Course: Communication Networks

Date: 10/13

Topics Covered: HW #3

- MATTHEW KRUEGER -

1. This question concerns transport layer multiplexing.

   (a) A browser opens a secure TCP connection to an HTTPS server. Given that the browser socket's source-destination label is (206.164.10.30,9339; 137.158.64.227,443), what is the HTTP server socket's source-destination label?

   (b) How many sockets must a UDP server open to service four clients? Explain. Now answer the same question for a TCP server.

   (c) Can four clients, each from a **different host**, connect to the same server port from the same port number using the same protocol? What about four clients from the **same host** and port number? Explain.

a) (Source IP port; destination IP port) Flip!

i.e. (destination IP Port; Source IP port) → (137.158.64.227:443 ; 206.164.10.30:9339)

b)

	client	server	Total	explanation
UDP	0	1	1	UDP connectionless; only needs listener = 1
TCP	4	1	5	TCP connection-oriented; 4 clients + 1 server = 5

c)

Question	Ans	Explanation
i) 4 on different Hosts	YES	different port numbers can connect to same server port
ii) 4 on same Host	NO	no clients on same host cannot connect to same server port simultaneously; however, with multiplexing, YES

iii) results in identical 4-Tuples. Without employing source ports, clients on same host wouldn't be able to simultaneously use server.

Course: Communication Networks

Date: 10/13

Topics Covered: HW #3

- MATTHEW KRUEGER -

2. Review the socket application programming interface (API) material introduced in lecture (you may find the Linux man pages useful, too; search for a function call on this site).

   (a) List three socket calls that you would expect every TCP *server* to issue that would not normally be issued by any *client*. Explain briefly why TCP clients would not typically issue these calls *(hint: Lec. 8 notes/code examples may be your friend).*

   (b) Briefly explain what happens when a TCP socket listen () call is issued.

   (c) Briefly explain what happens when a TCP socket connect() call is issued.

a)

1. bind() → assigns local IP address & port # to a socket to listen for connections
2. listen() → marks socket (from bind()) as a socket that is ready to accept connections on bound port
3. accept() → accept incoming connection request from client.

b)

TCP listen() turns state of socket from "bound" → "listening"

c)

TCP connect() initiates client/server 3-way handshake

connect() (client)

1. SYN (client → server)
2. SYN-ACK (server → client)
3. ACK (client → server)

Course: Communication Networks

Date: 10/13

Topics Covered: HW #3

- MATTHEW KRUEGER -

3. As described in lecture, UDP and TCP segments are protected by 16-bit 1s complement checksums with overflow carries wrapped and added back to the lowest order bit.

   (a) Compute the 16-bit 1s complement Internet checksum of the following three 16-bit words: 1100010010011001, 0001000010111001, and 1100010001010001. Show your work.

   (b) The Internet checksum detects all 1-bit errors. Could it miss a 2-bit error? What about a 3-bit error? Justify your answers.

   (Hint: start with two smaller, e.g., 4-bit, words that each have one bit flipped between their 'sent' and 'received' versions. Can you come up with words and bit flips that result in the same checksum, but have different values for word 1 and word 2, on the sending and receiving side? Can you derive a scenario, e.g., two words, where the checksums are the same if there are now three bit flips between the two words?).

a)

```
  1 1 0 0 0 1 0 0 1 0 0 1 1 0 0 1
+ 0 0 0 1 0 0 0 0 1 0 1 1 1 0 0 1
-----------------------------------
  1 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0
```

```
  1 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0
+ 1 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0
-----------------------------------
  1 1 0 1 1 0 0 1 1 0 1 0 0 0 1 1
+                               1
-----------------------------------
  1 0 0 1 1 0 0 1 1 0 1 0 0 0 1 0 0
```

b) YES: both 2 & 3 bit errors can be undetected by checksum method due to wraparound & offsetting errors. A combination of these can lead to same checksum for differing inputs.

Course: Communication Networks

Date: 10/13

Topics Covered: HW #3

- MATTHEW KRUEGER -

4. Using a sketch like the ones in Fig. 3.16 (p212) of your textbook, devise a situation that shows how **out-of-order delivery** may occur and induce failure of the stop-and-wait protocol (rdt3.0). Is a protocol like stop-and-wait suitable for channels such as the Internet that can reorder messages? Setup/Hints:

   - Recall this protocol's 'sequence #s' toggle between state 0 (M0) and state 1 (M1).
   - Set up the problem to transmit a sequence of characters (a, b, c, d) where each message's payload contains the next character in the sequence. (e.g., M0[a], M1[b], M0[c], M1[d]).
   - Useful scenario to induce failure: consider what happens when (perhaps due to congestion) both an ACK is substantially delayed, and the packet resent due to this delayed ACK **is also substantially delayed**.
   - Specific example: consider a case where M0[a] gets sent and received. The receiver then sends ACK0 which is very delayed (perhaps due to congestion). M0[a], however, times out before ACK0 is received so M0[a] is resent (and will encounter its own long delay).
   - Following the resend of M0[a], let's say the first ACK0 now comes in to the sender following its delay. What does the sender do when ACK0 is received?
   - *Super hint*: Can you devise the situation such that the receiver (from its perspective) correctly receives a, b, a, d instead of a, b, c, d?

SEE NEXT PAGE FOR DIAGRAM

4)

Stop-and-wait is NOT SUITABLE for internet communications

- inability to handle reordering & errors in high latency environments

Course: Communicatior
Date:
Topics Covered:

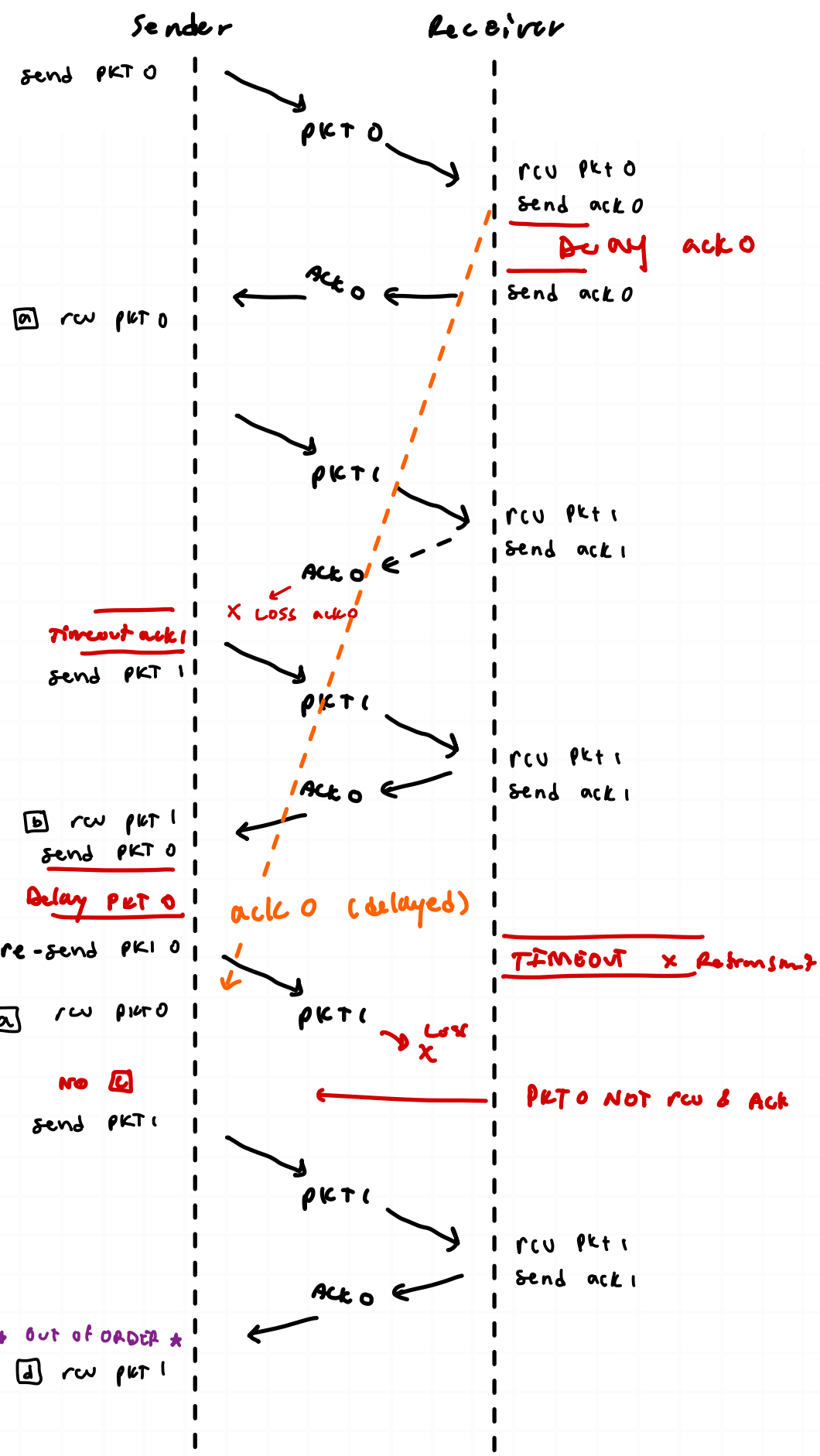
Sender window:

a b c d

N = 4

for question:

a b a d



Packets Received by sender: a b a d

Course: Communication Networks
Date: 10/13
Topics Covered: HW #3

- MATTHEW KRUEGER -

5. It's proposed that packets be sent over a link using a reliable data protocol that relies on negative, rather than positive, acknowledgements. Under this protocol: outgoing packets are assigned increasing sequence numbers by the sender; **upon receipt of an out-of-sequence packet**, the receiver rejects it and sends a negative acknowledgement of the expected sequence number; and upon receipt of a negative acknowledgement, the sender begins resending packets with increasing sequence numbers starting from the negatively acknowledged packet. *(Hint: recall there are no positive ACKs so the sender does not wait for an ACK or NACK. In other words, the sender keeps sending and corrects as needed when a NACK arrives).*

   (a) Using a sketch like in Fig. 3.22 on p220 of your textbook, show how the protocol recovers when a packet traveling from the sender to the receiver gets lost. *(Hint: how might the receiving side* ***detect*** *a packet loss?).*

   (b) Using a sketch like in Fig. 3.22 on p220 of your textbook, show how the protocol recovers when a negative acknowledgement traveling from the receiver to the sender gets lost.

   (c) A scenario exists where packets are lost and the protocol cannot recover. Sketch or describe this instance.

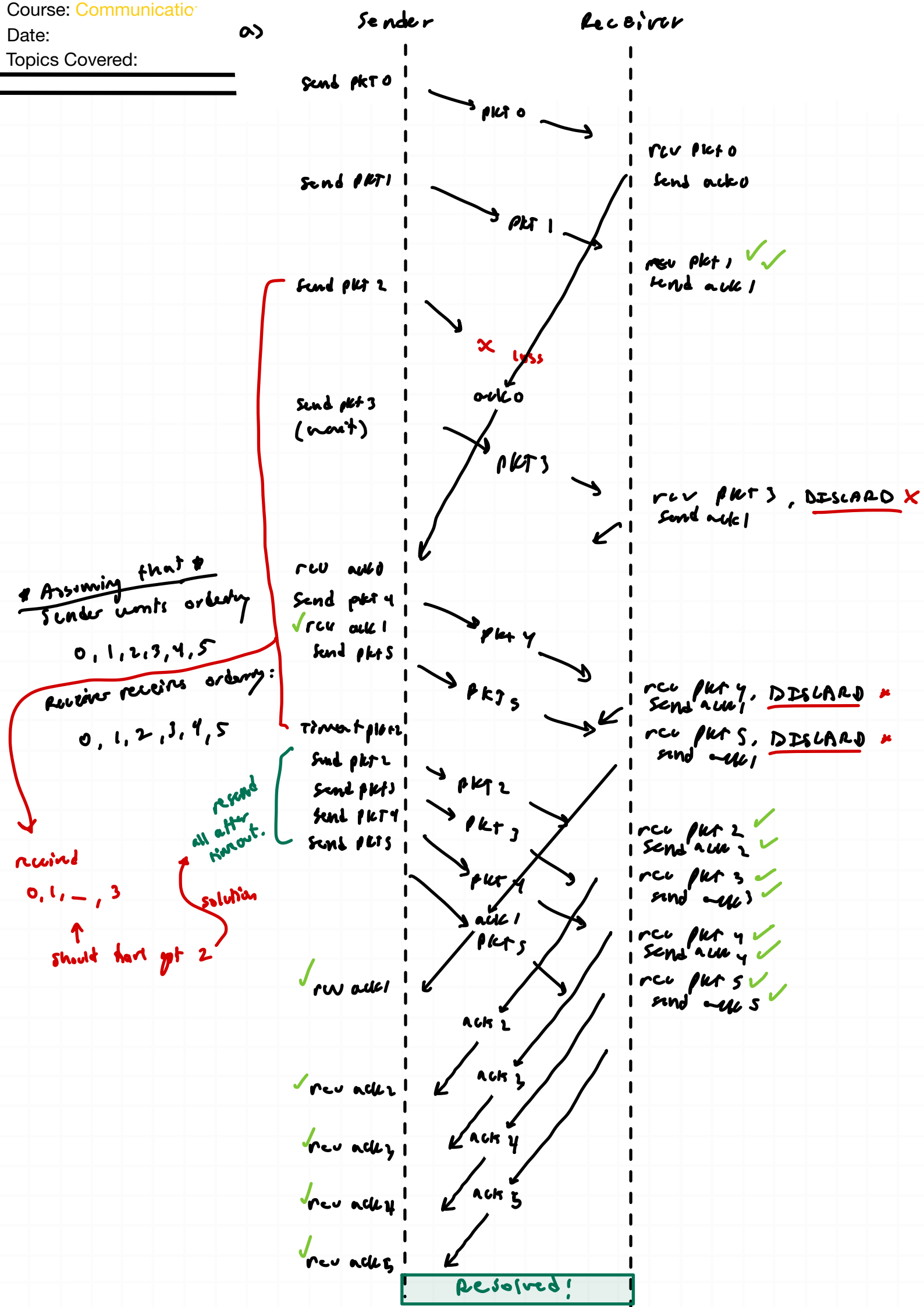
a) SEE NEXT PAGE

b) See two pages ahead

c) See three pages ahead

Course: Communicatio
Date:
Topics Covered:

a)



* Assuming that * sender wants ordering

0, 1, 2, 3, 4, 5

Receiver receives ordering:

0, 1, 2, 3, 4, 5

resend all after timeout.

received

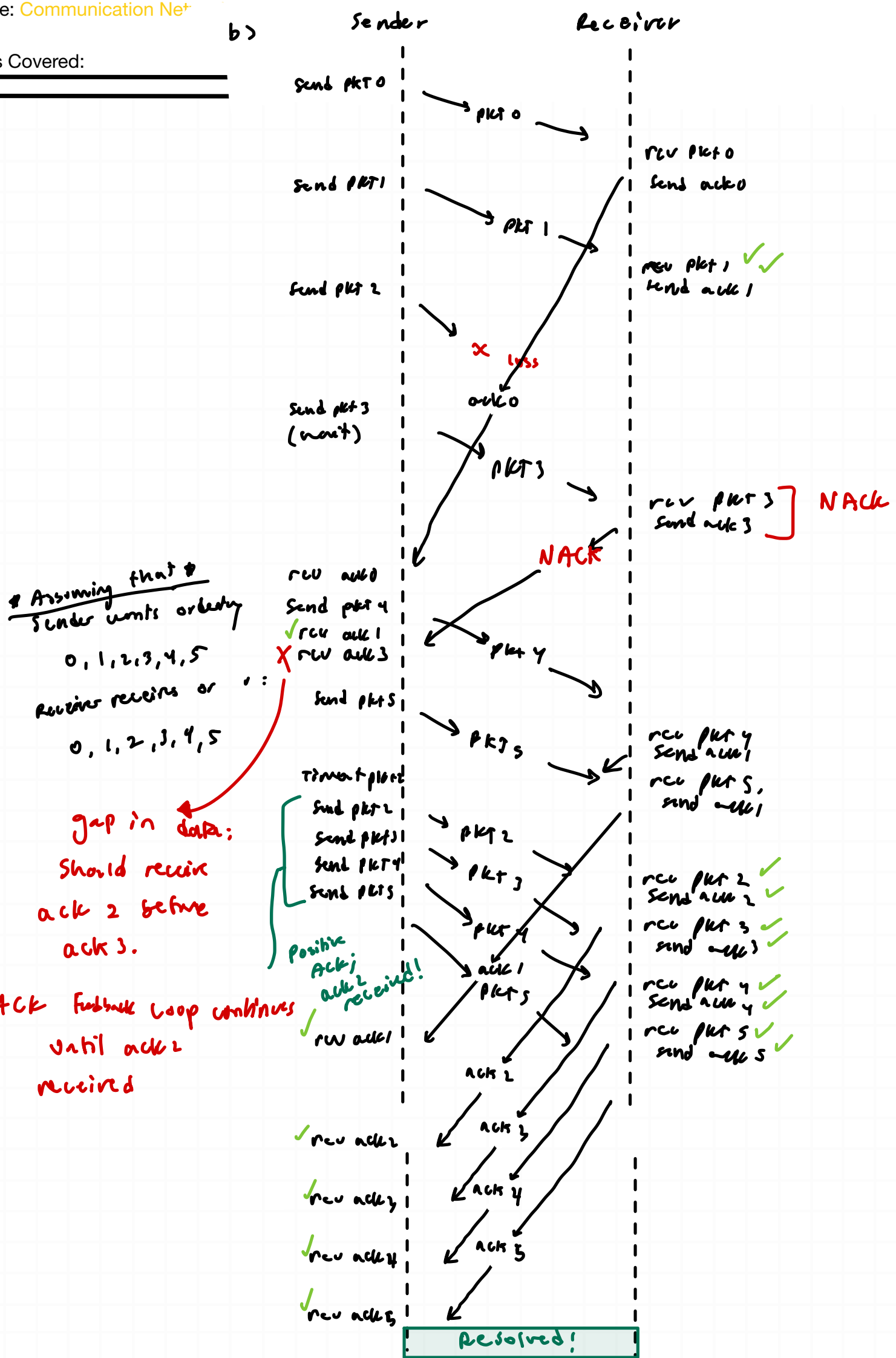
0, 1, _, 3

solution

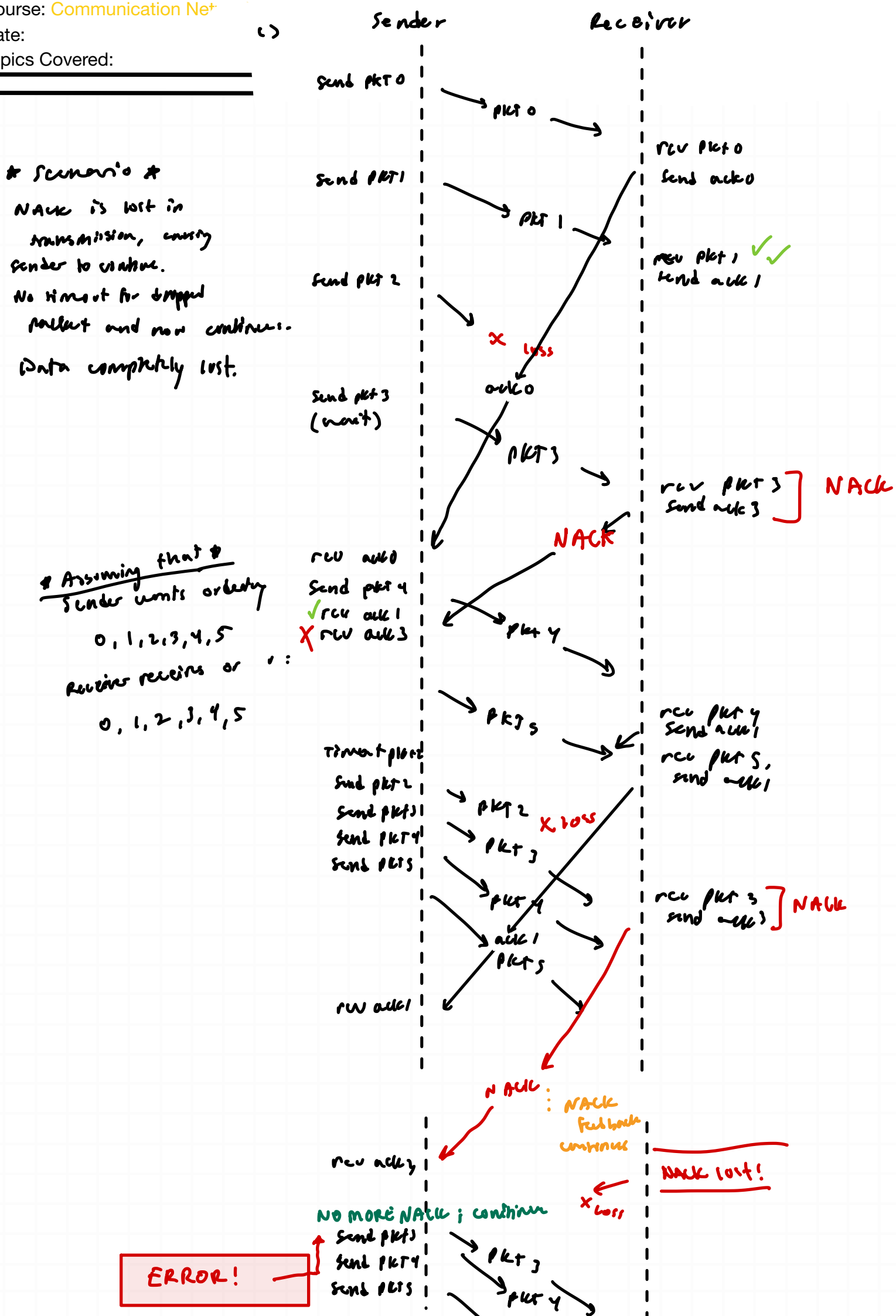
should have got 2

Course: Communication Net
Date:
Topics Covered:

b)



Course: Communication Net
Date:
Topics Covered:



Course: Communication Networks
Date: 10/13
Topics Covered: HW #3

- MATTHEW KRUEGER -

6. A sliding window protocol is to be used to provide reliable data transfer over a 1.1 km link with a propagation speed of 2×10^5 km/s. The link rate is 1 Gbps. The link packet size, including the header, is 100 bytes. The header has a 4-bit sequence number field.

   (a) Determine the maximum transmit and receive window sizes, W_t and W_r, allowable for proper operation of the stop-and-wait, go-back-N, and selective repeat protocols. *(Hint: Slide "Sequence #s and SR window size" of Lec. 12 will help).*

   (b) Which of the above protocols, and what window sizes, W_t and W_r, would be your best choices if your primary goal is to first maximize link utilization and then, as possible, minimize the average data transfer delay? What link utilization do your choices achieve? *(Hint: See Lec. 12 slides "Pipelining: increasing utilization" and "Sequence #s and SR window size." Recall for max utilization that "filling the pipe" $\Rightarrow U_{sender} \approx 1$, assuming small error probability).*

   (c) Rework (b) when the packet size, including the header, is 200 bytes instead of 100 bytes.

D: 1.1 km → 1100 m V: $2 \cdot 10^5$ km → $2 \cdot 10^8$ m R: 1 Gbps → $1 \cdot 10^9$ bits/s

L: 100 bytes → 800 bits

num sequence bits: 4
↳ $2^m \rightarrow 2^4 = 16$

a)

T_p:

$T_p = \frac{D}{V} = \frac{1100}{2 \cdot 10^8} = 5.5\ \mu s$

RTT:

$RTT = 2 \cdot T_p = 11\ \mu s$

Protocols

Stop-and-wait:
$W_t = 1$
$W_r = 1$

Go-Back-N:
$W_t = 15$
$W_r = 1$

SR:
$W_t = 8$
$W_r = 8$

$W_t \leq \frac{2^m}{2} = \frac{2^4}{2} = 8$

b)

$W_{t\,best} = \min\left(\frac{R \cdot RTT}{L} + 1,\ 2^m - 1\right)$

$= \left(\frac{(1 \cdot 10^9 \cdot 11 \cdot 10^{-6})}{800} + 1,\ 15\right)$

$= \left(\frac{11 \cdot 10^3}{800} + 1 \approx 14,\ 15\right)$

$= \min(14, 15)$

$= 14$

$T_t = \frac{L}{R} = \frac{800}{1 \cdot 10^9} = 0.8\ \mu s$

SR

$U = \frac{W_t T_t}{RTT} = \frac{8 \cdot 0.8\ \mu s}{11\ \mu s} = 0.5818$

Stop & wait

$U = \frac{T_t}{T_t + RTT} = \frac{0.8\ \mu s}{0.8\ \mu s + 11\ \mu s} = 0.0678$

Go-Back-N

$U = \frac{W_t T_t}{RTT}$

$= \frac{14 \times 0.8\ \mu s}{11\ \mu s} = 1.018 \rightarrow 1$ (capped)

Protocol U:

Best Go-Back-N: 1 FULL
↓ SR: 0.5818
Worst Stop-and-wait: 0.0678

Course: Communication Networks
Date: 10/13
Topics Covered: HW #3

- MATTHEW KRUEGER -

c) Now $L = 2000$ bytes $\rightarrow$ 16000 bits

$$T_t = \frac{L}{R} = \frac{16000}{1 \cdot 10^9} = 1.6\,\mu s$$

$$w_{t\,best} = \min\left(\frac{R \cdot RTT}{L} + 1,\ 2^{n} - 1\right)$$

$$= \left(\frac{(1 \cdot 10^9 \cdot 11 \cdot 10^{-6})}{2000} + 1,\ 15\right)$$

$$= \left(\frac{11 \cdot 10^3}{2000} + 1 = 56,\ 15\right)$$

$$= \min(56, 15) = 15$$

Stop & wait

$$U = \frac{T_t}{T_t + RTT} = \frac{1.6\,\mu s}{1.6\,\mu s + 11\,\mu s} = 0.127$$

SR

$$U = \frac{w_t T_t}{RTT} = \frac{8 \cdot 1.6}{11\,\mu s}\,\mu s = 1.16 \rightarrow 1 \text{ (capped)}$$

Go-Back-N

$$U = \frac{w_t T_t}{RTT} = \frac{8 \times 1.6}{11\,\mu s}\,\mu s = 0.145 \rightarrow 1 \text{ (capped)}$$

Protocol U:

Best Go-Back-N : 1
↓ SR : 1
Worst Stop-&-wait : 0.127