20

Sistemas Multimídia Distribuídos

Os aplicativos multimídia geram e consomem fluxos contínuos de dados em tempo real. Eles são compostos por grandes quantidades de áudio, vídeo e outros elementos de dados vinculados no tempo, e o processamento e a distribuição dos elementos de dados individuais (amostras de áudio, quadros de vídeo) respeitando critérios temporais são fundamentais. Os elementos de dados distribuídos com atrasos não têm valor e normalmente são eliminados.

Uma especificação para um fluxo multimídia é expressa em termos de valores aceitáveis para a taxa com que os dados passam de uma origem para um destino (a largura de banda), para o atraso da distribuição de cada elemento (latência) e com a taxa que os elementos são perdidos ou eliminados. A latência é particularmente importante nos aplicativos interativos. Um pequeno grau de perda de dados de fluxos multimídia frequentemente é aceitável, desde que o aplicativo possa voltar a sincronizar os elementos que vêm após aqueles que foram perdidos.

A alocação e o escalonamento dos recursos para atender às necessidades dos aplicativos multimídia, e outros, é referida como gerenciamento da qualidade do serviço. A alocação da capacidade de processamento, a largura de banda da rede e a memória (para o armazenamento em *buffer* dos elementos de dados distribuídos antecipadamente), tudo isso é importante. Eles são alocados em resposta às exigências da qualidade do serviço dos aplicativos. Uma requisição de qualidade do serviço bem-sucedida gera uma garantia de qualidade do serviço para o aplicativo e resulta na reserva, e no subsequente escalonamento, dos recursos solicitados.

Este capítulo recorre substancialmente a um tutorial de Ralf Herrtwich [1995] e agradecemos a ele pela permissão para usar seu material.

20.1 Introdução

Os computadores modernos podem manipular fluxos de dados contínuos (*streams*), baseados no tempo, como áudio e vídeo digital. Essa capacidade levou ao desenvolvimento de aplicações multimídia distribuídas, como as bibliotecas de vídeo interligadas em rede, a telefonia pela Internet e a videoconferência. Tais aplicações são viáveis com as redes e os sistemas de propósito gerais atuais, embora a qualidade do áudio e vídeo resultantes seja frequentemente insatisfatória. Aplicações mais exigentes, como videoconferência de larga escala, produção de TV digital, TV interativa e sistemas de supervisão com vídeo estão além da capacidade das tecnologias de interligação em rede e dos sistemas distribuídos atuais.

Os aplicativos multimídia exigem a distribuição de fluxos dos dados multimídia com restrições temporais para os usuários finais. Os fluxos de áudio e vídeo são gerados e consumidos em tempo real e a distribuição oportuna dos elementos individuais (amostras de áudio, quadros de vídeo) é fundamental para a integridade da aplicação. Em resumo, os sistemas multimídia são sistemas em tempo real: eles precisam executar tarefas e apresentar resultados de acordo com um escalonamento determinado externamente. O grau com que isso é conseguido pelo sistema subjacente é conhecido como *qualidade de serviço* (QoS – *Quality of Service*) usufruída por um aplicativo.

Embora os problemas do projeto de sistemas em tempo real tenham sido estudados antes do advento dos sistemas multimídia, e muitos sistemas em tempo real bem-sucedidos tenham sido desenvolvidos (veja, por exemplo, Kopetz e Verissimo [1993]), geralmente, eles não têm sido integrados em sistemas operacionais e redes de propósito gerais. A natureza das tarefas executadas pelos sistemas em tempo real existentes, como na aviação, controle de tráfego aéreo, controle de processos de fabricação e centrais telefônicas, diferem das executadas nos aplicativos multimídia. As primeiras geralmente lidam com volumes de dados relativamente pequenos, e poucas vezes têm *prazos finais rígidos*, mas o não cumprimento de um prazo pode ter consequências sérias ou mesmo desastrosas. Em tais casos, a solução adotada têm sido superestimar os recursos de computação e alocá-los com um escalonamento fixo que sempre garanta o atendimento desses requisitos de pior caso. Esse tipo de solução não existe para a maior parte das aplicações de fluxos multimídia da Internet em computadores de mesa, resultando em uma qualidade de serviço do tipo "melhor esforço", usando os recursos disponíveis.

A alocação e o escalonamento planejado dos recursos para atender às necessidades dos aplicativos multimídia, e outros, é referida como *gerenciamento de qualidade de serviço*. A maioria dos sistemas operacionais e redes atuais não inclui os recursos de gerenciamento da qualidade de serviço necessários para garantir a qualidade de aplicativos multimídia.

As consequências do não cumprimento dos prazos finais em aplicativos multimídia podem ser sérias, especialmente nos ambientes comerciais, como nos serviços de vídeo sob demanda, aplicações de videoconferência comercial e medicina à distância, mas os requisitos diferem significativamente daqueles de outros aplicativos em tempo real:

- Frequentemente, os aplicativos multimídia são altamente distribuídos e operam dentro de ambientes de computação distribuída de propósito geral. Portanto, eles competem com outros aplicativos distribuídos pela largura de banda da rede e pelos recursos de computação nas estações de trabalho dos usuários e nos servidores.
- Os requisitos de recursos dos aplicativos multimídia são dinâmicos. Uma videoconferência exigirá mais ou menos largura de banda de rede à medida que o número de participantes aumentar ou diminuir. O uso de recursos de computação na estação de trabalho de cada usuário também variará, pois, por exemplo, o nú-

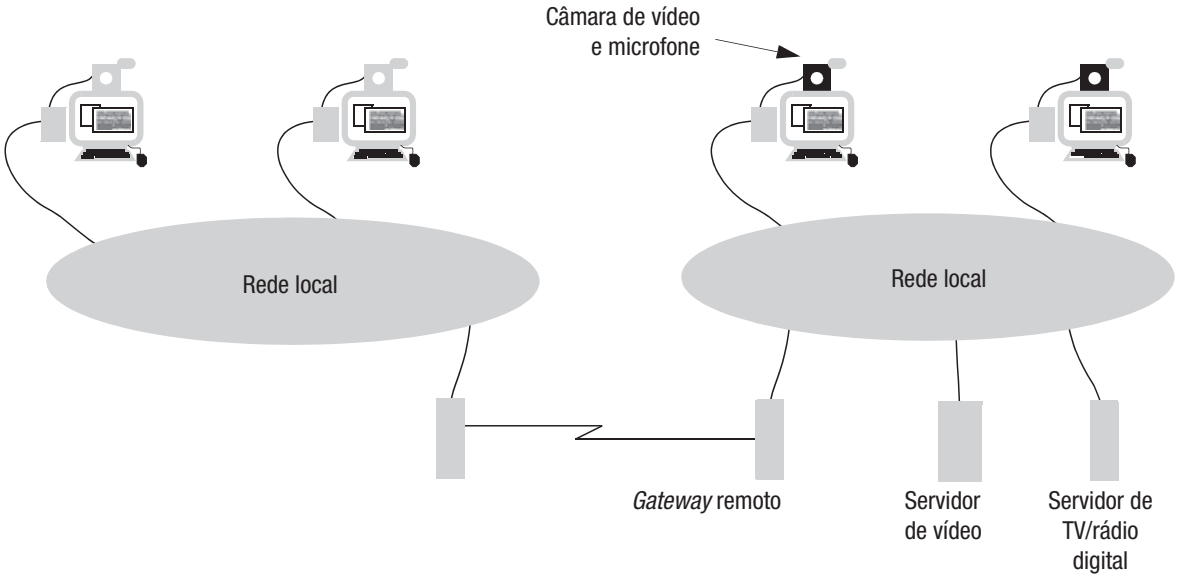


Figura 20.1 Um sistema multimídia distribuído.

mero de fluxos de vídeo que precisa ser exibido varia. Os aplicativos multimídia podem envolver outras cargas variáveis ou intermitentes. Por exemplo, uma aula expositiva multimídia em rede poderia incluir uma atividade de simulação com uso intenso do processador.

- Frequentemente, os usuários desejam contrabalançar os custos de recursos de um aplicativo multimídia com outras atividades. Assim, talvez eles queiram reduzir suas demandas por largura de banda de vídeo em um aplicativo de videoconferência para permitir que uma conversa de voz separada prossiga, ou que o desenvolvimento de um programa, ou de uma atividade de processamento de texto, continue enquanto estão participando de uma conferência.

Os sistemas de gerenciamento de qualidade de serviço se destinam a satisfazer todas essas necessidades, administrando os recursos disponíveis dinamicamente e variando as alocações em resposta às demandas variáveis e às prioridades do usuário. Um sistema de gerenciamento de qualidade de serviço deve supervisionar todos os recursos de computação e comunicação necessários para adquirir, processar e transmitir fluxos de dados multimídia, especialmente onde os recursos são compartilhados entre os aplicativos.

A Figura 20.1 ilustra um sistema multimídia distribuído típico, capaz de suportar uma variedade de aplicações, como videoconferência em *desktop*, fornecimento de acesso a sequências de vídeo armazenadas, transmissão de TV e rádio digital. Os recursos para os quais o gerenciamento de qualidade de serviço é exigido incluem largura de banda de rede, ciclos do processador e capacidade da memória. A largura de banda de disco no servidor de vídeo também pode ser incluída. Adotaremos o termo genérico *largura de banda do recurso* para nos referirmos à capacidade de qualquer recurso de *hardware* (rede, processador central, subsistema de disco) transmitir ou processar dados multimídia.

Em um sistema distribuído aberto, os aplicativos multimídia podem ser iniciados e usados sem organização prévia. Vários aplicativos podem coexistir na mesma rede e até na mesma estação de trabalho. Portanto, surge a necessidade do gerenciamento de quali-

dade de serviço, independentemente da *quantidade total* de largura de banda do recurso ou da capacidade de memória presente no sistema. O gerenciamento de qualidade de serviço é necessário para *garantir* que os aplicativos possam obter a quantidade de recursos necessária nos momentos exigidos, mesmo quando outros aplicativos estão competindo pelos recursos.

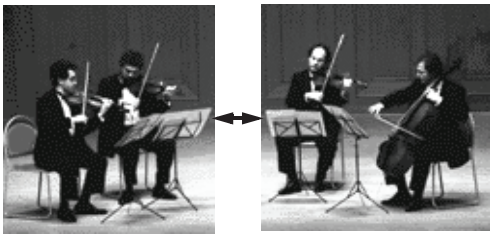
Alguns aplicativos multimídia têm sido implantados, mesmo nos atuais ambientes de computação e de redes baseados em melhor esforço e com pouco suporte à qualidade de serviço. Isso inclui:

Multimídia baseada na Web: são aplicativos que fazem os melhores esforços para a garantir a qualidade de serviço para acessar os fluxos de dados de áudio e vídeo publicados por meio da Web. Eles têm sido bem-sucedidos quando há pouca ou nenhuma necessidade de sincronização dos fluxos de dados em diferentes locais. Seu desempenho varia com a largura de banda e com as latências encontradas nas redes e é prejudicado pela incapacidade dos sistemas operacionais atuais de suportar escalonamento de recursos em tempo real. Contudo, aplicações como *YouTube*, *Hulu* e *BBC iPlayer* oferecem uma demonstração eficaz e popular da exiquibilidade da reprodução de fluxos multimídia em computadores pessoais pouco carregados. Elas exploram o uso extensivo de *buffers* no destino para atenuar as variações na largura de banda e na latência e conseguem uma reprodução contínua e suave de áudio de alta qualidade e de sequências de vídeo de média resolução, embora com um atraso da origem para o destino que pode chegar a vários segundos.

Serviços de vídeo sob demanda: esses serviços fornecem informações de vídeo em forma digital, recuperando os dados de grandes sistemas de armazenamento *online* e enviando-os para a tela do usuário final. Eles são bem-sucedidos onde está disponível uma largura suficiente de banda de rede dedicada e onde o servidor de vídeo e as estações receptoras são dedicados. Eles também fazem uso considerável de *buffers* no destino.

Os aplicativos altamente interativos apresentam problemas muito maiores. Muitos aplicativos multimídia são cooperativos (envolvendo vários usuários) e síncronos (exigindo que as atividades dos usuários sejam fortemente coordenadas). Eles abrangem um amplo espectro de contextos e cenários de aplicação. Por exemplo:

- Telefonia na Internet. Veja o quadro a seguir.
- Uma videoconferência simples, envolvendo dois ou mais usuários, cada um usando uma estação de trabalho equipada com uma câmara de vídeo digital, microfone, saída de som e capacidade de exibição de vídeo. *Software* aplicativo para suportar teleconferência simples surgiu há mais de uma década (*CU-SeeMe* [Dorcey 1995]) e agora é amplamente distribuído (por exemplo, *Skype, NetMeeting* [www.microsoft.com III], *iChat AV* [www.apple.com II]).
- Um recurso de ensaio e execução musical que permita músicos, em diferentes locais, tocar em conjunto [Konstantas *et al*. 1997]. Essa é uma aplicação multimídia particularmente exigente, porque as restrições de sincronização são muito severas.



Aplicações como essas exigem:

Comunicação com baixa latência: atrasos de ida e volta (RTT) de 100–300 ms, para que a interação entre os usuários pareça ser síncrona.

Estado distribuído síncrono: se um usuário interromper uma exibição de vídeo em determinado quadro, os outros usuários devem vê-la parada no mesmo quadro.

Sincronismo de mídia: todos os participantes de uma execução musical devem ouvi-la aproximadamente ao mesmo tempo (Konstantas *et al*. [1997] identificaram um requisito de sincronização dentro de 50 ms). Fluxos separados de trilha sonora e de vídeo devem manter "sincronização labial"; por exemplo, para o comentário de um usuário ao vivo em uma reprodução de vídeo ou para uma sessão de karaokê distribuída.

Sincronização externa: em conferências e em outras aplicações cooperativas podem existir dados ativos em outros formatos, como animações geradas por computador, dados de CAD, quadros-negros eletrônicos e documentos compartilhados. Suas atualizações devem ser distribuídas e agir de maneira que pareçam pelo menos aproximadamente sincronizadas com os fluxos multimídia baseados no tempo.

Na Seção 20.2, examinaremos as características dos dados multimídia. A Seção 20.3 descreverá estratégias para a alocação de recursos escassos para obter qualidade de serviço e a Seção 20.4 discutirá os métodos para seu escalonamento. A Seção 20.5 discutirá mé-

Telefonia na Internet – VoIP

A Internet não foi projetada para aplicações interativas em tempo real, como a telefonia, mas tornou-se possível usá-la para isso, como resultado do aumento na capacidade e no desempenho dos seus componentes básicos – seu *backbone* de enlaces de rede funciona a 10–40 Gbps e os roteadores que os interligam têm desempenho comparável. Normalmente, esses componentes funcionam com baixos fatores de carga (< 10% de utilização de largura de banda) e, portanto, o tráfego de IP raramente é atrasado ou eliminado como resultado da disputa por recursos.

Isso resultou na possibilidade de construção de aplicações de telefonia na Internet pública, por meio da transmissão de fluxos de amostras de voz digitalizadas da origem para o destino, através de datagramas UDP, sem nenhum preparativo especial para obter qualidade do serviço. As aplicações de VoIP (*Voice-over*-IP), como Skype e Vonage, contam com essa técnica, assim como os recursos de voz de aplicações de troca de mensagens instantânea, como AOL Instant Messaging, Apple iChat AV e Microsoft NetMeeting.

É claro que se trata de aplicações interativas de tempo real e permanece o problema da latência. Conforme discutido no Capítulo 3, o roteamento de pacotes IP acarreta um atraso inevitável em cada roteador pelos quais eles passam. Para rotas longas, esses atrasos podem ultrapassar facilmente 150 ms e os usuários observarão isso na forma de atrasos na interação das conversas. Por isso, as ligações telefônicas interurbanas (especialmente intercontinentais) na Internet sofrem muito mais com os atrasos do que as que usam a rede telefônica convencional.

Contudo, grande parte do tráfego de voz é transportada na Internet e a integração com a rede telefônica convencional está em andamento. O SIP (Session Initiation Protocol, definido no RFC 2543 [Handley *et al*. 1999]) é um protocolo em nível de aplicação para o estabelecimento de ligações de voz (assim como de outros serviços, como a troca de mensagens instantânea) pela Internet. Existem *gateways* para a rede telefônica convencional em muitos locais pelo mundo, permitindo que as ligações sejam iniciadas a partir de dispositivos conectados na Internet e terminem nos telefones convencionais ou em computadores pessoais.

	Taxa de dados (aproximada)	*Amostra ou quadro*	
		tamanho	*frequência*
Conversação telefônica	64 kbps	8 bits	8.000/s
Som com qualidade de CD	1,4 Mbps	16 bits	44.000/s
Vídeo de TV padrão (não compactado)	120 Mbps	até 640 × 480 pixels × 16 bits	24/s
Vídeo de TV padrão (MPEG-1 compactado)	1,5 Mbps	variável	24/s
Vídeo HDTV (não compactado)	1.000-3.000 Mbps	até 1920 × 1080 pixels × 24 bits	24-60/s
Vídeo HDTV (MPEG-2/MPEG-4 compactado)	6-20 Mbps	variável	24-60/s

Figura 20.2 Características dos fluxos multimídia típicos.

todos para otimizar o fluxo de dados em sistemas multimídia. A Seção 20.6 descreverá três estudos de caso de sistemas multimídia: o servidor de arquivos de vídeo Tiger, um sistema de baixo custo, escalável, para a distribuição de fluxos de vídeo armazenados, concorrentemente, para grandes números de clientes; o BitTorrent, como um exemplo de aplicação de compartilhamento de arquivos *peer-to-peer* que suporta o *download* de arquivos multimídia grandes; e o End System *Multicast*, da CMU, como um exemplo de sistema que suporta transmissão de conteúdo de vídeo pela Internet.

20.2 Características dos dados multimídia

Referimo-nos aos dados de vídeo e áudio como contínuos e baseados no tempo. Como podemos definir essas características mais precisamente? O termo "contínuo" se refere à visão que o usuário tem dos dados. Internamente, a mídia contínua é representada como sequências de valores discretos que substituem uns aos outros com o passar do tempo. Por exemplo, uma imagem é amostrada 25 vezes por segundo para dar a impressão de uma cena se movendo com a qualidade de uma TV; um sinal sonoro é amostrado 8.000 vezes por segundo para transmitir fala com a qualidade de um telefone.

Diz-se que os fluxos multimídia são denominados *baseados no tempo* (ou *isocrônicos*), pois elementos de dados temporais nos fluxos de áudio e vídeo definem a semântica ou o "conteúdo" do fluxo. Os tempos nos quais os valores são reproduzidos, ou gravados, afetam a validade dos dados. Portanto, os sistemas que suportam aplicações multimídia precisam preservar a temporização ao manipular dados contínuos.

Frequentemente, os fluxos multimídia são volumosos. Assim, os sistemas que suportam aplicações multimídia precisam mover dados com maior desempenho de saída do que os sistemas convencionais. A Figura 20.2 mostra algumas taxas de dados e frequências de quadro/amostragem típicas. Observamos que os requisitos de largura de banda de recurso para algumas delas são muito grandes. Isso é especialmente verdade no caso de vídeo de qualidade razoável. Por exemplo, um fluxo de vídeo de TV

padrão não compactado exige mais de 120 Mbps, o que ultrapassa a capacidade de uma rede Ethernet de 100 Mbps. Um programa que copia ou que aplica uma transformação simples nos dados em cada quadro de um fluxo de vídeo de TV padrão exige menos de 10% da capacidade da CPU de um PC. Os valores para fluxos de televisão de alta definição são ainda mais altos e devemos notar que, em muitas aplicações, como as videoconferências, há necessidade de manipular vários fluxos de vídeo e áudio concorrentemente. Portanto, o uso de representações compactadas para superar esses problemas é fundamental, embora transformações, como a mistura e edição de vídeo, sejam difíceis de realizar com fluxos compactados.

A compactação pode reduzir os requisitos de largura de banda por fatores entre 10 e 100, mas os requisitos de temporização de dados contínuos não são afetados. Há bastante pesquisa e atividades de padronização sendo realizadas com o objetivo de produzir representações eficientes de propósito gerais e métodos de compactação para fluxos de dados multimídia. Esse trabalho tem resultado em vários formatos de dados compactados, como GIF, TIFF e JPEG para imagens estáticas, e MPEG-1, MPEG-2 e MPEG-4 para sequências de vídeo.

Embora o uso de dados de vídeo e áudio compactados reduza os requisitos de largura de banda em redes de comunicação, ele impõe cargas adicionais substanciais sobre os recursos de processamento na origem e no destino. Frequentemente, isso tem sido fornecido usando-se *hardware* de propósito especial para processar e distribuir informação de vídeo e áudio – os codificadores/decodificadores (*codecs*) de vídeo e áudio encontrados nas placas de vídeo fabricadas para computadores pessoais. No entanto, o poder cada vez maior dos computadores pessoais e das arquiteturas de multiprocessadores agora permite que eles realizem grande parte desse trabalho em *software* usando filtros de codificação e decodificação de *software*. Essa estratégia oferece maior flexibilidade, com melhor suporte para formatos de dados específicos da aplicação, lógica de aplicação de propósito especial e o tratamento simultâneo de vários fluxos de mídia.

O método de compactação usado para os formatos de vídeo MPEG é assimétrico, com um algoritmo de compactação complexo e descompactação mais simples. Isso tende a ajudar no seu uso em videoconferências em *desktop*, no qual a compactação é frequentemente realizada por um *codec* de *hardware*, mas a descompactação dos vários fluxos que chegam ao computador de cada usuário é realizada por *software*, permitindo que o número de participantes na videoconferência varie sem considerar o número de *codecs* no computador de cada usuário.

20.3 Gerenciamento de qualidade de serviço

Quando as aplicações multimídia são executadas em redes de computadores pessoais, elas competem por recursos nas estações de trabalho que executam outras aplicações (ciclos de processador, ciclos de barramento, capacidade de *buffer*) e nas redes (enlaces físicos de transmissão, comutadores, *gateways*). Talvez as estações de trabalho e as redes tenham de suportar várias aplicações multimídia e convencionais. Há competição entre as aplicações multimídia e convencionais, entre diferentes aplicações multimídia e até entre os fluxos de mídia dentro de aplicações individuais.

O uso concorrente de recursos físicos para uma variedade de tarefas tem sido possível há muito tempo em sistemas operacionais multitarefa e em redes compartilhadas. Nos sistemas operacionais multitarefa, o processador central é alocado para tarefas (ou processos) individuais no esquema de rodízio, ou em outro esquema de escalonamento

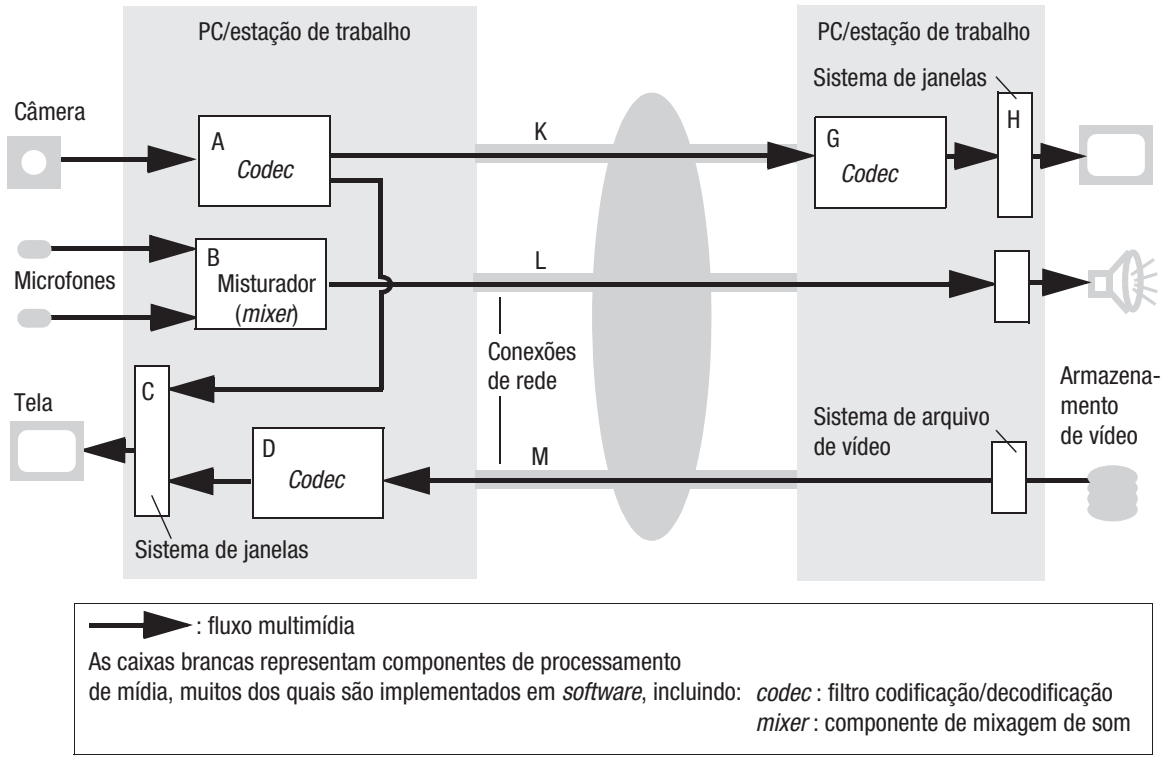


Figura 20.3 Componentes típicos de infraestrutura para aplicações multimídia.

que compartilhe os recursos de processamento na base do *melhor esforço*, entre todas as tarefas que estão competindo pelo processador central.

As redes são projetadas de forma a permitir que mensagens de diferentes origens sejam intercaladas, possibilitando a existência de muitos canais de comunicação virtuais nos mesmos canais físicos. A tecnologia de rede local predominante, a Ethernet, gerencia um meio de transmissão compartilhado na base do *melhor esforço*. Qualquer nó pode usar o meio quando ele estiver desocupado, mas podem ocorrer colisões no envio de quadros e, quando elas ocorrem, os nós de envio esperam por períodos aleatórios para evitar colisões repetidas. Provavelmente, as colisões ocorrem quando a rede está muito carregada e esse esquema não pode fornecer quaisquer garantias a respeito da largura de banda ou da latência em tais situações.

A principal característica desses esquemas de alocação de recursos é que eles tratam dos aumentos na demanda distribuindo os recursos disponíveis entre as tarefas concorrentes. O rodízio (*round robin*) e outros métodos de melhor esforço para compartilhamento de ciclos de processador e largura de banda de rede não podem satisfazer as necessidades das aplicações multimídia. Conforme vimos, o processamento e a transmissão de fluxos multimídia em momentos oportunos são fundamentais para eles. Uma distribuição atrasada não tem valor. Para conseguir a distribuição com restrições temporais, as aplicações precisam garantir que os recursos necessários sejam alocados e escalonados nos momentos exigidos.

O gerenciamento e a alocação de recursos para fornecer tais garantias são referidos como *gerenciamento de qualidade de serviço*. A Figura 20.3 mostra os componentes de infraestrutura para uma aplicação simples de conferência multimídia sendo executada em dois computadores pessoais, usando compactação de dados de *software* e conversão de

	Componente		*Largura de banda*	*Latência*	*Taxa de perda*	*Recursos exigidos*
	Câmera	Saída:	10 quadros/s, vídeo bruto 640x480x16 bits	–	Zero	–
A	Codec	Entrada: Saída:	10 quadros/s, vídeo bruto fluxo MPEG-1	Interativa	Baixa	10 ms CPU a cada 100 ms; 10 Mbytes RAM
B	Misturador	Entrada: Saída:	2 × 44 kbps áudio 1 × 44 kbps áudio	Interativa	Muito baixa	1 ms CPU a cada 100 ms; 1 Mbytes RAM
H	Sistema de janelas	Entrada: Saída:	Várias Framebuffer de 50 quadros/s	Interativa	Baixa	5 ms CPU a cada 100 ms; 5 Mbytes RAM
K	Conexão de rede	Entrada/ Saída:	Fluxo MPEG-1, aprox. 1,5 Mbps	Interativa	Baixa	1,5 Mbps, protocolo de fluxo de baixa perda
L	Conexão de rede	Entrada/ Saída:	Áudio 44 kbps	Interativa	Muito baixa	44 kbps, protocolo de fluxo de perda muito baixa

Figura 20.4 Especificações da qualidade de serviço para componentes de aplicações multimídia mostrada na Figura 20.3.

formato. As caixas brancas representam os componentes de *software* cujos requisitos de recursos podem afetar a qualidade de serviço da aplicação.

A figura mostra a arquitetura abstrata mais comumente usada para *software* multimídia, na qual *fluxos* de mídia de elementos de dados gerados continuamente (quadros de vídeo, amostras de áudio) são processados por um conjunto de processos e transferidos por meio de conexões entre eles. Os processos produzem, transformam e consomem fluxos contínuos de dados multimídia. As conexões ligam os processos em uma sequência de uma *origem* para um *destino*, no qual os elementos de mídia são representados ou consumidos. As conexões entre os processos podem ser implementadas por conexões de rede, ou por meio de transferências na memória quando os processos residem na mesma máquina. Para os elementos de dados multimídia chegarem a seu destino a tempo, cada processo deve receber tempo de CPU, capacidade de memória e largura de banda de rede adequadas para a execução da tarefa designada, e devem ser programados para usar os recursos de forma suficientemente frequente para enviar a tempo os elementos de dados de seu fluxo para o próximo processo.

Na Figura 20.4, especificamos os requisitos de recurso dos principais componentes de *software* e conexões de rede da Figura 20.3 (observe as letras correspondentes aos componentes nessas duas figuras). Claramente, os recursos exigidos só poderão ser garantidos se houver um componente de sistema responsável pela alocação e pelo escalonamento desses recursos. Vamos nos referir a esse componente como *gerenciador de qualidade de serviço*.

A Figura 20.5 mostra as responsabilidades do gerenciador de qualidade de serviço em forma de fluxograma. Nas duas próximas subseções, descreveremos as duas principais subtarefas do gerenciador de qualidade de serviço:

Negociação de qualidade de serviço: a aplicação indica seus requisitos de recurso para o gerenciador de qualidade de serviço. O gerenciador de qualidade de serviço avalia a possibilidade de atender aos requisitos com um banco de dados dos recur-

sos disponíveis e em relação aos comprometimentos de recurso correntes, e dá uma resposta positiva ou negativa. Se a resposta for negativa, a aplicação poderá ser reconfigurada para usar recursos reduzidos e o procedimento é repetido.

Controle de admissão: se o resultado da avaliação de recurso for positivo, os recursos solicitados serão reservados e a aplicação receberá um *contrato de recurso* indicando os recursos que foram reservados. O contrato inclui um limite de tempo. Então, a aplicação fica livre para ser executada. Se ela mudar seus requisitos de recurso, deverá notificar o gerenciador de qualidade de serviço. Se os requisitos diminuírem, os recursos liberados serão retornados ao banco de dados como recursos disponíveis. Se eles aumentarem, uma nova rodada de negociação e controle de admissão será iniciada.

No restante desta seção, descreveremos com mais detalhes as técnicas para executar essas subtarefas. É claro que, enquanto uma aplicação está em execução, há necessidade de escalonamento refinado de recursos como tempo de processador e largura de banda de rede para garantir que os processos em tempo real recebam a tempo seus recursos alocados. As técnicas para isso serão discutidas na Seção 20.4.

20.3.1 Negociação de qualidade de serviço

Para negociar a qualidade de serviço entre uma aplicação e seu sistema subjacente, a aplicação deve especificar seus requisitos de qualidade de serviço para o gerenciador de qualidade de serviço. Isso é feito por meio da transmissão de um conjunto de parâmetros. Três parâmetros têm maior interesse quando se trata do processamento e transporte de fluxos multimídia: *largura de banda*, *latência* e *taxa de perda*.

Largura de banda: a largura de banda de um fluxo, ou componente multimídia, é a taxa na qual os dados fluem por ela.

Latência: latência é o tempo exigido para um elemento individual de dados se mover em um fluxo, da origem até o destino. Isso pode variar, dependendo do volume de outros dados presentes no sistema e de outras características da carga do sistema. Essa variação é chamada *jitter* – formalmente, *jitter* é a primeira derivada da latência.

Taxa de perda: como a distribuição atrasada de dados multimídia não tem valor, os elementos de dados serão eliminados quando for impossível entregá-los antes de seu tempo de distribuição programado. Em um ambiente de qualidade de serviço perfeitamente gerenciado isso nunca deve acontecer, mas até agora existem poucos ambientes assim, pelos motivos mencionados anteriormente. Além disso, o custo em recursos para garantir a distribuição temporal de cada elemento de mídia é frequentemente inaceitável – para tratar com picos ocasionais, provavelmente haverá o envolvimento de muito mais do que o requisito médio de reserva de recursos. A alternativa adotada é aceitar certa taxa de perda de dados, isto é, quadros de vídeo ou amostras de áudio eliminados. Normalmente, as relações aceitáveis são mantidas baixas – raramente mais do que 1% e muito menos para aplicações cuja qualidade é crítica.

Os três parâmetros podem ser usados:

1. Para descrever as características de um fluxo multimídia em um ambiente específico. Por exemplo, um fluxo de vídeo pode exigir uma largura de banda média de 1,5 Mbps e, como ele é usado em uma aplicação de conferência, precisa ser transferido com um atraso máximo de 150 ms para evitar hiatos na palestra. O algoritmo de

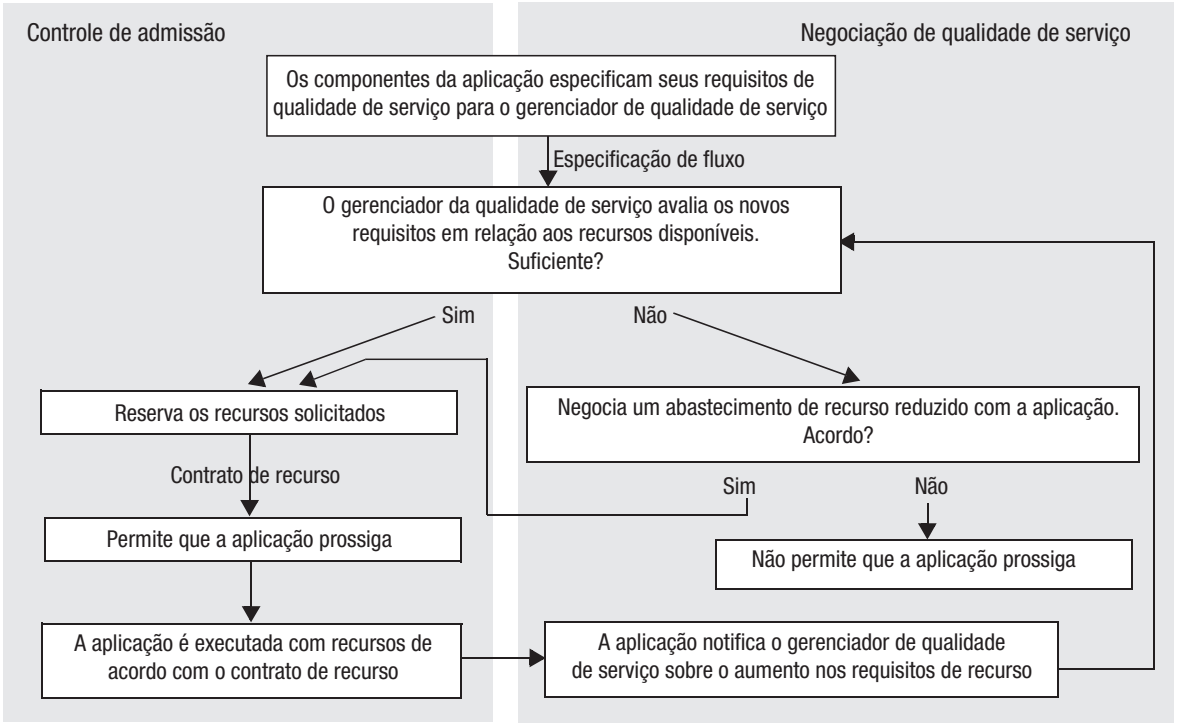


Figura 20.5 A tarefa do gerenciador de qualidade de serviço.

descompactação usado no destino pode produzir aceitavelmente imagens com uma taxa de perda de um quadro em 100.

2. Para descrever a capacidade dos recursos de transportar um fluxo. Por exemplo, uma rede pode fornecer conexões de largura de banda de 64 kbps, seus algoritmos de enfileiramento garantir atrasos de menos de 10 ms e o sistema de transmissão garantir uma taxa de perda menor do que 1 em 10^6.

Os parâmetros são interdependentes. Por exemplo:

- A taxa de perda nos sistemas modernos raramente depende de erros em bits devido a ruído ou defeitos; ela resulta no estouro do *buffer* e de dados dependentes do tempo chegando atrasados. Portanto, quanto maior puderem ser a largura de banda e o atraso, mais provável é que a taxa de perda seja baixa.
- Quanto menor a largura de banda global de um recurso, comparada com sua carga, mais mensagens serão armazenadas e maiores precisam ser os *buffers* para esse acúmulo, para evitar perda. Quanto maiores são os *buffers*, torna-se mais provável que as mensagens esperem que outras mensagens que estão na sua frente sejam atendidas – isto é, maior se tornará o atraso.

Especificando os parâmetros de qualidade de serviço para fluxos • Os valores dos parâmetros de qualidade de serviço podem ser declarados explicitamente (por exemplo, para o fluxo de saída da câmara da Figura 20.3, poderíamos exigir *largura de banda*: 50 Mbps, *atraso*: 150 ms, *perda*: < 1 quadro em 10^3) ou implicitamente (por exemplo, a largura de banda do fluxo de entrada para a conexão de rede *K* é o resultado da aplicação de compactação MPEG-1 na saída da câmara).

Contudo, o caso mais comum é precisarmos especificar um valor e um intervalo de variação permissível. Aqui, consideramos esse requisito para cada um dos parâmetros:

Largura de banda: a maior parte das técnicas de compactação de vídeo produz um fluxo de quadros de diferentes tamanhos, dependendo do conteúdo original do vídeo bruto. Para MPEG, a razão de compactação média está situada entre 1:50 e 1:100, mas isso variará dinamicamente, dependendo do conteúdo; por exemplo, a largura de banda exigida será maior quando o conteúdo estiver mudando mais rapidamente. Por isso, frequentemente é útil citar os parâmetros da qualidade do serviço como valores máximo, mínimo ou médio, dependendo do tipo de regime de gerenciamento de qualidade de serviço que será usado.

Outro problema que surge na especificação da largura de banda é a caracterização da *taxa de rajada* (*burst*). Considere três fluxos de 1 Mbps. Um fluxo transfere um único quadro de 1 Mbit a cada segundo, o segundo é um fluxo assíncrono de elementos de animação gerados por computador, com uma largura de banda média de 1 Mbps, e o terceiro envia uma amostra de som de 100 bits a cada microssegundo. Embora todos os três fluxos exijam a mesma largura de banda, seus padrões de tráfego são muito diferentes.

Uma maneira de lidar com as irregularidades é definir um parâmetro de rajada, além da taxa de transmissão e do tamanho do quadro. O parâmetro de rajada especifica o número máximo de elementos de mídia que podem chegar cedo – isto é, antes do que devam chegar, de acordo com a taxa de chegada normal. O modelo LBAP (linear-bounded arrival processes) usado em Anderson [1993] define o número máximo de mensagens em um fluxo durante qualquer intervalo de tempo t como $Rt + B$, onde R é a taxa de transmissão e B é o tamanho máximo de rajada. A vantagem de usar esse modelo é que ele reflete bem as características das origens multimídia: os dados multimídia lidos de discos normalmente são distribuídos em blocos grandes, e os dados recebidos das redes frequentemente chegam na forma de sequências de pacotes menores. Neste caso, o parâmetro de rajada define a quantidade de espaço no *buffer* exigida para evitar perda.

Latência: alguns requisitos de temporização em aplicações multimídia resultam do próprio fluxo: se um quadro de um fluxo não for processado com a mesma taxa com que os quadros chegam, haverá acúmulo de trabalho e a capacidade do *buffer* poderá ser excedida. Se isso precisa ser evitado, um quadro não deve permanecer em um *buffer*, em média, por mais do que $1/R$, onde R é a taxa de quadros de um fluxo, senão ocorrerá um acúmulo de trabalho. Caso ocorram acúmulos de trabalho, o número e o tamanho dos trabalhos acumulados afetarão o atraso de ponta a ponta máximo de um fluxo, além dos tempos de processamento e propagação.

Outros requisitos de latência surgem do ambiente da própria aplicação. Em aplicações de conferência, a necessidade de interação aparentemente instantânea entre os participantes torna necessário obter atrasos de ponta a ponta absolutos de não mais do que 150 ms para evitar problemas na percepção humana, enquanto que na reprodução de vídeo, para garantir uma resposta apropriada do sistema aos comandos, como *Pausa* e *Parar*, a latência máxima deve ser da ordem de 500 ms.

Uma terceira consideração para o tempo de distribuição de mensagens multimídia é o *jitter* – a variação no período entre a distribuição de dois quadros adjacentes. Embora a maioria dos dispositivos multimídia certifique-se de apresentar dados em sua velocidade normal, sem variação, as apresentações de *software* (por exemplo, em um decodificador de *software* para quadros de vídeo) precisam tomar um cuidado extra para evitar o *jitter*. O *jitter* é resolvido basicamente com o uso de *buffers*, mas sua capacidade de remoção é limitada, pois o atraso de ponta a ponta total é restrito pela consideração mencionada

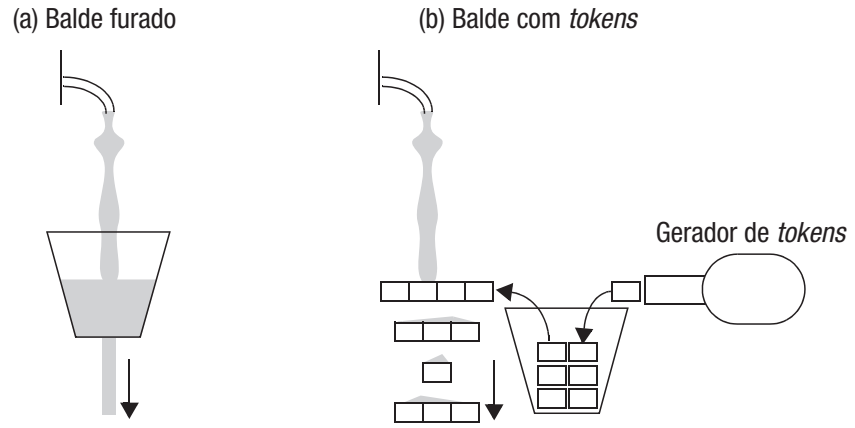


Figura 20.6 Algoritmos de conformação de tráfego.

anteriormente; portanto, a reprodução de sequências de mídia também exige que os elementos da mídia cheguem antes de prazos finais fixos.

Taxa de perda: a taxa de perda é o parâmetro de qualidade de serviço mais difícil de especificar. Os valores de taxa de perda normais resultam de cálculos de probabilidade sobre o estouro de *buffers* e mensagens atrasadas. Esses cálculos são baseados em suposições de pior caso ou em distribuições padrão. Nenhum deles é necessariamente uma boa solução para situações práticas. Entretanto, as especificações de taxa de perda são necessárias para quantificar os parâmetros de largura de banda e latência: duas aplicações podem ter as mesmas características de largura de banda e latência; elas parecerão substancialmente diferentes quando uma aplicação perder cada quinto elemento da mídia e a outra perder apenas um em um milhão.

Assim como nas especificações de largura de banda, em que é importante não apenas o volume dos dados enviados em um intervalo de tempo, mas também sua distribuição nesse intervalo de tempo, uma especificação de taxa de perda precisa determinar o intervalo de tempo durante o qual deve esperar certa perda. Em particular, taxas de perda dadas para períodos de tempo infinitos não são úteis, pois qualquer perda em um tempo curto pode ultrapassar a taxa de longo prazo significativamente.

Conformação de tráfego • Conformação de tráfego é o termo usado para descrever o uso de *buffers* de saída para suavizar o fluxo de elementos de dados. O parâmetro da largura de banda de um fluxo multimídia normalmente fornece uma aproximação ideal do padrão de tráfego real que ocorrerá quando o fluxo for transmitido. Quanto mais próximo for o padrão de tráfego real da sua descrição, melhor o sistema poderá manipular o tráfego; em particular, quando ele utilizar métodos de escalonamento projetados para requisições periódicas.

O modelo LBAP de variações de largura de banda exige regular a taxa de rajada dos fluxos multimídia. Todo fluxo pode ser regulado pela inserção de um *buffer* na origem e pela definição de um método com que os elementos de dados sejam consumidos do *buffer*. Uma boa ilustração desse método é a imagem de um balde furado (Figura 20.6a): o balde pode ser preenchido arbitrariamente com água, até ficar cheio e; por meio de um furo embaixo do balde, a água fluirá continuamente. O algoritmo do balde furado garante que um fluxo nunca será maior do que uma taxa R. O tamanho do *buffer* B define a rajada máxima que um fluxo pode estar sujeito sem perder elementos. B também limita o tempo durante o qual um elemento permanecerá no balde.

	Versão de protocolo
Largura de banda:	Unidade de transmissão máxima
	Taxa do balde com *tokens*
	Tamanho do balde com *tokens*
	Taxa de transmissão máxima
Atraso:	Atraso mínimo notado
	Variação máxima do atraso
Perda:	Sensibilidade de perda
	Sensibilidade de perda de rajada
	Intervalo de perda
	Garantia da qualidade do serviço

Figura 20.7 A especificação de fluxo RFC 1363.

O algoritmo do balde furado elimina completamente as rajadas. Tal eliminação nem sempre é necessária, já que a largura de banda é limitada em qualquer intervalo de tempo. O algoritmo do balde com fichas (*tokens*) consegue isso, permitindo que rajadas maiores ocorram, quando um fluxo estiver ocioso por algum tempo (Figura 20.6b). Trata-se de uma variação do algoritmo do balde furado, na qual são gerados *tokens* para enviar dados a uma taxa fixa *R*. Eles são coletados em um balde de tamanho *B*. Dados de tamanho *S* só podem ser enviados se pelo menos *S tokens* estiverem no balde. O processo de envio remove, então, esses *S tokens*. O algoritmo do balde com *tokens* garante que, em qualquer intervalo *t*, o volume de dados enviados não seja maior do que $Rt + B$. Trata-se, portanto, de uma implementação do modelo LBAP.

Picos altos de tamanho *B* só podem ocorrer em um sistema de balde com *tokens* quando um fluxo estiver ocioso por algum tempo. Para evitar essas rajadas, um balde furado simples pode ser combinado com um balde com *tokens*. A taxa de fluxo *F* desse balde precisa ser significativamente maior do que *R* para que esse esquema faça sentido. Seu único objetivo é fragmentar rajadas realmente grandes.

Especificações de fluxo • Um conjunto de parâmetros de qualidade de serviço é normalmente conhecido como *especificação de fluxo*. Existem vários exemplos de especificações de fluxo e todos são semelhantes. Na RFC 1363 [Partridge 1992], uma especificação de fluxo é definida como 11 valores numéricos de 16 bits (Figura 20.7), os quais refletem os parâmetros de qualidade de serviço discutidos anteriormente, da seguinte maneira:

- A unidade de transmissão máxima e a taxa de transmissão máxima determinam a largura de banda máxima exigida pelo fluxo.
- O tamanho do balde com *tokens* e a taxa determinam a taxa de rajada do fluxo.
- As características de atraso são especificadas pelo atraso mínimo que uma aplicação pode observar (pois queremos evitar a super-otimização para atrasos curtos) e o *jitter* máximo que ela pode aceitar.
- As características de perda são definidas pelo número total aceitável de perdas em certo intervalo e o número máximo de perdas consecutivas.

Existem muitas alternativas para expressar cada grupo de parâmetros. Em SRP [Anderson *et al*. 1990], a taxa de rajada de um fluxo é dada por um parâmetro de trabalho antecipado máximo, que define o número de mensagens em que um fluxo pode estar à frente de sua taxa de chegada normal, em qualquer ponto no tempo. Em Ferrari e Verma [1990], é dado um limite de atraso para pior caso: se o sistema não puder garantir o transporte dos dados dentro desse período de tempo, seu transporte será inútil para a aplicação. Na RFC 1190, a especificação do protocolo ST-II [Topolcic 1990], a perda é representada como a probabilidade de cada pacote ser eliminado.

Todos os exemplos anteriores fornecem um espectro contínuo de valores de qualidade de serviço. Se o conjunto de aplicações e fluxos a serem suportados for limitado, poderá ser suficiente definir um conjunto distinto de classes de qualidade de serviço: por exemplo, áudio com qualidade de telefone e de alta fidelidade, vídeo ao vivo e reprodução, etc. Os requisitos de todas as classes devem ser conhecidos implicitamente por todos os componentes do sistema; o sistema pode até ser configurado para certa mistura de tráfego.

Procedimentos de negociação • Para aplicações multimídia distribuídas, os componentes de um fluxo provavelmente estarão localizados em vários nós. Haverá um gerenciador de qualidade de serviço em cada nó. Uma estratégia simples de negociação de qualidade de serviço é seguir o fluxo de dados ao longo de cada fluxo, da origem até o destino. Um componente de origem inicia a negociação enviando uma especificação de fluxo para seu gerenciador de qualidade de serviço local. O gerenciador pode verificar, em seu banco de dados de recursos disponíveis, se a qualidade de serviço solicitada pode ser fornecida. Se outros sistemas estiverem envolvidos na aplicação, a especificação de fluxo será encaminhada para o próximo nó onde os recursos são exigidos. A especificação de fluxo percorre todos os nós, até que o destino final seja encontrado. Então, a informação sobre se a qualidade de serviço desejada pode ser fornecida pelo sistema é passada de volta para a origem. Essa estratégia simples de negociação é satisfatória para muitos propósitos, mas ela não considera as possibilidades de conflito entre negociações de qualidade de serviço concorrentes partindo de nós diferentes. Um procedimento de negociação de qualidade de serviço transacional distribuído exigiria uma solução completa para esse problema.

Raramente as aplicações têm requisitos de qualidade do serviço fixos. Em vez de retornar um valor booleano sobre se certa qualidade de serviço pode ser fornecida ou não, é mais apropriado o sistema determinar qual tipo de qualidade de serviço pode fornecer e deixar a aplicação decidir se ela é aceitável. Para evitar qualidade de serviço superotimizada, ou para cancelar a negociação quando se tornar claro que a qualidade desejada não pode ser obtida, é comum especificar um valor desejado e o pior valor para cada parâmetro de qualidade de serviço. Uma aplicação pode especificar que deseja uma largura de banda de 1,5 Mbps, mas que também poderia manipular 1 Mbps, ou que o atraso deve ser de 200 ms, mas que 300 ms seria o pior caso ainda aceitável. Como apenas um parâmetro pode ser otimizado por vez, sistemas como o HeiRAT [Vogt *et al*. 1993] esperam que o usuário defina valores de apenas dois parâmetros e deixam que o sistema otimize o terceiro.

Se um fluxo tem vários destinos, o caminho de negociação bifurca de acordo com o fluxo de dados. Como uma ampliação simples do esquema anterior, nós intermediários podem agregar mensagens de retorno de qualidade de serviço dos destinos, para produzir valores de pior caso para os parâmetros de qualidade de serviço. Então, a largura de banda disponível se torna a menor largura de banda disponível de todos os destinos, o atraso se torna o mais longo de todos os destinos e a taxa de perda se torna a maior de todos os destinos. Esse é o procedimento praticado pelos protocolos de negociação iniciada pelo remetente, como SRP, ST-II e RCAP [Banerjea e Mah 1991].

Nas situações com destinos heterogêneos, normalmente é inadequado atribuir uma qualidade de serviço de pior caso comum para todos os destinos. Em vez disso, cada destino deve receber a melhor qualidade de serviço possível. Isso exige um processo de negociação iniciada pelo receptor, em vez de orientada pelo remetente. O RSVP [Zhang *et al*. 1993] é um protocolo de negociação de qualidade de serviço alternativo no qual os destinos se conectam a fluxos. As origens comunicam a existência de fluxos e suas características inerentes para todos os destinos. Então, os destinos podem se conectar no nó mais próximo através do qual o fluxo passa e extrair dados de lá. Para que obtenham dados com a qualidade de serviço apropriada, talvez eles precisem usar técnicas como a filtragem (discutida na Seção 20.5).

20.3.2 Controle de admissão

O controle de admissão regula o acesso aos recursos para evitar sobrecarga de recurso e para proteger os recursos de requisições que eles não possam atender. Isso envolve rejeitar pedidos de serviço, caso os requisitos de recurso de um novo fluxo multimídia violem as garantias de qualidade de serviço existentes.

Um esquema de controle de admissão é baseado em algum conhecimento da capacidade global do sistema e da carga gerada por aplicação. A especificação do requisito de largura de banda de uma aplicação pode refletir a quantidade máxima de largura de banda que a aplicação exigirá, a largura de banda mínima que precisará para funcionar ou algum valor médio. Correspondentemente, um esquema de controle de admissão pode basear sua alocação de recursos em qualquer um desses valores.

Para recursos que possuem um único alocador de recursos, o controle de admissão é simples. Os recursos que possuem pontos de acesso distribuídos, como muitas redes locais, exigem uma entidade de controle de admissão centralizada, ou algum algoritmo de controle de admissão distribuído, que evite admissões concorrentes e conflitantes. O controle de acesso em uma rede de barramento, feito pelas estações de trabalho, pertence a essa categoria; entretanto, mesmo os sistemas multimídia que realizam alocação de largura de banda extensivamente não controlam o acesso ao meio (barramento), pois não se considera que a largura de banda do barramento esteja na janela de escassez.

Reserva de largura de banda • Uma maneira comum de garantir determinado nível de qualidade de serviço para um fluxo multimídia é reservar parte da largura de banda de recurso para seu uso exclusivo. Para atender durante todo tempo aos requisitos de um fluxo é necessário que seja feito uma reserva para sua largura de banda máxima. Essa é a única maneira possível de fornecer qualidade de serviço garantida para uma aplicação – pelo menos enquanto não ocorra nenhuma falha de sistema catastrófica. Ela é usada para aplicações que não conseguem se adaptar a diferentes níveis de qualidade de serviço ou que se tornam inúteis quando ocorrem quedas de qualidade. Exemplos incluem algumas aplicações na área médica (um sintoma pode aparecer em um vídeo de raio X exatamente no momento em que quadros de vídeo são eliminados) e o armazenamento de vídeo (em que quadros eliminados resultarão em uma falha visível sempre que o vídeo for reproduzido).

A reserva baseada nos requisitos máximos pode ser simples: ao se controlar o acesso a uma rede de determinada largura de banda B, podem ser admitidos s fluxos multimídia de largura de banda b_s desde que $\sum b_s <= B$. Assim uma rede *token ring* de 16 Mb/s pode suportar até 10 fluxos de vídeo digital de 1,5 Mb/s cada.

Infelizmente, os cálculos de capacidade nem sempre são tão simples como no caso de uma rede *token ring*. Alocar largura de banda de CPU da mesma maneira exige o conhecimento do perfil de execução de cada aplicação. Entretanto, os tempos de execução

dependem do processador usado e frequentemente não podem ser determinados com precisão. Embora existam várias propostas para o cálculo automático do tempo de execução [Mok 1985, Kopetz *et al*. 1989], nenhuma delas tem sido amplamente usada. Os tempos de execução normalmente são determinados por meio de medidas que frequentemente têm margens de erros grandes e portabilidade limitada.

Para codificações de mídia normais, como MPEG, a largura de banda real consumida por uma aplicação pode ser significativamente menor do que sua largura de banda máxima. As reservas baseadas nos requisitos máximos podem levar, então, ao desperdício de largura de banda de recurso: as solicitações de novas admissões são rejeitadas, embora pudessem ser atendidas com a largura de banda reservada, mas não usada realmente pelas aplicações existentes.

Multiplexação estatística • Devido à potencial subutilização que pode ocorrer, é comum reservar recursos além dos disponíveis. As garantias resultantes, frequentemente chamadas de estatísticas ou condicionais, para distingui-las das garantias determinísticas ou incondicionais, apresentadas anteriormente, só são válidas com alguma probabilidade (normalmente muito alta). As garantias estatísticas tendem a fornecer melhor utilização de recurso quando não consideram o pior caso. Contudo, assim como quando a alocação de recurso é baseada nos requisitos mínimos, ou médios, cargas de pico simultâneas podem causar quedas na qualidade de serviço; as aplicações precisam ser capazes de tratar dessas quedas.

A multiplexação estatística é baseada na hipótese de que para um grande número de fluxos, a largura de banda agregada exigida permanece quase constante, independentemente da largura de banda dos fluxos individuais. Isso presume que, quando um fluxo enviar um grande volume de dados, haverá também outro fluxo que enviará um volume pequeno e, no total, os requisitos serão equilibrados. Entretanto, isso só acontece com fluxos não correlacionados.

Conforme mostram as experiências [Leland *et al*. 1993], o tráfego multimídia em ambientes típicos contradiz essa hipótese. Dado um número maior de fluxos de transferência simultânea, o tráfego agregado ainda permanecerá repleto de transferências simultâneas. O termo *autossemelhante* tem sido aplicado a esse fenômeno, significando que o tráfego agregado mostra semelhança com os fluxos individuais dos quais é composto.

20.4 Gerenciamento de recursos

Para fornecer certo nível de qualidade de serviço para uma aplicação, um sistema não apenas precisa ter recursos suficientes (desempenho), como também precisa tornar esses recursos disponíveis para a aplicação quando eles forem necessários (escalonamento).

20.4.1 Escalonamento de recursos

Os processos precisam ter recursos atribuídos de acordo com suas prioridades. Um escalonador de recursos determina a prioridade dos processos com base em certos critérios. Os escalonadores tradicionalmente encontrados em sistemas operacionais frequentemente baseiam suas decisões de alocar a CPU a processos em função do tempo de resposta e na imparcialidade: as tarefas com uso intenso de E/S recebem alta prioridade para garantir resposta rápida as requisições de usuário, as tarefas ligadas à CPU recebem prioridades mais baixas e, de modo geral, os processos na mesma classe são tratados igualmente.

Os dois critérios permanecem válidos para sistemas multimídia, mas a existência de prazos finais para a distribuição de elementos de dados multimídia individuais muda a natureza do problema do escalonamento. Algoritmos de escalonamento em tempo real

podem ser aplicados a esse problema, conforme discutido a seguir. Como os sistemas multimídia têm de manipular mídia discreta e contínua, torna-se um desafio fornecer um serviço suficientemente ágil para tratar fluxos dependentes do tempo, sem causar inanição de acesso à mídia discreta e de outras aplicações interativas.

Métodos de escalonamento precisam ser aplicados (e coordenados) em todos os recursos que afetam o desempenho de uma aplicação multimídia. Em um cenário típico, um fluxo multimídia seria recuperado do disco e depois enviado, por meio de uma rede, para uma estação de destino, onde seria sincronizado com um fluxo proveniente de outra origem e, finalmente, exibido. Os recursos exigidos nesse exemplo incluem o disco, a rede e as CPUs, assim como memória e largura de banda, em todos os sistemas envolvidos.

Escalonamento imparcial (fairness) • Se vários fluxos concorrem pelo mesmo recurso, torna-se necessário ser imparcial e impedir que fluxos mal comportados ocupem largura de banda em demasia. Uma estratégia simples para garantir a imparcialidade é aplicar escalonamento em rodízio (*round-robin*) em todos os fluxos da mesma classe. Embora em Nagle [1987] tal método tenha sido introduzido pacote por pacote, em Demers *et al*. [1989] ele é usado bit por bit, o que proporciona uma maior imparcialidade com relação aos tamanhos de pacote variados e aos tempos de chegada de pacote. Esses métodos são conhecidos como *enfileiramento imparcial* (*fair queuing*).

Na prática, os pacotes não podem ser enviados bit a bit, mas dada uma taxa de quadros é possível calcular, para cada pacote, quando ele deve ser enviado completamente. Se as transmissões de pacote forem ordenadas com base nesse cálculo, se obterá quase o mesmo comportamento do rodízio de bit a bit real, exceto que, quando um pacote grande for enviado, ele poderá bloquear a transmissão de um pacote menor, o qual teria sido preferido no esquema bit a bit. Entretanto, nenhum pacote é atrasado por mais do que o tempo de transmissão de pacote máximo.

Todos os esquemas de rodízio básicos atribuem a mesma largura de banda para cada fluxo. Para levar em conta a largura de banda individual dos fluxos, o esquema bit a bit pode ser ampliado de modo que, para certos fluxos, um número maior de bits possa ser transmitido por ciclo. Esse método é chamado de *enfileiramento imparcial ponderado* (*weighted fair queuing*).

Escalonamento em tempo real • Vários algoritmos de escalonamento em tempo real foram desenvolvidos para atender às necessidades de escalonamento da CPU de aplicações como o controle de processo industrial. Supondo que os recursos de CPU não tenham sido alocados em demasia (o que é a tarefa do gerenciador de qualidade de serviço), esses algoritmos atribuem repartições de tempo de CPU para um conjunto de processos de uma maneira que garanta a execução de suas tarefas a tempo.

Os métodos tradicionais de escalonamento em tempo real se adaptam muito bem ao modelo de fluxos multimídia contínuos regulares. O escalonamento EDF (*Earliest-Deadline-First*) quase tem se tornado um sinônimo para esses métodos. Um escalonador EDF utiliza um prazo final, que é associado a cada um de seus itens de trabalho, para determinar o próximo item a ser processado: o item com o prazo final mais adiantado é processado primeiro. Nas aplicações multimídia, identificamos cada elemento de mídia que chega a um processo como item de trabalho. O escalonamento EDF se mostrou excelente para a alocação de um único recurso, baseado em critérios de temporização: se houver um escalonamento que atenda a todos os requisitos de temporização, o escalonamento EDF o encontrará [Dertouzos 1974].

O escalonamento EDF exige uma decisão de escalonamento por mensagem (isto é, por elemento multimídia). Seria mais eficiente basear o escalonamento nos elementos que exis-

tem por um tempo maior. O escalonamento RM (*Rate-Monotonic*) é uma técnica proeminente para escalonamento em tempo real de processos periódicos, que obtêm exatamente isso. Os fluxos recebem prioridades de acordo com sua velocidade: quanto maior a velocidade dos itens de trabalho em um fluxo, maior é a prioridade de um fluxo. O escalonamento RM tem se mostrado excelente para situações que utilizam apenas determinada largura de banda por menos de 69% [Liu e Layland 1973]. Usando-se tal esquema de alocação, a largura de banda restante poderia ser dada a aplicações que não fossem em tempo real.

Para suportar tráfego em tempo real com transferência simultânea, os métodos de escalonamento em tempo real básicos devem ser ajustados para distinguirem entre itens de trabalho de mídia contínua críticos e não críticos quanto ao tempo. Em Govindan e Anderson [1991], é introduzido o escalonamento de prazo final/trabalho antecipado. Ele permite que mensagens em um fluxo contínuo cheguem antes em rajadas, mas só aplica escalonamento EDF em uma mensagem no momento de sua chegada normal.

20.5 Adaptação de fluxo

Quando certa qualidade de serviço não pode ser garantida, ou só pode ser garantida com certa probabilidade, uma aplicação precisa se adaptar a níveis de qualidade de serviço mutantes, ajustando seu desempenho de acordo. Para fluxos de mídia contínuos, o ajuste se traduz em diferentes níveis de qualidade de apresentação de mídia.

A forma mais simples de ajuste é eliminar informações. Isso é feito facilmente em fluxos de áudio, em que as amostras são independentes umas das outras, mas pode ser notado imediatamente pelo ouvinte. Desaparecimentos em um fluxo de vídeo codificado em Motion JPEG, em que cada quadro tem posição livre, são mais toleráveis. Os mecanismos de codificação, como o MPEG, em que a interpretação de um quadro depende dos valores de vários quadros adjacentes, são menos robustos em relação às omissões: eles demoram mais para se recuperar de erros e o mecanismo de codificação pode, na verdade, amplificar os erros.

Se houver largura de banda insuficiente, e os dados não forem eliminados, o atraso de um fluxo aumentará com o passar do tempo. Para aplicações não interativas isso pode ser aceitável, embora possa eventualmente levar a estouros de *buffer*, à medida que os dados são acumulados entre a fonte e os destinos. Para aplicações de videoconferência e outras aplicações interativas, atrasos cada vez maiores não são aceitáveis ou só devem existir por um curto período de tempo. Se um fluxo estiver atrasado em seu tempo de execução designado, sua taxa de execução deverá ser aumentada até que ele esteja novamente no tempo certo: enquanto um fluxo está atrasado, os quadros devem aparecer na saída assim que estiverem disponíveis.

20.5.1 Mudanças de escala

Se a adaptação for realizada apenas no destino de um fluxo, a carga em qualquer gargalo no sistema não diminuirá e a situação de sobrecarga persistirá. É interessante adaptar um fluxo para a largura de banda disponível no sistema antes que ele necessite de um recurso em que há gargalos. Isso é conhecido como *mudança de escala*.

A mudança de escala é melhor aplicada quando existem amostras de fluxos ao vivo. Para fluxos armazenados, a facilidade de gerar um fluxo com diferente qualidade depende do método de codificação. A mudança de escala pode ser problemática, caso o fluxo inteiro tenha de ser descompactado e novamente codificado apenas para esse propósito. Os algoritmos de mudança de escala são dependentes da mídia, embora a estratégia de mudança de escala global seja a mesma: fazer uma nova amostragem de determinado

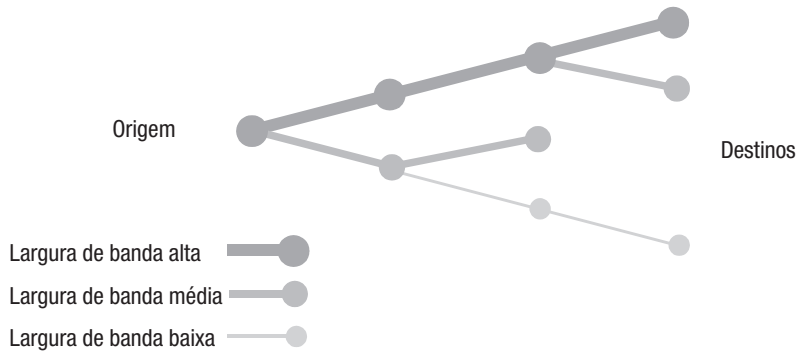


Figura 20.8 Filtragem.

sinal. Para informações de áudio, essa nova amostragem pode ser obtida reduzindo-se a taxa de amostragem de áudio. Ela também pode ser obtida eliminando-se um canal em uma transmissão em estéreo. Conforme esse exemplo mostra, diferentes métodos de mudança de escala podem trabalhar com diferentes granularidades.

Para vídeo, os seguintes métodos de mudança de escala são apropriados:

Mudança de escala temporal: reduz a resolução do fluxo de vídeo no domínio tempo, diminuindo o número de quadros de vídeo transmitidos dentro de um intervalo. A mudança de escala temporal é mais conveniente para fluxos de vídeo em que os quadros individuais são autocontidos e podem ser acessados independentemente. As técnicas de compactação delta são mais difíceis de manipular, pois nem todos os quadros podem ser eliminados facilmente. Portanto, a mudança de escala temporal é mais conveniente para Motion JPEG do que para fluxos MPEG.

Mudança de escala espacial: reduz o número de *pixels* de cada imagem em um fluxo de vídeo. Para a mudança de escala espacial, a organização hierárquica é ideal, pois o vídeo compactado está disponível imediatamente, em várias resoluções. Portanto, o vídeo pode ser transferido pela rede usando diferentes resoluções, sem gravação de cada imagem antes de finalmente transmiti-la. JPEG e MPEG-2 suportam diferentes resoluções espaciais de imagens e são muito convenientes para esse tipo de mudança de escala.

Mudança de escala de frequência: modifica o algoritmo de compactação aplicado a uma imagem. Isso resulta em alguma perda de qualidade, mas em um cenário típico, a compactação pode ser aumentada significativamente, antes que uma redução da qualidade da imagem se torne visível.

Mudança de escala de amplitude: reduz as intensidades das cores de cada *pixel* da imagem. Este método de mudança de escala é usado nas codificações H.261 para chegar a uma vazão (*throughput*) constante quando o conteúdo da imagem varia.

Mudança de escala do espaço de cores: reduz o número de entradas no espaço de cores. Uma maneira de perceber a mudança de escala do espaço de cores é trocar de apresentação em cores para escala de tons.

Se necessário, podem ser usadas combinações desses métodos de mudança de escala.

Um sistema para realizar mudança de escala consiste em um processo monitor no destino e um processo de mudança de escala na origem. O monitor controla os tempos de chegada de mensagens em um fluxo. Mensagens atrasadas são uma indicação de um gargalo no sistema. Então, o monitor envia uma mensagem de *diminuir escala* para a origem, e esta

reduz a largura de banda do fluxo. Após algum período de tempo, a origem aumenta a escala do fluxo novamente. Se o gargalo ainda existir, o monitor detectará novamente um atraso e reduzirá a escala do fluxo [Delgrossi *et al*. 1993]. Um problema para o sistema de mudança de escala é evitar as operações de *aumento de escala* desnecessárias e a oscilação do sistema.

20.5.2 Filtragem

Como a mudança de escala modifica um fluxo na origem, ela nem sempre é conveniente para aplicações que envolvam vários receptores: quando ocorre um gargalo na rota para um destino, esse destino envia uma mensagem de *diminuir escala* para a origem e todos os destinos recebem a qualidade degradada, embora alguns não tivessem problemas para manipular o fluxo original.

A filtragem é um método que fornece a melhor qualidade de serviço possível para cada destino, aplicando mudança de escala em cada nó relevante no caminho da origem para o destino (Figura 20.8). O RSVP (*Resource reservation protocol*) [Zhang *et al*. 1993] é um exemplo de protocolo de negociação de qualidade de serviço que suporta filtragem. A filtragem exige que um fluxo seja dividido em um conjunto de subfluxos hierárquicos, cada um adicionando um nível mais alto de qualidade. A capacidade dos nós em um caminho determina o número de subfluxos recebidos por um destino. Todos os outros subfluxos são filtrados o mais próximo da origem possível (talvez até na origem), para evitar a transferência de dados que posteriormente serão jogados fora. Um subfluxo não é filtrado em um nó intermediário, caso exista um caminho em algum lugar mais adiante que possa transportar o subfluxo inteiro.

20.6 Estudos de caso: Tiger, BitTorrent e End System Multicast

Conforme discutido no Capítulo 1, a multimídia é uma tendência básica nos sistemas distribuídos modernos. Em vez de ser uma área propriamente dita, a multimídia é mais considerada como algo que permeia todos os sistemas distribuídos e, assim, apresenta desafios que precisam ser levados em consideração no projeto de todos os aspectos desses sistemas. Nesta seção, apresentaremos estudos de caso que ilustram como a multimídia influencia três áreas importantes do desenvolvimento de sistemas distribuídos:

- o projeto de um sistema de arquivos distribuído para suportar arquivos de vídeo (*o servidor de arquivos de vídeo Tiger*);
- o projeto de um sistema de *download peer-to-peer* destinado a suportar arquivos multimídia muito grandes (*BitTorrent*);
- o projeto de um serviço de *fluxos multicast* em tempo real (*End System Multicast*).

Note que já vimos outro exemplo de serviço multimídia, quando examinamos a estrutura de rede de sobreposição adotada pelo Skype (veja a Seção 4.5.2).

20.6.1 O servidor de arquivos de vídeo Tiger

Um sistema de armazenamento de vídeos que fornece múltiplos fluxos de vídeo em tempo real simultaneamente é considerado um importante componente de sistema para suportar aplicações multimídia orientadas para o consumidor. Vários protótipos de sistemas desse tipo foram desenvolvidos e alguns até se transformaram em produtos (consulte [Cheng 1998]). Um dos mais avançados é o servidor de arquivos de vídeo Tiger, desenvolvido no Microsoft Research Labs [Bolosky *et al*. 1996].

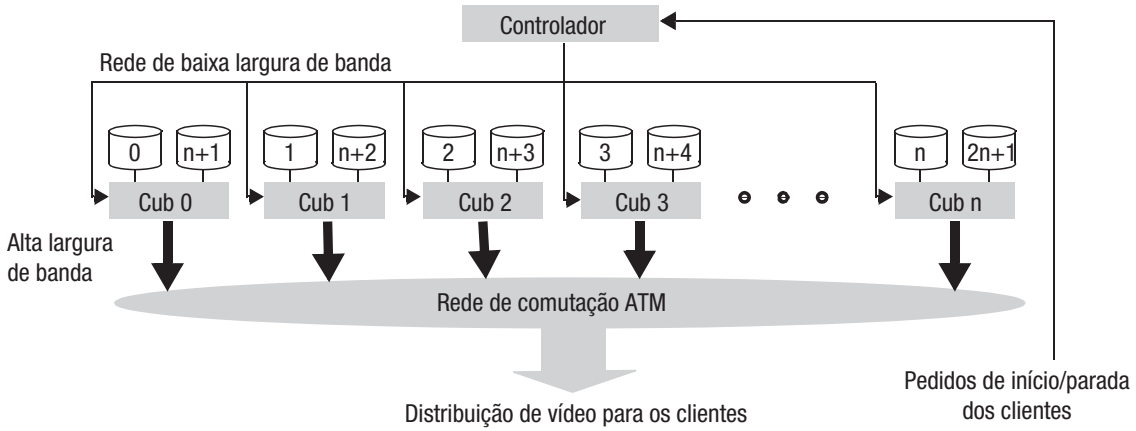


Figura 20.9 Configuração de *hardware* do servidor de arquivos de vídeo Tiger.

Objetivos de projeto • Os principais objetivos de projeto do sistema foram os seguintes:

Vídeo sob demanda para um grande número de usuários: a aplicação típica é um serviço que fornece filmes para clientes pagantes. Os filmes são selecionados em uma grande biblioteca de filmes digitais armazenados. Os clientes devem receber os primeiros quadros de seus filmes escolhidos dentro de poucos segundos após emitirem um pedido e devem ser capazes de executar operações de pausa, retrocesso e avanço rápido à vontade. Embora a biblioteca de filmes disponíveis seja grande, alguns filmes podem ser muito populares e serão motivo de vários pedidos não sincronizados, resultando em várias reproduções concorrentes, porém deslocadas no tempo.

Qualidade do serviço: os fluxos de vídeo devem ser fornecidos a uma velocidade constante, com uma flutuação (*jitter*) máxima determinada pela quantidade (presumida como sendo pequena) de uso de *buffers* disponíveis nos clientes e com uma taxa de perda muito baixa.

Sistema com escalabilidade e distribuído: o objetivo era projetar um sistema com uma arquitetura extensível (pela adição de computadores) para suportar até 10.000 clientes simultaneamente.

Hardware de baixo custo: o sistema era para ser construído usando *hardware* de baixo custo (PCs comerciais com unidades de disco padrão).

Tolerante a falhas: o sistema deveria continuar a funcionar sem degradação notável após a falha de qualquer computador servidor ou unidade de disco.

Tomados em conjunto, esses requisitos exigem uma estratégia radical para o armazenamento e a recuperação de dados de vídeo e um algoritmo de escalonamento eficiente, que harmonize a carga de trabalho entre um grande número de servidores semelhantes. A principal tarefa é a transferência de fluxos de dados de vídeo de largura de banda alta do armazenamento em disco para uma rede, e é essa carga que precisa ser compartilhada entre os servidores.

Arquitetura • A arquitetura de *hardware* do Tiger aparece na Figura 20.9. Todos os componentes são produtos de prateleira. Os computadores *filhotes* (*cub*) mostrados na figura são PCs idênticos, com o mesmo número de unidades de disco rígido padrão (normal-

mente, entre 2 e 4) ligadas em cada um. Eles também estão equipados com placas de rede Ethernet e ATM (veja o Capítulo 3). O *controlador* é outro PC. Ele não está envolvido na manipulação de dados multimídia e é responsável apenas por manipular os pedidos de clientes e pelo gerenciamento das programações de execução dos *cubs*.

Organização do armazenamento • O principal problema de projeto é a distribuição dos dados de vídeo entre os discos ligados aos *cubs* para permitir que eles compartilhem a carga. Como a carga pode envolver o fornecimento de vários fluxos do mesmo filme, assim como o fornecimento de fluxos de muitos filmes diferentes, é improvável que qualquer solução baseada no uso de um único disco para armazenar cada filme atinja o objetivo. Em vez disso, os filmes são armazenados em uma representação segmentada (*strips*) entre todos os discos. Isso leva a um modelo de falha no qual a perda de um disco, ou de um *cub*, resulta em uma lacuna na sequência de cada filme. Isso é tratado por meio de um esquema de espelhamento do armazenamento que replica os dados e de um mecanismo de tolerância a falhas, conforme descrito a seguir.

Segmentação (stripping): um filme é dividido em *blocos* (trechos de vídeo com tempo de reprodução igual, normalmente em torno de 1 segundo, ocupando cerca de 0,5 Mbytes) e o conjunto de blocos que compõem um filme (normalmente, cerca de 7.000 deles para um filme de duas horas) é armazenado nos discos ligados aos diferentes *cubs*, em uma sequência indicada pelos números de disco mostrados na Figura 20.9. Um filme pode começar em qualquer disco. Quando o disco de número mais alto é atingido, o filme "circula", de modo que o próximo bloco é armazenado no disco 0 e o processo continua.

Espelhamento: o esquema de espelhamento divide cada bloco em várias partes, chamadas de *secundários*. Isso garante que, quando um *cub* falhar, a carga de trabalho extra do fornecimento de dados para blocos no *cub* defeituoso recaia sobre vários dos *cubs* restantes e não apenas sobre um deles. O número de secundários por bloco é determinado por um *fator de desagrupamento*, *d*, com valores típicos no intervalo de 4 a 8. Os secundários de um bloco armazenado no disco *i* são armazenados nos discos $i + 1$ a $i + d$. Note que, desde que existam mais do que *d cubs*, nenhum desses discos é ligado no mesmo *cub* que o disco *i*. Com um fator de desagrupamento igual a 8, aproximadamente 7/8 da capacidade de processamento e da largura de banda de disco dos *cubs* pode ser alocada para tarefas isentas de falha. Os 1/8 restantes de seus recursos devem ser suficientes para servir secundários, quando necessário.

Escalonamento distribuído • O centro do projeto Tiger é o escalonamento da carga de trabalho dos *cubs*. O escalonamento é organizado como uma lista de *repartições* (*slots)*, em que cada repartição representa o trabalho que deve ser feito para reproduzir um bloco de um filme – isto é, lê-lo do disco relevante e transferi-lo para a rede ATM. Existe exatamente uma repartição para cada cliente em potencial receber um filme (chamado de *visualizador*), e cada repartição ocupada representa um visualizador recebendo um fluxo de dados de vídeo em tempo real. O estado do visualizador é representado no escalonamento pelo(a):

- endereço do computador cliente;
- identidade do arquivo que está sendo reproduzido;
- posição do visualizador no arquivo (o próximo bloco a ser distribuído no fluxo);
- número de sequência de exibição do visualizador (a partir do qual pode ser calculado um tempo de distribuição para o próximo bloco);
- por algumas informações de contabilidade.

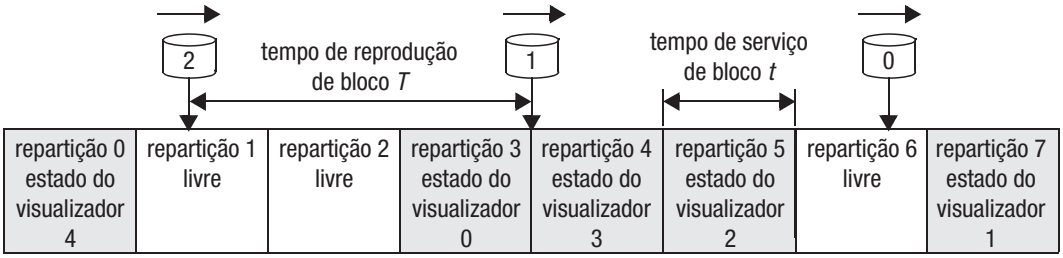


Figura 20.10 Escalonamento Tiger.

O escalonamento está ilustrado na Figura 20.10. O *tempo de reprodução do bloco T* é o tempo que será exigido para um visualizador exibir um bloco no computador cliente; normalmente, cerca de 1 segundo, e supõe-se ser o mesmo para todos os filmes armazenados. Portanto, o Tiger deve manter um intervalo de tempo *T* entre os tempos de distribuição dos blocos em cada fluxo, com uma pequena flutuação (*jitter*) permitida, que é determinada pelo uso de *buffers* disponíveis nos computadores clientes.

Cada *cub* mantém um ponteiro para o escalonamento de cada disco que controla. Durante cada tempo de reprodução de bloco, ele deve processar todas as repartições com números de bloco que caiam nos discos que controla, e tempos de distribuição que caiam dentro do tempo de reprodução de bloco corrente. O *cub* percorre o escalonamento em repartições de processamento em tempo real, como segue:

1. Lê o próximo bloco no armazenamento em *buffer* no *cub*.
2. Empacota o bloco e o envia para o controlador de rede ATM do *cub*, com o endereço do computador cliente.
3. Atualiza o estado do visualizador no escalonamento para mostrar o novo próximo bloco e exibir o número de sequência, e passa a repartição atualizada para o próximo *cub*.

Supõe-se que essas ações ocupam um tempo máximo *t*, que é conhecido como tempo de serviço de bloco. Conforme pode ser visto na Figura 20.10, *t* é substancialmente menor do que o tempo de reprodução de um bloco. O valor de *t* é determinado pela largura de banda de disco ou pela largura de banda de rede, o que for menor. (Os recursos de processamento em um *cub* são adequados para executar o trabalho programado de todos os discos anexos.) Quando um *cub* tiver concluído as tarefas programadas para o tempo de reprodução corrente do bloco, ele estará disponível para tarefas não programadas até o início do próximo tempo de reprodução. Na prática, os discos não fornecem blocos com um atraso fixo e, para acomodar sua distribuição desigual, a leitura do disco é iniciada pelo menos um tempo de serviço de bloco antes que o bloco seja necessário para empacotamento e distribuição.

Um disco pode fazer o trabalho de atender *T/t* fluxos e os valores de *T* e *t* normalmente resultam em um valor > 4 para essa relação. Isso e o número de discos no sistema inteiro determinam o número de visualizadores a que um sistema Tiger pode atender. Por exemplo, um sistema Tiger com cinco *cubs* e três discos ligados a cada um pode distribuir aproximadamente 70 fluxos de vídeo simultaneamente.

Tolerância a falhas • Devido à segmentação de todos os arquivos de filme entre todos os discos em um sistema Tiger, a falha de qualquer componente (uma unidade de disco ou um *cub*) resultaria em uma interrupção do serviço para todos os clientes. O projeto do Tiger resolve isso por meio da recuperação de dados das cópias secundárias espelhadas, quando um bloco primário está indisponível devido à falha de um *cub* ou de uma unidade de disco. Lembre-se de que os blocos secundários são menores do que os primários na

relação do fator de desagrupamento *d* e que os secundários são distribuídos de modo a caírem em vários discos ligados em diferentes *cubs*.

Quando um *cub* ou um disco falha, o escalonamento é modificado por um *cub* adjacente para mostrar vários *estados do visualizador espelho*, representando a carga de trabalho dos *d* discos que contêm os secundários desses filmes. Um estado do visualizador espelho é semelhante ao estado de um visualizador normal, mas com diferentes números de bloco e requisitos de temporização. Como essa carga de trabalho extra é compartilhada entre *d* discos e *d cubs*, ela pode ser acomodada sem interromper as tarefas nas outras repartições, desde que exista uma pequena quantidade de capacidade sobressalente no escalonamento. A falha de um *cub* é equivalente à falha de todos os discos ligados a ele e é tratada de maneira semelhante.

Suporte de rede • Os blocos de cada filme são simplesmente passados para a rede ATM pelos *cubs* que os contêm, junto ao endereço do cliente relevante. Depende-se das garantias de qualidade de serviço dos protocolos de rede ATM para distribuir os blocos para os computadores clientes em sequência e a tempo. O cliente precisa de armazenamento suficiente em *buffer* para conter dois blocos primários, o que está sendo reproduzido no momento na tela do cliente e o que está chegando da rede. Quando os blocos primários estiverem sendo servidos, o cliente só precisará verificar o número de sequência de cada bloco recebido e passá-lo para a rotina de exibição. Quando os secundários estiverem sendo servidos, os *d cubs* responsáveis por um bloco desagrupado distribuem seus secundários para a rede na sequência, e é responsabilidade do cliente reuni-los e montá-los em seu armazenamento em *buffer*.

Outras funções • Descrevemos as atividades críticas quanto ao tempo de um servidor Tiger. Os requisitos de projeto exigidos para fornecimento de funções de avanço rápido e retrocesso. Essas funções exigem a distribuição de uma parte dos blocos do filme para o cliente, para dar o retorno visual normalmente fornecido pelos gravadores de vídeo. Isso é feito com base no melhor esforço por parte dos *cubs*, em um tempo não programado para execução.

As tarefas restantes incluem o gerenciamento e a distribuição do escalonamento e o gerenciamento do banco de dados de filmes, a exclusão de filmes antigos e a gravação de novos nos discos e a manutenção de um índice de filmes.

A implementação inicial, o gerenciamento do escalonamento e a distribuição do Tiger eram manipulados pelo computador controlador. Como isso constituía um único ponto de falha e um gargalo de desempenho em potencial, o gerenciamento do escalonamento foi subsequentemente refeito como um algoritmo distribuído [Bolosky *et al*. 1997]. O gerenciamento do banco de dados de filmes é realizado pelos *cubs* em um tempo não programado, em resposta aos comandos do controlador.

Desempenho e escalabilidade • O protótipo inicial foi desenvolvido em 1994 e usava cinco PCs Pentium de 133MHz, cada um equipado com 48 Mbytes de memória RAM, três unidades de disco SCSI de 2 Gbytes e um controlador de rede ATM, executando Windows NT. Essa configuração foi medida sob uma carga de cliente simulada. Ao servir filmes para 68 clientes, sem falhas no sistema Tiger, a distribuição dos dados era perfeita – nenhum bloco era perdido, nem distribuído com atraso para os clientes. Com um *cub* danificado (e, portanto, três discos), o serviço era mantido com uma taxa de perda de dados de apenas 0,02%, dentro do objetivo de projeto.

Outra medida feita foi a latência de inicialização para distribuir o primeiro bloco de um filme, após a recepção do pedido de um cliente. Isso será altamente dependente do número e da posição das repartições livres no escalonamento. O algoritmo usado para

isso colocaria inicialmente o pedido de um cliente na repartição livre mais próxima, no disco contendo o bloco 0 do filme solicitado. Isso resultou em valores medidos para a latência de inicialização no intervalo de 2 a 12 segundos. Um trabalho recente resultou em um algoritmo de alocação de repartição que reduz o agrupamento de repartições ocupadas no escalonamento, deixando as repartições livres distribuídas mais igualmente na programação e melhorando a latência de inicialização média [Douceur e Bolosky 1999].

Embora as experiências iniciais tenham sido feitas com uma configuração pequena, foram feitas medidas posteriores com uma configuração de 14 *cubs*, 56 discos e o esquema de escalonamento distribuído descrito por Bolosky *et al*. [1997]. A carga que poderia ser servida por esse sistema mudou de escala com sucesso, para distribuir 602 fluxos de dados de 2 Mbps simultâneos, com uma taxa de perda de menos de um bloco em 180.000, quando todos os *cubs* estavam funcionando. Com um *cub* defeituoso, menos de 1 em 40.000 blocos eram perdidos. Esses resultados são impressionantes e parecem corroborar a afirmação de que um sistema Tiger poderia ser configurado com até 1.000 *cubs*, servindo até 30.000–40.000 visualizadores simultâneos.

20.6.2 BitTorrent

O BitTorrent [www.bittorrent.com] é uma aplicação de compartilhamento de arquivos *peer-to-peer* popular, projetada especificamente para o *download* de arquivos grandes (incluindo arquivos de vídeo). Ele não se destina aos fluxos de conteúdo em tempo real, mas sim ao *download* inicial de arquivos a serem reproduzidos posteriormente. O BitTorrent foi mencionado brevemente no Capítulo 10, como um exemplo de protocolo de compartilhamento de arquivos *peer-to-peer*. Neste capítulo, vamos examinar o projeto do BitTorrent mais detalhadamente, dando ênfase ao suporte fornecido para o *download* de arquivos de vídeo.

A principal característica de projeto no BitTorrent é a divisão dos arquivos em trechos de tamanho fixo e sua subsequente disponibilidade em vários *sites* pela rede *peer-to-peer*. Os clientes podem, então, fazer o *download* de vários trechos em paralelo, a partir de diferentes *sites*, reduzindo a carga de qualquer *site* em particular para atender ao *download* (lembrando que o BitTorrent conta com os recursos de máquinas de usuário normais e também que pode haver muitas solicitações simultâneas de arquivos populares). Isso é melhor do que as estratégias mais centralizadas, nas quais um cliente faria o *download* de um arquivo de um servidor usando, por exemplo, HTTP.

O protocolo BitTorrent funciona como segue. Quando um arquivo se torna disponível no BitTorrent, é criado um arquivo *.torrent* que contém metadados associados a esse arquivo, incluindo:

- o nome e o comprimento do arquivo;
- a localização de um *rastreador* (especificado como um URL) – um servidor centralizado que gerencia os *downloads* desse arquivo em particular;
- uma *soma de verificação* (*checksum*) associada a cada trecho, gerada pelo algoritmo *hashing* SHA-1, a qual permite que o conteúdo seja verificado após o *download*.

O uso de rastreadores é uma transigência em relação aos princípios *peer-to-peer* puros, mas isso permite que o sistema mantenha as informações acima facilmente e de maneira centralizada.

Os rastreadores são responsáveis por monitorar o status do *download* associado a um arquivo em particular. Para se entender as informações mantidas pelo rastreador, é necessário voltar e considerar o ciclo de vida de determinado arquivo.

Termo	*Significado*
arquivo *.torrent*	Arquivo que mantém metadados sobre um arquivo disponível
rastreador	Servidor contendo informações sobre os *downloads* em andamento
trecho	Parte de tamanho fixo de determinado arquivo
seeder (semeador)	Par que contém uma cópia completa de um arquivo (consistindo em todos os seus trechos)
leecher (parasita)	Par envolvido no *download* de um arquivo e que, no momento, contém apenas uma parte de seus trechos
torrente (ou enxame)	Conjunto de *sites* envolvidos no *download* de um arquivo, incluindo o rastreador, os *seeders* e os *leechers*
tit-for-tat (olho por olho)	Mecanismo de incentivo que governa o escalonamento de *downloads* no BitTorrent
desafogamento otimista	Mecanismo para permitir que novos pares estabeleçam suas credenciais
mais raro primeiro	Esquema de escalonamento por meio do qual o BitTorrent prioriza os trechos raros dentro de seu conjunto de pares conectados

Figura 20.11 Terminologia do BitTorrent.

Qualquer par (*peer*) com uma versão completa de um arquivo (em termos de todos os seus trechos) é conhecido como *seeder* (*semeador*) na terminologia do BitTorrent. Por exemplo, o par que cria o arquivo fornece a semente inicial para sua distribuição. O pares que desejam fazer *download* de um arquivo são conhecidos como *leechers* (*parasitas*), sendo que, a qualquer momento, determinado *leecher* vai conter vários trechos associados a esse arquivo. Quando um *leecher* faz o *download* de todos os trechos associados a um arquivo, ele se torna um *seeder* para *downloads* subsequentes. Desse modo, os arquivos se espalham como vírus pela rede, com a difusão estimulada pela demanda. Com base nisso, o rastreador mantém informações sobre o estado atual dos *downloads* de determinado arquivo, em termos dos *seeders* e *leechers* associados. O rastreador, junto aos *seeders* e *leechers* associados, são referidos no BitTorrent como *torrente* (ou *enxame*) desse arquivo. (Um resumo da terminologia do BitTorrent pode ser encontrada na Figura 20.11.)

Quando um par quer fazer o *download* de um arquivo, ele primeiro entra em contato com o rastreador e obtém uma visão parcial da torrente, em termos do conjunto de pares que podem suportar o *download*. Depois disso, o trabalho do rastreador está terminado – ele não se envolve no escalonamento de *downloads* subsequente. Esse é um assunto para os vários pares envolvidos e, assim, essa parte do protocolo é descentralizada. Então, os trechos são pedidos e transmitidos em qualquer ordem para o par solicitante (compare isso com o CoolStreaming, apresentado no quadro a seguir).

O BitTorrent, junto a muitos protocolos *peer-to-peer*, conta com os pares se comportando como bons cidadãos, contribuindo e usufruindo o sistema. Fundamentalmente, o sistema tem um mecanismo de incentivos incorporado para recompensar essa cooperação, conhecido como *tit-for-tat* (olho por olho) [Cohen 2003]. Informalmente, essa estratégia dá preferência ao *download* a pares que já fizeram ou estão fazendo *upload* nesse *site*. Além de atuar como mecanismo de incentivo, o *tit-for-tat* também estimula padrões de comunicação em que o *download* e o *upload* acontecem concomitantemente, fazendo o melhor uso da largura de banda.

Em mais detalhes, determinado par suporta o *download* de *n* pares simultâneos por *desafogar* esses pares. As decisões sobre quais pares vão ser desafogados são baseadas

em cálculos de suas taxas de *download*, com a decisão revista a cada 10 segundos. O algoritmo também aplica *desafogamento otimista* em um par aleatório a cada 30 segundos, para permitir que novos pares participem e estabeleçam suas credenciais. Note que o esquema de incentivos tem sido o tema de pesquisa significativa, com esquemas alternativos também propostos – para exemplos, consulte Sirivianos *et al*. [2007]. O BitTorrent acopla isso à política do *mais raro primeiro* para escalonamento de *downloads*, por meio da qual um par prioriza o trecho mais raro em seu conjunto de pares conectados, garantindo que os trechos ainda não prontamente disponíveis sejam espalhados rapidamente.

20.6.3 End System Multicast (ESM)

Um dos maiores desafios técnicos nos sistemas multimídia distribuídos é suportar transmissão de vídeo em tempo real pela Internet. Esses sistemas são exigentes por diversas razões [Liu *et al*. 2008]:

- Os sistemas devem *possuir escalabillidade* para números de usuários potencialmente muito grandes;
- Eles são particularmente exigentes em termos de *utilização de recursos*, impondo restrições significativas de largura de banda, armazenamento e processamento no sistema;
- Rigorosos *requisitos de tempo real* devem ser satisfeitos para que a experiência do usuário seja satisfatória;
- Os sistemas devem ser *flexíveis* e capazes de se *adaptar* a condições variáveis na rede.

Apesar desses desafios, foram feitos avanços significativos e atualmente vários serviços comerciais estão disponíveis, incluindo o BBC iPlayer, o BoxeeTV [boxee.tv] e o Hulu [hulu.com]. Nesta seção, apresentaremos um exemplo de sistema influente nessa área: o End System Multicast [Liu *et al*. 2008], desenvolvido na CMU e agora comercializado pela Conviva [www.conviva.com]. Antes de examinarmos a estratégia técnica defendida pelo ESM, é interessante contextualizar este trabalho.

Contexto do ESM • As primeiras experiências em fluxos de vídeo pela Internet foram feitas diretamente sobre *multicast* IP, conforme descrito na Seção 4.4.1. Essa estratégia tem a vantagem de o suporte para *multicast* ser oferecido diretamente em uma camada baixa no sistema, contribuindo, assim, para o desempenho global. No entanto, a estratégia tem vários inconvenientes, incluindo a falta de suporte para *multicast* IP em muitos roteadores e a necessidade de manter estado tolerante nos roteadores para suportar *multicast*. Mais fundamentalmente, isso também transgride o princípio fim-a-fim, discutido na Seção 2.3.3, o qual defende que o suporte para funções de comunicação (neste caso, fluxos multimídia) só pode ser implementado completamente e de forma confiável com o conhecimento e a ajuda da aplicação que está nos pontos extremos do sistema de comunicação.

Como resultado, agora a maioria dos sistemas defende *estratégias de sistema final* (*end-system*) para fluxos de vídeo, em que o controle e a inteligência residem nas extremidades da rede e não nela própria. Essa estratégia também é chamada de *multicast em nível de aplicação* e implica na formação de uma rede de sobreposição para suportar o tráfego multimídia associado (veja na Seção 4.5 uma discussão sobre redes de *overlay*).

Levando isso um passo adiante, há um interesse considerável nas estratégias *peer-to-peer* para suportar transmissão multimídia na Internet, e o ESM é um exemplo importante de tal sistema. Mais especificamente, o ESM emprega técnicas *peer-to-peer* estruturadas pelas quais os próprio pares formam uma estrutura em árvore para a subsequente distribuição da mídia em tempo real. Como uma alternativa à estratégia estruturada de-

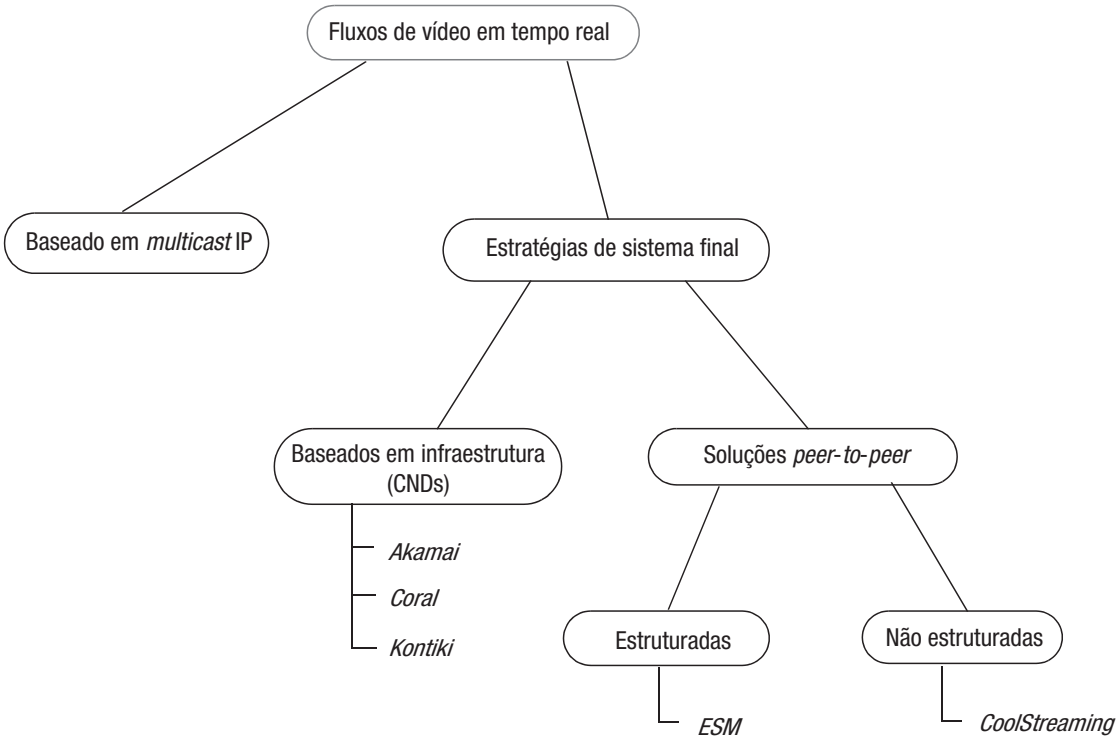


Figura 20.12 Estratégias de fluxo de vídeo em tempo real.

fendida pelo ESM, o CoolStreaming oferece uma estratégia não estruturada, ampliando a ideia do BitTorrent, discutido na Seção 20.6.2 (veja os detalhes no quadro a seguir).

Vários sistemas adotam a metodologia do sistema final, mas, em vez de uma estratégia *peer-to-peer*, oferecem uma infraestrutura fixa para manter várias cópias do conteúdo multimídia (ou outro) localizado em nós por toda a Internet, suportando, assim, uma distribuição mais rápida. Esses sistemas são referidos como *redes de distribuição de conteúdo* (ou CDNs, Content Distribution Networks). Os principais exemplos são o Akamai [www.akamai.com], o Coral [www.coralcdn.org] e o Kontiki [www.kontiki.com]. Tais sistemas suportam uma variedade de estilos de distribuição de conteúdo, incluindo aceleração da Web (maior desempenho no acesso ao conteúdo da Web) e fluxos de vídeo – por exemplo, o Kontiki foi usado na versão original do BBC iPlayer.

As diversas técnicas para suportar fluxos multimídia em tempo real estão resumidas na Figura 20.12.

Arquitetura do ESM • O ESM (End System *Multicast*) é uma solução *peer-to-peer* estruturada para *multicast* de vídeo em tempo real pela Internet. Inicialmente, a estratégia foi desenvolvida na Universidade Carnegie Mellon, como parte de uma pesquisa que investigava as propriedades da *estratégia de sistema final* para multicast, e o protótipo inicial foi usado para fluxos de vídeos em tempo real em diversas conferências importantes, incluindo SIGCOMM, INFOCOM e NOSSDAV [esm.cs.cmu.edu]. Conforme mencionado anteriormente, agora a estratégia é comercializada pela Conviva [www.conviva.com], que recentemente fez um acordo com a NBC Universal para utilizar a plataforma Conviva (denominada C3) na distribuição de seu conteúdo pela Internet.

Além de investigar a estratégia de sistema final, outro objetivo importante do ESM é ser flexível às mudanças, por meio de *auto-organização*. Em particular, os protocolos

de base são projetados para lidar elegantemente com o ingresso e a saída dinâmicas dos nós, com a falha de nós e com mudanças na configuração e no desempenho da rede subjacente. Em particular, eles promovem *adaptação com reconhecimento de desempenho*, por meio da qual as estruturas de sobreposição associadas ao sistema *peer-to-peer* são frequentemente reavaliadas para maximizar o desempenho global. Vamos discutir como isso é atingido, a seguir.

O ESM funciona construindo uma árvore para cada fluxo de vídeo, cuja raiz está na origem desse fluxo em particular. Os principais elementos algorítmicos para suportar isso são:

- como manter informações sobre participação como membro;
- como lidar com novos pares ingressando na árvore;
- como lidar com pares saindo da árvore (seja propositalmente ou por causa de falha);
- como adaptar as estruturas em árvore para favorecer o desempenho (adaptação com reconhecimento de desempenho, conforme mencionado anteriormente).

CoolStreaming: uma estratégia P2P não estruturada para fluxos de vídeo

Muitas estratégias de fluxos de vídeo são baseadas em uma estratégia *peer-to-peer* estruturada, construindo uma árvore (como no ESM) ou uma estrutura de sobreposição alternativa, como uma floresta – uma união disjunta de árvores – ou uma malha. O CoolStreaming [Zhang *et al*. 2005b] adota uma estratégia não estruturada radicalmente diferente para fluxos de vídeo, referida por seus criadores como estratégia *centrada em dados*. No CoolStreaming, os nós mantêm uma visão parcial da participação como membro, a qual é atualizada periodicamente por meio de um protocolo de fofoca (como no ESM). Quando um novo par ingressa, ele primeiro entra em contato com o nó de origem do fluxo de vídeo desejado (o qual presumidamente é anunciado) e esse nó escolhe outro aleatoriamente em seu conjunto de membros conhecidos, para atuar como representante (balanceando, assim, a carga entre todos os membros). Então, o novo nó obtém do representante um conjunto inicial de parceiros, carregando-os no sistema. Deve-se enfatizar que, ao contrário de uma estratégia baseada em árvore, esse conjunto de parceiros não significa quaisquer relações pai-filho para o *download*; em vez disso, o escalonamento do *download* é determinado dinamicamente e governado pela disponibilidade de dados, conforme explicado a seguir.

No CoolStreaming, um arquivo de vídeo é decomposto em vários *segmentos* de tamanho fixo, como no BitTorrent. Cada par cria um *mapa de buffer* para indicar a disponibilidade local de um ou mais segmentos de um arquivo e, então, troca essa informação com seus parceiros conhecidos. Essa informação é usada para obter todos os segmentos necessários de determinada origem de vídeo. Até aqui, isso é muito parecido com o BitTorrent, mas com duas diferenças importantes, devidas à necessidade de fluxos em tempo real. Primeiro, enquanto o BitTorrent pode fazer o *download* de trechos em qualquer ordem, o CoolStreaming deve satisfazer as restrições de tempo real desejadas para a reprodução do vídeo. Segundo, em dado momento, o CoolStreaming está interessado apenas em uma *janela de tempo variável*, desde o presente até um período no futuro próximo (na prática, uma janela de tempo variável de 120 segmentos de um segundo), em vez do arquivo inteiro. O cálculo do escalonamento associado é fundamental para o funcionamento do CoolStreaming e, embora encontrar uma solução perfeita seja o conhecido problema polinomial não determinista (*NP-hard*), o CoolStreaming adotou um conjunto bem-sucedido de heurísticas, baseadas em fatores que incluem o número de fornecedores em potencial de um segmento, a largura de banda dos fornecedores e o tempo disponível para processar a requisição.

O resultado final é uma arquitetura de sistemas distribuídos que pode satisfazer os requisitos de tempo real dos fluxos de vídeo e que é mais naturalmente flexível à falha de nós e às mudanças no desempenho ou na disponibilidade da rede.

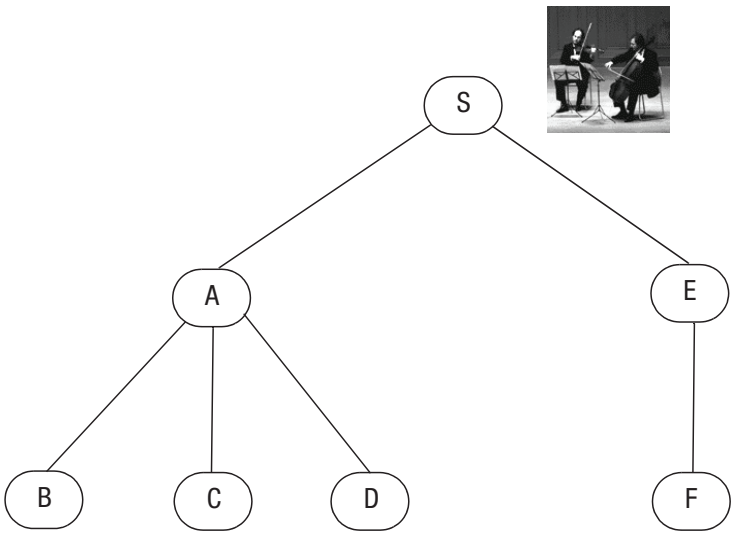


Figura 20.13 Um exemplo de árvore no ESM.

Vamos tratar de cada um desses elementos a seguir. Em cada uma das descrições, vamos nos referir ao exemplo de estrutura em árvore mostrada na Figura 20.13, a qual está transmitindo, em fluxos ao vivo, a apresentação de nossos músicos mencionados anteriormente (Seção 20.1).

Gerenciamento da participação como membro • Cada nó mantém uma visão parcial da participação como membro da árvore, com essa visão atualizada periodicamente com um *protocolo de fofoca* (um método popular para manter a participação como membro de grupo em estruturas *peer-to-peer*, descrito na Seção 18.4). Isso funciona com cada membro escolhendo periodicamente outro do grupo e enviando a ele um subconjunto de sua visão da participação como membro, anotada com informações sobre quando foi a última vez que ele soube de cada membro do grupo (na forma de um carimbo de tempo). Portanto, não há nenhuma tentativa de ter uma visão global consistente das informações sobre participação como membro do grupo, mas essa visão parcial é suficiente para o funcionamento do protocolo, conforme ficará claro a seguir.

Ingressando em uma árvore • É presumido que o nó de origem (a raiz da árvore) é anunciado e, portanto, conhecido pelo sistema. Um novo nó entra em contato com a origem e recebe um conjunto de nós selecionados aleatoriamente, extraídos da visão do grupo mantida pela origem. Esses são efetivamente candidatos a pais do novo nó, o qual deve selecionar um pai adequado nesse conjunto de possibilidades.

O protocolo de *seleção do pai* é fundamental para o funcionamento do ESM, com o objetivo global de otimizar a árvore para favorecer o desempenho (em particular, conforme veremos, para o desempenho de saída (*throughput*), sendo a latência uma consideração secundária).

A primeira fase da seleção do pai é sondar o conjunto de membros fornecido pela origem e coletar as seguintes informações em cada candidato:

- O desempenho que o nó está apresentando atualmente, em termos do *desempenho de saída* e da *latência* da origem;
- a saturação desse nó, em termos do número de filhos que ele já suporta.

A partir de um parâmetro obtido por sondagem, também é possível determinar a latência de ida e volta entre o novo nó e os vários nós candidatos.

O novo nó elimina os candidatos que considera *saturados* (definidos por uma constante interna) e, então, calcula o serviço que pode esperar de cada um dos outros candidatos, em termos de desempenho de saída e atraso. O desempenho de saída é estimado como o mínimo desempenho de saída relatado obtido por esse nó e por dados históricos do novo nó até esse candidato (esses dados podem estar disponíveis se, por exemplo, o novo nó conectou esse candidato em particular anteriormente). O atraso pode ser estimado com base no somatório do atraso relatado da origem e a latência experimentada pela sondagem. A seleção de nós é baseada na melhor largura de banda disponível até o novo nó; se as informações sobre a largura de banda não estão disponíveis, a seleção é feita com base nos valores de latência.

Voltando à árvore de exemplo da Figura 20.13, suponha que um novo nó *G* quer ingressar nesse fluxo de vídeo. *G* entra em contato com o nó de origem *S* e recebe (aleatoriamente) o seguinte conjunto de nós: *{A, C, E e F}*. *A* é eliminado imediatamente, pois é considerado saturado (supondo que a definição de saturação seja ter 3 filhos) e *C* relata características de desempenho de saída ruins, talvez porque *A* esteja saturado. Isso leva a uma escolha entre os nós *E* e *F*. *E* é escolhido por relatar os melhores valores de desempenho de saída disponíveis (talvez o enlace entre *E* e *F* seja por meio de uma conexão de largura de banda relativamente baixa) e, além disso, *G* se conectou a *E* anteriormente, tendo experimentado boas características de desempenho de saída.

Lidando com nós que saem • Os membros podem sair de uma árvore por causa de uma requisição de saída explícita ou devido a uma falha. No primeiro caso, para evitar rompimento, o membro que está saindo notifica os filhos sobre esse fato e deve continuar encaminhando dados por certo período, para evitar a interrupção do serviço mais abaixo na árvore. No último caso, os membros enviam periodicamente mensagens de *ativos* para seus filhos e a falha é detectada quando essas mensagens não são recebidas.

Nos dois casos, todos os filhos devem invocar o procedimento de seleção de pai, conforme definido anteriormente, com verificações extras realizadas para garantir que os candidatos já não sejam descendentes dos nós dados.

Suponha que logo após *G* ingressar na árvore, o nó *E* falhe. Nesse caso, tanto *F* como *G* devem executar o algoritmo de seleção de pai para restabelecer a conectividade.

Adaptação com reconhecimento de desempenho • Cada nó monitora continuamente o serviço que está recebendo de seu nó pai (e, conforme mencionado anteriormente, mantém essas informações de histórico para futura referência). A adaptação é ativada se a taxa detectada cai significativamente abaixo da esperada da origem. Para evitar o descarte (*thrashing*), um nó deve esperar por um período específico, conhecido como *tempo de detecção*, antes de optar por fazer a adaptação.

Uma vez tomada a decisão de adaptar, o nó invocará o algoritmo de seleção de pai para determinar um novo pai mais eficiente. Desse modo, a construção da árvore é constantemente reavaliada e se auto-organizará para otimizar o desempenho global.

Por exemplo, após um período de tempo, *C* pode decidir que o desempenho de saída recebido por meio de *A* é insatisfatório e executar o algoritmo de seleção de pai, resultando em *C* se tornando filho de *E*.

20.7 Resumo

As aplicações multimídia exigem novos mecanismos de sistema para permitir que eles manipulem grandes volumes de dados dependentes do tempo. Os mais importantes desses mecanismos se preocupam com o gerenciamento de qualidade de serviço. Eles devem alocar largura de banda e outros recursos, de uma maneira que garanta que os requisitos de recursos da aplicação sejam atendidos e devem programar o uso dos recursos para que os prazos finais exigentes das aplicações multimídia sejam atendidos.

O gerenciamento de qualidade de serviço trata as requisições de qualidade de serviço das aplicações, especificando a largura de banda, a latência e taxas de perda aceitáveis para os fluxos multimídia, e realiza o controle de admissão, determinando se há recursos disponíveis suficientes para atender a cada nova requisição e negociar com a aplicação, se necessário. Uma vez que uma requisição de qualidade de serviço seja aceita, os recursos são reservados e uma garantia é emitida para a aplicação.

A capacidade do processador e a largura de banda de rede alocada para uma aplicação devem, então, ser programadas para atender a suas necessidades. Um algoritmo de escalonamento de processador em tempo real, como o *earliest-deadline-first* (EDF) ou o *rate-monotonic* (RM), é exigido para garantir que cada elemento do fluxo seja processado a tempo.

Conformação de tráfego é o nome dado aos algoritmos que colocam dados no *buffer* em tempo real para suavizar as irregularidades da temporização que surgem inevitavelmente. Os fluxos podem ser adaptados para utilizar menos recursos, reduzindo a largura de banda da origem (mudança de escala) ou em pontos ao longo do caminho (filtragem).

O servidor de arquivos de vídeo Tiger é um exemplo excelente de sistema com capacidade de escalabilidade que fornece distribuição de fluxo em uma escala potencialmente muito grande, com fortes garantias de qualidade do serviço. Seu escalonamento de recurso é altamente especializado e ele oferece um exemplo excelente da estratégia de projeto alterada que frequentemente é exigida para tais sistemas. Os outros dois estudos de caso, BitTorrent e ESM, também fornecem sólidos exemplos de como suportar, respectivamente, o *download* e o fluxo de dados de vídeo em tempo real, novamente destacando o impacto que a multimídia tem no projeto de sistemas.

Exercícios

20.1 Descreva, em linhas gerais, um sistema para suportar um recurso de ensaio musical distribuído. Sugira os requisitos de qualidade de serviço convenientes e uma configuração de *hardware* e *software* que possa ser usada. *páginas 884, 889*

20.2 Atualmente, a Internet não oferece nenhuma facilidade de reserva de recurso ou de gerenciamento de qualidade de serviço. Como as aplicações de fluxo de áudio e vídeo baseados na Internet existentes obtêm qualidade aceitável? Quais limitações as soluções que eles adotam impõem às aplicações multimídia? *páginas 884, 893, 899*

20.3 Explique as diferenças entre as três formas de sincronização (estado distribuído síncrono, sincronização de mídia e sincronização externa) que podem ser exigidas nas aplicações multimídia distribuídas. Sugira mecanismos por meio dos quais cada uma delas poderia ser obtida, por exemplo, em uma aplicação de videoconferência. *página 885*

20.4 Descreva, em linhas gerais, o projeto de um gerenciador de qualidade de serviço para permitir que computadores *desktop* conectados por uma rede ATM suportem várias aplicações multimídia concorrentes. Defina uma API para seu gerenciador de qualidade de serviço, fornecendo as principais operações, com seus parâmetros e resultados. *páginas 889–891*

20.5 Para especificar os componentes de *software* dos requisitos de recurso que processam dados multimídia, precisamos de estimativas de suas cargas de processamento. Como essa informação pode ser obtida sem trabalho desnecessário? *páginas 889–891*

20.6 Como o sistema Tiger suporta um grande número de clientes, todos solicitando o mesmo filme em momentos aleatórios? *páginas 901–906*

20.7 A escalonamento do Tiger é potencialmente uma grande estrutura de dados que muda frequentemente, mas cada *cub* precisa de uma representação atualizada das partes que está manipulando no momento. Sugira um mecanismo para a distribuição do escalonamento dos *cubs*. *páginas 901–906*

20.8 Quando o Tiger está operando com um disco, ou com um *cub* defeituoso, blocos de dados secundários são usados no lugar dos primários ausentes. Os blocos secundários são *n* vezes menores do que os primários (onde *n* é o fator de desagrupamento). Como o sistema acomoda essa variabilidade no tamanho do bloco? *página 905*

20.9 Discuta os méritos relativos da política de *download* do *mais raro primeiro* no BitTorrent, em comparação com a estratégia de *download* sequencial mais tradicional. *páginas 906-908*

20.10 Além do Tiger (veja o Exercício 20.6), o ESM e o CoolStreaming também suportam acesso por fluxo ao mesmo filme, por, potencialmente, grandes números de usuários. Discuta as estratégias adotadas pelo ESM e pelo CoolStreaming para gerenciar esse acesso simultâneo e compare-as com as defendidas pelo Tiger. *páginas 901-906, 908-912*