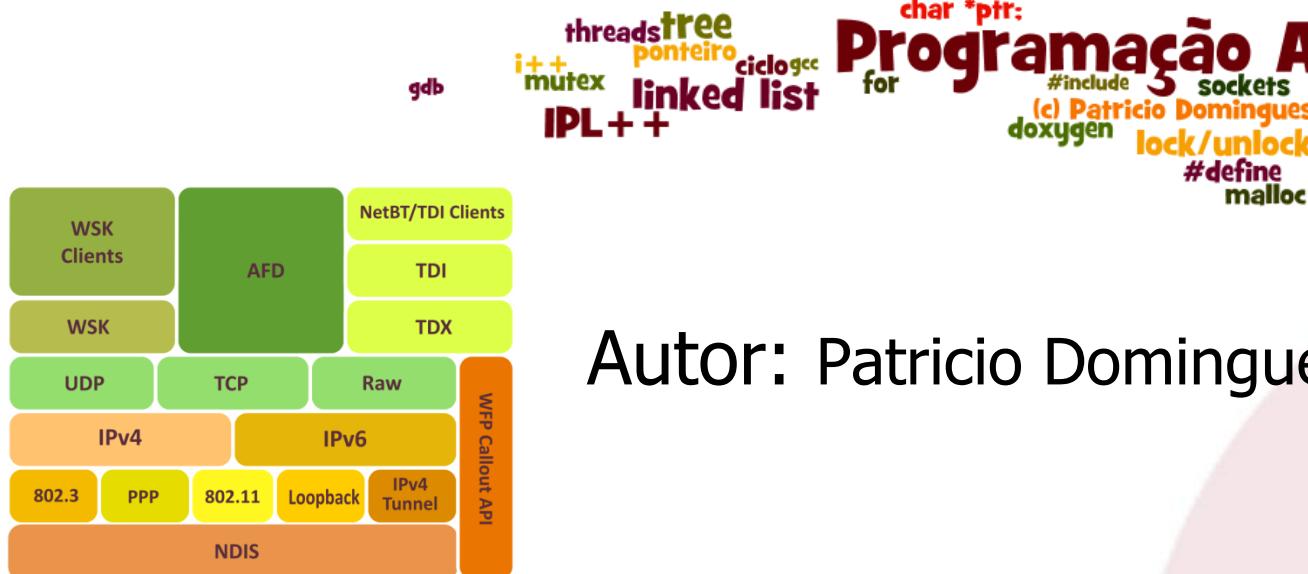
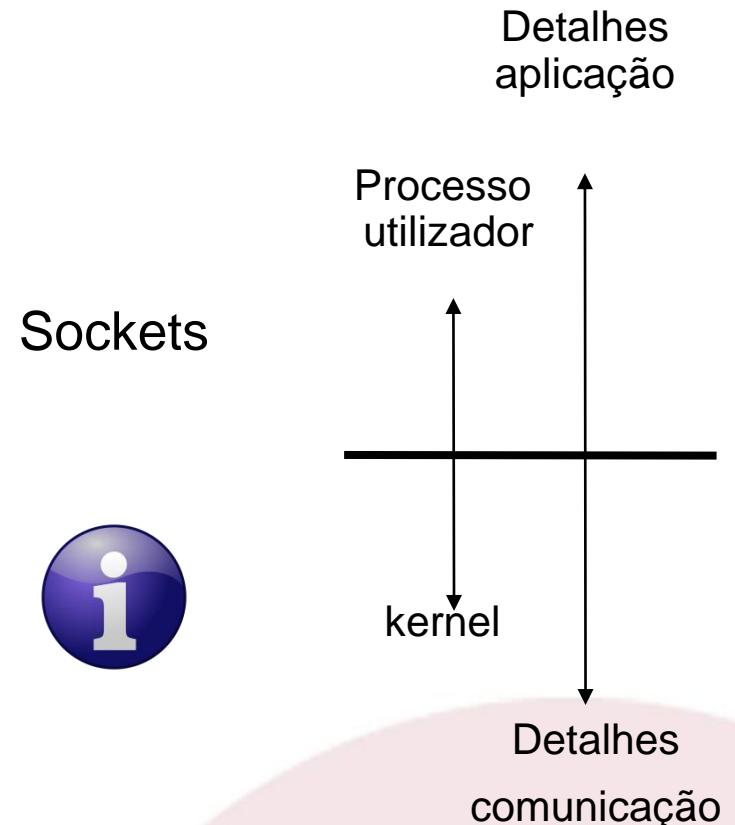
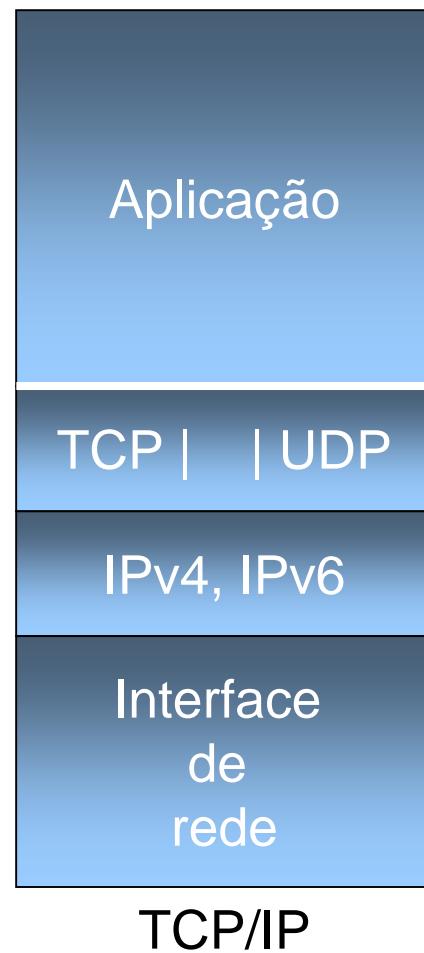


# TCP & UDP



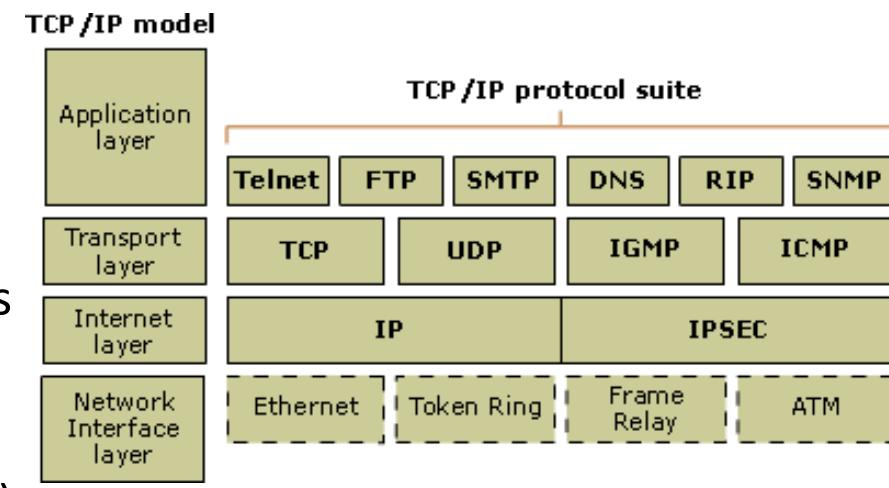
Autor: Patrício Domingues

# Modelo protocolar TCP/IP vs OSI

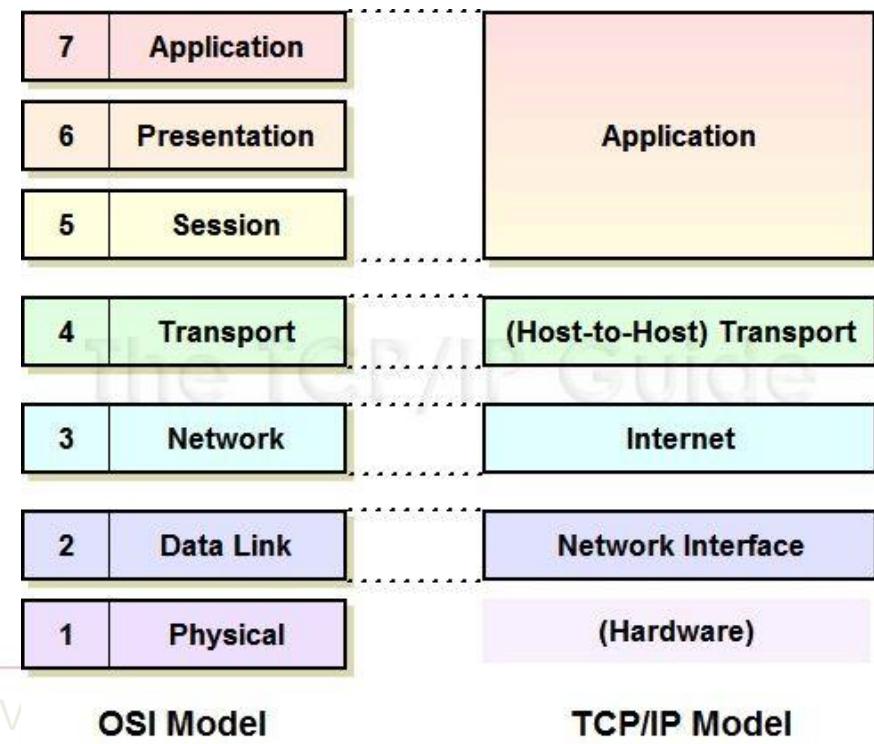


# Camada protocolar TCP/IP

- 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>

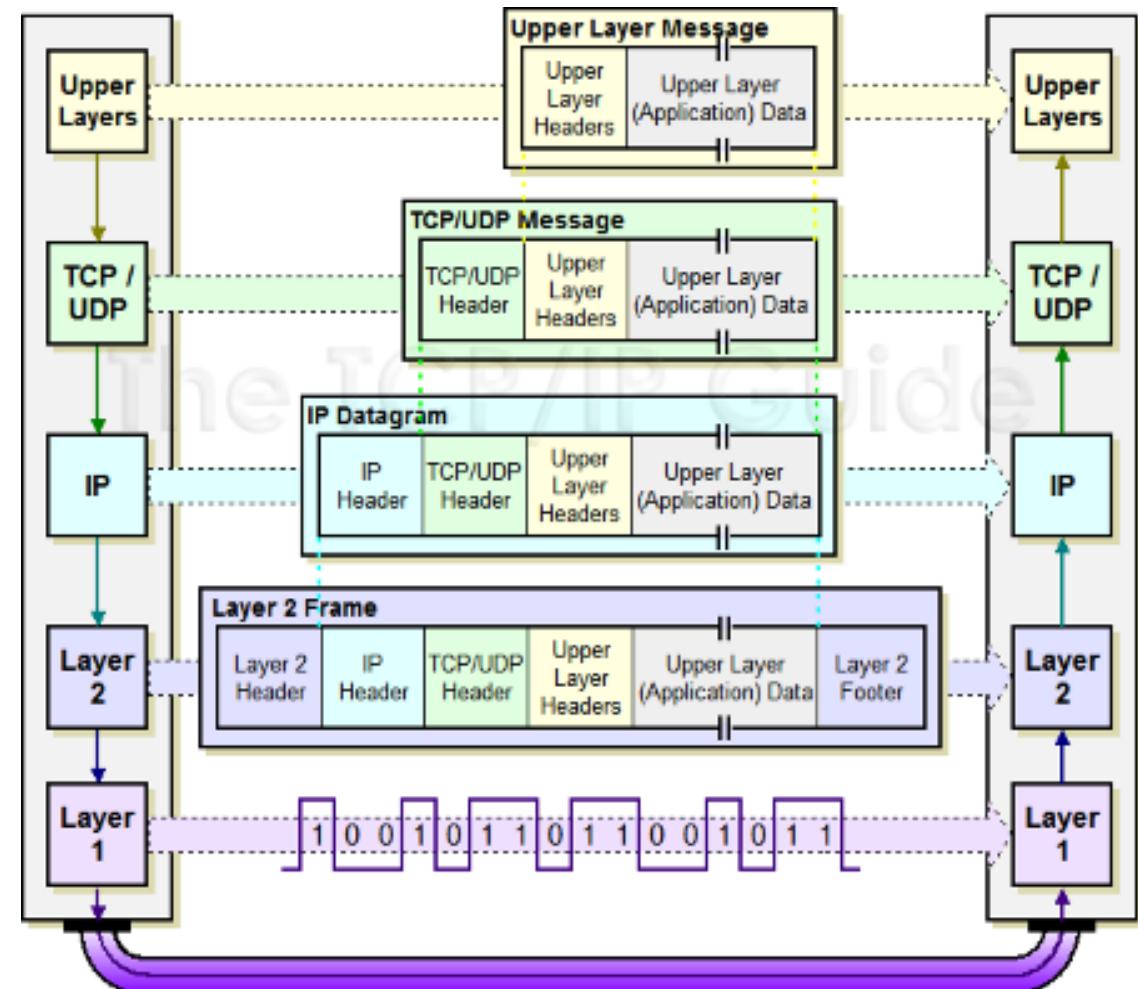


- Dois protocolos na camada de transporte TCP/IP
  - *User Datagram Protocol (UDP)*
    - Não orientado à ligação (envio de mensagens)
    - Não garante a entrega da informação
    - Não garante a ordem de entrega das mensagens
    - Especificação UDP – RFC 768 [Postel 1980]
  - *Transmission Control 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=Vdc8TCESlg8>)



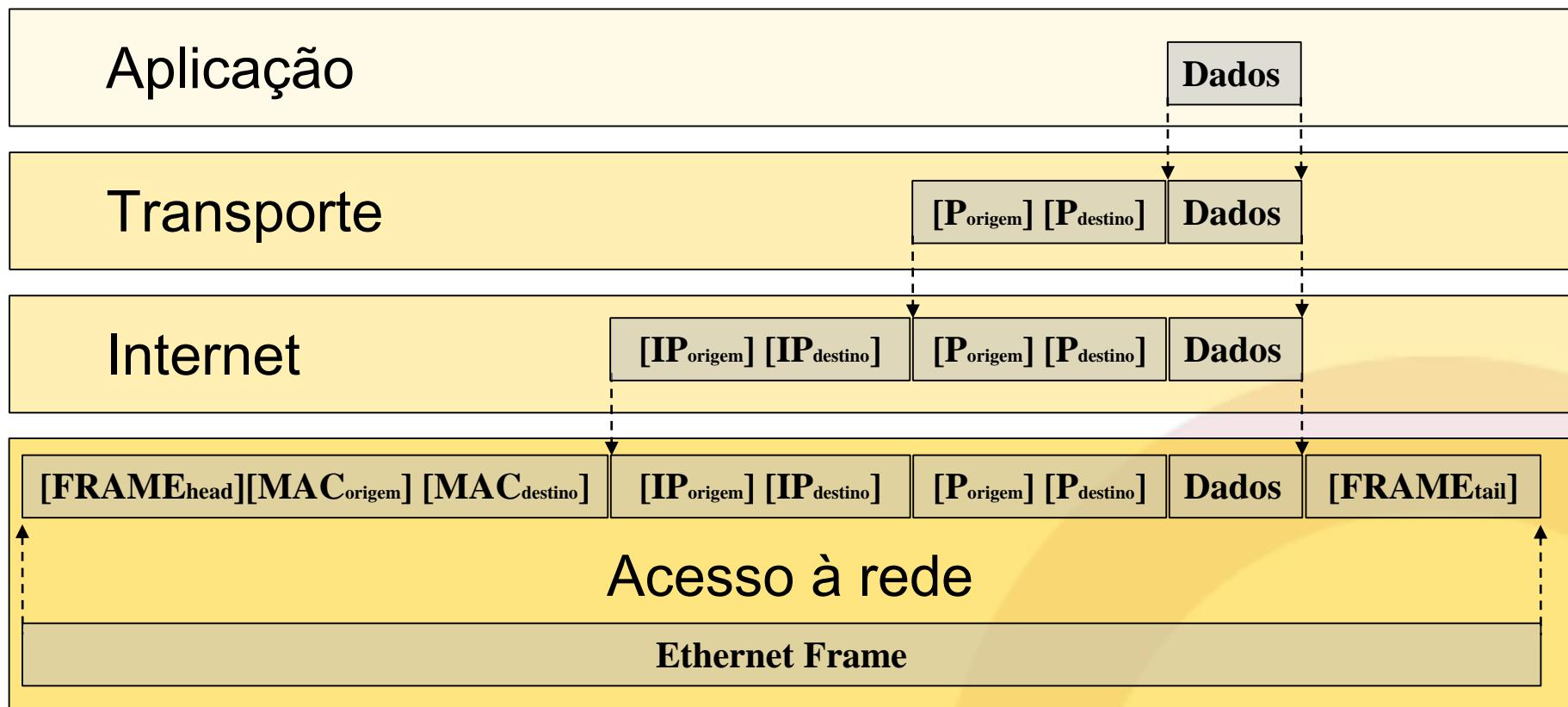
# As camadas protocolares

- Uma mensagem enviada por uma aplicação desce pelas camadas do emissor, subindo no receptor 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)



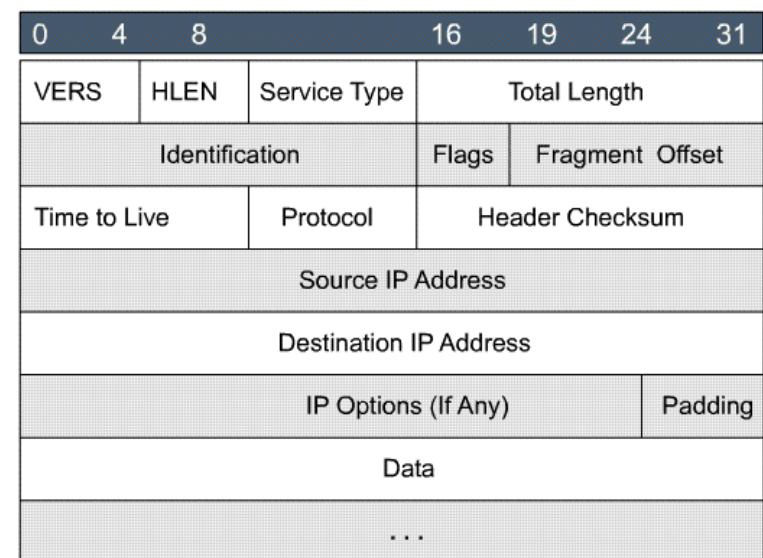
# Pilha protocolar TCP/IP

- 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

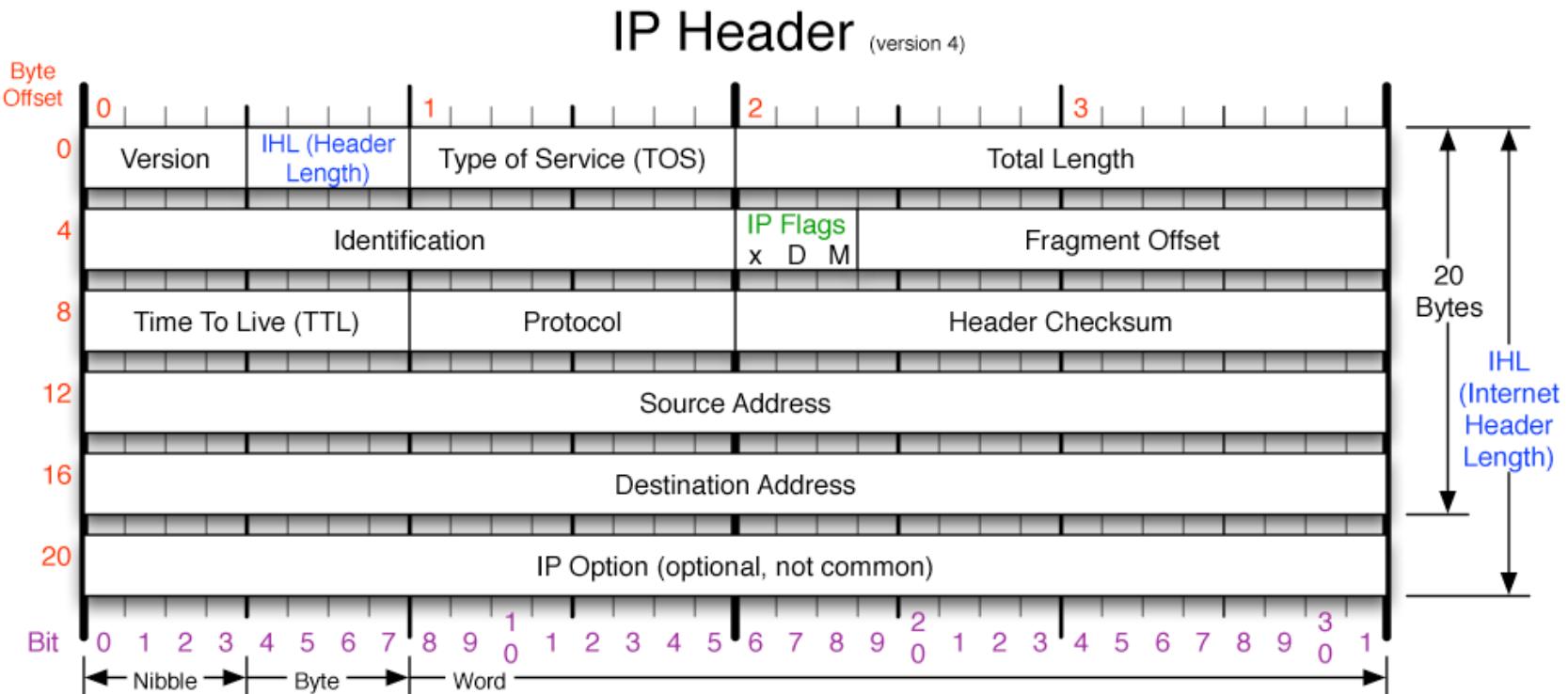


# Cabeçalho IPv4

- O 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



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

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

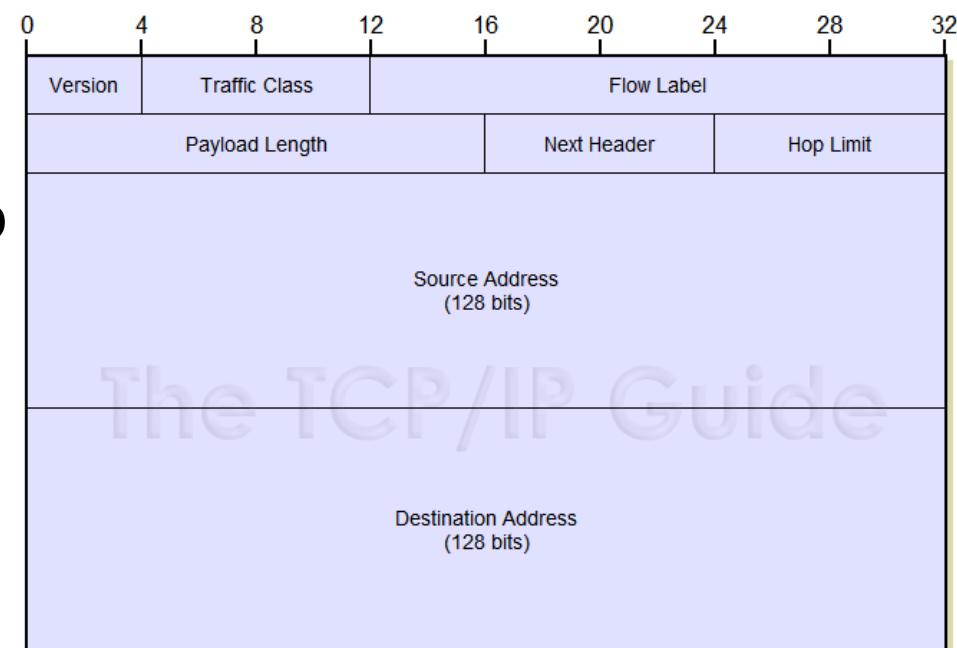
**IP Flags**  
**x D M**  
x 0x80 reserved (evil bit)  
D 0x40 Do Not Fragment  
M 0x20 More Fragments follow

RFC 791

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

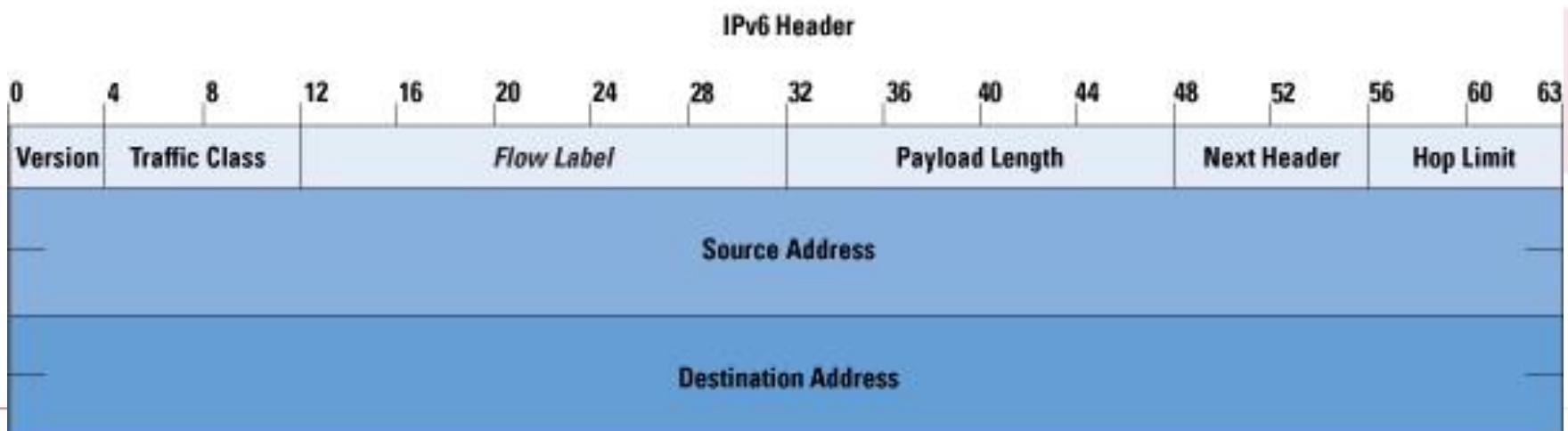
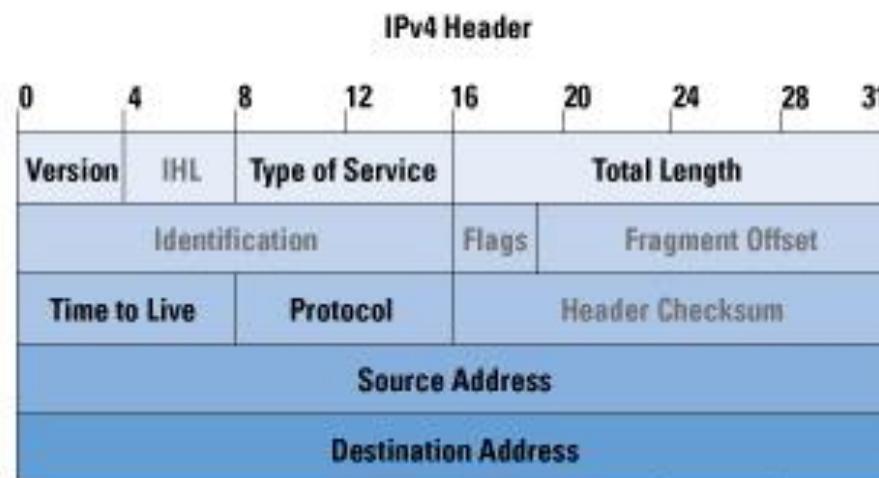
# Cabeçalho IPv6

- 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 tem 16 bytes
  - Exemplo:  
[2001:0db8:85a3:0042:1000:8a2e:0370:7334](http://2001:0db8:85a3:0042:1000:8a2e:0370:7334)
- Um endereço IPv4 tem 4 bytes
  - Exemplo: [192.168.200.5](http://192.168.200.5)



# Cabeçalho IPv4 vs IPv6

- Fonte: [http://www.cisco.com/web/about/ac123/ac147/images/ijp/ijp\\_9-3/93\\_ipv6\\_fig1\\_lg.jpg](http://www.cisco.com/web/about/ac123/ac147/images/ijp/ijp_9-3/93_ipv6_fig1_lg.jpg)



# Endereço IP

## ■ Limitações do endereço IP

- O protocolo de rede (camada 3) IP recorre a endereços IP para a identificação de uma máquina na rede
- Os protocolos de transporte fazem uso da camada IP
  - Uso do endereço IP para identificação das máquinas
- Um endereço IP apenas identifica uma máquina
  - Numa dada máquina podem existir várias aplicações a fazerem uso da rede
    - Navegador (*browser*), cliente email, cliente messenger, etc.
    - Mais, um navegador pode ter várias janelas abertas
  - Como identificar uma aplicação?
    - Conceito de **porto**



Porto >>

# Conceito de porto

## ■ Porto

- Identificador numérico (inteiro) com 16 bits
  - 16 bits pelo que existem  $2^{16}$  (65536) portos diferentes
- Deste modo, uma aplicação é identificada pelo:
  - Endereço IP
  - Por um porto (um de 65536)
- Uma dada máquina pode manter várias ligações em simultâneo com outras máquinas
  - IP + porto

## ■ Analogia

- “Porto” é similar a uma extensão de uma central telefónica
  - IP é como o número de telefone central



# 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)
  - 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



**IPL**

escola superior de tecnologia e gestão  
Instituto Politécnico de Leiria

# • Portos comuns

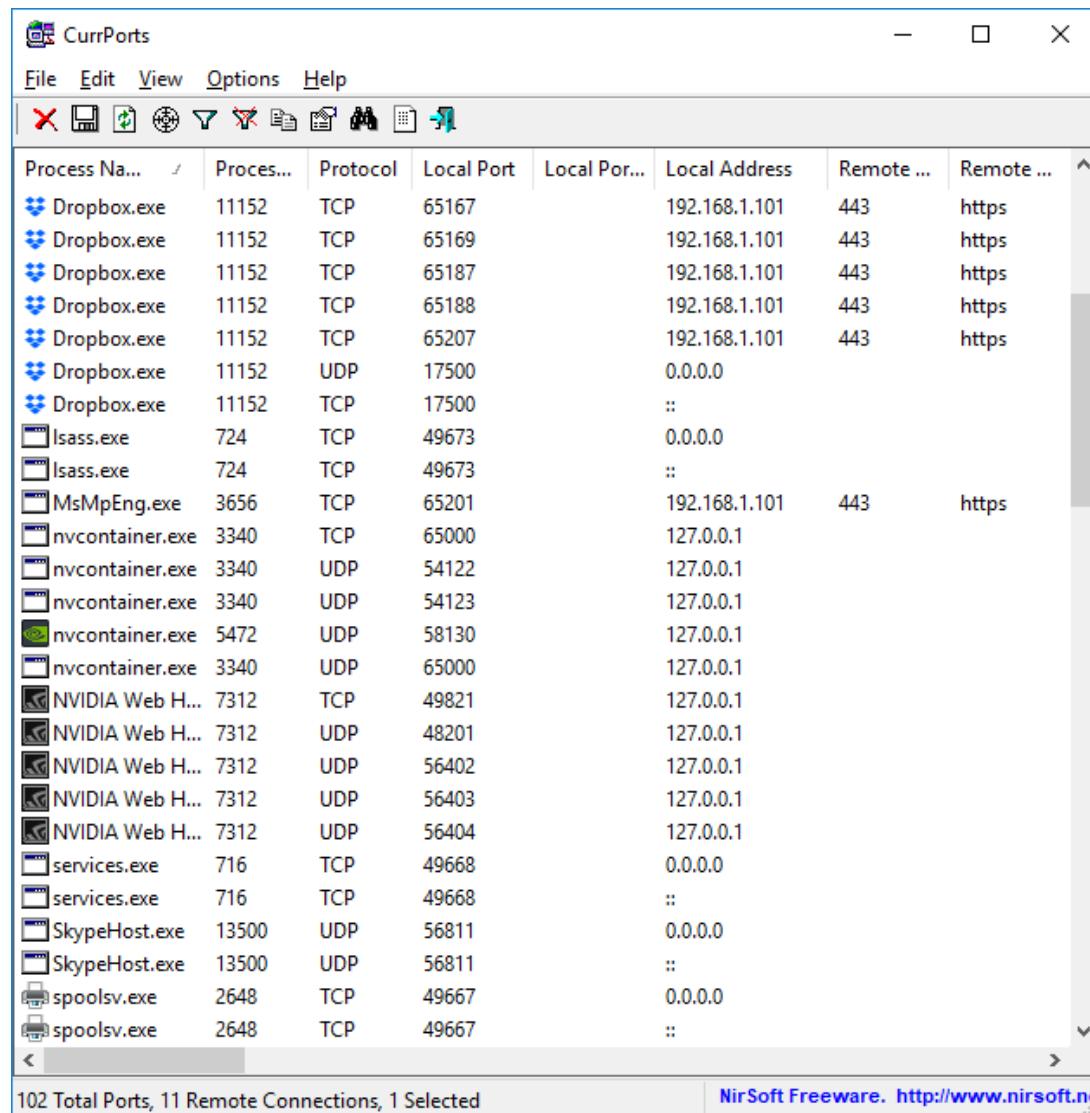
## – UDP e TCP

TCP/UDP Port Numbers			
7 Echo	554 RTSP	2745 Bagle.H	6891-6901 Windows Live
19 Chargen	546-547 DHCPv6	2967 Symantec AV	6970 Quicktime
20-21 FTP	560 rmonitor	3050 Interbase DB	7212 GhostSurf
22 SSH/SCP	563 NNTP over SSL	3074 XBOX Live	7648-7649 CU-SeeMe
23 Telnet	587 SMTP	3124 HTTP Proxy	8000 Internet Radio
25 SMTP	591 FileMaker	3127 MyDoom	8080 HTTP Proxy
42 WINS Replication	593 Microsoft DCOM	3128 HTTP Proxy	8086-8087 Kaspersky AV
43 WHOIS	631 Internet Printing	3222 GLBP	8118 Privoxy
49 TACACS	636 LDAP over SSL	3260 iSCSI Target	8200 VMware Server
53 DNS	639 MSDP (PIM)	3306 MySQL	8500 Adobe ColdFusion
67-68 DHCP/BOOTP	646 LDP (MPLS)	3389 Terminal Server	8767 TeamSpeak
69 TFTP	691 MS Exchange	3689 iTunes	8866 Bagle.B
70 Gopher	860 iSCSI	3690 Subversion	9100 HP JetDirect
79 Finger	873 rsync	3724 World of Warcraft	9101-9103 Bacula
80 HTTP	902 VMware Server	3784-3785 Ventrilo	9119 MXit
88 Kerberos	989-990 FTP over SSL	4333 mSQL	9800 WebDAV
102 MS Exchange	993 IMAP4 over SSL	4444 Blaster	9898 Dabber
110 POP3	995 POP3 over SSL	4664 Google Desktop	9988 Rbot/Spybot
113 Ident	1025 Microsoft RPC	4672 eMule	9999 Urchin
119 NNTP (Usenet)	1026-1029 Windows Messenger	4899 Radmin	10000 Webmin
123 NTP	1080 SOCKS Proxy	5000 UPnP	10000 BackupExec
135 Microsoft RPC	1080 MyDoom	5001 Slingbox	10113-10116 NetIQ
137-139 NetBIOS	1194 OpenVPN	5001 iperf	11371 OpenPGP
143 IMAP4	1214 Kazaa	5004-5005 RTP	12035-12036 Second Life
161-162 SNMP	1241 Nessus	5050 Yahoo! Messenger	12345 NetBus
177 XDMCP	1311 Dell OpenManage	5060 SIP	13720-13721 NetBackup
179 BGP	1337 WASTE	5190 AIM/ICQ	14567 Battlefield
201 AppleTalk	1433-1434 Microsoft SQL	5222-5223 XMPP/Jabber	15118 Dipnet/Oddbob
264 BGMP	1512 WINS	5432 PostgreSQL	19226 AdminSecure
318 TSP	1589 Cisco VQP	5500 VNC Server	19638 Ensim
381-383 HP Openview	1701 L2TP	5554 Sasser	20000 Usermin
389 LDAP	1723 MS PPTP	5631-5632 pcAnywhere	24800 Synergy
411-412 Direct Connect	1725 Steam	5800 VNC over HTTP	25999 Xfire
443 HTTP over SSL	1741 CiscoWorks 2000	5900+ VNC Server	27015 Half-Life
445 Microsoft DS	1755 MS Media Server	6000-6001 X11	27374 Sub7
464 Kerberos	1812-1813 RADIUS	6112 Battle.net	28960 Call of Duty
465 SMTP over SSL	1863 MSN	6129 DameWare	31337 Back Orifice
497 Retrospect	1985 Cisco HSRP	6257 WinMX	33434+ traceroute
500 ISAKMP	2000 Cisco SCCP	6346-6347 Gnutella	Legend
512 rexec	2002 Cisco ACS	6500 GameSpy Arcade	Chat
513 rlogin	2049 NFS	6566 SANE	Encrypted
514 syslog	2082-2083 cPanel	6588 AnalogX	Gaming
515 LPD/LPR	2100 Oracle XDB	6665-6669 IRC	Malicious
520 RIP	2222 DirectAdmin	6679/6697 IRC over SSL	Peer to Peer
521 RIPng (IPv6)	2302 Halo	6699 Napster	Streaming
540 UUCP	2483-2484 Oracle DB	6881-6999 BitTorrent	

IANA port assignments published at <http://www.iana.org/assignments/port-numbers>

# 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>



The screenshot shows the CurrPorts application window displaying a list of active network ports. The columns in the table are: Process Name, Process ID, Protocol, Local Port, Local Port Range, Local Address, Remote IP, and Remote Port Range. The application interface includes a menu bar (File, Edit, View, Options, Help) and a toolbar with various icons.

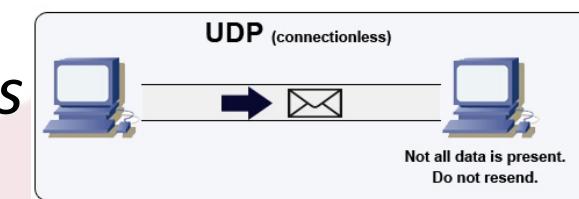
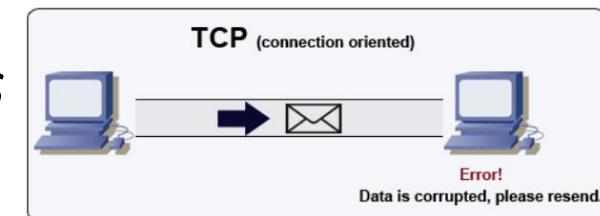
Process Name	Proces...	Protocol	Local Port	Local Port Range	Local Address	Remote IP	Remote Port Range
Dropbox.exe	11152	TCP	65167		192.168.1.101	443	https
Dropbox.exe	11152	TCP	65169		192.168.1.101	443	https
Dropbox.exe	11152	TCP	65187		192.168.1.101	443	https
Dropbox.exe	11152	TCP	65188		192.168.1.101	443	https
Dropbox.exe	11152	TCP	65207		192.168.1.101	443	https
Dropbox.exe	11152	UDP	17500		0.0.0.0		
Dropbox.exe	11152	TCP	17500		::		
lsass.exe	724	TCP	49673		0.0.0.0		
lsass.exe	724	TCP	49673		::		
MsMpEng.exe	3656	TCP	65201		192.168.1.101	443	https
nvcontainer.exe	3340	TCP	65000		127.0.0.1		
nvcontainer.exe	3340	UDP	54122		127.0.0.1		
nvcontainer.exe	3340	UDP	54123		127.0.0.1		
nvcontainer.exe	5472	UDP	58130		127.0.0.1		
nvcontainer.exe	3340	UDP	65000		127.0.0.1		
NVIDIA Web H...	7312	TCP	49821		127.0.0.1		
NVIDIA Web H...	7312	UDP	48201		127.0.0.1		
NVIDIA Web H...	7312	UDP	56402		127.0.0.1		
NVIDIA Web H...	7312	UDP	56403		127.0.0.1		
NVIDIA Web H...	7312	UDP	56404		127.0.0.1		
services.exe	716	TCP	49668		0.0.0.0		
services.exe	716	TCP	49668		::		
SkypeHost.exe	13500	UDP	56811		0.0.0.0		
SkypeHost.exe	13500	UDP	56811		::		
spoolsv.exe	2648	TCP	49667		0.0.0.0		
spoolsv.exe	2648	TCP	49667		::		

102 Total Ports, 11 Remote Connections, 1 Selected

NirSoft Freeware. <http://www.nirsoft.net>

# Camada de transporte

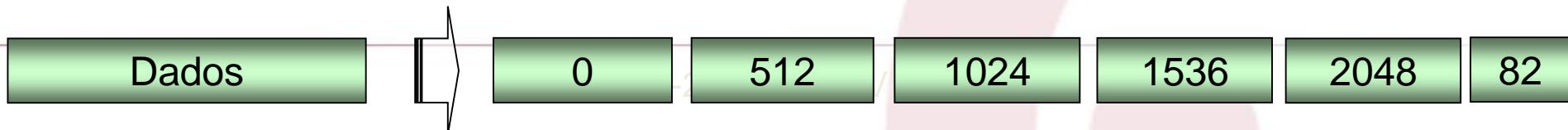
- Protocolos de transporte
  - TCP
    - mensagens transportadas por *segmentos*
  - UDP
    - mensagens transportadas por *datagramas*



Protocolo TCP >>

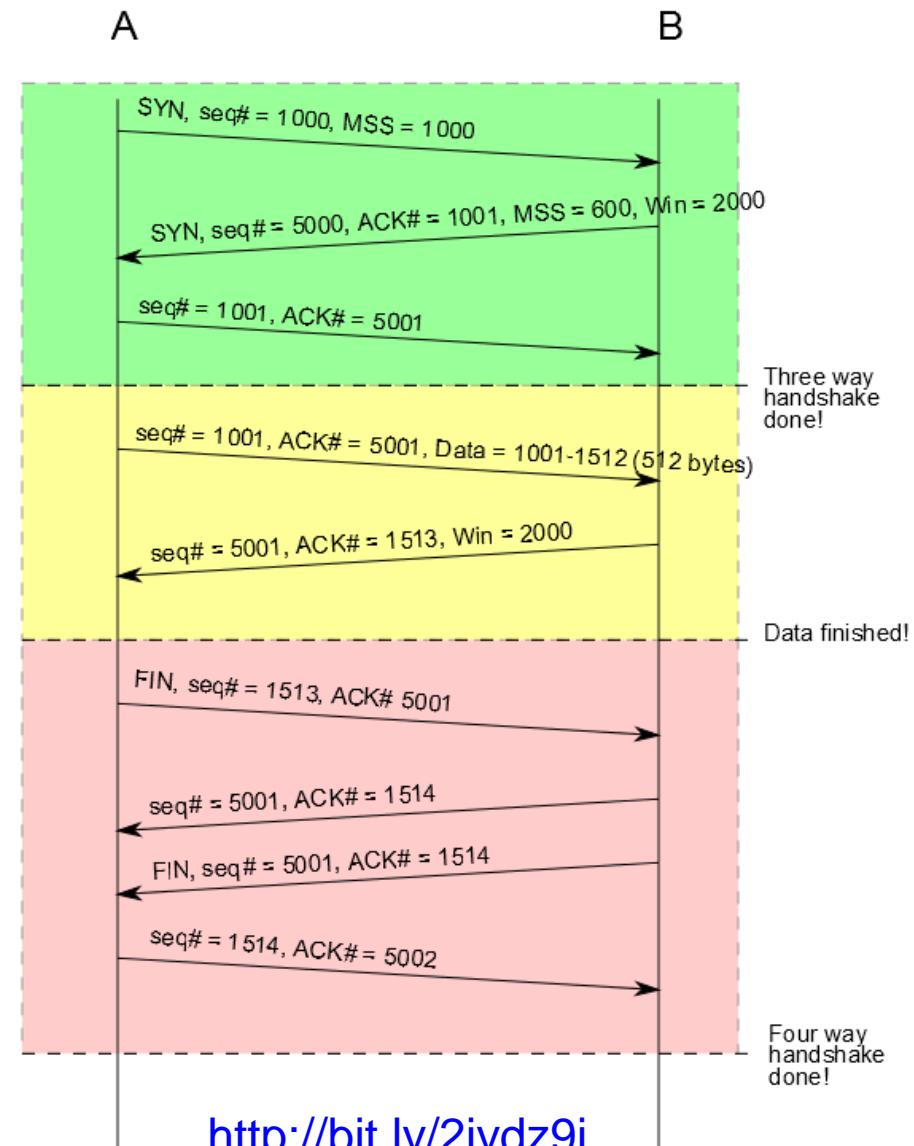
# Protocolo TCP: Introdução (1)

- 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



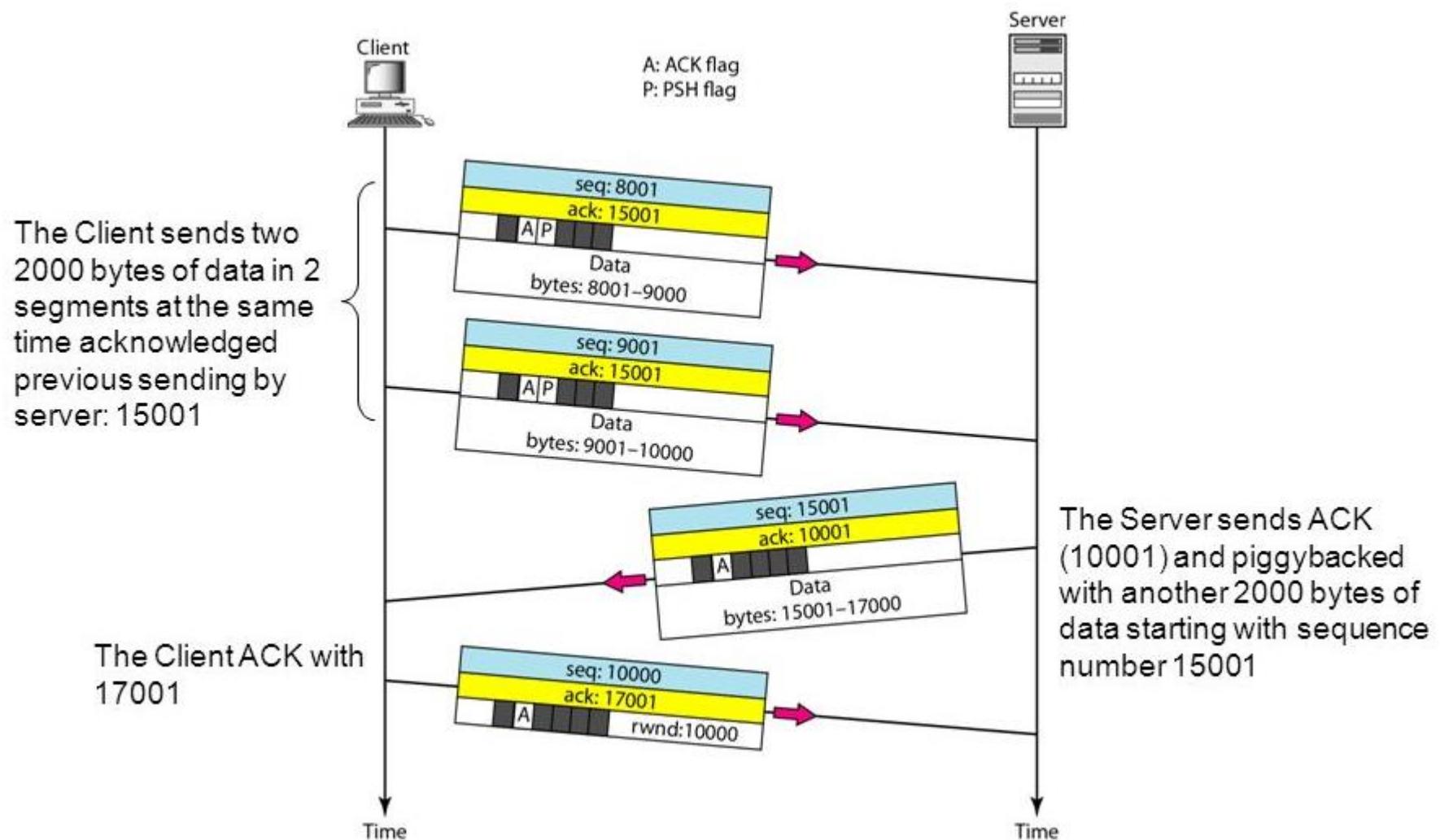
# Protocolo TCP: Introdução (2)

- 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 **ACKnowledge**
- O TCP garante a sequencialidade na entrega da informação
  - Numera sequencialmente os segmentos que envia



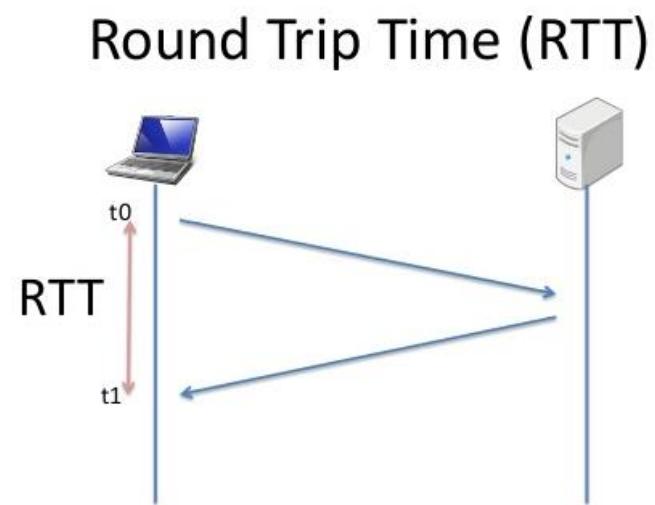
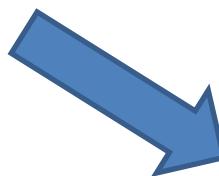
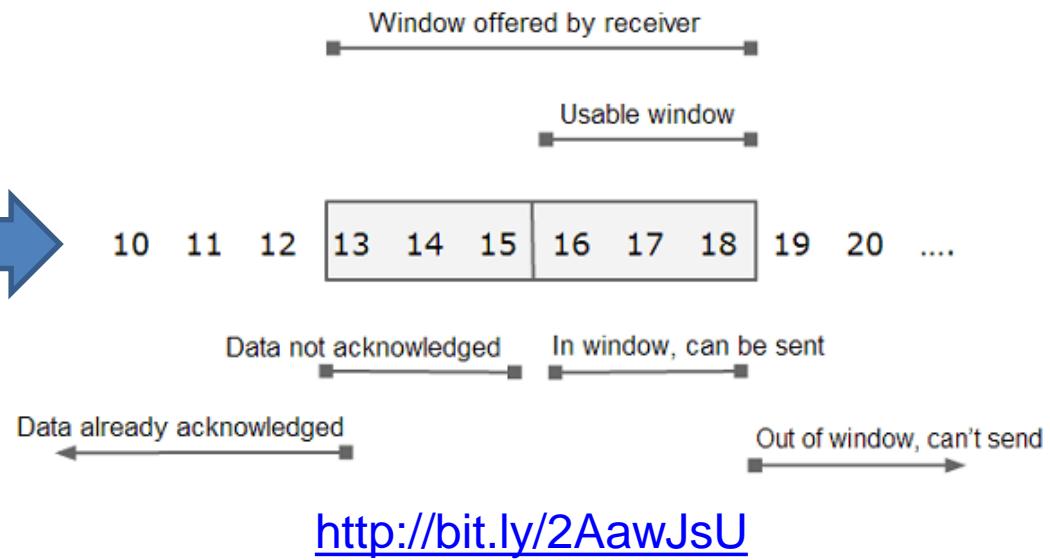
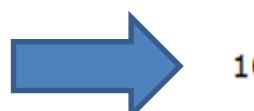
<http://bit.ly/2iydz9j>

- Segmentação de dados



# Protocolo TCP: Introdução (4)

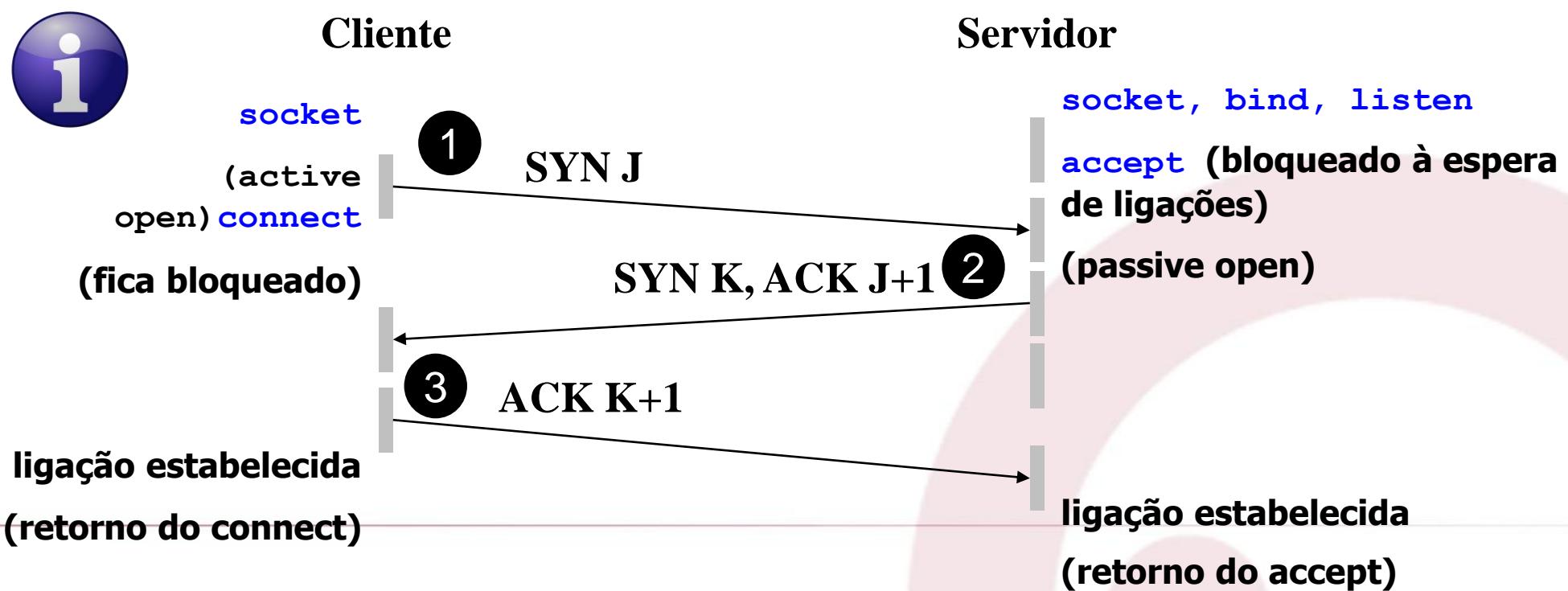
- O protocolo TCP adapta-se dinamicamente às condições da ligação
  - Mecanismo de janela (anuncia o número de bytes que está disposto a aceitar)
  - Estimação RTT (round-trip time) ou RTD (round-trip delay time)
    - Tempo que demora a enviar um segmento e receber a sua confirmação
  - Controlo de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



# Protocolo TCP: 3WHS (1)

## ■ Estabelecimento da ligação

- Algoritmo “*Three-Way Handshake*” (3WHS)
- Três etapas para o estabelecimento da ligação
  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



## ■ Legenda

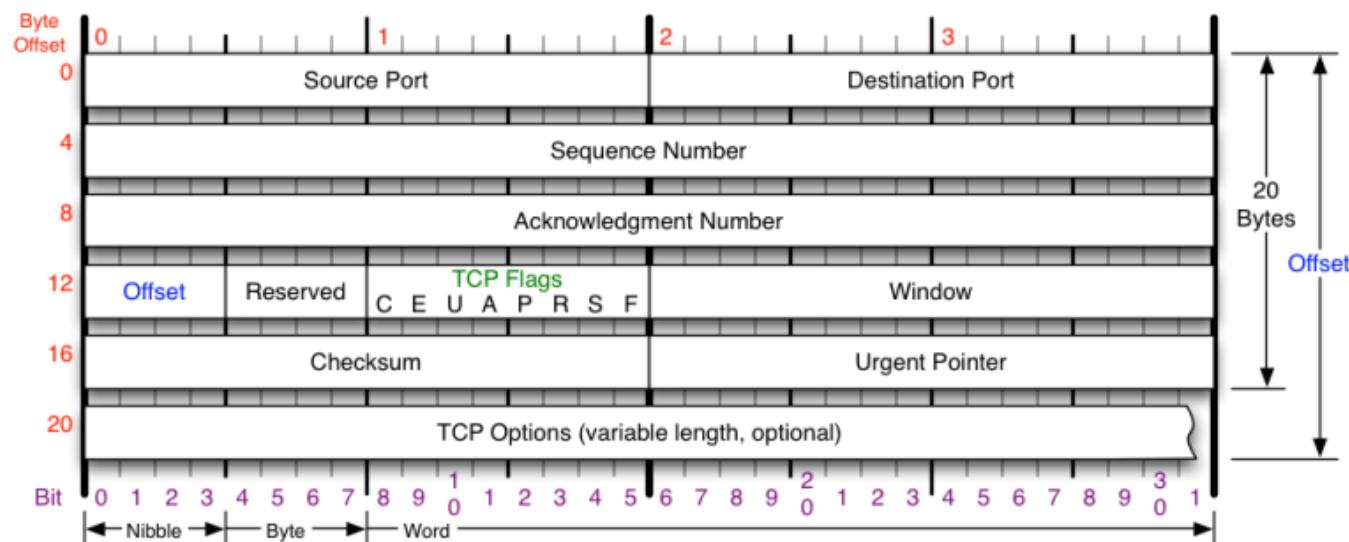
- SYN
  - Segmento para pedido de ligação **SYN**chronize
- ACK
  - Segmento de confirmação **ACK**nowledge
- J
  - número da sequência inicial do cliente a ser utilizado na comunicação
- K
  - número da sequência inicial do servidor a ser utilizado na comunicação
  - O TCP numera sequencialmente os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



# 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	Congestion Notification	TCP Options	Offset																																																																																												
<table border="1"> <tr><td>C</td><td>E</td><td>U</td><td>A</td><td>P</td><td>R</td><td>S</td><td>F</td></tr> <tr><td>Congestion Window Reduced (CWR)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>E 0x80</td><td>0x40</td><td>0x20</td><td>0x10</td><td>0x08</td><td>0x04</td><td>0x02</td><td>0x01</td></tr> <tr><td>ECN Echo (ECE)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>U 0x02</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>A 0x01</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>P 0x00</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>R 0x00</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>S 0x00</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>F 0x00</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </table>	C	E	U	A	P	R	S	F	Congestion Window Reduced (CWR)								E 0x80	0x40	0x20	0x10	0x08	0x04	0x02	0x01	ECN Echo (ECE)								U 0x02								A 0x01								P 0x00								R 0x00								S 0x00								F 0x00								<p>ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.</p> <table border="1"> <tr><td>Packet State</td><td>DSB</td><td>ECN bits</td></tr> <tr><td>Syn</td><td>0 0</td><td>1 1</td></tr> <tr><td>Syn-Ack</td><td>0 0</td><td>0 1</td></tr> <tr><td>Ack</td><td>0 1</td><td>0 0</td></tr> </table>	Packet State	DSB	ECN bits	Syn	0 0	1 1	Syn-Ack	0 0	0 1	Ack	0 1	0 0	<ul style="list-style-type: none"> <li>0 End of Options List</li> <li>1 No Operation (NOP, Pad)</li> <li>2 Maximum segment size</li> <li>3 Window Scale</li> <li>4 Selective ACK ok</li> <li>8 Timestamp</li> </ul>	<p>Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.</p>
C	E	U	A	P	R	S	F																																																																																								
Congestion Window Reduced (CWR)																																																																																															
E 0x80	0x40	0x20	0x10	0x08	0x04	0x02	0x01																																																																																								
ECN Echo (ECE)																																																																																															
U 0x02																																																																																															
A 0x01																																																																																															
P 0x00																																																																																															
R 0x00																																																																																															
S 0x00																																																																																															
F 0x00																																																																																															
Packet State	DSB	ECN bits																																																																																													
Syn	0 0	1 1																																																																																													
Syn-Ack	0 0	0 1																																																																																													
Ack	0 1	0 0																																																																																													
		Checksum	RFC 793																																																																																												
		Checksum of entire TCP segment and pseudo header (parts of IP header)	Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.																																																																																												

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



## ■ 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



## ■ 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 - flags

- Flags de um segmento TCP
  - Exemplo

Header length: 20 bytes

Flags: 0x11 (FIN, ACK)

000.	.....	.... = Reserved: Not set
...0	.....	.... =Nonce: Not set
....	0...	.... = Congestion Window Reduced (CWR): Not set
....	.0..	.... = ECN-Echo: Not set
....	..0.	.... = Urgent: Not set ←
....	...1	.... = Acknowledgement: Set ←
....	.... 0...	= Push: Not set ←
....	.... .0..	= Reset: Not set ←
....	.... ..0.	= Syn: Not set ←
[+]	.... .... .1	= Fin: Set ←

window size: 65888 (scaled)

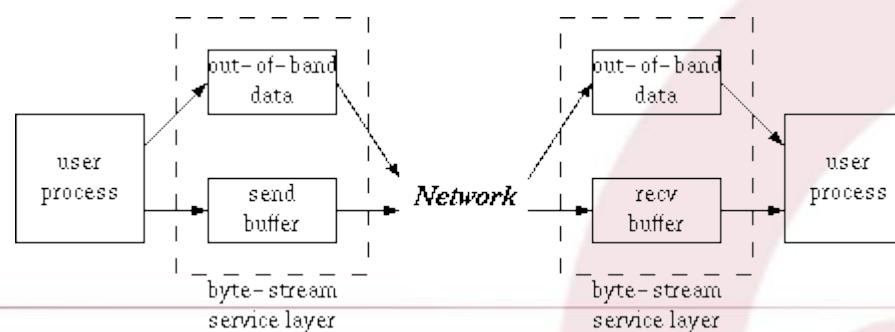
© Packet Crafter, 2011.

# Protocolo TCP: Segmento TCP (3)

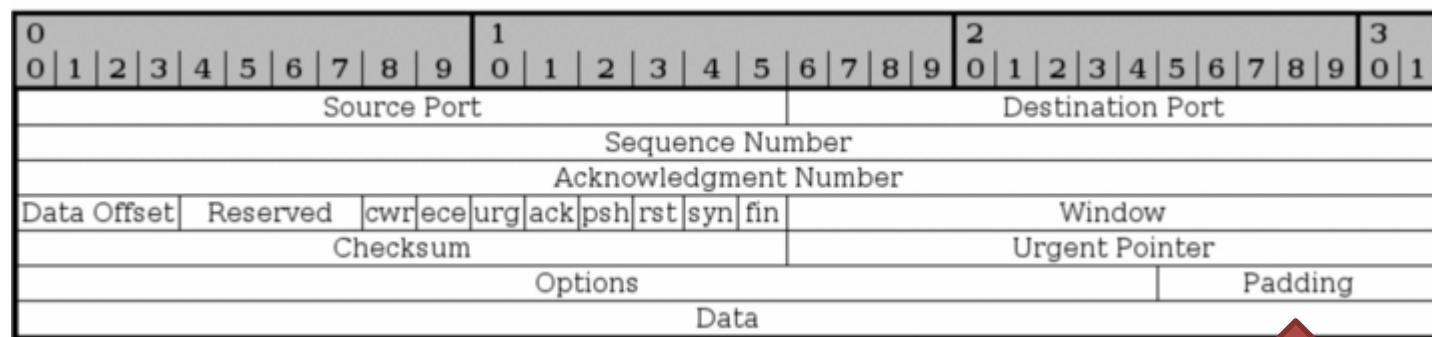
- Janela (16 bits)
  - Controlo de fluxo
  - Número de bytes que o destinatário pode receber sem confirmação (este número está condicionado pela opção WSO)
- Checksum (16 bits)
  - CRC16 para garantir a integridade dos dados do segmento
    - O cálculo do checksum abrange cabeçalho + dados, sendo o campo 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);
```



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

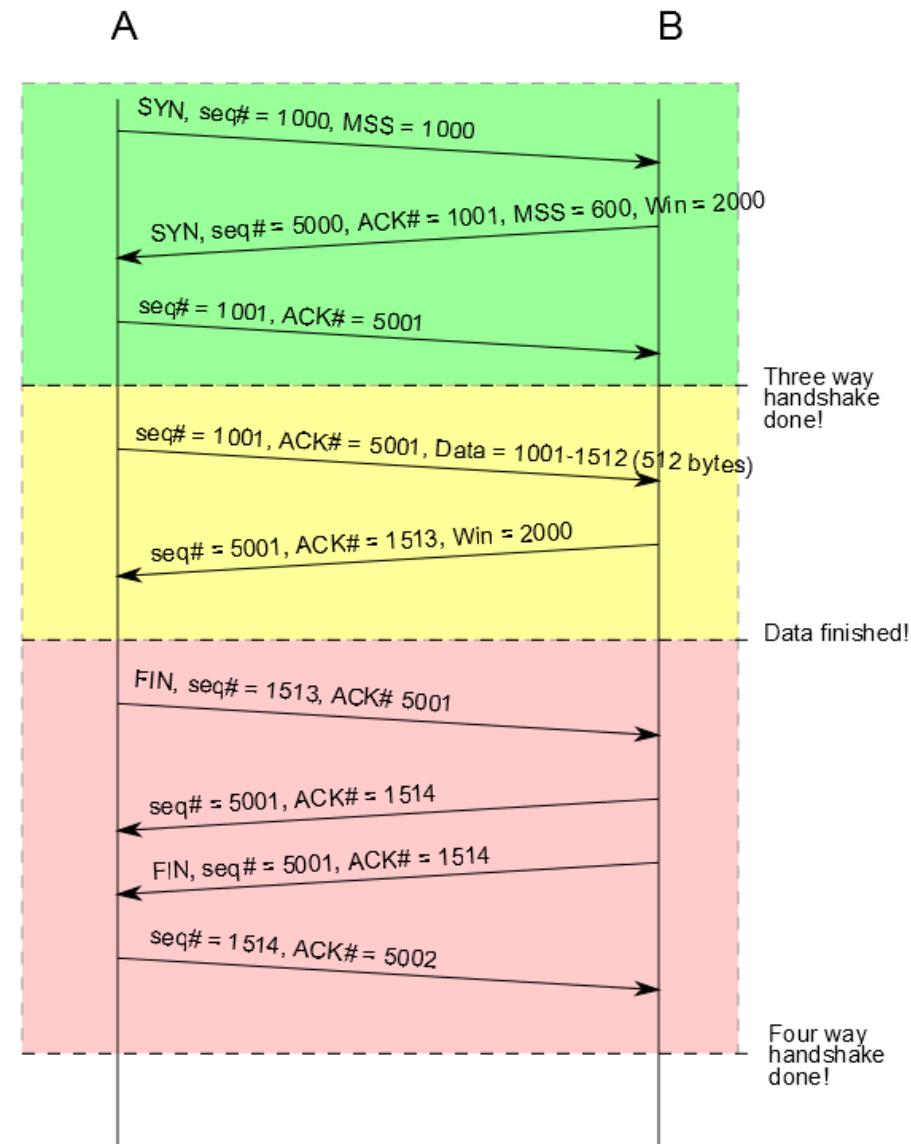


# Padding

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

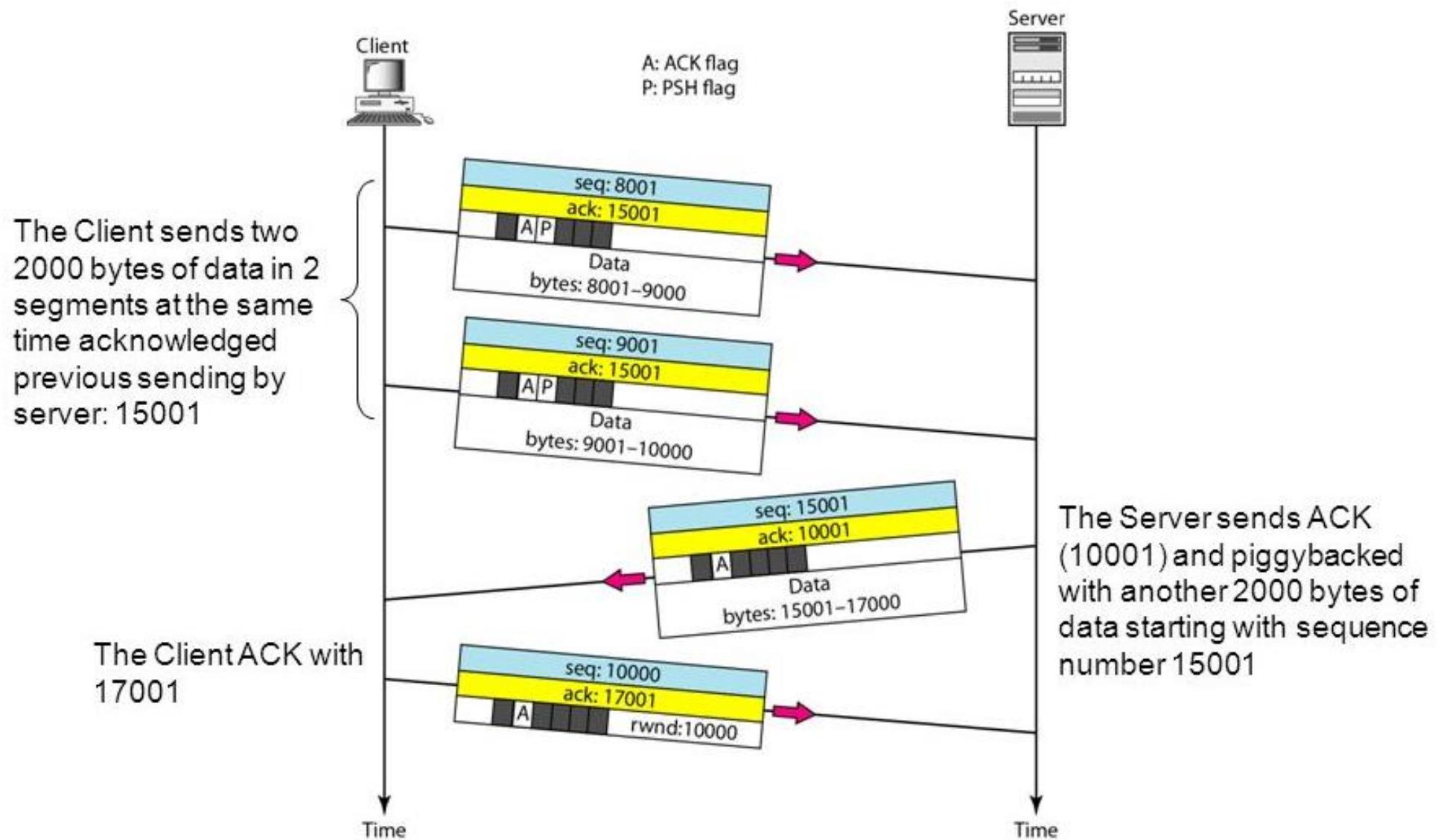
# Protocolo TCP: Troca de dados (1)

- Após o estabelecimento da ligação, podem ser trocados dados entre as entidades comunicantes
  - Cada segmento de dados 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)



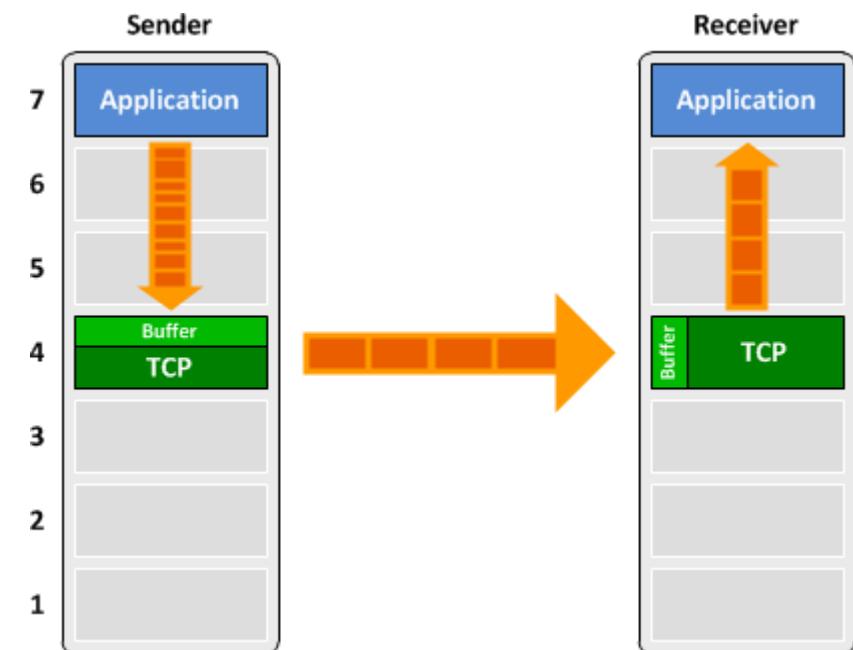
# Protocolo TCP: Troca de dados (2)

- Exemplo



# Protocolo TCP: Buffers (1)

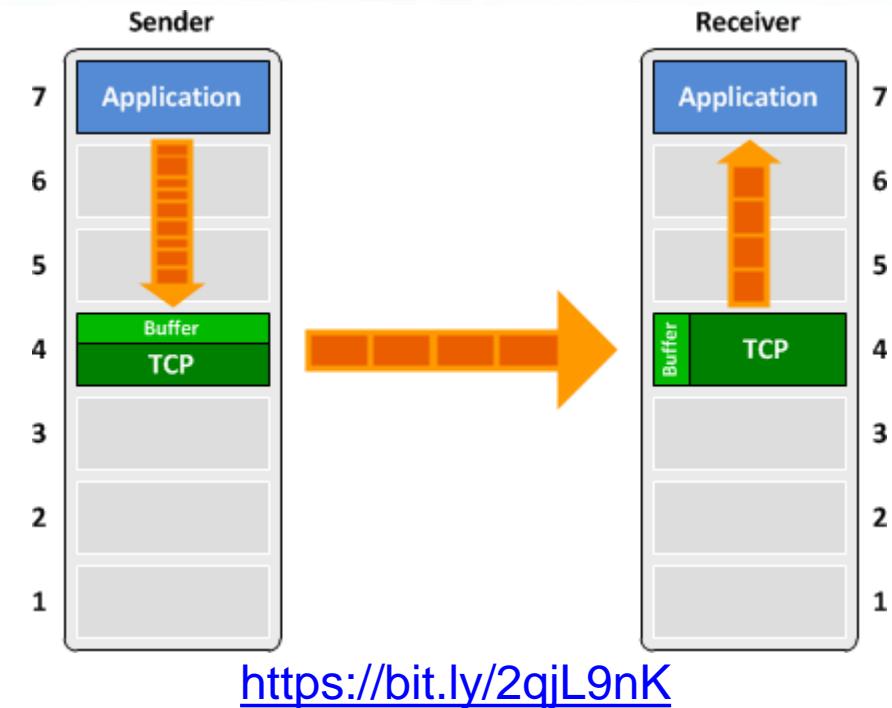
- A pilha protocolar TCP/IP encontra-se integrada no sistema operativo
  - Todos os detalhes da transmissão (ACK, segmentos perdidos, retransmissões, etc.) são tratados pelo TCP de forma “assíncrona”
  - É por esta razão que o desempenho da pilha protocolar difere de SO para SO
- A camada TCP desconhece quando a aplicação irá pedir os dados lidos
- 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 está condicionado pelo tamanho do *buffer*



<https://bit.ly/2qjL9nK>

# Protocolo TCP: Buffers (2)

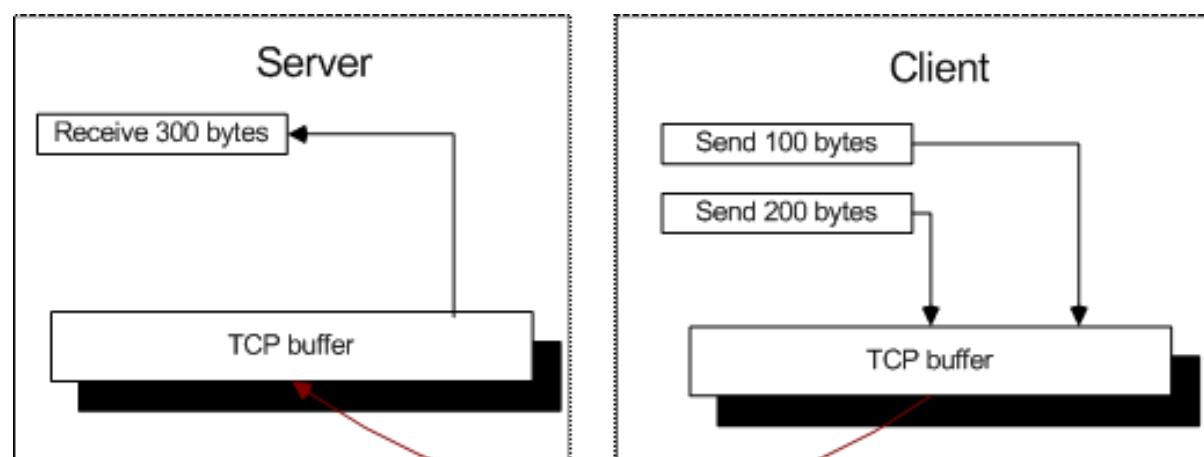
- A aplicação envia dados através de uma chamada ao sistema (**send(...)**, **write(...)**,etc.)
- Os dados são colocados na memória tampão de envio (ainda no emissor), onde permanecem até que sejam confirmados (ACK do receptor)
- A camada TCP só aceita dados da aplicação se ainda houver espaço na memória 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

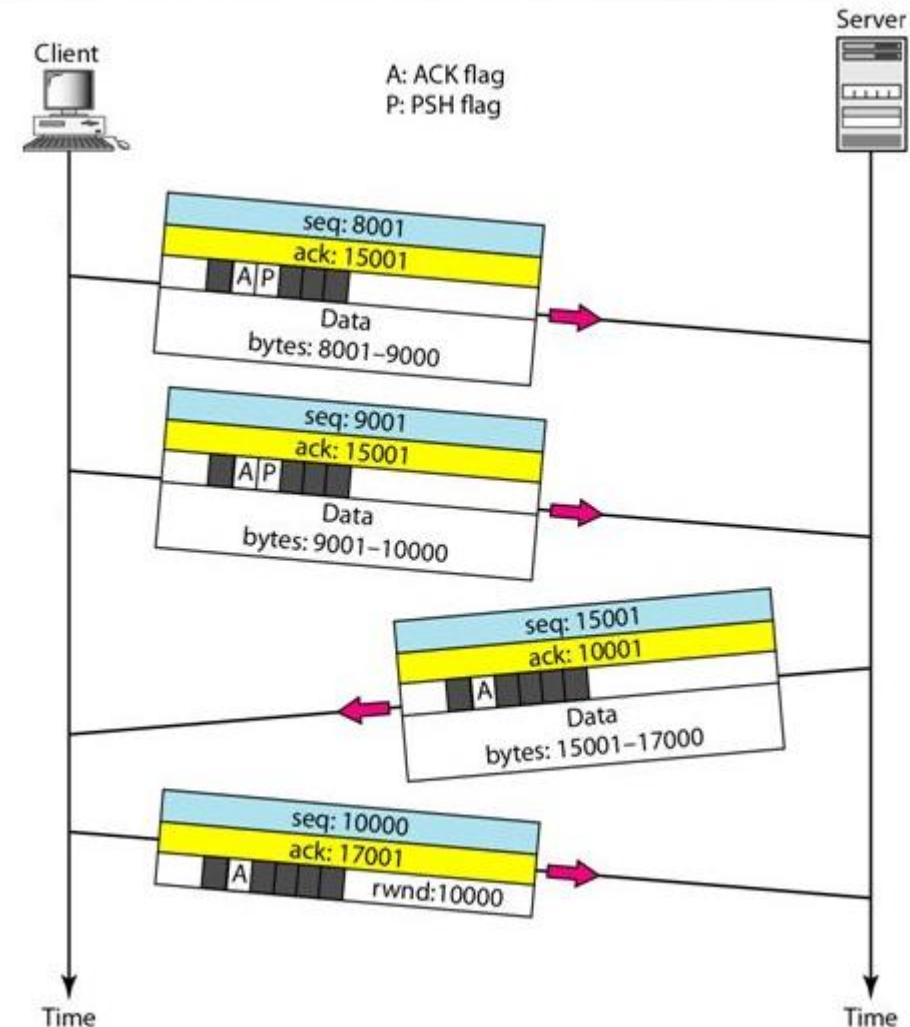


Fonte:

<http://www.codeproject.com/KB/IP/socketmessageboundary.aspx>

# Protocolo TCP: ACK

- Não existe necessariamente um segmento de ACK por cada segmento recebido.
  - O receptor pode através de um ACK confirmar vários segmentos
    - Normalmente o ACK é enviado quando a quantidade de dados recebida é igual ao tamanho definido pela janela
- Um segmento de ACK também pode conter dados
- Se um emissor não recebe um ACK após um determinado intervalo de tempo (MSL), o emissor procede ao reenvio dos dados
  - MSL = Maximum Segment Lifetime
    - Este valor depende da implementação da pilha protocolar, varia entre 30 s a 2 minutos



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

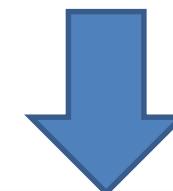
**MSL >>**

# 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  $2 \times \text{MSL}$  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



```
user@ubuntu: ~
File Edit Tabs Help
user@ubuntu:~$ cat /proc/sys/net/ipv4/tcp_fin_timeout
60
```

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

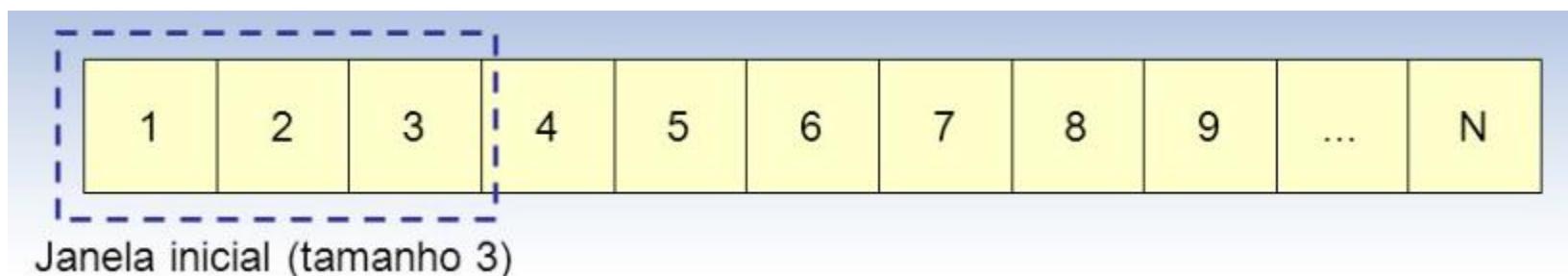


- A maioria das implementações TCP aceitam segmentos fora-de-ordem se ainda dispuserem de espaço na memória tampão de receção
- Logo que o segmento em falta chegue, um único ACK pode confirmar a recepção de todos os segmentos
- **Importante**
  - Os segmentos TCP são entregues pela camada IP
  - A camada IP não é confiável
    - Pode ocorrer perda de pacotes IP
    - Pode ocorrer que pacotes sejam entregues fora da ordem de envio
      - Seguiram rotas diferentes

Janela deslizante >>

# TCP – Janela deslizante (#1)

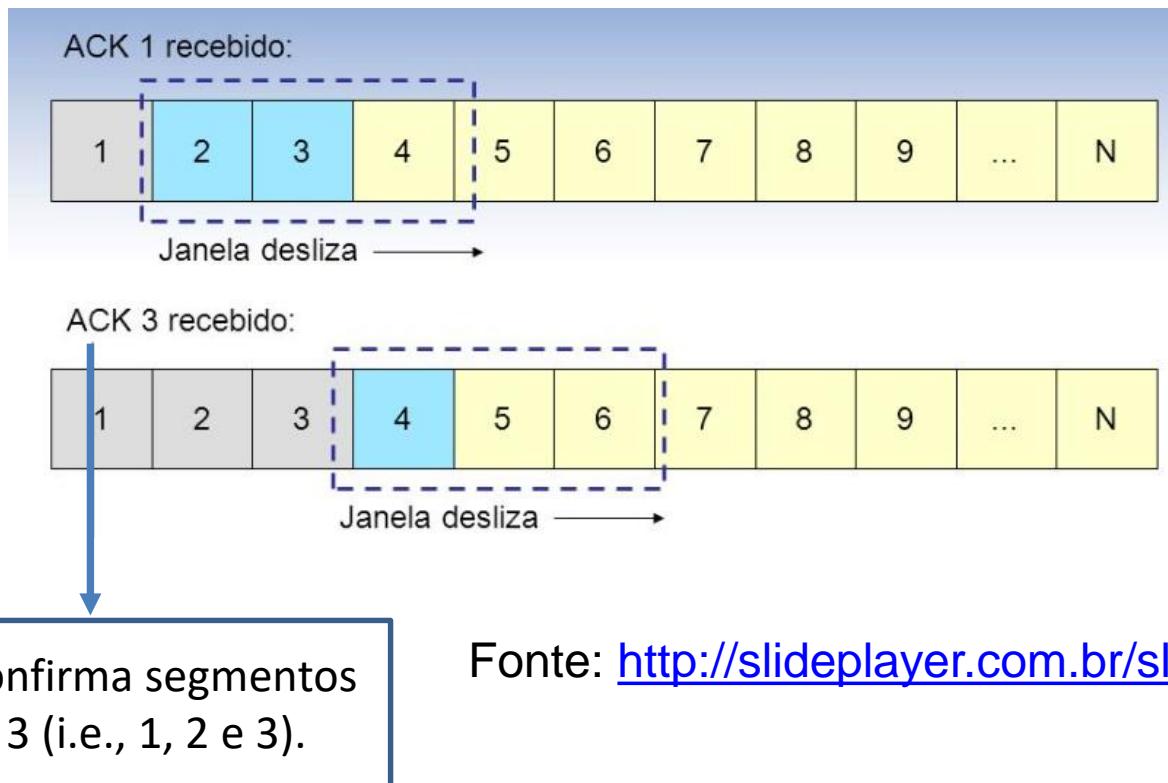
- Protocolo cria uma janela de tamanho fixo



Fonte: <http://slideplayer.com.br/slide/1240062/>

- Um segmento é não confirmado enquanto não tiver chegado um ACK a confirmar a receção
- O TCP pode transmitir todos os segmentos que se encontram na janela antes de receber uma confirmação (ACK)
  - O tamanho da “janela” define o número máximo de segmentos por confirmar

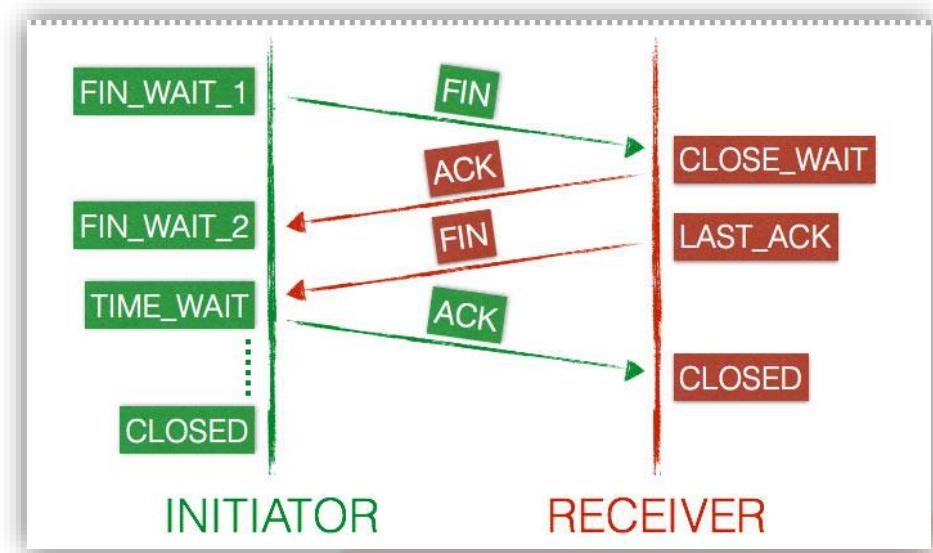
# TCP – Janela deslizante (#2)

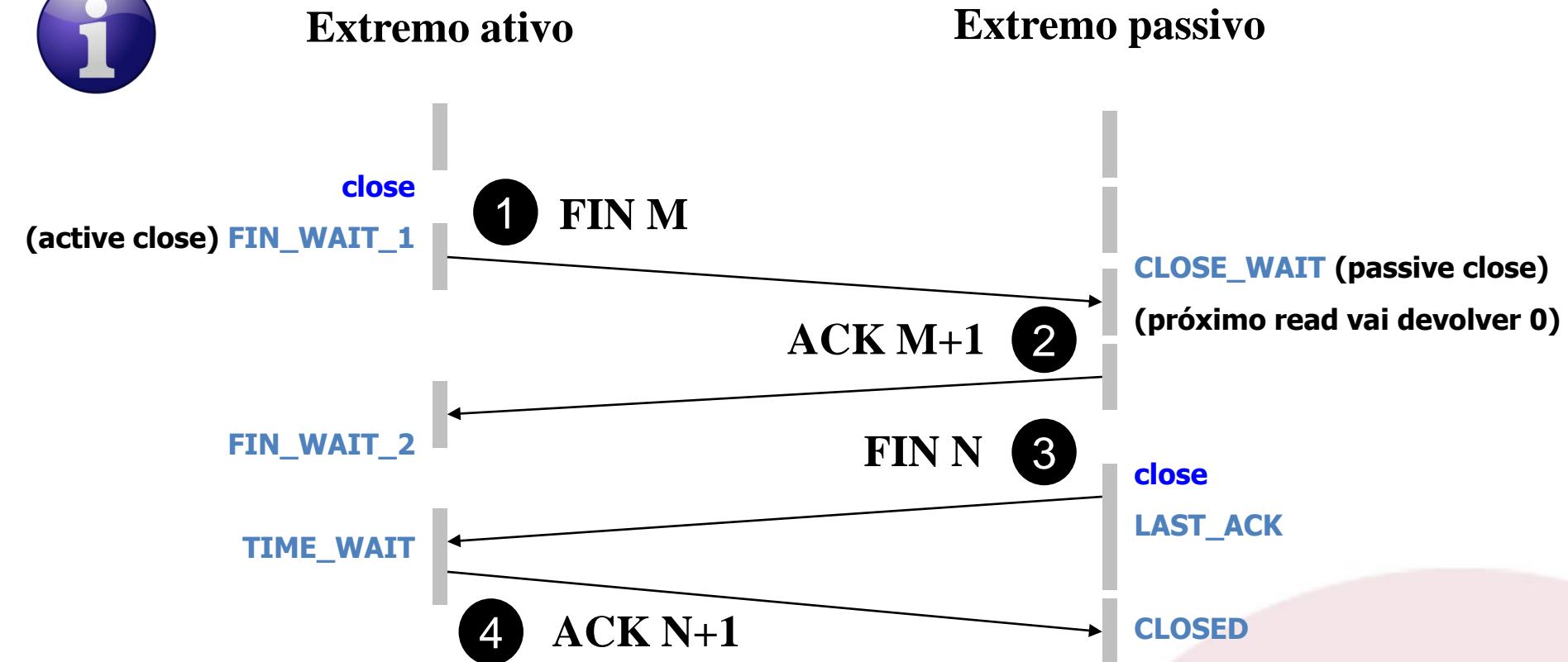


- Um segmento com ACK ativo confirma todos os segmentos anteriores

# Protocolo TCP: Término da ligação (1)

- Término da ligação: 4 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 0)
  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**

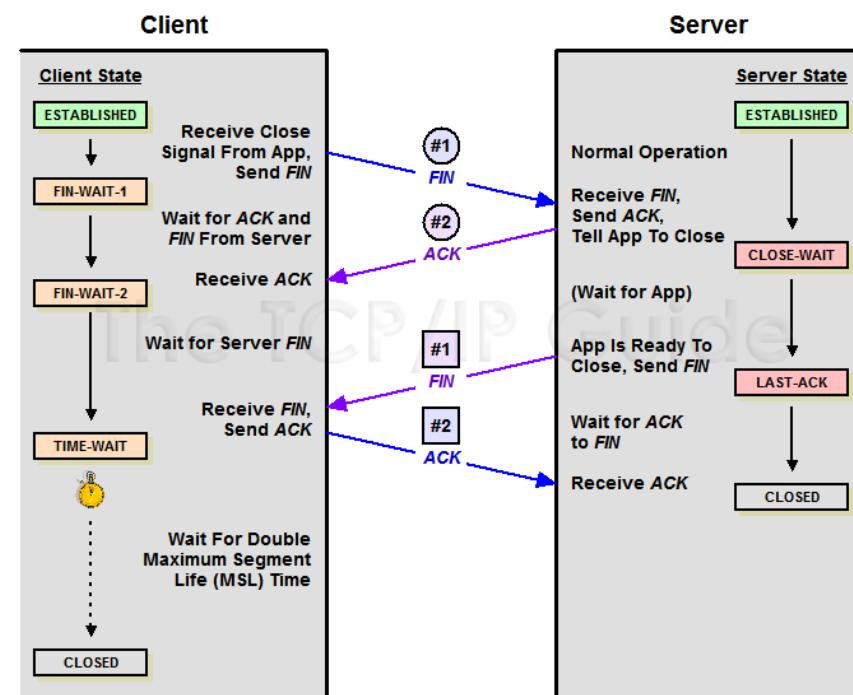




- Estado **TIME\_WAIT**
  - Enquanto o socket permanece no estado **TIME\_WAIT** continua a ocupar recursos

# 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

# Protocolo TCP: Estado da ligação

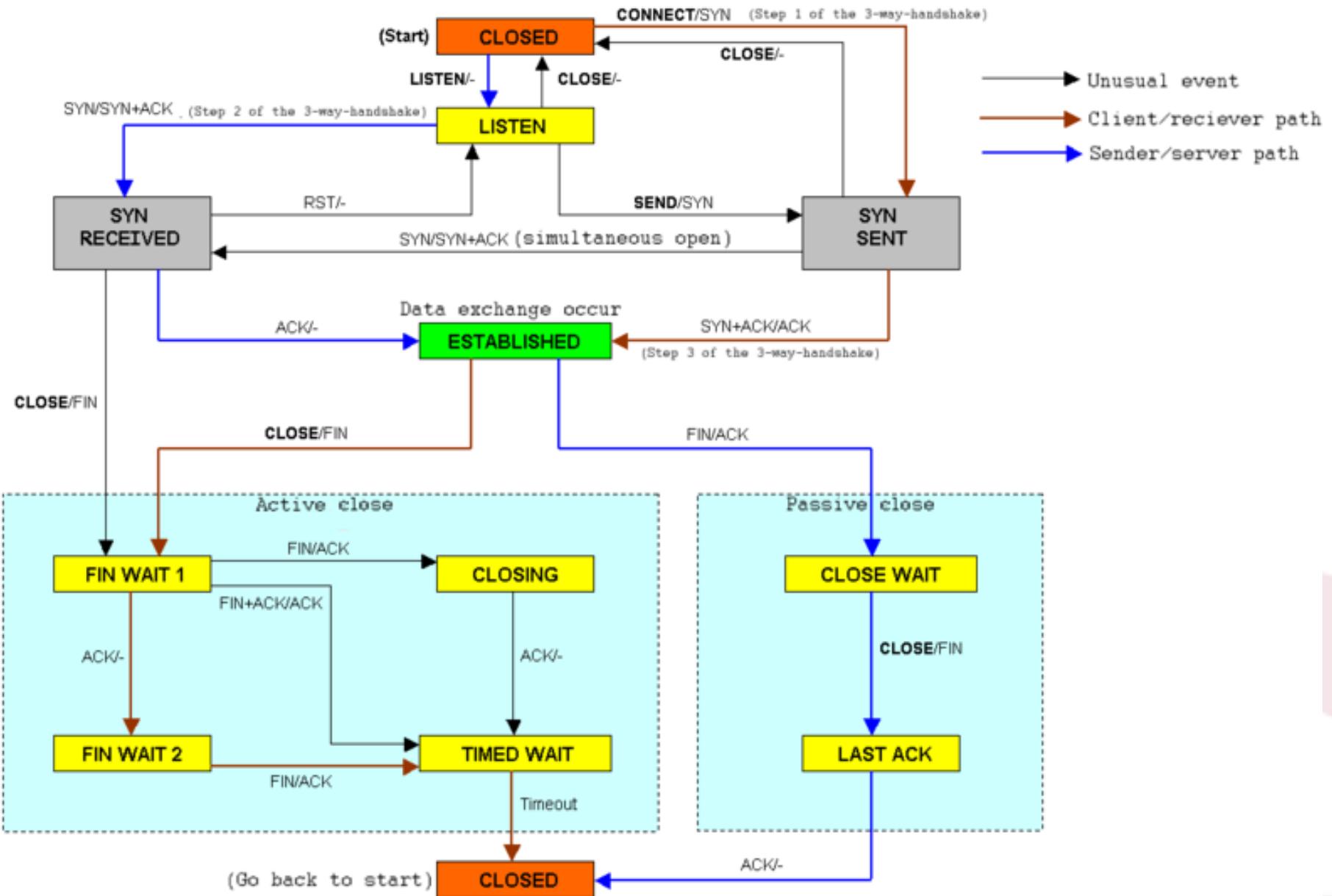
- Comando netstat
  - Existe na plataforma Linux e na plataforma Windows

```
[vmc@slinux vmc]$ netstat -t
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 slinu...estg.iplei.p:607 slinu...estg.iplei.p:898 TIME_WAIT
tcp      0      0 slinu...estg.iplei.p:613 slinu...estg.iplei.p:898 TIME_WAIT
tcp      0      0 slinu...estg.iplei.p:1017 slinu...estg.iplei.p:898 TIME_WAIT
tcp      0      0 slinu...estg.iplei.p:887 slinu...estg.iplei.p:898 TIME_WAIT
tcp      0      0 slinu...estg:netbios-ssn 192.168.5.180:1956 ESTABLISHED
tcp      0      0 slinu...estg.iplei.p:ssh 192.168.234.30:3057 ESTABLISHED
tcp      0      0 slinu...estg.iplei.p:ssh 192.168.234.30:3056 ESTABLISHED
tcp      0      0 slinu...estg.iplei.p:ssh 192.168.234.30:3058 ESTABLISHED
tcp      0      0 slinu...estg:netbios-ssn 192.168.234.60:1360 ESTABLISHED
tcp      0    216 slinu...estg.iplei.p:ssh 192.168.5.180:2180 ESTABLISHED
tcp      0      0 slinu...estg:netbios-ssn 192.168.234.30:3047 ESTABLISHED
tcp      0      0 slinu...estg:netbios-ssn 192.168.234.234 ESTABLISHED
tcp      0      0 slinu...estg.iplei.p:ssh 192.168.234.234 ESTABLISHED
tcp      0      0 slinu...estg.iplei.p:ssh 192.168.234.234 ESTABLISHED
tcp      0      0 slinu...estg.iplei.p:ssh 192.168.234.234 ESTABLISHED
[vmc@slinux vmc]$
```

```
C:\Windows\system32\cmd.exe
C:\Users\user>netstat -p TCP
Ligações activas

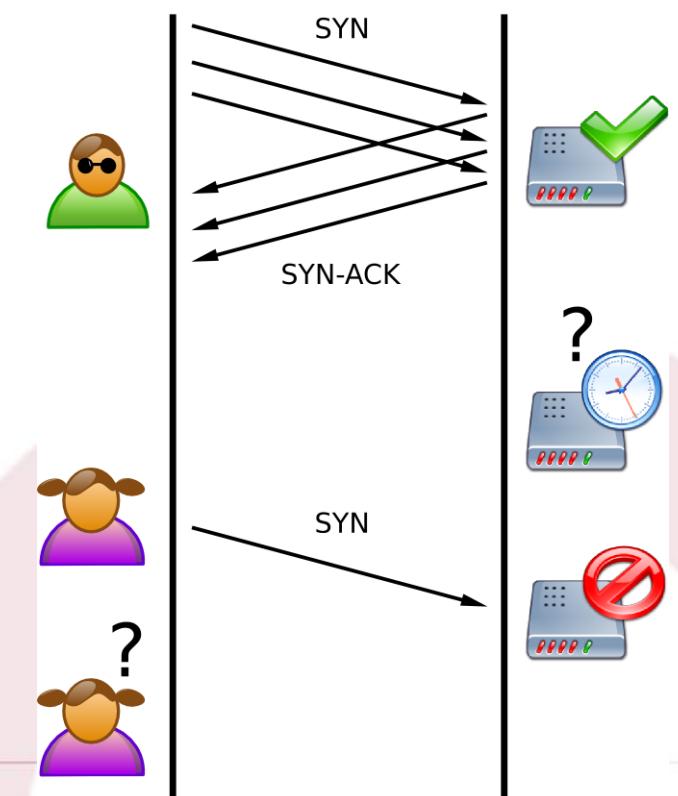
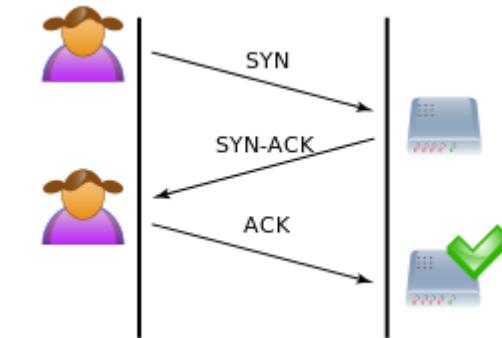
Proto Endereço local        Endereço externo      Estado
TCP   127.0.0.1:19872       patinhas_3:49170    ESTABLISHED
TCP   127.0.0.1:49170       patinhas_3:19872    ESTABLISHED
TCP   127.0.0.1:51809       patinhas_3:51811    ESTABLISHED
TCP   127.0.0.1:51811       patinhas_3:51809    ESTABLISHED
TCP   192.168.0.100:49172    sjc-not17:http    ESTABLISHED
TCP   192.168.0.100:49176    ec2-23-23-229-157:https CLOSE_WAIT
TCP   192.168.0.100:49335    fa-in-f125:5222    ESTABLISHED
TCP   192.168.0.100:49388    mrs02s05-in-f22:https ESTABLISHED
TCP   192.168.0.100:50622    fa-in-f125:5222    ESTABLISHED
TCP   192.168.0.100:51983    v-client-2b:https  CLOSE_WAIT
TCP   192.168.0.100:51984    ec2-23-23-193-246:https CLOSE_WAIT
TCP   192.168.0.100:51993    v-client-2b:https  CLOSE_WAIT
TCP   192.168.0.100:52166    mrs02s05-in-f9:https ESTABLISHED
TCP   192.168.0.100:52306    v-client-3b:https  CLOSE_WAIT
```

## Protocolo TCP: Diagrama de estados



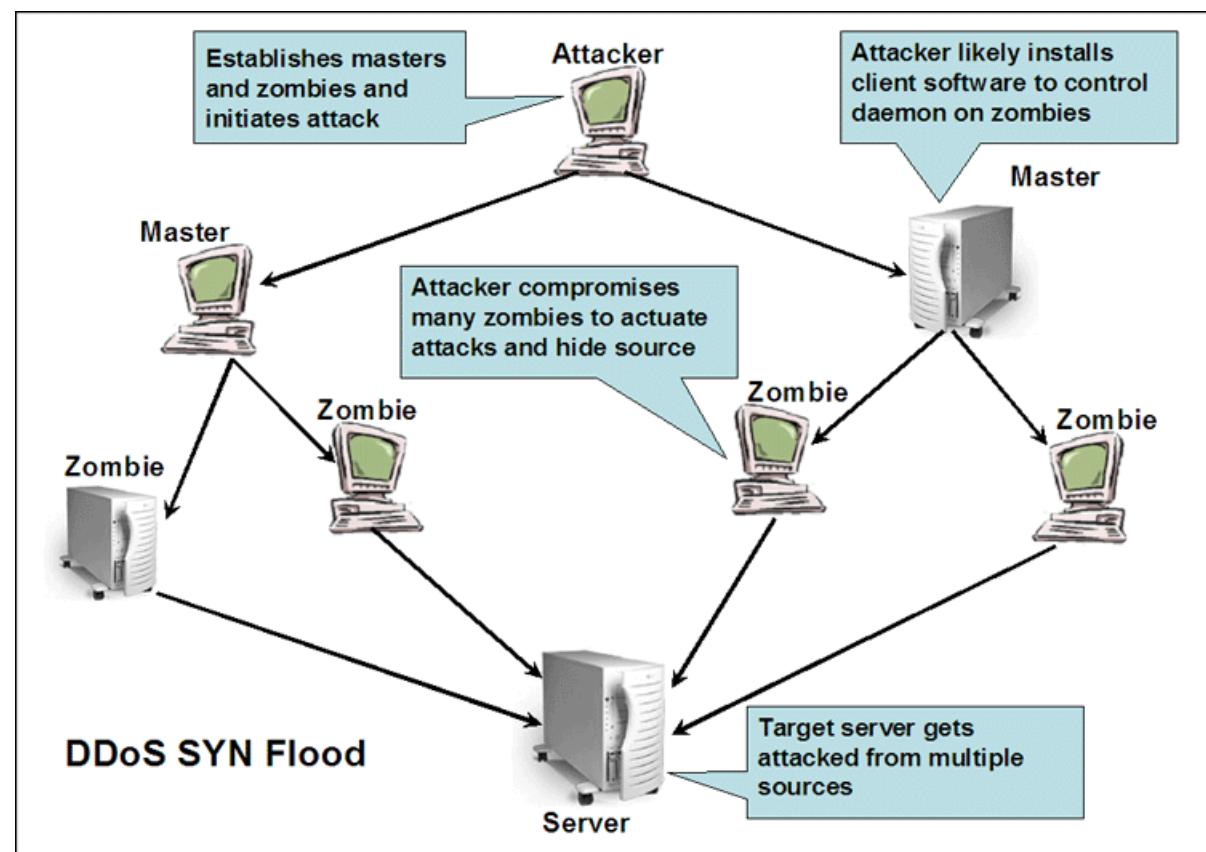
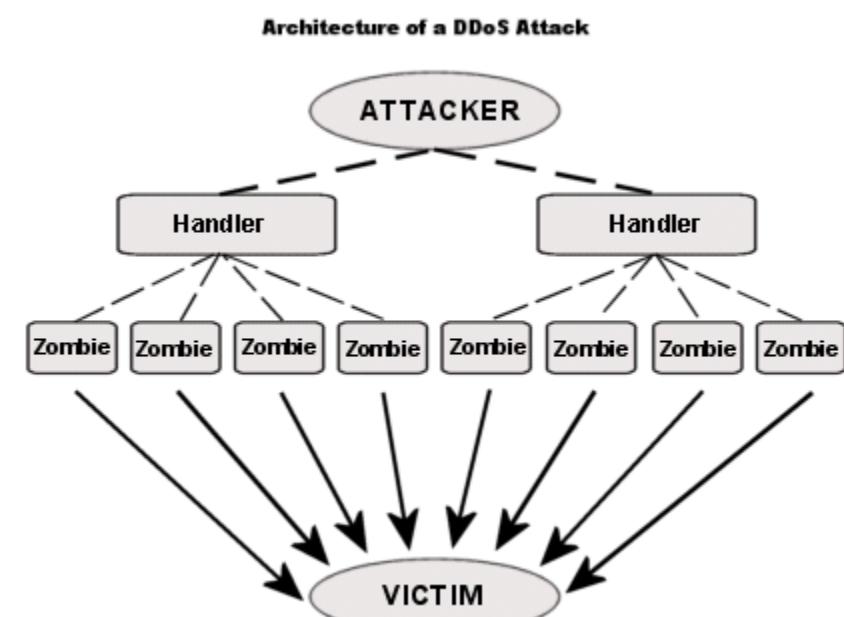
# TCP - SYN flood

- 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 existir muitos pedidos pendentes, servidor fica bloqueado...
    - Negação de serviço



# SYN flood distribuído

- Ataque SYN flood distribuído

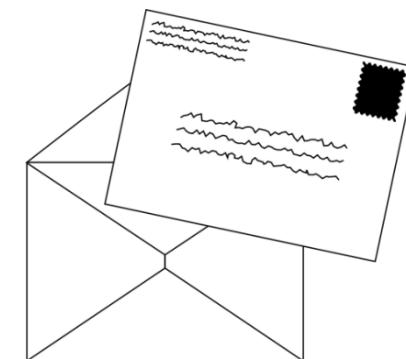


<http://bit.ly/2efcQ8O>



# Protocolo UDP: Introdução (1)

- A aplicação envia um *datagram* num canal UDP (*socket*)
  - O datagram UDP é encapsulado num datagram IP (IPv4 ou IPv6) e enviado para o seu destino
- O UDP é não orientado à ligação
  - Não garante entrega dos datagrams
    - “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



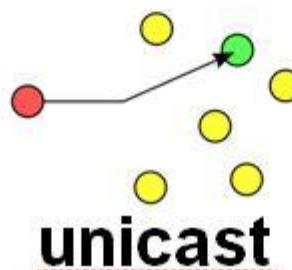
TCP



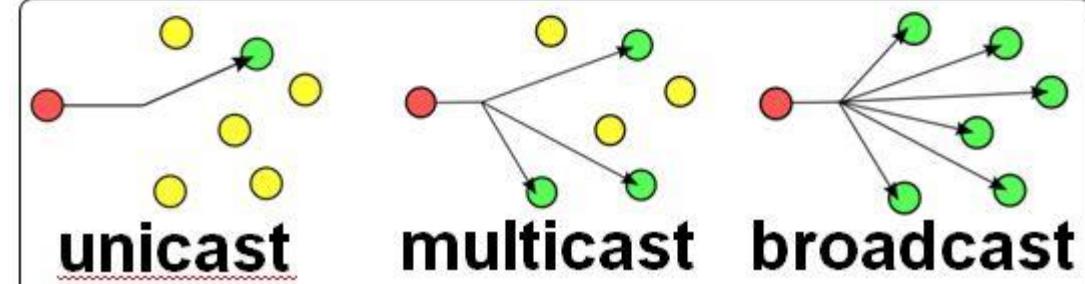
UDP

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

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

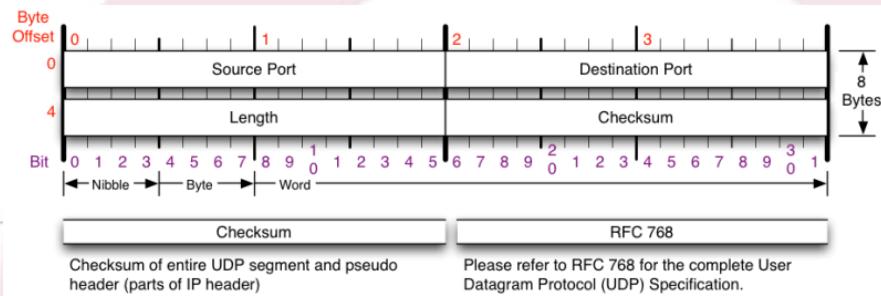


unicast



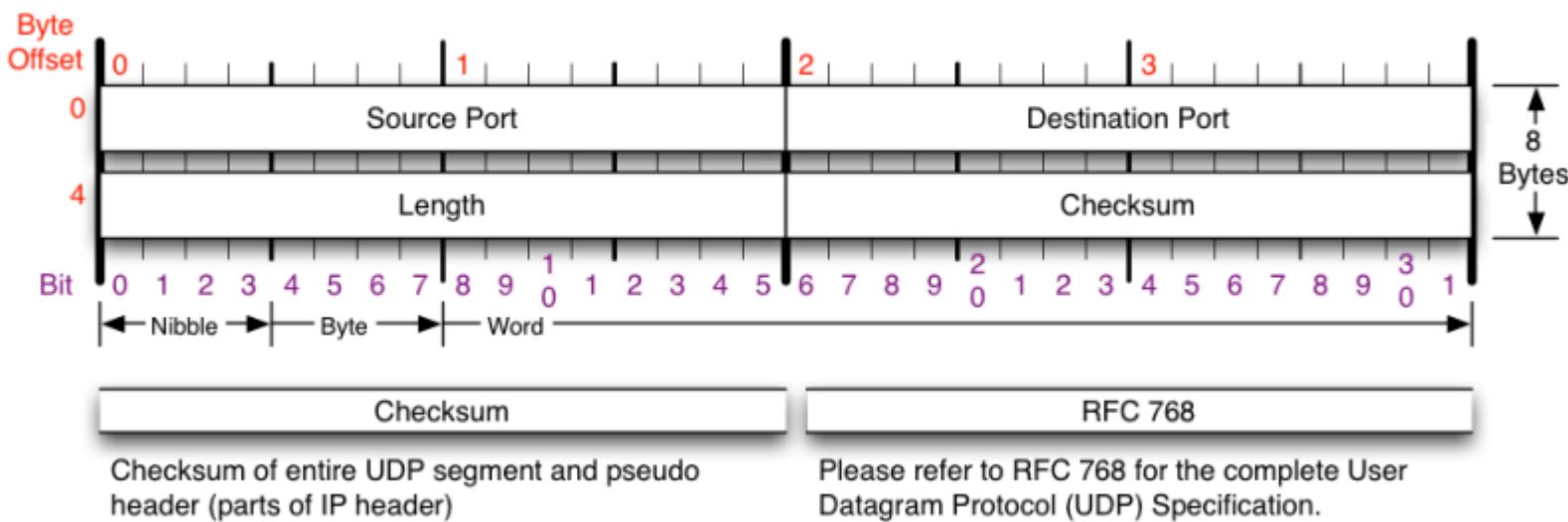
# Protocolo UDP: Introdução (2)

- Cada datagrama UDP tem um limite teórico de  $2^{16}$  bytes
  - Na prática:
    - Valor aceite como garantido são 576 bytes
      - Conhecido como o “*minimum maximum reassembly buffer size*”
      - Cada implementação deve ser capaz de agrregar 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.



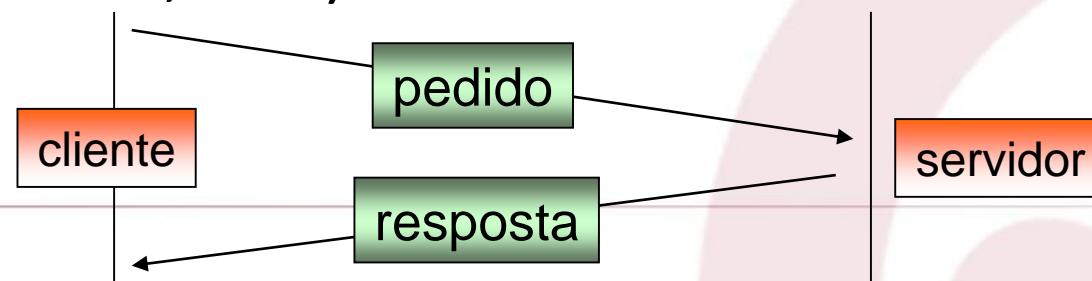
# Cabeçalho UDP

- 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 e dados)
  - *Checksum*: código de verificação da integridade do cabeçalho e dos dados



# Protocolo UDP: Características (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)

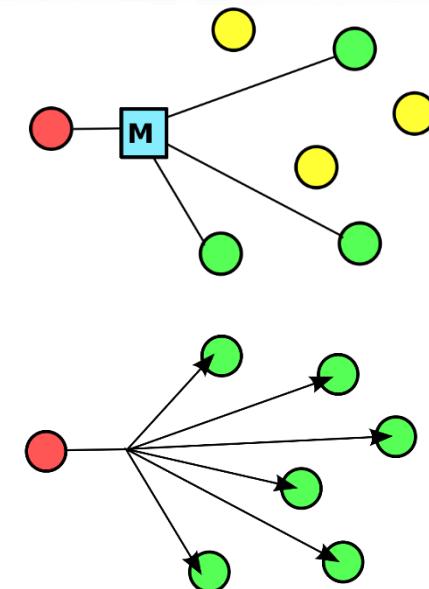


# Protocolo UDP: Características (2)

- 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 e de largura de banda da comunicação deve ser minimizado



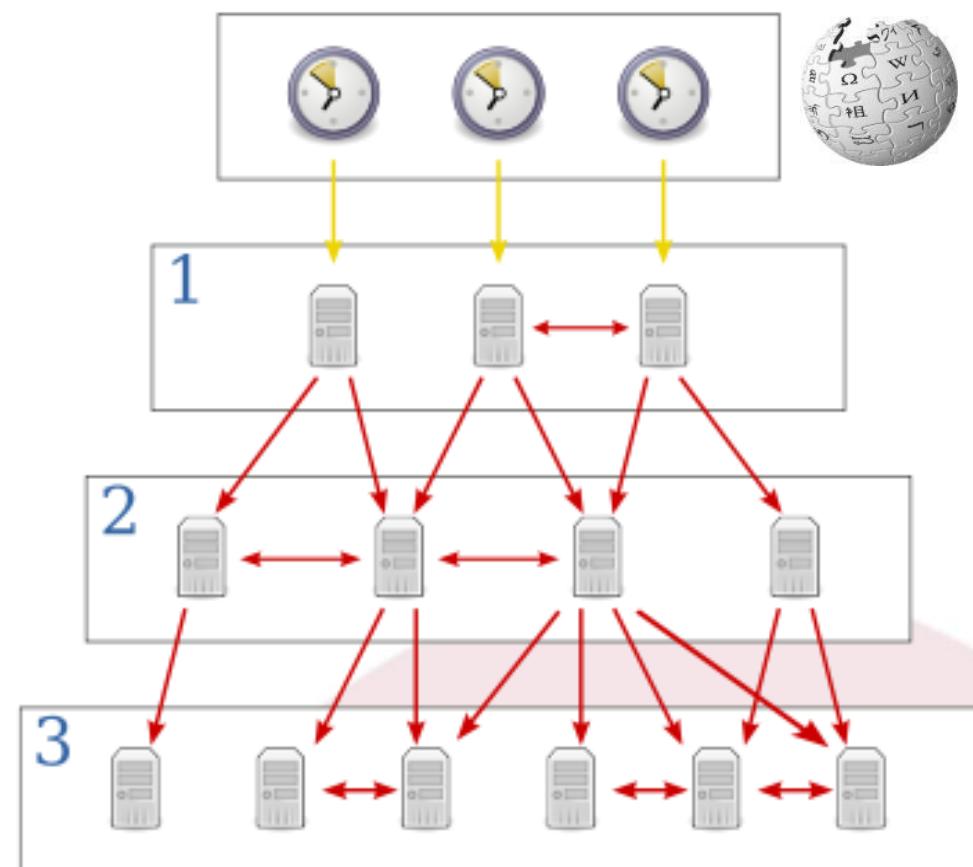
[https://en.wikipedia.org/wiki/Broadcasting\\_\(networking\)](https://en.wikipedia.org/wiki/Broadcasting_(networking))



# Network Time Protocol (NTP)



- Visa alcançar a sincronização dos relógios dos sistemas informáticos
  - É fundamental na informática o acesso a “hora certa”
- Suporta
- 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



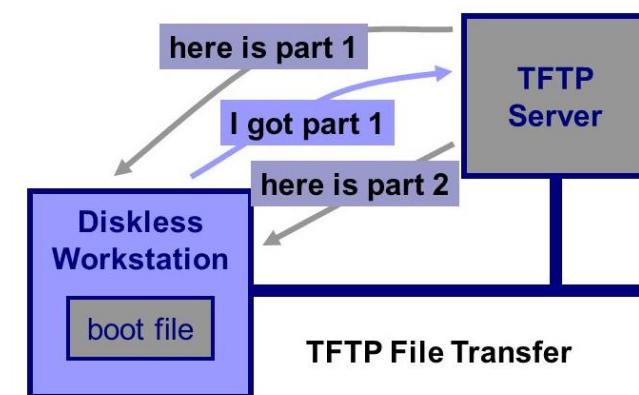
```
user@ubuntu:~$ ntptrace
```

```
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'
```

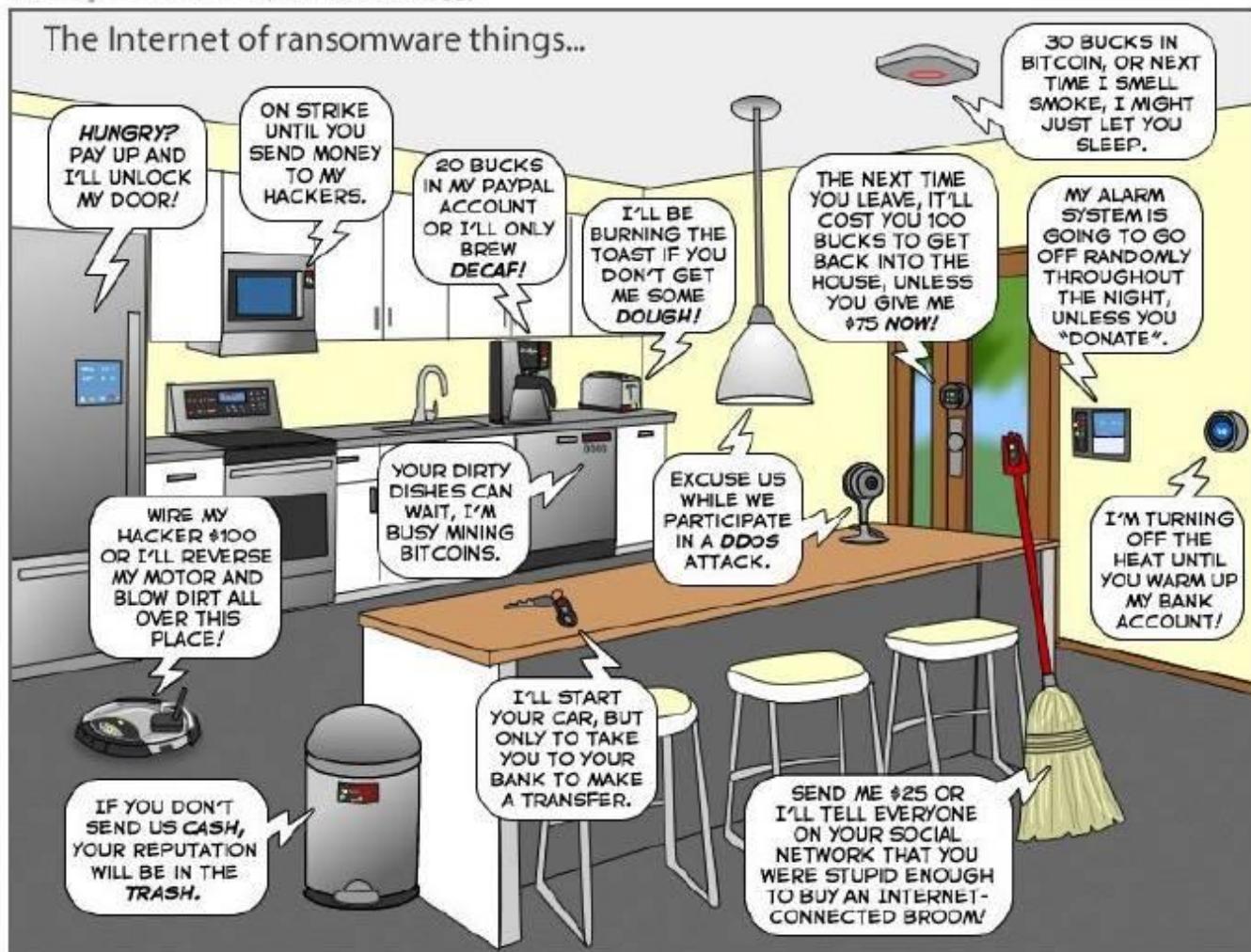
# Protocolo UDP: não utilizar quando...

- Não se deve usar UDP em:
  - Aplicações com vários ciclos consecutivos de pedido-resposta
  - Aplicações que movimentam volumes de dados importantes (e.g. transferência de ficheiros)
- 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)** recorre ao 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





The Joy of Tech™ by Nitroze & Snaggy



You can help us keep the comics coming by becoming a patron!  
[www.patreon.com/joyoftech](http://www.patreon.com/joyoftech)

[joyoftech.com](http://joyoftech.com)

# 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"*, 2<sup>nd</sup> edition, Richard Stevens, Addison-Wesley Professional, 2011
- *TCP/IP Network Administration*, Craig Hunt, Capítulo 2, “Building Blocks”, 3<sup>a</sup> edição, O'Reilly Networking, 2002.
- “*The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*”, Charles M. Kozierok, No Starch Press. 2005.
- *TCP performance*, Geoff Huston, *The Internet Protocol Journal*, Volume 3, Number 2, June 2000

