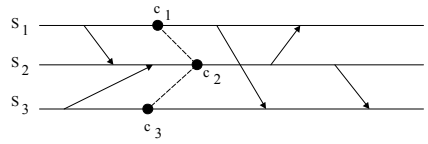
Cuts

A cut is a set of cut events, one per node, each of which captures the state of the node on which it occurs.



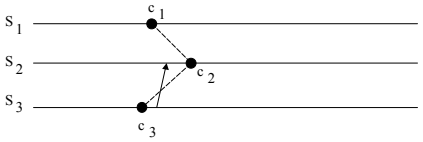
C = { c_1, c_2, c_3 }

Consistent Cut

A cut C = {c_1, c_2, c_3, … } is consistent if for all sites there are no events e_i and e_j such that:

(e_i --> e_j) and (e_j --> c_j) and (e_i -/-> c_i)

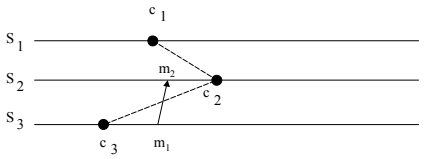
An inconsistent cut:



Ordering of Cut Events

The cut events in a consistent cut are not causally related. Thus, the cut is a set of concurrent events and a set of concurrent events is a cut.

Note, in this inconsistent cut, c_3 --> c_2.



Time of a Cut

Each cut event, c_i, is assigned a vector timestamp, VT_{c_i}. The cut set is assigned a vector timestamp VT_c where

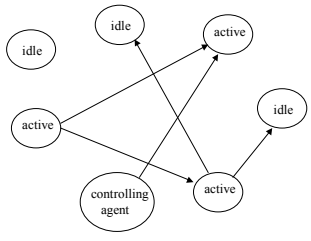
$$VT_c[i] = \max(VT_{c_1}[i], VT_{c_2}[i], \ldots, VT_{c_n}[i]).$$

The cut is consistent iff:

$$VT_c = (VT_{c_1}[1], VT_{c_2}[2], \ldots, VT_{c_n}[n])$$

because this implies that no message sent after cut event c_i has been received before some other cut event c_j.

Termination



sending a message: ⟶

Question:

In a distributed computation, when are all of the processes idle (i.e., when has the computation terminated)?

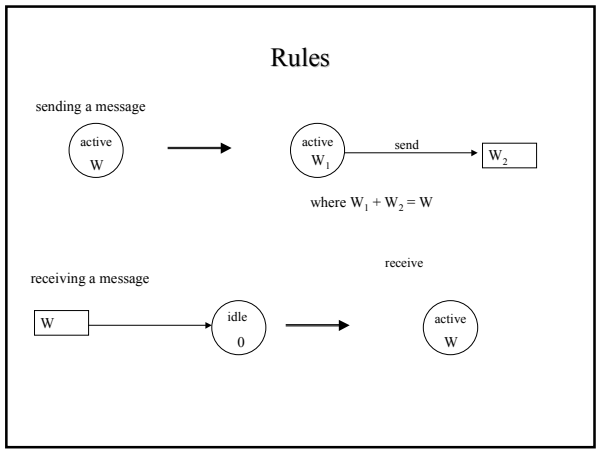
Huang's Algorithm

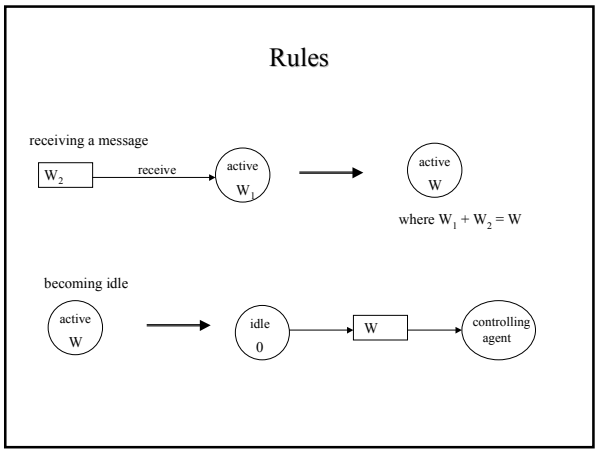
The computation starts when the controlling agent sends the first message and terminates when all processes are idle.

The role of weights:

- the controlling agent initially has a weight of 1 and all others have a weight of zero,
- when a process sends a message, a portion of the sender's weight is put in the message, reducing the sender's weight,
- a receiver adds to its weight the weight of a received message,
- on becoming idle, a process sends it weight to the controlling agent,
- the sum of all weights is always 1.

The computation terminates when the weight of the controlling agent reaches 1 after the first message.


Rules
sending a message
active
W
active
W_1
send
W_2
where $W_1 + W_2 = W$
receiving a message
receive
W
idle
0
active
W


Rules
receiving a message
W_2
receive
active
W_1
active
W
where $W_1 + W_2 = W$
becoming idle
active
W
idle
0
W
controlling
agent