Explain Server/Client examples in http://www.tutorialspoint.com/java/java_networking.htm to let students understand basic JAVA socket API. You may want to project the webpage on the screen.

True/False:

1. BSD Socket API was created at Stanford.
False: At Berkeley in 1980.

2. TCP guarantees reliable, in-order, and at most once delivery.
True.

Short Answer:

1. What does ACID stand for? Explain each of them.
Answer:
Atomicity: all actions in the transaction happen, or none happen
Consistency: if each transaction is consistent, and the database starts consistent, it ends up consistent, e.g.,
– Balance cannot be negative
– Cannot reschedule meeting on February 30
Isolation: execution of one transaction is isolated from that of all others
Durability: if a transaction commits, its effects persist

2. What are Serial schedule, Equivalent schedules, and Serializable schedule?
Answer:
Serial schedule: A schedule that does not interleave the operations of different transactions
– Transactions run serially (one at a time)
Equivalent schedules: For any storage/database state, the effect (on storage/database) and output of executing the first schedule is identical to the effect of executing the second schedule
Serializable schedule: A schedule that is equivalent to some serial execution of the transactions
– Intuitively: with a serializable schedule you only see things that could happen in situations where you were running transactions one-at-a-time

Long Answer:

1. Consider a deposit of $100 and a withdrawal of $50 from account A. Initially, account A holds $1000. Assume the two operations are implemented by two threads, and assume that each instruction is atomic.

```
Thread1:              Thread2:
t1 = A;               t2 = A;
t1 = t1 + $100;       t2 = t2 – $50;
A = t1;               A = t2;
```

a) What are the possible outcomes of the above transfer? For each outcome give a possible execution of the threads leading to that outcome.

Answer:

Case 1: A = 1050

```
t1 = A;
t1 = t1 + 100;
A = t1;
                      t2 = A;
                      t2 = t2 - 50;
                      A = t2
```

Case 2: A = 950 (Thread1's update is lost)

```
t1 = A;
t1 = t1 + 100;
                      t2 = A;
A = t1;
                      t2 = t2 - 50;
                      A = t2
```

Case 3: A = 1100 (Thread2's update is lost)

```
                      t2 = A;
t1 = A;
                      t2 = t2 - 50;
                      A = t2
t1 = t1 + 100;
A = t1;
```

b) Assume Thread1 repeats the deposit operation 3 times and Thread2 executes the withdraw operation 2 times (i.e., there are three deposits of $100 each, and 2 withdrawals of $50 each). What are the potential outcomes?

Answer:

900, 1000, 1050, 1100, 1150, 1200, 1250, 1300

c) Give a simple solution to avoid incorrect outcomes.

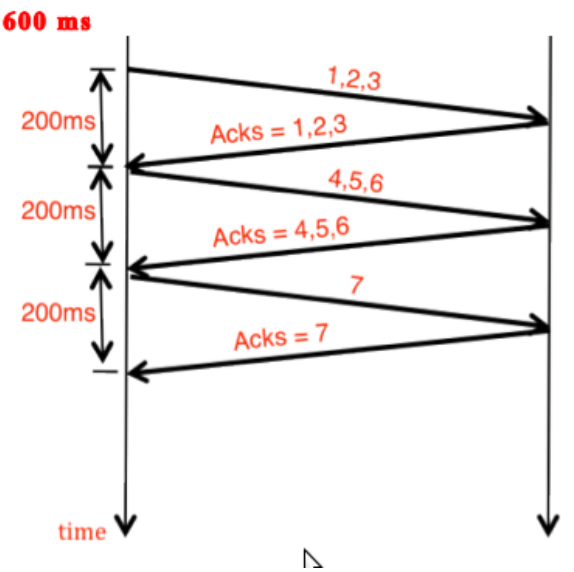
Answer:

Use a lock to access and modify variable A. Each thread acquires lock before starting and

releases after finishing (i.e., after writing back A).

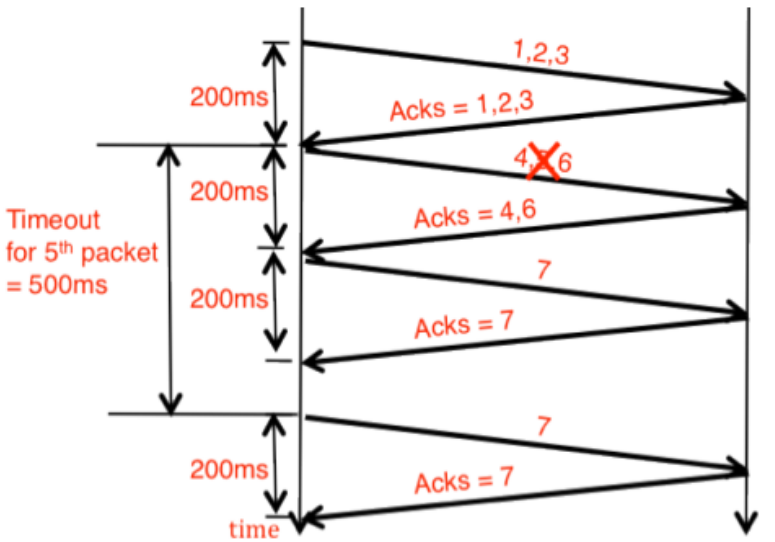
2. Assume two end-hosts using the sliding window protocol to implement flow control, and Go-back-n to implement reliability. Assume sender sends 7 packets. The window size at the receiver is 3 packets, the round-trip time is 200ms, and the retransmission timeout is 500ms. The transmission time of the packet is negligible, i.e., assume the size of a packet is 0. The time to send all packets is the interval between the time the sender sends the first packet and the time the sender receives the ack from the last packet.

a) How long does it take to send all packets, assuming no losses? Draw the time diagram.



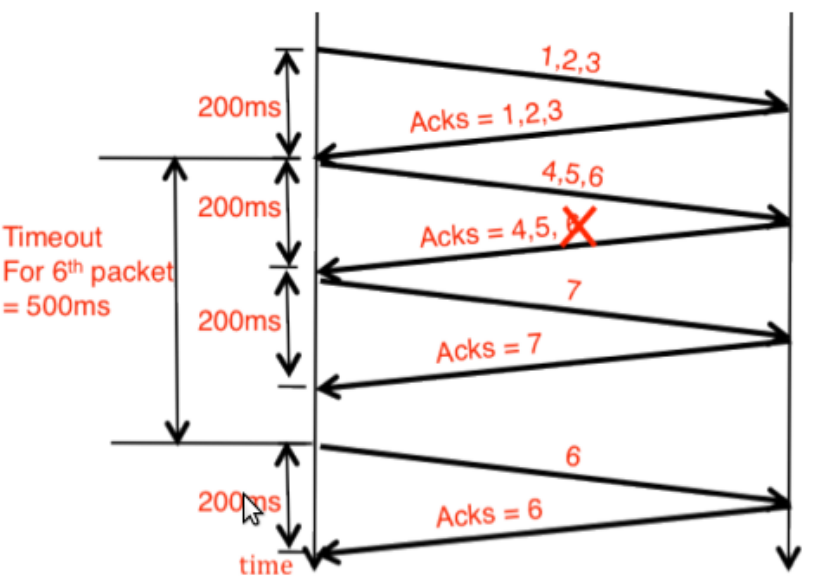
b) How long does it take to send all packets assuming the 5th packet is lost? Draw the time diagram.

900 ms

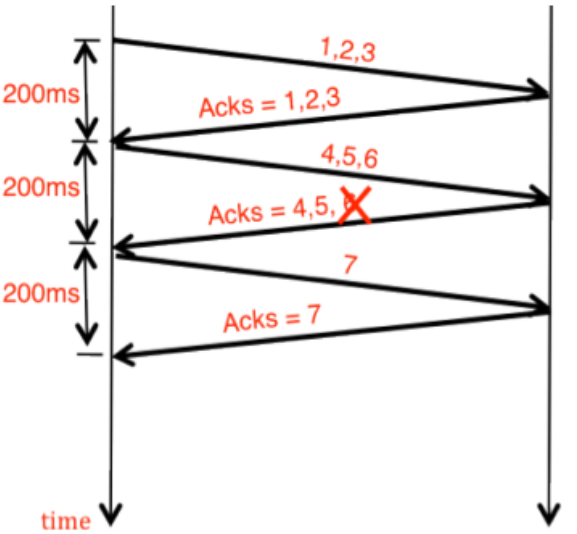


c) How long does it take to send all packets assuming the ack of the 6th packet is lost? Draw the time diagram.

900 ms



If the acks are cumulative then the answer is **600ms** (this answer was also considered correct):


1,2,3
200ms
Acks = 1,2,3
4,5,6
200ms
Acks = 4,5,
7
200ms
Acks = 7
time