University of California, Berkeley
College of Engineering
Computer Science Division – EECS

Spring 2017 Ion Stoica

Third Midterm Exam

April 24, 2017
CS162 Operating Systems

Your Name:	
SID AND 162 Login:	
TA Name:	
Discussion Section Time:	

General Information:
This is a **closed book and one 2-sided handwritten note** examination. You have 80 minutes to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points for that question. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time consuming.

Write all of your answers directly on this paper. ***Make your answers as concise as possible.*** If there is something in a question that you believe is open to interpretation, then please ask us about it!

Good Luck!!

QUESTION	POINTS ASSIGNED	POINTS OBTAINED
1	24	
2	15	
3	12	
4	16	
5	15	
6	18	
TOTAL	100	

P1 (24 points total) True/False and Why? **CIRCLE YOUR ANSWER**. For each question: 1 point for true/false correct, 2 point for explanation. An explanation cannot exceed 2 sentences.

a) When two processes mmap() the same region of a file, mmap() will always return the same set of virtual addresses to both processes, which represent a shared region of memory.

TRUE FALSE

Why?

b) A TCP sender can overflow the receiver's buffer.

TRUE FALSE

Why?

c) A process can wait for all of its children with a single call to wait(&status).

TRUE FALSE

Why?

d) Two-Phase Commit does not guarantee all ACID properties (Atomicity, Consistency, Isolation, and Durability).

TRUE FALSE

Why?

e) The end-to-end principle states that one should make the lower levels between the higher-level ends as reliable and secure as possible.

TRUE FALSE

Why?

f) FAT provides slower sequential access to a file than Unix File System.

TRUE FALSE

Why?

g) Using quorum consensus a writer can return before the data has been written to all replicas.

TRUE FALSE

Why?

h) With consistent hashing (e.g., Chord) if a new node is added to a system consisting of $N > 1$ nodes, this will lead to keys from every node to be moved around.

TRUE FALSE

Why?

P2 (15 points) **Transactions.** Consider two variables A and B which are initialized to 1, and 2, respectively:

A	1
B	2

Consider the following two transactions being attempted on A and B:

Transaction 1			Transaction 2		
A=3	B=4	Commit	A=5	B=6	Commit

For the following scenarios, list which ACID property was violated or "none" if the transactions obeyed ACID, explain your answer in 1 sentence:

a) (3 points) Transaction 1 successfully committed, then the computer crashed after step 2 (B=6) of transaction 2 (but before writing the commit record to the log).
State after Recovery:

A	3
B	4

b) (3 points) The transactions were run on two threads (at the same time) and have both committed. Note: concurrent transactions may complete in any order. State which order your answer assumes (if any).
State after Recovery:

A	3
B	4

c) (3 points) The transactions were run on two threads (at the same time) and have both committed. Note: concurrent transactions may complete in any order. State which order your answer assumes (if any).
Final State:

A	3
B	6

d) (3 points) **Transaction 1 successfully committed**, then the computer crashed during transaction 2 (before commit).
State after Recovery:

A	5
B	4

e) (3 points) Both transactions committed (in order txn1,txn2), then the computer crashed.
State after Recovery:

A	3
B	4

P3 (12 points) **Synchronization**. Consider the following C program.

```
int thread_count = 0;

void *thread_start(void *arg) {
 thread_count++;
 if (thread_count == 3) {
   char *argv[] = {"/bin/echo", "162 is the best!", NULL};
   /* Replace current process image with a new process image
    * that runs program pointed by "*argv" with arguments "argv" */
   execv(*argv, argv);
 }
 printf("Thread: %d\n", thread_count);
 return NULL;
}

int main(int argc, char *argv[]) {
 int i, status;
 for (i = 0; i < 2; i++) {
   pid_t pid = fork();
   if (pid != 0) {
     thread_count++;
     wait(&status);
   }
   pthread_t *thread = malloc(sizeof(pthread_t));
   /* start new "thread". The new thread starts execution by invoking
    * "thread_start()" with NULL arg. */
   pthread_create(thread, NULL, &thread_start, NULL);
   pthread_join(*thread, NULL);
 }
 return 0;
}
```

a) (8 points) Provide one possible output from this C program.

b) (4 points) Is the output deterministic? If yes, give a short explanation why. If no, explain how to make the output deterministic.

P4 (16 points) **Queueing theory** Consider a system with a single queue and a single server. The arrival rate of the requests follows a Poisson process, and the service time is exponential distributed. Thus, the system implements an M/M/1 queue. Assume that:

- The utilization U is 50%.
- The average service time T_{ser} = ⅙ seconds.

a) (4 points) What is the average service rate μ and arrival rate λ?

b) (4 points) What are the average time spent in the queue T_q and the response time?

c) (4 points) What is the average length of the queue L_q?

d) (4 points) Assume that we add an additional server with the same service time (T_{ser} = ⅙ seconds) to help service our queue. What is the average time spent in the queue T_q and response time for the new system, assuming the same arrival rate as at point (a), and that requests are balanced equally across the two servers?

P5: (15 points) **Caching and storage**: You have been hired at a hot startup located in downtown Berkeley and you are tasked with building a key-value store backed by spinning disks. Each disk has the following performance:

- Bandwidth: 100MB/s
- Disk controller latency: 1ms
- Average rotational latency: 3ms
- Seek latency: 3ms

In front of the backing store you have provisioned an in-memory caching layer. The performance of this layer is bounded by the capacity of the network, which is 100MB/s.

Each access to the key-value requests translates into one access into the cache and, if there is a cache miss, to one access to one backing disk. Each access results in reading 1 key-value of 100 KB, which is stored contiguously on the disk (i.e., on adjacent sectors.) Ignore the network and cache latency.

a) (5 points) Considering your workload hits the cache 50% of the time, what is the average service time (i.e., the time/latency to retrieve a key-value)?

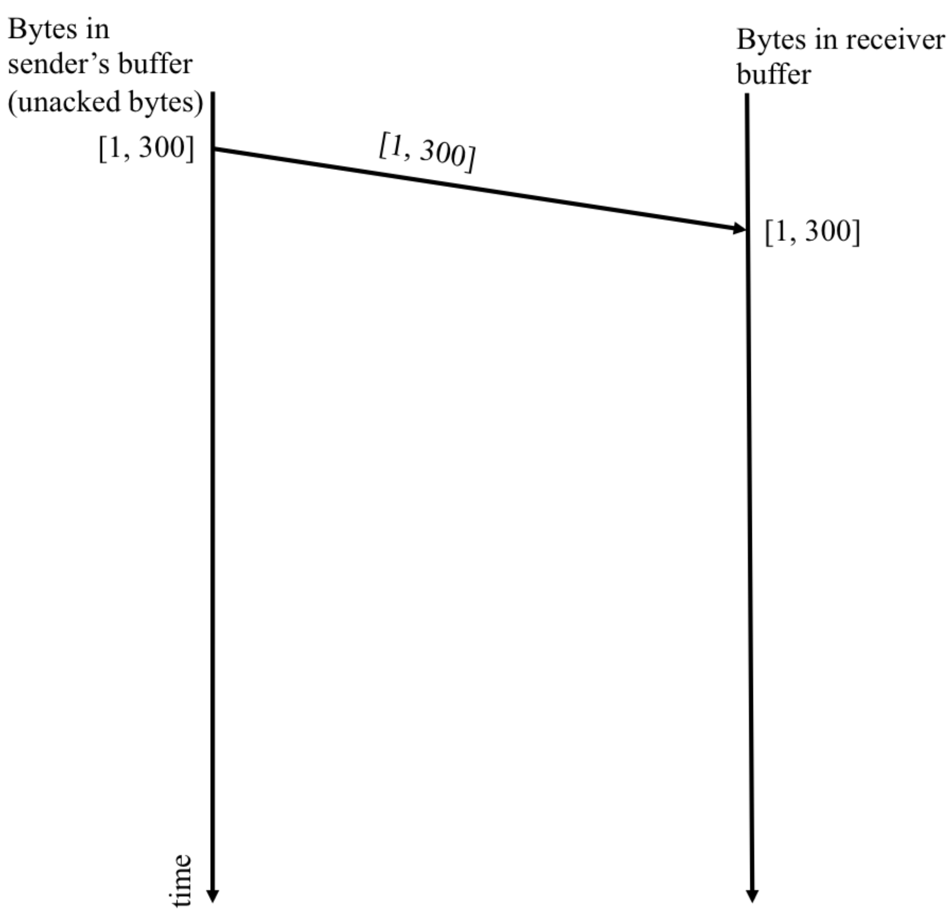
b) (5 points) What is the maximum number of requests (reads) per second that this system can handle?

c) (5 points) Now consider that you decide to change your spinning disks with faster SSDs to achieve an average service time (i.e., the time/latency to retrieve a key-value) of 1.25ms. Assume that the SSD controller latency is 1ms, the same as the HDD controller latency. What is the bandwidth of the SSD?

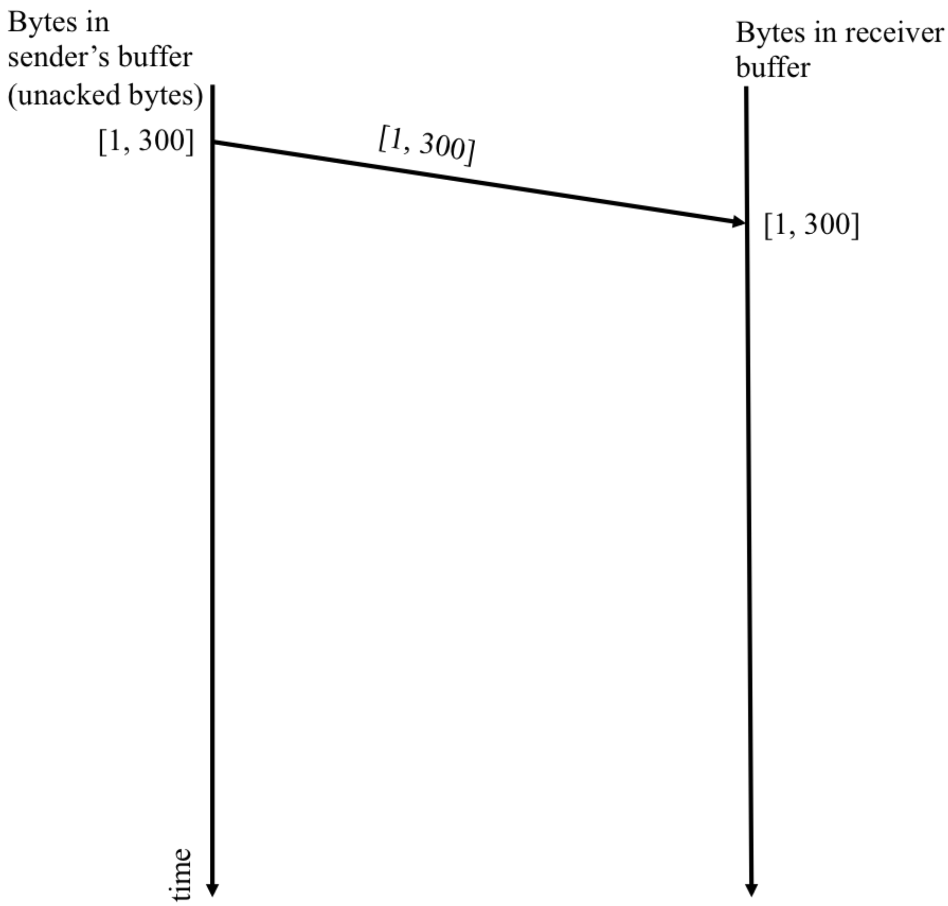
P6 (18 points) **TCP Flow Control.**
Assume host S uses TCP to send 1000 bytes of data to host R. We make the following assumptions:

- The maximum packet size is 300B.
- Receiving buffer of R is 500B.
- The delay between the machines is much larger than the time it takes to S to send 1KB, i.e, S could send 1KB faster than it takes to hear back an ack from R.
- R consumes the bytes as soon as it receives them (i.e., R's OS sends data to receiving app as soon as it gets them.)

a) (6 points) Assume no packet or acknowledgement is lost. Draw the packet diagram also indicating the bytes in R's sending buffer and the bytes in S's receiving buffer. The first packet with the associated information is drawn for you.



b) (6 points) Draw the packet diagram assuming the acknowledgement of the first packet is lost. No other packet or acknowledgement is lost.



c) (6 points) Draw the packet diagram assuming the third packet from the sender is lost. No other packet or acknowledgement is lost.