I. Activity NS 1.1

1. Prob(first 11 users active) = $0.1^{11} \times 0.9^{24}$

2. Prob(any 11 users active) = $\binom{35}{11} 0.1^{11} \times 0.9^{24}$

3. Prob(more than 10 users active) = $\sum_{i=11}^{i=35} \binom{35}{i} 0.1^{i} \times 0.9^{35-i}$

4. Since the system can only support 10 active users, the probability in Part (3) is the probability that the system is overloaded (i.e., users are unhappy). By

http://www.danielsoper.com/statcalc3/calc.aspx?id=71,

this probability is 0.00042. So we can support 35 users if it's acceptable to have unhappy users about 0.04% of the time.

II. Activity NS 1.2 (note: not required, and you don't have to submit it, but in case you are interested in attempting it anyways)

1. With A and C only, C will have to wait at most *T* time, where *T* is the length of the critical section S.

2. Now add B, and assume that it runs an infinite loop that never blocks. Consider the sequence of actions:

i. A runs and enters S.
ii. B runs and preempts A.
iii. C runs and tries to enter S. C is blocked since A is in S.
iv. The CPU scheduler chooses B to run since B has higher priority than A.
v. B keeps running in its infinite loop.
vi. As long as B is running, A can't run to exit S. And as long as A doesn't exit S, C keeps waiting to enter S. Hence, C's waiting time is *unbounded*.

Remark. B, in blocking a lower priority process A, blocks a *higher* priority process C *indirectly*. That's why it's called priority inversion, and it can last arbitrarily long.

III. Activity NS 2.1

P's delay = 67ms; throughput = 10Mb/s; loss rate = 23.05%

IV. Activity NS 2.2

1. Transfer of a huge file (e.g., migrate your library of high-def movies from local disk to iCloud, or download a full distribution of latex software).

2. Internet telephony (e.g., skype) – you say something, then wait/listen for any response/reaction from your friend. If the delay is high, you'll perceive bad interactivity.

3. Application A mainly wants high throughput (a "fat pipe"), whereas B mainly wants low delay.

4. The upgrade improves throughput, so it mainly benefits A.

5. Attempts to improve B's delay will be limited fundamentally by the speed of light over the required geographical distance (which can be long for say intercontinental skype calls).

6. (a) Without retransmission. There's a tradeoff between delay and loss controlled by the size of the router's buffer for waiting packets – a smaller buffer will lead to higher loss but smaller queuing delay.

(b) With retransmission. The sender will need at least one RTT to find out about a lost packet and hence retransmit it. Then we need at least another single trip time for the retransmitted packet to arrive at the receiver. Hence, a higher loss rate will lead to higher overall delay.

V. Activity NS 3.1

1. $n = 21$, $z = 12$.

2. $e = 5$ is relatively prime with z (12). $e \times d \bmod z = 25 \bmod 12 = 1$. So e and d satisfy the required mathematical properties.

3. The p and q won't work in practice, because they are too small.

I. **Activity NS 3.2**

1. Alice picks session key K_S.
2. Alice sends $K_B^+\{K_S\}$ to Bob.
3. Bob computes $K_B^-\{K_B^+\{\{K_S\}\}$ to get K_S.

I. Activity 4.1

1. 23.185.0.3

2. 202.65.347.31

3. The local DNS server (see the screenshot from the Mac OS/X computer).

4. The server is not officially responsible for the answer, but it has cached the answer from a previous lookup.

II. Activity 4.2

2. Use "`dig MX mit.edu`" to find out: `mit-edu.mail.protection.outlook.com.`

3. MIT outsources it.

4. Use "`dig mit-edu.mail.protection.outlook.com`" to find out:
   104.47.41.36
   104.47.42.36

There are two answers. They can be used for load balancing or fault tolerance.

Note that my lookup result also returns

```
;; AUTHORITY SECTION:
mail.protection.outlook.com. 1150 IN NS ns1-proddns.glbdns.o365filtering.com.
mail.protection.outlook.com. 1150 IN NS ns2-proddns.glbdns.o365filtering.com.
```

from which we can identity authoritative DNS servers for Q6 below.

5. Use who.domaintools.com to find the owner of, e.g., 104.47.42.36:

IP Location	United States Redmond Microsoft Corporation
ASN	AS8075 MICROSOFT-CORP-MSN-AS-BLOCK - Microsoft Corporation, US Mar 31, 1997)
Resolve Host	mail-by2nam030036.inbound.protection.outlook.com
Whois Server	whois.arin.net
IP Address	104.47.42.36

The owner is Microsoft Corporation.

6. Use

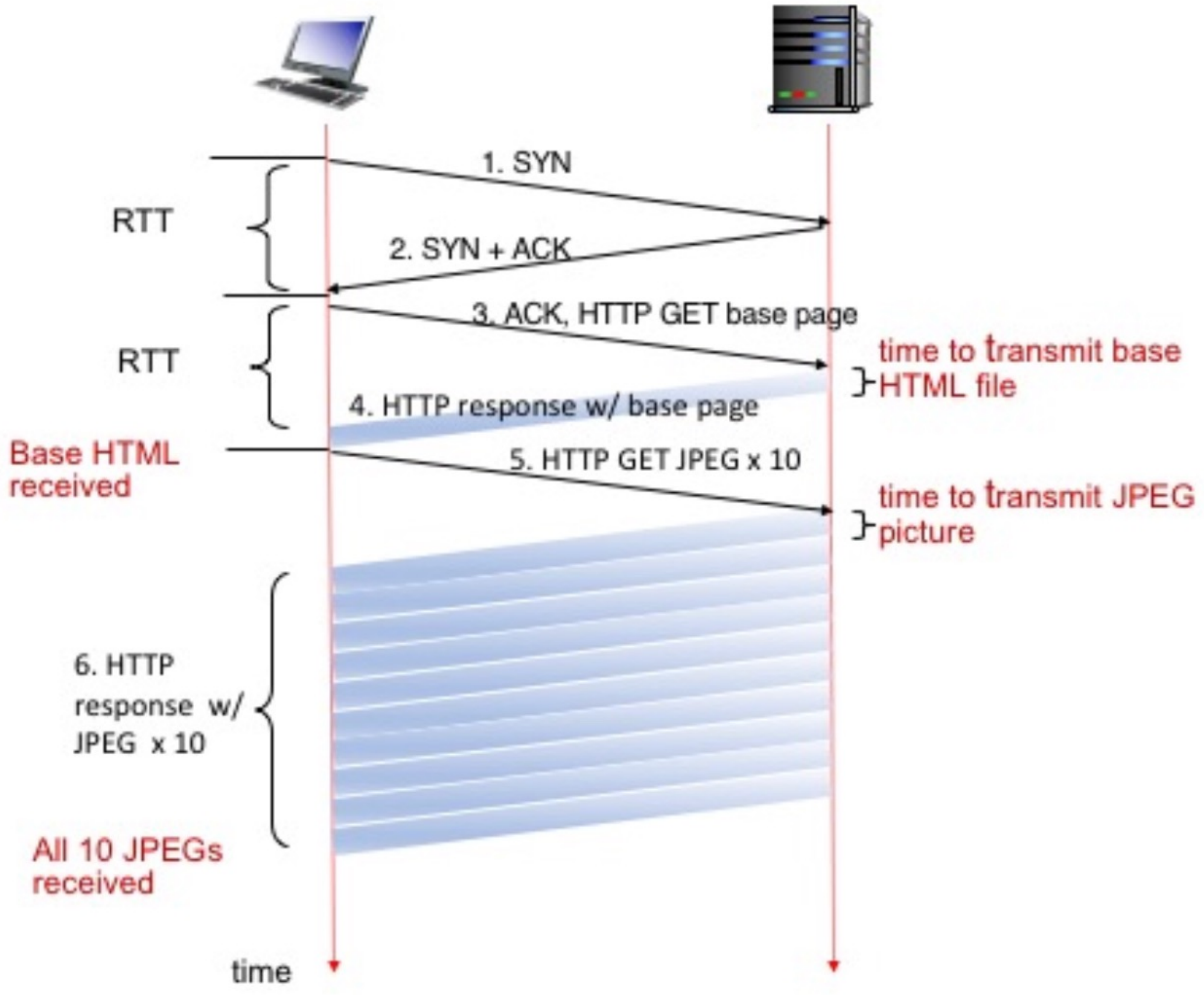
dig +noedns @ns1-proddns.glbdns.o365filtering.com mit-edu.mail.protection.outlook.com

Davids-SUTD-Powerbook-2018:~ david_yau$ dig +noedns @ns1-proddns.glbdns.o365filtering.com mit-edu.mail.protection.outlook.com

```
; <<>> DiG 9.10.6 <<>> +noedns @ns1-proddns.glbdns.o365filtering.com mit-
edu.mail.protection.outlook.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33606
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;mit-edu.mail.protection.outlook.com. IN A

;; ANSWER SECTION:
mit-edu.mail.protection.outlook.com. 10 IN A 104.47.41.36
mit-edu.mail.protection.outlook.com. 10 IN A 104.47.42.36

;; Query time: 223 msec
;; SERVER: 104.47.29.42#53(104.47.29.42)
;; WHEN: Sun Apr 14 19:36:15 HKT 2019
;; MSG SIZE  rcvd: 155
```

7. (i) Take note of the flags: aa means authoritative answer. (ii) No, the flag rd is set, but not ra. (iii) Yes, they match.

I. Activity NS 5.1



II. **Activity NS 5.2**

The telnet (or nc on OS/X High Sierra) connects you to TCP port 80 (HTTP) of www.sutd.edu.sg (web server). After you entered the minimal HTTP GET request (remember to enter a blank line after the sole header line), you should see a text message sent by the web server. The text is an HTTP response of code 302 Redirect, with a pointer to https://www.sutd.edu.sg/education/. A real browser would follow the redirect and load the page at the new URL (complete with embedded pictures, buttons, etc.) automatically. But you as a human user is interacting with the server using a text-only terminal, and you will have to interpret the response and follow it manually.

The openssl command connects you to TCP port 443 (HTTPS) of the web server. The minimal HTTP GET request now returns text that corresponds to the HTML content at www.sutd.edu.sg/education.

III. **Activity NS 5.3**

60% of the requests need to use the access link, so the average data rate through the link is $1.5 \times 0.6 = 9$ Mb/s, and the utilization of the link is $\frac{9}{1.54} = 0.584$.

Because the link utilization is below 0.7, the access link delay is several milliseconds (i.e., ~ms). The average delay for getting an object is 0.4 D_C + 0.6 D_N, where D_C is the delay if the object is served from the cache and D_N is the delay if it

is served from the public internet. D_N is dominated by the internet delay, so it is ~2s, and D_C is ~ms. Hence, the average delay is ~1.2s + ~ms, which is ~1.2s. Note that this delay is less than that for using a 15.4 Mbps access link.

IV. **Activity NS 5.4**

[http-ethereal-trace-3; message numbers below are according to the time order]

1. Request. GET request. HTTP 1.1. (See Message 8.)

2. Client was a Mozilla Netscape browser. It accepts a variety of objects including XML and HTML text, JPEG and GIF images, etc. It supports gzip compression.

3. The request took 57.19ms. (Inspect Message 14, the HTTP response.)

4. The request was successful. HTML text was returned. The size was 4500 bytes.

[http-ethereal-trace-4; message numbers are according to the time order]

1. URL is http://gaia.cs.umass.edu/ethereal-labs/lab2-4.html. HTTP 1.1. (See Message 10.)

2. Two embedded objects. One is a GIF image; the other is a JPEG image. (See Messages 17 and 20.)

3. New TCP connections were used. For the GIF (requested from 165.193.123.218), see Message 13, which is a TCP SYN to 165.193.123.218. For the JPEG, see Message 14 similarly.

4. The base page was obtained from server gaia.cs.umass.edu. The two images were obtained from servers www.aw-bc.com and manic.cs.umass.edu, respectively. Since all the three servers were different, the browser could not reuse TCP connections even though HTTP 1.1 allows persistent connections.

5. The connections were in parallel. (The second 3-way handshake to 134.241.6.82 for opening a TCP connection started before the first one to 165.193.123.218 finished.)