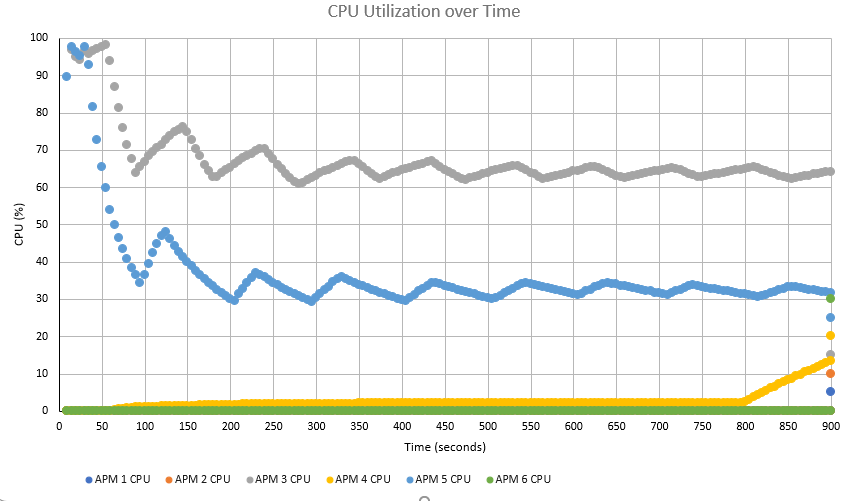
**NSSA-220 Project 1: Application Performance Monitoring**

**Ian Dinga, Jacob Neely, Ethan Gapay**

**Introduction**

This project was completed in order to create an Application Performance Monitoring Tool which can monitor the computer’s system and process metrics.Our tool captured the CPU% and memory % usage for the process metrics, and transmission and receiving data rates, disk writes, and available disk capacity for the system level metrics. This tool could be utilized to determine the amount of resources a program may use in its system as well as potential detecting malicious software being run in the background of a system. As a testing measure, our tool was used in tandem with a set of given C programs that were to run in the background and below our results after a 900 second test was shown.

**Process Level Metrics**

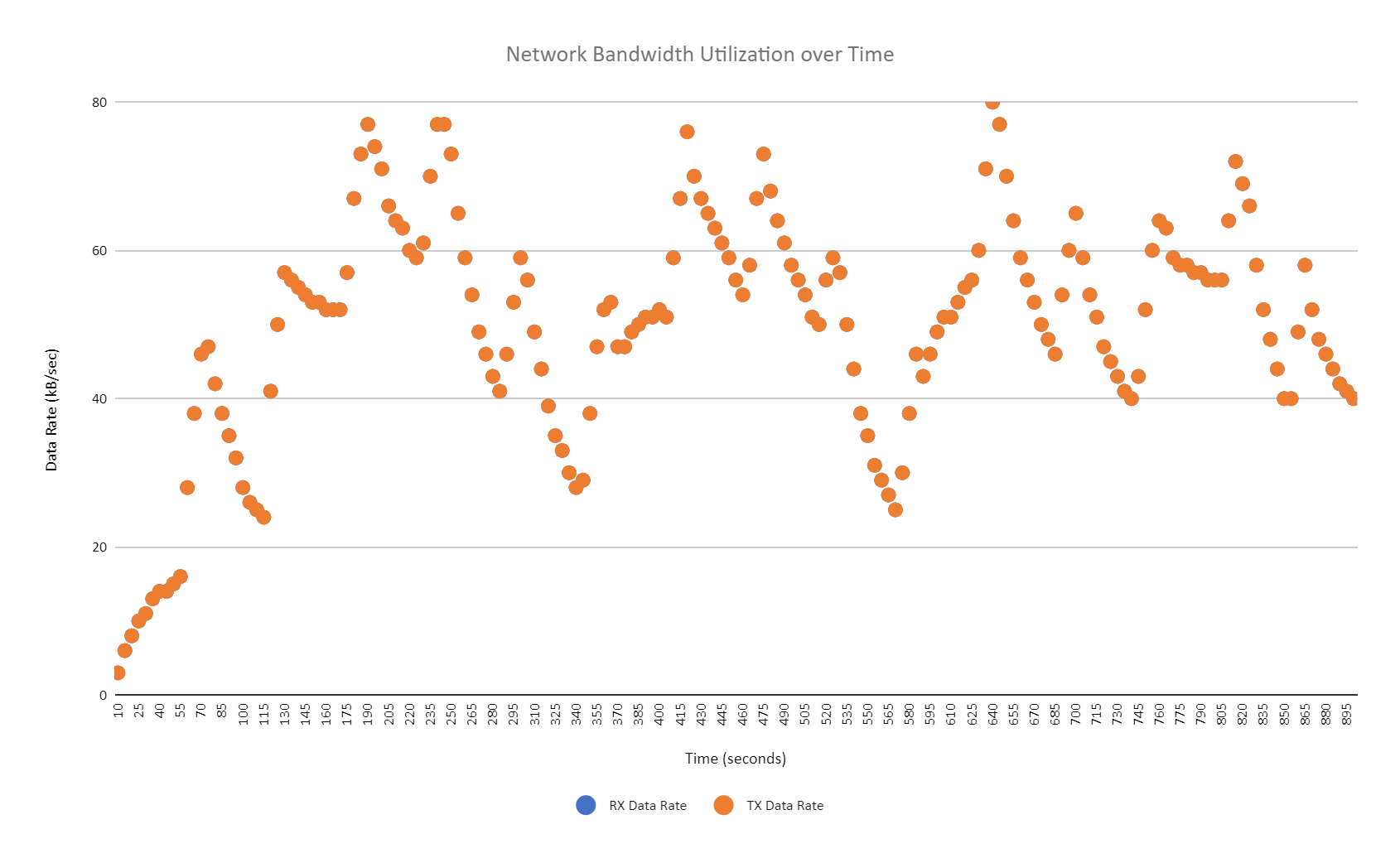


Shown above is the CPU utilization graph; it displays the % of CPU usage for each of the 6 APM programs that were run. It's clear to see that APM 3 used by far the most CPU %; it starts extremely high, peaking almost at 100%, and steadies out in a wave-like pattern between 60-70%. APM 5 follows a similar pattern, but it has a much larger fall than APM3 going from nearly 100% to steading at the 30-40% range. One more unique pattern is APM4 which starts small, but slowly increases, finally ending with a spike at the 800 second mark.

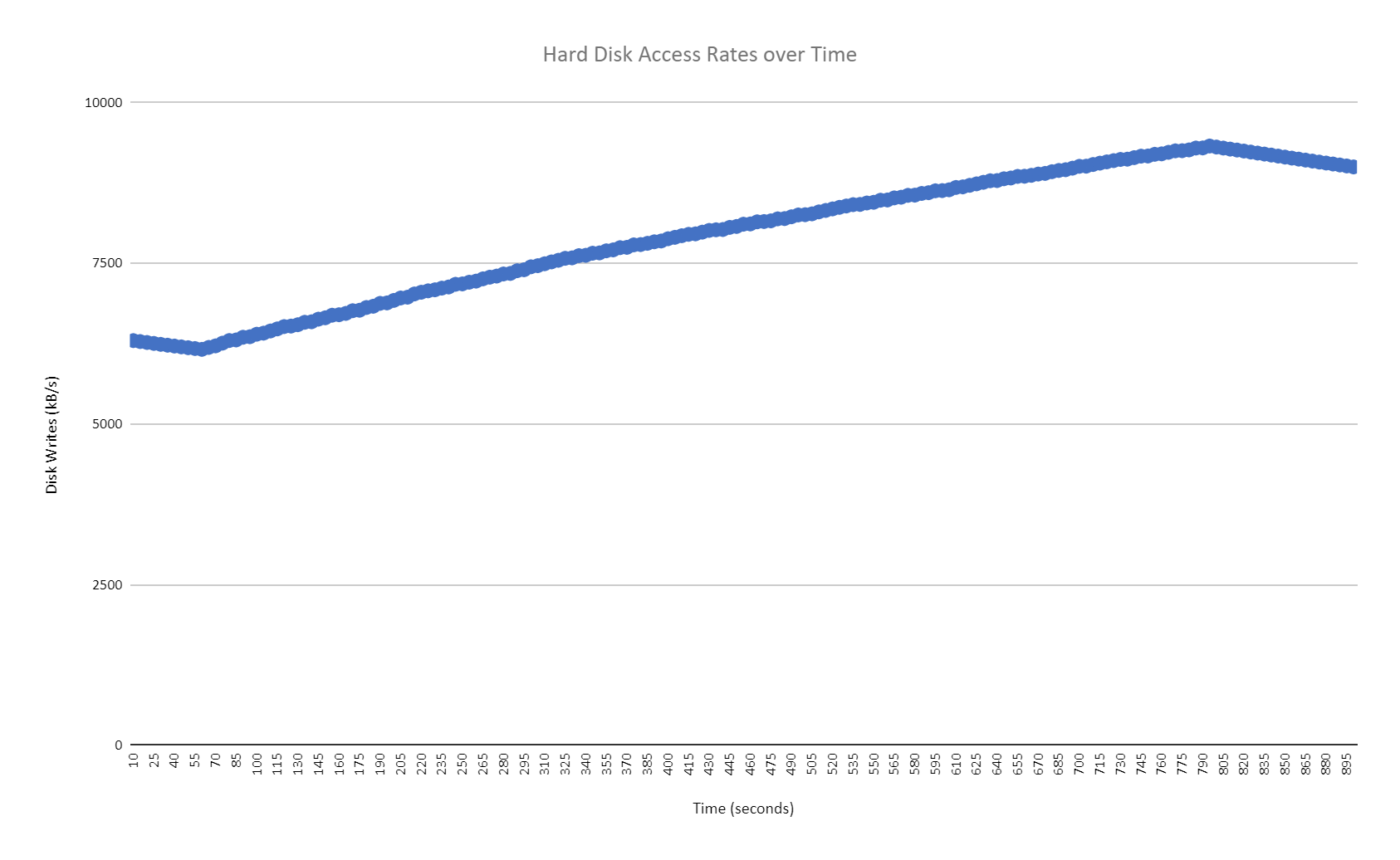


Above is the memory utilization graph, it shows the % of memory usage for each program in each of the 5 second intervals. While 3 of these programs had hardly any memory usage, APM 1 has a pattern of plateauing at a percentage and then dropping to none, overall it does have the highest usage at any given time. APM 6 is the other program that had significant memory usage, it created a simple sloped line, which is indicative of a memory leak, since over time it would use an absurd amount of memory.

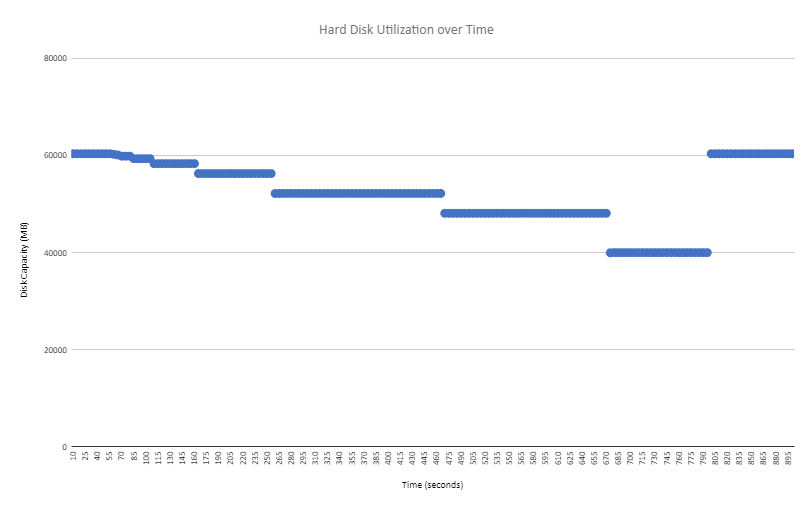
**System Level Metrics**



This network bandwidth utilization chart shows the amount of transmitted and received data over the 900 seconds of running the C programs. The C programs sent pings that were transmitted and received at basically the same rate according to the chart. It is shown that the pattern of RX/TX data rates moved in on the NIC became consistent over a 200 second interval fluctuating from about 22 kB/sec to 80 kB/sec.



Shown above is the hard disk access rates over time as the C files were constantly writing to an output file as the APM was running. It is shown that the disk writes started around 6000 kB/s and rose to about 9000 kB/s until it started to gradually go back down. The graph showed that the disk writes rose and fell at a steady rate.



Shown above is the Hard Disk utilization over time graph, this graph has an almost inverse relationship to the hard disk access rates graph, which is shown by the access rates graphs generally increasing pattern while hard disk utilization decreases steadily. This makes sense because as the hard disk writes in the access graph, it makes sense that storage should decrease in the utilization graph.

**Summary and Lessons Learned**

The VM we used did a pretty good job at handling the APMs. The network capacity was the only computing resource that struggled during the running of the APMs. In doing this project, it taught us how to read the data that was given when the different commands were being run. It required a collaborative effort in order to create a script to not only run the executables, but also record the usage of the VM to a file.