second protocol does not require processes to communicate only through broadcast messages. Both protocols require that the messages be delivered reliably (lossless and uncorrupted).

BASIC IDEA. The basic idea of both the protocols is to deliver a message to a process only if the message immediately preceding it has been delivered to the process. Otherwise, the message is not delivered immediately but is buffered until the message immediately preceding it is delivered. A vector accompanying each message contains the necessary information for a process to decide whether there exists a message preceding it.

BIRMAN-SCHIPER-STEPHENSON PROTOCOL

1. Before broadcasting a message m, a process P_i increments the vector time $VT_{P_i}[i]$ and timestamps m. Note that $(VT_{P_i}[i] - 1)$ indicates how many messages from P_i precede m.
2. A process $P_j \neq P_i$, upon receiving message m timestamped VT_m from P_i, delays its delivery until both the following conditions are satisfied.
   a. $VT_{P_j}[i] = VT_m[i] - 1$
   b. $VT_{P_j}[k] \geq VT_m[k] \quad \forall k \in \{1, 2, ..., n\} - \{i\}$
   where n is the total number of processes.
   Delayed messages are queued at each process in a queue that is sorted by vector time of the messages. Concurrent messages are ordered by the time of their receipt.
3. When a message is delivered at a process P_j, VT_{P_j} is updated according to the vector clocks rule IR2 (see Eq. 5.4).

Step 2 is the key to the protocol. Step 2(a) ensures that process P_j has received all the messages from P_i that preceed m. Step 2(b) ensures that P_j has received all those messages received by P_i before sending m. Since the event ordering relation "$\rightarrow$" imposed by vector clocks is acyclic, the protocol is deadlock free.

The Birman-Schiper-Stephenson causal ordering protocol requires that the processes communicate through broadcast messages. We next describe a protocol proposed by Schiper, Eggli, and Sandoz [20], which does not require processes to communicate only by broadcast messages.

SCHIPER-EGGLI-SANDOZ PROTOCOL

Data structures and notations. Each process P maintains a vector denoted by V_P of size $(N - 1)$, where N is the number of processes in the system. An element of V_P is an ordered pair (P', t) where P' is the ID of the destination process of a message and t is a vector timestamp. The processes in the system are assumed to use vector clocks. The communication channels can be non-FIFO. The following notations are used in describing the protocol:

- t_M = logical time at the sending of message M.
- t_{P_i} = present/current logical time at process P_i.

THE PROTOCOL

Sending of a message M from process P_1 to process P_2

- Send message M (timestamped t_M) along with V_P_1 to process P_2.
- Insert pair (P_2, t_M) into V_P_1. If V_P_1 contains a pair (P_2, t), it simply gets overwritten by the new pair (P_2, t_M). Note that the pair (P_2, t_M) was not sent to P_2. Any future message carrying the pair (P_2, t_M) cannot be delivered to P_2 until $t_M < t_{P_2}$.

Arrival of a message M at process P_2

If V_M (the vector accompanying message M) does not contain any pair (P_2, t)
then the message can be delivered
else (* A pair (P_2, t) exists in V_M *)
If $t \not< t_{P_2}$ then
the message cannot be delivered (*it is buffered for later delivery*)
else
the message can be delivered.

If message M can be delivered at process P_2, then the following three actions are taken:

1. Merge V_M accompanying M with V_P_2 in the following manner:
   - If $(\exists\ (P,t) \in V\_M,$ such that $P \neq P_2)$ and $(\forall (P', t) \in V\_P_2,\ P' \neq P)$, then insert (P, t) into V_P_2. This rule performs the following: if there is no entry for process P in V_P_2, and V_M contains an entry for process P, insert that entry into V_P_2.
   - $\forall P, P \neq P_2$, if $((P,t) \ni V\_M) \wedge ((P,t') \in V\_P_2)$, then the algorithm takes the following actions: $(P,t) \in V\_P_2$ can be substituted by the pair (P, t_{sup}) where t_{sup} is such that $\forall i, t_{\text{sup}}[i] = \max(t[i], t'[i])$. This rule is simply performing the step in Eq. 5.4 for each entry in V_P2.

     Due to the above two actions, the algorithm satisfies the following two conditions:

   a. No message can be delivered to P as long as $t' < t_P$ is not true.
   b. No message can be delivered to P as long as $t < t_P$ is not true.
2. Update site P_2's logical clock.
3. Check for the buffered messages that can now be delivered since local clock has been updated.

A pair (P,t) can be deleted from the vector maintained at a site after ensuring that the pair (P,t) has become obsolete (i.e., no longer needed) (see Problem 5.3).

5.6 GLOBAL STATE

We now address the problem of collecting or recording a coherent (consistent) global state in distributed systems, a challenging task due to the absence of a global clock and