Performance Measurements in a Manufacturing Communication System

Célio V. N. de Albuquerque
celio@coe.ufrj.br

Marcelo D. Nunes
bill@coe.ufrj.br

Otto C. M. B. Duarte
otto@coe.ufrj.br

COPPE/EE - Programa de Engenharia Elétrica
Universidade Federal do Rio de Janeiro
P.O. Box 68504 - CEP 21945-970 - (+55) 021-290-6626
Rio de Janeiro - RJ - Brasil

ABSTRACT

This paper presents and analyses a high performance manufacturing communication system. It consists in the standard TOP profile implemented in single processor computer connected to a LAN. High throughput is achieved by an efficient implementation architecture based on specific layer interfaces and data structures, specialized mechanisms of memory management, timer management and task scheduling. Performance measurement results show a throughput efficiency that attains 6 Mbit/s for a remote communication and 42 Mbit/s for loopback configuration. The most important bottlenecks are analysed and consist in the Transport checksum, Transport acknowledgment step frequency, LI.C memory copy, memory management and task scheduling.

INTRODUCTION

Integration of the various elements in a manufacturing system has become an important issue. This has led to the development of Computer Integrated Manufacturing (CIM) concepts [1], where proper communication is necessary to ensure adequate functioning of all the elements. Two communication protocols, namely, Manufacturing Automation Protocol (MAP) [2] and Technical and Office Protocols (TOP) [3], have emerged as a standard approach to enable easy and effective communication between the elements in CIM environments. Rather than specifying new protocols, MAP/TOP are based on international standards adopted by ISO, specifically the Reference Model for Open Systems Interconnection (RM-OSI). The RM-OSI defines a layered architecture structured in a hierarchical added value form.

The high bandwidth provided by the Local Area Networks (LAN) has moved the performance bottleneck to the protocol processing time. This work deals with the problem of implementing the standard TOP communication profile efficiently in a low cost and single processor environment.

This paper presents an implementation architecture of a TOP profile and its performance measurements in a local area network (LAN) environment. In order to achieve high performance communication protocols, an implementation architecture is defined. This architecture is based on specific layer interface and data structures, specialized modules of memory management, timer management and task scheduling. The proposed implementation architecture privileges flexibility, modularity and weak coupling between adjacent layers, minimizes the necessary retransmission buffers and takes into account the memory fragmentation. The measurements show a good performance and analyse the throughput due to each layer processing time, checksum mechanism and acknowledgment step frequency.

This work is organized as follows. Section 2 describes TOP profile and the implemented features of each layer. Section 3 presents the implementation architecture. Its performance measurements are given in Section 4. Finally, conclusions are presented in Section 5.

MANUFACTURING PROTOCOL PROFILES

CIM organizational hierarchy (Figure 1) is defined by the ISO Reference Model for factory automation. This model supports six hierarchical levels: the Equipment Level (realization of commands to the shop floor equipment -- robots, conveyors, vehicles, tools, etc.), The Station Level (numerical controllers, programmable logic controllers, etc., which direct and coordinate the activity of the shop floor equipment), the Cell Level (supervision of the various supporting devices), the Section/Area Level (coordination of production and resource allocation), the Facility/Plant Level (production planning and scheduling), and the Enterprise Level (corporate management).

MAP and TOP protocol profiles were specified as standards for manufacturing environments allowing different shop floor and

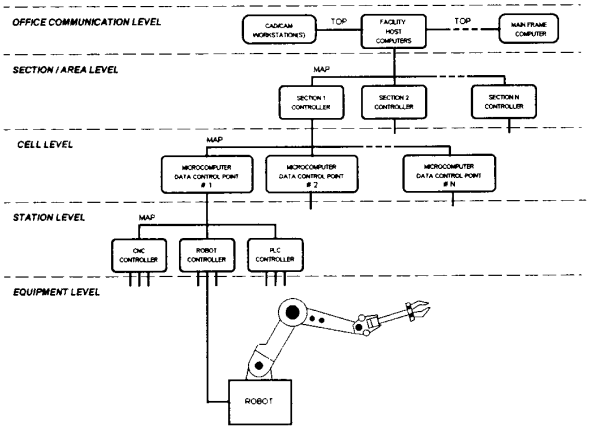


Figure 1: CIM Organizational Hierarchy

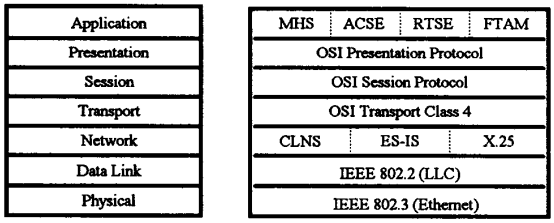


Figure 2: OSI Reference Model with the TOP profile

computer equipments to communicate. MAP first appeared in 1986, when General Motors (GM) begun to apply the concepts of CIM to its operations, and realized that communication was a bottleneck between devices on the plant floor. MAP has been specifically oriented to the needs of CIM, providing the communication that takes place on the factory floor between programmable devices, cell controllers and area or section computers, as depicted in Figure 1. The Boeing Company started work on the set of protocols that have evolved into TOP by the same time GM began defining MAP. TOP's main goal is to encompass everything but the manufacturing environment. It supports the transfer of files between different machines in the production and office environments, specifying standardized data formats. Figure 1 shows TOP's place in the organizational hierarchy and Figure 2, its layered protocol stack.

THE IMPLEMENTATION ARCHITECTURE

High performance and reliable data transfer in manufacturing environment are the main objectives of TOP implementation. Since this system's first protocol implementation work [4], it became clear that a careful implementation was necessary and that memory copy should be avoided. Hence an efficient implementation architecture was defined in order to obtain high performance.

Interface Data Structures

This implementation architecture defines a standard interface between layers (Figure 3). This interface consists of a pair of FIFO queues per connection and a set of data structure access procedures. It aims at normalizing the access to the services offered by adjacent layers, allowing flexibility, modularity and weak coupling between layers. This independence simplifies the division of the tasks of implementation, maintenance and updating of each layer protocol and leads to a better system management.

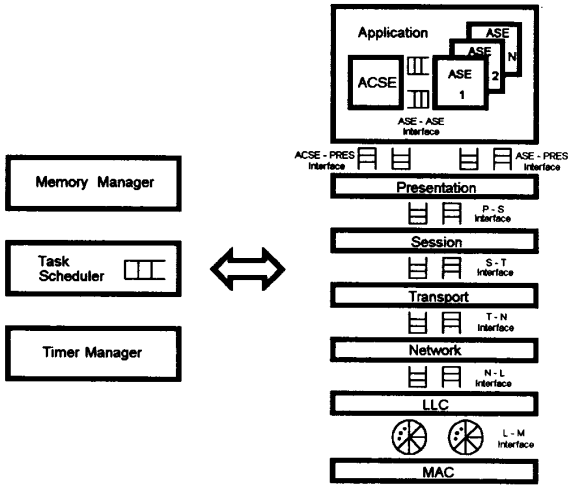


Figure 3: The Implementation Architecture.

Each FIFO element (Figure 4) is represented by a fixed size structure called *Primitive Structure* containing the following three fields:

- Primitive Structure Pointer -- used for linking with the next FIFO element;
- Primitive Name -- name of the incoming or outgoing service primitive;
- Parameter Structure Pointer -- pointer to a structure that contains the current primitive parameters that is called *Parameter Structure.*

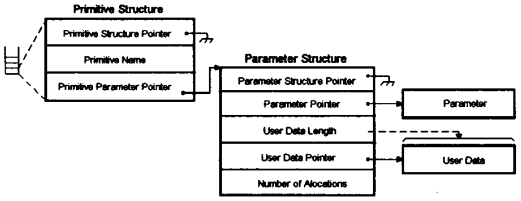
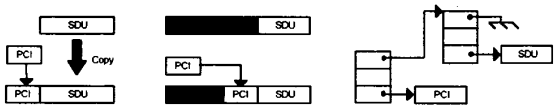


Figure 4: Interface Data Structures.

A connection setup corresponds to instantiating a pair of queues in the interface through where all service primitives corresponding to that connection are transferred.

Protocol Control Information -- PCI

A common processing for all layers, related to the communication between peer entities, is to insert (remove) PCI into (from) SDUs (Service Data Units). There are many methods of inserting PCI to SDUs [5,9], for example, copy, maximum allocation and chaining (Figure 5).



(a) copy (b) maximum allocation (c) chaining

Figure 5: PCI insertion techniques.

The chaining method of inserting PCI (Figure 5c) avoids data copy and waste of memory, consisting in:

1. allocating and filling up a PCI-size memory region;
2. associating this region to a *chaining structure*;
3. chaining this structure to the received SDU structure.

This approach seemed to be the best way of inserting PCI. Each queue is formed by a basic element composed of chained data structures, defined in order to avoid data copy, a relevant factor of degradation in communication system performance. Figure 4 shows the implemented data structures which base upon pointers and sets apart fixed size structures from variable size structures, to make a more efficient memory management possible. To implement this feature, the *Parameter Structure* was defined [6] containing the following five fields:

- Parameter Structure Pointer -- used for linking with other Parameter Structure;
- User Data Pointer -- pointer to (part of) the primitive user data parameter;
- User Data Length -- length in bytes of the respective user data parameter;
- Parameter Pointer -- pointer to the other primitive parameters;
- Number of Allocations.

The *Number of Allocations* field plays an important role for the system. It avoids several retransmission copies when protocols based on error recovery by retransmission were used (Data Link, Transport, Session and RTSE protocols). This field is incremented (decremented) every time a layer allocates (deallocates) that structure. The corresponding retransmission buffer is released from memory only when this field reaches zero. The Memory Manager is responsible for controlling this property.

Memory Management

A specific and optimized Memory Manager [6] was implemented. In order to avoid the excess of fragmentation, this module divides memory into transmission and reception regions. It is possible because transmission and reception flows are independent. These regions are still subdivided in three parts (Primitive Structure, Parameter Structure and data regions), allowing fixed and variable size memory allocations and deallocations.

Task Scheduling

The system's first version had a main procedure that checked whether the interface queues were not empty, i.e., if there were any elements (service primitives) to be processed. This procedure proved to be too much time consuming. A great improvement in performance was achieved with the creation of a Task Scheduler module [6]. This module consists of a single queue holding the sequence of tasks ready to be executed.

Timer Management

Protocol implementation includes timer-based procedures that are used in error recovery routines, connection establishment and release time-out control, etc. The Timer Manager implementation follows [7], where an analogy with a real clock is simulated by a cyclic counter. This implementation proved to be very efficient, avoiding unnecessary logical checks at each hardware interrupt, thus allowing the creation of a great number of run-time timers. In the implemented architecture, a time-out is considered as a service primitive, that is, time-out indication (event) is put into the appropriate interface queue, and the corresponding task into the Task Scheduler queue.

PERFORMANCE MEASUREMENTS

The implemented system is coded in C language and runs on IBM-PC-like personal computer environments using DOS operating system. The basic measurement scenario consisted of two identical 33 MHz 486 DX processors connected to an IEEE 802.3 local area network through NE2000 network adapters.

This performance analysis emphasizes on the throughput achieved by each layer user. The measurement methodology employed consisted in computing, at the originator, the data transfer time

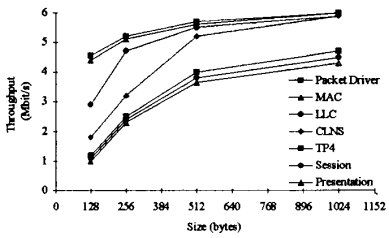


Figure 6: Throughput versus message size.

interval between the first message transmission and receipt of the last message reply. The elapsed time measurement approach makes use of PC tick-based function, providing a 55 ms accuracy. Therefore, in order to make the measurement error negligible, an information amount of 10,000 frames is transferred. In case of connection-oriented protocols, no setup or release time measurements were performed. The measurements also include overheads introduced by lower layer headers. This overhead reaches 33 bytes for Presentation Layer user. In fact, the measurements are lower bound throughputs because they include the user processing time due to message creation and destruction, message insertion (removal) into (out of) queues, memory allocation and deallocation, etc.

Figure 6 shows the throughput, in Mbit/s, for all layer users. Various message sizes are used, ranging from 128 bytes to 1024 bytes.

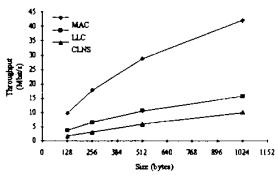


Figure 7: Throughput versus message size (loopback scheme).

The maximum theoretical IEEE 802.3 LAN throughput is 6 Mbit/s and 9.8 Mbit/s for minimum and maximum frame sizes, respectively. A specialized user software was developed to measure the NE2000 Packet Driver access routines' performance.

As depicted in Figure 6, as the frame size increases, throughput reaches approximately 6 Mbit/s. This result shows protocol processing time negligible when compared to transmission time, for large frames; i. e., Packet Driver access routines and LAN adapter constrain system throughput. Hence, only single computer measurements will be considered further here. A special loopback feature was developed at the Packet Driver level, consisting in frame copies from the transmission buffer to the reception buffer.

Figure 7 shows the system's performance improvement achieved with this new configuration. As an example, MAC sublayer throughput was about 43 Mbit/s, that is, approximately 7 times greater than the previous configuration throughput.

All communication controllers require sending data to be in a contiguous memory area. Thus, LLC sublayer performs an unavoidable copy of chained data to a contiguous memory region (transmission buffer in LLC-MAC interface). This data copy is an important performance degradation factor, responsible for a frame size dependent decrease in throughput.

In order to visualize the actual limitation introduced by copy, Figure 8 shows a pie chart for the specific LLC sublayer processing.

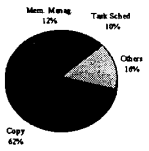


Figure 8: LLC sublayer processing a 1024 frame length.

It is also observed that memory allocation and deallocation and Task Scheduler queue insert procedures spend a considerable processing-time portion.

The Transport Layer is responsible for the greatest protocol performance decrease (Figure 6). This was already expected because it is the Transport Layer in data transfer phase, for the TOP protocol profile, that executes the greater number of functions [8]. The implementation of detection and error recovery services requires the use of a checksum mechanism, and its computational overhead can be quite high. Figure 9 proves the strong dependence between checksumming and frame size.

This protocol requires the transfer of Acknowledgment Transport Protocol Data Units (AK-TPDUs) for each DT-TPDU received, and this procedure overcharges system's processing. Acknowledgments are tied to window flow control, and a window size of 8 was used. Figure 10 emphasizes the influence of acknowledgments' exchange. To perform these measurements, checksum computing was disabled. Different acknowledgment steps were used, allowing one, two, three and four outstanding TPDUs.

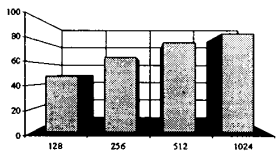


Figure 9: Transport checksum vs. message size.

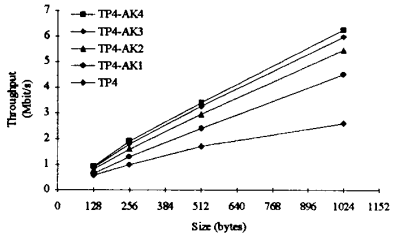


Figure 10: Transport user's throughput versus message size for various acknowledgment steps.

Finally, the Session and Presentation Layers execute, in the data transfer phase, a common processing for all layers: service primitives' receipt, decoding and treatment. They spend a minimum processing time due to interface management (queue checks and elements' insertion or removal), memory management (allocations and deallocations) and state machine execution, which causes a performance loss at each layer.

CONCLUSIONS

In this work an efficient TOP communication system was presented. Its implementation architecture was described, which is based on:

- asynchronous message interlayer communications modeled by paired FIFO queues. This results in a weak coupling between adjacent layers and a high modularity;
- interface data structures with pointers. Copies are avoided for interlayer communication, segmentation and retransmission buffers;
- a memory manager that attains high performance separating transmission and receiving memory regions. Fixed and variable size buffers allocations and deallocations are processed differently. Fragmentation is kept as a minimum;
- a task scheduler that avoids unnecessary queue tests and take advantage of required atomicity of some functions;
- a timer manager efficiently implemented and considered as a service primitive.

The measurement results show that the system performs well attaining almost the maximum throughput at the MAC Layer user (6 Mbit/s). In a loopback configuration this layer throughput has reached 42 Mbit/s. Also in a loopback configuration the throughput for Presentation Layer user was 2 Mbit/s, when TP4 was configured for an acknowledgment step of one and checksum computation. With an acknowledgment step of four and no checksum the Presentation Layer user throughput reaches approximately 6 Mbit/s.

The throughput results are quite good and fit well for most of the nowadays industrial applications.

ACKNOWLEDGMENTS

Prof. Marcelo Lanza, Prof. João Pereira, Luis Baginski, José Rezende, Rainer Schatzmayr, Rogério da Cunha, Fernando de Barros, Calixto Neto, Frederico Liporace and Marcelo Achá have all contributed to the implementation of parts of the system.

REFERENCES

[1] L. J. McGuffin, L. O. Reid and S. R. Sparks, "MAP/TOP in CIM Distributed Computing", *IEEE Network*, 2, 3, pp. 23-31, May 1988.

[2] "MAP 3.0 Implementation Release".

[3] "TOP 3.0 Implementation Release".

[4] L. F. Baginski, F. M. C. de Barros, R. Schatzmayr and O. C. M. B. Duarte, "Implementação e Análise de Desempenho em um Sistema de Transferência de Dados", in *X Congresso da Sociedade Brasileira de Computação*, Vitória, Brazil, pp. 157-169, July 1991.

[5] L. Svobodova, "Implementing OSI Systems", *IEEE J. Sel. Areas Comm.*, 7, 7, pp. 1115-1130, Sept. 1989.

[6] L. F. Baginski, "Ambiente de Implementação para Sistemas de Alto Desempenho", *Master Thesis*, PEE-COPPE/U.F.R.J., Jan. 1992.

[7] G. Varghese and T. Lauck, "Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility", in *Proc. 11th ACM-SIGOPS Symp. on Operating Systems Principles*, Austin, TX, pp. 25-38, 1987.

[8] W. T. Strayer, A. C. Weaver, "Performace Measurements of Data Transfer Services in MAP", *IEEE Network*, 2, 3, pp. 75-81, May 1988.

[9] C. N. Woodside and J. R. Montealegre, "The Effect of Buffering Strategies on Protocol Execution Performance", *IEEE Trans. Comm.*, 37, pp. 545-554, June 1989.