Best Available Copy
Modification to Internal Protocol Model

by

William R. Bush

December, 1978

Sponsored by
Defense Advanced Research Projects Agency (DoD)
ARPA Order No. 3079.3
Monitored by Naval Electronic Systems Command
Under Contract #N00039-78-G-0020

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.
**Title:** Modification to Internal Protocol Model

**Author:** William R. Bush

**Performing Organization:**
Center for Research in Computing Technology
Harvard University
Cambridge, Massachusetts 02138

**Controlling Office Name and Address:**
Department of the Navy
Naval Electronic Systems Command
Washington, D.C. 20360

**Report Date:** December 1978

**Number of Pages:** five

**Distribution Statement:** unlimited

**Abstract:**
An abstract model of the interprocess communication system MSG has been written as a test of software development and maintenance tools. This report evaluates the maintenance effort involved in changing the model to use the Transmission Control Program (TCP) network protocol.
Modification to Internal Protocol Model
by
William R. Bush

An abstract model of the interprocess communication system MSG has been written as a test of software development and maintenance tools. This report evaluates the maintenance effort involved in changing the model to use the TCP network protocol.

There are two aspects to converting to TCP. The first is disruption, that is, the minimum amount of work needed to get the old system to work in the new environment. The second is optimization made possible by the new environment, involving enhancements to the functioning system. The abstract model would be disrupted not at all by changing to TCP, and facilitates possible optimizations.

The model's interface to the network is its general paradigm for input-output, the channel. A channel is a data path between two processes. The primitives for manipulating channels are CHOPEN, with which channels are opened, SEND, for sending data, RECEIVE, for receiving data, and CHCLOSE, with which channels are closed. The notion of a channel is that of a TCP connection, and the channel primitives map directly to the TCP primitives OPEN, SEND, RECEIVE and
CLOSE, respectively. A channel identifier, the name given a channel, maps to the TCP socket. The only channel idea not in TCP is that of specifiable data byte size. All TCP connections pass only eight-bit bytes. This restriction does not affect the model, since inter-MSG network connections need only pass eight-bit bytes. In sum, changing to TCP would not change the abstract model at all, and the refinement of the four channel primitives to their TCP counterparts would be straightforward.

Changing to TCP would make possible substantial optimization of the inter-MSG protocol and a corresponding simplification of the abstract model. The fundamental idea is to model inter-MSG transactions in the same way as local transactions, with a separate channel per pending event. This would eliminate the negotiation currently required by the inter-MSG protocol. The optimization is made possible by TCP's reliable transmission feature, which obviates the need for inter-MSG confirmation messages.

In the abstract model, each MSG user primitive that cannot be satisfied immediately (a pending event), has associated with it a channel between the user process and MSG, used for passing the final result of the primitive to the user. In a similar manner each pending event can have associated with it a network channel to the appropriate remote MSG, over which inter-MSG communication passes. In the case of messages and alarms, after the user executes a
send primitive MSG executes a non-blocking channel SEND, which automatically opens the channel. When the receiving MSG is prepared to receive a message or alarm, it executes a corresponding RECEIVE. The process class and instance of the sender and receiver are encoded in their respective channel identifiers (socket numbers). When both automatic opens complete the data are transmitted and the channel closed. No confirmation from the receiving MSG is necessary since TCP will see to that. The case of direct connections is similar, the difference being that connection identifiers are exchanged and the channels are then passed to the user processes. The timing of pending events is done through timing the associated network channel primitive. The channel primitive is given the same timeout interval as the pending event. If the primitive fails, the pending event is aborted.

The above discipline simplifies the abstract model considerably, particularly the queue management routines. Twenty three routines are reduced to eleven -- the four message and four alarm routines that initiate output (EnQOutput...), initiate input (EnQReceive...), complete output (Record...Ok), and complete input (EnQInput...), and the three direct connection routines that initiate a connection (EnQOutputOpenConn), complete a connection (EnQInputOpenConn), and close a connection (EnQOutputCloseConn). These routines remain relatively unchanged. The lower level decision routines (exempli
gratia, HoldOrRejectMess) become unnecessary. The interface routine DeliverToRemoteHost is changed to reflect the channel strategy. The network server routines are simplified, since formatted MSG protocol items are no longer necessary. The cancelling of failed pending events proceeds as before, but the timer process is no longer needed, since it is subsumed by the timed channel primitives.
References

