Dept. Math. & Comp. Sc.	**Computer Networks**
Vrije Universiteit	**06.01.1999**

Part I

This part covers the same material as the midterm exam.

1a Explain under which conditions it is better to follow a CSMA protocol instead of a collision-free protocol. *5pt*

1b A pure ALOHA protocol will, at best, give a channel utilization of 18%. What does this actually mean? *5pt*

1c Why is it not appropriate to apply CSMA for wireless communication? *10pt*

2a Explain the *count-to-infinity* problem, and a widely deployed (not entirely complete) solution. *10pt*

2b What is the difference between a *leaky bucket* and a *token bucket*? *5pt*

2c Explain the difference between *circuit switching* and *packet switching*, and the difference between *connectionless service* and *connection-oriented service*. Give an example for each *(switching,service)* combination. *10pt*

Part II

3a Devise an algorithm for ending a connection, such that both parties agree on disconnecting. Assume that no messages are lost, but message ordering is not guaranteed. *5pt*

3b Disconnecting should preferably be done only when both parties agree. However, guaranteeing that agreement can be reached is impossible when messages can be lost. Explain why. *10pt*

3c Transport protocols generally use a *buffer credit grant* mechanism. Why? *5pt*

4a If Alice wants to send a secret (legally binding) offer to Bob over a network, what should she do? Explain why your solution works. *5pt*

4b Explain how the Diffie-Hellman shared key exchange algorithm works, and why this algorithm was invented. *10pt*

5 Explain what happens when a Web browser has to display the data referenced by URL ftp://ftp.cs.vu.nl/pub/steen/file.ps *10pt*

Final grade: *(1) Add, per part, the total points. (2) Let T denote the total points for the midterm exam ($0 \leq T \leq 45$); $D1$ the total points for part I; $D2$ the total points for part II. The final number of points E is equal to* $\max\{T, D1\} + D2 + 10$.

BIJLAGE BIJ TOETS COMPUTERNETWERKEN 22.10.1999

```
01 void protocol4 (void) {
02   seq_nr next_frame_to_send, frame_expected;
03   frame r, s;
04   packet buffer;
05   event_type event;
06
07   next_frame_to_send = 0; frame_expected = 0;
08   from_network_layer(&buffer);
09   s.info = buffer;
10   s.seq  = next_frame_to_send;
11   s.ack  = 1 - frame_expected;
12   to_physical_layer(&s); start_timer(s.seq);
13
14   while (true) {
15     wait_for_event(&event);
16     if (event == frame_arrival) {
17       from_physical_layer(&r);
18       if (r.seq == frame_expected){
19          to_network_layer(&r.info);
20          inc(frame _expected);
21       }
22       if (r.ack == next_frame_to_send){
23          from_network_layer(&buffer);
24          inc(next_frame_to_send);
25       }
26     }
27     s.info = buffer;
28     s.seq = next_frame_to_send;
29     s.ack = 1 - frame_expected;
30     to_physical_layer(&s); start_timer(s.seq);
31   }
32 }
```