Reducing control traffic in a distributed implementation of mutual exclusion

We consider a network of machines that can send messages to each other. Each machine is in 1 of 3 states, viz.

n	for "neutrally engaged",
d	for "delayed", or
c	for "critically engaged" .

A critical engagement lasts only a finite period and is immediately followed by a neutral engagement of the machine in question. Between a neutral and the subsequent critical engagement a delay may occur in view of the requirement that at any moment at most 1 machine be critically engaged (so-called "mutual exclusion"). The implied synchronization has to be implemented in such a manner that no delay lasts forever (so-called "fairness").

We introduce a single token, either held by one of the machines or being sent from one machine to another. Mutual exclusion is then achieved by maintaining

a critically engaged machine holds the token .

The machines maintain this by (i) not initiating a critical engagement unless holding the token, and (ii) not sending the token to another machine while being critically engaged.

Furthermore each machine maintains

the machine holding the token is not delayed

by (i) skipping the delay upon termination of a neutral engagement while holding the token, and (ii) initiating a critical engagement upon receipt of the token while delayed. Fairness is therefore ensured when each delayed machine receives the token within a finite period of time.

The rest of this note deals with the control of the movement of the token. To this end the machines are arranged in a ring, the two circular directions in which are called "to the left" and "to the right" respectively. The token is sent to the left, so-called <u>signals</u> are sent to the right. Each link connecting two neighbouring machines in the ring is in 1 of 3 states, viz.

u	for "unused"
t	for "carrying the token to the left"
s	for "carrying a signal to the right" .

The latter two states are postulated to last only a finite period of time.

The idea is that a delayed machine can send a signal to the right to call the token to the left. In order to mark ring segments traversed to the right by signals but not yet traversed to the left by the token, machines on such segments will be <u>black</u>; the others will be <u>white</u>. We shall now make this precise by developing the possible states of the ring, and begin by describing the coding conventions we shall use for the description of these ring states.

The state of a link will be coded by u, t or s. The state of a machine will be coded by the colour b or w, followed by n, d, or c; the machine holding the token is identified by writing its colour with a capital letter. The state of the ring is described by a string of the appropriate length in which machine states and link states alternate; the direction "to the right" in the string coincides with the circular direction "to the right" in the ring.

Strings are given by so-called "regular expressions". They enable us to describe the initial state by

$$Wn\ u\ (wn\ u)^* \quad ,$$

i.e. a white neutral machine holding the token, an unused link, followed by zero or more occurrences — that is what the star stands for — of a white neutral machine not holding the token, followed by an unused link. The above regular expression expresses: all machines are white and neutrally engaged, all links are unused and the token resides at some machine.

My intention is to build up the invariant as the transitive closure of the successor relation, hand in hand considering, as the need arises, more (local!) successor relations and the resulting extensions of the grammar of reachable states. We shall mark the local successor relations since they have all to be catered for in the algorithm we are heading for.

The white machine holding the token may complete its neutral engagement, but in that case we have already decided upon the transition

$$\mathrm{Wn} \rightarrow \mathrm{Wc} \qquad (0)$$

which extends reachability to

$$(\mathrm{Wn}|\mathrm{Wc})\ u\ (wn\ u)^*$$

– in which, as usual with regular expressions, the "|" is used to separate alternatives – .

Alternatively a white machine not holding the token may complete its neutral engagement, in which case I propose the transition

$$wn\ u \rightarrow wd\ s \qquad (1)$$

which, since it may occur repeatedly, extends reachability to

$$(wd\ s\ (wn\ u)^*)^*\ (\mathrm{Wn}|\mathrm{Wc})\ u\ (wn\ u)^*$$

The white machine may terminate its critical engagement; I propose the transition

$$\mathrm{Wc} \rightarrow \mathrm{Wn} \qquad (2)$$

which does not extend reachability. Let us now follow the carried signal; I propose

$$s\ wn\ u \rightarrow u\ bn\ s \qquad (3)$$

which extends reachability to

$$(wd\ (u\ bn)^*s\ (wn\ u)^*)^*\ (Wn|Wc)\ u\ (wn\ u)^* \quad .$$

Now a black machine may terminate its neutral engagement, for which I propose

$$bn \rightarrow bd \qquad (4)$$

which extends reachability to

$$(wd\ (u\ (bn|bd))^*s\ (wn\ u)^*)^*\ (Wn|Wc)\ u\ (wn\ u)^*$$

with the result that the transition

$$s\ wd \rightarrow u\ bd \qquad (5)$$

does not lead to further extension of the reachability.

Comes the moment that we have to consider the signal reaching the machine holding the token. I propose

$$s\ Wn \longrightarrow t\ wn \qquad (6)$$
$$s\ Wc \longrightarrow u\ \mathcal{B}c \qquad (7)$$

which -in view of (1)- extend reachability to

$$E\ ((Wn|Wc)\,u\ |\ wd\ (u\ (bn|bd))^*\ (t\ |u\ \mathcal{B}c\ u))(wn\ u)^*$$

with $E = (wd\ (u\ (bn\ |\ bd))^*\ s\ (wn\ u)^*)^*$.

Now a black machine may terminate its critical engagement, for which I propose

$$u\ \mathcal{B}c \rightarrow t\ wn \qquad (8)$$

which does not extend the reachability.

The last expression for reachable ring states invites us to consider the effect of the token on its way to the left. I propose

$$u\ bn\ t \rightarrow t\ wn\ u \qquad (9)$$
$$bd\ t \rightarrow Bc\ u \qquad (10)$$
$$wd\ t \rightarrow Wc\ u \qquad (11)$$

The transitive closure has been reached, the last expression for reachable ring states is our strongest invariant. It leads directly to obviously desirable conclusions such as

the recipient of a signal is white
the recipient of the token is black <u>or</u> is delayed
the machine holding the token is not delayed
a critically engaged machine holds the token
a neutrally engaged black machine does not hold the token
a delayed machine does not hold the token.

The components of the algorithm follow directly from the listed transitions.

```
upon completion of a neutral engagement {(0),(1),(4)}:
  if holding the token
           → {(0)} initiate critical engagement
  [] ¬ holding the token
           → if white → {(1)} send signal to the right
                              and initiate delay
             [] black → {(4)} initiate delay
             fi
  fi
```

```
upon conclusion of critical engagement {(2),(8)}:
   if white → {(2)} initiate neutral engagement
   ▯ black → {(8)} send token to the left, become
                   white and initiate neutral engagement
   fi

upon receipt of a signal {(3),(5),(6),(7)}
   if neutrally engaged
         → if holding the token
                 → {(6)} send token to the left
           ▯ ¬ holding the token
                 → {(3)} send signal to the right
                   and become black
           fi
   ▯ ¬ neutrally engaged
         → {(5),(7)} become black
   fi

upon receipt of the token {(9),(10),(11)}:
   if neutrally engaged ∧ black
         → {(9)} send token to the left and
           become white
   ▯ delayed → {(10),(11)} initiate critical
                 engagement
   fi
```

In retrospect

The above does not reflect at all the way in which I discovered this algorithm. I saw its possibility 9 days ago in a flash; thereafter it took me the major part of two evenings to write it down, since I had still to think about the precise rôle of the black/white distinction and did not know how to demonstrate its correctness. I mentioned a number of consequences of the invariant —as I have done in this text— but did <u>not</u> write down the invariant itself; not amazingly my program contained an error. The text ended with the remark that regular expressions might provide the means for formulating the invariant.

This suggestion was explored at the next session of the Tuesday Afternoon Club. After some notational experiments we came up with a draft invariant by means of trial and error; this took $1\frac{1}{2}$ hours. That afternoon we remembered that 5 years ago I had commented —in EWD668— on a very similar solution designed by Alain J. Martin (coded in a version of Hoare's Communicating Sequential Processes for which a fair daemon had to be assumed; so many years later I definitely disliked the argument of EWD668).

This version took me the major part of two evenings and an afternoon, not in the last place because I would like to present the solution in the form of a possible —be it untrue— design history. The idea of developing the invariant by successive weakenings came while I was writing; quite likely EWD840 —which is almost the record of a true design history— acted as inspiration.

The decision to develop the invariant gradually was taken for two reason. The minor reason was that, when I showed my draft invariant to C.S. Scholten, his first reaction was to suppress a shudder. The major reason was that it reflects a disciplined way of algorithm development that probably has wider applicability.

Some remarks about this develepment are in order.

0) I could never have done it, had I not been sufficiently familiar with regular expressions. That they could be used at good advantage seems again to say something about the complexity of the programming task.

1) My notational experiments of last Tuesday have not been recorded, and the reader should be reminded that it is not always obvious how best to apply known techniques.

2) Note the order in which transitions (4) and (5) have been introduced: it definitely simplified the derivation of the next regular expression. So did the postponement of (6) and (7).

Acknowledgements

I express my gratitude to K. Mani Chandy [0] for drawing my attention to the problem, my respect for A.J. Martin [1] for having found this solution 5 years ago, and to the members of the Tuesday Afternoon Club for their comments on the previous version.

References

[0] Chandy, K.M., Private Communication (1 March 1983)

[1] Martin, A.J., Private Communication (Spring 1978)

[2] Lamport, L., "Time, Clocks, and the Ordering of Events in a Distributed System" Comm. ACM, Vol. 21, No. 7 (July 1978) 558-565

[3] Ricart, G. and Agrawala, A.K., "An Optimal Algorithm for Mutual Exclusion in Computer Networks", Comm. ACM, Vol. 24, No. 1 (Jan 1981) 9-17

[4] Carvalho, O.S.F. and Roucairol, G. "On Mutual Exclusion in Computer Networks", Comm. ACM, Vol. 26, No. 2 (Feb. 1983) 146-148

Plataanstraat 5
5671 AL NUENEN
The Netherlands

19 March 1983
prof. dr. Edsger W. Dijkstra
Burroughs Research Fellow