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: Supports network-based applications (League, Twitch, etc) using protocols (HTTP/SMTP/FTP).

Transport Layer: Takes message from application layer and sends/receives it in packets at source or destination. Uses TCP/UDP protocols.

Network Layer: Routes message over the network, via whichever path is appropriate, from source to destination. Uses IP protocol.

Link Layer: Carries IP packets between nodes that the message encounters en route to its destination. Uses Ethernet/Token Ring protocols.

Physical Layer: Moves the message's bits between nodes, as the physical piece of copper wire, optical fiber, etc carrying electrical signals.

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, so 8 packets are required.
8 packets = 8*(32+256) = 2304 bytes total.

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 = 300 x 10^-6
Tw = 20000 / 20 x 10 ^ 7
Tf = 50 x 10^-3
R = 300 x 10^ -6
Transmission time = S + Tw + Tf + R = 0.0507 seconds
Observed Throughput = Message size / transmission time = 20000 / 0.0507 = 394477.3 bits/sec

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 and UDP. TCP treats the application data as a continuous byte stream, which is then grouped into chunks called segments before sending. UDP is datagram oriented, meaning data is divided into discrete chunks.

B. Describe the Stop and Wait protocol. Why is the sequence number necessary?
Stop and Wait sends a packet, stops sending, and waits for the receiver to send an acknowledgement (ACK) that the packet was received. Once this acknowledgement is read by the sender, it sends the next packet. If ACK is never detected within the timeout, the sender retransmits the packet. Repeat until all packets have been sent. Sequence number are

unique signatures which ensure that the sender/receiver can check the packets to see if some have been lost (sequence numbers will skip).

Part V: Networking Layer

A. What is the purpose of an IP address?
To be a multipart address that supports hierarchial addressing of networks such that an network or individual device can be specified within the hierarchy of the Internet.

B. Describe circuit switching and packet switching. Why does the Internet use packet switching?
Circuit switching refers to the physical link (over wires/fibers) broken into segments by a number of switches which together create a connection between two endpoints of a call. It guarantees quality of service but at the risk of underutilization.
Packet switching does not reserve all the segments between the two endpoints. Instead, at every switch, the packet is sent towards its destination on an appropriate link. Input and output buffers address queuing delay.
The Internet uses packet switching over circuit switching because it is simpler, allows complete utilization of bandwidth, and guaranteed transmission time is unimportant.

Part VI: The Link Layer

A. Explain how the Ethernet protocol works

It is similar to a bus, but unconfined to a box. Using CSMA/CD as its arbitration protocol, it can serve as a bus between an arbitrary number of devices over longer distances (within a building).

B. Explain the difference between CSMA/CD and CSMA/CA. Why is the latter used in some circumstances?
While both of them are multiple access protocols, CD assumes that the stations can tell when a collision occurs (such as on a wire), while CA does not. Simultaneously sending transmissions and detecting collisions is infeasible on a wireless connection. CA also resolves the hidden terminal problem by which two

stations' messages collide but neither station is aware of it.
C. Define RTS and CTS and explain their use in the MAC layer.

Ready-to-send and clear-to-send operate as a handshake process which gets explicit permission to send before transmitting. RTS and CTS signals are sent through the MAC layer from one NIC to another over the appropriate bridges and switches (which recognize specific MAC addresses).