1. The following modifications will occur:

Init ‘q’: a queue of tuples (int, pid, isWriter), initially empty

And numA: an integer counter, initially 0

On request,

1. Send a request message to all other processes with the tuple (logicalClock, i, isWriter)
2. Insert the tuple (logicalClock, i, isWriter) into q
3. Set numA to 0

On receive (request, (ts, j, isWriter)) from Pi:

1. Insert the tuple (ts, j, isWriter) into q
2. Send an acknowledgment message to Pi with the tuple (a, logicalClock, isWriter)

On receive (a, ts, isWriter):

1. Increment numA by 1
2. If numA == N - 1:
   1. If isWriter is false and there is no writer ahead in q, enter the critical section
   2. If isWriter is true and Pi is the smallest in q, enter the critical section

On receive from Pi:

1. Delete the request tuple from Pi from q
2. If numA == N - 1:
   1. If isWriter is false and there is no writer ahead in q, enter the critical section
   2. If isWriter is true and Pi is the smallest in q, enter the critical section Release:
3. Delete the request tuple from Pi from q
4. Send a release message to all processes

On release:

Delete the given request Pi from the q and send this release msg to other processes

2a) We continue to maintain a

* q: queue of (timestamp, pid, isWriter) initially empty
* numA: integer initially 0
* numI: integer initially 0
* int K

Request:

* Send a request message with (timestamp, i, isWriter) to all other processes
* Insert (timestamp, i, isWriter) into q
* Set numA to 0 On receive request message with (timestamp, j) from Pj:
* Insert (timestamp, j) into q
* Send an acknowledgement message with (timestamp) to Pj

On receive acknowledgement message with (timestamp):

* Increment numA
* If numA == N-1 and Process i has the smallest timestamp in q and numI < K:
  + Increment numI
  + Enter critical section

On receive release message from Pj:

* Delete the request process j from q
* If numA == N-1 and Process i has the smallest timestamp in q and numI < K:
  + Increment numI
  + Enter critical section

Finally, on Release:

* Decrement numI
* Send a release message to all other processes

2b)

Variables:

* q: list of process IDs, initially empty
* TS1: integer, initially infinity
* numA: integer, initially 0
* numB: integer, initially 0
* K

Request:

* Set TS1 to current logical clock
* Send a request message through TS1 to other processes
* Reset numA to 0

On receive(request, TS2) from process Pj:

* If TS2 from process Pj is less than TS1 or numB >= K, send the msg to process Pj
* Else, append Pj to q

On receive of acknowledgement from process Pj:

* Increment numA by 1
* If numA equals N - 1:
  + Increment numB by 1
  + Enter critical section

Release:

* TS1 = MAX\_INT
* Decrement numB by 1
* For each process ID j in q:
  + Send the msg to process j
* Clear q.