Capítulo 4: Comunicação entre Processos

- Troca de dados entre processos dentro do mesmo sistema ou em sistemas diferentes estabelecendo com diferentes sockets novos criados.Quando o socket for criado, o cliente (browser) cria um socket e se conecta ao endereço IP do servidor e ao número de porta publicado.
  - ***Comunicação Síncrona:***
    - Envio e recebimento para cada mensagem (operações “bloequeantes”)
    - Antes de continuar sua execução. Da mesma forma, quando um processo está aguardando uma mensagem, ele permanece bloqueado até que a mensagem seja recebida.
      - Ex: Transações Bancárias
  - ***Comunicação Assíncrona:***
    - Envio e recebimneto sem sincrooniização (não-bloqueante)
    - Um processo pode continuar sua execução imediatamente após enviar uma mensagem, sem esperar pelo recebimento, e pode estar pronto para receber uma mensagem a qualquer momento sem ficar bloqueado enquanto espera.
      - Ex: Weeze
- Duas operações básicas: **Send e Receive (Implementação típica envolve a adição da mensagem a uma fila (buffer) no lado do remetente)**
  - ***Comunicação com a Camada de Transporte***

    Quando uma aplicação deseja enviar uma mensagem, ela interage com a camada de transporte. A camada de transporte, por sua vez, utiliza números de porta para direcionar a mensagem ao serviço correto no processo de destino. Cada serviço em execução em uma aplicação

pode estar associado a um número de porta específico, permitindo a identificação precisa do serviço destinatário.

- Implementadas através de filas (buffers) de mensagens em cada lado da comunicação
    - ***Implementação com Filas (Buffers)***

      Cada processo que participa da comunicação tem uma fila (buffer) de mensagens. Essas filas são usadas para armazenar mensagens enviadas e recebidas.

Como fazer para que programas em diferentes PCs troquem dados de forma consistente?

- Duas abordagens:
    - Formato externo

Empacotamento e desempactoamento

- Relizados automaticamente pelas camadas do middleware, sem intervenção do usuario oui programador

PROTOCOLO BUFFER

- Protocolo de Serialização que permite que essas mensagens sejam enviadas de forma compacta antes de jogar no socket, usados para a troca e armazenamento de informações entre sistemas
- Ele converte dados em um formato binário compacto que pode ser transmitido entre sistemas e armazenado em arquivos.
- projetado para ser menor e mais rápido do que formatos como XML e JSON.
- Desenvolvida pela Google
- Útil se você quer acessar esses dados diretamente na memória sem criar cópias adicionais

- Através de um Objeto (como ‘Bytes’ ou ‘Bytesarray’) que utilizam uma interface de buffer para obter um ponteiro para os dados no buffer.Isso permite que você leia ou escreva diretamente nos dados.

FUNCIONAMENTO

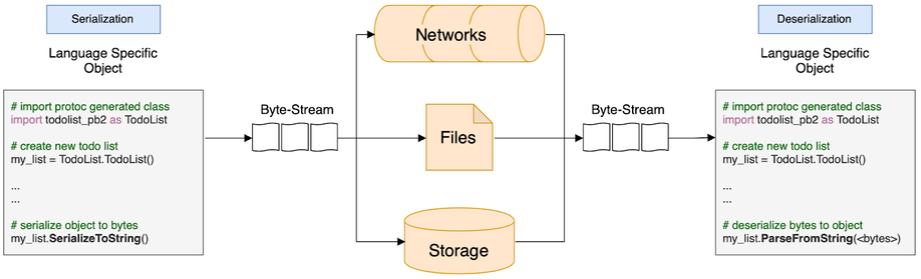

Figura 1: Essa lista de tarefas é, então, serializada e enviada pela rede, salva em um arquivo ou armazenada persistentemente em um banco de dados.O fluxo de bytes enviado é desserializado usando o método *parse*

DESVANTAGENS DE NÃO UTILIZAR

- Ocupação de espaço: mais espaço e são mais lentos para processar e transmitir.
- **Compatibilidade**: pode ser mais difícil manter compatibilidade entre versões diferentes do esquema de dados