Michael Reiter
V00831568

CSC 360 Assignment 2 Design Document

1. **How many threads are you going to use? Specify the task that you intend each thread to perform.**

   I will use N threads, where N is the number of flows in the input file. Each thread sleeps until its arrival time then transmits for its transmission time.

2. **Do the threads work independently? Or, is there an overall "controller" thread?**

   The threads work independently of each other. The transmission queue is sorted by scheduling logic within each thread [i.e. sortQueue()]. There is also a main thread (the main function) waiting for all threads to terminate so it can join them.

3. **How many mutexes are you going to use? Specify the operation that each mutex will guard.**

   I will use one mutex. It guards that two flows do not run at the same time. It is locked/ unlocked in requestPipe() and releasePipe() as well as by the convar.

4. **Will the main thread be idle? If not, what will it be doing?**

   The main thread will be idle as it waits for all other threads to terminate execution so they can be joined.

5. **How are you going to represent flows? What type of data structure will you use?**

   I will create a flow typedef struct with properties id, arrivalTime, transmissionTime and priority. All flows will be stored in an array.

6. **How are you going to ensure that data structures in your program will not be modified concurrently?**

   I will use a mutex to guard against concurrent modifications (lock the critical section).

7. **How many convars are you going to use? For each convar:**
   a) Describe the condition that the convar will represent.
   b) Which mutex is associated with the convar? Why?
   c) What operation should be performed once pthread_cond_wait() has been unblocked and re-acquired the mutex?

   I will use one conditional variable.
   a) It represents whether or not a flow is executing (i.e. whether the pipe is free for transmission).
   b) The single mutex corresponds to the single conditional variable. It is used when a thread is transmitting. The mutex is automatically released when the convar is signaled.
   c) When pthread_cond_wait() has been unblocked and the mutex has been reacquired, the flow that just finished transmitting is removed from the queue. Then the next flow in the queue should be resorted to determine the next flow that should transmit.

Michael Reiter
V00831568

8. **In 25 lines or less, briefly sketch the overall algorithm you will use.**

    Main:
    Parse flows text file into array of pointer to flow structs with properties id, arrivalTime, transmissionTime and priority
    Initialize mutex and conditional variable
    Create a thread for each flow in a joinable state
    Each thread executes the thread function
    Wait for all threads to terminate and join them
    Destroy the mutex and conditional variable

    Thread function:
    Sleep until arrival
    Print arrival
    Lock the mutex
    Insert flow into the queue
    Sort the queue
    If waiting for a flow to finish transmitting, print waiting
    While the front of the queue is not the current flow, wait on the convar
    Set the transmission flag to true
    Unlock the mutex
    Print transmission start
    Sleep for transmission time
    Print transmission finish
    Lock the mutex
    Broadcast on the convar
    Remove from the front of the queue
    Set the transmitting flag to false
    Unlock the mutex