Some Considerations for a High Performance Message-Based Interprocess Communication System*

John M. McQuillan
David C. Walden
Bolt Beranek and Newman Inc.
Cambridge, Massachusetts

1. Introduction

We continue to be concerned with interprocess communications systems (such as those described in references 1, 2, and 3 and called "thin-wire" communications systems in reference 4) which are suitable for communication between processes that are not co-located in the same operating system but rather reside in different operating systems on different computers connected by a computer communications network. Further, the systems with which we are concerned are assumed to communicate using addressed messages (e.g., reference 5) which are multiplexed onto the logical communications channel between the source process and the destination process, rather than using such traditional methods as shared memory (an impossibility for distributed communicating processes) or dedicated physical communications channels between pairs of processes desiring to communicate (which is considered to be prohibitively expensive).

2. Assumptions

The logical communications channel over which the processes must communicate is assumed to have the following properties (see references 6 and 7 for examples of computer communication networks having these properties):

a) finite, fluctuating delay (a result of the basic channel bandwidth, speed of light delays, possible queueing, possible errors and requisite recovery, etc.);

b) finite, fluctuating bandwidth (a result of overhead, possible errors and bandwidth loss due to recovery, contention for the channel, etc.);

c) finite error rate (e.g., messages delivered with bits in error, messages lost entirely, and duplicate copies of messages received -- how the latter two types of errors can occur is discussed in reference 8); and

---

*This work was supported by ARPA Contracts DAHC15-69-C-0197 and F08606-73-C-0027.

d) the possibility of delivery of messages to the destination in an order other than that with which they were transmitted from the source (see references 3, 7, and 8 for discussion of how this may happen). The source and destination are assumed to have, in general, finite storage and differing bandwidths.

3. Goals

We assume that the interprocess communication systems under consideration have two fundamental goals in the transmission of messages: low delay and high throughput. Each message should be handled with a minimum of waiting time, and the total flow of data should be as large as possible. The difference between low delay and high throughput is important. What the user wants is the completion of his data transmission in the shortest possible time. The time between transmission of the first bit and delivery of the first bit is a function of the delay in the communications system, while the time between delivery of the first bit and the delivery of the last bit is a function of the throughput of the communications system. For interactive users with short messages, low delay is more important, since there are few bits per message. For the transfer of long data files, high throughput is more important.

There is a fundamental tradeoff between low delay and high throughput, as is readily apparent in considering some of the mechanisms used to accomplish each goal. For low delay, a small message size is necessary to cut transmission time, to improve the possibility of pipelining where possible, and to shorten possible queueing latencies. Likewise, short queues are desirable. For high throughput, a large message size is necessary to decrease the percentage overhead, both in the communications channel and in processing bandwidth. That is, long messages increase effective channel bandwidth and processing bandwidth. Also, long queues may be desirable to provide sufficient buffering for full channel utilization.

To the goals of low delay and high throughput should be added the equally important goals of cost-effectiveness and high reliability. Individual messages should have a reasonable cost as measured in terms of utilization of network resources; further, the interprocess communications facility itself should be utilized in a cost-effective way. Messages should also be delivered with high probability of success.

How much effort the interprocess communications system has to put into obtaining each of these goals is a function of the subsystems on which the interprocess communication system is built and the level of performance desired. For instance, if the computer communications network guarantees very high reliability for message delivery, the interprocess communications system may have to pay only scant attention to this issue; and what

mechanisms the interprocess communications systems does have to assure reliability can be quite crude if they do not have to be called into action very often. A similar argument may be made with regard to delivering messages in the order sent. On the other hand, even though the subsystems provide very good performance, in some cases it may be desirable for the interprocess communications system to utilize quite sophisticated mechanisms (in addition to those in the subsystems) to handle the same issues which also occur at the interprocess communications system level. Further, in some cases (e.g., when the lower levels have occasional errors), it may be desirable for the interprocess communications system level to have its own mechanisms overcome sublevel troubles. However, if the lower levels perform too badly (e.g., extreme unreliability, poor throughput, poor delay), these represent bottlenecks which the interprocess communications system cannot overcome whatever mechanisms it attempts to bring to bear (in other words, what is done at the interprocess communications level is an addition to what must be done at lower levels, not a substitute).

In summary, we believe that delay, throughput, reliability, and cost are the four criteria upon which message-based interprocess communications system designs should be evaluated and compared. Further, it is the combined performance in all four areas which counts. For instance, poor delay and throughput characteristics may be too big a price to pay for "perfect" reliability.

The assumptions and goals stated above lead us to the strategies discussed in the remaining sections of this paper.

4. Buffering and Pipelining

Buffering is a technique of sending multiple messages over the communications channel before receiving an acknowledgment. Because of the finite delay of the communications channel, it may be desirable to have buffering for multiple messages simultaneously in flight between the source and destination to increase throughput. That is, a system without buffering may have unacceptable low throughput due to long delays waiting for acknowledgment betwen transmissions. For example, if one message of 2000 bits is allowed to be outstanding between the source and destination at a time, and the normal transit time through the communications channel including destination to source acknowledgment is 100 milliseconds, then the maximum throughput rate that can be sustained in only 20,000 bits per second. If poor responsiveness of the destination, great distance, etc. cause the normal transit time to be half a second, then the throughput rate is reduced to only 4,000 bits per second. Clearly, it is a poor design which allows performance (in this case throughput) to vary so greatly with the characteristics of the thin-wire communications medium, when such variance can be

avoided.

The characteristics of the communications channel (e.g., the computer communications network may be a store and forward network) may necessitate the use of relatively small messages so that the delay lowering affects of pipelining may be obtained. That is, the process of collecting and forwarding relatively large messages at each step through the store and forward network may result in excessive delay. By sending smaller messages it is possible to forward the first messages ahead of later ones. For instance, a message of length L traversing H hops of a store and forward network has a delay proportional to L*H; by breaking the message into P smaller, equal size pieces, the delay is proportional to (L/P)+H-1. Of course, in some instances (see reference 6), the computer communications network breaks long messages into smaller messages (so they may pipeline through the network) and then reassembles the original messages before delivery to the destination, all in a manner transparent to the interprocess communications system; in this case, the interprocess communications system has the option of not concerning itself with pipelining.

5. Error Control

We consider error control to comprise three tasks: detecting bit errors in the delivered messages, detecting missing messages, and detecting duplicate messages.

The former task is done in a straightforward manner through the use of checksums. A checksum is appended to the message at the source and checked at the destination; when the checksum does not check at the destination, the incorrect message is discarded, requiring it to be retransmitted from the source. Several points about the manner in which checksumming should be done are worthy of note: a) If possible, the checksum should check the correctness of the resequencing of the messages which possibly got out of order during their journey from the source to the destination. b) A powerful checksum is more efficient than alternate methods such as replication of a critical control field; it is better to extend the checksum by the number of bits that would have been used in the redundant field. c) Frequently people suggest encryption of messages to guarantee they cannot be read if accidentally delivered to the wrong destination; unless encryption is desirable for some other reason, it is simpler (and just as safe) to guarantee no misdelivery through the use of a powerful checksum (which covers the address of the message) than it is to use an encryption mechanism. d) The length of the checksum should be proportional to the log of the product of the desired time between undetected errors, the bit error rate, and the source to destination communication channel bandwidth; that is, checksum size does not depend on message size and it should be quite large.

As stated in the section on Assumptions, the communication channel between the source and the destination has the characteristic that some messages will fail to be delivered, and there will be some duplicate delivery of messages. Missing messages can be detected at the destination through the use of one state bit for each message which can simultaneously be in flight between the source and destination. An interesting detail is that for the purposes of missing message detection, the state bits must precisely cycle through all possible states. For example, stamping messages with a time stamp does nothing for the process of missing message detection because, unless a message is sent for every "tick" of the time stamp, there is no way to distinguish the case of a missing message from the cases where no messages were sent for a time.

Duplicate messages can be detected with an identifying sequence number such that messages which arrive from a prior point in the sequence are recognized as duplicates. The point to note carefully here is that if duplicate messages can arrive at the destination up to some quite possibly long time after the original copy, then the sequence number must not complete a full cycle during this period. For example, if a goal is to be able to transmit 200 minimum length messages per second from the source to the destination and each needs a unique sequence number, and if it is possible for messages to arrive at the destination up to 15 seconds after initial transmission from the source, then the sequence number must be able to uniquely identify at least 3000 messages. It is usually no trouble to calculate the maximum number of messages that can be sent during some time interval. What is usually more difficult is to limit the time after which duplicate messages will no longer arrive at the destination. One method is to put a timer in each message which is somehow counted down as the message journeys from the source to the destination; if the timer ever counts out, the message is discarded as too old, thus guaranteeing that no message older than the initial setting of the timer will be delivered to the destination. Alternatively, one might be able to calculate approximately the maximum arrival time through study of the communications channel between the source and the destination.

There must be mechanisms to resynchronize the sequence numbers between the source and destination at start up time, to recover from failures, etc. A good practice is to resynchronize the sequence numbers occasionally even though they are not known to be out of step. A good frequency with which to do redundant resynchronization would be every time a message has not been sent for longer than the maximum delivery time. In fact, this is the maximum frequency with which the resynchronization can be done (without additional mechanisms); if duplicates are to be detected reliably, the number at the destination must function without disruption for the maximum delivery interval after the "last message" has been sent. If it is desirable or necessary to resynchronize the sequence numbers more often than the maximum

time, an additional "use" number must be attached to the sequence number to uniquely identify which "instance" of this set of sequence numbers is in effect; and, of course, the messages must also carry the use number. This point is addressed in greater detail in references 9 and 10.

The next point to make about error control is that any message going from the source to the destination can potentially be missing or duplicated. In fact, the very control messages used in error control (e.g., sequence number resynchronization messages) can themselves be missing or duplicated, and a proper message transmission system must handle these cases.

Finally, there must be some inquiry-response system from the source to the destination to complete the process of detecting lost messages or messages discarded at the destination because of checksum detection of errors. When the proper reply or acknowledgment has not been received for too long, the source may inquire whether the destination has received the message in question. Alternatively, the source may simply retransmit the message in question. In any case, this source inquiry and retransmission system must also function in the face of duplicated or lost inquiries and inquiry response messages. Finally, the acknowledgment and retransmission system must depend on positive acknowledgments from the destination to the source and on explicit inquiries or retransmission from the source. Negative acknowledgments from the destination to the source are never sufficient (because they might get lost) and are only useful (albeit sometimes very useful) for increased efficiency.

6. Storage Allocation and Flow Control

One of the fundamental rules of communications systems is that the source cannot simply send data to the destination without some mechanism for guaranteeing storage for that data. In very primitive systems one can sometimes guarantee a rate of disposal of data, as to a synchronous line printer, and not exceed that rate at the data source. In more sophisticated systems there seem to be only two alternatives. Either one can explicitly reserve space at the destination for a known amount of data in advance of its transmission, or one can declare the transmitted copy of the data expendable, sending additional copies from the source until there is an acknowledgment from the destination. The first alternative is the high bandwidth solution: when there is no space, only tiny control messages (for the purpose of reservation of destination storage) travel back and forth between the source and destination. The second alternative is the low delay solution: the text of the message propagates as fast as possible.

In either case storage is tied up for an amount of time equal at least to one round trip time. This is a fundamental result -- the minimum amount of buffering required by a communications

system, either at the source or destination, equals the product of round trip time and the channel bandwidth. The only way to circumvent this result is to count on the destination behaving in some predictable fashion (an unrealistic assumption in the general case of autonomous communicating processes).

Our experience and analysis convinces us that if both low delay and high throughput are desired, then there must be mechanisms to handle each, since high throughput and low delay are conflicting goals. This is true, in particular, for the storage allocation mechanism. It has occasionally been suggested (e.g., reference 11), mainly for the sake of simplicity, that only the low delay solution be used; that is, messages are transmitted from the source without reservation of space at the destination. Those people making the choice never to reserve space at the destination frequently assert that high bandwidth will still be possible through use of a mechanism whereby the source sends messages toward the destination, notes the arrival of acknowledgments from the destination, uses these acknowledgments to estimate the destination reception rate, and adjusts its transmissions to match that rate. We feel that such schemes may be quite difficult to parameterize for efficient control and therefore may result in reduced effective bandwidth and increased effective delay. If the source never sends to the destination so fast that the destination must discard anything, then the delay is very low, but the throughput is not as high as it might be. Further, unless the source pushes now and then, it will never discover that the destination is able to increase its throughput. On the other hand, when the source is pushing hard enough, the destination may suddenly cut back on its throughput, causing all the messages which will be discarded at the destination due to the sudden cut back to have to be retransmitted increasing effective delay. If the destination could be predicted to accept traffic at a steady rate and vary this rate only very slowly, the above sort of feedback system might work. In this case unacknowledged messages should be retransmitted from the source to the destination shortly after the expected time for the acknowledgment to return has elapsed if minimum delay and maximum throughput are to be obtained (this is in contrast to the often suggested practice of keying retransmissions to the discard rate). However, in practice, the time for the acknowledgment to return is likely to be very difficult to predict due to variations (possibly rapid) in the transit time of the communications channel and particularly in the response time of the destination. Furthermore, the greater the sum of transit time and response time, the looser and less efficient the feedback loop will be. In fact, there appear to be oscillatory conditions which can occur where performance degrades completely. (Note, if there is much possibility of message loss, then the acknowledgment and retransmission system should allow quite selective retransmission of messages rather than, for instance, requiring a complete window of messages to be retransmitted to effect retransmission of the specific messages requiring it; otherwise, message retransmission will use excessive

bandwidth.)

The above discussion assumes that all mechanisms are attempting to minimize the probability of message discard. If, in addition to possible discards at the destination, the communications channel solves its internal problems (e.g., potential deadlocks) with cavalier discarding of messages, or if the destination solves its internal problems with cavalier discarding of messages, the detrimental effects of discarding (reduced effective bandwidth and increased effective delay) are probably drastically increased. Further, the above discussion assumed the destination was able to minimize the probability of discard. While this may be possible for a single source, we think it is unlikely that the destination will be able to resolve, in a way that does not entail excessive discards, the contention for destination storage from multiple uncoordinated sources. As reported in reference 12, detrimental contention for destination storage, in the absence of a storage reservation mechanism, happens practically continuously under even modest traffic loads, and in a way uncoordinated with the rates and strategies of the various sources. As a result, well-behaved processes may unavoidably be penalized for the actions of poorly-behaved processes.

A subtle point is that in addition to space to hold all data, there must also be space to hold all control messages. In particular, there must be space to record what needs to be sent and what has been sent. If a message will result in a response, there must be space to hold the response; and once the response has been sent, the information about what kind of answer was sent must be kept for as long as retransmission of that response may be necessary.

7. Multiplexing and Addressing

To this point in our paper, we have not been very specific about whether the above mentioned flow control, sequencing, error control, etc. mechanisms were performed for each pair of communicating processes, or whether several processes communicating between a given pair of source and destination operating systems share a set of these control mechanisms. The tradeoff is between overhead and precision of control. If many conversations are multiplexed on each instance of a source to destination control mechanism, the control overhead is lower than if each conversation has its own control mechanism. On the other hand, if several conversations are multiplexed on the same control mechanism, all the conversations tend to have to be treated equally (e.g., if one is stopped, all are stopped); while if each conversation has its own control mechanism, exact decisions about the allocation of various resources to the various conversations can be made. To give some examples of the latter, conversations over separate control mechanisms can be given differing

allocations, priorities, treatments of error conditions, etc.

Another issue is the management of the space available for control mechanisms when it is insufficient to handle the number of conversations competing for the communications channel. Should late-comers be left out until resources are available, or should some way be found to multiplex the available control mechanisms in time among the demanding conversations? We believe the latter should be done. The key here is to not allow users to explicitily acquire and hold resources (e.g., control mechanism space) needed for interprocess communication. Instead, the system should notice which users are actively communicating and dynamically gather the needed resources by garbage collecting the resources previously being used by users which appear inactive. This dynamic assignment of resources is obviously not fundamentally different from the scheduling of any other limited resource in an operating system (e.g., memory, CPU cycles, the I/O channel to the disk) and therefore has all the normal possibilities for thrashing, unfairness, and so on if care is not taken.

Once decisions in all of the above areas of multiplexing are made, one must choose the addressing mechanism and formats to be used. This is usually quite straightforward. The main point here is that addressing comes last; but very often we see designs begun by choosing the addressing system and format. A similar statement can be made about the choice of all other message formats.

Acknowledgments

We are grateful for help from several of our colleagues who are also actively interested in the areas discussed in this working paper: Jerry Burchfiel, Will Crowther, Alex McKenzie, Randy Rettberg, Bob Thomas, and Ray Tomlinson.

References

1. D.C. Walden, "A System for Interprocess Communication in a Resource-Sharing Computer Network," Communications of the ACM, Vol. 15, No. 4, April 1972, pp. 221-230; also in "Advances in Computer Communications," W.W. Chu (ed.), Artech House Inc., 1974, pp. 340-349.

2. R. Bressler, D. Murphy, and D. Walden, "A Proposed Experiment with a Message Switched Protocol," ARPA Network Working Group, Request for Comments No. 333, May 1972.

3. V. Cerf and R. Kahn, "A Protocol for Packet Network Intercommunication," IEEE Transactions on Communications, Vol. COM-22, No. 5, May 1974, pp. 637-648.

4. R.M. Metcalfe, "Strategies for Interprocess Communication in a Distributed Computing System," Proceedings of the Symposium on Computer Communications Networks and Teletraffic, Polytechnic Press of the Polytechnic Institute of Brooklyn, 1972.

5. P. Brinch-Hansen, "The Nucleus of a Multiprogramming System," Communications of the ACM 13, 4, pp. 238-250, April 1970.

6. F.E. Heart, R.E. Kahn, S.M. Ornstein, W.R. Crowther, and D.C. Walden, "The Interface Message Processor for the ARPA Computer Network," AFIPS Conference Proceedings, Vol. 36, June 1970, pp. 551-567; also in "Advances in Computer Communications," W.W. Chu (ed.), Artech House Inc., 1974, pp. 300-316.

7. L. Pouzin, "Presentation and Major Design Aspects of the Cyclades Computer Network," Proceedings of the Third ACM Data Communications Symposium, November 1973, pp. 80-88.

8. W.R. Crowther, F.E. Heart, A.A. McKenzie, J.M. McQuillan, and D.C. Walden, "Issues in Packet Switching Network Design," AFIPS Conference Proceedings 44, May 1975, pp. 161-175.

9. J.M. McQuillan, "The Evolution of Message Processing Techniques in the ARPA Network," July 1974, to appear in International Computer State of the Art Report No. 24: Network Systems and Software, Infotech, Maidenhead, England.

10. R.S. Tomlinson, "Selecting Sequence Numbers," this proceedings.

11. D. Belsnes, "Flow Control in Packet Switching Networks," INWG Note No. 63, October 1974.

12. R.E. Kahn and W.R. Crowther, "Flow Control in a Resource-Sharing Computer Network," Proceedings of the Second ACM/IEEE Symposium on Problems in the Optimization of Data Communications Systems, Palo Alto, California, October 1971, pp. 108-116; also in IEEE Transactions on Communications, Vol. COM-20, No. 3, Part II, June 1972, pp. 539-546; also in "Advances in Computer Communications," W.W. Chu (ed.), Artech House Inc., 1974, pp. 230-237.