AN ELECTION ALGORITHM FOR A MULTIPROCESSOR CONTROLLED DIGITAL EXCHANGE

G. NÉMETH

Institute of Communication Electronics,
Technical University, H-1521 Budapest

Received June 2, 1987
Presented by Prof. Dr. S. Csibi

Abstract

A multiprocessor controlled distributed digital exchange is studied in case of some failure types. The recovery from a failure is based on temporary suspension of the normal operation and the active processors elect a coordinator controlling the reconfiguration of the system. The algorithm developed for such an election is used at initial (cold) start of the system and in case of reintroducing one or more processors repaired as well. The failure environment is given and the assertions are proved verifying the algorithm described.

Introduction

The integration of communications and computer techniques makes attractive the development of multiprocessor controlled distributed digital exchanges. A pilot project to provide extensive phone and data services for widely varying number of subscribers led to the development of the system PRS (PCM Remote System) [1]. Its main features are the following: digital remote exchange concentrator with internal traffic, distribution of call processing functions, cooperation with the higher level exchange through common channel signaling, high level of modularity with respect to capacity and cost as well, extensive maintenance software and automatic reconfigurability in case of failures.

The control of a digital exchange differs from the control structures of other fields [2]. 1. High level of modularity is desirable in functions, capacity and services, 2. the effects of failures should remain as confined as possible, and 3. the accomplishment of services is not strictly required. These characteristics raise the possibility of a control system with multiprocessor structure (1. and 2.) and task queuing (3).

The use of single-channel PCM codecs and other communication-oriented LSI circuits make possible the introduction of PCM transmission principles down to the level of the subscribers. Thus the distribution of the exchange into smaller cooperating units, and the installation of these units as near to the subscribers as possible has become feasible. Such a distributed architecture, digital exchange is shown in Fig. 1.

The terminal unit PRT preprocesses the subscriber lines. To increase its availability its common functional units are separately duplicated providing flexible reconfigurability.

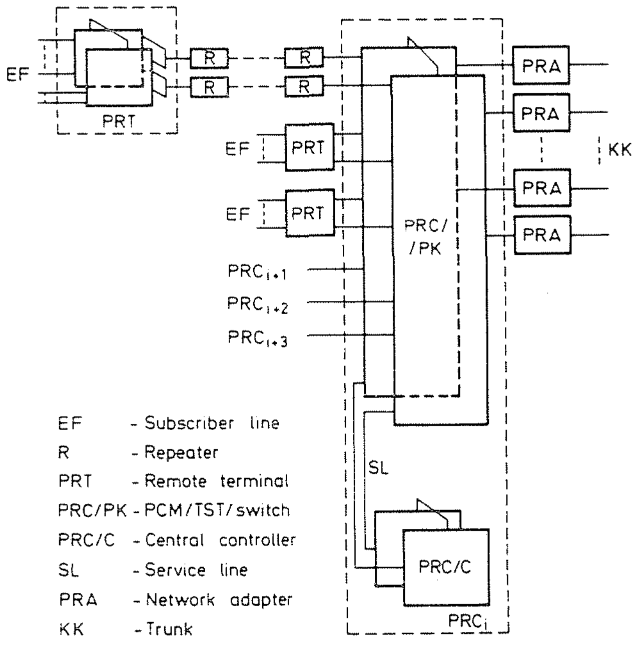

Fig. 1. Structure of the PRS (PCM Remote System) exchange

The dynamically reconfigurable switching stage PRC/PK performs time-space-time switching on 2 Mbit/s lines. The control of the switching stage and the majority of the call processing functions are performed by the control unit segment PRC/C.

The unit PRA connects the system to the higher level exchanges of the network. The man-machine communication (including the operational and maintenance functions) is provided through this unit as well.

From the point of the control structure the system may be considered as a distributed computer system consisting of autonomous nodes able to communicate with each other through virtual transmission pathes and working on a common goal. In case of one or more node failures the remaining active nodes must adapt themselves to the condition in order to be able to continue their work on their common goal.

Several strategies are possible to deal with this situation. In this paper a strategy is studied where the normal operation is temporarily suspended and then the system is reorganized. The active nodes elect a coordinator as the first step of the reorganization. This elected coordinator will control the reorganization of the system. Obviously, in a distributed computer system a distributed election algorithm is preferable. Besides, the election algorithm should preferably deal with the initial (cold) start of the system or a node, and with the reintroduction of nodes repaired after previous failures as well.

Election algorithm

Observe, please, this is a kind of the mutual exclusion problem, which can be solved by the use of a general finite state automaton. A rigorous design method may be used with the following steps:

1. Setting the ASSUMPTIONs corresponding to the system and its limitations.
2. Determining the CONSTRAINTs (the criteria of the proper solution) for the system studied.
3. Design of the FINITE AUTOMATON MODEL (internal states, events).
4. PROOF of the correctness of the algorithm chosen.

Several papers have studied such algorithms (e.g. see [3]), however, the algorithm depends upon the environment of the system. We have the following ASSUMPTIONS.

1. All nodes run the same election algorithm.
2. If a node fails it becomes inactive.
3. The communication subsystem does not fail.
4. The nodes have different integer values as their identifiers.
5. Node y processes the messages accepted from node x in the same order as they were sent by node x.
6. The message transmission delays between any node pairs are upper bounded.
7. A node sends immediately an acknowledgement of a message received.
8. All nodes send messages together with timestamps from their own clocks.
9. All nodes have their own clocks satisfying the following two conditions:
   9.1. If a and b are events in node x and a happens before b then $Cx\langle a\rangle < Cx\langle b\rangle$ ($Cx\langle a\rangle$ is the value of the clock of node x at the instant event a occurred);
   9.2. If a is the sending of a message from node x and b is the reception of the same message by node y then $Cx\langle a\rangle < Cy\langle b\rangle$.

Assumptions 1, 4, 7, and 8 are simply provided by the software of the processors. For assumption 2 the hardware and software error checking means change the failure sensed to a total crash. Assumption 9 means that each node has its own clock, running at different speeds and set to different clock values, but a form of logical clock system is implemented based upon timestamped message exchanges among the nodes [4]. Assumptions 3, 5, and 6 are provided by redundant transmission pathes dynamically reconfigurable in case of any errors (the messages are protected by proper coding) and each message has a serial number to be able to restore the proper order.

In normal operation the coordinator of the system sends periodically life messages to every active node (from its list, see below) and waits for acknowledgements. If within a fixed time period it does not receive an acknowledgement then it considers the node as a failed one and starts a new election-reorganization procedure. All other nodes wait for a life message from the coordinator for a fixed time period and if any one of them does not receive it within this period then considers the coordinator as a failed one and starts a new election-reorganization procedure.

The problem is to design an algorithm such that one can prove that there is a unique controller to be elected as a coordinator (CONSTRAINT A) in a finite time (constraint B). Unfortunately, it is not possible to fulfil constraint B, because a sequence of failures could provent any election protocol from ever electing a coordinator. So instead we require that if no infinite series of failures interfere with the election, then a new coordinator will eventually be elected (CONSTRAINT B').

The election algorithm is defined by the following four rules. Each node follows these rules independently of the others.

RULE 1:

A node whose life timer (if the node is active) or election timer (if the node is participating in an election but no node has become coordinator) awakes or the node is being (re)introduced into the system starts an election becoming a candidate.

RULE 2:

A candidate node sends timestamped candidate messages to every other nodes in the system and waits for acknowledgements. The candidate node lists the identifiers of the nodes from whom it received an acknowledgement.

RULE 3:

Upon receiving a timestamped candidate message the node sends an acknowledgement, and

a) If the receiving node is a candidate:
if the timestamp received is less than its own candidate message timestamp or equals to it but its identity is greater than that of the

source node then the receiving node drops its candidacy (changing its state to election), sets its election timer, and clears its acknowledgement list;

b) if the receiving node is not a candidate:
then changes its state to election, sets its election timer.

RULE 4:

When the candidate timer awakes the candidate node becomes coordinator and starts reorganization (knowing all working nodes from its own acknowledgement list).

Proof

In order to demonstrate that the election algorithm satisfies CONSTRAINT A, let us describe first the state-transition table of node *i*.

States of node *i*:

a. active, life timer is set;
b. election, candidate, candidate timer is set, sending timestamped messages, waiting for acknowledgement;
c. election, non-candidate, election timer is set;
d. coordinator, sending reorganizing messages;
e. coordinator in active mode (after reorganizing the system), sending life messages to active nodes;

Events:

1. life timer awakes;
2. candidate message is received with smaller timestamp (or equal but with smaller identifier);
3. candidate message is received with greater timestamp (or equal but with greater identifier);
4. candidate timer awakes;
5. election timer awakes;
6. reorganizing message is received;
7. reorganizing timer awakes.

Event / State	1	2	3	4	5	6	7
a	b	c	c	—	—	—	—
b	—	c	b	d	—	!	—
c	—	c	c	—	b	a	—
d	—	c	!	—	—	—	e
e	—	c	—	—	—	—	—

Note: The event denoted by ! may not occur.

We will use the following notation:

$I\ (x, 0)$	= instant of sending a timestamped candidate message by node x
$I\ (x, y)$	= instant of receiving by node y a timestamped candidate message sent by node x
T	= time limit for candidate timer
t	= time limit for election timer
k	= upper bound of message delay between any node pairs
R	= time limit for reorganization timer.

In order to simplify the explanation, temporarily assume that a failing node remains inactive forever, thus it will not interfere with any ongoing elections, and no new or repaired node will be (re)introduced into the system from the instant an election starts until the reorganization of the system after the election is completed.

The lower bound of the candidate timer value may be determined from the following constraint: node x, for example, will be the winning candidate; its candidate message arrives to node y with maximum time delay k, thus node y may send its own (loser) candidate message until time instant $I\,(x, 0)+k$, which may travel to node x with maximum time delay k, too. Node x may throw away this candidate message if $I\,(x, 0)+k+k<I\,(x, 0)+\ Tx\text{min}$, thus

$$Ti>2k$$

where Ti is the lower bound of the candidate timer value of node i measured by its own clock.

Suppose that node x and node y "simultaneously" become coordinators, thus violating CONSTRAINT A, and identity $(x)<$ identity (y). We will show that this situation is impossible.

Node x becomes coordinator if and only if it is in state b and event 2 does not occur between time instants $I\,(x, 0)$ and $I\,(x, 0)+Tx\text{min}$ (see second line in the state-transition table). Similarly, node y becomes coordinator if and only if it is in state b and event 2 does not occur between time instants $I\,(y, 0)$ and $I\,(y, 0)+Ty\text{min}$. Thus

$$I\,(y, x)>I\,(x, 0)+Tx\text{min}, \tag{1}$$

and

$$I\,(x, y)>I\,(y, 0)+Ty\text{min}. \tag{2}$$

However,

$$I\,(y, x)\leq I\,(y, 0)+k, \tag{3}$$

and

$$I\,(x, y)\leq I\,(x, 0)+k. \tag{4}$$

Since $Tx\min > 2k$ and $Ty\min > 2k$ from (1) and (3):

$$I(y,0) > I(x,0) + k,$$

and from (2) and (4):

$$I(y,0) < I(x,0) - k.$$

This is clearly a contradiction, thus only one node becomes coordinator.

The coordinator elected can successfully reorganize the system if and only if /2 and /3 between $I(x,0) + Tx$ and $I(x,0) + Tx + Rx$ (/i denotes non-occurrence of event i). A candidate message may be sent only by such a node, which had previously given up its candidacy, but its election timer awakes (event 5). Thus

$$I(x,0) + Tx\max + Rx\max < I(x,0) + ty\min$$

so

$$ty\min > Tx\max + Rx\max.$$

Since the nodes measure their timing by their own clocks, $ti > T + R$. However, reorganization involves sending reorganizing messages by the coordinator, and since the upper bound of message delay is k, $R \geq k$ must be ensured. From this the lower bound of $ti = 3k$.

So far it has been assumed that no nodes will be (re)introduced into the system until the end of reorganization. If one or more nodes may recover then a protection should be provided against two nodes becoming coordinators and reorganizators. More precisely, the situations denoted by ! in the state-transition table must be inhibited. This can be ensured by simply setting the clock value of the recovering node to 0 at the instant the node recovers (i.e. the timestamp of its candidate message will be 0).

Nodes are autonomous also at the initialization. No external action is needed as nodes will undertake spontaneously an election when the system is turned on.

Failure of the node which is precisely the one being elected by the other nodes as coordinator is not catastrophic; protection against infinite waiting is provided by the election timers; the election phase will only be longer than for a failure-free situation.

At any time, several nodes in the system may fail, so the coordinator at no time can know which nodes are active. This means that if the coordinator has some actions that it wishes a certain set of active nodes to perform, it cannot simply issue the necessary commands, instead it must use a two phase commit protocol to have the actions performed (or ignored) by all nodes in the set [3].

References

1. NÉMETH, G.: "Architectural problems of ISDN", Computer Network Usage: Recent Experiences (North Holland, 1986) 231—239.
2. NÉMETH, G.: "Multimicroprocessor control structures for small capacity electronic digital exchanges", 29. Int. Wiss. Koll. (Ilmenau, 1984), Vol. *2*, 187—190.
3. GARCIA-MOLINA, H.: "Elections in a Distributed Computing System", IEEE Trans. on Computers, Vol. C-31, No. 1, January 1982, 48—59.
4. LAMPORT, L.: "Time, Clocks, and the Ordering of Events in a Distributed System", Comm. ACM, *21*, 558—565 (1978)

Dr. Gábor NÉMETH H-1521 Budapest