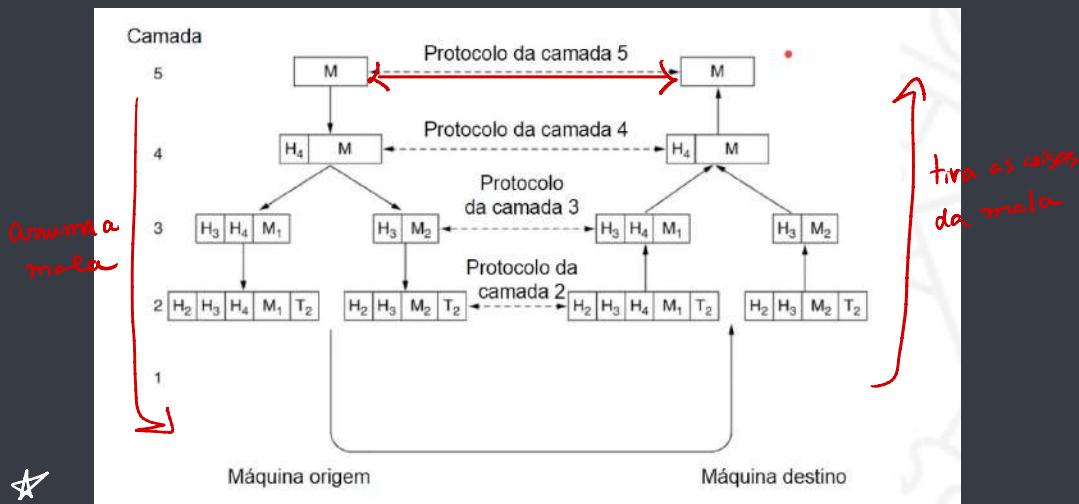


# Estudos Rudes I.

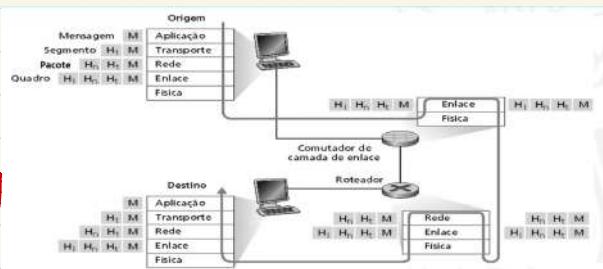
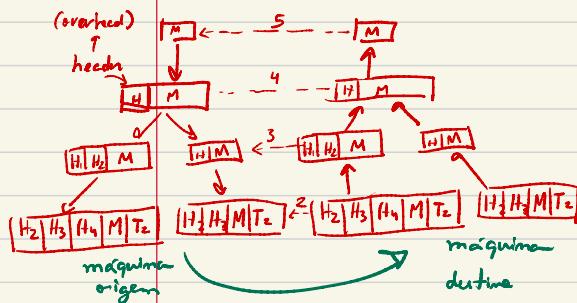


☆  
cada camada da pilha de camadas  
acegista que se comunica diretamente com seu proxima outra máquina

Sophia  
Carrazza

## Arquitetura TCP/IP

- baseada em uma pilha, em que cada camada é responsável por tarefas específicas.
- pilha de camadas → forma como as redes estão organizadas. Ela separa as tarefas, reduzindo complexidade do projeto.
  - horizontal
- comunicação virtual → entre a máquina de origem e destino, faz que elas achem que fazem comunicação direta entre si. É executada pelo protocolo da camada.
- comunicação real → a comunicação que realmente é feita, no nível + baixo de todos, entre entidades na mesma hierarquia através de um meio físico.



desta forma  
(TCP/IP veio antes  
do modelo OSI!!!)

## Modelos de Referência

- temos o OSI e o TCP/IP

módulo

arquitetura

OSI → interconexão de sistemas abertos (qualquer sistema que segue os padrões e cumpre de reinterectar)

→ possui 7 camadas: não é exatamente uma arquitetura, mas só explora o que deve ser feito em cada camada.

apresentação

rede e fechamento

- aplicação → comunicação entre pontos / processos
- internet → comunicação entre redes (roteamento)
- enlace → transforma pacotes em quadros
- física → fluxo de bits (envia e recebe)

flexíveis p/ suportar diferentes aplicações

**TCP/IP** → conjunto de protocolos que deveriam ter certas características (inicialmente p/ uso militar)

↳ removem algumas camadas de OSI:

fluxo de dados ← • aplicação ← → apresentação ← → negócio

segmento ← • transporte → TCP, UDP

pacote ← • internet rede → IPv4, IPv6, ICMP → controle e marcação de erros

frames/quadrados ← • enlace → Ethernet, WiFi, PPP

fluxo de bits ← • física → canais físicos tb não → transmissão de bits  
faltos na camada enlace → canal de bits

**Camadas** → (carga bits)

• física ⇒ transmissão de bits por meio do canal (enviam L, recebem L)  
↳ define interfaces elétricas, de sincronização, e outros.

• enlace ⇒ transmissão de dados entre vizinhos

↳ encadeamento = organiza os dados em unidades chamadas quadros

transporte → IP5 { ↳ controle de erros = detecta e corrige erros durante a transmissão

↳ controle de fluxo = controla a velocidade de transmissão de dados (evita que o transmissor envie dados + rápido que o receptor consegue processar)

• rede ⇒ roteamento de pacotes (pelo protocolo IP)

↳ ex: BH → ponto central PTT → ponto central Brasil → p.c. América → p.c. Asia → p.c. Japão → Tóquio  
no IPv4, cada endereço IP tem 32 bits e é representado em quatro octetos de 8 bits separados por um ponto. Ex: X<sub>1</sub>.X<sub>2</sub>.X<sub>3</sub>.X<sub>4</sub> → entre 0 e 255

↳ endereços 127.X<sub>2</sub>.X<sub>3</sub>.X<sub>4</sub> representam a máquina local (localhost)

• transporte ⇒ comunicação fim-a-fim entre processos (usando TCP/UDP)

↳ gerencia processos / serviços de comunicação na origem e destino.

↳ cada serviço disponível em uma máquina possui uma porta.

↳ prefixos do endereço de rede, número de IP e a porta de transporte

identifica a rede à qual

identifica cada

indica o processo / aplicação /

e dispositivo de destino

dispositivo (host)

que os dados pedidos

pertence.

na rede.

devem ser entregues

(ex: web, email, FTP, etc.)

\* O endereço de rede direciona o pacote até a rede correta.

\* O número de IP garante que o pacote chegue ao dispositivo certo dentro dessa rede.

\* A porta de transporte garante que, dentro desse dispositivo, os dados sejam entregues para o programa certo.

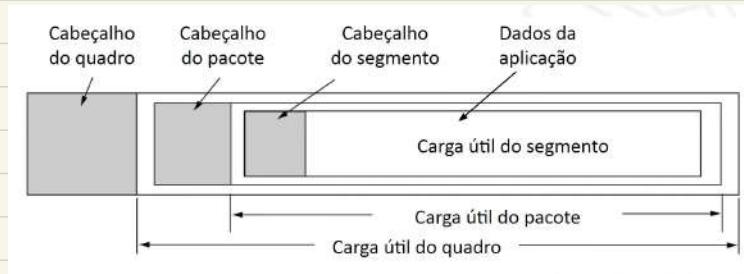
- redução → minimização, verificação, recuperação de troca de dados
- apresentação → representação de dados (criptografia, compactação, etc)
- aplicação → HTTP, email, FTP, áudio, vídeo e arquivos.

aplicação → apresentação → redução → transporte → rede → enlace → física

superiores: funcionários do serviço de transporte.      inferiores: prevedores de serviços de transporte.

### tipos de enlace:

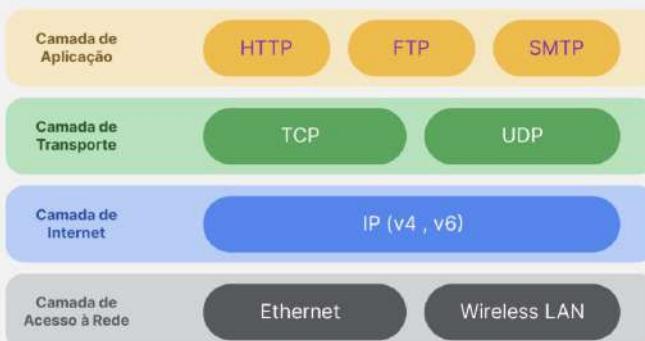
- processamento (CPU)
- enfileiramento (FIFO)
- serialização
- propagação física (lugares longe)



### > métricas de rede:

- latência = atraso / delay ou retraso (latência = melhor)
- bandwidth = largura da banda, qntd máxima de dados que pode ser transmitida em um canal. É propriedade física do canal
- throughput = n° de bits transmitidos por unidade de tempo
- custo, facilidade de instalação/mantenimento, taxa de erros = conceitos tradicionais

Modelo TCP/IP



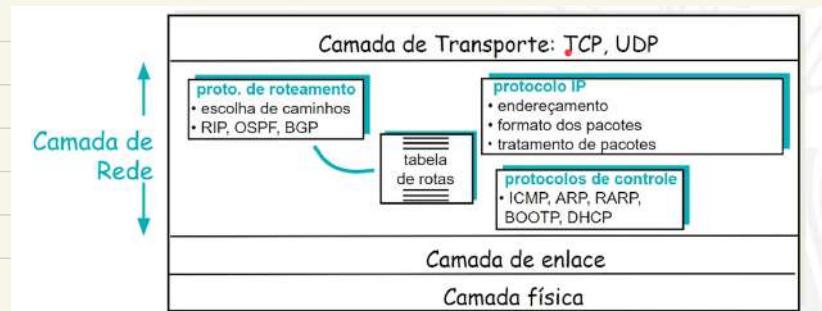
Modelo OSI



## Camada de Rede:

- ↳ transfere os pacotes da origem p/ o destino (onde tem + ex.)
- ↳ busca os melhores rotas e evita os perdes.
- ↳ durante o processo de roteamento, a camada + alta é a de rede, que é a camada que estará definindo essa rota.
- ↳ projeto da camada de rede:
  - computação de pacotes store-and-forward (quebra a mensagem em pacotes menores, salvando o pacote e quebrando em pedaços → cada nó do roteamento recebe, salva e segue o pacote p/ o próx. nó)
  - serviços oferecidos à Camada de Transporte (proporciona iluminação p/ origem e destino, fornecendo uma topologia de transporte que é independente à tecnologia de transporte, que para que a comunicação é direta).
  - serviços orientados (ou não) à conexão (se não for orientado à conexão, uma mensagem pode ser dividida em vários pacotes e cada um deles pode ser roteado por uma rota. / se for orientado à conexão, todos os pacotes seguem uma só rota, uma rede de circuitos virtuais).

Questão	Rede de datagramas	Rede de circuitos virtuais
Configuração de circuitos	Desnecessária	Obrigatória
Endereçamento	Cada pacote contém os endereços completos de origem e de destino	Cada pacote contém um pequeno número do circuito virtual
Informações sobre o estado	Os roteadores não armazenam informações sobre o estado das conexões	Cada circuito virtual requer espaço em tabelas de roteadores por conexão
Roteamento	Cada pacote é roteado independentemente	A rota é escolhida quando o circuito virtual é estabelecido: todos os pacotes seguem essa rota
Efeito de falhas no roteador	Nenhum, com exceção dos pacotes perdidos durante a falha	Todos os circuitos virtuais que tiverem passado pelo roteador que apresentou o defeito serão encerrados
Qualidade de serviço	Difícil	Fácil, se for possível alocar recursos suficientes com antecedência para cada circuito virtual
Controle de congestionamento	Difícil	Fácil, se for possível alocar recursos suficientes com antecedência para cada circuito virtual



\* A camada de redes pode ser vista como um conjunto de sub-redes ou SAs conectadas entre si.

## → Sistemas Autônomos

→ principios da canção de ninar?

- certifique-se de que funcionam
  - mantenha a simplicidade
  - faça escutas claras
  - explore a modularidade (leitura de código!) → focar no projeto
  - espere heterogeneidade → projeto
  - evite opções e parâmetros estáticos (pq é tudo muito dinâmico)
  - rigidez ao enviar mensagens e tolerância ao receber
  - escalabilidade

- custo benefício e desempenho  
→ não garante qualidade de serviço!

→ principal protocolo de rede que temos: Protocolo IP

- enderecamento (identificação dos nós)
  - formato dos pacotes
  - tratamento de pacotes

$$2^{16} = 65,535 \text{ bytes}$$

1

IP v4 → permite pacotes de até 64 KBytes

↳ elementos que mantêm a internet unida. Ela faz as interconexões de rede.

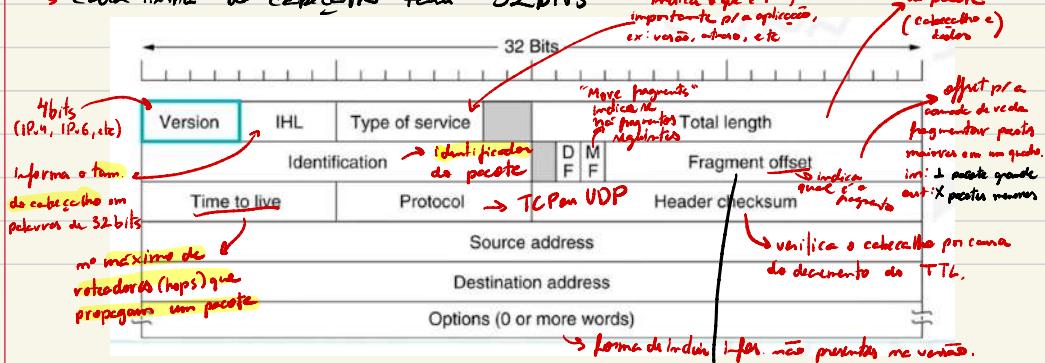
→ entrega os pacotes de origem p/ destino independentemente da ordem na encomenda ou não.

6 Entrega os pacotes c/ o maior eforço possível, mas não garante a menor entrega de distância, perde os pacotes, duplicações, entrega fora de ordem ou alterações.

Cabeçalho → parte fixa de 20 bytes

parte optional (también revisar)

↳ cada linha de cabeçalho tem 32 bits



↳ no fragment offset, a carga útil de cada fragmento é múltiplo de 8  
↳ (criado o último fragmento)

**MTU** → unidade máxima da transferência (quintal max. de dados que o protocolo consegue carregar (na camada de rede, e que também aplica a de rede)).

→ a camada de rede pregunta pacotes maiores que um quadro.

\* Fragmentação e remontagem não overhead!! → gate +

\* Cada roteador decrementa o TTL, limitando a vida útil do pacote (cada salto é múltiplo de 8).

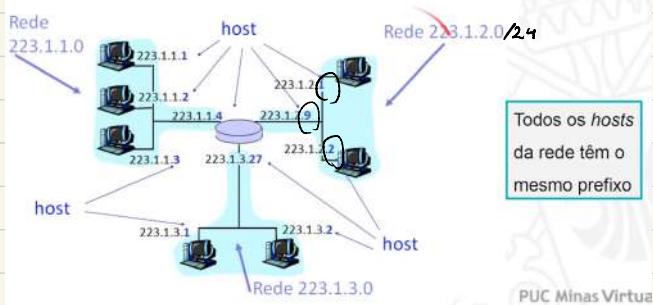
## Endereçamento

↳ endereço: número "único" de 32 bits que identifica cada máquina na Internet  
↳ em octetos

↳ na verdade, identifica cada interface de rede (Ex: no seu notebook tem a rede cabecada e a sem fio, cada uma c/ um endereço IP).

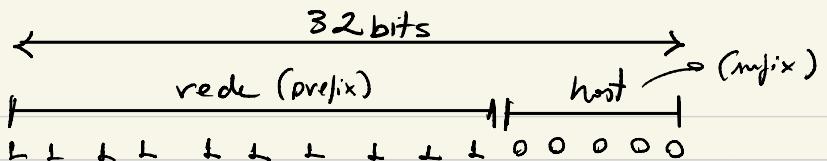
- ▶ Prefixo do endereço (tudo o que é antes) = identifica a rede que o host pertence (máscara de rede) e é definido globalmente
- ▶ Sufixo do endereço (tudo o que é depois) = identifica o host na rede, definido localmente pelo administrador da rede.

Exemplo com 24 Bits de Prefixo e 8 de Sufixo



Cada byte pode assumir  $2^8 = 256$  valores (entre 0 e 255)

△ **Máscara da Rede** = técnica para definir o tamanho do prefixo de rede. É um número de 32 bits, onde os bits de rede (prefixo) tem valor 1, e os de host, 0.



223.1.1.0/24

qu

223.L.I.O can  
meriae  
255, 255, 255. 0

- \* pode ser representada como:

  - barra (120)  $\rightarrow$  (20 bits de menor 1, restante 12 de zero)
  - em binário (LLL LLL LLL. LLL LLL LLL. LLL LLL LLL) 0000.0000.0000
  - em hexa : FF.FF.F0.00 (Octeto)
  - em decimal : 255.255.240.0

► Fazemos um AND bit a bit p/ definir o prefixo da rede

**ICANN** - corporação que controla os números de redes (IPs), evitando conflitos.

**Xrotcamento Hierárquico** → a tabela de rotas organiza rotas e prefixos, compactando todos os hots da unica rede rede.

↳ desvantagem → um endereço IP depende da sua localização

\* **sub-redes** → uma rede pode ser dividida em sub-redes, que podem ser divididas em + subredes, sem limite de n° de divisões.

- Se o roteador mais externo receber um pacote para o IP 128.208.2.151, esse pacote será encaminhado para qual sub-rede?
    - DCC: 128.208.2.151 and 255.255.128.0 (17 bits) = 128.208.0.0 que é diferente do prefixo do DCC (128.208.128.0)  

128.208.2.151  
and 255.255.128.0
    - DEE: 128.208.2.151 and 255.255.192.0 (18 bits) = 128.208.0.0 que corresponde ao prefixo do DEE  

128.208.2.151

**COLA**

0 [DEC]	=	0000.0000 [BIN]
128 [DEC]	=	1000.0000 [BIN]
2 [DEC]	=	0000.0001 [BIN]
192 [DEC]	=	1100.0000 [BIN]

128.208.2.151  
and 255.255.128.0

---

128.208.2.151  
and 255.255.192.0  
-----  
128.208.0.0

\* agrupação de votos → reduz o tamanho das tabelas de roteamento.

(takelad de rotel → nun rotelador fog una pergunta p/ cada pacote que ele encantaria).

## Considerações sobre as Tabelas de Rotas

- Um roteador faz uma pesquisa para cada pacote que ele encaminha
- Roteadores de núcleo podem encaminhar até milhões de pacotes por segundo, exigindo uma atenção especial dos fabricantes desses roteadores
- Os algoritmos de roteamento também podem exigir métricas especiais para cada caminho, aumentando o consumo de memória em cada roteador
- Soluções mais hierárquicas poderiam ser implantadas, contudo, isso demandaria IPs com mais de 32 bits

# Roteamento IP 4:

↳ Tabela de Roteamento = disponível em cada roteador

```

início
    se destino pertence à rede local então
        enviar o pacote diretamente para o destino
    senão
        para cada sub-rede cadastrada na tabela de roteamento faça
            faça um AND entre o IP do destino e a máscara da sub-rede
            se resultado igual ao prefixo da sub-rede então
                enviar o pacote para o próximo nó referente a sub-rede
            fim se
        fim para
    fim se
    se pacote não enviado então
        enviar o pacote para o default gateway (possui tabelas maiores)
    fim se
fim
  
```

→ fazendo um AND (IP destino, máscara rede/local)

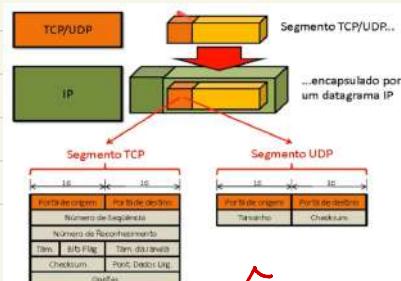
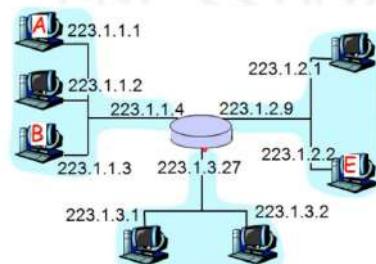
IP destino      vs      IP de prox. nó  
 aquele p/ quem deve ser entregue o pacote  
 (fica no cabeçalho de pacote)

aquele p/ quem o roteador deve encaminhar o pacote  
 (não está no cabeçalho)

exemplo

Suponha a rede abaixo em que o host A envia um pacote para B (mesma rede) e outro para E (outra rede)

Tabela de Roteamento de A		
Rede destino	Próx. Roteador	Núm. Saltos
223.1.1.0/24		1
223.1.2.0/24	223.1.1.4	2
223.1.3.0/24	223.1.1.4	2



## CAIXA NAT

- ↳ Os endereços IPv4 praticamente esgotaram
- solução = Network Address Translation (NAT)

- cada empresa tem apenas 1 endereço IP
- internamente, cada computador terá um IP exclusivo (fake, interno, inválido) dentro de uma faixa pré-definida.
- quando um pacote sai da empresa, ocorre uma conversão de endereços.

endereços de hosts:

Intervalos  
de endereços  
IPs privados

- 10.0.0.0 a 10.255.255.255/8 →  $2^{32-8} = 16.77.216$  hosts
- 172.16.0.0 a 172.31.255.255/12 →  $2^{32-12} = 1.048.576$  hosts
- 192.168.0.0 a 192.168.255.255/16 →  $2^{32-16} = 65.536$  hosts.

\* Quando um processo deseja estabelecer uma conexão TCP/UDP com outro remoto, ele se associa a uma porta TCP/UDP não usada.

\* Essa porta (porta de origem) informa para onde devem ser enviados os pacotes relativos a conexão (+ algumas informações).

> Como funciona o caixa NAT:

↳ Ele tem uma tabela de conversão entre porta de origem e IP privado (fake)

↳ quando um pacote de saída entra na caixa NAT:

- o IP privado da origem é substituído pelo da empresa
- o campo porta de origem é substituído por uma das entradas da tabela de conversão da caixa NAT.

↳ quando um pacote chega ao NAT vindo de ISP, o campo porta de destino do cabeçalho de TCP/UDP é extraído e usado como índice p/ a tabela de mapeamento da caixa NAT e que na origem vira destino, fazendo o pacote de volta.

Característica	IPv4	IPv6
Tamanho do Endereço	<u>32 bits</u>	<u>já ultrapassou</u> 128 bits
Espaço de Endereços	Aproximadamente <u>4,3 bilhões</u>	Aproximadamente <u>340 undecilhões</u>
Formato do Endereço	Numérico (e.g.,	<u>Alfanumérico</u> (e.g.,
Segurança	IPsec não é padrão	IPsec é padrão
Roteamento e Conectividade	Usa NAT para estender espaço de endereços	Conectividade direta de ponta a ponta <u>sem NAT</u>
Configuração	Configuração manual ou via DHCP	Configuração automática via ICMPv6 ou DHCPv6
Uso Atual	Predominante	Crescente

## IPv6

- Ver como solução p/ a escassez de endereços do IPv4
- tem 128 bits e não deve esperar outra escassez.
- (quanto o IPv4 tem 32bits)
- $2^{128}$  endereços  $\rightarrow 7 \cdot 10^{23}$  endereços por metro<sup>2</sup> de Terra

### Objetivos de IETF (1990) p/ o sucessor do IPv4:

- ter bilhões de hosts mas com alcance insuficiente
- reduzir o tamanho das tabelas de roteamento
- simplificar o protocolo p/ reduzir o tempo de processamento
- mais segurança
- auxiliar multicasting
- permitir a evolução do protocolo e a coexistência entre protocolos de importância os tipos de serviços.

→ no IPv4 tem 14 campos, no IPv6 não

### Cabeçalho do IPv6 (simplificado, reduzindo o processamento nos roteadores).

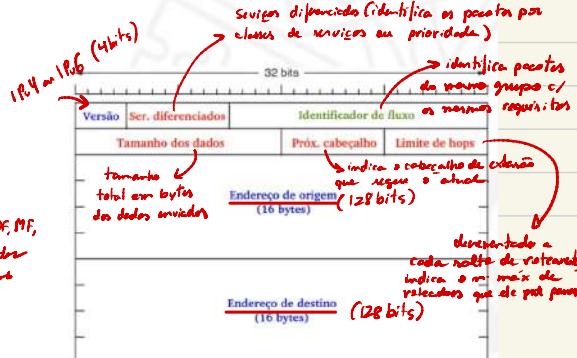
#### • Campos similares aos do IPv4

#### • Campos renomeados ou repositionados

#### • Novo campo

#### • Campos removidos

#### • Cabeçalhos de extensão

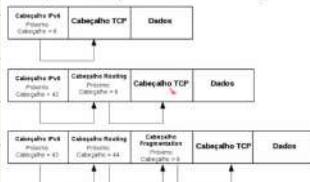


IHL → descrevendo pq o cabeçalho tem tam. fixo  
Id, DF, MF, Fragment Offset → migrados pr o cabeçalho de extensão de fragmentação  
Header Checksum → descartado pq outras camadas tb verificam o novo  
Options → verificado pelos cabeçalhos de extensão

## > Cabeçalhos de Extensão:

- ↳ permitem info. extras e úteis (facultativos)
- ↳ são 6 tipos:
  - hop-by-hop options (decisões de roteamento)
  - routing (suporte a mobilidade)
  - fragmentation (campos de fragmentação similares ao IPv4)
  - authentication header (segurança)
  - encapsulating security payload
  - destination options (tb suporte a mobilidade)

Localizam-se entre o cabeçalho base e da camada superior. Se tivermos múltiplos cabeçalhos de extensão, eles formam uma cadeia de cabeçalhos



## > Fragmentação no IPv6:

- ↳ roteadores intermediários não fazem fragmentações
- ↳ protocolo path MTU Discovery (PMTUD) descobre o MTU da rota.
- ↳ quando um roteador descobre uma rota com MTU maior que é considerada, ele envia um ICMPv6 Packet too Big pr o origem.

## > Endereçamento:

- ↳ número único de 128 bits que identifica cada máquina na internet ( $2^{128}$  endereços)
- ↳ cada hexadecateto tem = 16 bits, 2 bytes ou 4 dígitos hexadecimais

→ convenções de abreviação:

- omitir os zeros à virgula da cada octeto ( $0123 \rightarrow 123$ )
- substituir um ou + hexadecacifetos formados por zero por ":"  
( $00FF:0000:0000:0000:0000:0000:000F \rightarrow FF::F$ )
- endereços IPv4 podem ser escritos após um ":" ( $::192.31.20.46$ ).

→ representação de endereços IPv6 em URIs:

- entre colchetes, evitando ambiguidade e eventuais indicações de portas
- por ex: `http://[2001.12ff.0:4.22]/index.html`

→ endereços multicast → identificam um grupo de nós (interfaces) sendo que um nó pode pertencer a qualquer número de grupos multicast.

8 bits	4 bits	4 bits	112 bits
L L	FLAGS	SCOP	GROUP ID

- identifica o prefixo
- 3 bits da "L" → limita o escopo
- 4 bits de SCOP → de grupo da rede (Terra, galáxia, etc)
- 1 bit de direta
- 0 "L" → é endereço end. privado
- 1 "L" → é endereço provisório

→ endereços anycast → endereço atribuído a mais de um nó. Um pacote destinado a esse endereço é entregue ao nó + próximo, de acordo com os protocolos de roteamento.

→ prefixes de rede → padrão /64.

→ economia e redução do processamento no roteamento

↳ endereço de loop back

- no IPv4 → faixa 127.0.0.0/8 (normalmente, 127.0.0.1)
- no IPv6 → ::1.

> Tunelamento:

↳ os IPv6s podem rotear pacotes IPv4, mas o contrário não é verdade.

↳ assim, como a IPv4 pode rotear pacotes IPv6?

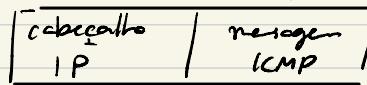
- Flag day (todos migrarem pro IPv6) é impossível
- a solução é o tunelamento

↳ O Tunelamento encapsula pacotes de uma versão do IP dentro de outra, permitindo que redes IPv6 se comuniquem por meio de IPv4 e viceversa (6in4 ou 4in6)

# Protocolos de Controle (do IPv4)

## ICMP → Internet Control Message Protocol

- ↳ reporta eventos inesperados nos roteadores.
- ↳ quando o roteador detecta um problema, ele envia um pacote ICMP p/ a origem do pacote reportando o problema.



valor inválido  
detectado  
se um dado  
pacote é recebido.

- ↳ Destination Unreachable, Time exceeded, Parameter Problem, Echo, etc
- ↳ traceroute → mostra o trajeto "atual" percorrido por um dado pacote IP.  
↳ origem envia pacotes c/ TTL=1 e a cada vez que um pacote "mexe" e retorna um envia, sabemos mais um nó da rota.

→ endereço físico = da placa

## ARP → Address Resolution Protocol

- ↳ endereço físico, ethernet ou MAC. Tem 6 bytes (48 bits) representados em hexadecimal e separados por : ou -.
- ↳ temos  $2,8 \cdot 10^{12}$  endereços físicos.
- ↳ o protocolo ARP faz o mapeamento de endereço IP em Físico de enlace (p/ a camada de enlace desconhece endereços IPs).

- Quando um nó deseja enviar um pacote e desconhece o endereço físico do próximo nó, ele envia uma mensagem ARP de difusão perguntando qual é o endereço físico do próximo nó
- Todos os nós da rede recebem a mensagem e a respondem informando seus respectivos endereços físicos
- Definido na RFC 826

## DHCP → o servidor DHCP entrega o IP p/ uso da máquina

- ↳ quando um nó entra na rede, ele faz a difusão de um pacote DHCP DISCOVER.

- ↳ o servidor DHCP responde c/ um DHCP OFFER contendo um endereço de IP livre e outras info.

- ↳ nó confirma c/ um DHCP REQUEST

- ↳ servidor DHCP confirma a alocação com DHCP ACK.

relacionado às portas de processos / de aplicações

## Camada de Transporte

- > mensagem → catálogo do quadro, do pacote e do segmento + <sup>carga</sup> útil
- > porta → número que identifica um processo de transporte → 16 bits (web, telnet, https, etc)

## Camada de Rede vs. de Transporte

- Responsáveis pelo **transporte de dados** entre origem e destino
- Contudo, têm preocupações distintas:
  - executa o transporte fina/fim usando datagramas ou circuitos virtuais
  - usa os caminhos físicos, através de roteamento
  - Camada de rede: **roteamento dos dados**
  - Camada de transporte: comunicação entre **processos** que **executam** na origem e destino. Tais processos executam os **protocolos de transporte**

Camada de Rede	Camada de Transporte
Transferência de dados entre sistemas finais	Transferência de dados entre processos
Funciona principalmente nos <u>roteadores</u>	Funciona <u>inteiramente</u> nas <u>máquinas dos usuários</u>
Identificam as máquinas nas redes (e.g., número IP)	Identificam os processos nas máquinas (e.g., porta)

	Rede orientada à conexão	Rede não orientada à conexão
Camada de Transporte orientado à conexão	✓	Exemplo: TCP/IP
Camada de Transporte não orientado à conexão	Combinação normalmente inexistente porque estabeleceria uma conexão para enviar um único pacote	Exemplo: UDP/IP

origem  
 multiplexação → reúne os dados provenientes de diferentes processos  
 demultiplexação → entrega os dados contidos em um segmento da camada de transporte ao processo de aplicação correto (destino)

**Protocolo UDP** → bom p/ dados simples, pequenos ou em tempo real (resposta ao ato)

- ↳ protocolo de transporte não confiável
- ↳ não há conexão → não há orientação entre o UDP emissor e o receptor.
- ↳ código de erro UDP é tratado de forma independente dos outros (uma pergunta, uma resposta).

→ Faz o serviço best-effort, não garantindo perdidos e fora de ordem.

→ Cabeçalho:



→ Funcionamento:

- cliente envia uma solicitação para o servidor
- cliente aguarda por uma resposta do servidor
- servidor recebe o pedido e devolve a resposta
- se o cliente não recebe a resposta, recebe timeout

→ não realiza controle de fluxo, controle de erros, retransmissão após segmento incorreto, controle de congestionamento, etc

→ bom para servidores de arquivo remoto, recepção de multimídia, protocolo de votação, telefonia por internet, etc.

Protocolo TCP → bom p/ dados grandes e muito importantes, sensível a perda de mensagens.

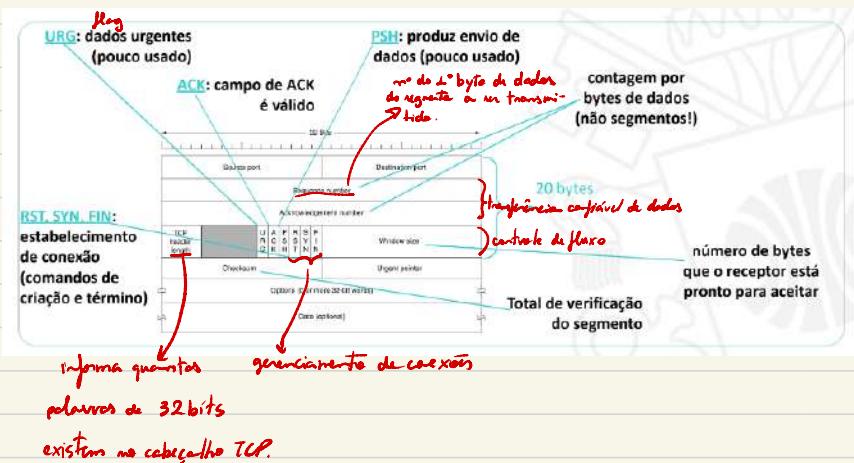
→ fornece um fluxo de bytes fim a fim confiável em uma rede nem sempre confiável

→ etapas do TCP:

- 1- entidade TCP emissor recebe o fluxo de dados da aplicação e o divide em segmentos
- 2- entidade TCP emissor envia cada segmento p/ a camada de rede e usa efeitos sua tarefa
- 3- ao receber os segmentos, a entidade TCP receptora restaura o fluxo de dados original

**soquetes** → permite a comunicação entre processos  
 → identificação com IP + nº de porta.

**Cabeçalho:** → 20 bytes de cabeçalho



\* Normalmente, um segmento TCP tem 1460 Bytes (parte útil + cabeçalho), fazendo com que cada segmento caiba em um único quadro Ethernet.

> **Confiabilidade:**

> O TCP provê garantia de entrega de dados confiável p/ os aplicativos  
 > trata pedos e atrasos sem sobrecarregar a rede e seus roteadores  
 > garante a ordem de bytes recebida no destino

\* Esta transferência confiável de dados é baseada em confirmações usando a flag ACK (Acknowlegment).

- 1- emissor envia um segmento e dispara um temporizador.
- 2- receptor recebe o segmento
- 3- receptor envia um segmento ACK (confirmação) (para pelos roteadores, cerca física, etc, etc chegar no transporte de receptor)
- 4- quando o emissor recebe o ACK, cancela o temporizador
- 5- se o temporizador expira e o emissor não receber o ACK, ele retransmite o segmento.

## > Conexão de TCP:

- ↳ não full duplex (tráfego pode ser feito em ambos direções ao mesmo tempo). (não tem diferença)
- ↳ não ponto a ponto (cada conexão tem 2 pontos terminais).
- ↳ as conexões não estabelecidas a partir do 3-way handshake (3 vias) ? SYN = synchronize, ou seja, indicando que de forma estabelecer uma conexão
- 1- cliente envia um segmento  $SYN=1 \wedge ACK=0$  ao servidor p/ informar o número inicial da seqüência
- 2- servidor responde c/ o segmento SYNACK ( $SYN=1, ACK=1$ ), receatendendo o SYN recebido, alocando buffers e especificando o nº inicial da sua seqüência (indica: recebi o SYN(x) e a próxima é o ACK(x+1)).
- 3- cliente recebe o SYNACK e envia um ACK.
- ↳ p/ finalizar a conexão TCP:  
1- o host 1 envia um segmento TCP FIN p/ o host 2 (flag)  
2- o host 2, recebe o FIN e responde c/ ACK, e fecha a conexão sentido host 1  $\Rightarrow$  host 2, mas o fluxo pode continuar no sentido oposto  
3- host 1 recebe o ACK e fecha o fluxo contrário.

## > Controle de Fluxo → capacidade do receptor

- ↳ evita que o emissor exoplete a capacidade do receptor
- ↳ usa o window size do cabeçalho TCP
- ↳ no estabelecimento da conexão, cada entidade TCP aloca um buffer de receção e informa o tamanho desse buffer p/ outra entidade. Em toda comunicação, enviar-se o espaço disponível nesse buffer (chamado janela)

## > Controle de Congestionamento → capacidade da rede

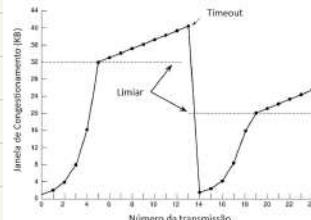
- ↳ quando a carga oferecida a qualquer rede é maior que sua capacidade = congestionamento.
- ↳ diferente de controle de fluxo! (o congestionamento tem a ver com a rede)

↳ logo, temos uma janela p/ o receptor (que faz o controle de fluxo), e uma p/ a rede, que é a janela de congestionamento.

↳ o mº de bytes que eu posso transmitir é o valor mínimo entre as duas janelas!



- **Um limiar é atingido:** as transmissões bem-sucedidas proporcionam um crescimento linear à janela de congestionamento
- **Acontece um timeout indicando a retransmissão de um segmento:**
  - O limiar é definido como a metade da janela de congestionamento atual
  - Janela de congestionamento ⇒ segmento máximo
  - Inicialização lenta é usada
- **Janela de recepção é alcançada:** janela de congestionamento pára de crescer e permanece constante



A transmissão de mensagens é feita de forma exponencial até atingir um limiar, quando ela passa a aumentar mais lentamente

## > Gerenciamento de Temporizadores

- ↳ RTT → Round Trip Time (tempo de viagem ida e volta, desde o envio do segmento até ele ser reconhecido).
- ↳ p/ cada conexão, o TCP tem um RTT, uma estimativa de tempo necessário → logo, o timer deve ser maior que o RTT!

$$RTT = \alpha RTT + (1-\alpha)M$$

fator de suavização ↗ último RTT medido

**Camada de Transporte (TCP):** O TCP é responsável por dividir os dados em pacotes, numera-los e garantir que eles sejam entregues corretamente ao destino. Ele estabelece uma **conexão confiável** entre o remetente e o destinatário.



**Camada de Rede (IP):** Os pacotes do TCP são então passados para a camada de rede, onde o IP adiciona informações de endereçamento, como o endereço IP de origem e destino. Isso permite que os pacotes sejam roteados através da rede até o destino correto.

## Camada de Aplicações

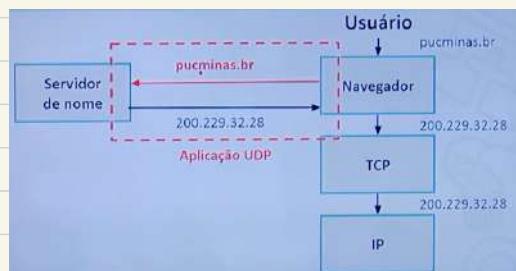
**Modelo OSI:** a camada de Serviços permite que máquinas se estableçam renas de comunicação entre elas.

### Aplicações:

- **DNS** = mapeamento de nome de máquinas
- **FTP** = transferência de arquivos
- **e-mail** = transferência de mensagens
- **web** = docs. em hipermédia

**DNS:** mapeamento entre nome e endereço, e vice-versa.  
mais ↗  
UDP ↘  
variável ↙  
fixo ↘

- ↳ Há um banco de dados distribuído responsável pelo mapeamento entre nomes e IPs.
- ↳ Busca de nomes hierárquico, baseado em domínio.



- ↳ O endereçamento é dividido em domínios, que podem ser divididos em sub-domínios, etc.
- cada domínio controla a colocação de seus subdomínios
- ex: .br, .pucminas, .edu, .icei, .gov, .com, etc.

↳ Conjunto de registros de recursos:

<Name, TTL, Class, Type, Value>

É implementada de forma recursiva e não recursiva.

## funcionamento:

- **Passo 1:** Uma aplicação chama o procedimento resolver passando como parâmetro o nome do computador desejado.
- **Passo 2:** Procedimento resolver envia uma mensagem para o servidor DNS local com nome desejado e esse retorna o devido endereço IP
- **Passo 3:** Procedimento resolver retorna o IP para a aplicação

>DNS dinâmico → o servidor DNS dinâmico armazena nome e IP atual, e sempre que é atualizada ele informa ao servidor de DNS

## FTP → File Transfer Protocol

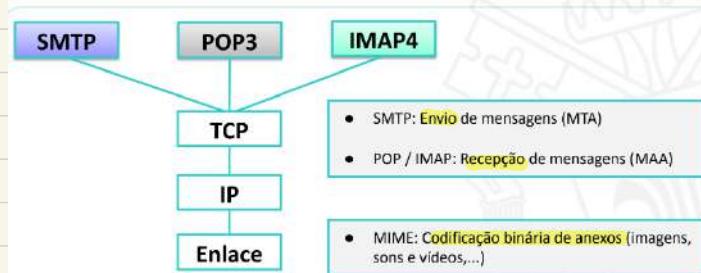
- ↳ faz a transferência de arquivos.
- ↳ usa cliente-servidor
- ↳ senha e usuário em anexo
- ↳ uma nova conexão de dados é criada p/ cada envio de dados.
- ↳ servidor adota a porta **TCP 20** p/ transferência de dados e **21** p/ controle
- ↳ p/ enviar comando e回报答

## Correio Eletrônico

- ↳ comunicação assíncrona
- ↳ Spams → lixo, técnica eficiente

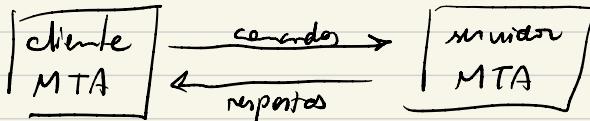
code envio  
de arquivos  
é uma nova  
conexão.

### Protocolos do Correio Eletrônico na Internet

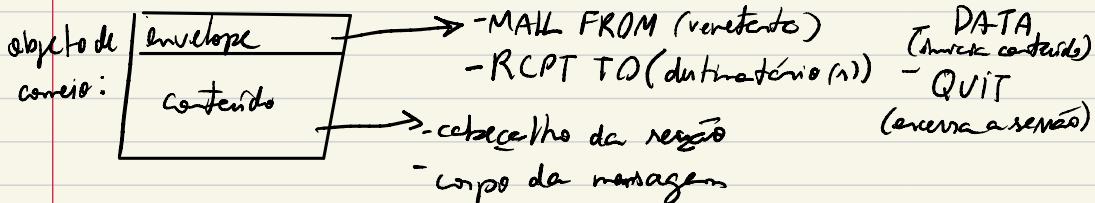


## Protocolo SMTP: (Simple Mail Transfer Protocol)

↳ transfere mensagens do remetente para seu servidor correio e entre os servidores de correio do destinatário.



↳ servidores email solicita endereço IP do registro MX do domínio (diferente do DNS padrão). Ele tem prioridades.



### MIME = Multipurpose Internet Mail Extensions

↳ converte dados não ASCII enviados pelo usuário, para dados ASCII para serem transmitidos pelo SMTP

↳ ele não completa o SMTP, não sendo responsável pela transferência dos dados.

### POP = Post Office Protocol

- ↳ protocolo que faz o "pull" para que o UA destino puxa os emails do seu servidor
- baixa mensagens armazenadas na mailbox do usuário
  - mensagens não baixadas são apagadas do servidor. Podem ser armazenadas somente na máquina local

### IMAP=

- ↳ também faz o pull p/ puxar os emails do servidor
- múltiplos usuários
  - mensagens de mensagens e pacotes p/ organização
  - salva os mensagens no servidor, podendo ser manipulados, excluídos, etc

## WWW - World Wide Web

↳ vasta coleção de páginas interligadas por links e acessadas via protocolo HTTP.

↳ W3C → padroniza protocolos e incentiva a interoperabilidade.

### 3. Funcionamento do Acesso Web

- O usuário solicita uma URL no navegador.
- O navegador consulta o DNS para obter o IP do servidor.
- Estabelece uma conexão TCP com o servidor web.
- Realiza requisições HTTP e recebe respostas, podendo solicitar objetos adicionais.
- O navegador exibe a página e encerra a conexão.

### 4. Tarefas do Servidor Web

- Aceitar conexões TCP, localizar e enviar arquivos, encerrar conexões.
- Utilizam cache, multithread e proxy reverso para otimizar desempenho.

## Protocolo HTTP => Hypertext Transfer Protocol

### Funcionamento do HTTP

1. **Requisição:** O cliente envia uma requisição HTTP ao servidor, especificando o método (como GET ou POST) e o recurso desejado 2 3.
2. **Processamento:** O servidor processa a requisição e envia uma resposta, que inclui o recurso solicitado e informações adicionais no cabeçalho 2 3.
3. **Conexão:** A comunicação ocorre via TCP/IP, garantindo uma conexão confiável 1 4.

### Características do HTTP

- **Stateless:** Cada requisição é independente, sem memória de transações anteriores 1 4.
- **Baseado em Texto:** As mensagens são em formato de texto, facilitando a leitura e o processamento 1 4.
- **Camada de Aplicação:** Atua na camada 7 do modelo OSI, responsável por interações entre aplicações 3 5.

### Mensagens HTTP

As mensagens consistem em uma linha inicial, cabeçalhos e um corpo opcional. Os cabeçalhos transmitem informações adicionais entre cliente e servidor 5.