# Project 2: I/O Elevators

McKenna Jones

April 27th, 2016

CS444: Operating Systems Two
Spring 2016

**Abstract**

This is the second project for CS444, Operating Systems Two. In this project we implement an I/O scheduler. Specifically, we implement the Shortest Seek Time First I/O scheduler. The SSTF, for short, is an elevator algorithm, also known as LOOK. As the name suggests, it functions as an elevator might, only servicing commands in one direction at a time.

## I. Design for SSTF Algorithm

To begin this project I started by looking at the implementation of the Noop scheduler. This is located within the block directory in the noop-iosched.c file. As the assignment description suggests, it is acceptable to base our solution off of this implementation. The Noop scheduler is fairly simple. Basically all it does is take whichever request is next in the queue and dispatch it. This is where SSTF is different. Instead of always dispatching the next request, SSTF dispatches based on the position of the request. It will select the I/O request that is closest to the current sector location, as long as it is in the same direction.

Based on this I decided to first add a few variables to the sstf data structure. First I added an integer which will be used to determine the direction scheduler. It will simply be a 0 or a 1. I also added a sector t variable to keep track of the current position of the scheduler.

My basic design for the dispatcher is as follows. First check if the list is currently empty. If it is not, get the next and previous nodes in the request queue. If next is equal to previous then we know there is only one request, and we can easily service it. Otherwise, we need to implement the elevator functionality. If we are going forward, we check the block location of the next request. If it is ahead of the node head, we service that request. But if it is behind the node head we reverse directions. The idea is exactly the same if we are currently going the opposite direction.

## II. Questions

### *A. What do you think the main point of this assignment is?*

The point of this assignment is first of all to become familiar with I/O schedulers. More specifically this assignment taught us not only how to understand them on a higher level, but also on a lower level. This is because we were actually required to dive deep into the kernel and figure out how a scheduler is implemented. Before this assignment, I understood what a scheduler was on a basic level, but I lacked the knowledge of how they were actually implemented. Now I actually understand the type of code that is used to implement one.

### *B. How did you approach the problem?*

I started by gaining a good understanding of how the Noop scheduler is implemented. This is where I spent a majority of my time, as I reused a large amount of this code in my SSTF implementation. Next I researched how a SSTF scheduler should work, in theory. After this I decided which parts of the Noop implementation I would need to change. For my approach I decided to only modify the dispatch function. My requests will be added to the queue always at the end. However, I will dispatch requests in an elevator fashion. My basic algorithm is to check for the closest request in the current direction, and if none are found, switch directions.

### *C. How did you ensure your solution was correct?*

To test my solution, I added printk statements to critical regions of my scheduler. From here I simply logged onto the VM and made some I/O requests. A simple example would be to do ls /. Based on the print statements I could see that some requests take longer to be dispatched than others, as they were in sectors far away from the current request. This verified that my scheduler was functioning properly.

### *D. What did you learn?*

I gained a wealth of information about I/O schedulers in this assignment. As discussed above, I now know the details of I/O scheduler. Other than that I have also learned how to apply linux kernel patches.

*E.*

## III. Version Control Log

| Date | Commit | Insertions | Deletions | Message |
|---|---|---|---|---|
| 4/24/16 | 43e9ac6d483ea20f116104a37a5623523fe39d2f | 234 | 0 | Added first version of assn 2 |
| 4/24/16 | 88ca886b2eaedc0e4d1d01a784a0a8457e45756f | 13,700 | 0 | Fresh Kernel |
| 4/25/16 | 7730733a240f0c39e182b4bc4589b7520e73742d | 158 | 0 | Changed Kconfig and makefile |
| 4/26/16 | 3977315c986953797d2fbc4b348cb3ca42b5d3e7 | 202 | 50 | Some progress from noop |
| 4/27/16 | e4787786352a2e2365661a6d2ebedddf8464d7d2 | 50 | 15 | Scheduler working |

## IV. Work Log

| Date | Hours | Work Done |
|---|---|---|
| 4/20/16 | 2 | Initial research on the topic |
| 4/22/16 | 2 | Working out an algorithm |
| 4/24/16 | 4 | Working on actual code |
| 4/25/16 | 4 | Got VM to actually use scheduler |
| 4/26/16 | 5 | Scheduler issues, most resolved |
| 4/27/16 | 7 | Finished writeup and scheduler |