Edwin W. Meyer, Jr.
MIT Project MAC
27 June 1970

The method of flow control described in RFC 54, prior allocation of buffer space by the use of ALL network commands, has one particular advantage. If no more than 100% of an NCP's buffer space is allocated, the situation in which more messages are presented to a HOST then it can handle will never arise.

However, this scheme has very serious disadvantages:

(i) chronic underutilization of resources,

(ii) highly restricted bandwidth,

(iii)considerable overhead under normal operation,

(iv) insufficient flexibility under conditions of increasing load,

(v) it optimizes for the wrong set of conditions, and

(vi) the scheme breaks down because of message length indeterminacy.

Several people from Project MAC and Lincoln Laboratories have discussed this topic, and we feel that the "cease on link" flow control scheme proposed in RFC 35 by UCLA is greatly preferable to this new plan for flow control.

The method of flow control proposed in RFC 46, using BLK and RSM control messages, has been abandoned because it can not guarantee to quench flow within a limited number of messages.

The advantages of "cease on link" to the fixed allocation proposal are that:

(i) it permits greater utilization of resources,

(ii) does not arbitrarily limit transmission bandwidth,

(iii)is highly flexible under conditions of changing load,

(iv) imposes no overhead on normal operation, and

(v) optimizes for the situations that most often occur.

Its single disadvantage is that under rare circumstances an NCP's input buffers can become temporarily overloaded. This should not be a serious drawback for network operation.

The "cease on link" method of flow control operates in the following

manner. IMP messages for a particular "receive" link may be coming in to the destination HOST faster than the attached process is reading them out of the NCP's buffers. At some point the NCP will decide that the input queue for that link is too large in relation to the total amount of free NCP buffer space remaining. At this time the NCP initiates quenching by sending a "cease on link" IMP message to its IMP. This does nothing until the next message for that link comes in to the destination IMP. The message still gets transmitted to the receiving HOST. However, the RFNM returned to the transmitting HOST has a special bit set. This indicates to the originating NCP that it should stop sending over that link. As a way of confirming the suspension, the NCP sends an SPD <link> "suspended" NCP control message to the receiving HOST, telling it that it indeed has stopped transmitting. At a future time the receiving process will have cut the input queue for the link down to reasonable size, and the NCP tells the sending NCP to begin sending messages by issuing a RSM <link> "resume" NCP control message.

The flow control argument is based on the following premises:

(1) Most network transmission falls into two categories:
Type 1 - short messages (<500 bits) at intervals of several seconds or more. (console communication)
Type 2 - a limited number (10 - 100) of full messages (8000 bits) in rapid succession. (file transmission)

(2) Most processes are ready to accept transmitted data when it arrives at the destination and will pick it up within a few seconds (longer for large files). Thus, at any particular instant the great majority of read links have no data buffered at the destination HOST. This assumes a sensible software system at both ends.

(3) Flow control need be imposed only rarely on links transmitting Type 1 messages, somewhat more frequently for Type 2 messages.

(4) Both the total network load and that over a single connection fluctuate and can not be adequately predicted by either the NCP or the process attached to an individual connection.

(5) Assuming adequate control of wide bandwidth transmission (Type 2), the probability that an NCP will be unable to accept messages from the IMP due to full buffers is quite small, even if the simultaneous receipt of moderately small messages over all active links would more than fill the NCP's input buffers.

(6) In the event that an NCP's buffers do fill completely, it may refuse to accept any transmission from the IMP for up to a minute without utter catastrophe.

(7) Under cases of extreme input load, if an NCP has large amounts of data buffered for input to a local process, AND that process has not read data over that connection for more than a minute, the NCP may delete that data to make space for messages from the IMP. This is expected to happen extremely rarely, except in cases where the process is the main contributor to the overload by maintaining several high volume connections which it is not reading.

Based on the preceding premises, I see the following disadvantages in the flow control proposed in RFC 54:

(1) Chronic Underutilization of Resources - A fixed allocation of buffer space dedicated to a single Type 1 connection will go perhaps 90% unused if we actually achieve responsive console interaction through the network. This is because it is empty most of the time and is very seldom more than partially filled. Stated conversely, a scheme of demand allocation might be expected to handle several times as many console connections using the same buffer resources. (It has been suggested that this problem of underutilization could be alleviated by allocating more than 100% of the available buffer space. This is in fact no solution at all, because it defeats the basic premise underlying this method of flow control: guaranteed receptivity. True, it might fail in less than one case in 1000, but that is exactly the case for which flow control exists.)

(2) Highly Restricted Bandwidth - At the same time that all that buffer space dedicated to low volume connections is being wasted (and it can’t be deallocated - see below), high volume communication is unnecessarily slowed because of inadequate buffer allocation. (Because of the inability to deallocate, it is unwise to give a large allocation.) Data is sent down the connection to the allocation limit, then stops. Several seconds later, after the receiving process picks up some of the data, a new allocation is made, and another parcel of data is sent. It seems clear that even under only moderately favorable conditions, a demand allocation scheme will allow several times the bandwidth that this one does.

(3) Considerable Overhead During Normal Operation - It can be seen that flow control actually need be imposed relatively rarely. However, this plan imposes a constant overhead for all connections due to the continuing need to send new allocations. Under demand allocation, only two RFC’s, two CLS’s, and perhaps a couple of SPD and RSM control messages need be transmitted for a single connection. Under this plan, a large fraction of the NCP control messages will be ALL allocation requests. There will probably be several times as many control messages to be processed by both the sending and receiving NCP’s, as under a demand allocation scheme.

(4) Inflexibility Under Increasing Load Conditions - Not only is this plan inflexible as to different kinds of load conditions on a single link, there is potential inflexibility under increasing total load. The key problem here is that an allocation can not be arbitrarily revoked. It can be taken back only if it is used. As an example of the problem that can be caused, assume the case of a connection made at 9 AM. The HOST controlling the receiving socket senses light load, and gives the connection a moderately large allocation. However, the process attached to the send socket intends to use it only to report certain special events, and doesn't normally intend to send much at all down this connection. Comes 12 noon, and this connection still has 90% of its original allocation left. Many other processes are now using the network, and the NCP would dearly love to reduce its allocation, if only it could. Of course it can't. If the NCP is to keep its part of the flow control bargain, it must keep that space empty waiting for the data it has agreed to receive.

This problem can't really be solved by basing future allocations on past use of the connection, because future use may not correlate with past use. Also, the introduction of a deallocation command would cause synchrony problems.

The real implication of this problem is that an NCP must base its allocation to a link on conditions of heavy load, even if there is currently only light network traffic.

(5) Wrong Type of Optimization - This type of flow control optimizes for the case where every connection starts sending large amounts of data almost simultaneously, exactly the case that just about never occurs. As a result the NCP operates almost as slowly under light load as it does under heavy load.

(6) Loss of Allocation Synchrony Due to Message Length Indeterminacy - If this plan is to be workable, the allocation must apply to the entire message, including header, padding, text, and marking, otherwise, a plethora of small messages could overflow the buffers, even though the text allocation had not been exceeded. Thomas Barkalow of Lincoln Laboratories has pointed out that this also fails, because the sending HOST can not know how many bits of padding that the receiving HOST's system will add to the message. After several messages, the allocation counters of the sending and receiving HOSTs will get seriously out of synchrony. This will have serious consequences.

It has been argued that the allocation need apply only to the text portion, because the header, padding, and marking are deleted soon after receipt of the message. This imposes an implementation restriction on the NCP, that it must delete all but the text part of the message just as soon as it gets it from the IMP. In both the TX2 and Multics implementations it is planned to keep most of the message in the buffers until it is read by the receiving process.

The advantages of demand allocation using the "cease on link" flow control method are pretty much the converse of the disadvantages of fixed allocation. There is much greater resource utilization, flexibility, bandwidth, and less overhead, primarily because flow control restrictions are imposed only when needed, not on normal flow.

One would expect very good performance under light and moderate load, and I won’t belabor this point.

The real test is what happens under heavy load conditions. The chief disadvantage of this demand allocation scheme is that the "cease on link" IMP message can not quench flow over a link soon enough to prevent an NCP’s buffers from filling completely under extreme overload.

This is true. However, it is not a critical disadvantage for three reasons:

(i) An overload that would fill an NCP’s buffers is expected to occur at infrequent intervals.

(ii) When it does occur, the situation is generally self-correcting and lasts for only a few seconds. Flow over individual connections may be temporarily delayed, but this is unlikely to be serious.

(iv) In a few of these situations radical action by the NCP may be needed to unblock the logjam. However, this will have serious consequences only for those connections directly responsible for the tie-up.

Let’s examine the operation of an NCP employing demand allocation and using "cease on link" for flow control. The following discussion is based on a flow control algorithm in which the maximum permissible queue length (MQL) is calculated to be a certain fraction (say 10%) of the total empty buffer space currently available. The NCP will block a connection if the input queue length exceeds the MQL. This can happen either due to new input to the queue or a new calculation of the MQL at a lower value. Under light load conditions, the MQL is reasonably high, and relatively long input queues can be maintained without the connection being blocked.

As load increases, the remaining available buffer space goes down, and the MQL is constantly recalculated at a lower value. This hardly affects console communications with short queues, but more queues for high volume connections are going above the MQL. As they do, "cease on link" messages are being sent out for these connections.

If the flow control algorithm constants are set properly, there is a high probability that flow over the quenched links will indeed cease before the scarcity of buffer space reaches critical proportions.

However, there is a finite probability that the data still coming in over the quenched links will fill the buffers. This is most likely under already heavy load conditions when previously inactive links start transmitting at at once at high volume.

Once the NCP's buffers are filled it must stop taking all messages from its IMP. This is serious because now the NCP can no longer receive control messages sent by other NCP's, and because the IMP may soon be forced to stop accepting messages from the NCP. (Fortunately, the NCP already has sent out "cease on link" messages for virtually all of the high volume connections before it stopped taking data from the IMP.)

In most cases input from the IMP remains stopped for only a few seconds. After a very short interval, some process with data queued for input is likely to come in and pick it up from the NCP's buffers. The NCP immediately starts taking data from the IMP again. The IMP output may stop and start several times as the buffers are opened and then refilled. However, more processes are now reading data queued for their receive sockets, and the NCP's buffers are emptying at an accelerating rate. Soon the reading processes make enough buffer space to take in all the messages still pending for blocked links, and normal IMP communications resume.

As the reading processes catch up with the sudden burst of data, the MQL becomes lower, and more and more links become unblocked. The crisis can not immediately reappear, because each link generally becomes unblocked at a different time. If new data suddenly shoots through, the link immediately goes blocked again. The MQL goes up, with the result that other links do not become unblocked.

The worst case appears at a HOST with relatively small total buffer space (less than 8000 bits per active receive link) under heavy load conditions. Suppose that high volume transmission suddenly comes in over more than a half-dozen links simultaneously, and the processes attached to the receive links take little or none of the input. The input buffers may completely fill, and the NCP must stop all input from the IMP. If some processes are reading links, buffer space soon opens up. Shortly it is filled again, this time with messages over links which are not being read. At this point the NCP is blocked, and could remain so indefinitely. The NCP waits for a time, hoping that some process starts reading data. If this happens, the crisis may soon ease.

If buffer space does not open up of its own accord, after a limited interval the NCP must take drastic action to get back into communication with its IMP. It selects the worst offending link on the basis of amount of data queued and interval since data was last read by this process, and totally deletes the input queue for this link. This should break the logjam and start communications again.

This type of situation is expected to occur most often when a process deliberately tries to block an NCP in this manner. The solution has serious consequences only for "bad" processes: those that flood the network with high volume transmission over multiple links which are not being read by the receiving processes.

Because of the infrequency and tolerability of this situation, the overall performance of the network using this scheme of demand allocation should still be much superior to that of a network employing a fixed allocation scheme.