Protocolos de Ligação de dados:

Sliding Window

$Se\ W \geq 1 + 2a$:
$U = 1$
$Se\ W < 1 + 2a$:
$U = \frac{W}{1+2a}$

$a = \frac{T_p}{T_f}$

$T_f = \frac{L}{R}$

$T_p = d * \tau_a = \frac{d}{V}$

Stop & Wait ARQ

$N_r = \frac{1}{1-P_e}$

$S = \frac{1-P_e}{1+2a}$

No Error Probability:
$P = (1-p)^n$
Error Probability:
$P = 1-(1-p)^n$
i Error Probability:
$P = \binom{n}{i} p^i (1-p)^{n-i}$

Go-Back-N ARQ (janelas pequenas)

$Se\ W \geq 1 + 2a$: tamanho máximo:

$S = \frac{1-P_e}{1+2a*P_e}$ $\quad W = 2^k - 1$

Se W < 1+2a :

$S = \frac{W(1-P_e)}{(1+2a)\cdot(1-P_e+W\cdot P_e)}$

$W_{max} = M - 1 = 2^k - 1$

K is number of bits used to code sequence numbers

$T_f \rightarrow tempo\ de\ transmissão(ms)$
$L \rightarrow tamanho\ da\ trama(bits)$
$T_P \rightarrow tempo\ de\ propagação(ms)$
$d \rightarrow distância(km)$
$\tau_a \rightarrow atraso\ de\ propagação(\mu s/km)$
$S \rightarrow eficiência$
$Nr \rightarrow nº\ médio\ tent.p.transmitir\ trama\ c.sucesso$
$Pe\ (FER) \rightarrow prob.de\ transmissão\ de\ trama\ com\ erros$
$W \rightarrow tamanho\ da\ janela$
$M \rightarrow Representação\ em\ mód.de\ nº\ de\ seq.$
$k \rightarrow nºbits\ necessários\ para\ codificar\ W\ tramas$
$d \rightarrow nº\ \min de\ erros\ nec.para\ erro\ ñ\ ser\ detetado$
$R \rightarrow data\ rate\ (bits\ /\ s)$

Nota: No caso de $P_o = 0$, as fórmulas de S são as de "Stop & Wait" e "Sliding Window"
Stop & Wait: quando a << 1 (eficiente apenas em distâncias curtas)
Selective Reject: quando a >> 1 (grandes distâncias, evitar retransmissões em caso de erro)

Parity Check -> d=2
Bi-dimensional Parity -> d=4
Internet Checksum -> d=2
Cyclic Redundancy Check (CRC) -> d>3

Débito máximo: $R_{MAX} = S * (R\ [ou\ C])$

$FER = 1 - (1-BER)^L$

Selective Reject/Repeat ARQ (janelas grandes)

Se $W \geq 1 + 2a$: tamanho máximo:

$S = 1 - P_e$ $\quad W = 2^{k-1}$

Se $W < 1 + 2a$:

$S = \frac{W\cdot(1-P_e)}{1+2a}$ $\quad W_{max} = \frac{M}{2} = 2^{k-1}$

- Link-by-Link ARQ - Repairs losses link by link. Store packets in case they have to be retransmitted (memory required)
- End-to-End ARQ - Switches/routers become simpler. Packets may follow different end-to-end paths. (Not acceptable when Packet Loss Ratio is high)

$T_{est} \rightarrow Tempo\ de\ estabeler\ ligação$
$T_{prop} \rightarrow Tempo\ de\ propagação \sim 0$
$T_{msg} \rightarrow Tempo\ de\ enviar\ dados\ \left(\frac{L}{R}\right)$
$T_i \rightarrow Tempo\ de\ enviar\ pacote\ numa\ ligação\ i\ \left(\frac{L}{R}\right)$
$T_{pi} \rightarrow Tempo\ de\ propagação\ numa\ ligação\ i$

paths. (Nc
high)

Circuit Switching
$T_{tot} = T_{est} + T_{prop} + T_{msg}$

Packet Switching
$T_{pac} = Sum(T_i)$
$T_i = T_{pi} + T_{msgi}$

Intro:

Physical Layer:

Lei de Shannon (capacidade max)

$C = B_c \log_2(1+SNR)$

$SNR \rightarrow Signal\ Noise\ Ratio = \frac{P_r}{N_0 B_c}$

$B_c \rightarrow Frequência\ do\ canal\ (Hz)$
$P_r \rightarrow Potência\ Recebida\ (W)$
$N_0 \rightarrow Ruido\ Branco\ \left(10^{-9}\frac{W}{Hz}\right)$
$N_0 B_c \rightarrow Potência\ do\ ruído\ recebido\ na\ frequência\ B\ (W)$

Cabos

$P_r = P_t \cdot Ganho\ (em\ W)$
$P_r = P_t + Ganho\ (em\ dB)$
$Ganho = \frac{1}{Atenuação}\ (em\ W)$
$Ganho = -Atenuação(em\ dB)$

$P_{dBW} = 10\log_{10} P$
$P_{dBm} = 10\log_{10}\left(\frac{P}{1mW}\right)$

M-PAM(amplitude)

$s(t) = A_i \cos(2\pi f_c t)$
$fase(\theta) = 0\ (zero)$

M-PSK(fase)

$s(t) = Acos(\theta_i + 2\pi f_c t)$
$A = constante$

M-QAM(ampli. e fase)

$s(t) = A_i \cos(\theta_i + 2\pi f_c t)$

Free Space Loss

$\frac{P_t}{P_r} = \frac{(4\pi d)^2}{\lambda^2} = \frac{(4\pi f d)^2}{c^2}$ $\quad \lambda f = c$

$P_t \rightarrow signal\ power\ at\ transmitting\ antenna$
$P_r \rightarrow signal\ power\ at\ receiving\ antenna$
$\lambda \rightarrow carrier\ wavelength$
$d \rightarrow propagation\ distance\ between\ antenas$
$c \rightarrow speed\ of\ light(3 * 10^8 \frac{m}{s})$

M níveis

$C = 2Blog_2(M)\ (C = 2B\ [2\ níveis])$
$Baudrate = 2B\left(\frac{symbols}{s}\ ou\ baud\right)$
$Bitrate = 2Blog_2(M) = C$

MAC:

Random Access (pouco eficiente em canais

Aloha
Transmite, espera por round-trip propagation delay, recebe ack e envia ou, atrasa e envia
- Pure Aloha
- Slotted Aloha

CSMA
Se canal livre transmite, senão espera. Há col.

$a = P_{col} = \frac{T_{prop}}{T_{frame}} \ll 1$

-Persistent: ocup-espera que fique livre
-Non-persistent:ocup-espera tempo aleatório
-p-persistent: espera até ter um slot livre para enviar

$vuln.time = 2T_{prop}$

CSMA/CD
Parecido a p-persistent
Ouve enquanto envia, colisão->aborta->binary exponencial back-off->retransmite só funciona se $T_f > 2T_{prop}$, senão não deteta colisão

CSMA/CA
Monitoriza canal até estar livre durante um período maior que DIFS (Distributed Inter-Frame Space) e transmite. Se estiver ocupado, define tempo de back off aleatório que vai diminuindo.
Necessita de ACK

Channel Partitioning (pouco eficiente em canais pouco carregados)
-Time Div. Multi.; -Freq. Div. Multi.

Taking Turns – Cada estação com o seu turno.
Estações com mais info -> turnos maiores.
-Polling-estação mestre.
Prob: Overhead; latência; ponto de falha
-Passagem de tokens- passsam tokens entre si para saber quem transmite.
Prob: same as above

Filas de Espera 1:

$C \rightarrow capacidade\ do\ canal\ (kbits/s)$
$L \rightarrow tamanho\ do\ pacote\ (bits)$
$R \rightarrow tráfego\ médio\ (kbit/seg)$
$\lambda \rightarrow taxa\ de\ chegadas\ (pacotes/seg)$
$\mu \rightarrow taxa\ de\ envios\ (pacotes/seg)$
$\rho \rightarrow intensidade\ média\ de\ tráfego(taxa\ de\ utilização)$
$T_a \rightarrow tempo\ médio\ de\ atraso\ dos\ pacotes\ (ms)$
$T_w \rightarrow tempo\ médio\ de\ espera\ na\ fila$
$T_s \rightarrow tempo\ médio\ de\ serviço$
$N \rightarrow nº\ de\ clientes\ no\ sistema$
$N_s \rightarrow nº\ de\ clientes\ a\ serem\ servidos$
$N_w \rightarrow ocupação\ média\ da\ fila\ de\ espera$
$V_s \rightarrow pacotes\ em\ processamento$
$V_S \rightarrow pacotes\ em\ espera$
$M \rightarrow nº\ de\ buffers$
$P_n \rightarrow nº\ de\ chegadas\ no\ intervalo\ T$
$P_b \rightarrow nºprob\ de\ bloqueio(perda\ de\ pacotes)$
$T_p \rightarrow$ Tempo nec. para trasm. um pacote
$C_c \rightarrow$ Capacidade canal
m → uma divisão da ligação

$\lambda = \frac{R}{L}$

$T_a = T_w + T_s = \frac{1}{\mu-\lambda} = \frac{1}{\mu(1-\rho)} = \frac{N}{\lambda}$

$T_w = \frac{N}{\mu}$

$T_s = \frac{1}{\mu}$

$\rho = \frac{\lambda}{\mu} = \frac{R}{C}$

$\mu = \frac{C}{L}$

$M = \frac{\log\left(\frac{P_b}{1-\rho}\right)}{\log(\rho)}, se\ \rho \neq 1$

Teorema Little
$N_s = \lambda T_s = \rho$
$N_w = \lambda T_w = \rho N$

$N = N_w + N_s = \lambda \cdot T_a = \frac{\rho}{1-\rho}$

$P_b = \frac{(1-\rho)\rho^M}{1-\rho^{M+1}}, se\ \rho \neq 1$

$P_b = \frac{1}{M+1}, se\ \rho = 1$

$P_n(T) = \frac{(\lambda T)^n e^{-\lambda T}}{n!}$

Statistical Multiplexing
$T_p = \frac{L}{C}$

Freq. Div. Multi.
$C_c = \frac{C}{m}$ $\quad T_p = L \cdot \frac{m}{C}$

Time. Div. Multi.
$C_c = \frac{C}{m}$ $\quad T_p = L \cdot \frac{m}{C}$

Filas de Espera 2:

M/M/1/B (B buffers)

Probabilidade de perder dados: $P(B) = \frac{(1-\rho)\rho^B}{1-\rho^{B+1}}$

$se\ \rho = 1 \rightarrow P(B) = \frac{1}{B+1}$

$se\ \rho \gg 1 \rightarrow P(B) = \frac{\lambda-\mu}{\lambda}$

D/D/1
Chegadas e atendimentos seguem distribuição determinista

M/D/1

$E[X] = \frac{1}{\mu}; E[X^2] = \frac{1}{\mu^2}$

$T_w = \frac{\lambda}{2\mu^2(1-\rho)} = \frac{\rho}{2\mu(1-\rho)}$

M/M/1 (Cadeias de Markov)

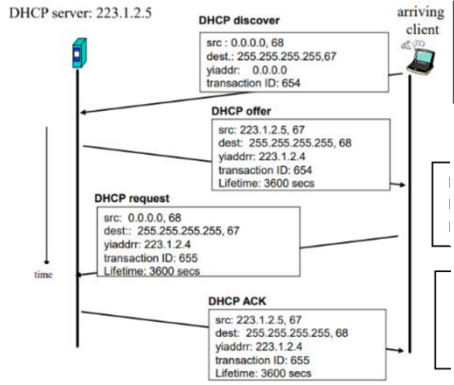
Chegadas -> Poisson; Attend. -> Exponencial
Prob. De estar em estado n: $P(n) = \rho^n(1-\rho)$
Tamanho médio de uma fila: $N = \frac{\lambda}{\mu-\lambda}$
Número médio clientes em espera: $N_w = N - \rho$
$E[X] = \frac{1}{\mu}; E[X^2] = \frac{2}{\mu^2}$ $\quad T = \frac{1}{\mu-\lambda}$
$T_w = T - T_s = \frac{1}{\mu-\lambda} - \frac{1}{\mu} = \frac{\lambda}{\mu^2(1-\rho)} = \frac{\rho}{\mu(1-\rho)}$

M/G/1
Chegadas->Poisson; Attend -> Arbitrário
Tempo de espera médio: $T_w = \frac{\lambda E[X^2]}{2(1-\rho)}$

$N = \lambda T = \lambda\left(T_w + \frac{1}{\mu}\right) = N_w + \rho$

Notação de Kendal (A/S/s/K)
A->Processo estatístico da chegada de clientes
S->Processo estatístico do serviço
s->número de servidores
K->capacidade do sistema em buffers

Extra:



$\log_b(x) = \frac{\log_k(x)}{\log_k(b)}$

$\log_2(x) = \frac{\log_b(x)}{\log_b(2)}$

Masks - nr hosts ($2^{\text{nr bytes host}} - 2$)
/27 – 30 /28 - 14
/29 – 6 /30 - 2

Quando uma trama é recebida por um Switch Ethernet e a tabela de encaminhamento do Switch não contém uma entrada para o endereço de destino da trama, o Switch envia a trama para todas as portas ativas exceto a porta através da qual a trama foi recebida.

Switch - Frame forwarding/flooding
When Switch receives a frame: 1. record link associated with sending host 2. index forwarding table using MAC destination address 3.

```
if (entry found in table) {
  if (destination is on segment from which frame arrived)
      drop the frame
  else
      forward the frame on interface indicated
} else
  flood <- (forward on all but the interface on which the frame arrived)
```

Bit stuffing – 1^5 -> 1^50
Byte stuffing – FLAG -> ESC FLAG.
Bit & Byte stuffing – 01^60 -> $01^501\ 01^201^30$

O Algoritmo Spanning Tree:
- Permite obter um caminho único entre nós Ethernet.
- Permite que uma única árvore seja calculada na rede, com raiz no nó com menor identificador.

Services provided by network layer
» Datagram network -> connectionless service (IP is datagram)
» Virtual Circuit network -> connection oriented service

Distance Vector Algorithm
♦ Iterative, asynchronous each local iteration caused by:
– local link cost change
– distance vector update message from neighbor
♦ Distributed
» node notifies neighbors only when its DV changes
♦ Neighbors then notify their neighbors, if necessary

Link-State Routing

♦ Each router keeps track of its incident links -> link up, link down ; cost on the link

♦ Each router broadcasts link state every router gets a complete view of the graph

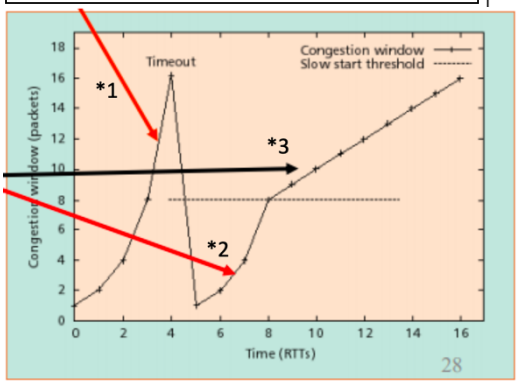
♦ Each router runs Dijkstra's algorithm, to -> compute the shortest paths ; construct the forwarding table

Transport:

TCP – Transmission Control Protocol
♦ Connection oriented
♦ Full-duplex
♦ Byte stream
♦ Flow control -> Reliability; ARQ mechanism; Avoids receiver's congestion
♦ Congestion control -> Avoids network's congestion

Additive Increase/Multiplicative Decrease
♦ Algorithm
» increases CongestionWindow by 1 segment
– for each RTT (Round Trip Time) -> additive increase
» divide CongestionWindow by 2
– when there is a packet loss -> multiplicative decrease
♦ In practice,
» Increases by ACK received
» Increment= MSS * (MSS / CongestionWindow)
» CongestionWindow += Increment
» MSS -> Maximum Segment Size

$$MaxWin = MIN(CongestionWindow, AdvertisedWindows)$$
$$EffWin = MaxWin - (LastByteSent - LastByteAcked)$$
$$Bitrate(byte/s) = CongestionWindows/RTT$$
network congestion decreases -> CongestionWindow Increases
network congestion increases -> CongestionWindow decreases



UDP - User Datagram Protocol (UDP)
♦ Datagram oriented
» Unreliable -> no error control mechanism
» Connectionless
♦ Allows applications to interface directly to IP with minimal additional protocol overhead
♦ UDP header
» Port numbers identify sending and receiving processes
» UDP length = length of packet in bytes
» Checksum covers header and data; optional

♦ Slow Start
» Sender starts with CongestionWindow=1sgm
» Doubles CongestionWindow by RTT (*1 on graph)
♦ When a segment loss is detected, by timeout
» threshold = ½ congestionWindow(*)
» CongestionWindow=1 sgm (router gets time to empty queues)
» Lost packet is retransmitted
» Slow start while
congWindow < threshold (*2 on graph)
» Then -> *Congestion Avoidance* phase (*3 on graph)

Congestion Avoidance
♦ Congestion Avoidance (additive increase)
» increments congestionWindow by 1 sgm, per RTT
♦ Detection of segment loss, by reception of 3 duplicated ACKs
» Assumes packet is lost, – Not by severe congestion, because following segments have arrived
» Retransmits lost packet
» CongestionWindow=CongestionWindow/2