**MSIM 406: Optimistic Algorithm**

Problem:

You are to implement the ability to roll back time in an optimistic time management simulation executive. You do not need to worry about rolling back the state, just time. However, this does involve cancelling improperly scheduled events created as a function of incorrectly executing other events.

Tasks:

1. Create a test platform to test your simulation executive. Create a single class with a single event. Given p LP's, each LP should have 1 object of this class created. On initialization, schedule n events for this object on each LP (so there will be np events in the system, this will be constant during the execution of the "simulation"). The event randomly schedules an event on an object (which may or may not be on another LP). It should schedule the event td in the future (simulation time), where td is exponentially distributed with a mean of td_mean. The event should be randomly scheduled on an LP (uniformly distributed from 0 to p-1, this can include itself) It should then "sleep" for a random amount of wallclock time, tw, which is uniformly distributed between 0 and tw_max. You should vary p, n, td_mean and tw_max to get interesting behavior, in particular, to get events scheduled in the past.
2. Implement a simulation executive where all events are saved; rather than removing events from an event list, a current event pointer just scans the list. It is suggested, but not necessary, to make the list a doubly linked list to facilitate the ability to move back and forth. When an event arrives in an LP's past, simply insert it in the list, and set the current event pointer to the newly scheduled event. Then allow the simulation to continue execution. This will result in the incorrect execution of the "simulation", and the growth of the number of events in the system as scheduled events will not be cancelled. Keep track of rollback, reporting the total number of roll backs on each LP and the average number of events rolled back each time. During debugging you should generate a useful trace of the event activities so you can see the behavior. You probably should give each event a unique id so you can see them rollback, and even get re-executed.
    a. Note: since there is only 1 event, and it does not "do" anything, you only need to communicate the scheduled time of the event in a message. On the receiving end, you can get an event from the test application and schedule that event in your event list.
3. Once 2 is working and you can observe roll back, implement anti-messages to cancel incorrectly scheduled messages. This will require creating an anti-message for each scheduled event and saving it with the event that did the scheduling. Note that there is only 1 scheduled event per event executed, so you do not require Then when rolling back, go back through each message and send the anti-message on its way. Make sure to generate a useful trace to observe the behavior.

Deliverables:

1. For task 2:
    a. Visual Studio project

    b. A discussion of the impact of your parameter selection and the reported results.
2. For task 3:
    a. Visual Studio project
    b. Select a section of your trace and discuss how the rollback and antimessages are reflected in the trace. Do not provide your whole trace, just an interesting section with a discussion.