MC714

Sistemas Distribuídos

2° semestre, 2014

Comunicação

Comunicação

- Comunicação entre processos: cerne de todo sistema distribuído.
- SDs: milhares a milhões de processos espalhados por uma rede com comunicação não confiável.
- Protocolos
- Modelos de comunicação: Remote procedure call - RPC, Message-Oriented Middleware – MOM; e fluxo de dados).
- Multicasting

Comunicação - fundamentos

Modelo ISO OSI

- Modelo OSI: até 7 camadas.
- Fig. 65
- Cada camada lida com um aspecto específico.
  - Problema quebrado em pedaços gerenciados independentemente.
- Cabeçalhos/trailers são adicionados em cada nível.
- Transmissão pela camada física.
- Cada camada retira e examina seu próprio cabeçalho.
- Fig. 66

Modelo ISO OSI

- Pilha de protocolos
  - Modelo de referência != protocolos utilizados
- Protocolos desenvolvidos para internet (TCP/IP) são os mais usados.

Modelo ISO OSI

- 3 camadas inferiores implementam funções básicas de rede
- Camada física
  - Transmitir bits
  - Quantos volts usar para 0 e para 1
  - Quantos bits por segundo
  - Ambas as direções simultaneamente?
  - Tamanho/forma do conector, número de pinos...
  - Padronização das interfaces elétrica, mecânica e de sinalização.

Modelo ISO OSI

- Camada de enlace
  - Agrupar bits em unidades (quadros)
  - Coloca padrão especial de bits no início e no final de cada quadro
  - Checksum
  - Número no cabeçalho para identificar sequencia
  - Remetente não localiza receptor: põe quadro na rede e o receptor o retira.

Modelo ISO OSI

- Camada de rede
  - Redes diferentes: mensagem pode fazer saltos
  - Escolher o melhor caminho
  - Roteamento: tarefa primária da camada de rede
  - Rota mais curta nem sempre a melhor
  - Mais utilizado: IP – internet protocol
  - Pacote – termo para uma mensagem na camada de rede
  - Roteamento independente, por pacote

Modelo ISO OSI

- Camada de transporte
  - Transforma rede em algo que um desenvolvedor pode usar
  - Serviço de transmissão entre remetente e receptor
  - Aplicação entrega mensagem com a expectativa que será entregue sem se perder.
  - Quebra mensagem, numera e envia.
  - Conexões de transporte confiáveis: rede orientada a conexão ou não – sequencia correta garantida ou não. Camada de transporte ordena.
  - Fornece comportamento de comunicação fim-a-fim.
  - Transmission Control Protocol – TCP – conexão
  - Universal Datagram Protocol – UDP – sem conexão

Modelo ISO OSI

- Camadas adicionais: internet agrupou camadas superiores (apresentação, sessão, aplicação) em uma única.
- Sessão: controle de diálogo – sincronização – pontos de verificação em transferências
- Apresentação: significado dos bits – definir campos da mensagem
  Aplicação: era para ser um conjunto de aplicações padronizadas (no modelo OSI).
  - Engloba qualquer aplicação que não se ajuste a uma das camadas subjacentes
- Da perspectiva do modelo OSI, middlewares são aplicações.

Protocolos de middleware

- Middleware: aplicação que reside logicamente, na maioria das vezes, na camada de aplicação, mas que contém protocolos de uso geral que justificam suas próprias camadas.
- Exemplos:
  - Protocolos de autenticação: não vinculados a uma aplicação específica.
  - Protocolos distribuídos de comprometimento (atomicidade).
  - Protocolo distribuído de bloqueio de recurso.

Protocolos de middleware

- Não substituem camadas do modelo de referência: aplicações podem, por exemplo, usar protocolos de transporte diferentes.

- Camada de middleware (protocolos) substitui camadas de sessão e apresentação na Fig. 65.

Tipos de comunicação

- Middleware: serviço intermediário na comunicação de nível de aplicação.
- Fig. 67
- Ex.: correio eletrônico
  - Comunicação é **persistente.**
  - Middleware armazena comunicação por tempo necessário
- **Comunicação transiente:**
  - Mensagem armazenada somente pelo tempo que a aplicação remetente e receptora estiverem executando.

Tipos de comunicação

- Comunicação assíncrona: remetente continua sua execução após ter apresentado a sua mensagem para transmissão.
- Comunicação síncrona: remetente bloqueado até saber que sua requisição foi aceita.
  - Sincronizar na apresentação da requisição (análoga: síncrona não bloqueante sem buffer no middleware)
  - Sincronizar na entrega da requisição (análoga: síncrona não bloqueante com buffer no middleware)
  - Sincronizar após processamento pelo servidor (análoga: síncrona bloqueante)

Chamada de Procedimento Remoto

RPC

- Permitir que programas chamem procedimentos localizados em outras máquinas.
- Processo A chama procedimento na máquina B
- A é suspenso
- Execução de procedimento ocorre em B
- Informações podem ser transportadas do chamador ao chamado, por parâmetros, e podem retornar no resultado do procedimento.
- Trocas de mensagem não são visíveis ao programador.

RPC

- Máquinas diferentes, espaços de endereçamento diferentes.
- Passar parâmetros e resultados: pode demandar conversões se máquinas não forem idênticas.
- Máquinas podem falhar.

- É possível lidar com muitos desses problemas: RPC é bastante utilizado.

RPC – operação básica

- **Primeiro:** rever chamada de procedimento convencional em C
- Count = read(fd, buf, nbytes)
  - fd inteiro que indica um arquivo
  - buf é um vetor de caracteres no qual dados são lidos
  - nbytes inteiro que informa quantos bytes ler
- Pilha fig. 68 (chamada pelo programa principal)

RPC – operação básica

- Parâmetro por valor: parâmetro é variável local com valor definido. Modificações não alteram valor original.
- Parâmetro por referência: ponteiro – i.e. endereço da variável
  - Chamada read: segundo parâmetro por referência (vetor).
  - Pilha: endereço do vetor de caracteres.
  - Modificar algo no vetor, modifica original.

RPC – operação básica

- Outro mecanismo (não usado em C): chamada por copiar/restaurar.
- Copiar valor para pilha, como na chamada por valor, e então copiá-lo de volta após a chamada, sobrescrevendo valor original.
- Mesmo efeito que passagem por referência em muitas situações. Mas não em algumas situações, como ... ?
- Qual mecanismo usar: projeto da linguagem.

RPC – operação básica

- Idéia da RPC: fazer com que uma chamada remota pareça uma local → seja transparente.
- Chamada local de *read:*
  - Rotina extraída da biblioteca pelo linker e inserida no programa objeto.
  - Interface entre o código de usuário e o sistema operacional local.

RPC – operação básica

- RPC consegue transparência de modo análogo:
  - Se *Read* é procedimento remoto, por ex. máquina servidor de arquivos.
  - Versão de read diferente: apêndice de cliente
  - Mesma sequencia de chamada da fig 68.
  - Também chamada SO local.
    - Não pede dados ao SO: empacota parâmetros em uma mensagem e requisita que seja enviada.
  - Chama send e depois receive, bloqueando até receber resposta.
  - Fig. 69.

RPC – operação básica

- Mensagem chega ao servidor: apêndice de servidor
- Equivalente ao apêndice cliente
  - Transforma requisições que vêm pela rede em chamadas de procedimentos locais.
  - Normalmente, terá chamado *receive* e bloqueado esperando mensagens.
  - Desempacota parâmetros e chama procedimento local.
- Servidor não precisa saber que chamador é remoto.
  - No exemplo do read, buffer será interno ao apêndice de servidor.

RPC – operação básica

- Apêndice retoma controle depois de read efetuado.
- Empacota resultados (i.e., o buffer) em uma mensagem e chama send.
- Apêndice de servidor, usualmente, chama *receive* novamente.
- No cliente, mensagem é copiada ao buffer que está esperando e processo cliente é desbloqueado.
- Apêndice de cliente desempacota resultado, copia ao cliente e retorna da maneira usual.
- Chamador não sabe que era remoto.

RPC – operação básica

- Ao cliente: chamada comum, não send e receive.
- Resumo:

1. Procedimento cliente chama apêndice cliente.
2. Apêndice cliente constrói uma mensagem e chama SO local.
3. SO cliente envia mensagem ao SO remoto.
4. SO remoto dá a mensagem ao apêndice servidor.
5. Apêndice servidor descompacta parâmetros e chama o servidor.

RPC – operação básica

6. Servidor faz serviço e retorna resultado para apêndice.
7. Apêndice de servidor empacota resultado em uma mensagem e chama SO local.
8. SO do servidor envia mensagem ao SO cliente.
9. SO cliente dá a mensagem ao apêndice de cliente
10. Apêndice desempacota resultado e retorna ao cliente.

- Efeito líquido: nem cliente nem servidor ficam cientes das etapas intermediárias ou da existência da rede.

RPC – parâmetros

- Apêndice de cliente:
  - Pegar parâmetros
  - Empacotar em mensagem
  - Enviar ao apêndice de servidor
  - Parâmetros por valor e parâmetros por referência

- Empacotar parâmetros em uma mensagem: montagem de parâmetros (parameter marshalling).

RPC – parâmetros por valor

- Parâmetros por valor
- Ex: procedimento remoto add(i,j).
- Apêndice de cliente coloca parâmetros em uma mensagem.
- Nome do procedimento.
- Servidor: apêndice examina e faz chamada apropriada.
- Ao final, apêndice do servidor retoma controle, empacota resultado em uma mensagem, envia ao apêndice de cliente.
- Fig. 70

RPC – parâmetros por valor

- Funciona bem para máquinas idênticas/ representações iguais.
- Little endian / big endian

- Fig. 71

- Informação sobre tipo

RPC – parâmetros por referência

- Como passar ponteiros?
- Ponteiro: significativo dentro do espaço de endereçamento do processo que o usa.
- Ex:.: Count = read(fd, buf, nbytes)
- Solução 1: proibir ponteiros
- Opção: passar o vetor e sobrepor resposta (copiar/ restaurar).

RPC – parâmetros por referência

- Otimização: saber se buffer é parâmetro de entrada ou de saída
  - Entrada: não precisa ser copiado de volta.
  - Saída: não precisa ser enviado.
- Não trata estruturas de dados arbitrárias.
  - Ponteiro para o apêndice
  - Requisição pode ser enviada de volta pela rede para cliente fornecer os dados referenciados.

RPC – parâmetros e apêndices

- Chamador e chamado precisam concordar com o formato das mensagens.
  - Seguir mesmo protocolo
- Fig 72: definir formato de mensagem

- Somente definir formato de mensagem não é suficiente
  - Informação sobre representação de estruturas de dados.
  - Ex.: Inteiros em complemento de dois, caracteres em unicode 16 bits, floats em IEEE #754, little endian.

RPC – parâmetros e apêndices

- Com codificação definida, necessário estabelecer regras para troca de mensagens.
  - TCP/IP
  - UDP + controle de erro
  - Etc.

- Com protocolo RPC definido, apêndices precisam ser implementados.
- Apêndices de um mesmo protocolo e procedimentos diferentes: interface diferente.
- IDL + compilação → apêndices + interfaces

Comunicação orientada a mensagem

Comunicação orientada a mensagem

- RPC contribui para transparência de acesso.
- Entretanto, nem sempre é adequado
  - Se receptor não está executando
  - Sincronismo de procedimento pode precisar ser substituído

- Outro mecanismo: troca de mensagens
  - Partes executando ou não (enfileiramento de mensagens)

Comunicação transiente orientada a mensagem - Sockets

- 1. Através de portas da camada de transporte
- Socket: terminal de comunicação
  - Aplicação pode escrever dados para a rede e ler dados da rede
- Primitivas para interface TCP
  - Servidores:
    - Socket: cria terminal de comunicação para o protocolo de transporte
    - Bind: associa endereço local com socket criado (IP+porta)
    - Listen: comunicação orientada a conexão; chamada não bloqueante que permite reservar buffers para um número de conexões.
    - Accept: bloqueia chamador até receber requisição; SO cria novo socket e retorna ao chamador; permite bifurcar processo.

Comunicação transiente orientada a mensagem - Sockets

- Primitivas para interface TCP
  - Clientes
    - Socket: SO pode alocar porta dinamicamente
    - Connect: especificando endereço para conexão; bloqueia até conexão ser estabelecida
    - Podem trocar informações por meio de send/receive.
    - Close fecha a conexão.

- Fig. 73

Comunicação transiente orientada a mensagem - MPI

- 2. Interface de troca de mensagens (MPI)
- Nível de abstração diferente de sockets
- Manipulação de diferentes formas de buffer e sincronização
- Redes e multicomputadores de alto desempenho: bibliotecas de comunicação proprietárias
  - Primitivas de alto nível eficientes, mas incompatíveis com outras bibliotecas
  - Problemas de interoperabilidade
- Definição de padrão para troca de mensagens: MPI – Message passing interface

Comunicação transiente orientada a mensagem - MPI

- Premissa: comunicação ocorre dentro de um grupo conhecido de processos.
- Cada grupo recebe identificador
- Cada processo recebe identificador (local no grupo)
- Par (groupID, processID) identifica fonte ou destinatário.
  - Usado no lugar do endereço de nível de transporte

Comunicação transiente orientada a mensagem - MPI

- Primitivas MPI:
  - MPI_bsend: assíncrona (copia para buffer local MPI e retorna).
  - MPI_send: pode bloquear até que receptor tenha iniciado operação de recebimento (sistema de execução MPI).
  - MPI_ssend: comunicação síncrona – bloqueia até que receptor receba mensagem.
  - MPI_sendrcv: bloqueia até receber resposta do receptor. Corresponde a uma RPC.
  - MPI_send e MPI_ssend: possuem variantes que evitam cópia de mensagens para buffers de sistema.
  - MPI_isend: remetente passa ponteiro para mensagem e continua.
  - MPI oferece primitivas para evitar sobrescrever buffer (verificar se terminou ou bloquear).

Comunicação transiente orientada a mensagem - MPI

- Primitivas MPI:
  - MPI_issend: remetente também passa somente ponteiro para sistema de execução MPI. Sistema indica que processou a mensagem e remetente continua.
  - MPI_recv: receber mensagem; bloqueia até chegar uma mensagem.
  - MPI_irecv: variante assíncrona (não bloqueante);

- Algumas vezes primitivas diferentes podem ser trocadas sem afetar correção de programa.
  - Variantes oferecem possibilidade de otimizar desempenho

Comunicação persistente orientada a mensagem

- Sistema de enfileiramento de mensagem ou middleware orientado a mensagem (MOM)
- Suporte para comunicação assíncrona persistente
- Capacidade de armazenamento de médio prazo para mensagens
  - Não exigem que remetente ou receptor estejam ativos.
- Suporte a transferências de mensagens que podem durara minutos ao invés de *ms*.

Comunicação persistente orientada a mensagem

- Idéia básica: aplicações se comunicam inserindo mensagens em filas específicas.
- Remetente → servidores → destinatário (mesmo se offline quando remetente enviou)

- Cada aplicação tem sua fila onde outras aplicações enviam mensagens
  - É possível aplicações compartilharem uma fila

Comunicação persistente orientada a mensagem

- Em geral, remetente sabe apenas que mensagem foi inserida na fila: entrega depende do receptor.
- Permite comunicação fracamente acoplada
- 4 combinações
- Fig. 74

- Mensagens podem conter qualquer dado
- Importante para middleware é que sejam adequadamente endereçadas.

Comunicação persistente orientada a mensagem

- Endereçamento: nome exclusivo no âmbito do sistema da fila destinatária.
- Tamanho de mensagem pode ser limitado ou pode ser fragmentada pelo sistema subjacente.
- Interface pode ser simples:
  - Put: anexe mensagem na fila especificada
  - Get: bloqueie até que a fila especificada esteja não vazia e retire a primeira mensagem
  - Poll: verifique uma fila especificada em busca de mensagens e retire a primeira. Nunca bloqueie
  - Notify: instale um manipulador a ser chamado quando uma mensagem for colocada em uma fila específica

Comunicação persistente orientada a mensagem

- Arquitetura geral de sistema de enfileiramento de mensagens
- Fig. 75
- Sistema de enfileiramento fornece:
  - Filas de fonte
  - Filas de destino
  - Providencia transferência entre filas
- Deve manter mapeamento de filas para localizações de rede: banco de dados de nomes de filas (análogo ao DNS).

Comunicação persistente orientada a mensagem

- Filas: gerenciadores de fila
- Interage com aplicação...
- ... ou como repassadores/roteadores.
  - Similar a roteamento em redes.
  - Fig. 76
  - Repassadores podem ser usados para multicasting.

Comunicação persistente orientada a mensagem - Brokers

- Aplicações diversas em sistemas distribuídos: formatos de mensagem variados.
- Em sistemas de enfileiramento, conversões são manipuladas por nós chamados brokers de mensagens.
- Converter mensagens que chegam para que sejam entendidas pela aplicação destino.
- Fig. 77
- Para sistema de enfileiramento, broker é uma aplicação

Comunicação persistente orientada a mensagem - Brokers

- Broker: de reformatador de mensagens a gateway de nível de aplicação, p.ex. Conversor entre aplicações diferentes de bancos de dados.
  - Nem sempre pode-se realizar conversão
- Comum broker para EAI (Enterprise Application Integration) – integração de aplicações empresariais.
- Converte mensagens e combina aplicações com base nas mensagens que são trocadas.
  - Publish/subscribe