CS 452 – Train Control 1

March 7, 2017

Zhengkun Chen

Student ID: 20557054

Xiwen Liu

Student ID: 20468618

Operating Instructions

Full pathname to executable: /u/cs452/tftp/ARM/z283chen/kernel.elf

To load and run the program in the Redboot terminal, run the following command:

> load -b 0x00218000 -h 10.15.167.5 "ARM/z283chen/kernel.elf" ; go

The executable can also be created by going into the directory "kernel" and calling make:

> cd kernel

> make

And then copy the elf to wherever you want to run it.

Data Structures and Algorithms

Calibration

Data structures:

- The track structure from track_data.c, from the course website, was used to store the track's information.
  - Additional fields index and switch direction were added to the track structure to better reflect the state of the track dynamically.
- A 4 dimensional array train_velocity[train number][speed][start sensor][end sensor] stores the number of ticks needed for a train to travel from one sensor to the next, at a set speed.
- A 2-d array default_speed[train number][speed] stores the default velocity of a train, if there is no data in train_velocity.
- A 3-d array train_acc[train number][speed][type] stores a specific train's acceleration at different types of sensors. For example, E14/15 and E5/6 are considered the same "type" of sensor, since they lie on the same track orientation.
- These arrays offer constant time access, are easy to manage, and have very little overhead.

Algorithms:

- As a baseline, the velocity of the train at a number of different sensors was calculated by recording the average time difference of when a sensor and the following sensor was activated. During execution, the times are dynamically adjusted as the trains move, using a weighted average of the new time and the old time.
  - Entering the command "cal [train number] [speed]" will print the new velocities, which can then be imported into our program.
  - All time is measured in ticks, where 1 tick is 0.01 seconds.
  - All distances are measured in millimetres.

Path Finding

Data Structures:

- The track structure from track_data.c is used to represent the graph of the track. This structure is a linked list of nodes, where a node is a sensor, switch, or an end of the track.

Algorithms:

- A path is found with DFS algorithm, since the graph of the track is sparse. The path is not the shortest path, but the algorithm will be replaced in future. If switches need to be flipped in the path, the switches are flipped when the train is two sensors before the switch.

Stopping

Data Structures:

- A stop wait queue keeps track of the amount of delay time and the sensor number that the train needs to stop at.

Algorithms:

- A worker task sends the stop command to the train when it is required. This way, the worker task does not block the calling task.
- To determine when the stop command should be sent to the train, the stopping distance of the train was measured. Once a path has been determined, the stopping distance is subtracted from the end point to determine the distance of the train from the preceding sensor when the stop command needs to be issued. Then the distance is converted into a delay time based on the experimental initial velocity at the preceding sensor.
- Commands:
    - stat [train number] [sensor name] [distance]
        - "**St**op **at**" stops the train at the specified sensor. The optional parameter "distance" tells the program how far after the sensor to stop in millimeters. For example, "stat 64 B6 20" will stop train 64 2 cm after sensor B6. The distance can be excluded from the command to stop the train exactly at a sensor. For example, "stat 64 B6" will stop the train at sensor B6.
    - staf [train number] [sensor name]
        - "**St**op **af**ter" sends the stop command when the train triggers the chosen sensor.

Things to note

- For this assignment, only train 64 was fully calibrated on track A, on speeds 6 and 10. Trains 71 and 76 were partially calibrated for only speed 10, so the results for those trains may vary.

Source Code

Git repository: https://git.uwaterloo.ca/x272liu/cs452

Sha1 of the commit: 3abf8cc90fb4b8ddfb0246e79a7f7dd8aa5212cb

All files in the repository are part of this submission.