

# Formulário Pcp - André Senna 28/280

## Camada Rede

- Routing e IP
- ICMP
- encapsula segmentos

## IP

- máx de 60 bytes de Header
- nos fragment em switching L2
- o campo TTL é inferior a 256
- Fragment offset - posição
- detecta erros efeitos de cabecalho
- quem manda pode pedir para nos fragmentar
- endereço: 32 bits

32 bits		32 bits	
ver	hlen	type	length
1 bit identifier	flag	fragment offset	
TTL	upper layer	header checksum	
source IP		dest IP	
options		data (...)	

## DHCP - uma UDP

- encade de timeout, o OS gera o IP na rede 169.254.0.0/16
- DHCP Discovery é em difusão
- mantém necessidade de ARP
- envia IP, server DNS e outros
- Server → Client pode ser broadcast ou unicast
- relay → server sempre unicast

## NAT

- proxy de portas
- encade de IP v4
- viola o princípio end-to-end
- numa rede privada, remota ou routers precisam de NAT
- é necessário recalcular check
- 16-bit de port number

## ICMP

- echo reply é request pelo mesmo caminho (ttl=0)
- message = type, code + 20 bytes do payload do echo + 8 bytes do header (timestamp +)

## IPv6

32 bits		flow label
ver	len	next header hop life
source IP (128 bits)		
dest IP (128 bits)		
Data (...)		

## Routers

- 0.0.0.0/0 é rota por omnião
- tabela não é consultada IGP.
  - ↳ preenchida dinamicamente

## Ethernet

- não corrige erros
- em estrela não há colisão
- não existe backoff
- campo q verifica a integridade permite 10Gb/s em fibra
- des de colisão → back off
- retransmite em caso de colisão
- payload dim máx de 1500bytes
  - ↳ 10Gb/s - 100 Gbps
- não viajável (CSMA CA)
- CSMA/CD - colisão detectada
- pacetable (fbytes) para sincronizar clock rates
- address (6 byte) de MAC type - protocolo decimal (IP)

## CMC - detecção de erros (no receptor)

## MAC Address

- único atribuído pelo fabricante
- FF...FF = broadcast
- 48 bits - 6 bytes (parte inferior gerida pelo fabricante)

## ARP

- resposta é unicast
- sempre é broadcast
- MAC/IP / TTL permite outras MAC através do

- DHCP Policy Agent é necessário nos routers dos PCs nessa mesma interface

- A 1s resposta de um traceroute não pode ser echo reply (em UDP, não em ICMP)
  - ↳ a última, se vier, é port unreachable

half-duplex → um de cada vez

full-duplex → trocando simultaneamente

## Redes Privadas

- 1.10.0.0/8 → 10.255.255.255
- 172.16.0.0/12 → 172.31.255.255
- 192.168.0.0/16 → ... .255.255

## CDP - IP e MAs

- Data plane - HW (má)
- Control plane - SW (má)
- SW → ciclicamente
- WFQ → maior prioridade de 1.0

# Formulário Pcp - 48280 André Jesus

## Cálculo da eficiência

$$T_p = \frac{D}{V_p} \rightarrow \text{distança}$$

$\rightarrow \text{velocidade de propagação}$

$$T_{tx} = \frac{\# \text{bits}}{\text{bit/s do canal}} \rightarrow \text{Gbits/s (10^9)}$$

$$a = \frac{T_p}{T_{tx}} \rightarrow 1+2a$$

$$P_f = 1 - (1 - P_e)^L \rightarrow \# \text{bits}$$

	Canal Sem Erros	Canal com Erros
Stop-and-wait	$U = \frac{t}{1+2a}$	$U = \frac{1-P_f}{1+2a}$
Selective Repeat	$U = \frac{N}{1+2a}, N < 1+2a$	$U = \frac{N(1-P_f)}{1+2a}, N < 1+2a$ $U = 1 - P_f, N \geq 1+2a$
Go-back-N	$U = 1, N \geq 1+2a$	$U = \frac{N(1-P_f)}{(1+2a)(1+P_f(N-1))}, N < 1+2a$ $U = \frac{1-P_f}{1+P_f(N-1)}, N \geq 1+2a$

DSL → Permite transmissão de TV sobre IP sobre cabos de cobre

- Dedicado
- Assimétrico
- DSLAM (Multiplexor)
- Up < 1 Mbps / Down < 10 Mbps
- Splitter e modem (Telefones)
- Cabo Coaxial

- Partilhado
- FDM
- Assimétrico
- Comutador de Pacotes
- Up < 21 Mbps / Down < 50 Mbps
- Splitter e cable modem (net e TV)
- Euro DCCSIS 3.0

- ATT do Selective Repeat > do Go-back-N
- UDP - 8 bytes / TCP - 20 a 60 bytes
- DNS - 512 bytes (UDP 99,9% e TCP 0,1%)
- upstream e downstream → central
- Go-back-N apenas descarta duplicados
- stateless - não mantém dados de pedidos anteriores
- Não há estabelecimento de ligação em UDP

GPON

- Partilhado
- Assimétrico
- TDM
- ONT usa timeslot para OLT / TV e Net

Up < 2,214 Gbps (TDMA)  
Down < 2,488 Gbps

Comutação

- Circuitos
- Dedicado
- Switches
- Largura de banda constante
- Estabelecimento de ligação (maneira alternativa que é com sucesso)

Pacotes

- Partilhado
- Routers
- Stop-and-forward
- não na ordem
- Largura de banda dinâmica
- Internet

SMTP → os comandos SACPT TO e MAIL FROM,  
nos permitem de link a origem e destinatário  
no User Agent

↳ Persistente TCP (Port 25)

↳ Transferência Direta

↳ + Bit ASCII

↳ Entrega/Armazenamento ao Server

↳ é possível forçar a origem de um mail

IMAP

↳ entrega ao User Agent

↳ guarda mensagens no Server

↳ mantém o estado do User

POP

↳ entrega ao User Agent

↳ tem autenticação

↳ nos mantém o estado do User

DNS → Serviços TLD são guardados em cache  
nos servidores de nomes locais  
(forwarders)

↳ Port 53 (UDP)

↳ recursivo (máx para a root)

↳ iterativo (utilizado)

A → hostname e IP

NS → nome do Domínio

↳ hostname do Server Autoritário

PTR → IP → hostname

CNAME → alias do nome real

↳ nome canonical (real)

MX → nome + Mail Server

nslookup www.ipsel.pt.

↳ port de onde queremos IP

8.8.8.8

Port que DNS Server queremos enviar a query

TCP Segment Structure

32 bits	
Source Port	Dest Port
Seq. Number	
Ack. Number	
Len SYN...  Receive Window	
Checksum	Pointer
options	
data	

UDP Datagram

32 bits	
Src Port	Dest Port
Length	checksum
data	
Options e portas de destino no seu utilização	

HTTP

↳ não-persistente; 1 object por ligação  
persistente: múltiplos por ligação

↳ stateless

↳ headers

- Etag - versão do recurso
- Connection → se a conexão se manteve aberta
- Keep-Alive → quanto tempo a conexão persistente deve durar

$$\text{Estimated RTT} = (1-\alpha) \text{Estimated RTT} + \alpha \text{sample}$$

$$\text{DevRTT} = (1-\beta) \text{DevRTT} + \beta |\text{sample} - \text{EstRTT}|$$

$$\alpha = 0,125 \quad | \text{Timeout Interval} = \\ \beta = 0,25 \quad = \text{Estimated} + 4 \text{ DevRTT}$$

↳ deltas não depende da quantidade de pacotes assim como a curva de transferência

• TCP faz multiplexação de canais lógicos utilizando o porto  
↳ se o porto é o mesmo, vai para máquinas diferentes

• Em CDN nem todos os nós têm o mesmo ficheiro

• Em Bit Torrent, o mais rápido só transfere do próximo

• Fisical define o caminho de linha usada no meio de transmissão

• Envio ao meu top-four, recebo de múltiplos

• Rede WiFi residencial utiliza IPs privados e necessita de NAT

DNS Messages

Question	#Answers
#Answers	#Additional
#Authority	Answers
questions	Answers
answers	Authoritative
authoritative	Additional info

loss por timeout:

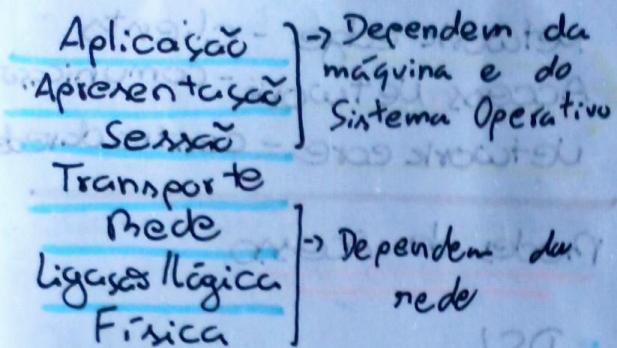
- currd = 1 loss
- slow start

• Duplicado (TCP NACK)

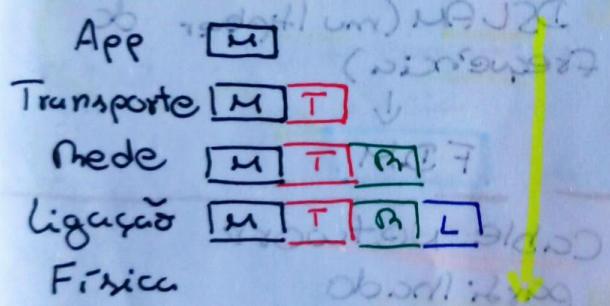
- currd/2 e aumenta imediatamente
- retrans
- currd = 1

## Redes 1

### Modelo OSI (7)



- Modelos em camadas para que uma mudança / nova implementação numa delas, não interfira com as outras.
- Cada camada presta serviços à camada superior.



Cada uma destas mensagens é interpretada por camadas do mesmo tipo.

**PDU** Protocol Data Unit

O nome que se dá às mensagens:

- App - Mensagem
- Transporte - Segmento
- Rede - Pacote / Datagrama
- Ligação - Trama / Frame
- Física - Trama física

## Redes 1

### Modelo TCP/IP (5)

9.0T - betwixt O mitosma  
 → transmission control Protocol  
 → camadas

Presentación e Session;

Aplicação - suporta aplicações de rede (ex: HTTP)

Transporte - transferência de dados extremo-a-extremo (ex: TCP, UDP)

Rede - encaminhamento dos pacotes da origem ao destino (ex: IP)

Ligaçāo - transferência de tramas entre elementos de rede vizinhos \* (ex: Ethernet, Wi-Fi)

Física - tudo relacionado com hardware (bits)

Notas:

- Máquinas (PCs) têm todas as camadas;
- Switchs têm L e F;
- Routers têm R, L e F;

\* Trama / Frame structure



## Connection Oriented - TCP

transmission  
control  
protocol

- ↳ existe uma conexão em camada de transporte;
- ↳ sessão é iniciada (3 way hand shake):
  - 1º o remetente envia uma pedido de sincronização
  - 2º o destinatário manda ACK
  - 3º o remetente manda ACK

## transmissão de dados:

- 1º remetente envia dados
- 2º destinatário dá ACK
- 3º se não for recebido o ACK, o remetente reenvia.

## Connectionless - UDP

- ↳ envia constantemente dados
- ↳ mais rápido (internet)
- ↳ menor seguro

Nota: TCP é muito seguro mas causa sobre carga da rede elétrica

- \* RTT (Round Trip Time)
  - ↳ tempo desde o envio do dado até à receção do ACK

## Entrutura da Rede

Network edge - clients

Access Network - comunicar

Network core - centro da rede

## Modos de Acesso

- DSL
- Cable Network
- Fiber (PON)

## Digital Subscriber Line

- ↳ dedicado
- ↳ assimétrico (diferentes frequências)
- ↳ DSL AM (multiplexor de frequência)

FDM

## Cable Network

- ↳ partilhado
- ↳ assimétrico
- ↳ multiplexor de várias zonas FDM

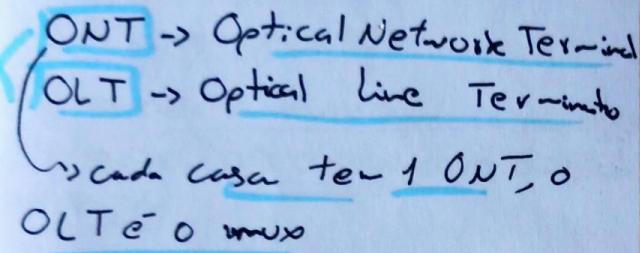
DSL → dedicado → cada conexão liga ao Mux que liga à central

Cable → partilhado → várias conexões partilham o cabo que entra no Mux que vai à central

## Fibra (PON)

- Passive Optical Network
- assimétrica
- partilhada
- cada ONT está ligado a outros até chegar ao OLT.

TDM



## Acesso múltiplo

FDM → multiplexagem da frequência

TDM → multiplexagem do tempo

Nota: NAT e Network

### Address Translation:

descrever os endereços IP de origem de um pacote que passar por um router ou firewall

## Redes 2

## Redes de Comutação

Comutação de Circuitos (switches)

Comutação de Pacotes (routers)

### Circuitos:

- caminho dedicado
- pacotes chegam em ordem
- falha de 1 switch é fatal
- largura de banda constante
- largura de banda possivelmente desperdiçada

### Pacotes:

- caminho partilhado
- store-and-forward \*
- pacotes não chegam em ordem
- largura de banda dinâmica (rajada - Ex: internet)

\* O pacote inteiro tem de chegar ao router para ser transmitido again

### Packet delay

dtrans (Transmission Delay)

↳ colocar os bits no cabo

dprop (Propagation Delay)

↳ mover no cabo

dqueue (Queueing Delay)

↳ tempo de espera no router

dproc (Processing Delay)

↳ verificar o endereço

## Capítulo 2

### Estruturas de Aplicações

#### Client-Server

#### Peer-to-Peer (P2P)

#### Client-Server

server always-on;

IP permanente

clientes comunicam com server

clientes não comunicam diretamente entre si

#### Peer-to-Peer

no always-on server

as pessoas (peers) estão todos conectados e comunicam entre si

#### Sockets

process - program running within a host

o process envia/recebe mensagens para/da socket

client process - inicia a conexão

server process - espera a conexão

para receber mensagens o process tem de ter um identifier

#### identifier

IP Address

Port numbers

#### Web e HTTP

URL - Uniform Resource Locator

Protocol :// Kont :Port /Path

↓  
HTTP      )      Opcional  
HTTPS     www.isel.pt      80  
(...)      (...)      home  
                   dashboard  
              (...)

HTTP -> Hypertext Transfer Protocol

#### Client-Server

- client: o browser requests (pede) e recebe os objetos

- server: o envia os objetos

usa TCP

- o cliente cria a socket, estabelecendo conexão

- o server aceita a conexão

- HTTP mensagens trocadas

- a conexão acaba

## HTTP Request Message

- ↳ request
- ↳ response

**NOTE:**  
Podemos usar o telnet

Request:

- ↳ request line (GET, POST, ...)
- ↳ header lines (Host, ...)
- ↳ carriage return ("\\r\\n")

method URL version cr lf

GET /index.html HTTP/1.1 lf lf

GET - pedir data específica

POST - enviar data para o server

PUT - é semelhante ao POST

mas sempre que o POST é  
chamado tem resultados diferentes

HEAD - semelhante a GET só que  
se responde

DELETE - apaga um recurso  
específico

OPTIONS - descreve o comércio  
(entre) o target resource  
ao

## HTTP Response Message

- ↳ status line

• protocol + status code +  
+ status phrase

- ↳ headers

- ↳ data (HTML file, ...)

## Tipos de Status Message

1xx → informação

2xx → successful

3xx → redirection

4xx → client error

5xx → server error

## Status - Code

200 Ok → request com sucesso

301 Moved Permanently

↳ objeto pedido foi movido

400 Bad Request → não  
reconhecido

404 Not Found → objeto não  
encontrado

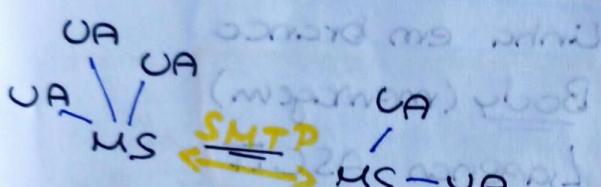
505 HTTP Version Not  
Supported

## Capítulo 2.4

### Correio Eletrônico

#### Intervenientes:

- user agent ("mail reader")
- mail servers (Outlook, etc...)
- SMTP (Simple Mail Transfer Protocol)



UA - user agent

MS - mail server

#### User Agent

#### mail reader

compõe, edita e lê mensagens

#### Ex: Outlook

#### Mail Server

mail box contém as mensagens enviadas ao user

message que contém as mensagens a serem enviadas

SMTP entre mail servers para enviar mails

### SMTP (RFC 2821)

usa TCP para transferir do cliente para o servidor, com a porta 25

Transferência direta de servidor para servidor

#### 3 Fases de Transferência:

- handshaking
- transferência de mensagem
- conclusão

#### Interações Comando/Resposta

- Comando: ASCII (7 bits)
- resposta: Status-Code

#### Comandos SMTP

HELO - inicia a conversa, identificando o server e é seguido pelo nome do domínio

HELO gonicul.com

MAIL FROM - o remetente

identifica o endereço do user

MAIL FROM: <alice@gonicul.com>

RCPT TO → identifica o destinatário, se for + 9 +, é repetido

RCPT TO: <bob@hotmail.com>

DATA → é possível começar a enviar dados, acabar com uma linha com ↴

DATA

(... resposta)

Tudo bem?

QUIT → termina a conversa

Alguns Status Code

220 - Serviço SMTP pronto

221 - Fecho do serviço

250 - Prequest action  
↳ completed

354 - Start message input

SMTP Conclusões

↳ usa conexões persistentes (TCP)

↳ mensagens em ASCII 7 bits

↳ CRLF, CRLF determinam  
fim da mensagem

↳ múltiplos objetos enviados

em multipart message

(em HTTP, cada objeto vai  
na própria resposta)

Formato do Mail

RFC 822 → formato de mensagens de texto  
standard

Header Lines

↳ To:  
↳ From:  
↳ Subject:

Linha em branco

Body (mensagem)

↳ apenas ASCII

Mail Access Protocols

UA → Sender's MS

UA → Receiver's MS

UA → Mail Access Protocol

SMTP → entrega/armação  
ao servidor do destinatário

MAP → entrega do servidor  
ao UA

MAP's:

↳ POP → Post Office Protocol  
(RFC 1939) - autorizações, download

↳ IMAP: Internet Mail Access Protocol  
(RFC 1730) - mais funções incluindo manipulação no servidor

↳ HTTP : gmail, Hotmail, ...

## POP3 Protocol

## Capítulo 2.3

→ Fase de Autorização

↳ comandos do client:

• USER:

• PASS:

↳ <sup>respostas</sup> comandos do server:

• +OK

• -ERR

→ Fase de transação (client:)

↳ list: list message numbers

↳ retr: receive message

↳ dele: delete message by number

↳ quit: message by number

## POP3 vs IMAP

### IMAP

↳ guarda todas as mensagens no server

↳ o user pode organizá-las em pastas

↳ mantém o estado do user

### POP3

↳ cópias das mensagens em clientes diferentes

↳ não mantém o estado do user

## Capítulo 2.3

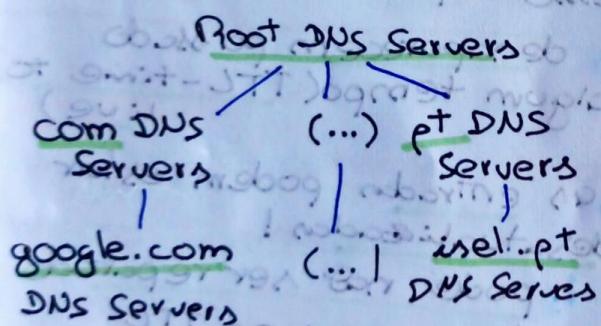
### DNS - Domain Name System

- Faz a translaçao Ip/Name do servidor;
- Base de dados distribuida implementada com hierarquia de muitos servidores de nomes.

### Serviços DNS

- Traduzir de host name para IP
- Host aliasing
- mail server aliasing

### DNS : Database



exemplo: Cliente quer IP de www.amazon.com google

- Client queries root para obter .com
- " " .com para obter google.com
- " " google.com para obter o IP de www.google.com

### Root Name Servers

- são contactados por servidores locais que não conseguem encontrar um nome
- retornam imediatamente TLD para o servidor local

### TLD - Top-Level Domain

#### Servidores TLD:

- são responsáveis por A dominios com, org, net, etc e por dominios dos países

#### Servidores DNS Autoritários

- são de algumas empresas que produzem hostnames para IPs de hosts da empresa

- mantidos pela organização ou pelo serviço que os provê

## Servidor de nome DNS Local

- ↳ Pode não pertencer à hierarquia
- ↳ Cada ISP (provedor de internet) tem um
- ↳ Quando a host faz uma query, é enviada ao servidor DNS local (forwarder)

## DNS Name Resolution

A host PC quer o IP para server.isel.pt

### Query Recursiva:

- ↳ server local pergunta a todo? "eu não sei, mas sei quem pede saber" DNS
- 
- ```

graph TD
    PC[PC] --> LS[Local S.]
    LS --> RootS[Root S.]
    RootS --> TLD[TLDS]
    TLD --> Authoritative[Authoritative S.  
dns.isel.pt]
    Authoritative --> Server[server.isel.pt]
  
```
- The diagram illustrates a recursive DNS query. A PC sends a query to its local server (LS). The local server then queries the root server (Root S.) for the authoritative server for the domain. The root server returns the address of the authoritative server for the domain (dns.isel.pt), which is then resolved by the local server and returned to the PC.

### Query Recursiva

- ↳ passa a pergunta ao outro servidor
  - ↳ mau por causa da sobrecarga do todo da hierarquia
- 
- ```

graph TD
    PC[PC] --> LS[Local S.]
    LS --> RootS[Root S.]
    RootS --> TLD[TLDS]
    TLD --> Authoritative[Authoritative S.  
dns.isel.pt]
    Authoritative --> Server[server.isel.pt]
  
```
- The diagram illustrates an iterative DNS query. A PC sends a query to its local server (LS). The local server queries the root server (Root S.) for the authoritative server for the domain. The root server returns the address of the authoritative server for the domain (dns.isel.pt), which is then resolved by the local server and returned to the PC.

## DNS: Caching / Updating Records

- ↳ Quando um server aprende um mapeamento, ele cache esse mapeamento (cache)
- ↳ desaparecem passado algum tempo (TTL - time to live)
- ↳ entradas podem estar desatualizadas!
- ↳ pode não ser repartido até expirar o TTL

## DNS Records (Registers)

- ↳ DNS é uma database que guarda registros de recursos (RA)

RA format:

(name, value, type, ttl)

### Types

- ↳ name é noname  
value é IP address



## Capítulo 2.4

### DNS Records

- NS → name é o domínio  
(ex: foo.com)
  - value é o host name do servidor autoritário
- PTR → name é o IP reservado
  - value é o host name
- CNAME → name é um alias
  - name para algum nome real (canonical)
  - value é o canonical name
- MX → value é o nome do mail server associado com o nome

### Protocolo DNS

• A query e a reply têm o mesmo formato

• Cabeçalho:

- 2 bytes de identificação
- " " de flags

identificação	flags
# questions	resposta RR,
# RRs autoritários	adicionais RRs
questions (variável)	
answers	
authority	
info adicional	

questions: nome e tipo da query

answers: RRs da resposta

authority: registros dos Auth... servers

Arquitectura P2P

- Não é sempre no servidor
- Peers estão conectados e trocam endereços IP

Ex:

- Distribuição de ficheiros
- Streaming
- Skype

Distribuição de FicheirosClient-Server

- Servidor envia sequencialmente N cópias do ficheiro
- Cada Cliente descarrega uma cópia

P2P

- Servidor carrega pelo -em cada uma cópia
- Cada cliente descarrega uma cópia
- Os clientes como agregado, descarregam NF Bittorrents

Bittorrent

- tracker → trata os peers que estão no torrent
- torrent → grupo de peers a trocar chunks de um ficheiro

Bittorrent

- Peer entra no torrent e não tem chunks, mas vai acumulando
- Usa o tracker para ter a lista de peers e conectar com os vizinhos
- Enquanto descarregam, upload para outros peers
- Peers saem e saem (Churn)
- Quando tem o file, pode sair ou ficar

**tit-for-tat**: quando um encontra um melhor

trocando partilha, é reciproco e ficar on de in no um do outro

**tcp-fair**  
melhor parceiro, transferências mais rápidas

## 2.6 CDNs

### Video Multimédia

**video**: sequência de imagens displayed numa taxa constante

**digital image**: array of pixels

**CBR**: Constant Bit Rate

↳ video encoding rate fixed

**VBR**: Variable Bit rate

### Streaming Multimédia

**DASH**: Dynamic Adaptive Streaming over HTTP

- server:

↳ divide o video file em chunks

↳ manifest file: provides URLs para os diferentes chunks

- client:

↳ consultando o manifest, pede um chunk de cada vez

↳ determina quando pedir o chunk, qual a encoding rate a pedir (qualidade) e onde pedir o chunk

↳ bemede na largura da banda

### CDN (Content Distribution Networks)

↳ Guardar múltiplas cópias do conteúdo nos sites geograficamente distribuídos

## Transporte

### Serviços e Protocolos

↳ originam comunicação lógica entre processos de Apps de hosts diferentes

→ Lado do envio: parte as mensagens da App em segmentos e dá à NL (Network Layer)

→ Lado do Destinatário: volta a montar a mensagem e dar à App.

### Connectionless - UDP

↳ não há handshaking entre o remetente e destinatário

↳ cada segmento é independente dos outros

↳ Utilizações:

- streaming
- DNS
- SNMP

### Reliable data transfer rdt

↳ Considera unidirecional data transfer

↳ mas o controle de flow é nas duas

### rdt 1.0

↳ canal subjacente perfeitamente reliable (confiável)

↳ no bits errors

↳ no loss of packets

### rdt 2.0

↳ canal subjacente pode causar bit errors

ACKs → o destinatário diz que o pacote tá Ok

NACKs → o destinatário diz que tem errors

o remetente reenvia o pacote

é melhor que rdt 1.0 porque:

- tem detecção de errors;
- mensagens de controle (ACKs e NACKs)

• Começa Aqui → rdt 2.1

↳ Quando ACK ou NACK estiverem corrompidos, manda novamente, podendo causar um duplicado

↳ O remetente adiciona um número de sequência a cada pacote

↳ O destinatário descarta pacotes duplicados → como msm nº de seq?

### rdt 2.1 \*

↳ O remetente adiciona o número ao pacote (0, 1)

↳ O Destinatário verifica se os pacotes recebidos são duplicados

## ndt 2.2

Mesmo que 2.1, mas só com ACKs.

O destinatário envia ACK do último pacote bem recebido.

Se o remetente recebe ACK duplicado, funciona com um NACK e reenvia.

## ndt 2.3.0 → Stop-And-Wait

canais subjacentes podem perder pacotes (ACKs e data)

O remetente espera um tempo razoável pelo ACK

reenvia passado o tempo; se o ACK apenas tava atrasado, o seq. # resolve

## Pipelined Protocols

remetente envia múltiplos pacotes to-be-acknowledged

go-back-N

selective repeat

Número mínimo de Seq. Numbers:

Stop-and-Wait →  $2 \text{ (0e1)}$

Selective Repeat →  $2N$

Go-back-N →  $N+1$

| N-Window Size |

Go-back-N \* envia todos a partir do ACKs

remetente envia N unacked packets

O destinatário envia ACK acumulativos

O remetente tem um timer para os unacked packets mais velhos

Quando ele acaba, reenvia todos os ACKs

## Selective Repeat

remetente tem N unacked packets

Destinatário envia ACKs individuais

Tem um timer para cada unacked packet; quando expira, reenvia individual!

## Formulário

$$T_p = \frac{D}{V_p}$$

→ Tempo de Propagação

↳ D é distância

↳ Vp é velocidade de propagação

$$T_{tx} = \frac{\text{Nº de Bits}}{\text{ritmo do canal}} \rightarrow \text{tempo de transmissão}$$

$$a = \frac{T_p}{T_{tx}} \rightarrow \text{nº de Tramas necessárias para encher o canal}$$

(uma direção)

$1+2a$  → em duas direções

BER - Bit Error Rate  
 $L$  - Nº de Bits

Calcular eficiência (U)

	Canal sem Erros	Canal com Erros	
Send and Wait	$U = \frac{1}{1+2a}$	$U = \frac{1-P_f}{1+2a}$	$P_f = 1 - (1 - BER)^L$ ↳ Probabilidade de envio
Selective Repeat	$U = \frac{N}{1+2a}$ , se $N < 1+2a$	$U = \frac{N(1-P_f)}{1+2a}$ $N < 1+2a$ $U = 1 - P_f$ $N \geq 1+2a$	
Go-Back-N	$U = 1$ , se $N \geq 1+2a$	$U = \frac{N(1-P_f)}{(1+2a)(1+P_f(N-1))}$ $N \geq 1+2a$ $U = \frac{1-P_f}{1+P_f(N-1)}$ $N > 1+2a$	

## Transfer Control Protocol

TCP

ponto-a-ponto

remetente e destinatário

reliável

Pipelined

conexão bi-direcional

Orientado a conexão

handshaking

## Entrada do segmento TCP

src e dest ports

Sequence Number

Acknowledgment Number

URG: urgent Data

ACK: ACK válido

PSH: push data now

RST, SYN, FIN, estabelecimento da conexão

Sq # → nº do 1º byte no

primeiro segmento

Ack → Seq # do próximo

byte a ser enviado pelo

outro lado

cumulativo

## Estabelecer Conexão

3-way hand shake

PC1

Server

SYN  
Seq=n

SYN+ACK  
Seq=k ACK=n+1

ACK;  
Seq=n+1  
ACK=k+1

## Terminar Conexão

FIN  
Seq=x

ACK  
ACK=x+1

FIN  
Seq=y

ACK  
ACK=y+1

## Reliable Data Transfer

use seq services

segmentos pipelined

acks cumulativos

single retransmission timer

ignorar duplicados

## Eventos do Remetente

↳ Criar segmento da data recebida pela App

↳ Começar contador/timer para cada segmento

↳ Retransmitir se houver timeout do ACK

↳ ACK da ACK dos anteriores

## Fast Retransmission

↳ Se o remetente recebe 3 ACKs duplicados (4 iguais), reenvia o segmento SEM ACK com menor # de seq.

↳ Nas espera pelo timeout

## RTT e timeout

RTT → round Trip Time

↳ tempo desde o envio até o recebimento do ACK

Sample RTT é ignorado na retransmissão

Estimated RTT =  $(1-\alpha) \text{Estimated RTT} + \alpha \text{ Sample RTT}$

tipico valor de  $\alpha = 0,125$

## Tempo de intervalo

↳ Estimated RTT mais uma margem de segurança

↳ Desvio do Sample e do Estimated

$$\text{Dev RTT} = (t - \beta) \text{ Dev RTT} +$$

→  $\beta \times \text{Sample} \rightarrow \text{Estimated}$

normalmente  $\beta = 0,25$

Timeout Interval =

$$= \text{Estimated} + 4 * \text{Dev RTT}$$

## Flow Control

↳ O receiver controla o sender, para o sender não sobrecarregar o buffer do receiver com muita informação muito rápida

↳ O receiver avisa quanto espaço livre tem o buffer no valor rwnd no TCP Header

↳ Remetente limita a quantidade de unacked data tendo em conta esse valor

↳ O buffer não vai sobre carregar

## Transporte

### Congestion Control

Muitas (m) fontes a enviar muito data muito rápido para a rede



### Congestion

Problemas:

perda de pacotes  
delays longos

### Idealizar

Sender apenas envia quando o buffer estiver disponível  
Sender apenas reenvia se sabe que o pacote foi perdido

### Realidade - duplicados

Pacotes são perdidos porque os buffers estão cheios  
Sender reenvia pacotes e os ambar entregues

### TCP Congestion Control

Sender aumenta a janela até ocorrerem perdas (limite), additive increase:

aumentar cwnd por  $\pm$  MSS a cada RTT até detectar perda  
multiplicative decrease:

diminuir cwnd (cada pela metade) após a perda

Notas: MSS - Maximum Segment Size  
RTT - Round Trip Time

cwnd - Congestion Window

rwnd - Receiver Window

### Slow Start

O ritmo inicial é lento mas começa a incrementar exponencialmente

### Deteção da Perda (loss)

Loss indicado pelo timeout

cwnd fica a  $\pm$  MSS

volta a slow start

Loss indicado por 3 duplicados (TCP RENO)

cwnd cortado a metade e aumenta linearmente

TCP Tahoe  $\rightarrow$  cwnd = 1

## UDP User Datagram Protocol

- ↳ connection-less
- ↳ cada segmento é independente dos outros
- ↳ é usado em streaming, DNS e SNMP.
- ↳ no congestion control

## UDP Checksum

- ↳ objetivo é detectar erros nos segmentos transmitidos

## Capítulo 4

### Network Layer

- Transporta o segmento entre hosts
- No lado do remetente, encapsula segmentos em datagramas
- No lado do destinatário, envia segmentos para a transp.
- Os protocolos estão em todos os hosts (routers)

### Funções

- Routing protocols
- IP Protocol
- ICMP Protocol

### IP

- addressing conventions
- formato de datagramas
- packet handling conventions

### Formato do Datagrama

20 bytes:

- versão (ver)
  - head. Len (bytes) ~ header
  - type of service (data)
  - length (bytes total)
    - identifier
    - flags
    - fragment offset
- } Para  
Fragmentos  
e  
Reassembly

- source e dest. addresses
- TTL
- Upper Layer (Protocol)
- checksum
- Data + options

### IP Addressing

- 32 bits de identificador do host
- interface: conexão entre host / router e ligação física

### CIDR - Classless Inter-Domain Routing

porção de subrede do address

• formato do endereço:

a.b.c.d/x

x - # de bits na porção de sub

High bits - Subnet part

Low bits - host part

### Subnet

- dispositivos têm a mesma parte de subnet no IP
- conseguem comunicar entre si sem necessidade de passar pelo router

## ICANN

- Internet Corporation for Assigned Names and Numbers
- ↳ guarda endereços
- ↳ manage DNS
- ↳ atribui nomes de domínios, etc

## DHCP

- Dynamic Host Configuration Protocol

↳ obter dinamicamente os addresses de um server

### Comunicação:

- host envia "DHCP discover"
- server DHCP responde "DHCP offer"
- host pede/aceita o IP "DHCP request"
- server DHCP envia IP "DHCPack"

### Envia também:

- endereço do primeiro router
- nome e IPs do servidor DNS
- network mask

## NAT

### Network address translation

- consegue alterar os IPs dos dispositivos locais sem influenciar o resto do mundo
- consegue mudar o ISP sem influenciar os dispositivos locais

### ISP: Internet Service Provider

- troca os source IPs do datagrama para o NAT IP, e os outros clientes viram o NAT IP apenas

- lembra na NAT table todos os tradusões de IP para NAT IP
- troca os datagramas recebidos para o NAT IP para o IP.

## IP Fragmentation e Reassembly

- Network Link's term MTU (Max transfer size)
- Um datagrama grande é fragmentado (volta a ser montado no destino)
- O header do IP é utilizado para identificar a ordem

## Capítulo 4.2

### ICMP

#### Internet Control Message Protocol

- Utilizado pelos hosts e routers para comunicar informações do nível da rede

↳ error reports; protocol; port

#### ICMP messages nos datagramas IP

##### ICMP Message

- type
- code
- 20 bytes do IP datagram causing error
- 8 bytes do transport header

### Traceroute

- Source envia séries de segmentos UDP para o destino

- Quando o datagrama  $n^{\text{th}}$  chega ao  $n^{\text{th}}$  router, o router deixa fora o datagrama e envia uma Mensagem ICMP à source (com ~~the reason~~, nome do router e o IP)

### IPv6

- 40 bytes fixos de headers
- fragmentação não é permitida

### IPv6 Datagram

pri - priority entre pacotes

flow label - identifica pacotes no mesmo "flow"

next header - identifica o protocolo da rede de cima

128 bytes de source & address  
16 " de destination "

### Outras mudanças para o IPv4

- checksum removido para reduzir tempo de processamento
- opções permitidas vêm fora do cabeçalho; indicado pelo cabeçalho "Next Header"

### ICMPv6

- novos tipos de mensagens (ex: "Packet Too Big")
- funções de gerenciamento de grupo multicast

## key words → network-layer functions

forwarding - mover pacotes do input de um router para o output apropriado

routing - determinar a route dos pacotes desde a source ao destino

## Data plane

- local
- per-router function
- determina como o pacote que entra no router é encaminhado
- forwarding function

## Control plane

- network-wide logic
- determina como o pacote é roteado entre routers, desde a source ao destino

traditional routing algorithms  
implementados nos routers

software-defined networking (SDN)  
implementado em servidores (switches)

- Per-route control plane
  - ↳ local forwarding table
- Logically centralized control plane
  - ↳ controlador remoto interage com os routers

## Router

processor de routing (software)

switching fabric (hardware)

forwarding table

opera em nanosegundos

routing; control plane

opera em milisegundos

## Input Port functions

- 1 - Line termination (Physical layer)
- 2 - Link layer protocol
- 3 - lookup, forwarding queueing

se os pacotes chegam mais rápido do que a tabela precisa

## destination-based forwarding

com base no IP (tradicional)

## generalized forwarding

com base em qualquer campo do header

## Longest Prefix Matching

### Destination Address Range

Destination Address Range	Interface
11001000 00010111 0001***	0
11001000 00010111 00011000	1
11001000 00010111 00011***	2

otherwise

Router Switching Fabrics

- transferir um pacote do input buffer até ao output buffer apropriado

Switching rate3 tipos:

- memória
- bus
- crossbar

Switching via Memory

- computadores tradicionais em que o CPU controlava isso
- pacote copiado para a memória
- velocidade limitada à largura de banda

Switching via a bus

- via bus partilhado
- bus contention: velocidade limitada à largura de banda do bus

Switching via interconnection networks

- supera as limitações do bus
- conectar processadores em multiprocessador
- fragmentar pacotes em células de tamanho fixo e trocar durante o trânsito

Input port queuing

- queuing delay e perda devido ao input do buffer sobrellenregado

- Head-of-the-line (HOL) blocking: o pacote na frente da queue pode que os outros avancem

Output Ports

- buffer de pacotes (queuing)
- link layer protocol
- line termination

- buffering necessário para ser mais rápido; pacotes podem ser perdidos devido a congestionamento
- scheduling datagramas

Queuing

- buffering quando a taxa de chegada é superior à de saída

média de buffering =

$$= \frac{RTT \times C}{\sqrt{N}} \rightarrow \text{capacidade} \\ \rightarrow N \text{ flows}$$

## Mecanismos de agendamento

- escolher o próximo pacote a enviar
- FIFO scheduling: enviar na ordem de chegada

## Capítulo 5.1

### Link Layer

- nodes - hosts e routers
- links - cunca de comunicacão que conectam nodes adjacentes
  - ↳ wired links
  - ↳ wireless links
  - ↳ LANs
- PDU da layer - Frame (Trama)
  - ↳ encapsula o datagrama

### Serviços

- ↳ framing
- ↳ encapsular Datagramas e adicionar headers
- ↳ NAC endereços utilizados
- ↳ transferência viável entre nodes adjacentes
- ↳ controlo de flow
- ↳ deteção de erros (pelo receiver)
- ↳ correção de erros (pelo receiver)
- ↳ half-duplex e full-duplex
  - ↳ dois nodes podem transmitir, mas não ao mesmo tempo.

### Implementações

- ↳ hosts, routers e switches
- ↳ adaptadores

### Deteção e Correção de Erros

↳ node é 100% viável

#### Parity check

- ↳ single bit (deteta 1 bit)
- ↳ two-dimensional bit (deteta e corrige 1 bit)

### Cyclic Redundancy check (CRC)

- deteção de erros mais poderosa
- usado na prática (Wi-Fi, ATM, Ethernet)

### Multiple Access Protocols

#### 2 tipos de links:

↳ Point-to-Point

↳ broadcast (partilhado)

↳

↳ unic播 broadcast partilhado

- pode haver múltiplas transmissões
- colisão se mais de um node recebe mais do que uma transmissão

### Protocolo

- algoritmo que determina como os nodes partilham o canal e quando podem transmitir

## MAC Protocols: taxonomy

### channel partitioning

- divide o canal em pequenas partes (time slots, ...)

### random access

- canal não dividido
- permite colisões
- recupera da colisão

### taking "turns"

- redução de turnos, mas os que têm mais a mandar demoram mais
- turnos maiores

## Channel Partitioning

### → TDMA (time division multiple access)

- acesso ao canal por turnos
- cada estação tem um slot de tempo fixo
- length = tempo de transmissão do pacote
- slots não usados ficam idle

### → FDMA (frequency division multiple access)

- espectro do canal dividido em bandas de frequência
- cada estação tem uma frequência

### Cica

- bandas não usadas ficam idle

## Random Access Protocols

- o nó mundo não sabe que tipos pacotes a mandar, sem coordenar com outros
- colisões
- deteta e recupera das colisões

### CSMA (carrier sense multiple access)

- ouve antena de transmitir
- adica se o canal estiver ocupado
- "não interromper o outro"
- colisões podem ocorrer por delay

### CSMA/CD (collision detection)

- colisões detectadas em porto tejo
- colisões fácil de detectar em wire, mas difícil em wireless
- "converrador educado"

CSMA/CD → Ethernet

.. / CA → 802.11

## Capítulo 5.2

54

### MAC Address

48

48 bits

- network-layer address para interface
- usado para forwarding (Layer 2)
- usado localmente entre interfaces conectadas fisicamente
- portatil, pode mudar de uma LAN para outra (OIP não é portatil)

### ARP (Address Resolution Protocol)

tabela:

- cada node ou LAN tem uma tabela
- mapeamento de endereços
- IP / MAC e TTL

TTL → tempo em q o mapeamento

vai ser esquecido (zummo normalmente)

"plug-and play" - seu interno  
deve dizer noq entra

### Ethernet

- LAN por cabo dominante
- single chip ; <sup>multiple</sup> veridida
- 10Mbps - 100 Gbps

bus → antigaente (cabo comum)

star → hoje em dia

↳ switch no centro dos PCs

- connectionless (não há handshaking)
- unreliable (new ACK e NACK)
- CSSA/CD com binary backoff

### Ethernet Frame

- preamble : 8 bytes com padrão sincronizar clock rates
- endereços : 6 bytes source/dst MAC
- Type : protocolo da camada 3
- crc : cyclic redundancy check

### Switch (Ethernet)

- store and forward frames
- seleciona e envia destino baseado no endereço MAC
- transparente para hosts
- flap não precisam de reconfiguração
- tem uma tabela com suas interfaces e MAC's
- aprende!
- se não sabe "flood" (inunda) para todos até saber
- podem ser conectadas a outras