Events

```
typedef struct
{
    pthread_cond_t cond;
    pthread_mutex_t mutex;
    int count;
} event_t;
```

Custom semaphores implemented using pthread mutex and conditions.

Related Functions:

```
void event_init(event_t **event_ptr);
```

```
void wait_event(event_t *event);
```
Semaphore wait. Decrease count by one.
If count >= 0, returns.
Else, waits for event to be signaled.

```
void signal_event(event_t *event);
```
Semaphore signal. Increase count by one.
Signal the event.

```
void broadcast_event(event_t *event, int n);
```
Increase count by n.
Broadcast the event.

```
void reset_event(event_t *event);
```
Set count to 0.

```
int timed_wait_event(event_t *event, long ms);
```
Decrease count by one. Wait event but time bounded. If the signal is not received, count is incremented to get its old value.

Atomic Integers

```
typedef struct
{
    pthread_mutex_t mutex;
    int value;
} atomic_t;
```

Atomic integers are integers that can be updated or read without any race condition. They are secured by mutexes. Functions related to atomic integers are trivial. Except these two:

```
int atomic_cond_set(atomic_t *atomic, int cond, int value);
```
Set the atomic int value to given value if its old value is equal to cond.

```
void atomic_cond_signal_event(atomic_t *atomic, int cond, event_t *event);
```
Signal the given event if atomic int is equal to cond. This function is used to implement a barrier for determining if all commentators decided on answering the question or not.

Queue

The queue implemented for this assignment is providing atomic pop and push functions. So race is prevented for queue related operations.

Implementation

We have 2 main functions: commentator_main and moderator_main
These functions are runners of the commentator threads and the moderator thread.

In each round, commentor_main calls commentator_round and moderator_main calls moderator_round. These functions are contain procedures in which commentators and moderators must do every round.

Implementing the 0 answers for the question case was simple. We have a while loop which is popping thread ids until there are no ids left. Since in the 0 answer case, the queue is empty, the while loop never iterates. Program continues with the next round.

breaking_news_start is broadcasted from main function each second with probability B. The listeners of this breaking_news_start event are news reporter thread and the current commentator who is speaking. Additional to this event, there is an atomic variable to indicate if breaking news is happening. This atomic variable is used by moderator to check breaking news, if it is occurring at that moment, it waits until breaking_news_end event is signalled. This event is signalled by news reporter thread after its 5 second sleep is done.

Current Problem

We detected that sometimes (rarely), the code gets stuck in an infinite loop of breaking news. When this happens, the program does not go on with commentators or the moderator. A news start, then end. Immediately after this, another one starts. But this happened only once. We didn't see this happening in our latest tests.