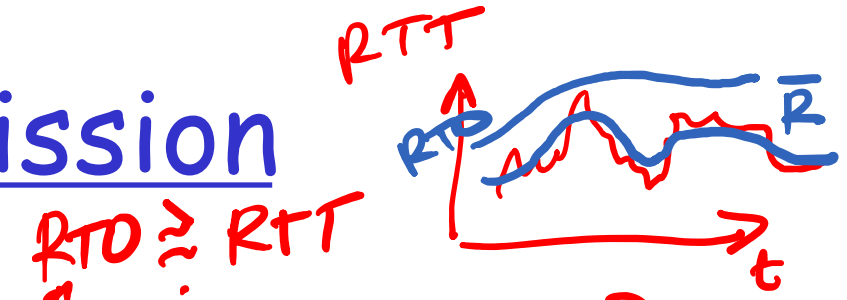
Adaptive Retransmission



Original Algorithm

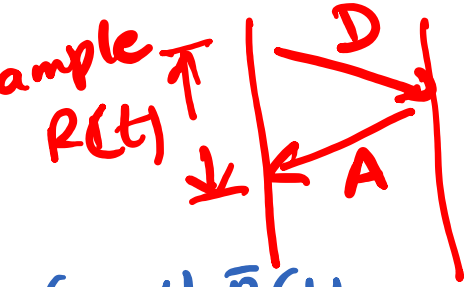
- Measure **SampleRTT** for a segment/ACK pair
- Compute weighted average of RTT (EWMA)
  - **EstimatedRTT =**

    (1-α) x EstimatedRTT+α x SampleRTT
    - where α is recommended to be 0.125 (=1/8)
- Set timeout based on **EstimatedRTT**
  - **TimeOut = 2 x EstimatedRTT**

$RTO \geq RTT$

adaptive

EWMA



$\bar{R}(t)_{new} = (1-\alpha)\,\bar{R}(t+1)_{old} + \alpha \cdot R(t)$

$0 < \alpha < 1$

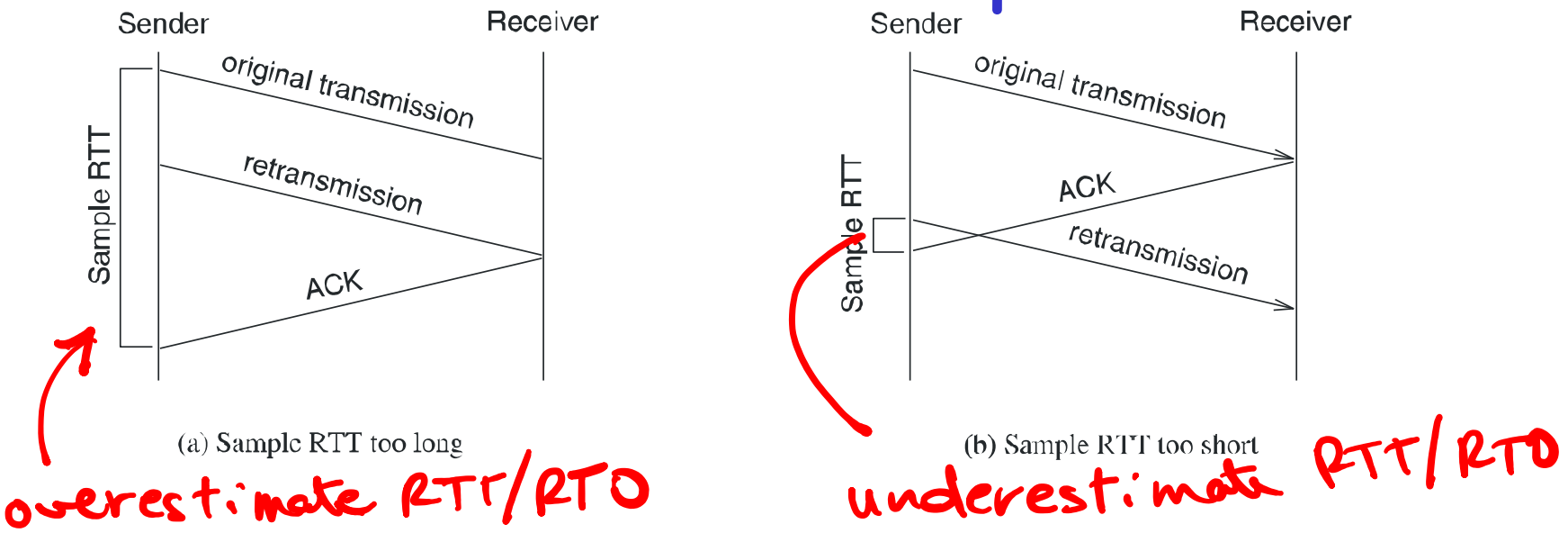
$\bar{R}(1) = (1-\alpha) R_0 + \alpha R(1)$

$\bar{R}(2) = (1-\alpha)[(1-\alpha) R_0 + \alpha R(1)] + \alpha R(2)$

$RTO = \gamma \cdot \bar{R}(t)$

$RTO = 2 \cdot \bar{R}(t)$ $\gamma > 1$

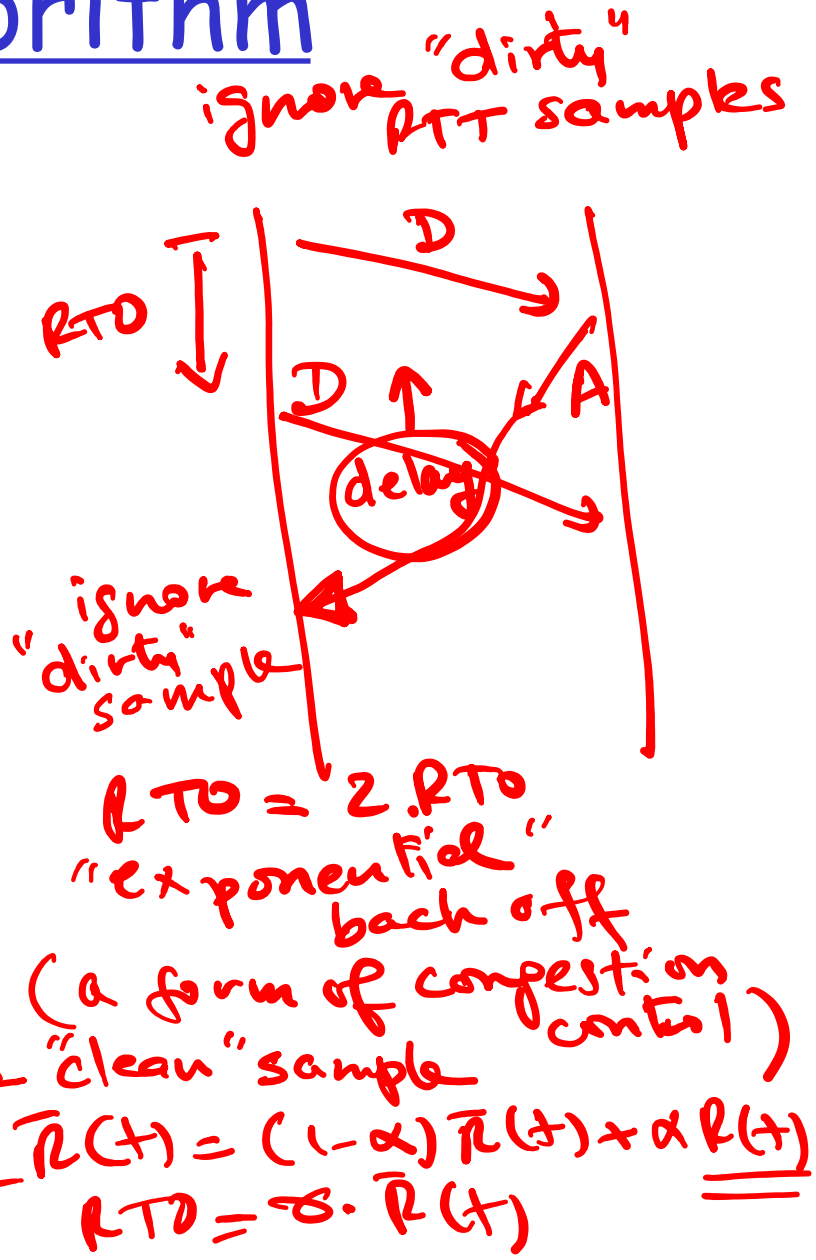
How to calculate SampleRTT?



- in (a), EstimatedRTT can grow without bound as datagrams get lost
- in (b), EstimatedRTT is observed to converge to half the correct estimate

Karn/Partridge Algorithm

- Ignore RTT samples for retransmitted segments
- **Problem:** ??
- timeout not updated to reflect increased delay!
- **Solution:** ??
- double timeout after each retransmission (*exponential backoff* - a form of congestion control!)
- Back to RTT estimation formula when receiving ACK for a segment that did ***not*** require retransmission



Adaptive Retransmission: Jacobson/Karels Algorithm

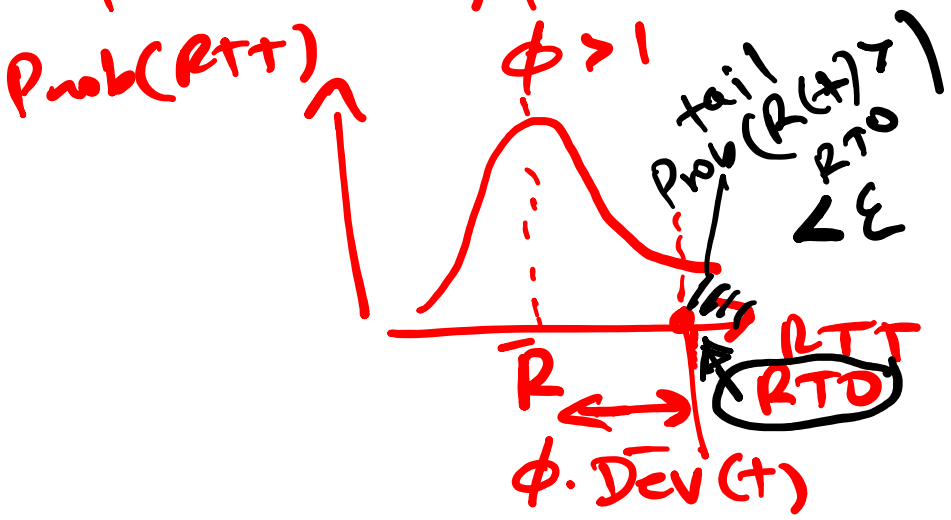
$$RTO = 2 \cdot \bar{R}(t)$$

$$\bar{Dev}_{new}(t) = (1-\beta) \cdot \bar{Dev}_{old}(t) + \beta \cdot Dev(t)$$

$$RTO = \bar{R}(t) + \phi \cdot \bar{Dev}(t)$$

$$Dev(t) = |R(t) - \bar{R}(t)|$$

$\phi > 1$



$\phi = 4$

Adaptive Retransmission: Jacobson/Karels Algorithm

- New calculation for average RTT
  - Difference = SampleRTT - EstimatedRTT
  - EstimatedRTT = EstimatedRTT + (α x Difference)
  - Deviation = Deviation + β (|Difference| - Deviation)
    - where β is recommended to be 0.25
- Consider variance when setting timeout value
  - TimeOut = EstimatedRTT + φ x Deviation
    - where φ = 4

Jacobson/Karels Algorithm

- Fast computation using integer arithmetic

$$\bar{R}(t) = (1-\alpha)\cdot\bar{R}(t) + \alpha\cdot R(t)$$

$$\Rightarrow \bar{R} = \bar{R} + \alpha(R - \bar{R})$$

$$\alpha = \frac{1}{8}$$

$$8\bar{R} = 8\bar{R} + (R - \bar{R})$$

$$\underbrace{8\bar{R}}_{\overline{SR}} = \overline{SR} + (R(t) - (\overline{SR} \gg 3))$$

Let $\overline{SR} = 8\cdot\bar{R} \Rightarrow \bar{R} = \frac{\overline{SR}}{8 = 2^3} = \overline{SR} \gg 3$

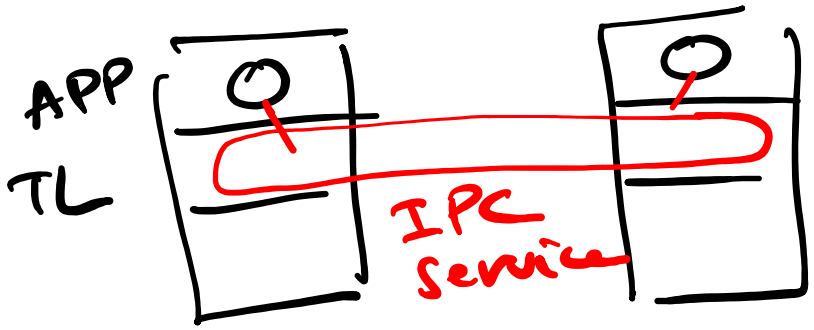
Jacobson/Karels Algorithm

- Fast computation using integer arithmetic

- scale by α and β, i.e. multiply by 8 (>>3) and 4 (>>2)
- keep **SampleRTT** and **TimeOut** unscaled
- Difference = SampleRTT – EstimatedRTT ' >>3
- EstimatedRTT' = EstimatedRTT' + Difference
- If (Difference < 0) Difference = - Difference
- Deviation' = Deviation' + (Difference - Deviation'>>2)
- TimeOut = EstimatedRTT ' >>3 + Deviation'

Recap: Midterm Review (1)

- What is the main goal of networking?
- Comm. channels (API, cap. v. thruput, delay, BxD)
- What is the main job of a transport layer?
- Mux/demux (UDP v. TCP)
- What is a protocol?
- Messages+actions, push v. pull, state-less/ful, non/persistent, FSM



Recap: Midterm Review (2)

- What is scalability?
- Caching, distributed servers, P2P
- What is reliability?
- Mechanisms: checksum, ACK, timer, sequence number, sliding window
- vs. Policies: S&W/GBN/SR, cum./sel./dup ACK, SWS setting = min (BxD, rawnd)
  send window size

Problem 1 (Error Control)

A 3000-km long, 1 Mbps link is used to transmit 1000-bit data packets using the *Selective Repeat* protocol. If the speed of light in this link is 2 x 10^5 km/second, how many bits the sequence numbers should be? Assume no flow control, and negligible transmission and processing times for acknowledgments. Take 1M = 1000,000.

$$SWS_{ideal} = B \times D = \frac{1\ Mbps}{1000} \times RTT \quad packets$$

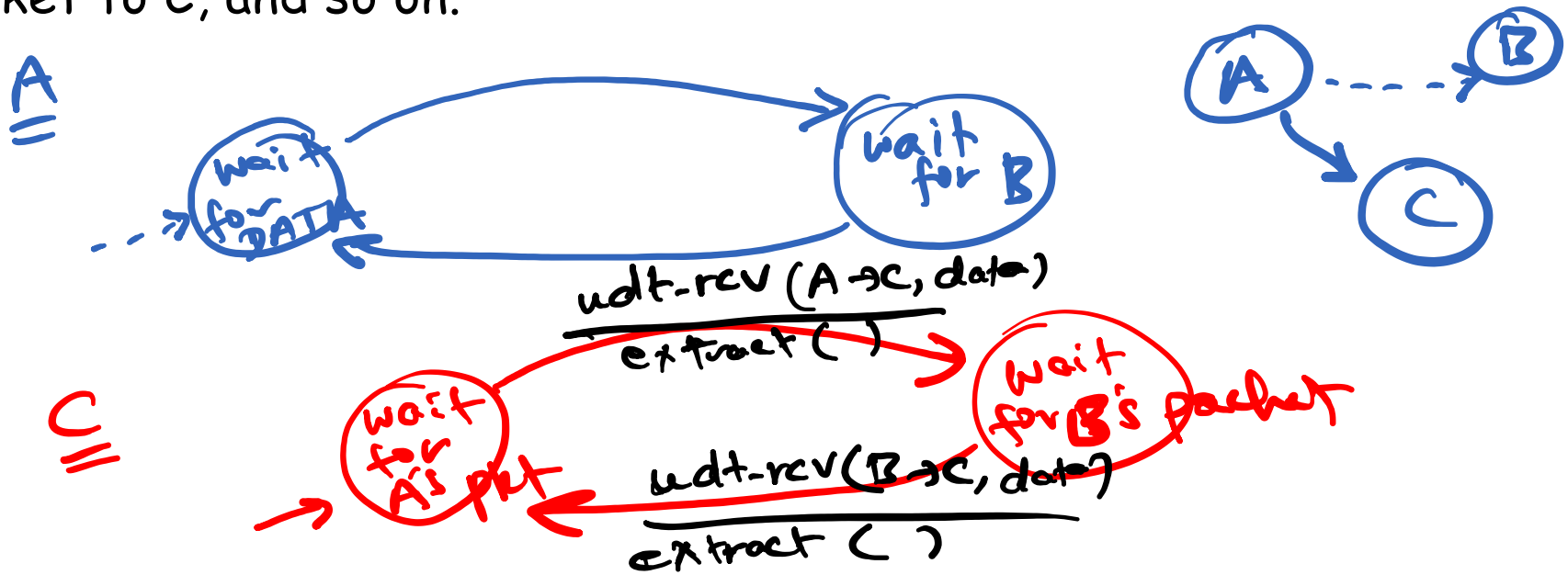
$$RTT_{(per\ packet)} = trans.delay + 2 \times prop.delay$$

$$\#\ seq\ nos. = \begin{cases} 2 * SWS & \text{for Selective Repeat} \\ SWS + 1 & \text{GBN} \end{cases}$$

$$\#\ bits\ for\ seq\ no. = \lceil log_2 (\#\ seq\ nos.) \rceil$$

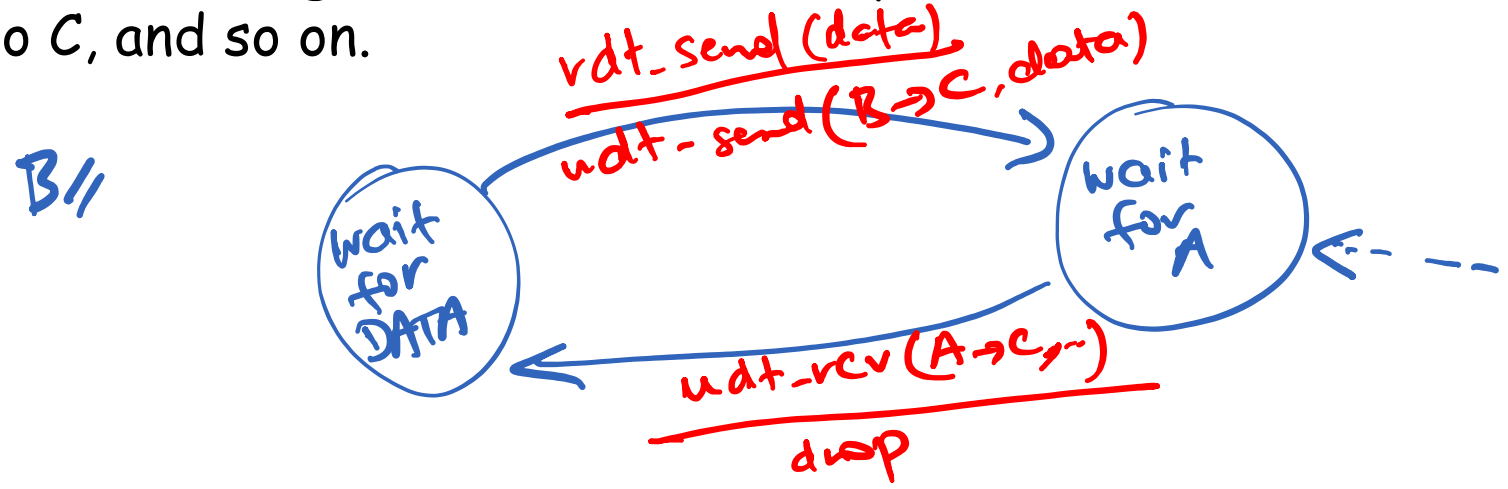
Problem 2 (Protocol Specification)

Consider a scenario in which Host A and Host B want to send messages to Host C. A, B, and C are connected by a ***perfect broadcast*** channel (that is, any message sent will be received by the ***other two*** entities correctly; the channel will ***not*** corrupt, lose, or re-order packets). Also, assume that any message sent will be received by the other two entities at the same exact time. The transport layer at Host C should alternate in delivering messages from A and B to the layer above; that is, it should first deliver the data from a packet from A, then the data from a packet from B, and so on. Host A should first get data from the layer above before it sends a packet to C, then B gets data from the layer above before it sends a packet to C, and so on.



Problem 2 (Protocol Specification)

Consider a scenario in which Host A and Host B want to send messages to Host C. A, B, and C are connected by a ***perfect broadcast*** channel (that is, any message sent will be received by the ***other two*** entities correctly; the channel will ***not*** corrupt, lose, or re-order packets). Also, assume that any message sent will be received by the other two entities at the same exact time. The transport layer at Host C should alternate in delivering messages from A and B to the layer above; that is, it should first deliver the data from a packet from A, then the data from a packet from B, and so on. Host A should first get data from the layer above before it sends a packet to C, then B gets data from the layer above before it sends a packet to C, and so on.

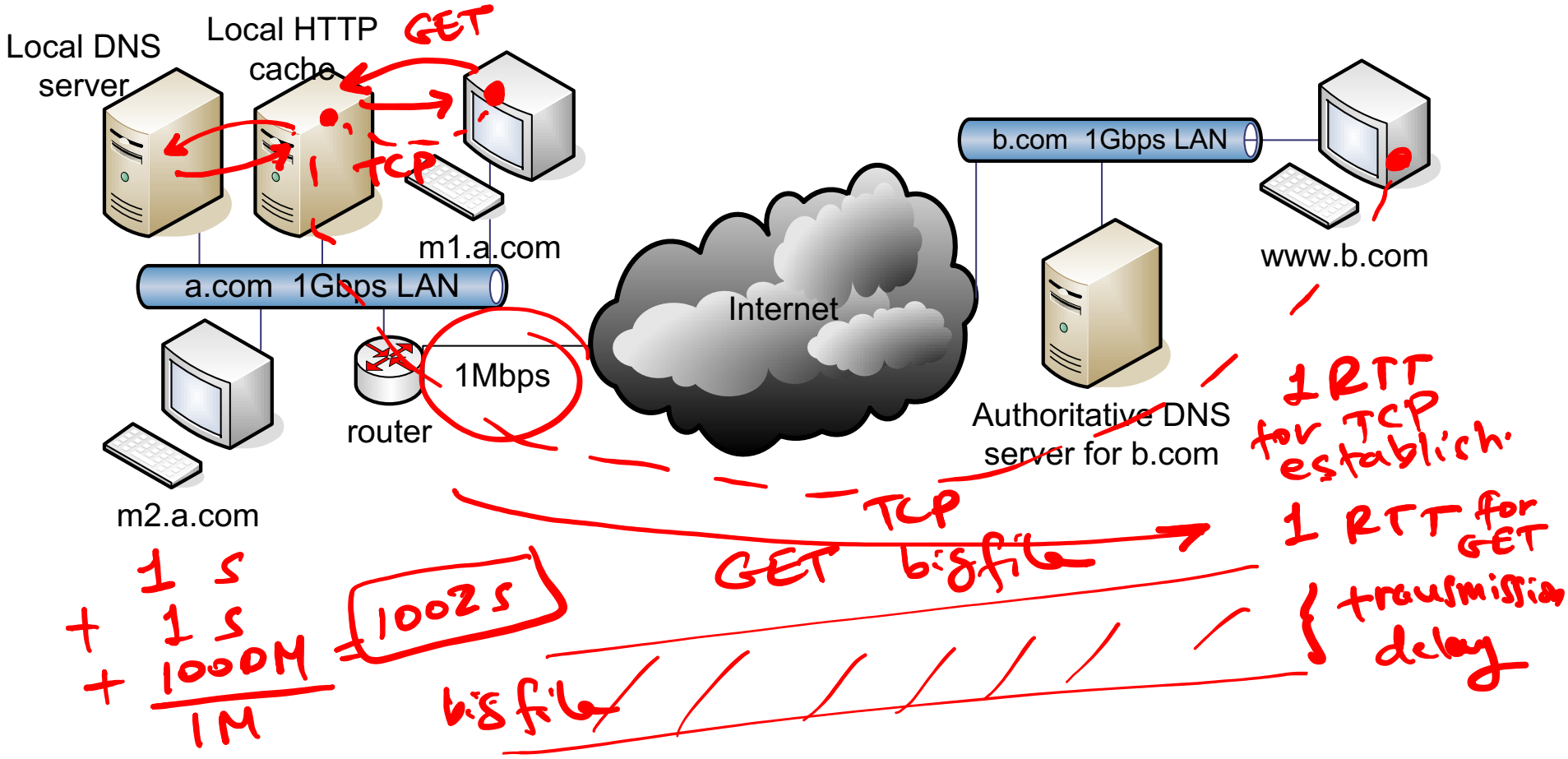


Problem 3 (HTTP Performance)

Suppose a web client wants to download a base html page of size O = 100K bits from a web server. This base html page contains ten embedded objects (img01.jpg, img02.jpg, … , img10.jpg) of the same size O = 100K bits each, all on the same web server. The (minimum) round-trip *propagation* delay RTP = 300 msec, and the channel rate R = 100 Mbps. Assume the client uses persistent HTTP (HTTP 1.1) with pipelining to retrieve the ten embedded objects, how long is the *response time*? Assume error-free transmission, consider TCP connection establishment (1 RTP) and the data transmission delay. Ignore header / control bits.

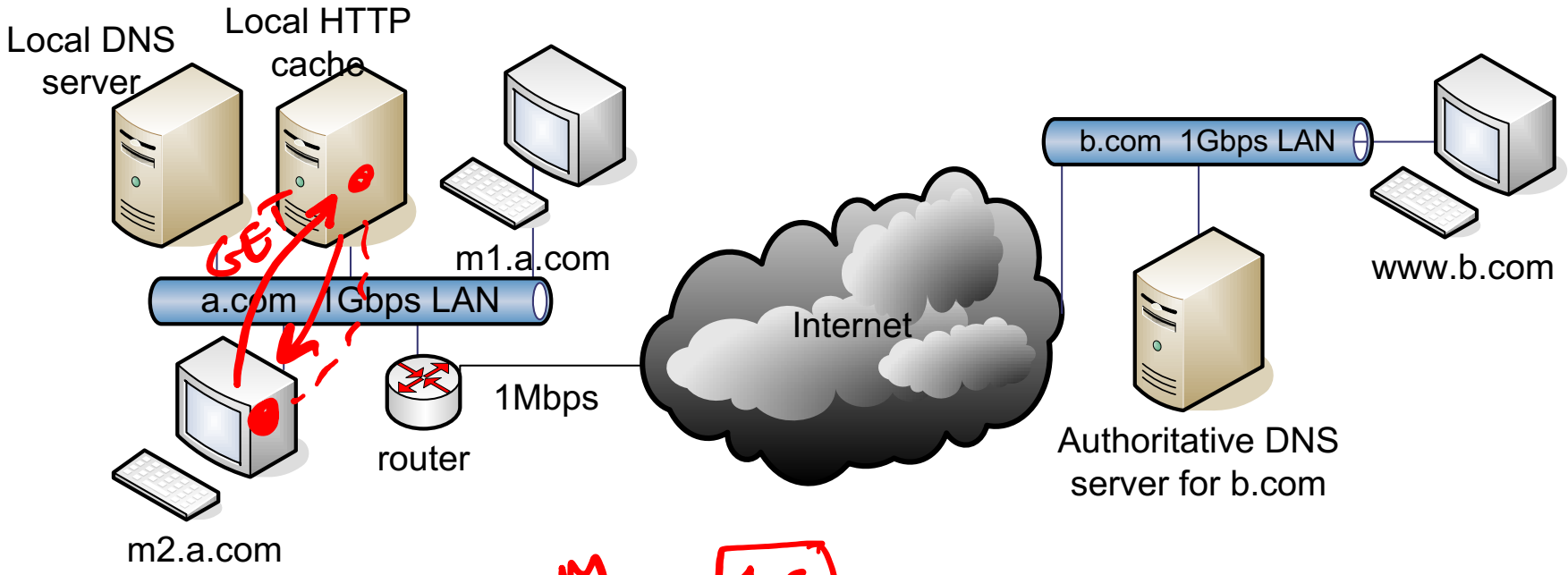
Problem 4 (Caching)

Suppose the user at m1.a.com types in the URL www.b.com/bigfile.htm into a browser to retrieve a large file of 1G bits (1000M bits). How long does it take? Assume that the local DNS server already has a mapping of www.b.com to its IP address, a user machine knows the IP addresses of both the local HTTP cache and the local DNS server, Internet RTT = 1 second, and include TCP connection establishment.



Problem 4 (Caching)

Now assume that machine m2.a.com makes a request to the same URL that m1.a.com requested. Consider now that the file is cached and will be directly served from the HTTP cache. What is the response time in this case?



$\frac{1000M}{1G} = 1s$

only transmission over LAN

Problem 4 (Caching)

If the local DNS server does *not* have a mapping of www.b.com to its IP address, how much does this mapping resolution add to the response time? Assume that to resolve a non-local hostname, the local DNS server first queries a Root DNS server, which knows how to reach the .com DNS server. Also assume all DNS requests are processed iteratively.