**Portos**

O protocolo de transporte anuncia o tipo de comunicação (TCP ou UDP)

O número do porto identifica a aplicação a correr no target host.

**Se o porto é escolhido pela aplicação ou serviço (server), então é um porto não dinámico ou efémero.**

Cada aplicação pode escolher / usar mais que um porto, o sistema operativo assegura-se que as aplicações escolham portos diferentes entre si.

**Portos dinâmicos / Efémeros:**

São usados para identificar um cliente numa comunicação cliente-servidor.

IANA sugere que se usem os portos de 49152 a 65535 = 16383 portos, 16k.

É escolhido aleatoriamente pelo sistema operativo.

**UDP:**

A performance é dada pela rede IP

Troca de dados / informação pode ser feita entre aplicações no mesmo host, e não apenas entre hosts, com o uso de portos.

Não tem nenhum mecanismo de recuperação de mensagens perdidas

Não dá nenhum mecanismo de ordem / ordenação para a receção de dados.

Deixa mandar dados / informação para múltiplos destinos simultaneamente (point-to-multipoint communications).

**Quando a porta do servidor está fechada:** se o host tentar mandar um UDP packet, o servidor responde com ICMP Port Unreachable, porque não está aberta, ou seja, está fechada.

**TCP:**

A informação / dados chegam em ordem sem qualquer perda de informação.

**Sessão**: Os **EndPoints** estabelecem um canal lógico em que podem trocar data. O TCP tradicionalmente suporta apenas comunicações com dois intervenientes, no entanto, no multipath TCP. É estabelecida uma comunicação 3Way Handshake entre dois terminais, criando um subflow. Subflows adicionais podem ser estabelecidos da mesma maneira do Handshake, mas, em vez de ser em sessões diferentes, é tudo na mesma sessão. A data pode ser mandada para qualquer um dos subflows ativos, usando Joint Sequence e Acknowledgment Numbers.

Apple Siri usa o Multipath TCP.

**TCP Buffers e Windows:**

**MSS:** Maximum Segment Size, é o tamanho do maior segmento que o sender pode enviar / transmitir.

**SYN** = bit de sincronização

**SN** = Sequence Number (aleatório)

**AN** = Acknowledge Number

**Estabelecimento de uma conexão TCP: 3Way Handshake check:**

1. O terminal escolhe um random SN, com uma dada Window com tamanho de w.

manda uma mensagem:

*“SYN, SN = X, NA = 0, W = w”*

1. O servidor escolhe um random SN, faz Acknowledge do SN, e manda uma dada Window com tamanho de w.

Manda a mensagem

*“SYN+ACK, SN = y, AN = X+1, W = w”*

1. O terminal incrementa o seu SN manda a mensagem:

*“ACK, SN = X+1, AN = Y+1, W = w”*

**Fechando uma conexão TCP:**

1. Terminal:

*“FIN, SN = c”*

1. Servidor:

*“FIN, ACK, SN = d, AN = c+1”*

1. Terminal:

*“ACK, SN = c+1, AN = d+1”*

**TCP Congestion Control:**

**Janela de Congestão (CWND):** número máximo que o enviador consegue enviar. Buffer. **O máximo vai ser igual ao RWND.**

**Janela de Recepção / Receiver Window (RWND):** Como o nome indica, é o buffer / janela de recepção que o receiver pode receber de cada vez.

**NACK:** Not Acknowledge Bytes. Bytes que ainda não foram Acknowledge.

**Allowed Window (AWND) = min(CWND – NACK, RWND)**

Exemplo: Uma aplicação abriu uma sessão TCP com Sequence Number de 5000 usando TCP Reno.

1. Se a janela de congestão inicial do terminal for de 10MSS, MSS 1400 bytes, janela de recepção de 20000 bytes, quantos pacotes de 1000 bytes podem ser enviados sem receber um Acknowledge do outro terminal?

Então: CWND = 10\*1400 = 14000 bytes, RWND = 20000.

Como estamos no início: 0 NACK.

Logo: AWND = min(CWND-NACK, RWND) = min(14000-0, 20000) = 14000

Podemos enviar 1000 bytes de cada vez, logo 14000/1000 = 14 mensagens/pacotes de cada vez sem receber bits de Acknowledge.

1. Num dado momento da ligação TCP, a janela de congestão é de 20000, a janela de recepção é de 15000, 4000 bytes foram enviados pelo terminal sem receber Acknowledgment, quantos pacotes de 1000 bytes de dados podem ser neste momento enviados pelo terminal?

A janela de congestão tem que ser igual à do RWND no máximo. Então não pode ultrapassar os 15000.

Logo: AWND = min(CWND-NACK, RWND) = (15000-NACK, 15000). 4000 bytes ainda não foram Acknowledge, então: 4000 not acknowledgeble. Logo: min(15000-NACK, 15000) = min(15000-4000/1000 = 4 bits , 15000), min(11000, 15000) = 11000.

11000/1000 = 11 pacotes / packets.

1. Numa sessão TCP, assumindo que a terminação foi bem sucedida, se o AN (Acknowledgment Number ) do ultimo pacote enviado pelo terminal, for de 100000, quantos bytes de dados foram enviados peloo terminal?.

SN inicial: 5000

AN = 100000.

No inicio, do Handshake, o AN é SN+1, então no inicio fica: AN = 5000+1

Durante a transmissão de dados, o AN vai ser o SN + bytes transmitidos, sendo que são os dados transmitidos + SN ao final.

AN = 5000 + 1 + data

Ao fechar a channel, o AN vai ser o AN anterior + 1, logo:

AN = 5000 + 1 + data + 1 = 100000 ⬄ data = -100000 + 5000 + 1 + 1

⬄ data = -94998.

**TCP Fast Retransmit:**

Um segmento é considerando perdido quando receber 3 ou mais ACK repetidos do data segment anterior.

TCP retransmite imediatamente o perdido.

**TCP Slow Start:**

No inicio, o CWND é usualmente 2, 3, 4, 10 MSS e o SSThresh = RWND. O CWND cresce muito rápido enquanto continuar a ser CWND < SSThresh. Nesse mesmo tempo o CWND é incrementando com Nº ACK, quando recebe um Acknowledgment,. ACK.

Continua até receber a dizer que foi perdid.

A partir daí o CWND reduz para metade, CWND/2 e refaz.

Quando o CWND atinge o SSThresh, cresce linearmente.

CWND < SSThresh? Exponencialmente

CWND = SSThresh? Linearmente

Occorreu erro? Reset, CWND/2

A close up of a map

Description automatically generated

**TCP**

**Congistion Avoidance:**

Quando um ACK chega, o CWND é atualizado; CWND = (CWND + Nª ACK) / CWND -> Linear

**Algoritmos TCP:**

**Tahoe:** Utiliza Fast Retransmit, Slow Start e Congestion Avoidance

**Reno:** Utiliza Fast Retransmit, Congestion Avoidance: Avança logo linearmente.

**New Reno:** *“When a loss occurs, CWND is defined as β\*CWND, with β=0.5. When a ACK arrives, CWND is updated as CWND=CWND+α, with α=1 MMS.”*

**Cubic:** *“”Uses a cubic function to control the CWND.*