MC833A – Programação de Redes de Computadores

Professor Nelson Fonseca

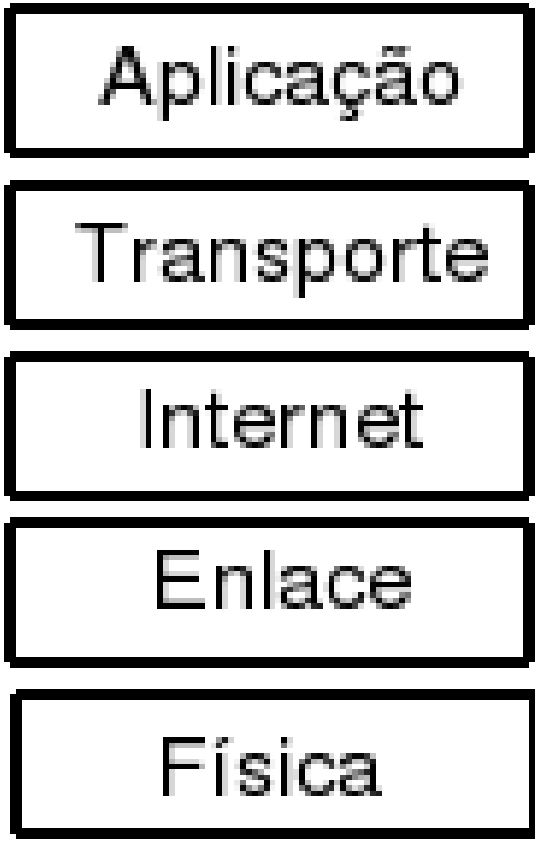
http://www.lrc.ic.unicamp.br/mc833/

Roteiro

- **Objetivo: revisar conceitos aprendidos em MC822 e uma breve introdução a programação com sockets (Capítulos 1 e 2 do livro texto)**

- Arquitetura Internet
- Detalhes de uma comunicação via Internet
- TCP x UDP
- Portas e serviços
- Protocolos e serviços
- Sockets em SOs Unix-like
- Programas úteis no GNU/Linux
- Como seriam os algoritmos?

Arquitetura Internet (1)

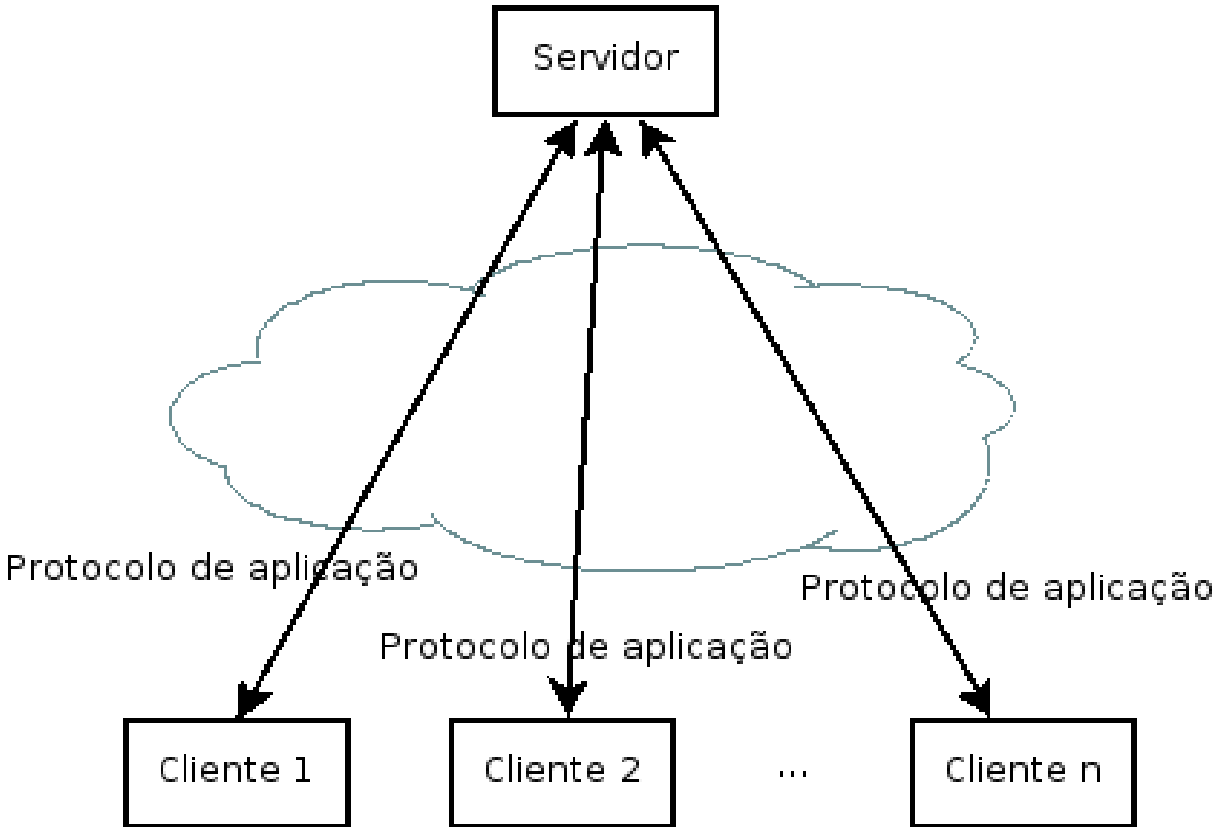


Arquitetura Internet (2)

- “TCP/IP”
- Funções bem definidas
- Encapsulamento
- Abstração
- Facilidade na implementação

Obs.: Foco em cliente-servidor

Cliente-Servidor



Passo-a-passo da comunicação - cliente

- Usuário acessa máquina pelo nome
  - Protocolo de aplicação DNS descobre o endereço IP
  - No acesso todas camadas envolvidas
  - Pode haver cache
  - Pode haver mais de um endereço IP
- Aplicação solicita serviço da camada de transporte
  - Prepara conexão ou começa a enviar dados
  - Define valores que garantem exclusividade na comunicação (cliente-servidor)
  - Faz verificação de erro (na verdade em muitas camadas)

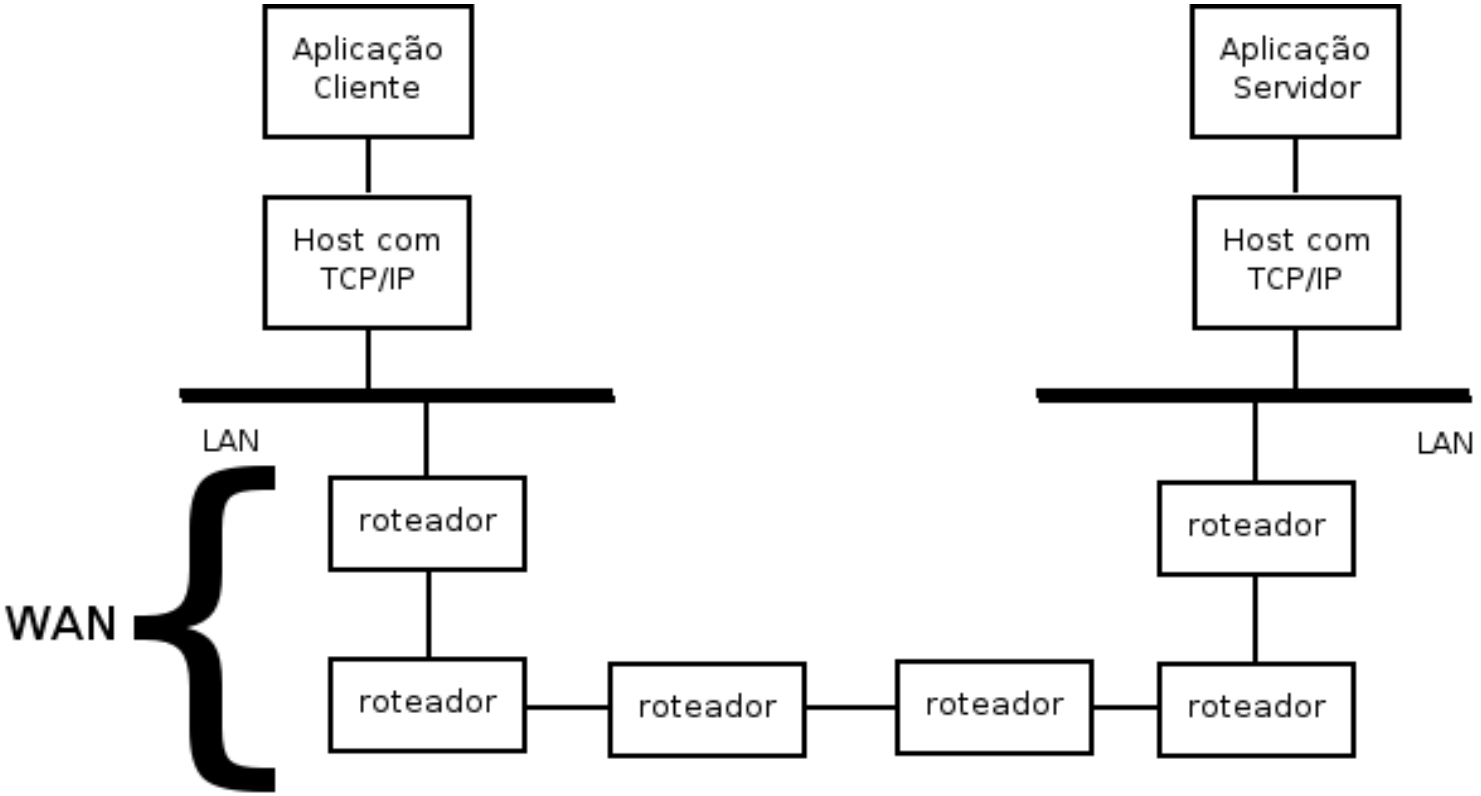
Passo-a-passo da comunicação - cliente

- Transporte solicita da rede o serviço de buscar o caminho
  - Análise do endereço IP, máscara, gateway padrão
  - Protocolo de roteamento
- Rede solicita ao enlace que identifique os pontos da comunicação
  - Endereço de hardware
  - Rede local
  - Switches
- Enlace solicita da física que transmita os bits
  - Volts, cabos, luz

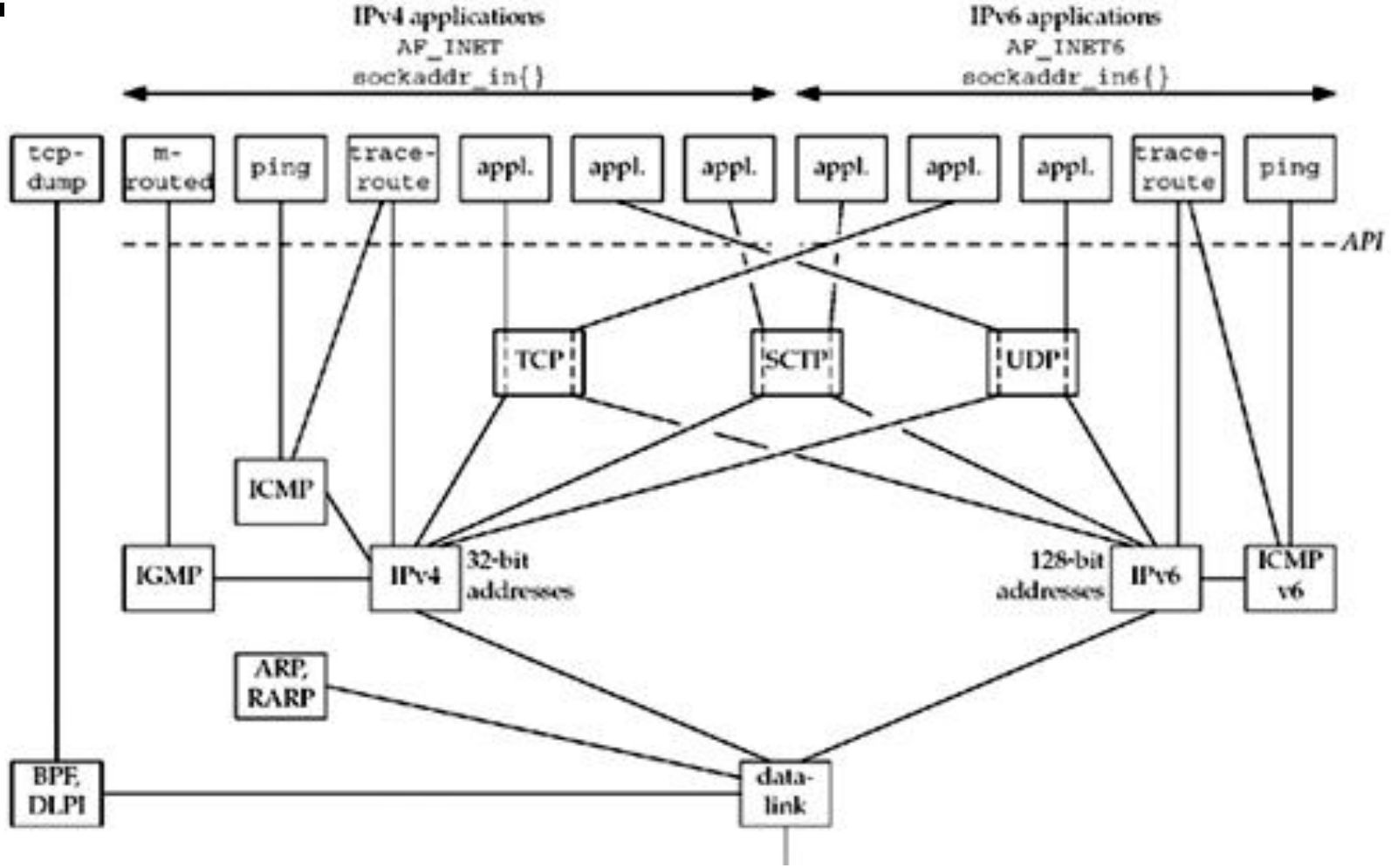
Passo-a-passo da comunicação - servidor

- Física informa chegada de dados na placa de rede
  - Luz, volts transformados em bits internamente
- Enlace verifica os dados de endereço físico e passa restante para a camada de rede
- Camada de rede verifica endereço IP e repassa para transporte
- Transporte verifica porta e repassa para aplicação (Aplicação descoberta)
- Aplicação detecta pedido do cliente e toma a ação necessária (DNS reverso, respostas, confirmação da conexão, negação)

Em resumo (1 cliente e 1 servidor)

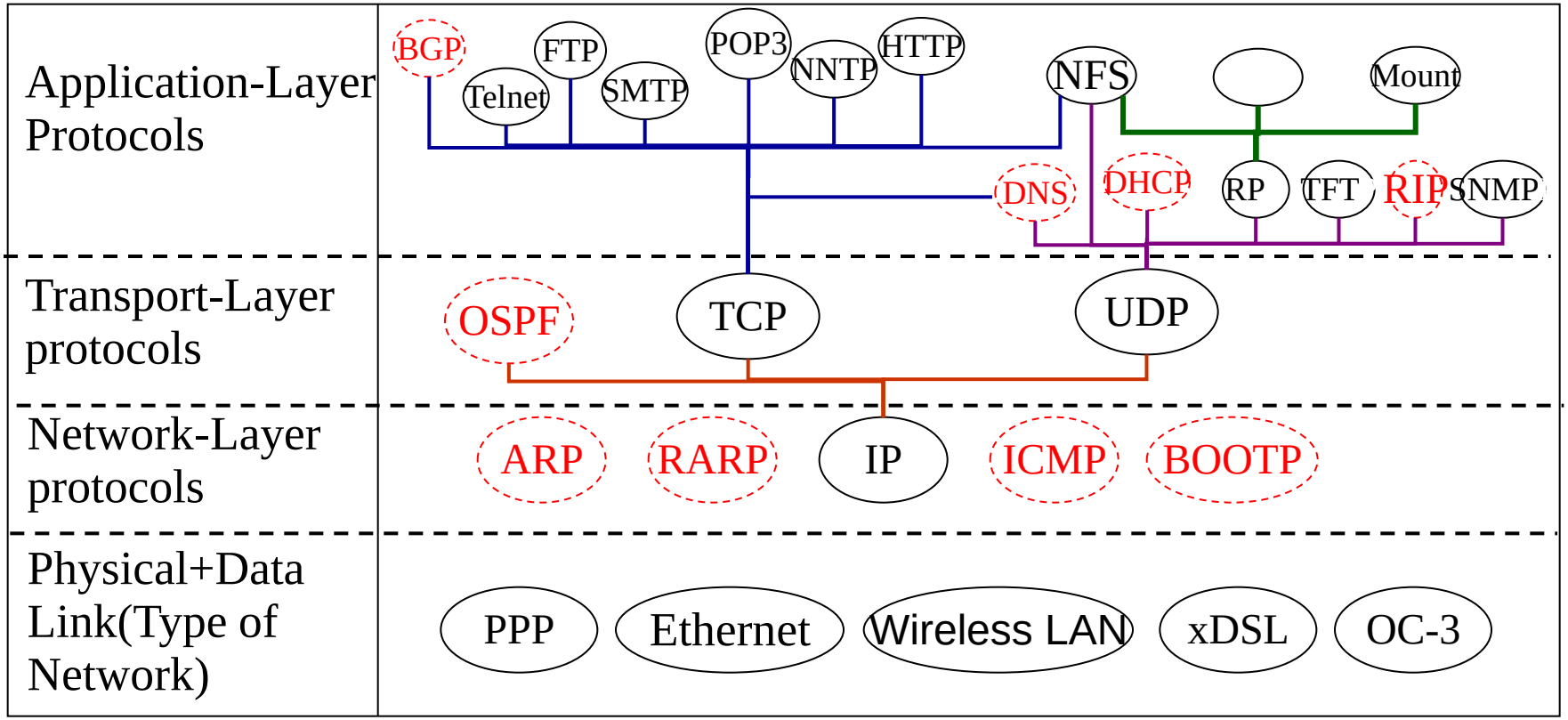


Arquitetura Internet (visão geral) "TCP/IP"



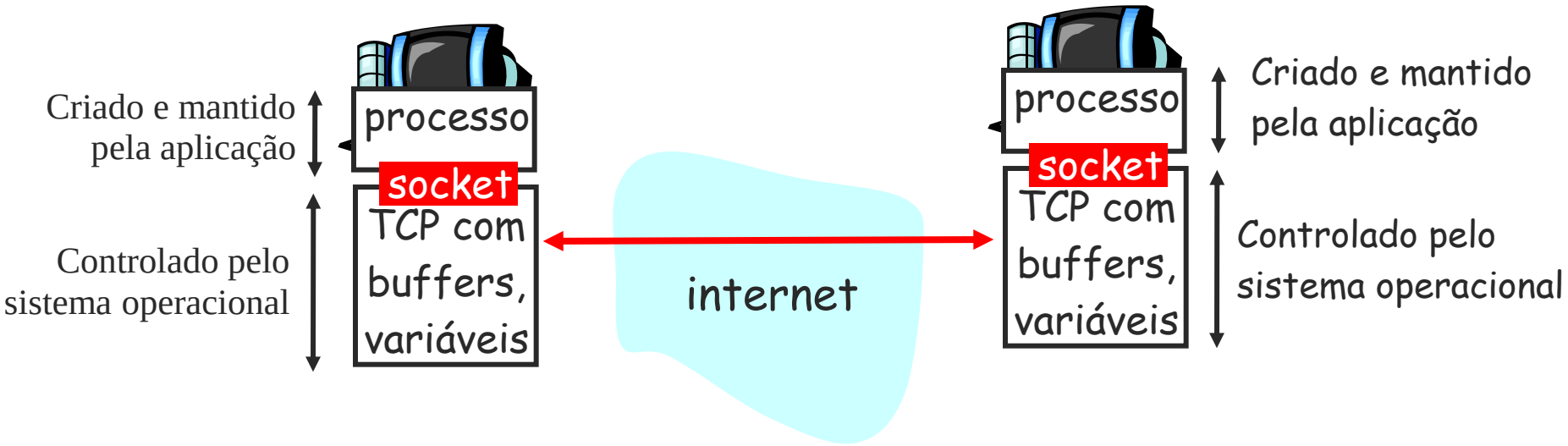
Internet Protocol Tree

Commonly Used Protocols



TCP

- Transferência confiável
- Um “pipe” entre dois processos remotos
- Entrega em ordem e garantida (buffers fazem um papel fundamental)



TCP

- Estabelecimento de conexão
  - 3-way handshake
- Garantia de entrega
  - ACK
- Controle de fluxo, controle de congestionamento
  - Janelas deslizantes
  - Crescimento exponencial, aditivo, decrescimento multiplicativo
- Full-duplex
- Cabeçalho: Porta fonte, Porta destino, Número de sequência, ACK, flags, janela, checksum, etc...

Quais as vantagens?

- Garantia de entrega
- "Network friendly"

Quais as desvantagens?

- Overhead no estabelecimento, fechamento e no cabeçalho

Ideal para transferência de arquivos

Em uma rede com muitas conexões TCP e datagramas UDP, algum protocolo é mais prejudicado?

TCP: Seq e ACK

A

B

envia 'C'

Seq=42, ACK=79, data = 'C'

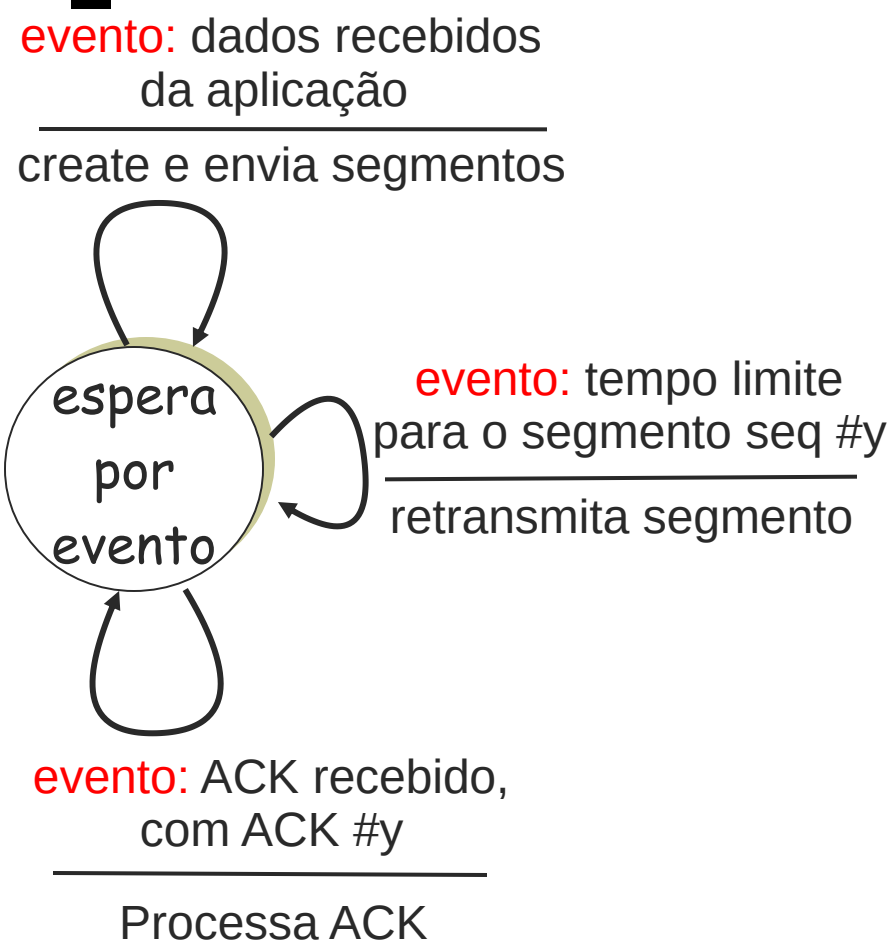
Seq=79, ACK=43, data = 'C'

Seq=43, ACK=80

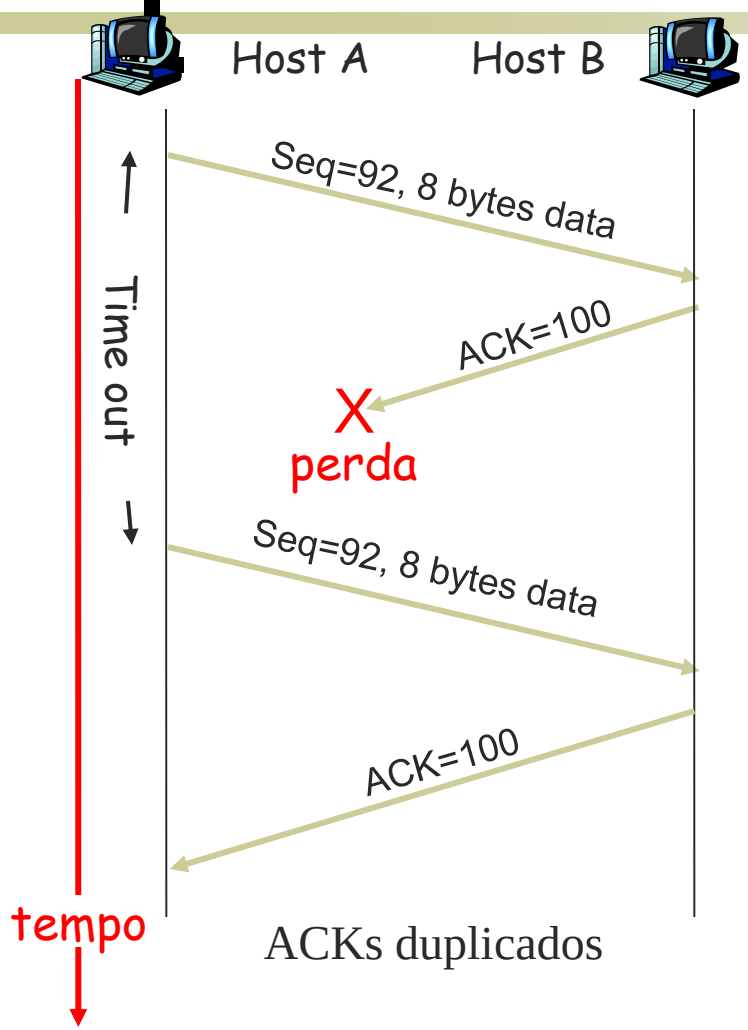
tempo

cenário simples de telnet

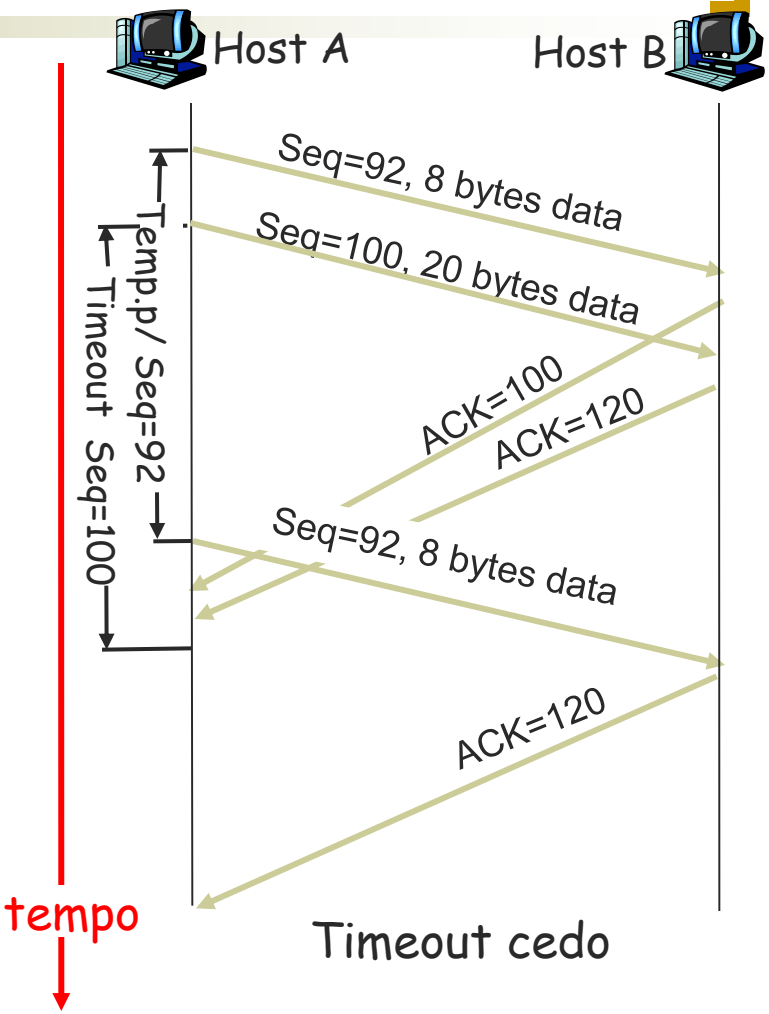
TCP: transferência confiável



TCP: cenários de retransmissão

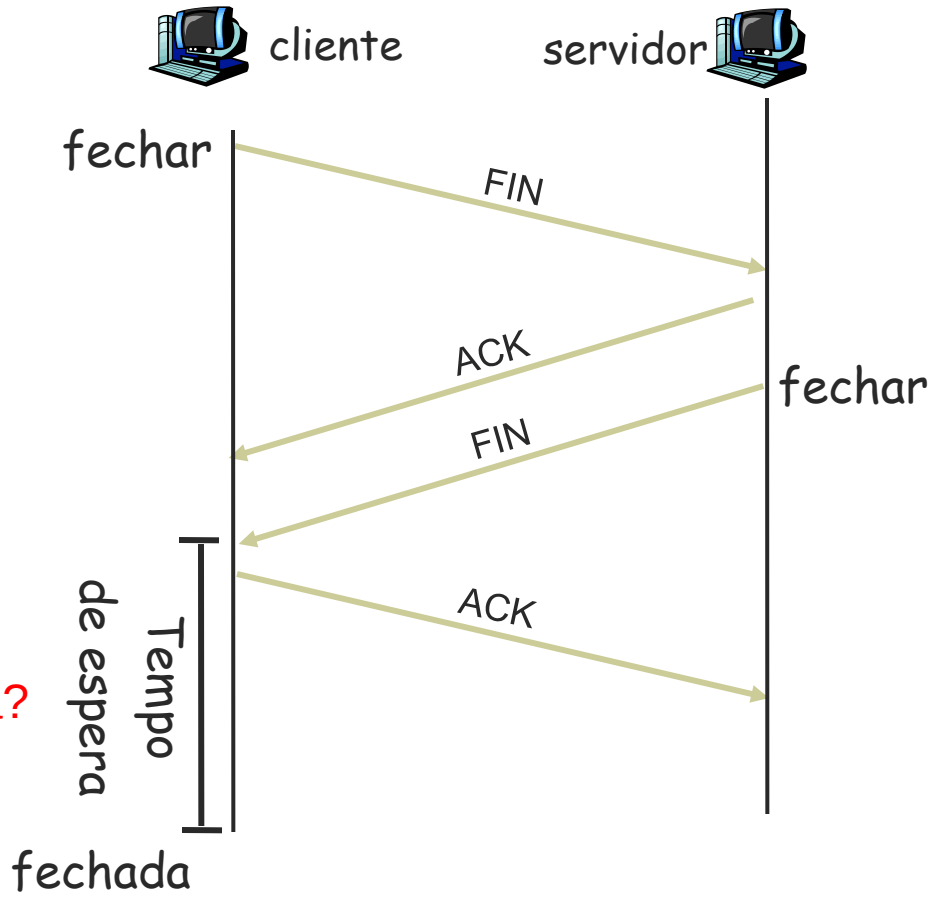


ACKs duplicados



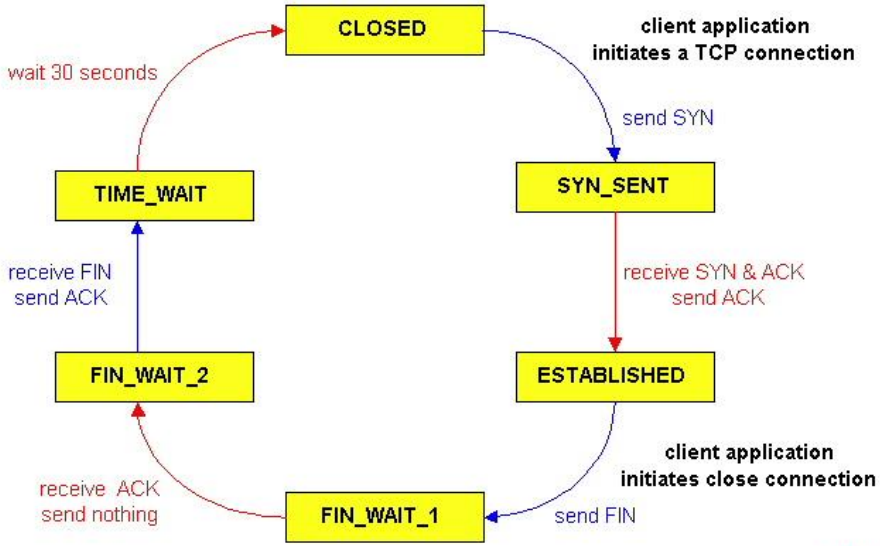
Timeout cedo

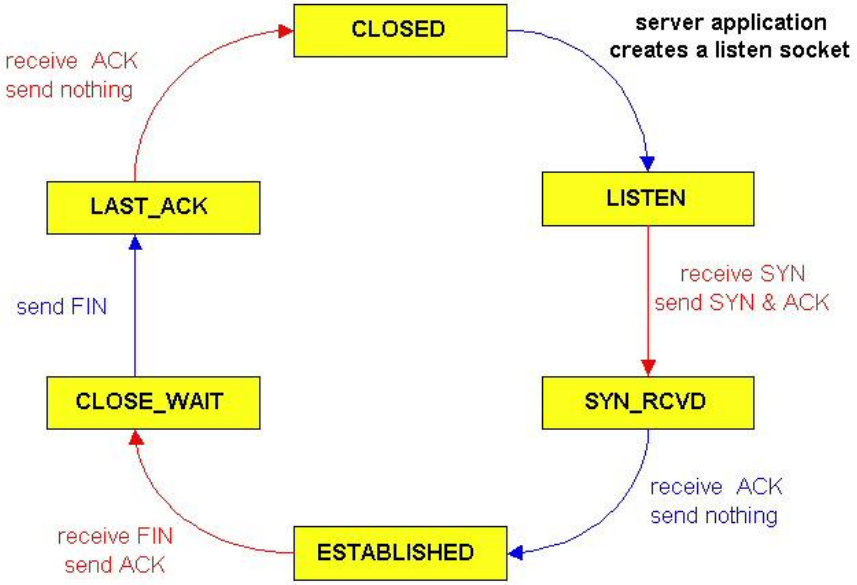
TCP: fechamento da conexão



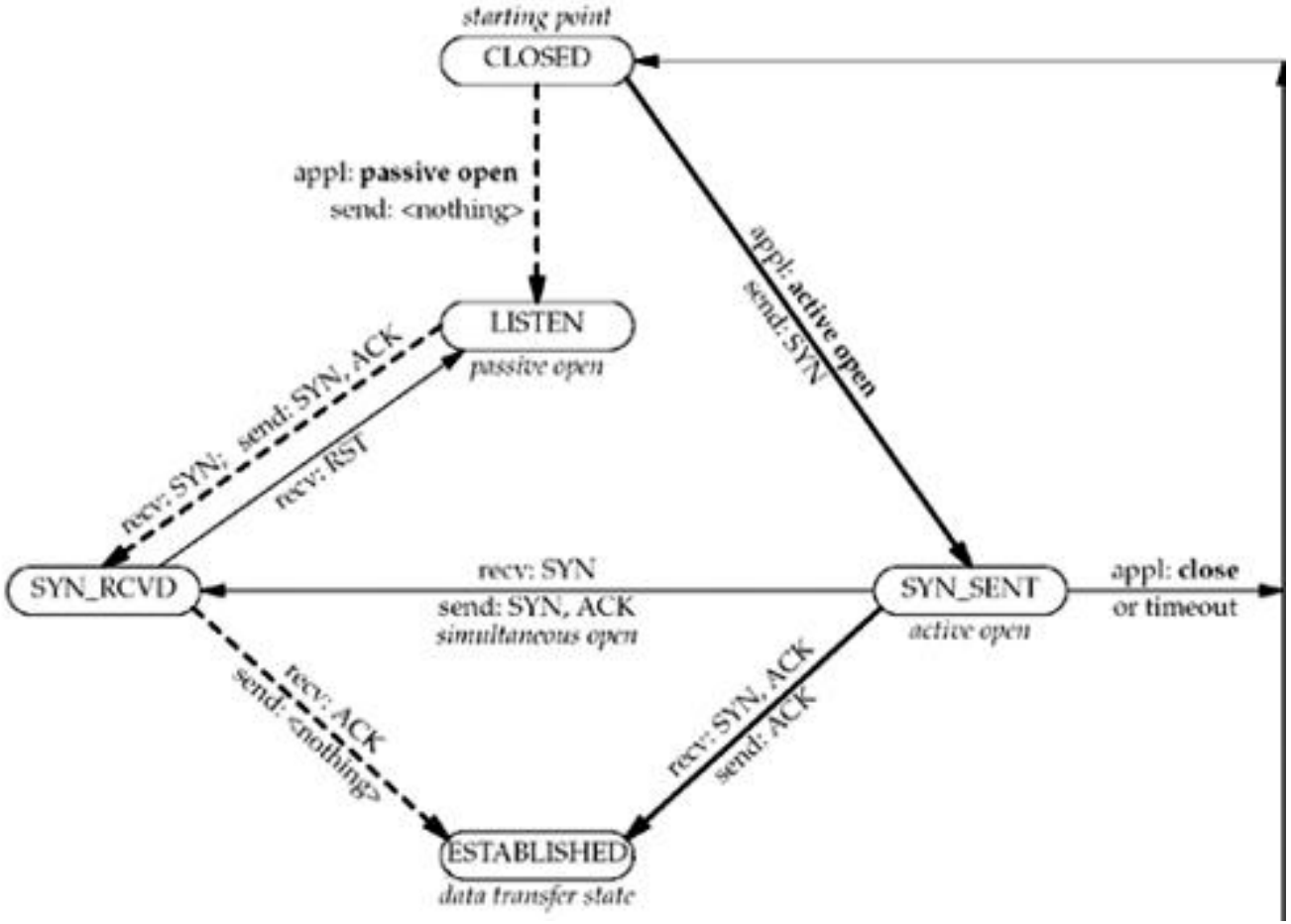
Para que o tempo de espera?

TCP: gerência da conexão

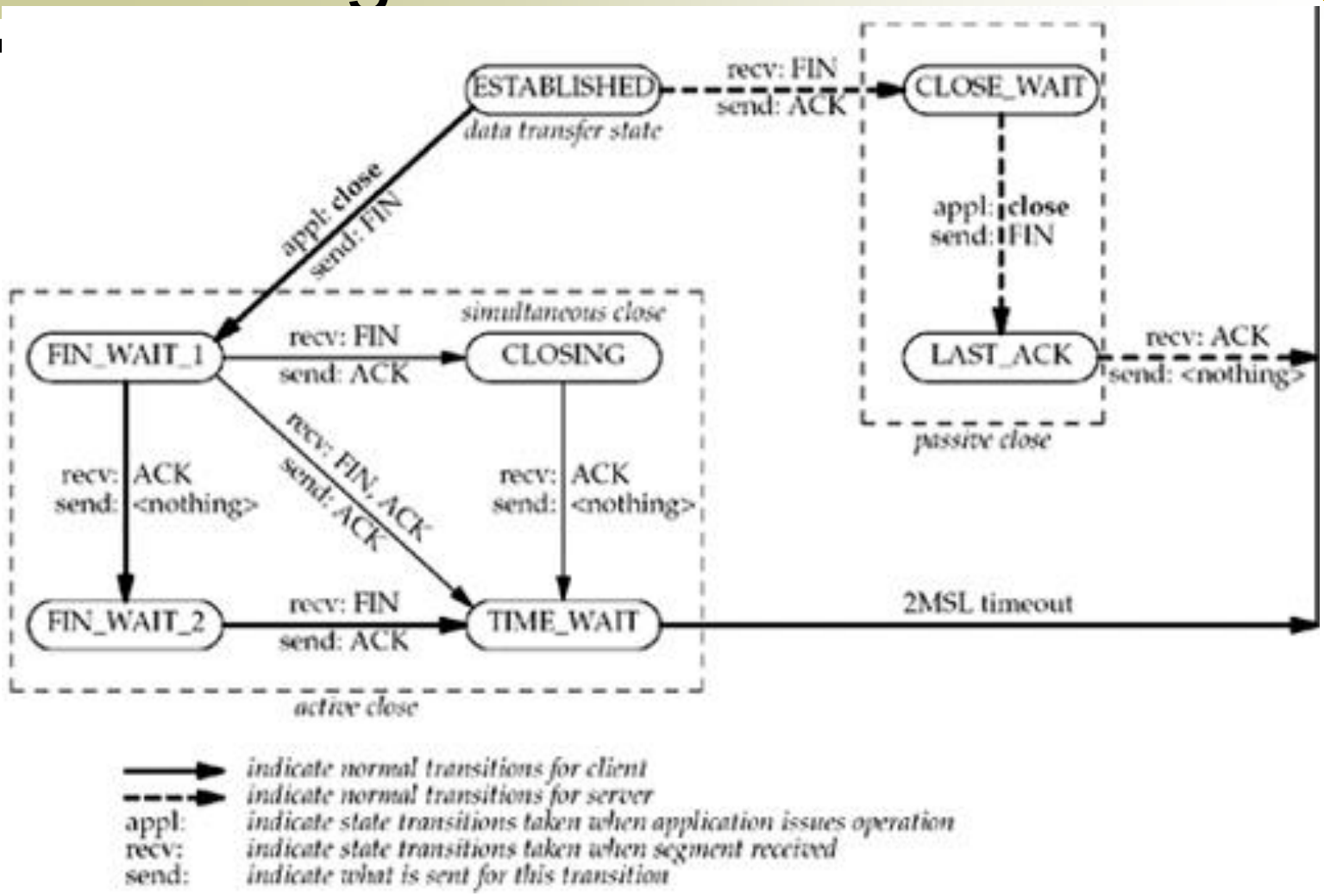




TCP: transição de estados



TCP diagrama de estados



UDP

- Serviço melhor esforço
- Segmentos UDP podem ser:
  - Perdidos
  - Entregues fora de ordem
- **Sem conexão:**
  - Não há estabelecimento de conexão
  - Cada mensagem é processada individualmente
- Cabeçalho: Porta fonte, Porta destino, comprimento e checksum

Quais as vantagens?

Sem o overhead do estabelecimento das conexões

Sem overhead do cabeçalho maior

Quais as desvantagens?

Sem controle de congestionamento

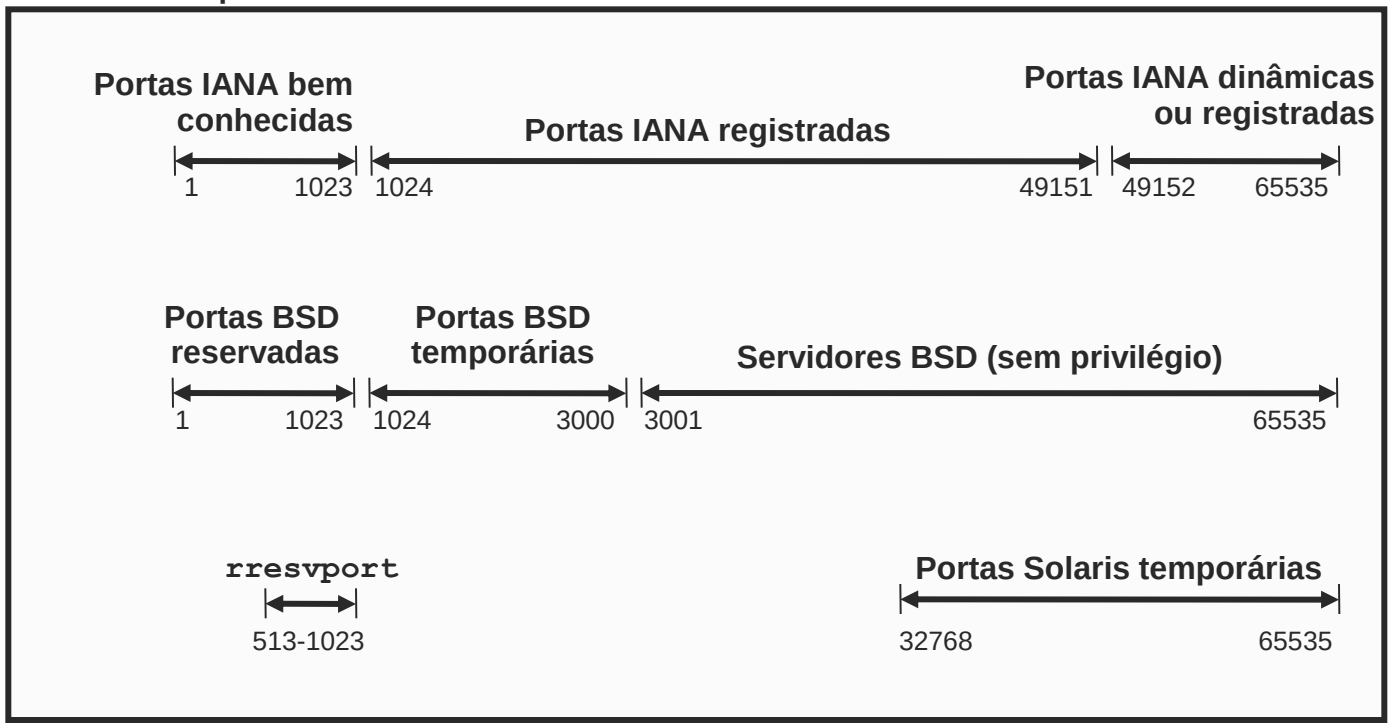
Sem controle de fluxo

Ideal para multimídia

Pode ser usado em transmissão de arquivos?

Números de portas

- IANA
- Portas bem conhecidas são de 0 a 1023
  - Em um SO Unix-like elas são acessíveis somente pelo root
  - Arquivo `/etc/services`



Números de portas

- Como implementar a associação de portas temporárias?
- Quais os cuidados que devem ser tomados?

Aplicações e protocolos

Application	IP	ICMP	UDP	TCP	SCTP
ping		•			
traceroute		•	•		
OSPF (routing protocol)	•				
RIP (routing protocol)			•		
BGP (routing protocol)				•	
BOOTP (bootstrap protocol)			•		
DHCP (bootstrap protocol)			•		
NTP (time protocol)			•		
TFTP			•		
SNMP (network management)			•		
SMTP (electronic mail)				•	
Telnet (remote login)				•	
SSH (secure remote login)				•	
FTP				•	
HTTP (the Web)				•	
NNTP (network news)				•	
LPR (remote printing)				•	
DNS			•	•	
NFS (network filesystem)			•	•	
Sun RPC			•	•	
DCE RPC			•	•	
IUA (ISDN over IP)					•
M2UA,M3UA (SS7 telephony signaling)					•
H.248 (media gateway control)			•	•	•
H.323 (IP telephony)			•	•	•
SIP (IP telephony)			•	•	•

Sockets

API Sockets

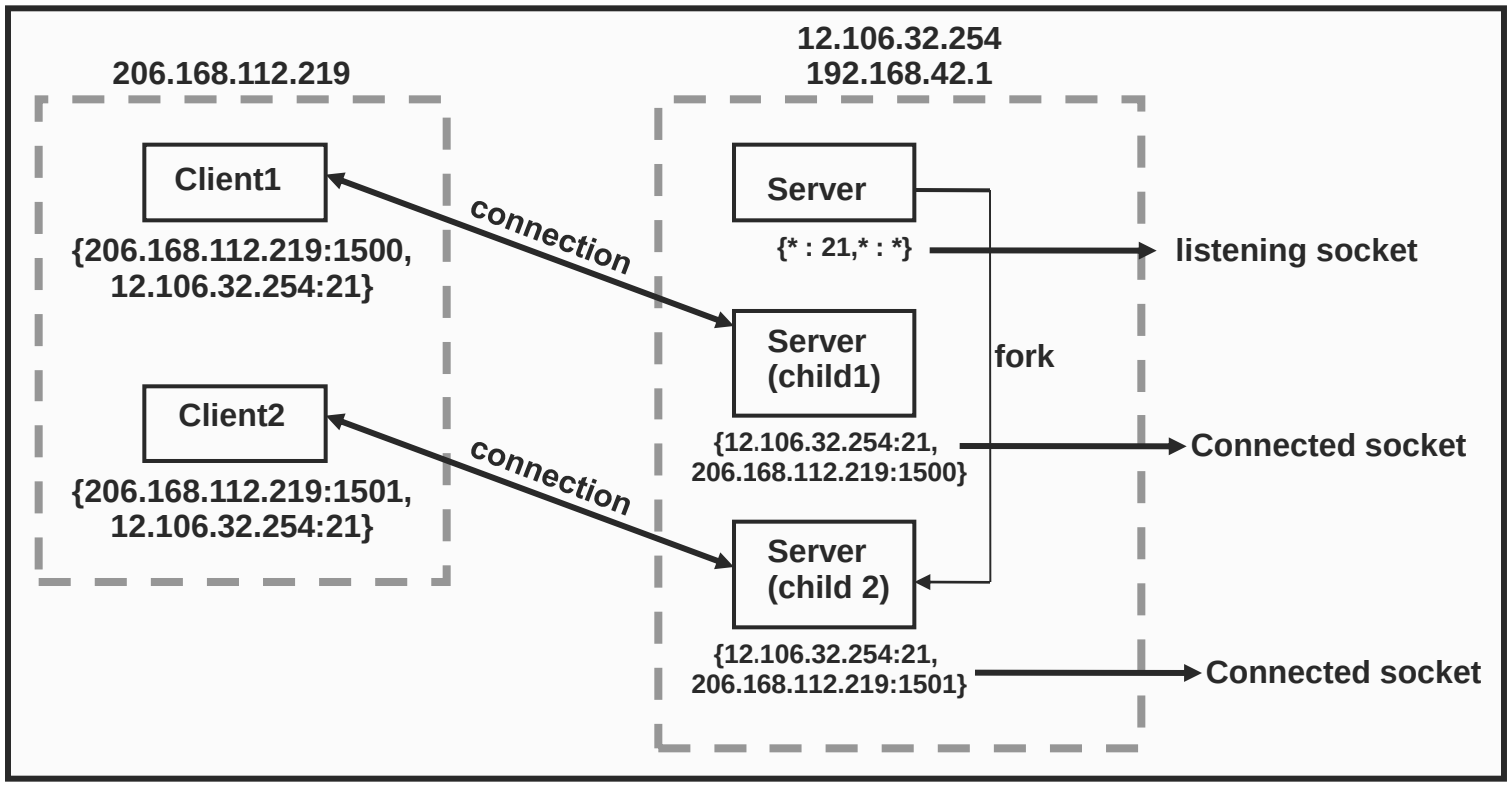
- Em SOs Unix-like, tudo é arquivo!
- Socket = “o arquivo para comunicação de programas via rede”
- cliente/servidor
- Dois tipos de serviço de transporte via API Sockets
  - Datagrama, entrega não confiável
  - Fluxo de bytes, entrega confiável

socket

Uma interface (“porta”),
local,
criada e mantida pela
aplicação e controlada
pelo sistema operacional ,
que permite mandar
e receber mensagens para
processos remotos

Identificação de sockets

- <endereço IP local, porta local, endereço IP remoto, porta remota>



1.4 Open Source Implementations

Open vs. closed

Taxonomy of open source packages

Software architecture in Linux systems

Kernel modules

Interface drivers

Clients and daemon servers

Open vs. Closed

What to open: interface or implementation?

Open: Internet (interface), Linux (implementation)

Closed: IBM SNA (System Network Architecture), Microsoft

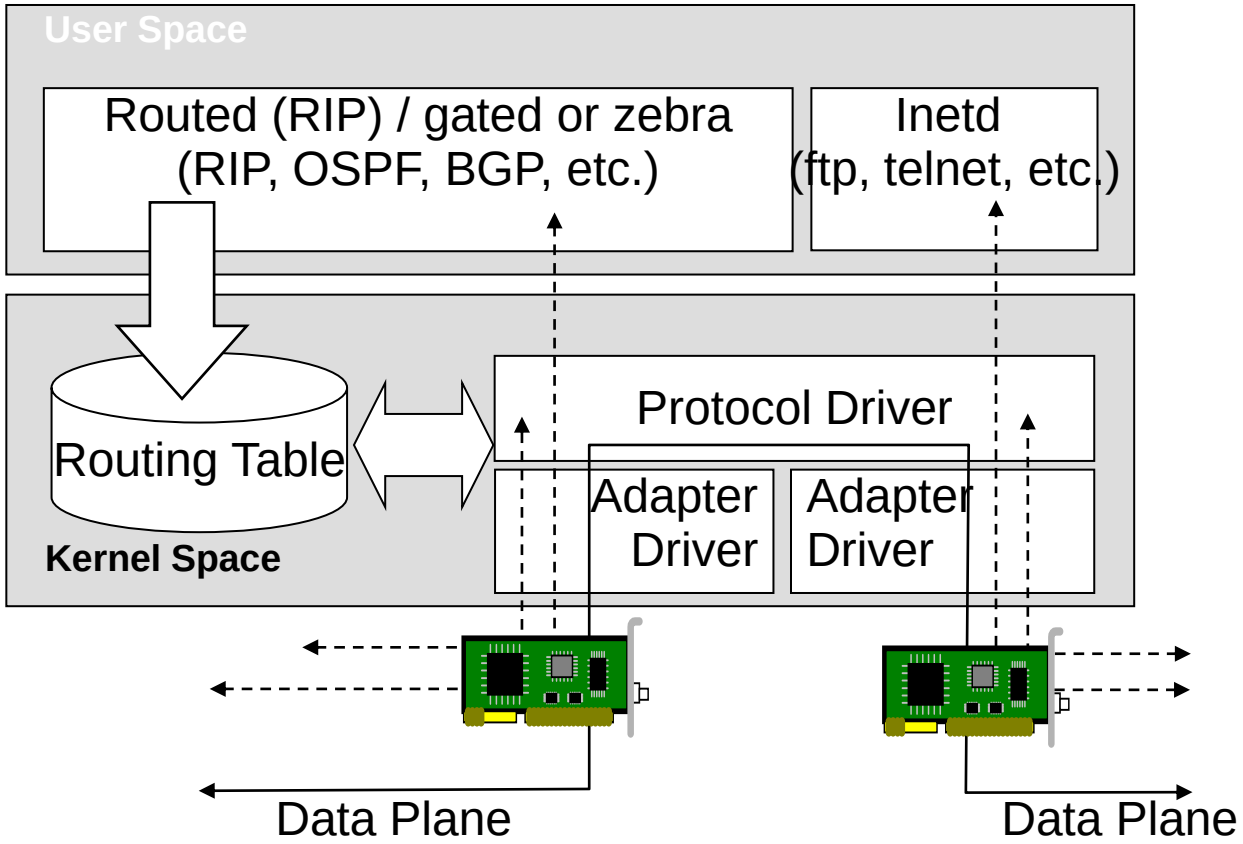
Virtues to open interface

- Interoperability

Virtues to open implementation

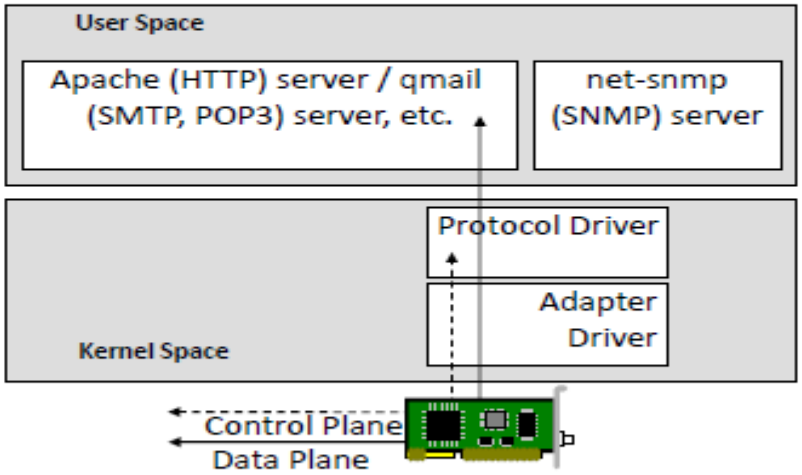
- World-wide contributors
- Fast updates and patches
- Better code quality

Software Architecture in Linux Systems: Router

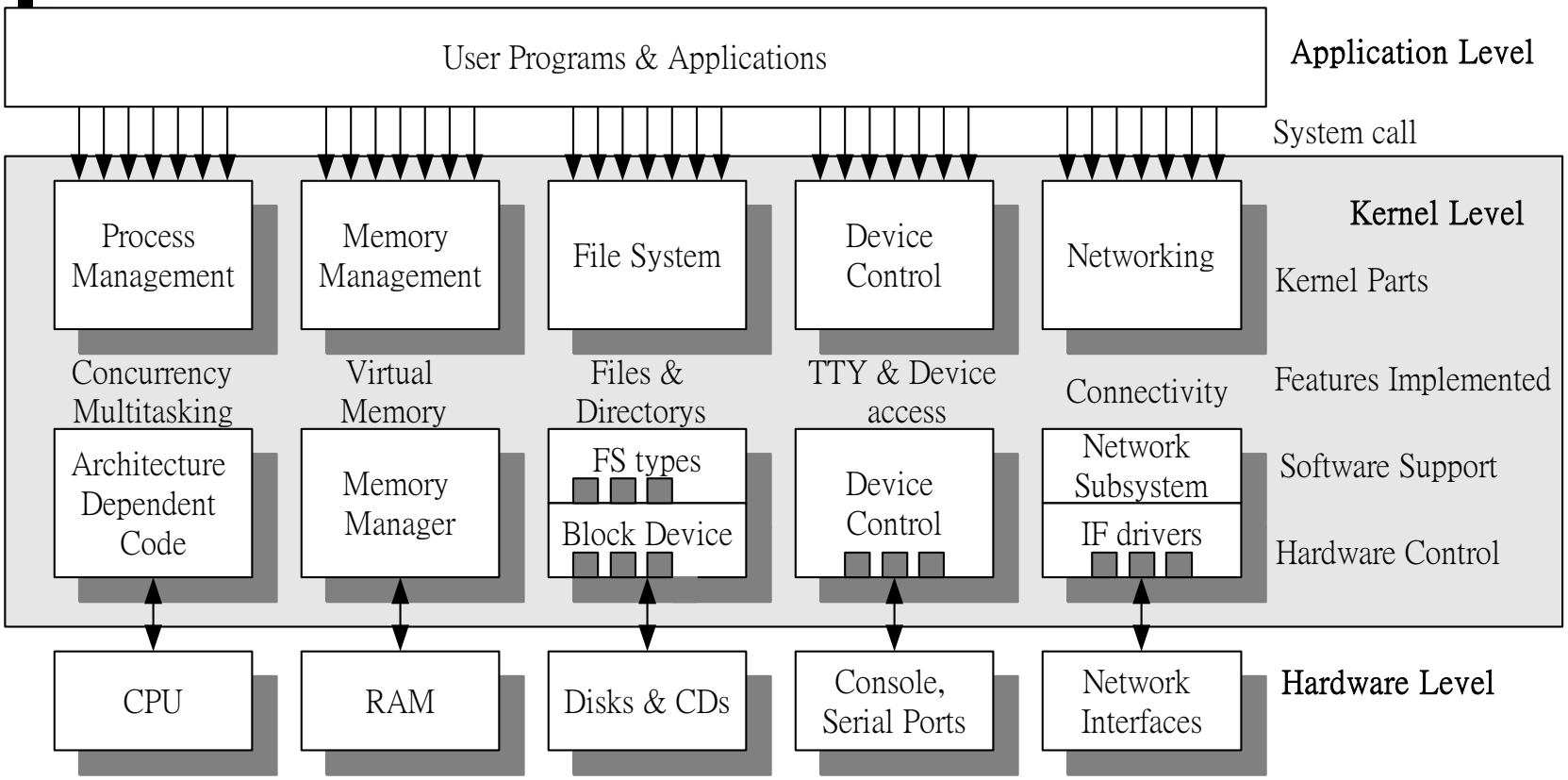


Software Architecture in Linux Systems: Host

Software Architecture in Linux Systems: Host

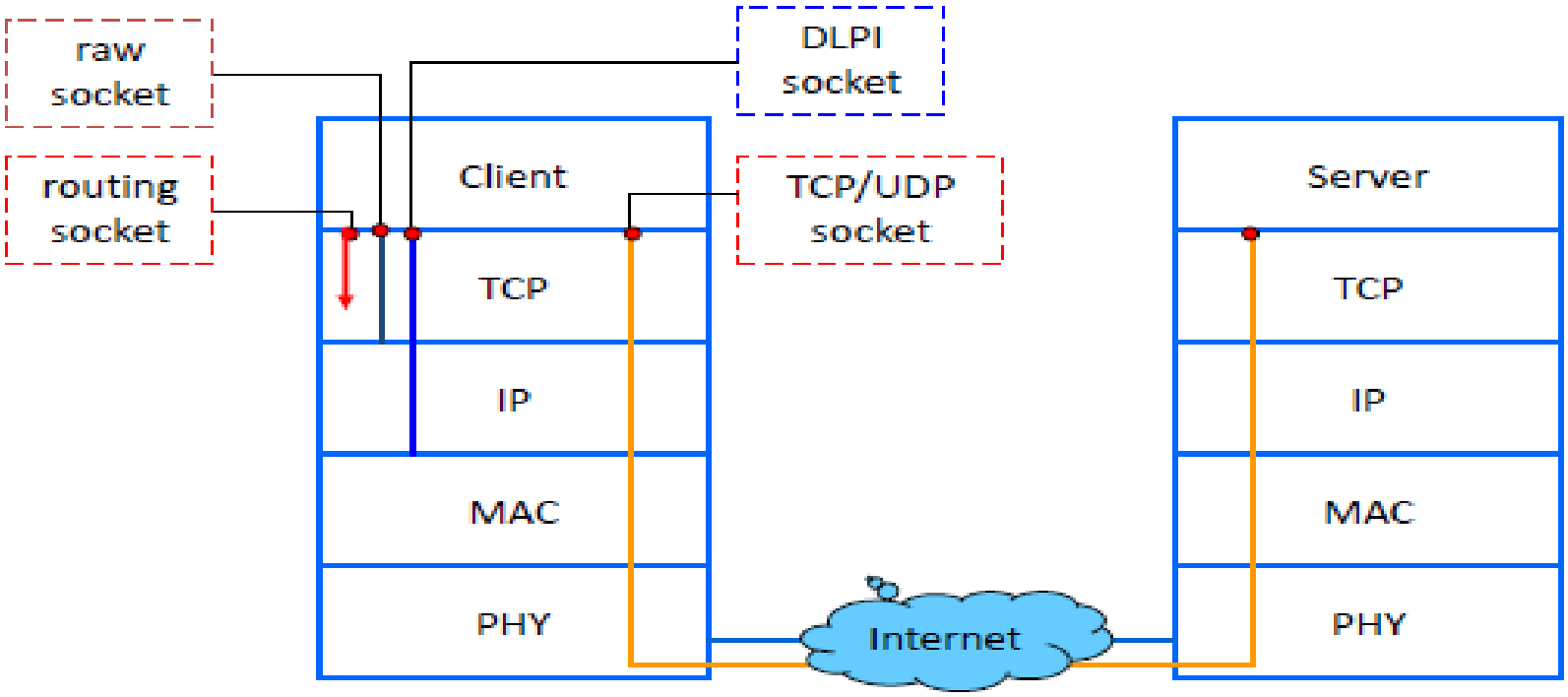


Kernel Components

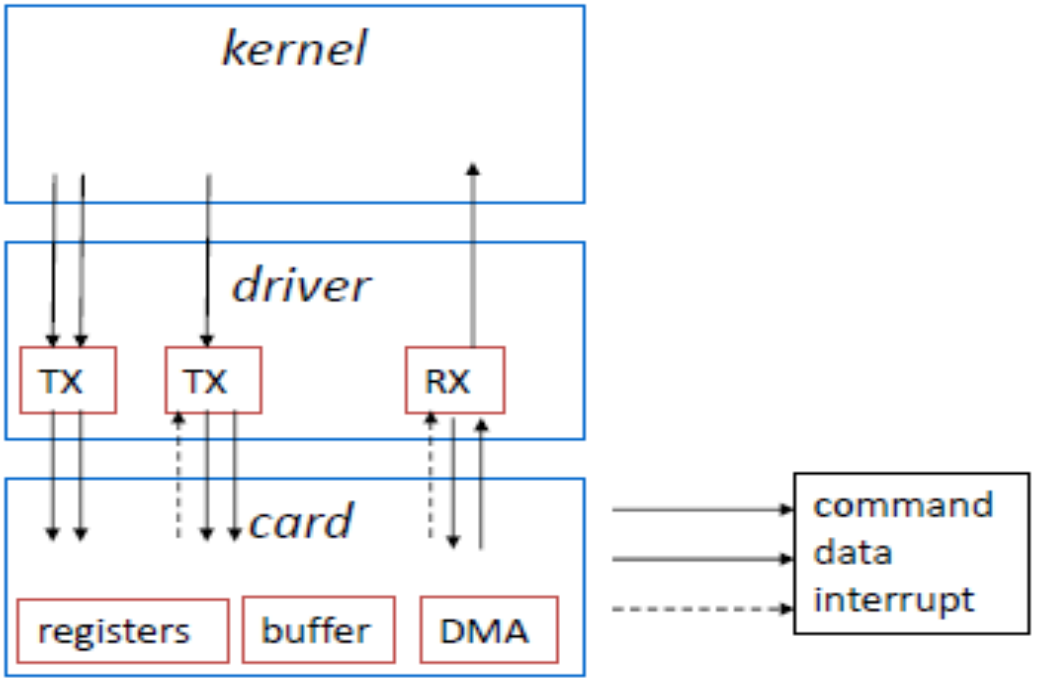


Clients and Daemon Servers

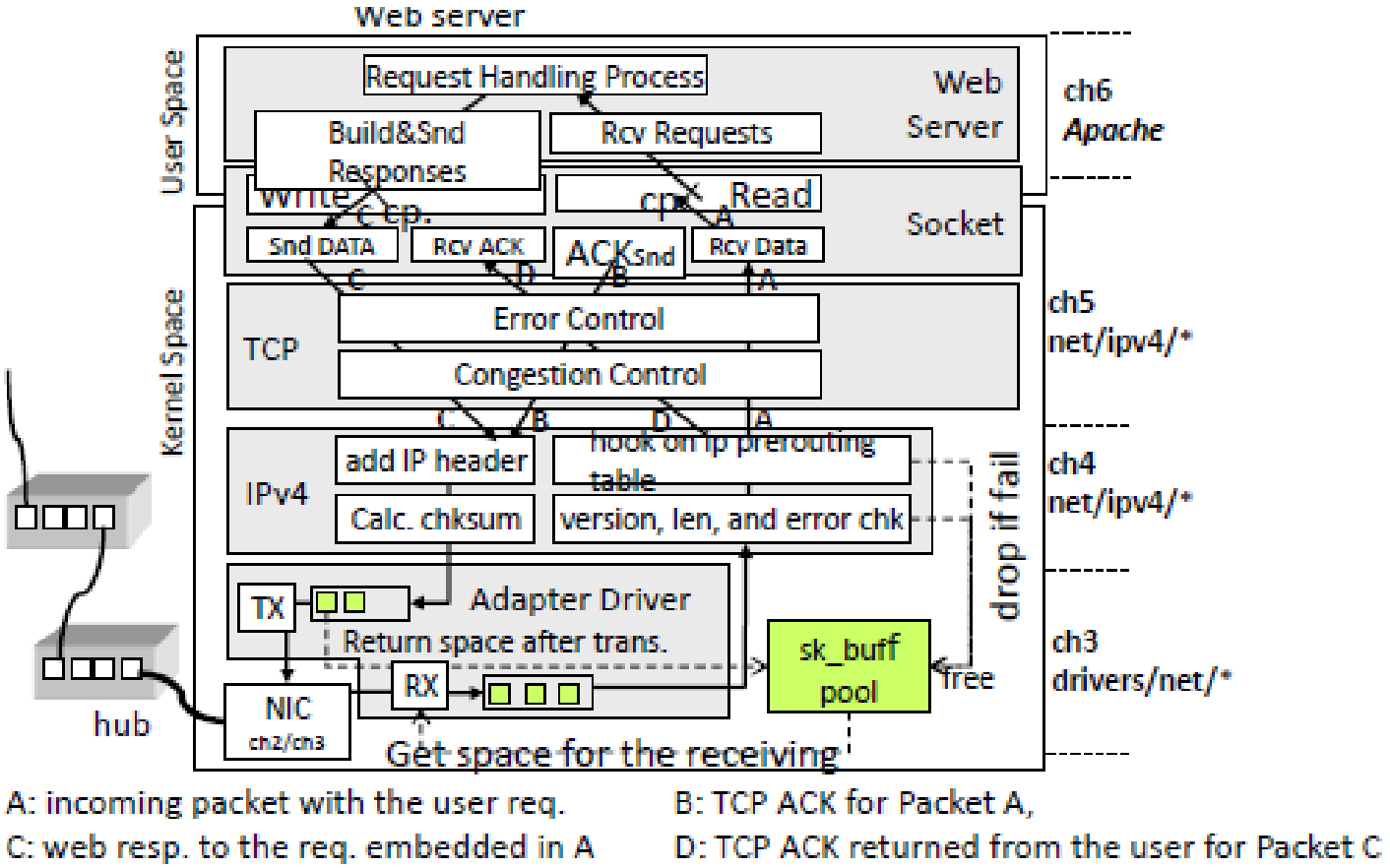
Socket APIs: TCP, UDP, raw, link, routing



Interface Drivers: In and Out

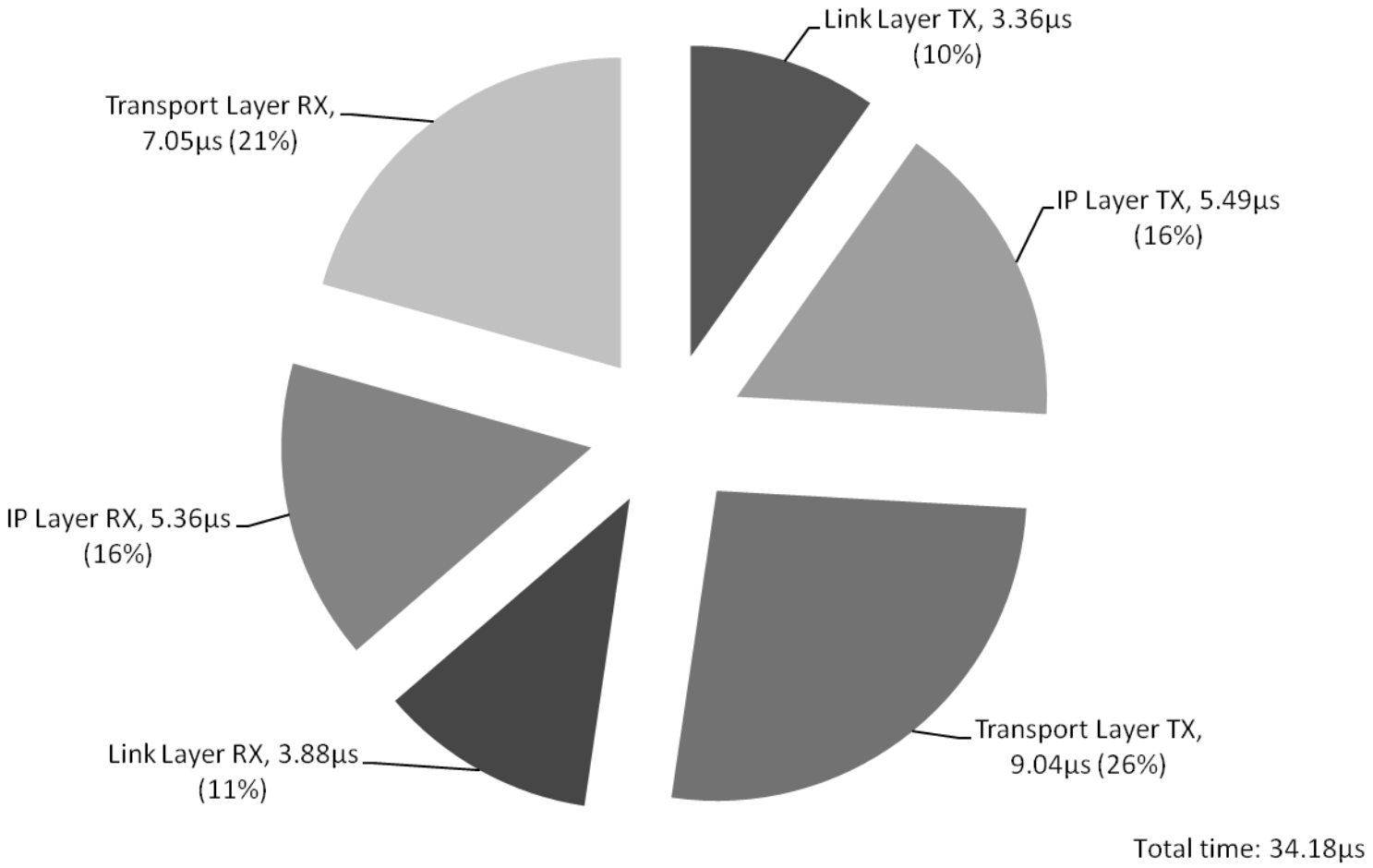


Book Roadmap
Packets' Life in a Web Server

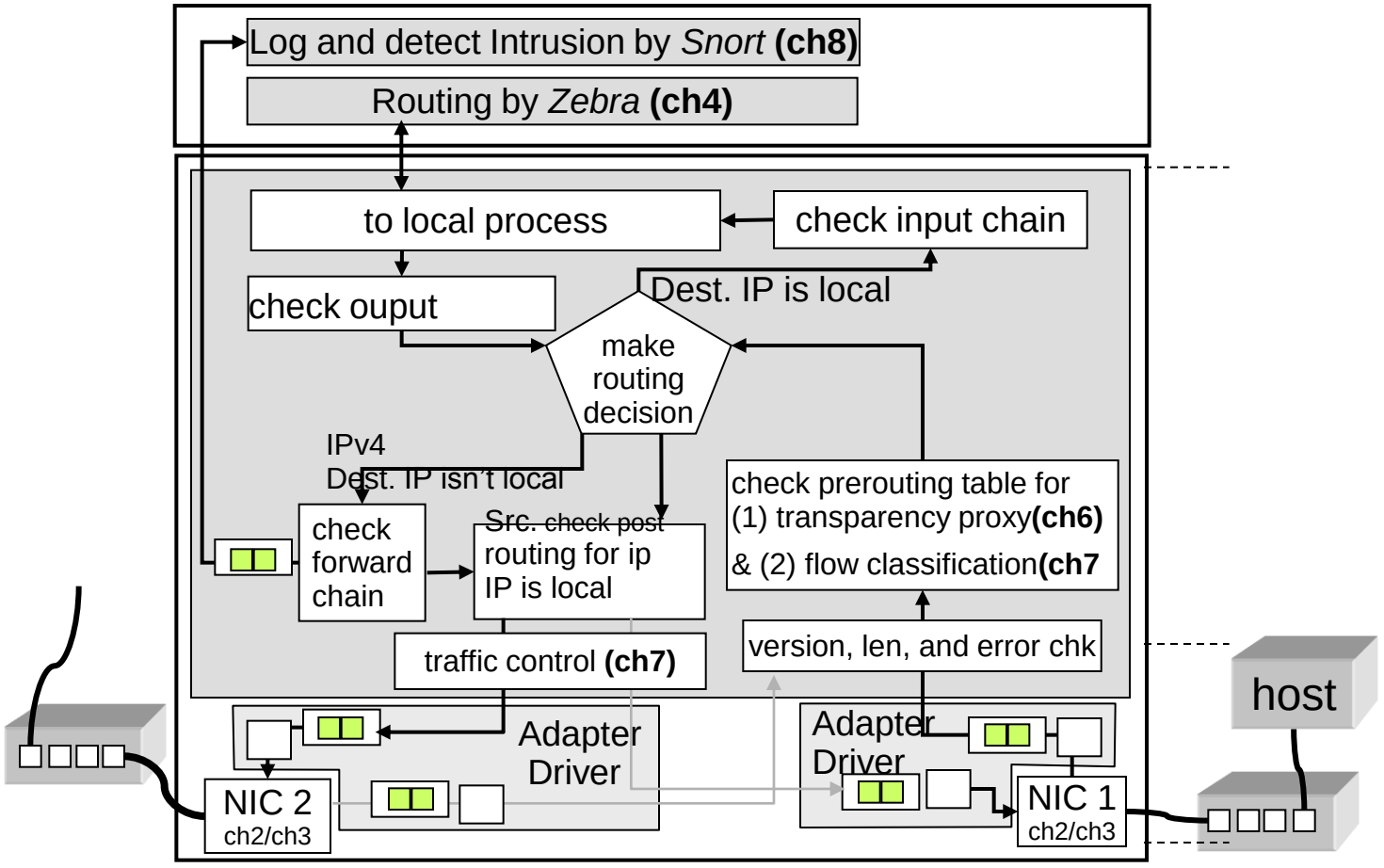


A: incoming packet with the user req.
B: TCP ACK for Packet A,
C: web resp. to the req. embedded in A
D: TCP ACK returned from the user for Packet C

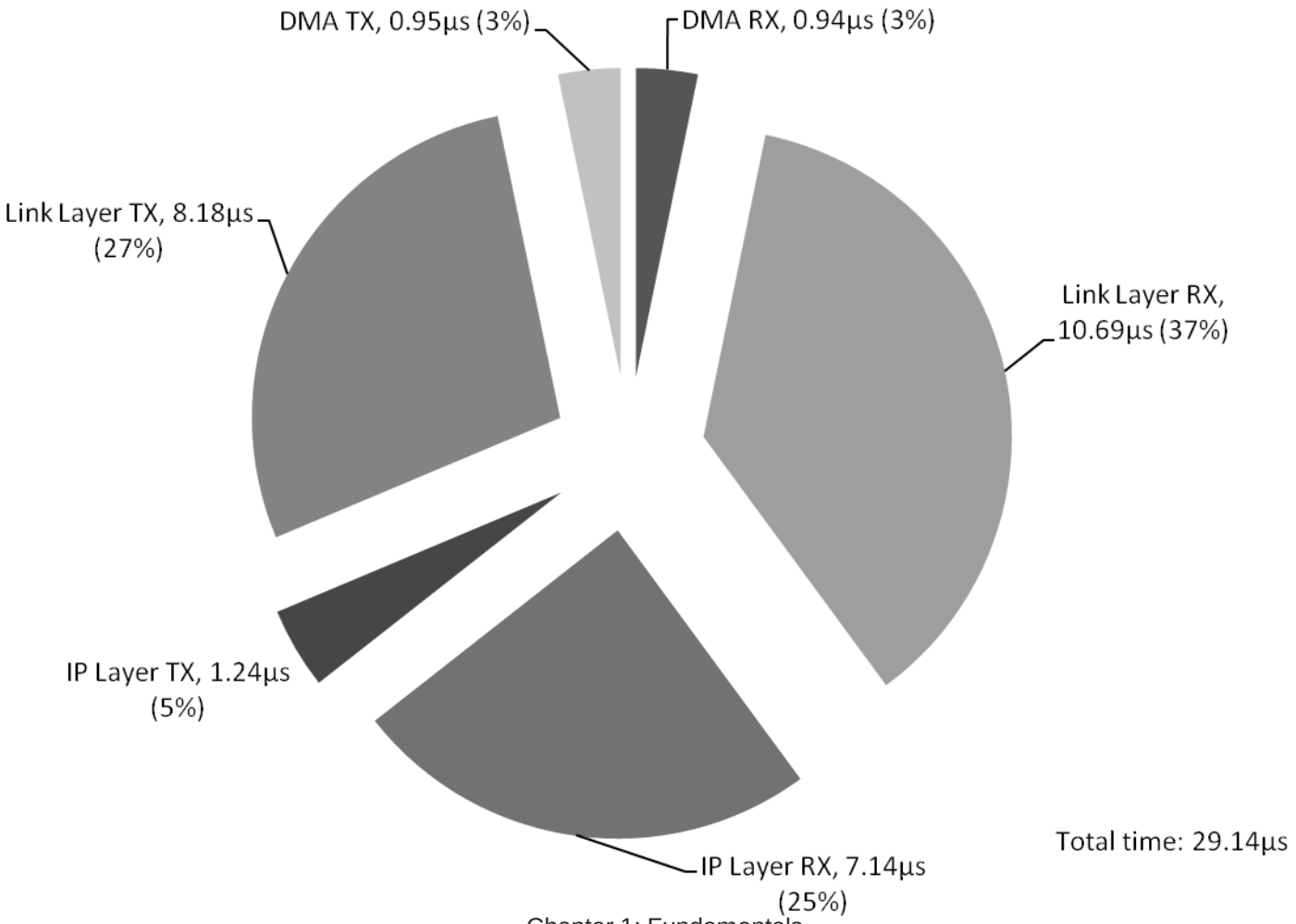
Performance Matters: From Socket to Driver within a Server



Book Roadmap
A Packet's Life in a Router



Performance Matters: From Input Port to Output Port within a Router



Algoritmos (Exercício)

- Escrever os algoritmos (alto nível)
  - Cliente/servidor TCP
  - Cliente/servidor UDP