Project Overview
Clock synchronization deals with the idea that the internal clocks may differ between several computers. Even when initially set accurately, real clocks will differ after some amount of time due to clock drift, caused by clocks counting time at slightly different rates. This causes a problem in distributed systems, especially when system-wide events must be carefully coordinated, and when distributed applications use timestamps. For this project, you will implement two methods for synchronizing physical clocks – a master/slave system, and a distributed system. You may work alone, or in a group of two.

Implementation Details
In a master/slave system, a single node (the master) propagates the official system time to all other machines (the slaves) at regular intervals. For this scenario, you must handle the following scenarios:

- For this project, the nodes in the system must decide which machine among them is elected the “time master.” The master machine must be selected by the nodes in the system. This will be accomplished via an election. The algorithm you choose for this election is up to you, so long as the process is clear. For example, you could randomize a weight for each machine, and perform election by bullying. Describe your approach.
- The time master may lose communication with some or all of the nodes, or the time master may crash completely. In this case, the nodes must choose to hold a new election after a “reasonable” period of silence from the time master. Describe your approach.

In a fully distributed system, messages are frequently passed among all nodes, and the nodes collectively determine the system time. For this scenario, consider the following:

- You are free to choose whatever algorithm you like to determine the overall system time. For example, you could attempt to average the local times of each node, or you could update to the maximum time received, or some other algorithm. Describe your approach.
- Much like the master/slave system, you should find some way to handle calculating the overall system time if nodes crash or are unreachable, or if there are significant delays in message transit times. Describe your approach.

For both systems, there are additional factors you must consider:

- Each process you create should have an artificially-high randomized drift rate, between half of normal speed and double normal speed. This drift rate will remain constant throughout the runtime of the system, except for when the process is correcting itself to match system time. After the process has matched what it believes is the current system time, the process will return to its original drift rate.
- Clocks are never allowed to travel backward in time. Rather, they must slow until matching the agreed upon system time.
Your system should be able to handle a variable number of simulated systems. For this project, assume that the range will be between 10 and 100 processes.

NTP (Network Time Protocol) is implemented as a daemon process in UNIX systems, and Windows includes the Windows Time Service, which syncs the system clock to an NTP server. For this project, you are not permitted to use any NTP service to determine the time of your distributed system after the initial startup. However, you will use NTP to evaluate the correctness of your implementations (more details on this in System Assessment).

The processes in your distributed system could operate on multiple physical machines, they could be multiple simulated processes on a single physical machine, or a combination of both. Note that processes that run on the same hardware will be able to communicate faster than processes that run on different machines, due to inherent network latency. It is up to you to determine how to best handle the discrepancies that results from network delay between machines when calculating the overall system time.

All design decisions that you make, whether from a decision made due to a constraint above or due to other issues, should be clearly justified in your report. Extra credit may be awarded for creative solutions, or for handling additional conflicts not specified above (extra credit at the discretion of the TA; you may want to consult with the TA prior to implementing). If you wish to claim extra credit for any of your solutions, please contact the TA before Monday, October 3rd, so that the value of the extra credit (if any) may be evaluated.

System Assessment
You will assess correctness of your systems in comparison with NTP (Network Time Protocol) over a ten-minute period. After your distributed system has run for ten minutes (real-time), all nodes should output their current time, as well as the overall system time. Additional details may also be relevant (and interesting to report), such as whether the node is a master or a slave (and if it spent time in both capacities, how long for each), how many time-updating messages it had sent or received (i.e., the overhead of the scheme), or the mean square error (or some other statistical measure of the goodness of your schemes).

You will not be penalized for having a system time that is wildly different from the correct time, so long as you have justified your design decisions and have an explanation for the discrepancy. However, you should try to optimize your system to approach the correct time – if in your early system testing you discover a sizable anomaly, you should make every effort to determine the cause and tweak your algorithms appropriately.

Report
You will submit a 3-5 page, single-spaced report (with no penalty for going over, should it be necessary, unless it is too much over). This report will contain:

- (1-2 pages) A brief, high-level overview of the implementation details for both systems your group has implemented. In this section, you must justify any design decisions you have made, and discuss the effects of these decisions.
- (3-4 pages) An analysis of the performance of your systems, on a variable number of processes. At a minimum, you should run experiments on 10, 25, 50, and 100 processes. Your analysis should compare your two implementations to each other and to NTP.
This report should be light on the implementation details, heavy on reporting the results of your experiments. The level of detail of your analysis and the items you analyze (e.g., message counts, latency, or error) is up to you, but a better structured and more detailed evaluation of system performance will be awarded more points.

**Demonstration**

You will be allotted 45-60 minutes for a live demonstration of your project/implementation. The location of the demonstrations is your choice (so long as it is in the CS building) and your responsibility – if you wish to use a computer lab, be sure that you have access at the time you demonstrate; if you wish to use the machines in your office or your personal machines, have them set up and ready when for when your demonstration time begins.

The demonstration will consist of the following components:

- The group will run their two systems, given a number of processes that will be determined by the TA at runtime. The two systems may be run sequentially or in parallel. You should create a startup script which will automatically generate the requested number of processes, rather than manually creating the processes during the demonstration.
- While the systems are running, you will be asked details about your implementations, your design decisions, and the distribution of workload inside your group during the implementation process.
- After both systems have finished running, we will determine which system kept time more accurately, and why this is so.
- Each student may be asked to perform minor modifications to your system to handle unusual circumstances, such as losing communication with the time master. You will be asked to explain how these changes are made, implement them, and run the system again.
- You may also be asked questions regarding general time synchronization theory, beyond physical clocks alone.

In all cases during the demonstration, when a question is directed at a specific member of the group, only that group member will be permitted to respond. One group member will not be penalized for another group member’s lack of knowledge; however, this component of your grade will be separate. If your group divided the work such that each group member implemented one system, you should familiarize yourself with your partner’s system, as questions about it could be directed to you.

All demonstrations will take place between Sunday, October 9th and Tuesday, October 11th, except for emergency cases. “Our systems don’t work yet” is not an emergency consideration. Your group’s individual demonstration time will be determined at a later date, on a first-come-first-served basis, sent in an email to the class mailing list.
Grading
Grading will be broken down as follows:
- System implementation: 50%
- Report: 25%
- Demonstration: 25%

The demonstration score will be an individual grade; the system implementation and report segments will be a collective group grade.

Submission Details
By the deadline (again, that is Sunday, October 9\textsuperscript{th} at 9:00 AM EST) you must submit the following:
- The source code for both systems your group has implemented.
- Your report

Your submission must be added to a compressed file or tarball, and emailed to the TA at jew51@pitt.edu no later than the deadline above.

Questions
Any additional questions about the details of this assignment should be directed to the course TA, either by email or in office hours.