TCP & UDP

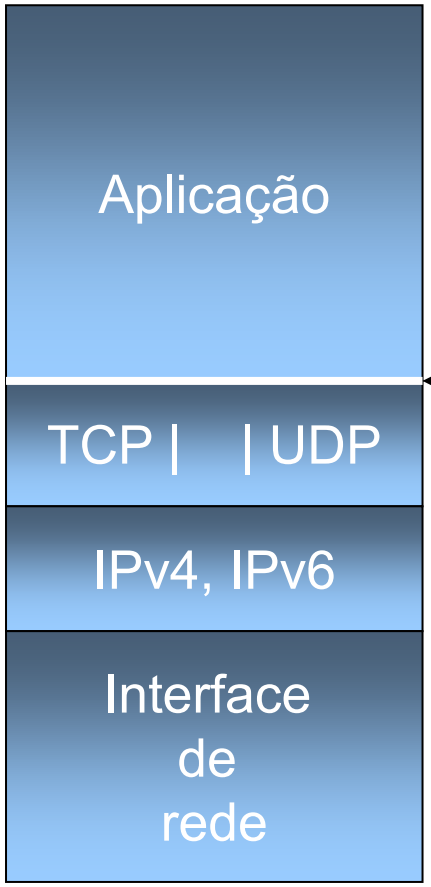
Programação Avançada

Patricio Domingues

PILHA PROTOCOLAR TCP/IP

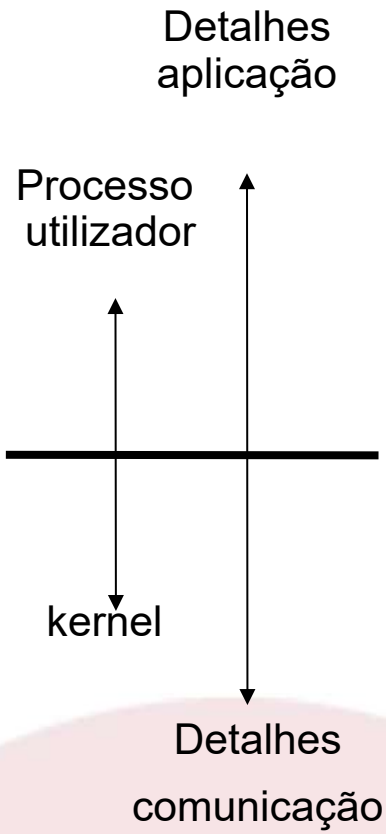
Modelo protocolar TCP/IP vs. OSI

Modelo OSI



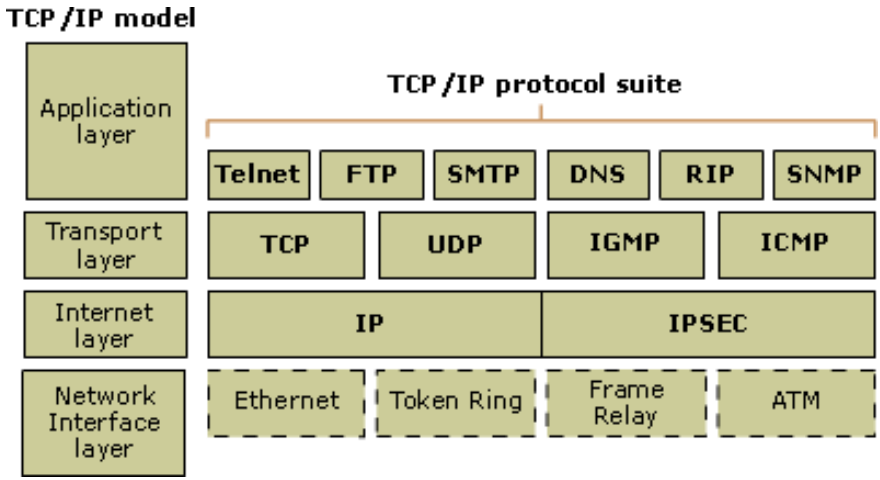
TCP/IP

Sockets

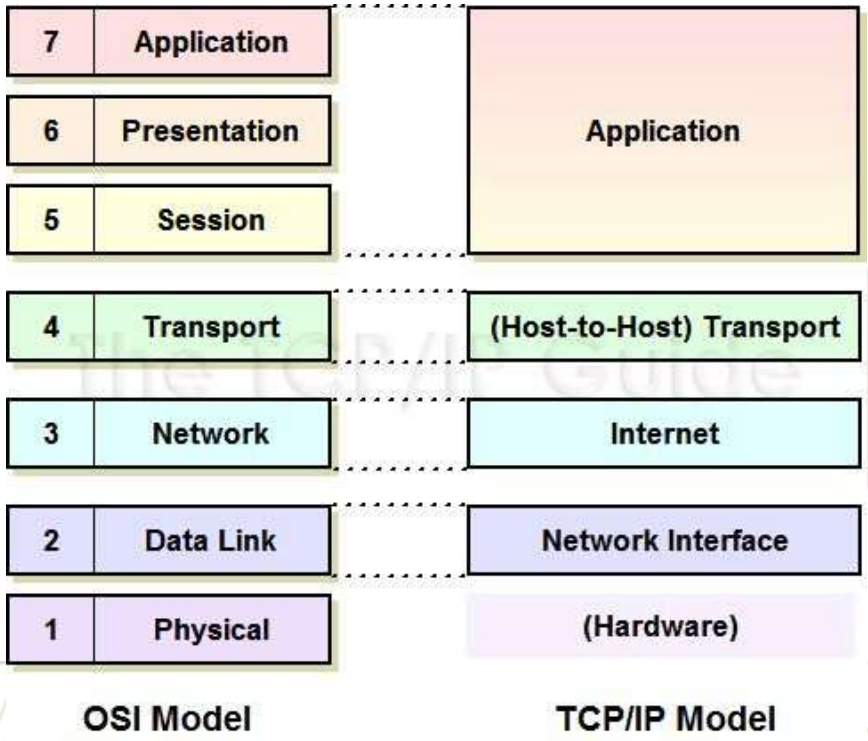


Camada protocolar TCP/IP

- Camada "aplicação"
  - Suporta as aplicações do nível do utilizador
    - mail, ftp, http (www), etc.
  - Providencia comunicação entre processos/aplicações em máquinas diferentes
- Camada "transporte"
  - Providencia transferência ponto-a-ponto com suporte de multiplexagem (conceito de porto)
    - TCP, UDP
  - Providencia alguma confiabilidade
- Camada "internet"
  - Camada responsável pelo endereçamento (IP)
  - Providencia capacidades de endereçamento e encaminhamento através de várias redes (IP ou ARP)
- Camada "acesso à rede"
  - Associada à ligação física entre elementos comunicantes
  - Não é definido pelo TCP/IP: depende da tecnologia de rede

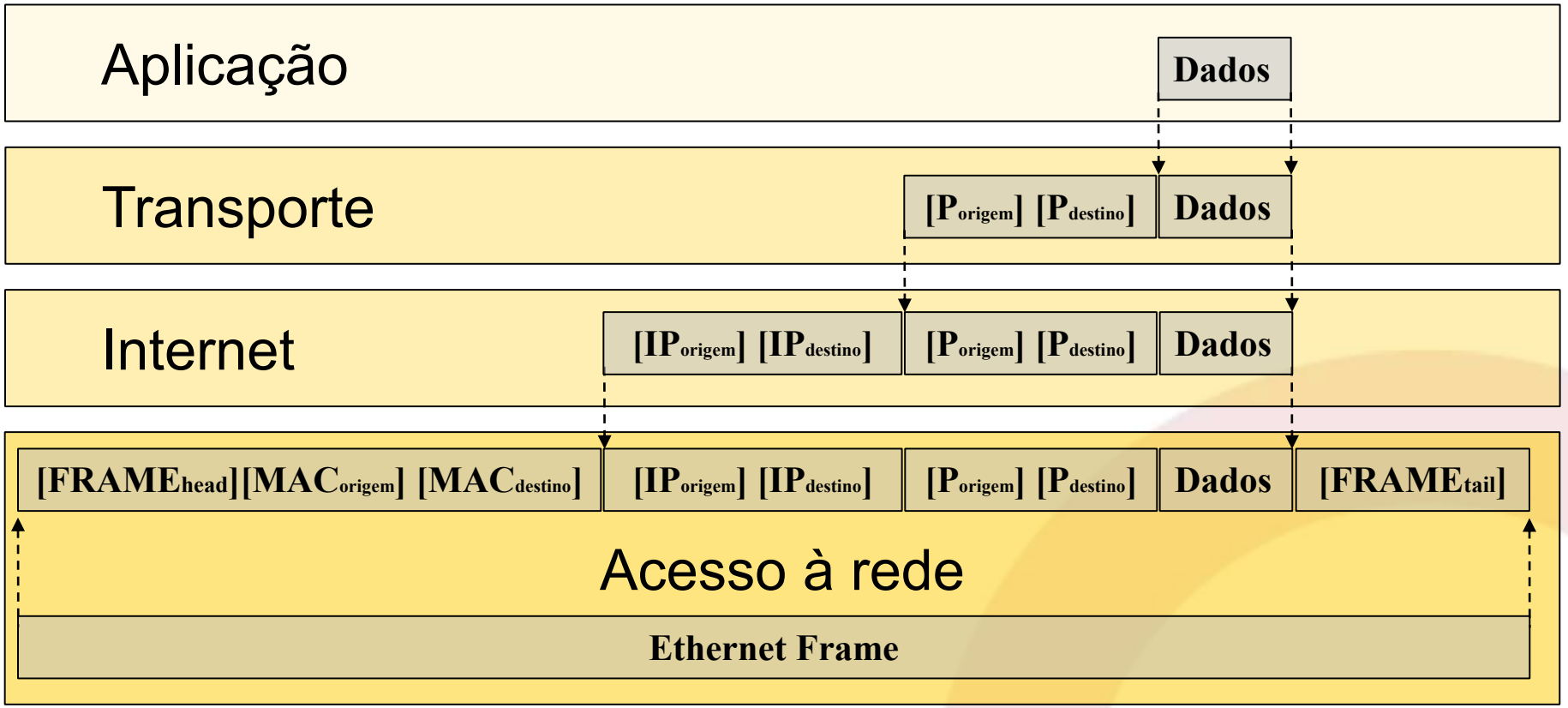


http://i.technet.microsoft.com/dynimg/IC197700.gif



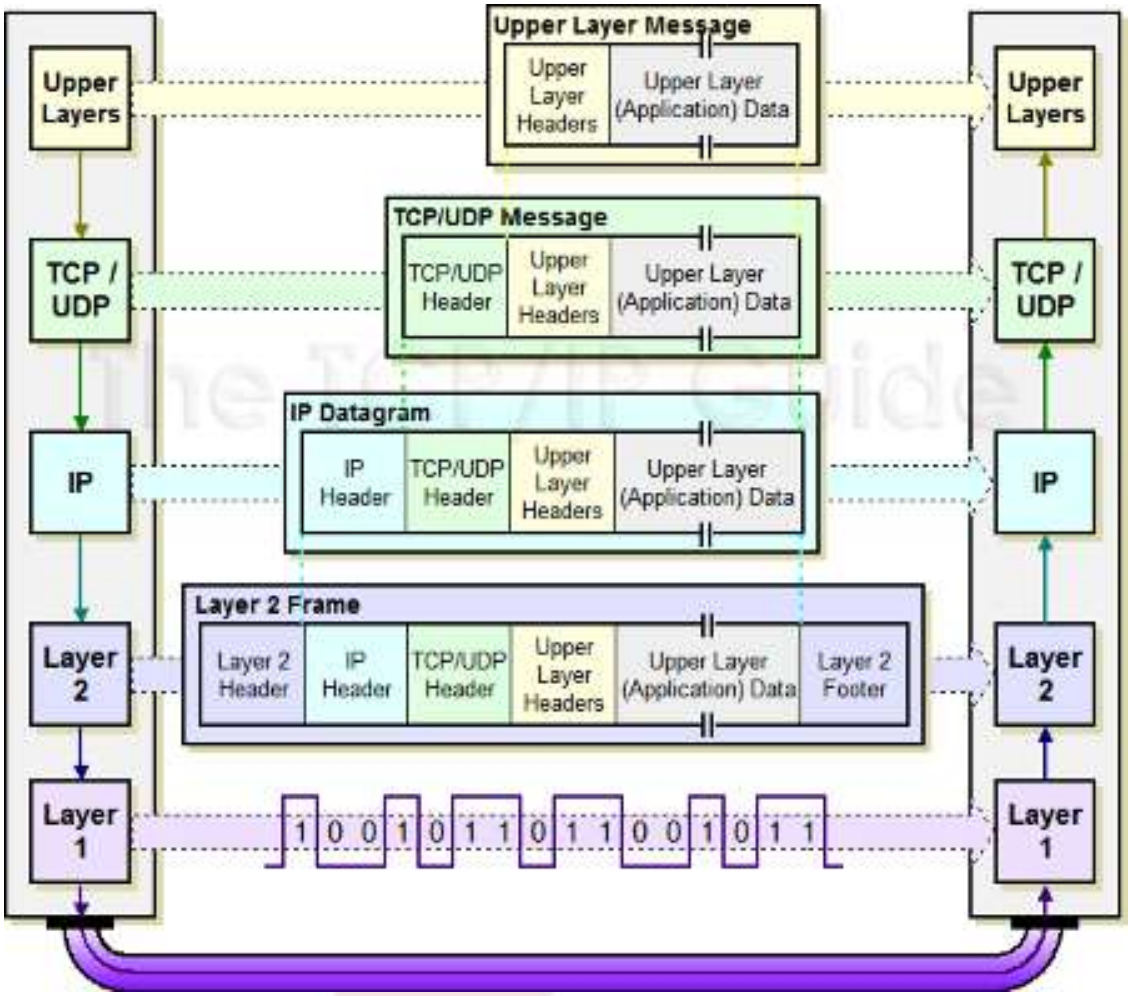
Pilha protocolar TCP/IP

- Construção de mensagens
  - À medida que uma mensagem desce pelas várias camadas da pilha protocolar TCP/IP, são acrescentados cabeçalhos (um por cada camada) à mensagem



As camadas protocolares

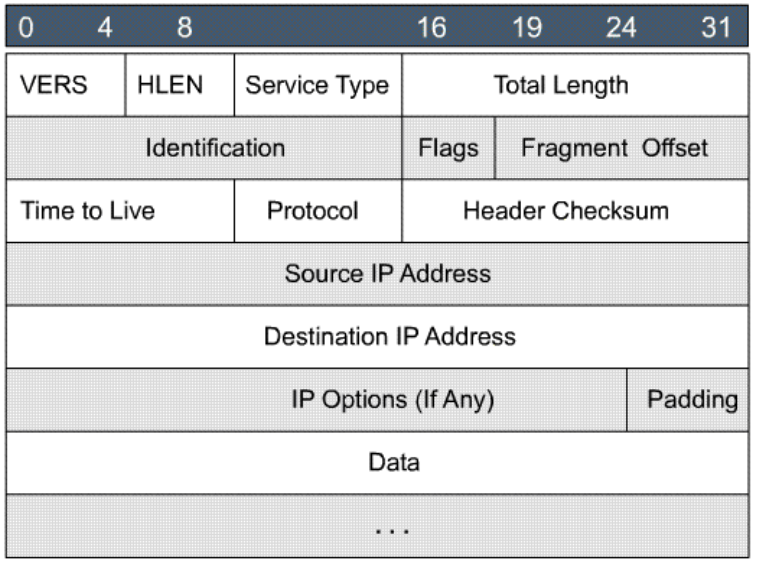
- Uma mensagem enviada por uma aplicação desce pelas camadas do emissor, subindo no recetor até ser entregue à aplicação
  - Conceito de pilha protocolar
  - Para a aplicação, a comunicação ocorre na camada da aplicação interagindo apenas com a camada de transporte (TCP/UDP)



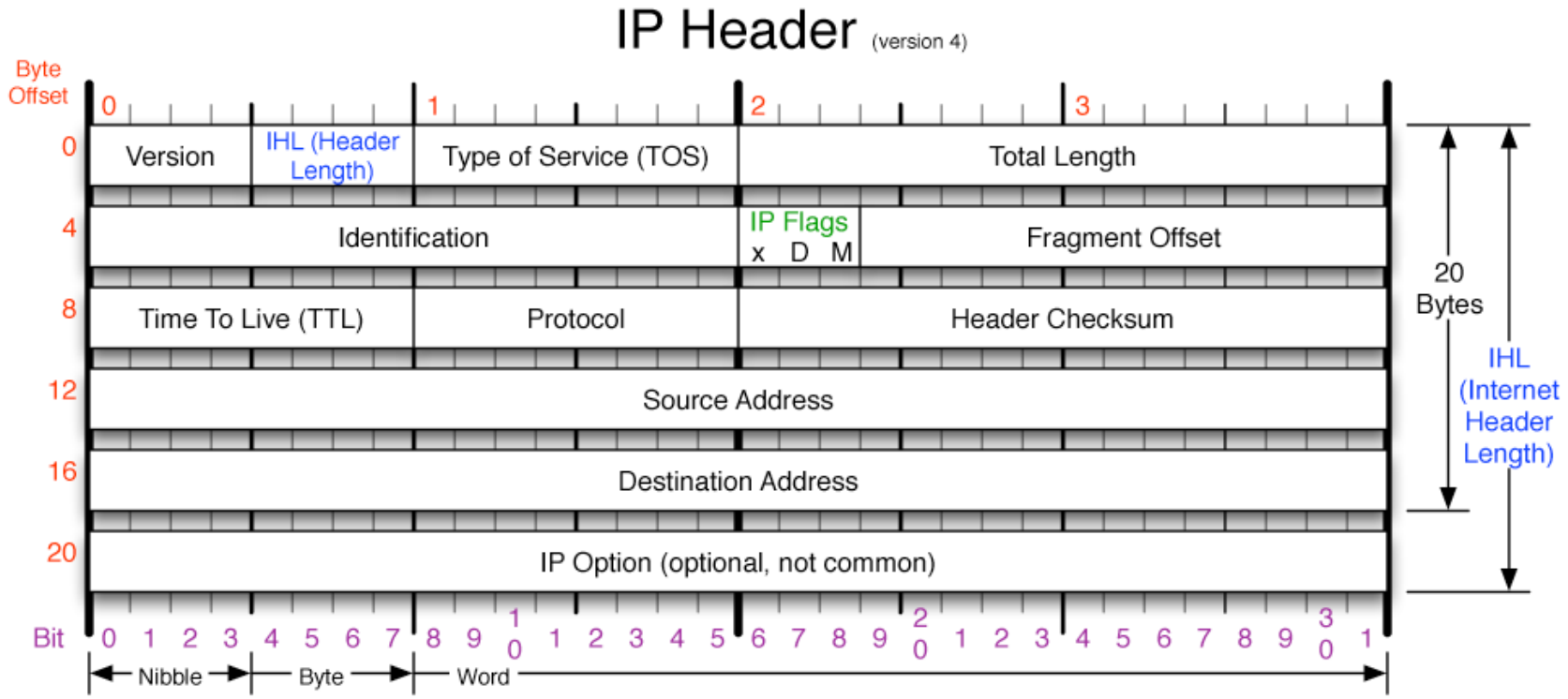
CAMADA DE REDE – IPV4 E IPV6

Cabeçalho IPv4

- Os protocolos TCP e UDP assentam em cima do protocolo IP
  - IP é camada Internet/Rede
- Cabeçalho IPv4
  - Parte fixa: 20 Bytes
  - Parte variável: [0...10] x 4 bytes = 40 bytes (máximo)
  - Tamanho máximo do cabeçalho IPV4 é de 60 bytes
  - O tamanho do cabeçalho é mantido numa grandeza de 4 bits que indique o número de palavras de 32 bits
    - Campo HLEN (*header length)*
  - *Os campos source IP e destination IP são inteiros de 32 bits sujeitos a endianness*



Cabeçalho IPv4



Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

Header Length

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

Protocol

IP Protocol ID. Including (but not limited to):

1 ICMP	17 UDP	57 SKIP
2 IGMP	47 GRE	88 EIGRP
6 TCP	50 ESP	89 OSPF
9 IGRP	51 AH	115 L2TP

Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

Header Checksum

Checksum of entire IP header

IP Flags

x D M

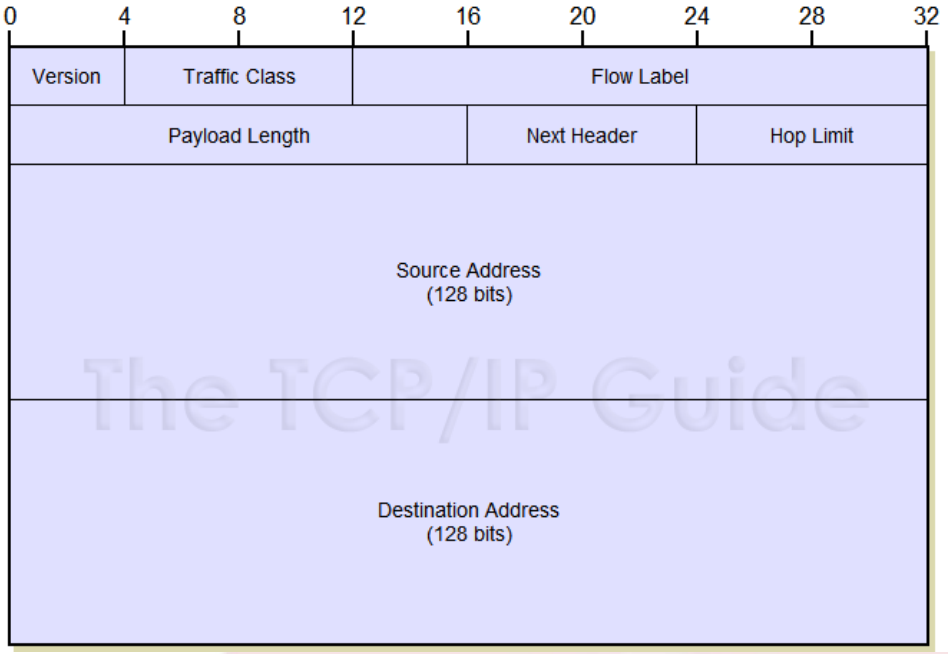
x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow

RFC 791

Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

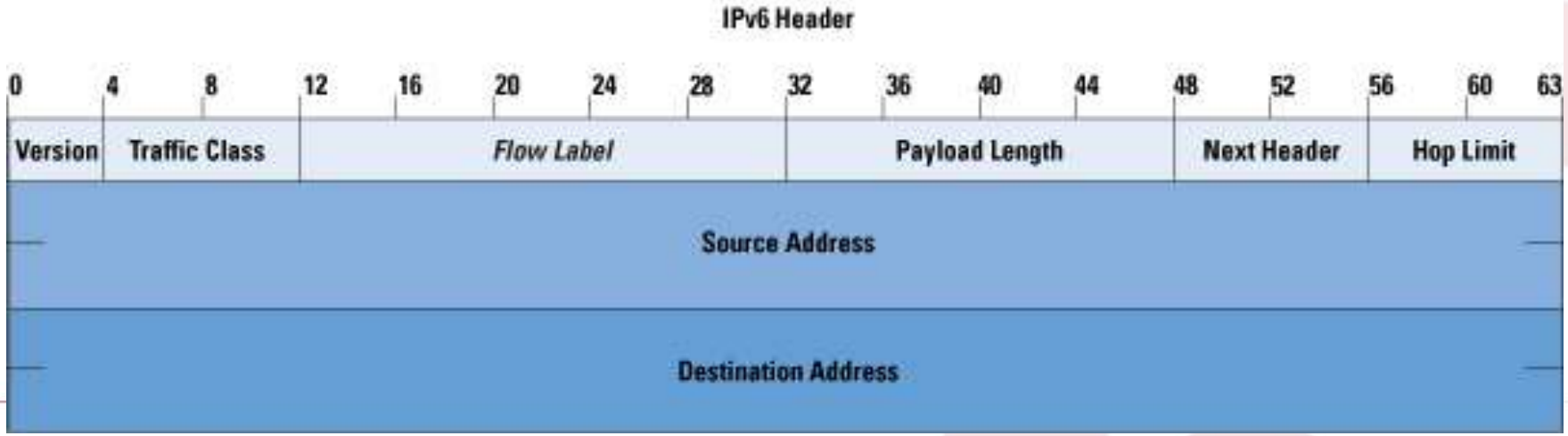
Cabeçalho IPv6

- Cabeçalho IPV6 tem **sempre** 40 bytes (8+16+16)
  - Cabeçalho mais uniforme e simples do que o IPv4
  - Dado ser mais simples, o processamento é mais rápido do que o IPv4
- Um endereço IPv6 é uma sequência de 16 bytes
  - Exemplo: 2001:0db8:85a3:0042:1000:8a2e:0370:7334
- Um endereço IPv4 tem 4 bytes
  - Exemplo: 192.168.200.5



Cabeçalho IPV4 vs IPV6

- Fonte: http://www.cisco.com/web/about/ac123/ac147/images/ipj/ipj_9-3/93_ipv6_fig1_lg.jpg



CAMADA DE TRANSPORTE

Camada de transporte / TCP-IP

- Fornecer comunicação lógica entre processos de aplicações em execução em máquinas remotas
- Ações de protocolos de transporte:
  - Emissor: divide as mensagens da aplicação em segmentos/datagrams e entrega-as à camada de rede
  - Recetor: reograniza segmentos/datagramas em mensagens e entrega-as à camada de aplicação
- Dois principais protocolos de transporte
  - TCP e UDP

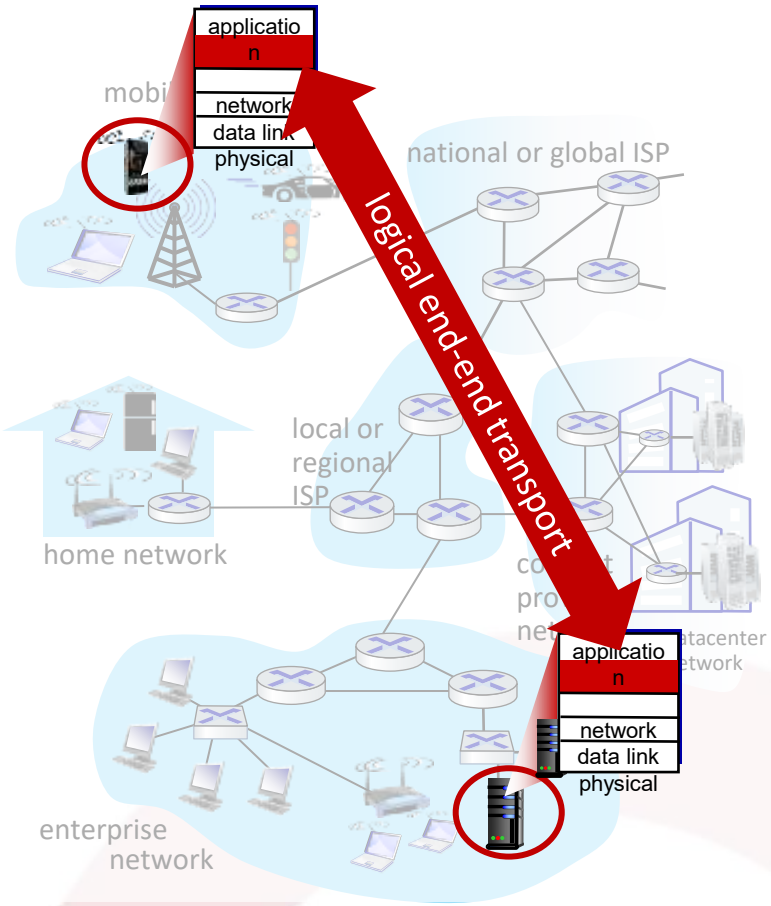
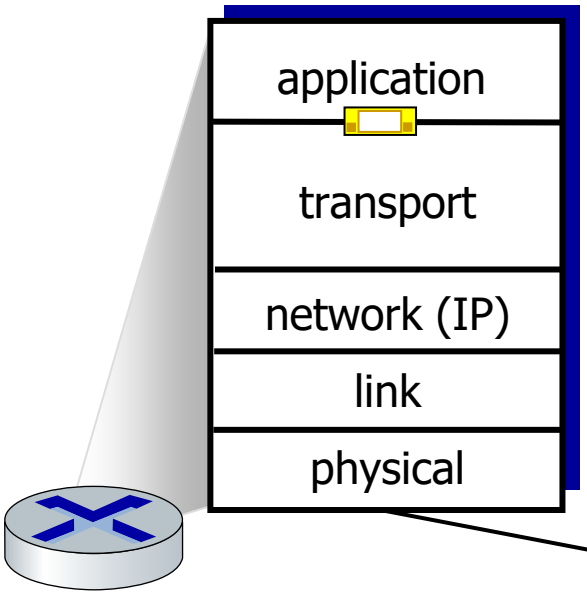


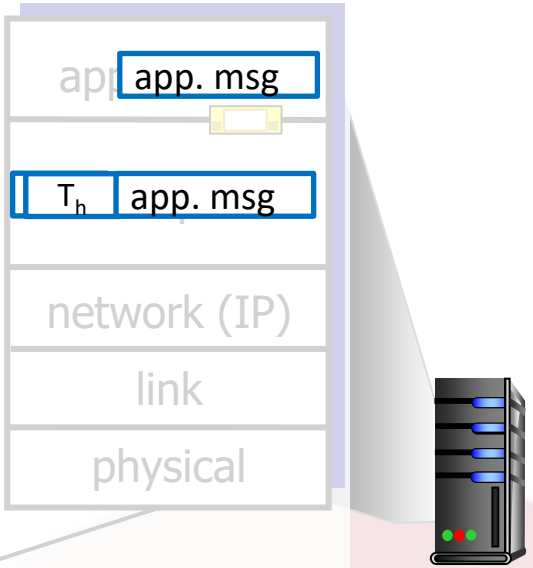
Figura: copyright 1996-2023 J.F Kurose and K.W. Ross

Camada de transporte em ação - Emissor

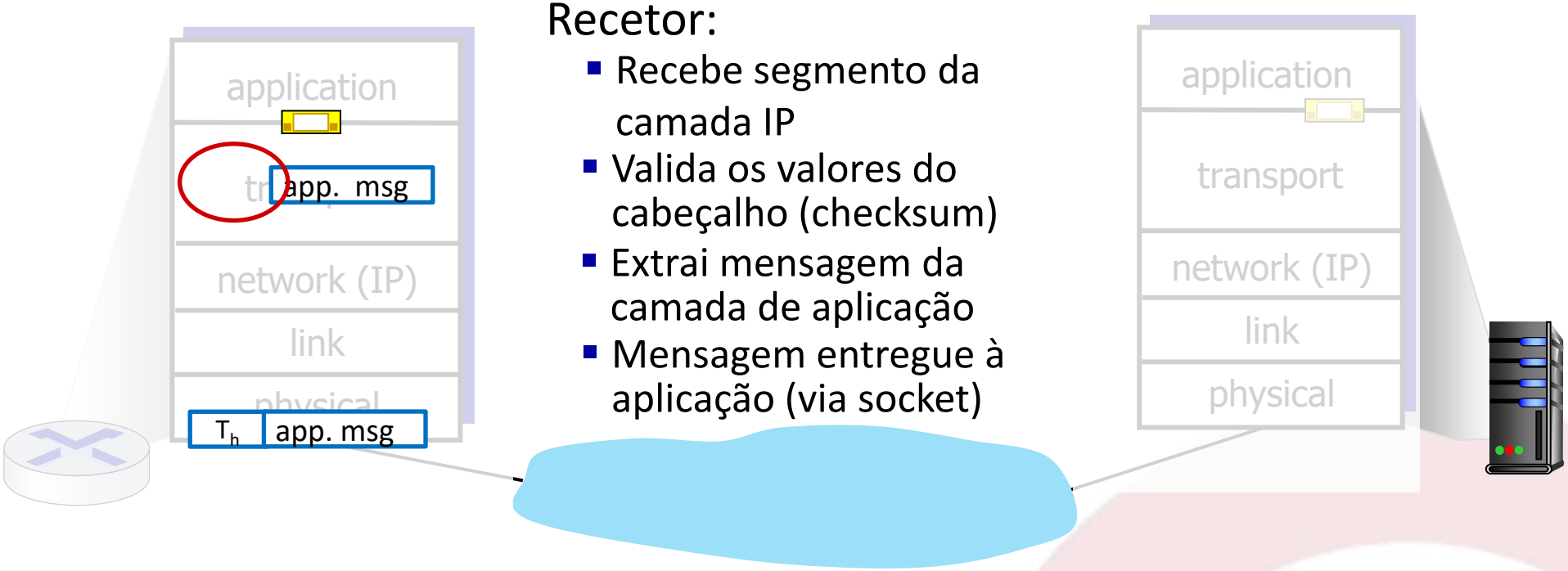


Emissor:

- É passada uma mensagem da camada de aplicação
- Determina os valores dos campos de cabeçalho de segmento
- Cria segmento/datagrama
- Passa segmento/datagrama para IP



Camada de transporte em ação - Recetor

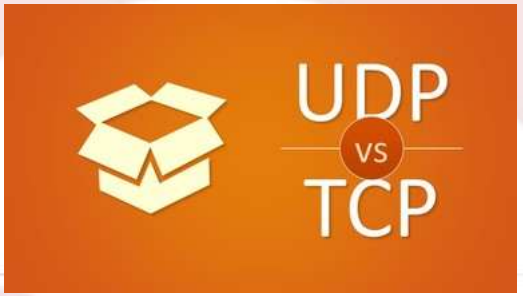


Recetor:

- Recebe segmento da camada IP
- Valida os valores do cabeçalho (checksum)
- Extrai mensagem da camada de aplicação
- Mensagem entregue à aplicação (via socket)

Camada de transporte TCP/IP

- Principais protocolos da camada de transporte TCP/IP
  - *User Datagram Protocol* (UDP)
    - Não orientado à ligação (envio de mensagens)
    - Não garante a entrega da informação
    - Não garante a ordem de entrega das mensagens
    - Especificação UDP – RFC 768 [Postel 1980]
  - *Transmission Control Protocol* (TCP)
    - Orientado à ligação (criação de uma stream)
    - Garante a entrega da informação (se tal for possível)
    - Mais complexo (a nível de implementação)
    - Especificação TCP – RFC 793 [Darpa Program 1981]
- Sugestão
  - "UDP and TCP: Comparison of Transport Protocols" (https://www.youtube.com/watch?v=Vdc8TCESIg8)



CONCEITO DE PORTO

Camada de transporte

Conceito de "porto" (porta) #1

- O Endereço IP (na Camada de Rede) identifica de forma única o hospedeiro (a máquina) na rede.
- Contudo, um hospedeiro executa múltiplas aplicações/serviços em simultâneo.
- **Pergunta:** Como é que os dados de entrada são entregues à aplicação correta (processo) no hospedeiro de destino?
- **Resposta:** O **porto** de Comunicação!
  - NOTA: é também empregue a palavra "porta" com a mesma finalidade

- **Endereço de Aplicação** = Endereço IP + Número de Porto

Conceito de “porto” (porta) #2

- O Porto (Porta) é um mecanismo lógico de endereçamento na Camada de Transporte.
- É um número de 16 bits, sem sinal, que identifica, de forma única, um processo/serviço em execução num hospedeiro.
- Alcance: Os números de porto variam de 0 a 65535 (2^16 - 1).
- Este par permite a comunicação processo-a-processo (ou aplicação-a-aplicação).
- O endereço completo de uma aplicação/serviço na rede é a combinação:
  - Endereço IP +
  - Número de Porto (socket address).

Portos TCP vs. Portos UDP

- Ambos os protocolos da Camada de Transporte (TCP e UDP) utilizam o conceito de porto
- O Porto 80/TCP é diferente do Porto 80/UDP; são *namespaces* separados no mesmo hospedeiro
- TCP (Transmission Control Protocol): Orientado à conexão, fiável, mais lento.
- UDP (User Datagram Protocol): Não orientado à conexão (sem garantia de entrega), mais rápido, para aplicações em tempo real.

- Exemplos:
  - O serviço web HTTP usa o porto 80/TCP
  - O serviço web HTTPS usa o porto 443/TCP (versão 1 e 2 do protocolo HTTP)
    - No HTTP3 é empregue o porto 443/UDP
  - O DNS usa o Porto 53/UDP, embora nalgumas situações recorra ao porto 53/TCP

Ainda sobre portos

- No TCP/IP os portos dividem-se em gamas
  - Classificação **BSD**
    - 1..1023: reservado para aplicações oficialmente registadas
      - ssh – porto 22
      - smtp – porto 25
      - http – porto 80 (web)
      - https – porto 443 (web cifrada)
      - Ver ficheiro /etc/services numa máquina UNIX
    - 1024...5000: atribuídos automaticamente pelo sistema
    - 5001...65535: portos não privilegiados (acesso livre)
  - Classificação **IANA** (Internet Assigned Numbers Authority)
    - 1..1023: portos bem conhecidos
    - 1024..49151: atribuídos/registados pela IANA
      - IANA - Internet Assigned Numbers Authority
    - 49152..65535: acesso livre

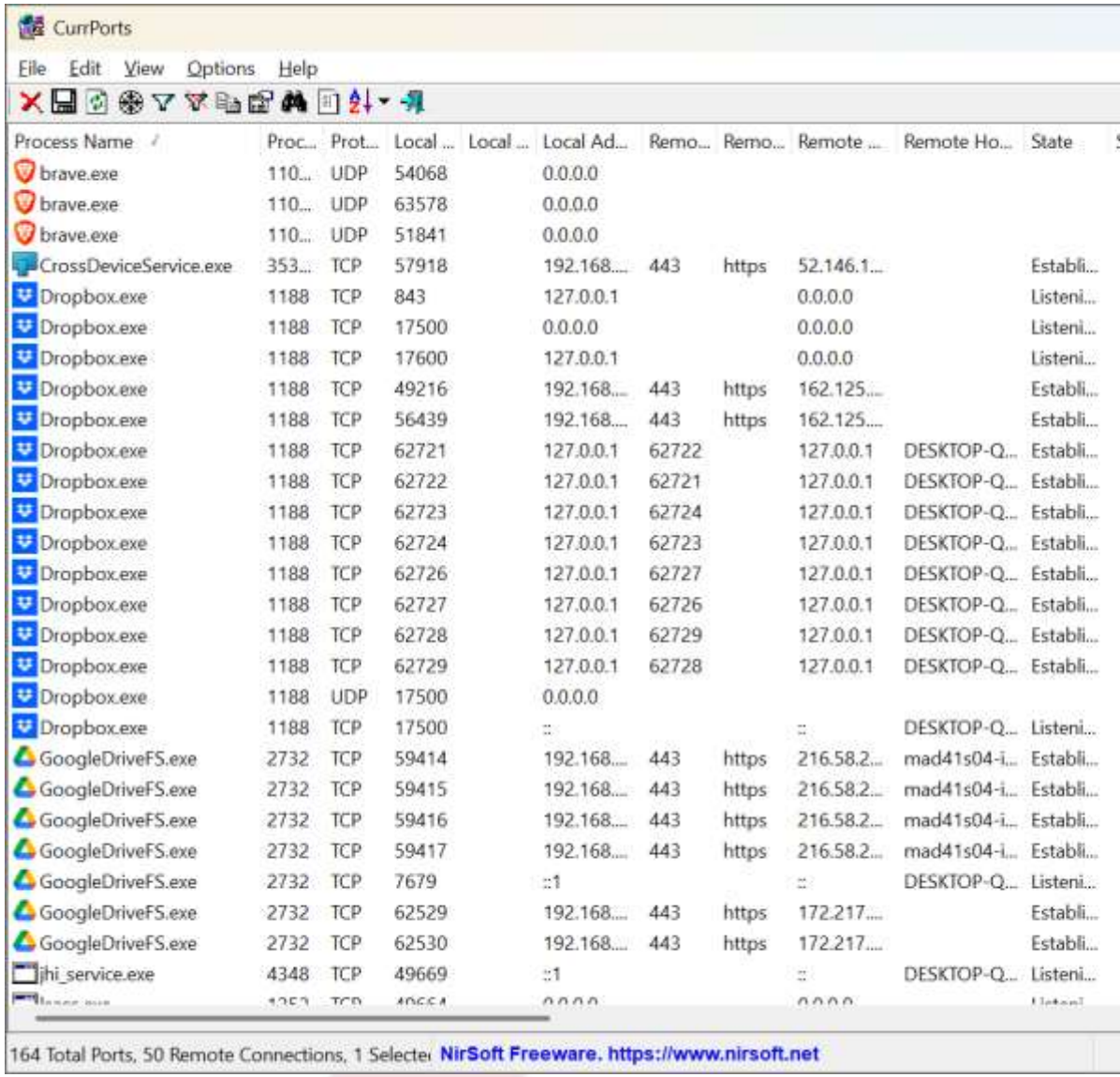
Alguns portos "importantes"

- Portos comuns
  - UDP e TCP

Porto	Serviço	Protocolo Comum	Descrição
20/21	FTP	TCP	Transferência de Ficheiros
22	SSH	TCP	Acesso Remoto Seguro
25	SMTP	TCP	Envio de E-mail
53	DNS	UDP/TCP	Resolução de Nomes de Domínio
80	HTTP	TCP	World Wide Web (Não Cifrado)
443	HTTPS	TCP	World Wide Web (Cifrado - SSL/TLS)

Utilitário currPort (nirsoft.net)

- Visualiza portos ativos no sistema Windows
- Pode ser executado em modo linha de comando
  - Relatórios, fechar portos, etc.
- Freeware da *Nirsoft.net*
  - http://www.nirsoft.net/utils/cports.html

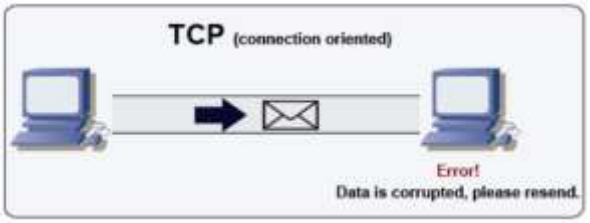


PROTOCOLO TCP

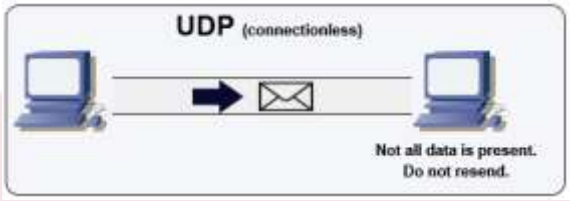
Camada de transporte

Camada de transporte

- Protocolos de transporte
  - TCP
    - mensagens transportadas por *segmentos*



  - UDP
    - mensagens transportadas por *datagramas*



Protocolo TCP >>

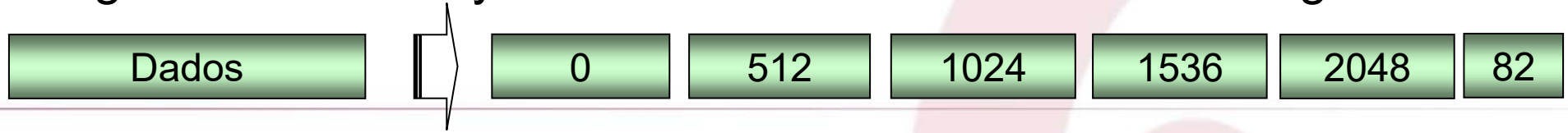
Características fundamentais do TCP

- **Orientado à ligação:** Requer um estabelecimento de ligação prévio
  - Three-Way Handshake
- **Fiável:** Garante que todos os dados chegam ao destino, na ordem correta e sem erros
- **Controlo de Fluxo:** Previne que o emissor sobrecarregue o recetor com mais dados do que este pode processar.
- **Controlo de Congestionamento:** Ajusta a taxa de envio para evitar o colapso da rede.
- **Comunicação Full-Duplex:** Permite que os dados fluam em ambas as direções simultaneamente.
- **Entrega Byte-Stream:** Os dados são transferidos como um fluxo contínuo de bytes, divididos em Segmentos TCP.

Exemplo

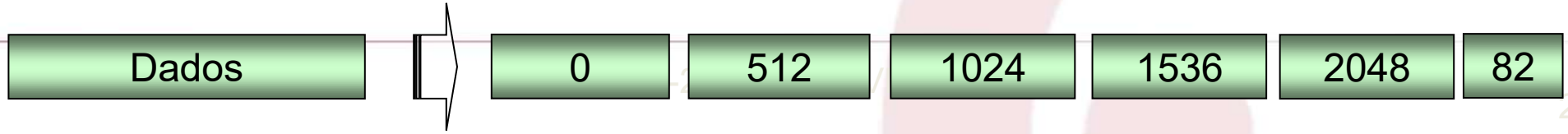
Envio de 2130 bytes (e.g., ficheiro com imagem)

Segmento de 512 bytes: os dados são enviados em 5 segmentos



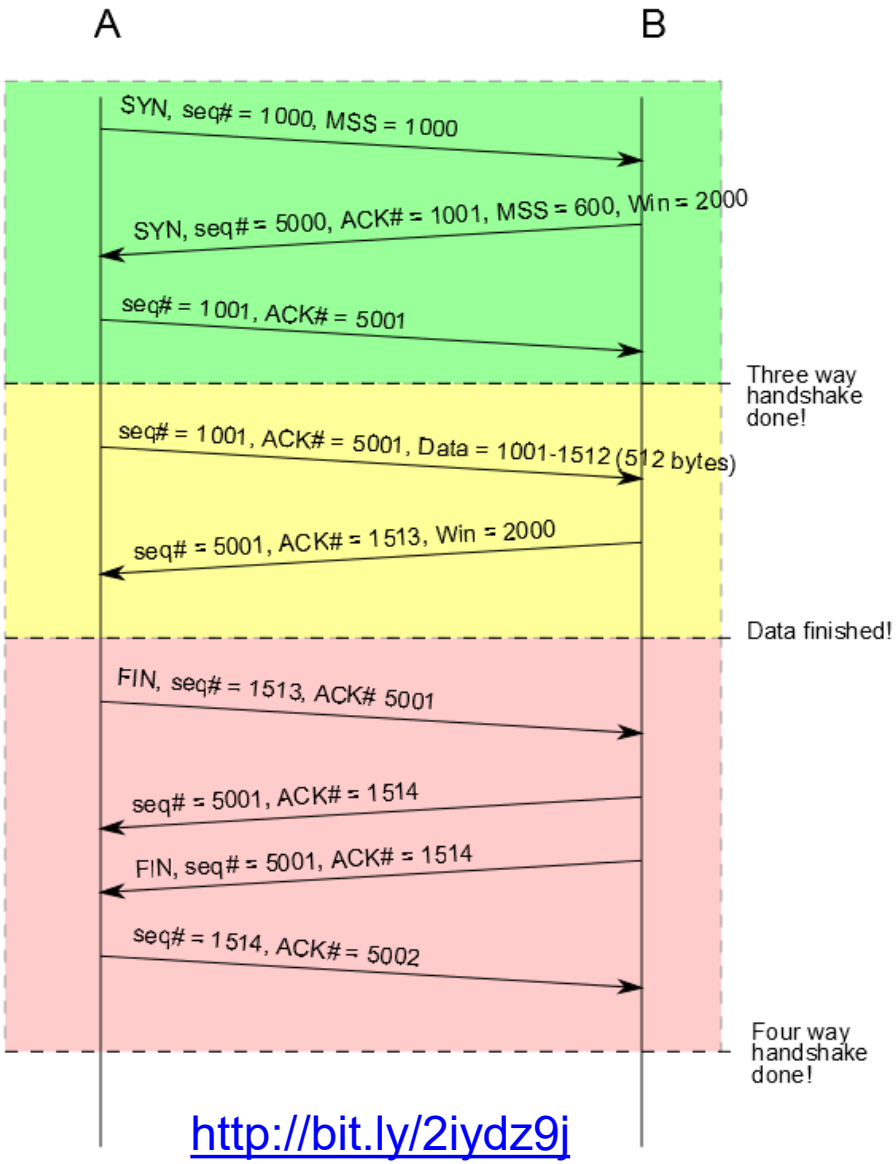
Protocolo TCP: Introdução (1)

- O protocolo TCP é dito “orientado à ligação”
  - Requer o estabelecimento de uma ligação entre os pares comunicantes
  - Ligação full-duplex
    - Pode enviar dados nos dois sentidos simultaneamente
- O TCP sequencia os dados em segmentos
  - Exemplo:
    - Envio de 2130 bytes (exemplo: ficheiro com imagem)
    - Exemplo
      - Segmento de 512 bytes: os dados são enviados em 5 segmentos



Protocolo TCP: Introdução (2)

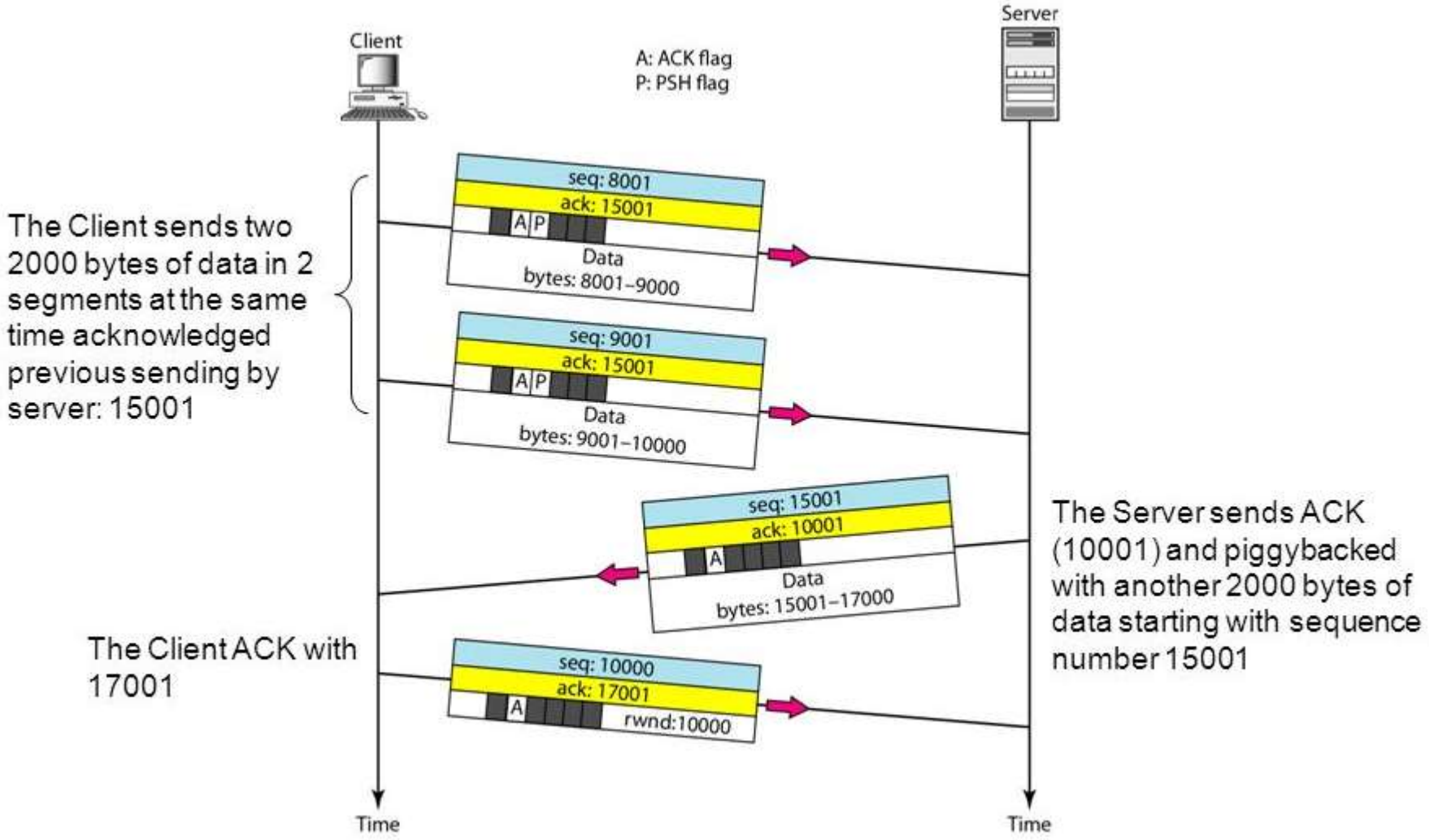
- O TCP garante a entrega da informação (se tal for possível)
  - Quando envia dados, o TCP requer que a outra extremidade comunicante lhe confirme a receção dos dados através de **ACK**nowledge
- O TCP garante a sequencialidade na entrega da informação
  - Cada segmento inclui um número de sequência que identifica o primeiro **byte** do segmento
  - Cada segmento (com dados ou sem dados) inclui um número de confirmação (ACK) que indica a sequência do próximo segmento que espera receber (dados foram recebidos + 1)



http://bit.ly/2iydz9j

Protocolo TCP: introdução (3)

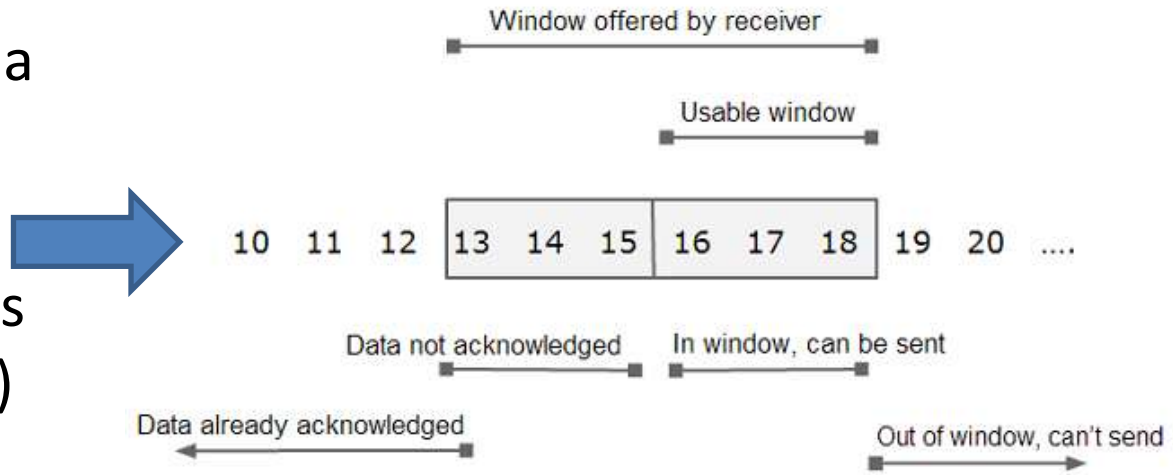
- Segmentação de dados



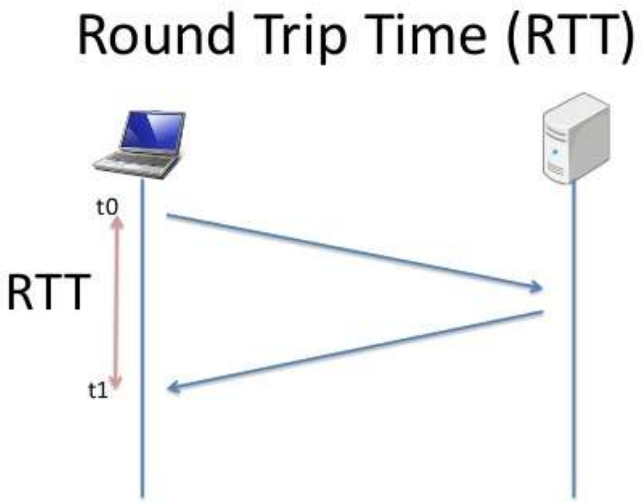
https://slideplayer.com/slide/9219608/

Protocolo TCP: Introdução (4)

- O protocolo TCP adapta-se dinamicamente às condições da ligação
  - Mecanismo de janela (anuncia o número de bytes que está disposto a aceitar)
  - Estimação RTT (round-trip time) ou RTD (round-trip delay time)
    - Tempo que demora a enviar um segmento e receber a sua confirmação
  - Controlo de fluxo (e.g. redução do débito de dados caso se estejam a perder segmentos)



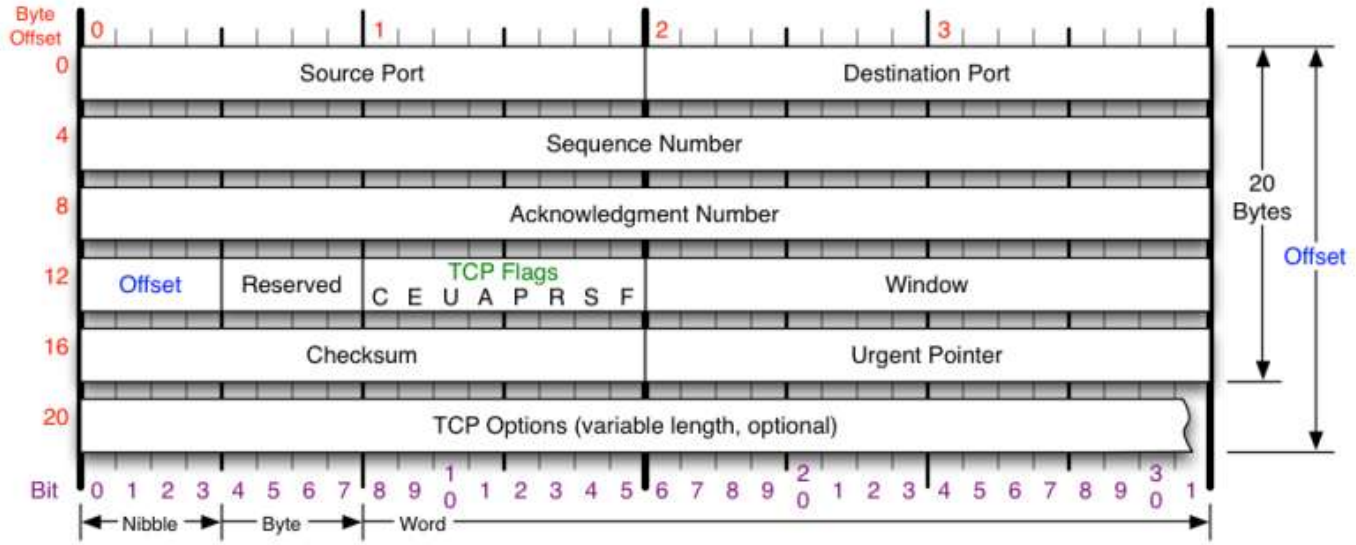
http://bit.ly/2AawJsU



CABEÇALHO TCP

Cabeçalho TCP

- Campos
  - Porto origem
  - Porto destino
  - Número de sequência
    - Garante sequência dos segmentos
  - Número de confirmação
  - Offset
    - Tamanho do cabeçalho em *words* de 32 bits
  - Flags
  - Window
    - Janela deslizante
    - Nº d bytes que podem ser enviados sem confirmação
  - Checksum
  - *Urgent pointer*



TCP Flags

C E U A P R S F

Congestion Window
C 0x80 Reduced (CWR)
E 0x40 ECN Echo (ECE)
U 0x20 Urgent
A 0x10 Ack
P 0x08 Push
R 0x04 Reset
S 0x02 Syn
F 0x01 Fin

Congestion Notification

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

Packet State	DSB	ECN bits
Syn	0 0	1 1
Syn-Ack	0 0	0 1
Ack	0 1	0 0
No Congestion	0 1	0 0
No Congestion	1 0	0 0
Congestion	1 1	0 0
Receiver Response	1 1	0 1
Sender Response	1 1	1 1

TCP Options

0 End of Options List
1 No Operation (NOP, Pad)
2 Maximum segment size
3 Window Scale
4 Selective ACK ok
8 Timestamp

Checksum

Checksum of entire TCP segment and pseudo header (parts of IP header)

Offset

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

RFC 793

Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.

http://nmap.org/book/tcpip-ref.html

Protocolo TCP: Segmento TCP (1)

- Campos
  - Porto origem (16 bits)
  - Porto destino (16 bits)
  - Número de sequência (32 bits)
    - Contém o número do 1º byte do segmento e não o número de segmentos já enviados
  - Número de confirmação (ACK, 32 bits)
    - Empregue para confirmação de recepção (ACK bit activo)
    - Indica o número de sequência do próximo segmento esperado
  - Data offset (4 bits)
    - Número de palavras de 32 bits do cabeçalho TCP
      - especifica onde começam os dados
  - Reservado (6 bits)
    - Reservado para uso futuro, deve possuir o valor zero

Protocolo TCP: Segmento TCP (2)

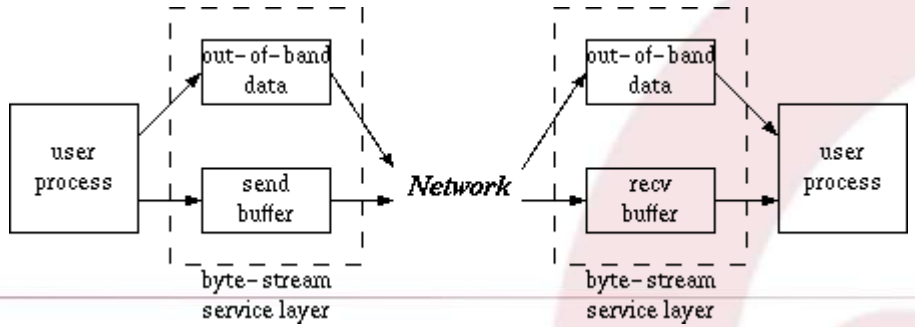
- Campos (continuação)
  - Bits de control (6 bits)
    - SYN
      - Pedido de estabelecimento de ligação
    - ACK
      - Indica que o campo de "acknowledge" do segmento deve ser considerado (i.e. confirma uma recepção)
    - PUSH
      - Todos os dados da ligação existentes no buffer de envio da máquina "emissora" devem ser enviados ao destinatário ("flush" da ligação)
    - URGENT
      - Indica que o ponteiro de urgência do segmento deve ser considerado (dados "Out-of-Band", OOB)
    - RESET
      - Quebra abrupta da ligação
    - FIN
      - Término regular da ligação

Protocolo TCP: Segmento TCP (3)

- Janela (16 bits)
  - Controlo de fluxo
  - Número de bytes que o destinatário pode receber sem confirmação (este número está condicionado pela opção WSO)
  - Checksum (16 bits)
    - CRC16 para garantir a integridade dos dados do segmento
      - O calculo do checksum abrange cabeçalho + dados, sendo o campo de checksum considerado a 0 para efeitos do cálculo
  - Urgent Pointer (16 bits)
    - Offset (relativo ao "sequence number") indicando o último dado urgente (OOB). Apenas é válido se o bit de controlo URGENT estiver activo
      - Nota: uso da opção "MSG_OOB" no **send** e **recv** para envio e receção de dados out-of-band.

OOB

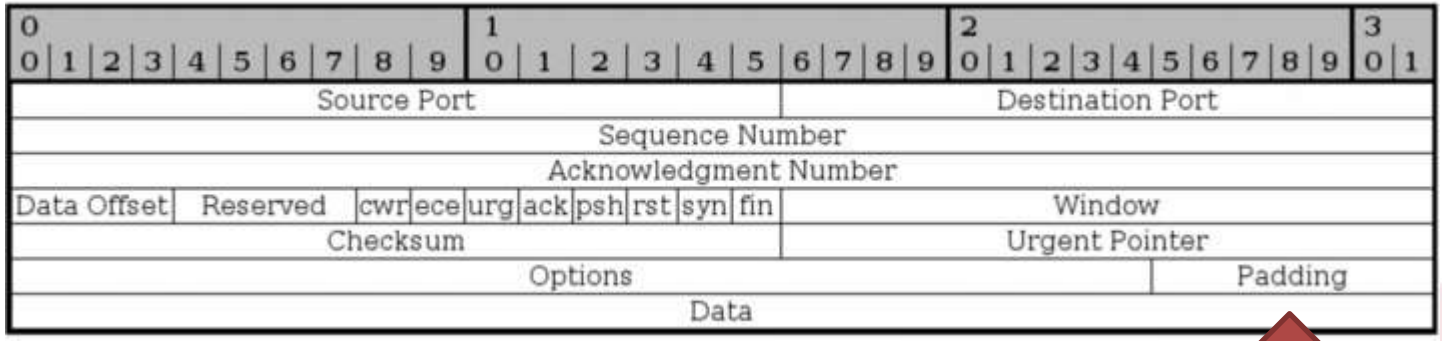
```
sent = send(fd,msg,len,MSG_OOB)
n = recv(connfd, buff, sizeof(buff)-1,MSG_OOB);
```



http://bit.ly/2j5tHCR

Protocolo TCP: Segmento TCP (4)

- Options (variável)
  - Múltiplos de 8 bits
- Padding (variável)
  - Conjunto de zeros para garantir que o cabeçalho TCP termina numa palavra de 32 bits



Padding
Para garantir que o cabeçalho tem um tamanho múltiplo de 32 bits

PROTOCOLO 3 WAY-HANDSHAKE

Protocolo de inicialização de ligação TCP

Protocolo TCP: 3WHS (1)

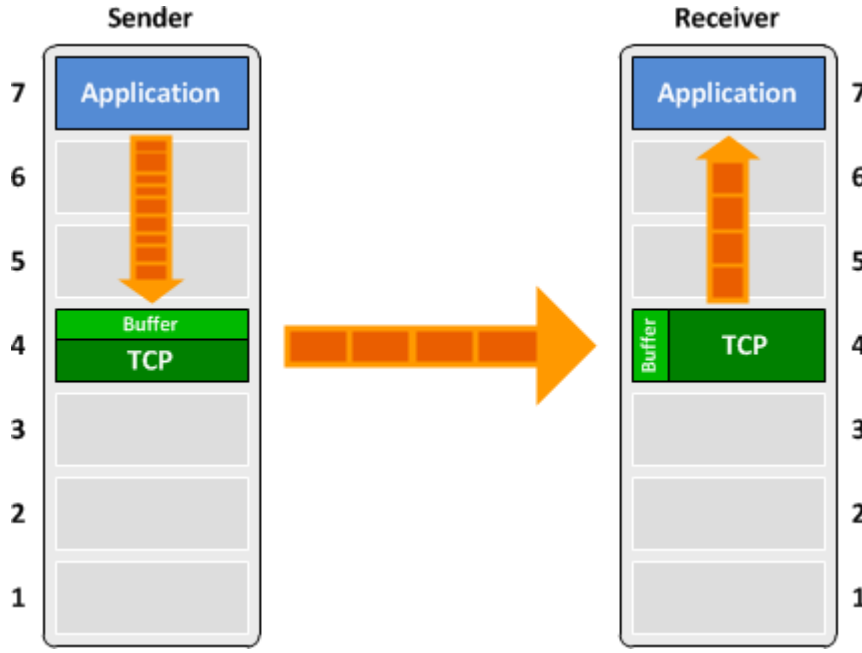
- Estabelecimento da ligação
  - Algoritmo "*Three-Way Handshake*" (3WHS)
  - **Três** etapas para o estabelecimento da ligação TCP entre duas entidades
    1. Cliente pede ligação: segmento SYN J
    2. Servidor confirma (ACK) e pede ligação: segmento ACK J+1, SYN K
    3. Cliente confirma (ACK): segmento ACK K+1

Protocolo TCP: 3WHS (2)

- Legenda
  - SYN
    - Segmento para pedido de ligação SYNchronize
  - ACK
    - Segmento de confirmação ACKnowledge
  - J
    - número da sequência inicial do cliente a ser utilizado na comunicação
  - K
    - número da sequência inicial do servidor a ser utilizado na comunicação
    - O TCP numera sequencialmente os segmentos que envia (J, K)

Protocolo TCP: Buffers (1)

- A pilha protocolar TCP/IP encontra-se integrada no sistema operativo
  - Todos os detalhes da transmissão (ACK, segmentos perdidos, retransmissões, etc.) são tratados pelo TCP de forma "assíncrona"
  - É por esta razão que o desempenho da pilha protocolar difere de SO para SO
- A camada TCP desconhece quando a aplicação irá pedir os dados lidos
  - Válido para qualquer aplicação TCP
- O TCP guarda os dados que recebe em memória "tampão" ("TCP buffers") por forma que os dados estejam prontos quando a aplicação efetuar a leitura
  - O tamanho da janela deslizante está condicionado pelo tamanho do *buffer*

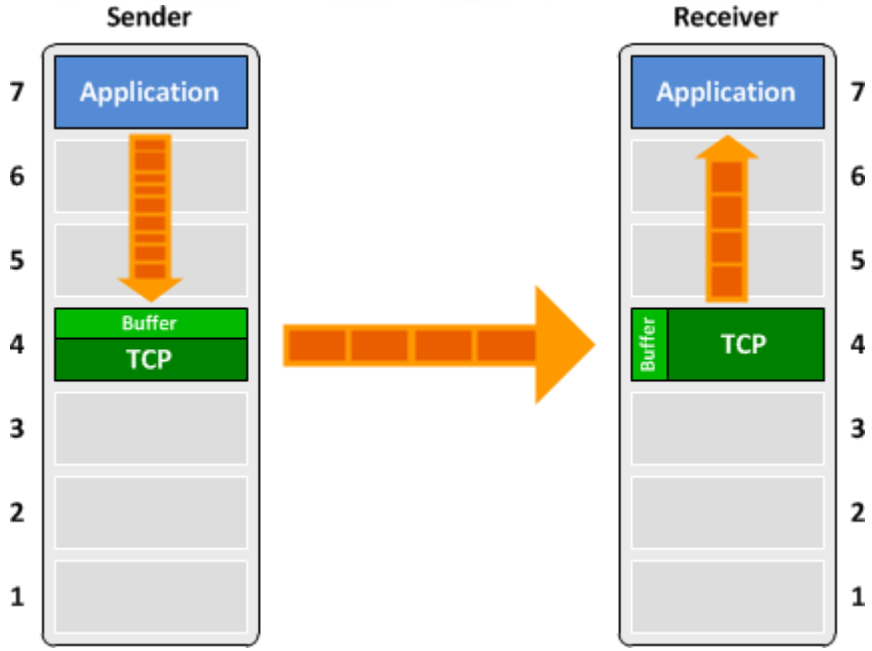


https://bit.ly/2qjL9nK

Protocolo TCP: Buffers (2)

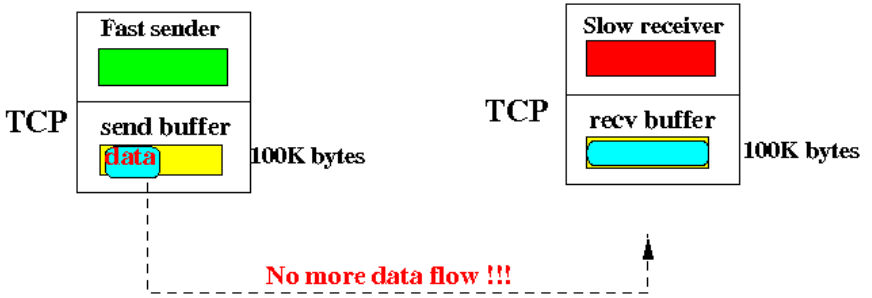
- A aplicação envia dados através de uma chamada ao sistema (**send(...)**, **write(...)**,etc.)

- Os dados são colocados na memória tampão de envio (ainda no emissor), onde permanecem até que sejam confirmados (ACK do receptor)



https://bit.ly/2qjL9nK

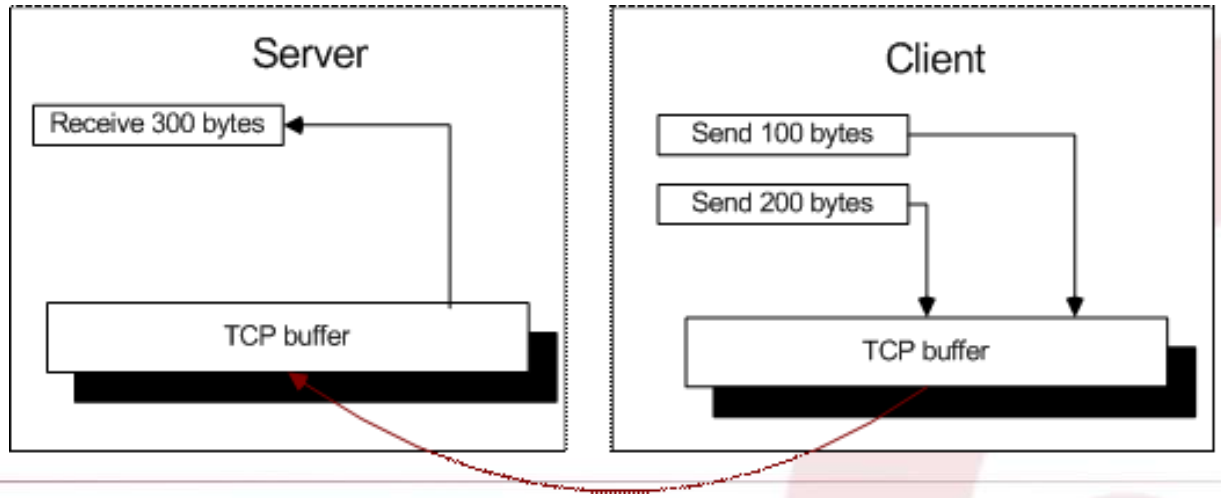
- A camada TCP só aceita dados da aplicação se ainda houver espaço na memória tampão
  - Se não houver espaço, o processo que pretende enviar os dados fica bloqueado até que haja espaço



https://tinyurl.com/ycwusa47

Protocolo TCP: Buffers (3)

- Comunicação TCP entre emissor e recetor
  - Emissor e recetor vão alternando (ligação bidirecional)
  - A comunicação é desemparelhada – exemplo
    - Emissor envia *100 bytes* e de seguida mais *200 bytes*
    - Recetor poderá receber *300 bytes* de uma só vez
    - Não existe emparalhamento direto entre envio e receção



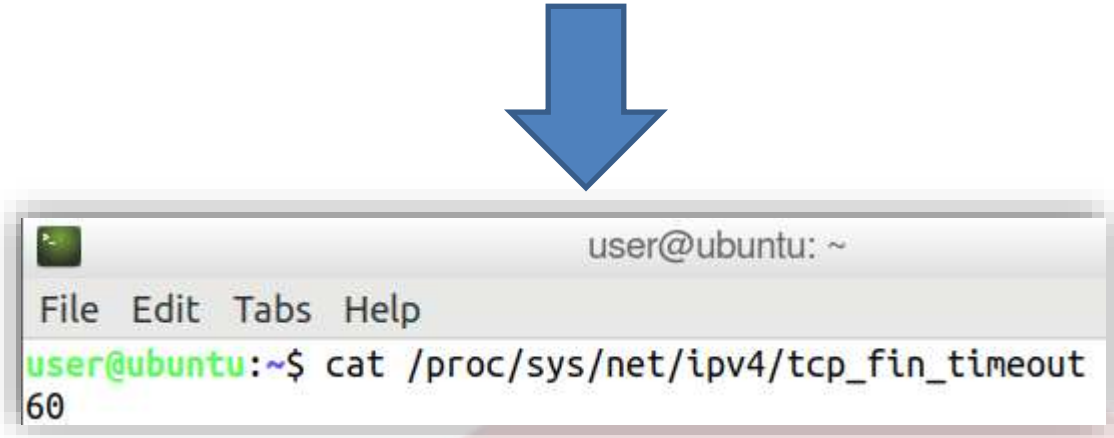
Fonte:
http://www.codeproject.com/KB/IP/socketmessageboundary.aspx

Maximum Segment Lifetime

- MSL: Maximum Segment Lifetime
  - Máximo tempo de vida expectável para um segmento TCP
  - Relevante quando se encerra uma ligação
    - A pilha protocolar deve aguardar 2 x MSL unidades de tempo antes de permitir a reutilização de um porto previamente empregue para uma ligação TCP
    - Racional
      - Evitar que um segmento atrasado de uma ligação anterior seja interpretado como sendo parte da nova ligação

Linux

- Valor MSL controlado pelo pseudo-ficheiro tcp_fin_timeout
- 60 segundos no Lubuntu



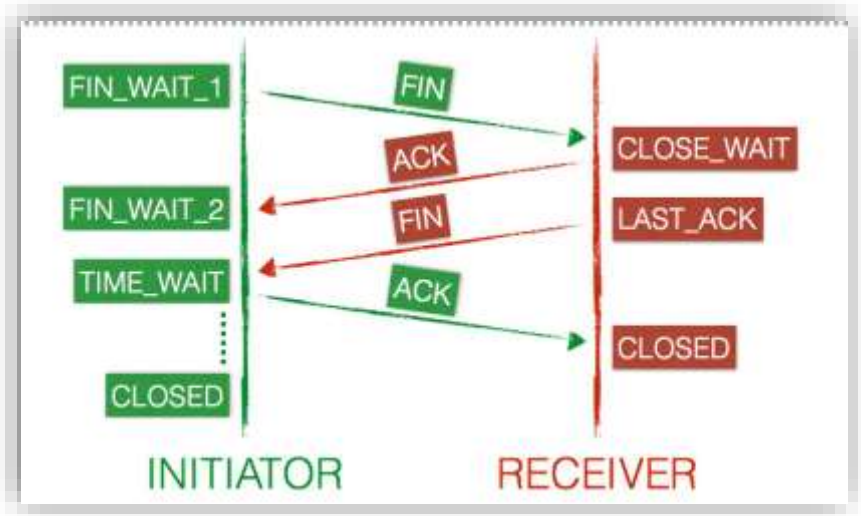
`/proc/sys/net/ipv4/tcp_fin_timeout`

TCP: TÉRMINO DA LIGAÇÃO

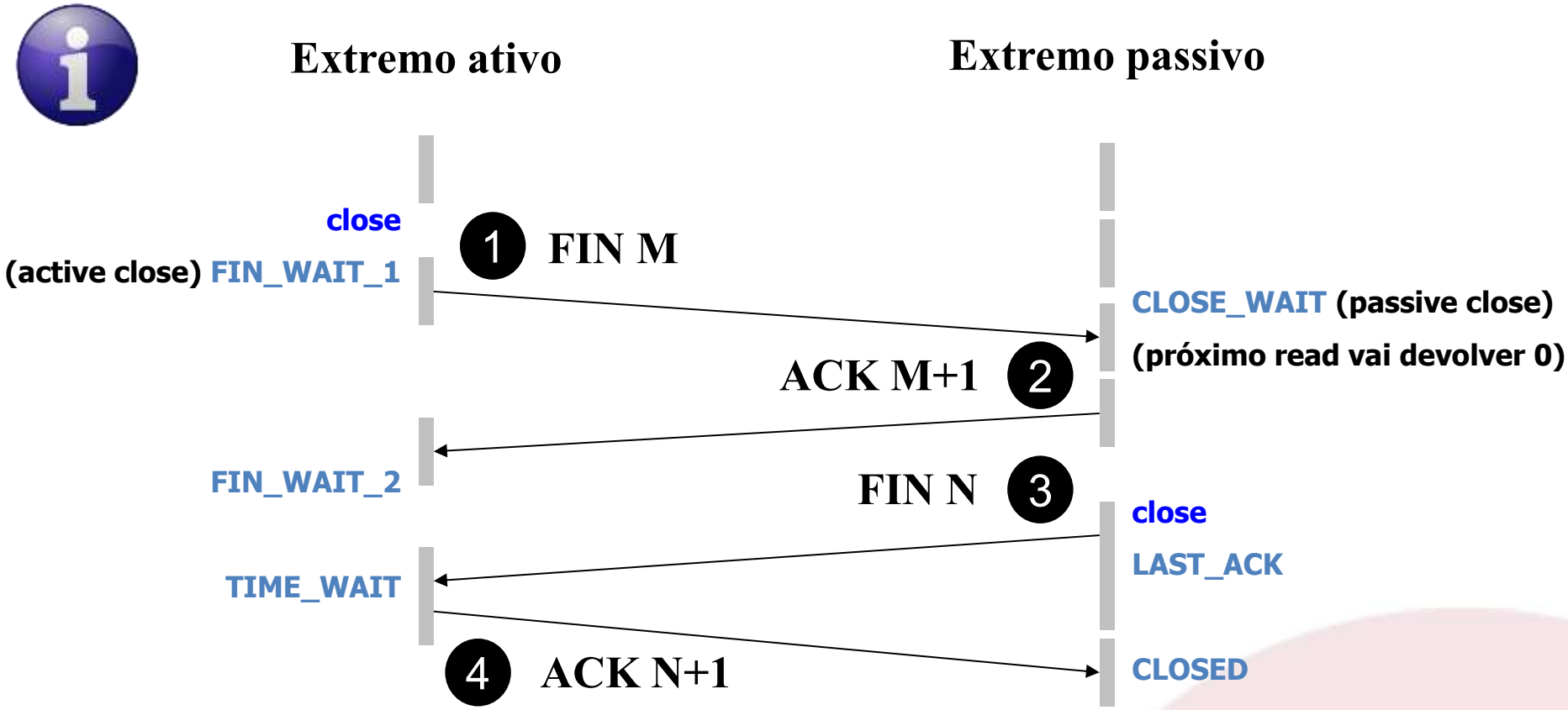
Protocolo TCP: Término da ligação (1)

- Término da ligação: **quatro etapas**

1. Um dos extremos (extremo activo) chama o close (*active close*)
   - É enviado um segmento FIN
2. O outro extremo (extremo passivo) recebe o segmento FIN (passive close)
   - Confirma a recepção com ACK
   - Envia fim-de-linha (EOF) à aplicação na próxima leitura (i.e., o próximo read vai devolver zero)
3. Quando a aplicação do extremo passivo (*passive close*) detecta o fim de ligação, fecha (close) a sua extremidade comunicante, originando o envio de um segmento FIN
4. O extremo activo (*active close*) recebe e confirma o segmento FIN



Protocolo TCP: Término da ligação (2)

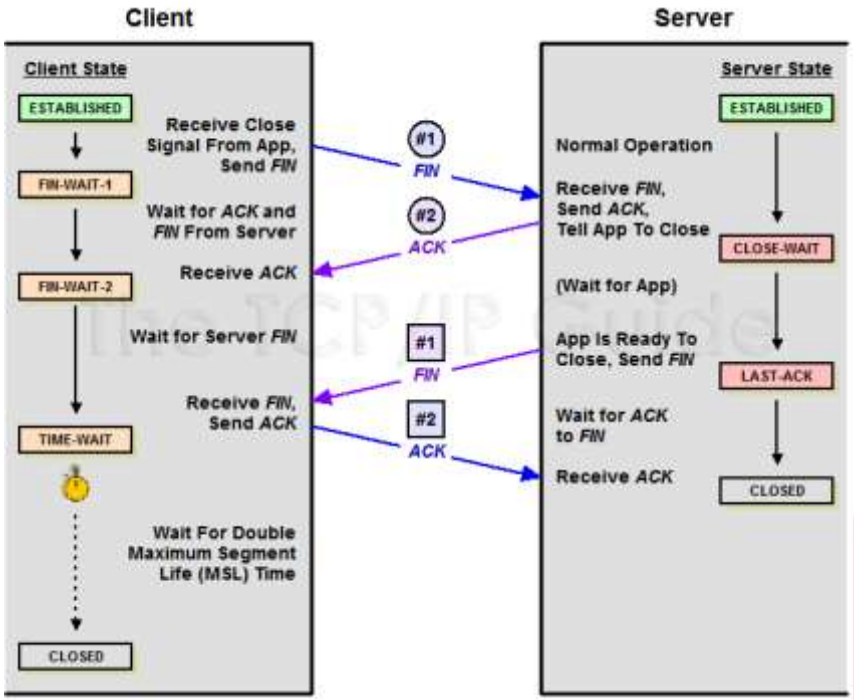


- Estado TIME_WAIT
  - Enquanto o socket permanece no estado TIME_WAIT continua a ocupar recursos

Protocolo TCP: Término da ligação (3)

- Entre as etapas 2 e 3 é possível enviar dados dado que o canal apenas está meio-fechado
  - Está fechado no sentido do extremo ativo para o extremo passivo
  - O extremo passivo pode continuar a enviar dados para o extremo activo

- Quando uma ligação terminou (último ACK enviado) podem existirem assuntos pendentes
  - E se o ACK se extraviar? O último FIN será reenviado requerendo um ACK
  - E se existirem segmentos duplicados ou perdidos que cheguem ao destino após uma "longa" pausa ?
    - TCP fica à espera durante um intervalo de tempo (de 1 a 4 minutos ~2 x MSL, dependendo da implementação)
    - Fica no estado TIME_WAIT



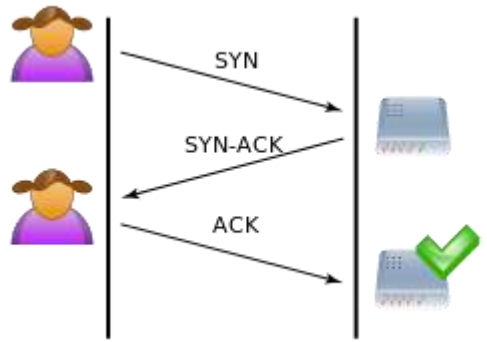
Fonte: TCP/IP guide

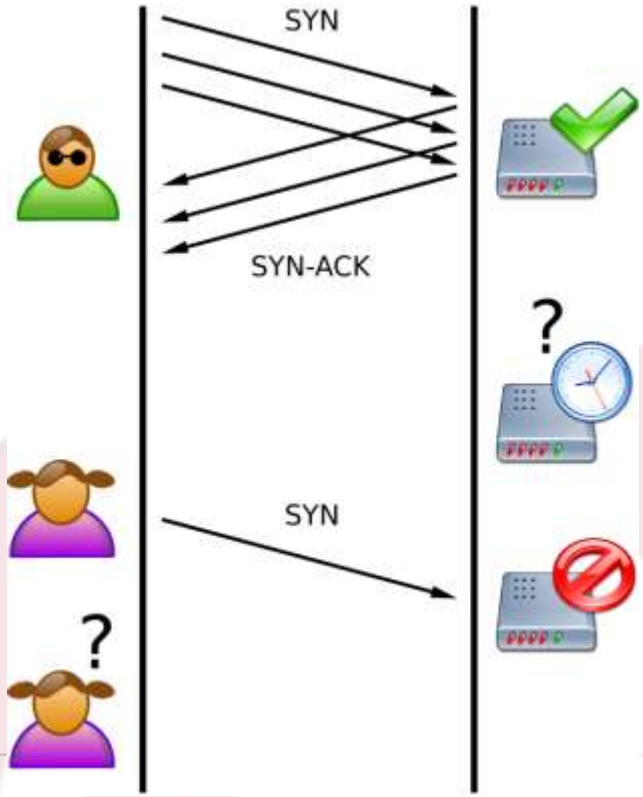
CIBERATAQUES – NEGAÇÃO DE SERVIÇO VIA TCP

Negação de Serviço (DoS/DDoS): Tornar um recurso (servidor, rede) indisponível para utilizadores legítimos.

TCP - SYN flood (#1)

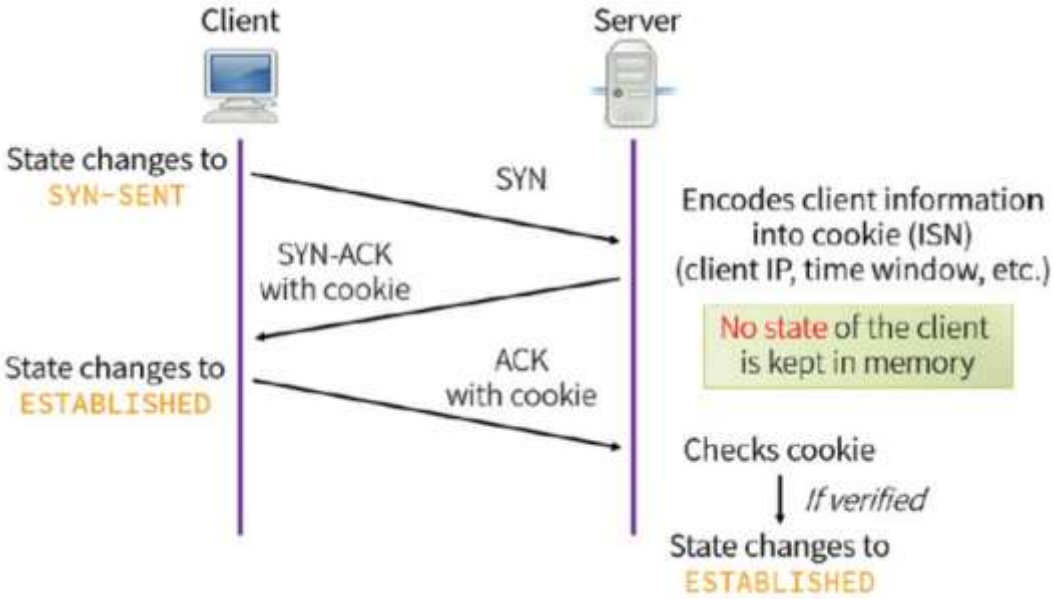
- SYN flood – negação de serviço
  - Atacante envia pedido de ligação TCP – SYN ao servidor
  - Servidor responde com ACK e pedido de ligação
  - Atacante nunca responde
    - Durante um certo intervalo de tempo, o servidor mantém o registo do pedido de ligação do atacante
  - Cada pedido pendente consome alguns recursos
  - Se existirem muitos pedidos pendentes, o servidor fica bloqueado…
    - Negação de serviço





TCP - SYN flood (#2)

- Prevenção anti-SYN flood
  - Sistema usa ***cookies SYN***
  - Os ***cookies SYN*** ajudam a evitar a exaustão de recursos
  - O servidor responde com um pacote SYN-ACK modificado (cookie)
  - O cookie contém informações sobre a ligação (*hash* do endereço IP, porto e número de sequência)
  - O servidor não mantém informação de estado, e assim, não são ocupados recursos



https://images.app.goo.gl/J5iPC8Ks8Bsg5E2f7

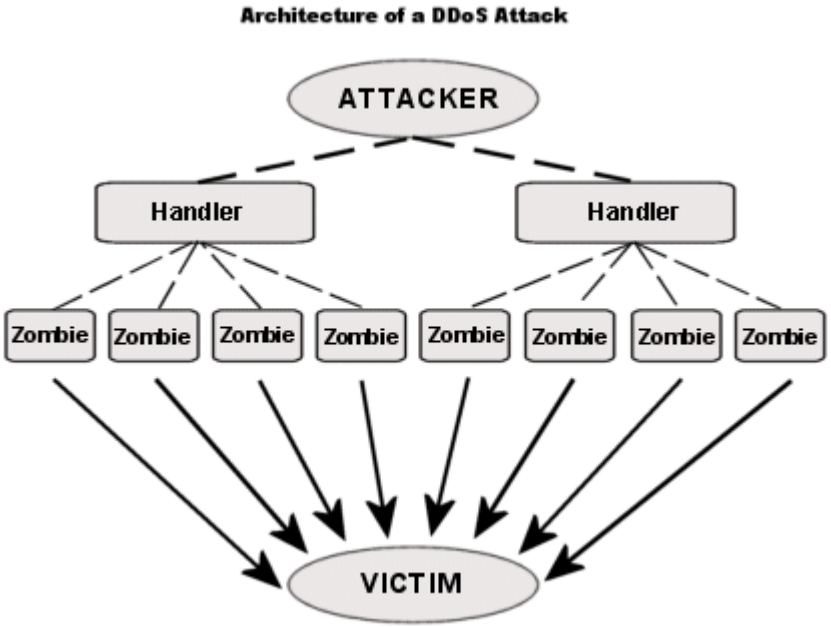
SYN flood distribuído (#1)

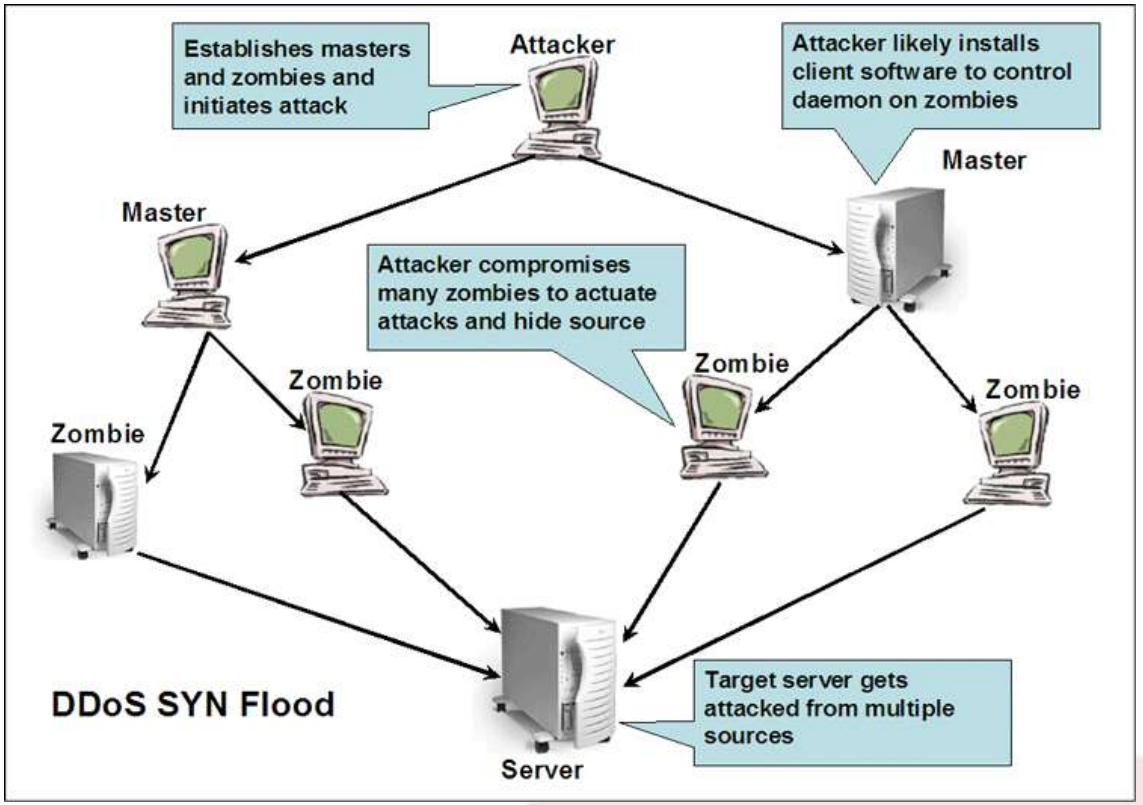
- Ataque SYN flood distribuído
  - Caso de *Distributed Denial of Service* (DDOS)
  - N sistemas informáticos solicitam de forma simultânea pedidos de ligação ao servidor "alvo"
  - Os N sistemas informáticos pertencem usualmente a *botnets*
    - *Botnet*: rede de sistemas informáticos infectados por software malicioso e controlados remotamente por um cibercriminoso
    - Sistemas informáticos podem ser infetados através de engenharia social, uso de software "*crackado*", vulnerabilidade no SO e aplicações, etc.
  - Existem serviços na *dark web* de *DDOS as a service*, i.e., o serviço podem ser contratado
  - DDOS SYN flood é um dos exemplos de DDOS

SYN flood distribuído (#2)

- Ataque SYN flood distribuído

http://bit.ly/2efcQ8O





- Exemplos

4.2 Tbps of bad packets and a whole lot more: Cloudflare's Q3 DDoS report

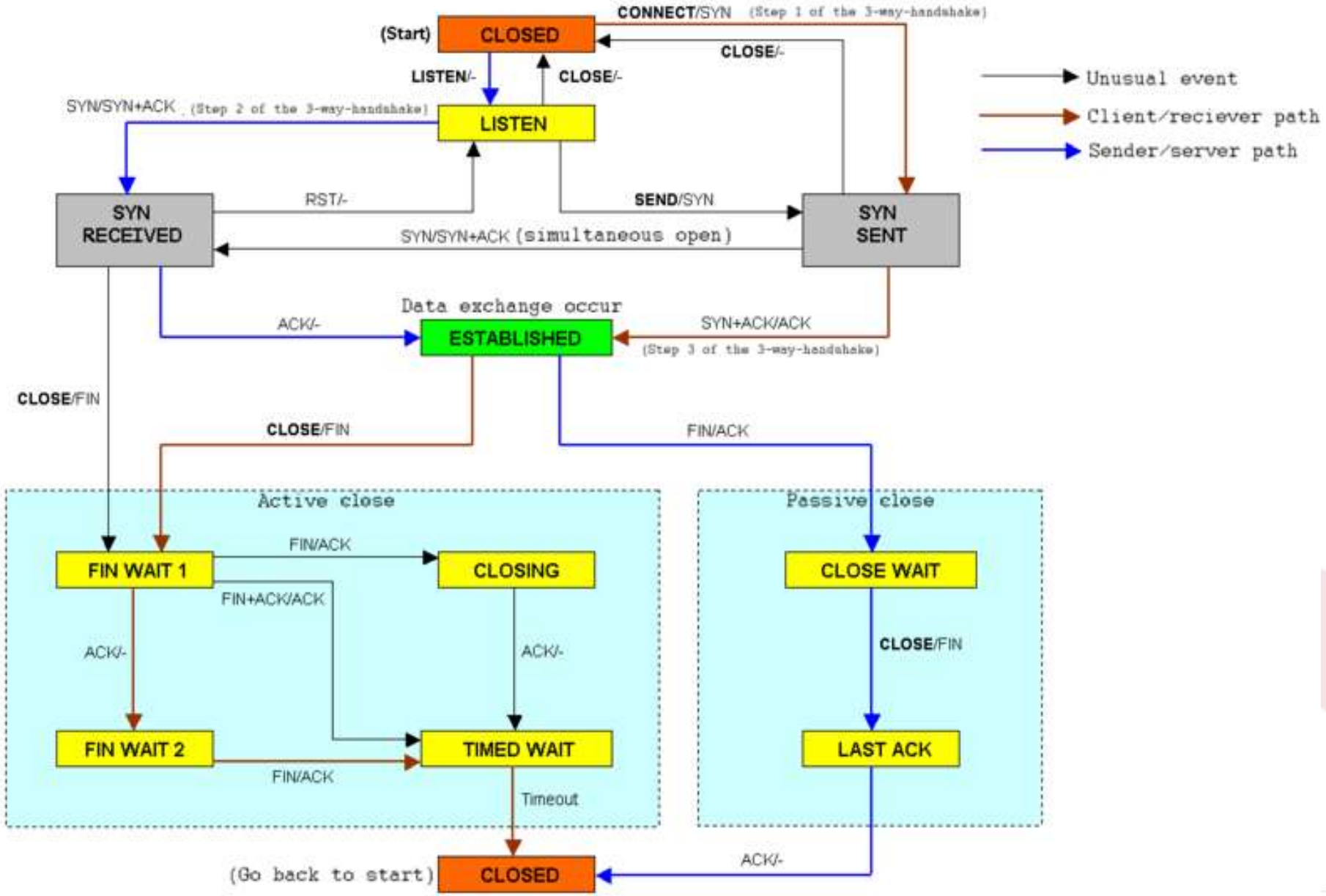
2024-10-23

https://tinyurl.com/2s3cp44z

Attack vectors

In Q3, we saw an even distribution in the number of network-layer DDoS attacks compared to HTTP DDoS attacks. Of the network-layer DDoS attacks, SYN flood was the top attack vector followed by DNS flood attacks, UDP floods, SSDP reflection attacks, and ICMP reflection attacks.

Protocolo TCP: Diagrama de estados



UDP

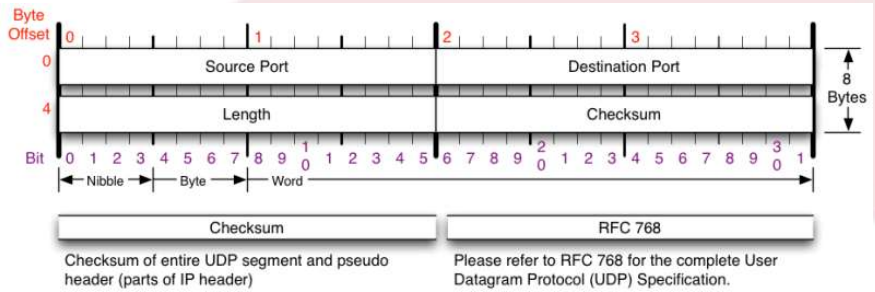
Protocolo de transporte

Protocolo UDP: Introdução (#1)

- A aplicação envia um *datagrama* num canal UDP (*socket*)
  - O datagrama UDP é encapsulado num datagrama IP (IPv4 ou IPv6) e enviado para o seu destino
- O UDP é não orientado à ligação
  - Não garante entrega dos datagrams
    - É um "best-delivery effort"
  - Não garante a ordem de entrega dos datagramas
    - Um datagrama pode chegar antes de outro que tenha sido enviado anteriormente (i.e. a sequencialidade não é garantida)
  - Não filtra datagramas duplicados
- UDP vs. TCP
  - O UDP pode ser comparado a um serviço de correio, o TCP a um serviço de telefone

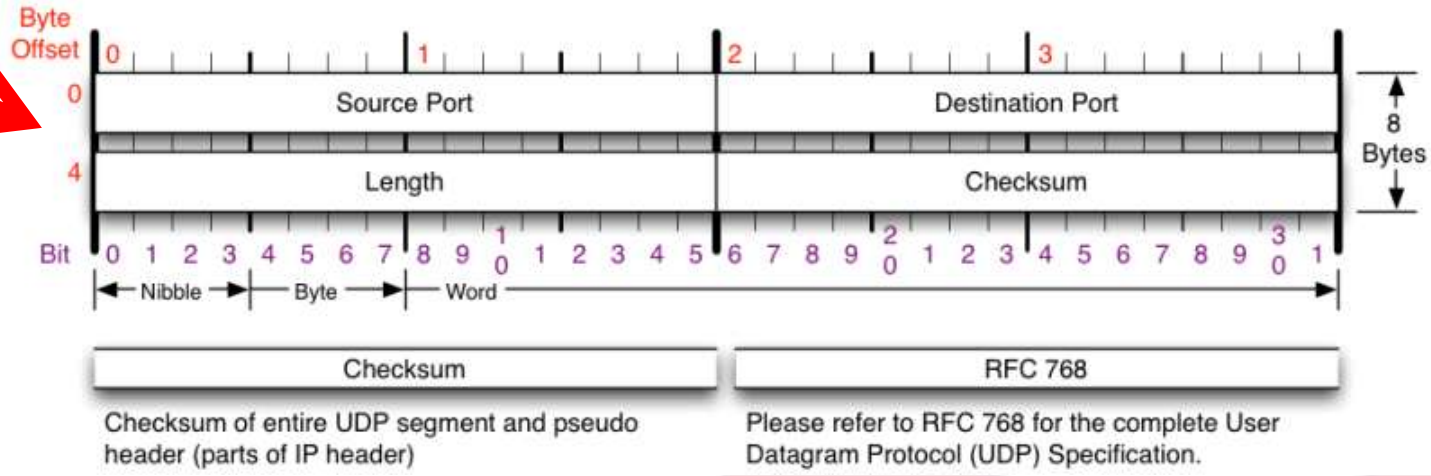
Protocolo UDP: Introdução (#2)

- Cada datagrama UDP tem um limite teórico de 2^{16} bytes, i.e., 65536 bytes.
  - Na prática:
    - Valor aceite como garantido são 576 bytes
      - Conhecido como o *"minimum maximum reassembly buffer size"*
      - Cada implementação deve ser capaz de agregar datagramas com esse tamanho (RFC 1122)
    - Exemplo: o protocolo DNS usa datagramas com 512 bytes
- UDP Exige pouco recursos
  - Reduzida sobrecarga (datagrama contém dados e pouco mais)
  - Reduzido uso de processamento
    - Útil para serviços de gestão de rede
      - DNS, SNMP, BOOTP, NTP, etc.



Cabeçalho UDP

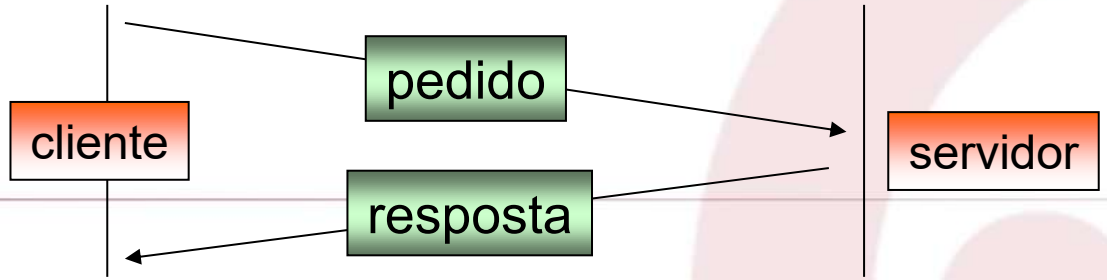
- Muito mais simples que o cabeçalho TCP
- Elementos
  - *Source port*: porto origem (16 bits)
  - *Destination port*: porto destino (16 bits)
  - *Length*: tamanho de todo o datagrama (cabeçalho + dados)
  - *Checksum:* código de verificação da integridade do cabeçalho e dos dados



UDP - CARACTERÍSTICAS

Protocolo UDP: Caraterísticas (#1)

- UDP suporta operações "broadcast" e "multicast"
  - broadcast – difusão para todos os elementos da rede
  - multicast – difusão para todos os elementos de um grupo de difusão (grupo de *multicast*)
- No UDP não existe estabelecimento de ligação
  - UDP não é orientado à ligação
  - Não é gasto tempo, nem pacotes em *estabelecimento de ligação*
  - Característica importante em aplicações de "pedido-resposta único" (e.g. DNS, NTP)



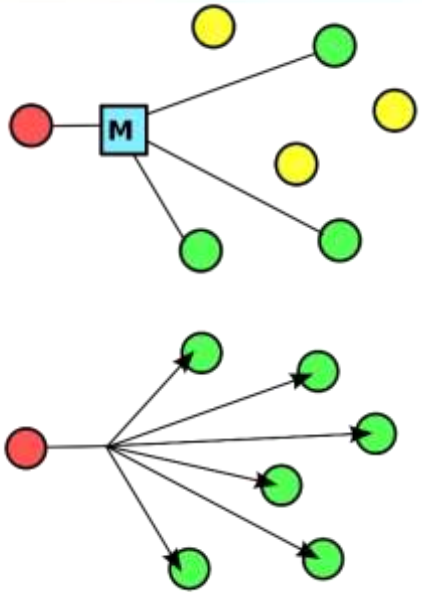
Protocolo UDP: Caraterísticas (#2)

- UDP **não** tem
  - Confirmação de recepção
  - Garantia de entrega ou notificação de "não entrega"
  - Garantia na ordem dos datagramas
  - Não deteta datagramas duplicados
  - Controlo de fluxo
  - Arranque lento e controlo de congestionamento

- Caso se pretenda qualquer desses mecanismos numa aplicação UDP será necessário implementá-los (o que não é fácil)

Protocolo UDP: Aplicabilidade

- Deve-se optar pelo protocolo de transporte UDP nos seguintes tipos de aplicações:
  - Aplicações para difusão de mensagens ("broadcast" e/ou "multicast")
  - Aplicações de "pedido-resposta único"
    - Contudo é necessário que a aplicação providencie detecção de erros
    - Exemplos
      - Domain Name System (**DNS**), Simple Network Management Protocol (**SNMP**), Network Time Protocol (**NTP**) e Simple Network Time Protocol (**SNTP**)
  - Aplicações em que o custo computacional, a largura de banda e latência da comunicação devem ser minimizado
    - Streaming de áudio/vídeo; jogos online



https://en.wikipedia.org/wiki/Broadcasting_(networking)

TCP vs. UDP

TCP

- **Transporte confiável** entre o processo de envio e o processo recetor
- **Controle de fluxo:** o remetente procura não sobrecarregar recetor
- **Orientado à ligação:** configuração necessária entre os processos comunicantes
- **TCP não providencia:** temporização, garantia de débito mínimo, segurança

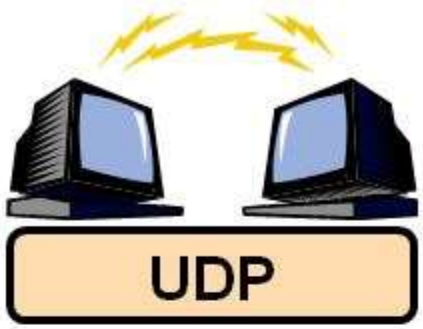
UDP

- **Transferência de dados** não fiável entre o processo de emissor e o recetor
- **UDP não providencia:** Confiabilidade, controlo de fluxo, controlo de congestionamento, temporização, garantia de taxa de transferência, segurança ou configuração de ligação

TCP vs. UDP

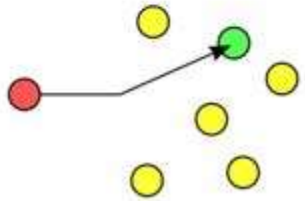
TCP

- **Slower but reliable transfers**
- **Typical applications:**
  - Email
  - Web browsing

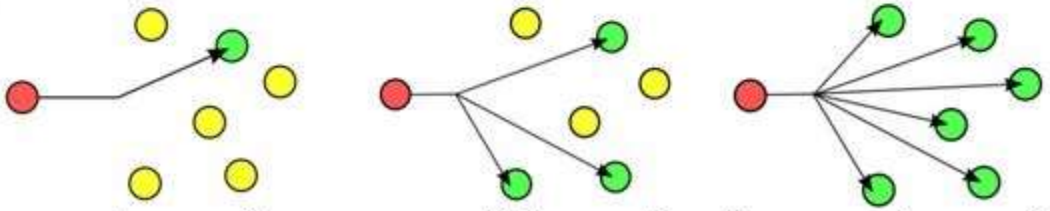


UDP

- **Fast but non-guaranteed transfers ("best effort")**
- **Typical applications:**
  - VoIP
  - Music streaming



unicast



unicast **multicast** **broadcast**

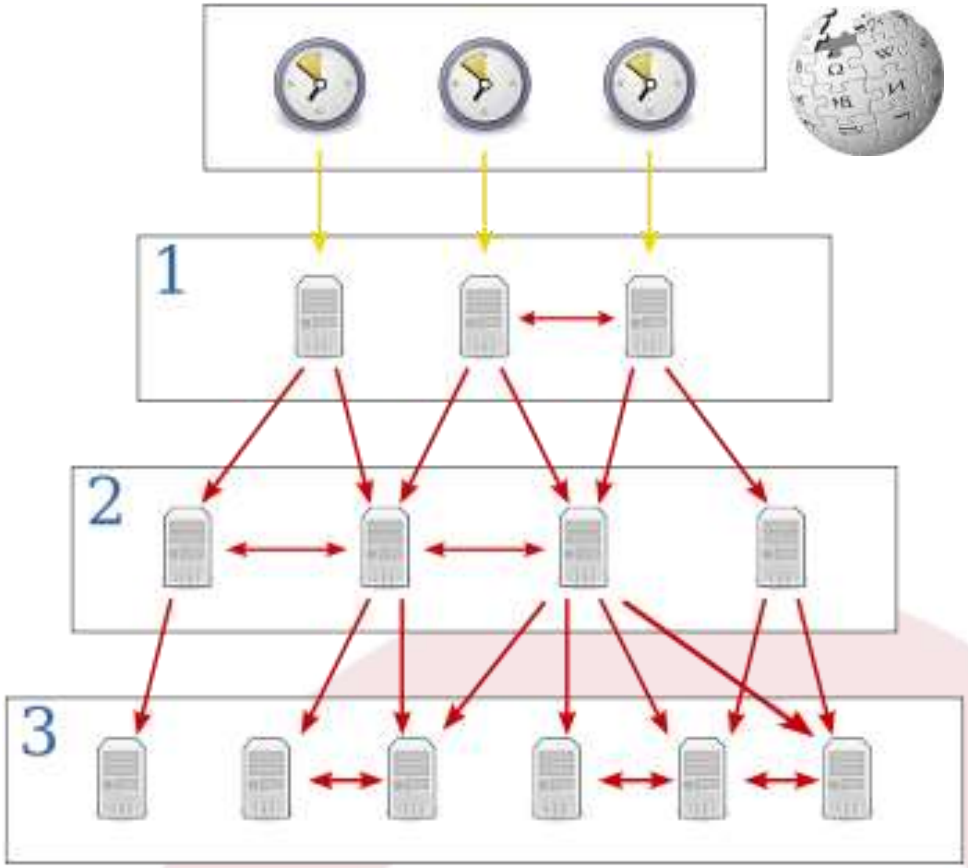
https://bit.ly/2Q49o4A

NETWORK TIME PROTOCOL

Caso de estudo de protocolo aplicacional que usa UDP

Network Time Protocol (NTP)

- Visa alcançar a sincronização dos relógios dos sistemas informáticos
  - É fundamental na informática o acesso à "hora certa"
- Usa o porto 123 UDP
- NTP funciona por níveis
  - Nível 0 é a fonte de tempo (relógio atómico, relógio GPS, estações rádio)
  - Nível 1 tem ligação direta à fonte de tempo
  - Nível 2 liga ao nível 1...
  - Etc.
  - Quando mais afastado da fonte, menor é a precisão




```
user@ubuntu:~$ ntptrace
localhost: stratum 2, offset -0.000029, synch distance 0.040446
85.199.214.100: stratum 1, offset 0.000001, synch distance 0.000001, refid 'GPS'
```

UDP: QUANDO NÃO USAR…

Protocolo UDP: não utilizar quando... (#1)

- Não é aconselhado usar UDP em:
  - **Aplicações com vários ciclos consecutivos de pedido-resposta**
    - Em operações não idempotentes podem surgir problemas de consistência de dados
    - O que é uma operação idempotente?
      - Operação que não altera dados, pelo que pode ser repetida (e.g., consulta saldo bancário, pedido DNS, leitura de um ficheiro)
      - Operação não idempotente: operação altera dados (e.g., transferência bancária, apagar de um ficheiro, etc.)
  - **Aplicações que movimentam volumes de dados significativos (e.g. transferência de ficheiros)**
    - UDP não garante a entrega dos dados

Exceções >>

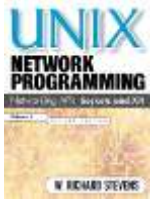
Protocolo UDP: não utilizar quando...

- Algumas exceções
  - **Trivial FTP (TFTP)** – transferência de ficheiros emprega UDP. **Porquê?**
    - Implementação simples
      - Requer pouco espaço. Importante para sistemas embebidos (*firmware*)
    - O TFTP é empregue como mecanismo de transferência de ficheiros de booting em protocolos de *remote booting*.
  - **Network File System** (NFS, sistema de ficheiros remoto) assenta no UDP (embora já exista NFS-TCP). **Porquê ?**
    - Em meados da década de 80, as implementações de UDP eram (bem) mais rápidas do que o TCP
  - **HTTP/3 – versão 3 do HTTP assenta no UDP. Porquê?**
    - Poupança de recursos do lado servidor, possibilitando maior escalabilidade
    - Reduzir a latência na troca de dados
    - Desenvolvimento novo protocolo denominado QUIC

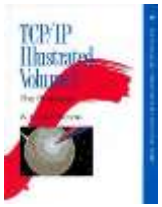
BIBLIOGRAFIA

Bibliografia

UNIX Network Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI, Prentice Hall, 1998, ISBN 0-13-490012-X.

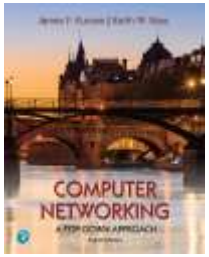


*TCP/IP Illustrated, Volume 1: "*The protocols", 2nd edition, *Richard Stevens,* Addison-Wesley Professional, 2011



TCP performance, Geoff Huston, The Internet Protocol Journal, Volume 3, Number 2, June 2000

- *"Computer Networking: A Top-down Approach ", Kurose, J. F., & Ross, K. W. (2022). Pearson, Hoboken, NJ.*



- *"The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference", Charles M. Kozierok, No Starch Press. 2005.*