Comunicação entre processos

- Comunicação local entre processos
- Comunicação remota entre processos

Processos cooperantes

- *Processos independentes* não podem afetar ou ser afetados pela execução de outros processos.
- *Processos cooperantes* podem afetar ou ser afetados pela execução de outros processos.
- Vantagens de cooperação entre processos:
  - Compartilhamento de informação
  - Aceleração de computação
  - Modularidade
  - Conveniência

Problema do produtor-consumidor

- Paradigma para processos cooperantes: processo produtor produz informação que é consumida por um processo consumidor.
- Área para armazenar informação pode ser:
  - *Ilimitada(unbounded-buffer):* não existe limite prático na área do buffer de comunicação.
  - Limitada(*bounded-buffer):* assume que existe um buffer de tamanho fixo.

Bounded-Buffer – Solução por memória compartilhada

- Dados compartilhados

```
#define BUFFER_SIZE 10
typedef struct {
   . . .
} item;
item buffer[BUFFER_SIZE];
int in = 0;
int out = 0;
```

Bounded-Buffer – Processo produtor

```
item nextProduced;

while (1) {
        while (((in + 1) % BUFFER_SIZE) == out)
                ; /* não faz nada */
        buffer[in] = nextProduced;
        in = (in + 1) % BUFFER_SIZE;
}
```

Bounded-Buffer – Processo consumidor

```
item nextConsumed;


while (1) {
        while (in == out)
                ; /* não faz nada */
        nextConsumed = buffer[out];
        out = (out + 1) % BUFFER_SIZE;
}
```

Comunicação entre processos (IPC)

- IPC é um mecanismo para que processos possam se comunicar e sincronizar suas ações.
- IPC disponibiliza duas operações:
  - **send**(*message*) – tamanho da mensagem fixo ou variável
  - **receive**(*message*)
- Se dois processos *P* and *Q* querem se comunicar, então eles precisam:
  - Estabelecer uma ligação de comunicação(*communication link)* entre eles
  - Trocar mensagens via send/ereceive
- Implementação de ligação de comunicação:
  - físicas (memória compartilhada)
  - lógica (propriedades lógicas)

Questões de implementação

- Como as ligações são estabelecidas ?
- Uma ligação pode estar associado a mais de dois processos ?
- Quantas ligações podem existir entre dois pares de processos comunicantes ?
- O que é a capacidade de uma ligação ?
- O tamanho da mensagem que uma ligação pode acomodar é fixo ou variável ?
- Uma ligação é unidirecional ou bidirecional ?

Comunicação direta

- Processos precisam ter nomes visíveis:
  - **send** (*P, mensagem*) – envia uma mensagem para o processo P
  - **receive**(*Q, mensagem*) – recebe uma mensagem do processo Q
- Propriedades da ligação de comunicação
  - Ligações são estabalecidas automaticamente.
  - Uma ligação é associada com exatamente um par de processos em comunicação.
  - Entre cada par de processos existe exatamente uma ligação.
  - Uma ligação pode ser unidirecional, mas é, usualmente, bidirecional.

Comunicação indireta

- Mensagens são enviadas e recebidas em caixas-postais (também conhecidas como portas).
  - Cada caixa postal tem um único id.
  - Processos somente podem se comunicar se eles compartilham a mesma caixa postal.
- Propriedades da ligação de comunicação
  - Ligação estabalecida somente se os processos compartilham a mesma caixa postal
  - Uma ligação pode estar associada com vários processos
  - Cada par de processos pode compartilhar várias ligações de comunicação
  - Ligação pode ser unidirecional ou bidirecional

Comunicação indireta

- Operações
  - Criar uma nova caixa-postal
  - Enviar e receber mensagens através da caixa-postal
  - Destruir uma caixa-postal

- Primitivas de comunicação são definidas como:

  send(*A, mensagem*) – envia uma mensagem para a caixa-postal A

  receive(*A, message*) – recebe uma mensagem da caixa-postal A

Comunicação indireta

- Compartilhamento da caixa-postal
  - P_1, P_2, e P_3 compartilham a caixa-postal A.
  - P_1, envia; P_2 e P_3 recebem.
  - Quem pega a mensagem ?
- Soluções:
  - Permitir que uma ligação seja associada com, no máximo, dois processos.
  - Permitir que somente um processo utilize, por vez, a primitiva de receive.
  - Permitir que o sistema selecione arbitrariamente o destinatário. Remetente é notificado sobre qual destinatário recebeu a mensagem.

Buffering

- Fila de mensagens associada à ligação.
- Formas de implementação da fila:

1. Capacidade 0 – 0 mensagens. Remetente precisa esperar o destinatário (rendezvous).
2. Capacidade limitada – tamanho finito de n mensagens. Remetente precisa esperar se o buffer está cheio.
3. Capacidade ilimitada – tamanho infinito. Remetente nunca espera.

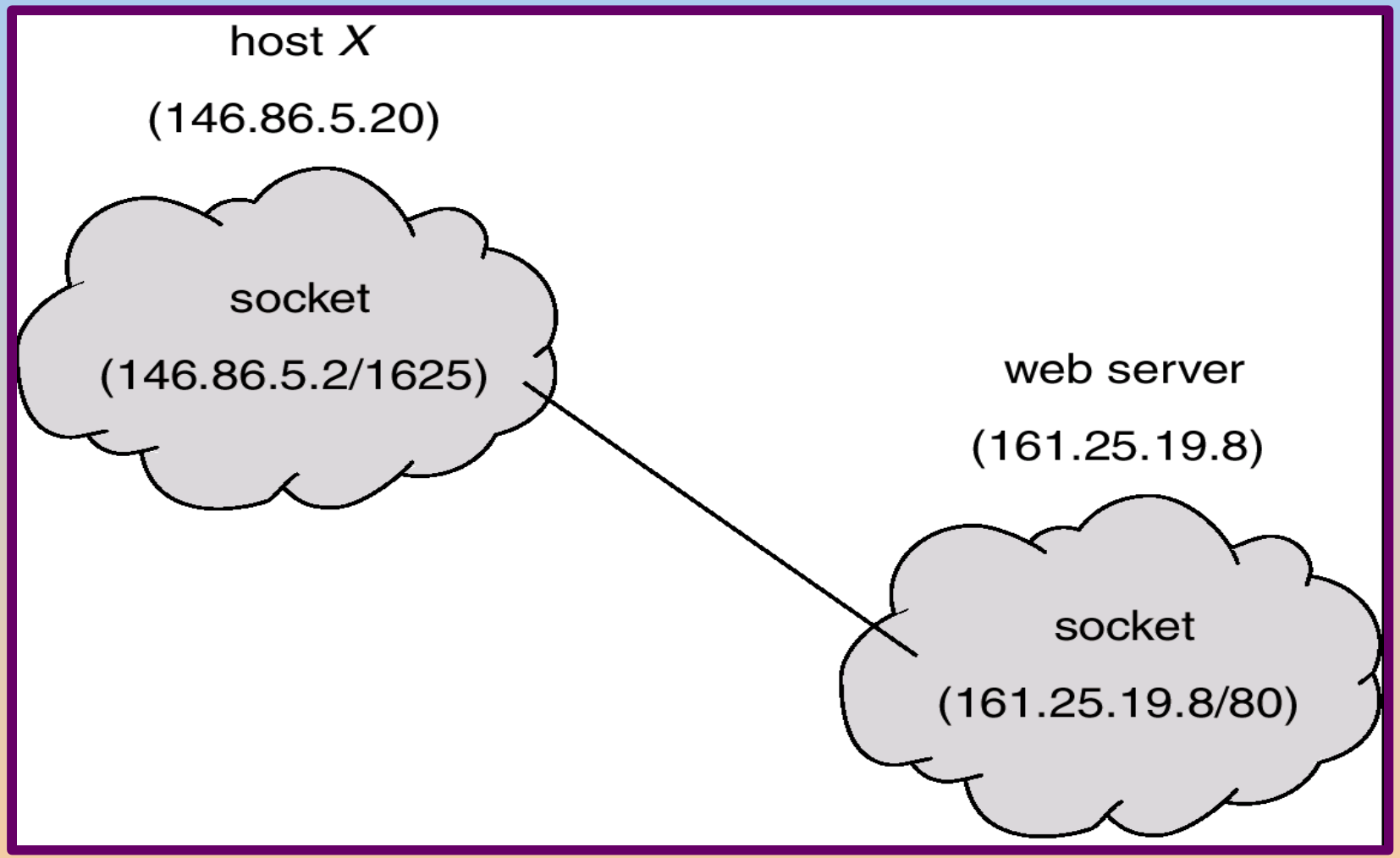
Comunicação cliente-servidor

- Soquetes (sockets)
- Chamada de procedimento remoto (RPC)
- Invocação de método remoto (Java RMI)

Sockets

- Um socket é definido com um ponto extremo(endpoint) para comunicação.
- Um socket é formado de um endereço IP e uma porta.
- O socket **161.25.19.8:1625** refere-se à porta **1625** no servidor **161.25.19.8**
- Comunicação consiste de pares de sockets.

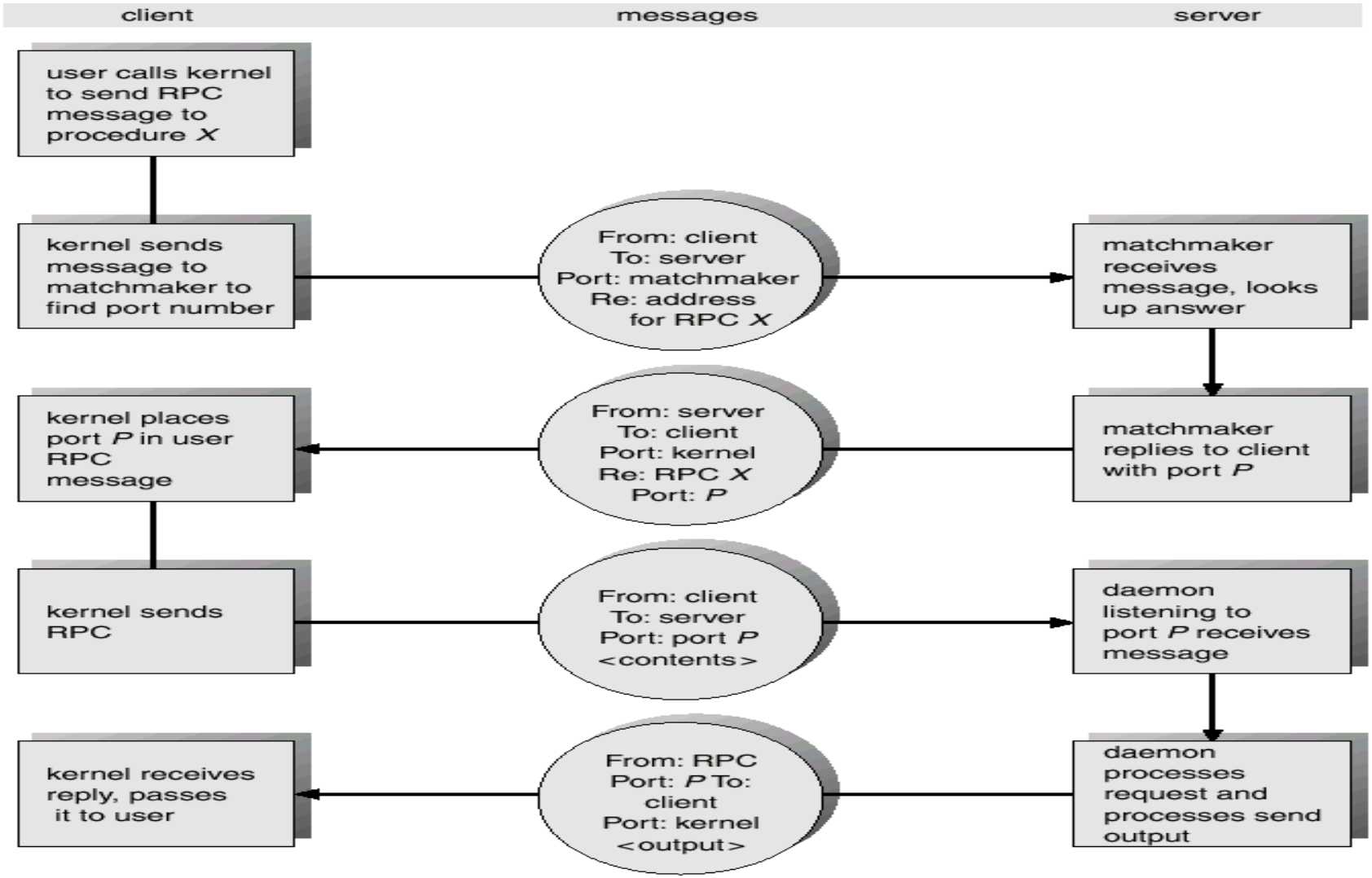
Socket Communication



Chamada de procedimento remoto

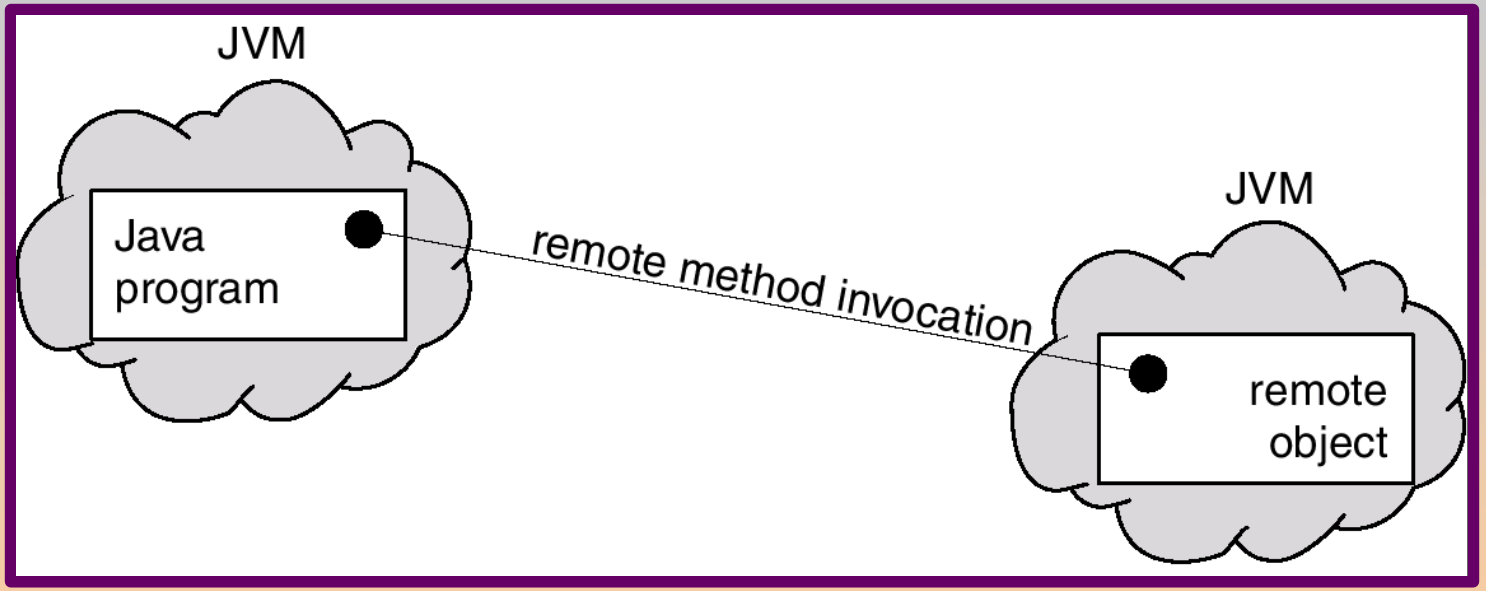
- Chamada procedimento remoto(RPC) realiza chamadas de procedimentos entre processos distribuídos sobre uma rede de computadores.
- **Stubs** – proxy do lado do cliente para os procedimentos acessáveis no servidor.
- O stub do lado do cliente localiza o servidor e ordena (marshaling) os parâmetros para envio.
- O stub do lado servidor recebe esta mensagem, desempacota os parâmetros e executa a chamada no servidor.

Execução de RPC



Invocação de método remoto

- Invocação de método remoto (RMI) é um mecanismo Java similar a RPC
- RMI permite que um programa Java em uma máquina invoque um método de um objeto remoto.



Ordenação de parâmetros