Sistemas Distribuídos

Modelos de sistema

António Rui Borges

Resumo

- *Modelos*
- *Tipos de modelos*
- *Modelos arquitetônicos*
  - *Servidor cliente*
  - *Comunicação entre pares*
  - *Assinante-editor*
- *Primitivas de comunicação*
- *Modelos fundamentais*
  - *Interação*
  - *Falha*
  - *Segurança*
- *Leitura sugerida*

Modelos

Sistemas distribuídos destinam-se a operar no mundo real, o que significa que devem ser projetados para funcionar adequadamente mesmo quando submetidos a uma ampla variedade de ambientes operacionais e/ou enfrentando cenários difíceis e ameaças previsíveis.

Um usa *descrição modelos* para enumerar as propriedades comuns e o design suposições características de uma determinada classe de sistemas.

Assim, o objetivo de um modelo é fornecer uma descrição simplificada e abstrata, mas consistente, de características relevantes do sistema em discussão.

Tipos de modelos

- *modelos arquitetônicos*–eles definem a forma como os diferentes componentes do sistema são mapeados nos nós da plataforma paralela subjacente e como eles interagem entre si
  - *mapeamento sábio*: visam estabelecer padrões eficientes de distribuição de dados e cargas de processamento
  - *interação inteligente*: descrevem o papel funcional atribuído a cada componente e seus padrões de comunicação
- *modelos fundamentais*–eles discutem as características sistêmicas que afetam a confiabilidade
  - *tipo de interação*: eles lidam com questões como largura de banda de comunicação e latência
  - *tratamento de falhas*: especificam o tipo de falhas que podem ocorrer nos processos intervenientes e canais de comunicação
  - *segurança*: eles discutem possíveis ameaças que são colocadas ao sistema distribuído e que afetam seu desempenho

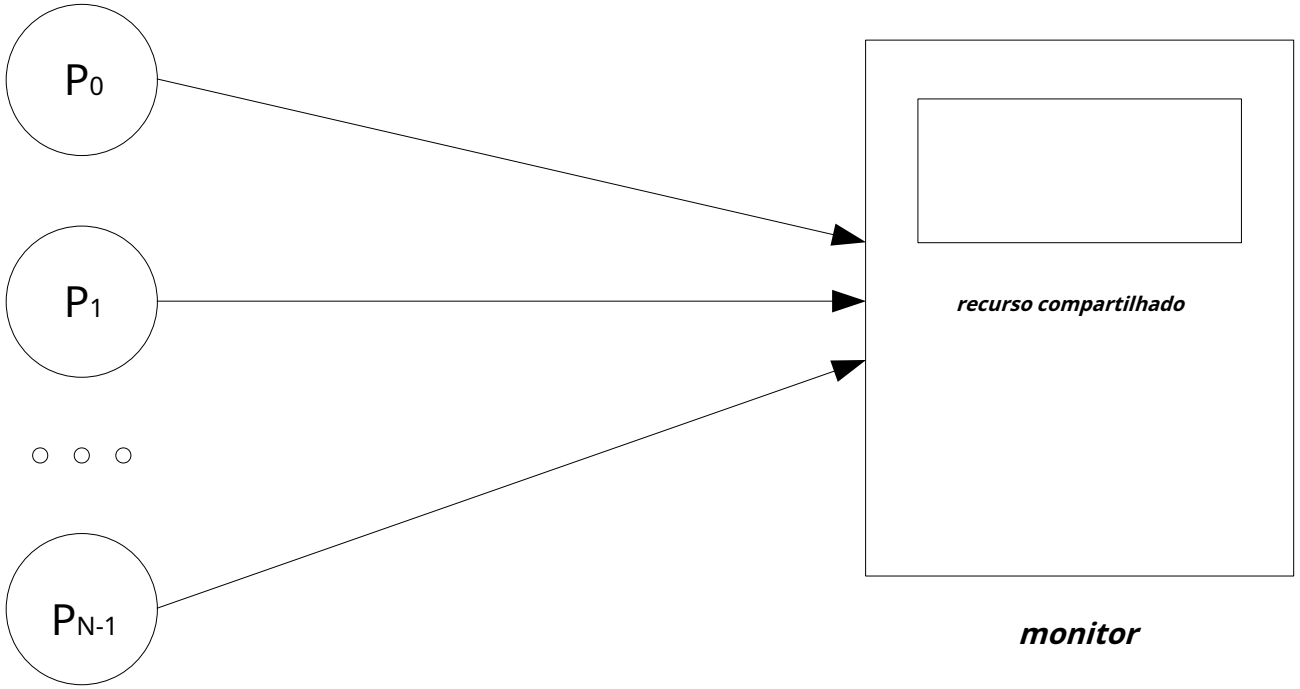
Modelos arquitetônicos

Em um *sistema distribuído*, os processos intervenientes cooperam entre si na realização de uma tarefa comum. A divisão de responsabilidades e o mapeamento em nós de processamento específicos da plataforma paralela subjacente estão entre as questões mais distintivas do projeto.

Uma abordagem de trabalho consiste em considerar primeiro uma solução concorrente para o problema (válida num sistema informático monoprocessado) e realizar a seguir um conjunto de transformações que conduzem à migração da solução para uma rede informática.

Ao projetar a solução concorrente, qualquer um dos paradigmas de comunicação pode ser usado, *variáveis compartilhadas* ou *passagem de mensagem*, pois são equivalentes. O primeiro, entretanto, leva a implementações mais intuitivas.

Solução simultânea - 1



processos cooperativos

Solução simultânea - 2

- o recurso compartilhado pertence ao espaço de endereçamento de todos os processos intervenientes
- a *monitor* protege o acesso ao recurso, impondo a exclusão mútua
- a sincronização do processo é feita dentro do monitor através *variáveis de condição*
- o modelo de interação é *reativo*: cada processo é executado até o bloqueio ou encerramento
- a interação em si é baseada na chamada de operações no monitor
- alternativamente, se a linguagem de programação não tiver uma semântica de simultaneidade, o recurso compartilhado pode ser protegido por um *acesso* semáforo (ou algum dispositivo similar) e a sincronização também sejam realizadas por semáforos, localizados agora fora da região crítica para evitar impasses

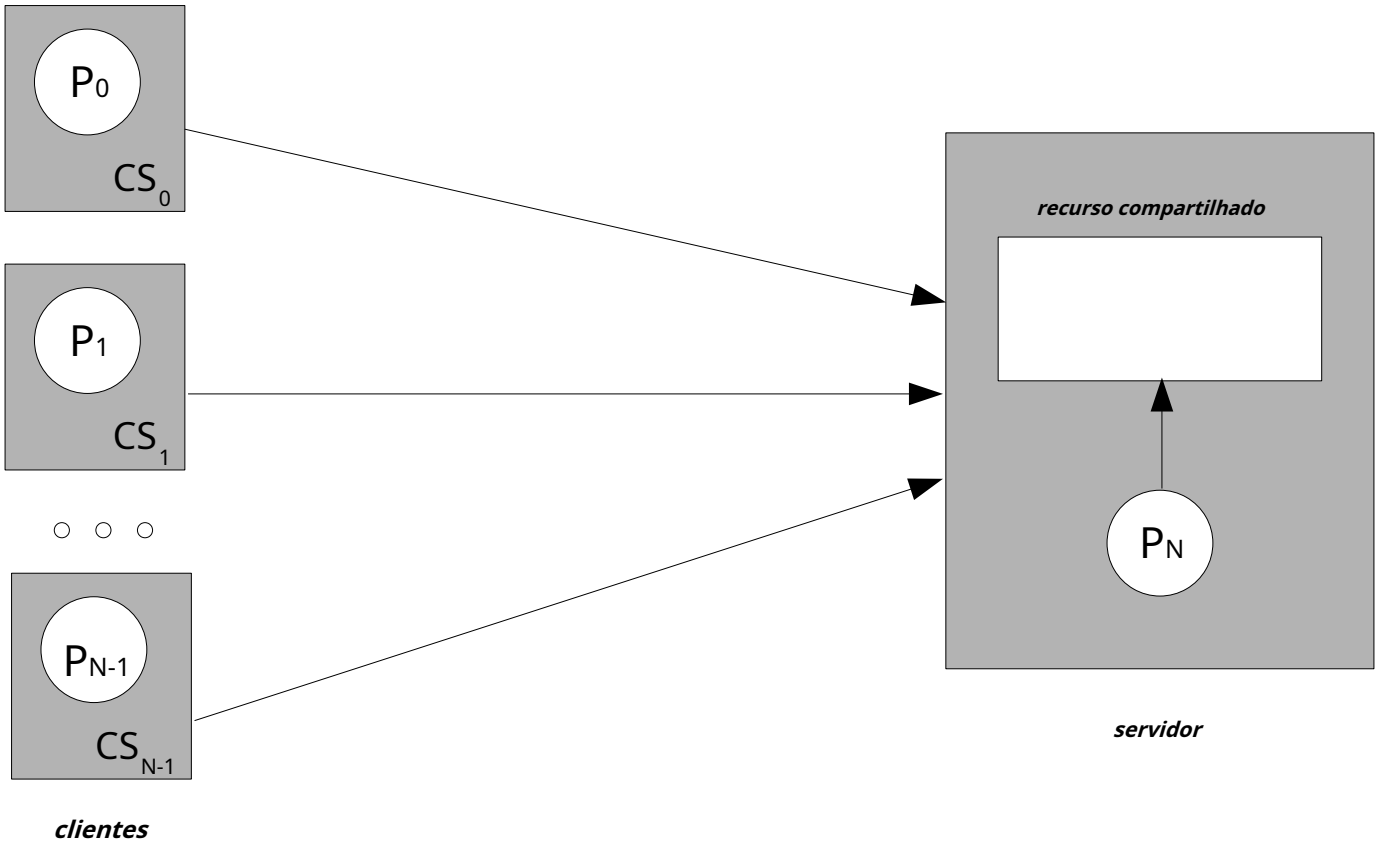
Modelo cliente-servidor – 1

Ao migrar processos e recursos compartilhados para diferentes sistemas computacionais, seus espaços de endereçamento tornam-se disjuntos e a comunicação deve ocorrer, de forma implícita ou explícita, por passagem de mensagens em um canal de comunicação comum.

Uma questão crítica a considerar é que, sendo o recurso partilhado uma entidade passiva, a gestão do acesso ao mesmo requer a criação de um novo processo, cuja função não é apenas comunicar com os outros processos, mas também executar localmente as operações chamadas no recurso partilhado. recurso.

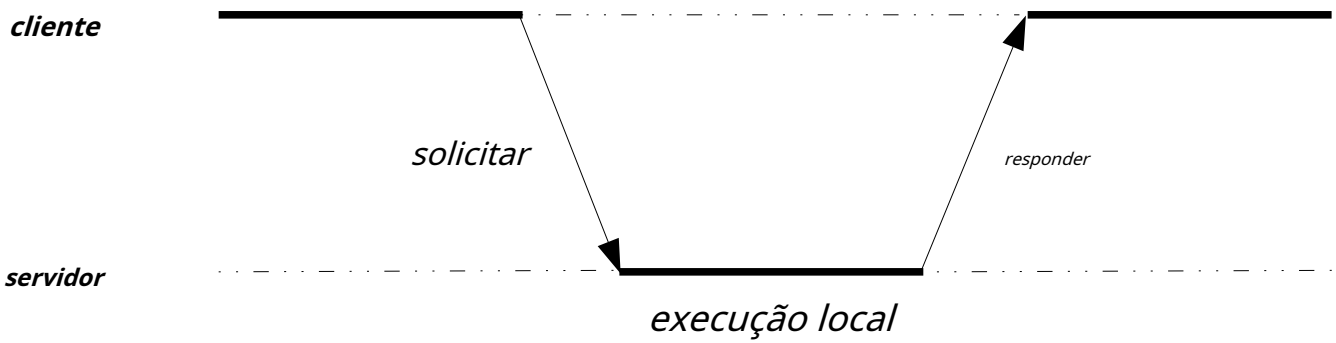
Portanto, um modelo operacional onde a manipulação de recursos possa ser entendida como *prestação de serviço*torna-se aparente. O processo que gerencia localmente o recurso compartilhado é visto como a prestação de serviço, geralmente chamada*servidor*. Os restantes processos, que pretendem aceder ao recurso, são vistos como as entidades que solicitam o serviço, denominadas*clientes*.

Modelo cliente-servidor – 2



Modelo cliente-servidor – 3

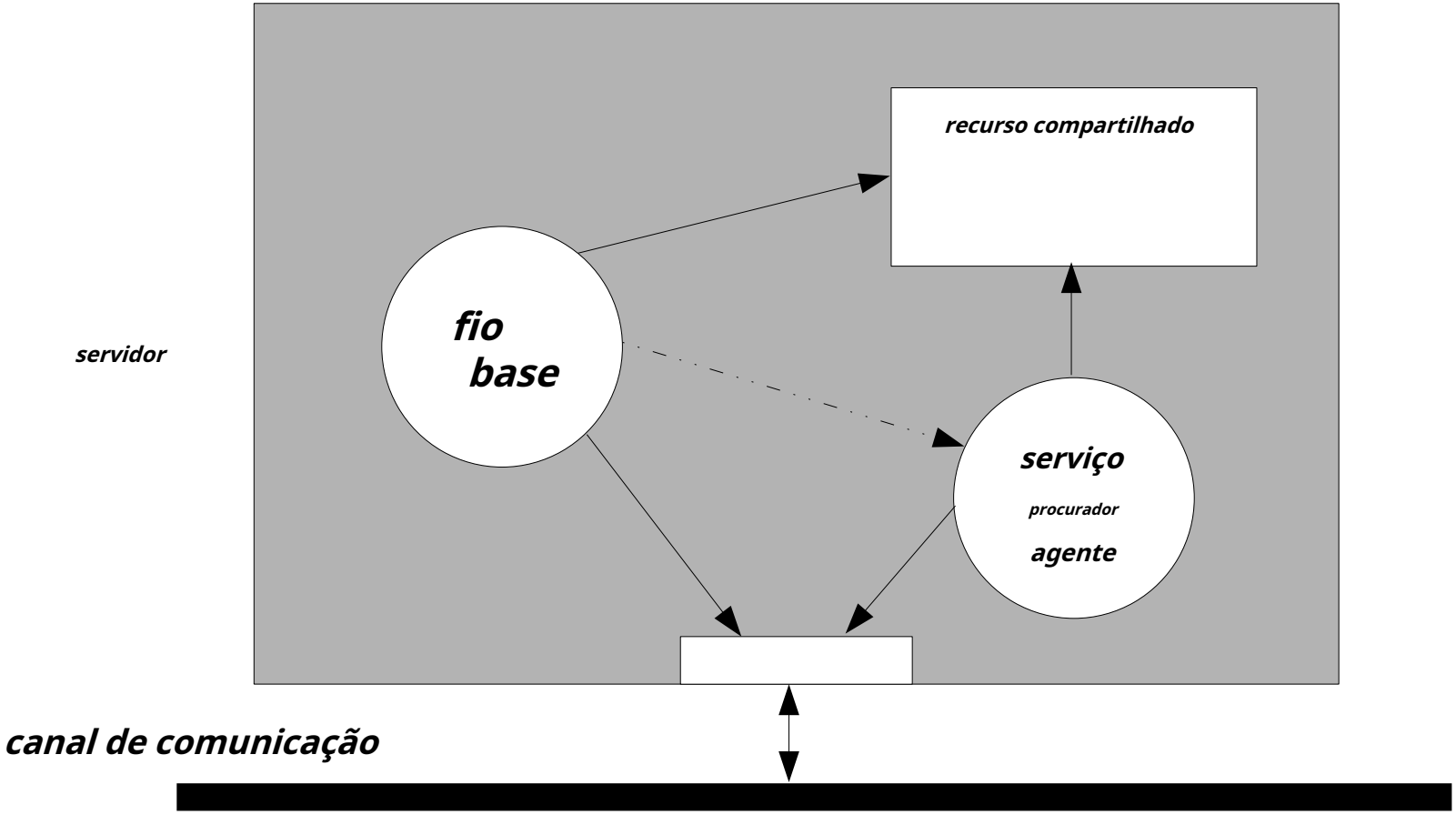
- operações chamadas no recurso compartilhado podem ser divididas em
  - *solicitar*:*cliente* processar chamadas *servidor* processo solicitando que ele execute uma operação em seu nome
  - *execução local*: execução por *servidor* processo da operação solicitada no recurso compartilhado
  - *responder*:*servidor* processar respostas para *cliente* processo comunicando o resultado da operação



Modelo cliente-servidor – 4

- o *servidor* processo tem normalmente duas funções, atuando como um
  - *gerente de comunicação*: aguarda solicitações de serviço por *cliente* processos
  - *agente proxy de serviço*: executa as operações no recurso compartilhado como um proxy do *cliente* processos
- o modelo de comunicação é assimétrico
  - *servidor-**público**/ clientes-**privado***: a operação do serviço exige que o serviço seja conhecido por todas as partes interessadas, enquanto os usuários do serviço não precisam ser previamente conhecidos pelos prestadores de serviço
  - *servidor-**eterno**/ clientes-**mortais***: a disponibilidade do serviço exige que ele esteja permanentemente operacional, enquanto seus usuários só se manifestam de tempos em tempos
  - *controle centralizado*: todas as operações de serviço estão centradas em uma única entidade

Arquitetura básica de servidor – 1

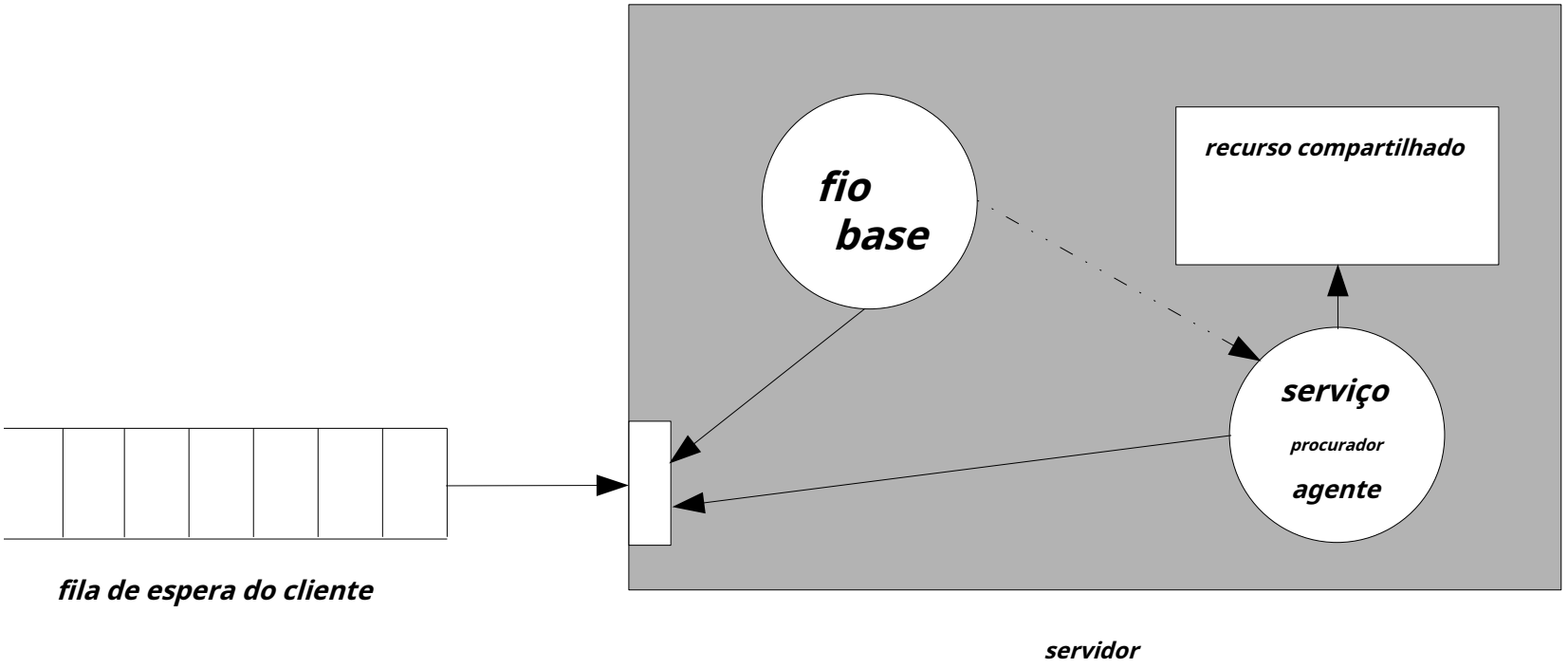


Arquitetura básica de servidor – 2

- papel do fio *base*
  - instanciar o recurso compartilhado
  - instanciar o canal de comunicação e mapeá-lo para um endereço público conhecido
  - comece a ouvir no canal de comunicação
  - quando uma conexão de um *cliente* processo é estabelecido, crie um *agente proxy de serviço* tópico para lidar com o *cliente* solicitar
- papel do fio *agente proxy de serviço*
  - determinar a operação *cliente* processo deseja ser executado no recurso compartilhado (*solicitar*)
  - executar a operação em seu nome (*execução local*)
  - comunicar o resultado da operação para *cliente* processo (*responder*)

Variantes do modelo cliente-servidor – 1

Solicitar serialização



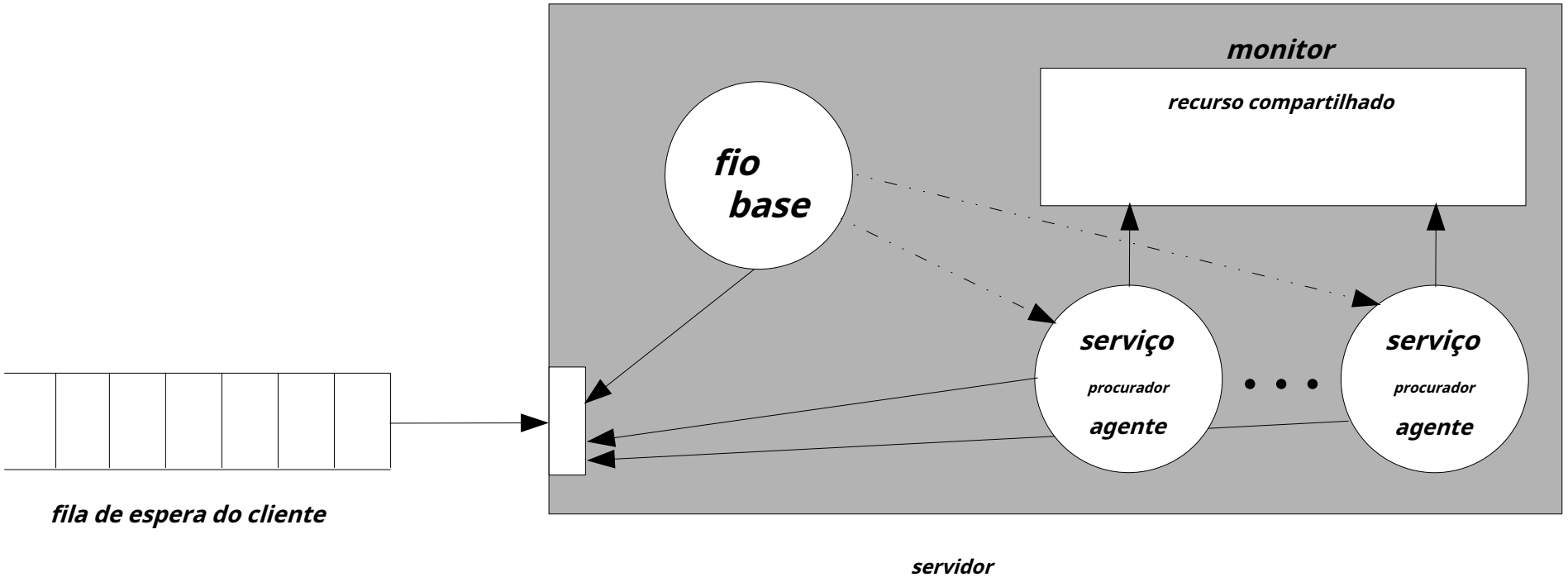
Variantes do modelo cliente-servidor – 2

Variante tipo 1 (***solicitar serialização***)

- apenas um *cliente* o processo é atendido por vez: isso significa que o thread *base*, ao receber uma solicitação de conexão, instancia um *agente proxy de serviço* e aguarda seu término antes de começar a ouvir novamente
- o recurso compartilhado não necessita de nenhuma proteção especial para garantir a exclusão mútua no acesso, uma vez que existe um único ativo *agente proxy de serviço* fio
- é um modelo muito simples, mas bastante ineficiente, uma vez que
  - o tempo de atendimento não é minimizado, pois não se aproveitam os tempos mortos de interação devido à falta de concorrência
  - dá origem a *ocupado esperando* na tentativa de sincronizar vários processos clientes no mesmo recurso compartilhado.

Variantes do modelo cliente-servidor – 3

Replicação de servidor



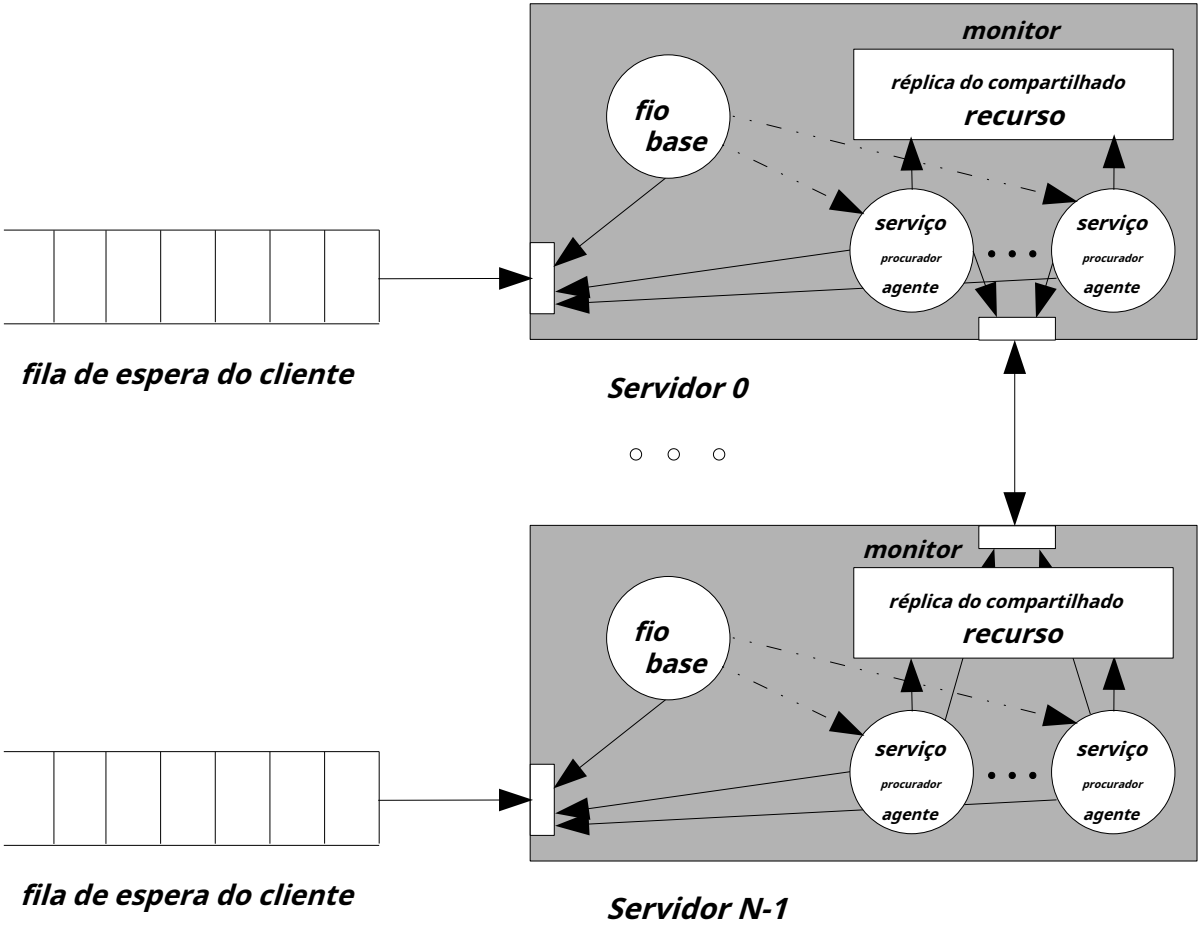
Variantes do modelo cliente-servidor – 4

Variante tipo 2 (***replicação de servidor***)

- *cliente* processos são atendidos simultaneamente: isso significa que o thread *base*, ao receber uma solicitação de conexão, instancia um *agente proxy de serviço* e começa a ouvir novamente
- o recurso compartilhado é transformado em monitor para garantir a exclusão mútua no acesso, uma vez que agora existem múltiplos ativos *agente proxy de serviço* tópicos ao mesmo tempo
- é o jeito tradicional *servidores* são configurados, na medida em que se tenta aproveitar ao máximo os recursos do sistema computacional onde o servidor está localizado
  - o tempo de serviço é minimizado, pois aproveita-se os tempos mortos da interação através da simultaneidade
  - permite a sincronização de diferentes *cliente* processos no mesmo recurso compartilhado.

Variantes do modelo cliente-servidor – 5

Replicação de recursos

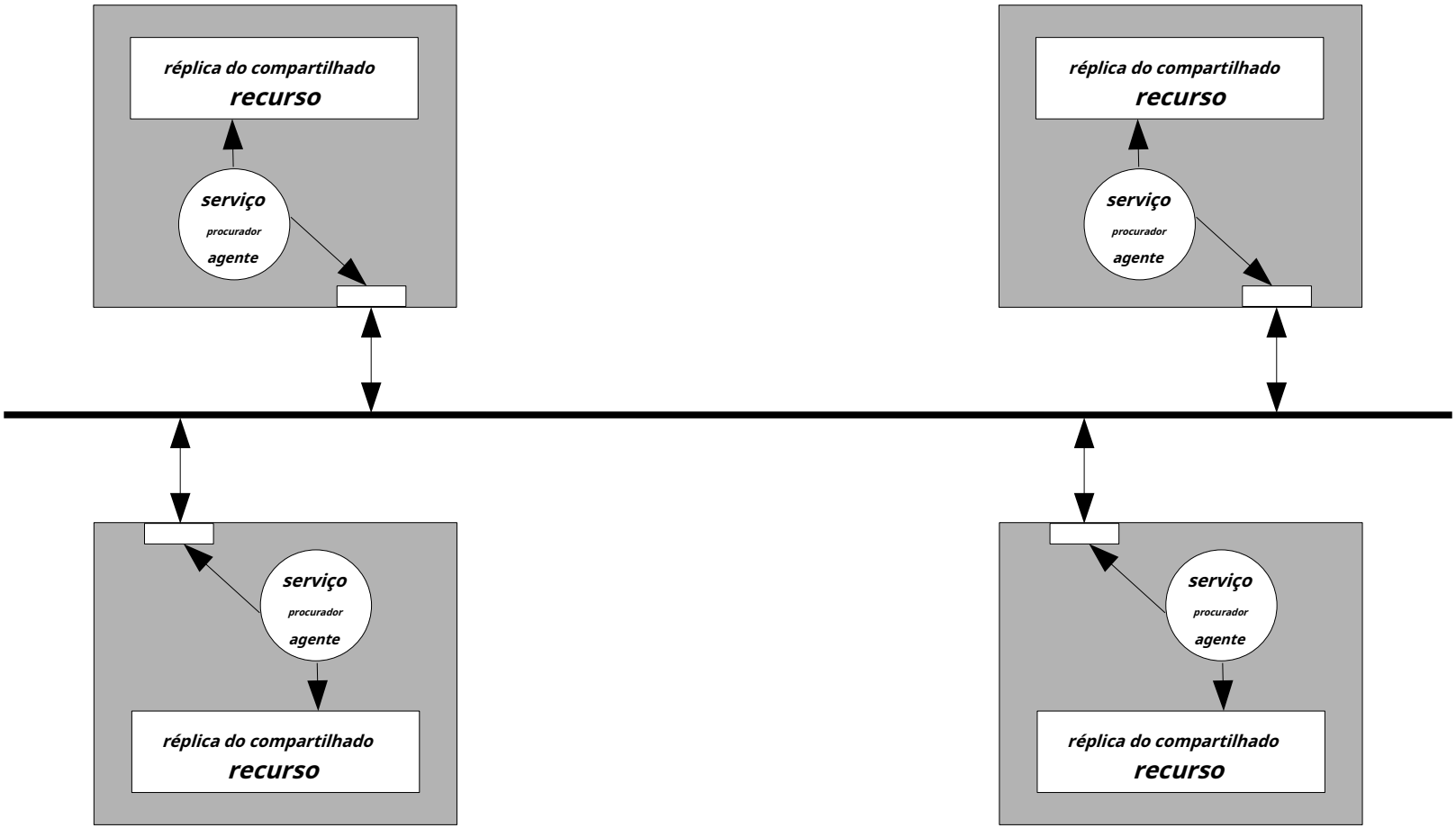


Variantes do modelo cliente-servidor – 6

Variante tipo 3 (***replicação de recursos***)

- o serviço é disponibilizado simultaneamente em vários sistemas informáticos, cada um executando uma variante tipo 2 do *servidor*
- o recurso partilhado é, assim, replicado em cada servidor, dando origem a múltiplas cópias
- é um modelo sofisticado que visa tanto maximizar a disponibilidade do serviço como minimizar o tempo de serviço, mesmo em situações de pico de carga (potencializando assim a escalabilidade)
  - o serviço é mantido operacional contra a falha de determinados *servidores*
  - *cliente* as solicitações são distribuídas entre os servidores disponíveis através de uma política, fornecida pelo serviço DNS, de associação geográfica para solicitações globais e de associação rotativa para solicitações locais
  - quando há uma alteração de dados locais em uma das réplicas do recurso compartilhado, surge a necessidade de manter consistentes as diferentes réplicas.

Comunicação entre pares – 1



Comunicação entre pares – 2

- o serviço é disponibilizado simultaneamente em vários sistemas informáticos, desempenhando cada um a mesma função; isto é, no que diz respeito ao serviço, não há diferença entre os diferentes nós de processamento, aqui chamados *pares*
- o recurso compartilhado é, assim, replicado em cada peer, dando origem a múltiplas cópias
- é um modelo sofisticado que visa tanto maximizar a disponibilidade do serviço como minimizar o tempo de serviço, mesmo em situações de pico de carga (potencializando assim a escalabilidade)
  - o serviço é mantido operacional contra a falha de determinados *pares*
  - em situações especiais, um dos pares tem de assumir um papel de liderança para que o sistema como um todo possa fazer uma transição suave de um estado estável para outro; a escolha do líder está normalmente sujeita a um processo eleitoral onde o consenso deve ser alcançado.

Primitivas de comunicação - 1

Comunicação através *passagem de mensagem* assume duas entidades: o *despachante*, que envia a mensagem; e a *destinatário*, que o recebe.

O *enviar* primitivo possui pelo menos dois parâmetros: o endereço de destino e uma referência a um buffer no espaço do usuário contendo os dados a serem enviados. Da mesma forma, o *receber* primitivo também possui pelo menos dois parâmetros: o endereço de origem e uma referência a um buffer no espaço do usuário onde os dados recebidos serão armazenados.

Normalmente, a comunicação é *tamponado* pelo sistema operacional: isto é, em uma operação de envio, os dados são primeiro transferidos para um buffer de kernel antes de serem devidamente entregues à rede; em uma operação de recebimento, os dados são armazenados primeiro em um buffer do kernel, após a recepção, e só são transferidos para o buffer do usuário quando o *receber* primitivo é chamado.

Primitivas de comunicação - 2

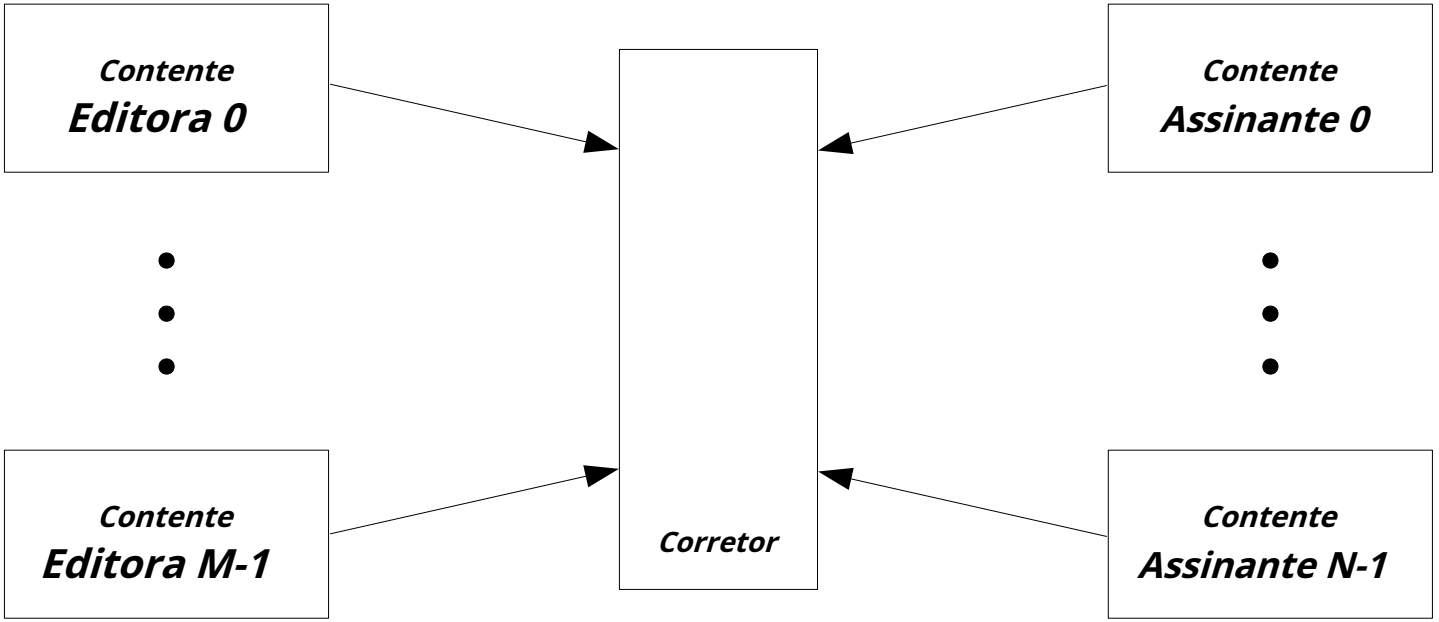
As primitivas de comunicação podem ser divididas em

- *síncrono*–se*enviar*e*receber*primitivas são acopladas entre si: a operação de envio só é concluída quando o nó de processamento de encaminhamento está ciente de que, no nó de processamento do destinatário, a operação de recebimento também foi concluída
- *assíncrono*–se*enviar*e*receber*as primitivas são totalmente desacopladas: a operação de envio é concluída assim que os dados são transferidos do buffer no espaço do usuário; não há primitivo de recebimento assíncrono.

Eles podem ser ainda divididos em

- *bloqueando*–se o controle só retornar ao processo invocador depois que a operação, seja síncrona ou assíncrona, for concluída
- *sem bloqueio*–se o controle retornar ao processo invocador imediatamente após a primitiva ser chamada, mesmo que a respectiva operação ainda não tenha sido concluída.

Modelo editor-assinante - 1

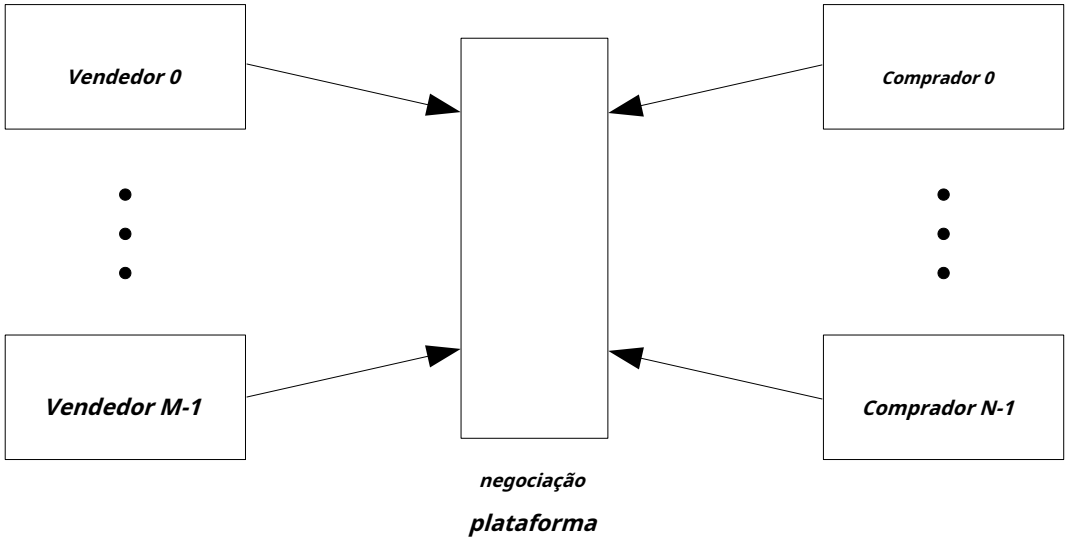


Modelo editor-assinante - 2

- existem vários prestadores de serviços, o *editores* e vários destinatários de serviços, o *assinantes*, que são totalmente dissociados entre si através da mediação de um serviço intermediário, o *corretor*
  - *editores* produzir informações de acordo com diferentes temas e atuar como clientes do *corretor* que o armazenam e o disponibilizam para *assinante* grupos que assinaram explicitamente o tópico ao qual estão associados
  - *assinantes* atuar como clientes tanto do *editores* e do *corretor*: o *editores*, pois consomem informações específicas por eles produzidas; o *corretor*, pois informam sobre os temas que lhes interessam e que devem ser alertados quando novos dados forem disponibilizados
- a principal característica é que, em contraste com o convencional *modelo cliente-servidor*, não há interação síncrona ocorrendo aqui.

Modelo editor-assinante - 3

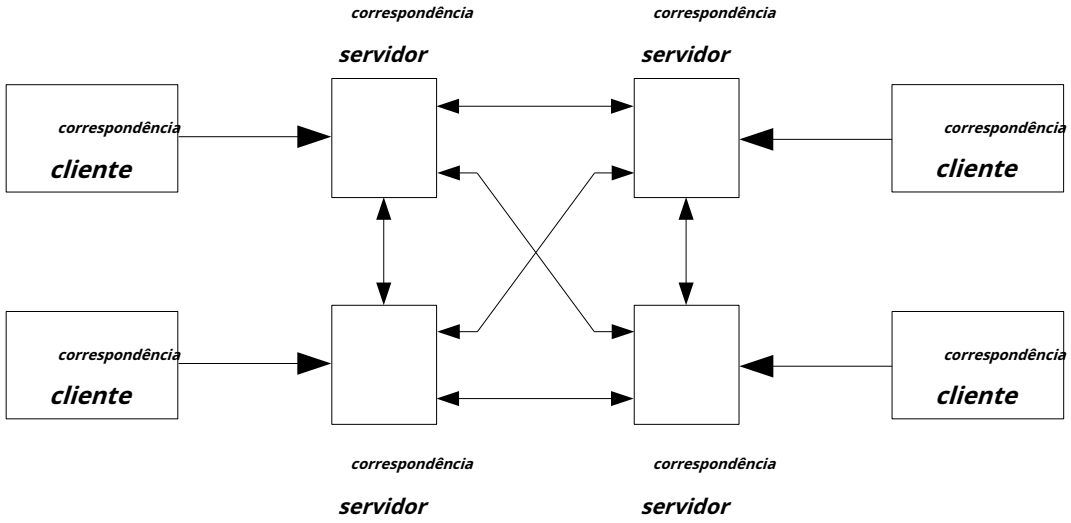
Negociação eletrônicaou**comércio eletrônico**



- o*vendedores*agir como*editores*que publicam dados sobre os produtos que desejam vender na plataforma de negociação
- o*compradores*agir como*assinantes*que solicitam à plataforma de negociação informações sobre os produtos que desejam comprar
- o*Plataforma de negociação*é o*corretor*quem gerencia possíveis transações entre eles.

Modelo editor-assinante - 4

Correio eletrônicooue-mail



- o*servidores de correio*agir como ambos*corretores*para local*clientes de e-mail*e outro*servidores de correio*, para gerenciar mensagens locais e como*editores*para outros*servidores de correio*, para encaminhar mensagens remotas
- o*clientes de e-mail*agir como ambos*editores*para enviar mensagens e como*assinantes*para receber mensagens.

Modelos fundamentais

Num sistema distribuído, onde os processos cooperam e comunicam através de uma rede, o desempenho está diretamente relacionado com a eficácia da troca de mensagens. Quando um serviço remoto faz parte dele, a velocidade com que a troca ocorre é determinada não apenas pela carga e pelo desempenho do servidor, mas também pelas capacidades de roteamento e transferência da rede de interconexão e pelos atrasos em todo o software. componentes envolvidos. Para alcançar tempos de resposta interactivos curtos, os sistemas devem ser compostos por relativamente poucas camadas de software e a quantidade de dados transferidos por interacção deve ser pequena.

Mas uma resposta rápida não é tudo o que importa para um bom*qualidade de serviço*a ser alcançado. *Confiabilidade*,*segurança*e*adaptabilidade*atender às mudanças nas configurações do sistema e na disponibilidade de recursos também são considerados importantes. Em aplicações que lidam com *tempo crítico*dados, a disponibilidade dos recursos de computação e de rede necessários nos momentos apropriados é fundamental.

Desempenho de um canal de comunicação

Os canais de comunicação podem ser implementados por fluxos de dados que conectam os pontos finais de comunicação ou por simples passagem de mensagens pela rede de computadores.

As seguintes propriedades são relevantes

- *latência*–o atraso entre o momento em que o processo remetente começa a transmitir a mensagem e o processo receptor começa a recebê-la; depende do tempo que os serviços do sistema operacional em ambas as extremidades levam para processar os dados, do atraso para acessar a rede e da sobrecarga de roteamento
- *largura de banda*–a quantidade total de informações que podem ser transmitidas pela rede de computadores em um determinado momento
- *nervosismo*–a variação de tempo necessária para entregar uma sequência de mensagens semelhantes entre os dois terminais.

Variantes do modelo de interação

Em um sistema distribuído, é muito difícil definir limites precisos no tempo necessário para a execução do processo, entrega de mensagens e desvio do relógio local.

Embora situações extremas opostas sejam

- *sistemas distribuídos síncronos*
  - o tempo para executar cada etapa do processo tem limites inferiores e superiores conhecidos
  - cada mensagem transmitida através de um canal de comunicação é recebida dentro de um limite superior conhecido
  - cada processo tem um relógio local cuja taxa de desvio de *tempo real* tem um limite superior conhecido
- *sistemas distribuídos assíncronos*
  - o tempo para executar cada etapa do processo tem um limite superior arbitrário, mas finito
  - cada mensagem transmitida através de um canal de comunicação é recebida dentro de um limite superior arbitrário, mas finito
  - cada processo tem um relógio local cuja taxa de desvio de *tempo real* é arbitrário.

Modelo de falha

Num sistema distribuído, tanto os processos intervenientes como os canais de comunicação podem *falhar*. Isso significa que seu comportamento pode se afastar do padrão definido como desejável ou correto.

As falhas são geralmente classificadas como

- *falhas de omissão*–quando as ações prescritas para serem realizadas simplesmente não ocorrem
- *falhas de tempo*–quando as ações previstas para serem realizadas não obedecerem aos prazos previamente estabelecidos
- *falhas arbitrárias ou bisantinas*–quando um erro inesperado pode ocorrer temporária ou permanentemente como resultado de um mau funcionamento de qualquer componente do sistema.

Classificação de falha

Falha Aula	***Afeta***	***Descrição***
falha	processo	um processo para e permanece parado (presume-se que os processos funcionam corretamente ou param; outros processos podem detectar seu estado)
colidir	processo	um processo para e permanece parado (outros processos não conseguem detectar seu estado)
omissão	canal	uma mensagem colocada no buffer de mensagens de saída do encaminhador nunca chega ao buffer de mensagens de entrada do destinatário
enviar omissão	processo	um processo conclui uma operação de envio, mas nenhuma mensagem é colocada no buffer de mensagens de saída
receber omissão	processo	uma mensagem é colocada no buffer de mensagens recebidas, mas uma operação de recebimento do destinatário não a obtém
relógio	processo	o relógio local do processo excede os limites de sua taxa de desvio do tempo real
desempenho	processo	o processo excede seus limites no tempo de execução entre duas operações
desempenho	canal	um tempo de transmissão de mensagem excede o limite declarado
arbitrário (ou bisantino)	qualquer processo ou canal	um processo/canal apresenta um comportamento errático: pode transmitir mensagens arbitrárias em momentos arbitrários, cometer omissões; ou um processo pode parar ou executar uma ação incorreta

Modelo de segurança - 1

A segurança de um sistema distribuído pode ser alcançada protegendo os processos e os canais de comunicação utilizados em suas interações e protegendo contra acesso não autorizado aos recursos que eles encapsulam.

Os recursos destinam-se a ser usados de diferentes maneiras por diferentes usuários. Eles podem manter dados privados para usuários específicos, acessíveis a classes especiais de usuários ou compartilhados por todos que lidam com o sistema. Para apoiar esse ambiente, *direitos de acesso* são definidos e especificam quais operações estão disponíveis e quem tem permissão para realizá-las de forma diferenciada no recurso.

Os utilizadores são, assim, incluídos no modelo como beneficiários de direitos de acesso aos recursos. Uma autoridade, chamada *diretor* e sendo um usuário ou um processo, está associado a cada invocação de operação e resultado de operação.

O *servidor* é responsável por verificar a identidade do responsável por cada solicitação de serviço e verificar se ele possui os direitos de acesso necessários para que a operação possa ser realizada em seu nome; caso contrário, rejeite a solicitação. O *cliente*, por outro lado, deve verificar a identidade do principal por trás do servidor para ter certeza de que a resposta vem do pretendido.

Modelo de segurança - 2

Para modelar ameaças à segurança, um *inimigo*, ou um *adversário* como às vezes é chamado, é postulado. O *inimigo* é capaz de enviar qualquer mensagem para qualquer processo e/ou ler qualquer mensagem trocada por um par de processos. O *inimigo* pode atacar um sistema distribuído a partir de uma plataforma de computador que esteja legitimamente conectada à rede ou que esteja conectada de maneira autorizada.

As ameaças podem ser divididas em

- *ameaças aos processos*–um processo projetado para lidar com solicitações recebidas pode receber uma mensagem na qual não é necessário identificar o encaminhador
- *ameaças aos canais de comunicação*–um inimigo pode copiar, alterar e/ou injetar mensagens à medida que elas trafegam pela rede de computadores, colocando assim em risco a privacidade, a integridade e a disponibilidade das informações do sistema.

Leitura sugerida

- *Sistemas Distribuídos: C uma vez n, 4ªEdi ção, Coulouris*, Dollimore, Kindberg, Addison-Wesley
  - Capítulo 2:*Modelos de sistema*
- *Sistemas Distribuídos: Princípios e Paradigmas, 2ªEdição,*Tanenbaum, van Steen, Pearson Education Inc.
  - Capítulo 2:*Arquiteturas nd Design*
    - Seções 2.1 a 2.3