Information Processing Letters 38 (1991) 157–161 17 May 1991

A deadlock-free communication kernel for loop architecture

P. Pramanik and P.K. Das

Department of Computer Science and Engineering, Jadavpur University, Calcutta 700 032, India

A.K. Bandyopadhyay

Department of Electronics and Telecommunication Engineering, Jadavpur University, Calcutta 700 032, India

D.Q.M. Fay

Department of Computer Science, The Queen's University of Belfast, Belfast, United Kingdom BT7 1NN

Communicated by P. Henderson

Keywords: Operating systems, parallel processing, message-passing multiprocessors, deadlock prevention

1. Introduction

In the early networks, meant for geographically distributed systems, flow control for relieving congestion and prevention of deadlock were achieved by "Isarithmic Control", "Buffer Storage Control" and other means [1,3]. Deadlock in a concurrent message-passing computer system [2,6] occurs when no message can advance through the interconnection network toward its destination because the queues at all the nodes are full [4]. The last paper also surveys contemporary works in this area.

The present paper reports a deadlock-free communication structure for a unidirectional loop of an arbitrary number of message-passing computers. The basic loop structure may however be utilized to synthesize other structures like bidirectional loop, mesh, hypercube etc. Indeed the suggested structure can be used to form the communication kernel of a generalized distributed operating system.

The method suggested here is advantageous over ordinary flow control algorithms because it does not generate any additional message traffic for its implementation. It is advantageous over the method suggested in [4], because it does not have to decompose a cyclic structure into an acyclic one explicitly via virtual links.

2. Assumptions

We consider a unidirectional loop consisting of an arbitrary number of message-passing computers connected in a manner as shown in Fig. 1. From now on we shall refer to these computers as nodes. Each node can

(a) receive a message from its previous node and either absorb it or pass it on to the next node,

(b) also generate a message which it then passes on to the next node.

We propose here an identical configuration of processes at each node which ensures the deadlock-free condition in a totally distributed manner. Each node, according to this scheme, consists of a group of four processes—a Receiver (R), a Buffer (B), a Sender (S), and a message

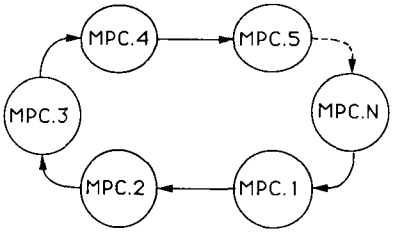

Fig. 1. A unidirectional loop of Message Passing Computers (MPC).

Originator (O), as shown in Fig. 2. We make the following assumptions:

(i) Processes are singly buffered and can, therefore, hold only one message at a time.

(ii) All buffers are of the same size, equalling that of a predefined message length.

(iii) The interarrival time of messages from the O processes are arbitrary but finite.

(iv) While the S and R processes can delay the outputting of a message that they hold by any arbitrary period of time (attributable to internal processing), the B processes present no such processing delay. They output the messages immediately as these are received at the input.

(v) Transfer of message between any pair of processes is synchronized and therefore takes place only when both the processes taking part in the transaction are ready for the transaction.

3. Systems description

The four processes, R, B, S and O, can be categorized under two basic groups—(i) those

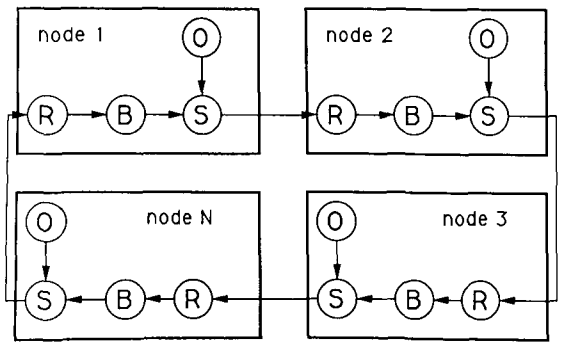

Fig. 2. A unidirectional loop of nodes each containing four processes; R: Receiver, B: Buffer, S: Sender, O: Originator.

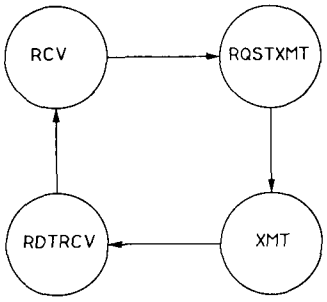

Fig. 3. State transition diagrams for groups "RT" and "T"; RCV: Receive, RQSTXMT: Request To Transmit, XMT: Transmit, RDTRCV: Ready To Receive, "RT": Receive & Transmit, "T": Transmit only.

which *receive* and then *transmit*, viz., R, B and S, that belong to the "RT" group and (ii) those which *transmit only* and do not receive, viz. O, which forms the "T" group. The two groups are characterized by the state transition diagrams as shown in Fig. 3. Note that a process can stay in any state for a finite amount of time but the transitions are instantaneous. It is also important to note that all the states except those connected with the actual act of message transmission and reception are independently controlled by the individual processes. Message transfer, on the other hand, is an activity between a *pair* of processes and assumption (v) above requires that both processes of the pair make the state transitions exactly simultaneously. A corollary of this requires, for example, that if two processes P_1 and P_2 intend to transmit messages to another process P_3, they first send a *request* for the intended transfer. The process P_3, when *ready*, accepts one of the requests depending perhaps upon the relative priorities of P_1 and P_2.

In order to bring out the true nature of the synchronized message transfer between two

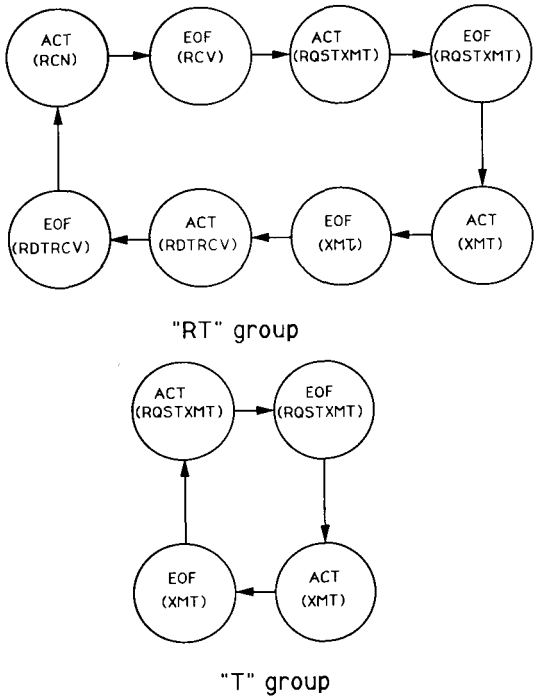

Fig. 4. Modified state transition diagrams for groups "T" and "RT"; ACT: Activity, EOF: End Of.

processes we split the states indicated in Fig. 3 into substates. For this we define two predicates ACT( ) and EOF( ). ACT(state) represents the ACTivity within the state. The EOF(state), on the other hand, is a *marker state* which marks the termination of the state. Semantically EOF(state) means the readiness of "state" to proceed to the *following* ACT(state). Note that a process only stays in the EOF(state) until all the conditions for the following ACT(state) are fulfilled. For example, a transmitting process waits in the state EOF(RQSTXMT) till the receiving process reaches the state EOF(RDTRCV) while a receiving process waits in EOF(RDTRCV) till the transmitting process reaches EOF(RQSTXMT), in accordance with our previous assumption. Here RQSTXMT and RDTRCV stand for Request_to_transmit and Ready_to_receive respectively. The modified state diagrams for the "RT" and "T" groups are shown in Fig. 4.

4. Deadlock-free communication structure

Let us assume that the time taken by process B in node m to go from the EOF(RCV) state to the EOF(RQSTXMT) state be d_{mb}.

It is then possible to ensure deadlock-free operation of a loop of message passing nodes by

(i) introducing a wait state of duration d_m between the EOF(XMT) and ACT(RDTRCV) in the S process of the mth node such that

$$d_m \geqslant d_{mb} \tag{1}$$

and

(ii) by assigning in the S process a higher priority to the input coming from B over that from O when inputs are simultaneously present from both these processes.

Definitions

We define that there is a conceptual "hole" present in a process if there is no message in it. The concept of the hole is analogous to the hole-electron concept in electronics from which the term has been borrowed. A message can be passed on to a process only if there is a hole in it. After the transfer of the message the hole passes from the receiver to the sender. Thus the direction of the hole flow is opposite to the direction of the message flow.

We set the following conditions:

(i) Messages are always ready at all the O processes to enter into the system.

(ii) There is no sink within the system.

Absence of any of these conditions in a physical system however will actually aid further in avoiding the occurrence of deadlock.

5. Proof

Let us consider that there are n nodes in the system. This means that there will be $3n$ processes in the system which form a loop. As each of the processes has a single buffer there will be $3n$ buffers in the system. As a result, deadlock will occur if and only if $3n$ messages are allowed to be present in the system simultaneously.

First we consider the situation in which the system contains $3n-1$ messages. We shall prove that the system will never allow any new message to enter into the system (from any of the O processes) and will therefore avert the deadlock situation. Let us initially suppose that another message *is* allowed to enter into the system causing a deadlock. We investigate the mth group of R–B–S processes, which has allowed this $(3n)$th input to enter into the system.

Since a new message can enter into the system from O_m through S_m, it implies that just prior to the introduction of this $(3n)$th message, S_m must have had a hole. Again because there were $3n-1$ messages prior to this introduction, each of the $3n-1$ processes must have had a message each. This in turn implies that B_m also must have contained a message.

Now a message can only be transferred to a process which contains a hole as stated earlier. So the presence of a hole in S_m implies that the immediately previous hole must have been in the R of the subsequent group of processes, i.e. the $(m+1)$th group. So, the transfer between R_m and B_m must have taken place earlier than the transfer between S_m and R_{m+1}.

In the light of the previous discussions, and the presence of a priority at the input to S_m, the system will not accept any new message if S_m and B_m are in the EOF(XMT) and EOF(RQSTXMT), respectively. Hence the $(3n)$th message will *not* enter the system and deadlock cannot occur.

So, we consider the other possible situation in which S_m and B_m are in the EOF(XMT) and ACT(RQSTXMT) states, respectively. We do not care whether this state of condition is reachable or not from the initial state of the system. Our only consideration is that this is the only possible situation in which a deadlock may occur.

Since there is no way of determining where in the ACT(RQSTXMT) state the process B_m may reside, we assume for reasons of stringency that it is about to enter the ACT(RQSTXMT) state. This in turn implies that B_m is in the EOF(RCV) state. Clearly this is the most stringent condition as far as the occurrence of deadlock is concerned.

If the time taken by process B_m to reach EOF(RQSTXMT) from EOF(RCV) is d_{mb}, and S_m is delayed by this period between EOF(XMT) and ACT(RDTRCV), B_m will reach the EOF(RQSTXMT) before S_m can enter the ACT(RDTRCV) state. Again, since a higher priority is given to the input coming from B_m over the input from O_m, no new message can enter into the system and hence the system will be free of deadlock.

Next, let us consider the situation where there are K number of holes in the system, i.e. $3n-k$ number of messages. If these K holes are distributed in exactly K number of S processes, then and then only, K new messages may enter the system, causing a deadlock. However, if sufficient delay has been introduced in each of the S processes, then none of the S processes will allow any message to enter into the system, and the total number of messages in the system will remain at $3n-k$.

However, it is possible, although very unlikely, that these K holes are distributed in exactly $(K/2)$ S processes and their corresponding B processes. Only under this condition, $K/2$ number of new messages can be introduced into the system. In other words, we need at least two holes in the system for introducing one message.

Let the number of messages in the system in any arbitrary state i be represented by N_i. Then it can be seen that

if $N_i = 3n-k$

then $\max(N_{i+1}) = 3n-k+\lfloor K/2 \rfloor$. (2)

This process of message introduction continues till the condition $k \geqslant 2$ remains satisfied. When K becomes equal to 1 the system becomes stabilized through the self-lock mechanism discussed earlier.

6. Discussion

It may be asked whether the solution could not have been achieved with only two processes R and S. The answer is in the affirmative. However, it should be kept in mind that in an actual system the process R may go into a state in which it processes the message after receiving it. This introduces an arbitrary amount of delay which may not

be easily estimated. On the other hand, the sole purpose of process B, defined as a buffer, is to receive and then transmit. Hence it is relatively simpler to calculate the time which the process B takes to inform the process S that it has a message for the latter.

We have intentionally not considered any sink in our analysis of the system in order to make it simple. This is contrary to any physical system where a message once generated reaches its destination after a finite interval of time. However, the presence of sinks aids to prevent the occurrence of deadlock by paving the way for new messages to enter into the system.

The message–transaction protocol assumed in this paper matches with the one defined in Occam and the communication structure presented here was implemented with Transputers. On experimentation the system was found to run completely free of deadlock for a wide range of message interarrival times starting from zero. This has been reported in [5].

7. Conclusion

In this paper we have shown how a unidirectional loop structure may be made free of deadlock with the help of three processes in each node of the system, the introduction of priority, and the incorporation of a certain delay. We do not, however, claim this delay to be optimum for the prevention of deadlock.

As has been stated earlier, three types of congestion–control algorithms are in existence. The one reported here solves the problem of deadlock in a manner which is totally distributed. It is advantageous over isarithmic control, because the empty packets in the latter also generate traffic and overhead. It is advantageous over the method suggested in [4], because it does not have to decompose a cyclic structure into an acyclic one explicitly via virtual links.

It is also obviously at an advantage over end-to-end flow control, because there is no need to reserve buffers in the receiver which also contributes to system overhead.

This structure behaves as if a predetermined number of messages $(3n - 1)$ may reside in the system at any point in time. Whenever the number of messages reaches that value, no more inputs are accepted. But the structure is self-regulatory and the regulation occurs automatically without any explicit communication taking place among the nodes. This self-regulatory mechanism prevents the occurrence of deadlock.

The structure ensures the presence of at least one hole in the system. The presence of this hole causes a temporal ordering of the processes forming the loop and implicitly converts the loop into an acyclic structure, thus preventing the occurrence of deadlock.

This basic technique can be used to synthesize other structures such as bidirectional loop, mesh, hypercube etc. and may form the communication kernel of a distributed operating system. This will be reported in a future paper.

References

[1] V. Ahuja, *Design and Analysis of Computer Communication Networks* (McGraw-Hill, New York, 1982).

[2] W.C. Athas and C.L. Seitz, Multicomputers: Message Passing Concurrent Computers, *IEEE Computer* (August, 1988) 9–24.

[3] C.G. Bell, A. Newell and D.P. Siewiorek, *Computer Structures: Principles and Examples* (McGraw-Hill, New York, 1982).

[4] W.J. Dally and C.L. Seitz, Deadlock-free message routing in multiprocessor interconnection networks, *IEEE Trans. Comput.* **C-36** (5) (1987).

[5] P.K. Das and D.Q.M. Fay, Performance studies of multitransputer architectures with static and dynamic links, in: *Proc. Euromicro '88 Conf.*, Zurich, Switzerland (1988) 281–289.

[6] C.L. Seitz, W.C. Athas, W.J. Dally, R. Faucette, A.J. Martin, S. Mattisson, C.S. Steele and W.K. Su, *Message-Passing Concurrent Computers, Their Architecture and Programming* (Addison-Wesley, Reading, MA, 1986).