Network Working Group
Request for Comments: 626
NIC #22161

L. Kleinrock (UCLA-NMC)
H. Opderbeck (UCLA-NMC)
Mar 1974

"On a Possible Lockup Condition in the IMP Subnet Due to Message Sequencing"

Lockup or deadlock conditions are one of the most serious system malfunctions that can occur in a computer system or network. Communication protocols have to be designed very carefully to avoid the occurrence of these lockups. Their common characteristic is that they occur only under unusual circumstances which were not foreseen or deemed too unlikely to occur by the protocol designers. (However, these designers often are not the ones in a position to evaluate such likelihoods quantitatively.)

The best known lockup that has occurred in the ARPANET is the reassembly lockup [1]. The store-and-forward lockup, also described in Reference 1, has been avoided in the new IMPSYS by carefully observing Kahn's heuristics [1]. The last lockup in the subnet we know of occurred on December 21, 1973 (Christmas lockup"). This dormant lockup conditions was brought to light by collecting snapshot measurement messages from all sites simultaneously. The Christmas lockup happened when snapshot messages arrived at ther UCLA IMP which had allocated reassembly storage for them and no reassembly blocks were free. (A reassembly block is a piece of storage used in the actual process of reassembling packets back into messages) [2].

Deadlock conditions have not only been observed in the subnet but also in higher level protocols. The original design of the ICP, for example, had a flaw that was discovered only after a few months of use [3,4]. More recently BBN reported a deadlock problem involving the exchange of HOST status information by the RSEXEC server (RSSER) programs [5].

As long as it is not possible to design practical communication protocols which guarantee deadlock-free operation it is vital to continually check those protocols that are currently in use for any such failures - even if they appear "very unlikely" to occur. In this RFC we comment on a possible deadlock condition in the IMP subnet which, to our knowledge, has not yet occurred, neither has it been identified. Though we have never seen this problem actually happen it may occur in the future if no precautions are taken. This possible lockup condition is due to the sequencing of messages in the subset.

To avoid the occurrence of reassembly lockup, the flow control mechanism in the subnet was modified in some significant ways. Currently there is a limit of four messages that can simultaneously be in transmission between any pair of source and destination IMPs. As a result of removing the link-handling from the old IMPSYS, it became necessary to introduce a message sequencing mechanism.

Messages leave the destination IMP in the same order as they entered the source IMP. (Note that the sequencing is done on an IMP-to-IMP basis, not a HOST-to-Host basis. This may introduce undesirable "sequencing delay" if two HOSTs attached to the same destination IMP receive messages from the same source IMP).

Sequencing of messages has, in general, the potential of introducing deadlock conditions. The reason for this is that any message, say message (n+1), which is out of order (and therefore cannot be delivered to its destination HOST) may use up resources that are required by message (n) which must be delivered next. Therefore, message (n) cannot reach its destination IMP which, in turn, prevents the other messages (n+1), etc) that are out of order from being delivered to their destination HOST(s). For this reason one has to be very careful not to allocate too many resources (e.g., buffers) to messages that are out of order.

To avoid lockup conditions the current IMPSYS takes the following two precautions:

1. Requests for buffer allocation are always services in order of message number; i.e., no 'ALLOCATE' is returned for message (n+1) if message (n) (or a request for buffer allocation for message (n)) has not yet been received and serviced.

2. Single packet messages (regular and priority) that arrive at the destination IMP out of order are not accepted unless they were retransmitted in response to a previous buffer allocation. These messages are treated rather as a request for the allocation of one buffer (according to 1 above) and then discarded.

With these two precautions a deadlock condition appears to be impossible to occur. However, there is a second buffer allocation mechanism that is not tried to the message sequencing, namely the 'ALLOCATE' that is piggy-backed on the RFNM for a multiple-packet message. The piggy-backed ALLOCATE represents a buffer allocation for the next multiple-packet message, and NOT for the next message in sequence. Thus, if the next message in sequence is a single-packet message, the piggy-backed ALLOCATE in effect allocates buffers to a message that is out of order.

Let us see how this situation can lead to a deadlock condition. Assume there is a maximum number of 24 reassembly buffers in each IMP.

Let IMPs A, B, and C continually transmit 8-packet messages to the same destination IMP D such that all 24 reassembly buffers in IMP D are used up by this transmission of multiple packet messages. If now, in the stream of 8-packet messages, IMP A sends a single-packet message it will generally not be accepted by destination IMP D since there is no reassembly buffer space available. (There may be a free reassembly buffer if the single-packet message happens to arrive during the time one of the three 8-packet messages is being transmitted to its HOST). The single-packet message will therefore be treated as a request for buffer allocation. This request will not be serviced before the RFNM of the previous multiple-packet message is sent. At this time, however, all the free reassembly buffers have already been allocated to the next multiple-packet message via the piggy-backed ALLOCATE mechanism. The only chance for the single-packet message to get its allocation request satisfied is to grab a reassembly buffer from one of the other two 8-packet messages. This attempt may be unsuccessful because it depends on the timing of events in the IMP. A deadlock condition can occur if IMP B and IMP C also send a single-packet message in their stream of 8-packet measures which cannot be serviced for the same reason. In this case, the three 8-packet messages that will arrive later at IMP D cannot be delivered to their destination HOST(s) because they are out of order. The three single-packet messages that should be delivered next, however, will never reach the destination IMP since there is no reassembly space available.

A possible sequence of event that leads to a deadlock condition can be described as follows:

Examples for Notation:

event:	A8 arrives ->	all 8 packets of the 8-packet message from IMP A have arrived at IMP D
event:	C1 arrives ->	a single packet message from IMP C has arrived at IMP D (and is treated as a request for buffer allocation)
event:	B8 complete ->	the last packet of the 8-packet message from IMP B has been received by its destination HOST
event:	A8 RFNM/ALL ->	a RFNM with the piggy-backed ALLOCATE is sent to IMP A

		# of Allocated Reassembly Buffers	# of Reassembly Buffers in Use	# of Free Reassembly Buffers
	Initially	24	0	0
1.	A8 arrives	16	8	0
2.	B8 arrives	8	16	0
3.	C8 arrives	0	24	0
4.	Al arrives	0	24	0
5.	B1 arrives	0	24	0
6.	C1 arrives	0	24	0
7.	A8 complete	0	16	8
8.	B8 complete	0	8	16
9.	C8 complete	0	0	24
10.	A8 RFNM/ALL	8	0	16
11.	B8 RFNM/ALL	16	0	8
12.	C8 RFNM/ALL	24	0	0
13.	A8 arrives	16	8	0
14.	B8 arrives	8	16	0
15.	C8 arrives	0	24	0
16.	- deadlock -			

Note that an ALLOCATE for one of the single-packet messages A1, B1 and C1 can only be returned to source IMP A, B, and C, respectively, after the RFNM (with its piggy-backed ALLOCATE) for the previous 8-packet message has been sent. If these RFNMs are sent in sequence, i.e., without an ALLOCATE for one of the single-packet messages in between, the temporarily freed reassembly storage (events (7) through (9) is implicitly allocated to the next multiple-packet messages (events (10) through (12) and not to any of the single-packet messages. The next 8-packet messages are, however, out of order and cannot be delivered to their destination HOST(s).

Right now it looks as if such a lockup can only occur if the number of reassembly buffers is a multiple of eight. Indeed, the probability of a lockup in this latter case is much higher. However, deadlocks can also occur if the number of reassembly buffers is not a multiple of eight.

Let us assume there are 26 instead of 24 reassembly buffers. Assume also that, due to alternate paths, line failure, or retransmission, the second packet of a 2-packet message arrives at IMP D before a single-packet message from the same source IMP A. The single-packet message has a smaller sequence number and must therefore be delivered to its destination HOST before the 2-packet message. When the second packet of the 2-packet message arrives at IMP D the IMP realizes that only 2 of the allocated 8 buffers will be needed. Therefore

6 buffers are returned to the pool of free reassembly buffers. If there were 26-3x8=2 buffers in the pool before, the pool size is increased by 6 to 8 buffers. These 8 buffers, however, are just enough to send out another piggy-backed ALLOCATE. The single-packet message will therefore find the pool of free reassembly buffers empty although the total number of reassembly buffers is not a multiple of eight. The 2-packet message cannot be delivered to its destination HOST because it is out of order. Therefore the deadlock condition we described before may occur again.

We agree that the above mentioned sequence of events is unlikely to occur (otherwise one would have observed it already). This is particularly true since the current maximum number of reassembly buffers (58) is much larger than 24. Judging from past experience with computer systems and networks, however, we know that even very unlikely events have a tendency to occur in the long run. Also, the probability of this deadlock condition increases with increasing traffic in the net. Therefore, it is certainly worthwhile to modify the IMPSYS in such a way that this deadlock cannot occur. It turns out that a minor modification already achieves the desired effect. Recall that that described deadlock can only occur because single- and multiple-packet messages use the same pool of reassembly buffers. If we set aside a single reassembly buffer (or one for each destination HOST) that can be used only by single-packet messages this lockup condition due to message sequencing cannot occur.

Another solution to this problem is, of course, to more the message sequencing from the IMP to the HOST level. This does not mean that similar lockup problems cannot occur on the HOST level. Also, it is currently much easier to resolve this problem by a slight modification of the IMPSYS rather than having to coordinate all the various HOSTs. Another alternative is to discontinue the use of multiple-packet messages. However, this also involves much more drastic changes which require careful consideration.

REFERENCES

1. Kahn, R. E. and W. R. Crowther, "Flow Control in a Resource-Sharing Computer Network," IEEE Transactions on Communication, Volume COM-20,3 June 1972.

2. BBN Report No. 2717, "Interface Message Processors for the ARPA Computer Network," Quarterly Technical Report No. 4, January 1974.

3. Naylor, W., J. Wong, C. Kline, and J. Postel, "Regarding Proffered Official ICP," RFC 143, NIC 6728.

4. Postel, J. B. "A Graph Model Analysis of Computer Communications Protocols," PhD dissertation, University of California, Los Angeles.

5. BBN Report No. 2670. "Natural Communication with Computers IV," Quarterly Progress Report No. 12, October 1973.