6.033—Computer System Engineering | Handout #8—February 24, 1999

QUIZ #1. 1994

This quiz contains six problems whose relative weights are indicated below. Please place each answer on the page with the question and hand in this booklet at the end of the examination.

Put your name on this cover sheet AND at the bottom of each page of this booklet. The booklet may be separated for grading so that each problem can be evaluated by one grader. Be sure that you don't put part of the answer to a problem on the back of a sheet for another problem!

Some parts of some problems may be much harder than others. Read them all through first and attack them in the order that allows you to make the most progress. Write down any assumptions and show your work, or you risk losing partial credit. Be neat. If we can't figure out your answer we can't give you credit. On answers that involve numbers, be sure to specify clearly the units of your answer.

Circle your recitation instance:

1. 10:00—Tennenhouse/McDonald
2. 11:00—Tennenhouse/DeHon
3. 1:00—Kaashoek/Manning
4. 2:00—Kaashoek/Joseph
5. 11:00—Corbató/McDonald
8. 1:00—Corbató/Joseph
7. 11:00—Waldspurger/Manning
6. 12:00—Waldspurger/DeHon

For Official Use Only

Problem	Your Score	Maximum Score
1		10
2		5
3		25
4		25
5		25
6		10
Total		100

 Name ____________________

1. *(10 points)* According to the paper by Schroeder et al., "Except during reconfiguration, Autonet never discards packets." (p. 1319, col 2). Louis Reasoner takes this statement to mean that if he is careful never to launch a packet during a reconfiguration, he can dispense with end-to-end verifications and simply assume that all packets will be delivered. Set Louis straight by pointing out two substantially different situations that may occur and that require that the sender verify that the intended recipient got the packet, even in Autonet.

Situation 1.

Situation 2.

2. *(5 points)* The X Window system deals with both big-endian and little-endian clients by providing two different network ports. Big-endian clients send requests and data to one port, while little-endian clients send requests and data to the other. The server may, of course, be implemented on either a big-endian or a little-endian machine. This approach is unusual. Most network servers provide only one network port, and require that all data be presented at that port in little-endian form. Explain the advantage of the structure chosen by X, as compared with the usual structure.

3. *(25 points)* For each of the following statements, circle whether it is True or False, and briefly explain. An answer without an explanation will receive no credit. And if you give a good enough explanation you may be able to get credit even if you say False to a point we thought was True, or vice-versa. *But keep it brief!*

a. One advantage of a microkernel over a monolithic kernel is that it reduces the load on the Translation Lookaside Buffer (TLB), and thereby increases its hit rate and its consequent effect on performance. TRUE/FALSE

b. Ethernet is not very scalable because a centralized mechanism is used to control network contention. TRUE/FALSE

 Name ______________________________

c. The primary way that hierarchy reduces complexity is that it reduces the size of individual modules. TRUE/FALSE

d. The primary way that modularity reduces complexity is that it eliminates incommensurate scaling. TRUE/FALSE

e. As linear feature size goes down, the number of components that can be placed on a chip goes up with the inverse square. But overall computation capacity improves with the inverse cube of feature size. TRUE/FALSE

4. *(25 points)* Ben Bitdiddle has written a "style checker" intended to uncover writing problems in technical papers. The program picks up one sentence at a time, computes very hard for a while to parse it into nouns, verbs, etc., and then looks up the resulting pattern in an enormous database of linguistic rules. The database was so large that it was necessary to place it on a remote server and do the lookup with a remote procedure call.

Ben is distressed to find that the RPC's have such a long latency that his style checker runs much more slowly than he hoped. Since he is taking 6.033 he wonders if adding multiple threads to the client could speed up his program.

a. Ben's checker is running on a single-processor workstation. Explain how multiple client threads could reduce the time to analyze a technical paper.

b. Ben implements a multithreaded style checker, and runs a series of experiments with various numbers of threads. He finds that performance does indeed improve when he adds a second thread, and again when he adds a third. But he finds that as he adds more and more threads the additional performance improvement diminishes, and finally adding more threads leads to reduced performance. Give an explanation for this behavior.

c. Suggest a way of improving the style-checker's performance without introducing threads. (Ben is allowed to change only the client.)

5. *(25 points)* Suppose the longest packet you can transmit across the Internet can contain 480 bytes of useful data, you are using a lock-step end-to-end protocol, and you are sending data from Boston to California. You have measured the round-trip delay and found that it is about 100 milliseconds. (A lock-step protocol is one where you must wait for the recipient to send an acknowledgement of your previous packet before you can send the next packet.)

a. If there are no lost packets, estimate the maximum data rate you can achieve.

b. Unfortunately, 1% of the packets are getting lost. So you install a resend timer, set to 1000 ms. Estimate the data rate you now expect to achieve.

c. On Tuesdays the phone company routes some westward-bound packets via satellite link, and we notice that 50% of the round trips now take exactly 100 extra milliseconds. What effect does this delay have on the overall data rate when the resend timer is not in use. (Assume the network does not lose any packets.)

d. Ben turns on the resend timer, but since he hadn't heard about the satellite delays he sets it to 150 ms. What now is the data rate on Tuesdays?(Again, assume the network does not lose any packets.)

e. Usually, when discussing end-to-end data rate across a network, the first parameter one hears is the data rate of the slowest link in the network. Why wasn't that parameter needed to answer parts a-d of this question?

6. *(10 points)* OutTel corporation has been delivering j486 microprocessors to the computer industry for some time, and Metoo systems has decided to get into the act by building a microprocessor called the “clone486+”, which differs from the j486 by providing twice as many processor registers.

Metoo has simulated many programs and concluded that having twice as many processor registers reduces the number of loads and stores to memory by an average of 30%, and thus improves performance, assuming of course that all programs are recompiled to take advantage of the extra registers.

The most popular operating system for the j486 is microkernel—based. As suggested, the hoped-for performance improvement won’t be realized unless all applications, both client and server, and the microkernel itself are recompiled. Point out one other reason why Metoo may find the performance improvement to be less than their simulations of individual programs predicts. If you think of more than one reason, choose the one that is likely to reduce performance the most.

Quiz #1 1995

1. *(18 points)* For each of the following statements, circle whether it is True or False, and briefly explain. An answer without an explanation will receive no credit. And if you give a good enough explanation you may be able to get credit even if you say False to a point we thought was True, or vice-versa. *But keep it brief!*

1a. The paper “On building systems that will Fail”, written by Corbató, was on creating planned catastrophes. TRUE/FALSE

1b. If one created a graph of personal friendships, one would have a hierarchy. TRUE/FALSE

1c. One of the mistakes that the Therac-25 programmers made was to not reuse software from earlier radiation therapy machines. TRUE/FALSE

1d. Firewalls completely prevent any bug in a server from affecting clients. TRUE/FALSE

1e. Modularity and abstraction control propagation of effects. TRUE/FALSE

1f. Variable delay is an intrinsic problem of isochronous networks. TRUE/FALSE

2. Ben Bitdittle is designing a virtual memory system, VM, for his new microkernel system Pico. In particular, he is interested in what page size to use for his VM system and whether VM should be implemented inside the Pico kernel or outside.

2a. *(5 points)* Ben's computer has a disk, which can be accessed in 20 msec. Once accessed Ben's computer can blast data to the disk at 10 Mbyte/s. Give a formula for the time to write an n-byte page to disk, in milliseconds.

2b. *(5 points)*Using the formula compute how long it takes to write a 1-Kbyte page and how long it takes to write an 10-Kbyte page to disk.

The time for a 1-Kbyte page =

The time for a 10-Kbyte page =

2c. *(5 points)*Given the results of b) Ben decides to use a 10-Kbyte page size.Why might Ben be better off choosing a smaller page size?

2d. *(5 points)*Suppose that crossing the kernel boundary and delivering a message takes about 100 cycles on Ben's 200-Mhz computer. Should Ben handle page faults in the kernel or in a trusted server?

3. Two threads, A and B, execute a function with the following code (using the syntax from Birrell's thread paper), but always at different times (i.e., only one of the threads is in the function at a given time).

```
lock mutex_a do
    lock mutex_b do
    ...
    end
end

lock mutex_b do
    lock mutex_a do
    ...
    end
end
```

3a. *(9 points)*Assuming that no other code in the program ever acquires more than one lock at a time, can the program deadlock? (if yes, give an example, if not, argue why not)

3b. *(9 points)*Now, assume that the two threads *can* be in the code fragment above at the same time, can the program deadlock? (if yes, give an example, if not, argue why not)

4. Threads in a new multithreaded WWW browser periodically query a nearby WWW server to retrieve documents. On average, a browser's thread performs a query every N instructions. Each request to the server incurs an average latency of T milliseconds before the answer is received.

4a. *(5 points)*For $N = 2{,}000$ instructions and $T = 1$ msec, what is the smallest number of such threads that would be required to keep a single 100 MIPS processor (i.e., executing one hundred million instructions every second) 100% busy? Assume the context switch between threads is instantaneous, and the scheduler is optimal.

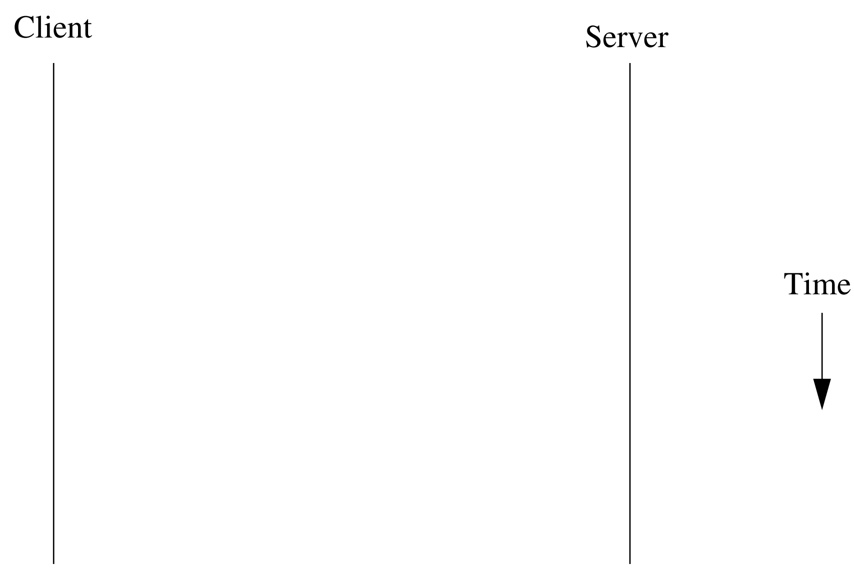
4b. *(5 points)*But context switches are not instantaneous. Assume that a context switch takes C instructions to perform. Recompute the answer to part a) for $C = 500$ instructions.

4c. *(5 points)*What property of the application threads might cause the answers of parts a) and b) to be incorrect? That is, why might more threads be required to keep the processor running the browser busy?

4d. *(5 points)*What property of the actual computer system might make the answers of a) and b) gross overestimates?

5. Birrell and Nelson's RPC, as described in their paper "Implementing Remote Procedure Calls", guarantees *at-most-once* semantics (*i.e.*, the RPC is carried out at most one time, but possibly none at all).

5a. *(8 points)*Give a scenario (*i.e.*, a sequence of events) in which the same request may arrive twice at the server?



5b. *(8 points)*Ben gets really excited about RPC and implements an RPC package himself. Like Birrell and Nelson, he starts a thread for each new request at the server. The thread performs the operaton specified in the request, sends a reply, and terminates. After measuring the RPC performance, Ben discovers that the creation of the thread and context switching to it is a major part of the cost of performing an RPC. After thinking about it, Ben comes up with a brilliant optimization: instead of creating a new thread for each new request, Ben's optimized RPC package executes the RPC on the current running thread. To his surprise the server deadlocks with this new optimization. What is happening? (Hint: think about locks that the current thread might be holding.)

5c. *(8 points)*Frustrated with optimizing his RPC package, Ben decides to use a brute force solution: he buys a much faster network. The latency for a sending packet from A to B on the new network is twice as low. Ben measures the performance of small RPCs (*i.e.,* each RPC contains only a couple of bytes) on the new network. To his surprise the performance is hardly better. What might be the reason that his RPCs are not twice as fast?

Quiz #1 1996

1. (*15 points*) We've discussed 4 principles for dealing with complexity: modularity, abstraction, hierarchy, and level definition. For each system organization below, choose *the most important* principle it uses or implements that makes it effective, and explain *briefly* why the system organization is an example of that principle. *An answer without explanation will receive no credit.* If you think two or more principles are equally important, choose one, and explain it well.

1a. *(3 points)* Protocol stack

1b. *(3 points)* Client/server

1c. *(3 points)* Virtual memory

1d. *(3 points)* RPC

1e. *(3 points)* Microkernel

2. (*18 points*) Ben Bitdiddle has written a program with 16 major modules of code. (Each module contains several procedures.) In the first implementation of his program, he finds that each module contains at least one call to every other module. Each module contains 100 lines of code.

2a. (*1 point*) How long is Ben's program in lines of code?

2b. (*4 points*) How many module interconnections are there in his implementation? (Each call from one module to another is an interconnection.)

Ben decides to change the implementation. Now there are 4 main modules, each containing 4 submodules in a one-level hierarchy. The 4 main modules each have calls to all the other main modules, and within each main module, the 4 submodules each have calls to one another. There are still 100 lines of code per submodule, but each main module needs 100 lines of management code.

2c. (*2 points*) How long is Ben's program now?

2d. (*5 points*) How many interconnections are there now? Include module-to-module and submodule-to-submodule interconnections.

2e. (*6 points*) Was using hierarchy a good decision? Why or why not?

3. (*21 points*) Mike R. Kernel is designing the Intel P97 computer system, which currently has one page table in hardware. The first tests with this design show excellent performance with one application, but with multiple applications, performance is horrible. Suggest 3 design changes to Mike's system that would improve performance with multiple applications, and explain your choices briefly. You cannot change processor speed, but any other aspect of the system, from disk to MMU design, is fair game.

4. (*22 points*) Alyssa P. Hacker is implementing a client/server spell-checker. The client scans an ASCII file, sending each word to the server in a separate message. The server checks each word against its database of correctly spelled words and returns a one-bit answer. The client displays the list of incorrectly spelled words.

4a. (*5 points*) The client's cost for preparing a message to be sent is 1 millisecond, regardless of length. The network latency is 10 milliseconds, and network bandwidth is infinite. The server can look up a word and determine whether or not it is mispelled in 100 microseconds. Since the server runs on a supercomputer, its cost for preparing a message to be sent is zero milliseconds. Both the client and server can receive packets with no overhead. How long will Alyssa's design take to spell check a 1,000 word file if she uses RPC for communication (ignore acknowldgements to requests and replies, and assume that packets are not lost or reordered)?

4b. (*5 points*) Alyssa does the same computations that you did and decides that the design is too slow. She decides to group several words into each request. If she packs 10 words in each request, how long will it take to spell check the same file?

4c. (*6 points*) Alyssa decides that grouping words still isn't fast enough, so she wants to know how long it would take if she used asynchronous messaging (with grouping words) instead of RPC. How long will it take to spell check the same file? (She is careful to resend packets that are not acknowledged but, for your calculations, you may assume that packets are not lost or reordered.)

4d. (*6 points*) Alyssa is so pleased with the performance of this last design that she decides to use it (without grouping) for a banking system. The server maintains a set of accounts and processes requests to debit and credit accounts (i.e., modify account balances). One day Alyssa deposits $10,000 and transfers it to Ben's account immediately afterwards. The transfer fails with a reply saying she is overdrawn. But when she checks her balance afterwards, the $10,000 is there! Draw a time diagram explaining these events.

5. (*24 points*) Louis P. Hacker-Reasoner (no relation) bought a used Therac-25 for $14.99 at a yard sale. After some slight modifications, he's hooked it up to the Ethernet as a computer-controllable turbo-toaster, which can toast one slice in under 2 milliseconds. (Louis likes his toast black.) Louis decides to use RPC to control the Toastac-25. Each toasting request starts a new thread on the server, which cooks the toast and returns an acknowledgement (or perhaps a helpful error code, such as "Malfunction 54"). Here's what the server thread looks like:

```
server thread:
    lock(message_buffer_mutex);
    decode message;
    lock(accelerator_mutex);
    unlock(message_buffer_mutex);
    cook toast;
    lock(message_buffer_mutex);
    put acknowledgement in message buffer;
    send message;
    unlock(accelerator_mutex);
    unlock(message_buffer_mutex);
    exit
```

5a. (*6 points*) To his surprise, the toast server stops cooking toast the first time it is heavily used! What has gone wrong? Refer specifically to the code above.

5b. (*6 points*) Once Louis fixes the multithreaded server, the Toastac gets more use than ever. However, when the Toastac has many simultaneous requests (i.e., there are many threads), he notices system performance degrading badly -- much more than he expected. Performance analysis shows that competition for locks is not the problem. What could be going wrong? Explain briefly.

5c. (*6 points*) An upgrade to a supercomputer fixes that problem, but it's too late -- Louis is obsessed with performance. He switches from RPC to an asynchronous protocol, which groups several requests into a single packet if they are made within 2 milliseconds of one another. On his network, which has a very high latency, he notices that this speeds up some workloads far more than others. Describe a workload that is sped up and a workload that is not sped up. (An example of a possible workload would be one request every 10 milliseconds.)

5d. (*6 points*) As a design engineering consultant, you are called in to critique Louis's decision to move from RPC to asynchronous client/server. How do you feel about his decision? Remember that the Toastac software sometimes fails with a "Malfunction 54" instead of toasting properly.

Quiz #1 1997

1. (*20 points*) Reading comprehension (multiple choice: select the one best answer to each question, and circle the number to its left).

a. "PC-losering" is

1) AI-lab slang for lowering oneself to the use of a UNIX system.

2) describing losing ideas using politically correct terminology.

3) backing up the saved program counter on a supervisor call.

4) a recurring OS problem in which the appropriate value to be restored into the PC is lost.

5) a term that has not appeared in 6.033 readings assigned so far this term.

b. Hierarchic systems

1) include artificial but not naturally-occuring systems.

2) exploit structures that incorporate little or no redundancy.

3) are exemplified by the structure of the World Wide Web.

4) are systems whose components are not themselves hierarchic.

5) exhibit a property called "near decomposability".

 Name ____________________

c. Therac-25 failures were difficult to reproduce in laboratory tests because they depended on

1) the technician's drunken stupor.

2) the absence of a potentiometer which was readily available in the laboratory.

3) the subject's actual cancer symptoms.

4) fast typing by the operator.

5) cosmic rays striking the field-light collimator.

d. The design of X Windows separates policy from mechanism by

1) the use of icons rather than names.

2) a policy of free distribution of source code.

3) support for big-endian as well as little-endian machines.

4) symmetric support for all platforms.

5) the demotion of the window manager to ordinary-program status.

e. Congestion is said to occur in a store-and-forward network when

1) communication stalls due to cycles in the flow-control dependencies.

2) the throughput demanded of a network link exceeds its capacity.

3) the volume of email received by each user exceeds the rate at which users can read email.

4) the buffering demanded of a network node exceeds its capacity.

5) the amount of space required to store routing tables at each node becomes burdensome.

2. (*80 points*) Quarria is a new country formed on a 1 Km rock island in the middle of the Pacific Ocean. The founders, recent 6.033 dropouts, have organized the Quarria Stock Market in order to get the economy rolling. The stock market is very simple, since there is only one stock to trade (that of the Quarria Rock Company). Moreover, due to local religious convictions, the price of the stock is always precisely the wind velocity at the highest point on the island.Rocky, Quarria's president, proposes that the stock market be entirely network based. He suggests running the stock market from a server machine, and requiring each investor to have a separate client machine which makes occasional requests to the server using a simple RPC protocol. The two requests Rocky proposes supporting are

Balance() - requests that the server return the cash balance of a client's account. This service is very fast, requiring a simple read of a memory location.

Trade(nshares) - requests that *nshares* be bought (assuming *nshares* is positive) or -nshares be sold (if *nshares* is negative) at the current market price. This service is potentially slow, since it potentially involves network traffic in order to locate a willing trade partner.

Quarria implements a simple RPC protocol in which a client sends the server a request packet of the format

```
struct Request {
   int Client;          /* Unique code for the client */
   int Opcode;          /* Code for operation requested */
   int Argument;        /* integer argument, if any */
   int Result;          /* integer return value, if any*/
};
```

and the server replies by sending back the same packet, with the Result field changed. We assume that link- and network-layer error checking detect and discard garbled packets, and that Quarria investors are scrupulously honest; thus any received packet was actually sent by some client (although sent packets might get lost).

a. Is this RPC design appropriate for a connectionless network model, or is a connection-based model assumed?

The client RPC stub blocks the client thread until a reply is received, but includes a timeout provision allowing any client RPC operation to return with a TIMEOUT error code if no response is heard from the server after Q seconds.

b. In a single sentence, give a reason for preferring returning a timeout error code over simply having the RPC operation block forever.

c. In a single sentence, give a reason for preferring returning a timeout error code over having the RPC stub transparently retransmit the packet.

d. Suppose you can bound the time taken for a request, including network and server time, at 3 seconds. What advantage is there to setting the timeout, Q, to 4 seconds instead of 2 seconds?

Unfortunately, no such bound exists for Quarria's network.

e. Q. What complication does client packet retransmission introduce into the RPC semantics, in the absence of a time bound?

 Name ______________________________

Rocky's initial implementation of the server is as follows:

```
#define N 1000                /* Total number of clients*/
int Cash[N],                  /* Cash balance of each client */
   Shares[N];                 /* Stock owned by each client*/

Server()
{
   struct Request *req;          /* Pointer to request packet*/
   while (1)   {                 /* loop forever...*/
         req = GetNextRequest();/* take next incoming request,*/
         if (req.Opcode==1)      /* ... and dispatch on opcode.*/
            Balance(req);        /* Request 1: return balance*/
         if (req.Opcode==2)      /* Request 2: buy/sell stock*/
Trade(req);
   }
}

/* Process a Balance request.. */
Balance(struct Request *req)
{
   int client=req->Client;          /* Get client number from request */
   req->Result = Cash[client];      /* Return his cash balance*/
   Reply(req);                      /* and send the reply.*/
}

/* Perform a trade: buy/sell Argument/-Argument shares of
* stock, and return the total number of shares owned after
* trade. */
Trade(struct Request *req)
{
   int client = req->client;     /* The client who is requesting*/
   int p=StockPrice();           /* Price, using network requests*/
   int nshares=req->Argument;    /* Number of shares to buy/sell*/
   int actual;
   /* See how many shares we can trade */
   actual = ConfirmTrade(req, p,nshares);

   Cash[client] = Cash[client]+p*actual;/* Update our records*/
   Shares[client] = Shares[client]+actual;
   req->Result = actual;
   Reply(req);
}
```

Note that ConfirmTrade uses network communication to check on available shares, executes the trade, and returns the number of shares that have actually been bought or sold.

Rocky tests this implementation on a single server machine by having clients scattered around the island sending Balance requests as fast as they can. He discovers that after some point adding more clients doesn't increase the throughput----the server throughput tops out at 1000 requests/second.

f. Rocky is concerned about performance, and hires you to recommend steps for improvement. Which, if any, of the following steps might significantly improve Rocky's measured 1000 **Balance** requests/sec?

Y___ N___: Use faster client machines.

Y___ N___: Use multiple client threads (each making Balance requests) on each client.

Y___ N___: Use a faster server machine.

Y___ N___: Use faster network technology.

Stone Galore, a local systems guru, has another suggestion to improve the performance generally. He proposes multithreading the server, replacing calls to service procedures like

```
   Balance(req);              /* Run Balance, to service request */
```

with

```
   NewThread(Balance,req);/* Fork thread to run Balance(req)*/
```

The NewThread primitive creates a new thread, runs the supplied procedure (in this case Balance) in that thread, and deactivates the thread on completion. Stone's thread implementation is preemptive.

Stone changes the appropriate three lines of the original code according to the above model, and re-tries the experiment with Balance() requests. He now measures a maximum server throughput of 500 requests/second.

g. Explain, in a single sentence, the performance degradation.

h. Is there an advantage to the use of threads in other requests? Explain, in a single sentence.

i. Select the best advice for Rocky regarding server threads:

A. Don't use threads; stick with your original design.

B. Don't use threads for Balance requests, but use them for other requests.

C. Continue using them for all requests; the benefits outweigh the costs.

Independently of your advice, Stone is determined to stick with the multithreaded implementation.

j. Should the code to Trade be changed to reflect the fact that it now operates in a multithreaded server environment? Explain, suggesting explicit changes as necessary.

k. What if the client is multithreaded and can have multiple request outstanding? Should the code to Trade be changed? Explain, suggesting explicit changes as necessary.

Rocky decides that multithreaded clients are complicated and abandons that idea.

Rocky left MIT just before the lecture on RPC, and isn't sure whether his server requires **at-most-once** RPC semantics.

l. Which of the requests require at-most-once RPC semantics? Explain, in a single sentence.

m. Suggest, in a sentence or two, how one might modify Rocky's implementation to guarantee at-most-once semantics. You can ignore the possibility of crashes, but should consider lost packets and retransmissions.

Quiz #1 1998

1. (*30 points*) ***Circle*** exactly one answer to the following questions (choose the best answer):

   a. Recitations in 6.033:

      1) Are not scheduled by the registrar, and require you to call Prof. Kaashoek at home to find a suitable time.
      2) Are an essential part of the 6.033 educational experience, and participation in recitation discussions contributes to your final grade.
      3) Are no longer held, and by not attending them you have cleverly saved a great deal of time waiting in empty rooms.
      4) Have been recently increased in length from 50 minutes to 3 hours due to popular demand.

   b. The errors in the Therac-25 system:

      1) Were in the terminal handler, and that is why the company suggested removing the terminal up-arrow key.
      2) Were dependent on timing.
      3) Were caused by writing the system in assembly language which caused compatibility problems when the code was ported from the Therac-20 system.
      4) Could have been fixed by correctly decoding the turntable position signals.

   c. The process primitives of the UNIX system:

      1) Create a new "forked" process and initialize its address space to zero.
      2) Do not permit processes to wait because processes are easily terminated and new ones started.
      3) Are used by the shell to create a single process for the command "ls | more" .
      4) Let newly created processes inherit pipes to facilitate inter-process communication.

d. The Eraser system:

1) Detects synchronization errors by computing the "happens before" relation.

2) During compilation places checks at lock and unlock commands to log what data is being protected by locks.

3) Considers a variable “shared-modified” as soon as the first store to it executes.

4) May miss synchronization errors.

e. In Birrell's RPC system:

1) A remote procedure call will always be executed completely, exactly once, or not at all.

2) A remote procedure call will always be executed at least once, and may be executed more than once.

3) A remote procedure call will always be executed at most once, which includes the chance that the call may not be executed at all or may be partially executed.

4) In the absence of failures it is impossible for the client to know if a remote procedure call was executed or not.

2. *(40 points)* Ben Bitdiddle is called in to consult for Microhard. Bill Doors, the CEO, has set up an application to control the Justice department in Washington, D.C. The client running on the TNT operating system makes RPC calls from Seattle to the server running in Washington, D.C. The server also runs on TNT (surprise!). Each RPC call instructs the Justice department on how to behave; the response acknowledges the request but contains no data (the Justice department always complies with requests from Microhard). Bill Doors, however, is unhappy with the number of requests that he can send to the Justice department. He therefore wants to improve TNT’s communication facilities.

Ben Bitdiddle observes that the Microhard application runs in a single thread and uses RPC. He also notices that the link between Seattle and Washington, D.C. is reliable. He then proposes that Microhard enhance TNT with a new communication primitive, pipe calls.

Like RPCs, pipe calls initiate remote computation on the server. Unlike RPCs, however, pipe calls return immediately to the caller and execute asynchronously on the server. TNT packs multiple pipe calls into request messages that are 1000 bytes long. TNT sends the request message to the server as soon as one of the following two conditions becomes true: 1) the message is full, or 2) the message contains at least 1 pipe call and it has been 1 second since the client last performed a pipe call. Pipe calls have no acknowledgements. Pipe calls are not synchronized with respect to RPC calls.

Ben quickly settles down to work and measures the network traffic between Seattle and Washington. Here is what he observes:

One-way Seattle to Washington, D.C. latency:	12.5×10^{-3} seconds
One-way Washington, D.C to Seattle latency:	12.5×10^{-3} seconds
Channel bandwidth in each direction:	1.5×10^{6} bits / second
RPC or Pipe data per call:	10 bytes
Network overhead per message	40 bytes
Size of RPC request message (per call)	50 bytes = 10 bytes data + 40 bytes overhead
Size of pipe request message:	1000 bytes (96 pipe calls per message)
Size of RPC reply message (no data):	50 bytes
Client computation time between requests:	100×10^{-6} seconds
Server computation time per request:	50×10^{-6} seconds

Note that the Microhard application is the only one sending packets on the link.

Please show your calculations and put your final answer in the box.

a. What is the transmission delay the client thread observes to transmit an RPC request message (the time required to transmit an entire message at the data rate of the link)?

Answer:

b. Assuming that only RPCs are used for remote requests, what is the maximum number of RPCs per second that will be executed by this application?

Answer:

c. Assuming that all RPC calls are changed to pipe calls, what is the maximum number of pipe calls per second that will be executed by this application?

Answer:

d. Assuming that every pipe call includes a serial number argument, and serial numbers increase by one with every pipe call, how could you know the last pipe call was executed? (*Circle the best answer*)

1) Ensure that serial numbers are synchronized to the time of day clock, and wait at the client until the time of the last serial number.
2) Call an RPC both before and after the pipe call, and wait for both calls to return.
3) Call an RPC passing as an argument the serial number that was sent on the last pipe call, and design the remote procedure called to not return until a pipe call with a given serial number had been processed.
4) Stop making pipe calls for twice the maximum network delay, and reset the serial number counter to zero.

 Name ____________________

3. *(30 points)* Ben Bitdiddle has just developed a $15 single chip Network Computer - NC - and plans to scoop Intel and create a revolution in computing. In the NC network system the network interface thread calls the procedure Packet_Arrived when a packet arrives. The procedure Wait_For_Packet can be called by a thread to wait for a packet.

   Part of the code in the NC is as follows:

```
1  VAR m: Thread.Mutex;
2  VAR packet_here: BOOLEAN;
3  VAR Packet_Present: Thread.Condition;
4
5  Packet_Arrived: PROCEDURE ();
6     BEGIN
7        packet_here := TRUE;
8        Thread.Broadcast(Packet_Arrived);
9     END;
10
11 Wait_For_Packet: PROCEDURE ();
12    BEGIN
13       LOCK m DO
14          WHILE NOT packet_here DO
15             Thread.Wait(m, Packet_Arrived);
16          END;
17       END;
18    END;
```

   a. It is possible that Wait_For_Packet will wait forever even if a packet arrives while it is spinning in the WHILE loop. Give an execution ordering of the above statements that would cause this problem. Your answer should be a simple list such as 1, 2, 3, 4.

   a. Write new version(s) of Packet_Arrived and/or Wait_For_Packet to fix this problem.

Answer: