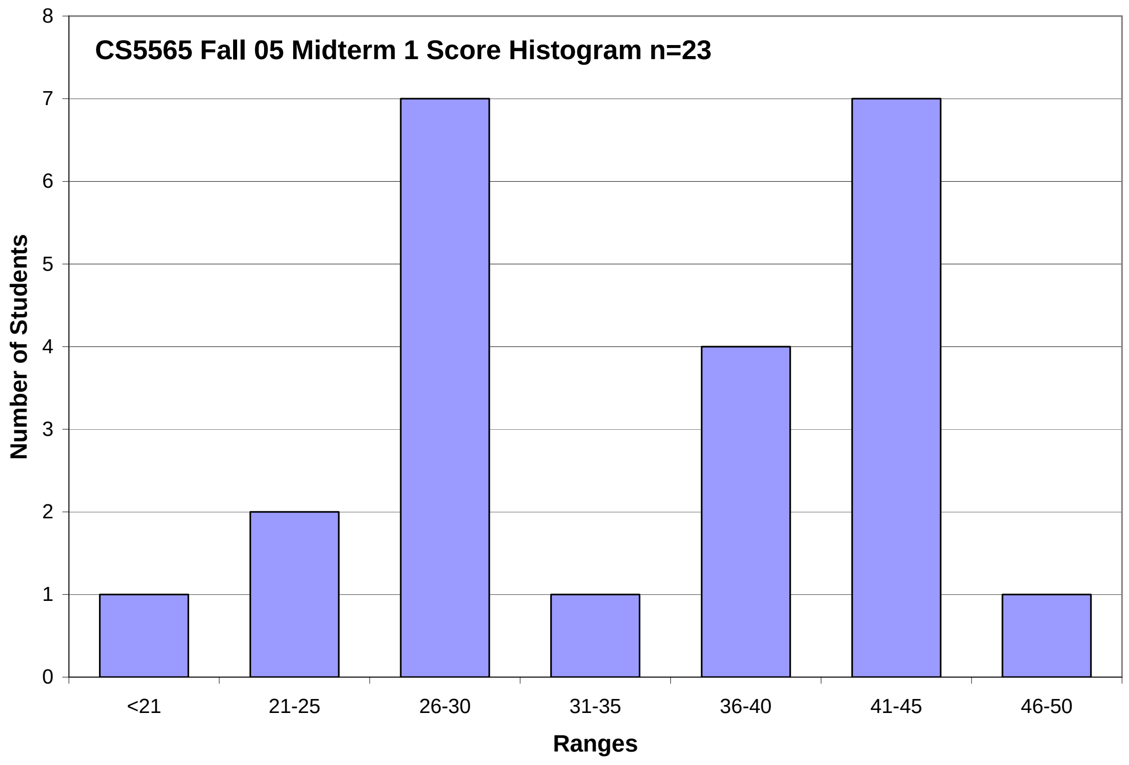
CS 5204 Midterm 1 Solutions

23 students took the midterm. The table below shows who graded which problem. Please contact the grader first with questions about your scores.

Problem	1	2	3	4	Total
Pts possible	10	16	12	12	50
Maximum	10	16	12	12	46
Minimum	2	9	0	0	19
Median	8	12	8	6	37
Average	7	12	8	7	34
Grader	Godmar	Wensi	Wensi	Godmar	

The chart below shows a histogram of the score distribution.



1. Threading vs Message Passing (10 pts)

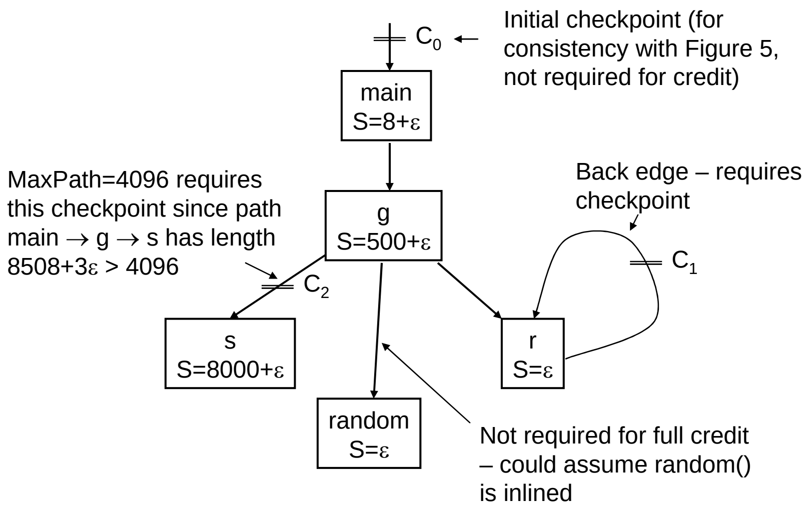
Lauer and Needham make the following claim in their 1978 paper on the duality of operating system structures: *We assert without proof that the facilities of each of our two canonical models can be made to execute as efficiently as the corresponding facilities of the other model.* Considering what you have learned from the SEDA and Capriccio papers, give one example of where L & N were correct, and one example where they were wrong.

a) (5 pts) Some examples of where it could be argued that Lauer & Needham were correct
   - o L&N state that it is possible to make the "basic actions of the two models behave identically with respect to the scheduling and dispatching of client processes." This is exactly the argument Capriccio makes when pointing out that their blocking graph was directly inspired by SEDA's stages and explicit queues.
   - o L&N expect the same performance from each model: the fact that Capriccio can roughly match the performance of SEDA appears to support this.

b) (5 pts) Some examples of where it could be argued that Lauer & Needham were wrong
   - o L&N say that "sending a message" has the same complexity as "forking to an entry" procedure. While true in theory, no threading system comes even close in thread creation times to low overhead in passing a message, in particular in shared-address space server systems such as the one considered in SEDA/Capriccio.
   - o L&N say that "virtual memory and paging and swapping can even be used with equal effectiveness" in either model. Generally, threading systems impose much larger pressure on the VM system that message-based systems: even a system such as Capriccio, which goes to great length to minimize the virtual address space overhead of thread stacks, will suffer from larger virtual address space consumption than a message-based system, all other things being equal.

2. Capriccio (16 pts)

a)
b)
c) (4 pts each) Draw the callgraph for this program, starting with function *main* at the root. Label each function with the approximate size of its stack frame, similar to Figure 5 in the paper. Suppose we set MaxPath = 4096 and MinChunk = 1024.


C_0
Initial checkpoint (for consistency with Figure 5, not required for credit)
main
S=8+ε
MaxPath=4096 requires this checkpoint since path main → g → s has length 8508+3ε > 4096
g
S=500+ε
Back edge – requires checkpoint
C_1
C_2
s
S=8000+ε
r
S=ε
random
S=ε
Not required for full credit – could assume random() is inlined

Note that MinChunk does not affect where to put in checkpoints. It only affects how often those checkpoints will have to allocate and link new stack chunks. (For instance, with MinChunk = 1k, about every (1,000/ε)th call of checkpoint C_1 would result in a new chunk being allocated.)

d) (4 pts) Futexes make only sense in a kernel-level implementation – Capriccio is a user-level threading implementation, where threads cannot be preempted outside of specific scheduling points. Implementing locks in Capriccio does not require atomic instructions.

3. VTRR (12 pts)

Your friend has implemented VTRR in a simulator and asks you to help him test it. His first test involves the following 4 tasks:
A, B, C, D with shares $\{S_A = 10, S_B = 5, S_C = 2, S_D = 1\}$.

a) (4 pts)

A B C C A B A B C A B A B A B A B B

Short answer: VTRR achieves perfect proportional fairness at the end of a scheduling period, which is 18 time units (tu). This schedule clearly does not, as can be seen easily.
After 18 time units (tu), A should have received 10 tu, B 5 tu, C 2 tu and D 1 tu. In the schedule shown, D didn't get any, (and C got 3, A got 7, B got 8) which can't be correct.

b) (4 pts)

A B C D A B **A C** B A B A A A B A A A

Although this schedule shows perfect proportional fairness, it is also not correct under VTRR, because VTRR schedules in decreasing share order, here: A B C D. It starts with A, and either goes down the list one by one or starts over with A. It will never schedule C after A without having scheduled B in between.

c) (4 pts)

A B C D A B A B C A B A A A B A A A

The service error is defined as $E_A(t_1,t_2) = W_A(t_1,t_2) - (t_2 - t_1)\frac{S_A}{\sum_i S_i}$

A has received 3 tu of work after 9 tu, versus the 5 it should have, so its service

error is -2: $E_A(0,9) = 3 - (9)\frac{10}{18} = -2$

4. Leases (12 pts)

a) (6 pts) The Leases paper assumes that write-through caches are being used; they claim that *extending the mechanism to support non-write-through caches is straightforward.* Explain what would need to be done in order to use a write-back cache with leases and what additional problem(s) this could cause!

(3 pts) To support write-back caches, modified data needs to be written back when a write lease is invalidated or expires.
(3 pts) Applications will then have to recover from lost writes because these writes may fail. The write must then be either retried or if that fails, such failure must be communicated back to the client that issued the write, which however may have continued operating assuming that the data has been written. See page 203, left column, paragraph 2.

b) (6 pts) When discussing options for Lease Management, the paper states that *multiple files per lease can also result in a form of false sharing*. Explain what is meant by that!

If multiple files share a lease, then an update to any single one of them will require invalidating the lease for all of them. If host A updates file *F* in the set, and host B updates a different file *G* in the set of files covered by a shared lease, it will falsely appear as though A and B are sharing files, which in truth they do not.
This can either lead to increased lease invalidation traffic, and it could delay writes if say client A decides to not give up its lease before it expires.