

# Sistemas Multimídia

## Aula 9: Protocolos de Comunicação e Sessões de Mídia

Professora Ana Luiza Scharf

IFSC - São José  
Departamento de Telecomunicações

Semestre 2026.1

# Agenda da Aula

- 1 Protocolos de Comunicação para Multimídia
- 2 Requisitos Funcionais para o Projeto
- 3 Arquitetura Cliente-Servidor
- 4 Exemplos e Implementação
- 5 Conclusão

# O que é um Protocolo de Comunicação?

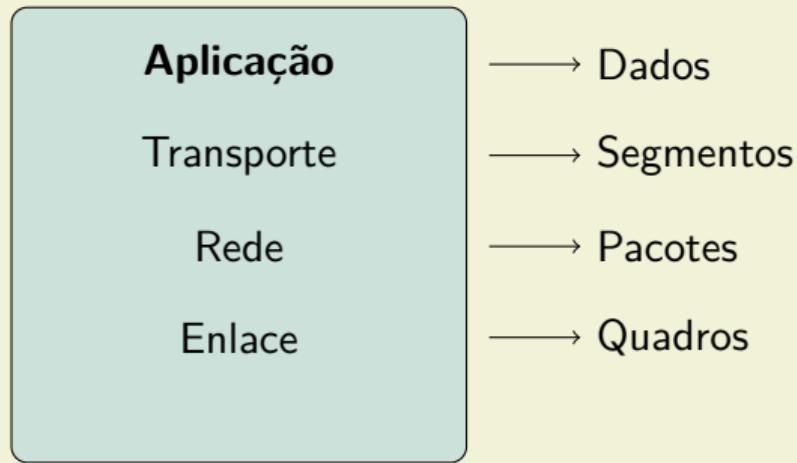
## Definição

Conjunto de regras e formatos que governam a comunicação entre dispositivos em uma rede.

## Elementos essenciais:

- Sintaxe (formato dos dados)
- Semântica (significado)
- Temporização (sincronização)

# O que é um Protocolo de Comunicação?



Camadas de protocolo e unidades de dados

# SIP (Session Initiation Protocol)

## Padrão para Comunicação Multimídia

- Protocolo de sinalização para criar, modificar e terminar sessões
- Usado em VoIP, videoconferência, mensagens instantâneas
- **Pode ser usado no projeto** (mas cuidado com complexidade)

# SIP (Session Initiation Protocol)

## Exemplo de mensagem SIP

```
INVITE sip:alice@ifsc.edu.br SIP/2.0
Via: SIP/2.0/UDP cliente.ifsc:5060
From: <sip:bob@ifsc.edu.br>
To: <sip:alice@ifsc.edu.br>
Call-ID: 12345@cliente.ifsc
CSeq: 1 INVITE
```

# Alternativas ao SIP

Protocolo	Características
<b>SIP (Recomendado)</b>	Padrão IETF, robusto, mas complexo
<b>WebRTC</b>	Moderno, navegador-based, mais simples
<b>WebSocket</b>	Bidirecional, bom para mensagens
<b>MQTT</b>	Leve, pub/sub, IoT
<b>Protocolo Customizado</b>	Simples, sob medida para o projeto

**Recomendação:** Entenda a lógica do SIP, mas implemente algo mais simples se necessário.

# Requisitos Funcionais Essenciais

## Dois principais (Entregas 3/4)

- 1 Registro e Presença:** Usuários devem se registrar e indicar disponibilidade
- 2 Estabelecimento de Sessão:** Convidar e conectar jogadores em sessão de mídia

## Requisitos Adicionais (para pensar à frente)

- Identificação única de jogadores
- Organização em salas/canais
- Descrição de mídia (codec, formato)
- NAT traversal (para Mês 3)

# Requisito 1: Registro e Presença

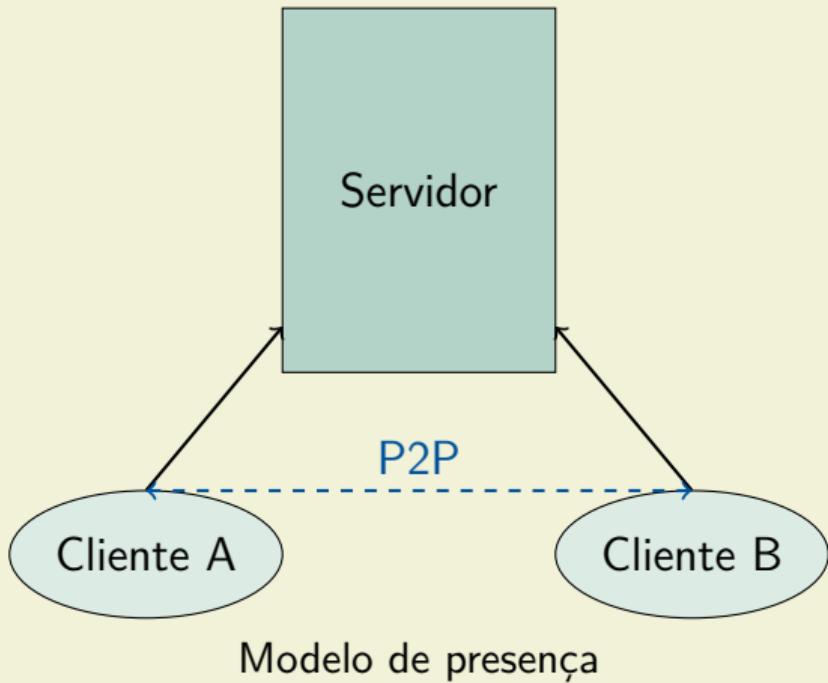
## O que deve funcionar:

- Cliente registra no servidor
- Status online/offline
- Lista de usuários disponíveis
- Atualização de status

## Exemplo de fluxo:

- ① Cliente → Servidor: REGISTER
- ② Servidor valida e armazena
- ③ Cliente periodicamente: PRESENCE UPDATE

## Requisito 1: Registro e Presença



## Requisito 2: Estabelecimento de Sessão

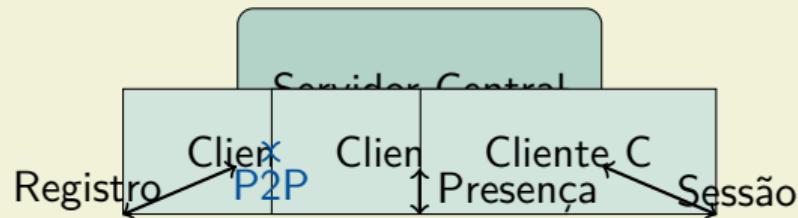
### Cenário: Dois jogadores

- Jogador A convida Jogador B
- Negociação de parâmetros de mídia
- Conexão peer-to-peer ou via servidor
- Início da transmissão de mídia

### Fluxo simplificado

- ① INVITE (de A para B, via servidor)
- ② 200 OK (B aceita)
- ③ ACK (confirmação de A)
- ④ Estabelece sessão RTC/WebRTC

# Arquitetura Básica do Sistema



Arquitetura cliente-servidor com comunicação ponto a ponto (P2P)

## Exemplo: Protocolo Customizado Simples

### Mensagem de Registro (JSON)

```
{  
  
  "type": "register",  
  "user_id": "aluno123",  
  "username": "Ana",  
  "timestamp": "2026-06-21T10:30:00Z"  
  
}
```

### Resposta do Servidor

```
{  
  
  "type": "register_response",  
  "status": "success",  
  "assigned_id": "user_789",  
  "message": "Registro completo"  
  
}
```

# Exemplo: Protocolo de Presença

**Atualização de status:** {

```
"type": "presence_update",
"user_id": "user_789",
"status": "online",
"activity": "available"
}
```

**Lista de usuários:** {

```
"type": "user_list",
"users": [
    {"id": "user_789", "name": "Ana", "status": "online"},
    {"id": "user_790", "name": "Bob", "status": "offline"}
]
```

## Exemplo: Protocolo de Presença

### Mensagem de Convite

```
{  
  "type": "invite",  
  "from": "user_789",  
  "to": "user_790",  
  "session_id": "sess_456",  
  "media_type": "video"  
}
```

# Estrutura de Código Recomendada

## Organização do Projeto

- /client – Código do cliente
- /server – Código do servidor
- /protocol – Definições do protocolo
- /docs – Documentação
- /tests – Testes

## Arquivos essenciais

- protocol.md – Especificação completa
- server.py – Servidor principal
- client.py – Cliente básico
- requirements.txt – Dependências Python

# Resumo da Aula

## Pontos Principais

- Entregas 3 e 4 estão interconectadas - planeje-as juntas
- Protocolo pode ser SIP ou customizado (recomendado mais simples)
- Foco atual: registro, presença e estabelecimento de sessão
- Arquitetura evolutiva: de 1 para múltiplos clientes

## Próxima Aula

- Dúvidas sobre especificação do protocolo
- Exemplos práticos de implementação
- Discussão sobre problemas comuns

# Perguntas?