Cohen

IEN 119

ST - A Proposed Internet Stream Protocol

by

James W. Forgie

M. I. T. Lincoln Laboratory

7 September 1979

1.0 INTRODUCTION

The internet stream protocol (ST) described in this document has been developed to support efficient delivery of streams of packets to either single or multiple destinations in applications requiring guaranteed data rates and controlled delay characteristics. The principal applications with these requirements are point-to-point speech communication and voice conferencing. While ST has been developed as a part of the ARPA Internet Program and has been formulated as an extension to the presently defined Internet Protocol (IP), it is not likely to find useful application in the current ARPA internet environment where the networks and gateways lack the capacity to handle significant speech communication. Instead, ST is aimed at application in wideband networks, in particular those intended to carry a large fraction of packet voice in their traffic mixes. Work is currently underway on such networks both for local access and long haul use. These networks will serve as vehicles for research on techniques for flow and traffic control and as testbeds for evaluating the potential of packet technology for providing economical speech communication. The design of the ST protocol represents a compromise among the sometimes conflicting requirements of compatibility with the existing IP and the gateways which handle it, the need for flexibility in supporting flow and traffic control research, and transmission efficiency.

The concepts in this protocol originated in the deliberations of a working group consisting of Danny Cohen, Estil Hoversten, and the author. They have been influenced by interactions with many other people. In order to examine the cost and feasibility of the protocol, the author has fleshed out some aspects of the protocol in detail. The other working group participants have not had an opportunity to approve or modify these detailed aspects of the protocol, and consequently all responsibility for them lies with the author.

The state of the protocol is such that, while there are still details to be worked out, implementation could begin if the protocol were acceptable to those interested.

2.0 MOTIVATION AND GENERAL DESCRIPTION

It is reasonable to ask why a new protocol is required to handle applications such as point-to-point speech and voice conferencing. This section addresses that question and begins with a brief statement of the requirements for packet speech communication. They are:

1. The network must be able to keep up with the data rate requirements of the speech terminal. Because no bits need be transmitted during silent intervals, the average data rate for conversational speech can be expected to be between 40 and 50% of the peak data rate for commonly used constant-rate encoding techniques. Experimental variable-rate encoding techniques have exhibited higher peak-to-average ratios. The network must be able to sustain the peak rate for the duration of talkspurt that can be as long as 20 seconds.

2. The stream of packets containing a talkspurt (a continuous segment of speech between silent intervals) must be delivered with a delay dispersion whose spread does not exceed some value that can be estimated with a high probability of success prior to the start of the talkspurt. Since the individual packets of the spurt will experience different delays as they pass through the net, delay must be added at the receiver to allow continuous speech to be played out for the listener. It is necessary to predict the value of this smoothing delay before starting to play out the talkspurt. Packets that are delayed more than the predicted worst-case value will arrive too late to be used, and gaps will occur in the output speech.

3. Overall delay should be kept low. If the overall round-trip delay is less than about 1/4 second, conversations are carried out in a "normal" fashion with considerable feedback from "listener" to "talker" taking place. When greater delay is experienced, people switch to a more formal mode in which feedback utterances are mostly suppressed, and the listener generally waits until the talker indicates that he has finished before saying anything. User satisfaction declines with increasing delay, but systems remain usable for delays as long as several seconds.

4. The amount of speech for any one talker contained in a packet (the basic unit subject to transmission loss) should be kept small. The loss of small (50 msec or less) chunks of speech produces a degradation of quality, but sentence intelligibility tends to be

preserved up to fairly high percentage losses. Larger chunks of speech represent whole syllables or words, and their loss can change the meaning of sentences.

5. Listeners will tolerate some packet loss without downgrading the acceptability of the system, but the probability of loss due to either failed or late delivery must be kept in the order of 1% or less to be considered acceptable for everyday (non-crisis) use.

6. As a network approaches its load limit it should reject (or hold off) new offered load on a call basis not on an individual packet basis. Continuing to accept new calls beyond capacity can result in unsatisfactory communication for many users.

7. If packet-switched speech transmission is to become economically competitive with circuit-switched transmission, a further requirement must be met. The product of packet efficiency and average link utilization must equal or exceed the efficiency of circuit switching. That efficiency is defined as one minus the fraction of the time that silence occurs in conversational situations. Estimates of this fraction for real-world conversations give values for efficiency between 40 and 50%. We will use 45% as a convenient figure for comparative purposes. Packet switching can easily take advantage of the silent intervals in a conversation by not transmitting packets, but that advantage may be lost through the combination of overhead bits in packet headers (packet efficiency) and the difficulty of operating communication links at high average utilization while keeping queueing delays within reasonable bounds.

Conventional datagram networks are unsatisfactory for speech communication except under conditions of light overall load or where speech constitutes a small fraction of the overall load and can be given priority service. The difficulty with datagram nets comes from their inability to provide the controlled delay and guaranteed data rate required for speech. Delay increases with offered load, slowly at light load, but dramatically as average load approaches capacity. Flow control strategies tend to be aimed at buffer management and fairness goals, both of which will operate to restrict the effective data rate available to an individual user as load increases. Traffic control strategies are mainly concerned with congestion control and are primarily defensive, resulting in offered datagrams being held off or refused when difficulties are detected. Unfortunately for the speech user, by the time congestion is detected, it is already too late. For satisfactory speech

service, congestion due to overload must be prevented. Since a datagram net has no knowledge of the a priori requirements of users, it cannot develop traffic control strategies to meet these requirements.

Another disadvantage of datagrams for speech is their packet efficiency. The speech content of an individual user packet can be anything from 50 or so bits up to 1200 or 1300 bits depending upon the speech digitization technique in use. The need to carry full source and destination addresses as well as other packet-handling information in each packet penalizes datagrams relative to other packet and circuit switching techniques. In the internet case the penalty is worse since two sets of header information have to be carried.

For example, IP datagrams on SATNET carrying 40-msec chunks of 16-kbps speech (a reasonaable chunk size and popular data rate) would have a packet efficiency of about 56% and would require utilization factors of about 80% to break even with respect to circuit switching. It is unlikely that delay characteristics would be satisfactory at this level of load.

The goal of the ST design effort has been to attempt to overcome both of the difficulties associated with datagrams. ST uses abbreviated internet headers and also allows speech from many talkers to be aggregated into single packets for transmission on wide-band networks where such aggregation is possible. For the case of ST messages on a wide-band SATNET each carrying ten 40-msec chunks of 16-kbps speech for ten different talkers, packet efficiency would be about 86% allowing break-even link utilization to occur at 52%, a much more comfortable level for assuring desirable delay characteristics.

Overcoming the inability of datagram nets to maintain data rates and delay characteristics as offered load increases is more difficult to achieve than improving packet efficiency. Circuit switching solves the problem by dedicating communication capacity to individual streams. The goal of ST is to support traffic control policies that match stated user requirements to available resources taking into account the statistical properties of these requirements rather than the peak requirements used in circuit switching. ST does not itself specify the traffic control algorithms to be used. The development of such algorithms is an area of research that the protocol is intended to support. Some algorithms may need only rough statistical knowledge of user requirements and network behavior. Others may want more detailed knowledge and need to monitor the behavior of individual streams. The protocol is intended to be general enough to support both extremes. A successful traffic control algorithm would retain much of the statistical multiplexing advantages of datagram nets while at the

same time gaining much of the guaranteed data rate and controlled delay capabilities of circuit switched nets. A packet net using ST also has the ability of a circuit switched network to deny access to, or negotiate lower rates with, users whose demands would exceed capability.

The ST protocol requires users to state the data rate and delay requirements for a data stream before accepting any stream data. These requirements are used by ST agents (host processes and gateways) to determine whether or not resources are available in the catenet to support the offered stream load. The determination is based on knowledge of the stated requirements of other users, negotiation with networks such as SATNET which have built-in resource allocation mechanisms, and statistical load estimates of traffic on networks that lack such mechanisms. In order to accept the offered stream load, the cooperating agents must find a route through networks with sufficient uncommitted capacity to handle the new stream. In the process of routing the stream, intermediate agents retain information about the stream. The existence of this information allows the use of abbreviated headers on stream data packets and the efficient distribution of the multi-addressed packets required for conferencing.

The process used by ST agents in finding a route with sufficient capacity between source and destination is assumed to use a distributed routing algorithm to take advantage of the robustness and flexibility characteristic of distributed packet routing techniques. In the simplest case, the result would be a fixed-path internet route (a fixed set of intermediate agents (gateways)) for the stream packets. In the event of gateway or network failure, rerouting would be required. This can be undertaken automatically, but success is not guaranteed, since loss of the failed element or elements is likely to result in inadequate capacity to carry the original load. Fixed-path routing is not required by the protocol. If desired, dynamic alternate routing of stream packets can be used at the discretion of individual agents, but gateway implementation and the routing process will be more complex if that option is chosen. The protocol described in this document assumes fixed-path routing.

The goal toward which the cooperating ST agents in a catenet work is the maintenance of a controlled delay, guaranteed data rate environment in which packet speech communications can take place in a satisfactory fashion. Obviously, other non-cooperating users of the networks involved in the catenet can make it impossible to achieve that goal. Some independence of other users can be obtained in some networks such as SATNET by the use of dedicated resources. Gateways can be programmed to throttle non-cooperating internet traffic. To some extent, networks with poor delay characteristics can be avoided in the routing process. Priority service can be used in some nets to

allow small quantities (proportionately) of ST traffic to be handled satisfactorily in spite of the activities of other users. However, the success of ST or any other approach to handling packet speech will require either the cooperation of all network users or the involvement of the networks themselves in providing the required controlled delay, guaranteed data rate services.

3.0 RELATIONSHIP TO IP

ST is intended to operate as an extension of the presently defined internet protocol (IP). ST provides a different kind of service than the datagram service offered by IP. ST must operate on the same level as IP in order to access local net resources such as SATNET streams and to be able to take advantage of any available local net multi-address delivery capabilities to support conferencing applications. If an ST agent shares a local net port with an IP datagram handler, the two must cooperate in the use of the port to regulate traffic flow through the port.

In order to get the advantage of abbreviated headers on stream packets, ST uses a different header format than that used for IP datagrams. Packets with this format (see Section 5.0 for details) are called ST packets in this document. They pass from one ST agent to another, and the abbreviated header information changes on a hop-to-hop basis. However, ST packets cannot be transmitted until a route for the stream has been found and intermediate agents have built routing tables to translate the abbreviated headers. Since end-to-end negotiation between ST users is often desirable before stream routing takes place, for example to agree on vocoder type and data rate, and it is convenient for a user to interface to only one protocol handler, ST provides a second type of service. This service uses IP datagrams with an "ST" value in the IP Protocol Field. These packets are called "IP.ST" packets. They pass through datagram handlers in gateways and reach ST agents only at their destination hosts.

A third type of packet is allowed by the protocol. This type is realized by embedding an ST packet in an IP.ST packet. This method of sending an ST packet allows it to pass through gateways that do not support the ST protocol but do support IP datagrams. Of course, the packet efficiency and traffic control benefits of ST are lost in such a case, but the use of this artifice could be justified on the grounds that any communication is better than none.

4.0 CONCEPTS

The key concept in ST is that of a connection. Connections are supported by entities called agents which are made aware of the connection during a setup process that precedes use of the connection for data transfer.

4.1 Agents

There are four types of agents that may be involved in supporting ST connections. They are:

4.1.1 ST Hosts

The users of connections are processes that run in host computers and communicate over connections through other processes or software modules that adhere to the ST protocol. Hosts having these processes or modules are called "ST hosts" (or "hosts," when the context permits). ST hosts perform the functions of gateway halves in interacting with gateways for internet traffic. ST hosts share the management of local net ST resources with the other agents on the local net and are capable of routing connections to other agents as may be required. In networks with local multi-addressing capability, ST hosts make use of this capability in routing conference connections. In networks lacking such capability, ST hosts may need to replicate messages for conference connections unless a special agent called a "replicator" is available in the local net. In some local nets it may be desirable for hosts to forward traffic for conference connections. The protocol allows but does not require the latter capability.

4.1.2 ST Gateways

ST gateways perform routing and forwarding functions very similar to those performed by IP gateways. Unlike IP gateways, they store information about the connections they support and share the management of resources in the nets to which they are connected with the other agents in those nets. Like hosts, ST gateways may have to replicate packets for conference connections.

4.1.3 Replicators

In networks that lack multi-addressing or broadcast capability it may be desirable to provide special server hosts to handle the replication required for conferences. Replicators are needed in situations where the load caused by replication would produce congestion at a gateway port. Use of a replicator adds delay and is probably not warranted unless the number of copies needed in a particular net exceeds some threshold that depends

upon network port capacity. In worst-case situations a daisy-chain type of replication might be required because the peak rate could not be sustained at any network site. The existence of a replicator does not eliminate the need for replication in hosts and gateways. For example, a host in a conference with some participants on the same net but others on other nets may need to send packets to one or more gateways for speedy internet delivery as well as to a replicator for automatic distribution to other local net participants.

4.1.4 Access Controllers

The management of ST conference connections involves the services of an access controller. The functions of an access controller are to control conference participation and provide a central source for information about the data rate requirements of a conference connection. Ideally, access control services would be provided by a set of hosts distributed throughout the catenet that shared information about the connections being controlled. The addresses of these public access controllers would be known to all other agents, and a query to any one controller would provide information about any connection. In the absence of public access controllers, the protocol allows any host to serve as a private access controller. It is proposed to use a bit in the conference connection name to allow agents to determine whether a public or private access controller is responsible for a particular conference. The name identifies the "owner" of the conference. The owner is also the access controller in the private case.

4.2 Connections

Most applications for ST connections require full-duplex (bi-directional) communication between the parties in a point-to-point connection and omni-directional communication among the participants in a conference connection. In the design of the protocol two different approaches to realizing the desired capability have been considered. The first, called the simplex approach, uses a combination of simplex (one-way) connections. For example, in the simplex approach the caller requests a simplex connection to the called party, who, after accepting the connection, requests another simplex connection for the return path to the caller. In the second, called the full-duplex approach, the caller requests a full-duplex connection at the outset, and as soon as the called party has accepted the connection, data can flow in both directions.

For conference connections, the simplex approach requires each participant to request a simplex connection to all the others. The full-duplex approach requires that a participant request connection only to those that have not already requested

connection to him.

Both approaches can provide workable bases for the required capabilities. The pros and cons for both may be summarized as follows:

1. Simplex connections can take maximum advantage of available resources by using different routes for the forward and return paths. The routing of a full-duplex connection is more likely to fail since a path with the desired capacity in both directions must be found. This advantage for simplex connections is most pronounced in networks where load is assymetrical, a situation to be expected in nets carrying relatively heavy data loads.

2. Full-duplex connections can, except perhaps under conditions of heavy load, be set up more rapidly and with less control message traffic. The difference is most pronounced for conference connections. With full-duplex components of a conference connection, m-1 connection requests are required for an m-participant conference, since each new participant must connect to all those already in the conference. In the case of simplex components each new participant must also connect to all those already in the conference; but, in addition, those already in must connect to each newcomer. This activity adds sigma (m-1) connection requests (and responses) to the setup procedure.

3. Simplex connections have an advantage in situations in which two parties attempt to call each other at the same time. The two simplex connections can easily be combined into the required full-duplex connection. If the two parties start out with full-duplex connections, one of them must be refused or disconnected, a somewhat more complex task for the higher level protocol requesting the connection.

This document proposes a full-duplex basis for ST connections because the author believes that the advantage of relative simplicity and efficiency in setting up conference connections outweighs the advantages of the simplex basis. To allow connections with assymetrical flow requirements, the protocol allows users to specify different data rates in the two directions.

Even though traffic can flow in both directions on an ST connection, the connection has an orientation, and packets are said to move in either the "forward" or "backward" direction depending on whether they are moving away from or toward the

originator of the connection.

ST provides two types of connections: Point-to-Point (PTP) and Conference (CONF). PTP connections use different packet header formats and setup procedures to reduce overhead and allow faster setup for that more frequently used type.

4.2.1 Point-to-Point (PTP) Connections

PTP connections are set up in response to a CONNECT command from an originating process to an ST agent. The CONNECT specifies the following:

1. The NAME of the connection. The NAME is obtained by concatenating the ST address of the originating process (ORIGIN) with an arbitrary number. The ST address is the internet host address (ala IP) concatenated with an "extension" field (32 bits) to specify a process in the host (a telephone for NVP applications). It is the responsibility of the originating process to provide arbitrary numbers that keep the names of all outstanding connections unique.

2. The internet address of the process to which the connection is desired. This address is called the "TARGET." The terms "ORIGIN" and "TARGET" are used instead of "SOURCE" and "DESTINATION" because the latter terms will be used to refer to the senders and receivers of packets travelling on the connection. Thus the ORIGIN process can be both SOURCE and a DESTINATION for packets on the full-duplex connection.

3. A flow specification (FLOW-SPEC) that tells ST agents about the desired characteristics of the connection. In addition to information about the data rate requirements for both directions of the full-duplex connection, the FLOW-SPEC has a PRECEDENCE value that agents can use as a basis for the preemption of this or other connections as part of the traffic control strategy. The FLOW-SPEC is discussed in more detail in Section 4.5.

4. An arbitrary 16-bit number that the agent is to use to identify all ST packets that it will send to the originator on the connection (the backward direction). This identifier is called the "CID.B." If the connection request is accepted, the originator will be given a CID.F to be used to identify all packets it sends in the forward direction on the connection. These CID's allow abbreviated headers to be used on ST

packets and provide a means for agents to rapidly locate the stored forwarding table involved in handling a received packet. CID's are assigned by the agents receiving packets and need be only locally unique since they are reassigned on a hop-by-hop basis. The CID to be used on the next hop is stored in the agent's forwarding table.

During the setup procedure the CONNECT command propagates from agent to agent until it reaches the TARGET process. This propagation differs from ordinary packet forwarding in that the intermediate agents inspect the command, take appropriate action, and retain information about the requested connection. If the TARGET process agrees to the connection, it sends an ACCEPT command that is propagated back through the same intermediate agents that handled the CONNECT. The agents take appropriate action as they process the ACCEPT. If the TARGET process is not willing to accept the connection, it issues a REFUSE command which propagates back in the same fashion as the ACCEPT. REFUSE's are generated by intermediate agents if they find themselves unable to support a requested connection. An agent receiving such a REFUSE tries alternate routes and passes the REFUSE back another hop only when it has exhausted its routing alternatives. Appropriate REASON codes are included in the REFUSE commands.

After a connection has become established (an ACCEPT has reached the ORIGIN), changes to the FLOW-SPEC can be accomplished by the ORIGIN issuing a new CONNECT or the TARGET issuing a new ACCEPT command. (Actually, the TARGET can issue a new ACCEPT at any time after issuing the first ACCEPT, and it can also at that time begin sending packets on the connection although there is some hazard in doing so since they may pass the ACCEPT enroute and be discarded.) For the case where the FLOW-SPEC calls for a connection whose rate can be varied at the discretion of the catenet, intermediate agents issue CONNECT's and ACCEPT's to inform other agents and the end users about rate changes. These commands are marked to distinguish them from end user commands.

The ACCEPT command contains the same kinds of information as the CONNECT except that the backward connection identifier (CID.B) is replaced by a forward identifier (CID.F). In addition, the FLOW-SPEC will generally be different and will indicate the data rates and delay characteristics accepted by the agents. The CONNECT that arrives at the TARGET will be similarly modified from the CONNECT that was issued by the ORIGIN and will match the ACCEPT received by the ORIGIN. See Section 4.5 for a discussion of the changes that can occur to the FLOW-SPEC.

4.2.2 Conference (CONF) Connections

The type of connection required for voice conferencing is one in which any participant can send messages to all the others. Connections of this type have been called "omniplex" connections. ST realizes a conference connection by means of a superposition of tree-like components that start from an origin process (the root) and extend to a set of targets (the leaves). The set of participants in a conference is represented by a bit map. Each participant has a location in the conference bit map that is assigned by the access controller (AC). When a conference CONNECT command is given, a TARGET-BIT-MAP (TBM) is used to specify the set of targets to which connection is requested. The TBM is supplied by the AC when a participant joins a conference. The tree-like components all have the same NAME, and intermediate agents combine branches from the components whenever possible to minimize resources committed to the conference. Because of this combining, an ORIGIN-BIT-MAP (OBM) is needed to represent the set of originators that have requested connection to a particular participant.

The list of participating processes in a CONF connection is not carried in the CONNECT request but is is maintained by the AC and provided to agents and participants when needed. Another function of the AC is to provide the FLOW-SPEC for the connection to any agent on request. The reason for assigning these tasks to an access controller is to prevent unauthorized connection to a conference and to assure that all components of the connection use the same FLOW-SPEC.

The first step in establishing a conference is to install a list of participants and a FLOW-SPEC in an AC. The list of participants may be fixed at the outset or be allowed to grow during the course of the conference. A participant may depart from a conference, but his position in the list and the bit maps is not reused. The method by which the list of participants is made known to the AC is not of concern to ST itself and is not specified in this document. Higher level protocols such as a network voice protocol (NVP) engage in communications between participant processes and the AC in the process of setting up a conference. For example, an NVP issues a JOIN command to request access to a conference. If the NVP process is on the participant list or is otherwise acceptable, the AC responds with a WELCOME command that among other things tells the participating NVP its location in the CONF bit map. The NVP then sends TELL-ME messages to the AC to obtain the participant list and FLOW-SPEC for the CONF connection. This information is provided in INFO messages from the AC. Several of these messages may be required to transmit all the information about a large conference. The messages exchanged between participants and the AC are IP.ST datagrams. They cannot be ST packets because no ST connection

exists between the participants and the AC.

Once a participant has received a WELCOME message from the AC, it can issue a CONNECT.CONF command to its ST host agent. It uses a TARGET-BIT-MAP (TBM) that it received as part of the data in the WELCOME message. This TBM has bits set for all the previous joiners of the conference. The CONNECT.CONF will thus attempt to establish a full-duplex path to each of the previous joiners. These paths will make use of common links where possible and will result in a connection resembling a tree rooted at the site of the process originating the connection. When the CONNECT.CONF is issued by the originator it contains an ORIGIN-BIT-MAP (OBM) with a single bit set corresponding to the originating participant. If the CONNECT.CONF is successful (i.e., some subset of the targets are reached), an ACCEPT.CONF will be returned with bits set in the TBM indicating the participants to which connection has been achieved. In a CONF connection attempt, success may not be achieved with the entire set of targets specified by TBM. Some may be unreachable for any of a number of reasons. REFUSE.CONF messages will be returned for all such failures with bits in the TBM identifying the unreachable participants. If the failures in a particular attempt are due to more than one REASON, at least one REFUSE.CONF will be returned for each reason.

The technique for setting up conference connections proposed for ST results in each participant actively connecting to some subset of the others while accepting connections from the rest. The first participant does not issue a CONNECT and accepts all the others. The last connects to all the others and accepts none. Each participant can maintain up-to-date information about participation in the conference by utilizing the information in the CONNECT and ACCEPT messages it receives.

The CONNECT.CONF messages received by agents during the setup procedure do not contain information about the identity of the participants. In order to route the connection, the agents must acquire this information, and they do so by sending TELL-ME messages to the AC and getting INFO messages in response. They need to retain this information only during the routing phase of connection setup. Once the connection is established, bit map information in forwarding tables combined with a FORWARDING-BIT-MAP (FBM) in the ST packet is sufficient to handle the forwarding of packets on the connection. The FBM is used to specify the set of destinations for the packet. Thus a packet can be sent to all or any subset of the connection participants. The source of the packet is identified by a number representing the position of the source participant in the conference bit map.

In the case of a voice conference, no useful purpose is accomplished when many people speak at the same time. It is expected that a higher level protocol (part of NVP) would regulate the activity of the conference and would normally allow one or perhaps two persons to transmit speech at the same time. ST is not involved in this aspect of conference control except to the extent that if there are too many simultaneous talkers, the traffic-handling capability of the connection could be exceeded, and ST might discard some of the packets. The higher level control protocol should set the FLOW-SPEC for the connection to accommodate the expected traffic flow. Thus, for a simple one-at-a-time conference, ST would be asked for a data rate corresponding to a single speech stream.

The above discussion has described a connection arrangement suitable for supporting voice conferences in which any participant can transmit and be heard by all others. ST also provides another kind of multi-address message delivery capability. If only one participant issues a CONNECT.CONF command with a TBM specifying connection to all the others, a tree-like connection will be set up that allows the ORIGIN to send packets to all the others and receive from any of the others, but packets sent by the others will be received only by the ORIGIN.

4.2.3 Taking Connections Down

The process of taking a connection down is initiated either by an ORIGIN issuing a DISCONNECT message or a TARGET issuing a REFUSE. These messages propagate from agent to agent along the connection path so that intermediate agents can take appropriate action to clean up their stored information about the connection.

Connections can also be taken down as a result of intermediate agents detecting a faulty link or gateway or deciding to preempt the connection. In this case the agent or agents involved issue a DISCONNECT/ REFUSE pair that propagate in the appropriate directions. A REASON code in the messages informs the users as to the cause of the disconnection.

In the case of conference connections, bit maps allow selective disconnection and refusal.

4.3 Types of Service

ST offers two types of service for packets travelling on connections. Neither type has any delivery guarantees, i.e., there are no acknowledgements or retransmissions on either a hop-by-hop or an end-to-end basis. Neither type guarantees packet integrity; i.e., if local nets offer a type of service

that can deliver packets with bits in error, ST may use that type of service. The headers of ST packets are sum-checked by ST agents, but the data portions are not.

The two types of service differ in whether or not they use the channel capacity nominally allocated to the connection and also in the strategy used by intermediate agents in buffering them. The two types are:

1. Stream Packets (called ST.ST packets). These packets use the allocated resources and are buffered for a short time only, since they are intended for applications such as speech communication where a late packet is not worth delivering. They are discarded by intermediate agents if queue conditions indicate that they cannot be delivered in a timely fashion.

2. Datagrams (called ST.DG packets). These packets have the same form as ST.ST packets except for a flag bit in the header and travel over the same connection path. They use allocated resources only when spare capacity exists, e.g., when the ST.ST flow drops below the allocated value. Otherwise they share local net resources with other IP datagram traffic. They are buffered with a queueing strategy appropriate for datagram traffic and are discarded only when agent buffer resources approach exhaustion. They are intended for use by higher level protocols such as NVP in applications such as dynamic control of the "floor" in a conference. They are also used by ST itself for connection management.

4.4 Packet Aggregation

ST allows any ST packets, stream or datagram, to be aggregated together that have the same next-agent local-net destination. "Aggregation" is a form of multiplexing, but is given a different name to distinguish it from the multiplexing done in the IP Multiplexing protocol that allows multiplexing only for packets with the same end-to-end source and destination. The term "envelope" is used to refer to any ST message sent from one agent to another. An envelope may contain one or more ST packets and is limited in size by the maximum size of packet that the local net can carry. The envelope has a short header in addition to the header of the individual aggregated packets. See Section 5.0 for a description of header formats.

The ST aggregation technique requires agents to look inside of received envelopes and handle the packets as individual entities. This procedure adds to the computing load of gateways, but can achieve significant communication savings in networks

with high per-packet overhead such as SATNET, particularly when many short packets must be handled.

4.5 Flow Specifications

The FLOW-SPEC that is carried by CONNECT and ACCEPT messages contains several fields. Some are specified by the originator of the CONNECT. Others are produced either during the process of setting up the connection or changing its allowed flow characteristics. Some apply in common to both directions of the full-duplex connection. Others apply individually to allow different flows in the two directions and appear in pairs in the control messages.

Data rate, the basic quantity used in traffic control computations, is specified by means of three parameters; a stream interval (SI), a packet length (PL), and a duty factor (DF). The average expected data rate can be computed by taking the product of PL, DF, and the reciprocol of SI. The FLOW SPEC allows for one value each for SI and DF for each direction. However, as many as four values of PL can be provided as options, allowing the ST agents flexibility in allocating resources for some types of traffic flow.

The flow type (TYPE) parameter is intended to allow ST to take into account a variety of different user load characteristics. The set of possible types can be expected to grow with experience, but a relatively few types seem to be adequate to deal with presently contemplated voice encoding techniques. These are:

1. Fixed Rate. The data rate is held fixed for the life of the connection. A simple speech encoder that can run at only one rate would use this type value with all four PL's set to the same value. A somewhat more complex encoder that could run at more than one rate but could not change rates on the fly would use the fixed-rate type but could offer a choice of up to four values for PL. A variable-rate vocoder such as the LPC2 vocoder used in the ARPANET that has a rate that varies depending on the short time behavior of the speech signal would also use the fixed-rate type but would set the duty factor to a lower value than the 0.5 or so used by a simple encoder.

2. Multiple Rate. The data rate allowed can be of any of the four specified by the four PL's and the agents are free to change rates at any time to accommodate to network load changes. Whenever an agent changes the rate, it sends appropriate CONNECT and ACCEPT messages to tell other agents and the users about the change.

Since such rate changes require extra communication and processing in the catenet, agents would have to avoid frequent changes. This flow type would be used by enoders that run at a variety of rates and can switch rates rapidly but need to do so explicitly either because packet formats must change with rate changes or because some parameter such as sampling rate must be changed at sender and receiver. This flow type could also be useful for sending data rather than voice over ST connections.

3. Prioritized Variable Rate. This flow type is intended for use by certain advanced encoders of a kind called "embedded" where subsets of the coded bit stream can be stripped en route without loss of intelligibility. There is, of course, some loss of quality and/or ability to withstand acoustical background noise when stripping occurs. For this flow type each of the four PL's corresponds to one of the four packet priorities that can be attached to ST.ST packets. The encoder would place the bits needed for its lowest rate in the highest priority packet, the next lowest in the second highest, etc. When pressed for channel capacity, agents would be free to discard the lower priority packets for this flow type. The overall precedence of the connection would also affect the probability of packet discard. It is not anticipated that agents would send explicit messages to announce that discarding was taking place.

Another set of parameters in the FLOW-SPEC is concerned with transmission delay. ST does not allow the user to specify a delay requirement, but it does allow some control over the tradeoff between delay and data rate options during the routing process. A ROUTING-STRATEGY parameter is provided for this purpose. Currently, two strategy options for PTP connections are envisioned, but others could be added if desired. One gives preference to minimizing delay at the expense of data rate. The other gives preference to data rate over delay. The ROUTING-STRATEGY options are meaningful only when data rate options are available. Otherwise data rate is as absolute requirement in routing.

While a user cannot specify a delay requirement to ST, ST does provide the user with an estimate of both minimum delay and delay dispersion in fields of the FLOW-SPEC. The estimates are based on a priori statistics relating delays to average network loads. When an agent propagates a CONNECT packet, it adds values from tables indexed on the current load estimate to the MIN-DELAY and DISPERSION fields of the FLOW-SPEC for the forward direction. It performs the same function for the backward direction as it

propagates the ACCEPT. The MIN-DELAY is the simple sum of the hop-to-hop contributions, but the DISPERSION is a sum of squares. The receiver can compute an estimate of overall delay by adding the MIN-DELAY to the square root of the DISPERSION. The DISPERSION estimate by itself can be useful in setting the reconstitution delay value needed to play out satisfactory speech for listeners. The proper value can vary over a wide range depending on the path through a catenet of networks with very different delay characteristics.

Another parameter set by agents during the routing process is the ACCEPTED-RATE field. This field informs the users as to which of the four possible data rate options (PL's) have been accepted for each of the two directions of the connection. Of course, if none were acceptable, a REFUSE would be returned with a REASON code indicating unavailability of resources at the requested precedence level. Another flow-related reason for refusal could be an inability of the networks to handle a too-short stream interval.

All FLOW-SPEC parameters except PRECEDENCE and ROUTING-STRATEGY can be independently specified or are reported separately for each of the two directions of the full-duplex connection. The exceptions are required to apply to the entire connection to simplify the task of gateways in handling connections.

The ROUTING-STRATEGY field has other control functions in addition to weighting the tradeoff between data rate and delay. For CONF connections it indicates whether or not data rate options must match in both directions (a requirement for voice conferencing) or can be negotiated independently. If ST agents support split routing, (a capability to divide the traffic on a connection among two or more paths) the ROUTING-STRATEGY field will indicate whether or not this technique is to be applied to the connection. Split routing also requires additional fields to indicate the fraction of the nominal traffic that has been accepted or is requested to be handled. This document does not propose the implementations of split routing in the first version of ST.

5.0 PACKET FORMATS

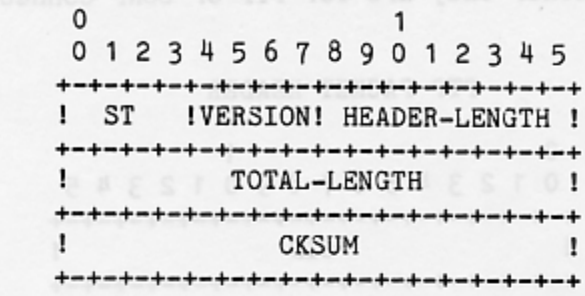
The messages sent between ST agents on connections are envelopes containing one or more ST packets. The envelope consists of an envelope header (EH) followed by one or more packet headers (PH's) followed by the data portions of the packets in the same order. The envelope thus has the form:

EH, PH1, PH2, . . .PHn, DATA1, DATA2, . . . DATAn

The reason for aggregating the headers separately from the data is that doing so allows the header region to be checksummed easily as a unit before attempting to parse the envelope. It is expected that ST will be used in networks that can deliver messages with bits in error and that some non-negligible fraction of the messages will have such errors. To require the entire envelope to be error-free in order to use any of it would result in an excessive rate of lost packets.

Since ST operates as an extension of IP, the envelope arrives at the same network port that IP uses to receive IP datagrams. It is proposed to use a unique code in the first field of the message to identify it as an ST envelope. The first four bits of an IP datagram are defined to be the Version Number field. It is therefore proposed to use one of the 16 possible IP versions to distinguish ST envelopes from IP datagrams. With this convention an envelope header will have the following format:

ENVELOPE HEADER



ST is the particular IP Version Number assigned to identify ST envelopes.

VERSION is the ST version number. This document is a proposal for VERSION 1.

HEADER-LENGTH* is the length in words of the envelope header (3) plus the sum of the header lengths of the aggregated packets.

TOTAL-LENGTH is the length of the entire envelope. It does not include any local net headers or trailers.

CKSUM covers the envelope header and all packet headers.

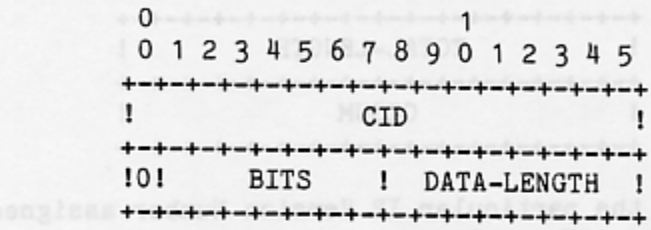
++++++++++++++

*All ST communications use the 16-bit word as a basic unit. All lengths are in word units.

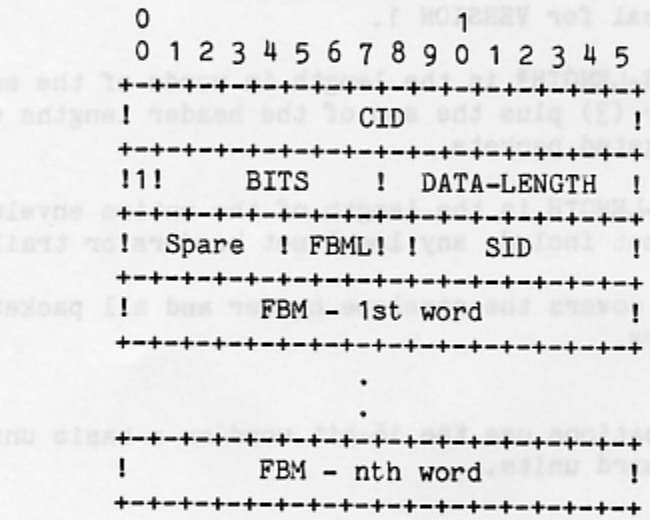
++++++++++++++

The individual packet headers have one of two formats depending on whether they are for PTP or CONF connections. These formats are:

PTP PACKET HEADER



CONF PACKET HEADER



CID is an arbitrary identifier assigned by the agent receiving the packet for the purpose of identifying the connection on which the packet is travelling. Since the CID is unique only to the agent that assigned it, it will generally have a different value on each hop of the connection path.

BITS are defined as follows:

Bit 1 distinguishes stream packets (ST.ST) from datagrams (ST.DG) (1 = DG).

Bits 2 and 3 define the packet priority (00 = highest priority).

Bits 4 and 5 are spares.

Bits 6 and 7 are unused (may be used by higher level protocols if desired).

DATA-LENGTH is the length of the data field in words.

FBML (3 bits) is one less than the length of the Forwarding Bit Map (FBM) in words.

SID (7 bits) identifies the source of the packet on a CONF connection (the source is implicit for a PTP connection packet). The value of SID corresponds to the bit position of the source in the conference bit map. Bit numbers start with zero, and positions start with the left-most (most significant) bit of the first word of the bit map.

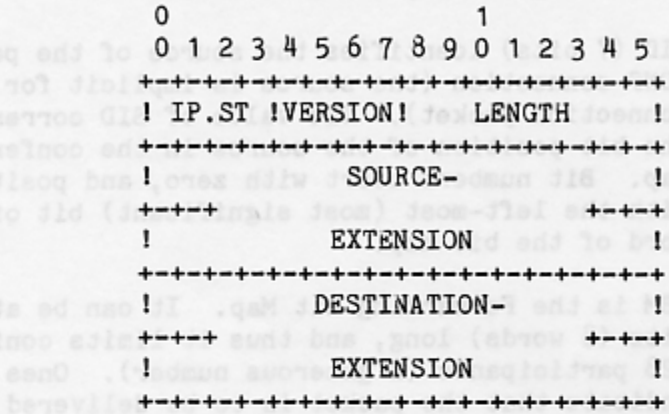
FBM is the Forwarding Bit Map. It can be at most 128 bits (8 words) long, and thus it limits conferences to 128 participants (a generous number). Ones in the FBM indicate that the packet is to be delivered to the corresponding participants. The FBM is allowed to increase in one word increments to allow new participants to enter during the course of a conference, but it does not shrink when participants leave, and bit positions are not reused.

As pointed out in Section 3.0, ST supports a second type of communication called IP.ST datagrams. These are ordinary IP datagrams with an "ST" value in the protocol field. They are used to allow higher level protocols to communicate prior to the setting up of an ST connection, and they are also used for communication between access controllers and other ST agents during the setup of CONF connections. They are strictly point-to-point communications since they are IP datagrams. According to the conventions for IP datagrams, these messages would have the form:

IP Header, IP.ST Header, Data

The IP.ST Header has the following form:

IP.ST PACKET HEADER



IP.ST is a value chosen to be different from the "ST" value used in the first four bits of the ST envelope. This field allows IP.ST datagrams to be distinguished from ST envelopes embedded in IP.ST packets, a technique that can be used to get ST envelopes through gateways that do not support ST.

VERSION is the ST version number.

LENGTH is the total length of the IP.ST packet excluding IP and local net headers, etc.

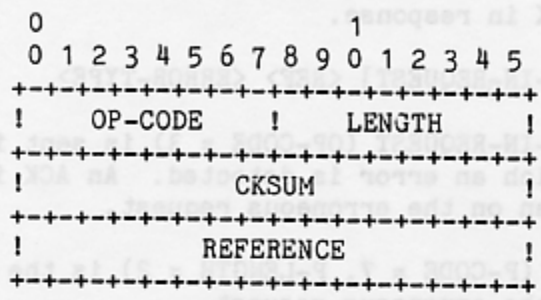
SOURCE- and DESTINATION-EXTENSION's are 32-bit fields used to identify the source and destination processes. Like ARPANET NCP process identifiers, they are not specified by the protocol. The source and destination host addresses are carried in the IP header.

6.0 CONTROL MESSAGES

With the exception of communications with access controllers, ST control messages are sent from agent to agent as ST.DG packets with the CID set to zero. This convention is similar to the ARPANET NCP use of Link 0 for control. Communication with AC's uses IP.ST packets. The form is otherwise the same. The control protocol follows a request-response model with all requests expecting responses and all responses expecting acknowledgements. Retransmission after timeout is used to allow for lost or ignored messages. A packet may contain more than one control message. Control messages do not extend across packet boundaries.

Control message headers have the following format:

CONTROL MESSAGE FORMAT



OP-CODE specifies the request or response.

LENGTH is the length of the control message in words.

CKSUM is the checksum of the control message. Because the control messages travel in envelopes that may be delivered with bits in error, each control message must be checked before it is acted upon.

REFERENCE is an arbitrary reference number used to associate requests with responses and acknowledgements.

The header is followed by parameters as required for the particular OP-CODE. Each parameter is identified with a P-CODE byte that is followed by a P-LENGTH byte indicating the length of the parameter (including the P-CODE, P-LENGTH word) in words. Parameters can be sent in any order. The format of individual parameters is specified in the following sections in connection with the OP-CODE's with which they are used.

Control messages fall into two categories according to whether they deal with PTP or CONF connections. There are four messages that are independent of connection type. These are:

6.0.1 [ACK]

ACK (OP-CODE = 1) has no parameters. The REFERENCE in the header is the REFERENCE number of the message being acknowledged. ACK's are used to acknowledge responses to requests and in some cases constitute responses or partial responses themselves.

6.0.2 [HELLO]

HELLO (OP-CODE = 2) is used to determine whether or not another agent is alive and well. It has no parameters and expects an ACK in response.

6.0.3 [ERROR-IN-REQUEST] <REF> <ERROR-TYPE>

ERROR-IN-REQUEST (OP-CODE = 3) is sent in response to a request in which an error is detected. An ACK is expected. No action is taken on the erroneous request.

REF (P-CODE = 7, P-LENGTH = 2) is the REFERENCE number of the erroneous request.

ERROR-TYPE is not yet specified.

6.0.4 [ERROR-IN-RESPONSE] <REF> <ERROR-TYPE>

This message (OP-CODE = 4) is sent in lieu of an ACK for a response in which an error is detected. No ACK is expected. Action taken by the requester and responder will vary with the nature of the request.

REF identifies the erroneous response.

ERROR-TYPE is not yet specified.

6.1 Control Messages for PTP Connections

PTP connections are set up and taken down with the following messages:

6.1.1 [CONNECT.PTP] <NAME> <TARGET> <FLOW-SPEC> <CID.B>

CONNECT.PTP (OP-CODE = 5) requests the set up (routing) of a PTP connection or asks for a change in the flow specification of a connection already routed. Its parameters are:

NAME (P-CODE = 1, P-LENGTH = 6) is the ST address of the process that originated the CONNECT.PTP (the ORIGIN) concatenated with a 16-bit number chosen to make the name unique. An ST address is a 32-bit IP host address concatenated with a 32-bit EXTENSION identifier chosen to identify a particular process in the host. The EXTENSION is provided by some higher-level protocol and is assumed by ST to be unique to the host. For NVP use the EXTENSION identifies a particular telephone and is presumably a well-known

quantity.

TARGET (P-CODE = 2, P-LENGTH = 5) is the ST address of the target process.

FLOW-SPEC (P-CODE = 3, P-LENGTH = 18) is a complex parameter that both specifies and reports on the flow requirements and expected delay characteristics of the full-duplex connection. See Section 4.5 for further information.

CID.B (P-CODE = 4, P-LENGTH = 2) is the connection identifier to be used on packets moving in the backward direction on the connection.

CONNECT.PTP expects a response. There are four response possibilities: ACCEPT.PTP, REFUSE.PTP, ACK, and ERROR-IN-REQUEST. Receipt of an ACK means that the agent receiving the request is working on it, and the requester should wait for a future ACCEPT or REFUSE. ERROR-IN-REQUEST will be returned only when a format error is detected in the CONNECT.PTP. Other errors, if detected, will elicit REFUSE messages.

The processing of CONNECT messages requires care to avoid routing loops that could result from delays in propagating routing information among gateways. The example in Section 7.0 describes in some detail the actions of agents in handling CONNECT requests while routing a connection.

6.1.2 [ACCEPT.PTP] <NAME> <TARGET> <FLOW-SPEC> <CID.F)

ACCEPT.PTP (OP-CODE = 6) is returned to indicate that the requirements of a CONNECT.PTP have been met or that a change in flow specifications has occured. Parameters are the same as for CONNECT.PTP except that a CID.F (P-CODE = 5, P-LENGTH = 2) is returned for use on packets travelling in the forward direction. The FLOW-SPEC will be modified to show the accepted rate and accumulated delay information (See Section 4.5).

ACCEPT messages expect ACK's or ERROR-IN-RESPONSE's. ERROR-IN-RESPONSE will be returned if an ACCEPT is sent to an agent that has no knowledge of the connection. This may occur if an ACCEPT is generated at the same time that a DISCONNECT is being propagated.

6.1.3 [REFUSE.PTP] <NAME> <REASON>

REFUSE.PTP (OP-CODE = 7) is returned to indicate that agents have failed to set up a requested connection or that a previously established connection has been lost. REFUSE's are also returned to indicate routing failure, and in such a case may not end up

propagating back to the origin. TARGET's also issue REFUSE's to take down connections intentionally.

REASON (P-CODE = 6, P-LENGTH = 2) indicates the reason for connection refusal. REASON codes apply also to DISCONNECT messages and include the following:

CODE	EXPLANATION
0	No explanation
1	Target refuses connection
2	Target does not respond
3	Target cannot be reached
4	Connection preempted
5	STREAM INTERVAL too short
6	Requested data rate cannot be handled
7	Connection broken due to network fault
8	Connection broken by ORIGIN
9	Conflicting FLOW-SPECs in CONF connections

REFUSE's are ACK'ed and are propagated by intermediate agents if meaningful (i.e., the agents had tables for the connection). The backward propagation of a refuse may be halted at an intermediate agent if an alternate route exists that has not been tried, and the REASON indicates that it is reasonable to try the alternate route. (I.e., it does not indicate that the target refuses or does not respond).

6.1.4 [DISCONNECT.PTP] <NAME> <REASON>

DISCONNECT.PTP (OP-CODE = 8) is sent to request that a previously requested connection be taken down. It can be generated either by the originator of the CONNECT or by an intermediate agent that executes a preemption or detects a fault.

REASON uses the same codes as REFUSE although not all codes apply.

DISCONNECT expects an ACK and is propagated in the forward direction so long as agents are encountered that know about the connection.

A connection can be taken down either by a REFUSE or a DISCONNECT (or both) depending upon which end first decides to initiate the process. If both start within a propagation time of each other, neither message will reach the opposite end.

6.2 Control Messages for CONF Connections

CONF connections are set up and taken down with CONNECT, ACCEPT, REFUSE, and DISCONNECT messages, but the CONF versions of these messages have somewhat different parameters. In addition, CONF connection setup requires that agents communicate with access controllers by means of TELL-ME and INFO messages. These latter messages are sent as IP.ST datagrams. The former are sent as ST.DG packets with CID = 0.

6.2.1 [CONNECT.CONF] <NAME> <OBM> <TBM> <CID.B>

CONNECT.CONF (OP-CODE = 9) requests the setup (routing) of a CONF connection or asks for a change in flow specifications of a connection already routed. The parameters NAME and CID.B have the same form and interpretation as they do for CONNECT.PTP except that NAME is the name of the owner of the conference, not the originator of the CONNECT message. The new parameters OBM and TBM allow the message to deal with multiple ORIGIN and TARGET processes. The FLOW-SPEC for the connection is obtained from the access controller.

OBM (P-CODE = 8, P-LENGTH = 2-9) is the ORIGIN-BIT-MAP. Bits set in the map identify originating processes. When a CONNECT.CONF is first issued by a user process only one bit is set in OBM identifying the issuer. However, as the message propagates, intermediate agents may find that they have other CONNECT.CONF messages for the same connection on hand at the same time. In that case, they can merge the requests so that more bits become set as the message approaches its targets.

TBM (P-CODE = 9, P-LENGTH = 2-9) is the TARGET-BIT-MAP. Bits set in the map identify the target processes. In general, the user process will have set many bits in TBM when it first issues a CONNECT.CONF. As the message propagates it will split many times, each split reducing the number of bits left set in TBM. When the CONNECT.CONF's reach their targets only one bit will be left set in each.

Since the CONNECT.CONF message does not tell its receiver anything about the actual identities of the target processes, intermediate agents must get this information, as well as the FLOW-SPEC, from the access controller by sending TELL-ME messages and receiving INFO messages in response. The agents use the NAME to locate the AC, using a bit in the name to distinguish between a public or private AC. The NAME is the ST address of a process concatenated with a 16-bit number to make the NAME unique. It is proposed that the most significant bit of that 16-bit number be used to distinguish public from private ACs. A zero in that bit would indicate a private AC and in that case, agents would send

TELL-ME messages to the process address in the NAME. In the public case, the agent would communicate with an AC whose address was known a priori to the agent.

6.2.2 [ACCEPT.CONF] <NAME> <OBM> <TBM> <FLOW-SPEC> <CID.F>

ACCEPT.CONF (OP-CODE = 10) is similar in function to ACCEPT.PTP. NAME, FLOW-SPEC, and CID.F have the same form and interpretation. OBM specifies the set of originators to which the ACCEPT is to be propagated. TBM specifies the set of targets that have accepted the connection. This set may be a sub-set of the targets requested in the CONNECT to which an ACCEPT responds. The FLOW-SPEC is included in the ACCEPT because it reflects the actual resources granted to the connection.

6.2.3 [REFUSE.CONF] <NAME> <OBM> <TBM> <REASON>

REFUSE.CONF (OP-CODE = 11) is similar in function to REFUSE.PTP. As for ACCEPT.CONF, OBM specifies the set of originators to which the REFUSE is to be propagated. TBM specifies the set of targets that cannot be reached, have refused, etc. A single REASON applies to all the targets in the TBM. If more than one REASON applies to a set of targets, as many REFUSE's as REASON's will be sent.

6.2.4 [DISCONNECT.CONF] <NAME> <OBM> <TBM> <REASON>

DISCONNECT.CONF (OP-CODE = 12) is similar in function to DISCONNECT.PTP. As for REFUSE.CONF, OBM and TBM specify the sets of originators and targets to which the DISCONNECT applies.

6.2.5 [TELL-ME] <NAME> <PART-NUM> <FLOW-SPEC-REQ>

TELL-ME (OP-CODE = 13) is sent from an agent or a participant process to an access controller . The AC is expected to return an INFO message with the requested information. Either of the latter two parameters may be omitted.

PART-NUM (P-CODE = 10, P-LENGTH = 2) specifies the number of the first participant about which information is requested. The response will be a participant list starting with the specified participant and continuing until the maximum packet size is reached or the list is exhausted.

FLOW-SPEC-REQ (P-CODE = 11, P-LENGTH = 2) requests the AC to send the FLOW-SPEC for the connection.

6.2.6 [INFO] <NAME> <STATUS> <PART-LIST> <FLOW-SPEC>

INFO (OP-CODE = 14) is sent from an AC to an agent or participant process in response to a TELL-ME. It provides the requested information. STATUS is always present. PART-LIST and FLOW-SPEC are present only when requested by the TELL-ME.

STATUS (P-CODE = 12, P-LENGTH = 2) carries 2 bytes of information. Byte 1 is the CONF-TYPE. Byte 2 gives the length of the participant list. The following values for CONF-TYPE are defined:

Type	Meaning
0	No conference defined with this NAME
1	Conference with closed participant list
2	Conference with open list and password
3	Conference with completely open list (no password needed).

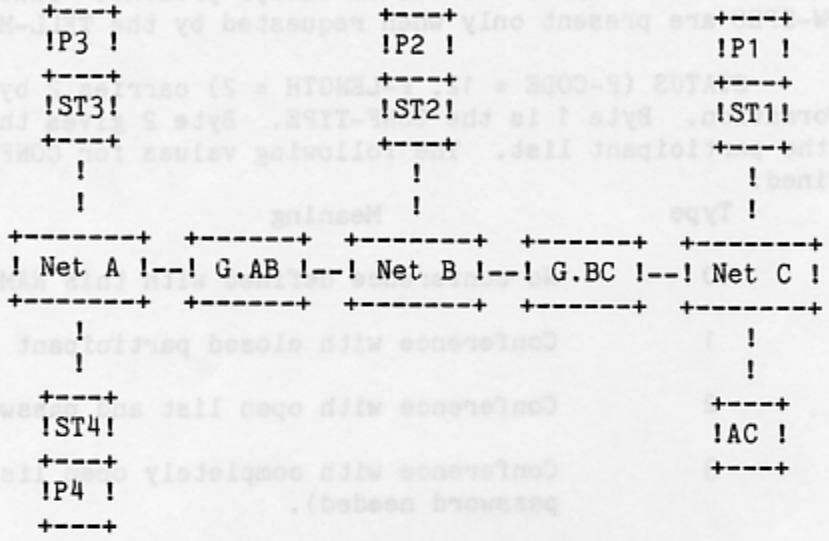
PART-LIST (P-CODE = 13, P-LENGTH = (4m + 2)) provides a section of the participant list starting at the location (PART-NUM) requested in the TELL-ME and continuing until either the end of the list or packet capacity is reached. The items in the PART-LIST are the ST addresses (64 bits) of the participating processes. The addresses are present whether or not the participants are active. The addresses are preceded by a word giving the number of the first participant on the list.

FLOW-SPEC is the nominal FLOW-SPEC for the conference.

7.0 AN EXAMPLE OF CONF CONNECTION SETUP

This section is a rather detailed example of the actions called for by ST in setting up a connection for a conference with four participants. In addition to showing the control message flow, it also indicates the information used and retained by gateways in supporting the connection. For the sake of simplicity, it is assumed that the flow requirements are always met. The ".CONF" suffix is omitted from OP-CODE's, and parameters such as NAME and FLOW-SPEC that are always the same are also omitted. In addition, ACK's are not shown but are assumed to occur where required.

The example uses the following network configuration:



Each participant (Pi) communicates through a host agent called "STi." The communications between the P's and their local ST's are written out as control messages to show the logical flow even though in actual implementations they might be handled very differently.

The actions involving ST start after the participants have joined the conference by communicating with the access controller (AC) and have received TARGET-BIT-MAPs (TBMs) telling each Pi to which other Pi's connections are to be set up. The notation "{ A, B, C }" is used to indicate a bit map with bits set for A, B, and C. The participants are assumed to have joined in the order of their numbers. Thus P1 got an empty TBM ({ }), and P4 got TBM = { P1, P2, P3 }. According to the rules, P1 issues no CONNECT messages, but waits for the others to connect to it. The action thus begins with P2 sending:

P2->ST2: [CONNECT] <OBM = { P2 }> <TBM = { P1 }> <CID.B = 3>

ST2->AC: [TELL-ME] <PART-NUM = 1> <FLOW-SPEC-REQ>

AC->ST2: [INFO] <PART-LIST = ADDR.P1, ADDR.P2, ADDR.P3, ADDR.P4>
<FLOW-SPEC> <STATUS>

These last two commands are executed independently by all agents when they first receive a CONNECT. They will be replaced by the phrase "X gets info" in the following.

ST2 observes that ADDR.P1 is not in its local net and lacking routing knowledge decides to try G.AB (the wrong direction).

ST2->G.AB: [CONNECT] <OBM = { P2 }> <TBM = { P1 }> <CID.B = 17>

G.AB gets info and decides that net C is unreachable except through net B from whence the CONNECT came.

G.AB->ST2: [REFUSE] <OBM = { P2 }> <TBM = { P1 }>
<REASON = 3 (Target cannot be reached)>

ST2 decides to try another gateway.

ST2->G.BC: [CONNECT] <OBM = { P2 }> <TBM = { P1 }> <CID.B = 17>

G.BC gets info, builds a connection entry, and sends:

G.BC->ST1: [CONNECT] <OBM = { P2 }> <TBM = { P1 }> <CID.B = 1001>

ST1 gets info and sends:

ST1->P1: [CONNECT] <OBM = { P2 }> <TBM = { P1 }> <CID.B = 1>

Since P1 has already joined the conference and recognizes P2 as another participant, it sends:

P1->ST1: [ACCEPT] <OBM = { P2 }> <TBM = { P1 }> <CID.F = 1>

ST1->G.BC: [ACCEPT] <OBM = { P2 }> <TBM = { P1 }> <CID.F = 32>

At this point G.BC would have the following stored information (neglecting bookkeeping items such as pointers).

1. A connection block with NAME, FLOW-SPEC, and CID.IN = 1001 (the same CID can be used for all inputs for the connection). This information is retained for the life of the connection. The PART-LIST used in processing may be discarded once an ACCEPT (or REFUSE) has been received and the forwarding tables have been created. However, since there are likely to be other CONNECT's to be processed, it would be efficient to keep the PART-LIST for a time (say several minutes).

2. Two forwarding tables, one for each packet that might be sent in response to an input.

ITEMS	#1	#2
NET-PORT	B	C
ADDRESS	ST2	ST1
MASK.OBM	{ P2 }	{ }
MASK.TBM	{ }	{ P1 }
CID.OUT	17	32

The principal function of the masks is to facilitate packet forwarding. When a packet arrives, the following computation is made for each forwarding table to compute the output FORWARDING-BIT-MAP (FBM):

FBM.OUT = FBM.IN & (MASK.OBM U MASK.TBM)

If FBM.OUT has no bits set, it is not necessary to send a packet to the address in the table. Otherwise a packet is sent using the NET-PORT, ADDRESS, and CID.OUT from the table.

Having built its tables, G.BC sends:

G.BC->ST2: [ACCEPT] <OBM = { P2 }> <TBM = { P1 }> <CID.F = 1001>

ST2->P2: [ACCEPT] <OBM = { P2 }> <TBM = { P1 }> <CID.F = 10>

At this point P2 and P1 are connected and could begin talking, if permitted by the higher level protocol.

In connecting P3 and P4 we will assume that both initiate requests at essentially the same time so that they propagate concurrently.

P3->ST3: [CONNECT] <OBM = { P3 }> <TBM = { P1, P2 }> <CID.B = 5>

P4->ST4: [CONNECT] <OBM = { P4 }> <TBM = { P1, P2, P3 }>
<CID.B = 1>

ST3 and ST4 get info. ST3 notices that P1, P2 both are outside the local net, but ST4 notices as well that P3 is on the same net as P4.

They send:

```
ST3->G.AB: [CONNECT] <OBM = { P3 }> <TBM = { P1, P2 }>
           <CID.B = 135>

ST4->ST3: [CONNECT] <OBM = { P4 }> <TBM = { P3 }> <CID.B = 27>

ST4->G.AB: [CONNECT] <OBM = { P4 }> <TBM = { P1, P2 }>
           <CID.B = 27>
```

ST3 forwards the CONNECT to P3, which accepts, and ST3 responds to ST4 with:

```
ST3->ST4: [ACCEPT] <OBM = { P4 }> <TBM = { P3 }> <CID.F = 135>
```

Meanwhile G.AB gets info and notices that it has two CONNECT's for the same NAME. It decides to merge them and sends:

```
G.AB->ST2: [CONNECT] <OBM = { P3, P4 }> <TBM = { P2 }>
           <CID.B = 2356>
```

and

```
G.AB->G.BC: [CONNECT] <OBM = { P3, P4 }> <TBM = { P1 }>
            <CID.B = 2356>
```

ST2 forwards the CONNECT to P2, which accepts, and ST2 sends:

```
ST2->G.AB: [ACCEPT] <OBM = { P3, P4 }> <TBM = { P2 }>
           <CID.F = 17>
```

Now G.AB will not continue to propagate the ACCEPT because the CONNECT on which it is working asked for connection to P1 as well as P2. It will wait for an ACCEPT or REFUSE from P1.

G.BC already knows about the connection to P1, but it does not assume that P1 will accept P3 and P4, so it propagates the CONNECT.

```
G.BC->ST1: [CONNECT] <OBM = { P3, P4 }> <TBM = { P1 }>
           <CID.B = 1001>
```

ST1 forwards to P1, which accepts, and ST1 responds:

```
ST1->G.BC: [ACCEPT] <OBM = { P3, P4 }> <TBM = { P1 }> <CID.F =
32>
```

In the latter exchange G.BC and ST1 used the same CID's they had used before for this connection. If either had chosen to use a different CID, the newer value would supercede the earlier one in the forwarding table.

It should be noted that the protocol could allow G.BC to accept the connection from P3 and P4 without forwarding the CONNECT to ST1 because G.BC already knows it has a connection to P1. This shortcut is not taken because it denies P1 the information about the connection requests from P3 and P4 and the opportunity to refuse those connections if desired.

To finish the setup we have:

```
G.BC->G.AB: [ACCEPT] <OBM = { P3, P4 }> <TBM = { P1 }>
            <CID.F = 1001>
```

G.AB will now accept for P1 and P2.

```
G.AB->ST3: [ACCEPT] <OBM = { P3 }> <TBM = { P1, P2 }>
           <CID.F = 2356>

G.AB->ST4: [ACCEPT] <OBM = { P4 }> <TBM = { P1, P2 }>
           <CID.F = 2356>
```

When ST3 and ST4 propagate the ACCEPT's to P3 and P4 the conference connection is complete.

At this point the forwarding tables in G.BC are the following:

ITEM	#1	#2	#3
NET-PORT	B	C	B
ADDRESS	ST2	ST1	G.AB
MASK.OBM	{ P2 }	{ }	{ P3, P4 }
MASK.TBM	{ }	{ P1 }	{ }
CID.OUT	17	32	2356

If at some later time G.BC should decide to preempt the connection, it would issue one message for each forwarding table entry:

```
G.BC->ST2: [REFUSE] <OBM = { P2 }> <TBM = { P1 }>
           <REASON = 4 (Connection preempted)>

G.BC->ST1: [DISCONNECT] <OBM = { P2, P3, P4 }> <TBM = { P1 }>
           <REASON = 4 (Connection preempted)>
```

G.BC->G.AB: [REFUSE] <OBM = { P3, P4 }> <TBM = { P1 }>
<REASON = 4 (Connection preempted)>

Having issued these messages and received ACKs in response (or timed out in the absence of an ACK), the gateway can delete the table entries and reclaim the CID for future use. The REFUSE sent to G.AB would, of course, be propogated to ST3 and ST4.

8.0 AREAS NEEDING FURTHER WORK

This document does not completely specify the protocol. Further work is needed to specify error conditions and their handling. The FLOW-SPEC parameter is not yet laid out in detail. Rerouting has not been thought through sufficiently. The whole area of routing strategies and the information to be exchanged among gateways has not been given much consideration. There is also a need for agents to exchange information (not yet specified) about local net resources. For example, if agents are to make use of local net multi-addressing capability, the selection of a CID for a connection is no longer at the discretion of an individual agent. A convention is needed to avoid conflicting use of CID's as well as requesting duplicate resources to serve a CONF connection. The CONNECT control message needs to be extended to allow agents to indicate local net resources that are already committed to a CONF connection.