POTENTIAL PERFORMANCE IMPROVEMENTS OF MIL-STD 188-220A THROUGH PARALLELISM *

Thomas P. Way
Cheer-Sun Yang
Lori L. Pollock

Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716

ABSTRACT

Given the currently available high speeds of physical networks, a major challenge to increasing the performance of delivering data across these networks is efficient protocol implementations. The growing use of shared memory multiprocessors as desktop workstations and server platforms suggests that one approach to this problem is to utilize parallelism at the point of the communication bottleneck. Researchers have shown that communication protocols can be parallelized in a number of different ways. This paper examines the applicability of the different parallelization approaches to implementations of the MIL-STD 188-220A protocol standard, the proposed standard for interoperability over Combat Net Radio, and a key to the Army's efforts to digitize the battlefield, and standardize protocols for the physical, data link and network layers.

Keywords: communication protocol, parallelization.

INTRODUCTION

Advancements in high-speed networking technology have transferred the performance bottleneck from the physical transmission media to the transport component, namely the communications protocol and the underlying processor and operating system. Fiber optic data rates in the gigabit range are achievable yet current transport systems are slower by an order of magnitude or more.

*Prepared through collaborative participation in the Advanced Telecommunications/Information Distribution Research Program (ATIRP) Consortium sponsored by the U.S. Army Research Laboratory under Cooperative Agreement DAAL01-96-2-0002.

The growing use of shared memory multiprocessors as desktop workstations and server platforms suggests that one approach to this problem is to utilize parallelism at the point of the communication bottleneck. Indeed, it has been shown that parallelism can be used to improve performance of network transport systems [11]. Since network communications are protocol driven, the natural candidate for parallelization is the protocol itself.

This paper focuses on the potential use of parallelism for enhancing the performance of Army communication protocol implementations. In particular, we investigate the various approaches to parallelizing the data link and intranet layers of MIL-STD 188-220A. MIL-STD 188-220A is the proposed standard for interoperability over Combat Net Radio, is key to the Army's efforts to digitize the battlefield, and standardizes protocols for the physical, data link and network layers. The target model of parallel programming is multithreaded programs for shared memory multiprocessors, given the increasing use of these architectures for single nodes in a communication network.

After describing the primary methods used for parallelization of communication protocols, we give a brief survey of the various experiences in parallelizing protocols. An overview of the MIL-STD 188-220A protocol is presented, with identification of the pertinent functionality. We discuss the applicability of each of the major parallelization methods to MIL-STD 188-220A and conclude with a briefing on our future plans.

PARALLELIZATION METHODS

The two approaches commonly used for improving performance of protocols are [5, 14]:

1. To implement existing protocols such that the processing time is minimized.

2. To design new protocols with mechanisms specifically architected for high-bandwidth, low-delay, low-error-rate communications.

Parallelism can be exploited in either of these approaches. Whether the solution is implemented in hardware, software or a combination, the same general techniques for protocol parallelism apply. Methods for achieving protocol parallelism can be classified as follows [2, 7, 13]:

- **Connection parallelism**: Connection parallelism involves establishment of multiple links, or connections, between end systems comprised of single processors or threads. Speedup is achieved with multiple connections allowing concurrent communication.

- **Packet parallelism**: Packet parallelism assigns each arriving packet to one processor or thread, from a pool of processors or threads, allowing several packets to be processed in parallel. This approach distributes packets across processors, achieving speedup both with multiple and single connections.

- **Pipelined or layered parallelism**: Pipelined or layered parallelism exploits the layering principle common in protocol design, assigning each layer to a separate processor. Performance gains are achieved mainly through pipelining effects.

- **Functional parallelism**: Functional parallelism decomposes functions within a single protocol and assigns them to processing elements. Speedup is achieved when packets require different independent functional processing, which can be performed simultaneously.

- **Message parallelism**: Message parallelism associates a separate process with each incoming or outgoing message. A processor or thread receives a message and escorts it through the parallelized portion of the protocol stack.

No single approach is a win in every situation. Factors that affect the success of one approach over the others include the node architecture, number of connections, thread scheduling policies, whether it is a hardware or software implementation, and the cost of primitives for context switching and locking.

EXPERIENCES IN PARALLELIZED PROTOCOLS

Previous study of protocol parallelization covers nearly all forms of parallelization while focusing on upper layers more than lower layers and the TCP/IP protocol. Functional and packet parallelism tend to be the most often used.

Experimental evaluation of connection-level parallel extensions of TCP/IP and UDP/IP protocol stack implementations demonstrate that connection-level parallel protocol stacks, where connections are mapped to processors or threads, scale well with the number of processors and deliver high throughput [17]. Throughput is best with a one connection per processor or thread approach. Multiplexing can be viewed as a form of connection-level parallelism. Feldmeier's study of multiplexing in communication systems that handle multiple QoS attributes [6] concluded that data streams should be multiplexed at the lowest layer that provides more than one QoS. The HPTB architecture [19] achieves a high degree of connection parallelism by having network attachment units that perform frame handling tasks and a buffering component to allow very high data arrival rates on multiple ports. Both connection-level and packet-level parallelism have been shown to increase the performance of encryption and decryption protocols on a symmetric shared-memory multiprocessor [13].

An experimental performance study of packet-level parallelism on a shared-memory multiprocessor showed that only limited packet-level parallelism can be achieved within a single connection under TCP; however, multiple connections can improve available parallelism [13]. Packet-level parallelism has been used to achieve 1 Gbps rates for end-to-end protocol processing with hardware-support lower-layer processing and no significant assumptions about the protocol [9]. A throughput of 1 Gbps was sustained using as few as 5 processors, but required some additional modest hardware extensions.

La Porta and Schwartz achieved three times the throughput of a serial implementation of the Transport layer by using parallelism based on streams of packets, or blocks of packets [11]. Using threads to shepherd a packet through the protocol stack provided significant

(1.6 to 3.5 times) improvement in performance on 2 to 4 processors over a sequential implementation [7]. Similar results were reported when using packet parallelism on a multiprocessor architecture for OSI layers 2 through 6 [8].

These studies show that the advantages of packet-level parallelism include no interlayer communication overhead, performance improvements regardless of how traffic is distributed over connections, a reasonable degree of synchronization, and no extra context switching overhead. However, significant hurdles remain. Throughput is inhibited by the serialization of processing at higher layers due to the requirement of in-order packet processing at these layers. Also, a large amount of processing is required for transfer syntax conversion. Throughput could be improved by using a standard data representation format, rather than a common transfer syntax, to avoid data type conversion.

By decomposing and then parallelizing the functional components of the TCP protocol, it is possible to achieve modest gains in performance [10]. However, memory bandwidth is a critical factor limiting performance. By overlapping protocol layers via functional parallelism, a speedup of 4 on 8 transputers was achieved [18]. A transport subsystem called PATROCLOS achieves functional parallelism by decomposing the protocol into concurrently executable finite state machines [1].

The T9000 transputer and C104 router technologies were used in a simulation of a high-speed parallel protocol processing architecture based on functional and packet parallelism [2]. A Transputer implementation of the OSI Transport Class 4 (TP4) achieved 100 Mbps performance, but demonstrated that TP4 is too general to fit with high speed protocol requirements [4].

Experiments comparing task-based (funciton and layered) and message-based parallelism [16] on a shared-memory multiprocessor platform reported several significant tradeoffs. Task-based parallelism incurs higher context switching and synchronization overhead, resulting in lower performance. However, task-based threading architectures are easier to implement than message-based process architectures.

In summary, the key limiting factors to success with parallelization of protocols are: (1) when dealing with a single connection under TCP, only very limited packet parallelism is possible, (2) in-order packet processing requirement is a point of serialization, (3) memory bandwidth can be a crucial limiting factor when the data transfer rate exceeds a memory system's access speed, (4) achieving better speedup may require redesigning protocols based on the notion of parallel finite state machines, and (5) the upper layers tend to be the bottleneck limiting performance gains [3, 15].

Notable among the successes are: (1) connection parallelism used when there are one or more processors or threads per connection combined with packet parallelism provides improved performance, (2) special hardware for lower layer processing with packet parallelism can achieve sustained throughput of 1 Gbps with only limited numbers of processors, (3) four times speedup is achievable with functional parallelism of TCP, and (4) upper layers provide far more opportunities for successful parallelization than lower layers.

THE MIL-STD 188-220A STANDARD

The military standard MIL-STD 188-220A "Interoperability Standard for Digital Message Transfer Device (DMTD) Subsystems" [12] protocol specifies interoperability of command, control, communications, computers, and C^4I systems via Combat Net Radio (CNR) on the battlefield. The architecture of MIL-STD 188-220A uses the ISO 7-layer reference model. The April 1995 draft of the standard uses TCP/IP in the Transport and Session Layers and does not address layers 5, 6 or 7. Thus, the focus of our interest in 188-220A is restricted to the physical, datalink, and network layers.

The **physical layer** provides basic control functions for connection activation, maintenance, and deactivation [12]. The standard specifies the use of a frequency hopping packet scheme, output transmission rates ranging in general from 75 to 32000 bits per second (bps), and half-duplex transmission mode. The physical layer may be realized over radio, wireline and/or satellite links.

The **datalink** layer provides control functions to ensure information transfer across the physical channel, framing of data, link encryption, and error control [12]. Four basic modes of communication are specified:

Type 1 - Unacknowledged Connectionless Operation
Type 2 - Connection-mode Operation
Type 3 - Acknowledged Connectionless Operation
Type 4 - Decoupled Acknowledged Connectionless Operation

A significant amount of processing is performed by the

datalink layer. Due in part to the use of TCP/IP in the upper layers, the functionality of the datalink layer was augmented to include operations normally handled in the upper layers and deemed necessary for the protocol. The functionality of the datalink layer includes: flow control, Forward Error Correction coding (FEC) (optional), group and global multicast of messages, priority queuing of messages (urgent, priority, routine), scrambling and unscrambling of data (optional), Time Dispersive Coding (TDC) (optional), and zero-bit insertion (*bit-stuffing*).

The **intranet** or **network** layer accommodates packet routing and information sharing among nodes within the same radio network [12]. The primary functions of the intranet layer are: exchanging topology and connectivity information which is essential for a node to initiate and/or perform intranet relay of packets, packet aging via hop counter and timers, and other typical network layer functions, routing packets between a source and multiple destinations within the same radio network, and update of topology and connectivity information, determined by passively listening to a node's traffic and/or actively exchanging information.

PARALLELIZATION POSSIBILITIES

Possible avenues to parallelism of MIL-STD 188-220A are explored within the framework of the five forms of parallelism mentioned earlier. As the protocol specified by MIL-STD 188-220A is well defined and not apt to change substantially in the near term, the general approach is to create a parallel implementation of the existing protocol.

Connection Parallelism. The low bandwidth of the physical transmission media and the half-duplex communication mode disallow true connection parallelism. For connection parallelism to be successful, multiple simultaneous connections must be established, a scheme for which the physical layer (CNR) and the MIL-STD 188-220A protocol are not designed.

At such time as the proposed 24 Mbps DBS link is incorporated into operations, some form of connection parallelism may be feasible. Assuming separate channels for CNR and DBS, a prerequisite for connection parallelism, concurrent communication, could be achievable. In addition, parallel flow control is inherent in connection parallelism.

Multicasting of messages, and portions of topology and connectivity information updating, conceptually are making use of connection parallelism already. By virtue of CNR being a shared broadcast channel, multiple simultaneous messages are sent from a node to all other nodes within range of the first node. This concurrent one-to-many broadcast has the appearance of connection parallelism, but is clearly limited to a single one-way transmission at any given time. The technology involved in the physical transmission media is not designed for connection parallelism.

Packet Parallelism. Given a large enough amount of processing to be performed at the nodes, packet parallelism is promising. Use of parallelism at the packet level allows multiple packets to be processed concurrently, allowing full utilization of the already loaded communications link.

Functions within the datalink layer that may be suitable for packet parallelism include: Forward Error Correction coding (FEC), Scrambling and unscrambling, Time Dispersive Coding (TDC), and Zero-bit insertion. The appropriateness of parallelizing these functions at the packet level will depend on the existence of inherent parallelism in the algorithms used for each function.

Packet routing is another area that may be particularly well suited to packet parallelism within the MIL-STD 188-220A protocol. Assigning packets to individual processors, those packets requiring routing can be handled simultaneously with packets destined for the local node.

Pipelined Parallelism. Given the potential for asynchronous communication, pipelining can be applied to a packet parallel scheme to further improve protocol performance. As packets are received and processed by the datalink layer, they are passed to the intranet layer for further processing. This processing at the subsequent layer is handled by a separate processor.

Pipelining is most beneficial when the pipeline length is sufficiently long. Since the protocol at hand covers the lower layers only, and performance benefits of pipelining are experimentally proven primarily for upper layers (i.e., transport layer and above), the use of pipelined parallelism may not be as beneficial as some of the other approaches under consideration.

Functional Parallelism. The use of functional parallelism to improve performance of the MIL-STD 188-220A protocol is promising. Functions which may lend themselves well to functional parallelism include: Preparation of individual messages with a combination

of FEC, TDC, Scrambling and prioritizing within the datalink layer, and the topology and connectivity information exchange and update operations.

The standard approach to take is twofold: (1) use profiling information gathered during simulated or actual runs of an implementation of the protocol to identify the most expensive functions, and (2) analyze the algorithms used for each function to identify opportunities for parallelism.

Message Parallelism. Message parallelism relies on the ability to parallelize the protocol stack, or as much of the stack as possible, to be maximally effective. The MIL-STD 188-220A protocol considers only the physical, datalink and intranet layers. Message parallelism, as well as most other forms, perform experimentally best when applied to upper layer protocols. Thus, message parallelism as a means for improving the performance of this particular protocol is not proposed.

FUTURE DIRECTIONS

We are in the process of obtaining an implementation of MIL-STD 188-220A. We plan to experimentally investigate each of the most promising kinds of protocol parallelism on a shared-memory multiprocessor workstation. The results of such experimentation will provide valuable information about the potential benefits of utilizing available parallelism of modern network node processors for enhancing the performance of MIL-STD 188-220A implementations.

References

[1] Torsten Braun Braun and Martina Zitterbart. Parallel transport system design. In *High Performance Networking, IV*, pages 397–412. Elsevier Science Publishers B.V., 1993.

[2] Toong Shoon Chan and Ian Gorton. A parallel approach to high-speed protocol processing. Technical report, University of New South Wales, March 1994.

[3] David D. Clark and David L. Tennenhouse. Architectural considerations for a new generation of protocols. In *ACM SIGCOMM*, pages 200–208, Philadelphia, 1990.

[4] Christophe Diot and Michel Ng.X. Dang. A high performance implementation of OSI transport protocol class 4; evaluation and perspectives. In *15th Conference on Local Computer Networks*, pages 223–230, Minneapolis, Minnesota, 1990.

[5] Willibald A. Doeringer, Doug Dykeman, Matthias Kaiserswerth, Bernd Werner Meister, Harry Rudin, and Robin Williamson. A survey of light-weight transport protocols of high-speed networks. *IEEE Communications Magazine*, 38(11):2025–2039, November 1990.

[6] David C. Feldmeier. Multiplexing issues in communication system design. In *ACM SIGCOMM*, pages 209–219, Philadelphia, 1990.

[7] Murry W. Goldberg, Gerald W. Neufeld, and Mabo R. Ito. A parallel approach to OSI connection-oriented protocols. In *IFIP Workshop Protocols for High Speed Networks*, pages 219–232. Elsevier Science Publishers B.V., 1993.

[8] Mabo R. Ito, Len Y. Takeuchi, and Gerald W. Neufeld. A multiprocessor approach for meeting the processing requirements for OSI. *IEEE Journal on Selected Areas in Communications*, 11(2):220–227, February 1993.

[9] Niraj Jain, Mischa Schwartz, and Theodore R. Bashkow. Transport protocol processing at GBPS rates. In *ACM SIGCOMM*, pages 188–199, Philadelphia, 1990.

[10] O.G. Koufopavlou, A.N. Tantawy, and M. Zitterbart. Analysis of TCP/IP for high performance parallel implementations. In *17th Conference on Local Computer Networks*, pages 576–585, Minneapolis, Minnesota, 1992.

[11] Thomas F. La Porta and Mischa Schwartz. The multistream protocol: A highly flexible high-speed transport protocol. *IEEE Journal on Selected Areas in Communications*, 11(4):519–530, May 1993.

[12] Military Standard. Interoperability standard for digital message transfer device subsystems. (MIL-STD-188-220A), July 1995.

[13] Erich M. Nahum, David J. Yates, James F. Kurose, and Don Towsley. Performance issues in parallelized network protocols. In *Firat Symposium on Operating Systems Design and Implementation*, 1994.

[14] Arun N. Netravali, W.D. Roome, and K. Sabnani. Design and implementation of a high-speed transport protocol. *IEEE Communications Magazine*, 38(11):2010–2024, November 1990.

[15] J.-R. Scherrer and S. Spahni. New opportunities for processing OSI-7 layer protocols using parallel processing (transputers). *International Journal of Bio-Medical Computing*, 34(1):387–398, jan 1994.

[16] Douglas C. Schmidt and Tatsuya Suda. The performance of alternative threading architectures for parallel communication subsystems. *submitted to: Journal of Parallel and Distributed Computing*, pages 1–19, 1996.

[17] David J. Yates, Erich M. Nahum, James F. Kurose, and Don Towsley. Networking support for large scale multiprocessor servers. Technical report, University of Massachuttes, jul 1994.

[18] Martina Zitterbart. High-speed transport components. *IEEE Network Magazine*, pages 54–63, jan 1991.

[19] Martina Zitterbart, Ahmed N. Tantawy, and Dimitrios N. Serpanos. A high performance transparent bridge. *IEEE Transactions on Networking*, 2(4):352–362, aug 1994.