Sumário

- Desempenho de sistemas de comunicação.
- Arquitecturas para comunicações.
- Ambientes de desenvolvimento e execução.
- Software de comunicações.

Desempenho

- O desempenho é um aspecto crítico no desenvolvimento de sistemas de comunicação.

  O sistema de comunicação pode ser o ponto de estrangulamento de um sistema distribuído.

- Três importantes aspectos de desempenho são:

  Débito: quantidade de informação por unidade de tempo.

  Latência: atraso entre o envio e a recepção.

  Previsibilidade: grau de variação dos factores anteriores.

Factores limitativos ao desempenho

- Débito dos elos.
- Acesso a recursos partilhados: incluindo o meio.
- Filas de espera.
- Perca de mensagens.
- Processamento.
- Protocolos de estabelecimento de ligação.

Aspecto a privilegiar

- Infelizmente, alguns aspectos são dífices de conciliar.

  Para obter previsibilidade é necessário mascarar as faltas e isso obriga ao envio de mais mensagens, reduzindo o débito.

- Diferentes sistemas têm ênfases diferentes.

Obtenção de desempenho

- Para obter desempenho satisfatório é necessário:

  Obter recursos suficientes.

  Escolher a configuração correcta.

  Utilizar as técnicas de concretização adequadas.

Recursos

- É impossível obter bom desempenho sem os recursos necessários.

  Fundamental estimar a carga a que o sistema vai ser sujeito: frequência das mensagens, distribuição no tempo, tamanho.

  Obter recursos suficientes: meios de comunicação, memória para armazenamento de mensagens, capacidade de processamento.

Configuração

- Escolher os protocolos adequados.

  Usar só os níveis necessários.

  Usar o método de acesso ao meio mais apropriado.

  Mecanismos de controlo de fluxo em função da distribuição do tráfego.

- Configurá-los correctamente:

  Valores dos temporizadores. Parâmetros como tempo de rotação do testemunho no token-bus, número de máquinas por segmento de rede na Ethernet, etc.

  Técnicas analíticas ou tentativa-e-erro (afinação).

Estudo analítico do token-bus

- N é o número de nós.
- R taxa de transmissão em bits por segundo.
- Mensagens chegam com uma distribuição exponencial com tempo de chegada λ.
- $E[X]$ é o tamanho médio das mensagens em bits.
- $\tau/3$ é o tempo de propagação médio no cabo.
- N_m é o número médio de mensagens que um nó tem para enviar (a descobrir).
- T_c é o tempo médio de rotação do testemunho (a descobrir).

Estudo analítico do token-bus (cont.)

- Taxa de transmissão em mensagens por segundo:

$$\mu_T = R/E[x]$$

- Tempo de passagem do testemunho:

$$\omega = X_t/R + \tau/3$$

Estudo analítico do token-bus (cont.)

- (1) Tempo médio rotação do testemunho:

$$T_c = N(N_m E[X]/R + \omega)$$

- (2) Número médio de mensagens que um nó tem para enviar (lei de Little):

$$N_m = \lambda T_c$$

Estudo analítico do token-bus (cont.)

- Combinado as fórmulas anteriores (1 e 2):

$$T_c = \frac{N\omega}{1 - N\frac{\lambda}{\mu_T}}$$

- Se $N\lambda > \mu_T$ (mais mensagens do que é possível enviar) a fórmula não se aplica.

  Porque, existe um tempo máximo para a passagem do testemunho.

  No entanto, as filas vão encher.

Concretização

- Fazer código a pensar no desempenho.
- Usar uma ambiente de desenvolvimento apropriado.

  Bibliotecas, sistema operativo, etc.

- Escolher a arquitectura adequada.

Codificação

- Evitar cópias de informação desnecessárias.
- Têr cabeçalhos "pré-fabricados" semi-prenchidos.
- Optimizar o "caminho crítico".

  Reduzir o número de instruções para o caso mais comum.

Cópias de informação

- Tentar passar os dados por referência entre diversos módulos.
- Se necessário, reduzindo os mecanismos de protecção.
- Escrever directamente nos blocos de memória que vão ser utilizados pelo controlador da rede.

Cabeçalhos pré-fabricados

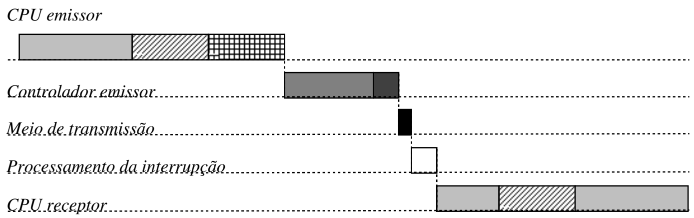
- Cada nível protocolar acrescenta um cabeçalho a cada trama enviada.
- Parte desta informação é invariante para um número elevados de tramas.

  Endereço de origem, tipo de trama, etc.

- Copiar toda esta informação em bloco e depois só alterar os campos que variam.

  Em vez de preencher o cabeçalho campo a campo, em cada transmissão.

Caminho crítico



Bibliotecas

- Concretizações optimizadas de funcionalidades frequentemente necessárias.

  Gestão de temporizadores.

  Gestão eficiente de memória (adição e remoção de cabeçalhos a mensagens, alocação e libertação de blocos de tamanho fixo, etc).

- Suporte à modularidade.

  Mecanismos eficientes de troca de informação entre módulos de software.

Gestão de memória

- Reserva e libertação de blocos de memória de tamanho variável é mais difícil que a gestão de blocos de tamanho fixo.

  Problema da fragmentação de memória.

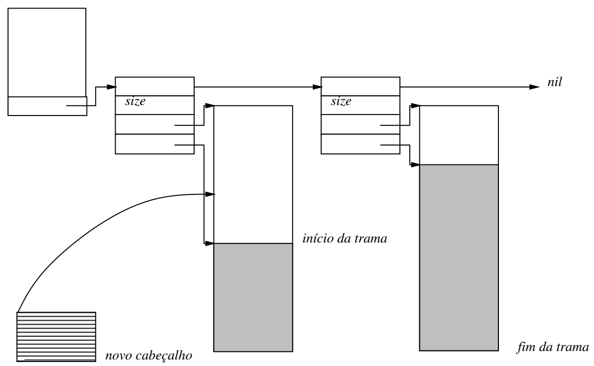
- Nas arquitecturas de comunicação usam-se frequentemente, colecções pré-reservadas de blocos do mesmo tamanho.

  Se os blocos são muito grandes existe fragmentação interna.

  Bloco deve ter o tamanho suficiente para conter a mensagem típica.

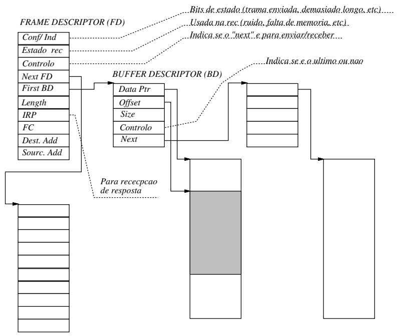
  Mensagens maiores podem ser memorizadas ligando vários blocos.

Estrutura das mensagens



Estrutura usada pelo TBC

Token Bus Controller (Motorolla 68824)



Suporte à modularidade

- Aproximação pouco eficiente: *concretizar cada nível protocolar como um processo, que comunica com os níveis adjacentes através de caixas do correio.*

  Grande número de trocas de contexto e cópias de informação.

- Geralmente, os vários níveis executam-se no mesmo espaço de endereçamento.

  Suporte de software para configurar o sistema (indirecções, tabelas e escalonamento de eventos, etc).

Configuração

```
/* chama sempre a mesma funcao */
codigo_nivel_x (msg) {
  ....
  codigo_nivel_y (msg);
  ....
}
/* outra alternativa:
   vai a uma tabela ver qual a funcao a chamar
   (esta tabela pode ser carregada durante a configuraçao) */
codigo_nivel_x (msg) {
  ...
  (*nivel_seguinte[NIVEL_X])(msg);
  ...
}
```


Sistemas operativos

- Para maior eficiência os protocolos são geralmente executados em modo previligiado, fazendo parte do núcleo do sistema.

- Quando se utilizam processadores dedicados para as comunicações, recorre-se a núcleos eficientes, apenas com as funcionalidades requeridas.

  Tais como os núcleos multi-tarefa leccionados na cadeira.

Arquitectura

- Hardware versus software.

- Acesso à memória (DMA, etc), comunicação entre componentes.

Hardware versus software

- De um modo geral, concretizações em hardware são mais eficientes do que concretizações em sofware.

  No entanto, têm flexibilidade limitada e maior custo fixo.

  São sobretudo utilizadas para concretizar normas/serviços bem definidos, com pouca probabilidade de evolução e elevado número de utilizadores.

- Existem no mercado controladores para a maioria das redes locais, os quais concretizam em harware os protocolos até ao nível MAC.

Sumário

- Factores que limitam o desempenho.
- Configuração do sistema.
- Análise de desempenho.

  Exemplo: tempo de rotação no token-bus.

- Arquitectua:

  Software e hardware.

- Codificação.