

<b>CS433/533: Computer Networks</b>
-------------------------------------

## **Exam 1: Application, Transport, and Network**

**03/29/2006**

**2:30-4:00 pm**

- This exam is closed book. However, you may refer to a sheet of 8.5"x11" paper of your own design.
- *Keep your answer concise.*
- Show your reasoning clearly. If your reasoning is correct, but your final answer is wrong, you will receive most of the credit. If you just show the answer without reasoning, and your answer is wrong, you may receive no points at all.

**Name:** \_\_\_\_\_

Applications (20 points)	Transport (26 points)	Routing (29 points)	Total (75 points)

**1. [20 points] Application Design**

a) [5 points] A challenge in designing a large-scale, distributed application is how to obtain the IP address of the server. Joe from Yale CS designs a new service called `wickedpedia`, which allows users to search and update a large database using keywords. The service is implemented by a server running on a zoo machine named `wickedpedia.cs.yale.edu`. The server listens at port 6789. Assume the domain `cs.yale.edu` is a sub-domain of `yale.edu`. When a remote client tries to connect to `wickedpedia.cs.yale.edu`, which servers might be queried to obtain the IP address of the server?

b) [5 points] The new service runs on top of TCP, and in the initial design, the server is implemented in a single process with the main body being:

```
listens at local port 6789
while (true) {
    if (a new connection has finished three-way-handshake) {
        create a connection socket from the connection
        read the request from the new socket
        process the request and write the reply to socket
        close the connection socket
    } // end of if
} // end of while
```

Joe's friends argue that this design has problems. What are the major problems of this design and can you propose a fix?

c) [5 points] Assume the new service becomes very popular, and now multiple computers are needed to handle the requests. Joe wants that the service still be known to be at `wickedpedia.cs.yale.edu`. However, he wants the ability to distribute the requests to multiple server programs running on multiple zoo computers each listening at 6789. Joe's friends have access to `cs.yale.edu` DNS server. Can Joe use DNS to distribute the requests among multiple servers?

d) [5 points] Wickedpedia is more popular than ever. Joe is thinking about turning his service into a peer-to-peer service so that others can contribute and manage different sets of keywords without a single bottleneck or point of failure. In your opinion, which P2P architecture is more appropriate, Napster-style, Freenet, or CAN? Please justify briefly.

## 2. [26 points] TCP Reliable Transport and Congestion Control for Long-Delay, High-Bandwidth Links

a) [5 points] One problem of running TCP on long-delay links is the delay introduced by three-way-handshake (TWH). Some people propose that we remove TWH for long-delay links. Do you support or disagree with this proposal? Please justify.

b) [5 points] TCP can also be slow to fully utilize high-bandwidth, long-delay links such as optical fibers. Assume round-trip time 200 ms and packet size 1000 bits. Approximately how long does it take for the slow start (assume no delayed ACK) of a flow to reach the capacity of a 10 Gbps link?

c) [6 points] Another criticism of TCP is that additive increase (AI) is too slow after a multiplicative decrease (MD). Some people worry that the utilization of a link will be only 50% when buffer overflows and the rates of all flows are cut to half. Some others explain that due to link buffers, TCP can maintain full link utilization. How can a link buffer help and approximately how big the *buffer* should be for a link which can transmit  $B$  packets per RTT? Hint: recall the zigzag curve we used to derive Reno throughput when losses are deterministic.

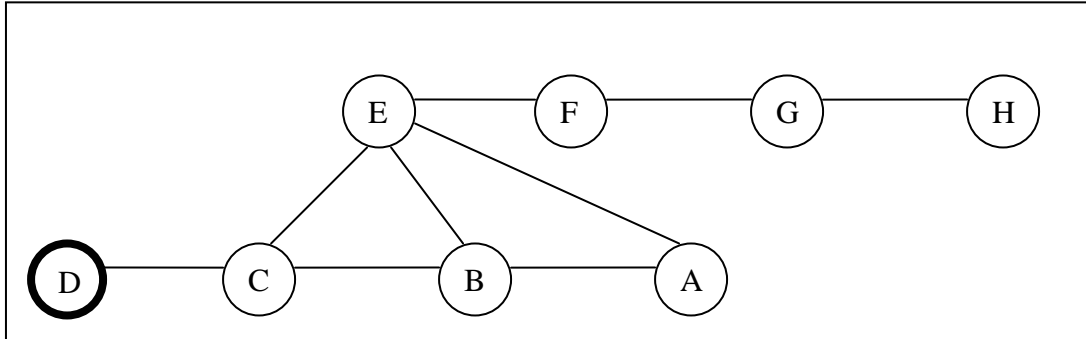
With some router support, a link could achieve high utilization without using TCP or a large buffer. One proposal is the following. A link with capacity  $C$  keeps track of  $n$ , the number of active flows using the link. The link also keeps track of  $R$ , the aggregated rate of all  $n$  flows. To reduce overhead, the link cannot keep track of the individual rates of the individual flows. The link estimates  $n$  and  $R$  every roundtrip time. Then it computes  $(C-R)/n$  and passes this value to one data packet of each flow. Note that  $(C-R)$  can be either positive or negative. The receiver of a flow extracts this value, and returns it to the sender in an ACK. The congestion control algorithm of the sender is simplified to just increasing its rate by the received value.

d) [5 points] Using this scheme, roughly how many round-trips are needed for a single flow starting at a very low rate to fully utilize a link?

e) [5 points] Some people argue that the proposed scheme might have a fairness problem. Is there really a fairness problem? If yes, can you propose a simple fix?

### 3. [29 points] Routing Protocols

Consider a network with 8 nodes: A, B, C, D, E, F, G, and H. All links have length 1. We consider D as the destination.



- a) [4 points] Using the *synchronous* Bellman-Ford algorithm (SBF), please compute the distances to D. The initial distance estimation of all nodes except D is  $\infty$ .

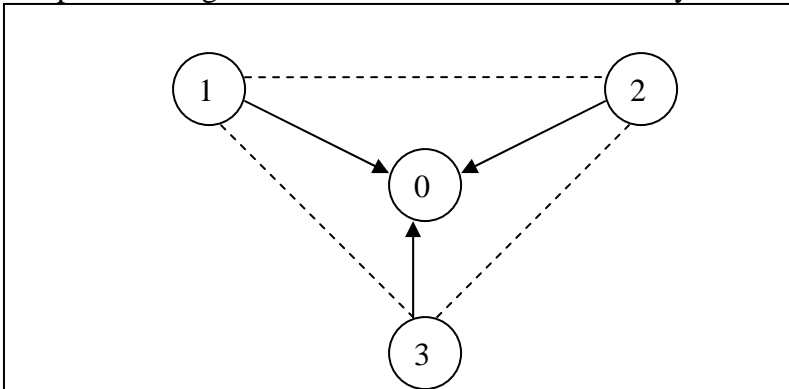
round	A	B	C	E	F	G	H	D
0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	

- b) [4 points] Few networks run SBF; instead they run asynchronous Bellman-Ford (ABF). However, ABF may have “churns,” i.e., frequent routing message updates before convergence. For the preceding network, what is a worst-case message update sequence when running asynchronous BF?

- c) [4 points] Another issue facing Bell-Ford based algorithms is count-to-infinity. One way to address this problem is to use reverse poison. Can this technique completely prevent count-to-infinity in the sample network? If yes, please briefly justify. If no, please give a scenario.
- d) [4 points] One way to avoid “churns” and count-to-infinity of ABF is to adopt a link-state protocol such as OSPF. But link state protocols face two problems in link state broadcast: link state sequence number reset after a router reboots and reconnection of two subnetworks after partition. How can these two problems be addressed?

- e) [4 points] Assume the IP addresses of the 8 nodes A, B, C, D, E, F, G, and H are 130.132.5.32, 130.132.5.33, ..., 130.132.5.39. Assume that the sample network is an autonomous system in the Internet with AS number 0. Node E is the BGP gateway of the AS. If E announces 130.132.5.0/29 as the aggregated address of the network, is it valid? If no, please propose a valid one.

- f) [4 points] E is connected to three providers: ASes 1, 2, and 3. ASes 1, 2, and 3 are peers of each other. All ASes follow standard (typical) Internet economy in export and path ranking. What are the BGP routes chosen by ASes 1, 2, and 3 to reach AS 0?



- g) [4 points] What are the BGP paths available (i.e., exported by its neighbors) at equilibrium at AS 2 to reach AS 0?