Quiz 1:

[Binomial Combination Calculator](https://stattrek.com/online-calculator/binomial)

transmission delay = L/R

packets per second = floor( R/L )

propagation delay = d/s (km to m = km\*1000)

total delay = transmission delay+propagation delay

probability a specific user is transmitting:

p\*(1-p)^Nps-1  (Nps = number of users)

exactly one is transmitting: Nps \* p \* (1-p)^Nps-1

What is the probability that any 16 users (of the total 29 users) are transmitting and the remaining users are not transmitting?

[Choose Calculator](https://www.calculatorsoup.com/calculators/discretemathematics/combinations.php)

choose(29, 16) \* p16(1-p)13

The probability that more than 15 users of the total 29 users are transmitting is Σ i=16,29 choose(29, i) \* pi(1-p)29 – i

Queuing Delay = I(L/R)(1 - I) \* 1000

[Floor Calculator](https://www.symbolab.com/solver/floor-calculator)

Packets left in buffer = a - floor(1000/delay)  where a is the number of packets

Packets dropped = packets - buffer size

Stacks Layers Related to Phrases:

Application

handles messages from a variety of network applications'

Transport

'handles the delivery of segments from the application layer, may be reliable or unreliable

Network

moves datagrams from the source host to the destination host

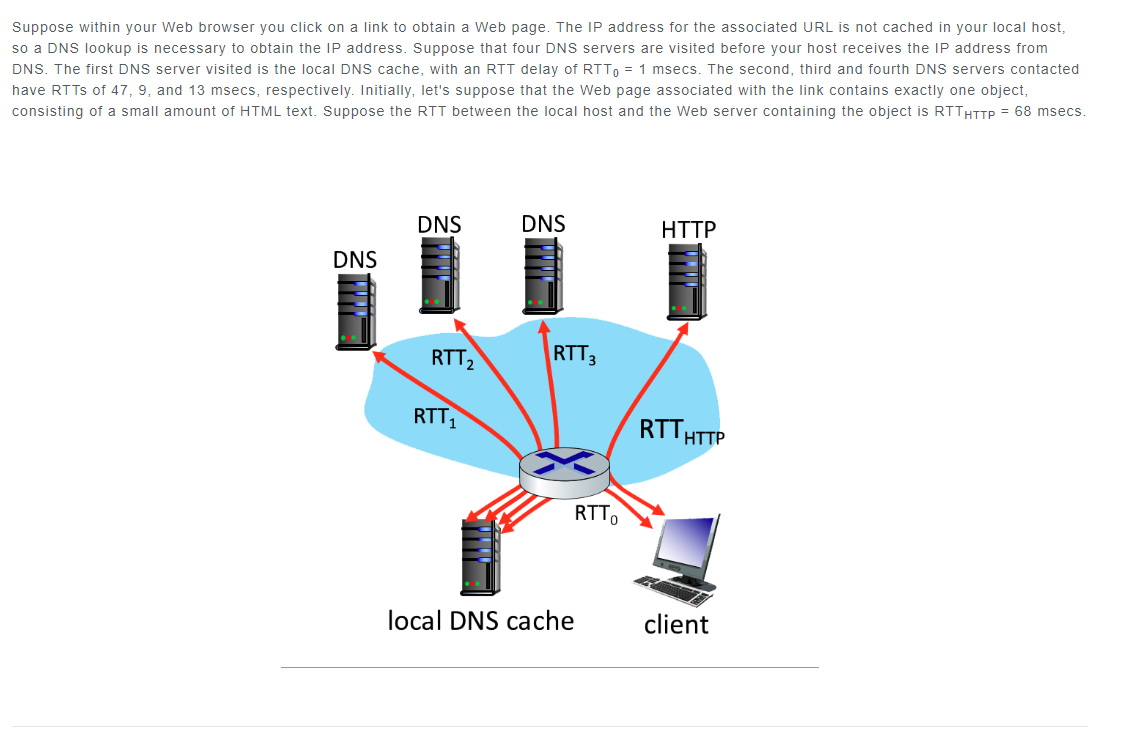
Link

'passes frames from one node to another across some medium'

Physical

bits live on the wire

Quiz 2: DNS and HTTPS delays



Assuming zero transmission time for the HTML object, how much time (in msec) elapses from when the client clicks on the link until the client receives the object?

RTT0 + RTT1 + RTT2 + RTT3 + 2\*RTTHTTP =x msec

Now suppose the HTML object references 9 very small objects on the same server. Neglecting transmission times, how much time (in msec) elapses from when the client clicks on the link until the base object and all 9 additional objects are received from web server at the client, assuming non-persistent HTTP and no parallel TCP connections?

RTT0 + RTT1 + RTT2 + RTT3 + 2\*RTTHTTP + 2\*#ofobjects\*RTTHTTP = xmsec

Suppose the HTML object references 9 very small objects on the same server, but assume that the client is configured to support a maximum of 5 parallel TCP connections, with **non-persistent** HTTP

Hint: for every 5 objects it takes 2 rtthttp

Initial delay = all rtt delay added together (no rtthttp)

Since there are 9 objects, there's a delay of 70 msec for the DNS query, two RTTHTTP for the base page, and 4\*RTTHTTP for the objects since the requests for 5 of these objects can be run in parallel (2 RTTHTTP) and the rest can be done after (2 RTTHTTP). .

If #object/parallels < 1 , =1, Otherwise, round up to nearest whole number

dnsDelay + 2\*RTTHTTP) + (#objects/#parallels \* 2 \* rtthttp)

Suppose the HTML object references 9 very small objects on the same server, but assume that the client is configured to support a maximum of 5 parallel TCP connections, with **persistent** HTTP.

RTT0+RTT1+RTT2+RTT3+(2 x RTThttp)+( ceiling(objects/parallel tcp connections) x RTThttp)

Since there are 9 objects, there's a delay of 70 msec for the DNS query. There's also a delay of two RTTHTTP for the base page, and 2 RTTHTTP for the objects. The total is 70 + 136 + 136 = 342 msec.As in 2 and 3 above, two RTTHTTP delays are needed to fetch the base HTML object - one RTTHTTP to establish the TCP connection, and one RTTHTTP to send the HTTP request, and receive the HTTP reply containing the base HTML object. However, with persistent HTTP, this TCP connection will remain open for future HTTP requests, which will therefore not incur a TCP establishment delay. Once the base object is received at the client, the maximum of five requests can proceed in parallel, each retrieving one of the 9 embedded objects. Each (in parallel) requires only one RTTHTTP delay to perform the HTTP GET/response for an embedded object. Once these first five objects have been retrieved, (if necessary) the remaining embedded objects can be retrieved (in parallel). This second round of HTTP GET/response to retreive the remaining embedded objects takes only one more RTTHTTP, since the TCP connection has remained open.

If #object/parallels < 1 , =1

Dnsdelay + 2\*rtthttps + ((#objects/parallels) \* rtthttps)

What's the fastest method we've explored: Nonpersistent-serial, Nonpersistent-parallel, or Persistent-parallel?

The delay when using persistent parallel connections is faster than using nonpersistent parallel connections, which is faster than using nonpersistent serial connections.

Quiz 3

( requests \* rtt ) + ( requests \* % not cached \* transmission delay )