Midterm #2
Fall 2000 15-712

Please write your name on the following line:

____

This midterm is open book and open notes.

You should write brief, precise, and legible answers. Rambling brain-dump answers may be punished twice — you run out of time writing them and we run out of patience reading them. :–)

There are eight questions but we will only grade six. (We will also look at the two bonus questions, but those questions do not count towards the six you need to complete.) List the six questions we should grade:

____

Name: ____________________

1. Lamport gives an approach to defining a total order on events in a distributed system. Explain briefly how this total ordering could be agreed upon by all nodes in the system in the face of Byzantine failures.

   The nodes can use the solution to the Byzantine Generals problem to come to consensus on the source and logical clock assigned to each event. In detail, rather than just sending the 1-bit information "attack/retreat" from the original Byzantine Generals problem, the nodes come to agreement on the source and clock values for each event. With Lamport's "happened before" function, the nodes can come to agreement on the total order. This is assuming, of course, that there are at least $3m + 1$ *nodes when* m *nodes can fail.*

Name: ____________________________

2. A "barrier" in parallel programming is a synchronization point at which each thread waits until all threads have reached it. Explain how one could implement barrier synchronization in a distributed parallel program.

   Let's begin assuming the threads communicate via RPC. Instantiate one serializer thread in the parallel program. Each thread sends an RPC to it when it reaches the barrier point, and the serializer thread only answers when it has heard from everyone.

   If instead the threads communicate via shared memory, then this can be solved with a monitor and condition variable. Let's say there are n threads in the program. The monitor protects a counter that is initialized to zero. The barrier function increments the counter. If the counter is less than n, it calls "wait" on the condition. If the counter equals n, the code resets n to zero and calls "broadcast" on the condition, waking all of the waiting threads.

Name: ___________________________________

3. When using locks, it's often possible to get into deadlock conditions. Explain how transactions can be used to detect and break deadlocks.

   Ongoing transactions that are thought to be in deadlock (e.g., because they've been blocked on a lock for a very long time, or a transaction manager has detected deadlock via a graph traversal algorithm) can be selectively aborted. This causes any locks they hold to be freed, which in turn can allow other transactions in the lock cycle to make forward progress. When the aborted transaction is later restarted, the deadlock conditions may be cleared.

Name: ____________________________________

4. Many security researchers have assumed that criminals would focus their attacks on cryptanalysis. List at least 2 reasons why this is rarely true, explain their significance, and give an example that illustrates each.

   The question is worded awkwardly. There's one meta-reason why criminals don't focus their attacks on cryptanalysis: there are easier ways than cryptanalysis to break security. What we were looking for were two specific examples of why or how systems are easier to compromise. Here are a few such examples:

   1. Implementations are often buggy. This is significant because some of the bugs (e.g., unchecked buffer sizes or parameter settings) can allow attackers to take control of, disrupt, or otherwise compromise a system. The most common example of this is the canonical buffer overflow, as illustrated in the ftp bug exploited by the Internet worm.

   2. Systems are often configured poorly or incorrectly. This is significant because a poor configuration opens the door to attacks much like buggy implementations. One example of this is poorly chosen passwords. A similar example is poorly protected encryption keys.

   3. Operators can be fooled or bribed. This is significant because the right insider already has the access rights required to do whatever the criminal desires. An example of this is pretending to be a system administrator and asking for the password.

Name: ________________________________

5. The X Window System uses something like Hauser's serializer thread to handle graphics commands from clients of the X server. These commands are essentially RPCs to the X server. The single server thread handles requests in a FIFO order. Given that different client threads may have different priorities (and CPU scheduling levels), explain what problem this could cause. How could it be fixed?

   The problem is priority inversion, that is, a higher priority task being unnecessarily delayed by a lower priority task. Specifically, a FIFO serializer will cause X Window clients to be scheduled according to their request rate, not according to their priorities. Imagine two processes, one with high and one with low priority. After the high priority process queues a command, it will go to sleep, enabling the low priority process to queue a command. So each process will get 50% service. One way to fix this is to have different queues for different priority tasks inside of X instead of just one FIFO queue. This would effectively introduce priority inheritence into the X server. (Note: this requires preemptive request processing to completely eliminate the inversion.)

Name: ____________________________________

6. Consider a collection of distributed clients accessing records on a central server using optimistic concurrency control to ensure serializability. Occasionally, a client could resort to traditional locking during a transaction. What circumstance might cause this to happen?

   If there is high contention on one record, the clients involved might continually fail the validation stage in optimistic concurrency control, causing continual restarts. To avoid starvation, the system can resort to locking for the transaction.

Name: ____________________________________

7. With authenticated messages, Byzantine consensus is possible with any number of faulty nodes. Argue informally why this is true.

   Each loyal node can detect when traitors send inconsistent values. If the general is loyal, then the loyal lieutenants will get the same values from the general and from the other lieutenants (since no traitor can sign a different message from the general). Thus, agreement is achieved. If the general is a traitor, then two lietenants can receive conflicting orders from the general. The lietenants exchange this information and apply the same "choice" function to this information, again coming to consensus.

Name: ____________________________________

8. Streaming audio or video services are usually implemented over UDP instead of TCP. Assuming that both impose roughly the same computational burden on the processor, explain why this the case?

   For these applications, it is more important to keep up than to receive every packet. If a packet is dropped, TCP halts the stream until it is retransmitted and received. For these applications, it is often better to keep going instead of holding things up to do retransmit; by the time the retransmitted packet is received, it is often stale and would just slow things down and load the apparently-busy network even more.

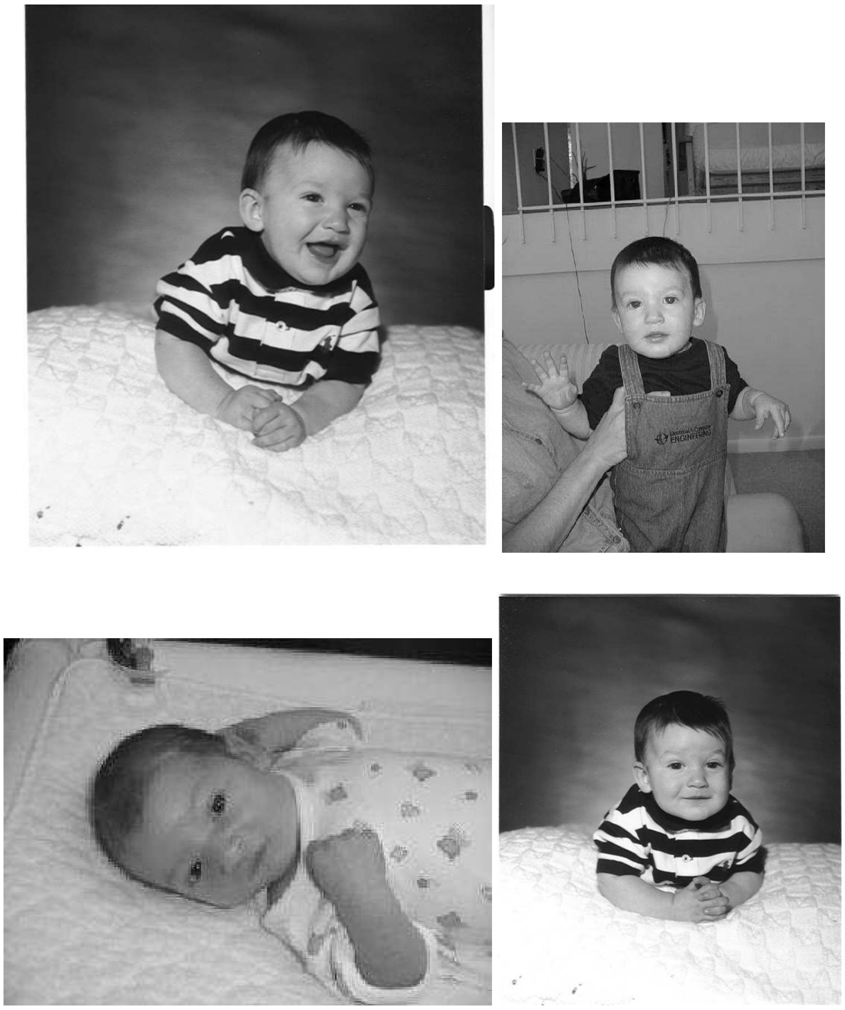
Name: ______________________________

- (Bonus, 1 point) Identify a typo in a math expression in the paper "Time, Clocks, and the Ordering of Events in a Distributed System." The typo we spotted is not in the appendix. (We're not cruel.)

  P. 563, 1st column, in the paragraph that begins "To avoid anomalous behavior..." The expression $C_i(t+\mu) - C_i(t) > (1-\kappa)\mu$ *should be* $C_i(t+\mu) - C_j(t) > (1-\kappa)\mu$.

Name: ______________________________

- (Bonus, 1 point) What is this cute baby's name?



Timmy.