Network Working Group
Request for Comments: 675
NIC: 2
INWG: 72

Vinton Cerf
Yogen Dalal
Carl Sunshine
December 1974

SPECIFICATION OF INTERNET TRANSMISSION CONTROL PROGRAM

December 1974 Version

1. INTRODUCTION

This document describes the functions to be performed by the internetwork Transmission Control Program [TCP] and its interface to programs or users that require its services. Several basic assumptions are made about process to process communication and these are listed here without further justification. The interested reader is referred to [CEKA74, TOML74, BELS74, DALA74, SUNS74] for further discussion.

The authors would like to acknowledge the contributions of R. Tomlinson (three way handshake and Initial Sequence Number Selection), D. Belsnes, J. Burchfiel, M. Galland, R. Kahn, D. Lloyd, W. Plummer, and J. Postel all of whose good ideas and counsel have had a beneficial effect (we hope) on this protocol design. In the early phases of the design work, R. Metcalfe, A. McKenzie, H. Zimmerman, G. LeLann, and M. Elie were most helpful in explicating the various issues to be resolved. Of course, we remain responsible for the remaining errors and misstatements which no doubt lurk in the nooks and crannies of the text.

Processes are viewed as the active elements of all HOST computers in a network. Even terminals and files or other I/O media are viewed as communicating through the use of processes. Thus, all network communication is viewed as inter-process communication.

Since a process may need to distinguish among several communication streams between itself and another process [or processes], we imagine that each process may have a number of PORTs through which it communicates with the ports of other processes.

Since port names are selected independently by each operating system, TCP, or user, they may not be unique. To provide for unique names at each TCP, we concatenate a NETWORK identifier, and a TCP identifier with a port name to create a SOCKET name which will be unique throughout all networks connected together.

A pair of sockets form a CONNECTION which can be used to carry data in either direction [i.e. full duplex]. The connection is uniquely identified by the <local socket, foreign socket> address pair, and the same local socket can participate in multiple connections to different foreign sockets [see Section 2.2].

Processes exchange finite length LETTERS as a way of communicating; thus, letter boundaries are significant. However, the length of a letter may be such that it must be broken into FRAGMENTS before it can be transmitted to its destination. We assume that the fragments will normally be reassembled into a letter before being passed to the receiving process. Throughout this document, it is legitimate to assume that a fragment contains all or a part of a letter, but that a fragment never contains parts of more than one letter.

We specifically assume that fragments are transmitted from Host to Host through means of a PACKET SWITCHING NETWORK [PSN] [ROWE70, POUZ73]. This assumption is probably unnecessary, since a circuit switched network could also be used, but for concreteness, we explicitly assume that the hosts are connected to one or more PACKET SWITCHES [PS] of a PSN [HEKA7O, POUZ74, SCWI71].

Processes make use of the TCP by handing it letters. The TCP breaks these into fragments, if necessary, and then embeds each fragment in an INTERNETWORK PACKET. Each internetwork packet is in turn embedded in a LOCAL PACKET suitable for transmission from the host to one of its serving PS. The packet switches may perform further formatting or other operations to achieve the delivery of the local packet to the destination Host.

The term LOCAL PACKET is used generically here to mean the formatted bit string exchanged between a host and a packet switch. The format of bit strings exchanged between the packet switches in a PSN will generally not be of concern to us. If an internetwork packet is destined for a TCP in a foreign PSN, the packet is routed to a GATEWAY which connects the origin PSN with an intermediate or the destination PSN. Routing of internetwork packets to the GATEWAY may be the responsibility of the source TCP or the local PSN, depending upon the PSN Implementation.

One model of TCP operation is to imagine that there is a basic GATEWAY associated with each TCP which provides an interface to the local network. This basic GATEWAY performs routing and packet reformatting or embedding, and may also implement congestion and error control between the TCP and GATEWAYS at or intermediate to the destination TCP.

At a GATEWAY between networks, the internetwork packet is unwrapped from its local packet format and examined to determine through which network the internetwork packet should travel next. The internetwork packet is then wrapped in a local packet format suitable to the next network and passed on to a new packet switch.

A GATEWAY is permitted to break up the fragment carried by an internetwork packet into smaller fragments if this is necessary for transmission through the next network. To do this, the GATEWAY produces a set of internetwork packets, each carrying a new fragment. The packet format is designed so that the destination TCP may treat fragments created by the source TCP or by intermediate GATEWAYS nearly identically.

The TCP is responsible for regulating the flow of internetwork packets to and from the processes it serves, as a way of preventing its host from becoming saturated or overloaded with traffic. The TCP is also responsible for retransmitting unacknowledged packets, and for detecting duplicates. A consequence of this error detection/retransmission scheme is that the order of letters received on a given connection is also maintained [CEKA74, SUNS74]. To perform these functions, the TCP opens and closes connections between ports as described in Section 4.3. The TCP performs retransmission, duplicate detection, sequencing, and flow control on all communication among the processes it serves.

2. The TCP INTERFACE to the USER

2.1 The TCP as a POST OFFICE

The TCP acts in many ways like a postal service since it provides a way for processes to exchange letters with each other. It sometimes happens that a process may offer some service, but not know in advance what its correspondents' addresses are. The analogy can be drawn with a mail order house which opens a post office box which can accept mail from any source. Unlike the post box, however, once a letter from a particular correspondent arrives, a port becomes specific to the correspondent until the owner of the port declares otherwise.

In addition to acting like a postal service, the TCP insures end-to-end acknowledgment, error correction, duplicate detection, sequencing, and flow control.

2.2 Sockets and Addressing

We have borrowed the term SOCKET from the ARPANET terminology [CACR70, MCKE73]. In general, a socket is the concatenation of a NETWORK identifier, TCP identifier, and PORT identifier. A CONNECTION is fully specified by the pair of SOCKETS at each end since the same local socket may participate in many connections to different foreign sockets.

Once the connections is specified in the OPEN command [see section 2.3.2], the TCP supplies a [short] Local Connection Name by which the user refers to the connection in subsequent commands. In particular this facilitates using connections with initially unspecified foreign sockets.

TCP's are free to associate ports with processes however they choose. However, several basic concepts seem necessary in an implementation. There must be well known sockets [WKS] which the TCP associates only with the "appropriate" processes by some means. We envision that processes may "own" sockets, and that processes can only initiate connections on the sockets they own [means for implementing ownership is a local issue, but we envision a Request Port user call, or a method of uniquely allocating a group of ports to a given process, e.g. by associating the high order bits of a port name with a given process.]

Once initiated, a connection may be passed to another process that does not own the local socket [e.g. from logger to service process]. Strictly speaking this is a reconnection issue which might be more elegantly handled by a general reconnection protocol as discussed in section 3.3. To simplify passing a connection within a single TCP, such "invisible" switches may be allowed as in TENEX systems.

Of course, each connection is associated with exactly one process, and any attempt to reference that connection by another process will be signaled as an error by the TCP. This prevents stealing data from or inserting data into another process' data stream.

A connection is initiated by the rendezvous of an arriving internetwork packet and a waiting Transmission Control Block [TCB] created by a user OPEN, SEND, INTERPUPT, or RECEIVE call [see section 2.3]. The matching of local and foreign socket identifiers determines when a successful connection has been initiated. The connection becomes established when sequence numbers have been synchronized in both directions as described in section 4.3.2.

It is possible to specify a socket only partially by setting the PORT identifier to zero or setting both the TCP and PORT identifiers to zero. A socket of all zero is called UNSPECIFIED. The purpose behind unspecified sockets is to provide a sort of "general delivery" facility [useful for logger type processes with well known sockets].

There are bounds on the degree of unspecificity of socket identifiers. TCB's must have fully specified local sockets, although the foreign socket may be fully or partly unspecified. Arriving packets must have fully specified sockets.

We employ the following notation:

x.y.z = fully specified socket with x=net, y=TCP, z=port

x.y.u = as above, but unspecified port

x.u.u = as above, but unspecified TCP and port

u.u.u = completely unspecified

with respect to implementation, u = 0 [zero]

We illustrate the principles of matching by giving all cases of incoming packets which match with existing TCB's. Generally, both the local (foreign) socket of the TCB and the foreign (local) socket of the packet must match.

	TCB local	TCB foreign	Packet local	Packet foreign
(a)	a.b.c	e.f.g	e.f.g	a.b.c
(b)	a.b.c	e.f.u	e.f.g	a.b.c
(c)	a.b.c	e.u.u	e.f.g	a.b.c
(d)	a.b.c	u.u.u	e.f.g	a.b.c

There are no other legal combinations of socket identifiers which match. Case (d) is typical of the ARPANET well known socket idea in which the well known socket (a.b.c) LISTENS for a connection from any (u.u.u) socket. Cases (b) and (c) can be used to restrict matching to a particular TCP or net.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.