An Algorithm for Detecting and Resolving Store-and-Forward Deadlocks in Packet-Switched Networks

CHEUNG-WING CHAN AND TAK-SHING P. YUM, SENIOR MEMBER, IEEE

***Abstract*—Freedom from store-and-forward (S/F) deadlocks in a packet-switched network can be guaranteed with the use of deadlock avoidance protocols. However, these protocols put so many restrictions on the use of buffers that even under normal circumstances the buffer utilization is small.**

We propose instead a deadlock detection and resolution algorithm that is completely invisible under normal circumstances. As soon as certain channels in the network have trouble in accepting and transmitting packets due to the lack of buffers, the deadlock detection phase of the algorithm is invoked. When a deadlock is identified, the deadlock resolving phase of the algorithm is executed. Once the deadlock is resolved, the control is removed. The algorithm can be used in conjunction with either the complete partitioning or the sharing with maximum queue lengths output buffer allocation strategies. A proof on the correctness of the algorithm is given. Simulation results show that the network can maintain a relatively high throughput even when deadlocks are being detected and resolved. In addition, several properties of deadlocks are shown: i) deadlocks start to increase abruptly once the network operates beyond its capacity; and ii) under heavy load conditions, increasing the buffer pool size will not delay the occurrence of deadlocks.

I. INTRODUCTION

DIRECT and indirect S/F deadlocks [1] in a packet-switched network, if left unattended, can result in the entire network becoming inoperative. Several techniques have been developed to resolve this problem [2]–[7].

Merlin [2] proposed a structured buffer pool technique for avoiding S/F deadlocks. This technique is easy to implement and has an implicit hop-level flow control [8]. However, as the network size grows, this technique requires a drastic increase of buffers at each node. Recently, Wimmer [3] proposed the use of "barrier graph" on the structured buffer pool technique and Gopal [4] proposed an improved version of the same technique that has the advantage of requiring fewer reserved buffers.

Another interesting solution by Gelernter [5] attempts to prevent S/F deadlocks by a flow control procedure. This procedure has the advantages of 1) no restriction in the routing of packets, 2) no partitioning of the buffer pool, and 3) the size of buffer pool at each node being independent of the network's size. Unfortunately, as network congestion increases, this algorithm requires rerouting of packets and, in the worst case, it may even lose some of them. Another deadlock-resistant flow control procedure is proposed by Blazewicz [6].

An algorithm based on the detection and removal of deadlocks was proposed by Gambosi [7]. Gambosi pointed out two important criteria that a deadlock control algorithm should fulfill.

1) Deadlock detection should be performed in a distributed fashion since any form of centralized control would certainly result in unreliable and inefficient operation of the algorithm.

2) The algorithm should exhibit negligible overhead for nodes not involved in a deadlock. In other words, normal network operation should not be affected by any kind of deadlock control traffic.

He then proposed a two-part algorithm: one for deadlock detection and the other for deadlock recovery. Deadlocks can be detected by constructing a site blocking graph (SBG). Once a deadlock is detected, then based on the SBG, a distributed deadlock recovery procedure is applied to resolve the deadlock. Nevertheless, the whole algorithm relies on a SGB whose construction is quite time-consuming.

In this paper, a different deadlock detection and resolution algorithm is proposed. The algorithm, first based on a complete partitioning (CP) output buffer allocation strategy [9] and later extended to the sharing with maximum queue lengths (SMXQ) strategy, not only satisfies the Gambosi criteria, but also shares the three advantages of the Gelernter algorithm. Moreover, packets will not get lost with this new algorithm.

In the following, we first introduce a network model (Section II). We then describe the deadlock control algorithm for the CP strategy (Section III). This is followed by a correctness proof of the algorithm (Section IV). A modified deadlock control algorithm for the SMXQ strategy is then introduced (Section V). Finally, using simulation, the performance of the algorithm is determined and some interesting properties of deadlocks are presented.

II. NETWORK MODEL

Consider a S/F packet-switched network with reliable channels. Let all inputs to this network be fixed-size packets, each occupying one unit of buffer space.

A typical model of a S/F node with Complete Partitioning buffer allocation strategy is depicted in Fig. 1. There is a nodal processor to handle all internal transmission of packets. One buffer is reserved for each input channel. Since the nodal processor will process and move received packets immediately upon their arrival, there is always room for another packets at the input channels. All output channels are modeled as first-in-first-out (FIFO) queues. A special output buffer called the reserved buffer is permanently allocated at each output channel for deadlock resolution purposes.

Static routing is used. This means that all packets from the same source-destination pair follow the same path. For each packet transmitted, a positive or negative acknowledgment will be received depending on whether or not it is accepted. Negative acknowledged packets are retransmitted and no new packets are transmitted until the present one is successful.

The deadlock control algorithm is executed by the channel processors, each of which works independently. An output channel can be in one of the following three states: 1) Normal

Paper approved by the Editor for Communication Switching of the IEEE Communications Society. Manuscript received January 22, 1986; revised November 5, 1986.

The authors are with the Department of Electronics, The Chinese University of Hong Kong, N.T., Hong Kong.

IEEE Log Number 8715688.

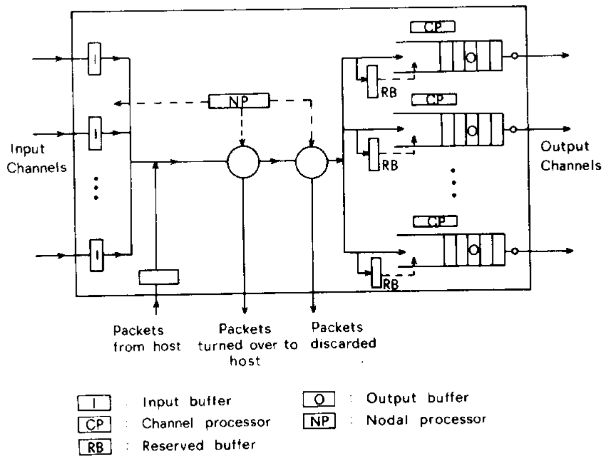


Fig. 1. Model of an S/F node.

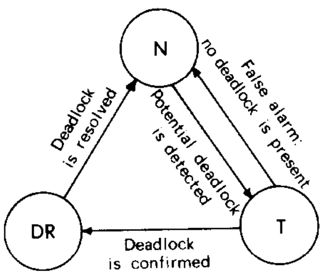


Fig. 2. State transition diagram for CP strategy.

state (N), 2) Test state (T), and 3) Deadlock-resolving state (DR). A state transition diagram is depicted in Fig. 2.

Depending on the channel state, outgoing packets are attached with different headers:

1) Normal header. It has a normal header identification, source and destination node addresses and a header checksum.

2) Test header. Besides a test header identification and all the normal header information, it also includes i) an x field indicating the total number of DR packets (packets with DR headers attached) to be transmitted and received upon detecting a deadlock and ii) an I field recording a set of channel identities in state T.

3) DR header. It contains x, a DR header identification and all the normal header information.

Upon receiving a packet from an adjacent node, the node processor will check whether the packet is destined for the present node or not. If it is, the processor will immediately turn the packet over to the local host. Otherwise, the packet will be routed to one of the output queues, say queue i. If queue i is full, then except for the DR packets which are put into the reserved buffer associated with queue i, all other kinds of packets are discarded after extracting the header information and a negative acknowledgement is sent back.

Note that a packet, once stored in a buffer, does not have to be physically moved. Movement of packets depicted in Fig. 1 may be accomplished by changing software pointers. Similarly, packets from the local host are accepted if queue i is not full.

III. The Deadlock Control Algorithm with CP Strategy

Conceptually, a S/F deadlock refers to the situation where there is a cycle of buffer requests among a set of nodes, all of which have no empty buffers left. Our algorithm is based on detecting the presence of these cycles and then resolving them efficiently. We will neglect, in our model, those packets destined for the local node as they will be turned over to the host immediately and will not impose a demand on the output buffer pool.

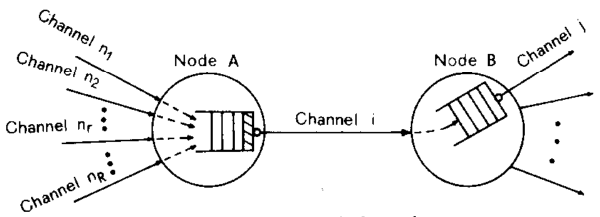


Fig. 3. A typical channel.

In the following discussion, we focus on a typical one-way channel, say channel i, that can transmit packets from node A to node B (Fig. 3). Let channels n_1, n_2, $\cdots$, n_R be the set of input channels to node A. Note that all these channels identifiers are unique throughout the network.

Here for channel i, the algorithm requires three control parameters: i) an integer y denoting the total number of DR packets to be transmitted and received by each channel in state DR before returning to state N, ii) an array $S_x = [s_x(1), s_x(2), \cdots, s_x(R)]$ where $s_x(r)$ records the integer x collected from channel n_r, and iii) a set of channel identities S_I. When channel i is in state N, these parameters are set to

$$y=0 \tag{1}$$

$$S_x=[s_x(1), s_x(2), \cdots, s_x(R)] = [M, M, \cdots, M] \tag{2}$$

$$S_I=\{\ \} \tag{3}$$

where M is an integer larger than the output buffer pool size of channel i.

Normally, channel i is in state N. It will change to state T if a potential S/F deadlock is detected. If it is a false alarm, then channel i will go back to state N. Otherwise it will change to state DR which, after the deadlock is resolved, goes back to state N. Therefore, the algorithm consists of three procedures, one for each of the three states.

A. Procedure for State N

In state N, all normal and test packets received are placed in the output buffer queue of channel i. But if the queue is full, the received packets are discarded.

Channel i will change from state N to state T if two conditions are satisfied: 1) the buffer queue of channel i is full and 2) the head packet (the packet in the first position of the output queue) has waited at the head of the queue for longer than a timeout period, say T_{out} s.

Comments: We declare that a potential S/F deadlock involving channel i is detected when channel i cannot receive and transmit any packet in a finite time T_{out} s.

B. Procedure for State T

In state T, channel i will discard all normal packets received. When a test packet is received from, say channel n_r, the channel i processor will discard the packet body and extract the x and I fields from the packet header. It then checks if its channel identity i is in the I field. If so, channel i will declare the detection of a deadlock, change its state to DR and set y to x. If i is not found in I, $s_x(r)$ and S_I are updated as follows:

$$s_x(r) \leftarrow x$$
$$S_I \leftarrow S_I \cup \{r\}.$$

The receipt of a DR packet indicates that channel i is already involved in a deadlock. The DR packet will be accepted and placed in the reserved buffer of channel i. Also y

is set to x which is obtained from the DR packet header. Channel i then changes its state from T to DR.

When channel i is in state T, only test packets are transmitted. If the head packet is to be routed to channel j (Fig. 3), then the x and I fields in its header are set as

$$x = \min\,[k_{ij}, s_x(1), s_x(2), \cdots, s_x(R)] \qquad (4)$$

$$I = S_I \cup \{i\} \qquad (5)$$

where k_{ij} denotes the total number of packets in the output buffer of channel i to be routed to channel j. Once a new test header is created, S_x and S_I are reset according to (2) and (3) (i.e., all their entries are used once only).

On the other hand, when the head packet transmitted by channel i is accepted by channel j, channel i will change from state T to N.

Comments: As mentioned before, the sufficient conditions for the existence of a deadlock are a) the presence of a cycle of buffer requests and b) all buffers in that cycle being full. It is equivalent to having a set of channels, all in state T, forming a loop as depicted in Fig. 4. All channels in that loop will transmit test packets. The I field in the headers of these packets will, according to (5), gradually accumulate the identities of these channels. Sooner or later, one or more channels in that loop will receive test packets with their own identities included in I. If that happens, these channels realize immediately that they are involved in a deadlock and change their state from T to DR. Moreover, they will, in turn, transmit DR packets to inform the other channels in the loop that a deadlock is present.

C. *Procedure for State DR*

At state DR, all normal and test packets received are discarded. When a DR packet is received, it is placed in the reserved buffer of channel i and joins the end of the output queue. When a copy of a DR packet from channel i is accepted by the neighboring node, the buffer storing the DR packet is freed and treated as the reserved buffer for receiving other incoming DR packets.

Note that only those packets having the same destination channel as that of the head packet are selected as DR packets to be transmitted and there are always y or more of these in the output buffer. The x fields in these DR packet headers are all set to y. Other packets remain in the queue until the deadlock is resolved.

After y DR packets are transmitted and received, channel i will change back to state N.

Comments: During the process of resolving a deadlock, the number of DR packets to be forwarded and received is y which is, after repeated use of (4), equal to $\min\{k_{ij}, k_{jk}, k_{kl}, \cdots, k_{n_r i}\}$ where $i, j, k, l, \cdots, n_r$ are the channel identities of the closed loop.

Meanwhile, since DR packets will only be transmitted to those channels involved in a deadlock, channel i in state N will not receive DR packets.

Figs. 5–7 show the flowcharts for the above procedures.

IV. Proof of the Distributed Algorithm

To prove the correctness of the distributed algorithm, we first model the S/F network as a directed graph described as follows.

Directed Graph (DG) Model: A transmission channel i, in state T or in state DR, with the head packet to be routed for another transmission channel j can be represented by vertex i, denoted as V_i, with an outgoing edge directed towards V_j. For channel i in state N, it is represented as V_i with no outgoing edge. To illustrate, Fig. 8 shows three cascaded channels i, j, and k in states DR, T and T, respectively.

Based on the DG Model representation, we can, at any moment, use a directed graph to model the state of a S/F

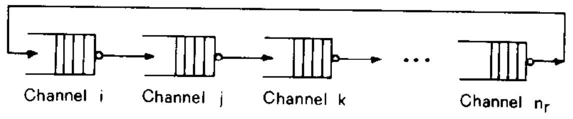


Fig. 4. A closed loop of channels.

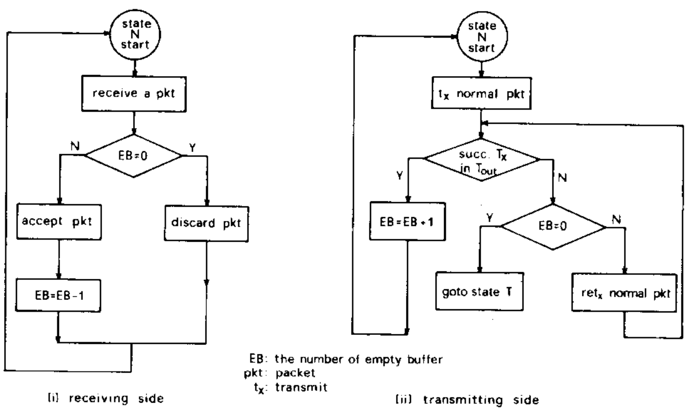


Fig. 5. Flowchart for state N.

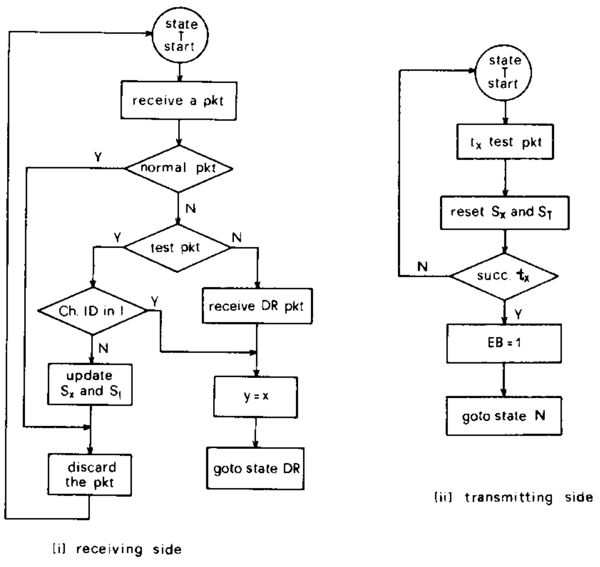


Fig. 6. Flowchart for state T.

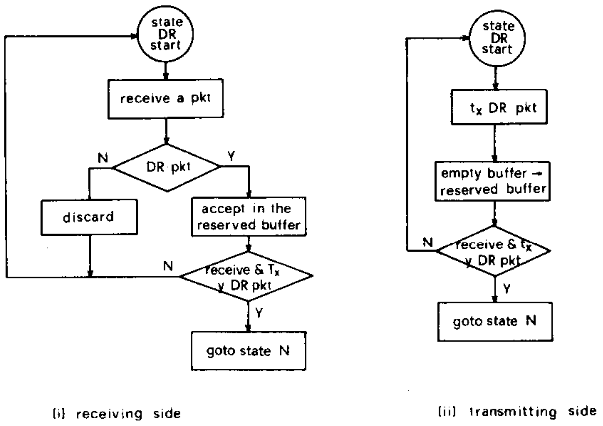


Fig. 7. Flowchart for state DR.

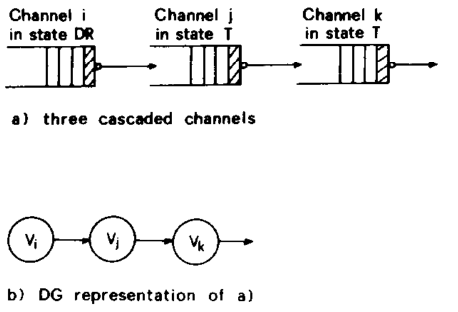


Fig. 8. A DG model representation.

network. Note that each vertex can have at most one outgoing edge, but can have several incoming ones. Such a directed graph shows only those packet transmissions that are relevant to the deadlocks. Those not relevant are neglected.

Lemma 1: Different closed loops in a directed graph are vertex disjointed.

Proof: Suppose the closed loops L_1 and L_2 have a common vertex, say V_i. Then V_i must have two outgoing edges: one belongs to loop L_1 while the others belongs to loop L_2. But it contradicts our DG model representation of a vertex which has at most one outgoing edge. By a similar argument, we can disprove the existence of two or more common vertexes in three or more closed loops.

Lemma 2: An S/F deadlock exists in a network if and only if a closed loop exists in the corresponding directed graph.

Proof: As mentioned in Section III, a deadlock exists when there is a cycle of buffer requests. This condition is revealed when a set of channels in state T have formed a closed loop (refer to Fig. 4). Then by using the DG model representation, it immediately yields the corresponding closed loop in the directed graph. Q.E.D.

Based on the preceding lemmas, Fig. 9 depicts the general situation for the presence of a deadlock in the directed graph. Note that V_i, V_j, V_k, $\cdots$, V_l, V_m, $\cdots$, V_n form a closed loop L and that V_w, V_x, V_y, $\cdots$, V_z form a path P.

Theorem 1: Every vertex in the closed loop L can detect the deadlock and enter into state DR while vertexes not in loop L will not.

Proof:

1) Vertexes in the closed loop L. When the closed loop L is formed, all its vertexes are in state T. Let us assume that all their states remain unchanged and consider the detection of a deadlock by V_i. First V_i forwards its test packets to V_j and receives test packets from V_n and V_z. After this interchange of packets, the test packets sent by V_i will have their headers' I field set to $\{\cdots, n, z, \cdots\} \cup \{i\}$. As the algorithm is distributed, every vertex in loop L does the same and eventually, V_i will find its own identity in I of a test packet received from V_n. By the procedure for state T, V_i will change to state DR indicating the detection of a deadlock.

Next, let V_l be the first vertex to detect the deadlock in the path Q from V_l to V_i and consider V_i for its deadlock detection. According to the procedure for state DR, V_l will send DR packets to V_m. Upon receiving the first DR packet, V_m will (by the procedure of state T) change to state DR and start transmitting DR packets. This transmission of DR packets and change of state then propagate along the path Q and finally, V_i will receive a DR packet from V_n to indicate the detection of a deadlock.

2) Vertexes not in the closed loop L. Consider the vertex in path P which is not in loop L but has a directed edge, originated at V_w, toward loop L. All these vertexes must be in state T when path P is first formed. By Lemma 1, all the loops formed must be vertex disjointed. Therefore, path P cannot be the segment of any loop. So even though all vertexes in path P forward test packets with their own identities inserted in the I

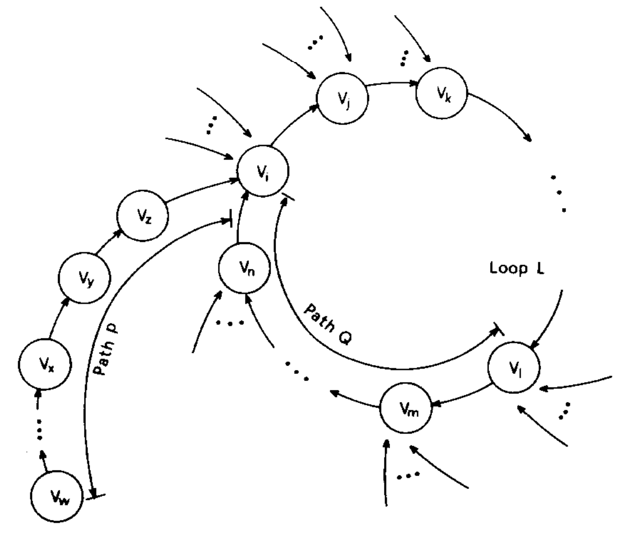


Fig. 9. A general closed loop.

field of the packet headers, they will not receive test packets with their identities in I. In addition, these vertexes will not receive DR packets because none of them can enter into state DR. Q.E.D.

Note that during the detection of a deadlock, there must be at least one vertex which enters into state DR by the successful search of their node identities in I of the receiving test packets. These vertexes will forward DR packets to inform the next vertexes for the existence of a deadlock.

Theorem 2: The transmission of DR packets is deadlock-free.

Proof: When the vertexes of a closed loop detect a deadlock, their associated reserved buffers are always ready for receiving DR packets. Since in state DR, only DR packets can propagate in loop L, the transmission of DR packets is deadlock-free. Q.E.D.

Theorem 3: By the time a deadlock corresponding to loop L is resolved, all test packets generated by the vertexes of a loop L are discarded.

Proof: Consider V_i in state T which forwards all its test packets to V_j. Since V_j must be either in state T or in state DR, all test packets received are discarded. Moreover, V_j in state DR will make a possible change of state to N only when a DR packet is received from V_i. But V_i is in state DR and will not generate test packets. Hence, when V_j returns to state N, all test packets forwarded from V_i to V_j for the search of this deadlock are discarded. Q.E.D.

We can conclude from Theorems 1, 2, and 3 that:

a) Only those output channels involved in a deadlock can detect the deadlock.

b) This deadlock can be resolved.

c) By the time when the deadlock is resolved, all test packets generated for the search of this deadlock are discarded; and so will not interfere with the detection of other potential deadlocks.

VI. The Deadlock Control Algorithm with SMXQ Strategy

We now extend the algorithm so that the output buffers allocation strategy is SMXQ. Let B_i be the total number of packets on channel i and b be the maximum queue size allowed for each channel. Besides states N, T, and DR, a new wait state, W, is needed for the modified algorithm. The state transition diagram for the modified algorithm is shown in Fig. 10. The following are the procedures of each state for a typical channel, say channel i.

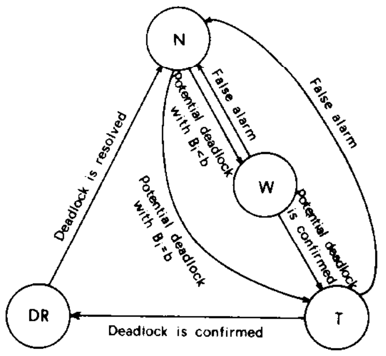

N
W
T
DR
False alarm
False alarm
Potential deadlock with $B_i < b$
Potential deadlock with $B_i = b$
Potential deadlock is confirmed
Deadlock is resolved
Deadlock is confirmed

Fig. 10. State transition diagram for SMXQ strategy.

A. Procedure for State N

At state N, all normal and test packets received are placed in the output buffers of channel i. If $B_i = b$ or there is no empty buffer in the common buffer pool (CBP), the received packets are discarded.

Channel i will change to state T if (1) $B_i = b$ and (2) the head packet has waited in the output queue for T_{out} seconds. Channel i will change to state W if 1) $B_i < b$, 2) a request for an empty buffer from the CBP fails, and 3) the head packet has waited for T_{out} seconds.

B. Procedure for State W

When channel i receives a packet and an empty buffer is successfully allocated from the CBP, the packet is accepted and channel i will change back to state N. If no empty buffer is available, the received packet is discarded.

All outgoing packets in this state are attached with normal headers.

Once channel i is in state W, it checks whether all other outgoing channels in the same node are in states T or W. If they are, channel i will change its state to T and trigger all other channels in state W to change to state T.

C. Procedures for State T and State DR

A channel in state T or DR is not allowed to request buffers from the CBP for incoming packets. Besides that, the procedures are the same as that for the CP strategy in Section III.

Comments:

1) to account for the additional restriction in the procedures for states T and DR, consider the deadlock loop in Fig. 4. Let us say there is no such restriction as described in Subsection VI-C. Let channel j, in state T or DR, obtains an empty buffer from the CBP. Then channel j may have a chance of accepting a test packet from channel i. Doing so will allow channel i to go back to state N, and thus a deadlock cannot be detected.

2) Since all outgoing packets of channel i are still attached with normal headers, a channel i in state W is represented, similar to state N, by V_i with no outgoing edge. Thus, the directed graph created is the same as that for the CP case. The correctness proof of the algorithm therefore is also the same.

VII. Simulation Results

Fig. 11 shows an eight-node network connected by eleven homogeneous full-duplex links. Each link is modeled as a FIFO $M/D/1$ queue with one reserved buffer permanently allocated and other buffers allocated according to the CP or SMXQ strategy. Let the processing time, the packet transmission time and the timeout period T_{out} be 0.01, 1, and 3 time units, respectively. The shortest path routing rule is used. The input traffic is homogeneous with all r_{ij} (traffic rate from node i to node j) equal to constant r. Each simulation run lasts for 11 000 time units with the data from the first 1000 time units discarded.

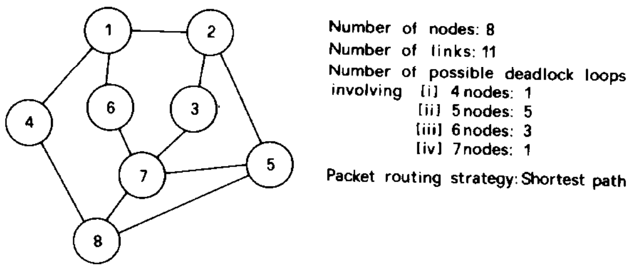

1
2
3
4
5
6
7
8
Number of nodes: 8
Number of links: 11
Number of possible deadlock loops involving (i) 4 nodes: 1
(ii) 5 nodes: 5
(iii) 6 nodes: 3
(iv) 7 nodes: 1
Packet routing strategy: Shortest path

Fig. 11. Network I.

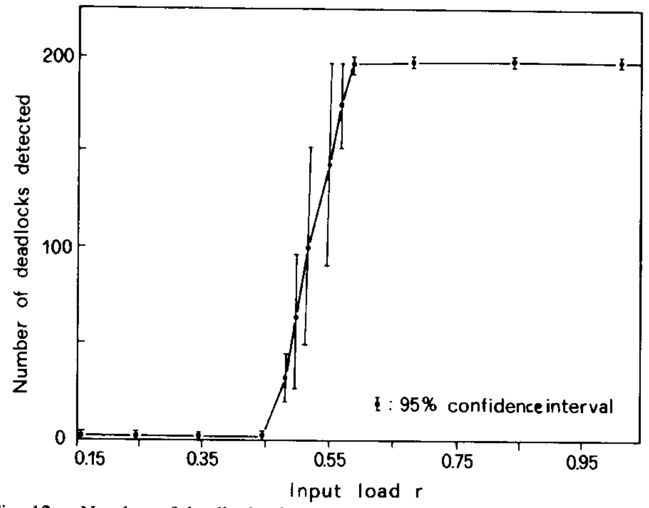


Fig. 12. Number of deadlocks detected versus the input load for buffer size = 4.

In Fig. 12, we plot the number of deadlocks that occurred against input load r with output buffer size pool equal to four. The CP strategy is used. It is observed that deadlocks rarely occur under normal traffic condition ($r < 0.45$). But once beyond the network capacity ($r > 0.58$), the average number of deadlocks detected increases abruptly to about 200. In between ($0.45 < r < 0.58$) the number of deadlocks that occurred has a very large variance. Fig. 13 shows the case with buffer size equal to five. We observe that the curve is similar except that the high variance region is shifted to $0.59 < r < 0.75$.

Fig. 14 shows the network throughput under normal traffic condition ($r < 0.45$). The output buffer for each channel is four. Here we show three curves: Curve C shows the network throughput with no deadlock control algorithm implemented while Curves A and B represent the throughput with the deadlock control algorithms implemented with the SMXQ and CP strategies, respectively. It is readily seen that very high network throughput can be maintained when the network is not saturated. But for Curve C, the network breaks down. In addition, the SMXQ starts to give slightly higher throughput than the CP at $r > 0.4$. When one-third of the channels are loaded with twice the amount of input (i.e., under asymmetric traffic condition), similar results are found as shown in Fig. 15.

Fig. 16 shows the time for the first occurrence of a deadlock (starting from an empty system) versus the input load using the CP strategy. Under heavy loading, we see that the first deadlock occurrence time is nearly a constant, independent of the number of buffers available at each channel. On the other hand, under moderate traffic condition, increasing the buffer size pool can indeed delay the occurrence of deadlock. When the traffic is very light, deadlock is still very unlikely to occur even with a very small buffer pool size.

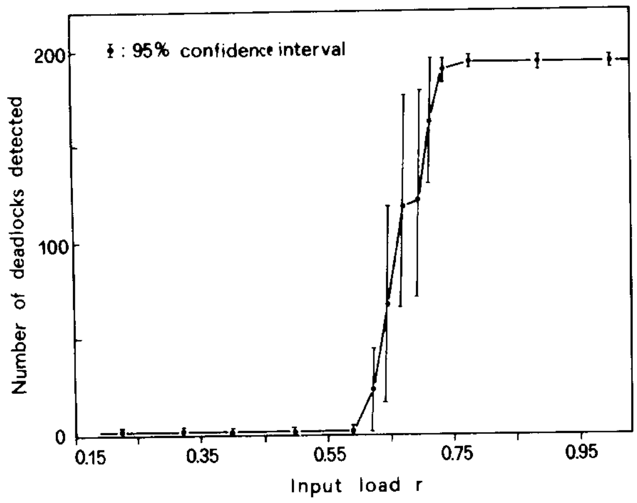


Fig. 13. Number of deadlocks detected versus the input load for buffer size = 5.

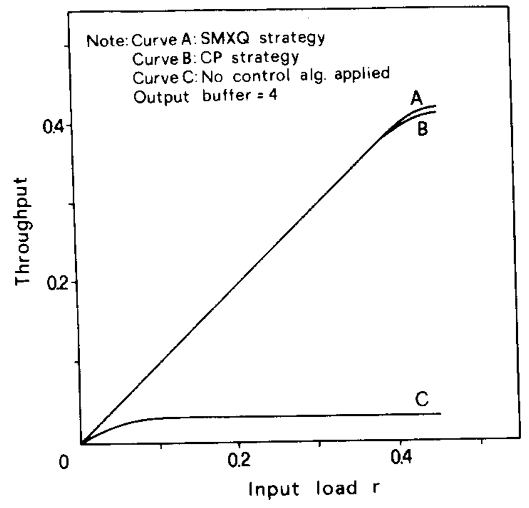


Fig. 14. Network throughput versus the input load r (symmetric load condition).

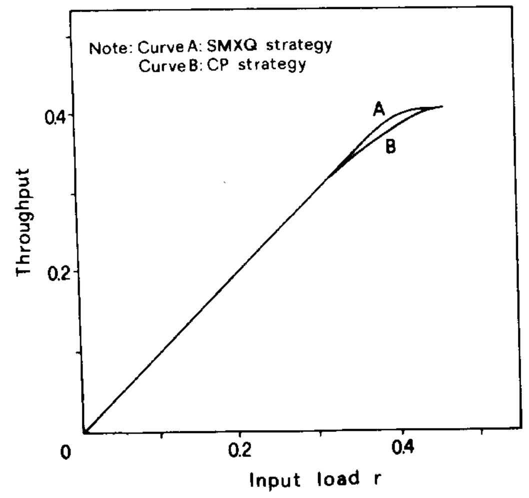


Fig. 15. Network throughput versus the input load r (asymmetric load condition).

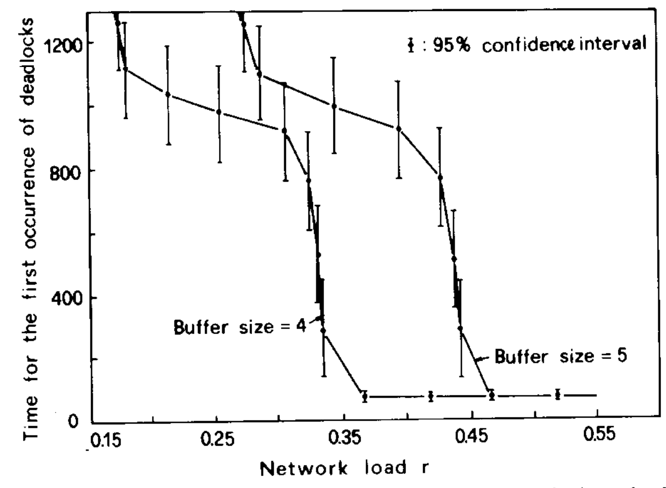
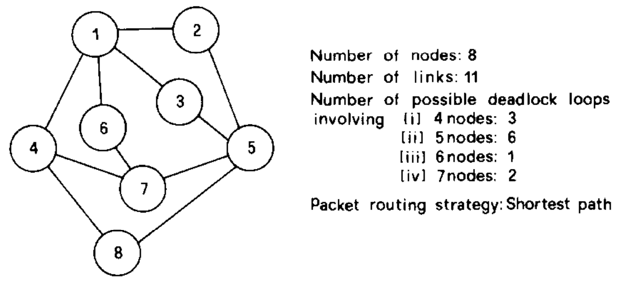


Fig. 16. Time for the first occurrence of deadlocks versus the input load.



1
2
3
4
5
6
7
8
Number of nodes: 8
Number of links: 11
Number of possible deadlock loops involving (i) 4 nodes: 3
(ii) 5 nodes: 6
(iii) 6 nodes: 1
(iv) 7 nodes: 2
Packet routing strategy: Shortest path

Fig. 17. Network II.

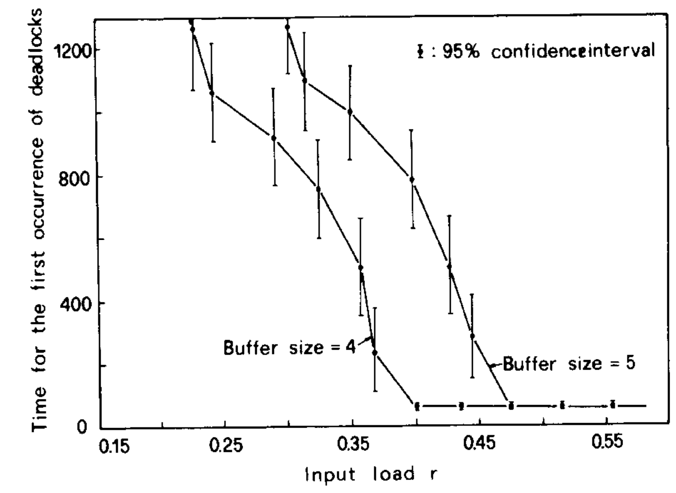


Fig. 18. Time for the first occurrence of deadlocks versus the input load.

Figs. 17 and 18 show the case for another eight-node network. The exact same phenomena are observed.

VIII. Conclusion

Deadlocks can render a communication network inoperative. They can occur even when the network is not heavily loaded. This paper proposes a distributed deadlock detection and resolution algorithm that introduce negligible overhead on network resources. The correctness of the algorithm is proven. Simulation results are provided to verify the properties we claimed for this algorithm and in addition, some interesting deadlock phenomena are observed.

IX. Acknowledgment

We would like to thank the reviewers for their valuable comments and suggestions.

References

[1] K. D. Gunther, "Prevention of deadlocks in packet-switched data transport systems," *IEEE Trans. Commun.*, vol. COM-29, pp. 512–524, Apr. 1981.

[2] P. M. Merlin and P. J. Schweitzer, "Deadlock avoidance in S/F network I: S/F deadlocks," *IEEE Trans. Commun.*, vol. COM-28, pp. 345–354, Mar. 1980.

[3] W. Wimmer, "Using barrier graphs for deadlock prevention in communication networks," *IEEE Trans. Commun.*, vol. COM-32, pp. 897–901, Aug. 1984.

[4] I. S. Gopal, "Prevention of store-and-forward deadlock in computer networks," *IEEE Trans. Commun.*, vol. COM-33, pp. 1258–1264, Dec. 1985.

[5] D. Gelernter, "A DAG-based algorithm for prevention of S/F deadlock in packet networks," *IEEE Trans. Commun.*, vol. COM-30, pp. 709–715, Oct. 1981.

[6] J. Blazewicz *et al.*, "Deadlock-resistant flow control procedures for S/F networks," *IEEE Trans. Commun.*, vol. COM-32, pp. 884–887, Aug. 1984.

[7] G. Gambosi, D. P. Bovet, and D. A. Menasce, "A detection and removal of deadlocks in S/F communication networks," *Performance of Computer-Communication Systems,* W. Bux and H. Rudin, Eds. IBM Zurich Research Laboratory, 1984, pp. 219–229.

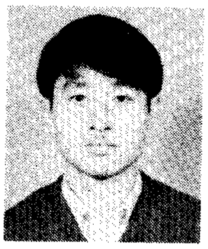
[8] M. Gerla and L. Kleinrock, "Flow control: A comparative survey," *IEEE Trans. Commun.*, vol. COM-28, Apr. 1980.

[9] M. I. Irland, "Buffer management in a packet-switch," *IEEE Trans. Commun.*, vol. COM-26, pp. 328–337, Mar. 1978.

Cheung-Wing Chan received the B.Sc. (with honors) and M.Sc. degrees in electronics from the Chinese University of Hong Kong in 1984, and 1986, respectively.

He is currently a Sales Executive with Wang Pacific Ltd. His research interests are flow controls and deadlocks in computers networks.

Mr. Chan was a recipient of a Croucher Foundation Award for continuation of his studies.



Tak-Shing P. Yum (S'76–M'78–SM'86) was born in Shanghai, China, on February 26, 1953. He received the B.S., M.S., and Ph.D. degrees from Columbia University, New York, NY, in 1974, 1975, and 1978, respectively.

From 1978 to 1980, he was a member of the Technical Staff at Bell Laboratories, Holmdel, NJ, and worked mainly on the performance analysis of the common channel interoffice signaling network. From September 1980 to June 1982, he was an Associate Professor at the Institute of Computer Engineering, National Chiao-Tung University, Taiwan. Since August 1982, he has been a Lecturer in the Department of Electronics, Chinese University of Hong Kong. His main research interest is in the design and analysis of computer communication networks.