CS452 Train Project

Christoph Ulshoefer (20751216) and Thomas Broadley (20522223)

April 6, 2018

1 About this report

The final report includes more insight into why we have implemented features in the way we have, as well as why we didn't implement certain features. We felt that we didn't discuss these topics enough in earlier reports, so we are including the discussion now.

2 Running the program

```
load -b 0x00218000 -h 10.15.167.5 "ARM/csulshoe/project.elf"
go
```

If you're feeling fancy and want to run the latest kernel that we ran:

```
l  ARM/csulshoe/m;g
```

The ELF file for this version is stored at `/u/cs452/tftp/ARM/csulshoe/project.elf` in the CS Student Computing Environment. The source code can be found at https://git.uwaterloo.ca/csulshoe/cs452/tree/master.

2.a Commands

Command	Description
`tr <train_number> <train_speed>`	Set a train's speed.
`rv <train_number>`	Reverse a train.
`sw <turnout_number> <turnout_direction>`	Switch a turnout.
`sd <train_number> <train_speed>`	Run automated stopping distance calibration.
`v <train_number>`	Run automated velocity calibration.
`loop <train_number> <train_speed>`	Enter a loop around the track.
`r <train_number> <sensor> <offset>`	Stop at a location.
`t2start <train_number>`	Start random repeated routing.
`t2start <train_number>`	Stop random repeated routing.
`group <name> <train_number> [train_numbers]`	Link together a group of trains. Single train commands for grouped trains are not supported.
`ungroup <name>`	Go back to individual train control.
`trg <name> <speed>`	Set a group's speed.
`rvg <name>`	Reverse a group.
`set`	Set parameter. Useful parameters: `trains`: active trains. `track`: 0 for track A, 1 for track B. `spacing`: space between each pair of trains in a group. See `/usr/train/parameters.c` for more.
`go`	Start the train controller.
`stop`	Stop the train controller.
`q`	Quit.

3 Vision

For our final project, we wanted to implement a system for creating, dissolving, and moving groups of trains as single units[1]. We also wanted to integrate reservations and routing with reversal into this system. Our goal was to have two or more groups of trains simultaneously routing to random locations on the track.

4 Features and implementation

Two trains with a 10-centimeter spacing can run as a group for over 10 minutes. Three trains with a 20-centimeter spacing can run for several minutes. Groups of trains can also be reversed. It is also possible for a single train to route to random destinations on sidings while a group of trains runs in a loop.

4.a Routing one train in the presence of a group

We allow one independent-minded, free-wheeling, anarchist train to route in parallel to a group. This train will constantly check whether it will run into the group (by projecting ahead its movements, and the movements of the other trains), and pick the highest speed that guarantees a certain distance from the other trains, for the foreseeable future (i.e. 500 ticks into the future or 140 centimeters, whichever comes first).

[1]Similar to how trucks platoon (they are even called "road trains"). https://youtu.be/lx9EFJ6qgZc

If lucky, you can encounter the independent train sneaking ("interleaving") inbetween the trains in a group, if the spacing in the group is big.

4.b Handling broken turnouts

To handle broken turnouts, we treat both branches of the turnout as if they connect to the same node in the track graph. This solution was easy to implement and allows trains to One downside is that we must recompile the project to change the list of broken switches. If we had more time, we would consider implementing automatic detection of broken turnouts.

4.c Sunsetting tasks

We came across a couple of situation where we wanted to kill a task and its children. Rather than changing our `Kill` syscall, we decided to create a message type `MESSAGE_SUNSET`. When a message of this type is sent to a task that is prepared to receive it, the task is expected to kill or sunset its children, then exit.

One disadvantage of this approach is that only servers can be sunset, whereas changing the `Kill` syscall would apply this behaviour to all tasks. Luckily, we only needed to apply this behaviour to the train conductor and multi-train conductor tasks, which are both servers.

5 Task structure

Note that the graph above is a DAG except for `Send()`s between the command dispatcher and the train conductors. These `Send()`s happen in a controlled fashion; in particular, the command dispatcher only ever sends to a train conductor if it is ready for a new train command and is thus send-blocked. In this way, we hope to make deadlock less likely.

5.a Train coordinate courier

We previously implemented the train conductor as a courier that regularly queried various servers to decide how to drive the train. As a result of this design, we wrote a lot of duplicated code and found it difficult to add new features to the conductor.

We decided for our final project to break the train conductor into two tasks: a server (the conductor) and a courier. The conductor calculates locations at which it needs to change the train's speed, reverse it, or flip a switch, then tells the courier to wake it up once the train has reached that location. The courier regularly polls the train coordinates server and sends a message to the conductor when the train has reached one of the specified locations. We found that the separation of location-tracking logic and train command generation into two tasks made it easier to implement new train behaviour.

5.b Multi-train conductor

To drive a group of trains as a single unit, we decided to implement a multi-train conductor and a corresponding multi-train coordinates courier. We thought this would be easier to implement than

The multi-train conductor is responsible for driving an entire group of trains. It is created when a `group` command is issued. It then creates a multi-train courier. This courier, in addition to waking the conductor when the first train in the group has reached given points on the track,

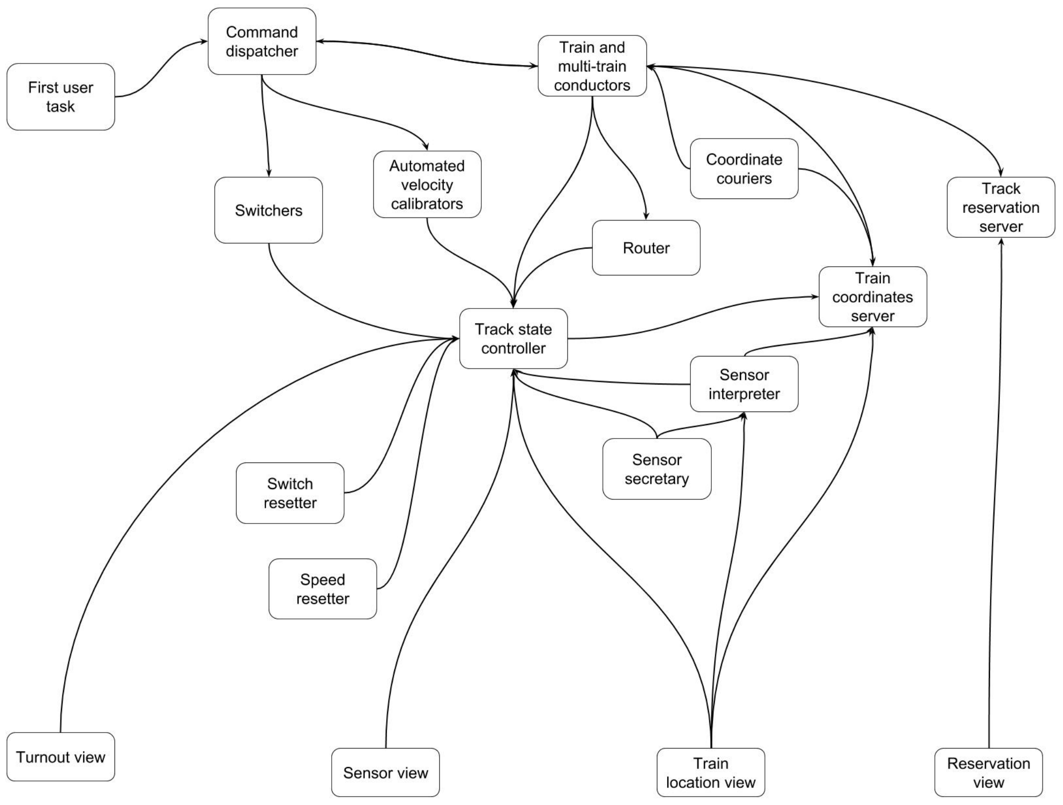

Figure 1: Diagram of our final project task structure, with arrows indicating `Send` calls between tasks. The clock and UART servers are not included in this diagram because many tasks send to them and they do not send to any task. Notifiers are not included because each only sends to its corresponding server.

sends a message when the spacing between two consecutive trains in the group is out of a tolerated range. In this way, the multi-train conductor makes sure that all the trains in a group stay at a given distance from each other.

5.b.1 Speed modulation: an alternative solution

We also considered creating a task that would rapidly change a train's speed between two integer speeds, with the goal of moving the train at an average velocity between the velocities of the two speeds. We decided to implement spacing control as described above because we believed it would be simpler. However, in hindsight, we think that the modulator approach might have resulted in less variation in spacing. We could have programmed the modulator to choose an average speed that would keep it near the optimal spacing, rather than only correcting the spacing once it is outside the acceptable spacing range.

6 Unimplemented optimizations

We could not convince ourselves that optimizing our performance is necessary. Although our idle time with `-O2` enabled is 60% on average, this did not seem to degrade our project's performance. If it had been necessary, we did have several ideas for optimizations to make:

- Change the CPU speed[2]
- Optimize our `memcpy` implementation
- Use smaller messages in most cases (since every message sent is over 200 bytes)
- Run selected `-O3` flags (since building with `-O3` breaks the kernel)
- Use a linked list instead of a min-heap for the clock server queue
- Reduce kernel data structure size
- Fix `-O3`
- Apply link-time optimization
- Use vectored interrupts

7 File and repository hashes

The last commit on the branch `master` has SHA `TODO`.

7.a MD5 hashes

`TODO`

[2]https://wiki.embeddedarm.com/wiki/TS-7200#Setting_the_CPU_Clock_Speed