CS 2200 Intro to Systems and Networks

HW 5 – MT & Networking

Part I: Producer-Consumer

This problem has you solve the classic "bounded buffer" problem with one producer and multiple consumer threads.

The program takes the number of consumers as an argument (defaulting to 1) and a sequence of numbers from stdin. We give you a couple of test sequences: shortlist and longlist. For more explanation of how this works, see the comment at the top of hw4.c

The producer thread reads the sequence of numbers and feeds that to the consumers. Consumers pick up a number, do some "work" with the number, then go back for another number.

The program as provided includes output from the producer and consumers. For reference, a working version of the code with a bounded buffer of size 10 running on shortlist with four consumers produces this output (the comments on the right are added): (NOTE: Your printed console output may not match what is shown identically due to the randomness of thread scheduling. However, your output should show all entries being produced in the correct order and consumed in the correct order).

```
turku% ./hw4 4 < shortlist

main: nconsumers = 4

  consumer 0: starting

  consumer 1: starting

  consumer 2: starting

  consumer 3: starting

  producer: starting

producer: 1

producer: 2
```

```
producer: 3

producer: 4

producer: 5

producer: 6

producer: 7

producer: 8

producer: 9

producer: 10

producer: 9

producer: 8

producer: 7

producer: 6

  consumer 0: 1

producer: 5

  consumer 1: 2

producer: 4

  consumer 2: 3

producer: 3

  consumer 3: 4

producer: 2

  consumer 0: 5

producer: 1

  consumer 1: 6

producer: read EOF, sending 4 '-1' numbers

  consumer 2: 7

  consumer 3: 8
  consumer 0: 9

  consumer 1: 10

producer: exiting

  consumer 2: 9

  consumer 3: 8

  consumer 0: 7

  consumer 1: 6

  consumer 2: 5

  consumer 3: 4

  consumer 3: exiting

  consumer 0: 3

  consumer 0: exiting

  consumer 2: 1

  consumer 2: exiting

  consumer 1: 2

  consumer 1: exiting
```

Finish the bounded-buffer code in hw4.c , adding synchronization so that the multiple threads can access the buffer simultaneously.

Part II: The Internet Protocol Stack

A. Describe the purpose of each of the 5 layers of the Internet Protocol Stack.

Application Layer – Support network-based applications, , It has protocols such as HTTP(web), SMTP(electronic mail), FTP(file transfer), and these protocols provide a common language for application-level entities to communicate.

Transport Layer – It takes the message from application-layer, break them into packets and deliver them between communication. TCP and UDP are two most used protocols today. As described in Network Layer part of the book, it also collates the packets into message for application layer.

Network Layer – It routes the packet from source to destination. At sending side, it finds a way to get the packet(from transfer layer) to destination address. At receiving end, it passes packet back to transport layer. It also breaks IP packets into fragments and reconstruct them back. It uses routing algorithms, tables, and destination address for packet to determine the next hop.

Link Layer – When IP packets received from Network-Layer, Link-layer delivers fragments of packets to next hop and repeats until the packet gets to destination by choosing physical medium, such as Ethernet, token ring, and 802.11.

Physical Layer – Physically move bits of the packet from one node to next. Link layer uses physical media to move the bits. Example of physical-layer protocol is Ethernet.

Brief Summary –
Source side : AL give message to TL ➔ TL breaks message into packets, and request delivery ➔ NL finds the route to destination for packet, breaks them into fragments if necessary ➔ LL delivers those fragments of packets to hops until it gets to destination using physical media ➔ PL physically move the bits of packet.
Destination side : PL receive bits from destination PL ➔ NL reconstruct those fragments into original IP packet ➔ TP reconstruct those packets into original message ➔ Destination AL receives the message.

Part III: Basic Networking Concepts

A. Assume that packets have a header that is 32 bytes long and a payload of 256 bytes. If you wish to send a message that is 2048 bytes long, how many packets will be required? How many bytes in total will need to be sent?
2048 / 256 = 8 packets
(32 + 256) * 8 = 2304 bytes

B. Given these parameters:

wire bandwidth	20Mbps (20 x 10^7 bits per second), full-duplex
time-of-flight	50ms (50 x 10^-3 seconds)
sender overhead	300us (300 x 10^-6 seconds)
receiver overhead	300us (300 x 10^-6 seconds)

We are sending a 20000-bit message. What is the observed throughput? Hint: Use the equations on Page 690 of Chapter 13 and follow the worked out examples that use these equations.)

S + Tw + Tf + R = message transmission
S = 300 * 10^-6 = 0.0003
Tw = 20000 / (20 * 10^7) = 2 / 20000 = 1 / 10000 = 0.0001
Tf = 50 * 10^-3 = 0.05
R = 300 * 10^-6 = 0.0003
Message transmission time = 0.0507 sec
Throughput = 20000 / message transmission ➔ 20000 / 0.0507 = 394477.318

Part IV: Transport Layer

A. What are the two main protocols used for the Transport Layer? How do they differ? In what cases would you use one or the other?
TCP, UDP
TCP (transmission control protocol) does in-order delivery and is guaranteed to deliver application data. It gives stream semantics for data transport.
UDP (user datagram protocol) is not reliable (data could be lost, not in order). It gives datagram semantics.
They differ in the fact the TCP is slow because it is more secure, UDP is fast because they don't care if not all data is transmitted.
In case of Video stream, you would use UDP because you don't care if some data is not transmitted in video stream as you will barely notice the difference and its still doable.
In case of sending mail, or receiving mail, and web, you would use TCP for security purpose and make sure data is sent (ack from receiving end).

B. Describe the Stop and Wait protocol. Why is the sequence number necessary?

When the sender sends a packet, it stops to wait for ACK from receiver so that sender can continue to send next packet. It ACK is not received for certain period of time, then sender retransmit the packet. Reason for no ACK is two, Packet lost en route, or ACK lost en route.
Sequence Number is important in this case because it identities each packet therefore it will reduce the possibilities of duplicate call in case of either data packet or ACK packet loss. (retransmitting packet or retransmitting ACK). Also it helps reconstructing the packets into message therefore making out of order deliver okay, and doesn't matter the size of the message.

Part V: Networking Layer

A. What is the purpose of an IP address?
IP address works as unique identity for each device/network to identify a device from another device within network OR network from another network in inter-network communication. Therefore being able to communicate with other device via IP packet

B. Describe circuit switching and packet switching. Why does the Internet use packet switching?
Circuit switching – A dedicated point to point connection (circuit) has to be established during calls. During this, other network requests are denied.
Packet switching – when packet arrives at a switch, the switch places the packet on appropriate outgoing link after examining the destination address and routing information of the packet; switch looks for the most efficient route.
Internet also uses circuit switching for certain purpose in my knowledge. Anyway, internet is mostly suitable for packet switching because there are many users in the network, and packet switching allows shared data path for delivering packets, which is more cost efficient than to have an established path. Even though packet switching may have some delay (lag), it is not that significant for most uses (like loading a most website doesn't take that long, but something like video gaming would require higher performance and may cause more delays).

Part VI: The Link Layer

A. Explain how the Ethernet protocol works

It controls how individual devices access network, configured in network, and fast data transfer, therefore, It can deals with problem of large distances and arbitrary number of units concurrently competing for use of medium.

B. Explain the difference between CSMA/CD and CSMA/CA. Why is the latter used in some circumstances?

Both are how Ethernet arbitrate the uses of medium for unit.

(collision detected, or collision avoided).

CSMA/CD is when device wants to transmit, it waits on medium to become an idle state (medium not in use) before it starts the transmission. However, after the start of transmission it also listens for collision because multiple stations sense the cable and it may also start their transmission as they both get to conclusion that medium is idle. Upon detecting collision, the transmission stops and waits for random amount of time before repeating sensing, transmitting, and listen for collision part.

However, CSMA/CA, collision avoid require source to explicitly get permission to send before sending frame to the destination. Steps to get permission; source sends RTS(request to send) packet to destination, destination responds with CTS(clear to send) packet back to source.

CSMA/CA is used in wireless medium as we cannot determine if there was a collision on the medium.

CSMA/CD

C. Define RTS and CTS and explain their use in the MAC layer.

RTS is request to send, this control packet is sent when source wants to do data transmission and ask destination for permission. CTS is clear to send, this control packet is sent after destination receives RTS and responds the permission to the source by sending CTS.

Both are used during CSMA/CA protocol to achieve data transmission while avoiding collision. Using RTS and CTS, MAC layer can control and access to physical medium such as all the device in the local area network can hear RTS and CTS control packets and therefore they don’t try to send an RTS packet into the medium until the data transmission is complete to avoid collision.