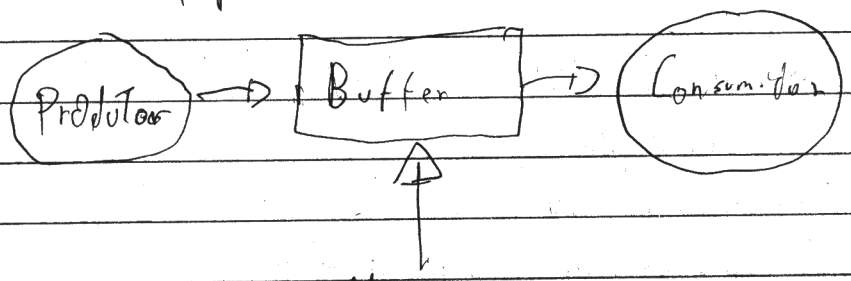
3) Produtor / Consumidor

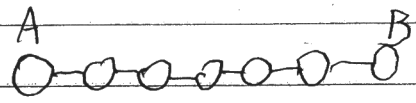
Um produz, outro consome, podem estar organizados em uma pipeline.



Níveis críticos
Cheio / Vazio

Exemplos: Unix Pipes
cat lixo.txt | sort | uniq

Transmissão de Mensagens em Caminhos



Tamanho L
$\alpha + L\gamma$.

M

α - Start up
γ - Inverso da banda passante

Dividir a mensagem em p pedaços

$$\alpha + \frac{L}{p}\gamma + \alpha + \frac{L}{p}\gamma$$

4) Cliente / Servidor

Padrão dominante em sistemas distribuídos.
Cliente: Solicita serviços, espera pela resposta
Servidor: Aguarda e processa pedidos (1 por vez ou multi-thread) se em máquinas diferentes usa RPC.

5) Interação entre pares

Ocorre em programas distribuídos quando existem vários processos que executam praticamente o mesmo código, e trocam mensagens para realizar uma tarefa

Exemplos: multiplicação de matrizes distribuída

⇒ troca de mensagens

Mestre-escravo

Calcular $a \times b$ onde a e b são matrizes $n \times n$ usando n processos um por máquina

```
process worker [i = 0 to n-1] {
    double a[n]; # linha i da matriz a
    double b[n,n]; # matriz b
    double c[n]; # linha i da matriz c
    receive (valores de a e b)
    for [j = 0 to n-1] {
        c[j] = 0.0;
        for [k = 0 to n-1]
            c[j] = c[j] + a[k] * b[j,k];
    }
    send (vetor c);
}
```

```
process coordinator {
    double a[n,n];
    double b[n,n];
    double c[n,n];
    inicializa a e b
    for [i=0 to n-1] {
        send(linha i de a para worker[i]);
        send(matriz b para worker[i]);
    }
    for [i=0 to n-1]
        receive(linha i de c do worker[i]);
    imprime resultados
}
```

send/receive - primitivas para troca de mensagens Bloqueantes

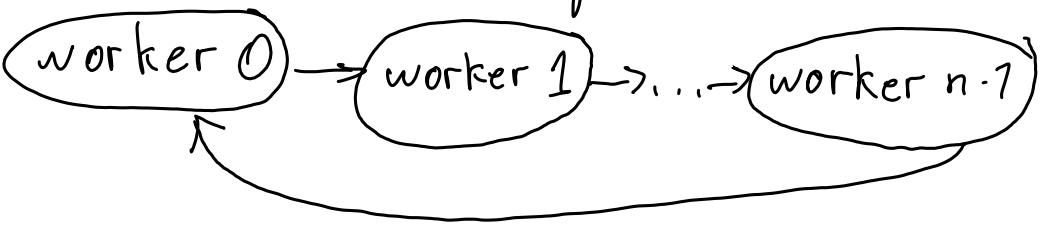
coordinator: nor nindr, fazer o primeiro com CO e colocar fora o segundo, trocando o por broadcast

broadcast - difusão

gosiping - troca completa

Outro exemplo: pipeline circular

Suponha que inicialmente cada processo possua apenas a linha i de a e a coluna i de b. Para calcular a solução, as colunas de b têm que circular



```
process worker [i=0 to n-1]{
    double a[n] b[n], c[n];
    double sum = 0.0,
    int vertcol = i;
    receive (linha i de a e coluna i de b)
    # calcula c[i,i]
    for [k=0 to n-1]
        sum = sum + a[k]*b[k];
    c[vertcol] = sum;
    for [j=1 to n-1] {
        send (coluna b para o próximo worker)
        receive (coluna b do worker anterior)
        sum = 0.0;
        for [k=0 to n-1]
            sum += a[k]*b[k];
        if (vertcol == 0)
            vertcol = n-1;
```

```
    else
        nertcol = nertcol - 1;
    c[nertcol] = num;
  }
  send(c para o coordenador);
}
```