CS452 aka ‘The Trains Course’ – Final Examination

Winter, 2016

Instructor: Bill Cowan

Starting time: 12.30, 19 April, 2016.
Ending time: 15.00, 20 April, 2016.

Please e-mail your answers to the instructor at wmcowan@cgl.uwaterloo.ca before the ending time.

Things to remember when reading and writing this examination.

Make sure that you understand what is written below. Term after term more marks are lost to not following instructions than to any other cause.

0. Do Question 1 and any one (1) of the following non-bonus questions. Doing the bonus questions is almost never a good use of your time: marks are added more quickly as you respond to other questions. It does, however, give you an opportunity to put the marker in a good mood, by your interest in something that obviously interests him, and by the scope for making him laugh.

1. You must work independently.

2. You may use any source of information you want on this examination. Any information from sources you consult MUST be referenced. (Your memory, course notes and lectures are the only exceptions.)

3. There may be places in the examination where you can say something clever by making assumptions. Do so, but explain your assumptions and how they are related to what you are saying.

4. I take it for granted that you know CS452 to be the REAL-TIME course. Pretty well every question requires an answer that takes time into account even though time is rarely mentioned explicitly in question descriptions.

5. Remember that in real-time programming we care most about deadlines, the time between an event occurring in the outside world and the time when the correct response must be completed. In real-time systems a too soon response is equally bad as a too late one.

6. Any time you can provide a quantitative estimate of performance you are gaining more marks! All numbers should have units.

7. Have a dictionary at hand when reading and look up any unfamiliar words.

8. Submit your answers by e-mailing them to wmcowan@cgl.uwaterloo.ca.

9. All questions are open-ended: most marks come from going beyond simple answers; explicit instructions are intended as prompts to get you started in the right direction. You gain marks for the thoughts that you contribute to your answers. Write other people’s thoughts, including mine, only to the extent that I need them to understand your answer.

10. In the examination ‘your kernel’ means the kernel you actually created, not the kernel you wish you had created. Similarly, ‘your OS’ it means your kernel plus the other tasks (couriers, notifiers, servers) that provide the kernel API. On the other hand, ‘your train application’ means the application you planned to create in its final form.
Question 1. A Novel I/O Device

Note. You must do this question. In addition do one of questions 2–4. Do bonus questions only after doing the best you can on the other questions.

Note. As you read through the exam questions you will, no doubt, find spelling and grammar mistakes. They are inevitable when typing the order of four thousand words. If you find them early in the exam e-mail them to me: I will correct the on-line copy and send out errata.

Now the question itself starts. Developing your kernel included implementing servers (and their attendant tasks) for only two types of device, a timer and a UART.

Part 1.1. Timer and UART

Both devices are simple, which eases implementation; both are low performance, which eases debugging.

i. What makes them simple compared to more modern devices such as USB? (This question has a variety of answers, philosophical, financial (marketing), technical, sociological, educational, etc. The more you put into your answer, and the more that you show their interconnexions, the more fun it is to read your exam.)

ii. Why does simplicity make implementation easier?

iii. Why does low performance make debugging easier? (This question and the one above do NOT have the same answer.)

Suppose we were to add a device with the following properties:

• a large (hundreds of KBtyes) buffer located in the I/O part of the address space,
• a receive interrupt which is asserted when a block of data has been received and is in the buffer,
• a transmit interrupt, which is asserted when transmission of a block of data from the buffer is complete, and
• registers, one each for receiving and transmitting, containing
  • the starting buffer address of the current data block,
  • the size of the current data block, and
  • the destination or source address of the current data block.

You may recognize this as a very schematic ethernet adapter, receiving and transmitting one packet at a time. In this imaginary device a hardware controller provides simplicity to the programmer’s view. (As so often occurs, the real world is exactly the reverse of the imagined one: in your laptop a fairly substantial CPU provides an ethernet – or wireless connection – having context too big for my simple mind even to consider!)

Part 1.2. Kernel redesign

This question allows you to show me that you understood the general architecture of the I/O services provided by your kernel. The description above is
deliberatively sparse. In your answer tell me what you think is unusual and interesting. When you make assumptions tell me what they are.

i. Design a kernel interface for the AwaitEvent primitive and describe any special treatment its implementation would require.

ii. Design a Notifier to wait on the AwaitEvent primitive. Describe any special features that its implementation requires.

iii. Design a Server to distribute input to clients and to transmit output from clients.

iv. Describe any auxiliary tasks that solve problems specific to this device.

Part 1.3. Application redesign

The device described above is not easy to fit into a real time application, and so far I don’t expect your answer to have worried too much about real time. Now is the time for that.

i. The most common application you know for devices like this one is the movement of large amounts of data long distances in and out of your local machine. Neither the operating system driver in your local machine nor the network architecture of your ISP is, however, is not compatible with doing so in real-time. (A quick listen to the sound quality of Skype or VoIP should be enough to convince you.)

I assume the components you described above would have in providing real-time I/O performance. If not, patent! Describe the performance limitations you would expect to occur and how you might alleviate them.

ii. Suppose you are using this device as part of a collection of hardware, all of which is dedicated to your real-time application. That is, assume that you are implementing both the data that comes in through the device as well as the data that goes out through the device. You can then establish conventions that assure real-time performance.

Describe conventions you could establish to provide real time behaviour, and estimate the response times you would expect from obeying them.

Part 1.4. Special hardware

There are a variety of ways in which hardware assists in handling big buffers by minimizing the load on interfaces to high speed buses or even to the bus itself. Below are a few possibilities; there are many more.

- Direct Memory Access (DMA). Special hardware that transfers memory contents in bulk without CPU intervention.
- Memory mapping. Mapping the address space of I/O data buffers directly into the address space of the device drivers, or even user space.
- Special bus operations that increase the speed of data transfer.
- Special bus architectures that create extra data paths.

Look up one or two of these, or other, possibilities and explain how you might use it to improve the real-time performance of your design.
Question 2. Calibration accuracy.

Note. Do one of questions 2–4.

In class we talked about the unavoidable error that occurs when we get sensor data from the train controller: we don’t know exactly when the sensor was read, but only a finite time interval in which it was read. That is, there are intervals of time \([t_n, t_{n+1})\) and when you get a sensor reading you know only that the time of the reading is \(t_n \leq t < t_{n+1}\). This question is about how you (dealt/would deal) with this problem when calibrating velocity.

Part 2.1. How big is the interval?

i. What bounds the interval? A specific event occurs in the train set at \(t_{n-1}, t_n, t_{n+1}\), etc. that separates one interval from the next. What is this event? Why is it significant?

ii. How long was the interval in your project? How (did/could) you change your implementation to shorten the interval?

iii. How far does a train travel during your interval? Was this good enough for your project? Explain your reasoning.

Part 2.2. Sequence of events

There is a sequence of data movements and transformations that are repeated during each interval.

i. Write a list, in order, of the the events in a single interval.

ii. Estimate the amount of time between each two events, showing how you made each estimate. The sum of the estimates should come close to the length of the interval you gave above.

iii. When improving the performance of an application you seek, either
   • code that is easy to improve, or
   • code that executes a lot.

   Identify one example of each and explain either how you might improve its performance, or why your code is so good that its performance cannot be improved.

Part 2.3. Systematic error

We can break error up into two parts:
   • systematic error, which is the same for every measurement, and
   • random error, which varies, measurement to measurement.

i. We minimize random error by convention. Systematic error is defined such that the expectation of the random error (the average of the random error over many measurements) is zero. Draw one interval and indicate where in the interval the random error is defined to be zero. Explain.

ii. In your project you (used/should have used) differentiation to remove systematic error from important measurements. Explain how you (did/would have done) it when calibrating velocity.
Part 2.4. Random error

You can find the answers to these questions in pretty well any elementary statistics text, but you can probably figure out the answers faster than looking them up.

i. Draw the probability distribution function for the random error in the measurement discussed in this question. Label the diagram. Calculate the value of the mean and the standard distribution.

ii. The probability distribution function for the average of two measurements is different than the one for a single measurement. Draw it on a labelled diagram; calculate the mean and standard deviation. This is a simple example of what general principle for dealing with random error.

iii. In your project you certainly experienced a few failures of the general principle. Explain at least one.
Question 3. Fluidity of a Railroad System

Note. Do one of questions 2–4.

When railroaders talk about the operational health of a railroad fluidity is usually a central concept. They see trains moving on the tracks as fluid moving in pipes or water moving in rivers. Trains flow into a classification yard, where they are broken down into parts by destination and reassembled into new trains with all cars having the same destination, and then flow out again. The yard is like a buffer; as long as there is enough space in the buffer trains move smoothly through it. But if the buffer fills, trains back up on the incoming track and everything stops because the cars you need to complete leaving trains can’t get into the yard: the system has frozen.

Part 3.1. Theoretical limits

The track configuration in the trains lab was deliberately created to challenge fluidity. Real railroads have kilometre after kilometre of track quite free of turn-outs, with occasional interlocks, where trains can move from the north to the south track, or vice versa, which is not much of a challenge for trains students.

Most projects reserve track breadth first, reserving a fan-shaped collection of track along all the routes that might be occupied in the future. We can make a little model of this system for you to explore. We assume the following.

• Reservable sections of track (edges) are bounded by sensors, merges and branches (vertices). Merges and branches are turn-outs, the first seen as two tracks coming together into one, the second as one track splitting into two.

• Merges can be counted as sensors: that is, loops are removed from the tree by duplicating edges. They can also be ignored.

• Branches are a fraction $p$ of section divisions.

We now describe a process by which bigger reservations are constructed systematically from smaller reservations, creating the reservation fan.

1. A train sits on an edge, between two vertices. That edge is reserved, owned by the train sitting on it.

2. Forward, as seen by the train, points to one of the vertices, the leading vertex.

3. When the train wishes to move it asks for edges beyond the leading vertex. If the leading vertex is a branch it asks for two additional edges, otherwise one.

4. After the reservation has been granted either three edges have been reserved, with two leading vertices, or two edges have been reserved, with one leading vertex. The former happens with probability $p$, the latter with probability $1 - p$. Thus, the average number of reserved edges is $3p + 2(1 - p) = 2 + p$, and the average number of leading vertices is $2p + (1 - p) = 1 + p$.

5. As long as the train needs more free distance it continues to reserve track. Suppose that when the fan is $n$ deep the average number of leading vertices
is $l(n)$ and the average number of reserved sections is $s(n)$. Then

$$l(n+1) = 2pl(n) + (1-p)l(n) = (1+p)l(n) \quad (4.1)$$

and

$$r(n+1) = r(n) + 2pl(n) + (1-p)l(n) = r(n) + (1+p)l(n). \quad (4.2)$$

i. Check that (4.1) and (4.2) are correct for $l(1)$ and $r(1)$, noticing that $l(0) = r(0) = 1$.

ii. Solve the recursions, showing your work.

(I think that the answer is $l(n) = (1+p)^n$ and $r(n) = (1 - (1+p)^n+1/p$.

At least in the limit as $p$ goes to zero $r(n) = n + 1$ is correct.)

These numbers grow fast with $n$, but not so fast as to be unmanageable on a track the size of ours.

Part 3.2. The quality of the model

We make theoretical models not because they are ‘correct’, but because they give an understanding of ideal cases. We extend this understanding to more complex cases by faith and analogy. When we make a model we must estimate how it falls short of the reality with which we must work in practice. The current model is both overly optimistic and overly pessimistic.

In considering the shortcomings of a model always recall what we are trying to understand. In this case, we are trying to understand why and when reservation start to interfere with one another in the specific track configurations in which we will demo our projects.

i. **Overly optimistic.** The model is overly optimistic in neglecting the details of track geometry. Some sections of track are more likely than others to produce reservation conflicts. Explain. (Reservation conflicts are the primary reason why trains must wait, which limits fluidity.)

ii. **Overly pessimistic.** The model is overly pessimistic in neglecting merges, which reduce the amount of track reserved. Explain.

Part 3.3. Using the model

The model abstracts away all details of how sections of track are attached to one another, the topology of the track. It does, however, retain some differences of how sections of track are joined together, *viz* how branches create fans.

In addition, the model assumes that several sections of track are required for a reservation: that is, a train requires a distance to stop that is bigger than the average section length. This assumption is valid. However, it measures the size of a reservation by the number of sections it contains, abstracting away the large variability of section length.

i. How many section dividers of what kind are there on your favourite track? How many sections are there? What’s an appropriate value to use for $p$? (Please don’t send me e-mail asking me the right way to count: in this
question I want to know if you can translate vague descriptions of counting into pointing at actual things, saying ‘One, two, three,...’

ii. What is your estimate for \( p \)? Explain.

iii. What is your estimate for the number of sections required to stop a train at the speeds and on the sections of track that you used in your project? Explain.

iv. What is a typical reservation size, in sections, required by a train in your project?

v. How many trains do you think you could keep moving on the track in the lab? *This number is not simply the result of algebraic manipulation of the numbers you put down above. First of all, it is likely to be a range, not a single number. Second, it will be the result of thinking and evaluating as well as of calculating. Tell me what you are thinking in your answer.*

The purpose of making models like the one in this question is to limit ignorance, which is often worse than stupidity. (What’s worse than stupidity is not ‘limiting ignorance’ but ignorance itself.) Computer graphics students from time to time draw at Alpha Centauri, which is in the logical display space, but not the physical one, then wonder for days why they can’t see their drawing. The trains course equivalent is carefully designing a project that will be able to run twenty-five trains simultaneously. When models like this one prevent errors like those ones they are worth their weight in gold. (Which is?)
Question 4. Relativistic Reservations

Note. Do one of questions 2–4.

I suppose AM radio stations still have programs to which you can phone in and dedicate songs. If songs, why not examination questions? So..., this question is dedicated to Simon Parent, a previous student and long-time TA of CS452, who was viscerally unhappy with the reservation system I taught when he was a student.

Here is a different method for preventing the collision of trains. It’s based on the following ideas.

- There are dangerous places to stop on the track, such as on top of switches, or on curves where you can’t get going again.
- The rest of the track is safe.
- A reservation (dark grey) is a region of space-time, as shown in the image (which isn’t there yet), where the train starts stationary in one safe region (light grey), and ends stationary in another safe area.
- Another reservation (black) allows the train to remain in a safe place for a finite amount of time.
- At the end of that time the train receives another reservation to move to a third safe area.
- Notice that the reservations expand as they move from one safe area to the next. The expansion allows for unpredictable variations in velocity. Because these reservations use space-time pictures like the one on the right, which also occur frequently in Einstein’s theory of special relativity, we call them relativistic reservations (RR).

Part 4.1. Shape of an RR

i. An RR has a minimum width in space. Why? How wide must it be?
ii. The two RRs shown above have straight sides. Is this realistic? Why or why not?

Part 4.2. RR rules

In class I described a reservation system, basing it on a set of rules distributed across several tasks. The rules guaranteed that there would never be a collision.

i. Give a set of rules for RR systems, and explain why it avoids collisions.
ii. Describe how to distribute responsibility so that rules are followed, using the task structure of your train application as a model.

Part 4.3. Sharpening up RRs

As the train travels it passes sensors.

i. How might you use sensor feedback to improve an RR as the train travels?

ii. Sharpening up reservations means that they are temporally dynamic. Explain how the reservations you used changed over time. Draw a changing reservation as drawn above.

iii. Draw on a diagram like the one above showing your reservations changing over time.

iv. Another way to have changing reservations is to have two time dimensions. Try drawing a changing RR in a 3D diagram with two temporal dimensions.

Part 4.4. Failures of train control

As all of you know from the project demos trains that refuse to move as directed are common in the real world.

i. Explain how you would modify RRs to make them more resistant to trains that stall or travel unexpectedly slowly.

ii. Explain how you would modify RRs to make them more resistant to trains that go the ‘wrong way’ through a turn-out.
Bonus Question 1. Course Upgrades

It probably occurred to you while doing question 1 that I was talking a little about the trains course of the future. The COM port is a thing of the past in the home computer world: modern computer interfaces to model train sets communicate with computers using USB (blah!), ethernet, or even CaNet. If we succeed in getting the school to buy us a new train set, I can still use 3-wire RS-232 and a UART for communicating with the terminal, but you will have to implement another server for communicating with the train. I am most optimistic about handling ethernet using programmed I/O. If that works the students will have to do something like Question 1 as the final part of the kernel. This question asks you for your comments, particularly from a wider point of view.

i. This question asks you for your comments, particularly from a wider point of view. As I have often said the course should be difficult enough, but not too difficult. Adding another device to the kernel increases the difficulty.

ii. Along with another device there will be pressure to move to a multicore SoC. This is actually a lot easier than it sounds: the kernel you made generalizes very naturally into the multiprocessor world.

iii. Faster I/O to the train probably goes along with faster communication within the train set itself. The result is almost certainly a substantial diminution in the polling time, which makes the project significantly easier.

iv. Many think it would be natural to introduce video cameras into the course, either the driver’s eye view or the overhead plan view. This is not desirable in my view: it either introduces an unreasonably large amount of computer vision programming, which is orthogonal to the course or it introduces a large-scale library like OpenCV which probably makes locating the trains trivial while removing the you-did-it-all-yourself element of the course.

According to student comments from the last five years the overall balance of what I have laid out above is right: kernel a little more difficult, project a little easier. (It also allows students to put in more time early in the term and less late in the term, which I think is good.) But, including iii.), in my opinion, goes too far in weakening the system aspect of the course, however it’s done.
Bonus Question 2. Snow Scenes

I know that you are probably fed up with snow, but..., when I talked in class about projects I gave you a link to a video showing what I said I would like to see as part of a project. In what province was the video made? How might you arrange to have what’s in the video as part of a project?