Self-stabilizing systems in spite of distributed control.

by Edsger W.Dijkstra *)

Abstract. In a sparsely connected system with distributed control the local rules of behaviour can guarantee within a bounded number of steps convergence of the system as a whole towards satisfying a global requirement.

Keywords: multiprocessing, networks, self-stabilization, synchronization, mutual exclusion, robustness, sharing, error recovery, distributed control, harmonious co-operation, self-repair.

CR Category: 4.32

*) Author's address: BURROUGHS
Plataanstraat 5
NUENEN - 4565
The Netherlands

Self-stabilizing systems in spite of distributed control.

The synchronization task between loosely coupled cyclic sequential processes (as can be distinguished in, for instance, operating systems) can be viewed as keeping the relation "the system is in a legitimate state" invariant. As a result each individual process step that could possibly cause violation of that relation has to be preceded by a test deciding whether the process in question is allowed to proceed or has to be delayed. The resulting design is readily --and quite systematicly-- implemented if the different processes can be granted mutually exclusive access to a common store in which "the current system state" is recorded.

A complication arises if there is no such commonly accessible store and "the current system state" must be recorded in variables distributed over the various processes and furthermore the communication facilities are limited in the sense that each process can only exchange information with "its neighbours", i.e. a small subset of the total set of processes. The complication is that the behaviour of a process can only be influenced by that part of the total current system state description that is available to it: local actions taken on account of local information must accomplish a global objective. Such systems (with what is quite aptly called "distributed control") have been designed, but all such designs I was familiar with were not "self-stabilizing" in the sense that, when once (erroneously) in an illegitimate state, they could --and usually did!--

remain so forever. Whether the property of self-stabilization --for a more precise definition, see below-- is interesting as a starting procedure, for the sake of robustness or merely as an intriguing problem falls outside the scope of this article. It could be of relevance on a scale ranging from a world-wide network to common bus control. (I have been told that the first solution shown below was used a few weeks after its discovery in a system where two resource-sharing computers were coupled via a rather primitive channel along which they had to arrange their co-operation.)

We consider a connected graph in which the majority of the possible edges are missing and with a finite state machine placed at each node; machines placed in directly connected nodes are called each other's neighbours. For each machine one or more so-called "privileges" are defined, i.e. boolean functions of its own state and the states of its neighbours; when such a boolean function is true, we say that the privilege is "present". In order to model the undefined speed ratios of the various machines, we introduce a central daemon --its replacement by a distributed daemon falls outside the scope of this article-- that can "select" one of the privileges present. The machine enjoying the selected privilege will then make its "move", i.e. is brought into a new state that is a function of its old state and the states of its neighbours; if for such a machine more than one privilege is present, the new state may also depend on the privilege selected. After completion of the move the daemon will select a new privilege.

Furthermore there is a global criterion, telling whether the system as a whole is in a "legitimate" state. We require that

1) in each legitimate state one or more privileges will be present, and

2) in each legitimate state each possible move will bring the system again in a legitimate state, and

3) each privilege must be present in at least one legitimate state, and

4) for any pair of legitimate states there exists a sequence of moves transferring the system from the one into the other.

We call the system "self-stabilizing" if and only if regardless of the initial state and regardless of the privilege selected each time for the next move, always at least one privilege will be present and the system is guaranteed to find itself in a legitimate state after a finite number of moves. For more than a year it has --at least to my knowledge-- been an open question whether non-trivial (e.g. all states legitimate is considered trivial) self-stabilizing systems could exist. It is not directly obvious whether the local moves can assure convergence towards satisfaction of such a global criterion; the non-determinacy as embodied by the daemon is an added complication. The question is settled by each of the following three constructs. For brevity's sake most of the heuristics that lead me to find them and the proofs that they satisfy the requirements have been omitted and --to quote Douglas T.Ross's comment on an earlier draft-- "the appreciation is left as an exercise for the reader". (For the cyclic arrangement discussed below the discovery that not all machines could be identical was the crucial one.)

In all three solutions we consider N+1 machines, numbered from 0 through N. In order to avoid avoidable subscripts I shall use for machine nr. i:
L: to refer to the state of its lefthand neighbour, machine nr. $(i-1) \underline{mod} (N+1)$,
S: to refer to the state of itself, machine nr. i,
R: to refer to the state of its righthand neighbour, machine nr. $(i+1) \underline{mod} (N+1)$.
In other words, we confine ourselves to machines placed in a ring (a ring being roughly the sparsest connected graph I could think of); machine nr. 0 will also be called "the bottom machine", machine nr. N will also be called "the top machine". For the legitimate states I have chosen those states in which exactly one privilege is present. In describing the designs we shall use the format: "if privilege then corresponding move fi".

a set of

Solution with K-state machines $(K > N)$.

Here each machine state is represented by an integer value S, satisfying $0 \leq S < K$. For each machine, one privilege is defined, viz.
for the bottom machine: if $L = S$ then $S := (S+1) \underline{mod}\ K$ fi
for the other machines: if $L \neq S$ then $S := L$ fi .
Note 1. With a central daemon the relation $K \geq N$ is sufficient.
Note 2. This solution has been generalized by C.S.Scholten [1] for an arbitrary network in which the degree of freedom in the legitimate state is that of the special Petri-nets called "event graphs": along each independent cycle the number of privileges eventually converges towards an arbitrary predetermined constant.

Solution with four-state machines.

Here each machine state is represented by two booleans xS and upS. For the bottom machine upS = true by definition, for the top machine upS = false by definition: these two machines are therefore only two-state machines. The privileges are defined as follows:

for the bottom machine: <u>if</u> xS = xR <u>and</u> <u>non</u> upR <u>then</u> xS:= <u>non</u> xS <u>fi</u>

for the top machine: <u>if</u> xS $\neq$ xL <u>then</u> xS:= <u>non</u> xS <u>fi</u>

for the other machines: <u>if</u> xS $\neq$ xL <u>then</u> xS:= <u>non</u> xS; upS:= <u>true</u> <u>fi</u>;
<u>if</u> xS = xR <u>and</u> upS <u>and</u> <u>non</u> upR <u>then</u> upS:= <u>false</u> <u>fi</u> .

The four-state machines may enjoy two privileges. The neighbour relation between bottom and top machine is not exploited; we may merge them into a single machine which is then also a four-state machine for which also two privileges have been defined.

<u>Solution with three-state machines</u>.

Here each machine state is represented by an integer value S, satisfying $0 \leq S < 3$. The privileges are defined as follows:

for the bottom machine: <u>if</u> (S+1)<u>mod</u> 3 = R <u>then</u> S:= (S-1)<u>mod</u> 3 <u>fi</u>

for the top machine: <u>if</u> L = R <u>and</u> (L+1)<u>mod</u> 3 $\neq$ S <u>then</u> S:= (L+1)<u>mod</u> 3 <u>fi</u>

for the other machines: <u>if</u> (S+1)<u>mod</u> 3 = L <u>then</u> S:= L <u>fi</u>;
<u>if</u> (S+1)<u>mod</u> 3 = R <u>then</u> S:= R <u>fi</u> .

Again the machine nr. i with $0 < i < N$ may enjoy two privileges, the neighbour relation between bottom and top machine has been exploited.

Acknowledgements are due to C.S.Scholten who unmasked an earlier effort as fallacious and since then has generalized the first solution, to C.A.R.Hoare and M.Woodger whose fascination by the first two solutions was an incentive to find the third one and to the referees whose comments regarding the presentation of these results have been most helpful.

[1] Scholten, C.S. Private communication.

NUENEN, 4th June 1974

prof.dr.Edsger W.Dijkstra
Research Fellow