastante estranho e decidiu acrescentar uma opção de liberação disciplinada à camada de transporte, solucionando dessa forma o problema e ao mesmo tempo tornando o padrão dos E.U.A diferente do que era usado no resto do mundo. As pessoas que lidam com padrões precisam com freqüência escolher entre faze-lo do modo certo ou da forma que todos fazem. Esse é o motivo pelo qual existem tantos padrões.

O endereçamento é outra área que a camada de sessão e a camada de transporte se diferenciam, embora apenas ligeiramente. Para estabelecer uma sessão, deve-se estabelecer o endereço PASS (em lugar do endereço PAST) a fim de se conectar a ele. Embora os padrões não informem como deve ser construídos os endereços PASS, é provável que um endereço PASS, na prática, em um endereço PAST complementado por algumas informações adicionais de identificação.

Outra maneira na qual o intercâmbio de dados de sessão difere do intercâmbio de dados de transporte é o número de diferentes tipos de dados. A camada de transporte apresenta dois fluxos de dados logicamente independentes, os dados regulares e os dados expedidos. A camada de sessão possui ambos ou mais dois (dados tipificados e dados de capacidade). Os dois outros fluxos se relacionam a recursos da camada de sessão que ainda não abordamos; assim, adiaremos a nossa discussão sobre eles até mais adiante neste capítulo.

12.1.3 GERENCIAMENTO DE DIÁLOGOS

Em princípio, todas as conexões OSI são full-duplex, ou seja, as UDPS podem se mover em ambos os sentidos simultaneamente sobre a mesma conexão. A comunicação full-duplex é mostrada na figura 7-4(a). Contudo, existem muitas situações em que o software da camada superior está estruturada de forma esperar que os usuários se revezem (comunicação half-duplex). Esse projeto não tem nenhuma relação com as limitações de alguns terminais antigos ainda existentes que não podem funcionar de modo full-duplex, mas sim com o projeto de software da camada superior de uma forma conveniente.

Como exemplo, considere um sistema de gerenciamento de bancos de dados que pode ser acessado a partir de terminais remotos (ex: a reserva de passagem aérea ou o banco em casa). O modo mais comum de operação é aquele que o usuário envia uma consulta ao sistema de banco de dados e espera a resposta. Permitir que os usuários transmitissem uma segunda e uma terceira consultas antes que a primeira tenha sido respondida complica desnecessariamente o sistema. Logicamente, é desejável o funcionamento do sistema no modo half-duplex: ou é a vez de o usuário transmitir ou a vez do sistema do banco de dados.

O controle (e o encorajamento) de quem deve ter a vez de conversar é chamado gerenciamento de diálogos e é um dos serviços que podem ser fornecidos pela camada de sessão quando solicitados.

A forma de implementação de um gerenciamento de diálogos é pelo uso de token de dados. Quando uma sessão é estabelecida, a operação half-duplex é uma das opções que podem ser selecionadas. Se é escolhida a operação half-duplex, a negociação inicial também determina qual dos lados recebe o primeiro token. Somente o usuário que retém o token pode transmitir os dados; o outro tem que permanecer em silêncio. Quando detentor do token encerra sua transmissão, passa o token ao seu parceiro usando a primitiva S – TOKEN-GIVE_pedido, como mostra a figura.

O que acontece se o usuário que não retém o token deseja transmitir dados?

Ele pode pedir polidamente o token, utilizando a primitiva S – TOKEN-PLEASE_pedido. O detentor do token pode concordar e passar o token, ou pode recusar e, nesse caso o outro usuário terá simplesmente de aguardar (ou enviar uma mensagem de emergência usando dados pedidos, que não exigem o token). Se a operação full-duplex é selecionado quando a sessão é estabelecida, não se usa nenhum token para transmissão de dados.

12.1.4 SINCRONIZAÇÃO

O outro serviço da camada de sessão é a sincronização, que é utilizada para devolver as entidades de sessão a um estado conhecido na eventualidade de um erro ou divergência. À primeira vista esse serviço pareceria desnecessário, por que a camada de transporte foi cuidadosamente projetada para se recuperar de modo transparente de todos os erros de comunicação e quedas de sub-rede. Contudo um estudo mais profundo mostra que a camada de transporte foi projetada somente para mascarar os erros de comunicação. Ela não pode se recuperar de erros de uma camada superior.

Como exemplo dos problemas que podem ocorrer, considere o serviço de teletexto que está substituindo gradualmente o telex (TWX) para a remessa de mensagens impressas de uma companhia para a outra. Os assinantes desse serviço utilizam um dispositivo contendo uma CPU, teclado, monitor, impressora, modem, telefone, e, algumas vezes, uma unidade de disco. Primeiro, o usuário compõe uma mensagem utilizando o monitor e o teclado. Então, a CPU chama a mensagem a empresa a que se destina a mensagem, estabelece uma sessão e transfere a mensagem. Se o dispositivo de recepção não possui armazenamento em disco, uma possibilidade real nas versões mais econômicas, as mensagens têm de ser impressas em tempo real, à medida que são recebidas.

Tão logo seja recebida, cada UDPS é confirmada e o texto é levado à impressora. Agora suponha que ocorra um problema com o papel ou com a fita. Mesmo que o operador humano perceba o problema rapidamente e aperte um botão do dispositivo a fim de interromper a impressão, algumas informações podem se perder. Mesmo que já tenham sido reconhecidas, o transmissor não terá mais uma cópia e a transmissão da mensagem falhará.

Não há nada que a camada de transporte possa fazer com relação a esse problema; afinal, afinal ela realizou com perfeição o seu trabalho: deslocou todos os bits de modo confiável do transmissor até o receptor. Só depois de transferir as UDPS para a camada de sessão é que ela enviou de volta a confirmação. Não é culpa da camada de transporte se as camadas superiores tenham falhado.

A solução se encontra na camada de sessão. Os usuários de sessão podem dividir o texto em páginas e inserir um ponto de sincronização entre elas. No caso de ocorrer algum problema, é possível reinicializar o estado de sessão a um ponto de sincronização anterior ou continuar a partir dele. É evidente que a fim de tornar possível este processo, chamado de ressincronização, o usuário de sessão transmissor (e não a entidade de sessão) tem que continuar a reter os dados pelo tempo em que poderiam ser necessários.

É importante compreender o que é a semântica de sincronização da camada de sessão. Os usuários da sessão podem introduzir pontos de sincronização no fluxo da mensagem. Cada ponto de sincronização contém um número de série. Quando um usuário emite uma primitiva a fim de solicitar um ponto de sincronização, o outro recebe uma indicação. De forma semelhante, quando um deles emite uma primitiva para ressincronização, o outro recebe também uma indicação disso. O salvamento de mensagem e a subsequente retransmissão se da acima da camada de sessão. Tudo que a camada de sessão proporciona é um meio de conduzir sinais numerados de sincronização e ressincronização através da rede.

O mecanismo é realmente um pouco mais complexo do que descrevemos até agora. Existem dois pontos diferentes de pontos de sincronização, o principal e o secundário, cada um com suas próprias primitivas. As unidades delimitadas pelo ponto de sincronização principal são chamadas de unidades de diálogos e representam logicamente partes significativas do trabalho. Por exemplo, quando se transmite um

livro, os capítulos poderiam estar delimitados por pontos de sincronização principais e as páginas por pontos de sincronização secundários.

Os pontos de sincronização diferem entre si de diversas maneiras. Uma delas é que, quando se dá ressincronização, só é possível retornar a ponto de sincronização mais recente e nada mais. Portanto, no intervalo de tempo entre os pontos de sincronização 6 e 7 na figura 7-6 é permitido ressincronizar voltando ao ponto 6, mas não aos pontos 1,2,3,4 ou 5. Erguendo um muro em cada ponto de sincronização principal, o transmissor sabe que os dados que foram enviados antes dele podem ser descartados com segurança. Os pontos de sincronização secundários não possuem essa propriedade. Apenas antes do ponto de sincronização 6 na Figura 7-6 é permitido ressincronizar para 1, 2, 3, 4, ou 5. É claro que a escolha do tipo e do local de utilização do ponto de sincronização é tarefa dos usuários de sessão.

Os pontos de sincronização principais são tão significativos que cada um deles inserido no fluxo de dados é confirmado explicitamente. Os pontos de sincronização secundários não são confirmados. Em um canal de satélite, onde o retardo mínimo para enviar a mensagem e obter a confirmação é de 540ms, essa distinção pode ser importante.

Fixar um ponto de sincronização, seja ele principal ou secundário, requer a posse das fichas relevantes. Estão disponíveis duas fichas independentes (para sincronização principal e secundária). Cada uma delas se distingue da outra e as duas são diferentes das fichas usadas para fluxo de dados de controle nas conexões half-duplex. Quando ocorre a ressincronização, todas as fichas são restauradas as posições que elas tinham no instante em que foi sincronizado o ponto de sincronização.

12.1.5 Gerenciamento de atividade.

Outro recurso fundamental da camada de sessão, intimamente relacionada à sincronização, é o gerenciamento da atividade. A idéia por trás do gerenciamento da atividade é permitir que o usuário divida o fluxo de mensagens em unidades lógicas denominadas atividades. Cada atividade é completamente independente de quaisquer outras atividades que possam ter vindo antes ou que venham depois dela.

Cabe ao usuário determinar o que é uma atividade. Como primeiro exemplo, considere uma sessão definida para a finalidade de transferir diversos arquivos entre computadores. É necessário algum meio de assinalar o local onde termina um arquivo e onde começa o próximo. Utilizar o caractere ASCII (file separator) não é uma boa idéia porque, se os arquivos contiverem informações binárias, esse caractere poderia aparecer nos dados e indicar acidentalmente o final do arquivo quando não fosse esta a intenção. Uma possibilidade é utilizar alguma forma de preenchimento de caracteres, como é feita na camada de enlace de dados, mas na camada de sessão o preenchimento terá de ser feito provavelmente em software em vez de hardware e assim será lento.

O que realmente é necessário, é algum meio de inserir um marcador no fluxo de mensagem que seja ele próprio distinto de uma mensagem de dados (algumas vezes chamado de informação fora de faixa). Uma forma de alcançar este objetivo é definir cada transferência de arquivo como uma atividade diferente. Antes de ser iniciada cada transferência de arquivo, o transmissor emite uma primitiva S – ATIVIDADE – INICIO.pedido, que aparece aparece do outro lado como uma S – ATIVIDADE – INICIO.indicação, assinalando o inicio do arquivo. De forma similar,

depois que cada transferencia de arquivo se completa, pode ser utilizada a primitiva S – ATIVIDADE – FIM para denotar o final do arquivo.

É importante enfatizar aqui do que constitui uma atividade é feita pelos usuários, e não pela camada de sessão. Tudo que a camada de sessão faz é assegurar que, quando umas solicitações da S – ATIVIDADE [e feita por um usuário, o outro usuário receba indicação correspondente. Não é de interesse da camada de sessão quando tais solicitações são feitas e como o receptor reage às indicações. A camada de sessão se interessa apenas pela execução das primitivas, e não pelo seu significado (semântica) ou sua utilização. Vale a pena notar que uma atividade abrange todo o tráfego enviado em ambos os sentidos].

Como um segundo exemplo da maneira pela qual o gerenciamento da atividade pode ser utilizado, considere um sistema de “banco em casa” no qual as pessoas podem fazer pagamentos de contas usando seus computadores pessoais para transferir dinheiro de suas contas para as das companhias que emitem as contas. O programa em execução no computador pessoal poderia começar pedindo o número da conta a ser

debitada e enviando essa informação como primeira mensagem. Depois, ele poderia pedir sucessivamente o número da conta a receber o crédito enviar esses itens como as mensagens dois e três.

Quando a primeira mensagem chega ao computador do banco, o registro em disco que contém a conta a ser debitada é localizado e bloqueado a enviar qualquer acesso concorrente. Quando chega a segunda mensagem, o registro da conta a ser creditada também é bloqueado. Ao chegar a terceira mensagem, o dinheiro é transferido e as duas contas são desbloqueadas.

Imagine o que aconteceria se acontecesse uma falha de energia elétrica que deixasse a casa do usuário no escuro exatamente depois que a primeira mensagem foi enviada, recebida no banco e processada. A transação nunca se completaria e a conta ficaria para sempre bloqueada. Para evitar situações como essa, a transação bancária poderia ser estruturada como uma atividade da camada de sessão. Depois de receber a S-ATIVIDADE-INÍCIO.indicação, o computador do banco poderia simplesmente acumular as mensagens que chegasse até a S-ATIVIDADE-FIM.indicação. Assinalasse que não haveria mais nenhuma. Só então, começaria o processamento e o bloqueio. Dessa forma, nenhuma falha externa poderia fazer o computador do banco se perder no meio de uma transação.

A técnica de juntar mensagens em um buffer de entrada até que todas elas tenham chegado antes de iniciar o processamento de qualquer uma delas é denominado isolamento. Nos primeiros esboços da camada de sessão. Mais tarde, contudo, o comitê da ISO concluiu que o isolamento poderia ser realizado igualmente bem pelo uso do gerenciamento de atividade, e assim o serviço de isolamento não foi incluído no padrão publicado.

No terceiro e último exemplo de gerenciamento de atividade faz uso de uma propriedade que ainda não mencionamos: as atividades podem ser interrompidas (ou seja, suspensas) e mais tarde reiniciadas sem perdas de informações. Considere o caso de alguém que começou a transferir um arquivo muito longo de seu computador do trabalho para o computador pessoal de sua casa. Em meio a transferência, ele precisa fazer uma chamada telefônica urgente e precisa examinar o número de telefone no catálogo telefônico on-line do escritório, de preferência sem arruinar a transferência do arquivo.

Existe uma solução para esse problema. A transferência do arquivo é iniciada como uma atividade. Em certo momento do processo, é possível emitir um S-

ATIVIDADE-INTERRUPÇÃO. pedido a fim de suspender a transferência do arquivo. Então, outra atividade pode ser iniciada e completada e finalmente a atividade original pode ser retomada a partir do ponto em que foi interrompida.

O gerenciamento de atividades é a forma principal de estruturar uma sessão. Por essa razão, é essencial que ambas as partes concordem em relação ao que seja a estrutura de atividades. Poderia surgir um problema se ambas tentassem iniciar simultaneamente atividades. Para evitar que ocorra esse evento, o gerenciamento de atividades é controlado por uma ficha (na verdade a mesma ficha usada para os pontos de sincronizações principais). Para invocar um serviço de atividade, um usuário deve estar de posse da ficha pode ser passada e solicitada de forma independente dos dados e das fichas de sincronização secundária.

Na realidade, a situação é pouco mais complicada do que descrevemos. Ela começou como mostramos; porém, a ISSO concluiu mais tarde que poderiam ocorrer problemas se um usuário iniciasse uma atividade enquanto o outro estivesse fazendo uma sincronização secundária. Para evitar essa situação, as regras foram alteradas d modo a exigir que um usuário retenha tanto a ficha de atividade quanto à de sincronização secundária, como também a ficha de dados (se utilizada) antes de iniciar uma atividade ou uma operação de sincronização. Essa estratégia eliminou os problemas originais, mas criou um novo: o que acontece se um usuário retém a ficha de atividade e o outro retém a ficha de sincronização secundária, e ambos querem as duas fichas? Cada um fica em um loop, emitindo uma primitiva S-PEDIR-FICHA, sem chegar a lugar nenhum. O resultado é um empasse. A única solução é todas as aplicações tentarem ser muito cuidadosas. Em retrospecto, uma única ficha poderia ter sido melhor.

As atividades estão intimamente relacionadas aos pontos de sincronização. Quando uma atividade é iniciada, os números de serie de sincronização são reinicializados em 1 e é estabelecido um ponto de sincronização principal. É possível estabelecer pontos de sincronização adicionais, sejam eles principais ou secundários, dentro de uma atividade.

Como inicio de cada atividade também corresponde a um ponto de sincronização principal, uma vez que uma atividade tenha se iniciado, não é possível ressincronizar a um ponto anterior ao de inicio dessa atividade. Em particular não é possível ressincronizar voltando a um ponto de sincronização de uma atividade anterior.

12.1.6 Relatório de Exceções

outro recurso de camada de sessão é um mecanismo de uso geral para relatar erros inesperados. Se o usuário tiver problemas por qualquer motivo, esses problemas podem ser relatados ao parceiro pelo uso de uma primitiva S-U-RELAT-EXCEÇÃO. pedido. alguns dados do usuário podem ser transferidos usando essa primitiva. Em geral, os dados do usuário irão explicar o que aconteceu.

Como exemplo, suponha que um programador projete um protocolo complexo de sete camadas de uso geral para alguma aplicação específica. Durante a depuração do protocolo(ou mesmo depois de que ele esteja supostamente depurado) podem ocorrer erros de protocolo. Estes podem ser relatados ao parceiro usando o mecanismo de relatório de exceções da camada de sessão.

O relatório de exceções não se aplica apenas a erros detectados pelo usuário. O fornecedor do serviço pode gerar uma S-F-RELAT-EXCEÇÃO. indicação a fim de

notificar o usuário sobre problemas internos na camada de sessão ou problemas relatados a ele pela camada de transporte ou pelas camadas inferiores. Esses relatórios contêm um campo descrevendo a natureza da exceção. Cabe ao usuário decidir que ação executar, se for o caso.

12.1.7 As Primitivas de Serviço de Sessão no OSI

Nesta seção iremos examinar sistematicamente todas as primitivas de sessão OSI e descrever a função de cada uma. Elas estão listadas na figura 7-10. Cada linha da tabela corresponde a um grupo de uma a quatro primitivas.potencialmente, cada tipo de primitiva tem suas versões *pedido*, *indicação*, *resposta* e *confirmação*. Entretanto, nem todas as combinações são validas. Por exemplo, S-DADOS tem somente *pedido* e *indicação*.as confirmações (i. e.,*resposta* e *confirmação*) são de responsabilidade das camadas mais baixas.

Primitiva de sessão do OSI	Ped.	Indic.	Resp.	Conf.	Significado
S-CONEXÃO	X	X	X	X	Estabelece uma sessão
S-LIBERAÇÃO	X	X	X	X	Encerra uma sessão de forma elegante
S-U-ABORTA	X	X			Liberação abrupta iniciada pelo usuário
S-F-ABORTA		X			Liberação abrupta iniciada pelo fornecedor
S-DADOS	X	X			Transferência de dados normal
S-DADOS-XPRESSOS	X	X			Transferência de dados expedidos
S-DADOS-TIPIFICADOS	X	X			Transferência de dados fora de faixa
S-DADOS-DE-CAPACIDADE	X	X	X	X	Transferência de dados de informações de controle
S-ENTREGAR-FICHA	X	X			Entrega uma ficha ao parceiro
S-PEDIR-FICHA	X	X			Solicita uma ficha ao parceiro
S-ENTREGAR-CONTROLE	X	X			Entrega todas as fichas ao parceiro
S-SINC-PRINCIPAL	X	X	X	X	Insere um ponto de sincronização principal
S-SINC-SECUNDÁRIO	X	X	X	X	Insere um ponto de sincronização secundário
S-RESSINCRONIZAÇÃO	X	X	X	X	Retorna a um ponto de sincronização anterior
S-ATIVIDADE-INICIAR	X	X			Inicia uma atividade
S-ATIVIDADE-FINALIZAR	X	X	X	X	Finaliza uma atividade
S-ATIVIDADE-DESCARTAR	X	X	X	X	Abandona uma atividade
S-ATIVIDADE-INTERROMP	X	X	X	X	Suspende uma atividade
S-ATIVIDADE-RETOMAR		X			Reinicia uma atividade suspensa
S-U-RELAT-EXCEÇÃO	X	X			Relatório de uma exceção de usuário
S-F-RELAT-EXCEÇÃO	X				Relatório de uma exceção do fornecedor

(a)

S-UNIDADES DE DADOS		X			Transferência de dados sem conexão
---------------------	--	---	--	--	------------------------------------

(b)

Figura 14-10 (a) *Primitivas do serviço de sessão baseadas em conexões do OSI* (b) *Primitivas do serviço de sessão sem conexões do OSI*

Existem 58 primitivas de serviço de sessão baseadas em conexões. Podem ser agrupadas em sete grupos para uma melhor compreensão:

1. Estabelecimento de conexões.
2. Liberação de conexões.
3. Transferência de dados.
4. Gerenciamento de fichas.
5. Sincronização.
6. Gerenciamento de atividades.
7. Relatório de exceções.

1. Estabelecimento de Conexões.

Contém seis primitivas da forma S-CONEXÃO.*xxx*: S-CONEXÃO.*pedido* especifica um identificador de sessão, os endereços de PASS (chamador e chamado), a qualidade de serviço, o número do ponto de sincronização inicial, atribuição inicial das fichas, alguns dados do usuário (opcionais) e possivelmente opções diversas.

As opções são fornecidas porque nem todas as sessões exigem todos os serviços que estão potencialmente disponíveis a sincronização, o gerenciamento de atividades, o relatório de exceções e certos tipos de transferências de dados, a serem discutidos em seguida, podem ser ativados ou desativados individualmente para cada sessão, dependendo das necessidades dos usuários.

2. Liberação de conexões.

Existem cinco primitivas relacionadas com a liberação de sessões. S-LIBERAÇÃO.*pedido* é usada para solicitar o término disciplinado de uma sessão. Uma alternativa a essa liberação comum é uma liberação negociada, que utiliza uma ficha de liberação. Quando essa opção tiver sido selecionada no momento do estabelecimento da sessão, somente o usuário de posse dessa ficha pode iniciar uma liberação. Esse recurso é algumas vezes útil quando as duas extremidades de uma sessão são iguais, exibindo um relacionamento mestre-escravo.

Também são fornecidas duas formas de liberação abrupta, uma iniciada pelo usuário e outra pelo fornecedor. A ultima somente ocorre se for detectado um erro fatal pela entidade de sessão.

3. Transferência de dados.

Conforme mencionamos rapidamente antes, existem quatro fluxos de dados independentes. São eles:

- 1.Dados regulares
- 2.Dados expedidos
- 3.Dados tipificados

4.Dados de capacidade

Os dois primeiros tipos já foram bastante discutidos, e assim vamos examinar os outros dois. Os dados tipificados são como dados regulares, exceto pelo fato de que sempre podem ser enviados, não importa quem seja o proprietário das fichas. Em contraste com a chegada de dados comuns, assinalada por uma primitiva S-DADOS.*indicação*, de modo que receptor pode informá-las em deparado. Os dados tipificados podem ser utilizados pelo usuário do serviço de sessão para mensagens de controle ou qualquer outra finalidade.

Os padrões da camada de sessão não especificam como os dados tipificados devem ser usados. Entretanto, a intenção do comitê da ISO foi proporcionar um fluxo de dados fora de faixa para informações de controle das camadas mais altas, a manutenção de dados e o gerenciamento do sistema. Por esse motivo, os dados tipificados podem ser enviados a qualquer tempo, sem se importar com fichas. Já examinamos antes este conceito de canal especial para informações de controle. No protocolo X.25, existe um bit Q no cabeçalho para indicar dados qualificados tipificados pudesse ser enviados usando o bit Q. infelizmente, o serviço da camada de transporte não tem nenhum conceito de dados tipificados. Assim, não existe meio para a entidade de sessão informar à camada de transporte que uma mensagem contém dados tipificados.

O quarto fluxo de dados, que é chamado de dados de capacidade, também foi elaborado para fins de controle, mas para controle da própria camada de sessão. Sua função real é permitir que as opções e parâmetros da sessão sejam modificados enquanto ela esteja em andamento (embora em teoria possam ser usados para qualquer finalidade com a qual concordem os usuários da sessão). Ao contrário dos dados tipificados, os dados de capacidade são completamente reconhecidos. Além disso, para evitar confusão, os dados de capacidade só podem ser transmitidos fora das atividades(i.e., entre elas), e somente quando há posse das fichas de dados, de sincronização e de atividade.

4 .Gerenciamento de fichas

A camada de sessão têm quatro fichas, como mostra a figura 7-11. A primitiva S-ENTREGAR-FICHA.*pedido* pode ser usada para passar uma ou mais fichas à entidade parceira. Os parâmetros especificam quais fichas devem ser entregues. A primitiva S-PEDIR-FICHA.*pedido* pode ser usada para anunciar que o usuário que emite a primitiva quer a fichas especificadas. Finalmente, usa-se a primitiva S-ENTREGAR-CONTROLE.*pedido* para resgatar todas as fichas de uma só vez. Essa primitiva só pode ser usada fora das atividades; logicamente, não é necessária, mas foi incluída a pedido do CCITT para compatibilidade com seu protocolo de teletexto, que funciona em termos de controle e não em termos de fichas.

Tipo de ficha	Controle
Ficha de dados	Transferência de dados no modo half-duplex
Ficha de liberação	Início da liberação disciplinada
Ficha de sincronização secundaria	Inserção de pontos de sincronização

	secundaria
Ficha principal/atividade	Atividade ou pontos de sincronização principais

Figura 7-11 *As fichas da camada de sessão*

5. Sincronização.

São fornecidas primitivas tanto para a sincronização principal quanto para a secundaria, bem como a resincronização. Todas as primitivas são confirmadas. Cada primitiva especifica o número de séries do ponto de sincronização que ela deseja estabelecer ou ao qual deseja retornar. Esses números de série estão na faixa de 0 a 999.999. Todas as primitivas de sincronização exigem a posse das fichas correspondentes.

6. Gerenciamento de atividades.

As atividades podem ser iniciadas, interrompidas, retomadas e descartadas (abandonadas). Como a sincronização, o gerenciamento de atividades é controlado por fichas.

7 .Relatório de exceções.

Observe que S-F-EXCEÇÃO, como S-F-ABORTAR, não pode ser solicitada. O fornecedor do serviço decide se e quando emiti-la.

As primitivas do OSI são implementadas pela entidade de sessão através do uso do protocolo de sessão. Discutiremos esse protocolo mais adiante, neste capítulo.

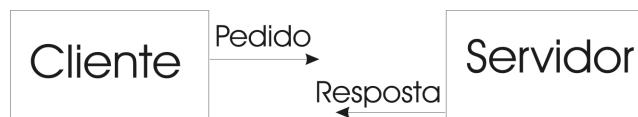
12.2 CHAMADA A PROCEDIMENTOS REMOTOS

A principal tarefa da camada de sessão é gerenciar o dialogo e lidar com erros (ex., quedas do sistema) que ocorrem acima da camada de transporte. No modelo de Referencia OSI, uma sessão sem conexões faz pouco sentido. Entretanto, existe pesquisa considerável nas universidades e na industria sobre um modelo radicalmente diferente para o controle do diálogo e dos erros com base no modelo sem conexões. Esse trabalho, com nome de RPC (Remote Procedure Call [chamada a Procedimentos Remotos]), foi amplamente implementado em redes e especialmente em sistemas distribuídos.

A RPC que não cabe muito bem no modelo de referencia OSI, foi projetada para ser rápida e, por conseguinte, não contem uma estrutura em múltiplas camadas. Logicamente, a chamada a procedimentos remotos se relaciona basicamente com as mesmas questões que envolvem a chamada de sessão, embora de uma perspectiva muito diferente. Assim, nós a examinaremos neste capítulo. Contudo, observe que a RPC também pode ser implementada na camada de aplicação, porém com menor eficiência. Portanto, as vezes encontramos discussões a seu respeito naquele contexto.

12.2.1 O Modelo Cliente Servidor

Até agora, temos pressuposto taticamente que os dois processos que se comunicam por uma conexão de sessão ou de transporte são simétricos. Na prática, essa suposição é violada com freqüência. Um exemplo comum é uma rede de computadores pessoais ou estações de trabalho sem unidades de disco, chamadas clientes, que se comunicam por uma rede com um servidor de arquivos que possui um disco no qual todos os arquivos estão armazenados. Neste sistema, os clientes acessam seus dados enviando pedidos ao servidor, o qual executa um trabalho solicitado e devolve as respostas. A comunicação sempre usa o formato de pares pedido-resposta, sempre iniciados pelos clientes, nunca pelo servidor. Esse modelo é chamado de modelo cliente-servidor



Conquanto seja obviamente possível estabelecer sessões entre clientes e servidores e depois usar a comunicação half-duplex sobre essas sessões, o alto overhead causado por múltiplas camadas de conexões é freqüentemente desestimulante para aplicações, tais como servidores de arquivos em que o desempenho é critico. Uma forma de comunicação totalmente sem comunicação, totalmente sem conexões construída exatamente na parte superior do recurso de datagrama simples (especialmente nas LANs) é quase sempre uma opção muito melhor.

Mesmo se os problemas de desempenho puderem ser solucionados pelo uso do modo sem conexões, o modelo ainda tem um defeito importante: a base conceitual de toda a comunicação é a I/O (entrada/saída). Os programas se comunicam uns com os outros usando comando tais como X-DADOS.*pedido* e X-DADOS.*indicação*, a primeira delas uma I/O e outra uma interrupção. Dificilmente essas seriam ferramentas apropriadas para a construção de aplicações bem estruturadas.

A linha de pensamento da RPC aborda o modelo cliente-servidor de um ponto de vista completamente diferente. Nessa perspectiva, um cliente que envia uma mensagem a um servidor e obtém uma resposta é como um programa que chama um procedimento e recebe um resultado. Em ambos os casos, o chamador inicia uma ação e aguarda até que ela seja completada e os resultados estejam disponíveis. Embora no caso comum (local) o procedimento rode em um equipamento diferente, o chamador não precisa estar ciente dessa distinção.

Para ajudar a oculta ainda mais a diferença entre chamadas locais e remotas, é possível incorporar à RPC à linguagem de programação. Supomos, por exemplo, que nós proporcionamos a cada um dos clientes no servidor de arquivos um procedimento (de biblioteca) *ler*, que pode ser chamado com três parâmetros: um indicador informando o arquivo a ser lido, um buffer para receber os dados lidos e uma contagem do numero de bytes a ler. Uma chamada como:

Ler (idarquivo, buffer, contagem)

É então uma chamada comum a um procedimento local (ou seja, um procedimento incluído pelo linkeditor no espaço de endereços do chamador). Esse procedimento transmite uma mensagem ao servidor de arquivos e espera pela resposta. Só depois que chega a resposta é que *ler* devolve o controle ao chamador.

A beleza desse esquema é que a comunicação cliente-servidor utiliza agora o formato de chamadas a procedimento em vez de comando I/O(ou pior ainda, interrupções). Todos os detalhes de funcionamento da rede podem ser ocultados do programa de aplicação pela sua inserção nos procedimentos locais tais como *ler*. Esses procedimentos são denominados stubs.

Nesse exemplo, o procedimento stub realmente transfere dados, mas um procedimento stub pode igualmente bem enviar uma mensagem solicitando que o servidor realize uma operação arbitaria. Por exemplo:

Eliminar(nome-de-arquivo)

Poderia fazer com que o procedimento stub *eliminar* enviasse uma mensagem ao servidor de arquivos pedindo a ele para destruir o arquivo especificado. Fornecendo procedimentos stub apropriados, podemos fazer com que o cliente invoque ações arbitrárias no servidor de uma forma muito mais natural para o programador de aplicações do que o trabalho com I/O e interrupções. O objetivo final é fazer com que uma chamada a procedimento remoto não aparente qualquer diferença em relação a uma chamada local.

12.2.2 Interpretação da Chamada a um Procedimento Remoto

Nesta seção examinaremos mais de perto a forma pela qual a RPC é implementada. Podem ser encontradas mais informações no trabalho de Birrell e Nelson (1984).

A chamada remota engloba dez etapas. A etapa 1 consiste no programa (ou procedimento) do cliente que chama o procedimento stub encadeado no interior do seu próprio espaço de endereços. Os parâmetros podem ser passados de maneira usual. O cliente não percebe nada fora do normal em relação essa chamada, por ser ela uma chamada local comum.

Então, o stub do cliente recolhe parâmetros e os condiciona em uma mensagem. Essa operação é conhecida como coleta parâmetros. Depois de construída a mensagem é entregue à camada de transporte para transmissão (etapa 2). Em um sistema de LAN sem conexões, a entidade de transporte provavelmente irá apenas anexar um cabeçalho a mensagem e coloca-la na rede sem trabalho adicional (etapa 3). Em uma WAN, a transmissão real pode ser mais complicada. Em muitos sistemas, a etapa 2 é um desvio para o sistema operacional.

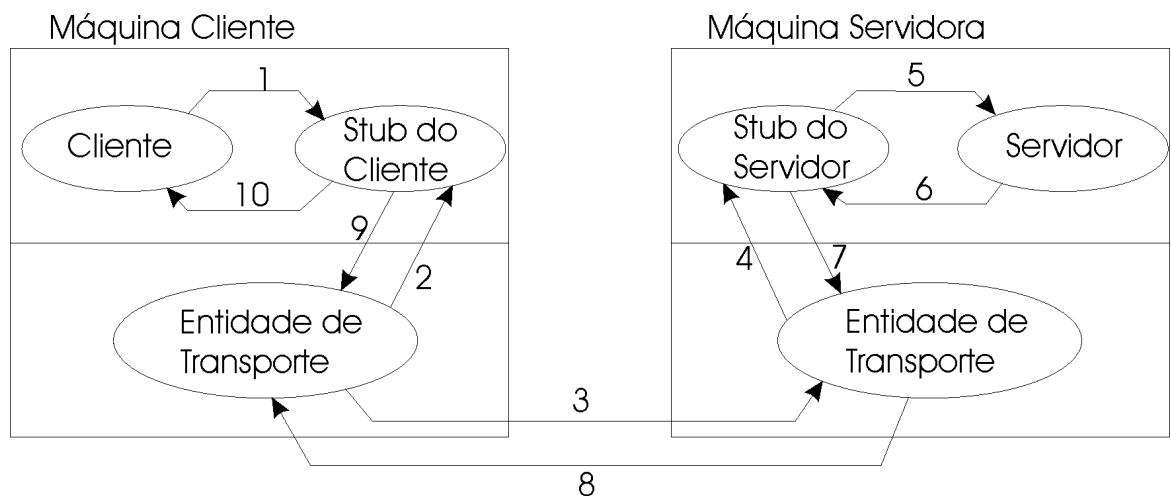


Figura 14-13. As dez etapas necessárias para a execução de uma chamada a procedimento remoto

Quando a mensagem chega ao servidor, a entidade de transporte desse lado a entrega ao stub do servidor (etapa 4), que desagrupa os parâmetros coletados. O stub do servidor chama então o procedimento do servidor (etapa 5), passando dos parâmetros na forma padrão. O procedimento do servidor não tem meios de saber que está sendo ativado remotamente porque seu chamador imediato é um procedimento local e obedece a todas as regras padrão. Somente os stubs sabem que algo peculiar está em andamento.

Depois de completar seu trabalho, o procedimento do servidor retorna (etapa 6), da mesma maneira que qualquer outro procedimento retorna quando se encerra. Ele também pode devolver um resultado a seu chamador. Então, o stub do servidor coleta o resultado em uma mensagem e a entrega à interface de transporte (etapa 7), possivelmente fazendo uma chamada ao sistema, exatamente como na etapa 2. Depois que a resposta retorna à máquina cliente (etapa 8), ela é entregue ao stub do cliente (etapa 9). Finalmente, o stub do cliente retorna ao seu chamador, o procedimento do cliente. Qualquer valor retornado pelo servidor na etapa 6 é entregue ao cliente na etapa 10.

O objetivo de todo o mecanismo é dar ao procedimento do cliente a ilusão de que ele está fazendo uma chamada direta ao procedimento servidor distante. Na medida em que a ilusão é bem sucedida e o cliente não pode saber que o servidor é

remoto, o mecanismo é dito ser transparente. Contudo, uma inspeção mais atenta revela algumas dificuldades para se conseguir transparência total.

O principal problema ocorre com passagem de parâmetros. É fácil passar por valores inteiros, números em ponto flutuante e strings de caracteres. O stub do cliente simplesmente os insere na mensagem. Na pior hipótese poderia ser necessária uma convenção para algum formato padrão da rede (tais conversões são parte da camada de apresentação e serão discutidas em detalhes no próximo capítulo). A passagem de estruturas, registros ou vetores desses tipos é igualmente direto.

O problema surge quando a linguagem permite que os parâmetros sejam passados por *referência*, em vez de por *valor*. Para uma chamada local, um ponteiro (o endereço do parâmetro) é passado normalmente ao procedimento chamado. Este procedimento sabe que esta lidando com um parâmetro de referência e assim pode seguir o ponteiro a fim de acessar o parâmetro.

Esta estratégia falha completamente para uma chamada remota. Quando o compilador produz o código para o servidor, ele não sabe nada sobre a RPC, e gera as instruções usuais para seguir o ponteiro. É evidente que o objeto que esta sendo apontado não se encontra nem mesmo na máquina do servidor e, ainda que estivesse lá, não teria o mesmo endereço que tinha na máquina do cliente. Como resultado quando um cliente tenta utilizar um parâmetro de referência, recebe o valor errado e a computação falha.

Uma solução possível é substituir o mecanismo de parâmetros chamados por referência pela chamada por copia/restauração. Com a copia/restauração o stub do cliente localiza o item que esta sendo indicado e o transmite ao stub do servidor. Este último o insere em algum lugar da memória e passa ao procedimento do servidor um ponteiro para ele. O servidor é capaz então de acessar o item da forma usual. Quando o procedimento do servidor devolve o controle ao stub, este transmite o item de dados (possivelmente modificado) de volta ao stub do cliente, que por sua vez utiliza para sobrescrever o parâmetro de transferência original.

Embora o mecanismo de copia/restauração freqüentemente funcione, ele pode falhar em certas situações patológicas. Pó exemplo, considere o programa da figura 7-14. quando esse programa roda localmente, ambos os parâmetros na chamada a *duploincr* são ponteiros para *a*, que é incrementado duas vezes e o numero 2 é impresso.

```

program teste(output);
var a : integer;

procedure duploincr(var x, y : integer);
begin
    x := x + 1;
    y := y + 1;
end;

begin {programa principal}
    a := 0;
    duploincr(a, a);
    writeln(a);
end.

```

Figura 14-14 Se o procedimento *duploincr* for executado remotamente, o programa falha

Agora vejamos o que acontece se duploincr é chamado como um procedimento remoto usando copia/restauração. O stub do cliente processa separadamente cada parâmetro, e assim envia duas cópias de a ao stub do servidor. O procedimento do servidor incrementa uma vez cada cópia, e ambas são devolvidas ao stub do cliente que então as restaura seqüencialmente. Primeiro, a é restaurado ao valor 1 e depois é restaurado novamente ao valor 1. Portanto, o valor final é 1 em vez do valor 2, que seria a resposta correta.

Os ponteiros geram problemas semelhantes aos parâmetros por referência. Eles são especialmente incômodos se apontam para o meio de listas ou grafos complexos ou para estruturas de dados envolvendo registros variantes. Os parâmetros de procedimentos ou funções também são difíceis de manipular, embora seja possível ao stub do servidor substituí-los por procedimentos locais ao equipamento do servidor que invoquem os procedimentos chamados na máquina do cliente através de uma RPC invertida.

Muitos sistemas de RPC simplificam logo todo o problema, pela proibição do uso de parâmetros de referência, ponteiro e parâmetros de procedimento ou funções nas chamadas remotas. Tal decisão torna a implementação mais fácil mas degrada a transparência, porque as regras para chamadas locais e remotas ficam diferentes nesse caso.

Vamos agora sair da passagem de parâmetros para outra questão de implementação: como o stub do cliente sabe quem chamar? Nas redes baseadas em conexões tradicionais, as sessões são estabelecidas entre PASSs, cada um dos quais possui um “número de telefone” fixo. Para RPC é necessário um esquema mais simples, ainda que mais dinâmico.

Birrell e Nelson (1984) descreveram um esquema envolvendo não apenas clientes e servidores, mas também um tipo especializado de sistemas de bancos de dados. No seu método, quando um servidor é inicializado, registra-se no sistema de bancos de dados pelo envio de uma mensagem contendo o seu nome (na forma de um string ASCII), seu endereço na rede (ex., um PASR, PAST ou PASS) e um identificador único (ex., um inteiro aleatório de 32 bites). Esse registro é feito fazendo com que o servidor chame um procedimento *export*, o qual é manipulado pelo stub(etapas 1 e 2).

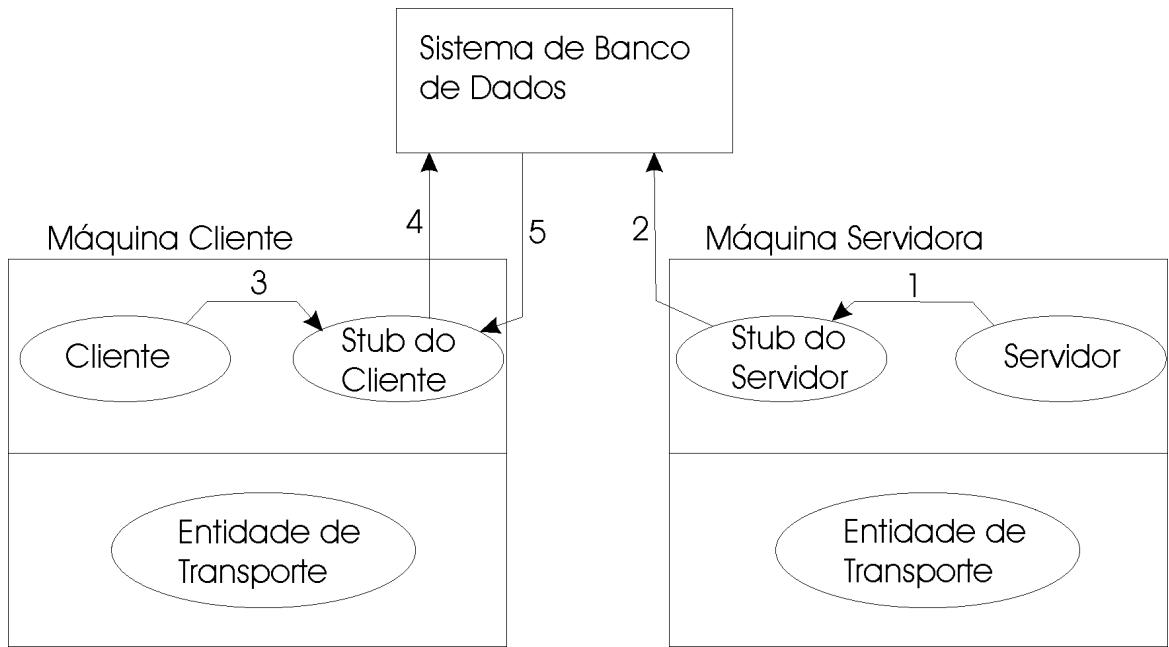


Figura 14-15 A vinculação cliente-servidor é feita através de um banco de dados

Mais tarde, quando o cliente faz sua primeira chamada (etapa 3) e seu stub fica diante do problema de localizar o servidor, o stub envia seu nome em ASCII, que é também o nome do servidor, ao sistema de bancos de dados (etapa 4). O sistema de bancos de dados devolve então o endereço de rede do servidor e o identificador único (etapa 5). Esse processo é chamado de vinculação (binding). Desse ponto em diante, o stub sabe como localizar o servidor, e assim a vinculação não é necessária nas chamadas subsequentes.

O identificador único de 32 bits é incluído e cada chamada RPC é utilizada pela entidade de transporte no equipamento do servidor para informar qual dos muitos stubs do servidor deve receber a mensagem que chega. Ele também tem outro papel. Se o servidor cair e for reiniciado, ele se registra novamente no sistema de bancos de dados, usando um novo número único. As tentativas do cliente de se comunicar com ele usando o antigo identificador único falharão, fazendo com que fiquem clientes da queda e forçando-os a fazer nova vinculação.

Outra questão fundamental da implementação é o protocolo utilizado. No caso mais simples, o protocolo da RPC pode consistir em duas mensagens: um pedido e uma resposta. Tanto o pedido quanto a resposta contêm o número único que identifica o servidor, um identificador de transação e os parâmetros. Quando envia um pedido, o stub do cliente (em alguns sistemas) pode fixar um timer. Se o timer disparar antes de a resposta disparar, o stub pode pedir ao servidor para verificar se o pedido chegou. Se não, ele pode retransmiti-lo. O objetivo do identificador de transação é permitir que o servidor reconheça e rejeite pedidos duplicados. O reconhecimento de pedidos duplicados só é possível se o servidor controlar o identificador de transação mais recente de cada cliente.

Uma última questão da implementação é o tratamento de exceções. De modo diferente de chamadas a procedimentos locais, em que nada pode dar errado, podem ocorrer muitos erros na RPC como, por exemplo, o servidor estar inativo. Se a linguagem de programação o permitir, a ocorrência de um erro de RPC não devolveria o controle ao chamador, mas surgiria uma exceção a ser manipulada por um tratador

de exceções. O projeto de mecanismo de tratamento de exceções é dependente da linguagem, mas é claramente necessário um método para distinguir chamadas mal sucedidas daquelas que obtiveram sucesso.

12.2.3 Semântica da Chamada a Procedimento Remoto

Também de modo diferente das chamadas a procedimento locais, as chamadas a procedimentos remotos estão sujeitos à perda de mensagens, quedas do servidor e quedas de clientes. Esses efeitos influenciam na semântica da RPC e no objetivo de torná-la transparente. Nesta seção, examinaremos os problemas causados por falhas no servidor, na seguinte, iremos examinar falhas nos clientes.

Considere o que acontece se o servidor falhar depois de executar o pedido, mais antes de enviar a resposta. Há pelo menos três formas possíveis de programar o stub do cliente para lidar com essa situação.

1. Simplesmente se manter para sempre esperando pela resposta que nunca virá.
2. Interromper-se e gerar uma exceção ou relatar a falha do cliente.
3. Interromper-se e retransmitir o pedido.

A primeira abordagem é semelhante ao que acontece quando um programa chama o procedimento contendo um loop infinito. Nenhum timer é usado localmente e o procedimento nunca retorna. É obrigatória a intervenção manual para encerrar o programa. Se o objetivo é tornar a semântica da RPC semelhante a chamadas a procedimentos locais, exige algo a ser dito sobre essa abordagem.

A segunda abordagem para tratar as falhas no servidor é fazer com que o stub do cliente se temporize e gere uma exceção (se a linguagem de programação suportar exceções), ou relate um erro em caos contrario. Esse método é análogo ao que acontece se um procedimento chamado recebe um erro de proteção de memória ou tenta fazer a divisão por zero. Se tiver ativado um tratador adequado, a exceção será captada e processada. Caso contrario, o programa será encerrado.

A terceira forma de abordagem convenciona que o stub do cliente temporize e retransmita o pedido. Tendo em vista que o servidor registrará outra vez um novo identificador único no sistema de bancos de dados após uma queda, a retransmissão será rejeitada pela entidade de transporte na máquina do servidor quando ele encontrar o identificador único antigo, agora inválido.

Se a entidade de transporte transmitir de volta uma resposta com erro, o stub do cliente pode desistir ou revincular e tentar novamente. Se o stub do cliente repetir a vinculação e a chamada, é concebível que a operação seja realizada duas vezes. De fato, podemos ter a repetição da operação mesmo sem quedas, se a resposta for pedida e permitirmos ao stub do cliente tentar transmitir novamente.

O problema desta abordagem reside no fato de se é aceitável a repetição da operação em execução. Se a operação não altera o estado do sistema(ou seja, é idempotente), não há mal em que ela repita-se.

Por outro lado, se a operação consistir em acrescentar um bloco ao final de um arquivo, por exemplo, então é muito importante o número de vezes que a operação é realizada (já que ele acarreta uma mudança de estado do sistema).

Se todas as operações pudessem ser modeladas em uma forma idempotente, então a terceira abordagem seria obviamente a melhor. Infelizmente, para algumas

operações isto não é possível (transferência de dinheiro de uma conta bancária por exemplo).

Como resultado destas dificuldades, a semântica exata dos sistemas de chamadas a procedimentos remotos pode ser classificadas de diversas maneiras, segundo Nelson (1981).

A mais desejável das espécies de semântica é a *exatamente uma vez*, na qual toda chamada é executada exatamente uma vez, nem mais nem menos. Esse objetivo é inatingível, porque depois da queda de um servidor não é sempre possível informar se a operação foi ou não realizada.

É possível uma forma mais frágil da semântica de exatamente uma vez, se a linguagem de programação suportar o tratamento de exceções. Nessa variação, se uma chamada retornar normalmente, significa que não ocorreu nenhuma queda e que a operação foi executada exatamente uma vez.

Se o servidor falhar ou for detectado um erro sério, não há qualquer retorno da chamada. Em vez disso, é gerada uma exceção e invocado o tratador de exceções apropriado. Com esta semântica, os procedimentos no cliente não tem de ser programados para lidar com erros, mas sim os tratores de exceção.

Um segundo tipo de semântica é o de *no máximo uma vez*. Quando é usada essa forma, o controle sempre retorna ao chamador. Se tudo correu bem, a operação terá sido realizada exatamente uma vez.

Entretanto, se foi detectada uma falha no servidor, o stub do cliente irá parar e retornar um código de erro. Não há tentativas de retransmissão. Nesse caso, o cliente sabe que a operação foi realizada zero vezes ou uma vez, não mais do que isso. A recuperação adicional é trabalho do cliente.

Uma terceira espécie de semântica para a RPC é a de *pelo menos uma vez*. O stub do cliente mantém-se tentando indefinidamente até receber uma resposta apropriada. Quando o chamador recebe o controle de volta ele sabe que a operação foi realizada uma ou mais vezes.

Para operações idempotentes, essa situação é ideal. No caso das operações não-idempotentes, podemos distinguir diversas variações. É provável que a mais útil seja que o resultado retornado é com certeza o resultado das operações finais, não o das primeiras. Se o stub do cliente utilizar um identificador de transação diferente em cada retransmissão, será possível para ele saber qual resposta pertence a qual pedido e, portanto filtrar todas elas, com exceção da ultima. Essa semântica é chamada última dentre várias.

Em resumo, o objetivo de chegar a uma semântica *totalmente transparente* dificultado pela possibilidade de desastres que afetem o servidor, mas não o cliente. Em sistemas de um único processador, essa situação não ocorre, porque uma falha que atinge o servidor também atinge o cliente.

ÓRFÃOS

Quando um cliente falha depois de iniciar uma chamada a um procedimento remoto, seu processo continua em execução no servidor. Um processo em execução sem um superior (ou “pai”) é denominado *órfão*.

Os órfãos podem causar diversos problemas, como desperdício de ciclos de CPU e bloqueamento de arquivos e outros objetos, impedindo o acesso a eles por outros processos. Finalmente, se um cliente começa outra vez uma RPC, os resultados devolvidos pelos órfãos podem provocar confusões.

Nelson descreve quatro maneira de se lidar com órfãos. Na primeira forma, chamada *extermínio*, quando um equipamento se recupera de um desastre, ele verifica se havia qualquer RPC em curso no momento da queda. Se havia, ele pede à(s) máquina(s) do servidor para eliminar os processos em execução sob seu nome.

Para usar o algoritmo de extermínio, é necessário que os stubs do cliente registrem RPCs antes de executá-las. Quando as RPCs se completam, suas entradas são eliminadas do registro de RPCs pendentes. O registro deve ser mantidos de forma persistente, que sobreviva à quedas do processador.

Os órfão podem eles próprios fazer chamadas e procedimentos remotos, gerando outros órfãos. Portanto, o algoritmo de extermínio deve ser recursivo. A técnica de extermínio pode falhar nas eliminações de todos os órfãos (como uma rede particionada com alguns nodos inalcançáveis por exemplo).

A segunda forma de eliminar órfãos é a *expiração*. Essa técnica não exige qualquer registro. Quando uma RPC é iniciada, o servidor recebe um certo prazo para completar a chamada. Se a chamada não completar-se dentro do intervalo original, o stub do servidor deve pedir ao stub do cliente um novo prazo.

Se a máquina do cliente tiver sofrido uma queda ou ter sido reiniciada, esse evento será detectado, o timer não será renovado e o servidor terá permissão para se desativar. Quando é utilizada a expiração, tudo que um cliente tem a fazer é se certificar de que decorreu um certo prazo desde a ultima queda antes de emitir a primeira RPC.

A terceira técnica para se livrar dos órfãos é chamada *reencarnaçāo*. Nesse método, o tempo é dividido em épocas numeradas seqüencialmente. Quando um cliente em recuperação falha na tarefa de exterminar seus órfãos, ele transmite a todas as máquinas o início de uma nova época. Elas reagem eliminando todos os seus processos servidores. Como todos os pedidos e respostas sempre contém o número da época em que foram iniciados, quaisquer respostas que eventualmente chegarem de órfãos carregarão um numero de época obsoleto e portanto serão detectáveis.

A quarta abordagem para detecção de órfãos é semelhante à terceira. Ela também utiliza épocas porém , em lugar de cada máquina eliminar toda atividade remota quando uma nova época é declarada, ela tenta localizar o cliente que deu partida ao servidor. Somente se esse cliente não puder ser encontrado, o servidor é eliminado. Esse algoritmo é denominado *reencarnaçāo suave*.

Até agora assumiu-se que, tão logo um órfão possa ser encontrado, ele pode ser eliminado. Porém um órfão pode estar rodando no interior de uma região crítica, ou pode ter bloqueado alguns arquivos. Sob essas circunstâncias, eliminar o órfão pode criar impasses ou inconsistências em bancos de dados.

Alem disso, um órfão já pode ter preparado seu trabalho futuro. Por exemplo, um órfão pode ter feito uma entrada em uma fila para um arquivo a ser impresso mais tarde ou para alguma outra ação futura a ser executada. Mesmo se o próprio órfão for eliminado, o trabalho enfileirado pode ser eventualmente executado. O problema da eliminação de órfãos é discutido em maiores detalhes por Shrivastava e Panziere (1982).

DISCUSSÃO DA RPC

Existem alguns aspectos com que se defronta o projetista de qualquer sistema de RPC. Eles dividem-se em quatro categorias:

- ★ Projeto para interface;
- ★ Projeto do cliente;
- ★ Projeto do servidor;
- ★ Projeto do protocolo.

No *projeto para interface*, o núcleo do assunto é até que ponto a transparência deverá ser perseguida. Dados os problemas com parâmetros e ponteiros, e a remota possibilidade de conseguir a semântica de exatamente uma vez em face de falhas, Hamilton(1984) argumentou que a transparência não deveria nem mesmo ser tentada. Ele defende o acréscimo de uma palavra chave, remoto, a frente da declaração de qualquer procedimento que possa ser chamado remotamente, para alertar o compilador e evitar confusões para o programador.

Por outro lado, embora no papel pareçam imensos, os problemas do RPC são reduzidos. Quedas são raras e a maior parte dos outros problemas pode ser manipulada pelo projeto cuidadoso de stubs e compiladores.

Uma questão importante é como os stubs são produzidos. Uma possibilidade é que sejam escritos a mão. Outra é que o compilador os produza como resultado do processo de compilação. A última possibilidade é muito conveniente mas bastante complicada, a menos que a linguagem seja muito fortemente tipada.

Outro tema é o tratamento de exceções. Se não existir nenhum mecanismo de exceção adequado o procedimento do cliente terá de testar todos os retornos de erros possíveis e estar preparado para manipulá-los, uma situação não-transparente.

Por último, vem a questão da vinculação. A forma precisa pela qual os servidores exportam seus nomes, o modo como os clientes podem selecionar uma instância específica de um servidor quando existem muitas instâncias idênticas e o momento em que a revinculação funciona são temas importantes.

O *projeto do cliente* envolve temporizações e órfãos. Os stubs do cliente devem se temporizar depois de algum tempo sem receber nenhuma resposta ou eles devem simplesmente esperar para sempre? Se interrompem-se, que ações devem executar, que semântica lhes deve ser fornecida e como os órfãos são tratados?

A principal questão referente ao *projeto do servidor* é o paralelismo. Em um extremo, sempre que chega um pedido para execução de um procedimento do servidor, um novo processo é criado e o procedimento roda como parte desse processo. Se chegarem outros pedidos de outros clientes antes que o primeiro tenha terminado, mais processos são criados e todos eles se executam em paralelo, independentes uns dos outros. No outro extremo, existe um processo único associado antes que o primeiro se tenha encerrado, ele deve esperar sua vez.

A primeira abordagem implica o ônus da criação de processos a cada RPC, mas permite a execução em paralelo das chamadas. A segunda é mais simples e rápida se houver apenas uma chamada por vez, mas não permite qualquer paralelismo se existirem chamadas múltiplas.

No *projeto do protocolo* o essencial é conseguir um alto desempenho. Houve grande controvérsia sobre esse ponto. Muitos pesquisadores acreditam que a única forma de tornar a RPC mais rápida é executá-la na parte superior da interface de pacotes bruta, com camadas de transporte e de rede nulas.

Por outro lado, algumas pessoas com mentalidade mais ligada a conexões imaginam que a RPC pode rodar de forma adequada na camada de aplicações, mesmo

se houver muitas camadas de protocolos abaixo dela. Essa é a direção que vem tomando a ISO.

Nesse modelo, existem as primitivas VINCULAR, DESVINCULAR, INVOCAR e RESULTADO, cada uma com *pedido*, *indicação*, *resposta* e *confirmação*.

Capítulo

13

Implementação da Camada de Sessão

PEQUENA INTRODUÇÃO:

A tarefa básica do nível de sessão é utilizar o serviço provido pelo nível de transporte, que simplesmente move bits de uma máquina para outra e adicionar alguns serviços orientados ao usuário.

Genericamente falando, o nível de sessão deveria assegurar que eventuais falhas ao nível de transporte fossem transparentes aos níveis superiores.

SERVIÇOS PROVIDOS PELO NÍVEL DE SESSÃO

Segundo o padrão ECMA-75, os serviços definidos para o nível de sessão proveriam facilidades de cinco tipos. Cada tipo de serviço de acesso é definido por meio de primitivas. As primitivas associadas a cada tipo de serviço são as seguintes:

FACILIDADES	NOME DA PRIMITIVA
Estabelecimento da conexão	S-CONNECT
Termino da conexão	S-RELEASE S-DISCONNECT S-ABORT
Transferência de dados de sessão	S-DATA S-EXPEDITED
Quarentena	S-QUARANTINE-DELIVER S-QUARANTINE-CANCEL
Sincronização	S-SYNC S-END-DU S-RESYNC S-TOKEN-GIVE S-PLEASE

Cada uma destas primitivas tem seus parâmetros e existe um protocolo para gerenciar as trocas entre os níveis de sessão.

UNIDADES FUNCIONAIS

Unidades funcionais são grupamentos de serviços relacionados definidos no padrão ISSO, para fins de negociação na fase de estabelecimento da conexão de sessão e para fins de referencia para outros padrões internacionais. Aqui será implementada apenas a unidade funcional denominada KERNEL, que suporta os serviços básicos de sessão, exigidos para estabelecer uma conexão de sessão, transferir dados normais e liberar a conexão de sessão.

Abaixo é relacionado à unidade funcional Kernel e os serviços a ela associados.

Unidade Funcional	Serviços	Cód.	SPDU
Kernel	Session Connection	CN	CONNECT
	Normal Data Transfer	AC	ACCEPT
	Ordery Release	RF	REFUSE
	U-Abort	FN	FINISH
	P-Abort	DN	DISCONNECT
		AB	ABORT
		AA	ABORT ACCEPT
		DT	DATA TRANSFER

O kernel constitui a unidade funcional mínima que deve ser implementada para que um sistema possa interagir com outros OSI.

PROTOCOLO DE SESSÃO

O protocolo de sessão proposto pela ISSO especifica:

- Os procedimentos para transferir dados e informação de controle entre duas entidades de sessão pares;
- Os meios para selecionar as unidades funcionais a serem usadas pelas entidades de sessão;
- A estrutura e codificação das unidades de dados do protocolo de sessão usados para transferência de dados e informações de controle.

De acordo com este trabalho a entidade de sessão comporta-se como uma máquina de estado finita. Através de um ASAP (Application Service Access Point), são recebidos os dados de um usuário do serviço de sessão, os quais são estruturados de acordo com o protocolo de sessão definido pela ISO, e enviados na forma de SPDUs a entidade de sessão par, usando os serviços providos pelo nível de transporte ao qual são entregues os TSDUs (Transport Service Data Units).

As entidades de sessão intercambiam UDPS (Unidades de Dados de Protocolos de Sessão), referidas também como SPDUs (Session Protocol Data units) que contém dados e/ou informação de controle.

A interação de uma máquina provedora de serviço de sessão com o usuário do serviço de sessão se dá por meio de primitivas do serviço de sessão (S-CONNECT request, S-DATA indication, etc).

A interação da entidade de sessão com o serviço de transporte, por ela usado, também se dá por meio de primitivas, mas usando as primitivas do serviço de transporte de quem a entidade de sessão é usuária (T-CONNECT request, T-DATA request, T-DISCONNECT request, T-DATA indication, etc).

Como neste trabalho o protocolo de transporte não é orientado à conexão e sua implementação foi assim realizada, o protocolo de sessão não troca primitivas através de SSAP, mas apenas passa a estes os SPDUS.

Formato dos SPDUS

O Formato dos SPDUS é mostrado na figura abaixo.

S1	L1	Campo de Parâmetros	Campo de Informações do Usuário
----	----	---------------------	---------------------------------

S1 – Tipo de SPDUS – Inclui os campos C1 e R1 definidos na recomendação S.62 do CCITT.

C1 – Identificador de comando.

R1 – Identificador de resposta

L1 – Comprimento do campo dos parâmetros (número binário)

Unidades PGI (Parameter Group Identifier)

PGI	L1	Campo de Parâmetros
-----	----	---------------------

PGI – Identifica o grupo de parâmetros

L1 – Comprimento do campo dos parâmetros

Campos de parâmetros:

- Simples
- Uma ou mais unidades PI (Parameter Identifier)

Unidades PI

PGI	L1	Campo de Parâmetros
-----	----	---------------------

PI – Identifica o parâmetro

L1 – Comprimento do campo dos parâmetros

Campos de parâmetros = valor do parâmetro

ANALISE DA NORMA REFERENTE AO PROTOCOLO DE SESSÃO

Consultando o “DRAFT PROPOSAL ISSO/DP 8327” cujo título é “Information Processing Systems – Open Systems Interconnection – Basic Connection session Protocol Specification”, foi analisada a mesma dedicando maior atenção ao capítulo 8 que trata da estrutura e códigos do SPDUs e ao anexo A que mostra a tabela de estados de todas as unidades funcionais.

Devido ao melhor encaminhamento das atividades, para se obter êxito na conexão micro mainframe, optou-se por implementar apenas a unidade funcional do kernel. Para obter a máquina de estados da unidade funcional kernel a partir da tabela de estados do anexo A foi necessário percorrer as tabelas seguindo ordenadamente a ocorrência de eventos e troca de estados, considerando apenas as primitivas e SPDUs envolvidas ba unidade funcional kernel. O diagrama de classes resultante do procedimento acima é mostrado na folha de papel milimetrado, ou seja, na próxima página.

Como o protocolo de transporte foi implementado de forma não orientado a conexão estando sempre ativo, algumas modificações foram realizadas na máquina de estados finitos do protocolo de sessão, estas modificações aparecem na página 07 (folha de papel manteiga).

Comparando o primeiro diagrama com o segundo, observa-se que o estado STA1B (await TCONcnf) da não existe mais e todos os eventos que aconteciam devido às primitivas do serviço de transporte também deixam de existir.

Para exemplificar observe que quando no STA1 ocorre evento de entrada SCONreq a saída será o evento de saída CN e o próximo estado serão STA2A e não o STA1B como acontecia antes.

Observando o diagrama de estados da SPM (Session Protocol Machine) Kernel com transporte não orientado a conexão procurou-se estabelecer uma comunicação mínima entre duas entidades pares de sessão, considerando a solicitação de abertura de conexão de parte o usuário de sessão, a transferência de dados e a desconexão. Para isso foi necessário, novamente, alterar a máquina de estados.

A comunicação entre as entidades de sessão é mostrada na página 09, e o correspondente diagrama de estados é mostrado na página 10.

Na página 09 são mostradas as SPDUs e primitivas envolvidas na comunicação entre entidades pares. Neste caso, primitivas são trocadas com o nível superior e SPDUs com o nível inferior (nível de transporte) que passa as SPDUs para a outra entidade correspondente.

Em relação às primitivas uma ↑ (seta) à esquerda da mesma indica que é um evento de saída da SPM, enquanto que uma ↓ (seta) à direita da mesma indica que é um evento de entrada da SPM.

Em relação as SPDUs uma ← (seta) à esquerda do mesmo indica que é um evento de saída da SPM, enquanto que uma → (seta) à direita da mesma indica que é um evento de entrada da SPM.

A transição entre os dados com os respectivos eventos de entrada e saída é mostrada para a fase de conexão, transferência de dados e desconexão.

O diagrama de estados correspondente está mostrado na página 10. Convém ressaltar que este será implementado inicialmente para ser expandido futuramente, com o objetivo de manipular com as estruturas e possibilitar experiência no trato das estruturas utilizadas implementação mais simplificada da SPM.

	Entidade A	Entidade B
Conexão	STA1	STA1
	SCONreq ↓	CN ←
	← CN	↑ SCONind
	STA2A	STA8
	AC ←	SCONrsp+ ↓
	↑ SCONcnf+	← AC
Transferência De Dados	STA713	STA713
	SDTreq ↓	DT ←
	← DT	↑ SDTind
	STA713	STA713
	DT ←	SDTreq ↓
	↑ SDTind	← DT
Desconexão	STA713	STA713
	SRELreq ↓	FN-nr ←
	← FN-nr	↑ SRELind
	STA3	STA9
	DN ←	SRELrsp+ ↓
	↑ SRELcnf+	← DN
	STA1	STA1

Comunicação entre entidades pares de sessão

Código dos SPDUs do Kernel

Nome do SPDU	Cód. Do SPDU	Código
CONNECT	CN	13
ACCEPT	AC	14
REFUSE	RF	12
FINISH	FN	09
DISCONNECT	DN	10
ABORT	AB	25
ABORT ACCEPT	AA	26
DATA TRANSFER	DT	01

Código atribuído às primitivas usadas nesta implementação:

Código	Abreviatura	Descrição
01	SCON req	S-CONNECT request primitive
02	SCON rspt	S-CONNECT (accept) response primitive
03	SCON rsp -	S-CONNECT (reject) response primitive
04	SDT req	S-DATA request primitive
05	SREL req	S-RELEASE request primitive
06	SREL rsp+	S-RELEASE (accept) response primitive
07	SREL rsp-	S-RELEASE (reject) response primitive
08	SUAB req	S-U-ABORT request primitive
09	SUER req	S-U-EXCEPTION-REPORT request primitive
20	TIM	Time Out

Dos eventos de saída:

Código	Abreviatura	Descrição
10	SCON ind	S-CONNECT indication primitive
11	SCON cnf +	S-CONNECT (accept) confirm primitive
12	SCON cnf -	S-CONNECT (reject) confirm primitive
13	SDT ind	S-DATA indication primitive
14	SPAB ind	S-P-ABORT indication primitive
15	SREL ind	S-RELEASE indication primitive
16	SREL cnf +	S-RELEASE (accept) confirm primitive
17	SREL cnf -	S-RELEASE (reject) confirm primitive
18	STD ind	S-TYPED-DATA indication primitive
19	SUA Bind	S-U-ABORT indication primitive

A seguir são mostrados os formatos dos SPDUS, com seus respectivos campos, usados na implementação do subconjunto do SPM Kernel, que está sendo implementado.

Os valores para os devidos campos de SPDU foram escolhidos baseando-se no capítulo 8 da norma da ISO referente ao protocolo de sessão. Este capítulo trata dos códigos e formatos dos SPDUS.

IDENTIFICADORES E CAMPOS DE PARÂMETROS ASSOCIADOS A CADA SPDU:

SPDU CONNECT (CN)

13	2	1	0
----	---	---	---

SPDU ACCEPT (AC)

14	2	1	0
----	---	---	---

SPDU FINISH (FN)

9	0
---	---

SPDU DISCONNECT (DN)

10	0
----	---

SPDU DATA TRANSFER (DT)

1	3	25	1	X
---	---	----	---	---

X Indica que este campo receberá o valor correspondente às condições abaixo citadas.

VALOR	CONDIÇÕES
01	Início de SPDU
02	Fim de SPDU
03	Início e fim de SPDU
00	SPDU intermediário

DOCUMENTAÇÃO DO PROGRAMA

O programa que implementa SPM é dividido em cinco partes que é programa principal e quatro procedimentos. Cada uma destas partes é descrita a seguir.

PROGRAMA PRINCIPAL

No programa principal são declaradas as duas “TASKs T e TF”; as filas SSAP-TS, SSAP-ST, ASAP-SA, ASAP-AS e FILA. Os arquivos de entrada e saída respectivamente; REM e REN e algumas variáveis globais. Depois é inicializada a SPM colocando STA = 1 e STB = 1 e disparados os processos SESSÃO e FANTASMA No final do programa as “TASKs” são encerradas.

O programa principal dispara os processos referentes aos procedimentos SESSÃO e FANTASMA.

O procedimento SESSÃO juntamente com os procedimentos PRIMIT e SPDUS satisfaz os requisitos de implementação do subconjunto da SPM Kernel. O procedimento FANTASMA tem a função de testar os demais procedimentos colocando e retirando mensagens de suas filas; imprimindo resultados de acordo com os dados de entrada. Quando o procedimento FANTASMA causa o evento NUNCA, são encerradas as duas ”TASKs” causando o fim do programa.

PROCEDIMENTO SESSÃO

Para melhor entendimento do procedimento SESSÃO é bom observar a figura que mostra os diversos níveis da ligação micro-mainframe mostradas a seguir.

Mensagens associadas a cada fila:

TSDUIN	_____	SSAP-TS
TSDUOUT	_____	SSAP-ST
SSDUOUT	_____	ASAP-SA
SSDUIN	_____	ASAP-AS

O procedimento SESSÃO aguarda que chegue uma mensagem na fila SSAP-TS oriunda do nível de usuário dos serviços se SESSÃO, ou seja, do nível de Apresentação para cima. Quando uma destas mensagens chegar, ela será removida da fila e de acordo com o seu conteúdo serão tomadas as decisões conforme o protocolo de sessão da proposta da ISO. Isto é implementado utilizando um trecho de programa similar ao mostrado a seguir.

```

1: % WAIT (SSAP-TS. QINSERT, ASAP-ASQINSERT)
CASE 1 OF
BEGIN
 1: % CHEGOU UMA MENSAGEM NA FILA SSAP-TS
 T: = REMOVE (TSDUIN, SSAP-TS);
 % TRATA A MENSAGEM, DESENCAPOULA E REAGE DE
 % ACORDO COM O PROTOCOLO DE SESSÃO KERNEL SPDUS;
 % SE OCORREU UM SPDU DE TRANSFERÊNCIA DE DADOS,
 % ENTÃO CONCATENAR OS DADOS JUNTO COM A PRIMITIVA %
 CORRESPONDENTE,
 % SENÃO PASSAR APENAS A PRIMITIVA PARA O NÍVEL
 % SUPERIOR
 ALLOCATE (SSDUDUT, TAM);
 INSERT (SSDUOUT, ASAP-SA);

2: % CHEGOU UMA MENSAGEM NA FILA ASAP-AS
 T: = REMOVE (SSDUIN, ASAP-AS);
 % TRATA A MENSAGEM, REAGE DE ACORDO COM O
 % PROTOCOLO SSESÃO KERNEL E ENCAPSULA PRIMIT;
 % CORFORME A PRIMITIVA MONTA O SPDU E INSERE NA FILA %
 PARA O TRANSPORTE
 % SE A PRIMITIVA FOR DE DADOS E OS DADOS TIVEREM      %
 MAIS DE 128 BYTES; DIVIDE A MENSAGEM EM           % PARTES ≤
 128.
 ALLOCATE (TSDUOUT, TAM);
 INSERT (TSDUOUT, SSAP-ST);
 END;
```

PROCEDIMENTOS SPDUS E PRIMIT

Uma boa maneira de implementar uma máquina de estados é usando o comando CASE. Procedendo desta forma pode-se associar cada estado a um número. Abaixo são mostradas as relações entre os estados do diagrama de estados da SPM Kernel e os números associados nos procedimentos SPDUS e PRIMIT.

Estados	Nome	Número
STA 1	Idle no Tc	01
STA 1A	await AA	02
STA 1B	await TCONonf	03
STA 1C	Idle TC con	04
STA 2A	await Ac	05
STA 3	await Dn	06
STA 8	await SCONrsp	07
STA 9	await SREL rsp	08
STA 16	await TDISind	09
STA 713	data transfer	10

Capítulo

14

Camada de Apresentação

A Camada de Apresentação relaciona-se com a sintaxe e a semântica das mensagens, conversão de códigos entre máquinas e outros serviços de conversão.

A principal tarefa desta camada é codificar dados estruturados de acordo com o formato interno do transmissor à um formato adequado para transmissão dos mesmos e depois decodificá-los de acordo com o exigido no equipamento destino.

São funções da Camada de Apresentação:

- g)Sintaxe e semântica de mensagens;
- h)Codificação de dados (Ex: ASCII, EBCDIC);
- i)Compatibilização das estruturas de representação de dados;
- j)Compressão de dados, e
- k)Autenticação.

Como exemplo de incompatibilidade de representação de dados, podemos citar os microprocessadores Intel e Motorola: os processadores da Intel (família 8086) representam binariamente os valores direita para esquerda, de forma invertida, enquanto que os microprocessadores Motorola (família 68000) os representam da esquerda para direita. Além desta diferença existem outras incompatibilidades entre processadores menos difundidos.

A conversão da forma de representação dos dados pode ser feita de duas maneiras: cada receptor decodifica os dados recebidos ou o transmissor e o receptor codificam os dados para um formato de transmissão e os decodificariam para sua representação particular.

A primeira solução é inconveniente pois o receptor deve ser capaz de identificar as diferenças entre ele e os demais transmissores com os quais comunica-se para ser capaz de adaptar os dados recebidos. Já para a segunda solução têm-se um algoritmo mais simples: o codificador e o decodificador poderiam se basear em uma estrutura padrão para transmissão e o formato de representação interna dos dados seria irrelevante. É esse o objetivo da notação ASN.1, discutida mais adiante.

A compactação dos dados faz-se necessária dado o custo de transmissão dos dados.

É também na camada de apresentação que os dados podem ser criptografados. Com isso documentos de real importância poderiam ser enviados de forma sigilosa.

Quando é feita uma criptografia dos dados, uma unidade de criptografia é inserida entre os sistemas comunicantes, sendo que inicialmente os dados são codificados de forma a serem entendidos por ambos os sistemas e depois criptografados de forma que só as entidades comunicantes possam interpretar os dados.

14.1. A NOTAÇÃO PARA SINTAXE ABSTRATA UM

ASN.1 ou *Abstract Syntax Notation One* é uma notação que permite definir tipos de dados simples e complexos e especificar valores que estes tipos podem assumir.

Os valores transmitidos podem ser de diversos tipos. Existem os tipos simples e tipos complexos estruturados, formados por vários tipos simples combinados. Cada tipo recebe uma denominação (rótulo ou “tag”) que o distingue dos demais.

Algumas das maneiras de definir novos tipos são:

- ★Uma sequência (ordenada) de tipos existentes;
- ★uma sequência não ordenada de tipos existentes;
- ★uma seleção de um dentre um conjunto de tipos.

Um mesmo rótulo pode ser atribuído a mais de um tipo cuja particular identificação será decidida pelo contexto em que for usado. Existem quatro classes de rótulos:

- l)**UNIVERSAL**: pode ser atribuído a um tipo simples ou a um mecanismo de construção, conforme especificado na tabela A.1.
- m)**APLICAÇÃO**: rótulos atribuídos a tipos por padrões específicos. Num particular padrão os rótulos da classe de APLICAÇÃO somente podem ser atribuídos a um único valor.
- n)**PRIVADA**: rótulos usados numa empresa específica.
- o)**ESPECIFICADO-POR-CONTEXTO**: interpretado de acordo com o contexto em que é usado.

Tabela 16.1 Rótulos atribuídos na classe universal

UNIVERSAL 1	tipo booleano
UNIVERSAL 2	tipo inteiro
UNIVERSAL 3	tipo string de bits
UNIVERSAL 4	tipo string de octetos
UNIVERSAL 5	tipo nulo
UNIVERSAL 6	tipo identificador de objeto
UNIVERSAL 7	tipo descritor de objeto
UNIVERSAL 8	tipo externo
UNIVERSAL 9-15	reservados para adendos ao padrão
UNIVERSAL 16	tipo SEQUENCE e SEQUENCE-OF
UNIVERSAL 17	tipo SET e SET-OF
UNIVERSAL 18-22, 25-27	tipos string de conjuntos de caracteres
UNIVERSAL 23-24	tipo hora
UNIVERSAL 28-...	Reservados para adendos ao padrão

O valor do rótulo é especificado indicando-se sua classe e o número dentro da classe (que deve ser inteiro não negativo), em notação decimal.

As regras de codificação sempre conduzem o rótulo do tipo, explícita ou implicitamente, bem como alguma representação do valor do tipo.

As regras de codificação constituem outro padrão que aplicadas ao valor de um certo tipo definido pela ASN.1 resultam na especificação completa dos valores daquele tipo durante a transferência.

Supondo que a definição de um registro de funcionário tenha sido definido e recebido a denominação de DADOS-PESSOAL. Os campos pertinentes a tal tipo de registro poderiam ser como os seguintes:

Nome: Joao P. Silva
 Cargo: Diretor
 Numero: 51
 Data de admissão: 17 de setembro de 1971
 Nome da esposa: Maria Silva
 Numero de filhos: 2

Informações sobre filhos

Nome: Rafael Silva
 Data de nascimento: 11 de novembro de 1957
 Informações sobre filho
 Nome: Suzana Silva
 Data de nascimento: 17 de julho de 1959
 A descrição formal deste registro, usando a notação ASN.1 seria:

```

Registro_pessoal ::= [APPLICATION 0] IMPLICIT SET
{ Nome,
  Cargo [0] ISO646 String,
  NumeroEmpregado,
  DataIngresso [1] Date,
  NomeEsposa [2] Name,
  Filhos [3] IMPLICIT SEQUENCE OF
              InformaçãoFilho DEFAULT {
} }

InformaçãoFilho ::= SET
{ Nome,
  DataNascimento [0] Date}

Nome ::= [APPLICATION 1] IMPLICIT SEQUENCE
{ Nome ISO646 String,
  Inicial ISO646 String,
  Sobrenome ISO646 String,

  NumeroEmpregado ::= [APPLICATION 2] IMPLICIT
INTEGER
  Date ::= [APPLICATION 3] IMPLICIT ISO 646
String -- AAAAMMDD

  O valor ou conteúdo de um registro deste tipo
seria:
  { nome "Joao", inicial "P", sobrenome "Silva"},
```

Os detalhes desta especificação serão apresentados na próxima sessão.

ASN.1 ou Abstract Syntax Notation One é uma notação que permite definir tipos complexos e especificar valores destes tipos. As regras de codificação constituem outro padrão que aplicadas ao valor de um certo tipo definido pela ASN.1 resultam na especificação completa dos valores daquele tipo durante a transferência. As regras de codificação sempre forçam a transmissão do rótulo de um tipo, implícita ou explicitamente, juntamente com a representação do seu valor.

Supondo que a definição de um registro de funcionário tenha sido definido e recebido a denominação de DADOS-PESSOAL. Os campos pertinentes a tal tipo de registro poderiam ser como os seguintes:

Nome: Joao P. Silva
 Cargo: Diretor
 Numero: 51
 Data de admissão: 17 de setembro de 1971
 Nome da esposa: Maria Silva
 Numero de filhos: 2

Informações sobre filhos

Nome: Rafael Silva
 Data de nascimento: 11 de novembro de 1957
 Informações sobre filho
 Nome: Suzana Silva
 Data de nascimento: 17 de julho de 1959
 A descrição formal deste registro, usando a notação ASN.1 seria:

```

Registro_pessoal ::= [APPLICATION 0] IMPLICIT SET
{ Nome,
  Cargo [0] ISO646 String,
  NumeroEmpregado,
  DataIngresso [1] Date,
  NomeEsposa [2] Name,
  Filhos [3] IMPLICIT SEQUENCE OF
    InformaçãoFilho DEFAULT {
  }
}

InformaçãoFilho ::= SET
{ Nome,
  DataNascimento [0] Date }

Nome ::= [APPLICATION 1] IMPLICIT SEQUENCE
{ Nome ISO646 String,
  Inicial ISO646 String,
  Sobrenome ISO646 String,

  NumeroEmpregado ::= [APPLICATION 2] IMPLICIT
INTEGER
  Date ::= [APPLICATION 3] IMPLICIT ISO 646
String -- AAAAMMDD

```

O valor ou conteúdo de um registro deste tipo seria:

```
{ nome "Joao", inicial "P", sobrenome "Silva"},
```

Os detalhes desta especificação serão apresentados na próxima sessão.

14.2 A notação usada em ASN.1

ASN.1 utiliza alguns tipos primitivos (elementos de dados) com os quais podem ser compostas estruturas mais complexas. Os tipos primitivos ou básicos são:

- BOOLEAN
- INTEGER
- BITSTRING
- OCTETSYSTRING
- NULL

Estruturas complexas são definidas agregando-se tais tipos primitivos de algumas formas previstas na ASN.1. As principais formas de estruturação de tipos compostos ou "construídos" são:

- SEQUENCE-lista ordenada de tipos
- SEQUENCE OF-iteração ilimitada de um único tipo Mecanismos de
- SET-lista não ordenada de tipos estruturação
- SET OF-interação ilimitada de um único tipo (a ordem não é importante)
- CHOICE-um campo que consiste de uma valor dentre os tipos listados.

ASN.1 define uma notação para especificação de valores e para definição de tipos. A definição de um tipo consiste numa coleção de campos que, no mais baixo nível, consiste de um identificador, uma possível etiqueta (rótulo ou "tag"), uma referência e uma possível indicação de que aquele campo é opcional (pode ser omitido).

No menor nível os campos da definição são combinados usando mecanismos de estruturação que começa com o nome do tipo da estrutura e então, em geral, uma lista de campos separados por vírgulas e envolvidos em chaves "{}". É válido usar aninhamento de estruturas.

O identificador comprehensível (letra minúscula) serve apenas para auxiliar a compreensão mas é ignorado pelas regras de codificação básicas.

O rótulo ("tag") permite que dois tipos aparentemente idênticos sejam separados tal que seu uso possa ser distinguido em construções tal como CHOICE ou SET.

Assim, um tipo será definido, segundo a ASN.1 usando-se uma das sequências de tipo válidas, que são:

```
Type ::= BuiltinType | DefinedType
```

```
BuiltinType ::= BooleanType |
    IntegerType |
    BitStringType |
    OctetStringType |
    NullType |
    SequenceType |
```

```

SequenceOfType |
SetTypeType |
SetOfType |
ChoiceType |
TaggedType |
AnyType |
CharacterSetType |
UsefulType |

```

O DefinedType especifica sequências externas usadas para referir definições de tipos e valores.

Os mais significativos tipos de primitivos serão definidos a seguir:

- O tipo **BOOLEAN** é usado para representar valores de variáveis lógicas (isto é que somente podem assumir dois estados). Quando escolher um nome para representar uma variável boolena deve ser escolhido um que descreva o estado verdadeiro, por exemplo:

Casado ::= **BOOLEAN**

- O tipo **INTEGER** é usado para modelar os valores designativos de ordem (cardinal) ou inteiros. Exemplo:

SaldoDaConta ::= **INTEGER** -- em centavos; negativo significa débito

Pode ser definido o valor mínimo e máximo, por exemplo:

DiaDoMes ::= **INTEGER** {primeiro(1),ultimo(31)}

Também pode-se usar inteiros como símbolos:

DiaDaSemana ::= **INTEGER** {Domingo(1), Segunda(2), Terça(3),
Quarta(4), Quinta(5), Sexta(6), Sábado(7)}

EstadoMarital ::= **INTEGER** {solteiro(0),casado(1),viúvo(2)}

- Usa-se o tipo **BITSTRING** para modelar dados binários de formato e comprimento qualquer (inclusive não múltiplo de 8). Pode-se usar este tipo para representar um mapa em que se indica pela posição dos bits ligados a ocorrência de algum fato, por exemplo:

DiasComSolNoMes ::= **BITSTRING**{primeiro(1),ultimo(31)} -- Um dia foi ensolarado se o bit correspondente a ele no mapa tem valor 1

- O tipo **OCTET STRING** é usado para modelar dados de qualquer formato e comprimento múltiplo de 8 bits. Exemplo:

CorpoDoPacote ::= **OCTET STRING**

OU

Sobrenome ::= **PrintableString**

A segunda forma é preferível, quando possível.

- O tipo **NULL** é usado para indicar ausência de algum elemento numa sequência, tal como no exemplo seguinte:

```

IdentificaçãoPaciente ::= SEQUENCE
{ nome          OCTET STRING,
  numeroQuarto CHOICE
    { INTEGER,
      NULL --se o paciente já saiu do
hospital-- } }

```

- O tipo **SEQUENCE OF** é usado para modelar uma coleção de variáveis cujo tipo é o mesmo, cujo número é grande e imprevisível e cuja ordem é significativa. Exemplo:

```
NomeNaçõesMembros ::= SEQUENCE OF ISO646String
-- na ordem em que se integraram
```

- O tipo **SEQUENCE** é usado para modelar uma coleção de variáveis cujo tipo e o mesmo, cujo número e conhecido e modesto e cuja ordem e significante, desde que a composição do conjunto dificilmente varie de uma versão do protocolo para outra. Exemplo:

```
NomeDosFuncionarios ::= SEQUENCE
{ presidente ISO646String,
  vicePresidente ISO646String,
  secretaria ISU646String }
```

Também pode ser usado o tipo **SEQUENCE** quando o tipo dos integrantes difere, mas o número e conhecido e modesto e cuja ordem e significante, desde que a composição do conjunto dificilmente mude de uma versão do protocolo para outra. Exemplo:

```
Credenciais ::= SEQUENCE
{ nomeUsuario ISO646String,
  password ISO646String,
  numeroConta INTEGER }
```

Se os elementos de uma sequência são em número fixo mas de vários tipos, um nome de referência deve ser atribuído a cada elemento cujo objetivo não seja completamente evidente de seu tipo. Exemplo:

```
Arquivo ::= SEQUENCE
{ TipoConteudo,
  outros AtributosArquivo,
  conteudo ANY }
```

- O tipo **SET** é usado para representar uma coleção de variáveis cujo número e conhecido e modesto e cuja ordem não e significante. Neste caso, cada variável deve ser rotulada neste contexto. Exemplo:

```
NomeUsuario ::= SET
{ nomePessoal [0] IMPLICIT ISO646String,
  nomeOrganização [1] IMPLICIT ISO646String,
  nomePaís [2] IMPLICIT ISO646String }
```

A palavra **OPTIONAL** pode ser colocada após uma variável do conjunto cujo aparecimento e opcional. Se os membros de um tipo SET são em número fixo, também deve ser atribuído um nome a cada membro cujo fim não é evidente de seu tipo. Exemplo:

```
AtributosArquivo ::= SET
{ proprietario [0] IMPLICIT
  NomeUsuario,
  tamanhoDoConteúdoEmOctetos [1] IMPLICIT
  INTEGER,
  [2] IMPLICIT
  ControleAcesso }
```

- O tipo **SET OF** pode ser usado para representar uma coleção de variáveis cujos tipos são os mesmos e cuja ordem não e significante. Ex:

```
PalavrasChave ::= SET OF ISO646String --em qualquer ordem
```

- O tipo rotulado é usado para definir um tipo de dados de uso geral, independente de aplicação que deve ser distingível de qualquer outro, por meio de sua representação. Exemplo:

```
ChaveCriptografia ::= [UNIVERSAL 24] IMPLICIT OCTET STRING --
sete octetos
```

Também podem ser usados tipos rotulados para definir um tipo de dados de uso amplo e disperso, num particular contexto de apresentação e que deve ser distingível (por meio de sua representação) de todos os outros tipos de dados usados naquele contexto de apresentação. Exemplo:

```
NomeArquivo ::= [APPLICATION 8] IMPLICIT
SEQUENCE
{ nomeDiretorio ISO646String,
  nomeRelativoArquivoDiretorio ISO646String }
```

Os rótulos específicos de um contexto são usados para distinguir os membros de um conjunto. Os rótulos numéricos devem começar de zero se sua única meta e distinguir os tipos. Exemplo:

```
RegistroCliente ::= SET
{ nome          [0] IMPLICIT ISO646String,
  endereçoPostal [1] IMPLICIT ISO646String,
  numeroConta    [2] IMPLICIT INTEGER,
  débito         [3] IMPLICIT INTEGER --em
  centavos-- }
```

Quando um particular membro do conjunto recebeu um rótulo a nível de aplicação não necessita receber rótulo específico de contexto. Ex:

```
RegistroProduto ::= SET
{ CódigoUniforme,
  descrição      [0] IMPLICIT ISO646String,
  númeroEstoque [1] IMPLICIT INTEGER }
CódigoUniforme ::= [APPLICATION 13] IMPLICIT
INTEGER
```

Rotulação específica de contexto também é usada para distinguir as alternativas de uma escolha (CHOICE). Rótulos numéricos começam em zero se seu único propósito e distinguir os diversos tipos. Exemplo:

```
AtributoDeCliente ::= CHOICE
{ nome          [0] IMPLICIT ISO646String,
  endereçoPostal [1] IMPLICIT ISO646String,
  númeroDaConta [2] IMPLICIT INTEGER,
  débito         [3] IMPLICIT INTEGER -- em
  centavos-- }
```

Os rótulos de uso privativo são usados para rotular tipos de dados que são de uso apenas no escopo de uma particular organização ou país e que devem ser distinguíveis de todos os demais tipos de dados usados por aquela organização ou país. Exemplo:

```
NúmeroCartãoAcme ::= [PRIVATE 2] IMPLICIT INTEGER
```

- O tipo **CHOICE** (seleção) é usado para representar uma variável que é selecionada dentre uma coleção de variáveis cujo número e conhecido e modesto. Cada variável do conjunto deve ser identificada por um rótulo específico do contexto. Exemplo:

```
IdentificadorArquivo ::= CHOICE
{ nomeRelativo [0] IMPLICIT ISO64String, --
  nome do arquivo
  nomeAbsoluto [1] IMPLICIT ISO64String, --
  nome do arquivo e do

  diretorio que o contem<
  numeroSerial [2] IMPLICIT INTEGER
  --identificador atribuido pelo sistema }
```

O tipo seleção é usado para representar uma variável cujo tipo e um dentre alguma particular alternativa de um **CHOICE** previamente definido.

Exemplo: a definição seguinte é possível

```
AtributosCorrentes ::= SEQUENCE
{ data-ultimo-uso AtributoArquivo,
  nome-arquivo AtributoArquivo }
```

se existir a seguinte definição:

```
AtributoArquivo ::= CHOICE
{ data-ultimo-uso INTEGER,
  nome-arquivo ISO64String }
```

- O tipo **ANY** é usado para modelar uma variável cujo tipo não é especificado ou é especificado em outro ponto usando ASN.1. Exemplo:
ConteudoMensagem ::= ANY
-- um elemento de dados cujo tipo é especificado na norma XXX
- O tipo **EXTERNAL** é usado para referenciar uma variável cujo tipo não é especificado ou que é especificado em outro local usando ASN.1.

14.3 Basic Encoding Rules

As "Basic Encoding Rules" provêem um algoritmo que especifica como um valor de qualquer estrutura (tipo) definida usando ASN.1 deve ser codificada para transmissão. O uso do algoritmo no sentido contrário permite que qualquer receptor que tinha conhecimento da definição do tipo ASN.1, possa decodificar os bits que chegam em um valor daquele tipo.

A codificação de um tipo de dados ASN.1 (usando as Basic Encoding Rules) permite que um receptor, sem conhecimento da definição de tipo, reconheça o inicio e o fim das construções (SEQUENCE, SET, etc) e os octetos representando os tipos básicos de dados (BOOLEAN, INTEGER). No uso mais simples da notação, e também possível determinar, a partir da codificação, as construções efetivamente usadas e os tipos de dados básicos.)

Quando o rótulo ASN.1 ("[]") é empregado, a codificação porta tanto o valor do rótulo como o valor identificando a construção do tipo de dado básico que foi rotulado.

Também é possível, na notação ASN.1, usar o que é chamado rotulação IMPLÍCITA a qual indica que, nas situações em que é usada, a rotulação não é necessária durante a transferência dos dados, pois pode ser percebida pelo contexto e assim, é minimizada a quantidade de octetos transferida. A alternativa IMPLICIT não deve ser usada com os tipos CHOICE, ANY ou EXTERNAL.

Exemplo:

```
ExemploTipo ::= CHOICE
{ primeiraEscolha [0] IMPLICIT INTEGER,
  segunda-escolha [1] IMPLICIT BOOLEAN }
```

Neste caso, o valor do rótulo (zero ou um) estará presente na codificação gerada pela regras de codificação mas os códigos para identificar que os tipos são INTEGER ou BOOLEAN serão suprimidos. Evidentemente, o receptor deverá conhecer a definição dos tipos para saber, pelo rótulo, qual o tipo recebido.

Cada valor de dados a ser codificado terá uma representação consistindo de:

- T tipo
- C comprimento - quando indefinido, termina com 2 octetos nulos
- V valor - pode ser em si uma série de componentes TCV

As regras de codificação sempre conduzem o rótulo do tipo, explícita ou implicitamente, bem como alguma representação do valor do tipo. Estas regras constituem outro padrão que aplicadas ao valor de um certo tipo definido pela ASN.1 resultam na especificação completa dos valores daquele tipo durante a transferência. Exemplo de uso da ASN.1 para descrição de uma estrutura de dados concernente a um registro de pessoa. O registro contém os seguintes campos:

Nome: John T. Smith

Cargo: Director

Numero: 51

Data de admissão: 17 de setembro 1971

Nome da esposa: Mary Smith

Numero de filhos: 2

Informações sobre filhos

Nome: Ralph T. Smith

Data de nascimento: 11 de novembro 1957

Informações sobre filho

Nome: Susan B. Jones

Data de nascimento: 17 de julho de 1959

Nome: John T. Smith

Cargo: Director

Número: 51

Data de admissão: 17 de setembro 1971

Nome da esposa: Mary Smith

Número de filhos: 2

Informações sobre filhos:

Nome: Ralph T. Smith

Data de nascimento: 11 de novembro 1957

Informações sobre filhos:

Nome: Susan B. Jones

Data de nascimento: 17 de julho de 1959

A descrição formal deste registro, usando a notação padronizada seria:

```

Registro pessoal ::= [APPLICATION 0] IMPLICIT SET
{ Nome,
  cargo [0] ISO646 String,
  numero NumeroEmpregado,
  dataDeIngresso [1] Date,
  nomeDaEsposa [2] Name,
  filhos [3] IMPLICIT SEQUENCE OF
    InformaçãoFilho DEFAULT { }
}
InformaçãoFilho ::= SET
{ Nome,
  dataNascimento [0] Date}
Nome ::= [APPLICATION 1] IMPLICIT SEQUENCE
{ nome ISO646 String,
  inicial ISO646 String,
  sobrenome ISO646 String,
}
NúmeroEmpregado ::= [APPLICATION 2] IMPLICIT INTEGER
Date ::= [APPLICATION 3] IMPLICIT ISO 646 String - AAAAMMDD

```

O valor ou conteúdo de um registro deste tipo seria:

```

{ nome "John", inicial "T", sobrenome "Smith",
  titulo "Director",
  numero 51,
  dataDeIngresso "19710917",
  nomeDaEsposa {nome "Mary", inicial "T", sobrenome "Smith"},
  filhos
    {{nome "Ralph", inicial "T", sobrenome "Smith} ,
     dataDeNascimento "19571111" },
    {{nome "Susan", inicial "B", sobrenome "Jones"} ,
     dataDeNascimento =19590717" } }

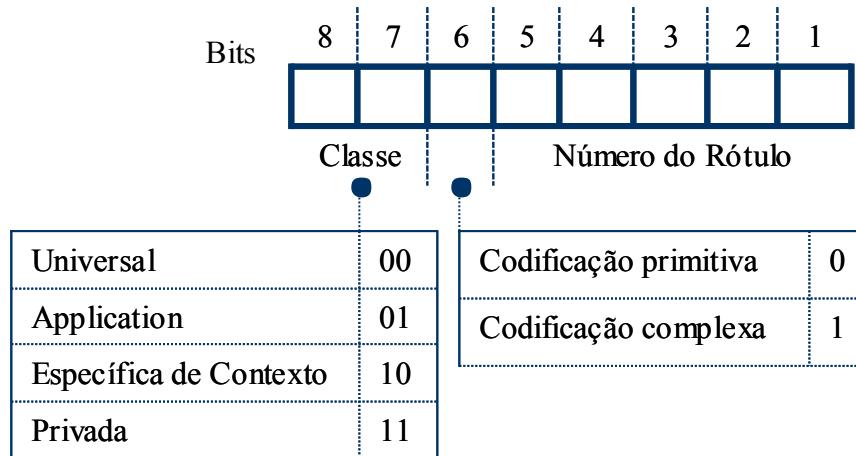
```

14.3.1 Regras gerais para codificação

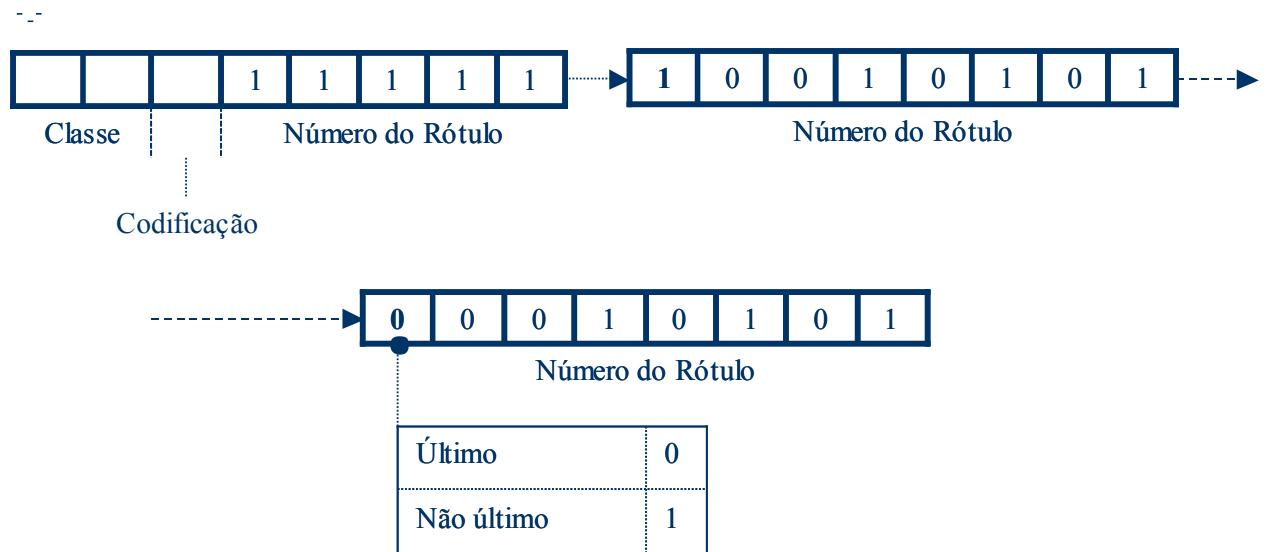
A codificação de um valor de todos os tipos, exceto os externos deverá consistir de 4 octetos, os quais deverão aparecer na seguinte ordem:

- Identificação;
- Comprimento;
- Conteúdo;
- Fim-de-contúdo.

14.3.1.1. Os octetos de identificação indicam a classe e número:



Quando o número do rótulo não puder ser expresso com apenas 5 bits, usa-se a seguinte forma :

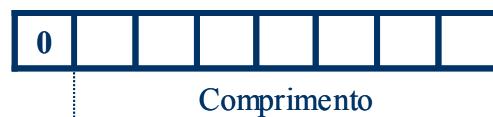


14.3.1.2. Octetos de comprimento

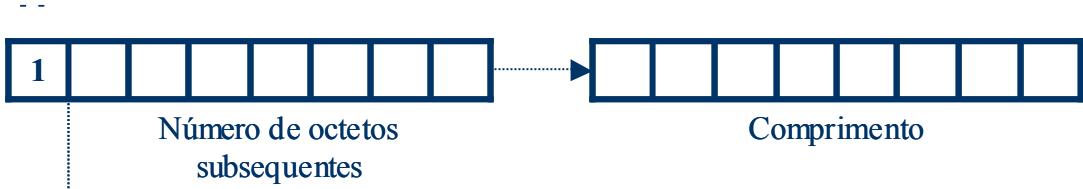
Poderão ser indicados de duas maneiras:

- Um ou mais octetos indicando o comprimento do conteúdo;

Neste caso, se o comprimento for menor do que 127 o formato será:



Caso o comprimento for maior do que 127 :



- Indicando apenas o final do campo, da seguinte maneira:



Isto indica que o final do conteúdo será sinalizado por um octeto de fim-de-contúdo.

14.3.1.3. Octetos de conteúdo

Zero ou mais octetos codificando os valores sendo transmitidos.

14.3.1.4. Octetos de fim-de-contúdo

Dois octetos zero. Este campo somente estará presente quando o comprimento do conteúdo não for conhecido ao ser iniciada sua transmissão; neste caso, no octeto de comprimento será sinalizada esta forma de delimitação de conteúdo.

14.3.2. Regras de codificação para valores

14.3.2.1. Booleano (primitivo)

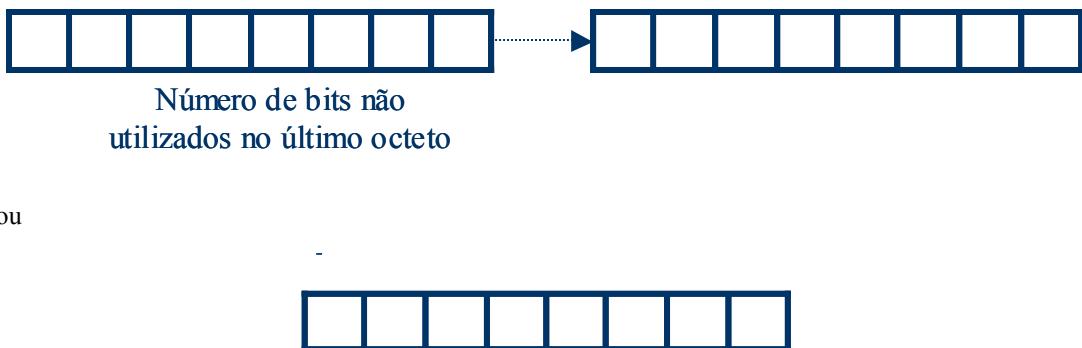
- FALSE : octeto zero;
- TRUE : qualquer valor não nulo.

14.3.2.2. Integer (primitivo)

Consiste de um ou mais octetos concatenados cujo conteúdo deverá ser o complemento de dois do número binário inteiro igual ao valor inteiro. Se mais de um octeto é usado, os primeiros nove bits não deverão ser todos nulos ou todos "1".

16.3.2.3. String binário (BIT STRING)

- Primário



Nenhum octeto subseqüente, se o string binário está vazio

- Composto

Consiste de zero ou mais codificações completas de valores de dados, cada qual sendo a codificação de um string de bits.

14.3.2.4. String de octetos (OCTET STRING)

- Primário: zero ou mais octetos;
- Composto: zero ou mais codificações completas de zero ou mais valores dados.

14.3.2.5. Valor nulo primitivo

Nenhum octeto e o octeto de comprimento será zero.

14.3.2.6. Valores seqüenciados (composto)

Contém codificação completa de um valor de dados para cada um dos tipos listados na definição ASN.1, na ordem em que aparecem na definição.

Observação: não é possível transferir codificação de valores de dados posteriores na definição ASN.1 sem que os anteriores sejam conhecidos.

16.3.2.7. Seqüência de valores (composto)

Consiste zero, um ou mais codificações completas de valores com os tipos listados na definição ASN.1. A ordem das codificações dos valores deve ser a mesma que a ordem dos valores a serem codificados ou produzidos por decodificação.

14.3.2.8. Conjunto de valores diferentes ou iguais (SET ou SET-OF) composto

Consiste na codificação completa de um valor de dados de cada um dos tipos listados na definição ASN.1, na ordem escolhida pelo emissor.

14.3.2.9. Escolha (CHOICE)

A codificação de um valor escolhido deve ser a mesma de um valor do tipo escolhido.

14.3.2.10. Seleção (SELECTION)

A codificação de um valor selecionado deverá ser a mesma que a codificação de um valor do tipo selecionado. A codificação pode ser primitiva ou composta, dependendo do tipo selecionado.

14.3.2.11. Valor rotulado

A codificação de um valor rotulado deve ser derivada da codificação completa do correspondente valor do tipo aparecendo na notação dos tipos rotulados (denominada a base da codificação).

14.3.3 Codificação do registro de pessoal exemplificado segundo as Basic Encoding Rules

<i>Rotulo</i>	<i>Comprimento</i>	<i>Conteúdo</i>		
60	8185			
	61	10		
		16	4	"John"
		16	1	"T"
		16	5	"Smith"
	A0	0A		
		16	08	"Diretor"
	42	01	33	
	A1	0A		
		43	08	"19710917"

Capítulo

15

Camada de Aplicação

15.1 Temas de Projeto da Camada de Aplicações

Faremos nesta seção uma breve introdução e uma abordagem geral das aplicações que serão estudadas neste capítulo, apenas para ter uma idéia de que tipos de questões devem ser tratadas no estudo da camada de aplicações. Esses temas incluem a transferência de arquivos, o correio eletrônico e os terminais virtuais, entre outros. Mais adiante, retornaremos a cada tópico para examiná-los em detalhes.

Além disso, também examinaremos brevemente dois blocos de construção que são oferecidos com freqüência por essas e outras aplicações. Um deles lida com gerenciamento de conexões e o outro se relaciona com a concorrência.

15.1.1 Transferência, Acesso e Gerenciamento de Arquivos

A transferência de arquivos e o acesso a arquivos remotos são duas aplicações das mais comuns em qualquer rede de computadores. Pessoas que trabalham juntas em um projeto precisam em geral compartilhar arquivos. Uma saída é ter um equipamento onde o original de cada arquivo esteja contido e fazer com que suas cópias sejam transferidas para outras máquinas conforme necessário. Outra abordagem é ter cada arquivo “ativo” na máquina em que foi criado (ou em que é mantido), e fazer com que os usuários em outras máquinas peçam cópias quando precisarem deles.

Outra situação em que é utilizada a transferência de arquivos é em uma universidade que tenha várias estações de trabalho sem unidades de disco espalhadas pelo campus juntamente com um ou mais computadores com discos de grande capacidade. Os estudantes podem se registrar em qualquer estação de trabalho e acessar seus arquivos por meio da rede. Em um projeto alternativo, as estações de trabalho podem ser equipadas com pequenas unidades de disco, permitindo aos estudantes transferirem seus arquivos para suas estações de trabalho no início de cada sessão no terminal e transferi-los de volta para a máquina principal quando tiverem terminado. Essa disposição significa que um estudante pode trabalhar em qualquer estação do campus (ou talvez até mesmo no dormitório), não apenas na estação de trabalho específica em que estão seus arquivos.

O acesso a arquivos remotos (como no caso das estações de trabalho sem unidades de disco) é semelhante à transferência de arquivos, exceto pelo fato de que somente partes de arquivos são lidas ou escritas, em vez de arquivos inteiros. As técnicas usadas para a transferência de arquivos e acesso a arquivos remotos são similares, de modo que não faremos grande distinção entre elas até chegarmos à discussão sobre a duplicação de arquivos. Além disso, o acesso a um arquivo localizado em um computador remoto que tem seus próprios usuários é bastante

diferente de acessar um arquivo em um equipamento servidor de arquivos dedicado que não tem nenhum usuário local, e portanto não faremos grande distinção entre esses dois casos. Para simplificar, iremos supor que os arquivos estão localizados em máquinas servidoras de arquivos, com os usuários em equipamentos **clientes** desejando transferir esses arquivos no todo ou em parte, para leitura e escrita.

Em contraste com as camadas de sessão e de apresentação, sobre as quais havia pouquíssimos trabalhos fora do OSI, houve uma intensa pesquisa sobre a transferência de arquivos nas universidades e na indústria. Muitos servidores de arquivos foram construídos e experimentados (Svobodova, 1984). Como resultado, nossa discussão sobre transferência de arquivos refletirá tanto o trabalho do OSI quanto às pesquisas fora do OSI.

A idéia fundamental dos servidores de arquivos mais modernos é a de um **armazenamento de arquivo virtual**, um servidor de arquivos abstrato, que esteja pronto ou em execução como um processo em um computador de tempo compartilhado. O armazenamento virtual apresenta a seus clientes uma interface padronizada e oferece um conjunto de operações padronizadas que os clientes podem executar. As transferências de e para o armazenamento virtual são protocolos padronizados. Se o servidor de arquivos real tiver uma estrutura interna diferente da que tem o armazenamento virtual, ele precisará de algum software de camada de aplicações para ocultar a verdade dos clientes e tornar visível somente a interface de armazenamento virtual. Pela padronização de uma determinada interface de armazenamento virtual, como fez OSI, é possível aos programas de aplicação acessarem e transferirem arquivos remotos sem conhecer todos os detalhes de numerosos servidores de arquivos incompatíveis.

Veremos neste capítulo algumas características do armazenamento virtual do OSI, e discutiremos ainda alguns temas relacionados com o projeto e a implementação de servidores de arquivos e protocolos de transferência de arquivos.

15.1.2 Correio Eletrônico

Quando a ARPANET foi colocada em operação pela primeira vez, seus projetistas esperavam que o tráfego de processo para processo fosse dominar. Mas estavam errados. Logo a partir do seu começo, o volume de correio eletrônico entre pessoas superou o volume de comunicações entre processos. Conquanto nem a neve, nem a chuva, nem o calor, nem a escuridão da noite possam impedir os estafetas do serviço postal de completarem com presteza suas rotinas determinadas, a capacidade da ARPANET de entregar uma mensagem de costa a costa em poucos segundos deu início a uma revolução na maneira como as pessoas se comunicam.

Evidentemente, a atração do correio eletrônico vem do fato de ele ser muito rápido. Contudo, existem outras vantagens que não são igualmente bem conhecidas. O telefone também proporciona acesso instantâneo, mas as pesquisas têm demonstrado que cerca de 75% de todas as chamadas comerciais falham em seu objetivo (“Lamento muito, mas Sr. Smith está em reunião/fora da cidade/longe de sua mesa.”). O correio eletrônico tem a velocidade do telefone sem exigir que as duas partes estejam disponíveis no mesmo instante. E também deixa uma cópia escrita da mensagem que pode ser enviada a muitas pessoas de uma só vez.

Embora o correio eletrônico possa ser visto simplesmente como um caso especial de transferência de arquivo, ele tem uma série de características específicas que não são comuns a todas as transferências de arquivo. Por um lado, os transmissores e receptores finais são quase sempre pessoas, e não máquinas. Esse fato resultou em que os sistemas de correio eletrônico fossem construídos como duas partes distintas, mas intimamente relacionadas: uma para prover a interface humana (p.ex., composição, edição e leitura da correspondência), outra para o transporte postal (p.ex., gerenciar listas de correspondência e fornecer notificação de entrega).

Outra diferença entre o correio eletrônico e a transferência de arquivos de uso geral é que as mensagens de correio são documentos altamente estruturados. Em muitos sistemas, cada mensagem apresenta um grande número de campos além do seu conteúdo. Esses campos incluem o nome e o endereço do transmissor, o nome e o endereço do destinatário, a data e a hora da postagem, uma lista de pessoas que devem receber cópias em carbono, a data de expiração, o nível de importância, certificados de segurança e muitos outros.

Muitas companhias telefônicas e PTTs estão interessadas em oferecer o correio eletrônico como um serviço padrão para companhias e assinantes individuais. Para prevenir o caos em nível mundial, o CCITT definiu em 1984 uma série de protocolos para o que ele chama de **MHS (Message Handling Systems – Sistemas de tratamento de mensagens)** em sua série de recomendações X.400. O ISO tentou incorporá-las a camada de aplicação do OSI sob o nome **MOTIS (Message-Oriented Text Interchange Systems – Sistemas de Intercâmbio de Texto Baseados em Mensagens)**, embora tal incorporação não seja inteiramente direta, dada a carência de estrutura na série X.400. Em 1988, o CCITT modificou o X.400, em uma tentativa de convergência com o MOTIS. Discutiremos o MOTIS mais a diante neste capítulo. Material adicional sobre correio eletrônico pode ser encontrado em Huffman (1987); Hutchison e Desmond (1987); Solman (1987) e Taylor (1988).

15.1.3 Terminais Virtuais

Por um motivo ou outro, a padronização dos terminais foi um fiasco completo. Praticamente todo terminal aceita certas seqüências de caracteres chamadas **seqüências de escape**, para movimentar o cursor, entrar e deixar o modo de vídeo reverso, inserir e eliminar caracteres e linhas, etc. Existem padrões do ANSI e outros para essas seqüências de escape, mas ninguém os utiliza. Cada fabricante tem suas próprias seqüências de escape, incompatíveis com as de qualquer outro fabricante. Além disso, o problema da entrada (teclado) é ainda pior do que o problema da saída (tela).

Como resultado, torna-se difícil para qualquer pessoa escrever um editor de tela que funcione com um conjunto de teclado e vídeo arbitrário. Ainda que um dado terminal tivesse uma tecla rotulada “inserir caractere”, é muito improvável que existisse um editor capaz de utilizar esta tecla com a funcionalidade para a qual ela foi concebida.

De forma semelhante, se um programa de reservas de passagens aéreas exibisse uma lista de vôos disponíveis, seria ótimo se o usuário pudesse apenas mover o cursor para o vôo desejado usando as teclas de setas no teclado ou o mouse e depois pressionar a tecla de retorno de cursor ou dar um clique em um botão do mouse para selecionar o vôo. As companhias aéreas resolvem esse problema adquirindo o computador principal, todos os terminais e o software de um único fabricante, de

modo que tudo funcione em conjunto. Entretanto, à medida que mais e mais pessoas têm acesso aos sistemas de reservas a partir de seus terminais e computadores pessoais através de redes, o problema da incompatibilidade não pode mais ser desprezado.

A abordagem do OSI para solucioná-lo é definir um **terminal virtual**, na realidade uma estrutura de dados abstrata que representa o estado abstrato do terminal real. Essa estrutura de dados pode ser manipulada pelo teclado e pelo computador, sendo seu estado atual refletido no monitor. O computador pode consultar a estrutura de dados abstrata para descobrir algo sobre a entrada pelo teclado e pode modificar a estrutura de dados abstrata para fazer com que a saída apareça na tela. Na seção sobre terminais virtuais, descrevemos com alguns detalhes como funciona essa idéia.

15.1.4 Outras Aplicações

Várias outras aplicações foram ou estão sendo padronizadas. Muitas delas pertencem a alguma indústria específica, tal como o ramo bancário. No entanto, há algumas outras que são bastante gerais e por isso merecem pelo menos algumas palavras. Entre estas últimas estão serviços de diretório, entrada de jobs remota, gráficos e telemática.

Os serviços de diretório são o equivalente eletrônico da lista telefônica: fornecem um meio para encontrar o endereço de rede de pessoas e serviços disponíveis na rede. Mesmo deixando de lado a questão de que esse serviço seria o PASR, o PAST, o PASS, o PASP ou alguma outra coisa, existe uma variedade bastante grande de questões de projeto interessantes sobre o serviço de diretório, que discutiremos neste capítulo.

A entrada de jobs remota permite a um usuário trabalhando em um computador controlar a execução de um job em outro computador. Em geral, é o usuário de um computador pessoal carregando um job em lote para execução em um grande mainframe em algum outro lugar. Em muitos casos, o programa, os arquivos de dados e as instruções de controle do job têm de ser todas coletadas, possivelmente de máquinas diversas, organizadas e executadas como uma unidade. Por fim, a saída deve ser direcionada para seus destinos apropriados.

Nem todas as aplicações são baseadas em texto. Algumas utilizam principalmente desenhos. Assim, existe a necessidade de enviar, por exemplo, esquemas de engenharia através de redes para exibição e plotagem remotas. Tem sido feito um trabalho considerável nessa área; discutiremos algumas idéias relacionadas com o assunto mais adiante.

A **telemática** é o nome coletivo para os serviços de informação pública para uso no lar ou no escritório. O **teletexto** é um sistema simples no qual uma pequena quantidade de informações pode ser enviada a um grande número de pessoas usando a difusão convencional de televisão. O **videotexto** (chamado formalmente de **viewdata**) é um serviço interativo, no qual os usuários podem acessar extensos bancos de dados públicos e realizar transações simples como fazer reservas.

15.1.5 Elementos de Serviço do OSI – ACSE e CCR

Depois de a ISO ter trabalhado nessas aplicações e em outras por um certo período, tornou-se aparente que muitas aplicações apresentavam algo em comum. Quase todas precisavam gerenciar conexões e muitas tinham de coordenar atividades entre três ou mais partes. Para evitar que cada nova aplicação fosse forçada a desenvolver esses programas mais uma vez, a ISO decidiu padronizar tais programas em blocos básicos de construção. Descrevemos em seguida dois dos mais importantes entre eles.

Elemento de Serviço do Controle de Associação

O ACSE (Association Control Service Element – Elemento do Serviço de Controle de Associação) foi projetado para gerenciar conexões, que são chamadas na camada de aplicações de associações. A figura 9-1 mostra as primitivas do ACSE.

Cada primitiva do ACSE faz um mapeamento de um para um com a primitiva de serviço correspondente da camada de apresentação. Em consequência, seria legítimo perguntar que utilidade elas têm. Seguramente, as aplicações poderiam muito bem apenas utilizar diretamente as primitivas de apresentação. Há duas razões para sua existência. A primeira é a simetria. Todas as outras camadas têm suas próprias primitivas para o estabelecimento de conexões, de forma que seria estranho se a camada de apresentação não as tivesse. A segunda razão é que alterações futuras no ACSE podem dar mais trabalho para A-ASSOCIATE, como por exemplo, a autenticação de usuários. Tal função é evidentemente imprópria para a camada de apresentação.

Compromisso, Concorrência e Recuperação

O CCR (Commitment, Concurrency and Recovery – Compromisso, Concorrência e Recuperação) é um elemento de serviço que coordena interações entre múltiplas partes de uma forma imune a falhas, mesmo em face de repetidas quedas do sistema. Praticamente todas as aplicações que necessitam operar de modo confiável utilizam o CCR.

Para ver que tipo de problema o desenvolvimento do CCR buscava solucionar, considere uma transação bancária simples. Marvin instruiu seu banco para transferir um milhão de dólares de sua conta em Amstelveen. A instrução foi passada para a câmara de compensação eletrônica que lida com todas as transferências interbancárias.

O protocolo fácil de compreender é aquele em que a câmara de compensação envia mensagens simultâneas aos dois bancos instruindo-os para debitar ou creditar na conta apropriada e depois confirmarem que o serviço foi feito. O problema surge quando um dos bancos cumpre a tarefa e o outro é incapaz de executá-la por alguma razão, tal como uma falha na rede. O que precisamos é de um protocolo que tenha sucesso completo ou não faça absolutamente nada, em lugar de um que realiza parte do trabalho e deixa o sistema bancário em um estado inconsistente.

O CCR resolve esse problema através de ações atômicas. Uma ação atômica é um conjunto de mensagens e operações que se realizam por completo ou podem ser levadas de volta ao estado original, como se nenhuma ação tivesse ocorrido.

As ações atômicas são implementadas utilizando um **comprometimento de duas fases**, uma técnica utilizada inicialmente em bancos de dados distribuídos mas que é aplicável a praticamente qualquer operação de múltiplas partes. Isto está ilustrado na Figura 17-2.

Na primeira fase, o mestre (a câmara de compensação do nosso exemplo) diz a cada escravo (outro banco) o que deseja que seja feito. Cada escravo verifica se pode executar seu pedido. Se puder, ele registra o pedido no **armazenamento estável** (algum meio, tal como um disco duplicado, reconhecido capaz de sobreviver a desastres), bloqueia os dados para que nenhum outro pedido de outros mestres possa interferir, e relata de volta o seu sucesso. Ele também registra o estado inicial dos dados (p.ex., o saldo da conta) no armazenamento estável. Por outro lado, se não puder executar o pedido, o escravo simplesmente informa de imediato a falha e não faz nada além disso.

<u>Mestre</u>	<u>Escravo</u>
Iniciar ação atômica;	
Enviar Pedido 1;	
.	
.	
Enviar Pedido n;	
Enviar mensagem "Preparar para cumprir";	
	if (ação pode ser realizada) then
	begin
	Bloquear dados;
	Armazenar estado inicial em disco;
	Armazenar pedidos em disco;
	Enviar mensagem "OK";
	end
	else
	Enviar "Falha";
if (todos os escravos disseram OK) then	
enviar mensagem "cumprir"	
else	
enviar mensagem "retornar";	
Esperar pelas confirmações;	
	if (mestre disse cumprir) then
	begin
	Fazer trabalho;
	Desbloquear dados;
	end;
	Enviar mensagem "confirmação";

Figura 15-2 *O comprometimento de duas fases*

Quando todas as respostas retornam, o mestre verifica se todos os escravos podem cumprir as tarefas que lhes são atribuídas. Se isso ocorrer, ele envia a cada escravo uma mensagem de **comprometimento**, instruindo-o para ir em frente e realizar a tarefa. Como a perda de mensagem é tratada pelas camadas mais baixas, o único detalhe que pode sair errado é um escravo sofrer uma pane antes de cumprir sua tarefa. Entretanto, ele pode recuperar-se quando retornar à atividade, examinando o armazenamento, estável para ver qual era o estado inicial e o que ele devia ter feito.

Se qualquer dos escravos informar uma falha na primeira fase, o mestre aborta a ação e avisa a todos os escravos para desbloquearem seus dados e voltarem ao estado inicial. O resultado final é que nenhum trabalho é feito e nenhum dado é alterado.

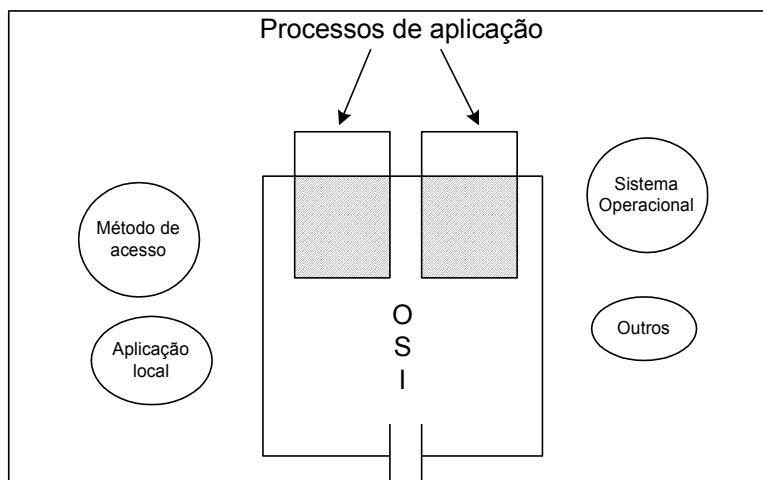
Primitiva CCR do OSI	Da parte de	Descrição
C-INICIAR	Mestre	Inicia uma ação atômica
C-PREPARAR	Mestre	Fim da fase 1; prepara para registrar
C-PRONTO	Escravo	O escravo está apto a fazer seu trabalho
C-RECUSAR	Escravo	O escravo não é capaz de fazer seu trabalho
C-REGISTRAR	Mestre	Cumpre a ação
C-RETORNAR	Mestre	Aborta a ação
C-REINICIAR	Qualquer um	Anuncia que ocorreu uma falha

Figura 17-3 *As primitivas do CCR*

O CCR fornece primitivas para cada uma das ações essenciais do comprometimento de duas fases que podem ser vistas na Figura 9-3. As seis primeiras já foram discutidas. A última é utilizada quando o mestre ou um escravo foi reinicializado depois de uma queda. Sua finalidade é restabelecer o estado da ação à sua condição inicial, para que ela possa ser repetida.

Um elemento arquitetural é definido com o propósito de esclarecer melhor o que se passa no nível da camada de aplicação. É o conceito de ASE (Application Service Element) que é o elemento de busca desta camada.

Uma entidade de aplicação pode conter um ou mais (até vários) ASEs. É importante prever meios para gerenciar as atividades destes ASEs e suas instâncias de comunicação e associações.



A camada de aplicações contém os programas do usuário (também conhecidos como aplicações ou aplicativos) que fazem o verdadeiro trabalho para o qual os computadores foram adquiridos. Esses programas utilizam os serviços oferecidos pela camada de aplicações para suas necessidades de comunicação. Entretanto, certas aplicações, tais como a transferência de arquivos, são tão comuns que foram desenvolvidos padrões a fim de eliminar a necessidade de cada empresa desenvolver

sua própria aplicação e para garantir que todos usem os mesmos protocolos.

Processos e entidades de aplicação

O processo de aplicação é responsável pelo funcionamento das aplicações distribuídas.

A **entidade de aplicação** é a parte do processo responsável por gerar comunicações. Somente a entidade de aplicação é normalizada. O processo de aplicação pode ser elaborado de maneira livre.

Elemento de serviço da camada de aplicação (ASE – Application Service Element)

Ex: FTAM usa a noção de ASE.

- ASEs são específicos a cada aplicação;
- Uma entidade de aplicação pode conter um ou vários ASEs;

Objeto de associação única (SAO – Single Association Object)

- Um SAO é o componente de uma entidade de aplicação que serve para modelar um conjunto de funções e de informações de controle referentes a uma única associação;
- Existe um ASE que é apresentado em todos SAO que é o ACSE, permitindo gerenciar a associação que une um SAO local a um SAO remoto;

Todas as regras de coordenação de funcionamento dos ASEs são gerenciados pelo

Processos e entidades de aplicação

O processo de aplicação é responsável pelo funcionamento das aplicações distribuídas.

A **entidade de aplicação** é a parte do processo responsável por gerar comunicações. Somente a entidade de aplicação é normalizada. O processo de aplicação pode ser elaborado de maneira livre.

Elemento de serviço da camada de aplicação (ASE – Application Service Element) é um agrupamento que suporta um conjunto de necessidades de comunicação.

Ex: FTAM usa a noção de ASE.

- ASEs são específicos a cada aplicação;

- Uma entidade de aplicação pode conter um ou vários ASEs;

Objeto de associação única (SAO – Single Association Object)

- Um SAO é o componente de uma entidade de aplicação que serve para modelar um conjunto de funções e de informações de controle referentes a uma única associação;
- Existe um ASE que é apresentado em todos SAO que é o ACSE, permitindo gerenciar a associação que une um SAO local a um SAO remoto;
- Todas as regras de coordenação de funcionamento dos ASEs são gerenciados pelo SACF (Single Association Control Function);
- Os ASEs contidos em um SAO colaboram, visando a prestação de serviços a um usuário. Eles vão utilizar a mesma associação estabelecida entre as entidades de aplicação homologas;

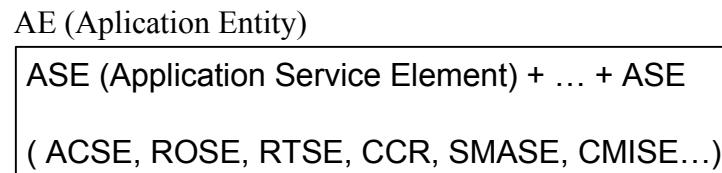
Conceito de associações múltiplas

MACF (Multiple Association Control Function) permite gerir vários SAOs, no caso de uma Entidade de Aplicação que estabelece várias associações múltiplas.

A camada de aplicação OSI é definida para suportar diferentes aplicações. No modelo de referência OSI, é a camada que dá suporte para os aplicativos (ou aplicações) dos usuários. Os aplicativos é que fazem o verdadeiro trabalho para o qual os computadores foram adquiridos. Os aplicativos utilizam-se dos serviços da camada de aplicação para suas necessidades de comunicação.

Um Processo de Aplicação (AP - Application Process) é um processo que “roda” na camada de aplicação e uma Entidade de Aplicação (AE) representa os aspectos de comunicação dos processos de aplicação.

Vários agrupamentos de funcionalidades foram definidos para preencher as necessidades das aplicações. Um elemento de serviço de aplicação (ASE - *Application Service Element*) é um agrupamento que suporta um conjunto de necessidades de comunicação de uma aplicação. Cada AE é composta por um ou mais elementos de serviço de aplicação (ASEs). Alguns exemplos de ASEs são o ACSE - *Association Control Service Element* (usado para estabelecer e gerenciar uma associação entre entidades pares de aplicação), o ROSE - *Remote Operation Service Element* (usado para a execução de operações remotas), o RTSE - *Reliable Transfer Service Element* (oferecendo um serviço de transferência confiável), o CCR - *Commitment Concurrency and Recovery* (que possibilita a execução de várias operações de uma forma atômica).



Alguns processos de aplicação (APs), tais como o FTAM - *File Transfer Access Management* (usado para a transferência e manipulação remota de arquivos) e o MHS - *Message Handling System* (usado para a transferência e manipulação de mensagens) também são utilizados por outras aplicações devido às facilidades que eles oferecem.

Elementos de Serviço para aplicações de gerenciamento

Uma aplicação de gerenciamento, como qualquer outra aplicação no Modelo OSI, utiliza-se dos serviços oferecidos pelos elementos de serviço ASEs (*Application Service Element*) da camada 7 do RM-OSI (*Reference Model - Open System Interconnection*).

Para o caso especial da aplicação de gerenciamento, dois elementos de serviço genérico são necessários: o ACSE (*Association Control Service Element*), que oferece serviços para o estabelecimento e controle das associações de gerenciamento e o ROSE (*Remote Operations Service Element*) que fornece serviços para a execução de operações remotas.

Além destes elementos de serviço, dois outros são definidos para dar suporte às aplicações de gerenciamento: o CMISE (*Common Management Information Service Element*) e o SMASE (*System Management Application Service Element*).

Na figura abaixo estão representados os elementos de serviço **ACSE**, **ROSE**, **SMASE** e **CMISE**, utilizados por uma aplicação de gerenciamento, com as respectivas primitivas de serviço.

Para garantir um perfeito entendimento do fluxo das informações de gerenciamento trocadas entre gerentes e agentes, é necessário conhecer, com detalhes, os elementos de serviço envolvidos.

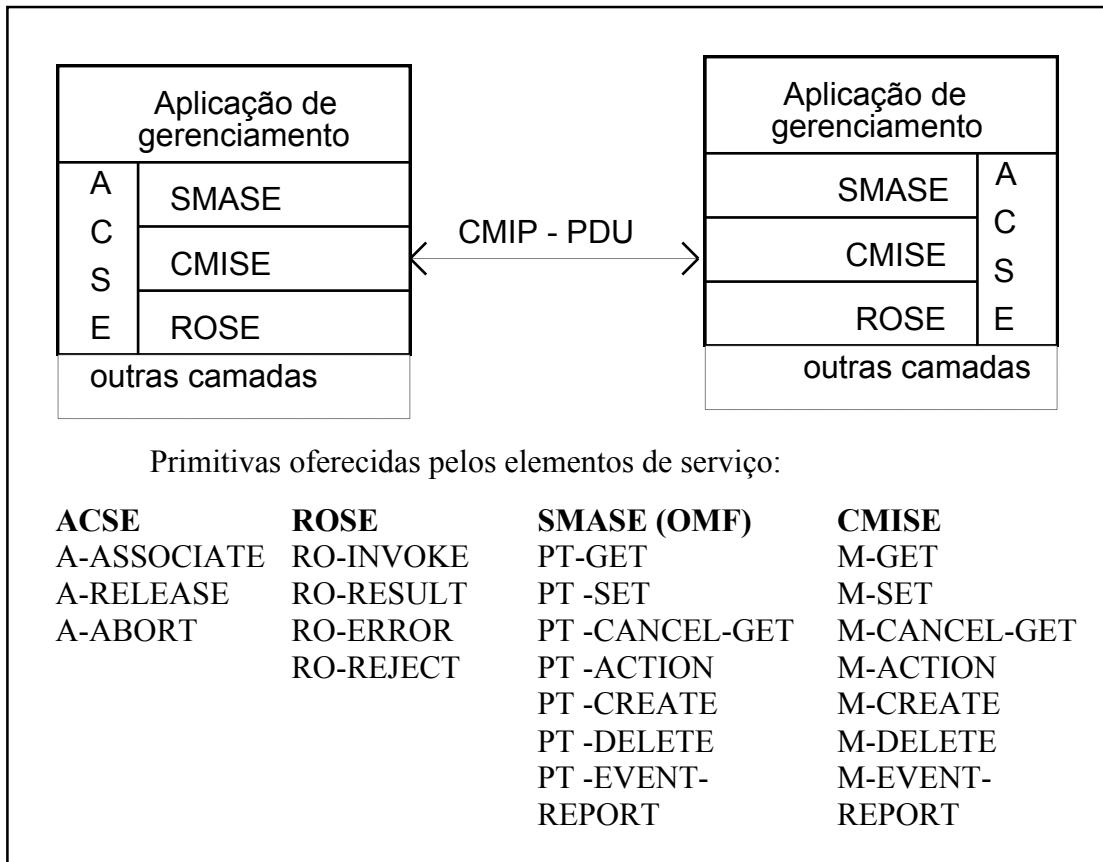


Figura 15.1 - Elementos de Serviço para o suporte às aplicações de gerenciamento

Controle de Associação (ACSE)

A função principal do ACSE é prover facilidades básicas para o controle de uma associação de aplicação entre duas entidades de aplicação, as quais se comunicam por meio de uma conexão realizada na camada de apresentação.

Os serviços ACSE são providos pelo uso do protocolo ACSE em conjunto com os serviços P-CONNECT, P-RELEASE, P-U-ABORT e P-P-ABORT da camada de apresentação.

Operações Remotas (ROSE)

O elemento de serviço ROSE oferece serviços de suporte à aplicações interativas. De uma forma geral, opera de maneira equivalente a uma chamada de procedimento remoto (RPC - Remote Procedure Call).

Este elemento de serviço é bastante útil no suporte às aplicações distribuídas.

Existe sempre uma aplicação que solicita a execução de uma operação a outra aplicação, que pode devolver ou não o resultado correspondente.

A entidade que solicita uma operação é chamada entidade invocadora e a que

recebe a solicitação é chamada entidade executora.

É importante observar que as entidades de aplicação somente podem utilizar os serviços do ROSE se já houver uma associação de aplicação estabelecida. Neste caso, podem existir 3 classes de associação:

Classe 1: somente a entidade iniciadora da associação pode invocar operações.

Classe 2: somente a entidade respondedora da associação pode invocar operações.

Classe 3: ambas as entidades, iniciadora e respondedora da associação, podem invocar operações.

São definidas, também, cinco classes de operação, que são caracterizadas pelo tipo de interação (síncrona ou assíncrona) e pelo comportamento da entidade executora. O tipo de interação diz respeito à forma como a entidade iniciadora se comporta após a solicitação de uma operação, isto é, se fica bloqueada aguardando o resultado da operação (operação síncrona) ou se fica livre para executar outras operações (operação assíncrona). O comportamento da entidade executora é modelado em termos de emissão de uma resposta à operação (sucesso ou falha da operação) ou se nenhum resultado é emitido (operação não confirmada).

O ROSE geralmente é utilizado por aplicações envolvendo o serviço de mensagens MHS (*Message Handling Service*), o serviço de diretório DS (*Directory Service*) ou o serviço de informação de gerenciamento CMIS. O usuário do ROSE dispõe de um conjunto de primitivas de serviço descritas na tabela 1.

Primitiva	Significado	Tipo de serviço
RO-INVOKE	Invoca uma operação	não confirmada
RO-RESULT	Resultado de operação bem sucedida	não confirmada
RO-ERROR	Resultado de operação sem sucesso	não confirmada
RO-REJECT-U	Rejeição iniciada pelo usuário	não confirmada
RO-REJECT-P	Rejeição iniciada pelo provedor	só indicação

Tabela 17.1 - Primitivas de serviço do ROSE

Protocolo ROSE

Para cada serviço iniciado pelo usuário do ROSE é criada uma APDU ROSE, que é mapeada para a primitiva RT-TRANSFER do RTSE ou para o serviço P-DATA de apresentação.

Fluxo da informação de gerenciamento

Uma aplicação de gerenciamento (exercendo o papel de gerente ou de agente), solicita uma associação com uma entidade par, através da primitiva de serviço de associação A-ASSOCIATE.request oferecida pelo ACSE, indicando as unidades funcionais que suporta e outras informações relevantes para a associação a ser estabelecida. A entidade par responde ao pedido através da primitiva A-ASSOCIATE.response, confirmando ou negando o estabelecimento da associação. A fase de troca de informações de gerenciamento só pode ser iniciada caso a associação tenha sido estabelecida com sucesso. Na resposta afirmativa para o estabelecimento da associação, a entidade respondedora confirma ou restringe o conjunto das

funcionalidades que podem ser utilizadas nesta instância de associação. A figura 2 mostra um exemplo de fluxo de informações em uma associação entre duas entidades de aplicação de gerenciamento.

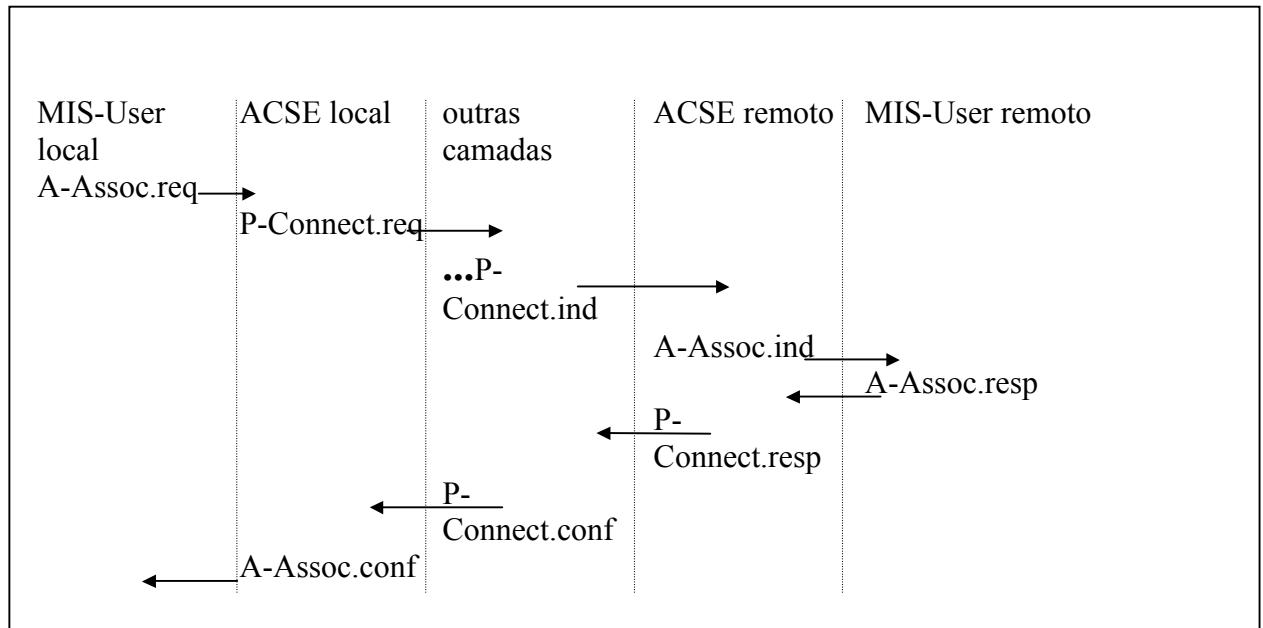


Figura 15.2- Fase de Associação entre dois MIS-Users

O elemento de serviço CMISE é composto da especificação de um conjunto de serviços denominado CMIS (*Common Management Information Service*) e de um protocolo denominado CMIP (*Common Management Information Protocol*). O CMIS é composto de um conjunto de primitivas que oferecem serviços para a criação (*M-CREATE*) e destruição (*M-DELETE*) de objetos gerenciados, para a execução de ações (*M-ACTION*) sobre objetos gerenciados e, ainda, operações de leitura (*M-GET*) e modificação (*M-SET*) de valores de atributos de objetos gerenciados. Um serviço para o cancelamento de um pedido de leitura de valores de atributos de vários objetos (*M-CANCEL-GET*) é oferecido de modo opcional, isto é, o suporte a este serviço não é obrigatório.

As operações e notificações de gerenciamento são solicitadas diretamente ao CMISE (que utiliza os serviços oferecidos pelo ROSE) ou, em sistemas que apresentam alguma funcionalidade, ao SMASE. O CMISE utiliza os serviços oferecidos pelo ROSE para transferir as operações e notificações para o MIS-User remoto. No caso de serem solicitadas ao SMASE, estas são repassadas por um serviço de *Pass-Through* para o CMISE. O serviço de *Pass-Through* para as operações e notificações de gerenciamento é oferecido pela Função de Gerenciamento de Objetos, denominada OMF (*Object Management Function*). A figura 3 mostra o cenário de comunicação entre dois MIS-Users, na solicitação de uma leitura de valor de atributo.

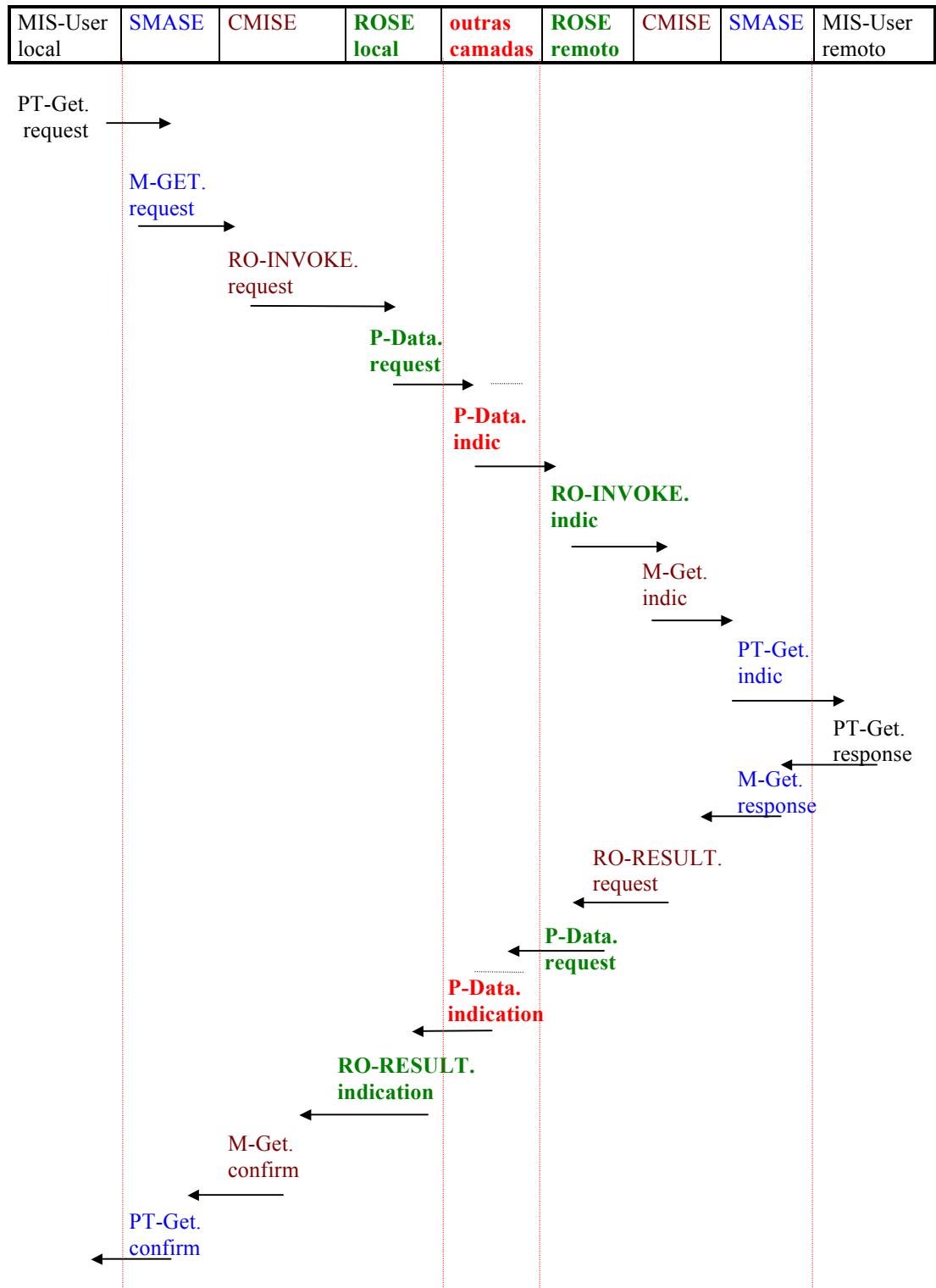


Figura 17.3 - Cenário de comunicação para a operação de leitura de valor de atributo

1. A-ASSOCIATE.request	1. M-SET.indication
2. P-DATA.request	2. PT-SET.indication
3. P-DATA.indication	3. PT-SET.response
4. A-ASSOCIATE.indication	4. M-SET.response
5. A-ASSOCIATE.response	5. RO-RESULT.request
6. P-DATA.request	6. P-DATA.request
7. P-DATA.indication	7. P-DATA.indication
8. A-ASSOCIATE.confirm	8. RO-RESULT.indication
9. PT-SET.request	9. M-SET.confirm
10. M-SET.request	10. PT-SET.confirm
11. RO-INVOKE.request	11. A-RELEASE.request
12. P-DATA.request	12. P-DATA.request
13. P-DATA.indication	13. P-DATA.indication
14. RO-INVOKE.indication	14. A-RELEASE.indication

Capítulo

16

Redes Sem Fio e QoS

1.1 Redes sem fio

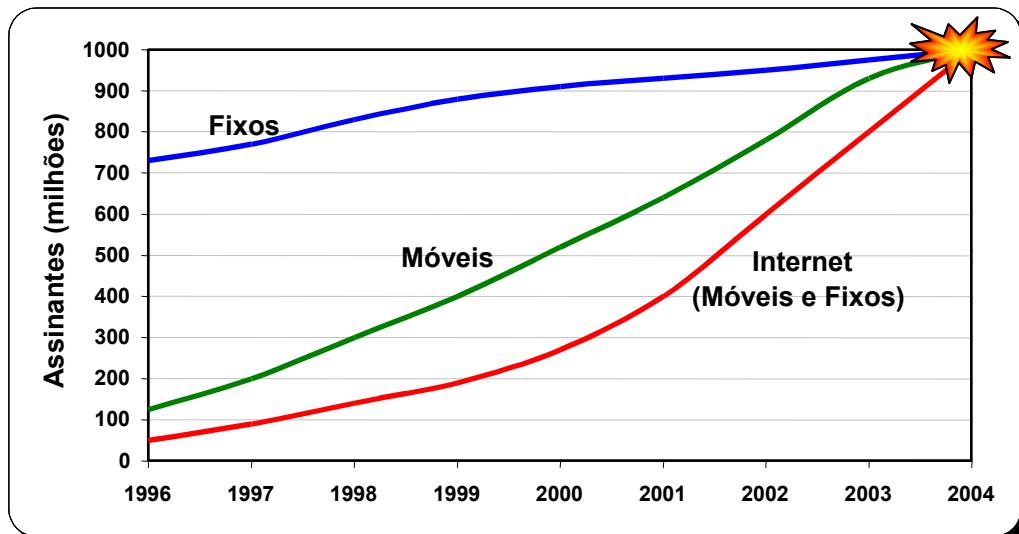
Com o extraordinário crescimento ocorrido nos últimos anos nas áreas de comunicação celular, redes locais sem fio e serviços via satélite, a tendência é que em um futuro próximo, informações e recursos possam ser acessados e utilizados em qualquer lugar e a qualquer momento. Dado o atual crescimento do segmento de computadores pessoais portáteis e PDA's (*Personal Digital Assistants*), estima-se que em poucos anos, dezenas de milhões de pessoas terão um laptop, palmtop ou algum tipo de PDA. Independente do tipo de dispositivo portátil, a maior parte desses equipamentos deverá ter capacidade de se comunicar com a parte fixa da rede e, possivelmente, com outros computadores móveis. A comunicação sem fio elimina a necessidade do usuário manter-se conectado a uma infra-estrutura fixa e em geral, estática. A esse ambiente de computação dá-se o nome de computação móvel. A necessidade de tomada de decisão em tempo real, tem levado usuários dos mais diversos setores e profissões a buscarem tecnologias que possibilitem o acesso, tratamento e uso de informações em tempo reduzido, conforme a necessidade de quem o utiliza.

O telefone celular é um ótimo exemplo, estimulou nos usuário a busca por maior comodidade, agilidade e capacidade de mobilidade exigindo cada vez mais o aumento de qualidade nos mais diversos tipos de tecnologia móvel.

A Internet é um exemplo de tecnologia fixa, que também teve que evoluir para se adequar às necessidades latentes e mutantes dos usuários atuais e potenciais. O quadro a seguir ressalta o acima exposto:

Como se pode perceber através da análise do quadro, existe uma forte tendência de crescimento de assinantes na utilização de novas formas de tecnologia, tanto fixa quanto móvel [33].

A seguir são relacionados os conceitos básicos de redes *ad hoc*, qualidade de serviço, largura de banda e padrão IEEE 802.11.



Quadro 1: Análise crescimento na utilização de novas tecnologias

1.2 Redes *Ad Hoc*

A seguir é apresentado um breve histórico das redes *ad hoc*, bem como seu conceito e classificação.

1.2.1 Histórico das redes *ad hoc*

As redes *ad hoc* tiveram seu início com as pesquisas realizadas na década de 70 pela United States Defense Advanced Research Projects Agency (U.S DARPA) com o projeto Packet radio network (PRNET).

Tal projeto tinha como intuito explorar e ampliar os usos e aplicações de redes de pacote de rádio em um ambiente tático militar com o objetivo principal de melhorar a comunicação de dados.

Em 1983, a agência americana (U.S Darpa) fez o lançamento do programa Survivable Adaptive Network, conhecido como SURAN, o qual foi desenvolvido para ampliar a base tecnológica que havia sido desenvolvida no projeto PRNET.

O SURAN foi desenvolvido com o intuito de suportar grandes redes e também desenvolver protocolos de rede adaptativos os quais tivessem a capacidade de adaptar-se às rápidas e voláteis mudanças nas condições de ambientes táticos.

O GloMo (Global Mobile Information Systems) foi o ultimo programa iniciado pela U.S DARPA em 1994, e foi desenvolvido para satisfazer os requisitos de defesa para sistemas de informações consistentes e com capacidade ampla de expansão.

Com o passar do tempo as aplicações das redes *ad hoc* ultrapassaram o uso meramente militar e passaram a servir a outros fins como busca e salvamento de pessoas, conferências entre outras aplicações.

1.2.2 O que são redes *Ad Hoc*

Ad hoc não é um termo necessariamente novo, já é utilizado em outras áreas do conhecimento como por exemplo, Administração. A denominação vem do latim e significa literalmente “para isto”, ou ainda, “apenas para este propósito”, ou seja, tem o caráter de temporalidade. Em outras palavras, *ad hoc* pode ser entendido como algo que é criado ou ainda utilizado para a resolução de um problema específico e imediato.

Geralmente, numa rede *ad hoc* não há topologia predeterminada, e nem controle centralizado, por isso, esse tipo de rede não requer uma infra-estrutura tal como backbone, ou pontos de acesso configurados antecipadamente.

O tipo de ambiente onde os usuários podem utilizar comunicação sem fio para acessar recursos distribuídos faz parte da linha de pesquisa de Rede Móvel sem fio, e é chamada de Rede Móvel *ad hoc*. Uma rede móvel *ad hoc* é uma rede onde os nodos são móveis e podem se comunicar diretamente entre si [20], através de uma placa de interface de rede sem fio que está instalada em todos os dispositivos que fazem parte desta rede e com capacidade de compartilhar recursos.

Redes móveis *ad hoc* são também conhecidas como MANET (*Mobile ad hoc Network*). Numa MANET uma rota entre dois computadores pode ser formada por vários *hops* (saltos) através de um ou mais computadores na rede, conforme pode ser observado na Figura 1.

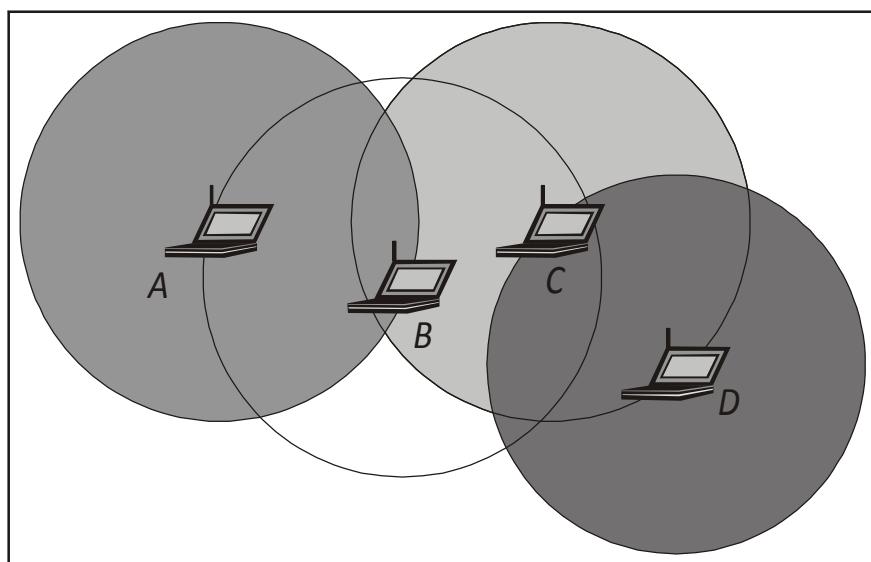


Figura 1: Exemplo de uma comunicação entre os computadores de uma MANET

Na figura 1 os círculos demonstram o alcance da comunicação das unidades móveis e, por exemplo, as mensagens de A para D devem passar pelos nodos B e C para chegar em D.

Neste tipo de rede, os nodos podem se movimentar livremente e se comunicar diretamente com outro nodo que esteja dentro de sua área de alcance. Deste modo, a topologia da rede muda freqüentemente de forma imprevisível. Isso significa que um computador intermediário I, que num determinado instante faz parte de uma rota entre dois hosts A e B, pode não fazer parte dessa mesma rota mais tarde. [20]

1.2.3 Direct Sequence Spread Spectrum (DSSS)

DSSS é a segunda forma de espalhamento de freqüência de rádio. Nesta tecnologia de transmissão um sinal de dados na estação emissora é combinado com uma taxa maior de sequenciamento de bits, o “*chipping code*”, que separa os dados do usuário de acordo com a taxa de espalhamento. Este código é um padrão redundante do bit para transmissão, aumentando a resistência do sinal para transferência. Se um ou mais bits no padrão são perdidos durante a transmissão, o dado original pode ser recuperado devido à redundância de transmissão.

A camada física DSSS usa uma seqüência de 11 bits para espalhar os dados antes de transmiti-los. Cada bit transmitido é modulado por esta seqüência. Este processo espalha a energia de rádio-freqüência em torno de uma ampla largura de banda que pode ser necessária para transmitir o dado. A carga de processamento do sistema é definido como sendo 10 vezes o logaritmo da taxa de espalhamento (também conhecido como taxa de chip) para o dado. O receptor concentra o sinal de rádio-freqüência recebido para recuperar o dado original. A vantagem desta técnica é que ela reduz os efeitos de interferência de fontes de banda estreita.

No quadro dois é apresentada uma comparação das características distintas entre as duas tecnologias, embora tenham os mesmos objetivos. Já a tabela 1, caracteriza as tecnologias.

FHSS	X	DSSS
Complicada	X	Fácil e simples
Alta potência	X	Baixa potência
Alto período de latência	X	Curto período de latência
Conexão de entrada lenta	X	Conexão de entrada rápida
Alto alcance interno	X	Curto alcance interno
Altas taxas de transferências	X	Baixas taxas de transferência

Quadro 2: Comparação entre FHSS e DSSS [35]

CARACTERÍSTICAS	FHSS	DSSS
Banda	2.4 GHz	2.4 GHz
Padrão	IEEE 802.11	IEEE 802.11b
Técnica de Modulação	Modulação de Freqüência (FM)	Modulação de Amplitude (AM)
Canal da Portadora	Envia dados sobre canais 1 MHz	Fixado em canal de 17MHz
Serviços Suportados	Dado, vídeo, voz	Dado
Máximo de Canais Independentes	15	3
Tecnologia da Indústria	HomeRF, Bluetooth	802.11b

Tabela 1: Características das tecnologias FHSS e DSSS [42]

A tecnologia FHSS está baseada em freqüências, ou seja, diferentes freqüências em diferentes instantes de tempo, pré-acordados entre emissor e receptor. Já a tecnologia DSSS, está baseada em um código que identifica o emissor e o sinal é espalhado em várias freqüências com menor amplitude.

Na seqüência estudaremos Qualidade de Serviço (QoS), fator muito importante na transmissão dos dados em uma rede.

2.5 Qualidade de Serviço (*QoS – Quality of Service*)

Em relação a QoS dois conceitos são de fundamental importância:

- * O gerenciamento eficiente dos recursos de rede para fornecer serviços adequados, passa necessariamente pela capacidade de controle dos mecanismos de tratamento de tráfego;
- * Regras previamente definidas para as políticas de rede.

Com base nos conceitos acima apresentados, pode-se definir QoS como os procedimentos e ações que tem como intuito controlar os meios de tratamentos de tráfego na rede, com objetivo de prestar serviços específicos aos usuários, no entanto sujeito às políticas da rede. Outro aspecto importante a ser explicitado, diz respeito à alocação da banda variável a qual representa uma forma de implementar a QoS, a qual deve adaptar-se às condições específicas da rede.

De outra forma, *QoS* trata de uma política para dividir mais racionalmente a banda disponível e tentar garantir, para certos serviços, a latência e largura de banda. Necessária especialmente em momentos de sobrecarga da rede, pois aqueles serviços definidos como prioritários deverão continuar funcionando perfeitamente, mesmo que isto prejudique a performance de outros de menor importância [23].

QoS é uma maneira de fornecer classes de serviços diferenciados em níveis de prioridade para fluxos de dados. Os serviços diferenciados podem ser definidos, em parte, pelos seguintes parâmetros: largura de banda, taxa de erros, atraso e variação no tempo e na seqüência da entrega das informações (*jitter*). *QoS* pode atribuir níveis de

prioridade e então tentar fornecer estes serviços diferenciados para fluxos de dados ou *hosts* específicos [13]. Não são todas as aplicações que realmente necessitam de garantias fortes e rígidas de qualidade de serviço (*QoS*) para que seu desempenho seja satisfatório.

A *QoS* oferecida em redes é um importante aspecto de implantação e operação para as redes de pacote como um todo, tornando-se fundamental para o desempenho das aplicações. A *QoS* se refere à habilidade da rede em prover o melhor serviço para cada tipo de tráfego, sobre as diferentes tecnologias de rede. Entretanto, para extrair o máximo de recursos e a garantia de um nível adequado de serviço para os diversos tipos de aplicações, é necessário implementar mecanismos que garantam a qualidade de serviços.

A *QoS* em redes independe do seu tamanho, sendo formada pelo encadeamento da qualidade em cada um dos *hops* dentro de uma rota de comunicação. As redes IP tradicionais utilizam o roteamento *Best Effort*, o qual tem seu cerne baseado no princípio das filas FIFO (*First In First Out*). O emprego de tais filas tem como objetivo produzir latência e minimizar a perda de pacotes quando as redes encontram-se congestionadas a fim de minimizar a ocorrência de *jitter* ocasionado pelo atraso entre os pacotes.

Desta forma, é possível destacar um primeiro requisito técnico que é a *QoS* por *hop*. Por se tratar do menor elemento controlável dentro de uma rede, o qual pode integrar dois ou mais canais de comunicação, é imprescindível que cada um tenha uma forma característica para tratar *QoS* nas comunicações.

Outro elemento de destaque é a possibilidade de viabilização de diversas rotas, dentro do conceito das redes Ad Hoc, existe a possibilidade de existência de vários caminhos paralelos entre dois pontos, dependendo do grau de mobilidade dos nós que fazem parte dessas rotas, os mesmos podem existir ou deixar de existir com uma relativa freqüência. O uso de várias rotas traz como vantagem a sensível redução da carga no caminho entre dois pontos, o que implica obviamente na redução da possibilidade de perdas de pacotes e do *jitter*.

Os mecanismos que possibilitam o tratamento do tráfego da rede são determinantes do nível de qualidade de serviço. Tais mecanismos têm a função de implementar e exercer controles considerando as políticas vigentes da rede, que são duas: por usuários e por aplicações.

Por analogia, pode-se usar o exemplo da Internet e das Intranets, no qual a largura de banda é de suma importância. Há um uso crescente dos benefícios gerados pela Internet e imposto pelas turbulências e urgência de uso eficaz e racional do tempo aliados à rápida obsolescência das informações.

A gigantesca quantidade de dados que precisam ser transmitidos diariamente através da Internet vem crescendo exponencialmente. Novos aplicativos, como RealAudio, RealVídeo, Internet phone e sistemas de videoconferência necessitam cada vez mais largura de banda em comparação aos aplicativos usados nos primeiros anos de Internet.

Ao passo que aplicativos tradicionais da Internet como WWW, FTP ou Telnet não toleram perda de pacotes, mas são menos sensíveis aos retardos variáveis. A maioria dos aplicativos em tempo real apresenta exatamente o comportamento oposto, pois podem compensar uma quantidade razoável de perda de pacotes, mas são, normalmente, muito críticos com relação aos altos retardos variáveis.

Isso significa que sem algum tipo de controle de largura de banda, a qualidade dos fluxos de dados em tempo real depende da largura de banda disponível no momento.

Larguras de banda baixas, ou mesmo larguras de banda melhores, no entanto, instáveis, causam má qualidade em transmissões de tempo real, com eventuais interrupções ou paradas definitivas da transmissão.

Mesmo a qualidade de uma transmissão usando o protocolo de tempo real (RTP) depende da utilização do serviço de entrega IP subjacente. Por isso, se faz urgente a aplicação de conceitos cada vez mais adequados às necessidades reais e potenciais no intuito de aumentar a probabilidade de níveis elevados de QoS específica para aplicativos em tempo real na Internet.

QoS pode ser conceituada como um conjunto de parâmetros que descrevem a qualidade de um fluxo de dados específicos. A pilha do protocolo IP básica propicia somente uma QoS que é chamada de "melhor tentativa".

Os pacotes são transmitidos de um ponto ao outro sem qualquer garantia de uma largura de banda especial ou retardo mínimo, ou seja, os equipamentos de rede enviam informações e vão progressivamente aumentando a taxa de transferência, para tentar atingir o máximo de desempenho.

No momento de tráfego de "melhor tentativa", as requisições na Internet são processadas conforme a estratégia do "primeiro a chegar, primeiro a ser atendido". Isso significa que todas requisições têm a mesma prioridade e é processada uma após a outra, sendo assim, não há possibilidade de fazer reserva de largura de banda para conexões específicas ou aumentar a prioridade de uma requisição especial.

Atualmente, há dois princípios básicos para conseguir QoS:

- p) Serviços Integrados (IntServ);
- q) Serviços Diferenciados (DiffServ);

O IntServ é um modelo baseado em reserva de recursos, ao passo que, os serviços diferenciados (DiffServ) representam uma proposta na qual os pacotes são marcados de acordo com a classe de serviços pré-determinada.

Os conceitos e características de cada um deles serão destacados na seqüência.

2.5.1 Serviços Integrados (IntServ)

O modelo de serviços integrados é caracterizado pela reserva de recursos. Antes de iniciar uma comunicação, a emissora solicita ao receptor a alocação de recursos necessária para definir-se uma boa qualidade na transmissão dos dados.

O protocolo RSVP (*Resource Reservation Protocol*) é utilizado, nesse modelo, para troca de mensagens de controle de alocação dos recursos. A alocação de recursos, diz respeito à largura de banda e ao tempo em que será mantida a conexão. Neste período de tempo, o emissor daquele serviço tem uma faixa de largura de banda disponível para transmitir seus dados.

O IntServ caracteriza-se pela alocação de recursos para dois novos tipos de serviços:

- os serviços de carga controlada para aplicações que requerem segurança e destacam o serviço de melhor esforço;
- os serviços garantidos para aplicações que necessitam de um atraso constante [46].

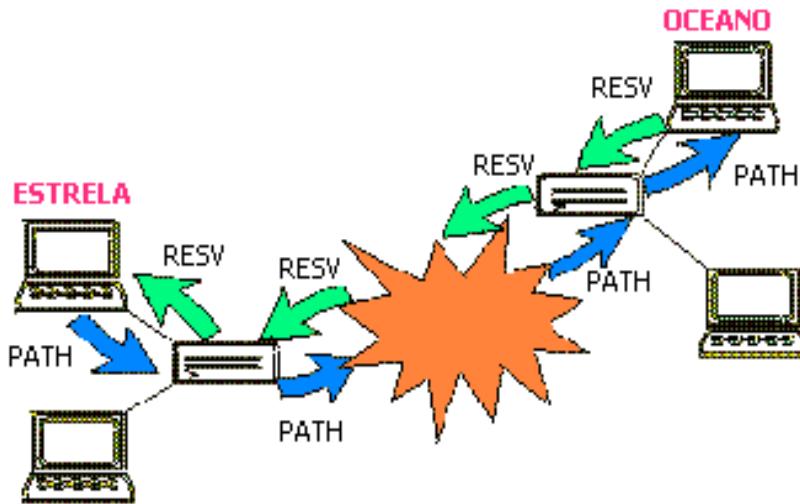


Figura 6: Sinalização RSVP [46]

Na figura acima, supõe-se que a máquina JAZZ deseja fazer comunicação de voz através da Internet (voz sobre IP) com a máquina CIRANDA. Esta aplicação requer baixo atraso e baixa variação do atraso (*jitter*) para que sejam mantidos os requisitos de qualidade. Então, JAZZ envia uma mensagem especificando as características para o tráfego (*path*) para CIRANDA. Quando a mensagem de *path* chega a CIRANDA, inicia-se o procedimento de reserva de recursos (*resv*) por todo caminho entre este dois nós da rede. Todos os roteadores entre os dois pontos passam pelo processo de alocação de recursos e qualquer um deles pode rejeitar a solicitação, informando para JAZZ que a solicitação não foi aceita. Caso todos os roteadores tenham condições de disponibilizar os recursos solicitados, é alocada a largura de banda e *buffer* necessários para a aplicação.

Durante a transmissão dos pacotes, são feitas classificações nos roteadores para cada fluxo, colocando-os em filas específicas para a aplicação. Como o controle é feito basicamente nos roteadores, isso exige grande capacidade de processamento, armazenamento e bons algoritmos para tratamento de filas, ou seja, aumenta o grau de complexidade nos roteadores.

O IntServ foi projetado para estabelecer QoS "end-to-end", ou seja, aumentar a probabilidade de que a qualidade será adequada, exatamente como foi estabelecido na configuração original, entre os dois pontos que estão conectados através deste sistema.

O que ocorre, é que várias conexões virtuais são estabelecidas como em Frame-Relay e ATM, e os roteadores armazenam em tabelas o estado de cada conexão.

Portanto, para estabelecer um canal IntServ entre dois pontos, o aplicativo do usuário irá perguntar para o roteador se ele pode oferecer o recurso que a aplicação necessita no momento, e este roteador irá perguntar ao seu próximo, que irá perguntar ao próximo, até chegar na outra ponta. Se todos tiverem recurso disponível para oferecer, o canal é estabelecido; caso contrário, não é feita a conexão IntServ [49].

Embora a idéia e o conceito sejam muito interessantes, o IntServ não funciona bem em grandes redes, pois nenhum roteador é capaz de guardar uma tabela com muitas conexões IntServ e comparar a cada pacote IP, em contrapartida, é mais recomendado para redes pequenas em função de diversos fatores, como por exemplo o custo.

A seguir, apresenta-se com mais detalhe o Protocolo RSVP.

2.5.1.1 Protocolo RSVP (*ReSerVation Protocol*)

Resource ReSerVation Protocol (RSVP) pode ser conceituado como um protocolo de sinalização que opera no nível três reservando recursos em redes habilitadas para tratar da QoS, tem ainda a função de coordenar a aplicação de mecanismos de tratamento de tráfego em múltiplos dispositivos, com o objetivo de obter a maior QoS fim-a-fim.

De forma sucinta, o RSVP opera da seguinte maneira: envia uma requisição de pedido de recursos em somente uma direção e trata de forma distintiva o emissor e o receptor, mesmo que paralelamente a aplicação atue tanto como emissora quanto receptora. É importante fazer uma ressalva, esse protocolo não transporta dados de aplicações, ou seja, executa somente em *background*, não tendo a função de enviar dados.

O RSVP foi desenvolvido para que o *host* possa se beneficiar de *QoS* específicas conforme a aplicação a qual se destina. No entanto, o RSVP não tem a função de atuar como roteador e sim para operar com protocolos unicast e multicast, tanto atuais, quanto futuros. A diferença fundamental é que os protocolos de roteamento determinam o destino dos pacotes enviados enquanto o RSVP tem a função específica de primar pela *QoS* desses pacotes que são enviados conforme o roteamento determina.

O mecanismo de funcionamento do RSVP, opera da seguinte maneira: é transmitida uma mensagem PATH que descreve para os receptores o dado a ser transmitido e o caminho que o mesmo deverá tomar. Como resposta, os receptores enviam mensagens RESV que segue o mesmo caminho de volta da mensagem PATH, informando que o perfil de tráfego solicitado é suportado pelo receptor que está respondendo. Os tipos de informações suportadas por esse tipo de protocolo são:

- a) Informação de classificação (meio de identificação do tráfego);
- b) O tipo de informação requerido da rede (IntServ);
- c) Parâmetros quantitativos descrevendo o tráfego;
- d) Informações de política (identificando o usuário e a aplicação correspondente).

Os dispositivos que contam com o mecanismo RSVP extraem informações de políticas da rede e fazem sua verificação. Tais solicitações de recursos podem não ser aceitas diante dessas políticas de rede evitando que a mensagem siga o seu destino. Quando esse fato ocorre, é enviada uma mensagem de rejeição. Quando as mensagens não são rejeitadas por políticas de rede ou indisponibilidade de recursos dá-se a reserva a qual significa que os recursos que foram solicitados serão alocados para o tráfego pertencente ao fluxo solicitante. Quando dá reserva, uma requisição do RSVP QoS é encaminhada para dois módulos de decisão local:

- *Admission Control* – determina se o nó tem recurso suficiente para suportar o QoS solicitado.
- *Policy Control* – determina se o usuário tem permissões administrativas para fazer a reserva.

Se ambas as verificações procederem positivamente são estabelecidos parâmetros *packet classifier* e também na interface do enlace com o objetivo de obter o QoS necessário. No entanto, se alguma rejeição ocorrer é retornado uma notificação

de erro pelo protocolo RSVP para o processo de aplicação que originalmente solicitou a requisição.

2.5.2 Serviços Diferenciados (*Diffserv*)

A idéia fundamental da Arquitetura de Serviços Diferenciados (DiffServ) é definir um conjunto pequeno de mecanismos que possam ser implementados nos nós da rede e que suportem uma grande variedade de serviços oferecendo QoS com escalabilidade; sem estado para cada fluxo e sinalização a cada nó. O tratamento oferecido por um nó compatível com essa abordagem, é efetuado sobre uma agregação de fluxos. Não mais para um fluxo individual. Os pacotes que não pertencem a nenhum fluxo contratado recebem o serviço padrão, o qual é equivalente ao serviço de melhor esforço, atualmente oferecido.

O diffserv se difundiu devido a sua praticidade e baixo custo, e hoje é amplamente usado. Toda sua operação é baseada em marcar um campo no cabeçalho dos pacotes IP, que não era utilizado até então. Este campo de 8 bits, hoje chamado de TOS (Type of Service) no Ipv4, conforme pode ser observado na Figura 7, e TCF (Traffic Class Field) no Ipv6 – conforme pode ser observado na Figura 8 – recebe um valor numérico que os roteadores usam para saber que tratamento dar a este pacote. Sua marcação pode ser feita usando como regra o endereço de origem ou destino, porta de origem ou destino, horário, MAC Address, e outras, além de poder combinar várias delas [49].

Quando um pacote chega a uma máquina ele será analisado pelo “Classificador”, e dependendo do conteúdo de seu campo de TOS ou TCF ele será encaminhado para uma fila. Com isto, os roteadores deixam de ter filas únicas do tipo *FIFO* (primeiro a entrar é o primeiro a sair) e passam a ter diversas filas, que recebem pacotes de acordo com sua marcação. Os roteadores que são habilitados a usar diffserv, tratam o pacote de maneira diferenciada, inserido-o em uma fila de maior ou menor prioridade, e os roteadores que não têm este recurso, apenas o ignoram, e repassam o pacote com o campo intacto [47].

Dessa forma, embora o diffserv não seja um mecanismo que garanta 100% a qualidade da rede, ele é muito fácil e prático de implementar, bastando um sistema operacional para microcomputadores PCs comuns, com um microcomputador. Ele também é escalável, pois os roteadores não têm que manter em memória e processar o estado de cada pacote, devem apenas colocar em filas de acordo com marcações simples. A figura abaixo ilustra as diferenças e similaridades no preenchimento dos campos do cabeçalho do Ipv4 e Ipv6:

IPv4

Version	IHL	TOS	Total Length		
Identification			Flag	Fragment Offset	
Time to Live	Protocol	Header Checksum			
Source Address					
Destination Address					

Figura 7: Modelo Cabeçalho do Ipv4 [44]

IPv6

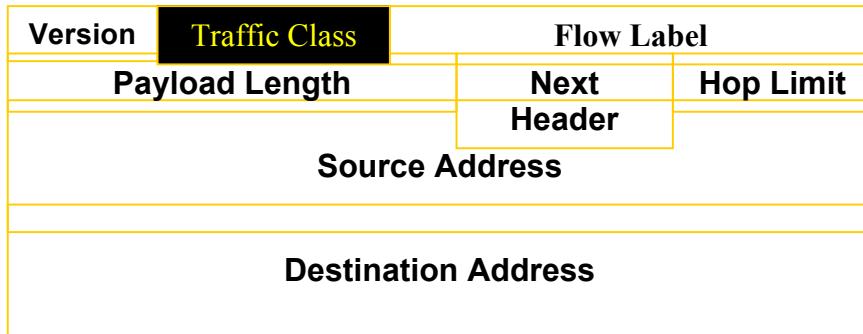


Figura 8: Modelo Cabeçalho do Ipv6 [44]

A identificação da agregação de fluxos no interior de um domínio DiffServ é efetuada através da marcação de um novo campo chamado DS (*Differentiated Services*). O campo DS é obtido pela renomeação do campo TOS, no caso do Ipv4, ou no campo do *Traffic Class*, no caso do Ipv6.

Formato do byte DS:

DSCP (Differentiated Services Code Point)

CU (Currently Undefined)

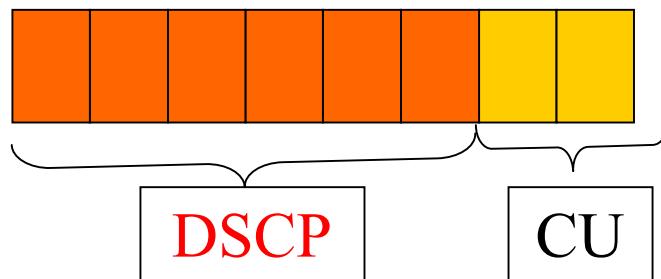


Figura 9: Formato do byte do Differentiated Service (DS) [44]

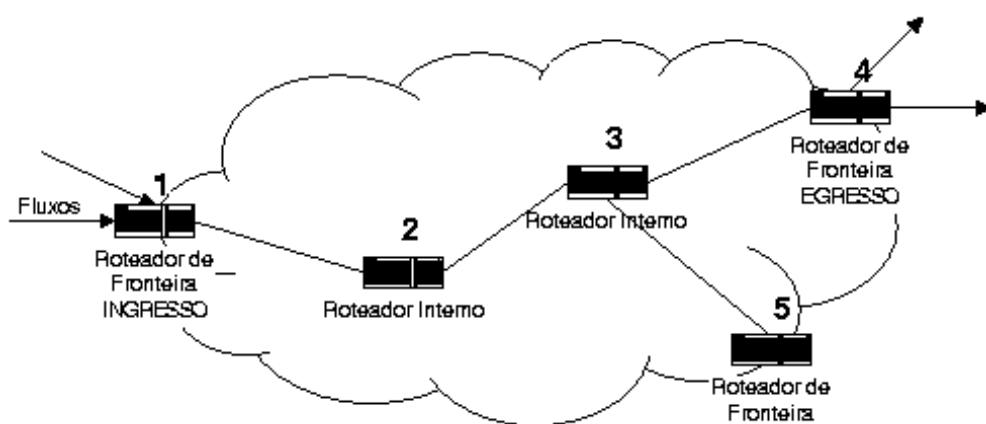


Figura 10: Domínio de Serviços Diferenciados – DS [48]

A figura acima representa um domínio DiffServ. No entorno do domínio estão localizados os nós de fronteira 1, 4 e 5. Nesses nós, os fluxos são agregados e estas agregações são encaminhadas pelo nós interiores 2 e 3. Esta arquitetura somente provê a diferenciação de serviços em uma única direção do fluxo de tráfego, sendo portanto assimétrica. Os nós de fronteira assumem, dependendo do sentido do tráfego tratado, o papel de nós de ingresso ou egresso. Na ilustração da Figura 10, o nó 1 representa um roteador de ingresso e o nó 4 um roteador de egresso. [48]

No modelo DiffServ o encaminhamento das agregações é feito segundo uma política de Comportamento por Nó (*Per-Hop Behavior*, PHB). O resultado de uma transmissão, isto é, o serviço fim a fim é conseguido através da combinação dos PHBs entre os domínios ao longo do caminho. Atualmente, existem duas propostas de PHB para a implementação de Serviços Diferenciados: o Encaminhamento Expresso (*Expedited Forwarding* – EF) e o Encaminhamento Assegurado (*Assured Forwarding* – AF). O primeiro oferece a garantia de uma taxa mínima de transmissão pré-configurada em cada nó compatível com a proposta de Serviços Diferenciados que o suporte. Assim, o PHB EF pode ser utilizado para a obtenção de um serviço fim-a-fim com baixa perda, baixo retardo, baixa variação do retardo (*jitter*) e banda passante assegurada através de um domínio DS. Entre os pontos finais de um domínio DS, o Serviço de Encaminhamento Expresso apresenta-se como um caminho virtual dedicado que possui a taxa de transmissão pré-configurada. [48]

O PHB de Encaminhamento Assegurado provê quatro classes, cada uma com uma certa quantidade de recursos (memória e banda passante) alocada. Em cada classe, os pacotes podem ser marcados com uma de três preferências para descarte. Havendo congestionamento, a preferência de descarte determina a importância relativa entre os pacotes de uma mesma classe. Não há o reordenamento de pacotes de um mesmo fluxo quando estes pertencem a uma mesma classe.

A abordagem Serviços Diferenciados também possui alguns problemas. Dentro eles, podemos citar a baixa granulosidade (não possui um bom mecanismo de distinção frente à diversidade de fluxos presentes) e o número limitado de classes.

2.5.2.1 Classes

Um aspecto diffserv é a capacidade de trabalhar com classes de serviços - tecnicamente conhecido como CBQ (Class Based Queue). Usando classes poderíamos dizer que todo download (ftp) feito por usuários não pode ultrapassar uma velocidade pré-determinada. Dizer também que o e-mail não é prioritário, ou seja, seus pacotes podem esperar pacotes de navegação web passarem para serem transmitidos.

Como se tudo isto não bastasse, ainda pode-se configurar as classes de forma a uma emprestar seus recursos a outra em momentos de ociosidade, mas tomá-las de volta se preciso. Outro exemplo seria um provedor de acesso a Internet com um *link* de 10Mbit.

Poderiam ser criadas duas classes principais, uma de 8Mbit no máximo para os clientes, e 2Mbit de velocidade máxima para o escritório do provedor. Entretanto, durante a noite não há ninguém trabalhando no escritório, e os 2Mbit destinados a ele seriam desperdiçados.

Para resolver este problema, pode-se configurar as classes para que a classe do escritório possa emprestar sua banda disponível para os usuários do provedor depois do horário de expediente.[47]

2.5.2.2 Utilidade e aplicabilidade

Os procedimentos relativos a QoS nas redes *ad hoc*, são especialmente úteis e necessários para:

- Eliminar ou minimizar congestionamentos da rede;
- Dar melhor suporte para largura da banda;
- Modelar e priorizar os aspectos ligados ao tráfego de informações, entre outros.

A adoção de procedimentos da QoS tem como função, a priori, melhorar as condições de utilização das redes *ad hoc*, pois quanto melhor a condição da rede, mais fácil prever problemas ou gargalos e gerir as mudanças necessárias ao longo do tempo.

2.6 Aspectos relevantes para latência e largura de banda

O assunto relacionado a enfileiramento, apresenta-se como de substancial importância, dada sua influência sobre QoS. A utilização do esquema de enfileiramento tem a função de proporcionar determinados níveis de qualidade através da capacidade de transmissão ou da largura disponível de bandas, considerando determinados fluxos de tráfegos e a consequente latência de pacotes sobre a QoS.

esquema de enfileiramento tem a função de proporcionar determinados níveis de qualidade através da capacidade de transmissão ou da largura disponível de bandas, considerando determinados fluxos de tráfegos e a consequente latência de pacotes sobre a QoS.

Existem basicamente duas formas para tratar desse assunto. A primeira é o levantamento de informações que destaque a capacidade de transmissão de um parâmetro para acompanhar a efetividade da transmissão, e o nível do seu fluxo. A segunda dá-se através da mensuração de latência de pacotes em determinados fluxos. O nível de latência tem relação com a capacidade da interface, a largura de banda existente e a seqüência na qual o fluxo da interface é servido.

Outro elemento a ser considerado e que afeta a latência é a variação desta entre os pacotes de um mesmo fluxo, ou seja, o *jitter* e paralelamente o tipo de mecanismo utilizado para enfileiramento.

2.6.1 Largura de Banda

Largura de Banda ou *Bandwidth* é a faixa de freqüência necessária para a sinalização. A diferença entre a freqüência mais alta e mais baixa de uma banda é medida em Hertz [21].

A largura de uma banda de freqüência eletromagnética representa a velocidade com que os dados fluem, seja numa linha de comunicação ou no barramento de um computador. Quanto maior a largura de banda, mais informações podem ser enviadas num dado intervalo de tempo, pode ser expressa em bits por segundo (bps), bytes por segundo (Bps) ou ciclos por segundo (Hz) [22].

As considerações a respeito da largura de banda são importantes para a Qualidade de Serviço, pois determinam a qualidade do tráfego das informações. Quanto maior a largura da banda, menor serão os problemas relativos ao tráfego. Logo, esta deve ser trabalhada no intuito de otimizar as aplicações e requisitos para atender as necessidades de gerenciamento da rede. No entanto, perde-se em eficiência quando há compartilhamentos. Em contra partida, os procedimentos relativos à largura da banda devem ser tratados de forma específica, respeitando as exigências de funcionalidade da rede. Permite ganho de eficiência nas redes ad hoc, como por exemplo, agilidade no tráfego de informações.

2.6.2 Tráfego de informação

Diferentes tipos ou intensidade de tráfegos são gerados se for considerada a diversidade das aplicações e usuários que, simultaneamente, demandam diferentes tipos de recursos de rede. Esta adversidade pode ser percebida através da variação das taxas de cargas a qual são submetidas às redes. Tais variações se dão em função das diferenças de tamanhos de pacotes e horários. Para que exista eficiência na alocação de recursos no sentido de atender as aplicações e usuários, quatro requisitos se fazem necessários:

- *Bandwidth* (largura de banda) – taxa de tráfego necessária para uma aplicação;
- *Perda de Pacotes* - percentual de pacotes de dados perdidos;
- *Latência* – indica o atraso que uma aplicação pode tolerar para transmissão de pacotes;
- *Jitter* – variação da latência.

O tráfego de informações também afeta a qualidade de serviços. Quando o recebimento de informação for maior do que sua capacidade de liberação, ocorre o congestionamento. Para lidar com este problema existem três abordagens.

1. Mecanismos de enfileiramento – quando ocorre o congestionamento as informações são armazenadas na memória dos dispositivos, sendo que o maior problema é a variação de latência.
2. Descarte de pacotes – quando há o congestionamento e a informação se torna obsoleta, os pacotes são descartados.
3. Classificação do tráfego – determinação prévia da ordem de prioridade da informação.

Deste modo, a qualidade do serviço é determinada em parte pela capacidade de utilização otimizada dos mecanismos de controle de tráfego. Para tanto existem as formas de classificação, onde os pacotes são alocados em diferentes filas. Tais filas se valem de algoritmos específicos que tem como função determinar os caminhos mais adequados os quais os pacotes devem percorrer.

A configuração da rede deve levar em consideração dois pontos principais, no sentido de manter a qualidade do serviço:

- Informação de classificação para que os dispositivos separem o tráfego em diferentes filas;

- Filas e algoritmos de filas, que aloquem os pacotes em diferentes filas.

Basicamente, existem duas formas de QoS:

a) Reserva de recursos (Serviços Integrados): os critérios de políticas de gerenciamento de banda determinam a maneira como os recursos serão alocados para atender os níveis necessários de QoS das aplicações.

b) Priorização (Serviços Diferenciados): Novamente os critérios adotados nas políticas de gerenciamento de banda, determinam de que maneira o tráfego é classificado e quais recursos devem ser alocados para atendê-lo.

O tráfego de informações afeta substancialmente a qualidade de serviços. Por isso, é necessário ter mecanismos refinados de controle que classifique, enfileire e escalone de forma distintiva os diversos tipos de tráfego adequando-se aos padrões de qualidade nos serviços prestados pela rede *ad hoc*.

2.7 Padrão para Redes Locais sem Fio - IEEE 802.11

IEEE 802.11 representa o primeiro padrão para produtos de redes locais sem fio de uma organização independente e internacionalmente reconhecida, a IEEE (*The Institute of Electrical and Electronics Engineers, Inc.*).

A maioria dos produtos para redes locais sem fio disponíveis no mercado é solução proprietária *spread spectrum* operando nas bandas de freqüência ISM (*Industrial, Scientific, and Medical*) de 902 MHz e 5,85 GHz [32]. Soluções proprietárias são tipicamente customizadas e obrigam os usuários finais a adquirirem produtos de uma única fornecedora de equipamentos. No entanto, assim que os produtos são introduzidos em conformidade a um padrão, os usuários podem escolher dentre um número de fornecedores produtos que são compatíveis, favorecendo a competição. Interoperabilidade, baixo custo e estímulo de demanda de mercado são algumas das vantagens que soluções baseadas em padrões oferecem.

O padrão IEEE 802.11 define protocolo para dois tipos de redes:

- Rede *ad hoc*: uma rede simples onde as comunicações são estabelecidas entre múltiplas estações em uma certa área de cobertura, sem o uso de um ponto de acesso ao servidor. O padrão especifica os critérios que cada estação deve observar, de modo que todos tenham acesso ao meio sem fio. Ele provê métodos para gerenciar requisições para o uso da mídia, garantindo máxima performance para todos os usuários.
- Rede Infra-Estruturada (Cliente/Servidor): usa um ponto de acesso para controle da alocação de tempo de transmissão para todas as estações e habilita estações móveis a realizar *roaming* (ou seja, a capacidade que uma estação sem fio tem de sair de sua rede e migrar para outra). O ponto de acesso é usado para manipular tráfego do rádio móvel para o *backbone*, com ou sem fio, da rede cliente/servidor. O ponto de acesso direciona os dados entre as estações e outras estações sem fio direcionam para o servidor da rede.

O comitê de padrões IEEE 802 formou o Grupo de Trabalho do Padrão para Redes Locais sem fio 802.11 (*802.11 Wireless Local Area Networks Standards Group*)

em 1990. Este grupo desenvolveu uma padrão global para equipamento de rádio e redes operando na banda de freqüência não-licenciada de 2,4 GHz para taxas de dados de 1 e 2 Mbps. O padrão foi concluído em junho de 1997 e não especifica tecnologia de implementação, mas simplesmente especificações para a camada Física e de Controle de Acesso ao Meio (MAC – Medium Access Control). O padrão permite aos fabricantes de equipamentos de redes locais sem fio construir equipamentos de redes interoperacionais.

2.7.1 Camada MAC (Medium Access Control)

A camada MAC é projetada para poder assegurar autenticação e serviços sensíveis a tempo. Adotou-se o mecanismo de acesso múltiplo com prevenção de colisão CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*), onde este trabalha da seguinte maneira: Uma estação solicitada para transmitir escuta no meio. Se o meio está ocupado com a transmissão da outra estação, então a estação atrasa sua transmissão por um determinado período de tempo [24].

A especificação da camada MAC para 802.11 tem semelhanças ao padrão 802.3. O protocolo para 802.11 usa o CSMA/CA. Este protocolo evita colisões (ao contrário de detectar uma colisão, como fazia o algoritmo usado no 802.3), porque é difícil detectar colisões em uma rede de transmissão rádio-freqüência.

A sub-camada física usa um algoritmo (CCA – *Clear Channel Assentment*) para determinar se o canal está livre. Este é acompanhado pela medida da energia de rádio-freqüência na antena e determina a força do sinal recebido. Se a força do sinal recebido é menor que um determinado valor, o canal é declarado livre e a camada MAC recebe o estado de canal livre para transmissão de dados. Caso contrário, as transmissões de dados são adiadas de acordo com as regras do protocolo.

Esta técnica é mais seletiva, uma vez que ela verifica se o sinal é do mesmo formato de um transmissor 802.11. O protocolo CSMA/CA (*Carrier Sense Multiple Access / Collision Avoidance*) permite minimizar colisões pela transmissão dos quadros RTS (*Request To Sender*) e CTS (*Clear To Sender*), dos dados e do sinal ACK (*Acknowledge*). No método CSMA/CA pode ocorrer colisões e esse método não garante a entrega correta dos dados. Com isso, uma estação após transmitir um quadro, necessita de um aviso de recebimento que deve ser enviado pela estação receptora. A estação remetente aguarda um tempo (*timeout*) pelo aviso do recebimento do quadro por parte das estações destino. Caso este aviso não chegue no tempo considerado, a estação origem realiza novamente a transmissão do quadro.

Para melhorar a transmissão de dados, o protocolo DFWMAC acrescenta ao método CSMA/CA com reconhecimento, um mecanismo opcional que envolve a troca de quadros de controle RTS e CTS antes da transmissão de quadro de dados [20]. A comunicação funciona da seguinte forma:

- A comunicação é estabelecida quando um dos nós sem fio envia um sinal RTS, sendo que este possui as funcionalidades de reservar o meio para a transmissão do quadro de dados, e de verificar se a estação destino está pronta para receber o quadro de dados, sendo que nesta última funcionalidade a estação destino pode estar operando no modo de economia de energia (*modo power save*). O sinal RTS inclui ainda o destinatário e o

o tempo reservado para o envio do quadro de dados. Esta informação é utilizada por outros nodos que podem escutar tanto o transmissor como o receptor para atualizar seus *Net Allocation Vectors* (NAV) – um temporizador que está sempre decrementando se não for zero. A um nodo não se permite que inicie transmissão se o seu NAV for diferente de zero. Refere-se a utilização de NAV a fim de determinar se o canal está livre ou ocupado como o mecanismo de **Detecção Virtual de Portadora**. Como nodos que podem ouvir ou o transmissor ou o receptor evitam enviar informação durante a transmissão do quadro de dados, a probabilidade de sucesso é grande.

Entretanto, o aumento de probabilidade é obtido às custas de um *overhead* considerável envolvendo as trocas dos quadros CTS e RTS, que pode ser significativo para quadros de dados pequenos.

Os serviços sem contenção, providos por PCF, são gerenciados pelo *Point Coordinator* (PC), que deve operar junto aos pontos de acesso da célula da rede sem fio e que define qual estação tem o direito de transmitir. Essencialmente, o PC pergunta a cada estação se esta deseja transmitir – um mecanismo típico de *polling*.

O PCF controla quadros durante o **período livre de contenção** – *Contention Free Period* (CFP) –, que é seguido por um **período de contenção**, controlado pelo mecanismo DCF, anteriormente descrito. O PC ganha o controle de CFP e tenta manter o controle pelo período inteiro porque espera um tempo menor para transmitir do que estações utilizando procedimento de acesso DCF. Este intervalo de tempo, pouco menor que DIFS, mas maior que SIFS, é denominado de PIFS (*PCF InterFrame Space*).

No início de cada CFP, após esperar um tempo PIFS, o PC informa, através de um

quadro *beacon*, qual o tempo total de CFP e quando ocorrerá novamente. Todas as estações devem colocar como valor de NAV a duração total de CFP a fim de evitar com que alguma estação tome o controle do meio durante este período. Após um intervalo de tempo SIFS, o PC pode enviar dados, requisitar que estações envoiem dados, confirmar dados que recebeu, (empregando apenas um quadro MAC quando possuir mais de uma tarefa a ser realizada) ou acabar com o CFP. Durante CFP, somente estações que estiverem na lista de *polling* do PC podem transmitir, mas todas as estações podem receber dados. O PC pode terminar CFP a qualquer momento, mesmo que o tempo de duração informado no quadro *beacon* não tenha se esgotado, baseado no tráfego disponível e no tamanho de sua lista de *polling*.

Todos as estações que estão na lista de *polling* e, portanto, que respondem a pedido de transmissão do PC, ignoram o mecanismo de detecção virtual de portadora (temporizador NAV), verificando apenas se o meio está livre após um intervalo SIFS. Estações fora da lista que recebem quadros de dados devem confirmá-los segundo as regras do procedimento DCF. O mecanismo básico da camada MAC é conhecido como *Distribution Coordination Function* (DCF), que provê acesso múltiplo assíncrono e com contenção, com detecção de portadora e prevenção de colisão CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*). Opcionalmente, suportado pelo DCF, pode ser oferecido um mecanismo com serviços livres de contenção, denominado de *Point Coordination Function* (PCF) [FIS95a]. Através do DCF, uma estação que deseja transmitir só pode proceder a transmissão se o seu temporizador de *backoff* estiver em zero e se o meio estiver livre por um tempo

maior ou igual à constante DIFS (*Distributed InterFrame Space*). Caso o seu temporizador ainda não seja zero, mas o meio estiver livre, a estação decrementa continuamente o temporizador. O valor deste temporizador não pode ser diminuído enquanto o meio estiver ocupado.

Entretanto, se a estação deseja transmitir e o seu temporizador já for zero e o meio estiver ocupado, ela deve aguardar até o fim da transmissão atual e escolher um novo valor para o seu temporizador de *backoff*, que só poderá ser decrementado enquanto o meio estiver livre.

Empregam-se um mecanismo de confirmação, que indica a recepção exitosa ou não de cada quadro de dados (pois o transmissor não pode determinar se o quadro foi recebido corretamente escutando sua própria transmissão). O receptor transmite um quadro de confirmação (*acknowledgment frame*) após o intervalo de tempo SIFS (*Short InterFrame Space*), que é menor do que DIFS, imediatamente após a recepção do quadro de dados. Notase que o quadro de confirmação é enviado **sem** o receptor escutar o canal. Se o quadro de confirmação não for recebido, o transmissor assume que o quadro de dados foi perdido e uma retransmissão é planejada. Denomina-se este método como **Basic Access** (BA).

Caso uma estação inquirida não envie dados ou uma estação de fora da lista de *polling* não confirme dados recebidos, o PC assume o controle do meio após intervalo PIFS.

Caso uma estação inquirida não possua dados a serem enviados, deve retornar um quadro nulo (*null frame*), para o PC ter garantias que não houve problemas de transmissão (como interferência causada pela sobreposição de células de transmissão).

Uma estação define para a rede sem fio, no início de suas atividades, se deseja ou não estar na lista de *polling*, podendo mudar seu estado em relação à lista mais tarde. O PC também interfere na constituição da lista, integrando ou descartando estações da lista pela observação de seu tráfego nos períodos sem e de contenção.

Por fim, observa-se que nem todos as estações reconhecem o modo de operação PCF. Neste caso, jamais integrarão a lista e, se receberem dados, deverão confirmá-los como no regime DCF. Supõe-se que jamais obterão o controle do canal pois, em nenhum momento, o canal deverá ficar livre por um tempo igual ou maior do que DIFS (na verdade, o tempo máximo em que o canal ficará livre deve ser igual a PIFS).

Percebe-se que, apesar do mecanismo de *polling* sempre apresentar um *overhead*

considerável [CHE94], a escolha deste mecanismo para prover serviços livres de contenção é resultado de uma série de compromissos. Como exemplo, nem todos as estações precisam ter o mecanismo de PCF [FIS95a], simplificando a implementação da camada MAC. Mesmo assim, estas estações são capazes de receber e confirmar dados durante o período de PCF, pelo fato dos mecanismos de *acknowledgment* serem iguais nos dois períodos (DCF e PCF). Uma das alternativas estudadas para prover serviços livres de contenção em redes locais sem fio, o TDMA (*Time Division Multiple Access*), além de exigir um mecanismo rígido de sincronismo em relação a *slots* de tempo [BAU95], alia estações, que não o implementem, de se comunicarem durante o período livre de contenção.

As soluções adotadas para serviços com e sem contenção visam a atender principalmente requisitos de economia de energia. De fato, o mecanismo de detecção virtual da portadora possibilita que se desliguem os circuitos de transmissão e

recepção até o temporizador NAV atingir zero, pois a estação não pode transmitir e não deve receber nenhum quadro durante este tempo.

A troca de quadros especiais RTS e CTS restringe a probabilidade de ocorrer alguma colisão durante o período de envio destes quadros. Pode-se considerar esta troca como um mecanismo de detecção de colisão, uma vez que só se inicia a transmissão dos dados após o transmissor receber CTS em resposta a RTS. Caso este mecanismo falhe, ou porque o quadro RTS não chegou ao receptor, ou porque o transmissor não recebeu RTS, assume-se que houve uma colisão (pois alguém mais está transmitindo) e suspende-se temporariamente o envio de mensagem. Além disso, este mecanismo de detecção de colisão adequa-se ao fato de que, em equipamentos de redes locais sem fio, não haverá detecção de portadora enquanto se transmite. Por fim, a troca de quadros evita o problema do terminal escondido [BHA94], onde uma estação não percebe que o destinatário de sua transmissão (B) já está recebendo dados de outra estação (A), por estar fora do alcance de A, e começa a enviar para B. Ocorre, então, uma colisão no receptor B.

Por fim, deve-se ressaltar que a subcamada MAC apóia-se integralmente na capacidade do nível físico em determinar se o meio está livre ou não. Dado que as características do meio mudam constantemente ao longo do tempo, a camada física deverá ser implementada de forma bastante robusta para evitar falsas conclusões sobre o estado do meio.

Conclusão

O padrão 802.11 constitui-se em um dos mais complexos da família IEEE 802 [RYP96], por necessitar atender as peculiaridades inerentes ao meio de transmissão. O Grupo de Trabalho 802.11, ao oferecer duas técnicas de transmissão de rádio-freqüência para a banda de 2,4 GHz e uma de infravermelha difusa, com vazão de 1 a 2 Mbit/s, acredita que está fechando o leque de aplicações típicas de uma rede local sem fio [HAY96], quer sejam ambientes internos, como escritórios, lojas, quer sejam ambientes externos, como *campi* de uma Universidade ou em complexos prediais.

A transmissão infravermelha difusa deve permanecer restrita a nichos de mercado. A favor desta técnica de transmissão, pode-se apontar que sua propagação nunca passa além dos limites impostos por paredes e tetos do local onde está funcionando, fato que pode se consolidar como uma importante característica de segurança.

Em uma análise mais detalhada das especificações propostas para as técnicas de rádio-freqüência, a tolerância a interferências em suas mais diversas formas tem um peso considerável. A transmissão FHSS apresenta, desde sua definição, preocupação em lidar com interferências, prevendo, inclusive, a operação de diversas redes simultâneas em uma mesma área geográfica. Já a tecnologia DSSS, mesmo sendo a mais difundida no momento, não apresenta características de desempenho comparáveis a FHSS. A maioria dos fornecedores de redes locais sem fio, atualmente, possui projetos de desenvolvimento centrados em FHSS [PRO96]. Parece-nos que, tanto em termos de desempenho como de flexibilidade operacional, a opção pela tecnologia de FHSS no nível físico para redes locais sem fio deverá ser privilegiada nos próximos anos.

Independentemente de qual técnica de transmissão se utilize, um bom funcionamento dos mecanismos implementados na subcamada MAC depende do correto funcionamento destas. O nível físico deve ser capaz de avaliar corretamente se o meio está livre ou não, tarefa não trivial e que, pela presença de interferências, pode levar a conclusões falsas. Nunca é demais reforçar a importância do impacto de interferências, pois um sinal com um determinado alcance em área sem obstáculos pode interferir em uma área bem maior [RYP95].

A subcamada MAC deste padrão, por exemplo, constitui-se um exemplo de como o meio de transmissão pode influenciar além do nível físico. De fato, o esquema de troca de quadros RTS-CTS evita que seja necessário a detecção de portadora enquanto se transmite, ação bastante complicada para um ambiente sem fio. A subcamada MAC ainda é capaz de prover opcionalmente serviços sensíveis a tempo. Resta, contudo, um estudo mais detalhado sobre o desempenho deste esquema para serviços com tempo de resposta limitado.

Além disso, resta saber se, de fato, redes sem fio podem aparecer para a subcamada LLC e níveis superiores como qualquer outra rede IEEE 802. O padrão, ainda em elaboração, não definiu até que grau proverá mobilidade para as estações móveis [FIS95b]. Ou seja, deve-se definir ainda qual será o grau de mobilidade permitido para estações móveis poderem se deslocar de uma rede móvel para outra. Se houver permissão para este tipo de mobilidade, novos problemas podem surgir, como correto roteamento de pacotes de dados para as estações, questões não levantadas neste artigo.

Por fim, a inserção de redes sem fio junto ao mercado reside basicamente em três fatores, a saber: segurança, pois o meio de transmissão, por ser de domínio público, não é considerado confiável; custo, ainda bastante elevado, e vazão, considerada baixa para as atuais aplicações multimídia sensíveis a tempo. Para o primeiro item, o padrão contempla um esquema de segurança com o mesmo nível de privacidade de sistemas fixos (WEP – *Wired Equivalent Privacy*). O custo deve sofrer reduções, à medida em que mais fabricantes produzam equipamentos de acordo com o padrão e em que estas tecnologias amadureçam e começem a ser utilizadas. A questão da vazão só será resolvida com pesquisas para transmitir em canais com largura de banda maior do que a especificada até agora. Entretanto, componentes para frequência muito altas são mais caros, além de poder envolver riscos à saúde ainda não pesquisados.