

PROGRAMAÇÃO DISTRIBUÍDA

SISTEMAS DISTRIBUÍDOS

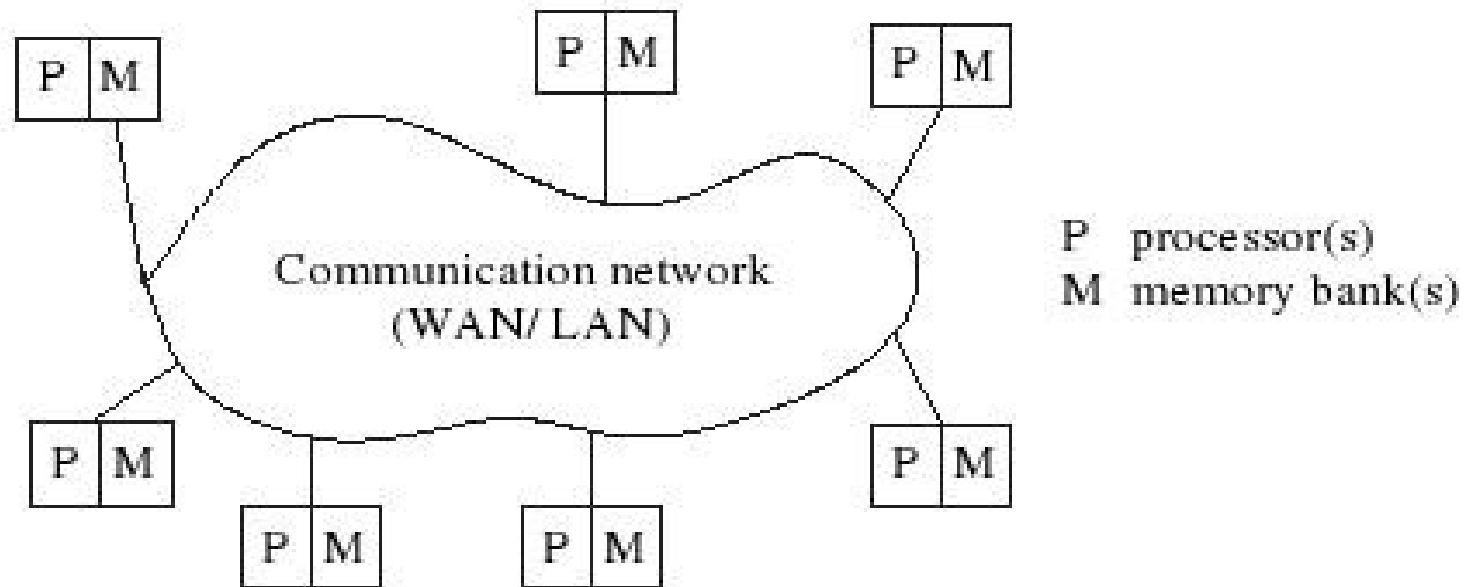
- . Conjunto de computadores ligados em redes
 - . Comunicação e sincronização de ações são feitos por passagem de mensagem
- . **Características:**
 - . Concorrência/Múltiplos processos
 - . Cada computador executa processos autônomos e concorrentes
 - . Não há um sinal de *clock* físico comum
 - . Sincronização é feita por troca de mensagens
 - . Limitação da precisão: Não há noção de tempo correto

- . Não há memória compartilhada
 - . Computadores não compartilham fisicamente a mesma memória
 - . Acessos a memória não-local deve ser feita via rede por troca de mensagens
- . Separação geográfica
 - . Computadores do sistema são fisicamente separáveis
 - . Podem estar distantes ou não
- . Meta coletiva comum
 - . Devem ter um objetivo comum a ser atingido

- . Autonomia e heterogeneidade
 - . Computadores são fracamente acoplados
 - . Tem diferentes velocidades
 - . Usam sistemas operacionais diferentes
 - . Não fazem parte de um sistema dedicado, mas cooperam oferecendo serviços e resolvendo um problema juntos.

ARQUITETURA DE UM SISTEMA DISTRIBUÍDO

- . Constituído de computadores conectados por rede
- . Cada computador possui memória local
- . Acesso a dados/serviços de outro computador somente via rede



MOTIVAÇÕES PARA APLICAÇÕES DISTRIBUÍDAS

- . Ambiente computacional geograficamente distribuído
 - . Muitas aplicações são inerentemente distribuídas
 - . Os agentes envolvidos estão separados geograficamente
 - . Rede bancária: Operações podem ser realizadas em qualquer agência; Transferências entre bancos, etc.
 - . Internet
- . Compartilhamento de recursos
 - . Recursos podem não ser replicados em todos os lugares
 - . Recursos alocados em um único lugar pode gerar um gargalo
 - . Devem ser distribuídos ao longo do sistema

- . Aumentar a confiabilidade

- . Permite distribuir e replicar recursos pelo sistema
- . Os recursos estão separados geograficamente
- . Dificultam a ocorrência de falhas no sistema

- . **Aspectos a serem procurados:**

- . **Disponibilidade:** Recursos estão sempre disponíveis
- . **Integridade:** Garantir que dados e estado dos recursos estejam corretos
- . **Tolerância a falhas:** Habilidade de recuperar as falhas do sistema

- . Aumentar a proporção desempenho/custo
 - . Com recursos compartilhados, o acesso é feito em computadores distintos, com melhor desempenho que um sistema centralizado e com custo inferior
- . Escalabilidade
 - . É possível acrescentar novas máquinas sem fazer um gargalo na rede
- . Modularidade e expansão incremental
 - . Os computadores são independentes quanto a arquitetura e desempenhos, desde que rodando

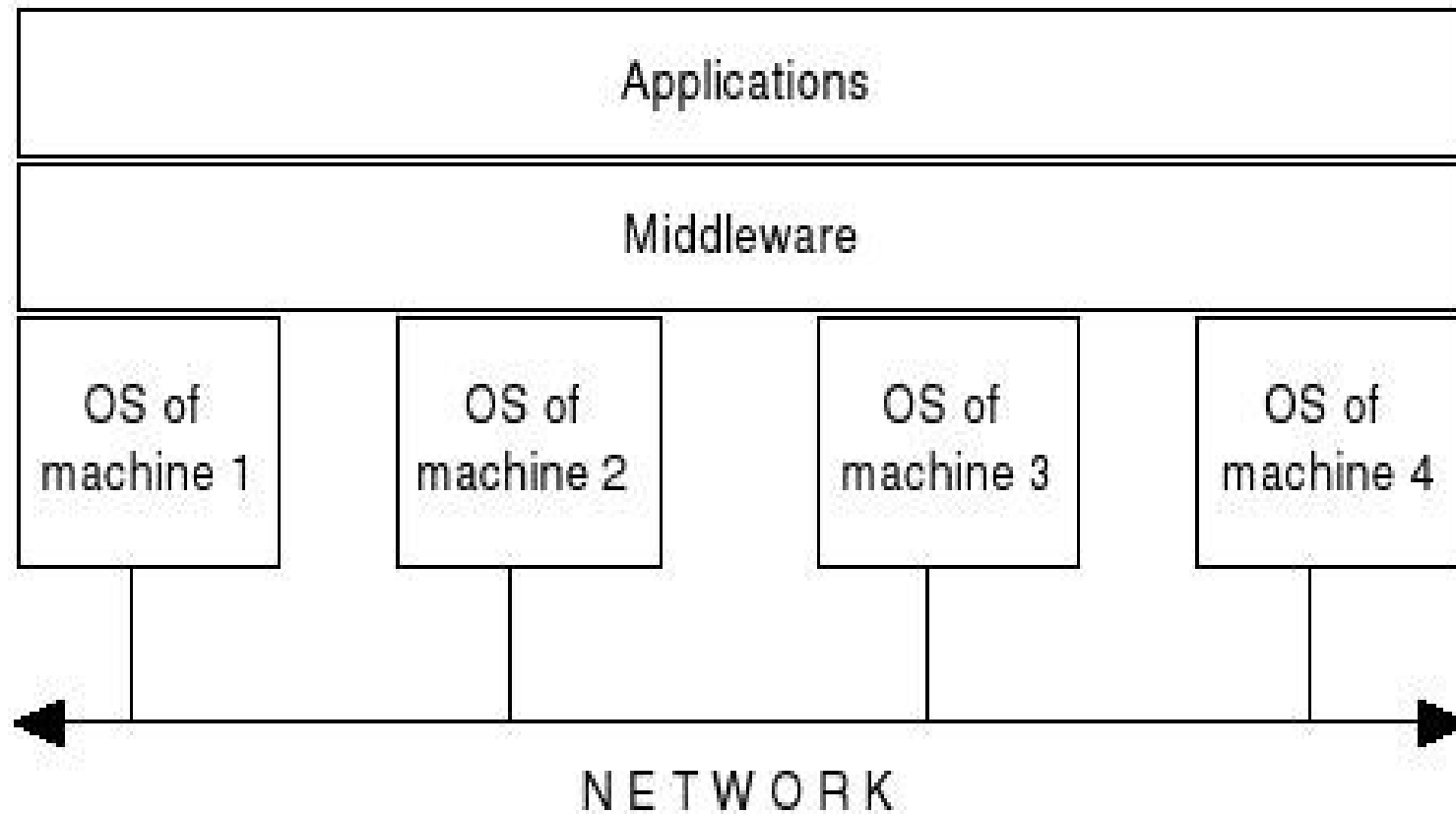
EXEMPLOS DE APLICAÇÕES DISTRIBUÍDAS

- . World Wide Web
- . Servidor de arquivos de redes
- . Sistema de rede bancária
- . Redes peer-to-peer (P2P)
- . Redes de sensores
- . Grade (grid) computacional

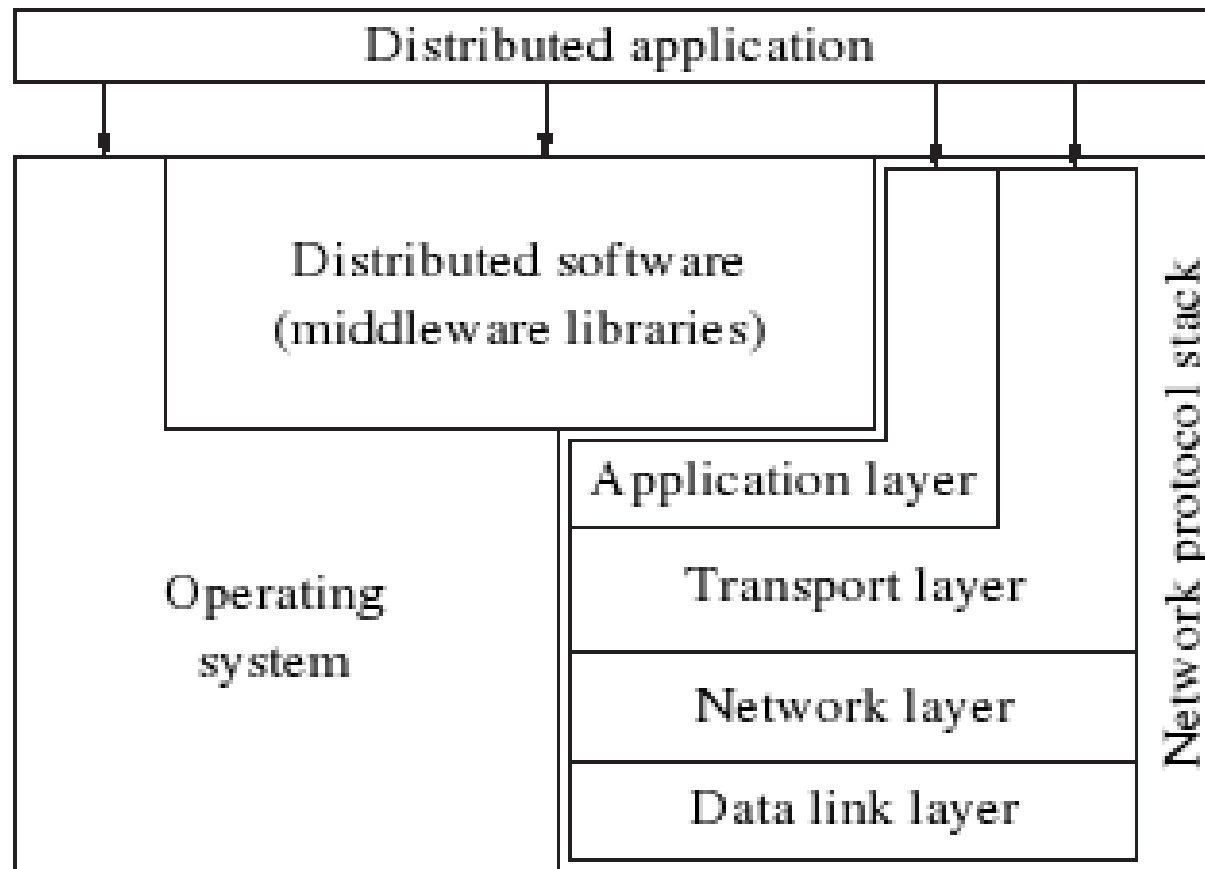
MIDDLEWARE

- . Camada de software (biblioteca) que:
 - . Atua como intermediário com o sistema
 - . Fornecem portabilidade, transparência e interoperabilidade
 - . Simplificam a implementação dos algoritmos
 - . Separam a aplicação do protocolo de comunicação
 - . Fornecem serviços para:
 - . Localizar os demais computadores
 - . Localizar os demais processos
 - . Fazer a comunicação com segurança

Situação Ideal:



Situação Comum:



PRIMITIVAS DE COMUNICAÇÃO DISTRIBUÍDA

- `send()`

- Usada para enviar dados para outro processo
- Contém dois parâmetros
 - **Destino** - identifica processo/máquina que os dados serão enviados
 - **Buffer** - Contém os dados a serem enviados
- Pode ser chamada com as opções:
 - **bufferizada** - Dados são transferidos do buffer da aplicação para um buffer do kernel (SO) antes do envio
 - **Não bufferizada** - Dados enviados diretamente do buffer da aplicação

- . `receive()`
 - . Usada para receber dados de outro processo
 - . Contém dois parâmetros
 - . **Origem** - identifica processo/máquina dos quais os dados serão recebidos (pode ser curinga)
 - . **Buffer** - Onde os dados a serem recebidos serão armazenados na aplicação
 - . A primitiva `receive()` sempre é bufferizada no kernel
 - . Os dados podem chegar antes da chamada da função

TIPOS DE PRIMITIVAS DE COMUNICAÇÃO

- . As primitivas podem ser:

- . **Síncronas**

- . As primitivas send() e receive() executam juntas
 - . A execução da send() só é completada quando uma correspondente receive() foi iniciada e terminada

- . **Assíncronas**

- . A primitiva send() termina quando os dados são copiados do buffer da aplicação
 - . Não faz sentido definir uma primitiva receive() assíncrona. (porquê?)

- . **Bloqueantes**

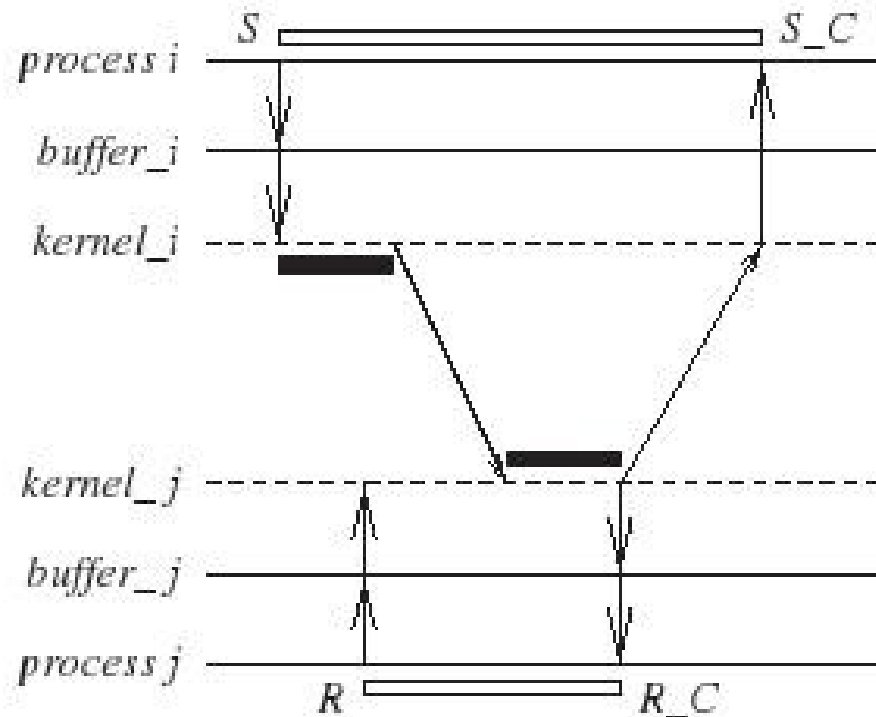
- . A primitiva é bloqueante se o fluxo de execução só retorna para o processo que a chamou após a conclusão da primitiva

- . **Não bloqueantes**

- . O fluxo de execução retorna imediatamente para o processo que chamou a primitiva
- . A primitiva retorna um manipulador que permite verificar a conclusão da primitiva

- . Os tipos síncrono/assíncrono podem ser combinadas com bloqueante/Não bloqueante.

Ilustração: `send()` síncrona bloqueante e `receive()` bloqueante.



■ Duration to copy data from or to user buffer

▬ Duration in which the process issuing send or receive primitive is blocked

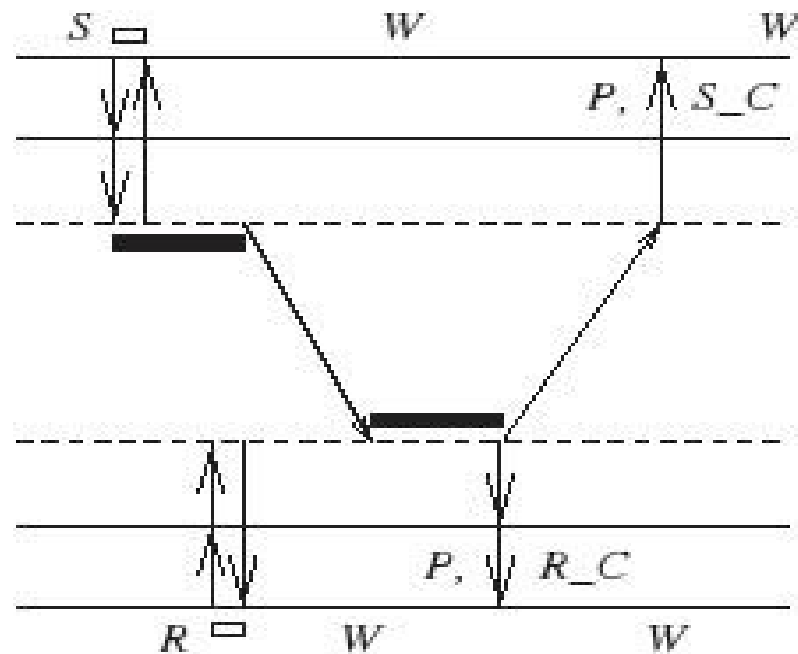
S *Send* primitive issued *S_C* processing for *Send* completes

R *Receive* primitive issued *R_C* processing for *Receive* completes

P The completion of the previously initiated nonblocking operation

W Process may issue *Wait* to check completion of nonblocking operation

Ilustração: send() síncrona não bloqueante e receive() não bloqueante.



■ Duration to copy data from or to user buffer

□ Duration in which the process issuing send or receive primitive is blocked

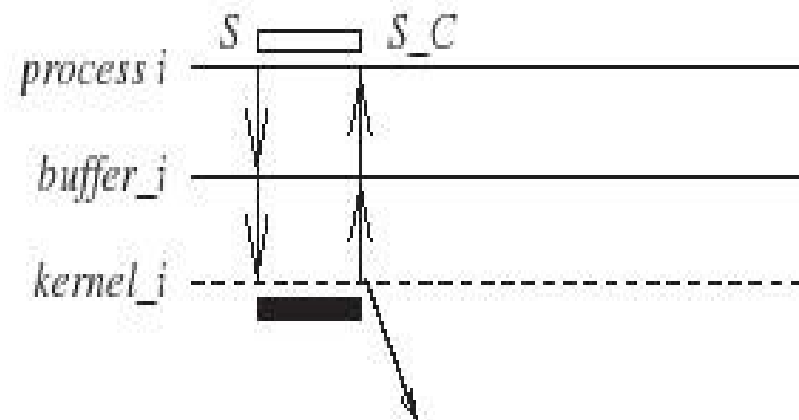
S Send primitive issued *S_C* processing for Send completes

R Receive primitive issued *R_C* processing for Receive completes

P The completion of the previously initiated nonblocking operation

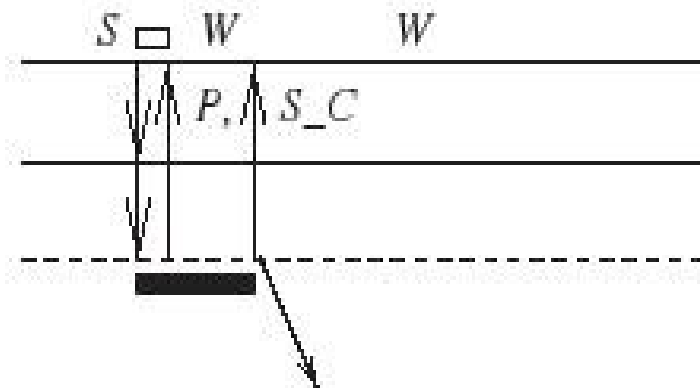
W Process may issue Wait to check completion of nonblocking operation



Ilustração: send() assíncrona bloqueante



- Duration to copy data from or to user buffer
- ▢ Duration in which the process issuing send or receive primitive is blocked
- S* *Send* primitive issued *S_C* processing for *Send* completes
- R* *Receive* primitive issued *R_C* processing for *Receive* completes
- P* The completion of the previously initiated nonblocking operation
- W* Process may issue *Wait* to check completion of nonblocking operation

Ilustração: send() assíncrona não bloqueante



	Duration to copy data from or to user buffer		
	Duration in which the process issuing send or receive primitive is blocked		
<i>S</i>	<i>Send</i> primitive issued	<i>S_C</i>	processing for <i>Send</i> completes
<i>R</i>	<i>Receive</i> primitive issued	<i>R_C</i>	processing for <i>Receive</i> completes
<i>P</i>	The completion of the previously initiated nonblocking operation		
<i>W</i>	Process may issue <i>Wait</i> to check completion of nonblocking operation		