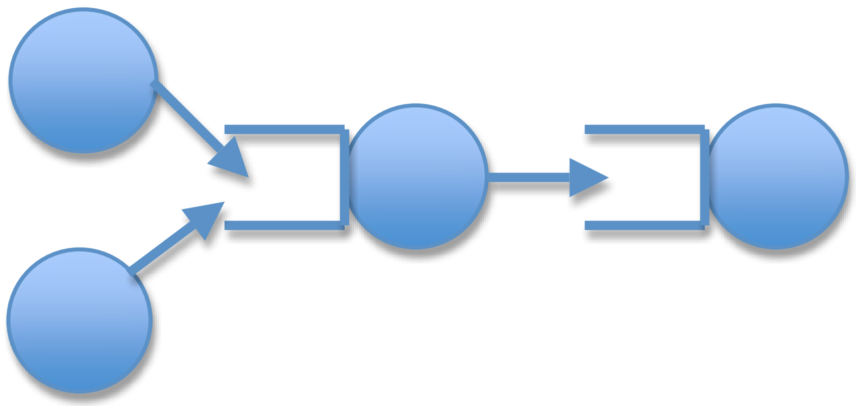
Project for COMP 655-A Time Warp Simulator

Below is an artists rendering of the queuing network which you will simulate in this project. Circles represent LPs and the rectangular shapes represent (input) buffers into which messages are sent by the LPs.

You will simulate this system via Time Warp, making use of two machines. Three of the LPs (on the left) should be in one machine, while the one on the (far) right should be in the second machine. You will also implement a single processor simulation of this simulation.



The single processor version of the simulation should make use of one heap, scheduling the events via the smallest time-stamp first algorithm.

In the Time Warp simulator, you should schedule events in each processor by using one heap for all of the LPs and using a smallest time-stamp first scheduling algorithm, i.e. pick the event with smallest time-stamp from the top of the heap and simulate it, irrespective of at which LP the event is located.

You will implement Samadi's GVT algorithm as part of the project. You will need to decide on the frequency with which it is called. You will also implement a fossil collection algorithm.

You can generate events by picking their inter-arrival times from a distribution, e.g. an exponential distribution. You should start the project by using a fixed (i.e. deterministic) inter-arrival time. This will make de-bugging easier for you to do. Then you can switch to an inter-arrival time picked from a distribution. Likewise, you should first use a constant processing time for an event and make use of a time picked from a distribution at a later time. The processing of an event consists of simply adding a numerical value to its time-stamp. For example, if the smallest element in the heap has time-stamp 15, processing this event consists of adding 1 (for example) to 15, and getting 16.

The input buffers should have a fixed length.

Here is a suggested approach to the project.

1. First implement the single processor simulation, followed by the Time Warp version. T
2. Partition the queuing network (as indicated above) and add message passing via MPI. If you are new to MPI, take a look at the references on the course web-page.
3. Finally, add the Time Warp features-input, output and state queues, anti-messages, cancellation, and fossil collection.

Here are the deliverables:

1. Documented source code, including a description of the purpose of each routine and the names/meanings of the variables.
2. Description of the program, including the data structures, message format, and the calling sequence (especially the main loop) of the program. I expect a thorough description, i.e. not something cobbled together at the last minute.

You will be expected to demonstrate the program, making use of an input script. As part of the demonstration, you will be asked to trace the sending of anti-messages, their subsequent annihilation and the re-instantiation of state during a rollback.