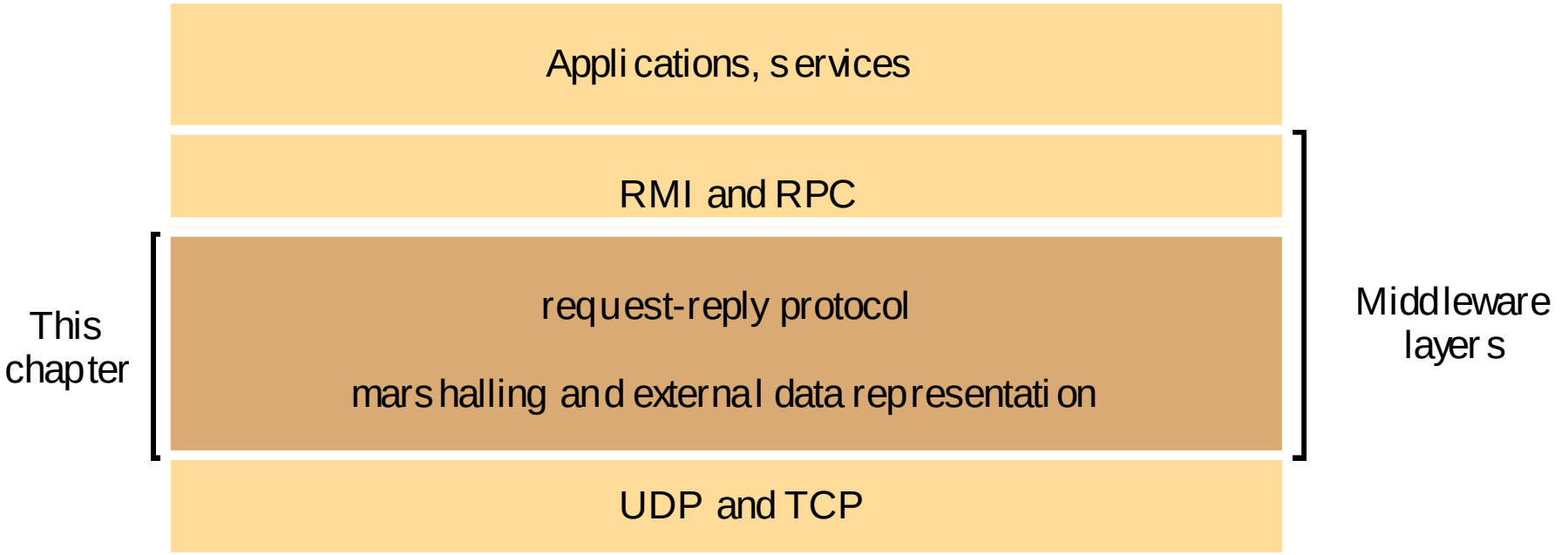
Comunicação Interprocessos Distribuídos

Capítulo 4 Sistemas Distribuídos
[Coulouris, Dolimore, Kindberg]

Prof. MSc. Rodrigo D. Malara

Nosso Escopo Hoje



- Protocolos de requisição e resposta
- Conversão e Representação de Dados (XDR)

Sistemas Distribuídos

- Sistemas Distribuídos são baseados em comunicação.
  - Geralmente é o fator que limita a performance em um SD
  - Fatores: latência, largura de banda, jitter
- Atualmente é possível se considerar que computadores são mais usados como dispositivos de comunicação do que dispositivos de computação.

Comunicação Interprocessos

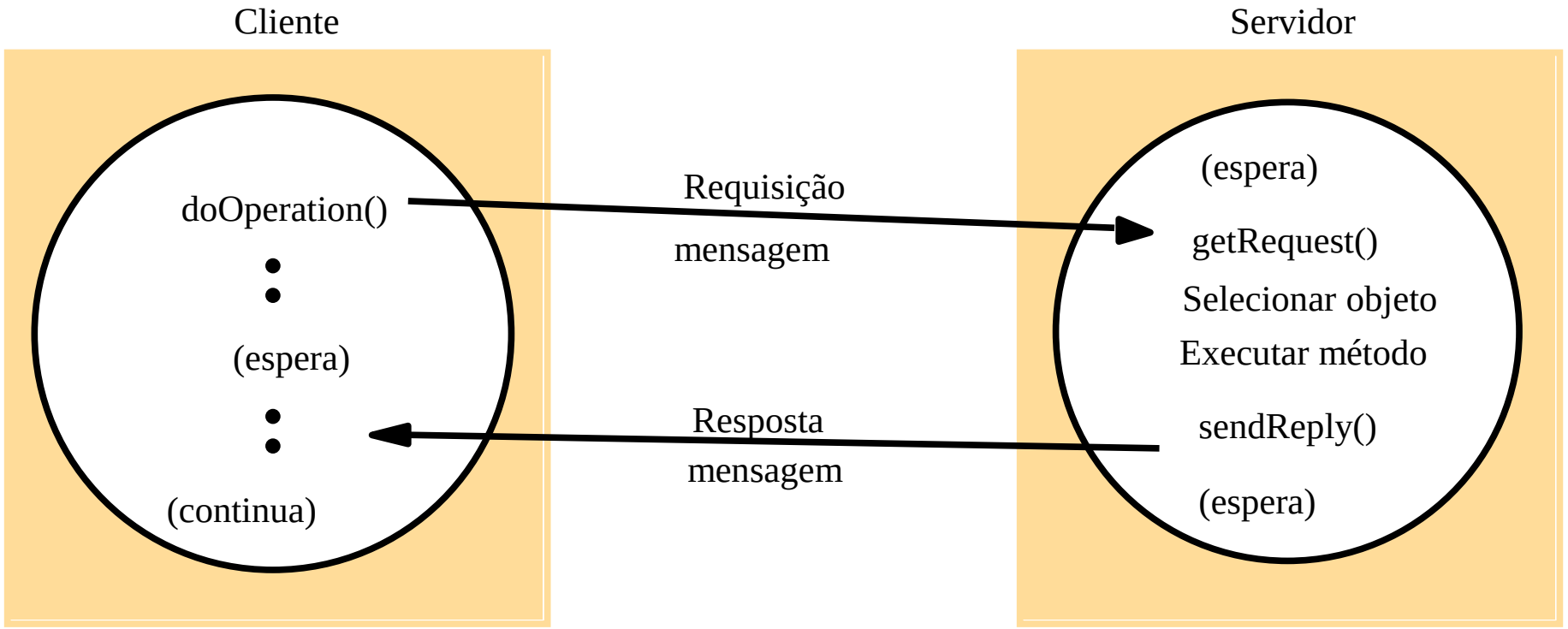
- Crítica para o sucesso de Sistemas Distribuídos
- Protocolos tendem a ser bem definidos
  - Conjunto de regras que devem ser seguidas para se padronizar uma comunicação.
- Comunicação interprocessos é baseada em troca de mensagens de baixo nível por uma rede de comunicação

CI - Conceitos Básicos

- Passagem de mensagens usando *send* e *receive*.
- Mensagens inseridas e retiradas de filas.
- Problemas:
  - Confiabilidade
    - Validade: todas as mensagens devem ser entregues mesmo que ocorram perdas de pacotes (TCP)
    - Integridade: sem corrompimento e sem duplicação (IP)
  - Ordenação
    - Pode ser qualquer ordem igual à ordem de envio (TCP)
- Não precisamos nos preocupar sobre isso!
  - Fica para pilha de protocolos do SO.
- Padrões de Comunicação:
  - Síncrona
  - Assíncrona

CI - Padrões de Comunicação

- Síncrona
  - Send é bloqueante
    - aguarda a resposta
  - Receive é bloqueante
    - não faz nada enquanto não receber a requisição (send)



CI - Padrões de Comunicação

- Assíncrona
  - Send não é bloqueante
  - Receive pode ser bloqueante ou não-bloqueante
    - Bloqueante – usamos threads
      - Cria-se um thread que vai ficar aguardando a resposta
      - Código mais simples – não tem sido muito utilizado
    - Não bloqueante é mais eficiente
      - Envolve uma complexidade maior
      - Uso de Callbacks (chamar de volta) – tem sido bastante utilizado
      - Envia um callback handler para que seja chamado pelo server junto com os dados da requisição
        - » Muito parecido com o funcionamento de interrupções no hardware

Java Sockets

Armazenando Dados

- Projeto de Arquiteturas foca em performance
  - Evolui muito rápido levando a mudanças de padrões
  - São realizadas por diferentes grupos de profissionais
- Dados são armazenados como seqüências de valores binários
- Vários tipos de dados podem coexistir:
  - Inteiros
  - Floats
  - Caracteres
- Devem obedecer à um padrão válido para quem escreve e para quem lê

Caractere	ASCII	EBCDIC
A	01000001	00011000
a	01100001	00010100

Formatos Numéricos

- Alguns computadores armazenam números de maneira diferente
- O valor 11110000 (240) pode ser armazenado de maneira que
  - 1111 é armazenado em uma localização mais baixa na memória
  - 0000 é armazenado em uma localização mais alta
  - Padrão BIG-ENDIAN – Ele inverte a posição: 00001111
    - Ganho de performance nas operações aritméticas no hardware projetado
- O mesmo inteiro pode ter significados diferentes se ele for signed (com sinal) ou unsigned (sem sinal)
- Existem vários formatos diferentes para armazenar números de ponto flutuante
- Computadores tem registradores de tamanho variado
  - 32 ou 64 bits.

Transferindo Dados

- Problemas !
  - Codificações diferentes
  - Ordens diferentes de armazenamento de dados em memória
- É necessário se definir formatos padrões a serem usados enquanto os dados estão em trânsito
- Cada arquitetura deve ter a sua própria maneira de converter os dados para a sua própria representação
- O formato intermediário é chamado ***eXternal Data Representation (XDR) – Representação eXterna de Dados***
- Notação para definir tipos e ordem de transferência de dados é chamada **Interface *Definition Language (IDL)***.

XDR - eXternal Data Representation

- Três padrões comuns de XDR:
- CORBA's Common Data Representation (CDR)
  - Pode ser usado em uma grande variedade de linguagens.
  - 15 tipos primitivos (short, long, char, boolean, …).
- Serialização de Objetos Java
  - Pode passar objetos complexos pela rede
  - Limitado ao Java apenas.
- Extensible Markup Language (SOAP/XML)
  - Pode representar dados de forma estruturada em ASCII
- JavaScript Object Notation (REST/JSON)
  - Pode representar dados de forma estruturada em ASCII

XDR - eXternal Data Representation

- Extensible Markup Language (SOAP/XML)
  - Pode representar dados de forma estruturada em ASCII

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:inv="InvoiCy">
   <soapenv:Header/>
   <soapenv:Body>
      <inv:exportardocumentos.Execute>
         <inv:Invoicyrecepcao>
            <inv:Cabecalho>
               <inv:EmpPK>PYcEsFuKroDBojfiFEl+Ms==</inv:EmpPK>
               <inv:EmpCK>39f89d5e31b5890e41383a12e3c82a1e</inv:EmpCK>
               <inv:EmpCO></inv:EmpCO>
            </inv:Cabecalho>
            <inv:Informacoes>
               <inv:Texto></inv:Texto>
            </inv:Informacoes>
            <inv:Dados>
               <inv:DadosItem>
                  <inv:Documento>&lt;ConsultaLote&gt;&lt;CnpjEmpresa&gt;99999999999999&lt;/CnpjEmpresa&gt;&lt;Prot
                  <inv:Parametros></inv:Parametros>
               </inv:DadosItem>
            </inv:Dados>
         </inv:Invoicyrecepcao>
      </inv:exportardocumentos.Execute>
   </soapenv:Body>
</soapenv:Envelope>
```

XDR - eXternal Data Representation

- JavaScript Object Notation (REST/JSON)
  - Pode representar dados de forma estruturada em ASCII

JSON for Query Request ☑ show command-line curl example

```
curl -XPOST -H "Content-Type: application/json" \
http://localhost:8094/api/index/travel-sample-index/query \
-d '{
  "explain": true,
  "fields": [
    "*"
  ],
  "highlight": {},
  "query": {
    "query": "restaurant"
  },
  "ctl": {
    "consistency": {
      "level": "at_plus"
    }
  }
}'
```

Marshalling e Unmarshalling

- Converter e desconverter
- Realizado pelo Middleware !
- A conversão de informações para uma forma transportável pela rede (XDR) seguindo especificações da IDL é chamado de ***marshalling***.
- A conversão do dado na forma XDR para o formato que a aplicação (e a arquitetura) pode entender é chamado de ***unmarshalling***.