CS/ECE 438 HW 2 - Application & Transport Layer

Jiaxin Wu, Jie Wang

TOTAL POINTS

70 / 80

QUESTION 1

1 DNS 9 / 14

- **0 pts** Correct
- **1 pts** 1a
- **1 pts** 1b
- **1 pts** 1c
- **1 pts** 1d
- **1 pts** 1e
- **1 pts** 1.2 Minor error
- **2 pts** 1.2 Major error
- **3 pts** 1.2 Unresolved
- **1 pts** 1.3 Minor error
- **2 pts** 1.3 Major error
- **3 pts** 1.3 Unresolved
- **1 pts** 1.4 Minor error
- **2 pts** 1.4 Major error
- **3 pts** 1.4 Unresolved

✓ **- 2.5 pts** *1.1: partial name server provided*

✓ **- 2.5 pts** *1.3: partial name server provided*

QUESTION 2

2 Client-Server 14 / 14

✓ **- 0 pts** *Correct*

- **1 pts** 2.1 Minor error
- **2 pts** 2.1 Major error
- **4 pts** 2.1 Unresolved
- **1 pts** 2.2 Minor error
- **2 pts** 2.2 Major error
- **4 pts** Unresolved
- **1 pts** 2.3 Minor error
- **3 pts** 2.3 Major error
- **6 pts** Unresolved

QUESTION 3

3 BitTorrent 12 / 16

- **0 pts** Correct
- **1 pts** 3.1 Minor error
- **2 pts** 3.1 Major error
- **4 pts** 3.1 Wrong approach
- **1 pts** 3.2 Minor error
- **2 pts** 3.2 Major error
- **4 pts** 3.2 Wrong approach
- **1 pts** 3.3 Minor error
- **2 pts** 3.3 Major error
- **4 pts** 3.3 Wrong approach

✓ **- 4 pts** *3.4 Wrong answer*

- **2 pts** 3.4 Not answered what is asked

QUESTION 4

4 DHT 16 / 16

✓ **- 0 pts** *Correct*

4.1

- **1 pts** Minor mistake
- **2 pts** Wrong answer or Unresolved

4.2

- **1 pts** Minor mistakes

- **2 pts** Wrong
- **1 pts** 4.3 Minor error
- **2 pts** 4.3 Major error
- **3 pts** 4.3 Unresolved
- **1 pts** 4.4 Minor error
- **3 pts** 4.4 Unresolved
- **1 pts** 4.5 Minor Error
- **2 pts** 4.5 Unresolved
- **4 pts** 4.6 Unresolved
- **2 pts** 4.4 Major error
- **2 pts** Not accurate or no argument
- **3 pts** 6.4 Unresolved

QUESTION 5

5 HTTP Connections 8 / 8

✓ **- 0 pts** *Correct*

- **2 pts** 5.1 Not accurate
- **4 pts** 5.1 Unresolved
- **2 pts** 5.2 Not accurate
- **4 pts** 5.2 Unresolved

QUESTION 6

6 Reliable Data Transfer 11 / 12

- **0 pts** Correct
- **1 pts** 6.1 Minor Issue
- **2 pts** 6.1 Major Issue
- **3 pts** 6.1 Unresolved
- **1 pts** 6.2 Minor Issue
- **2 pts** 6.2 Major Issue
- **3 pts** 6.2 Unresolved

✓ **- 1 pts** *6.3 Minor Issue*

- **2 pts** 6.3 Major Issue
- **3 pts** 6.3 Unresolved
- **1 pts** 6.4 Minor Issue
- **2 pts** 6.4 Major Issue

CS438 Assignment 2

02/17/2023

Wang, Jie [jiew5]
Wu, Jiaxin [jiaxin19]

1 DNS

The task requires using the **dig** command to provide answers. To ensure accurate results, it is recommended to perform these steps from a computer located on a campus network. The user can refer to the dig documentation to understand how to utilize it.

Notice: here we use our apartments' WIFI to dig the www.eecs.mit.edu
the miteecs.wpengine.com is intermediate web ISP provide by Google
We tried using rvpn to connect UIUC campus network, but it dons't work for our virtual box.

1. Starting from one of the root servers *a–m.root-servers.net*, perform an iterative lookup for the host www.eecs.mit.edu. For instance, you can initiate the search by using the following command:

dig @h.root-servers.net www.eecs.mit.edu

Please provide a list of the following information for each name server you visit during the lookup process:

- (a) Can you specify the **domain name** of the **name server** being visited?
    - **eecs.mit.edu**

  - TLD: edu. 172800 IN NS a.edu-servers.net.TLD: edu. 172800 IN NS a.edu-servers.net.

- (b) Can you provide the IP address of the name server that is currently being used?
    - **35.231.163.5**
- (c) How long did the query take?
    - **Query time: 95 msec**

- (d) What is the round-trip time (RTT) to the server, which can be determined by using the "ping" command to connect to the server?
    - **RTT** = min/avg/max/mdev = 89.990/90.973/99.823/2.098 ms
- (e) For how long can you store the results in cache?

TTL = 392 IN CNAME miteecs.wpengine.com.
TTL = 106 IN A 35.231.163.5 (miteecs.wpengine.com.)
TTL = 172800 IN NS a.gtld-servers.net. (the root server, there are a-m different root server, here I only display one)

2. Perform a recursive query using resolver.illinois.edu of the name www.eecs.mit.edu.

- Was this query faster or slower than the sum of the iterative steps?
- Why do you think that is?

Behavior:

- For the first time, It is much slower than the iterative steps.
    - recursive : 176 ms
    - iterative : 90 ms
- At the second time, it is much faster!
    - recursive : 4 ms
    - iterative : 92 ms

Possible Reason:

- At first, we are at the campus circle, whose ISP doesn't directly link to resolver.illinois.edu, the first time the dig need to ask for the request from UIUC resolver
    - At second time, there are already cache to the target server and resolver.illinois.edu, so it performs much faster.

```
wjx@wjx-VirtualBox:~$ dig +recurse @resolver.illinois.edu www.eecs.mit.edu

; <<>> DiG 9.16.1-Ubuntu <<>> +recurse @resolver.illinois.edu www.eecs.mit.edu
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62111
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: d3d1ea6b21951a22623dd96263f16702bb6f273da6f20dea (good)
;; QUESTION SECTION:
;www.eecs.mit.edu.              IN      A

;; ANSWER SECTION:
www.eecs.mit.edu.       900     IN      CNAME   miteecs.wpengine.com.
miteecs.wpengine.com.   120     IN      A       35.231.163.5

;; Query time: 176 msec
;; SERVER: 130.126.2.131#53(130.126.2.131)
;; WHEN: Sat Feb 18 18:02:10 CST 2023
;; MSG SIZE  rcvd: 123
```

Second time we run the command:

the query time is much faster!

```
wjx@wjx-VirtualBox:~$ dig +recurse @resolver.illinois.edu www.eecs.mit.edu

; <<>> DiG 9.16.1-Ubuntu <<>> +recurse @resolver.illinois.edu www.eecs.mit.edu
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27629
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 825a94d5fd81de78b071959663f1685a79bfc283f5093d4a (good)
;; QUESTION SECTION:
;www.eecs.mit.edu.              IN      A

;; ANSWER SECTION:
www.eecs.mit.edu.       554     IN      CNAME   miteecs.wpengine.com.
miteecs.wpengine.com.   120     IN      A       35.231.163.5

;; Query time: 4 msec
;; SERVER: 130.126.2.131#53(130.126.2.131)
;; WHEN: Sat Feb 18 18:07:54 CST 2023
;; MSG SIZE  rcvd: 123
```

a recursive query may be faster than a sum of iterative steps, because the resolver can cache intermediate results and reuse them in subsequent queries. This can reduce the overall query time, especially for frequently accessed domain names.

3. Perform an iterative reverse-mapping query for the address of www.eecs.mit.edu you found in the previous steps, using dig -x. List again the information asked in part 1.

(a) eecs.mit.edu -> CNAME miteecs.wpengine.com.

(b) SERVER: 66.253.214.16

(c) Query time: 16 msec

(d) RTT = min/avg/max/mdev = 0.013/0.031/0.044/0.007 ms

(e) LTT = 0.031 ms

4. Can you explain why the DNS protocol tends to utilize UDP rather than TCP, considering what has been previously discussed in this inquiry?

The DNS protocol tends to use UDP rather than TCP because the domain queries and responses process are typically small and can be transmitted efficiently with UDP. The content information is provided by the server online, while DNS do not need to establish any long-existing connection.

In addition, it is important to avoid using UDP for DNS helps to reduce the load on DNS servers and the network as a whole, as UDP messages have a smaller overhead and do not require the same level of reliability as TCP. However, in some cases, DNS queries and responses may be too large for UDP, and in those situations, the protocol can switch to TCP as needed.

Good Reference:

(129条消息) DNS基础之使用dig查询DNS解析过程*dig指定dns查询*夜已如歌_ok的博客-CSDN博客

(129条消息) Linux下解析域名命令-dig 命令使用详解*linux解析域名的命*令thlzjfefe的博客-CSDN博客

1 DNS **9 / 14**

- **0 pts** Correct
- **1 pts** 1a
- **1 pts** 1b
- **1 pts** 1c
- **1 pts** 1d
- **1 pts** 1e
- **1 pts** 1.2 Minor error
- **2 pts** 1.2 Major error
- **3 pts** 1.2 Unresolved
- **1 pts** 1.3 Minor error
- **2 pts** 1.3 Major error
- **3 pts** 1.3 Unresolved
- **1 pts** 1.4 Minor error
- **2 pts** 1.4 Major error
- **3 pts** 1.4 Unresolved

✓ **- 2.5 pts** *1.1: partial name server provided*

✓ **- 2.5 pts** *1.3: partial name server provided*

2 Client-Server

Think about spreading a F-bit file among N peers using a client-server structure. Let the server have an maximum upload capacity μ_s, and each client c has a download capacity d_c. Let $d_{min} = \min_c d_c$ be the minimum download rate. Assume that the server can serve multiple clients simultaneously and fluidly set the rate for each client, r_c, as long as $\forall_c r_c \leq d_c$ and $\sum_c r_c \leq \mu_s$.

1. Suppose that $\mu_s/N \leq d_{min}$. How would you set the rates r_c so that the file is fully distributed to all clients in a minimum time? (i.e., you are minimizing the time that the slowest client receives the file.) What would the distribution time be?
2. Suppose now that $\mu_s/N > d_{min}$. How would you set the rates r_c now to fully distribute the file to the clients in a minimum time? And what would this time be?
3. Consider a concrete example with 5 clients with $d_c = \{06, 12, 18, 24, 30\}$ and a server upload capacity of $\mu_s = 30$. How would you set the rates to get the smallest average download time *without* increasing the total distribution time from the previous part?

Ans:

1. Since $\mu_s/N \leq d_{min}$, the $r_c \leq d_c$ should be always true for arbitrary c, the data transfer speed is limited by r_c

   then we can let $\sum_c r_c = N \times r_c = \mu_s$, i.e. $r_c = \mu_s/N$

   As a result, the minimum distribution time is $T_1 = F/r_c = F * N/\mu_s$
2. Because $\mu_s/N > d_{min}$, the $r_c \leq d_c$ may not be always true for arbitrary c, the data transfer speed is limited by d_{min}

Then maximum $r_c = d_{min}$, it is the rate we need to set for all clients. (Avoid congestion at the client with smallest download bandwith)

> rc: download rate for a client set by the server.
> dc: max download rate for a client
> Therefore, the actual download speed for a client would be max{rc, dc}

As a result, the minimum distribution time is $T_2 = F/r_c = F/d_{min}$

3. **We assume the r_c can't change after initial setting**,

   $\mu_s/N = 30/5 = 06 \leq d_{min}$, then we can let $\sum_c r_c = N \times r_c = \mu_s$,

   i.e. $r_c = \mu_s/N = 30/5 = 6$

   As a result, the minimum distribution time is $T_3 = F/6$

2 Client-Server **14 / 14**

✓ **- 0 pts** *Correct*

- 1 pts 2.1 Minor error

- 2 pts 2.1 Major error

- 4 pts 2.1 Unresolved

- 1 pts 2.2 Minor error

- 2 pts 2.2 Major error

- 4 pts Unresolved

- 1 pts 2.3 Minor error

- 3 pts 2.3 Major error

- 6 pts Unresolved

3. BitTorrent

BitTorrent employs a choking mechanism for distributing bandwidth to peers. It selects the four peers that have provided the best download performance using a tit-for-tat strategy and one additional peer chosen randomly, referred to as an "optimistic unchoking". The selection of the four best peers and the random choice are updated every second.

Suppose Bob joins a BitTorrent swarm with 30 other peers. Let each peer, including Bob, have an upload speed of 80 $Mbps$ and unlimited download speed. Consider what happens in the phase of the protocol where for each pair of peers A and B, A has some blocks that B wants and vice versa; i.e, any peer can productively download data from any other peer.

1. If Bob intends to not upload any data and become a free rider, what would his average download speed be?
2. What would be the average download rate for the remaining peers?
3. Suppose that Bob runs a second client that pretends to be a separate peer, who also becomes a free rider. How fast can Bob download data now?
4. Suppose Bob switches his two clients to the regular BitTorrent code and they both start uploading as well. Will Bob's performance see a noticeable improvement compared to when he was not contributing?

1. If Bob does not upload any data, he will not be selected as one of anyone's four best peers. He can download from others only if he is selected as the additional random peer. The opportunity of one peer selecting Bob as the random peer is $\frac{1}{30-4} = \frac{1}{26}$. On average, Bob is the additional peer of $30 \times \frac{1}{26} = \frac{15}{13}$ peers. So his average download speed is $80 \div 5 \times \frac{15}{13} = 18.46 Mbps$.
2. The overall upload rate is $80 Mbps \times 30 = 2400 Mbps$. The average download rate for the remaining peers is $(2400 - 18.46) \div 30 = 79.38 Mbps$.
3. Each of Bob's client has an opportunity of $\frac{1}{31-4} = \frac{1}{27}$ to be selected by one peer as the additional random peer. On average, each of Bob's client is the additional peer of $30 \times \frac{1}{27} = \frac{10}{9}$ peers. So the average download rate for Bob is $2 \times 80 \div 5 \times \frac{10}{9} = 35.56 Mbps$.
4. If Bob's two clients start uploading, they can become the top four providers of some other peers, so they can download from more peers than before. On average, the download rate should be 80Mbps for each peer if everyone is uploading. So Bob's downloading performance should see a noticeable improvement.

3 BitTorrent **12 / 16**

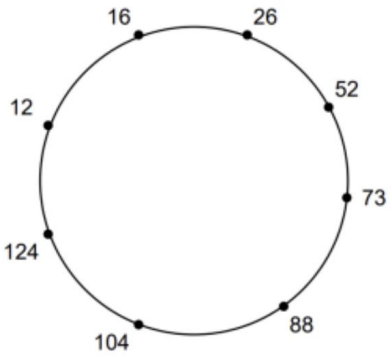
- **0 pts** Correct
- **1 pts** 3.1 Minor error
- **2 pts** 3.1 Major error
- **4 pts** 3.1 Wrong approach
- **1 pts** 3.2 Minor error
- **2 pts** 3.2 Major error
- **4 pts** 3.2 Wrong approach
- **1 pts** 3.3 Minor error
- **2 pts** 3.3 Major error
- **4 pts** 3.3 Wrong approach

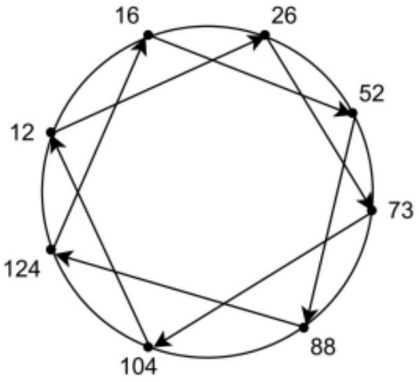
✓ **- 4 pts** *3.4 Wrong answer*

- **2 pts** 3.4 Not answered what is asked

4 . DHT

Consider a distributed hash table (DHT) with 8 peers and an 8-bit hash ID that operates in a circular fashion. The peers are represented by the values $\{12, 16, 26, 52, 73, 88, 104, 124\}$. The key-value pairs are assigned to the peer that immediately follows the key in the circular DHT.





1. What is hashing ID space?
2. In which peer is the key-value pair stored in the following circumstances:
   (a) The key has a value of 19?
   (b) The key has a value of 125?
3. If peer 88 initiates a search for the value associated with key 23, as shown in the illustration on the left, how many messages will be necessary to complete the query? This includes the message that initiates the query and the message that returns the value. Provide an explanation for the answer?
4. Assume each peer is given one cord and is now aware of the next two successors as shown in the right figure above. If peer 12 starts a query for the value associated with key 111, then how many messages (including the query message and return value message) are required to resolve this query? Explain why?
5. What is the maximum number of messages to resolve any query if
   (a) each peer is aware of its immediate successor as shown in the left figure above?
   (b) each peer is given one cord and is now aware of the next two successors?
6. Given that each peer is provided with one cord, can you propose a new arrangement of the cords that would minimize the maximum number of messages needed to resolve any query? Draw a representation of the circular distributed hash table and provide an argument to support why your solution is effective.

Ans:

1. Because id is 8-bit, so the maximum Hashing ID# $= 2^8 = 256$

   And the Hashing ID space = $\{0, 1, 2, 3, \ldots, 255\}$

2.

	key_value	peer_value
(a)	19	**26**
(b)	125	**12**

Because:

- rule: assign key-value pair to the peer that has the closest ID
- convention: closest peer is the immediate successor of the key

3. **6 times,**
   - the key 23 is associated with peer 26, there are 5 nodes between 26 and 88 $\{104, 124, 12, 16, 26\}$
   - node **26** needs one return value message, which is counted as one additional message

   therefore, it takes 6 messages in total

4. **5 times,**
   - the key 111 is associated with peer 124, there are 4 nodes between peers 12 and 124 $\{26, 73, 104, 124\}$ (using shortcuts)
   - node **124** needs one return value message, which is counted as one additional message

   therefore, it takes 5 messages in total

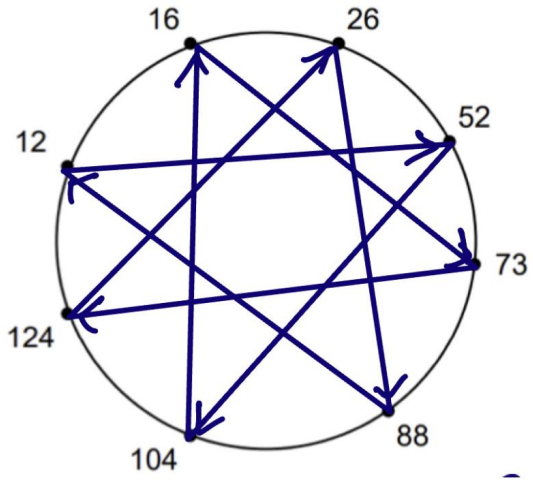
5. (a) the maximum time happens when the target peer is just the predecessor of initiation peer, i.e. the search goes through the whole circle table.

   $\rightarrow$ **8 times**

   (b) First we clarify, for current peer, its **next two successors are**
   - next node by walking the circle
   - next node by walking the cord

   Then, similar to (a) and 4, the maximum messages number is **5**

6. According to symmetry rule, each cord of a DHT with minimum inquiry time should have peer nodes with equal intervals. The following figure is the optimal cord arrangement diagram drawn by me:



   In which, the maximum messages number in each query search is **4.**
   - Take $16 \rightarrow 12$ as an example, it goes through $\{73, 124, 12\}$ 3 nodes, and need one extra response message. Since 12 is the predecessor of 16, we can confirm this is the longest routine.

   All other combinations are draw and test to be inefficient compared to this one.

Proof of the optimization of my arrangement:

Assume the minimized maximum number of messages needed is 3. (*)

Then for each search, there is at most one peer between the initial peer and the target peer. So the target peer must be an immediate or secondary successor of the initial peer.

Since each peer has one successor by walking the circle and another by walking the cord,each peer has two successors, adding up to 4 secondary successors. So each peer can have at most 2+4=6 peers which it can reach within two steps (three messages, which ack response).

However, there are 8 peers in total, so each peer has at least one peer requiring more than three messages. This implies assumption (*) contradicts to the problem, we can say there do not exist such an arrangement better than mine.

Therefore, the arrangement I draw is optimized.
Q.E.D.

4 DHT **16 / 16**

✓ **- 0 pts** *Correct*

4.1

- 1 pts Minor mistake

- 2 pts Wrong answer or Unresolved

4.2

- 1 pts Minor mistakes

- 2 pts Wrong

- 1 pts 4.3 Minor error

- 2 pts 4.3 Major error

- 3 pts 4.3 Unresolved

- 1 pts 4.4 Minor error

- 3 pts 4.4 Unresolved

- 1 pts 4.5 Minor Error

- 2 pts 4.5 Unresolved

- 4 pts 4.6 Unresolved

- 2 pts 4.4 Major error

- 2 pts Not accurate or no argument

5. HTTP Connections

1. How does a web server handle multiple HTTP connections at the same time, even though all of the data packets received from different sessions are all directed to its TCP port 80? In other words, how does the server ensure that each packet is correctly delivered to its corresponding socket?
2. Is there a restriction on the number of simultaneous connections a single host can make with a web server, such as "a host can only maintain one HTTP connection with the server at any given time"? If such limitations exist, please describe them. If not, please explain the reasons why there are no limitations.

Ans:

1. **Server based on Socket**

Socket distribution connects HTTP request with corresponding destination ip.

When a web server receives HTTP requests from multiple clients, each connection is established on a different socket at the server. Here although different clients used same port 80, they can be redirect back to the according socket and the header info.

Take our MP1 code for an example, the code is based on Linux system, and everything in Linux is a file. Therefore, the socket is distributed by different socket file descriptor (sockfd). Each time the server check the sockfd to accept new request, and fork the process to send back necessary HTTP header and file (if 200 OK).

In summary, it is a demultiplexing procedure, where web servers differentiate between different connections using sockets, allowing it to handle multiple HTTP connections simultaneously.

2. The answer is different on different operating system and HTTP architecture.

But, the overall restriction is the hardware limit and the socket fd number.

Take Linux for an example, since the socket file descriptor is a int16_t number, the maximum can only be 2^15 = 32768

However, you can still use some technique like CDN to enlarge this maximum. For instance, we can construct distributed server in different region, and connect user's request with the nearest one. Therefore, we can handle high-concurrency requirement in the application. For instance, the Amazon server during the Black Friday.

5 HTTP Connections **8 / 8**

✓ **- 0 pts** *Correct*

- 2 pts 5.1 Not accurate

- 4 pts 5.1 Unresolved

- 2 pts 5.2 Not accurate

- 4 pts 5.2 Unresolved

6. Reliable Data Transfer

1. How can the maximum utilization of the network be calculated, when Alice is sending data to Bob using the stop-and-wait method, given that the bottleneck bandwidth is 40 Mbps and the end-to-end delay is 25 ms, and each packet being sent is 1200 bytes with 32 bits reserved for the checksum and sequence number?
2. How should Alice and Bob set their timeout value?
3. What changes would you make to the protocols if the connection between Alice and Bob is ensured to not cause any packet damage?
4. What would be the impact on performance of this change?

1. The transmission dalay for a packet is $D_{trans} = \frac{L}{R} = \frac{1200bytes}{40\times10^6 bits/sec} = 0.24ms$
   The transmission delay for the reserved bits of a packet is $D'_{trans} = \frac{L_{reserved}}{R} = \frac{32bits}{40\times10^6 bits/sec} = 0.0008ms$
   The network utilization is $U = \frac{D_{trans}-D'_{trans}}{RTT+D_{trans}} = \frac{0.24-0.0008}{25\times2+0.24} = 0.004757$
2. The timeout value should be not be too low to avoid unneccessary resending. It should not be too high as well, or otherwise it takes more time for the sender to detect a loss and resend the packet.
   At least, it must be bigger than 50.2408ms.
3. They can use pipelined protocols where Alice can allow multiple, yet-to-be-acknowledged packets in pipeline.
4. The utilization of the network will be increased.
   The original sender utilization is $U_{sender} = \frac{L/R}{RTT+L/R}$ After using pipelined protocols, if Alice can allow N unacked packets in pipeline, the sender utilization becomes $U_{sender} = \frac{N*L/R}{RTT+L/R}$

Useful Reference:

https://campuswire.com/c/G8B863314/feed/399

Reliable Data Transfer (umn.edu)

6 Reliable Data Transfer **11 / 12**

- **0 pts** Correct
- **1 pts** 6.1 Minor Issue
- **2 pts** 6.1 Major Issue
- **3 pts** 6.1 Unresolved
- **1 pts** 6.2 Minor Issue
- **2 pts** 6.2 Major Issue
- **3 pts** 6.2 Unresolved

✓ **- 1 pts** *6.3 Minor Issue*

- **2 pts** 6.3 Major Issue
- **3 pts** 6.3 Unresolved
- **1 pts** 6.4 Minor Issue
- **2 pts** 6.4 Major Issue
- **3 pts** 6.4 Unresolved