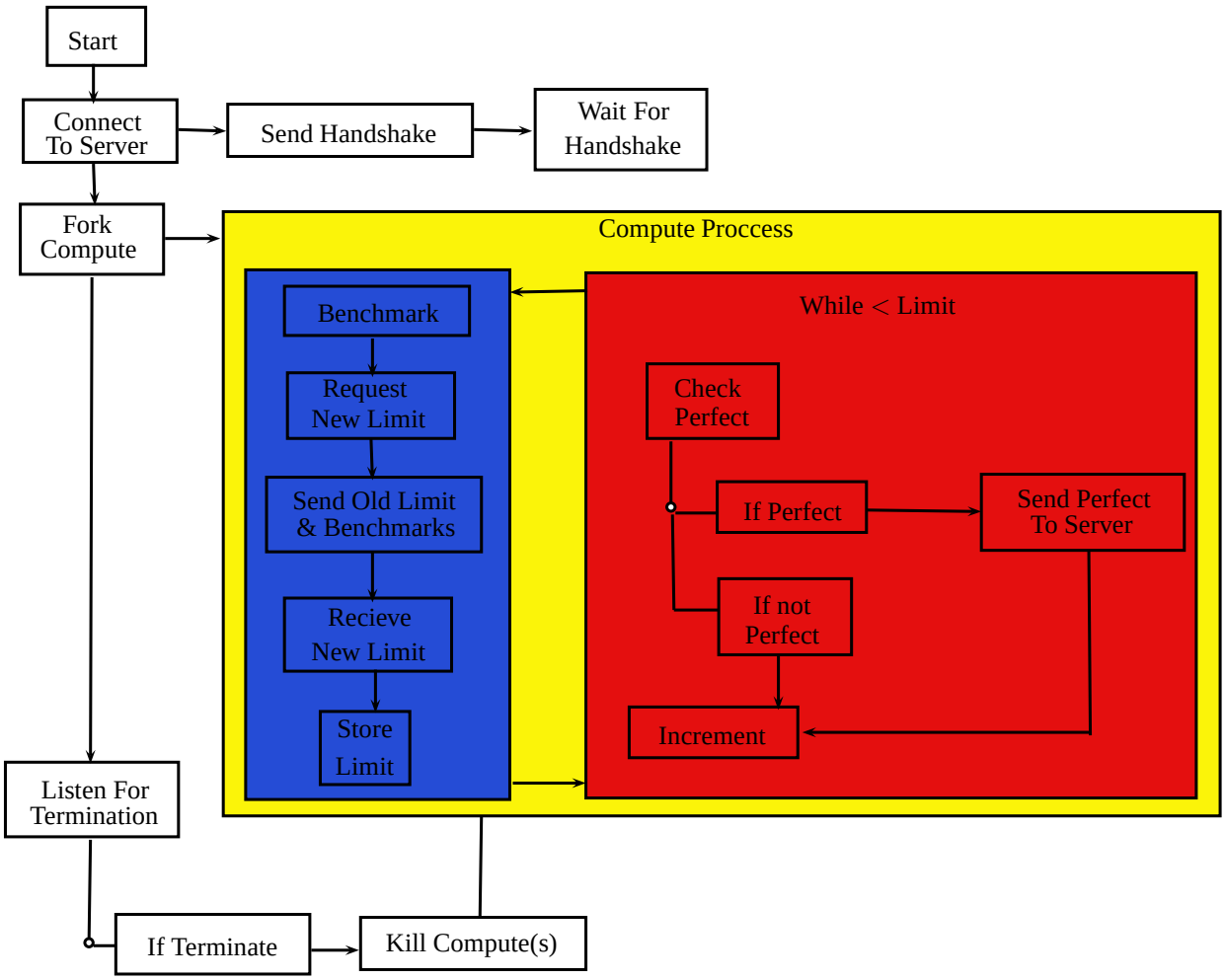
Project 5 Write Up

Sam Quinn

December 8, 2013

Design

compute.c Program



- Get start + stop range from **Manage** "Server".
- Calculate weather number is perfect or not within the range.*
- Within the timed loop calculate know number of operations store how long they take.
- Send performance information to **Manage** "Server", Use [FLOPS/IOPS].
- Have one process or thread that is dedicated to monitoring for a control socket to terminate.

*Note - The time to check the range from start to stop for perfect numbers should be 15 seconds.

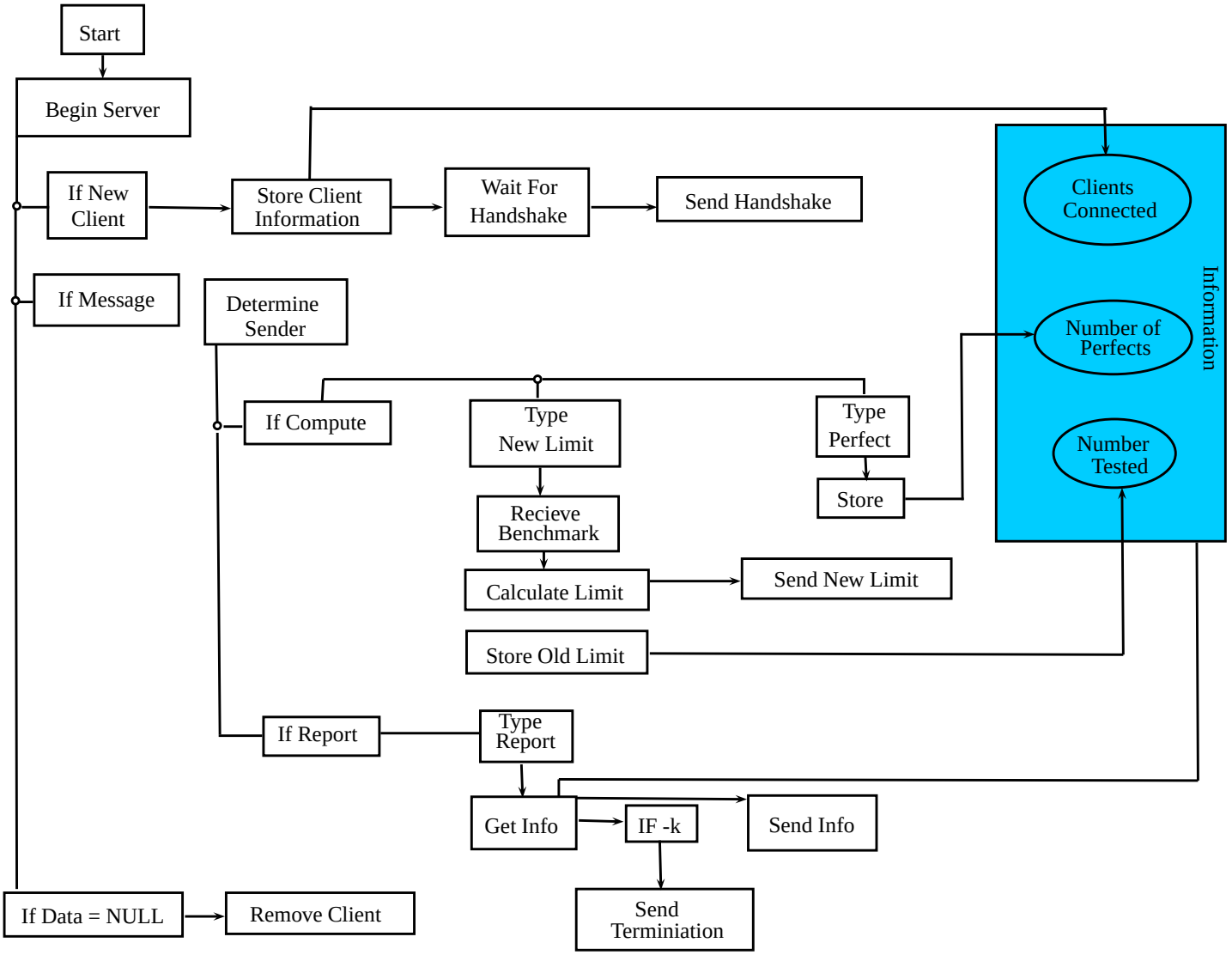
The **Compute** function will have multiple processes running in parallel, one of which will only be looking for the termination key sent from the server. This program will have to benchmark itself to receive the correct limit where it will only take 15 seconds to completely check every number in the range given to see if it is perfect of not. I will do this with a infinite loop that will use a *volatile* variable which means that for every iteration through the loop it will recheck the variable for changes. Once 15 seconds is reached it will return the number of modulo operations were executed to the server to get a new limit. The **Compute** function will continue checking for perfects, benchmarking itself, and requesting a new limit until an interrupt signal is received.

manage.py Program

- Get weather the number is perfect or not from **Compute**.
- Keep track of the threads or processes from the **Compute** program, so signals can be sent to each.
- Send **Compute** a range of numbers to check for perfect numbers within.*
- Receive interrupts from **Report** and send received signal to all processes or threads.

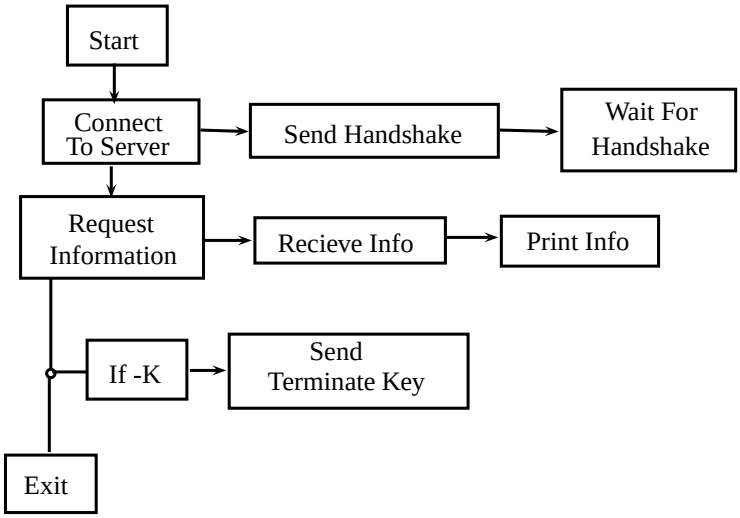
*Note - This should take 15 seconds for **Compute** to check for perfect numbers.

Manage is really the brains of this operation. **Manage** controls everything and is the only process that will talk to all of the other processes. **Manage** will first receive a connection request from any of the other programs and store their address in the connected clients list. The reason for storing these addresses is eventually we will need to kill them and this provides a nice way to access them all. **Manage** will receive benchmark statistics from **Compute** and calculate what the next limit should be for **Compute** to check for perfect numbers up to. **Manage** will also when requested by **Report** send How many perfect numbers **Compute** has found, the number of clients connected to **Manage**, and the number of numbers checked so far. When **Report** invokes the termination process for all the connected clients **Manage** will receive the kill key and be in charge of shutting every other process down.



report.py Program

The **Report** program has the least amount of work to do but has the most power out of all the programs. **Report** simply just receives the information regarding the number of perfect numbers found, the number of connected clients, and how many numbers have been checked so far from the server and prints this data out. The power comes from the -k flag used when running the program. When the -k flag is defined this will do the normal report stuff but once it has the information it goes on a killing spree killing every process it knows, no, but it does politely ask manage to kill every running process and then exits.



- Count the number of perfect numbers discovered from **Compute**.
- Count the number of numbers tested to be perfect.
- Count the number of processes or threads being used by **Compute**.
- -k will send an interrupt to **Manage** "Server" to kill all of the processes.
- Send a signal to **Manage** to kill **Report** and **Manage**.

Functions of Program Details

Work Log

```
commit 80ef4b567f4e13c19b334e5b40f81d6a48eca811
Author: Sam Quinn <quinnsa@os-class.engr.oregonstate.edu>
Date:   Sun Dec 8 18:27:41 2013 -0800
```

Manage.py - Deleted one debug print.

```
commit 59c88cd029ce7d5ba82d81295e62c34b6f07bf52
Author: Sam Quinn <quinnsa@os-class.engr.oregonstate.edu>
Date:   Sun Dec 8 18:23:43 2013 -0800
```

Manage.py - Final commit ready, Roger 1 Check!
Compute.c - Final Commit ready, Roger 2 Check!
Report.py - Final Commit ready, Roger 3 Check! Ready for lift off PSHHHHHHHHHEEEEWWWWWWWEWEWR-RRHH!

```
commit 3f17d90ce65b229e45869f6d78651c1266fffca7
Author: Sam Quinn <quinnsa@os-class.engr.oregonstate.edu>
Date:   Sun Dec 8 17:34:51 2013 -0800
```

Manage.py - Fully funtional and works great. Need to add comments to finish her off.
Compute.c - Fully functional and works great too, adding comments now.
Report.py - Fully functional and works great, Added comments already.

```
commit e216478b3a3c148ff20f6e80ae6aef21836c210f
Author: Sam Quinn <quinnsa@os-class.engr.oregonstate.edu>
Date:   Sun Dec 8 14:49:22 2013 -0800
```

Manage.py - Receives perfect numbers from compute and stores them and sends the perfect numbers to report.
Compute.c - Working on the benchmark function. Have the check perfect function working good and sending the data to manage.
Report.py - Works good nothing really need to work on here.

```
commit 3fac1cdb36ad58c5778c701944257c843ca2cef8
Author: Sam Quinn <quinnsa@os-class.engr.oregonstate.edu>
Date:   Sat Dec 7 17:30:16 2013 -0800
```

Manage.py - Most of the functionality is there it can receive -k from report and kill all of the processes, it sends new limits to compute, and it stays active.
Compute.c - Haven't gotten the bench mark to function to be used yet I think I have it working though just need to use it. Need to test my check perfect function I don't think that works. I need to change the number of bytes to write back to manage when sending the perfect number. Got the signal handler working well and everything.
Report.py - Have the -k function working and it displays fake number of perfects and number of processes and everything so just need to have manage send correct values and should be good!

```
commit f2d6d3d19930d81e0e7f34f1abe81203a36c0005
Author: Sam Quinn <quinnsa@os-class.engr.oregonstate.edu>
Date:   Fri Dec 6 18:15:51 2013 -0800
```

Manage.py - Sill trying to get multiple Computes to connect.
Compute.c - Have the fork in place and figured out that the main function of the compute program will be the one always listening for the termination.
Report.py - Haven't made any progress, am working on next though.

```
commit a4be32c84184e3d581278ed73f49b1236fedf95a
Author: Sam Quinn <quinnsa@os-class.engr.oregonstate.edu>
Date:   Mon Dec 2 22:20:56 2013 -0800
```

manage.py - Receives XML data from both report and compute ans stores their address in an array. Have not implemented parsing XML by hand I think I might try to do that later if I have time.
report.py - Sends XML handshake data to manage.
compute.c - Sends handshake XML data to manage.

```
commit 593d2c2b8aa69e92ae883c3b05a5619ffb2f3c9d
```

```
Author: Sam Quinn <quinnsa@os-class.engr.oregonstate.edu>
Date:   Sat Nov 30 17:59:23 2013 -0800
```

manage.py - Got a basic ECHO server set up. receives and sends same message back.
report.py - Got basic client running sends Hello world from python and recives
message back from manage. compute.c - Got basic client running sends Hello World and prints the message recived.

```
commit 6f4d627943405d053e1ea081e43140bda983108c
Author: Sam Quinn <quinnsa@os-class.engr.oregonstate.edu>
Date:   Thu Nov 28 20:43:29 2013 -0800
```

hw5_write_up.tex - Added for documentation of my design for this program.
compute.c - Added my code to find perfect numbers, I tested on paper and it worked never ran it on the computer yet.
report.py - Added some imports still no code just prints "Hello world!".
manage.py - No change from last commit just prints"Hello world!".
makefile - Added this I have changed the targets for the new programs and write up latex file.

```
commit 54bf68ad0806573c8d9575867cdbea97540441f1
Author: Sam Quinn <quinnsa@os-class.engr.oregonstate.edu>
Date:   Thu Nov 28 17:54:32 2013 -0800
```

Initial commit
Report.py - Prints "Hello World!"
Manage.py - Prints "Hello World!"
Compute.c - Prints "Hello World!"
Mentally preparing for this assignment and beginning design.

Challenges Overcame

This project was especially difficult to complete not only was I trying to study for finals, working on the final for this class, doing multiple other final projects I was teaching myself how to program in python but I got it done. I was very impressed with myself for completing this assignment as well as I did, I even made peace with the idea of turning in a non functioning program because I was so stressed out. The hardest part at first was learning how to program in python but after a few hours getting the basics done I realized it was easier than the c programing part. I got hung up on the multiplexing aspect in the **Manage** program but after looking at some examples on line and reading through the very well documented python.org I got it all working the way it should. Another thing that I had a rough time with was the benchmarking code in the **Compute** program. The benchmark was not hard to program but figuring out what the next limit should be was the difficult part for me. I designed my **Compute** program to work with multiple proceses running **Compute** but I left only tested it with one process for computing and one for listening for the termination key since in the assignment description it says T̈here may be more than one copy of compute running simultaneously“ but a quick change of the variable num_pro would implement multiple process for compute.