Fundamentos

em

Sistemas de Computação

- Resumos 2014 -

Aleardo Manacero Jr.
DCCE/Ibilce/Unesp

Capítulo 1

Introdução

1. O que são sistemas de computação
2. O que é Computação de Alto Desempenho
3. Como se pode obter CAD
4. CAD de baixo custo
5. Redes de computadores e sistemas distribuídos
6. O restante da disciplina

1.1 Sistemas de Computação

Não existe uma definição clara do que seriam sistemas de computação. Para o CNPq, por exemplo, dentro de sistemas de computação aparecem os sistemas digitais, compiladores, etc. Em nosso caso consideraremos apenas os sistemas que servem de interface entre o hardware e os programas aplicativos utilizados pelos usuários do sistema.

Partindo desse ponto de vista fica claro que os sistemas de computação englobam sistemas que permitem fazer um uso eficiente das máquinas, o que inclui sistemas operacionais e protocolos de redes de computadores, entre outras coisas.

1.2 Computação de Alto Desempenho

Entende-se Computação de Alto Desempenho (CAD daqui por diante) como a área da computação preocupada com a criação de condições para que processamentos de elevada carga computacional possam ser executados em intervalos de tempo factíveis, isto é, permitir que programas que levariam anos para fornecerem resultados em uma máquina comum possam faze-lo em bem menos tempo (horas ou dias), por exemplo.

Alguns exemplos de áreas que necessitam CAD são:

- Física de Altas Energias, em que se analisam volumes gigantescos de dados (muito acima de Terabytes hoje em dia), coletados em meses de experimentos e que levariam alguns séculos para serem completamente examinados em equipamentos convencionais.

- Bioinformática, em que o volume de dados (em genômica, por exemplo) também é gigantesco.
- Controle de Energia Elétrica, em que é preciso analisar diversas possibilidades de ação (em caso de curto-circuito, por exemplo) e tomar a decisão correta de forma quase que instantânea. Nesse caso, uma máquina convencional levaria alguns minutos para tomar a decisão, o que inviabiliza a sua execução antes da piora da situação.
- Mineração de dados, em que o volume de dados é alto e o processo de análise envolve uma quantidade elevada de restrições. Isso, por exemplo, é a base de funcionamento de mecanismos de busca como Google e outros.

1.3 Como obter CAD

CAD é obtida, sempre, através da paralelização das atividades a serem executadas pelo programa. Isto significa, por exemplo, fazer um programa que some duas matrizes linha por linha e não elemento por elemento, como fazemos. Para tanto, cada linha é entregue para um processador do sistema, que se encarrega de parte da tarefa e devolve os resultados para quem estiver interessado neles.

Fica claro então que uma condição necessária para a obtenção de CAD é a existência de vários elementos de processamento no sistema. A quantidade de elementos vai depender do grau de paralelismo do problema resolvido e da disponibilidade de recursos (dinheiro) existente. Máquinas com tal capacidade tem o seu preço iniciando em cerca de 500 mil dólares e indo bem além dos 50 milhões de dólares.

Deve ter ficado claro também, que outro componente importante em CAD é o mecanismo responsável pela comunicação entre os elementos de processamento. De forma simples, isso pode ser feito através do compartilhamento da memória ou através de uma estrutura de rede de computadores.

Do ponto de vista de arquitetura essas máquinas podem ser classificadas de várias formas (como deve ser visto no curso de Organização e Arquitetura de Computadores), sendo que para o nosso caso interessa apenas uma classificação rudimentar entre sistemas fortemente acoplados e sistemas fracamente acoplados, assim definidos:

- Fortemente acoplados: são sistemas em que a conexão entre os vários elementos de processamento é feita, em geral, através de memória e barramentos compartilhados. Isto implica em máquinas de alto custo e de grande velocidade de processamento global (graças ao pouco tempo gasto em comunicação).
- Fracamente acoplados: são sistemas em que a conexão é feita através de redes (o que implica em atrasos consideráveis de comunicação), trazendo uma grande redução no custo do sistema. Nessa categoria temos os chamados *clusters* e *grids*, bem como *clouds*.

Atualmente a evolução do *hardware* para CAD permite que se aplique soluções tipicamente de alto desempenho em aplicações relativamente ordinárias. O uso de processadores com vários núcleos (quase todos os atuais) e muitos núcleos (GPUs) tem tornado a computação paralela algo não apenas viável como também necessária para melhor usar os recursos disponíveis.

1.4 CAD de baixo custo

Do item anterior pode ser percebido que é possível obter (dentro de certas restrições é claro) CAD a um baixo custo. Para tanto se usam sistemas fracamente acoplados, que possuem um custo menor de implantação. Outras vantagens de tais sistemas são sua modularidade, que permite que o grau de paralelismo possa ser ampliado máquina a máquina, e sua maior utilização, pois como temos máquinas individualmente funcionais, podemos aproveitar a capacidade de processamento de cada uma delas, mesmo que a tarefa não exija alto desempenho.

A principal desvantagem da utilização desses sistemas está ligada ao custo alto de comunicação. Essa comunicação faz com que tais sistemas tenham um desempenho bastante inferior ao dos sistemas fortemente acoplados. Apesar disso, o seu uso tem se tornado cada vez mais intenso através da popularização dos sistemas distribuídos.

1.5 Sistemas distribuídos

O uso de sistemas distribuídos, que nada mais são do que redes de computadores com ambientes especialmente configurados para a paralelização de suas atividades e serviços, tem possibilitado o crescimento no número de sistemas de alto desempenho em uso. Entretanto, o funcionamento de um sistema distribuído é possível apenas graças a um ambiente de hardware e software preparado para a interação de programas. Isso implica, em outras palavras, na necessidade de um sistema operacional capaz de operar sobre várias máquinas e na existência de máquinas capazes de se comunicar entre si.

A infraestrutura de hardware é necessária para que a comunicação entre os programas paralelos possa ser feita de forma coordenada, sem a criação de eventuais problemas relacionados com a heterogeneidade das máquinas paralelas. Isto implica na criação de um protocolo de tradução entre as várias "linguagens"faladas nas diferentes máquinas.

Já a infraestrutura de software é mais trabalhosa para ser obtida, uma vez que para tanto é necessário que se transforme os mecanismos de gerenciamento de um sistema operacional convencional em mecanismos que operem sobre várias máquinas. Isto é feito para que o paralelismo possa ser aproveitado de modo transparente ao usuário, isto é, deve ser possível ao usuário solicitar a execução de um determinado programa e esse programa ser executado em qualquer máquina (ou mais de uma), sem que o usuário perceba que isso está ocorrendo.

1.6 A disciplina

No restante desse curso examinaremos como é feita a construção de um sistema operacional (convencional por simplicidade) e de uma rede de computadores (protocolos de comunicação). Ao final do semestre é esperado um bom domínio sobre esses tópicos, de tal forma que o aluno possa posteriormente realizar o projeto completo de um S.O. e estudar com mais profundidade os problemas relacionados aos sistemas distribuídos, além de poder iniciar o projeto de uma rede de computadores.

A partir deste capítulo introdutório o curso prossegue, inicialmente examinando

alguns dos requisitos básicos para a existência de uma rede de computadores e posteriormente, em momentos adequados, são examinados os mecanismos de gerenciamento de um sistema operacional convencional, que serão mapeados futuramente para o caso de um sistema distribuído.

1.6.1 Metodologia de ensino

Em cada aula os conceitos serão apresentados numa estratégia parecida com *sala invertida*, em que problemas serão apresentados e as suas soluções deverão ser encontradas através de discussões em sala. Isso exige uma postura diferente por parte dos alunos.

Assim, dos alunos espera-se sempre a leitura prévia, nos livros indicados, dos tópicos a serem examinados em sala de aula. Primeiro por garantir uma melhor compreensão do que será ali discutido e segundo por permitir uma maior familiaridade com os livros usados na disciplina.

Capítulo 2

Redes - Conceitos gerais

1. Topologia
2. Protocolos
3. Camadas e pacotes
4. Concorrência por serviços

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

- Computer Networks, Tanenbaum, cap. 1
- Redes de Computadores, Soares-Lemos-Colcher, cap. 1, 2 e 5
- Local & Metropolitan Area networks, Stallings, cap. 1, 2, 3 e 4

No capítulo anterior constatou-se a necessidade da operação eficiente de uma rede de comunicação para o funcionamento de um sistema distribuído. Isso leva à necessidade do estudo de redes de computadores, que será iniciado aqui, através de seus conceitos mais gerais.

2.1 Topologia

Num primeiro momento o que se faz é apresentar conceitos de topologia de redes (anel, barramento, estrela, completa, etc.), que nada mais é do que a descrição da forma de ligação elétrica entre as várias máquinas que fazem parte da rede. Examina-se então as diferenças entre essas topologias do ponto de vista de custo, tempo de transmissão e estabilidade.

2.2 Protocolos

A simples ligação elétrica entre os vários computadores não garante a comunicação entre os mesmos. Para que isso ocorra existe a necessidade de se definir um protocolo para a comunicação entre máquinas distintas. Isso significa definir um padrão

organizado para o estabelecimento de uma conversa, tal como ocorre, por exemplo, em uma ligação telefônica, quando a pessoa que inicia a chamada deve esperar por um sinal de linha, discar o número desejado, esperar o sinal de chamando e pelo atendimento no outro lado da linha, podendo então iniciar de fato a comunicação. Todos sabemos que não é possível estabelecer uma comunicação correta sem seguir todos esses passos.

Para a comunicação entre computadores isso se torna ainda mais crítico pois tais máquinas são essencialmente "burras", além de poderem falar "línguas" distintas (usarem processadores e/ou sistemas operacionais diferentes).

Assim é fundamental a criação de um protocolo de comunicação que seja não apenas possível mas também confiável. Vários protocolos têm sido propostos, com diversas implementações distintas para cada um deles. Em particular se destacam os protocolos IEEE 802-xx, o RM-OSI da ISO, o ATM e o TCP-IP.

É interessante notar que todos os protocolos têm em comum sua estruturação em camadas. Isso ocorre, primeiro, para tornar sua implementação mais simples e segundo para possibilitar que cada tipo de problema durante a comunicação possa ser resolvido em um ponto diferente do processo. Durante essa disciplina examinaremos com mais detalhes o RM-OSI, algumas variantes dos protocolos 802-xx e o TCO-IP, que na prática é mais uma implementação ad-hoc de protocolo criada com o UNIX.

2.3 Camadas e pacotes

As camadas dentro de um protocolo estão organizadas de forma vertical, isto é, um usuário ao iniciar um processo de comunicação acessa a camada superior do protocolo, que por sua vez acessa a camada imediatamente abaixo e assim por diante, até que se chegue ao meio físico que liga as duas máquinas envolvidas na comunicação. No outro lado da linha o processo se repete, agora indo da camada mais baixa até a mais alta e, finalmente, ao destinatário da comunicação. O fluxo da comunicação ao longo das camadas é feito por pacotes de bits enviados sequencialmente. Assim, toda vez que existir algo para ser transmitido, essa informação é transformada em pacotes e esses pacotes são passados de camada em camada através da prestação de serviços por cada uma delas.

2.4 Concorrência por serviços

Quando um processo de comunicação ocorre da forma descrita acima é fácil constatar que possivelmente teremos vários pacotes sendo transferidos (servidos) entre as (pelas) camadas do protocolo. Em algumas camadas podemos servir mais do que um pacote simultaneamente. Entretanto, em outras camadas o serviço deve ser oferecido a apenas um pacote por vez.

Essa situação caracteriza uma condição de corrida pelo acesso ao serviço, ou em outras palavras, a existência de concorrência por serviços. O tratamento dessa concorrência exige cuidados para evitar o acesso simultâneo ao serviço (exclusão mútua) e também para obter a sincronização entre os serviços de camadas diferentes. Esses cuidados são o objeto de estudo da **"Programação Concorrente"**, sendo examinados com maior rigor a seguir.

Capítulo 3

Programação Concorrente

1. Uma breve introdução aos processos
2. Mecanismos de exclusão mútua
3. Mecanismos de sincronismo
4. Tratamento de deadlocks

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

- Modern Operating Systems, Tanenbaum, cap. 2 e 6
- Operating Systems Concepts, Peterson/Silberschatz, cap. 5 e 6
- Outros bons livros de sistemas operacionais, nos capítulos sobre processos ou concorrência

3.1 Preâmbulo

A compreensão correta do que são processos, do que eles representam e como são tratados por um sistema computacional pode ser melhor entendida a partir da compreensão da evolução de sistemas operacionais. Esse tópico será muito rapidamente visto em sala de aula, desde as máquinas que não tinham SO até os vários modelos de SO existentes atualmente, mas aconselha-se que esse tema seja melhor visto em algum dos livros apontados como referência.

3.2 Processos

Existe uma grande discussão quanto a definição exata de processos. A definição que adotaremos aqui é a de que processos nada mais são do que programas em execução, isto é, um programa passa a ser denominado processo quando é carregado para execução. Neste momento diversos eventos ocorrem, dentre os quais o principal é definir a identificação desse processo durante a sua execução.

A execução coletiva de processos pode ser classificada como sequencial, quando a execução de cada processo for estritamente sequencial, ou concorrente, quando a

execução de dois ou mais processos puder ser feita de modo simultâneo, compartilhando ou não recursos do sistema.

As interações entre execuções de diversos processos podem ser representadas por grafos de processos (que identificam quais processos ativam outros processos) e por grafos de precedência (que identificam quais processos devem ser concluídos antes do início de um dado processo). O entendimento correto de como tais grafos determinam os processos é de grande importância para a compreensão do conceito de concorrência.

3.3 Exclusão mútua

A execução de dois ou mais processos concorrentes pode ser feita com ou sem interação entre eles. Quando não existe interação o tratamento dos mesmos é bastante simples. Já quando existe interação entre os processos é preciso tomar cuidado com vários detalhes. Um deles é a obtenção de exclusão mútua, que é necessária quando dois ou mais processos têm que acessar um dado recurso que não pode ser simultaneamente compartilhado. Essa situação pode levar a erros de execução, que podem ser evitados através de mecanismos como desabilitação de interrupções, soluções de software com espera ocupada (Dekker, Peterson, etc.) ou ainda por primitivas sem espera ocupada (semáforos, block e wake-up, monitores, troca de mensagens, etc.).

3.4 Sincronismo

Um segundo problema no tratamento da interação entre processos é a necessidade de sincronismo entre eles, em determinadas situações. O problema de sincronismo surge, por exemplo, em situações em que um processo produz informação a ser consumida por outro processo. Nesse caso o primeiro não pode produzir uma informação nova antes que a anterior seja consumida e o segundo processo não pode consumir nada que ainda não tenha sido produzido. Soluções para este problema podem ser obtidas com o uso das mesmas primitivas sem espera ocupada listadas acima.

3.5 Deadlocks

Um último, e bastante grave, problema no tratamento de processos concorrentes é a obtenção de processos livres de deadlock. Deadlocks ocorrem quando dois processos se travam na disputa por dois recursos, de tal forma que um dos processos possui o recurso A e espera pelo recurso B, enquanto o outro possui o recurso B e espera pelo A. Nessa situação se torna impossível que um deles (P1 por exemplo) avance, uma vez que para isso é preciso que o outro (P2) libere o recurso que possui, o que apenas é feito se ele (P2) obtiver o outro recurso (de posse de P1) para avançar. Isso caracteriza o que chamamos de espera circular e pode ser tratada evitando que pelo menos uma das condições necessárias para deadlock ocorra. O tratamento de deadlocks é realizado através de métodos como do avestruz, time-stamp, banqueiro ou da padaria.

Problemas semelhantes são o de starvation, quando um processo espera indefinidamente por algum recurso, e o de livelock, que é parecido com deadlock porém sem o bloqueio definitivo dos processos. O problema de starvation, em particular, é de grande interesse na análise de eficiência dos vários mecanismos de gerenciamento do SO, quando se procura identificar a possibilidade ou não de sua ocorrência.

Capítulo 4

Gerenciamento de Processos

1. Estados de um processo
2. Tabela de processos
3. Mecanismos de escalonamento de processos

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

- Modern Operating Systems, Tanenbaum, cap. 2
- Operating Systems Concepts, Peterson/Silberschatz, cap. 4
- Outros bons livros de sistemas operacionais, nos capítulos sobre gerenciamento ou escalonamento de processos

4.1 Estados de um processo

Do capítulo anterior é possível perceber que os vários processos executando em um sistema estão sempre disputando o acesso à CPU. Com isso, é preciso definir um conjunto de estados para representar os processos em cada condição. De modo geral os processos podem assumir os seguintes estados:

- **Ativo** (ou executando), quando estiver fisicamente ocupando a cpu;
- **Pronto**, quando estiver apenas esperando por uma chance de ocupar a cpu;
- **Em espera** (ou bloqueado), quando estiver esperando pela ocorrência de algum evento externo que o permita ficar esperando apenas pela cpu;
- **Inativo** (ou morto), quando não estiver executando.

Além desses estados é possível diferenciar outros estados a partir do fato do processo estar ou não fisicamente na memória.

A passagem de um processo de um estado para outro ocorre em determinadas circunstâncias, que são:

- **Pronto** para **Ativo**: quando o *dispatcher* (um dos componentes de um sistema operacional) escalonar o processo;
- **Ativo** para **Pronto**: quando o *dispatcher* decidir que outro processo deve ocupar a cpu (por estouro do tempo de cpu, por exemplo);
- **Ativo** para **Em espera**: quando o processo solicitar uma operação de entrada/saída ou for bloqueado por algum semáforo (ou outro mecanismo de bloqueio);
- **Ativo** para **Inativo**: quando o processo for retirado da memória ou terminar sua execução;
- **Inativo** para **Pronto**: quando o *scheduler* (um *dispatcher* mais eficiente) trazer o processo do disco para a memória;
- **Em espera** para **Pronto**: quando a operação de E/S for completada ou o processo for desbloqueado.

4.2 Tabela de processos

Para que o sistema tenha condições de manipular os processos (e seus estados) é necessário que se tenha uma estrutura para armazenar informações de cada processo. Isso é feito por uma tabela, normalmente chamada de **tabela de processos** ou **bloco de controle de processos - BCP**, que deve conter informações tais como o número do processo (PID), identificador de seu estado, instante de criação, tempo acumulado de execução, conteúdo dos registradores da cpu por ele usados, endereço na memória, pilha, etc.

É através da manipulação desta tabela que o sistema operacional pode controlar todos os processos e, também, fornecer informações para os algoritmos de escalonamento na cpu. Uma descrição mais detalhada dos dados de uma tabela de processos pode ser vista nos livros de SO.

4.3 Mecanismos de escalonamento de processos

A ocupação da cpu pelos processos que estejam no estado de pronto deve ser feita de forma organizada. Para tanto existem algoritmos de escalonamento que atuam de acordo com premissas estabelecidas pelo tipo de processo do sistema, necessidades do usuário, disponibilidade de recursos de hardware, etc. Esses algoritmos podem ser classificados em duas categorias principais: **algoritmos sem preempção** (FCFS - primeiro a chegar primeiro a executar, SJF - primeiro o mais curto, etc.), e **algoritmos com preempção** (Round-Robin, Prioridade, SRTF - primeiro quem falta menos tempo, Filas multinível, etc.).

Os algoritmos sem preempção não permitem que um processo seja interrompido após assumir o controle da cpu, o que passa a ser permitido nos algoritmos com preempção. A diferença entre algoritmos com preempção está centrada exatamente nos motivos que originam a preempção de um processo. De forma geral os algoritmos

com preempção são muito mais eficientes do que os sem preempção por fazer um melhor uso da cpu.

A eficiência de cada um desses algoritmos pode ser medida através de métricas de desempenho, tais como ocupação da cpu, *throughput* (taxa de saída ou vazão), tempos de espera, tempos de execução médios (*turnaround time*), etc. A adoção de uma dessas medidas de desempenho é feita de acordo com a necessidade do usuário, sendo que cada algoritmo de escalonamento pode resultar em medidas bastante distintas de desempenho para conjuntos de processos diferentes. Cabe ao usuário, portanto, escolher a medida que mais lhe interessa e determinar o algoritmo de escalonamento que melhor se adequa à sua situação.

Por fim, é preciso observar que esse cenário vem mudando com o surgimento de threads e processadores multicore. Atualmente é possível executar vários threads simultaneamente, o que aumenta as possibilidades de escalonamento para o gerenciador. Assim, as versões mais recentes dos SO disponíveis já fazem uso dessa capacidade, permitindo um aumento no throughput do sistema.

Capítulo 5

Gerenciamento de Entrada/Saída

1. Responsabilidade
2. Forma de atuação
3. Mecanismos de controle de E/S em disco
4. Problemas com dispositivos de entrada não controlada

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

- Modern Operating Systems, Tanenbaum, cap. 5
- Operating Systems Concepts, Peterson/Silberschatz, cap. 9
- Outros bons livros de sistemas operacionais, nos capítulos sobre gerenciamento de E/S ou gerenciamento de armazenamento em discos ou ainda gerenciamento de periféricos.

5.1 Responsabilidade

Na conclusão do último capítulo é indicado que as operações de E/S representam um fator preponderante no desempenho de um sistema operacional. Isso ocorre porque todo e qualquer processo depende de operações de entrada ou saída de dados e elas sempre tomam tempo para serem realizadas. É possível realizar E/S de várias formas, quando a operação é avaliada olhando-se para o responsável pelo seu gerenciamento. Nesse aspecto as formas são:

1. **E/S programada**, em que toda a ação é comandada pelo próprio processo que quer fazer E/S;
2. **E/S por interrupção**, em que a E/S é comandada por um processo especial, ativado por um sinal de interrupção gerado pelo processo que quer fazer E/S;
3. **E/S por roubo de ciclo**, quando a operação de E/S é controlada por um dispositivo especial (o **DMA** - Direct Memory Access), que rouba ciclos do relógio da CPU para transferir blocos de bytes entre a memória e o dispositivo de E/S;

4. **E/S em cadeia**, em que se permite que a fila de espera por operações de E/S seja controlada pelo DMA e não por algum mecanismo que execute na CPU.

As duas primeiras formas são estratégias que consomem muito tempo de CPU e, portanto, valem apenas para aplicações dedicadas. Com elas é possível ao programador desenvolver formas ótimas para se realizar a E/S. As duas últimas são equivalentes e tem amplo uso hoje em dia. Têm também a facilidade de esconder do usuário o que realmente ocorre para se realizar a operação.

5.2 Forma de atuação

O gerenciamento de E/S, independente de quem o realiza, é feito em duas etapas ou níveis: controle de E/S e controle de periféricos. O primeiro, mais alto, se preocupa com aspectos administrativos das solicitações de E/S realizadas, enquanto o segundo se preocupa com a operacionalização física desses pedidos.

Essa divisão faz com que o controle de periféricos trabalhe diretamente com o hardware envolvido, sendo necessário o desenvolvimento de uma interface diferente para cada dispositivo específico (são os drivers que instalamos no sistema). Ao fazer isso percebe-se que o S.O. não influencia esse controle, sendo o estudo do mesmo deixado para áreas mais ligadas à engenharia.

Já o controle de E/S independe de características elétricas/mecânicas do hardware por atuar apenas no controle lógico dos mesmos. Isso significa, em outras palavras, que o controle de E/S irá gerenciar filas de controle de acesso aos dispositivos de E/S e não executar a atividade de E/S propriamente dita. Isto possibilita, portanto, que um mesmo controle possa ser aplicado para toda uma família de dispositivos.

Esses dois níveis de controle se diferenciam ainda na forma em que tratam dispositivos que exijam acesso privativo (como impressoras, por exemplo) e dispositivos com acesso compartilhado (como discos ou rede, por exemplo). Em qualquer desses casos o controle de periféricos não faz distinção entre essas categorias de dispositivos por ser apenas o executor da operação de E/S. A diferenciação entre o que pode ter acesso compartilhado ou não é feita no controle de E/S, que é quem trata logicamente todas as solicitações realizadas.

5.3 Mecanismos de controle de E/S em disco

O controle de E/S em disco é feito de forma a otimizar a movimentação da cabeça de leitura/escrita e do próprio disco. Existem vários algoritmos propostos para a realização dessa otimização, sendo que a maioria deles procura otimizar o chamado tempo de busca (seek time) pela informação no disco. Para entender melhor esses algoritmos é preciso antes entender como um disco é dividido fisicamente, ou seja, entender o que são trilhas e setores de um disco e que os tempos de busca estão associados com o acesso às trilhas e os tempos de latência com os setores. O tempo de acesso ao disco é portanto a soma desses dois tempos. Como o primeiro é bem superior ao segundo, passa a ser a restrição mais importante no momento de escolher um algoritmo para controlar o acesso ao disco.

Os algoritmos de acesso a disco mais usados são o FIFO (que escolhe o próximo pedido a ser atendido pela ordem de requisição), SSTF (shortest seek-time first, que atende o pedido para a trilha mais próxima da atual), o SCAN (que varre o disco atendendo os pedidos linearmente) e suas variações, como C-SCAN, LOOK, C-LOOK, Modified Scan, etc.

5.4 Problemas com entrada não controlada

Fazer o gerenciamento de dispositivos de E/S não envolve atividades muito complexas quando quem dispara a operação for o sistema operacional. Entretanto isso não é verdade para operações de entrada de dados com determinados dispositivos, como mouse, teclado e redes. Nesses casos o sistema deve prover algum mecanismo para que a recepção de dados ocorra sem perdas, pois não se sabe quando eles chegarão, qual processo os deve receber, qual o tamanho do bloco de dados e como sinais de controle devem ser tratados.

O tratamento de quem deve receber os dados e de quantos bytes virão é feito com a identificação através de um cabeçalho (header) no bloco de dados (quando estivermos usando rede) ou pela identificação do dispositivo (quando não for rede). Já o tratamento de quando os dados chegarão pode ser feito com a criação de buffers e sinais de interrupção, com técnicas como as de toggle buffers ou buffers circulares.

Por fim, o tratamento de sinais de controle pode ser feito de forma imediata (com detecção dos mesmos) ou durante o esvaziamento do buffer (com a sua leitura pelo processo destinatário). Em qualquer das hipóteses existe perda de informação, ficando a cargo do projetista do S.O. identificar qual tipo de informação perdida é menos prejudicial aos sistemas em que ele será aplicado.

Capítulo 6

Camada Física

1. Introdução
2. Funções
3. Suporte físico
4. Codificação e modulação

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

- Computer Networks, Tanenbaum, cap. 2
- Redes de Computadores, Soares et alii, caps. 3, 4, 6 e 9
- Outros bons livros de redes de computadores, nos capítulos sobre camada física do protocolo OSI

6.1 Introdução

Após a discussão sobre os problemas no gerenciamento de E/S para o dispositivo de rede, se torna interessante voltarmos ao estudo dos protocolos de redes envolvidos efetivamente com o envio e recebimento de sinais na rede.

Como já examinado, os protocolos de comunicação são estruturados em camadas. Começaremos nosso estudo pela camada física, que é aquela que trata diretamente das especificações de como as máquinas se conectam fisicamente para permitir a comunicação.

6.2 Funções

Como a função primária da camada física é o estabelecimento de regras para a ligação mecânica/elétrica entre máquinas, pode-se estabelecer um conjunto de serviços que a mesma deve prover à camada de rede. Esses serviços dependem de algumas características físicas do ambiente de rede, como:

1. Definição do suporte físico

2. Definição de características eletromecânicas dos conectores

3. Definição de técnicas de codificação

4. Definição de técnicas de modulação

Dessas características não examinaremos aqui os detalhes eletromecânicos dos conectores (RJ-45, RS-232, RS-449, RS-422, etc.), uma vez que são dados tabelados e que não interferem na especificação conceitual de uma rede. As demais funções serão examinadas a seguir.

6.3 Suporte Físico

Suporte físico é, na realidade, o elemento no qual serão transmitidas as mensagens pela rede. Assim, os tipos de suporte físico disponíveis são o par trançado (cabos UTP), o cabo coaxial, a fibra ótica, e o próprio ar. As diferentes capacidades de transmissão e de imunidade a ruídos diferenciam a aplicação de cada tipo de suporte. Outra restrição ao tipo de suporte está ligada ao tipo de topologia adotado, uma vez que nem sempre é conveniente usar um determinado tipo de cabeamento para determinados tipos de topologia.

Outros aspectos a serem considerados são o custo do cabeamento, sua flexibilidade e sua capacidade mecânica (resistência física). Esse conjunto de fatores é que determina, no final das contas, qual tipo de suporte físico será adotado, exceto em aplicações específicas, quando uma característica especial acaba sendo determinante sobre as demais.

Uma consequência importante da escolha por um dado tipo de suporte físico e da topologia usada é o mecanismo a ser adotado para o controle de acesso ao meio. Diferentes combinações de suporte e topologia demandam diferentes técnicas para definir quem pode transmitir a cada instante. Os protocolos IEEE 802-xx fazem exatamente a definição dessas técnicas para redes em barramento, anel, sem-fio, etc.

6.4 Codificação e modulação

A transmissão de informações entre duas máquinas envolve, na prática, o envio de sinais (elétricos, eletromagnéticos ou óticos) ao longo do suporte físico. Temos então ondas, que sofrem sempre alguma forma de atenuação por parte do meio em que estão sendo transmitidas. Essa atenuação é proporcional a dois fatores: a frequência do sinal e do seu espectro de potência.

A transmissão de sinais em banda básica, isto é, em frequência e valores exatamente iguais ao sinal a ser transmitido, é a forma mais simples de se transmitir um sinal. Esse modo, porém, apresenta forte atenuação em função de seu espectro de potência. Isso faz com que o sinal em banda básica não possa ser transmitido para distâncias maiores. Além disso, no sinal em banda básica podemos ter longas sequências de zeros ou uns, o que fatalmente faria as máquinas comunicantes perderem o sincronismo entre si.

Para corrigir o problema de perda de sincronismo e do espectro de potencia ruim faz-se a **codificação** do sinal a ser transmitido. Existem vários códigos distintos que

podem ser usados, como Manchester, Miller, Diferencial, etc. O interessante nesses códigos é que todos procuram provocar transições no sinal transmitido, mesmo que o valor do bit na infomação permaneça constante

Finalmente, a **modulação** do sinal corrige o problema de alcance em função da frequência do mesmo. A modulação faz com que os sinais transmitidos sejam de uma frequência maior (a portadora) e que o sinal modulado seja depois recebido pela máquina destinatária do sinal, que o demodularia para extrair o sinal real, num processo idêntico ao de transmissão de rádio ou televisão. Essa modulação pode ser feita classicamente em amplitude, em frequência ou ainda em fase, conhecidos pelas siglas ASK, FSK e PSK. Alternativas mais recentes de modulação para transmissão de dados envolvem técnicas de telefonia celular, como CDMA, GSM, etc.

Capítulo 7

Camada de Enlace

1. Introdução
2. Funções
3. Sub-camada de enlace lógico (LLC)
4. Sub-camada de controle de acesso ao meio (MAC)
5. Bridges

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

- Computer Networks, Tanenbaum, cap. 3 e 4
- Redes de Computadores, Soares et alii, caps. 7, 8 e 9
- Outros bons livros de redes de computadores, nos capítulos sobre camada de enlace do protocolo OSI

7.1 Introdução

Dando continuidade ao estudo dos protocolos de comunicação, passamos para a segunda das camadas do protocolo OSI, que é, também, uma das mais importantes para um bom desempenho da rede.

Nesse capítulo será dado um enfoque especial em dois dos serviços prestados pela camada de enlace, que são o controle de acesso ao meio e o controle de enlace lógico entre as estações que estiverem se comunicando. Esses serviços, dentro dos protocolos IEEE 802-xx, estão definidos em duas subcamadas, que são a **LLC** (Logic Link Control) e a **MAC** (Medium Access Control), arranjadas de modo que a subcamada MAC fica abaixo da LLC, sendo responsável pelas diferentes formas de acesso ao meio para as diferentes topologias existentes.

7.2 Funções

A função primária da camada de enlace é fazer a conexão entre as máquinas que estiverem trocando informações. Essa tarefa implica em alguns serviços que podem ser oferecidos às camadas superiores que não têm, na realidade, relação direta com o enlace das máquinas. O principal deles é a verificação, e possível correção, de erros nos pacotes sendo transmitidos. As principais funções são:

1. Definição de como mensagens são trocadas
2. Definição de quadros (framing) a serem transmitidos
3. Definição de como são tratados os erros
4. Definição de sincronismo entre pacotes na transmissão
5. Definição de como os pacotes acessam o suporte físico

Examinaremos aqui primeiro os detalhes do tratamento de erros e da definição de quadros (itens 2 a 4), para então concluir com um estudo do controle de acesso ao suporte físico (item 5).

7.3 Enlace lógico

7.3.1 Detecção de erros

A transmissão de um sinal ao longo da rede pode sofrer interferências por vários motivos. Independentemente do motivo, essas interferências podem fazer com que valores transmitidos cheguem trocados em seu destino. É preciso, portanto, uma técnica que determine se a mensagem recebida possui ou não erros, para que o receptor da mesma possa tratá-la convenientemente. As formas de se fazer isso são:

- Bits de paridade e paridade transveral
- Códigos de Redundância Cíclica (CRC)

Em paralelo à detecção de erros existem formas de recuperar erros que não sejam de grandes proporções, tal como é feito usando-se o Código de Hamming.

7.3.2 Sincronismo de mensagens

Um problema comum em qualquer processo de comunicação é identificar quem e quando cada processo "fala". Esse problema é agravado na comunicação entre duas máquinas, pois podem ocorrer erros na transmissão dos bits que compõe os dados. Esses erros implicam na necessidade da máquina que transmite ter conhecimento sobre o sucesso ou não da transmissão para que possa dar continuidade ao processo de comunicação. Isso pode ser feito através de vários protocolos distintos de controle de fluxo, entre os quais temos:

- *Stop-and-wait*
- *Sliding window* (janela deslizante)

7.4 Controle de acesso ao meio

Até agora temos examinado a transmissão de mensagens de forma não concorrente, isso é, uma máquina usa o suporte físico sem ter que disputá-lo com outras máquinas. Isso, no entanto, é completamente irreal (e ineficiente ou muito caro, dependendo da topologia usada).

A prática é fazer com que o suporte físico seja usado de modo concorrente, com técnicas para controle de seu acesso dependendo da topologia usada na rede. Examinaremos aqui apenas protocolos para redes em barramento, anel e sem-fio, deixando as demais topologias entendidas como casos particulares dessas, que podem ser resolvidos pelas mesmas técnicas ou por técnicas de roteamento.

- Redes em barramento
  - Aloha
  - Família CSMA
  - Token bus
- Redes em anel
  - Token
  - Slotted
- Redes sem-fio
  - MACAW

7.5 Bridges

Para concluir esse capítulo é preciso que se fale em dispositivos que permitem a interconexão de redes. Aqui, em particular, falaremos de um dispositivo chamado *bridge*, que nada mais é do que um conversor de protocolos que permite que os endereços de máquinas tratados possam ser de redes vizinhas, bastando que esses endereços sejam conhecidos pela *bridge* (e não por todas as máquinas da rede. O uso de *bridges* permite também a conversão de protocolos, permitindo que uma máquina localizada em uma rede em anel "converse"com outra localizada numa rede em barramento.

Um cuidado especial com o uso de *bridges* é evitar a formação de laços na conexão entre sub-redes. Na realidade é impossível evitar a formação física de tais laços, uma vez que não temos como controlar quais redes estarão conectadas a quais outras redes. O que se faz é evitar a formação lógica desses laços, com a aplicação de algoritmos de árvores geradoras (**spanning tree**).

Capítulo 8

Camada de Rede

1. Introdução
2. Funções
3. Estratégia de conexão
4. Roteamento
5. Congestionamento
6. Interconexão de redes

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

- Computer Networks, Tanenbaum, cap. 5
- Redes de Computadores, Soares et alii, caps. 10 e 11
- Outros bons livros de redes de computadores, nos capítulos sobre camada de rede do protocolo OSI

8.1 Introdução

A terceira camada do protocolo OSI é também a última em que todas as máquinas da sub-rede tomam parte. É na camada de rede que as máquinas determinam, finalmente, se a mensagem é de fato endereçada a ela ou não.

Isso implica, na prática, no desenvolvimento de atividades de endereçamento e roteamento, para que os pacotes de dados possam chegar corretamente aos seus destinos e de forma eficiente.

8.2 Funções

Como a função primária da camada de rede é o tratamento de endereços, os serviços por ela prestados à camada de transporte são:

1. Definição de endereços
2. Definição da estratégia de conexão
3. Definição de rotas a serem seguidas pelos pacotes
4. Definição de técnicas para tratamento de congestionamentos

Dessas características não examinaremos aqui os detalhes de como se definem endereços (e como eles são tratados), deixando esse item para ser discutido dentro da camada de transporte, no contexto do protocolo TCP/IP. Nos concentraremos portanto no tratamento de rotas, o que envolve os itens 2 a 4.

8.3 Estratégia de Conexão

Como mencionado, a camada de rede se preocupa essencialmente com atividades de endereçamento e roteamento. Entretanto, uma condicionante bastante severa à atividade de roteamento é a forma como se estabelecem conexões entre as entidades que irão se comunicar. Essa conexão diz respeito ao formato em que se define um caminho na rede, o que pode ser feito de duas formas distintas:

1. **Circuito Virtual**, em que o caminho é definido no estabelecimento da conexão, não sendo mais alterado a partir daí. Todos os pacotes seguem sempre a mesma rota, independentemente de situações como perda de linha ou congestionamento. É uma forma mais simples de se fazer o roteamento.
2. **Datagrama**, em que o caminho é definido para cada pacote individualmente, sendo o pacote denominado datagrama. Aqui é possível tentar usar sempre o melhor caminho possível. Entretanto isso cria um custo computacional adicional, que nem sempre é desejável.

8.4 Roteamento

A transmissão de informações entre duas máquinas apenas é possível se existir uma ligação entre as mesmas. Se essas máquinas fizerem parte da mesma sub-rede (domínio) não existe maior preocupação em como se estabelecer tal conexão. Para máquinas em sub-redes distintas, muitas vezes bem distantes uma da outra, é preciso que se defina o caminho que deve ser percorrido pelos pacotes transferidos entre cada máquina. Ao processo de definição do caminho denominamos roteamento, que é, na prática, uma das operações mais caras no funcionamento de uma rede.

Existem diversas formas de se fazer o roteamento, tais como:

- *Hot potato*
- *Flooding*
- Caminho mínimo (Dijkstra, Bellman-Ford)
- Por tabelas estáticas (ou dinâmicas), etc.

8.5 Congestionamento

Um problema intrinsicamente associado ao roteamento é o de tratamento de congestionamentos. A relação entre os dois é bastante óbvia, pois um caminho é mais ou menos eficiente dependendo de quantos pacotes estão trafegando por ele num dado instante. É natural que quanto mais pacotes trafegando, menor será a folga do canal e, portanto, maior a possibilidade de redução na velocidade do tráfego naquele ponto da rede.

Dentro da camada de rede se estabelecem, portanto, serviços para o tratamento de congestionamentos, que podem tanto se aproveitar dos algoritmos de roteamento, quanto ter funcionamento dedicado e independente, como é o caso do algoritmo de *Leaky Bucket*.

8.6 Interconexão de redes

Como a camada de redes é a responsável pelo roteamento de pacotes entre as várias sub-redes, cabe a ela hospedar os dispositivos que executarão tal função. Esses dispositivos são denominados **roteadores** (routers), possuindo um alto grau de processamento e "inteligência". O funcionamento estável e eficiente de uma rede depende, fundamentalmente, da eficiência dos roteadores. A velocidade do tráfego, além de restrições locais, depende das capacidades de transferência dos roteadores, uma vez que todo o tráfego entre duas redes quaisquer deve, obrigatoriamente, passar por um ou mais desses dispositivos.

Apenas como lembrança, vale dizer que enquanto um *bridge* se ocupa de apenas duas sub-redes, sendo portanto capaz de transferir dados de uma para outra, um roteador é capaz de identificar endereços de redes que distam vários passos (conexões entre sub-redes) entre si, estabelecendo rotas para se transferir pacotes entre uma e outra, além de poder conectar simultaneamente diversas sub-redes.

Capítulo 9

Camada de Transporte e TCP/IP

1. Introdução
2. Funções
3. Conexão/desconexão
4. QoS
5. Endereçamento
6. Multiplexação
7. TCP/IP

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

- Computer Networks, Tanenbaum, cap. 6
- Redes de Computadores, Soares et alii, cap. 12
- Outros bons livros de redes de computadores, nos capítulos sobre camada de transporte do protocolo OSI e sobre TCP/IP

9.1 Introdução

A camada de transporte do protocolo OSI é a primeira em que apenas as duas entidades que querem se comunicar tomam parte. É por isso que uma de suas funções é o estabelecimento de um padrão de qualidade de serviço (QoS) sobre o qual a comunicação deve ocorrer.

Além do estabelecimento da qualidade de serviço, cabe ainda à camada de transporte a definição de como uma conexão deve ser estabelecida, mantida e encerrada. Também é na camada de transporte (em conjunto com a de rede) que se materializa o protocolo TCP/IP, que é, sem sombra de dúvida, o padrão dominante na Internet.

9.2 Funções

A camada de transporte trata, fundamentalmente, do processo de conexão entre duas entidades que irão se comunicar. Nesse processo, as principais funções por ela oferecidas às camadas superiores são:

1. Definição da Qualidade de Serviço (QoS)
2. Definição do processo de conexão e desconexão de uma sessão
3. Definição de endereçamento dos processos comunicantes associado ao protocolo TCP/IP

9.3 Protocolos de Conexão e desconexão

Como a camada de transporte é a primeira em que os pacotes transferidos entre as duas entidades que se comunicam só circulam nas máquinas em que estão executando, cabe a ela fazer o controle da conexão e desconexão entre essas máquinas. Por apenas envolver as máquinas origem e destino, os protocolos dessa camada são chamados de protocolos ponto-a-ponto.

Dentro desse cenário a operação do protocolo de conexão é bastante simples, requerendo apenas que se enviem mensagens solicitando a conexão, aceitando o pedido e confirmando o estabelecimento da conexão após acordos sobre o padrão de QoS a ser admitido. Todo esse processo é bastante bem definido através de primitivas chamadas TPDUs (Transport Protocol Data Unit), que possuem sintaxe e relações de dependência delimitadas.

Os problemas surgem, de fato, no momento da desconexão entre as duas máquinas, o que deve ocorrer de modo a evitar que uma delas permaneça acreditando que a conexão ainda está em atividade. Isso ocorre pela falta de capacidade de se identificar que o parceiro da comunicação está pronto para se desconectar sem que ele lhe envie uma mensagem sobre isso. Ao receber tal mensagem, é o seu parceiro que ficará na dúvida sobre se você recebeu, ou não, a última mensagem enviada. Esse processo de dúvidas prossegue *ad infinitum* caso um dos dois parceiros não admita, sem necessidade de uma mensagem, que o outro está pronto para a desconexão.

A esse problema se aplicam soluções de concordância (***Agreement***), em que o protocolo de desconexão deve ser feito de modo que, num dado ponto, um dos parceiros admita que o outro iniciou a desconexão real caso não receba mensagem indicando o contrário.

9.4 QoS

Durante o processo de estabelecimento de uma conexão devem ser definidos alguns detalhes sobre como a mesma transcorrerá. Um desses detalhes é a definição da qualidade de serviço (QoS), a qual, na prática, define os limites mínimos aceitáveis para certos parâmetros de desempenho para que a conexão seja mantida. Os valores estabelecidos de QoS durante a fase de conexão são, portanto, pisos de desempenho para a conexão. Dentro do protocolo OSI se definem diversas classes de QoS, segundo

patamares de erros e perdas de pacotes. A classe de uma rede (do ponto de vista de qualidade) é definido pela classificação ISO.

Além dos tipos de classe do protocolo OSI ainda se definem, durante o estabelecimento da conexão, parâmetros de qualidade como:

1. Atraso de estabelecimento de conexão
2. Probabilidade de falha de estabelecimento de conexão
3. Vazão (velocidade) da rede
4. Atraso de trânsito
5. Prioridade
6. Resiliência
7. Taxa de erros residuais

9.5 Endereçamento

O problema de endereçamento consiste em permitir que várias conexões sejam estabelecidas, por uma mesma máquina, com uma ou mais máquinas simultaneamente, ou seja, é preciso definir como cada processo envolvido em alguma conexão é diferenciado dos demais processos. Isso implica em que cada conexão deverá ter endereços específicos, que permitam à camada de transporte diferenciar entre uma conexão e outra. Isso é feito através de endereços de portas de serviço, conhecidas por TSAP (Transport Service Access Point). As entidades que querem trocar informações devem, então, especificar endereços TSAP em que efetivarão tais trocas. No caso particular do TCP/IP as TSAPs são implementadas através de *sockets* para identificar os pontos (portas) de acesso.

Em situações específicas, como acesso a serviços considerados padrões na internet (acesso a um sistema de arquivos, por ex.), esse processo de endereçamento faz uso de portas (TSAPs) pré-definidas para a conexão, que depois são redirecionadas para portas gerais. No protocolo TCP essas portas pré-definidas estão abaixo da porta 49151, como por exemplo o serviço de telnet na porta 23, ssh na porta 22 e http na porta 80.

9.6 Multiplexação

Como a camada de aplicação trata de multiplas conexões simultâneas e as camadas abaixo dela podem tratar apenas de uma informação por vez, é necessário definir como isso pode ser acomodado. A técnica usada é a de multiplexação das conexões, ficando a camada de aplicação responsável por definir qual porta (TSAP) acessará a camada de rede por vez.

9.7 TCP/IP

Finalmente, para concluir o estudo da camada de transporte falta examinar o funcionamento do protocolo TCP/IP. Na realidade, esse protocolo trabalha em cinco camadas, que são as camadas física, de enlace, de rede, de transporte e de aplicação. Dessas camadas todas é nas camadas de rede e de transporte que aparecem as diferenças significativas entre TCP/IP e OSI. Em específico, é na camada de rede que está regulado o protocolo IP, enquanto o TCP funciona na camada de transporte para trocas de dados que demandem o estabelecimento de conexão. Além do TCP existe também na camada de transporte o protocolo UDP, usado quando não existe necessidade de conexão.

9.7.1 IP

O protocolo IP é encarregado de definir endereços das máquinas que participarão integralmente do processo de comunicação. Para essa tarefa se define um cabeçalho (o **IP header**) contendo informações tais como endereços internet da fonte e do destino do pacote, tamanho e identificação do pacote, opções de serviço (como restrições de roteamento, nível de segurança, etc.), entre outros dados, que podem ser vistos com detalhes na maioria dos livros de redes de computadores.

Os endereços IP são endereços de 32 bits (no caso do IPv4), divididos em classes de redes, que especificam o número máximo de máquinas em uma rede específica. A classificação de endereços IP também pode ser vista em livros da área, assim como um detalhamento das especificações do IPv6 (nele os endereços são de 128 bits). Vale observar que o protocolo IPv4 vem sendo substituído pelo IPv6, com endereços de 128 bits e uma gama bastante maior de serviços oferecidos ao processo de roteamento e identificação de endereços.

Como endereços IP na realidade são obtidos a partir de nomes, através de um serviço de nomes de domínio, eles identificam uma máquina de forma lógica. A localização física de uma máquina, ou o mapeamento entre endereço lógico e físico, é distinto do endereço IP. Na prática cada máquina possui uma placa de rede que possui um endereço Ethernet (48 bits, único para cada placa existente no mundo). Então é preciso que se transforme endereços IP para endereços Ethernet e vice-versa. Isso é feito por um par de protocolos definidos na camada de rede, chamados ARP (*Address Resolution Protocol*) e RARP (*Reverse Address Resolution Protocol*), que fazem, respectivamente, o mapeamento entre um endereço IP para um Ethernet e entre um endereço Ethernet para um IP.

9.7.2 TCP

O protocolo TCP faz a definição dos serviços listados aqui para a camada de transporte, em especial os serviços de QoS e de endereçamento, além de prover primitivas de controle de conexão (TPDUs para estabelecer, manter e fechar uma conexão).

Parte desse serviço é definido através de um cabeçalho (***TCP header***), que define as TSAPs envolvidas na comunicação, padrões de qualidade para o serviço, etc. (mais uma vez, a descrição completa do cabeçalho pode ser vista nos bons livros da área).

Uma observação importante a fazer nesse ponto é que a camada de transporte deve fornecer serviços equivalentes para uma comunicação quer seja ela orientada a conexão quer não seja. Em específico, dentro do TCP/IP, o serviço orientado a conexão é provido pelo protocolo TCP, enquanto que o não orientado a conexão (connectionless) é provido pelo protocolo UDP.

9.7.3 UDP

O protocolo UDP é utilizado para trocas de mensagens em que não se necessite do estabelecimento prévio de conexão entre as máquinas comunicantes. Existem várias aplicações em que isso é interessante e até mesmo necessário, devendo ficar claro que quando a transmissão ocorre por UDP não existe a garantia de que a mensagem foi entregue com sucesso. Endereços UDP são os mesmos do TCP, sendo a diferença apenas a necessidade de conexão prévia. O cabeçalho UDP também é descrito nos bons livros de redes de computadores, não sendo detalhado aqui.

Capítulo 10

Camada de Aplicação

1. Introdução
2. Funções

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

- Computer Networks, Tanenbaum, cap. 7
- Redes de Computadores, Soares et alii, cap. 15
- Outros bons livros de redes de computadores, nos capítulos sobre camada de aplicação

10.1 Introdução

Antes de falar da camada de aplicação é preciso mencionar que entre essa camada e a de transporte existem, no modelo OSI, outras duas camadas, responsáveis pelo gerenciamento de uma sessão (a camada de sessão) e pelo gerenciamento do perfil dessa sessão (a camada de apresentação). A descrição dessas camadas pode ser vista nos livros que tratam mais detalhadamente o protocolo OSI. Aqui não as trataremos pois grande parte de seus serviços podem ser anexados aos serviços das camadas de transporte e de aplicação, como é feito no protocolo TCP/IP, usado pela grande maioria das redes em funcionamento.

Assumindo-se essa falta, voltamos nossa atenção à camada de aplicação, que é a responsável pela especificação dos diferentes serviços de comunicação a serem disponibilizados para o usuário da rede. As funções a serem descritas a seguir resumem, na realidade, uma série de aplicações em rede que permitem aos usuários desses serviços a realização de tarefas diversas.

Não existem, portanto, serviços na camada de aplicação que tenham a mesma funcionalidade dos serviços presentes nas camadas anteriores. Em cada aplicação são definidos protocolos específicos para a aplicação, tendo em comum apenas a interface com a camada inferior (apresentação no protocolo OSI ou transporte no TCP/IP). Dessa forma, o estudo da camada de aplicação pode ser entendido (e assim realizado) como o estudo de cada uma das aplicações existentes. Isso, é claro,

requer um esforço grande demais para ser efetivo nesse ponto do curso. Assim, apenas listaremos algumas das aplicações existentes, com suas características principais, deixando apenas a aplicação de segurança para ser discutida um pouco mais detalhadamente no próximo capítulo.

10.2 Funções

1. Transferência de arquivos

   O serviço de transferência de aquivos permite aos usuários movimentar (carregar ou depositar) arquivos de uma máquina para outra. Aplicativos dessa categoria incluem o *ftp* (ou *sftp* por razões de segurança) e, hoje em dia, navegadores *web*. O serviço oferecido consiste em estabelecer a conexão entre duas máquinas, reconhecer as permissões de acesso e transferir arquivos entre as máquinas através de pacotes de mensagens;

2. Troca de mensagens

   Esse serviço consiste em permitir que usuários se comuniquem de forma assíncrona através de mensagens de texto (com ou sem imagens ou outros arquivos anexados). Esse serviço é provido por aplicativos como e-mail, usenet news (muito tempo atrás) e twitter, entre outros.

   Para e-mails o envio de uma mensagem exige que as máquinas que representam os endereços fonte e destino se conectem para que o envio da mensagem ocorra de fato. O não estabelecimento da conexão implica em colocar-se a mensagem numa fila e tentar-se o seu envio repetidas vezes durante alguns dias.

   No caso de serviços de news o envio de uma mensagem não é feito para um usuário (ou lista deles) específico e sim para um mural, que pode ser consultado por quem desejar. Funciona como um quadro de avisos (*bulletin board*), em que se afixam notas que são lidas apenas pelos interessados. Essa é uma forma mais eficiente de se trabalhar com listas de distribuição, embora exija a existência de um servidor (e muitos espelhos) para o mural;

   Temos ainda serviços de troca de mensagem em tempo-real, tais como os *chats* da internet, mas esses pouco se diferenciam dos mecanismos anteriores, exceto pelo fato da troca ocorrer de modo mais instantâneo.

3. Acesso remoto

   Permite aos usuários acessarem máquinas remotamente. Os maiores problemas para acesso remoto estão ligados à segurança e ao fornecimento de um ambiente de trabalho confortável ao usuário. A segurança envolve aspectos bastante específicos e não serão tratados aqui e o ambiente de trabalho envolve o fornecimento de características de conversão entre terminais gráficos, que serão tratados no próximo item. Aplicativos desse tipo incluem *telnet* (que deve ser evitado) e *ssh*, entre outros;

4. Terminal virtual

Como dito acima, o serviço de terminal virtual consiste em fornecer ao usuário um ambiente em que ele possa fazer acesso remoto à uma determinada máquina, através da conversão de controles específicos de interface (tela, teclado, mouse, etc.) da máquina em que se está fisicamente o usuário e a máquina que é acessada remotamente. Terminais virtuais incluem xterm, vt100, etc.;

5. Execução remota

   Serviços de execução remota são, na realidade, uma especialização do serviço de acesso remoto em que o usuário apenas solicita que uma determinada tarefa seja executada remotamente. Eles são uma das características determinantes de sistemas distribuídos, sendo representados, por exemplo, pelo serviço de RPC (*Remote Procedure Call*) ou de RMI (*Remote Method Invocation* em Java);

6. Segurança

   Esse serviço será tratado com mais detalhes no próximo capítulo. Entretanto, para não deixar em falta o mínimo de informação, é preciso dizer aqui que segurança em redes de computadores significa fornecer meios de controle de acesso aos serviços da rede. Isso implica em controlar a autenticação dos usuários e das permissões de acesso que esses usuários terão aos conteúdos, serviços e recursos disponíveis na rede.

Capítulo 11

Segurança em Sistemas de Computação

1. Introdução
2. Autenticação
3. Controle de permissões

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

- Computer Networks, Tanenbaum, cap. 7
- Modern Operating Systems, Tanenbaum, cap. 4
- Operating Systems Concepts, Peterson/Silberschatz, cap. 11
- Outros bons livros de redes de computadores ou de sistemas operacionais, nos capítulos sobre segurança

11.1 Introdução

No capítulo anterior mencionou-se que segurança seria um dos serviços a serem providos pela camada de aplicação de um protocolo de rede. Entretanto, a segurança de um sistema computacional não difere muito se tratada do ponto de vista de rede ou de uma máquina isolada, uma vez que o problema básico de segurança é o controle de acesso, que pode ser dividido entre as atividades de autenticação do usuário e de controle de permissões. Essas duas atividades, exceto pelo escopo de sua atuação e vulnerabilidade, são semelhantes quer aplicadas para uma ou mais máquinas.

Nesse capítulo faremos uma revisão rápida dos principais problemas envolvidos em fornecer segurança a um sistema computacional (máquina ou rede), examinando inicialmente os problemas envolvidos com autenticação e depois aqueles referentes ao controle de permissões. **Saliente-se que o exame desse tópico aqui não deve, nem pode, ser considerado como um curso introdutório de segurança de computadores**.

Assim sendo, é interessante começar com uma lista de alguns dos tipos mais comuns de ataques perpretados contra sistemas de computação. São eles:

1. **Denial of Service** (DoS) ou **Distributed Denial of Service** (DDoS), em que uma máquina recebe um número exorbitante de solicitações de serviço causando sua paralização;

2. **Brechas de sistema**, em que se aproveita de bugs na implementação de um sistema operacional para se ganhar acesso ao equipamento. Isso envolve, em geral, uso de scanners que procuram na rede por sistemas que apresentem tais vulnerabilidades;

3. **Sniffing**, em que se escuta as mensagens circulando pela rede na busca de nomes de usuários e de suas senhas.

11.2 Autenticação

O serviço de autenticação do usuário é o responsável por permitir que o acesso lógico ao sistema seja habilitado em forma e grau determinados para cada usuário. Tanto do ponto de vista de acesso à rede quanto de uma máquina, o processo de autenticação envolve a identificação do usuário.

Dependendo do grau de restrição de acesso ao sistema, o processo de autenticação exige procedimentos distintos. Por exemplo, numa máquina isolada, instalada em um quarto de uma residência, a autenticação pode envolver simplesmente o acesso físico ao equipamento, através da chave da residência. Já num ambiente de máxima segurança esse processo pode envolver mecanismos sofisticados, tais como verificação de timbre vocal, iris do olho, impressão digital, DNA, etc.

Os problemas associados ao processo de autenticação se restringem basicamente à possibilidade de burla de senhas ou outros mecanismos quaisquer. Numa máquina cabe ao usuário cuidar que sua senha não seja facilmente descoberta e que a mesma seja trocada com uma certa frequência. Numa rede o problema é agravado com a possibilidade de escutas na rede (sniffers), que podem capturar nomes e senhas de usuários que estejam fazendo acesso remoto. Esse problema pode ser minimizado se os mecanismos de acesso remoto fizerem uso de criptografia, como é o caso do ssh, por exemplo.

Na prática, então, o controle de autenticação depende de um bom mecanismo de proteção de informações, que vai desde serviços de criptografia (existem inúmeras técnicas para se criptografar, como PGP, DES, RSA, etc.) até mesmo ao cuidado do usuário em evitar que sua senha seja óbvia ou que o vejam digitando-a.

No contexto de redes de computadores, esses cuidados dever ser ampliados para incluir autenticação de máquinas. Isso significa garantir que as máquinas em conexão são, de fato, as máquinas esperadas. Uma forma de se resolver isso é através do um serviço de DNS reverso de encaminhamento confirmado (FRrDNS), que procura garantir as identidades das máquinas envolvidas.

11.3 Controle de permissões

Uma vez que o usuário tenha ganho acesso ao sistema ainda é preciso controlar suas ações dentro dele. Isso envolve basicamente a verificação das permissões dadas

a ele, controlando o acesso do mesmo aos vários recursos do sistema (máquinas, aplicativos, comandos, arquivos) de acordo com tais permissões.

O modo de se realizar esse controle varia de sistema para sistema e de recurso para recurso. No caso de arquivos (e diretórios), por exemplo, o controle normalmente é feito através de informações armazenadas sobre cada arquivo, que dizem o tipo de acessibilidade que o mesmo tem. A mesma estratégia pode ser usada para o acesso a aplicativos e comandos do sistema.

Para o acesso a máquinas, a estratégia pode envolver o uso de mecanismos de autenticação explícitos ou ainda, como é o caso do UNIX, fazer uso de arquivos que delimitam o acesso de cada máquina ou aplicação (arquivos .allow e .deny, por exemplo).

Vale ainda indicar aqui que no caso de uma máquina, todo esse controle de segurança é feito pelo sistema operacional, através de seu módulo de gerenciamento de arquivos (o Sistema de Arquivos), que será examinado no próximo capítulo.

Capítulo 12

Sistema de Arquivos

1. Introdução
2. Funções
3. Proteção
4. Nomes e endereços

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

- Modern Operating Systems, Tanenbaum, cap. 4
- Operating Systems Concepts, Peterson/Silberschatz, cap. 10
- Outros bons livros de sistemas operacionais, nos capítulos sobre sistema de arquivos

12.1 Introdução

O sistema de arquivos é o módulo responsável, dentro do SO, pelo gerenciamento do conteúdo dos discos do sistema. Esse gerenciamento inclui tanto o controle de acesso aos mesmos (segurança), como o controle da localização e manuseio dos conteúdos de cada arquivo.

Aqui não examinaremos os aspectos relativos à segurança do sistema, que foram apresentados no capítulo anterior. Com isso nos concentraremos mais nos aspectos relativos aos arquivos propriamente ditos, iniciando esse estudo pela definição das funções do sistema de arquivos.

Uma última observação é que vários dos conceitos examinados aqui podem ser mapeados para os mecanismos de gerenciamento de memória (próximo capítulo), uma vez que este se ocupa também de controlar a localização e o acesso aos dados contidos na memória.

12.2 Funções

Como dito acima, as funções de um sistema de arquivos se concentram no controle dos arquivos em disco. Dentro desse cenário as principais funções por ele desempenhadas são:

1. Proteção
2. Mapeamento de nomes
3. Endereçamento
4. Segurança
5. Contabilidade

Dessas funções examinaremos aqui as três primeiras. A função de **segurança** foi examinada no capítulo anterior e faz, como foi visto, o serviço de autenticação de usuários e de controle de permissões.

Já a função de **contabilidade** não diz respeito especificamente aos arquivos. Ela é normalmente incluída dentro do sistema de arquivos por não se encaixar em nenhum outro módulo do SO e por ser mais fácil executar os serviços de contabilidade (quem usou o sistema, por quanto tempo, fazendo o que, etc.) a partir dos arquivos (textos, aplicativos, comandos) acessados. Em geral os serviços de contabilidade incluem a geração de arquivos de registro (logs) sobre os eventos ocorridos no sistema e estão voltando a ter mais presença com a introdução de ambientes de nuvem (*cloud computing*) e de hospedagem, em que se paga pelos recursos utilizados.

12.3 Proteção

O serviço de proteção é muitas vezes confundido com o de segurança. A principal distinção a ser feita entre os dois é que segurança se ocupa primordialmente de acessibilidade, enquanto proteção cuida de integridade (ou consistência) dos arquivos. Com esse enfoque a proteção de arquivos em um sistema deve se preocupar com dois tipos de falha: aquela que corrompe grande parte dos arquivos (falha de disco) e aquela que corrompe um arquivo de forma individual (falha de gravação).

Nas falhas de disco temos, normalmente, algum tipo de erro no disco em que o mesmo se torna ilegível. Problemas desse tipo ocorrem, em geral, por falhas de energia elétrica, crashs de disco, etc. A solução para os mesmos é o uso intenso e cuidadoso de **backups**, incluindo-se o armazenamento desses backups em locais fisicamente distantes dos discos originais (evitando que se perca o disco e seu backup num incêndio, por exemplo).

Um sistema decente deve fornecer dois tipos de backup ao seu administrador, que são os backups físico e lógico. No primeiro o backup é realizado copiando-se sequencialmente os bytes do disco, sem preocupação com o conteúdo de cada byte. No último o backup é feito copiando-se arquivo por arquivo do disco. Com isso, o backup físico é mais rápido mas força que o sistema fique inativo enquanto ele estiver em andamento. Já o backup lógico é mais lento mas necessita apenas do

bloqueio do arquivo sendo copiado durante o processo, permitindo que o restante do sistema permaneça em atividade.

Independente do modo do backup é bastante importante que eles sejam programados adequadamente. Embora o sistema de arquivos possa fornecer aplicativos que realizem essa operação de modo automático, cabe sempre ao administrador definir a periodicidade em que ocorrem os backups e quais arquivos devem ser copiados a cada vez. Isso permite que áreas diferentes do disco tenham políticas diferentes de backup, de acordo com a frequência em que seus conteúdos são alterados.

Já os problemas de falha de gravação surgem quando um usuário está salvando um arquivo. Se no momento da gravação ocorrer uma falha no sistema (falta de energia, por exemplo), o arquivo provavelmente se tornará inconsistente pela perda de links entre suas diversas partes (blocos). Isto ocorre pois como o disco é dividido em trilhas e setores e um arquivo pode ter que ocupar diversas trilhas e muitos setores, é necessário saber a sequência de setores ocupados. Essa sequência é armazenada e gerenciada pelo sistema de arquivos, sendo que pode se tornar totalmente inconsistente se uma falha ocorrer durante o processo de escrita do mesmo no disco.

Para diminuir (infelizmente não dá para eliminar) o risco de perda desse tipo de informação (trilhas e setores ocupados por um arquivo), usa-se técnicas como as páginas sombra (*shadow pages*), em que o salvamento de um arquivo em disco passa a ser feito em duas etapas. Na primeira criam-se as páginas sombra do que havia sido anteriormente salvo em uma outra área do disco, juntando-se também as páginas novas do arquivo. Se essa fase terminar com sucesso então passa-se a mapear o arquivo para esse novo conjunto de páginas, eliminando-se sua versão anterior. Desse modo, a única possibilidade de perda de integridade é se a falha ocorrer durante o processo de mapear o arquivo para o novo conjunto de páginas. Em qualquer outro ponto do processo o prejuízo máximo é o de se perder o que foi editado após a última operação de salvamento do arquivo.

Finalmente, o último problema de integridade de arquivos envolve o seu compartilhamento de modo simultâneo. Essa situação implica na existência de diversas cópias de um arquivo no sistema, o que exige cuidados para se garantir que todos os usuários que o estejam acessando tenham cópias atualizadas do mesmo e, ainda, que ao final do uso a cópia resultante seja a realmente pretendida. Esse problema é ampliado em sistemas distribuídos, quando podemos ter réplicas de um mesmo arquivo espalhadas por várias máquinas.

12.4 Nomes e endereços

Essas são, sem dúvida, as funções mais importantes prestadas pelo sistema de arquivos. Independentemente de haver controle de acesso ou garantia de integridade dos arquivos, é preciso localiza-los no disco. Isso é propiciado pelas funções de mapeamento de nomes e de endereçamento.

12.4.1 Mapeamento de nomes

Consiste em identificar os arquivos de forma única dentro de um disco, a partir de chamadas feitas pelos processos em execução no sistema. Isso significa ter condições

de diferenciar arquivos com mesmo nome que sejam de fato distintos (ou identificar um mesmo arquivo com nomes diferentes).

Parte desse trabalho é feito através da estruturação de diretórios, que hoje em dia estão organizados na forma de árvores hierárquicas, com subdiretórios com permissões de acesso diferenciadas para cada usuário. Então, quando um processo solicita um dado arquivo, o que se faz é buscar na estrutura de diretórios o arquivo pedido (segundo algum caminho de busca pré-definido).

Outras formas de se organizar diretórios incluem diretórios planos (um único raiz) e de usuários (sem possibilidade de compartilhamento). Entretanto essas formas não são mais usadas pois a estrutura em árvore é muito mais eficiente.

Uma outra tarefa do serviço de nomes é permitir o compartilhamento de arquivos, ou mais precisamente, controlar a abertura de arquivos evitando que cópias desnecessárias de um arquivo coexistam na memória da máquina. Isso implica em verificar se o arquivo já está aberto no momento em que é solicitado (e com isso mapear nomes a identificadores únicos desse arquivo) e controlar o seu fechamento, evitando que algum usuário perca o seu ponteiro para o arquivo.

12.4.2 Endereçamento

Trata de como os arquivos são alocados (e localizados) ao longo do disco. Em geral a técnica para resolver esse problema é fazer com que a própria estrutura de diretórios possua algum tipo de mapeamento entre nomes e localização física no disco. Isso é feito de forma diferente para cada sistema operacional, com vantagens e desvantagens em cada uma delas.

Dentro desse enfoque, o primeiro obstáculo a ser resolvido é o de determinar como os setores serão divididos. Como um arquivo pode não caber em um único setor, e a alocação de um setor a um arquivo implica que todos os bytes do setor estariam reservados para esse arquivo (que pode não ocupar nem 10% dele), é prática comum dividir o setor em blocos (ou páginas) de tamanho fixo, alocando-se tais blocos aos arquivos.

Assim, ao armazenar um arquivo temos que, na realidade, armazenar também os endereços dos blocos que ele ocupa (e a ordem desses blocos na sequência lógica do arquivo). O tamanho do bloco deve ser determinado de tal forma que não seja grande nem pequeno demais, pois blocos grandes desperdiçam muito disco, apesar de serem mais fáceis de gerenciar, e blocos pequenos dificultam o gerenciamento, apesar de ocuparem mais integralmente o disco. Tamanhos razoáveis, nos dias de hoje, vão de 1024 a 8192 bytes.

O segundo obstáculo é, então, o mapeamento dos endereços ocupados por um dado arquivo em disco. Como dito acima, isso é feito de maneira diferente em cada sistema.

No D.O.S. isso era feito através da FAT (*File Allocation Table*), que embora de implementação simples (a localização dos arquivos era feita por listas implementadas por índices de um vetor), era limitada e ineficiente para localizar páginas finais em arquivos grandes.

No UNIX isso é feito através da chamada **I-list**, que é uma lista cujos elementos (os **I-nodes**) ocupam, cada um, uma página do disco, independente de se referirem a um arquivo comum ou a um diretório. Nessa página aparecem os dados genéricos

do arquivo, tais como nome, data de criação, alteração, proprietário, permissões de acesso, tamanho, etc. Também aparecem os endereços das primeiras dez páginas do arquivo (ou dos primeiros dez i-nodes de arquivos de um diretório). Para arquivos maiores existem ainda outros três endereços, que funcionam em três diferentes níveis de indireção, proporcionando o endereçamento rápido de até 16 Gbytes num sistema com páginas de 1024 bytes.

Já as versões mais recentes do Windows (a partir do XP) adotam esquemas intermediários entre o D.O.S. e o UNIX, com várias das idéias presentes no UNIX mas ainda carregando o peso de ser compatível com a proposta da FAT. Neles adota-se o sistema NTFS, que é derivado do antigo Windows NT.

Capítulo 13

Gerenciamento de Memória

1. Introdução
2. Endereçamento
3. Alocação
4. Memória virtual
5. Desempenho
6. Visão global do sistema

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

- Modern Operating Systems, Tanenbaum, cap. 3
- Operating Systems Concepts, Peterson/Silberschatz, caps. 7 e 8
- Outros bons livros de sistemas operacionais, nos capítulos sobre gerenciamento de memória

13.1 Introdução

O gerenciamento de memória, em conjunto com o gerenciamento de processos, forma o que se pode chamar de coração de um sistema operacional. Sua importância reside fundamentalmente no fato do processador executar instruções trazidas da memória, sobre dados trazidos da memória e guardando resultados na memória. Assim, passa a ser crucial o gerenciamento de como a memória estará ocupada e como as informações nela serão reconhecidas pelo processador.

Isso se torna ainda mais crítico se considerarmos que a memória não é apenas o conjunto de bytes disponíveis na chamada RAM do computador. Existem, na realidade, cinco diferentes níveis de memória, diferindo em tamanho, custo e velocidade, que são:

1. Registradores internos da CPU

2. *Cache*
3. Memória principal
4. Memória secundária
5. Bibliotecas

O gerenciamento de memória se ocupa da memória principal e das suas interações com cache e memória secundária. O gerenciamento dos registradores e de boa parte da cache é feito pelo compilador, que ao gerar o executável já determina como os registradores serão ocupados e como a cache pode ser ocupada para acelerar o processamento. Já o gerenciamento das bibliotecas é feito pelo próprio usuário (ou administrador) do sistema, enquanto boa parte do gerenciamento do disco é feito pelo sistema de arquivos, que foi o tema do capítulo anterior.

Desse modo, o gerenciamento de memória se ocupa fundamentalmente do controle de quais dados vão para a memória, de que forma são nela armazenados e como podem ser acessados. Isso envolve atividades de **endereçamento**, em que se mapeia endereços de disco para endereços de memória, de **alocação**, em que se determina quais espaços serão ocupados por quem, e de **memória virtual**, em que se amplia o conceito de memória principal para um tamanho infinito. Trataremos cada um desses pontos a seguir, fechando o capítulo com um estudo de como características de gerenciamento de memória e processos influencia no desempenho de um sistema.

13.2 Endereçamento

O problema de endereçamento surge quando se percebe que os endereços ocupados por um programa no disco não correspondem aos que ele ocupará na memória. Isso fica ainda pior quando se examina o código do programa, em que aparecem instruções do tipo **JMP 6000H** (seis-zero-zero-zero em hexadecimal), em que a possição 6000H se refere a um dado deslocamento a partir da origem do programa, que possivelmente não se refere ao endereço 6000H da memória. Esses diferentes conjuntos de endereços recebem os nomes de espaços de endereços lógicos, no disco, e endereços físicos, na memória. O problema de endereçamento passa a ser, portanto, determinar uma estratégia de conversão entre endereços lógicos e endereços físicos. Essa transformação deve ser feita de forma que a execução do programa seja efetivada com sucesso ao carregar-se o mesmo na memória. As estratégias para efetuar esse mapeamento são:

1. **Endereçamento Absoluto**, em que os endereços físicos são absolutamente iguais aos endereços lógicos;
2. **Relocação Estática**, em que os endereços lógicos são transformados em endereços físicos, recalculados a partir da diferença entre as origens do programa em disco e na memória, no instante do carregamento do programa;
3. **Relocação Dinâmica**, que faz o cálculo do endereço físico apenas no momento da execução de cada instrução, a partir do conteúdo de um registrador especial (originalmente chamado de *fence register*);

4. **Segmentação**, que nada mais é do que o uso de relocação dinâmica aplicado ao conceito de se dividir o programa em partes menores, como código, dados, etc.

13.3 Alocação

Tendo definido o método de conversão entre endereços lógicos e físicos, é preciso tratar algo que ocorre momentos antes de se carregar um arquivo ou programa na memória (observem que daqui por diante chamaremos programas e arquivos indistintamente como sendo segmentos). Para que se saiba o endereço base de um segmento é preciso que se defina em que lugar da memória ele será alocado. Essa alocação deve ocorrer sob regras bastante bem definidas para que não ocorram problemas de interferência entre os processos. Existem duas estruturas básicas de alocação de memória, que são em **espaços contíguos** e em **blocos**, descritas a seguir:

13.3.1 Espaços contíguos

A alocação de memória em espaços contíguos é o modelo mais simples de alocação, em que para o segmento ir para a memória ele deve caber inteiro em um único trecho contínuo, com todos os seus bytes alocados de modo consecutivo. Embora simples esse modelo de alocação trás diversos inconvenientes, como fragmentação externa, necessidade de recompactação, mapeamento complexo e dificuldade na escolha do espaço livre a ser ocupado. Como se pode observar, a alocação em espaços contíguos trás mais problemas do que vantagens.

Algumas soluções tentadas para corrigir seus problemas envolvem a alocação em segmentos de tamanho fixo e segmentos de tamanho fixo reconfiguráveis (sistemas buddy). Infelizmente, apesar de tais modificações terem facilitado o gerenciamento, elas não eliminaram a fragmentação externa e ainda criaram a fragmentação interna.

13.3.2 Blocos

A solução para o problema de fragmentação veio com a organização de espaços usada em discos. Ao dividir-se a memória em blocos de tamanho fixo, e permitir-se que um segmento seja quebrado em vários blocos, eliminou-se definitivamente a fragmentação externa, uma vez que um segmento apenas deixaria de ser carregado para a memória caso esta não tivesse blocos livres suficientes para o segmento (esse problema foi resolvido posteriormente com a introdução de paginação e memória virtual).

Dentro dessa estratégia um segmento é dividido entre várias páginas, cada uma podendo ser alocada nos espaços livres de memória disponíveis no momento da alocação, independentemente de serem próximos ou não. Nesse esquema a fragmentação externa deixa de existir e a interna fica limitada ao tamanho de uma página. O problema nessa estratégia é calcular o endereço físico correspondente a um endereço lógico qualquer e evitar que se façam muitos acessos à memória em busca de um único dado. Isso é resolvido de modo eficiente por uma forma rápida

para transformação de endereços lógicos em físicos e o uso do nível de memória cache para acelerar os acessos, como descrito a seguir:

- Cálculo de endereço:
  Dado um endereço lógico qualquer, seu endereço físico correspondente é determinado dividindo-o pelo tamanho da página, resultando em um número inteiro (o quociente) correspondente ao número da página e outro inteiro (o resto) correspondente ao deslocamento em bytes dentro da página. O problema nesse processo é que para acessar o conteúdo do endereço físico correspondente ao dado endereço lógico temos que fazer três acessos à memória (um para localizar a tabela de páginas, outro para localizar a página específica e o terceiro para acessar o endereço desejado), o que tomaria muito tempo.

- Otimização de acesso:
  O problema de três acessos pode ser substancialmente reduzido se fizermos uso de memórias cache (ou memórias associativas), em que parte das informações necessárias estariam disponíveis em cache.

O uso de memória alocada em blocos melhora significativamente o desempenho da memória. Entretanto ela não resolve a restrição de que se um segmento ocupar mais páginas do que as que estão livres, então ele não pode ser colocado para executar.

Entretanto essa restrição é desnecessária, uma vez que se sabe que durante a execução de um programa ele não precisa acessar todas as suas instruções e todos os seus dados ao mesmo tempo. Então, se for possível deixar na memória apenas aquilo que está sendo necessário num dado instante, poderemos usar melhor os espaços disponíveis. Isso é feito através do mecanismo de memória virtual, examinado a seguir.

13.4 Memória Virtual

A solução para a alocação de segmentos maiores do que o espaço disponível na memória (ou até mesmo maior que ela toda) veio com um dos conceitos mais importantes de otimização de programas e sistemas, que é o **Princípio da Localidade**. Este princípio diz que os endereços de memória não têm probabilidade igual de acesso, sendo mais provável que após executar uma instrução da página x, que acesse um dado da página y, é muito mais provável que a próxima instrução também esteja na página x e também acesse dados na página y.

Desse modo, se um segmento ocupa muitas páginas, seria possível dizer que se estamos acessando um certo número delas num dado instante, então nos manteríamos acessando-as por mais algum tempo, não necessitando portanto que as demais páginas estivessem na memória.

Esse é o princípio de memória virtual. Seu funcionamento ocorre através de paginação por demanda, isto é, uma página vai para a memória apenas no momento em que é requisitada por um processo. O problema passa a ser, então, o que fazer se não houver mais páginas livres na memória. A solução é escolher uma das páginas alocadas para sair da memória, liberando portanto seu espaço. Essa operação é

conhecida como ***swapping***, em que se faz o *swap-out* de uma página (a escolhida para sair) e o *swap-in* de outra (a demandada).

Antes de discutirmos o funcionamento de memória virtual precisamos definir alguns termos importantes para fazer a análise dos vários algoritmos de escalonamento. Esses termos são:

- *Falta de página*, que é o evento que ocorre quando se precisa acessar um endereço de uma página que não está na memória;
- *Conjunto residente*, que é o conjunto das páginas que estão na memória em um dado instante;
- *Tamanho do conjunto residente*, é o número de páginas ocupadas (pelo sistema ou segmento) num dado momento;
- *Sequência de referência*, que é uma sequência de páginas que deverão ser acessadas pelo sistema ao longo do tempo;
- *Anomalia de Belady*, que é o indesejável fenômeno de ocorrer aumento no número de faltas de página ao se aumentar o tamanho conjunto residente.

Todo o processo de memória virtual passa a ser o gerenciamento de operações de swapping, procurando obter o melhor resultado possível a partir do princípio da localidade. Existem diversos algoritmos propostos para fazer essa escolha, dentre os quais temos :

1. **FIFO**, que escolhe para sair a página que entrou na memória há mais tempo e está sujeito à anomalia de Belady.
2. **LRU**, que é um algoritmo de pilha em que o critério de escolha da página indica que a página excluída será aquela que não é referenciada há mais tempo.
3. **Optimal**, também é um algoritmo de pilha, mas escolhe para sair a página que levará mais tempo para ser novamente necessária.
4. **Clock-FINUFO**, que faz uma implementação simplificada do LRU, tomando por base valores aproximados dos reais quanto ao último acesso à página.
5. **Segunda chance**, que é similar ao FINUFO, porém a página escolhida para sair teria que ter os bits de acesso e de modificação zerados.

Além desses algoritmos diversos outros foram propostos. Alguns trabalhando com o conceito de que cada processo teria número fixo de páginas, quando a página a ser retirada seria sempre do processo que precisa de uma nova página. Outros com o conceito de tempo fixo entre faltas de páginas, em que o número de páginas de cada processo é variado para que o intervalo entre duas faltas consecutivas seja mantido constante.

13.5 Desempenho

O uso de memória virtual permite que mais segmentos sejam carregados na memória por vez, o que permite um aumento no número de processos executando. A partir disso é interessante perceber algumas situações que afetam o desempenho do sistema. Antes porém é preciso definir alguns conceitos básicos:

- Nível de multiprocessamento - indica o número de processos executando no sistema
- Taxa de falta de páginas - indica o número de faltas de páginas ocorridas no sistema
- Ocupação da CPU - indica a porcentagem de tempo em que a CPU está executando processos de usuários

Considerando esses aspectos, temos que:

- Quanto mais processos executando melhor o nível de ocupação da CPU, uma vez que quando um processo é interrompido para fazer E/S ou por bloqueio, temos vários outros para assumir seu lugar na CPU;
- Quanto mais processos executando mais falhas de páginas temos, uma vez que cada processo passa a ocupar menos páginas e, com isso, passa a ser mais provável que uma página requisitada não esteja na memória;
- Um número elevado de processos executando acaba tendo um péssimo nível de ocupação da CPU, apesar de com o crescimento do número de processos teríamos uma maior ocupação, mas com muitos processos é possível que todos tenham tão poucas páginas que ficam o tempo todo causando falta de páginas e, com isso, não podem ocupar a CPU. Essa situação recebe o nome de ***thrashing***.

Assim, temos algumas diretrizes para avaliar o desempenho de um sistema. Basta verificar taxas de ocupação da CPU, número de faltas de páginas (ou por quanto tempo o sistema fica ocupado realizando paginação), taxa de ocupação dos dispositivos de E/S, etc.

Com essas medidas em mãos um analista pode sugerir mudanças no sistema, como aumento de memória, troca de dispositivos de E/S, troca de mecanismos de *caching* (preenchimento da cache com dados), entre outras.

O importante aqui é lembrar sempre que uma análise cuidadosa do sistema, levando em consideração todas as variáveis possíveis e as diretrizes anteriormente indicadas, é uma tarefa que se bem executada resulta em ganhos consideráveis ao sistema e ao seu financiador, devendo portanto ser uma tarefa bastante bem remunerada.

13.6 Visão global do sistema

Agora que terminamos a análise individual de cada um dos mecanismos do SO podemos ter fazer uma avaliação de seu funcionamento global. Tomaremos como exemplo a ocorrência de uma falta de página e das ações decorrentes dela.

Na ocorrência de uma falta de página, identificada pelo gerenciamento de memória, é necessário:

- ativar ações do gerenciamento de processos, para bloqueio do processo que gerou a falta;
- ativar ações do gerenciamento de memória para definição de qual página sairá da memória, já marcando-a como fora da memória para evitar acessos incorretos;
- ativar ações do sistema de arquivos para a localização das páginas envolvidas no *swapping*;
- ativar ações do gerenciamento de disco para realizar fisicamente o *swapping*;
- ativar, quando necessário, os mesmos mecanismos quando da conclusão das ações disparadas.

Disso percebe-se que existe muita interação entre os vários mecanismos do SO, o que apenas é possível de ser feito eficientemente se forem implementados modularmente, na forma de *threads*, por exemplo.

Capítulo 14

Sistemas Distribuídos

1. Introdução
2. Características
3. Paralelismo
4. Transparência
5. Gerenciamento
6. Aplicações

Os tópicos a serem examinados podem ser encontrados nos seguintes capítulos de livros:

- Modern Operating Systems, Tanenbaum, cap. 9 em diante
- Operating Systems Concepts, Peterson/Silberschatz, caps. 12, 13 e 14
- Outros bons livros de sistemas operacionais, nos capítulos sobre sistemas distribuídos

14.1 Introdução

Para fechar o conteúdo dessa disciplina temos que voltar às suas origens, ou seja, computação de alto desempenho e baixo custo. Como foi dito no início, isso pode ser obtido com o uso de sistemas distribuídos, que nada mais são do que conjuntos de máquinas operando em rede. Daquela definição passamos, durante o semestre, ao estudo dos dois componentes básicos desses sistemas, que são as redes de computadores e os sistemas operacionais.

Nesse capítulo examinaremos como usar esses componentes em conjunto para obter um sistema distribuído. Isso implica em examinar como os mecanismos de gerenciamento de um sistema operacional devem ser modificados se o seu objeto de gerenciamento envolver uma rede de computadores. Pela sua complexidade, o estudo detalhado deste tópico fica para o curso de **Projeto de Sistemas Operacionais**, bastando nesse momento uma revisão rápida desses sistemas.

14.2 Características

Um sistema distribuído é mais do que uma rede com servidores espalhados pela mesma (como algumas vezes é dito). Ele envolve certas características que não aparecem numa rede comum, como:

1. Transparência, que é a característica de livrar o usuário de qualquer conhecimento sobre como o ambiente executa suas tarefas;

2. Processamento paralelo, que é a capacidade do sistema em oferecer multiprocessamento efetivo caso o usuário necessite (o que é o necessário para computação de alto desempenho), possivelmente sem o conhecimento do usuário sobre isso.

14.3 Paralelismo

O paralelismo num S.D. é dito de grão largo ou, mais precisamente, fracamente acoplado. Isso significa que o acoplamento (pontos de troca de informação) entre os processos é reduzido. Essa característica vem como consequência de o processo de troca de informações na rede ser bastante lento, sendo no máximo 100 Mbits/s em Ethernet comum, ou 320ns por palavra (considerando atraso zero no protocolo de rede e que seja possível transmitir apenas 4 bytes, o que é claramente um absurdo), contra 30ns no acesso a uma palavra na memória.

Essa lentidão faz com que se evite ao máximo a troca de informações entre processos, ou como se diz, que o grão de execução sequencial seja mais largo.

Técnicas de como realizar o processamento paralelo se agrupam em modelos mestre-escravo, *bag-of-tasks* e *interacting-peers*, sendo que sua programação envolve basicamente a construção de processos que interagem por mecanismos de exclusão mútua e sincronismo. O estudo detalhado desses mecanismos é feito no curso de Programação Concorrente.

14.4 Transparência

O conceito de transparência é o mais importante em um sistema distribuído. Como dito há pouco, transparência implica em o usuário não precisar saber como o sistema trabalha, o que, num sistema distribuído, resulta em vários tipos de transparência, como por exemplo:

1. **execução**, quando o usuário não sabe em que máquina o seu programa está executando. Isso ocorre graças ao conceito de migração de processos, em que o sistema move programas (e/ou dados) de uma máquina para outra tentando obter um melhor balanceamento de carga, desempenho, ou mesmo a viabilização de execução (por falta de recursos locais);

2. **acesso**, em que o usuário acessa o sistema de forma equivalente, independentemente do ponto em que esteja fazendo isso;

3. **paralelismo**, quando o próprio sistema decide sobre a conveniência ou não de se paralelizar a execução de uma tarefa.

14.5 Gerenciamento

Um sistema distribuído apresenta restrições ao seu gerenciamento que não estão presentes num sistema operacional convencional. Como mencionado, o estudo mais detalhado de um S.O.D. ocorrerá em Projeto de Sistemas Operacionais. Entretanto, precisamos indicar aqui algumas dessas restrições, separadas por módulos de gerenciamento:

1. Gerenciamento de processos - precisa levar em consideração o fato de que os processos por ele controlados estão em máquinas distintas, exigindo mecanismos mais complexos para o controle de exclusão mútua e sincronismo (pela impossibilidade de uso de memória compartilhada), além de mecanismos para balanceamento de carga entre as máquinas e migração de tarefas;

2. Gerenciamento de memória - precisa cuidar do fato de termos a possibilidade de compartilhamento de informações entre memórias distintas, o que dificulta enormemente o gerenciamento das *caches* distribuídas, apenas para citar uma complicação;

3. Gerenciamento de E/S - que não é muito diferente do convencional, pois é possível imaginar a rede (e tudo que é acessível através dela) como um dispositivo de E/S. O único problema é controlar dispositivos diretamente conectados à rede (impressoras com placas de rede, por exemplo), mas mesmo esse é de simples solução;

4. Sistema de arquivos - como agora envolvem arquivos em várias máquinas, temos problemas na localização desses arquivos e na manutenção de sua consistência se houverem múltiplas cópias (réplicas) do mesmo.

14.6 Aplicações

Além das aplicações tradicionais de alto desempenho, sistemas distribuídos podem ser (e são) usados em sistemas de bancos de dados (os chamados SBDD), de automação e controle (tempo-real e redes de sensores), imagens, etc.

Não examinaremos tais aplicações aqui, mas é importante mencionar que o uso de SBDD, STRD, etc., tem aumentado significativamente nos últimos anos, especialmente com a introdução maciça de *clusters* de estações e arquiteturas dedicadas (*Beowulf*, por exemplo), além de grades e *cloud computing*, o que indica que o mercado de trabalho para profissionais com esse tipo de conhecimento (e esses são raros) deve aumentar bastante.

Dentro da ênfase de sistemas de computação esse conhecimento vem dos cursos de **Projeto de Sistemas Operacionais**, em que se examina como gerenciar um S.O.D.; **Redes de Computadores**, que examina os detalhes de um protocolo de redes e seu projeto e gerenciamento; **Arquiteturas e Organização de Computadores**, em que são vistos os aspectos organizacionais de uma máquina (convencional ou não-convencional) e como ela pode ser usada mais eficientemente; e **Programação Concorrente**, com o estudo de técnicas de construção de programas paralelos e cooperativos.

Outros cursos, como os de **Sistemas de Tempo-Real**, com o exame das características necessárias ao controle de processos que demandem execução sob restrições de tempo; **Modelagem de Sistemas**, em que se estudam técnicas para a construção de um modelo computacional para um sistema real; e **Técnicas de Simulação**, que trata do estudo de como modelos podem ser simulados para a análise e projeto de sistemas, são considerados acessórios ao corpo central da ênfase, mas igualmente importantes para a formação mais completa do profissional de sistemas de computação.