Sistemas Multimídia

Aula 5: TCP/IP Avançado, QoS, HTTP Moderno e WebRTC

Professora Ana Luiza Scharf

IFSC - SJ
Departamento de Telecomunicações

Semestre 2026.1

Agenda da Aula

- Introdução e Contexto
- Revisão Detalhada da Pilha TCP/IP
- Qualidade de Serviço (QoS)
- Protocolos de Transporte
- Evolução do HTTP
- APIs Web Modernas para Multimídia
- Aplicações Práticas na Disciplina

Objetivos da Aula 5

- **Fechar lacunas** na pilha TCP/IP para prosseguir sem percalços
- **Revisar qualidade de serviço** (QoS) em IPv4 vs IPv6
- **Entender protocolos de transporte** (TCP, UDP, SCTP) e suas aplicações
- **Explorar evolução do HTTP** (1.1, 2, 3) e impacto no desempenho
- **Conhecer APIs web modernas** para multimídia
- **Preparar terreno** para comunicação em tempo real nos projetos

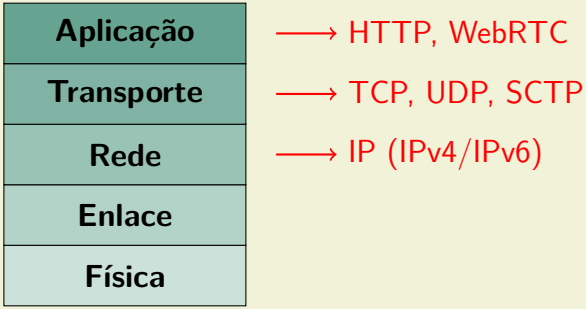
Contexto da Disciplina

Objetivo Principal

Ponte entre teoria de redes e desenvolvimento web para aplicações multimídia

- Mercado de Florianópolis: forte em desenvolvimento web
- Disciplinas tradicionais: TCP/IP, redes locais
- Necessidade: aproximar esses universos
- Projeto: jogo com comunicação em tempo real
- WebRTC + WebSockets + APIs modernas
- Aplicação prática dos conceitos de rede

Pilha TCP/IP: Foco nas 3 Camadas Superiores



Pressuposto

Camadas física e enlace estão funcionando - foco no que acontece acima

Camada de Internet (Rede)

- **Função principal**: Endereçamento e roteamento
- **Protocolo roteado**: IP (IPv4/IPv6)
- **Protocolos de roteamento**: OSPF, BGP, RIP
- **Endereçamento único** na rede
  - IPv4: 32 bits (ex: 192.168.1.1)
  - IPv6: 128 bits (ex: 2001:db8::1)

Tabela de Roteamento

Rede	Próximo Salto
192.168.1.0/24	Local
10.0.0.0/8	192.168.1.1
0.0.0.0/0	192.168.1.254

Endereçamento IP: Redes Locais

Faixas privadas (RFC 1918)

- **Classe A**: 10.0.0.0 - 10.255.255.255 (10.0.0.0/8)
- **Classe B**: 172.16.0.0 - 172.31.255.255 (172.16.0.0/12)
- **Classe C**: 192.168.0.0 - 192.168.255.255 (192.168.0.0/16)

IPv6 (RFC 4193)

- **ULA**: fd00::/8 (Unique Local Addresses)
- Exemplo: fd12:3456:789a::/48

Importante

NAT (Network Address Translation) traduz entre IPs privados e públicos

Por que QoS é Importante?

Problema

- Rede compartilhada
- Recursos limitados
- Aplicações com requisitos diferentes

Exemplos

- **Videoconferência**: baixa latência
- **Download**: alta vazão
- **VoIP**: jitter baixo
- **Email**: pode esperar

QoS no IPv4: Evolução Histórica

ToS (Type of Service) - RFC 791

- Campo de 8 bits no cabeçalho IP
- Precedência (3 bits) + Delay/Throughput/Reliability (3x1 bit)
- Pouco utilizado na prática

DSCP (Differentiated Services Code Point) - RFC 2474

- Evolução do ToS
- 6 bits para classificação
- Exemplos:
  - EF (Expedited Forwarding): 101110
  - AF41 (Assured Forwarding): 100010
  - CS0 (Class Selector 0): 000000

QoS no IPv6: Nativamente Projetada

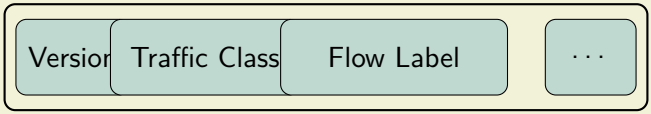
Traffic Class (8 bits)

- Similar ao DSCP do IPv4
- Padronizado desde o início
- Mais amplamente implementado

Flow Label (20 bits)

- **Novidade do IPv6**
- Identifica um fluxo de pacotes
- Permite tratamento especializado

QoS no IPv6: Nativamente Projetada



Cabeçalho IPv6 – campos relevantes para QoS

Arquiteturas de QoS

IntServ (Integrated Services)

- **RSVP** (Resource Reservation Protocol)
- Reserva recursos **antes** da transmissão
- Garantias rigorosas
- **Problema**: complexidade, escalabilidade

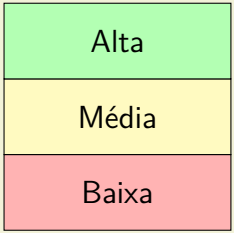
DiffServ (Differentiated Services)

- Classificação **por pacote**
- Políticas locais em cada roteador
- **Filas prioritárias** (PQ, CQ, WFQ)
- **Vantagem**: mais simples, escalável

DiffServ em Ação: Exemplo Prático

Configuração de Filas

- **Fila 1 (Alta prioridade)**: VoIP, videoconferência
  - Limite: 20% da banda
  - Se exceder: descarta pacotes
- **Fila 2 (Média)**: HTTP, HTTPS, SSH
  - Limite: 50% da banda
  - Se exceder: pode usar outras filas vazias
- **Fila 3 (Baixa)**: SMTP, FTP, P2P
  - Limite: 30% da banda
  - Se exceder: descarta agressivamente



Algoritmos de Enfileiramento

FIFO

- First In, First Out
- Simples, justo
- Sem prioridades

PQ

- Priority Queuing
- Filas com prioridades fixas
- Risco de inanição

WFQ

- Weighted Fair Queuing
- Pesos para diferentes fluxos
- Mais justo, complexo

RED (Random Early Detection)

- Descarta pacotes aleatoriamente **antes** do congestionamento
- Sinais para TCP reduzir taxa
- Evita sincronização global

TCP vs UDP: Visão Geral

TCP

- **Orientado a conexão**
- **Confiável** (ACKs, retransmissões)
- **Controle de fluxo** (janela deslizante)
- **Controle de congestionamento**
- **Ordem garantida**
- **Overhead maior**

UDP

- **Sem conexão**
- **Não confiável** (best effort)
- **Sem controle** de fluxo/congestionamento
- **Sem garantia** de ordem
- **Overhead mínimo**
- **Suporte a multicast**

SCTP: O Protocolo do Meio-termo

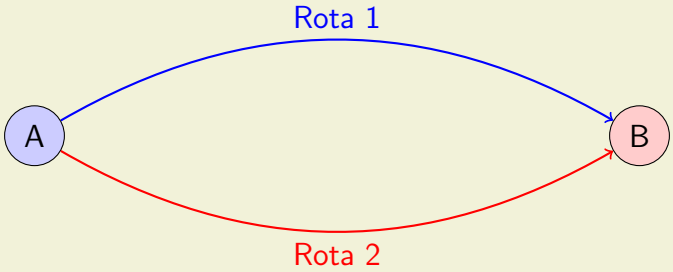
Características Principais

- **Orientado a conexão** como TCP
- **Mensagens** (não fluxo de bytes)
- **Multihoming**: múltiplos endereços IP
- **Multistreaming**: múltiplos fluxos independentes
- **Proteção contra ataques** (cookies 4-way handshake)

Aplicações Típicas

- **SIGTRAN** (sinalização telefônica sobre IP)
- **WebRTC** (data channels)
- Situações que precisam de confiabilidade **e** baixa latência

SCTP: Multihoming em Ação



Mesma conexão, múltiplos caminhos

Vantagens

- **Resiliência**: falha em um link não quebra a conexão
- **Balanceamento**: pode usar múltiplos caminhos
- **Otimização**: escolhe o melhor caminho dinamicamente

HTTP/1.0 vs HTTP/1.1

HTTP/1.0

- **1 conexão = 1 requisição**
- Sem keep-alive
- Ineficiente para páginas modernas

Problemas comuns

- **Head-of-line blocking**
- Muitas conexões paralelas
- Overhead de TCP handshake

HTTP/1.1

- **Conexões persistentes**
- **Pipelining** (teórico)
- **Chunked encoding**
- **Host header** obrigatório

Exemplo prático

Página com 100 recursos:

- HTTP/1.0: 100 conexões TCP
- HTTP/1.1: 1-6 conexões TCP

HTTP/2: Revolução no Transporte

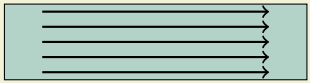
Principais Inovações

- **Multiplexação** em uma conexão
- **Streams** independentes
- **Server push**
- **Compressão de cabeçalhos** (HPACK)
- **Priorização** de streams

Limitação

Ainda sobre **TCP**: head-of-line blocking no nível transporte

1 Conexão TCP



Múltiplos Streams

HTTP/3: A Era do QUIC

QUIC (Quick UDP Internet Connections)

- Desenvolvido pelo **Google**
- **UDP** em vez de TCP
- **Criptografia** nativa (TLS 1.3)
- **0-RTT** handshake para conexões recorrentes

Vantagens

- **Sem HOL blocking**
- **Melhor com perdas**
- **Connection migration**
- **Nativo em HTTPS**

Adoção

- **Cloudflare, Google, Facebook**
- **Navegadores modernos**
- **Crescimento acelerado**

Demo: HTTP/1.1 vs HTTP/2 vs HTTP/3

Akamai HTTP/2 Demo

`https://http2.akamai.com/demo`

- **Cenário**: 300 pequenas imagens (sprites)
- **HTTP/1.1**: carrega em lotes de 6, sequencial
- **HTTP/2**: todas simultaneamente, muito mais rápido
- **HTTP/3**: ainda mais rápido em redes com perdas

Para testar em casa

Use o comando curl com as flags:

- `curl -http1.1 https://exemplo.com`
- `curl -http2 https://exemplo.com`
- `curl -http3 https://exemplo.com`

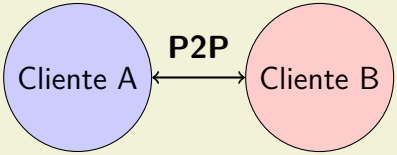
WebRTC: Comunicação em Tempo Real

Componentes Principais

- **getUserMedia**: acesso a câmera/microfone
- **RTCPeerConnection**: conexão P2P
- **RTCDataChannel**: dados arbitrários
- **ICE/STUN/TURN**: NAT traversal

Casos de Uso

- **Videoconferência** (Google Meet, Zoom Web)
- **Compartilhamento de tela**
- **Jogos multiplayer** no navegador
- **Ferramentas colaborativas**



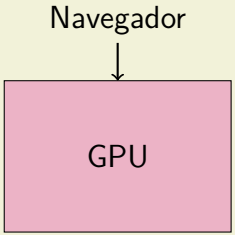
WebGPU: Computação de Alto Desempenho

Comparação com WebGL

- **WebGL**: apenas gráficos
- **WebGPU**: gráficos **e** computação geral
- Acesso direto à GPU
- Mais eficiente, mais controle

Aplicações

- **Jogos 3D** avançados
- **Simulações científicas**
- **Machine learning** no cliente
- **Processamento de vídeo**



WebAssembly: Código Nativo no Navegador

O que é?

- **Formato binário** para execução no navegador
- **Linguagens suportadas**: C, C++, Rust, Go, etc.
- **Desempenho** próximo ao nativo
- **Sandboxed** (seguro por padrão)

Vantagens

- **Performance** crítica
- **Reuso** de código existente
- **Portabilidade** entre plataformas

Exemplos Reais

- **AutoCAD Web**
- **Figma** (editor de design)
- **Photoshop Web**
- **Jogos** (Unity, Unreal)

Outras APIs Relevantes

Web Audio API

- Processamento de áudio
- Sintetizadores, efeitos
- Análise em tempo real

Web Speech API

- Reconhecimento de fala
- Síntese de voz
- Comandos por voz

Web Workers

- Threads em background
- Processamento paralelo
- Não bloqueia a UI

WebXR (Realidade Estendida)

- VR (Virtual Reality)
- AR (Augmented Reality)
- Aplicações imersivas

Cronograma

- **Semana 1-2**: Escolha e análise do código
- **Semana 3-4**: WebSockets para chat/texto
- **Semana 5-6**: WebRTC para áudio
- **Semana 7-8**: Data channels para sincronização
- **Semana 9-10**: Features adicionais (vídeo, screenshare)

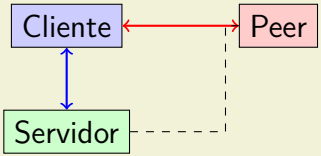
Exemplo: Jogo com Chat de Voz

Arquitetura

- **Cliente**: HTML5 + JavaScript
- **Comunicação**: WebSockets + WebRTC
- **Áudio**: Opus codec via WebRTC
- **Dados do jogo**: JSON sobre WebSockets
- **Sincronização**: WebRTC data channels

Servidores Necessários

- **HTTP/WebSocket**: lógica do jogo
- **STUN/TURN**: NAT traversal para WebRTC
- **Sinalização**: WebSocket ou HTTP



Considerações sobre QoS na Prática

Problemas Comuns

- **Wi-Fi doméstico**: interferência, congestionamento
- **NAT traversal**: STUN/TURN necessários
- **Buffer bloat**: latência variável
- **Jitter**: variação no delay

Estratégias de Mitigação

- **Adaptação de bitrate** (WebRTC)
- **FEC** (Forward Error Correction)
- **Packet loss concealment**
- **Jitter buffers** adaptativos
- **Congestion control** específico (Google Congestion Control)

Resumo da Aula

QoS

- DiffServ vs IntServ
- IPv6 nativo
- Filas prioritárias

HTTP

- 1.1 $\rightarrow$ 2 $\rightarrow$ 3
- QUIC sobre UDP
- Multiplexação

Web APIs

- WebRTC P2P
- WebGPU/WebAssembly
- Aplicações ricas

Lição Principal

Aplicações web modernas podem ter desempenho e funcionalidade de aplicações nativas, usando os conceitos de redes que estudamos.

Tarefas para Próxima Semana

1. **Escolher um jogo** do catálogo disponível
2. **Analisar o código** e entender a arquitetura
3. **Pensar em features** multimídia para adicionar
4. **Ler material** sobre WebSockets (já implementado)
5. **Explorar documentação** do WebRTC

Recursos

- **Repositório da disciplina**: códigos de exemplo
- **MDN Web Docs**: documentação oficial das APIs
- **WebRTC samples**: exemplos práticos
- **Discord da turma**: dúvidas e discussões

Obrigada!

Dúvidas?

ana.scharf@ifsc.edu.br