

# Prof. Esp. Thalles Canela

- **Graduado:** Sistemas de Informação - Wyden Facimp
- **Pós-graduado:** Segurança em redes de computadores - Wyden Facimp
- **Consultor de Tecnologia - [aXR6] Cyber Security e NtecSoftware**
- **Professor no Senac (contratado)**
- **Professor na Wyden Facimp (contratado)**
  - **Pós-graduação:** Segurança em redes de computadores - Wyden Facimp
- **Professor na Wyden Facimp (Efetivado)**
  - **Graduação:** Análise e desenvolvimento de sistemas - Wyden Facimp

## Redes sociais:

- **Linkedin:** <https://www.linkedin.com/in/thalles-canela/>
- **YouTube:** <https://www.youtube.com/aXR6CyberSecurity>
- **Facebook:** <https://www.facebook.com/axr6PenTest>
- **Instagram:** [https://www.instagram.com/thalles\\_canela](https://www.instagram.com/thalles_canela)
- **Github:** <https://github.com/ThallesCanela>
- **Github:** <https://github.com/aXR6>
- **Twitter:** <https://twitter.com/Axr6S>

# **PROTÓCOLOS DE APLICAÇÃO DA INTERNET**

# Camada de aplicação

- Princípios de aplicações de rede
- Web e HTTP
- FTP
- Correio eletrônico
  - SMTP, POP3, IMAP
- DNS

# Camada de aplicação

Nossos objetivos:

- Conceitual, aspectos de implementação de protocolos de aplicação de redes
  - Modelos de serviço da camada de transporte
  - Paradigma cliente-servidor
  - Paradigma peer-to-peer
  - Aprender sobre protocolos examinando protocolos da camada de aplicação populares: HTTP, FTP, SMTP/ POP3/ IMAP, DNS

# Algumas aplicações de rede

- E-mail
- Web
- Mensagem instantânea
- Login remoto
- P2P file sharing
- Jogos de rede multi-usuário
- Streaming stored vídeos
- Telefonia via Internet
- Videoconferência em tempo real
- Computação paralela massiva

# Criando uma nova aplicação de rede

- Escrever programas que:
  - Executem sobre diferentes sistemas finais;
- Se comuniquem através de uma rede.

Ex.: Web – software de servidor Web se comunicando com software do browser.

Nenhum software é escrito para dispositivos no núcleo da rede

- Dispositivos do núcleo da rede não trabalham na camada de aplicação
- Esta estrutura permite um rápido desenvolvimento de aplicação

# Camada de aplicação

- Princípios de aplicações de rede
- Web e HTTP
- FTP
- Correio electrónico SMTP, POP3, IMAP
- DNS

# Arquiteturas de aplicação

- Cliente-servidor
- **Peer-to-peer (P2P)**
- Híbrida de cliente-servidor e P2P



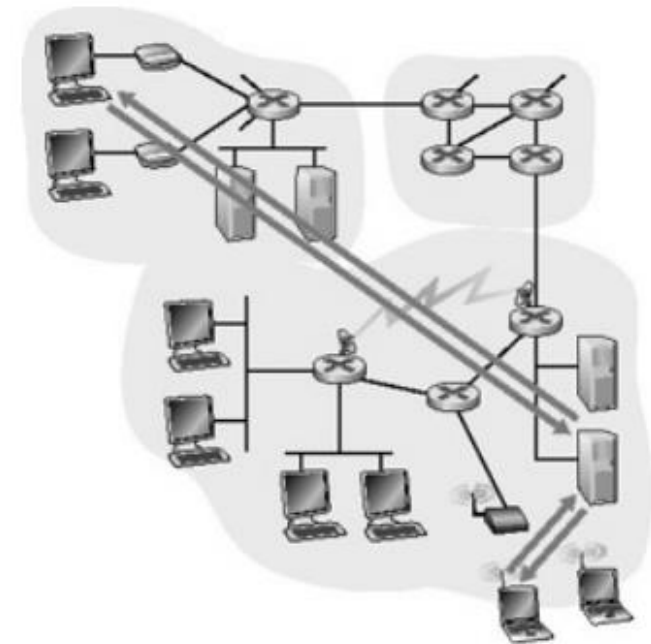
# Arquitetura cliente-servidor

- **Servidor:**

- Sempre ativo
- Endereço IP permanente
- Fornece serviços solicitados pelo cliente

- **Clientes:**

- Comunicam-se com o servidor
- Pode ser conectado intermitentemente
- Pode ter endereço IP dinâmico
- Não se comunicam diretamente uns com os outros

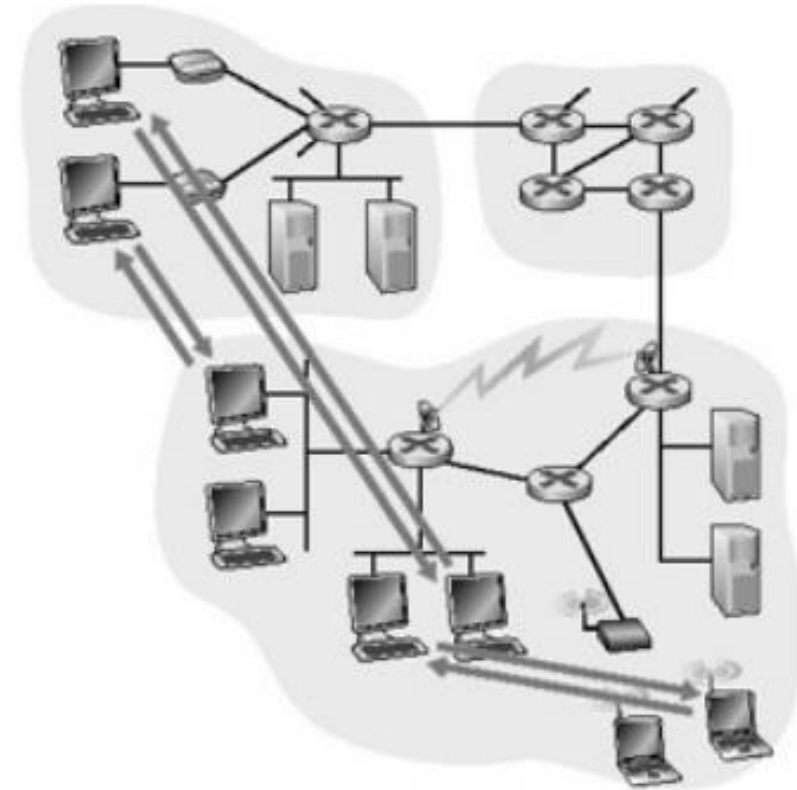


a. Aplicação cliente-servidor

# Arquitetura P2P pura

- Nem sempre no servidor
- Sistemas finais arbitrários comunicam-se diretamente
- Pares são intermitentemente conectados e trocam endereços IP
- Ex.: Gnutella

Altamente escaláveis mas difíceis de gerenciar



b. Aplicação P2P

# Híbrida de cliente-servidor e P2P

## **Napster**

- Transferência de arquivo P2P
- Busca centralizada de arquivos:
  - Conteúdo de registro dos pares no servidor central
  - Consulta de pares no mesmo servidor central para localizar o conteúdo

## **Instant messaging**

- Bate-papo entre dois usuários é P2P
- Detecção/localização centralizada de presença:
  - Usuário registra seu endereço IP com o servidor central quando fica online
  - Usuário contata o servidor central para encontrar endereços IP dos vizinhos

# Comunicação de processos

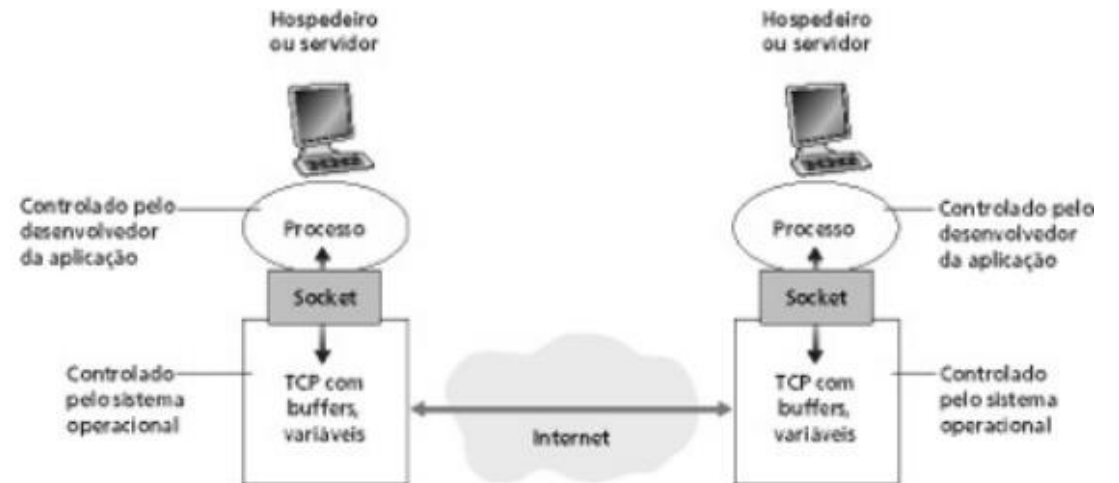
**Processo:** programa executando num hospedeiro

- Dentro do mesmo hospedeiro: dois processos se comunicam usando **comunicação interprocesso** (definido pelo OS)
- Processos em diferentes hospedeiros se comunicam por meio de troca de **mensagens**
- **Processo cliente:** processo que inicia a comunicação
- **Processo servidor:** processo que espera para ser contatado

**Nota:** aplicações com arquiteturas P2P possuem processos cliente e processos servidor

# Sockets

- Um processo envia/recebe mensagens para/de seu socket
- O socket é análogo a uma porta
  - O processo de envio empurra a mensagem para fora da porta
  - O processo de envio confia na infra-estrutura de transporte no outro lado da porta que leva a mensagem para o socket no processo de recepção.



# Processos de endereçamento

- Para um processo receber mensagens, ele deve ter um identificador
- Um hospedeiro possui um único endereço IP de 32 bits
- **P.:** O endereço IP do hospedeiro onde o processo está executando é suficiente para identificar o processo?
- **R.:** Não, muitos processos podem estar em execução no mesmo hospedeiro.
- O identificador inclui o endereço IP e o **número da porta** associada ao processo no hospedeiro
- Exemplos de números de porta:
  - Servidor HTTP: 80
  - Servidor de Correio: 25

# O protocolo de aplicação define

- Tipo das mensagens trocadas, mensagens de requisição e resposta
- Sintaxe dos tipos de mensagem: os campos nas mensagens e como são delineados
- Semântica dos campos, ou seja, significado da informação nos campos
- Regras para quando e como os processos enviam e respondem às mensagens

## Protocolos de domínio público:

- Definidos nas RFCs
- Recomendados para interoperabilidade
- Ex.: HTTP, SMTP

## Protocolos proprietários:

- Ex.: KaZaA

# De qual serviço de transporte uma aplicação necessita?

## Perda de dados

- Algumas aplicações (ex.: áudio) podem tolerar alguma perda
- Outras aplicações (ex.: transferência de arquivos, telnet) exigem transferência de dados 100% confiável

## Temporização

- Algumas aplicações (ex.: telefonia Internet, jogos interativos) exigem baixos atrasos para serem “efetivos”

## Banda passante

- Algumas aplicações (ex.: multimídia) exigem uma banda mínima para serem “efetivas”
- Outras aplicações (“aplicações elásticas”) melhoram quando a banda disponível aumenta



# Requisitos de transporte de aplicação comuns

Aplicação	Perdas	Banda	Sensível ao atraso
file transfer	sem perdas	elástica	não
e-mail	sem perdas	elástica	não
Web documents	tolerante	elástica	não
real-time áudio/vídeo	tolerante	áudio:5Kb-1 Mb vídeo:10Kb-5 Mb	sim, 100's mseg
stored áudio/video	tolerante	igual à anterior	sim, segundos
jogos interativos	tolerante	kbps	sim, 100's mseg
e-business	sem perda	elástica	sim

# Serviços dos protocolos de transporte da Internet

## Serviço TCP:

- **Orientado à conexão:** conexão requerida entre processos cliente e servidor
- **Transporte confiável** entre os processador de envio e recepção
- **Controle de fluxo:** o transmissor não sobrecarrega o receptor
- **Controle de congestionamento:** protege a rede do excesso de tráfego
  - **Não oferece:** garantias de temporização e de banda mínima

## Serviço UDP:

- Transferência de dados não confiável entre os processos transmissor e receptor
- Não oferece: estabelecimento de conexão, confiabilidade, controle de fluxo e de congestionamento, garantia de temporização e de banda mínima.

**P.:** Por que ambos? Por que existe o UDP?

# Aplicação e protocolos de transporte da Internet

Aplicação	Protocolo de aplicação	Protocolo de transporte
e-mail	smtp [RFC 821]	TCP
acesso de terminais remotos	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
transferência de arquivos	ftp [RFC 959]	TCP
streaming multimídia	RTP ou proprietário (ex.: RealNetworks)	TCP ou UDP
servidor de arquivos remoto	NSF	TCP ou UDP
telefonia Internet	RTP ou proprietário (ex.: Vocaltec)	tipicamente UDP

# Camada de aplicação

- 2.1 Princípios de aplicações de rede
- **2.2 Web e HTTP**
- 2.3 FTP
- 2.4 Correio eletrônico
  - SMTP, POP3, IMAP
- 2.5 DNS

# Web e HTTP

## Primeiro alguns jargões

- **Página Web** consiste de **objetos**
- Objeto pode ser arquivo HTML, imagem JPEG, Java applet, arquivo de áudio,...
- A página Web consiste de **arquivo-HTML base** que inclui vários objetos referenciados
- Cada objeto é endereçado por uma **URL**
- Exemplo de URL:

# Visão geral do HTTP

## HTTP: hypertext transfer protocol

- Protocolo da camada de aplicação da Web
- Modelo cliente/servidor
  - **Cliente:** browser que solicita, recebe e apresenta objetos da Web
  - **Servidor:** envia objetos em resposta a pedidos
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



# Visão geral do HTTP

## Utiliza TCP:

- Cliente inicia conexão TCP (cria socket) para o servidor na porta 80
- Servidor aceita uma conexão TCP do cliente
- mensagens HTTP (mensagens do protocolo de camada de aplicação) são trocadas entre o browser (cliente HTTP) e o servidor Web (servidor HTTP)
- A conexão TCP é fechada

## HTTP é “stateless”

- O servidor não mantém informação sobre os pedidos passados pelos clientes

## Protocolos que mantêm informações de “estado” são complexos!

- Histórico do passado (estado) deve ser mantido
- Se o servidor/cliente quebra, suas visões de “estado” podem ser inconsistentes, devendo ser reconciliadas

# Conexões HTTP

## HTTP não persistente

- No máximo, um objeto é enviado sobre uma conexão TCP
- O HTTP/1.0 utiliza HTTP não persistente

## HTTP persistente

- Múltiplos objetos podem ser enviados sobre uma conexão
- TCP entre o cliente e o servidor
- O HTTP/1.1 utiliza conexões persistentes em seu modo padrão



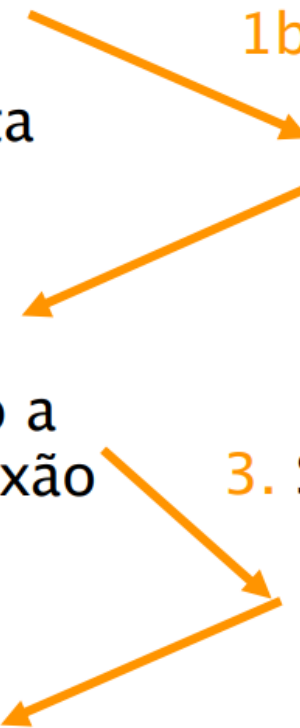
# HTTP não persistente

1a. Cliente HTTP inicia conexão TCP ao servidor HTTP (processo) em `www.someSchool.edu`. Porta 80 é a default para o servidor HTTP.


1b. Servidor HTTP no hospedeiro `www.someSchool.edu` esperando pela conexão TCP na porta 80. “Aceita” conexão, notificando o cliente

2. Cliente HTTP envia HTTP **request message** (contendo a URL) para o socket da conexão TCP

3. Servidor HTTP recebe mensagem de pedido, forma **response message** contendo o objeto solicitado (`someDepartment/home.index`), envia mensagem para o socket



# HTTP não persistente

4. Servidor HTTP fecha conexão TCP.
  5. Cliente HTTP recebe mensagem de resposta contendo o arquivo html, apresenta o conteúdo html. Analisando o arquivo html, encontra 10 objetos jpeg referenciados
  6. Passos 1–5 são repetidos para cada um dos 10 objetos jpeg.
- 

# Modelagem do tempo de resposta

**Definição de RRT:** tempo para enviar um pequeno pacote que vai do cliente para o servidor e retorna.

## **Tempo de resposta:**

- Um RTT para iniciar a conexão TCP
- Um RTT para requisição HTTP e primeiros bytes da resposta HTTP para retorno
- Tempo de transmissão de arquivo