The distributed snapshot of Chandy/Lamport/Misra

[The algorithm dealt with in this note was shown to me on 1 July 1983 by K. Mani Chandy and Jayadev Misra – both of the University of Texas at Austin – who gave considerable credit to Leslie Lamport – of SRI International – . The next 8 weeks or so I was otherwise engaged, then the last 2 sessions of the Tuesday Afternoon Club were devoted to it. The first session was devoted to a reconstruction of the problem and of its solution and to a summary of our proof obligations; in the second session we devised the correctness proof given below. If I have failed to do justice to the algorithm, I offer my apologies to its authors.]

We consider a finite network of sequential machines, strongly connected by uni-directional, first-in-first-out message buffers, and such that its activity can be described as a succession of atomic actions. Each atomic action takes place in one machine, may comprise the reception of one message from one of the machine's input buffers and may comprise the sending of at most one message per output buffer. Necessary initial conditions for an atomic action to take place are (i) that the machine in question is in an appropriate state, and (ii) that the buffer in question is not empty if the atomic action includes receipt of a message.

The state of the network is deemed to be fully described by the states of all machines and the contents of all buffers. For a given network a predicate is called "stable" if none of the atomic actions can falsify it, or equivalently, if it holds in all states reachable from a state in which it holds. (For example, in the now classical case of a diffusing computation the predicate "the diffusing computation has terminated" would be a stable predicate; in a token-preserving network the predicate "the number of tokens travelling on the network equals 7" would be stable.)

Finally we assume some fairness, more specifically that the system's nondeterminacy is resolved in such a manner that each message entering a buffer eventually leaves it at its other end. (Any daemons ensuring this fairness are not considered to be part of the system.)

The execution of the snapshot algorithm to be designed is merged with the network's original activity; its purpose is to collect information allowing the detection that a stable state has been reached.

To this end

(i) the repertoire of messages is extended with one new message, called "the red letter"

(ii) each machine is equipped with a bit recording whether the machine is white or red

(iii) each machine is equipped with a bit for each of its input buffers, recording whether that input buffer is white or red
(iv) the repertoire of atomic actions is extended —and modified, see later— with atomic actions for the receipt of red letters.

Initially all machines and all buffers are white. The snapshot algorithm is fired by one of the machines —the "starter"— by acting as if it has received a red letter (from "nowhere").

As far as red letter traffic and colouring is concerned, the atomic action describing the receipt of a red letter is as follows
(i) the input buffer from which the red letter is received (was white and) becomes red
— for the red letter received by the starter from "nowhere" this part of the atomic action is void —
(ii) if the receiving machine was white, it sends one red letter over each of its output buffers and becomes red, if the receiving machine was already red, it remains so and it sends no further red letters.

The above algorithm maintains the invariant that each buffer from a red machine to a white one contains precisely 1 red letter; on account of the fairness of that white machine, the latter will receive the red letter and turn red itself. Therefore, when the last machine has turned red, there is no buffer leading from a red machine to a white one;

hence, all machines reachable from the starter are in the final state red. Since all machines are reachable from the starter, all machines are red in the final state. Since each buffer is the output buffer of some machine that has turned red once, each buffer has accepted 1 red letter: when the last red letter is received, the only still white buffer becomes red. The state of the network at that moment will be referred to as "the final state (of the snapshot algorithm)"; "the initial state (of the snapshot algorithm)" refers to the state of the network just prior to the atomic action in which the starter turned red.

So much for the red letter traffic. Its sole purpose is to trigger local snapshots, according to the following rules

(i) upon turning red, each machine records its own state

(ii) each red machine records for each of its input buffers the stream of messages it receives from that buffer as long as the latter is white. (For a buffer from which a white machine received a red letter, the empty string is recorded; for all other buffers, the string ends with the red letter carried by that buffer.)

Hence, the snapshot algorithm records a machine state for each machine, and a message stream for each buffer. They together comprise what is called "the snapshot state".

Since the network is strongly connected, the record of the snapshot state can be collected in the starter, which then can investigate whether the snapshot state satisfies the stable predicate. (If not, the starter is supposed to start the snapshot algorithm a next time.)

In order to guarantee that it will be detected if the stable predicate holds, we shall prove about the snapshot algorithm

Lemma 0.
If the initial state of the snapshot algorithm satisfies a stable predicate, the snapshot state satisfies it.

In order to guarantee that no erroneous detection of stability can occur, we shall prove about the snapshot algorithm

Lemma 1
If the snapshot state satisfies a stable predicate, the final state of the snapshot algorithm satisfies it.

The two proofs are similar: in both cases we shall demonstrate that one state of the network is reachable from another. The proofs are rather operational in the sense that we shall take an arbitrary, but "fixed", execution of the snapshot algorithm as our reference, to which we shall refer as "reality"; in addition we shall consider an identically constructed shadow network in which some of the

atomic actions occurring in reality are faithfully copied, others suppressed. We shall describe this as if the two networks evolve in strict synchronism and shall define the states of the shadow network in terms of the given reality.

For the description of message streams we need some terminology with respect to the direction: a stream and a nonempty prefix of it contain the same oldest message; similarly a postfix of a stream is taken by omitting zero or more of its older messages; postfixing a stream is extending it at the end of its youngest message with still younger messages.

Proof of Lemma 0.

We have to show that the snapshot state is reachable from the initial state in reality. To this end, the shadow network starts in the same initial state as in reality and ends in the snapshot state (with all machines red).

In the shadow network we have at any moment

(i) the colour of a machine is as in reality
(ii) the state of a machine is:
    a) for a white machine as in reality,
    b) for a red machine as in the snapshot state
(iii) the contents of a buffer are:
    a) for a buffer from a white to a white machine as in reality

b) for a buffer from a red to a red machine as in the snapshot state
c) for a buffer from a red machine to a white one, from the contents in reality, the prefix up to and including the red letter
d) for a buffer from a white machine to a red one, the record of incoming messages as recorded in reality by the red machine, postfixed by the contents of the buffer in reality.

To begin with, we observe that on account of (i), (ii,a), and (iii,a), the initial state of the shadow system coincides with the initial state in reality; and we observe that on account of (i), (ii,b) and (iii,b), the final state of the shadow system coincides with the snapshot state.

The correspondence between the two processes is as follows: to an action performed in reality by a white machine corresponds in the shadow network exactly the same action, to one performed by a red machine corresponds in the shadow network a skip.

The action of a white machine is possible on account of (i), (ii,a), and (iii,a) or (iii,c); it leaves (i), (ii), and (iii) clearly invariant if the machine remains white. If it turns red, we observe
ad (i): that in both networks corresponding machines turn red simultaneously
ad (ii,b): that the snapshot is defined so as to establish the truth of (ii,b)

ad (iii, b): if the machine for which it is an input buffer turned red, (iii, b) is established on account of (iii, c) and the definition of the snapshot state; if the machine for which it is an output buffer turned red, (iii, b) is established on account of (iii, d), the definition of the snapshot state and the fact that the machine postfixed the buffer contents with a red letter.

ad (iii, c): on account of (iii, a) and the fact that buffer contents have been postfixed by a red letter.

ad (iii, d): on account of (iii, a) and the fact that the machine just turned red has recorded so far an empty string of messages.

(End of Proof of Lemma 0.)

Proof of Lemma 1.

We have to show that the final state in reality is reachable from the snapshot state. To this end the shadow network starts in the snapshot state (with all machines and buffers white) and ends in the final state in reality.

In the shadow network we have at any moment

(i) the colour of a machine is as in reality
(ii) the state of a machine is
  a) for a white machine as in the snapshot state
  b) for a red machine as in reality
(iii) the contents of a buffer are
  a) for a buffer from a white to a white machine

as in the snapshot state

b) for a buffer from a red machine to a red machine as in reality

c) for a buffer from a red machine to a white one, a postfix of the contents in reality, extending over the messages that the target machine in reality shall accept while red. (Note that if the red letter in in the buffer in reality turns its recipient red, the postfix in the shadow buffer extends up to and excluding that red letter and the snapshot string is empty; otherwise the snapshot string is not empty and the postfix in the shadow buffer extends up to and including the snapshot string.)

d) for a buffer from a white machine to a red one, reality postfixed by the messages up to and including the red letter that the white machine will enter in reality into the buffer.

The correspondence between the two processes is as follows: to an action performed in reality by a red machine corresponds in the shadow network exactly the same action, to an action performed in reality by a white machine corresponds in the shadow network a skip, except that a shadow machine turns red when its partner

in reality does so.

The action of a red shadow machine is possible on account of (i), (ii,b) and (iii,b) or (iii,d); it clearly leaves (i),(ii) and (iii) invariant. So does the skip performed by a white shadow machine despite the change in the "reality" in which (i), (ii), and (iii) are formulated.

For the switch from white to red we observe

ad (i): this is maintained by the switch

ad (ii): because the machine's snapshot state equals its current state in reality, (ii,a) prior to the switch ensures (ii,b) afterwards

ad (iii,b): if the machine for which it is an input buffer switches to red, the buffer in the shadow network has the same contents as its partner in reality on account of (iii,c); if the machine for which it is an output buffer switches to red, the buffer in reality just receives a red letter, hence the contents of the shadow buffer are as in reality on account of (iii,d).

ad (iii,c): in reality the buffer just receives the red letter, the shadow buffer remains unchanged, i.e. the snapshot string (empty or closed by a red letter). But that is what (iii,c) defines if the red letter is the youngest message in the buffer.

ad (iii,d): the machine of which it is an input buffer just turned red; (iii,d) defines the

the snapshot string as the contents of the shadow buffer; according to (1ii, a) the contents of the shadow buffer were the snapshot string prior to the switch, which has left these contents unchanged.

(End of Proof of Lemma 1.)

* * *

Remarks

I know that the above text can be shortened and polished; regrettably I also know that a very heavy schedule will prevent me from doing so in the near (and not so near) future. [One improvement is to record instead of the empty string a string containing just a red letter. The idea was to record in the snapshot the tree along which the blushing propagates from the starter over the network. But this propagation tree, though frequently mentioned during the discussions in the Tuesday Afternoon Club, has all but disappeared.]

I am very pleased by the obvious symmetry between the two proofs. [This symmetry only emerged in full force while writing this text.]

I am also pleased by my two-stage presentation of the snapshot algorithm: first the red-letter traffic, and then the recording it triggers. [The red-letter traffic is such a fundamental

process that it probably deserves to be named, described, and justified, in isolation. Then we can refer to it whatever we superpose on it.]

I think that the idea of introducing shadow networks next to reality was correct. The fact that there is not the slightest objection - as we did in the proof of Lemma 1 - to initialize the shadow system in a way that depends on the "future" computation in reality is very similar to the fact that there is no objection - on the contrary, it can often be done at great advantage! - to using the value of the "still unknown, perhaps even still undecided" answer in the formulation of the invariant. [This is another reason for <u>not</u> trying to understand computations in terms of cause-and-effect: people used to that are immensely troubled by this trick.]

I am mildly annoyed by the 4-fold case analysis under (iii) of the invariants, but only mildly since it is probably unavoidable since both source and target - the introduction of these terms would have helped! - of a buffer may have either colour.

I am profoundly annoyed, however, by my clumsy formulations of, in particular (iii,c) and (iii,d). My linguistic shortcomings are here painfully obvious; were I to rewrite this note, I would first think very hard how to

define these strings before setting a pen on paper.

Acknowledgements

My first acknowledgements are due to K. Mani Chandy, Leslie Lamport, and Jayadev Misra, the authors of this beautiful algorithm, in particular to the first one of the above triple since he took the trouble of explaining it to me. The algorithm is so beautiful because it separates the general process of taking a distributed snapshot from the particular process of testing against a specific stable predicate.

My next acknowledgements are due to the members of the Tuesday Afternoon Club. They greatly helped in taking the firm decision to ignore all relativistic worries and to consider the atomic actions in reality totally ordered in time. I think that was a wise decision.

Plataanstraat 5
5671 AL NUENEN
The Netherlands

8 September 1983

prof. dr. Edsger W. Dijkstra
Burroughs Research Fellow.