A Communications Operating System for the Homogeneous Machine

13 January 1982

Erik DeBenedictis

The research described in this document was sponsored by the Defense Advanced Research Projects Agency, ARPA Order number 3771, and monitored by the Office of Naval Research under contract number N00014-79-C-0597.

Ng.
4707

1. Introduction

What communications primitives will be available to programs running on the homogeneous machine? One requirement is that the CPU overhead be low when the communications is simple (e.g. systolic). On the other hand, the communications must be general enough to support dynamically allocated processes (e.g. COPE). If a general communications strategy existed that was compatible with these requirements it could be implemented as part of the ROM resident operating system.

2. Types of Problems and Necessary Communications Capabilities

A spectrum of communications strategies have been proposed. Each strategy trades simplicity and efficiency for capability. These are listed below:

Systolic: In systolic communication both the sender and the receiver must be waiting on the same communication event for the event to proceed. Systolic communication is modeled by two monitor calls: one to send a message over a particular channel, and one to receive a message. The characteristics of the calls is that they hang until function can be completed, locking out any other processor activity. No interrupts are required.

Processor Directed: In processor directed communication each message is accompanied by a specification of the processor that is to receive it. When a message enters a processor that message is either delivered to the program running in that processor or relayed to another processor. If relaying must occur the operating system decides which link is most appropriate to forward the message. Queueing can be implemented for relayed messages and/or for messages destined for the processor. Interrupts are required.

Process Directed: In process directed communication there may be more than one process in each processor. Furthermore, processes may move around between processors. Communication is directed to a particular process, even though the processor where that

process is currently resident may not be known. Implementation of this scheme is very involved.

3. Processor Directed Communication

Processor directed communication offers extremely high speed and sufficient generality to be the best choice. The communications primatives appropriate for this type of communication are shown below:

int outA(Dest) Specifies a message to another processor. Dest is the destination processor (0-63).

int outB(Buf,Len) Sends message data to the selected processor. Buf is pointer to an integer array that contains the information to be transmitted. Len bytes starting at Buf are transmitted. The value returned is 0 if the transmission was successful or -1 if output buffer space was full.

int inA() Receives a message. The value returned is the identification of the sending processor, or -1 if no message was available.

int inB(Buf,Len) Receives a message. Buf is a pointer to a Len word block where the message is deposited. If the remaining message is longer than the buffer, the buffer is filled with as much of the message as possible and 0 is returned. If the message fits in the buffer, the number of bytes transferred is returned.

4. Deadlock

Deadlock can potentially occur for two reasons: 1) the user program assumes too much queueing, or 2) the message passing system deadlocks. The following rule will prevent a user program from deadlocking:

User program deadlock prevention rule:

A user program may not wait on a failed out call unless in calls are processed during the wait.

The message passing system uses a deadlock free routing algorithm. This algorithm is described below (refer to figure 1 for notation):

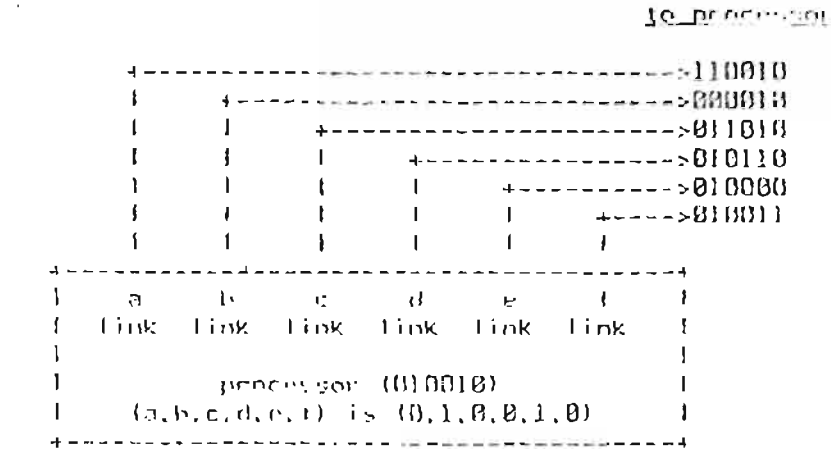

Figure 1: Notation Concerning Processors in a Hypercube

- Let processors be identified by their coordinate in the hypercube: (a.b.c.d.e.f). The letters a.b.c.d.e.f are either zero or one, and are transmitted as a binary number abcdef.

- Let the links to other processors be labeled a, b, c, d, e, and f. The labeling is chosen to correspond to the bit labeling of the previous paragraph. (I.e. the processors on either end of link c have identifications that differ only in the c bit.)

- When a message is available from a link the identification of its destination processor is analyzed before it is removed from the input queue. Analysis consists of XORing the destination processor identification with the identification of the processor with the message and performing one of two functions:

    1. If the identifications are the same then the message is destined for that processor. The message is made available for input. If the input buffer is full, the operating system runs the user program until that is no longer so.

    2. If the identifications are different then the message must be forwarded. The link to forward is determined by the following algorithm: the leftmost bit in the difference is located and that determines the forwarding link.

This system does not deadlock for the following reason: If a message arrives on

link f (least significant bit) then it must be destined for that processor. This is true because to be forwarded to link f by the previous processor all other bits must match. The user program on the processor must remove the message eventually. The system therefore cannot deadlock with messages in link f.

Now consider link e. If a message has arrived in link e, bits a-e must match. If bit f matches then the message can be made available for input and there is no problem. If bit f does not match the message is forwarded to link f. Since the system cannot deadlock with a message in link f, it will not deadlock in this condition. Therefore the system will not deadlock with a message in link e or f.

By induction, it has been proven that all links to the right of link x do not deadlock. If a message arrives on link x then it is known that all bits to the left and including x match. Therefore the message will either match exactly, or it will be forwarded to a link to the right of x. If an exact match occurs, the message is consumed by the user program. If the message is forwarded deadlock cannot occur because all links to the right of x do not deadlock. Therefore, link x does not deadlock.