

Message Packing as a Performance Enhancement Strategy with Application to the Totem Protocols*

P. Narasimhan, L. E. Moser, P. M. Melliar-Smith
Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106

Abstract

Packing a number of messages into a single packet for transmission can be used as a strategy to improve the performance of communication protocols. In this paper, we investigate this technique and its application to the Totem reliable ordered multicast protocols. The mechanism underlying message packing is also analyzed by means of a buffer model with exponential message lengths and exponential interarrival times between messages. The results obtained from the analysis and from the implementation of this strategy in the Totem protocols show that message packing is a very effective technique for improving the performance, especially for messages of small length.

1 Introduction

Many applications are built as distributed systems that rely on message passing as the means of communication between the processors in the network. Among the desirable performance characteristics of a communication protocol underlying a distributed system are high throughput and low latency. In this paper, we study message packing as a technique for improving the performance of communication protocols. The improvement obtained from the use of such a packing strategy demonstrates its viability and its value in the implementation and application of distributed systems.

The typical protocol data unit involved in the transmission of messages is a *packet*. Usually, a message that arrives from the higher layer is encapsulated into a packet along with a packet header, and is transmitted without delay. In such a case, the mean rate at which packets are transmitted equals the mean rate at which messages are transmitted. Thus, there is a direct one-to-one correspondence between messages and packets.

Alternatively, if we buffer the arriving messages up to a desired cumulative length and *then* transmit them in a packet, there exists a distinct possibility of improvement in performance, particularly if the messages are of small length. A potential drawback of such a scheme is the amount of time involved in buffering the messages prior to sending them, as opposed to transmitting them directly without a buffering delay.

If such a scheme is intelligently implemented so that the performance benefit (from the increased amount of useful data per packet owing to a larger number of messages per packet) outweighs the increased time taken to form a packet, the mechanism

of message packing is indeed worthwhile. The most noticeable improvement is visible in the increased throughput. The reason for this is that each packet incurs some overhead on transmission due to the characteristics of the underlying communication protocol. The overhead per packet is usually a fixed quantity. If there are multiple messages in a packet, the overhead per message is reduced as compared to the situation without message packing. Clearly, the more messages that constitute a packet, the lower the overhead per message but the greater the buffering time per message. Thus, a tradeoff exists between these two quantities.

The ubiquitous TCP/IP protocols employ a similar concept of batching messages with the aim of improving performance. Much closer to our work is the message packing strategy adopted for the Horus system [3]. That work indicates the potential of message packing as a tool for increasing the performance of communication protocols. In this paper, we evaluate the viability of message packing, not only by means of its implementation in the Totem protocols [1, 2, 6] but also by the development of a mathematical model to analyze the mechanisms underlying it.

2 Analytical Modelling

The model presented in this section presupposes exponential probability density functions for both the message lengths and the interarrival times of messages. The analysis can be extended to other probability distributions.

We assume that messages can be transmitted only in the form of a packet of fixed size. The headers of the packets are excluded from the analysis. Such finer details are discussed in Section 3 on the implementation of the message packing strategy in the Totem protocols. A useful quantity that recurs throughout the analysis is the *packing factor*, which we define as the mean number of messages per packet. The following assumptions are also embodied in the model.

Assumptions

- The length of a packet is fixed, while the actual length of the data that it contains varies. A packet is modelled as a buffer of B bytes in length.
- Message lengths X are exponentially distributed with mean λ^{-1} bytes. Thus, a sequence of message lengths X_1, X_2, \dots is a sequence of independent, identically distributed random variables.
- Message interarrival times are exponentially distributed with mean μ^{-1} secs.

*Research supported by DARPA contract number N00174-93-K-0097.

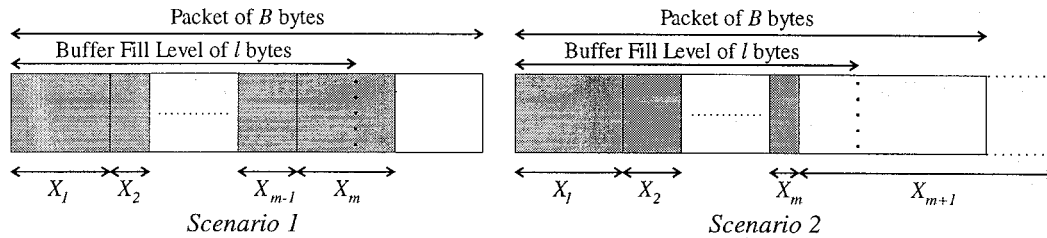


Figure 1: Message packing with a buffer fill level of l bytes. Two possible scenarios result in m messages being transmitted as a packet.

- Instead of packing messages to the fullest extent of the buffer, messages within the buffer are transmitted as soon as their cumulative length exceeds the *buffer fill level* of l bytes. Buffer overflow is considered as a special case.

2.1 Messages Sent without Packing

Here, each packet contains a single message. Thus, the packing factor is unity. The mean data length is λ^{-1} bytes. The throughput is μ messages/sec or μ packets/sec. The buffer utilization is $(B\lambda)^{-1}$. It should be noted that $B\lambda \geq 1$. Sending messages without packing is recommended when the mean message length is itself B bytes because in this case message packing has no effect.

2.2 Messages Sent with Packing

In this case, multiple messages are transmitted as a packet. The number of messages transmitted as a packet is m if either of the following two scenarios occurs.

Scenario 1

As shown in Figure 1 at the left, the cumulative length of the first $m - 1$ messages is less than l bytes and the arrival of the m th message causes the cumulative length to exceed l bytes without causing buffer overflow. In this case, the m messages are sent and the buffer returns to its initial empty state. This occurs with probability

$$\Pr \{0 \leq X_1 + \dots + X_{m-1} < l, l \leq X_1 + \dots + X_m \leq B\}$$

Scenario 2

As shown in Figure 1 at the right, the cumulative length of the first m messages is less than l bytes and the arrival of the $(m + 1)$ st message causes the buffer to overflow. The first m messages are sent as a packet and the $(m + 1)$ st message is retained as the first message of the next packet to be sent. This occurs with probability

$$\Pr \{0 \leq X_1 + \dots + X_m < l, B < X_1 + \dots + X_{m+1} < \infty\}$$

2.2.1 Mathematical Formulas

Under the conditions stipulated above,

$$\text{packing factor} = l\lambda + (1 - e^{-(B-l)\lambda}) \text{ messages/packet}$$

$$\text{mean useful data} = l + (1 - e^{-(B-l)\lambda})\lambda^{-1} \text{ bytes/packet}$$

$$\text{throughput} = \frac{\mu}{l\lambda + (1 - e^{-(B-l)\lambda})} \text{ packets/sec}$$

Note that the mean number of messages per second is still μ and, thus, we have the following relation

$$\text{packing factor} = \frac{\text{throughput in messages/sec}}{\text{throughput in packets/sec}}$$

which can also be seen intuitively.

The really interesting quantity that we derive is the mean waiting time of a message between the time that it arrives from the upper layer until it is transmitted on the medium. Since the messages in the buffer arrive at different times and are queued in the buffer until they are transmitted, they have different waiting times. The state transition diagrams in Figure 2 show the different states of the buffer as messages arrive.

Analysis and solution of the state transition equations reveal that

$$\text{mean message waiting time} = \frac{l\lambda + e^{-(B-l)\lambda}}{2\mu} \text{ secs}$$

2.2.2 Practical Constraints

Practical considerations arising from the implementation of the message packing scheme impose the constraint that the packing factor must be at least unity. It follows from this that

$$\text{throughput} \leq \mu \text{ packets/sec}$$

$$\text{mean amount of useful data} \geq 1/\lambda \text{ bytes/packet}$$

$$\text{mean message waiting time} \leq l\lambda/\mu \text{ secs}$$

2.2.3 Results

The graph for the variation of the message waiting time, shown in Figure 3 at the left, is interesting in that it demonstrates that smaller message lengths lead to larger waiting times, as is to be expected. The graph for the throughput, shown in Figure 3 at the right, indicates that messages of larger lengths are transmitted more quickly, thereby increasing the rate of packets being transmitted. Increasing the buffer fill level clearly increases the extent to which messages are packed.

As the graph in Figure 4 at the left indicates, the mean amount of useful data increases as the buffer fill level l increases. The graph in Figure 4 at the right shows the variation of the message waiting time with throughput and is useful in tailoring a message packing scheme to meet the requirements of the underlying

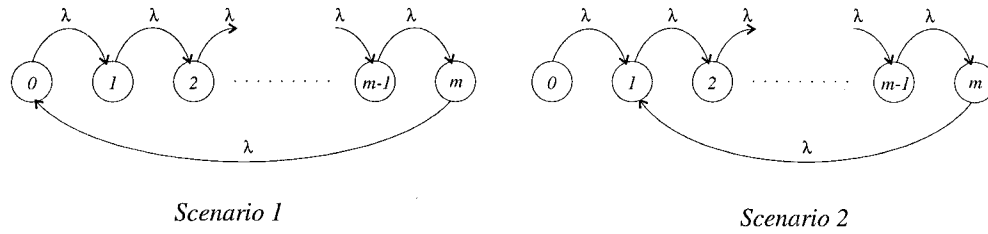


Figure 2: State transition diagrams. For a packet consisting of m messages, there are two state transition diagrams (corresponding to the two possible scenarios shown in Figure 1) with $m + 1$ states each. A state labelled i represents i messages in the buffer, given that m messages are transmitted as a packet.

communication protocol. For instance, given the maximum waiting time that the protocol can tolerate, we can arrive at a suitable buffer fill level and the throughput resulting from this choice.

The other interesting aspect of these graphs is that they advocate a partial buffer filling strategy (where $l < B$) over a scheme where the buffer is completely filled with messages. This is representative of the tradeoff between the message waiting time and the throughput in messages/sec.

2.3 Evaluation of the Strategies

Realistically speaking, every packet has a protocol (transmission) overhead, which is usually fixed for a given packet size. Let us assume that a packet size of B bytes gives rise to a fixed overhead of T seconds. For simplicity, also assume that the buffer fill level l equals B bytes, the length of the buffer itself. Without message packing,

$$\text{time between packets} = T + \mu^{-1} \text{ secs}$$

$$\text{time between messages} = T + \mu^{-1} \text{ secs}$$

With message packing and $l = B$,

$$\text{time between packets} = T + (B\lambda)\mu^{-1} \text{ secs}$$

$$\text{time between messages} = T(B\lambda)^{-1} + \mu^{-1} \text{ secs}$$

where $B\lambda$ is the packing factor.

From the above and using the fact that $B\lambda \geq 1$ when packing is implemented,

$$T(B\lambda)^{-1} + \mu^{-1} \leq T + \mu^{-1}$$

which implies that

$$\text{throughput with packing} \geq \text{throughput without packing}$$

where the throughput is in messages/sec.

If the transmission overhead T is considerable, a substantial improvement in performance can be realized by packing messages, thereby reducing the protocol overhead per message.

Another interesting fact surfaces. If the mean message size λ^{-1} decreases, the packing factor $B\lambda$ increases and, thus, the mean throughput in messages/sec also increases for a fixed packet size of B bytes. This means that the most dramatic improvement in performance can be realized for messages of small length, a fact which is borne out by the application of the message packing strategy to the Totem protocols.

As anticipated, if the mean message size λ^{-1} decreases, the extent of packing clearly increases and the mean waiting time of a message as given by $(1 + B\lambda)\mu^{-1}$ increases, demonstrating the potential danger of increasing the extent of packing beyond a certain limit.

3 Application to the Totem Protocols

3.1 Message Packing Scheme

The Totem protocols [1, 2, 6] provide reliable ordered delivery of messages over a single local-area network, or across multiple LANs interconnected by gateways. The Totem single-ring protocol uses a logical token-passing ring imposed on a single LAN that provides hardware broadcasts (such as an Ethernet) with a token that provides total ordering of messages, flow control, fault detection and fault recovery. The Totem system incorporates the idea of multicasting to process groups, and provides two types of message delivery guarantees, *agreed* and *safe*. A process can deliver a message in agreed order if it has already delivered all prior messages originated by processes in its current group membership and timestamped within the duration of that membership. Delivery in safe order requires, in addition, that before a process can deliver a message, it has determined that every other process in its current group membership has received the message, and will deliver the message unless that process fails. The latency of the Totem single-ring protocol without message packing has been studied [5]. Our research reveals that higher throughput of the protocol is obtained with the implementation of message packing, particularly for messages of small length.

Messages and packets have the structure shown in Figure 5. The packet header has a field labelled *packet_data_len*, which indicates the length of the data in the packet. The message header has a field labelled *total_mesg_len*, which indicates the length of the message, including the header. The packet length is limited by the capability of the Ethernet; a packet, including the header, has a fixed length of 1472 bytes. The packet and message header sizes are fixed at 68 and 56 bytes, respectively. These numbers are subject to change with future versions of Totem and the platform on which it runs.

The main difference between this discussion and the analysis in Section 2 is that we use deterministic interarrival times (by means of a periodic timeout used to control the rate of message generation) and deterministic message lengths in the application of the packing technique to the Totem protocols. Also, the

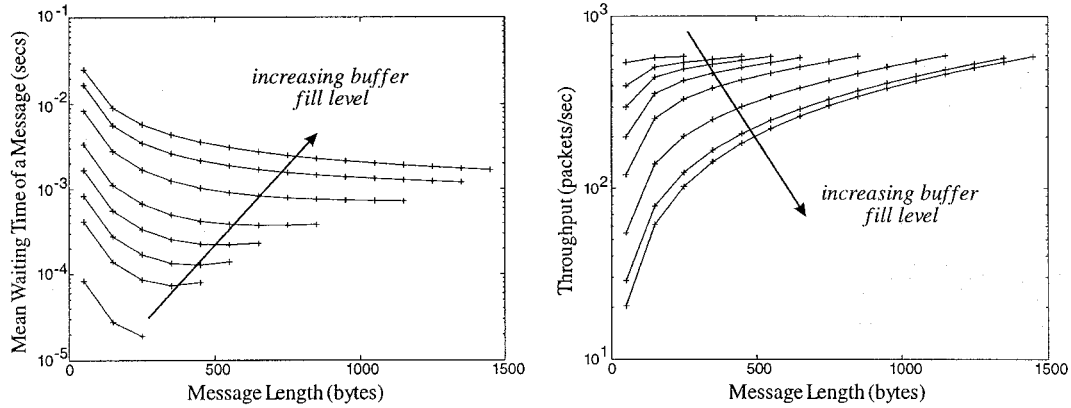


Figure 3: Mean waiting time and throughput as a function of message length. Each curve in either graph is drawn for a different fixed buffer fill level l bytes.

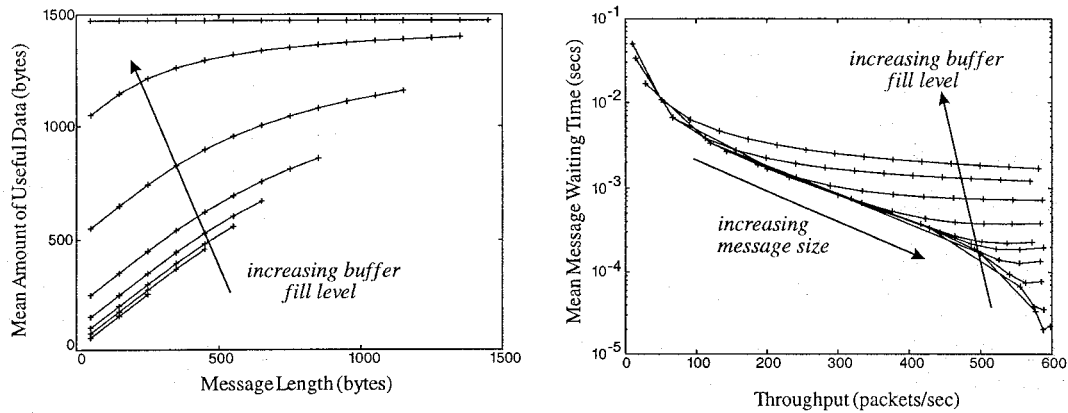


Figure 4: Useful data as a function of message length and message waiting time as a function of throughput. Each curve in either graph is drawn for a different fixed buffer fill level l bytes.

buffer fill level l equals B bytes, the size of the packet itself. Investigations of Totem are currently underway for schemes employing different message length distributions, as well as a partial buffer filling strategy.

3.1.1 Simple Strategy

The simplest packing scheme involves packing messages back-to-back without taking into account the redundancy of consecutive message headers. We discovered that even this simple scheme of message packing proves to be an effective mechanism for improving the performance. However, greater performance improvements are obtained with the more intelligent message packing scheme outlined below.

3.1.2 Refined Strategy

The information contained in the headers of consecutive messages in a packet is the same except for the message length, which could potentially differ from one message to the next (though the results presented pertain to messages of fixed length). The redundant information that is replicated in consecutive message headers includes the message type (agreed or safe), the sender id and the process groups to which the message is sent.

To avoid incorporating the entire message header with every message and thereby incurring a 56 byte overhead with every

message, the 56 byte header of the first message is left intact. Every message thereafter has its header stripped off; a two byte separator is inserted between messages to indicate the length of the message that follows the separator. Figure 5 shows several messages packed using this scheme.

3.2 Results

The Totem single-ring protocol was run on a network of eight Sun SPARC20 workstations connected by a 100 Mbps Ethernet. The message packing scheme was implemented with deterministic message lengths, deterministic message interarrival times, and a buffer fill level equal to the packet length.

The graph in Figure 6 compares the throughput in messages/sec with and without packing. It is evident that message packing improves the performance dramatically. The graph also indicates that message packing is most effective for messages of small length, as we concluded earlier from our analysis. Preliminary measurements indicate a 70-fold improvement in throughput for four byte messages, when message packing is implemented. Of course, the throughput in messages/sec reduces as the message size increases or as the extent of packing decreases.

It was also observed from the measurements that, although the throughput in messages/sec increases with a decrease in message length, a corresponding increase in the time to send a message

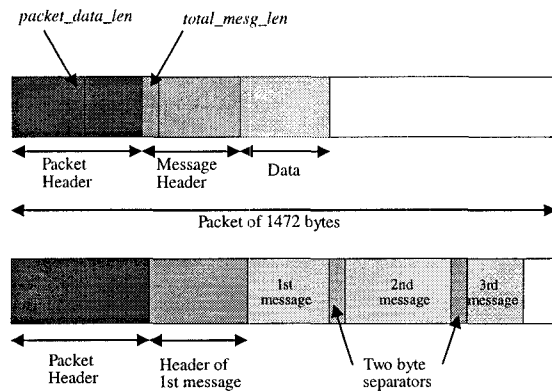


Figure 5: Message packing in the Totem single-ring protocol. The top of the figure shows the structure of a message and a packet in Totem. The bottom of the figure shows the refined packing strategy adopted using 2 byte separators.

is observed. This was anticipated from the analytical results and can be attributed to the increased buffering time per packet as the packing factor increases.

4 Conclusions and Future Directions

Applications that are built as distributed systems require high performance (high throughput and low latency) from the underlying communication protocol. The technique of message packing provides a means of enhancing the performance of such protocols.

While the buffering of messages increases the time between the transmission of consecutive packets, the amount of useful data per packet also increases. It is this tradeoff that makes message packing such an interesting subject of study. For small messages the performance improves dramatically, while for large messages the performance approaches that of the scheme without message packing.

Message packing is undoubtedly a useful and valuable technique in enhancing the performance of communication protocols, as our results show. However, many interesting issues in message packing remain to be explored including:

- Analysis of the integration of the message packing mechanism with the token-passing mechanism of the Totem single-ring protocol. We are currently developing a *symmetric, limited-service, polling system* model [4, 7, 8] with infinite buffers.
- Existence of an *optimal* buffer fill level. This would provide the best possible tradeoff between the mean message waiting time and the throughput.
- Analysis and implementation of the message packing scheme with other probability distributions for the message lengths, such as an Erlang distribution. This might allow us to model the system more accurately.
- Performance improvement with upgraded processors and communication cards.
- Analysis of the impact of message packing on the Totem multiple-ring protocol.

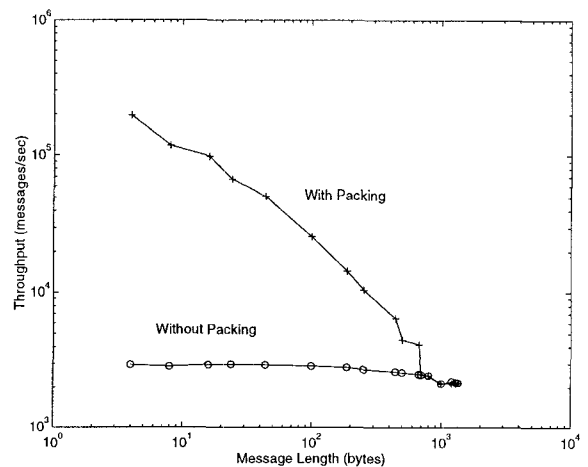


Figure 6: Throughput with and without packing.

References

- [1] D. A. Agarwal, *Totem: A Reliable Ordered Delivery Protocol for Interconnected Local-Area Networks*. PhD Dissertation. Department of Electrical and Computer Engineering. University of California, Santa Barbara, August 1994.
- [2] Y. Amir, L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal and P. Ciarfella, "The Totem single-ring ordering and membership protocol," *ACM Transactions on Computer Systems* 13, 4 (November 1995), 311-342.
- [3] R. Friedman and R. van Renesse, "Packing messages as a tool for boosting the performance of total ordering protocols," *Technical Report 95-1527, Department of Computer Science, Cornell University*.
- [4] A. Konheim, "Chaining in a loop system," *IEEE Transactions on Communications COM-24*, 2 (February 1976), 203-210.
- [5] L. E. Moser and P. M. Melliar-Smith, "Probabilistic bounds on message delivery for the Totem single-ring protocol," *Proceedings of the 15th IEEE Real-Time Systems Symposium*, San Juan, Puerto Rico (December 1994), 238-248.
- [6] L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, R. K. Budhia and C. A. Lingley-Papadopoulos, "Totem: A fault-tolerant multicast group communication system," *Communications of the ACM* 39, 4 (April 1996), 54-63.
- [7] H. Takagi, *Analysis of Polling Systems*, MIT Press, 1986.
- [8] H. Takagi, "Application of polling models to computer networks," *Computer Networks and ISDN Systems* 22 (1991), 193-211.