CMSC 327 Distributed Systems

Project 4: Group Communication Clarification
Due November 19, 2010

The book is a bit confusing about how to implement causally ordered multicast using vector clocks. The original paper by Birman[1] is much more precise: The relevant portions are from pages 280 & 281:

> 4.3 Vector Time
>
> Our delivery protocol is based on a type of logical clock called a *vector clock*. The vector time protocol maintains sufficient information to represent $\rightarrow$ precisely.
>
> A vector time for a process p_i, denoted $VT(p_i)$, is a vector of length n (where $n = |P|$), indexed by process-id.
>
> (1) When p_i starts execution, $VT(p_i)$ is initialized to zeros.
> (2) For each event $send(m)$ at p_i, $VT(p_i)[i]$ is incremented by 1.
> (3) Each message multicast by process p_i is timestamped with the incremented value of $VT(p_i)$.
> (4) When process p_j delivers a message m from p_i containing $VT(m)$, p_j modifies its vector clock in the following manner:
>
> $$\forall k \in 1 \cdots n : VT(p_j)[k] = \max(VT(p_j)[k], VT(m)[k]).$$
>
> That is, the vector timestamp assigned to a message m counts the number of messages, on a per-sender basis, that causally precede m.
>
> Rules for comparing vector timestamps are
>
> (1) $VT_1 \leq VT_2$ iff $\forall i: VT_1[i] \leq VT_2[i]$
> (2) $VT_1 < VT_2$ if $VT_1 \leq VT_2$ and $\exists i: VT_1[i] < VT_2[i]$
>
> It can be shown that given messages m and m', $m \rightarrow m'$ iff $VT(m) < VT(m')$: vector timestamps represent causality precisely.

and

> The protocol is as follows:
>
> (1) Before sending m, process p_i increments $VT(p_i)[i]$ and timestamps m.
> (2) On reception of message m sent by p_i and timestamped with $VT(m)$, process $p_j \neq p_i$ delays delivery of m until:
>
> $$\forall k : 1 \cdots n \begin{cases} VT(m)[k] = VT(p_j)[k] + 1 & \text{if } k = i \\ VT(m)[k] \leq VT(p_j)[k] & \text{otherwise} \end{cases}$$
>
> Process p_j need not delay messages received from itself. Delayed messages are maintained on a queue, the CBCAST *delay queue*. This queue is sorted by vector time, with concurrent messages ordered by time of receipt (however, the queue order will not be used until later in the paper).
> (3) When a message m is delivered, $VT(p_j)$ is updated in accordance with the vector time protocol from Section 4.3.

Two points of clarification:

1. Update the vector clocks *after* delivering the message to the application (i.e. not when the message is initially received, but after the two conditions have been met).

2. You don't increment the vector clock upon receipt or delivery, only upon send.

[1] "Lightweight causal and atomic group multicast." by Schiper, Birman, and Stephenson. http://portal.acm.org/citation.cfm?id=128742