This is the best assignment solution uploaded with some modification of the following student:

Name: **Samreen Laghari**
Student Id: **ms110400048**

Question No. 1:

a) Suppose that host A is connected to a router R1, R1 is connected to another router, R2, and R2 is connected to host B. Suppose that a TCP message that contains 900 bytes of data and 20 bytes of TCP header is passed to the IP code at host A for delivery to B. Show the Total length, Identification, DF (Don't Fragment), MF (More Fragment), and Fragment offset fields of the IP header in each packet transmitted over the three links. Assume that link A-R1 can support a maximum frame size of 1024 bytes including a 14-byte frame header; link R1-R2 can support a maximum frame size of 512 bytes, including an 8-byte frame header, and link R2-B can support a maximum frame size of 512 bytes including a 12-byte frame header.

Solution

The initial IP datagram will be fragmented into two IP datagrams at I1. No other fragmentation will occur.

Link A-R1
Total length = 940
Identification = ID = x
Don't Fragment = DF = 0
More Fragment = MF = 0
Fragment Offset = 0

Link R1-R2

1) Total length = 500
   Identification = ID = x
   Don't Fragment = DF = 0
   More Fragment = MF = 1
   Fragment Offset = 0
2) Total length = 460
   Identification = ID = x
   Don't Fragment = DF = 0
   More Fragment = MF = 0
   Fragment Offset = 60

Link R2-B

1) Total length = 500

Identification = ID = x
Don't Fragment = DF = 0
More Fragment = MF = 1
Fragment Offset = 0

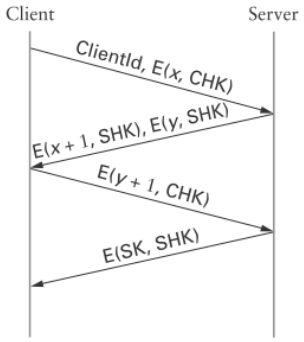
2) Total length = 460
Identification = ID = x
Don't Fragment = DF = 0
More Fragment = MF = 0
Fragment Offset = 60

b) Video applications usually run over UDP rather than TCP as they cannot tolerate retransmission delays. Nevertheless, this means video applications are not constrained by TCP's congestion-control algorithm. What impact does this have on TCP traffic? Be specific about the consequences. Fortunately, these video applications often use RTP, which results in RTCP "receiver reports" being sent from the sink back to the source. These reports are sent periodically (e.g., once a second) and include the percentage of packets successfully received in the last reporting period. Describe how the source might use this information to adjust its rate in a TCP-compatible way.

Solution

If the UDP Video streams go beyond the path capacity due to enough traffic, the competing TCP streams, if any, will back off and effectively killed. On the other hand, the UDP streams will get packets through since they tend to ignore congestion. We can say that UDP streams will get large percentage of bandwidth as they never get killed or back off due to congestion. The use of RTCP "receiver reports" to throttle the send rate in the way similar to TCP i.e. the exponential backoff and linear increase helps in making the video traffic much approachable and greatly helps in competing TCP performance.

c) In the three-way authentication handshake as shown in the following figure, why is the server unsure of the client's identity until it receives the third message? To what attack might a server be exposed if it trusted the client's identity before the third message was received?



Solution

An eavesdropper may have intercepted the messages exchanged in a previous transaction, and may be replaying the client's earlier responses. Until the real client has successfully decrypted the server's new challenge and sent back evidence of this, the server cannot be sure such a replay attack isn't in progress.

d) As a possible congestion control mechanism in a subnet using virtual circuits internally, a router could refrain from acknowledging a received packet until (a) it knows its last transmission along the virtual circuit was received successfully and (b) it has a free buffer. For simplicity, assume that the routers use a stop-and-wait protocol and that each virtual circuit has one buffer dedicated to it for each direction of traffic. If it takes T sec to transmit a packet (data or acknowledgement) and there are n routers on the path, what is the rate at which packets are delivered to the destination host? Assume that transmission errors are rare and that the host-router connection is infinitely fast.

Solution

The time can be slotted in units of T seconds; the situation of the three slots can be given as under:

- The source router sends the first packet in slot 1.
- At start of slot 2, second router has received the packet but cannot acknowledge it yet
- At start of slot 3, the third router has received the packet, but it also cannot acknowledge it, therefore all the routers behind this are still suspended.

When the destination host takes the packet from the destination router, only then the first acknowledgement can be sent.

Then the acknowledgement propagates back and takes full transits of the subnet, 2(n – 1) T sec, before the source router sends the second packet. Therefore the throughput is one packet every 2(n – 1) T sec i.e. $Throughput = \frac{1}{2(n-1)\ T\ \text{sec}}$.

Question No. 2:

a) The RPC-based "NFS" remote file system is sometimes considered to have slower than expected write performance. In NFS, a server's RPC reply to a client write request means that the data is physically written to the server's disk, not just placed in a queue.

(a) Explain the bottleneck we might expect, even with infinite bandwidth, if the client sends all its write requests through a single logical CHAN channel, and explain why using a pool of channels could help. Hint: You will need to know a little about disk controllers.

(b) Suppose the server's reply means only that the data has been placed in the disk queue. Explain how this could lead to data loss that wouldn't occur with a local disk. Note that a system crash immediately after data was enqueued doesn't count because that would cause data loss on a local disk as well.

(c) An alternative would be for the server to respond immediately to acknowledge the write request, and to send its own separate CHAN request later to confirm the physical write. Propose different CHAN RPC semantics to achieve the same effect, but with a single logical request/reply.

Solution

(a) Explain the bottleneck we might expect, even with infinite bandwidth, if the client sends all its write requests through a single logical CHAN channel, and explain why using a pool of channels could help. Hint: You will need to know a little about disk controllers.

The bottleneck we might expect even with infinite bandwidth is the servicing of reads which is not accomplished in FIFO order. Instead the disk controllers schedule writes by the use of "elevator" or "SCAN" algorithm. The "elevator" or SCAN algorithm sorts the pool of currently outstanding writes by disk track number. After this the writes are executed in increasing order of track numbers.

Writes would be forced to execute in sequence using single CHAN channel even in the situation when such sequence requires numerous unnecessary disk head motion. By using a pool of N CHAN-like channels, disk controller will have almost N writes to schedule at any time in the order.

(b) Suppose the server's reply means only that the data has been placed in the disk queue. Explain how this could lead to data loss that wouldn't occur with a local

disk. Note that a system crash immediately after data was enqueued doesn't count because that would cause data loss on a local disk as well.

Consider a situation when some data is written to the server by a client process. In the normal course of events, the client system shuts down graciously and flushing its buffers. At this stage, the data on a local disk is safe however; if a server crash happens to occur, this would cause loss of client data remaining in the server's buffers. In this situation, client will not be able verify that the data was safely written out by the server.

(c) An alternative would be for the server to respond immediately to acknowledge the write request, and to send its own separate CHAN request later to confirm the physical write. Propose different CHAN RPC semantics to achieve the same effect, but with a single logical request/reply.

The proposed CHAN RPC semantics to achieve the same effect are given as under:

Proposed Approach

- Modify CHAN to support multiple independent outstanding requests on a single logical channel
- Support replies in an arbitrary order, regardless of the order in which they were received

Proposed Approach Outcomes

This will result in the server responding to multiple I/O requests in the most convenient order. Following are the outcomes of the proposed approach:

- A subsequent request cannot be served as an ACK of a previous reply; Explicit and noncumulative ACKs would be needed now.
- Retransmission Management changes would be required such as :
  - A list of the requests that hadn't yet been answered should be maintained by the client
  - A list of replies that had been sent but not acknowledged should be maintained by the server
- Limit on the size of the lists corresponding to window size are required

b) Suppose BLAST runs over a 10-Mbps Ethernet, sending 32K messages.

(a) If the Ethernet packets can hold 1500 bytes of data, and option-less IP headers are used as well as BLAST headers, how many Ethernet packets are required per message?

(b) Calculate the delay due to sending a 32K message over Ethernet

i. directly

ii. broken into pieces as in (a), with one bridge

Ignore propagation delays, headers, collisions, and inter-packet gaps.

Solution

(a) If the Ethernet packets can hold 1500 bytes of data, and option-less IP headers are used as well as BLAST headers, how many Ethernet packets are required per message?

Number of bytes required by IP and BLAST headers = 40 bytes
Bytes per packet left for data = 1500 – 40 = 1460 bytes
Size of one message = 32 K = 32 × 1024 = 32768 bytes

$$Number\ of\ packets\ required\ for\ a\ message = \frac{Size\ of\ message}{Bytes\ per\ packet\ for\ data}$$

Putting values, we get;

$$Number\ of\ packets\ required\ for\ a\ message = \frac{32768}{1460} = 22.4$$

$$Number\ of\ bytes\ in\ 22\ packets = 22 \times 1460 = 32120$$

$$\text{Re}maining\ bytes = 32768 - 32120 = 648\ bytes$$

$$Number\ of\ packets\ required = 22\ full\ packets + one\ 648\ bytes\ packet$$

(b) Calculate the delay due to sending a 32K message over Ethernet

i. directly
ii. broken into pieces as in (a), with one bridge

i. delay due to sending a 32K message over Ethernet directly

$$32\ KB = 32 \times 8Kbits = 256Kbits \approx 260Kbits$$

Time taken to send this data can be given by the following formula:

$$Time\ to\ send\ the\ data = \frac{Size\ of\ the\ data}{Link\ Bandwidth}$$

Putting values, we get;

$$Time\ to\ send\ the\ data = \frac{260\ Kbits}{10Mbps} = \frac{260\ Kb}{10,000Kbps} = 0.026\,\text{sec} = 26m\,\text{sec}$$

$$Time\ to\ send\ the\ data = 26m\,\text{sec}$$

ii. broken into pieces as in (a), with one bridge

By fragmenting the pieces by BLAST, the delay would be 1.5 msec. The total delay can be given as follows:

$$Total\ Delay = 26m\,\text{sec} + 1.5m\,\text{sec} = 27.5m\,\text{sec}$$

$$Total\ Delay = 27.5m\,\text{sec}$$

c) Assume that TCP implements an extension that allows window sizes much larger than 64 KB. Suppose that you are using this extended TCP over a 1-Gbps link with

a latency of 100 ms to transfer a 10-MB file, and the TCP receive window is 1 MB. If TCP sends 1-KB packets (assuming no congestion and no lost packets):

(a) How many RTTs does it take until slow start opens the send window to 1 MB?

(b) How many RTTs does it take to send the file?

(c) If the time to send the file is given by the number of required RTTs multiplied by the link latency, what is the effective throughput for the transfer? What percentage of the link bandwidth is utilized?

Solution

(a) How many RTTs does it take until slow start opens the send window to 1 MB?

The size of the window gets doubled with every RTT in Slow Start. Mathematically, at the end of every k^{th} RTT, the window size is 2^K KB. Therefore, for the above problem, we have $1\text{MB} = 2^{10}\text{KB}$. Here $K = 10$, so it will take 10 RTTs until slow start opens the send window to 1 MB.

(b) How many RTTs does it take to send the file?

After 10 RTTs, the number of transferred KBs are $1\text{MB} - 1\text{KB} = 1023\text{KB}$ resulting in window size of 1MB. Until the maximum capacity of network is reached, slow start keeps on doubling the window with each RTT. The remaining 9 MB will be transferred in 4 further RTTs as follows:

RTT 1 → 1 MB
RTT 2 → 2 MB
RTT 3 → 4 MB
RTT 4 → 1 MB

All the above RTT sizes are below the maximum capacity of the link (12.5 MB).

Therefore, total number of RTTs in which the file is transferred is as follows:

$$Total\ number\ of\ RTTs = \underbrace{10\ RTTs}_{1\text{MB}} + \underbrace{4\ RTTs}_{9MB} = 14\ RTTs$$

$$Total\ number\ of\ RTTs = 14\ RTTs$$

(c) If the time to send the file is given by the number of required RTTs multiplied by the link latency, what is the effective throughput for the transfer? What percentage of the link bandwidth is utilized?

Time to send the file can be given by the following formula:

$$Time\ to\ send\ the\ file = Number\ of\ Required\ RTTs \times Link\ Latency$$

Putting the given values, we get;

$$Time\ to\ send\ the\ file = 14 \times 100\ m\sec$$

$$Time\ to\ send\ the\ file = 14 \times 100\ ms = 1.4\sec$$

$$Time\ to\ send\ the\ file = 1.4\sec$$

Now, the effective throughput can be calculated as follows:

$$Effective\ Throughput = \frac{Size\ of\ the\ file}{Time\ to\ send\ the\ file}$$

Putting the given values, we get;

$$Effective\ Throughput = \frac{10MB}{1.4\sec} = 7.14\ MBps = 7.14 \times 8Mbps = 57.12Mbps$$

$$Effective\ Throughput = 57.12Mbps$$

The utilized percentage of the link bandwidth can be given by the following formula:

$$Utilized\ Link\ Bandwidth = \frac{Effective\ Throughput}{Link\ Bandwidth}$$

Putting the given values, we get;

$$Utilized\ Link\ Bandwidth = \frac{57.12Mbps}{1Gbps} = \frac{57.12Mbps}{1000Mbps} = 0.057$$

$$Percentage\ of\ Utilized\ Link\ Bandwidth = 0.057 \times 100 = 5.7\%$$

$$Utilized\ Percentage\ of\ the\ Link\ Bandwidth = 5.7\%$$

Effective throughput is 5.7% of the available link bandwidth

d) Suppose a router has three input flows and one output. It receives the packets listed in following table all at about the same time, in the order listed, during a period in which the output port is busy but all queues are otherwise empty. Give the order in which the packets are transmitted, assuming

(a) fair queuing

(b) weighted fair queuing, with flow 2 having weight 2, and the other two with weight 1

Packet	Size	Flow
1	100	1
2	100	1
3	100	1
4	100	1
5	190	2
6	200	2
7	110	3
8	50	3

Solution

(a) order in which the packets are transmitted, assuming fair queuing

In order to find out order in which the packets are transmitted we need to calculate the finishing times F_i. Regarding the clock speed, we can take $A_i = 0$ for all packets. Therefore, we can express F_i, the cumulative per-flow size, as follows:

$F_i = F_{i-1} + P_i$

F_i corresponding to all the packets is given in the following table:

Packet	Size	Flow	F_i
1	100	1	100
2	100	1	200
3	100	1	300
4	100	1	400
5	190	2	190
6	200	2	390
7	110	3	110

8	50	3	170

Therefore, we will send packets in increasing order of F_i as follows:

Packet 1
Packet 7
Packet 8
Packet 5
Packet 2
Packet 3
Packet 6
Packet 4

(b) weighted fair queuing, with flow 2 having weight 2, and the other two with weight 1

To give flow 1 a weight of 1, we have $F_i = F_{i-1} + P_i$. To give flow 2, a weight of 2, we divide each of its F_i by 2 i.e. $F_i = F_{i-1} + P_i/2$. To give flow 3 a weight of 1, we have $F_i = F_{i-1} + P_i$. Assuming no waiting, F_i corresponding to all the packets is given in the following table:

Packet	**Size**	**Flow**	**Weighted** F_i
1	100	1	100
2	100	1	200
3	100	1	300
4	100	1	400
5	190	2	95
6	200	2	195
7	110	3	110
8	50	3	170

Therefore, we will send packets in increasing order of weighted F_i as follows:

Packet 5
Packet 1
Packet 7
Packet 8
Packet 6
Packet 2
Packet 3
Packet 4

Question No. 3:

Read the attached research paper entitled **"Performance Improvements of TCP by TCP Reno and SACK Acknowledgement"** with this assignment very carefully. Answer the following questions:

1. How proposed technique improved the performance of existing TCP variants? Differentiate among different TCP Variants discussed in this paper.

Read the attached research paper entitled **"On the Intertwining between Capacity Scaling and TCP Congestion Control."** with this assignment very carefully. Answer the following questions:

2. How interaction between capacity scaling and TCP Congestion Control affects the performance in terms of Quality of Service (QoS) and energy saving? Discuss in the light of paper.

Solution

Please! Consult the mentioned Research Paper.