Trabalho Prático: Implementação de Clientes para Protocolos de Comunicação

Disciplina: Sistemas Distribuídos

Objetivo: Comparação de Protocolos de Comunicação em Sistemas Distribuídos

1. VISÃO GERAL

Este trabalho prático tem como objetivo implementar clientes para três diferentes protocolos de comunicação, permitindo a análise comparativa de desempenho, eficiência e características específicas de cada abordagem em sistemas distribuídos.

1.1 Protocolos Disponíveis

Os servidores estão implementados e serão executados pelo professor em instâncias EC2 da Amazon Web Services com IPs públicos. Os alunos devem implementar os clientes para os seguintes protocolos:

1. **Protocolo Strings** (Porta 8080) - Comunicação baseada em strings estruturadas
2. **Protocolo JSON** (Porta 8081) - Comunicação baseada em mensagens JSON
3. **Protocolo Protocol Buffers** (Porta 8082) - Comunicação binária eficiente

1.2 Arquitetura do Sistema

```
Cliente (Implementação do Aluno) <---> Servidor EC2 (Fornecido pelo Professor)
- Protocolo JSON                       - IP Público AWS
- Protocolo Strings                    - Três portas diferentes
- Protocolo Protocol Buffers           - Base de dados SQLite
```

2. ESPECIFICAÇÕES TÉCNICAS

2.1 Protocolo Strings (Porta 8080)

Formato de Comunicação:

- Mensagens em texto plano
- Separadores: pipe (|) entre campos
- Codificação UTF-8
- Terminador: `\n`

Estrutura de Mensagem:

```
COMANDO|PARAM1|PARAM2|...|TIMESTAMP
```

Exemplos:

```
AUTH|202301001||2025-10-10T10:00:00
OP|echo|mensagem=Hello World|token123|2025-10-10T10:00:01
```

2.2 Protocolo JSON (Porta 8081)

Formato de Comunicação:

- Mensagens em formato JSON válido
- Codificação UTF-8
- Cada mensagem termina com `\n`

Estrutura de Requisição:

```
{
    "comando": "AUTH|OP|INFO|LOGOUT",
    "dados": {
        // parâmetros específicos do comando
    },
    "timestamp": "ISO 8601"
}
```

Estrutura de Resposta:

```
{
    "status": "OK|ERRO",
    "dados": {
        // dados de resposta
    },
    "timestamp": "ISO 8601"
}
```

2.3 Protocolo Protocol Buffers (Porta 8082)

Formato de Comunicação:

- Mensagens binárias serializadas
- Cabeçalho: 4 bytes indicando tamanho da mensagem
- Payload: dados Protocol Buffers serializados

Schema (arquivo .proto fornecido):

```
syntax = "proto3";

message Requisicao {
    oneof tipo {
        Auth auth = 1;
        Operacao operacao = 2;
        Info info = 3;
        Logout logout = 4;
    }
}

message Resposta {
    oneof resultado {
        RespostaOK ok = 1;
        RespostaErro erro = 2;
    }
}
```

3. OPERAÇÕES DO SISTEMA

3.1 Autenticação (AUTH)

Parâmetros:

- `aluno_id` : Matrícula do aluno (202301001 a 202301005)

Resposta:

- `token` : Token de sessão para operações subsequentes
- `nome` : Nome completo do aluno
- `matricula` : Número de matrícula

3.2 Operações Disponíveis (OP)

3.2.1 Echo

- **Descrição:** Retorna a mensagem enviada com informações adicionais
- **Parâmetros:** `mensagem` (string)
- **Retorno:** mensagem original, eco, timestamp, tamanho, hash MD5

3.2.2 Soma

- **Descrição:** Calcula estatísticas de uma lista de números
- **Parâmetros:** `numeros` (lista de números)
- **Retorno:** soma, média, máximo, mínimo, números processados

3.2.3 Timestamp

- **Descrição:** Retorna informações de tempo do servidor
- **Parâmetros:** nenhum
- **Retorno:** timestamp formatado, timezone, informações temporais

3.2.4 Status

- **Descrição:** Informações sobre o estado do servidor
- **Parâmetros:** `detalhado` (boolean, opcional)
- **Retorno:** status, operações processadas, estatísticas (se detalhado)

3.2.5 Histórico

- **Descrição:** Histórico de operações do aluno
- **Parâmetros:** `limite` (número, opcional)
- **Retorno:** lista de operações anteriores, estatísticas

3.3 Informações do Servidor (INFO)

Parâmetros:

- `tipo` : "basico" ou "detalhado"

Retorno:

- Informações sobre o servidor, protocolo e capacidades

3.4 Logout (LOGOUT)

Parâmetros:

- `token` : Token da sessão ativa

Retorno:

- Confirmação de logout

4. REQUISITOS DE IMPLEMENTAÇÃO

4.1 Requisitos Obrigatórios

1. **Três Clientes Funcionais**: Implementar cliente para cada protocolo
2. **Autenticação Completa**: Sistema de login/logout funcional
3. **Todas as Operações**: Implementar todas as 5 operações listadas
4. **Tratamento de Erros**: Gerenciamento adequado de erros de rede e protocolo
5. **Interface Consistente**: Interface similar entre os três clientes

4.2 Funcionalidades Mínimas

- Conexão com servidor remoto

- Autenticação com credenciais válidas
- Execução de todas as operações
- Exibição clara dos resultados
- Logout e encerramento de sessão
- Tratamento de timeouts e desconexões

4.3 Linguagens Suportadas

Os alunos podem implementar os clientes em qualquer linguagem que suporte:

- Sockets TCP/IP
- JSON parsing (para protocolo JSON)
- Protocol Buffers (para protocolo binário)

Sugestões: Python, Java, C++, JavaScript (Node.js), Go, C#

5. ENTREGÁVEIS

5.1 Código Fonte

- Três implementações de cliente (uma para cada protocolo)
- Documentação de instalação e execução
- Scripts de exemplo ou interface gráfica

5.2 Relatório Técnico

Estrutura sugerida:

1. **Introdução** - Objetivos e metodologia
2. **Implementação** - Decisões técnicas e arquitetura
3. **Testes** - Cenários testados e resultados
4. **Análise Comparativa** - Comparação entre protocolos
5. **Conclusões** - Lições aprendidas e recomendações

5.3 Demonstração

- Apresentação funcional dos três clientes
- Demonstração das operações principais

- Discussão dos resultados obtidos

6. INFORMAÇÕES COMPLEMENTARES

6.1 Credenciais de Teste

Alunos cadastrados no sistema:

- `202301001` - João Silva
- `202301002` - Maria Santos
- `202301003` - Pedro Costa
- `202301004` - Ana Oliveira
- `202301005` - Carlos Souza

6.2 Endereços dos Servidores

Importante: Os endereços IP serão fornecidos pelo professor no início do trabalho, pois as instâncias EC2 serão criadas dinamicamente.

Formato:

- Servidor JSON: `http://[IP_PUBLICO]:8080`
- Servidor Strings: `tcp://[IP_PUBLICO]:8081`
- Servidor Protocol Buffers: `tcp://[IP_PUBLICO]:8082`

6.3 Arquivos de Apoio

O professor fornecerá:

- Schema Protocol Buffers (`.proto`)
- Documentação detalhada da API
- Exemplos de mensagens para cada protocolo
- Bibliotecas ou dependências necessárias

6.4 Prazo de Entrega

Data de Entrega: [A ser definida pelo professor]
Formato: Repositório Git com código e documentação

Apresentação: [Data da apresentação a ser definida]

7. RECURSOS ADICIONAIS

7.1 Bibliografia Recomendada

- Coulouris, G. et al. "Distributed Systems: Concepts and Design"
- Tanenbaum, A. S. "Distributed Systems: Principles and Paradigms"
- Documentação oficial do Protocol Buffers

7.2 Ferramentas Sugeridas

- Wireshark (análise de tráfego de rede)
- Postman ou similar (testes de API)
- Git (controle de versão)
- IDE/Editor de código de preferência

7.3 Suporte

- Fórum da disciplina para dúvidas técnicas
- Horário de atendimento do professor
- Documentação técnica disponível no repositório da disciplina

Boa sorte com o trabalho! Este projeto proporcionará uma oportunidade de aprender sobre diferentes abordagens de comunicação em sistemas distribuídos.