Solutions for Homework #4

EE122: Introduction to Communication Networks

(Fall 2006)

Department of Electrical Engineering and Computer Sciences
College of Engineering
University of California, Berkeley

Vern Paxson / Sukun Kim / Dilip Antony Joseph

1. (10 points)

   (a) (10 points)

   Let F_i be the finishing time.

Packet	Size	Flow	F_i
1	200	1	200
2	200	1	400
3	160	1	560
4	80	2	80
5	240	2	320
6	90	3	90
7	180	3	270
8	180	3	450

   Packets are sent in the increasing order of F_i: packet 4, 6, 1, 7, 5, 2, 8, 3.

   (b) (optional)

   The ratio of share of flows 1, 2, and 3 is 2:4:3. Let W_i be this ratio ($W_1 = 2, W_2 = 4, W_3 = 3$). To weight these, let $(\textit{Weighted } F_i) = F_i/W_i$. Then, packets are sent in the increasing order of *Weighted* F_i.

Packet	Size	Flow	*Weighted* F_i
1	200	1	100
2	200	1	200
3	160	1	280
4	80	2	20
5	240	2	80
6	90	3	30
7	180	3	90
8	180	3	150

Packets are sent in the following order: packet 4, 6, 5, 7, 1, 8, 2, 3.

2. (15 points)

   (a) (5 points)
   To fully utilize the network, *AdvertisedWindow* needs to be larger than $(\mathit{Delay} \times \mathit{Bandwidth})$.

$$\begin{aligned}(\mathit{AdvertisedWindow}) &\geq (\mathit{Delay}) \times (\mathit{Bandwidth}) \\ &= 275\mathit{ms} \times 1\mathit{Gbps} \\ &= 275\mathit{Mbit} \\ &= 34.375\mathit{MB}\end{aligned}$$

$$\begin{aligned}2^{25} &= 33,554,432 \\ 2^{26} &= 67,108,864\end{aligned}$$

   Therefore, 26 bits are needed for $AdvertisedWindow$.

   SequenceNum needs be large enough that the sequence number does not wrap around before any delayed segments have left the network, which is presumed to occur within the maximum segment lifetime.

$$\begin{aligned}(\mathit{SequenceNum}) &\geq (\mathit{Maximum\ Segment\ Lifetime}) \times (\mathit{Bandwidth}) \\ &= 30\mathit{s} \times 1\mathit{Gbps} \\ &= 30\mathit{Gbit} = 3.75\mathit{GB}\end{aligned}$$

$$\begin{aligned}2^{31} &= 2,147,483,648 \\ 2^{32} &= 4,294,967,296\end{aligned}$$

   Therefore, 32 bits are needed for *SequenceNum*.

   (b) (5 points)
   If *AdvertisedWindow* is 16 bits, it is smaller than $(\mathit{Delay} \times \mathit{Bandwidth})$ product. Therefore, *AdvertisedWindow* is the limiting factor of the effective bandwidth. During a single *RTT*, only *AdvertisedWindow* amount of data can be transferred.

$$\begin{aligned}(\mathit{Effective\ Bandwidth}) &= \frac{(\mathit{AdvertisedWindow})}{\mathit{RTT}} \\ &= \frac{2^{16}\mathit{B}}{275\mathit{ms}} \\ &= 238.31\mathit{KB/s}\end{aligned}$$

(c) (5 points)

The average long-term performance achieved by a TCP sender is governed by the *TCP Throughput Equation*:

$$\begin{aligned}(\textit{Throughput}) &= \frac{\sqrt{1.5}B}{\textit{RTT}\sqrt{p}} \\ &= \frac{\sqrt{1.5}B}{275\textit{ms}\sqrt{0.005}} \\ &= (62.98 \times B) \text{ bytes/s}\end{aligned}$$

Assuming $B = 1500$ bytes:

$$\begin{aligned}(\textit{Throughput}) &= (62.98 \times B) \text{ bytes/s} \\ &= \text{94,475 bytes/s}\end{aligned}$$

3. (10 points)

In Jacobson/Karels algorithm, *TimeOut* value is calculated as follows.

$$\begin{aligned}\textit{Difference} &= \textit{SampleRTT} - \textit{EstimatedRTT} \\ \textit{EstimatedRTT} &= \alpha \times \textit{EstimatedRTT} + (1-\alpha) \times \textit{SampleRTT} \\ \textit{Deviation} &= \textit{Deviation} + \delta(|\textit{Difference}| - \textit{Deviation}) \\ \textit{TimeOut} &= \mu \times \textit{EstimatedRTT} + \phi \times \textit{Deviation}\end{aligned}$$

The parameters are set to $\alpha = 7/8$, $\delta = 1/8$, $\mu = 1$, and $\phi = 4$. (These are the standard ones, except the usual setting for δ is $1/4$, but the problem states to instead use $1/8$.) When we start, *EstimatedRTT* $= 4.0$ and *Deviation* $= 0.75$. From that point on, all *SampleRTT* are 1.0.

The following table shows the calculated values of *Difference*, *EstimatedRTT*, *Deviation*, and *TimeOut* over repeated measurements:

Iteration	*SampleRTT*	*EstimatedRTT*	*Deviation*	*Difference*	*TimeOut*
0	1.000	4.000	0.750		
1	1.000	3.625	1.031	-3.000	7.750
2	1.000	3.297	1.230	-2.625	8.219
3	1.000	3.010	1.364	-2.297	8.465
4	1.000	2.759	1.445	-2.010	8.537
5	1.000	2.539	1.484	-1.759	8.474
6	1.000	2.346	1.491	-1.539	8.309
7	1.000	2.178	1.473	-1.346	8.069
8	1.000	2.031	1.436	-1.178	7.774
9	1.000	1.902	1.385	-1.031	7.443
10	1.000	1.789	1.325	-0.902	7.088
11	1.000	1.691	1.258	-0.789	6.722
12	1.000	1.604	1.187	-0.691	6.352
13	1.000	1.529	1.114	-0.604	5.985
14	1.000	1.463	1.041	-0.529	5.626
15	1.000	1.405	0.969	-0.463	5.279
16	1.000	1.354	0.898	-0.405	4.947
17	1.000	1.310	0.830	-0.354	4.631
18	1.000	1.271	0.765	-0.310	4.332
19	1.000	1.237	0.703	-0.271	4.051
20	1.000	1.208	0.645	-0.237	3.788

It takes 20 iterations before *TimeOut* falls below 4.0.

4. (15 points)

We divide 60KB of data into 40 packets of 1500 byte each. Initially, CWND is set to 1 MSS, which is 1500 bytes. To calculate the average data throughput, we need to determine how much time elapses until we have sent the last of the packets. (The problem does not require us to consider when the last packet ultimately arrives at the receiver.) Note that since there is sufficient buffering at the receiver, window size is solely limited by CWND.

(a) (5 points)

With slow-start, CWND doubles every RTT:

Time (s)	Window Size (B)
t_1	1,500
$t_1 + 0.1$	3,000
$t_1 + 0.2$	6,000
$t_1 + 0.3$	12,000
$t_1 + 0.4$	24,000
$t_1 + 0.5$	48,000

At $t_2 = t_1 + 0.5$, we have sent

$$1{,}500 + 3{,}000 + 6{,}000 + 12{,}000 + 24{,}000 = 46{,}500 \text{ bytes}$$

leaving us 13,500 more bytes to send (9 packets). For a 100 Mbps link, these take 13.5 KB/s / 12.5 MBps = 1.08 ms to transmit (leave the sender). Since we're computing approximate throughput, we can ignore this additional amount, as it is much less than RTT.

Thus, about 500 ms elapses between when we start sending and when we finish, and therefore:

$$\begin{aligned} \text{Throughput} &\approx \frac{60 \text{ KB}}{500 \text{ ms}} \\ &\approx 120 \text{ KB/s} \end{aligned}$$

(b) (5 points)

In pure AIMD, CWND increases by 1 MSS every RTT.

Time (s)	Window Size (B)
t_1	1,500
$t_1 + 0.1$	3,000
$t_1 + 0.2$	4,500
$t_1 + 0.3$	6,000
$t_1 + 0.4$	7,500
$t_1 + 0.5$	9,000
$t_1 + 0.6$	10,500
$t_1 + 0.7$	12,000
$t_1 + 0.8$	13,500

At $t_2 = t_1 + 0.8$, we have sent 36 packets and 4 more remain to be sent. Again, the time these 4 take is negligible, so we have:

$$\begin{aligned} \text{Throughput} &\approx \frac{60 \text{ KB}}{800 \text{ ms}} \\ &\approx 75 \text{ KB/s} \end{aligned}$$

(c) (5 points)

In this scheme, CWND increases by MSS $\times$ (MSS / CWND) for every ACK:

ACK number	Window Size (B)	ACK number	Window Size (B)
0	1,500	20	9,865
1	3,000	21	10,093
2	3,750	22	10,316
3	4,350	23	10,534
4	4,867	24	10,748
5	5,330	25	10,957
6	5,752	26	11,163
7	6,143	27	11,364
8	6,509	28	11,562
9	6,855	29	11,757
10	7,183	30	11,948
11	7,496	31	12,136
12	7,796	32	12,322
13	8,085	33	12,504
14	8,363	34	12,684
15	8,632	35	12,862
16	8,893	36	13,037
17	9,146	37	13,209
18	9,392	38	13,380
19	9,632	39	13,548
		40	13,714

(your values may differ slightly depending on how you round)

We can then use the above table to see how many packets will be in flight during each round-trip:

Time (s)	Window Size (B)	Packets Sent (= Acks Received)	Starting Ack
t_1	1500	1	0
$t_1 + 0.1$	3,000	2	1
$t_1 + 0.2$	4,350	2	3
$t_1 + 0.3$	5,330	3	5
$t_1 + 0.4$	6,509	4	8
$t_1 + 0.5$	7,796	5	12
$t_1 + 0.6$	9,146	6	17
$t_1 + 0.7$	10,534	7	23
$t_1 + 0.8$	11,948	7	30
$t_1 + 0.9$	13,209	(remainder = 3)	37

Thus, at time $t_2 = t_1 + 0.9$, we have 3 more packets to send. Ignoring the time to send these gives us:

$$\begin{aligned}\text{Throughput} &\approx \frac{\text{60 KB}}{\text{900 ms}}\\ &\approx 66.7\text{ KB/s}\end{aligned}$$

5. (10 points)

   Let λ be the mean arrival rate of people, and let d be the mean time people spend in the office. Let N be the mean number of people in the office. Little's Law states

   $$N = \lambda d$$

   We have $\lambda = 3$ people/hour. We can compute:

   $$\begin{aligned} d &= \frac{\text{20 minutes} + \text{35 minutes} + \text{20 minutes}}{\text{3 people}}\\ &= \frac{\text{75 minutes}}{\text{3 people}}\\ &= \text{25 minutes/person}\end{aligned}$$

   Applying Little's Law:

   $$\begin{aligned} N &= \lambda d\\ &= \text{3 people/hour} \cdot \text{25 minutes/person}\\ &= 3 \times \frac{25}{60}\\ &= 1.25\end{aligned}$$

   Thus, the mean number of people in Dilip's office is 1.25.

6. (16 points)

   (a) (4 points)

   We find in */etc/services* that the port associated with TLS/SSL is 443. We run openssl to port 443 of *www.amazon.com* as follows:

   ```
   openssl s_client -connect www.amazon.com:443
   ```

   ```
   ee122-tb@c199: /download$ openssl s_client -connect www.amazon.com:443
   CONNECTED(00000004)
   depth=0 /C=US/ST=Washington/L=Seattle/O=Amazon.com Inc./CN=www.amazon.com
   verify error:num=20:unable to get local issuer certificate
   ```

```
verify return:1
depth=0 /C=US/ST=Washington/L=Seattle/O=Amazon.com Inc./CN=www.amazon.com
verify error:num=27:certificate not trusted
verify return:1
depth=0 /C=US/ST=Washington/L=Seattle/O=Amazon.com Inc./CN=www.amazon.com
verify error:num=21:unable to verify the first certificate
verify return:1
—
Certificate chain
0 s:/C=US/ST=Washington/L=Seattle/O=Amazon.com Inc./CN=www.amazon.com
i:/C=US/O=RSA Data Security, Inc./OU=Secure Server Certification Authority
—
Server certificate
——BEGIN CERTIFICATE——
MIIDrTCCAxqgAwIBAgIQXLQs7kNSZIYaovXXArxaATANBgkqhkiG9w0BAQUFADBf
MQswCQYDVQQGEwJVUzEgMB4GA1UEChMXUlNBIERhdGEgU2VjdXJpdHksIEluYy4x
LjAsBgNVBAsTJVNlY3VyZSBTZXJ2ZXIgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkw
HhcNMDUxMjIzMDAwMDAwWhcNMDYxMjIzMjM1OTU5WjBnMQswCQYDVQQGEwJVUzET
MBEGA1UECBMKV2FzaGluZ3RvbjEQMA4GA1UEBxQHU2VhdHRsZTEYMBYGA1UEChQP
QW1hem9uLmNvbSBJbmMuMRcwFQYDVQQDFA53d3cuYW1hem9uLmNvbTCBnzANBgkq
hkiG9w0BAQEFAAOBjQAwgYkCgYEAyFlt3lt72exxzm9ZHEnLFRCceYqCv8fRfhMp
4dlgJoSLyRW3wYrrYzbqdhh1OF6shlP239hPNb9Rfaqt2md3583TupEWNn6pWL2i
nhuAmhn0P7gV0zqX02pRA/4WZh5133jkmOVq9cCkJz+7yJunqxv844JLEmBrrSA3
jbXzy4kCAwEAAaOCAWQwggFgMAkGA1UdEwQCMAAwCwYDVR0PBAQDAgWgMDwGA1Ud
HwQ1MDMwMaAvoC2GK2h0dHA6Ly9jcmwudmVyaXNpZ24uY29tL1JTQVNlY3VyZVNl
cnZlci5jcmwwRAYDVR0gBD0wOzA5BgtghkgBhvhFAQcXAzAqMCgGCCsGAQUFBwIB
FhxodHRwczovL3d3dy52ZXJpc2lnbi5jb20vcnBhMB0GA1UdJQQWMBQGCCsGAQUF
BwMBBggrBgEFBQcDAjA0BggrBgEFBQcBAQQoMCYwJAYIKwYBBQUHMAGGGGh0dHA6
Ly9vY3NwLnZlcmlzaWduLmNvbTBtBggrBgEFBQcBDARhMF+hXaBbMFkwVzBVFglp
bWFnZS9naWYwITAfMAcGBSsOAwIaBBSP5dMahqyNjmvDz4Bq1EgYLHsZLjAlFiNo
dHRwOi8vbG9nby52ZXJpc2lnbi5jb20vdnNsb2dvLmdpZjANBgkqhkiG9w0BAQUF
AAN+AGlyQMXafinsEd7XYJcDB01Livgjwwgr/Roomib2LuWJIG3PjNxqslpvqpUY
GMPGWDJDKor1PvV5okbdx8F7dCHST0/XenAVG0z2AZYQc1Krlm+BwlRF3AS+UtUD
XxH8Ga2am3X6xEMZv1vpOgQzh/I8GxETS8ez1qphxcsD
——END CERTIFICATE——
subject=/C=US/ST=Washington/L=Seattle/O=Amazon.com Inc./CN=www.amazon.com
issuer=/C=US/O=RSA Data Security, Inc./OU=Secure Server Certification Authority
—
No client certificate CA names sent
—
SSL handshake has read 1091 bytes and written 330 bytes
—
```

```
New, TLSv1/SSLv3, Cipher is RC4-MD5
Server public key is 1024 bit
Compression: NONE
Expansion: NONE
SSL-Session:
Protocol : TLSv1
Cipher : RC4-MD5
Session-ID: 23D22A076344B8730B2C15D5D8496B4D8C25556DA7B7A782C2D9111BF6867E06
Session-ID-ctx:
Master-Key: B504465103C346DB06D1C3040AAEE533E67C1F885F019FA4674AD2BA7C5C0A76D2F514EE41
Key-Arg : None
Start Time: 1164760603
Timeout : 300 (sec)
Verify return code: 21 (unable to verify the first certificate)
—
```

From this we read out:

Country : US

State : Washington

City : Seattle

Organization : Amazon.com Inc.

(b) (4 points)

RSA Data Security, Inc.

(c) (4 points)

Country : US

State : California

City : Berkeley

Organization : UC Berkeley

RSA Data Security, Inc. provides ”Secure Server Certification Authority”.

(d) (4 points)

Country : US

State : California

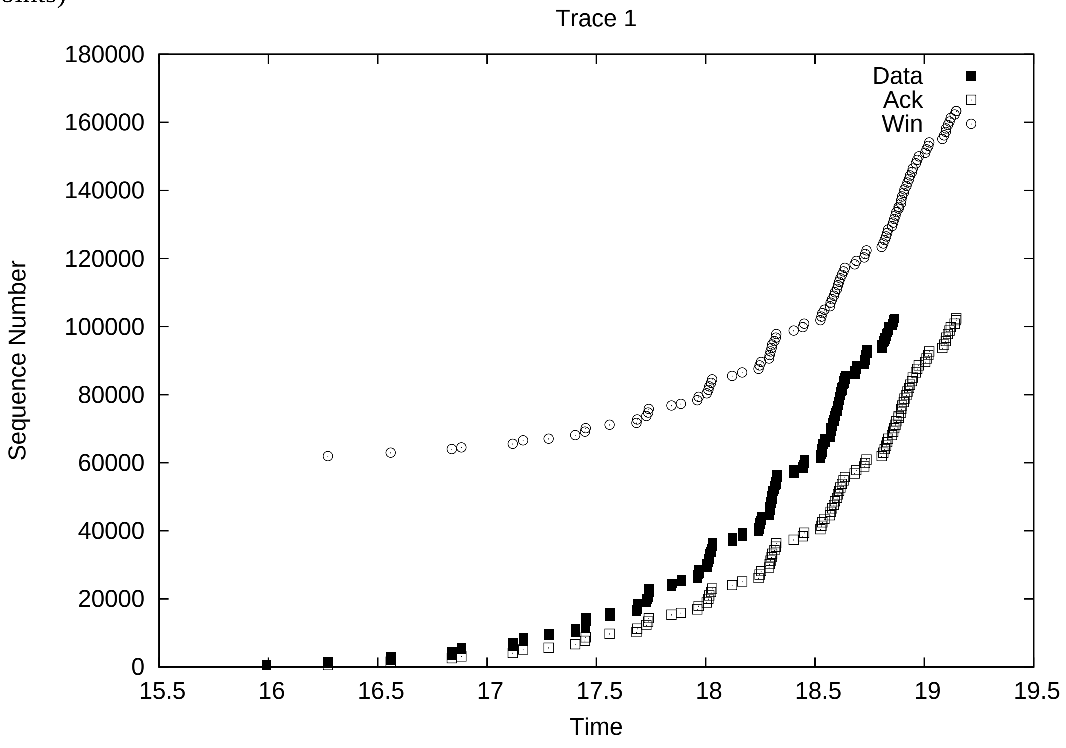
City : Berkeley

Organization : Lawrence Berkeley Laboratory

Entrust.net provides ”Secure Server Certification Authority”.

7. (24 points)

(a) (8 points)



i. Approximate RTT: 280 ms

ii. Overall throughput: 36 KB/s

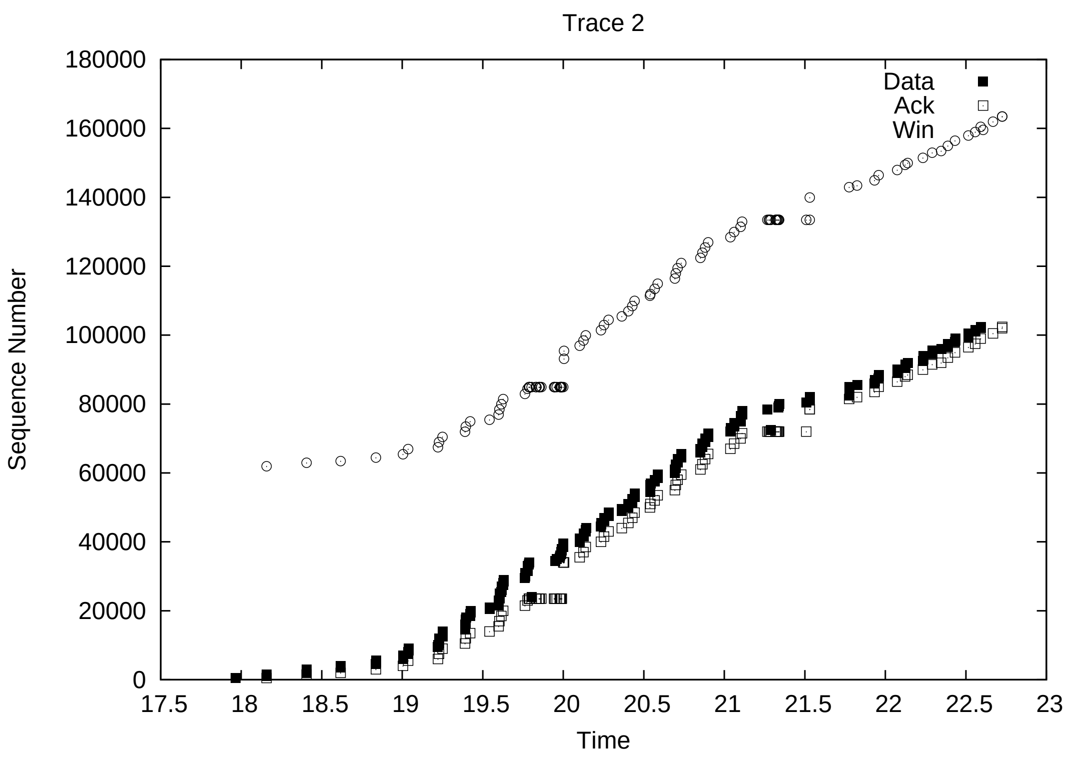
iii. Largest effective window size: 33 KB. Note, this refers to the maximum amount of data in flight (which here occurs near the end of the trace), *not* the advertised window (unless it happens to limit the maximum amount of data in flight—not the case for any of these traces).

iv. Maximum throughput: ≈ 90 KB/s (last two flights). Note, there are a lot of ways to measure throughput (data in flight, data until acknowledged, throughput as echoed in acknowledgments due to "ack clocking"), so we allowed any answer here within a fact of two of our best answer.

v. Advertised window has never been a limit.

vi. There is no retransmission.

(b) (8 points)



i. Approximate RTT: 190 ms initially; this drops to around 150 ms later in the trace.

ii. Overall throughput: $\approx$ 22 KB/s

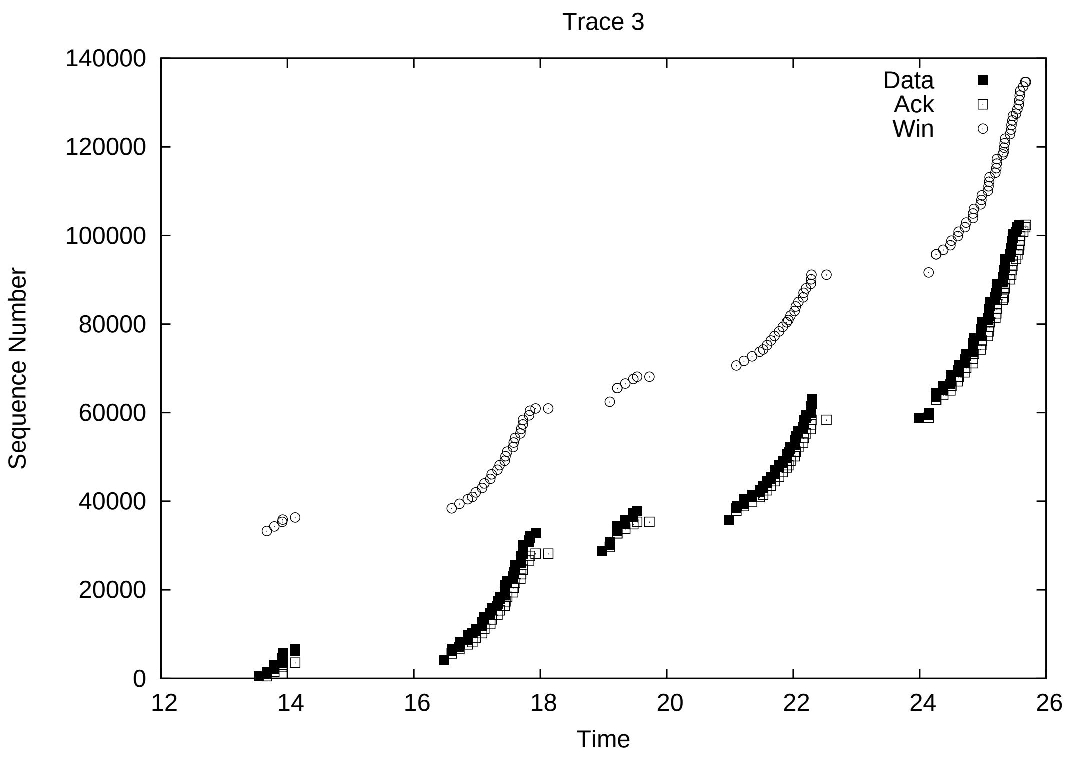
iii. Largest effective window size: a bit above 10 KB, just prior to the first lost.

iv. Maximum throughput: 40–60 KB/s

v. Advertised window has never been a limit.

vi. 24001, 72501 are retransmitted due to Fast Retransmission. We can tell that Fast Recovery is also used because as more duplicate ACKs arrive, additional data is sent.

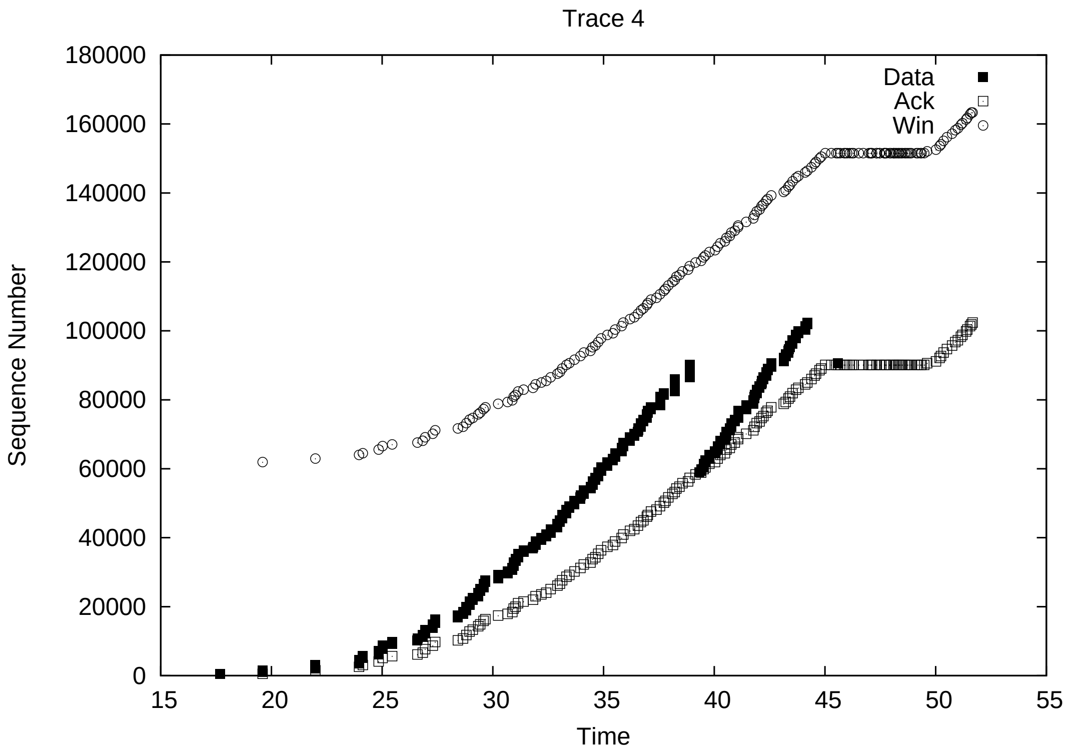
(c) (8 points)



i. Approximate RTT: 115 ms
ii. Overall throughput: 8.3 KB/s
iii. Largest effective window size: about 6 KB (near the end of the trace)
iv. Maximum throughput: 40–80 KB/s
v. Advertised window has never been a limit.
vi. 4097, 28673, 35841, 58881 are retransmitted due to timeout.

8. (+10 points)

(a) (+5 points)

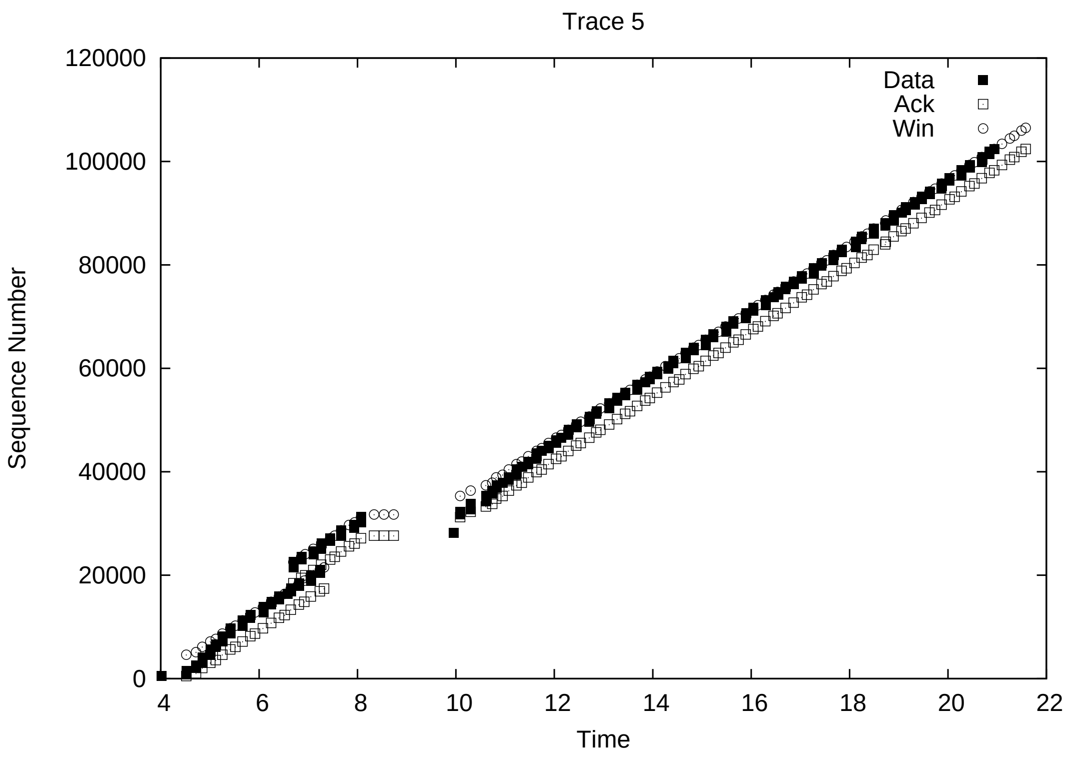


Error in the TCP sender's algorithms. The sender fails to adjust its timeout value sufficiently, leading to an unnecessary timeout retransmission of segment 58881. Note how soon after the retransmission that the ACK for that segment arrives—it's for the original transmission of the segment, rather than for the retransmission. Also note the immense RTT at this point: well over 4 seconds.

The steady increase in RTT as the connection proceeds indicates that the connection is *filling a queue*, such that its packets increasingly must wait behind their predecessors.

This is from a trace of an actual connection.

(b) (+5 points)



Measurement error. The packets starting with 21505 are shifted to the left due to *the clock used to record their timestamps jumping backward in time* by about 500 msec.

How does a clock jump backward? Because it is being set in an attempt to keep it accurate, in this case using the Network Time Protocol (NTP).

This again is from a trace of an actual connection.