PROTOCOLOS FUNDAMENTAIS INTERNET

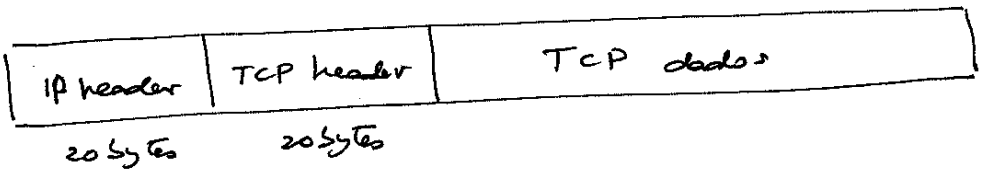
AULA 7 — TRANSMISSION CONTROL PROTOCOL (TCP)

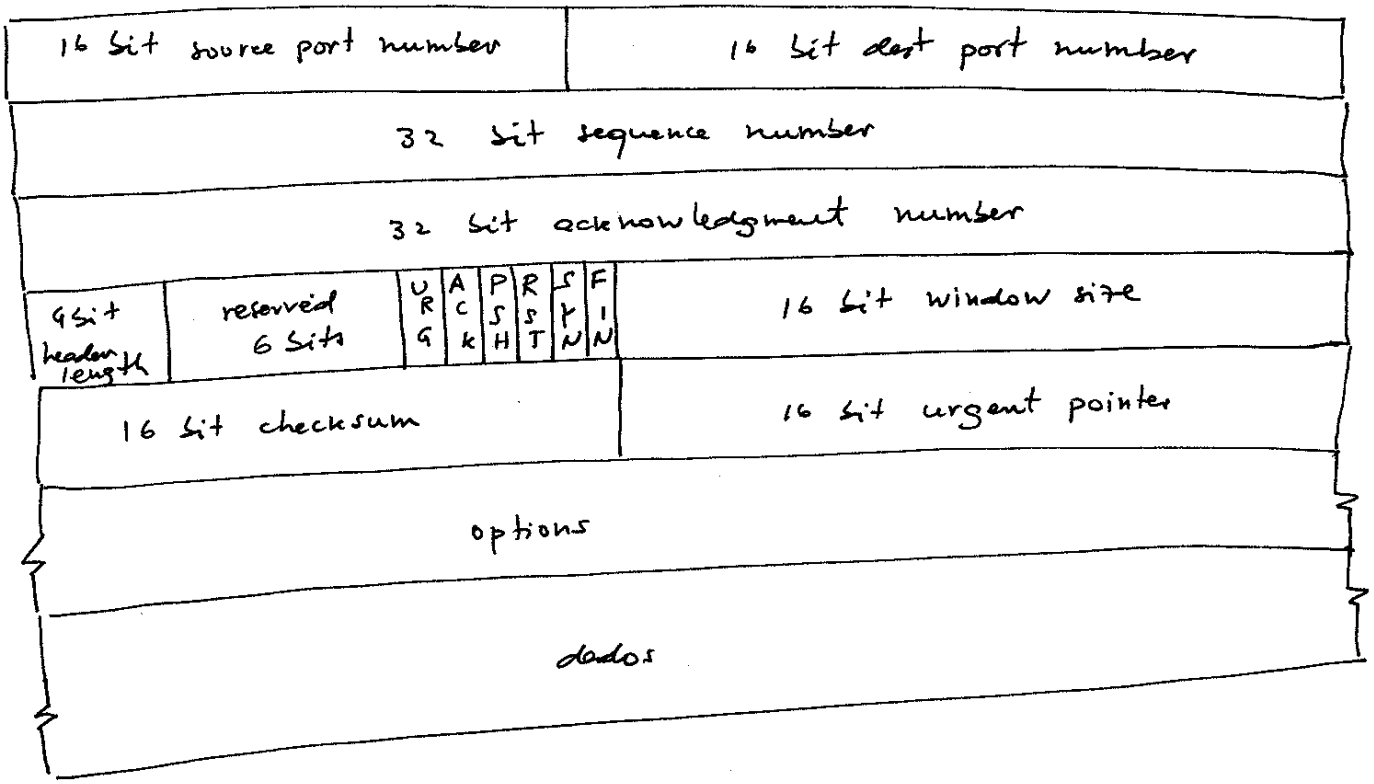
TCP oferece uma ligação, segura para transmitir um fluxo de bytes (connection oriented, reliable, byte stream service)

- ligação entre duas aplicações (cliente, servidor) que é estabelecida antes da troca de dados
- segura: (1) a chegada dos dados é sempre confirmada (acknowledge); se a confirmação não chegar ao fim de um "timeout" os dados são retransmitidos
  (2) um checksum é realizado sobre o cabeçalho e os dados; se o datagrama não chega intacto é ignorado
  (3) os segmentos IP podem chegar fora de ordem e repetidos; o protocolo garante entregar os dados correctamente à camada superior
  (4) há continuamente controlo do fluxo de dados, garantindo que o buffer do receptor nunca entra em overflow
- fluxo de bytes TCP envia bytes entre as duas aplicações não faz nenhuma ideia se são caracteres ASCII, binário etc

Ao contrário do datagrama UDP que é transmitido inteiro à camada IP, o datagrama TCP é normalmente feito em segmentos de comprimento suficientemente pequeno para não ser necessário fragmentação na camada IP

TCP header Stevens cover





- source port number
- dest port number
- número de sequência: todos os segmentos TCP são numerados: este número indica a posição do primeiro byte do segmento no datagrama TCP

  O primeiro número de sequência (initial sequence number) não é 1 mas sim um número associado com o relógio interno
- número de confirmação (acknowledgment number)

  indica o próximo byte que a aplicação está disposta a receber confirmando que os bytes anteriores foram recebidos
- comprimento do cabeçalho mínimo 20 bytes
  máximo 15 × 4 bytes = 60 bytes

URG - ponteiro urgente é válido

Ack - número de confirmação é válido

PSH - passar os dados à aplicação o mais rápido possível

RST - reset à ligação

SYN - início de ligação sincronizar os números de sequência

FIN - o locutor não quer ouvir mais dados

window size — tamanho do buffer ainda livre (número de bytes que o emissor pode enviar sem haver buffer overflow)

checksum - header e dados

urgent pointer - sequence number + urgent pointer = dados que devem ser entregues à aplicação o mais depressa possível

Opções (várias) a mais usada é o MSS maximum segment size

TCP ESTABELECIMENTO DA LIGAÇÃO E TERMINAÇÃO

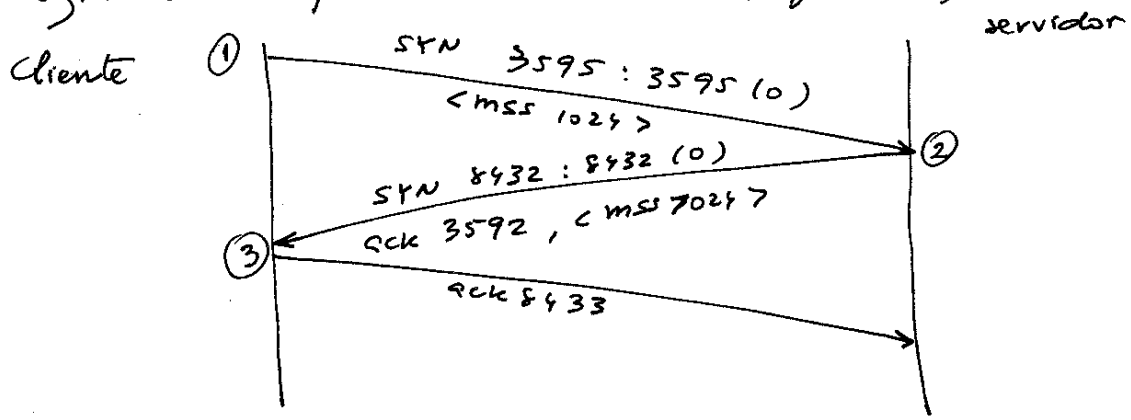
Handshake em 3 tempos

1) o cliente envia um segmento SYN especificando a porta do servidor a que se quer ligar e o número do segmento inicial (ISN, 1415531521)

2) o servidor responde com outro segmento SYN e o seu número de segmento inicial (ISN, 1823083524). O servidor envia tb a flag Ack activada para confirmar ter recebido o segmento do cliente

3) o cliente confirma o segmento que acaba de receber enviando um segmento com a flag Ack

diagrama temporal (stevens pag 232)



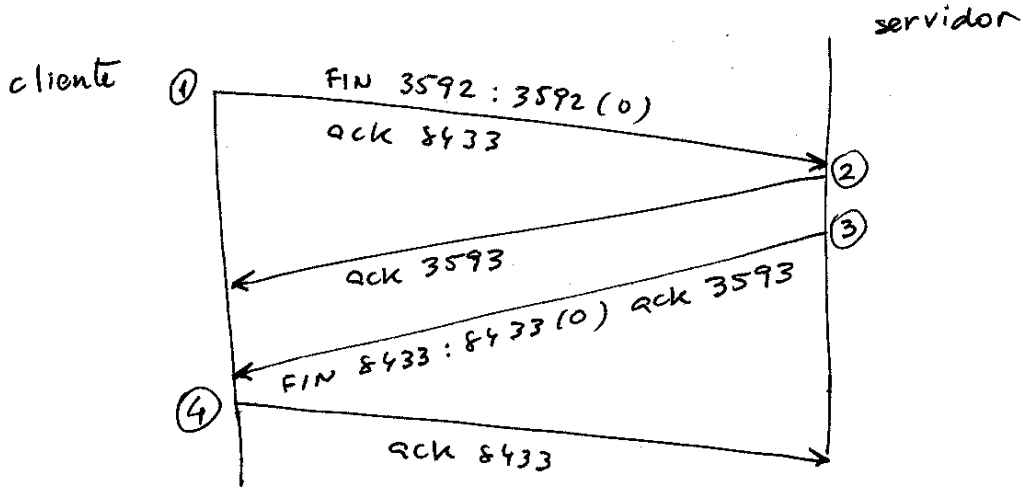
terminação em 4 tempos

1) o cliente envia um segmento FIN

2) o servidor envia um ACK . Neste momento a ligação está SEMI - FECHADA (half - close)

3) o servidor decide fechar o seu lado da ligação , enviando um FIN segmento

4) o cliente confirma a recepção enviando um segmento ACK



LAB stevens 230

telnet escorpiao discard
^] (ESCAPE)
telnet > quit

tcpdump -s host escorpiao and tcp

LAB stevens 235

timeout (server down — desligar cabo ethernet)

telnet peixes discard

tcpdump host máquinacliente

TCP Diagrama de estados stevens 241

LAB netstat -a -n -f inet
(ligar vários clientes telnet ...)

LAB stevens 247

reset segments (porta no servidor não está aberta)

telnet carneiro 20000

tcpdump host carneiro port 20000