XTP FOR THE NASA SPACE STATION

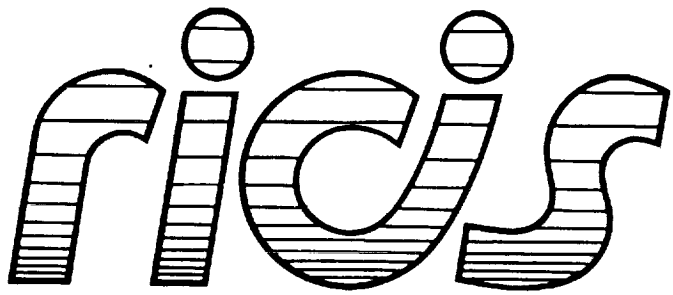
Alfred C. Weaver

Digital Technology

March 1990

Cooperative Agreement NCC 9-16
Research Activity No. SE.31

NASA Johnson Space Center
Engineering Directorate
Flight Data Systems Division



Research Institute for Computing and Information Systems
University of Houston - Clear Lake

T·E·C·H·N·I·C·A·L R·E·P·O·R·T

Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by Alfred C. Weaver and Digital Technology. Dr. George Collins, Associate Professor of Computer Systems Design, served as RICIS technical representative for this activity.

Funding has been provided by the Engineering Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA technical monitor for this activity was Frank W. Miller, of the Systems Development Branch, Flight Data Systems Division, Engineering Directorate, NASA/JSC.

The views and conclusions contained in this report are those of the author and should not be interpreted as representative of the official policies, either express or implied, of NASA or the United States Government.

XTP FOR THE NASA SPACE STATION

Alfred C. Weaver, Director
Computer Networks Laboratory
Department of Computer Science
Thornton Hall
University of Virginia
Charlottesville, Virginia 22903
U. S. A.

ABSTRACT

The NASA Space Station is a truly international effort, and therefore its communications systems must conform to established international standards. For that reason NASA is requiring that each Network Interface Unit implement a full suite of ISO protocols. However, NASA is understandably concerned that a full ISO stack will not deliver performance consistent with the real-time demands of Space Station control systems. Therefore, as a research project, we are investigating whether the Xpress Transfer Protocol (XTP) is a suitable candidate for use alongside a full ISO stack. This paper describes our initial plans for implementing XTP and for comparing its performance to ISO TP4.

XTP FOR THE NASA SPACE STATION

1. BACKGROUND OF THE SPACE STATION PROJECT

The NASA Space Station, now scheduled for launch in the mid-1990s, will be a distributed system supporting a global network of international science, technology, and commercial users. The Space Station Information System, or SSIS, is responsible for providing communications services between end users; users may be men or machines, and they may be located on-station or on the ground. SSIS must interface interoperably with data streams from many different sources and must transport a wide variety of data types and data rates, while at the same time remaining sufficiently flexible that it can accommodate the technology changes which will certainly occur over the Station's expected 30-year lifespan.

One important component of SSIS is the Data Management System (DMS) which provides the hardware resources and software services which support the data processing and communications needs of the Station's systems and payloads. DMS will provide a common operating environment and human-machine interface for operation and control of the Space Station. DMS has defined a set of services which it will provide to the user:

- file transfer and access in space, on the ground, and within international partners' networks
- on-board virtual terminal system
- remote job entry service
- real-time and non-real-time telemetry and telecommand
- electronic mail
- SSIS-wide, application-to-application messaging service
- local and remote access to on-board database
- connectionless multicast distribution of messages (ancillary data)
- globally-unique, location-independent names
- global naming authority

NASA's initial performance requirements for DMS define three grades of service. Grade I specifies a connection-oriented (i.e., virtual circuit) service in which every message is guaranteed to be delivered in order and without duplication. The underlying network must operate with a bit error rate of 10^{-12} or better. Grade II is a datagram service, in which messages may occasionally be lost, or duplicated, or delivered out of sequence. This service must support a bit error rate of better than 10^{-8}. Grade III is a

poorer datagram service, with the same specifications as Grade II except that the bit error rate may rise to 10^{-5}. Each of these grades of service fills a particular need. For example, grade I service would be required for program upload, while grade III would be sufficient for real-time voice. In addition, performance requirements for message latency are being established for four classes of messages: background, normal, isochronous, and emergency. Latency is the time elapsed from the moment the application process requests message transfer until the message arrives at its destination if it is on the same DMS network, or until it arrives at the appropriate gateway if it is bound for an international partner's module or for the space-to-ground RF links. While the exact latency requirements for the four message classes have not yet been finalized, they are expected to be in the range of tens of milliseconds.

Because the Space Station is truly an international project, with active participation from Europeans, Canadians, and Japanese, the communications system *must* be interoperable over multiple vendors and heterogeneous space and ground computer systems. This has led NASA to adopt the ISO OSI services and protocols as being the only hope for achieving international interoperability. But everyone recognizes that therein lies a dilemma. Since OSI was developed in the environment of international, packet-switched, wide area network communications, its design emphasis was on interoperability, not performance. Thus one challenge is to develop hardware and software which can communicate via the ISO protocols and yet achieve the throughput and latency requirements needed for SSIS — this challenge has been undertaken by Honeywell for the ground-based testbed and by IBM for the flight-qualified system. A second challenge is to determine whether or not another, more advanced technology (e.g., XTP) is suitable for use on Space Station and whether it has performance attributes consistent with the needs of real-time systems. This second challenge has been undertaken by the Computer Networks Laboratory at the University of Virginia.

2. WHY XTP?

NASA is understandably concerned that a full ISO protocol stack will not perform adequately for a real-time system. Various published measurements of ISO protocol performance [Janetzky87, Heatley88, Strayer88a, Strayer88b, Svobodova89] suggest that, however adequate ISO protocols may be for general purpose use (e.g., file transfer), they are not generally consider adequate for real-time control systems.

With funding from the United States Naval Ocean Systems Center, we investigated a number of protocol alternatives for real-time systems [Strayer88c], ranging from full seven-layer protocols (MAP, TOP) to transport protocols (ISO TP4, VMTP, XTP, GAM-T-103) to MAC-layer protocols (FDDI, HSRB). While none of these was perfect, we determined that XTP was the closest match to the needs of real-time systems. Those unfamiliar with XTP may refer to [Chesson87a, Chesson87b, Chesson88].

XTP is potentially a much higher performance protocol than ISO TP4 or TCP. Quoting from the XTP Protocol Definition [XTP88]:

> "The functional design for XTP arises from the needs of contemporary and future distributed systems. Existing protocol systems, e.g., TCP and ISO TP4, do not meet these needs. In addition to well-known performance and complexity issues, the most cited problem is that they provide only a "traditional" reliable stream service, whereas distributed systems need reliable real-time arbitrary-sized datagrams. This reflects the need of distributed systems for remote procedure calls, rapid request/response operations, and transaction-based file servers."

Unlike TCP or TP4, XTP is designed from the outset to be implementable in VLSI hardware, and its technology is designed to scale from 10 Mbit/s (e.g., Ethernet) to 100 Mbit/s (e.g., FDDI) to 1 Gbit/s networks. XTP is equally applicable to LANs, MANs, and WANs.

3. XTP IMPLEMENTATION PLANS

Our XTP design has only begun, so its description herein is necessarily incomplete. Figure 1 shows our strawman architecture.

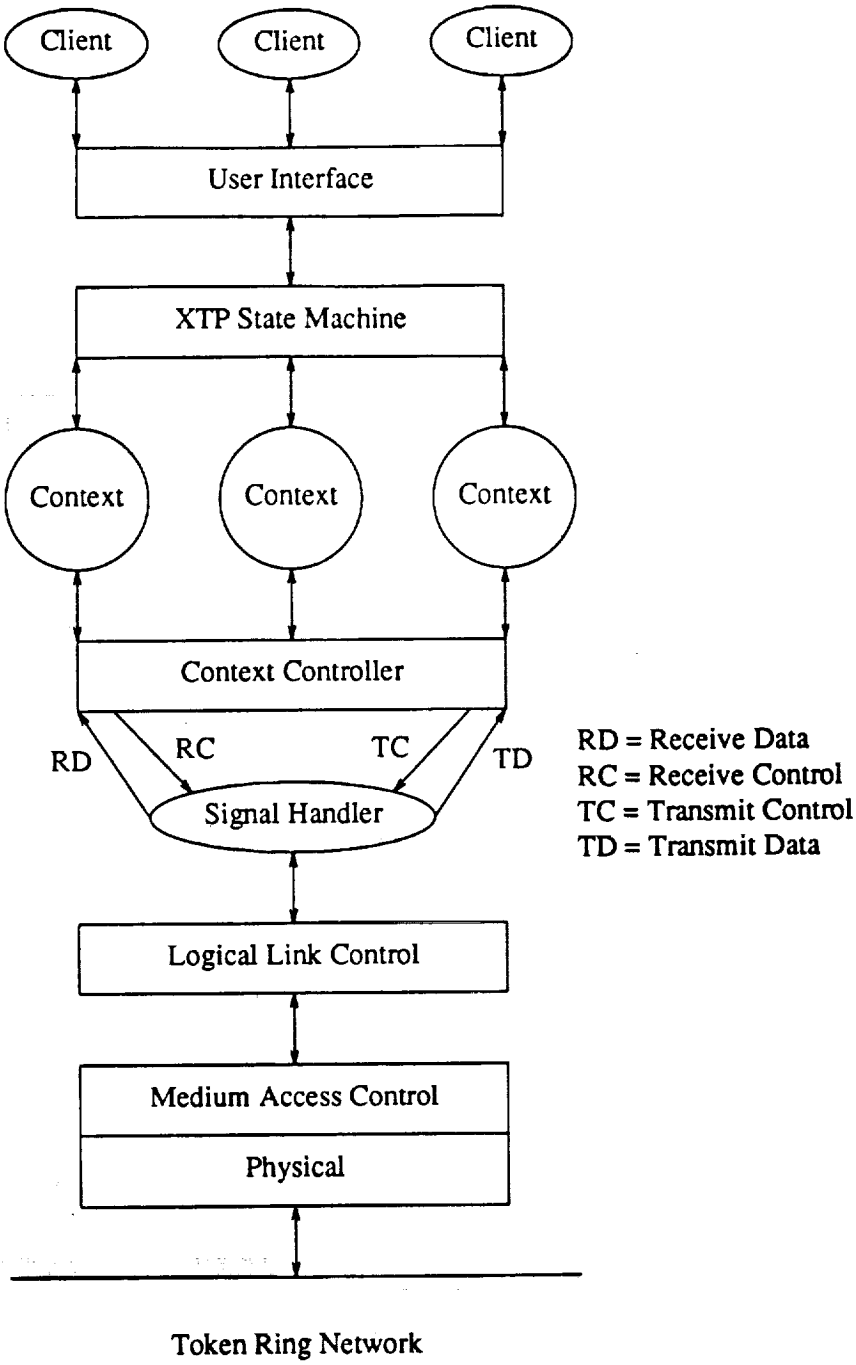


Figure 1.
Strawman Architecture for XTP

Clients are users of the XTP service. They may be application processes in their own right (such as real-time control programs), or they may be other communications services (such as the ISO session layer).

The *User Interface* is a set of communication services provided by XTP to the user. At the moment there is no official XTP service definition, so we are developing our own. One idea under consideration is that the user interface may look like a control block in which the user specifies an operation (e.g., send data), a size, and a pointer to the head of a buffer chain.

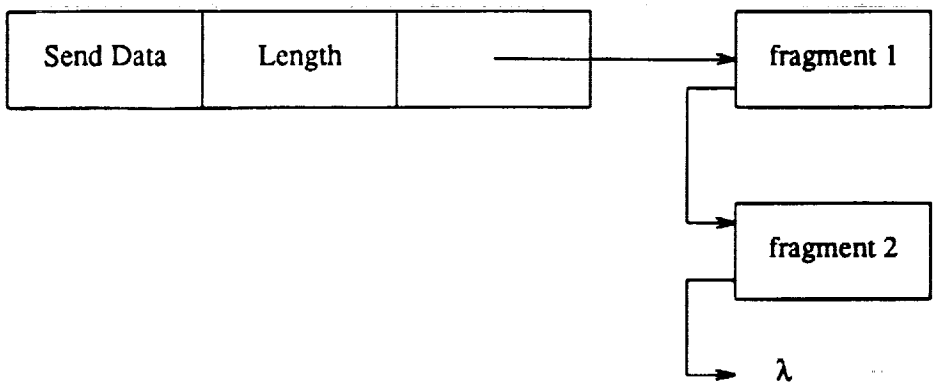


Figure 2.
Possible User Interface Data Structure

By configuring the buffer as a buffer chain, we believe that we can achieve memory economy for short (e.g., control) messages while still supporting arbitrarily large messages (e.g., files). A user wanting to transmit at 64MB file will have to segment it somewhere, so the linked segment approach provided by the buffer chain provides a conceptually simple way for the user (or his operating system) to accomplish it.

The *XTP State Machine* would be our code implementing XTP version 3.3 as per the definition in [XTP88]. We are making every effort to code XTP as the Finite State Machine (FSM) which it is. The advantages to us are three-fold: (1) we would expect our best performance to emerge from a FSM implementation; (2) the FSM representation will ease our eventual conformance testing against the XTP Software Reference Model (SRM); (3) a FSM implementation would allow our code to run as a micro-controller (similar to the Protocol Engine concept, but without the custom VLSI).

A *Context* is an XTP term meaning an active connection or datagram. A *Context Record* is created for each new connection or datagram processed. The XTP state machine manipulates the context record to modify the status of a connection.

The *Context Controller* is our code to multiplex and demultiplex multiple contexts (connections) across a single Logical Link Control (LLC) interface. It is a small finite state machine which performs two primary operations:

(1) When receiving data it performs a lookup (using the *key* field in the XTP header) to identify the proper context to be associated with the incoming message. If the connection has already been established, then the proper context record is found and the message is enqueued for that context. If this is a datagram or the first message of a connection, then a new context record is created. Various algorithms are under consideration to provide a fast lookup service (balanced trees, hash tables, etc.) but no decision has yet been made.

(2) When transmitting data the FSM makes a departure from standard protocol coding practice. When a message to be transmitted (either data or acknowledgement) is received from the XTP state machine, the context controller will indeed enqueue the message for transmission, but will not frame the message or otherwise initiate the transmission process. Instead, it continues to collect messages for transmission until it receives a signal from the *Signal Handler* advising that the transmitter is idle and needs work. Only then are the messages passed to the LLC for framing and transmission. The advantage of this technique is that we are still prepared to transmit at every legitimate opportunity (i.e., whenever the transmitter becomes idle) and we will transmit all enqueued messages (data and acknowledgements) in the minimum number of frames. By "piggybacking" acknowledgements with the data we both reduce the number of packets emitted by the transmitter and we provide the most recent acknowledgement information (for example, two acknowledgements for sequential messages can be combined into one for the most recent message).

The *Signal Handler* is another small finite state machine which further decouples the transmit and receive processes from the packetization process in the LLC. When the LLC receives, deframes, error checks, and accepts a packet it signals the *Context Controller* that one or more packets are in the receive queue. Only when the *Context Controller* is ready to process those messages are they physically delivered to it. When the LLC transmitter goes idle, the *Signal Handler* signals the *Context Controller* that the transmitter needs more work. All accumulated data and acknowledgements are then moved to the LLC transmitter where they are examined and then packetized into the fewest possible packets.

The *Signal Handler* operates a circular queue of buffers on both its transmit and receive sides. An interesting question is how best to operate the receive buffers. Suppose that the receive buffers are temporarily full and a new message arrives. With classical protocols the last message is simply lost due to buffer starvation (this can happen with or without flow control, depending upon the timing). In our design we want to assure that there is always an available receive buffer, but if the receive buffer pool is full, which buffer should be reused? If the data being sent is strictly sequential, then we should delete the most recently received message (because the oldest message is most important since it affects the sequencing and acknowledgements). If, on the other hand, this is datagram-type traffic (e.g., sensor readings or effector updates in which the most recent information is most valuable), then we should delete the oldest message in the queue. But the kind of data being transmitted is, in general, unknown, and "reading the mail" to determine what type of data a packet carries is not generally considered good procedure either. Thus, receiver buffer management is a timely research issue.

Similarly, it is unknown whether the receiver queue, described above as a simple circular buffer, should actually be a priority queue. Our studies of priority systems thus far [Peden87, Peden88a, Peden88b] have shown that *medium access priority strategies*, for example the "token priority" and "priority reservation" fields of IEEE 802.5 or SAE HSRB, make only small differences in overall message latency in the general case. However, *processing priority* within the protocol stack makes a great deal of difference, leading us to believe that our receiver should actually implement a priority queue. So this, too, in a fruitful research area.

We are building XTP upon a *Logical Link Control* (LLC) which is already fully functional. Our LLC design and its performance characteristics have already been reported in [Simoncic88a, Simoncic88b, Cain89], but in summary, our LLC is optimized to support real-time data transfer. In our SeaNET project [Simoncic88b], the LLC operates on PCs and PC/ATs over an IEEE 802.5 token ring. On an 8 MHz PC/AT, a single SeaNET station supports a continuous throughput in excess of 1.5 Mbit/s, a message transmission rate of 423 100-byte messages/second, and a true end-to-end delay (user memory to user memory, including all operating system overhead and all network transit time) of 3.8 ms for 100-byte messages. In our AirNET project [Cain89], a 6 MHz Intel 286 host on a Multibus I, using a Proteon ProNET-10 token ring, can support a continuous throughput of 1.8 Mbit/s, a message transmission rate of 250 100-byte messages/second, and a true end-to-end delay of 2.5 ms for 100-byte messages. Initial experiments with a 16 MHz Intel 386 version of AirNET suggest that we can eventually decrease the message transmission delay to about 1 ms.

We are presently converting our SeaNET and AirNET LLC's to operate on a 25 MHz ALR Flex-cache using very high speed cache memory. Our LLC is small enough to fit entirely in cache. We are also planning to implement it on a 25 MHz Motorola 68020 using a VMEbus during summer 1989. We think that these LLC implementations will provide adequate support for our XTP implementation until we advance to FDDI and gigabit LANs.

4. TESTING

Our group has much experience with protocol testing, evaluation, and performance measurement. We expect our initial implementation of XTP to occur on a 25 MHz ALR Flexcache using the 802.5 token ring, and our first tests will be performance studies of XTP vs. ISO TP4 in that environment. Our second implementation is expected to utilize a 25 MHz Motorola 68020 system, a VMEbus, and an FDDI network. Our second suite of tests will be XTP vs. ISO TP4 in this environment. Another set of tests will assess interoperability between our XTP implementation and the Software Reference Model (SRM) distributed by Protocol Engines Inc.

Plans beyond this point are uncertain, but it is our intent to demonstrate XTP operating on a very high speed (order of 1 Gbit/s) WAN backbone as part of the prototype for the National Research Network now being sponsored by the United States' Defense Advanced Research Projects Agency (DARPA), the National Science Foundation (NSF), and the Corporation for National Research Initiatives (NRI).

REFERENCES

[Cain89] Brendan Cain, Alfred Weaver, Robert Simoncic, and Alex Colvin, "AirNET: A Real-Time Communications Network for Aircraft," MILCOM'89, Boston, MA, October 1989 (submitted).

[Chesson87a] Greg Chesson, "Protocol Engine Design," *Usenix Conference Proceedings*, June 1987.

[Chesson87b] Greg Chesson, "The Protocol Engine Project," *Unix Review*, September 1987.

[Chesson88] Greg Chesson, "XTP/PE Overview," *Proc. 13th Local Computer Networks Conference,*, October 1988.

[Heatley88] Sharon Heatley and Daniel Stokesberry, "Measurements of a Transport Implementation Running Over an IEEE 802.3 Local Area Network," *Proc. IEEE Computer Networking Symposium*, Washington, D.C., April 1988.

[Janetzky87] D. Janetzky and K. S. Watson, "Performance Evaluation of the MAP Token Bus in Real-Time Applications," in *Advances in Local Area Networks*, IEEE Press, 1987.

[Peden87] Jeffery Peden and Alfred Weaver, "Performance of Priorities on an 802.5 Token Ring," *Proc. ACM SIGCOMM*, Stoweflake, Vermont, August 1987.

[Peden88a] Jeffery Peden and Alfred Weaver, "Are Priorities Useful on an IEEE 802.5 Token Ring?" *IEEE Transactions on Industrial Electronics*, Vol. IE-35, No. 3, August 1988.

[Peden88b] Jeffery Peden and Alfred Weaver, "The Utilization of Priorities on Token Ring Networks," *Proc. 13th Conference on Local Computer Networks*, Minneapolis, MN, October 10-12, 1988.

[Simoncic88a] Robert Simoncic, Alfred Weaver, Brendan Cain, and Alex Colvin, "SHIPNET: A Real-Time Local Area Network for Ships," *Proc. 13th Local Computer Networks Conference,"* Minneapolis, MN, October 1988.

[Simoncic88b] Robert Simoncic, Alfred Weaver, Brendan Cain, and Alex Colvin, "SeaNET: A Real-Time Communications Network for Ships," *Proc. Symposium on Mini- and Microcomputers*, Miami Beach, FL, December 1988.

[Strayer88a] Timothy Strayer and Alfred Weaver, "Performance Measurement of Data Transfer Services in MAP," *IEEE Computer*, May 1988.

[Strayer88b] Timothy Strayer and Alfred Weaver, "Performance Measurements of Motorola's Implementation of MAP," *Proc. 13th Local Computer Networks Conference*, Minneapolis, MN, October 1988.

[Strayer88c] Timothy Strayer and Alfred Weaver, "Evaluation of Real-Time Transport Protocols," Computer Science Report No. TR-88-21, University of Virginia, October 1988.

[Svobodova89] Liba Svobodova, "Measured Performance of Transport Service in LANs," *Computer Networks and ISDN Systems*, January 1989 (submitted).

[XTP88] Protocol Engines Inc., "XTP Protocol Definition," version 3.3, December 1988.