CMPT 371 Assignment 2

Junchen Li (301385486)
Wenqing Liu (301417132)

2021/7/10

1) Consider a host running a IPv4 resolver and a local DNS server. Do not consider IPv6 AAAA records. Assume the domain names given in A, B and C are hosts (not DNS servers). Assume the following domains are delegated by the DNS server for the nodes above them.

   ca. com. gov. postulates.ca. seas.com. mammal.gov. integers.postulates.ca. beaches.seas.com.

   Assume all other domains are not delegated.
   Both the host and the server have recently been rebooted. The server has made three queries since it was rebooted. The queries were for the A records for each of the following hosts. The hosts (not running DNS servers) are listed in the order the queries are made.

   i. numbers.integers.postulates.ca
   ii. landscapes.beaches.seas.com.
   iii. lion.cat.mammal.gov.

   Answers should all be in the form of DNS records for example
   - NS record for each DNS server for domain fresh.fruit.gov.
   - A record for each DNS server for domain favorit.color.ca.
   - AAAA record for fish.shark.seafood.com.

   a) **[4 points]** What DNS records would you expect to find in the cache of the local DNS server after the queries to resolve the address for numbers.integers.postulates.ca.?

Answer:

NS and A record for integers.postulates.ca.
NS and A record for postulates.ca.
NS and A record for ca.

   b) **[6 points]** Consider each of the following hosts. For each host answer the following two questions.
      - Which DNS server would be authoritative for each of the following hosts?
      - Why is that DNS server you chose authoritative for the host?

      I. Tiger.cat.mammal.gov.

Answer:

The mammal.gov. Because of this is the longest match domain name that has been delegated.

      II. Colorful.sunsets.beaches.seas.com.

Answer:

beaches.seas.com. Because of this is the longest match domain name that has been delegated. sunsets.beaches.seas.com has not been delegated.

III. Arithmetic.integers.postulates.ca.

Answer:

integers.postulates.ca. Because of this is the longest match domain name that has been delegated

c) **[6 points]** After the three queries above for hosts i, ii, and iii, an additional query for host wolf.mammal.gov. is made.

I. **[2 points]** What is the first DNS server queried? Why?

Answer:

query for the ip of wolf.mammal.gov. to mammal.gov.

Because of mammal.gov. is the longest match domain name could be found in the cache.

II. **[2 points]** What DNS record is the query requesting, in the query to the DNS server in 1?

Answer: A record

III. **[2 points]** What is returned by DNS query in II?

Answer: IP address of wolf.mammal.gov.

d) **[4 points]** For each of the following two domains you will state which DNS server would be contacted first and the number of iterative queries needed to obtain an answer to the query.

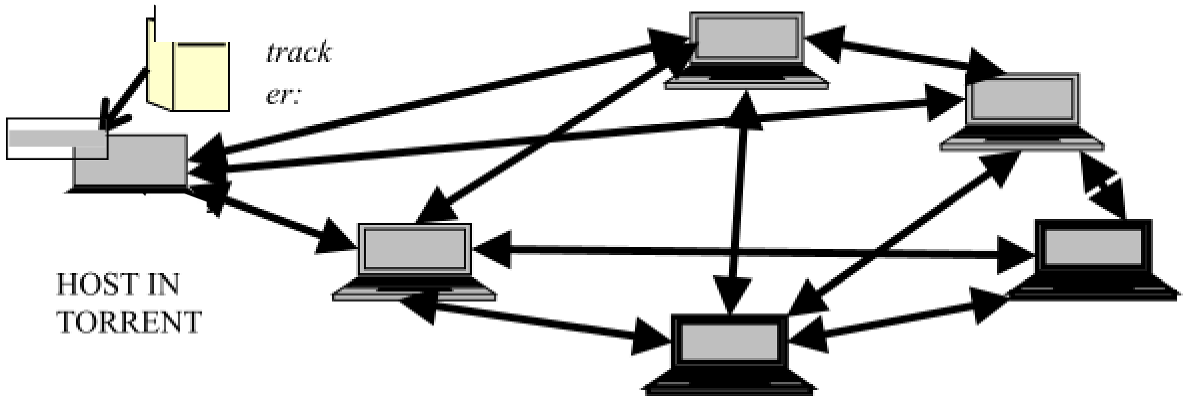
I. After the query for wolf.mammal.gov. the next query was for sandy.beaches.seas.com.

Answer: The beaches.seas.com. would be contacted first,
1 iterative query

II. After the query for sandy.beaches.seas.com the next query was for a.b.c.d.edu

Answer: the root would be contacted first,
1 iterative query

2) **[15 points]** Consider bit torrent as an example of a peer to peer application.



The host labeled X has just joined the illustrated bit torrent. As indicated below the host is the newest host in the torrent. Answer the following questions. Each answer should be no longer than three sentences.

a) How does host X obtain an initial list of potential peers?

Answer: X obtains an initial list of potential peers by tracker. Tracker chooses potential peers from the list of current peers.

b) What is the purpose of the tracker?

Answer: The purpose of the tracker is managing the torrent. The tracker provides a list of potential peers to each peer and updates it periodically. Also, the tracker tracks which peer has which block of data. Then, tell each peer what blocks data its potential peers have now.

c) How does a potential peer from X's list become a neighboring peer?

Answer: To be a neighboring peer, a TCP connection must be built. So, when a peer is prepared to send data, it becomes a neighboring peer.

d) What does it mean to unchoke a peer?

Answer: X would like to choose some neighboring peers that send data oftenly and recently, and upload data to them. The receivers will be considered as unchoked. Other peers who do not receive the data are considered as choked.

e) What does it mean for a peer to be optimistically unchoked?

Answer: X will choose a new peer to exchange with randomly and periodically, and unchoke the chosen peer. X would hope the unchoked chosen peer will send it data back.

3) **[50 points]** You will write two Python (compatible with v3.5) socket programs to implement a slightly modified version of the protocol (rdt3.0) from your text (that we will/have discussed in class. These two processes will exchange simple packets that are used to implement the protocol. Please note that the form of the packets has been changed from the example in your textbook. This means that the arguments of the makepkt function in the figures must be changed to a list of the variables in the fields of the packets described below. The arguments of makepkt should be in the order Field1, Field2, Field3, Field4
Your programs must run on the Linux machines in the CSIL labs. If your programs do not run in the Linux environment in CSIL you will receive a 0 for this problem. **Beware socket programs are often not easily portable between operating systems.**

The contents of the simple packets you send between your sender (client) and receiver (server) sockets will include 4 values.

- Field 1: Packet contents: a 32 bit integer (data).
  The integer cannot be assumed to follow any pattern Packet contents for an ACK or NACK packets is 0
- Field 2: Sequence Number for Packet: a Boolean Value is True if sequence number is 1 Value is False if sequence number is 0
- Field 3: Is this an ACK? Boolean
  True if the packet is an ACK, False if the packet is not an ACK
- Field 4: Sequence Number for ACK: a Boolean
  Value is True if sequence number is 1
  Value is False if sequence number is 0

In your **sender program** you will

- **Use one instance of a pseudo random number generator** random( ) to produce uniformly distributed random floating point numbers in the range [0.0, 5.0). These pseudo random numbers will be used to simulate random arrival time of packets. When a packet is sent, the send is followed immediately by a call to this random number generator. The pseudo random number returned will be interpreted as the delay in seconds before the next packet will be generated. This delay occurs after the process enters either "wait for call 0 from above" or "wait for call 1 from above". It is the time between entering one of these states and building the next packet (waiting for enough data for a packet).
- **Use a second instance of a pseudo random number generator** random() to generate random number that will be used to determine if an ACK ~~a nd NACK~~ that has just arrived has been corrupted. This instance of the pseudo random number generator random should generate uniformly distributed pseudo random numbers between [0.0 and 1.0). If the number generated is less that the input value of the probability that an ACK packet has been be corrupted, then the ACK packet that has just arrived will be considered to be corrupted.

- **Read the values of the following quantities** used in your program at the start of your program. You will not read any quantities not in this list into your program. You will read the specified quantities in the order they are specified below.
  - The seed for the random number generator used for timing
  - The number of packets to send
  - The seed for the random number generator used for determining if ACKs ~~o r NACKs~~ have been corrupted.
  - The probability that an ACK has been corrupted. Between [0 and 1)
  - The round trip travel time (to be used for timer for determining if a packet has not received an ACK)

- **Print three messages immediately before a packet is sent.**
  - You must print the messages exactly as given (character by character), with the sole exception of items in **bold** which will contain actual data values.
  - First message: one of the four following messages should be printed

    A packet with sequence number 0 is about to be sent

    A packet with sequence number 1 is about to be sent

    A packet with sequence number 0 is about to be resent

    A packet with sequence number 1 is about to be resent
  - Second message should be printed

    Packet to send contains: data = **123** seq = **0** isack=**False** ack = **0**
  - Third message: one of the two following messages should be printed.

    Starting timer for ACK0

    Starting timer for ACK1
- **Print the following messages immediately after an uncorrupted ACK packet is received.**
  - An ACK will be received. Print the message for the packet received followed by the third message.
  - You must print the messages exactly as given below, with the sole exception of items in **bold** which will contain actual data values.
  - First message: one of the two following messages should be printed.

    An ACK0 packet has just been received

    An ACK1 packet has just been received
  - Second message should be printed

    Packet received contains: data = **0** seq = **1** isack= ***False*** ack = **0**
  - Third message: one of the two following messages should be printed.

    Stopping timer for ACK0

    Stopping timer for ACK1
- **Print the following message immediately after a corrupted packet is received.**
  - You must print the message exactly as given below.

    A Corrupted ACK packet has just been received
- **Print the following messages if a timer expires.**

There are two reasons the timer may expire.
For a lost packet/ack the best way to handle this timeout is using the timeout associated with the socket recv().

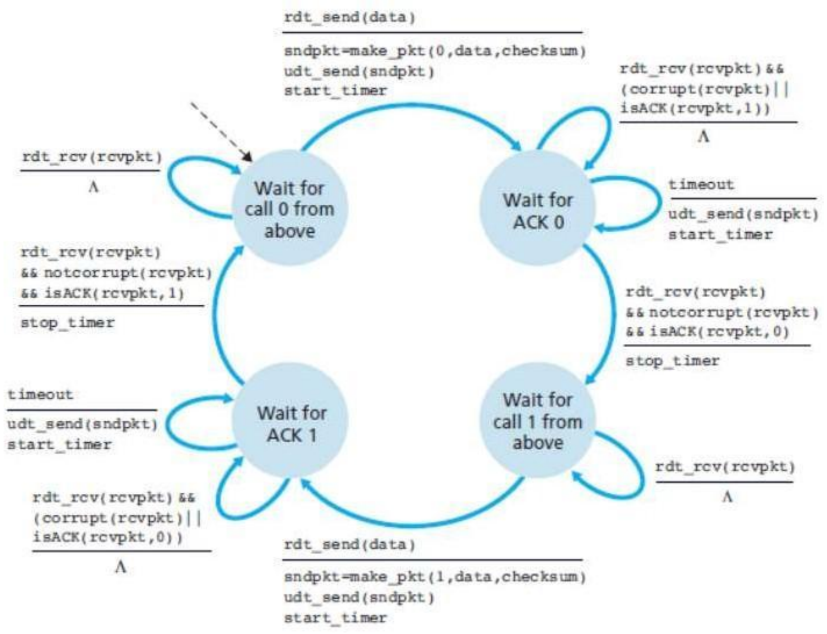
For a corrupted ack or an incorrect sequence number something will be received before the timeout. You will need to check what is received, if it is a corrupted ack or incorrect ack you will need to monitor the time until the timer expires then send your next packet. You must print the message exactly as given below.

- One of the two following messages should be printed.

  ```
  ACK0 timer expired
  ACK1 timer expired
  ```

- **Print the following messages immediately before the sender moves to another state or returns to the same state**.
    - Only the appropriate one of the following messages should be printed for each transition.
    - These messages must be printed exactly as given below.

        The sender is moving to state WAIT FOR CALL 0 FROM ABOVE
        The sender is moving to state WAIT FOR CALL 1 FROM ABOVE
        The sender is moving back to state WAIT FOR CALL 0 FROM ABOVE
        The sender is moving back to state WAIT FOR CALL 1 FROM ABOVE
        The sender is moving to state WAIT FOR ACK0
        The sender is moving to state WAIT FOR ACK1
        The sender is moving back to state WAIT FOR ACK0
        The sender is moving back to state WAIT FOR ACK1


rdt_send(data)
sndpkt=make_pkt(0,data,checksum)
udt_send(sndpkt)
start_timer
rdt_rcv(rcvpkt) && (corrupt(rcvpkt)|| isACK(rcvpkt,1))
Λ
rdt_rcv(rcvpkt)
Λ
Wait for call 0 from above
Wait for ACK 0
timeout
udt_send(sndpkt)
start_timer
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(rcvpkt,1)
stop_timer
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(rcvpkt,0)
stop_timer
timeout
udt_send(sndpkt)
start_timer
Wait for ACK 1
Wait for call 1 from above
rdt_rcv(rcvpkt)
Λ
rdt_rcv(rcvpkt) && (corrupt(rcvpkt)|| isACK(rcvpkt,0))
Λ
rdt_send(data)
sndpkt=make_pkt(1,data,checksum)
udt_send(sndpkt)
start_timer

In your **receiver program** you will

- **Use** a**n instance of a pseudo random number generator** random() to generate uniformly distributed pseudo random numbers between [0.0 and 1.0). These pseudo random numbers will be used to determine if a packet that has just arrived has been corrupted. If the pseudo random number generated is less that the input value of the probability that the packet has been be corrupted, then the packet that has just arrived will be considered to be corrupted.
- **Read the values of the following quantities** used in your program at the start of your program. You will not read any quantities not in this list into your program. You will read the specified quantities in the order they are specified below.
    - The seed for the random number generator used for determining if packets have been corrupted.

   - The probability that a packet has been corrupted

- **Print the following messages immediately before an ACK is sent.**
   - One of the first two messages should be sent followed by the third message.
   - You must print the messages exactly as given, with the sole exception of items in

     bold which will contain actual data values.

       An ACK0 is about to be sent
       An ACK1 is about to be sent
       Packet to send contains: data = 0 seq = 0 ***isack = True ack = 0***

- **Print the following messages immediately after an uncorrupted packet is received.**
   - One of the first four messages should be sent followed by the fifth message, including the actual contents of the packet just received
   - You must print the messages exactly as given, with the sole exception of items in

     bold which will contain actual data values.

       A packet with sequence number 0 has been received
       A packet with sequence number 1 has been received
       A duplicate packet with sequence number 0 has been received
       A duplicate packet with sequence number 1 has been received
       Packet received contains: data 333 seq = ***0 isack = True ack = 1***

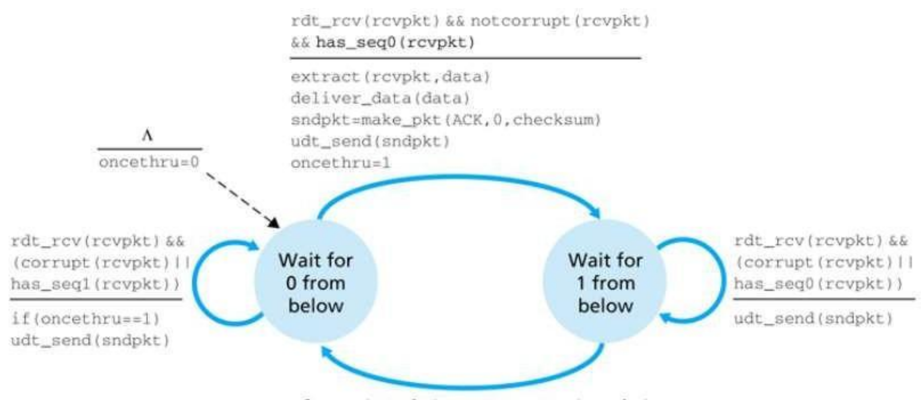
- **Print the following message immediately after a corrupted packet is received.**

  You must print the message exactly as given.

    A Corrupted packet has just been received

- **Print the following messages Immediately before the sender moves to another state or returns to the same state**
   - the appropriate one of the following messages should be printed

       The receiver is moving back to state WAIT FOR 0 FROM BELOW
       The receiver is moving back to state WAIT FOR 1 FROM BELOW
       The receiver is moving to state WAIT FOR 0 FROM BELOW
       The receiver is moving to state WAIT FOR 1 FROM BELOW


rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq0(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pkt(ACK,0,checksum)
udt_send(sndpkt)
oncethru=1
Λ
oncethru=0
rdt_rcv(rcvpkt) && (corrupt(rcvpkt) || has_seq1(rcvpkt))
if(oncethru==1)
udt_send(sndpkt)
Wait for 0 from below
Wait for 1 from below
rdt_rcv(rcvpkt) && (corrupt(rcvpkt) || has_seq0(rcvpkt))
udt_send(sndpkt)

4) Consider a link with the following properties:
- 75-meter link length
- Bit rate of 5Mibps (1Mib = 2^{20} bits) in each direction.
- Propagation velocity through the link is $2.5*10^8$ m/s.
- Size of one HTTP response (containing one HTTP object) is 80Kibits bits (1Kibit = 2^{10} `bits) including headers.
- Packets containing HTTP requests, SYNs, ACKs, and FINs are 1024 bits long including headers.

Assume that a request is made for a particular webpage. The initial response is a single object. While processing that single object 13 additional objects are requested. Consider how long it would take to obtain all 14 of these objects in each of the following scenarios. Include the time used by both the 3 way handshake to establish the TCP connection and the 3 way handshake to close the TCP connection.

a) **[5 points]** Assume that each object is requested using a non-persistent HTTP connection. Assume that only one non-persistent connections can be in use at any given time. Each non-persistent connection has a rate of 5Mibps. The first FIN sent is a separate packet (no piggybacked on an object or request).

Answer : transmitting time for SYN = L/R = 1024 bit /(5*2^20 bps) = 0.00019531s
propagation time = D/V = 75m / 2.5*10^8 = 0.0000003s
transmitting time for SYN+ACK =L/R=2048 bit /(5*2^20 bps) = 0.00039063s
transmitting time for response = L/R = (80*2^10 bit) / (5*2^20 bps)
= 0.015625s

time for one SYN, ACK, FIN = transmitting time + propagation time
= 0.00019531s + 0.0000003s = 0.00019561s
time for SYN+ACK = transmitting time + propagation time
= 0.00039063s + 0.0000003s = 0.00039093s
time for response = transmitting time + propagation time
= 0.015625s + 0.0000003s = 0.0156253s

total time for an object = (SYN) + (SYN + ACK) + (ACK) + (Request) + (Response) + (FIN) + (ACK + FIN) + (ACK)
= 5*SYN + 2*(SYN + ACK) + Response
= 5*0.00019561s + 2*0.00039093s + 0.0156253s
= 0.01738521s

total time for 14 object = 0.01738521s*14 = 0.24339294s

b) **[5 points]** Assume a single persistent HTTP connection is used for all 14 objects. No pipelining is used.

Answer:

total time = SYN + (SYN+ACK) + (ACK) + 14*(Request + Response) + FIN + (ACK + FIN) + ACK
= 4*SYN + 2*(SYN+ACK) + 14*(Request + Response)
= 4*0.00019561 + 2*0.00039093 + 14*(0.00019561+0.0156253)

= 0.22305704s