IEN # 4
Supercedes: None
Replaces: None

Barryl Rubin
SRI
16 May 1977

Section 2.5.4.2

NSC Note No. 107

Specification for a General Network Conferencing System
Voice Communication Supervisor
(Preliminary Documentation)

Barryl E. Rubin

Telecommunications Sciences Center
Stanford Research Institute
Menlo Park, California 94025

May 16, 1977

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense, and was monitored by the Naval Electronics Systems Command under Contract No. N00039-76-C-0364.

CONTENTS

PREFACE

This document contains a preliminary description of a new computer network conferencing control protocol. The information presented is not necessarily comprehensive enough for implementation; its orientation at this point is largely theoretical. The purpose of this presentation is to solicit feedback that will hopefully uncover any protocol problems, suggest useful extensions or improvements, and lead to an overall design that is acceptable to the research and user community.

1 Introduction

With the demonstrated feasibility of speech transmission via packet switched computer networks [1], attention has turned toward multi-participant conferencing. The processing power of packet switched networks provides an inherent capability for controlled real-time conferencing; simply needed is a VOICE protocol to mediate the exchange of intelligible speech across the VOICE data streams, plus a CONTROL protocol to direct the establishment and activity of these streams. Current conference CONTROL protocols [2] have been implemented as demonstration systems within specific networks; for more general user applicability, extensions to these systems would be desirable to provide for more flexible and 'natural' conferencing. To operate effectively in a user environment, a CONTROL protocol should:

(1) Have minimum CONTROL delays to avoid speech gaps and preserve real-time conferencing ability.

(2) Be capable of fully automatic operation to provide transparent, voice-directed conferencing.

(3) Allow formation of unscheduled conferences at user initiative.

(4) Provide for conferencing in any of several modes, including round-table (free floor), parliamentary (CHAIRMAN recognizes floor requests), structured and unstructured subconference, etc.

(5) Permit side-comments and interruptions.

(6) Keep the user informed of all pertinent information, such as participant list changes.

(7) Recover automatically from failures or error conditions with minimum real-time disruption of the conference.

(8) Be network-independent for internet applications.

Herein is presented a specification for a conference CONTROL protocol designed to satisfy the above criteria insofar as possible. The implementation of this protocol will be called the Voice Communication Supervisor (VCS). It is assumed that there exists an implementation of the VOICE protocol (denoted Voice Communication Transceiver or VCT) that interfaces to the VCS as described in Section IV. This document defines the logical language that VCS's use to communicate with foreign VCS's, local VCT's, and local users. The exact specification of this language in terms of actual message formats exchanged between network HOSTS is left to a future document.

II Design Approach

Owing to varied user needs and real-time requirements, network conference parameters can be highly dynamic and unpredictable. The VCS must therefore incur minimal CONTROL delay for functions such as SPEAKER or CHAIR handoff, while ensuring that such global reconfigurations be achieved robustly; the protocol must also be very modular to permit internet portability, as well as easy expansion or modification according to specific user needs.

At every network site supporting speech, there will be an arbitrary number of VCI's (extensions), and one VCS controlling them. A single user handset will be patched into each VCI, while all user interface devices (Section V) will communicate directly with the VCS. The VCS will initiate actions in response to messages received from other VCS's, from local users, and from local VCI's; these actions may include generating outbound CONTROL messages, SIGNALLING a VCI to perform some action, or sending information to the user interface device. A conference will consist of a set of fully-connected VCI's with one local user each, and the set of presiding VCS's which are in logical communication with the local user(s) and the CHAIRMAN VCS of the set.

The VCT VOICE network will be fully-connected to minimize SPEAKER handoff delays and to permit interruptions and side comments; this structure will also make the best use of the broadcast addressing mechanisms under development [3] while remaining compatible with single-destination protocols. The VCS CONTROL network will be a logical star centralized at the CHAIRMAN VCS (which can run automatically or under partial human control). The CHAIRMAN will serve as master of all negotiations and source of all mandatory directives. Since the VCS's will be homogeneous, the CHAIRMANSHIP will be relocatable to any participant site at any time by reconfiguration of the CONTROL network.

For rapidity, the CONTROL mechanism will be fully pipelined, and all CONTROL messages will have high information content. To avoid network specificity, references to network access will be made with respect to a hypothetical network interface (Section III) which provides the required high-level logical communication functions (without concerning the VCS about how the functions are accomplished). Since some network protocols are fully reliable (e.g., TCP), the conference CONTROL protocol will be specified to operate without command acknowledgments, but will provide them optionally if required.

III General Network Interface

Packet switched networks provide highly varied logical communication facilities. These include simplex-only connections (ARPANET NCP), full-duplex connections (TCP), and broadcast connections (SATNET and eventually PRNET). To regularize the VCS's operation within such diverse environments, a General Network Interface (GNI) has been designed. The GNI provides a minimum set of logical communication functions that should be realizable in most networks, and therefore permits network-related VCS actions to be specified in a network-independent manner. To run the VCS in a given network, the logical functions provided by the GNI must be made available, either by an actual implementation of the GNI, or by appropriate network management routines within the VCS itself.

All communication through the GNI is via communication streams called PATHS. A PATH is a logical mapping of a set of foreign addresses (PATHSET) onto the PATHNAME. PATHS are full-duplex communication streams; any packet placed on a given PATH is sent to all members of the PATHSET, and any packet received from a PATHSET member is associated with the PATHNAME upon delivery. The GNI provides the following asynchronous primitive operations:

(1) DEFINE <PATHNAME> [<PATHSET>]. Defines the logical mapping of foreign addresses. The GNI module must attempt to establish and maintain communication with every member of the PATHSET; this operation is network specific, and of no concern to the caller. If several foreign addresses reside at the same physical HOST, the GNI will open only one communication channel to that site, and the GNI there will distribute messages to all the destination processes. When finished, the GNI returns a message:

=> <PATHNAME> HAS [<PATHSET actually acquired>]

This message is issued any time the composition of the PATHSET changes (e.g., if a connection crashes). If the PATHSET in the DEFINE call is empty, a global receive-only listen PATH is created on which any packets not destined for a DEFINED PATH are received. Note that addresses in the PATHSET that are currently on other PATHS are implicitly moved to the new PATH.

(2) ADD <PATHNAME> [<PATHSET>]. Removes the specified foreign addresses from their current resident PATH(s), if any, and adds them to the (pre-defined) PATHNAME.

(3) REMOVE <PATHNAME> [<PATHSET>]. Removes the specified foreign addresses from the PATH. The GNI must terminate communication with those addresses (a connection close in most networks). If 'PATHSET' is empty, the entire PATH is undefined. Note that the caller can assume that the REMOVE operation is instantaneous in its action, even if connections in the host network close slowly, since the GNI will discard any data packets received while trying to close.

(4) TO [<PATHNAME>]: <MESSAGE>. Places the MESSAGE on the PATH; the GNI will send it to each foreign address. In a broadcast net, this would involve only one packet transmission, whereas with single-destination addressing, the packet would be sequentially sent to each destination (in the latter case, if several destinations exist at the same physical HOST, the GNI will send only a single copy of the message there; the receiving GNI will distribute it to all local processes that have a PATH containing the message source). Note that in place of a PATHNAME, a PATH subset may be specified by listing the foreign addresses (internet ID's) that comprise the subset. The message will be sent only to the PATH members listed.

(5) FROM [<PATHNAME>]:. Requests that the GNI deliver the next MESSAGE on the PATH; the PATHNAME may be a PATH

subset of foreign addresses as in the 'IU' primitive. The GNI will usually try to keep ahead of caller requests by acquiring a small received packet queue for each PARTSET member, but many strategies are possible.

In some (unreliable or broadcast) networks, it may not be possible for the GNI to explicitly know when communication with a foreign site has been established or lost. In those cases, the GNI will immediately return a fully successful 'HAS' message in response to a 'DEFINE', 'ADD', or 'REMOVE'. It is the caller's final responsibility (probably via timeouts) to determine when a foreign site is unavailable.

IV The VCT Interface

The VCT module implements the VOICE protocol of the network speech conferencing system. The specifics of this protocol are of no concern to the VCS, since the VCS directs only high-level VCT functions. Conferences are made possible by the selective control of these functions, which include VOICE network establishment, VCT parameter change, VOICE transmission/reception on selected streams, and enabling/disabling of limited feedback mode (Section XII). The VCS controls these activities and receives status reports through inter-process signals as follows:

(1) Network action signals (VCS to VCT).

(a) ACQUIRE [<PATHSET>]. This signal supplies the addresses of all (new) foreign VCT's in the conference. It evokes a 'DEFINE VOICENET [<PATHSET>]' by the local VCT if the VOICENET does not yet exist, else an 'ADD VOICENET [<PATHSET>]' is issued.

(b) DROP [<PATHSET>]. The inverse of an 'ACQUIRE'. Packet streams from members of the PATHSET are flushed.

(c) PARMS [<VALUES>]. Sets negotiable VCT parameters, including 'DATA-RATE', 'MESSAGE-LENGTH', 'VOCODER' type (Section VII).

(2) Network status signals (VCT to VCS).

(a) ACQUIRED [<PATHSET>]. Informs the VCS of the successful acquisition of foreign VCT's.

(b) [SQUELCH [<PATHSET>]. Informs the VCS that communication with some member(s) of the VOICENET has been lost.

(c) FAR [<PATHSET>]. When packets are received from a foreign VCT which has been silent for some time, the VCS is informed that Foreign Activity has been Detected.

(d) FAL [<PATHSET>]. When packet reception from a foreign VCT has stopped, the VCS is informed that Foreign Activity has been Lost.

(3) Local action signals (VCS to VCT).

(a) QUIET. Inhibits VCT VOICE transmission, except to the extent allowed by the VOICE protocol for limited feedback (Section XII).

(b) TALK [<PATHSET>]. Enables the VCT to send VOICE data to 'VOICENET' if the 'PATHSET' is empty. Otherwise, VOICE data is sent only to the specified subset of 'VOICENET' (for side-comment applications, Section IX). The first VOICE message generated after getting a 'TALK' is assigned a sequence number of zero to help distinguish the incipient VOICE stream from delayed remnants of a previous one.

(c) LISTEN [<PATHSET>]. Causes the VCT to output packets received from the specified foreign VCT's to the speech decoder, starting with sequence number zero on each stream (unless a timeout forces resynchronization). If the PATHSET has more than one member (Section XII), the VOICE streams will have to be mixed, whereas if the set is empty, the VCT is inhibited from further reception. The PATHSET of a LISTEN supercedes that of any previous LISTEN. For the VOICE protocol to provide limited feedback, all VOICE reception is done via 'FROM VOICENET:' calls; packets from the 'LISTEN' PATHSET are shuttled to the VOICE hardware, while others may be discarded or placed on small queues (when appropriate) for strategic insertion into the 'LISTEN' stream.

(d) FROM. Turns on limited feedback, allowing regulated speech transmission while 'QUIET', and regulated global speech reception from other PARTICIPANTS while 'LISTENING' to the SPEAKER.

(c) FBOFF. Disables limited feedback. No speech may be sent to 'VOICENET' while 'BUSY', and speech will only be accepted from the designated 'LISTEN' stream.

(f) TONE [<TONE TYPE>]. Causes the VCT to output a tone (or canned speech message) of the specified type to the handset. This signal is provided so that the VCS can cue a user lacking a visual interface when a foreign site is unavailable ('BUSY'), when he is granted the floor ('SPEAK'), when he has lost the floor ('SHUT-UP'), and when the foreign SPEAKER has been silent for some time ('IDLE').

(4) Local status signals (VCT to VCS).

(a) LAD. Issued when new local speech Activity is detected by the VCT ('coming out of silence'). This signal provides the capability for VCS floor requests based upon speech activity (to make a vocal floor request, the user might be required to utter a specific audio cue).

(b) LAL. Issued when the VCT determines that local speech Activity has been Lost ('going into silence').

V The User Interface

To utilize the network conferencing system, the user is interfaced to his VCT through a handset, and to the VCS through a special device. In the ARPANET experiments, a light box with pushbuttons was used, but the expanded capabilities of the VCS justify more elaborate user controls. It is suggested that the input device be a touchtone pad, and that output be via a sixteen character alphanumeric readout.

Through the interface device, users will be able to direct all VCS functions, even normally automatic ones if desired. The following list specifies most of these, with 'A' denoting a normally automatic function that is transparent to the user, 'M' a normally manual function initiated by the user, 'P' one that is under automatic and manual control simultaneously, and 'C' a function applicable only to the CHAIRMAN.

(1) Conference establishment.

(a) M: Foreign user call-up.

(b) M: Answering calls (actually, just picking up the handset could accomplish this).

(c) AC: Conference generation.

(d) M: Joining a conference in progress.

(e) AM: Leaving a conference or terminating a call.

(f) A: VCS parameter negotiation (e.g., PARTICIPATORY or ROUND-TABLE floor HANDOFF, VCS message ACK or NOACK, choice of VCS automatic or manual modes, etc.).

(g) A: VCT parameter negotiation (e.g., DATA-RATE, VOCODER type, MESSAGE-LENGTH).

(2) General configuration requests.

(a) AR: Getting a typeout of any aspect of the current conference configuration, including the 'ID' of the current CHAIRMAN or SPEAKER, composition of the PARTICIPANT list, VCT and VCS parameters in effect, etc.

(b) R: Requesting change of any parameter (e.g., floor HANDOFF mode change request of CHAIRMAN).

(c) RC: Accepting or rejecting a parameter (or mode) request.

(3) Chair management.

(a) R: CHAIR requests.

(b) RC: Granting CHAIR requests.

(c) MC: Passing the CHAIR.

(4) Floor management.

(a) AR: Floor requests (automatic if based on speech activity).

(b) AC: Granting the floor.

(c) AR: Passing the floor.

(d) R: Interruption requests.

(e) R: Granting or refusing interruption requests.

(f) ARC: Opening a VOICE channel to the SPEAKER (automatic only for a two-party conference to achieve full-duplex conversation, else manual to enable the CHAIRMAN to make private comments on the SPEAKER's otherwise unused LISTEN stream. See Section IX).

(g) MC: Closing the VOICE channel to the SPEAKER.

(5) Side comment management.

(a) M: Making side-comment calls.

(b) M: Answering or refusing side comment calls.

(6) Subconference formation.

(a) M: Subconference formation requests.

(b) MC: Issuing subconference formation directives.

(c) MC: Denying subconference formation requests.

(d) M: Accepting or rejecting a subconference directive or call.

(7) Voting.

(a) MC: Calling for a vote.

(b) M: Voting 'AYE', 'NAY', or 'ABSTAIN'.

Through the alphanumeric readout, the VCS will respond to user requests, and keep him abreast of all pertinent information, including:

(1) Conference establishment.

(a) 'RINGING user <ID>'.

(b) 'User <ID> is BUSY'.

(c) 'User <ID> is AVAILABLE'.

(d) 'Call from user <ID>'.

(e) 'CHAIRMAN is <ID>'.

(f) 'PARTICIPANTS are <ID list>'.

(g) 'PARTICIPANT <ID> has joined'.

(b) 'PARTICIPANT <ID> has left because <REASONS>'.

(2) Configuration requests.

(a) 'VCS parameters are:' (e.g., PARLIAMENTARY floor HANDOFF, command ACK).

(b) 'VCI parameters are:' (e.g., VOCODER TYPE CVSD, DATA-RATE 16 KBITS).

(c) 'Asking <user request> of CHAIRMAN' (i.e., VCS or VCI parameter request).

(d) C: '<User request> from user <ID>' (only seen for requests that must be granted manually).

(e) '<User request> accepted by CHAIRMAN' (e.g., request for PARLIAMENTARY floor HANDOFF accepted by CHAIRMAN).

(f) '<User request> denied by CHAIRMAN'.

(3) Chair management.

(a) C: 'User <ID> requests the CHAIR'.

(b) C: 'You have the CHAIR'.

(c) C: 'You have lost the CHAIR'.

(4) Floor management.

(a) C: 'User <ID> requests the floor' (seen only in manual floor HANDOFF mode).

(b) 'You have the floor'.

(c) 'You will lose the floor in <n> minutes'.

(d) C: 'User <ID> has passed the floor' (seen only in manual floor HANDOFF mode).

(e) 'You have lost the floor'.

(f) 'Interruption request from user <ID>'.

(g) 'Interruption request denied by user <ID>'.

(h) C: 'VOICE channel to SPEAKER opened'.

(i) C: 'VOICE channel to SPEAKER closed'.

(5) Side-comment management.

(a) 'Side-comment call from user <ID>'.

(b) 'Side comment call refused by user <ID>'.

(c) 'Side-comment channel opened to user <ID>'.

(d) 'Side-comment channel to user <ID> closed'.

(6) Subconference formation.

(a) C: 'Request from user <ID> to subconference with <ID list>'.

(b) 'Subconference request denied by CHAIRMAN'.

(c) 'Start a subconference with <ID list>'.

(d) 'Subconference call from user <ID>'.

(e) C: 'User <ID> refuses to subconference with <ID list>'.

(7) Voting.

(a) 'Vote "AYE", "NAY", or "ABSTAIN"'.

(b) C: 'Vote results: <counts>'.

In some instances, an output interface device may not be available or needed. The VCS is designed to be operable without any direction at all if desired, basing its actions upon speech activity, and informing the user of status with appropriate TONES (Section IV). However, in this mode, the user is unable to access some conference functions (such as status typeout).

VI The VCS Interface

VCS's communicate by exchanging messages according to the conference CONTROL protocol. These messages conform to a simple grammar which facilitates automation, modularization, and implementation of the protocol.

Three word classes comprise the grammar: VERBS, OBJECTS, and ADJECTIVES. Messages begin with a single VERB which specifies a kind of action, potential or intended, that is applicable to the rest of the message (the OBJECT LIST). VERBS include:

(1) DO: A directive, usually mandatory when issued by the CHAIRMAN VCS. 'DO' specifies an action that the source wants the destination to perform upon receipt.

(2) WANT: A request to perform some action, normally sent from PARTICIPANT to CHAIRMAN VCS, and reflecting a user desire (e.g., to assume the floor). The action will not be initiated without permission of the destination (i.e., getting back a 'DO').

(3) AM: Informs the destination of some aspect(s) of the source's state. 'AM' is also an acknowledgment to a 'DO' if command acknowledgment is in effect.

(4) WONT: A rejection of or negative acknowledgment to a 'DO' or 'WANT', often reflecting user unwillingness (to serve as CHAIRMAN, for instance).

(5) CAN: Generated by PARTICIPANTS, 'CAN' informs the CHAIRMAN of local (primarily VCT) capabilities, such as ability to use CVSD, LPC, etc.

(6) CANT: Infrequently used, 'CANT' informs the CHAIRMAN that some capability previously identified in a 'CAN' has been lost.

VERBS refer to a succeeding list of OBJECTS, some of which take mandatory ADJECTIVES; the message format is therefore:

<VERB> [<OBJECT>{,<ADJECTIVE>}; ...;<OBJECT>{,<ADJECTIVE>}]

The sections that follow specify the message exchange sequences that the VCS uses to establish the CONTROL network, as well as those interactions that invoke direction of the VCT's; the various OBJECTS and ADJECTIVES that comprise these messages are defined as they arise in context.

VII Negotiation

Inter-VCS negotiation is the fundamental mechanism by which a conference achieves the configuration desired by the users; for flexibility, any global conference parameter value like VOCODER type, or any FUNCTION (such as SPEAKER) is a negotiable OBJECT. The VCS utilizes a new negotiation philosophy designed to facilitate the rapid optimization of these parameters and the assignment of the conference FUNCTIONS; for efficiency, this process is integral to initial conference generation (see Section VIII), but may be re-triggered at any time.

Negotiation takes place between a single MASTER and an arbitrary number of SLAVES; in an established conference, the current CHAIRMAN is always MASTER, and the PARTICIPANTS are SLAVES. A negotiation session is initiated by a WANT message from one of the PARTICIPANTS:

P->C: WANT [<OBJECT LIST>]

A 'DO' will grant the request, and a 'WONT' will deny it. If the VCS wants to postpone replying to a request (such as to be SPEAKER), it can simply acknowledge receipt of the request with an 'AM RESPOND,<REQUEST OBJECT LIST>]' (see Section X), and send the 'DO' or 'WONT' later. The 'AM RESPOND' is optional if the VCS is running without command acknowledgment, since this mode is only used if CONTROL connections are fully reliable.

FUNCTIONS requested in the 'WANT' message usually require no further message exchange; when the CHAIRMAN is willing to assign the FUNCTION to the PARTICIPANT, he will send:

C->P: DO [<OBJECT LIST>]

The OBJECT LIST contains the FUNCTION.

Requested global parameters must be acceptable to all PARTICIPANTS before they can be effected; the CHAIRMAN will never send a 'DO' to a PARTICIPANT unless he knows that the PARTICIPANT has the specified CAPABILITY. To compile this kind of information, a message requesting a CAPABILITIES report is sent to all PARTICIPANTS when necessary:

C->All P's: DO [<CAPABILITY LIST>]

All P's->C: CAN [<CAPABILITY LIST>]

CAPABILITY LIST (CLIST) OBJECTS include:

(1) VOCODER: vocoder types available to the VCT (e.g., CVSD, LPC, RELP, DELCO).

(2) DATA-RATE: the VOICE bandwidth range that can be accommodated by the local VCT.

(3) MESSAGE-LENGTH: the range of message lengths that can be accommodated by the local VCT.

(4) VOCODER-PARMS: acceptable ranges for parameters specific to a given vocoder type (e.g, LPC PRE-EMPHASIS, LPC ACOUSTIC-CODING, LPC INFO-CODING).

(5) C-VERSION: which versions of the CONTROL protocol the VCS supports (e.g, ACK--command acknowledgment, NOACK--no acknowledgment, LOCK--FUNCTION handoff interlocking, NOLOCK--no interlocking). This CAPABILITY CLASS is intended for adapting VCS operation to the performance characteristics (reliability and delay) of a specific network communication facility.

(6) V-VERSION: which versions of the VOICE protocol the VCI supports (e.g., NVP-HEADER--NVP' header format for VOICE messages [1], FEEDBACK- ability to handle limited feedback, MIXING- ability to handle multiple SPEAKER's).

(7) FUNCTIONS: functions that the local VCS/VCI installation can perform (e.g, SPEAKER, CHAIRMAN, SCRIBE).

Notice that each CAPABILITY OBJECT takes ADJECTIVES. When empty in the 'DO' message, a 'CAN' is returned with ADJECTIVES to inform the CHAIRMAN of the PARTICIPANT's full range of CAPABILITIES for each unmodified CAPABILITY OBJECT (in command 'ACK' mode, the CHAIRMAN may indicate receipt of the 'CANS' via 'AM (RESPOND,<CLIST>]'); after a single round-trip time, the CHAIRMAN will therefore possess a data base that enables him to respond to all WANTS without querying other PARTICIPANTS. A CAPABILITY OBJECT accompanied by a single valued ADJECTIVE in a 'WANT' is a request to set that value, whereas in a 'DO', it is a directive to do so.

After inspecting all 'WANTS' and PARTICIPANT CAPABILITIES, the CHAIRMAN will select the set of global parameter values most acceptable to all and send:

C->All P's: DO [<OBJECT LIST>]

Each CAPABILITY OBJECT in the list has an ADJECTIVE that specifies what value to set. For any other requested CAPABILITY OBJECT values not globally acceptable, the requesting PARTICIPANT will get back a 'WONT'.

Whenever a PARTICIPANT receives a 'DO', he will take the specified action(s) immediately, returning an 'AM [<OBJECT LIST>]' if command

acknowledgment is in effect. Consequently, reconfiguring a conference can be as simple and speedy as a 'WANT'--global-'DO' exchange. Should a 'WONT' be returned to the global 'DO' (a rare occurrence that could only happen through user intervention), the CHAIRMAN will broadcast another 'DO' to reset the conference to its initial state; a 'WONT' reply to this message is grounds for automatic dismissal from the conference (you can't please everybody--see Section XII).

To further trim negotiation, a set of default CAPABILITIES will be specified (see Section XI). The CHAIRMAN can always assume that foreign sites have these CAPABILITIES unless (during conference generation) he receives a 'CANT [<OBJECT LIST>]' from some of them. A site may also generate such a message if a previously possessed CAPABILITY is lost for any reason. In this event, the CHAIRMAN may have to choose a new set of global conference parameters, or else dismiss the troubled PARTICIPANT from the conference.

Other notes:

(1) A 'WONT' on some OBJECT cancels a previous 'WANT'.

(2) A 'DO' on an OBJECT received subsequent to sending a 'WONT' or 'CANT' elicits another 'WONT' or 'CANT' respectively.

(3) The negotiation MASTER ignores repetitious 'WONTS' and 'WANTS'.

(4) 'WANTS' not within the requester's CAPABILITIES are 'WONTED'.

(5) 'WANTS' that would not change the requester's state if fulfilled are granted.

(6) 'NOS' that will not change the receiver's state are acknowledged with 'AK' if command acknowledgment is in effect, else they are ignored.

(7) Any other meaningless message is ignored.

(8) Objects from different CLASSES (e.g., CAPABILITIES, FUNCTIONS) may be mixed in the same message in any fashion.

VIII Establishing a Conference

A conference is formed when a set of users try to mutually call one another, or instruct their VCS's to do the 'dialing' for them; such calls are received on a global listen PATH ('CALL') for which VCS's always maintain outstanding FROM:'s (Section III). Normally, conferences will be scheduled in advance, and a single VCS (the designated CHAIRMAN) will call PARTICIPANTS in. However, the conference generation mechanism will function properly even if there are multiple active callers. OBJECTS relevant to conference generation include all CAPABILITIES, as well as:

(1) CONFER. References the act of conferring with a user. CONFER takes one of several ADJECTIVES which specify the reason for the reference, some of which include:

(a) SCHLD,<CONF ID>: Scheduled conference, assigned 'ID'.

(b) UNSCHED,<USER ID>: Unscheduled call from user 'ID'.

(c) SUBCONF,<SUBCHAIR ID>: Subconference CHAIRED by 'ID'.

(d) SIDE-COM,<USER ID>: Side-comment request from user 'ID'.

(e) C-HANDOFF,<NEW-CHAIR ID>: CHAIR handoff to user 'ID'.

(f) RECOVER,<CHAIR ID>: Trying to recover from loss of CHAIRMAN 'ID' (he crashed).

(g) UNAUTH,<USER ID>: Unauthorized to speak to 'ID'.

(h) IN CONF,<CHAIR ID>: In conference with CHAIRMAN 'ID'.

(i) PCALL,<USER ID>: Called by higher PRIORITY user 'ID', joining him.

(j) FAIL: Negotiation failure; site unserviceable.

(2) QUIT. References the act of leaving a conference or terminating communications. QUIT takes an ADJECTIVE which indicates the reason for leaving--

(a) END,<CONF ID>: Conference 'ID' has ended.

(b) ALTCON,<CHAIR ID>: Joining an alternate conference CHAIRED by 'ID'.

(c) UNSPEC: Reason for 'QUIT' unspecified.

(d) SUBCONF, C-HANDOFF, PCALL, SIDE-CONF, and FAIL as above.

(3) MEMBERS. Refers to the PARTICIPANT composition of the conference and the identity of the CHAIRMAN. MEMBERS takes as an ADJECTIVE the PARTICIPANT LIST (PLIST) of the conference, ordered by PARTICIPANT PRIORITY (Section IX); the CHAIRMAN is first on the LIST.

In the preceding compilation, 'CONF ID' is a logical name assigned to the conference in advance and distributed to each PARTICIPANT site, while 'USER ID' and 'CHAIR ID' consist of a). the internet ID of the VCS servicing the indicated user, b). his VCI extension number at that site, and c). his name. Once the PARTICIPANT LIST has been distributed to all sites (see below), CONTROL messages between VCS's that contain ID's need only include the internet ID and extension number portions of the user 'ID'. Conversely, messages for the user device containing a

user 'ID' will be formatted to include only the user name part of the 'ID', which can be found in the PARTICIPANT LIST.

To establish a conference, a VCS sends a 'DO [CONFER,<REASON>]' message to each destination on the desired PARTICIPANT LIST; the originator of this message is implicitly attempting to CHAIR the conference, and a positive reply implicitly accepts him:

C->PLIST: DO [CONFER,<REASON>]

P->C: WONT [CONFER,<REASON>] -- Rejects

P->C: WANT [CONFER,<REASON>; <WANT LIST>] -- Accepts

The negative reply terminates the communication for the specified REASON. The positive reply echoes back the REASON supplied in the 'DO', and usually includes negotiation requests (in the WANT LIST) for OBJECTS that need to be specified during conference generation. Pertinent OBJECTS include 'VOCODER', 'MESSAGE-LENGTH', 'DATA-RATE', and so on. If the WANT LISTS of the accepting PARTICIPANTS are incomplete or conflicting, the CHAIRMAN will have to obtain more complete CAPABILITY information at this point by issuing to all:

C->All P's: DO [<CAPABILITY LIST>]

The CLIST will consist primarily of unmodified CAPABILITY OBJECTS, but could include some directives as well if the CHAIRMAN were already able to finalize some parameters. When the CAN replies filter back:

C->Some P's: DO [CONFER; <DO LIST>] -- They are acquired

C->Other P's: WONT [CONFER,FAIL] -- They are rejected

As above, the 'WONT [CONFER]' terminates communication. If command acknowledgment is in effect, the accepted PARTICIPANTS return an AII which echoes the contents of the preceding DO:

P's >C: AII [CONFER,<REASON>; <DO LIST>]

Finally, the CHAIRMAN dispatches a message to the PARTICIPANTS to inform them of the conference PLIST:

C->P's: DO [MEMBERS,<PLIST>]

Each local VCS will display the PLIST to the user and direct its VCI(s) to establish the VOICENET; conferencing can begin.

Notice that throughout this conference generation mechanism, all PARTICIPANTS were handled simultaneously in a pipelined fashion, and that the conference is fully configured after only two or three round-trip message delays.

A caller does not necessarily have to attempt to assume the CHAIRMANSHIP; there is a passive call mechanism that will normally be used for unscheduled calls, and it implicitly puts the responder in the CHAIRMAN role. To make a passive call, the VCS simply sends:

P->C: WANT [CONFER,<REASON>; <WANT LIST>]

The responder need now only issue a 'WONT' or a 'DO', and proceed exactly as above. In fact, the only difference here is that a 'WANT [CONFER]' was dispatched without a 'DO [CONFER]' to evoke it. Note that it is possible to dispatch a 'WANT [CONFER]' in this way, and then receive that very 'DO [CONFER]' (i.e., passive call to a site that is simultaneously actively calling you); the 'DO' would be ignored.

In a well populated speech network, a VCS actively establishing a conference will occasionally receive a 'DO [CONFER]' from another VCS. If that VCS is not on his PARTICIPANT LIST, a 'WONT' will probably be returned ('REASON' being 'IN-CONF'). Otherwise, there is a potential deadlock since both are trying to initiate the same conference, and have exchanged 'DOS'; somehow, one must surrender control to the other. When a 'DO [CONFER]' (or 'WANT [CONFER]') collision occurs, it is suggested that the involved VCS's compare their respective internet ID's, treated as unsigned sixty-four bit integers. The smaller 'ID' has priority and will be CHAIRMAN. For a 'DO' collision, the larger 'ID' will therefore send the standard 'WANT [CONFER]', while for a 'WANT' collision, the lower 'ID' will issue a 'DO'. This will allow conference generation to get underway automatically and without delay when there are multiple active callers, and if they like, the users can reassign the CHAIR (Section IX) once the conference has started.

When one VCS surrenders control to another during conference establishment, it must also dismiss any PARTICIPANTS that have been acquired. This will be done via:

C->Acquired P's: DO [QUIT,PCALL,<USER ID>]

The surrendering VCS will also terminate communication with sites that are not yet fully acquired (i.e., 'DO [CONFER; <DO LIST>]' not yet sent):

C->Partially acquired P's: WONT [CONFER,PCALL,<USER ID>]

The 'HD' serves as a referral to the higher priority VCS so that the dismissed PARTICIPANTS can send it a 'WAIT' if desired. As usual, with command acknowledgment, an 'AM (QUIT,<REASON>)' will be returned to the CHAIRMAN. This same message is used any time a PARTICIPANT decides to leave a conference.

IX Conducting a Conference

Once the CONTROL and VOICE networks have been established, the conference can begin. Conferences are conducted by the CHAIRMAN through dynamic assignment of FUNCTIONS, usually in response to user requests. Therefore, the CHAIRMAN should receive a series of 'WANT' messages as soon as he has dispatched the 'DO [MEMBERS]'. Although not explicitly stated from here on, an accepted 'DO' will always evoke an 'AK' if command acknowledgment is in effect, while a 'WONT' will be returned to an unaccepted 'DO' (or 'WANT') regardless of the acknowledgment mode.

The FUNCTION most frequently requested and reassigned is:

SPEAKER. References possession of the floor, and takes as an ADJECTIVE a user ID.

To request the floor, a user will punch a button on his VCS interface device, or he might utter a special cue into the handset (the VCS would detect the 'LAD' signal). This evokes a 'WANT [SPEAKER,<ID>]' from the VCS, where 'ID' is the identity of the requesting user. To switch SPEAKERS takes only a single message transmission: the CHAIRMAN will send to all PARTICIPANTS a 'DO [SPEAKER,<ID>]', resulting in:

(1) The previous SPEAKER's VCT is SIGNALLED 'QUIET' and 'LISTEN [<SPEAKER ID>]'.

(2) The new SPEAKER's VCT is SIGNALLED 'TALK' and 'LISTEN [<CHAIRMAN ID>]'.

(3) All other VCT's are SIGNALLED 'LISTEN [<SPEAKER ID>]'.

Notice that when a PARTICIPANT assumes the floor, his VCT will receive no input from the VOICENET (except for limited feedback streams). His LISTEN stream is therefore defaulted to the CHAIRMAN, who may now talk to the SPEAKER by SIGNALLING 'TALK [<SPEAKER ID>]' to the CHAIRMAN VCT. This facility is most useful for a two-party conversation (a degenerate conference with a CHAIRMAN and only one PARTICIPANT) since it provides full-duplex VOICE communication between CHAIRMAN and SPEAKER. For two-party calls, the CHAIRMAN VCS will therefore automatically open this VOICE channel to the SPEAKER. Otherwise, the CHAIRMAN user can open the channel by special command to his VCS when he wants to make comments to the SPEAKER.

Large, widely variable message delays are typical in some networks, necessitating interlocking of FUNCTION HANDOFF's. Interlocking requires that command acknowledgment be in effect ('C-VERSION ACK'), and is negotiated with the 'C-VERSION LOCK' OBJECT (Section VII); it ensures that all other PARTICIPANTS are aware of the HANDOFF before the FUNCTION is actually reassigned. When interlocking, floor HANDOFF is achieved by:

(1) The 'DO [SPEAKER,<NEW SPEAKER ID>]' is broadcast to all but the new SPEAKER.

(2) The CHAIRMAN awaits reception of all 'AT [SPEAKER,<NEW SPEAKER ID>]' acknowledgments.

(3) The CHAIRMAN sends 'DO [SPEAKER,<NEW SPEAKER ID>]' to the new SPEAKER.

The CHAIRMAN may utilize any of several floor HANDOFF decision strategies, negotiable through:

F-HANDOFF. References the floor HANDOFF algorithm used by the CHAIRMAN. ADJECTIVES include:

(1) PARLIAMENTARY. The CHAIRMAN recognizes requesting PARTICIPANTS in any order at his discretion (e.g., via Robert's Rules of Order). In this mode, the user at the CHAIRMAN site may want to perform SPEAKER HANDOFF manually. Otherwise, the CHAIRMAN VCS will utilize some appropriate function of request order and requester PRIORITY.

(2) ROUND-TABLE. The floor is granted on a first-come-first-served basis. ROUND-TABLE mode is best handled automatically.

The default mode is ROUND-TABLE, although a user may request another mode at any time in a 'WANT' message; the CHAIRMAN will reply 'WONT' if the requested mode is unacceptable, else 'AM'.

The CHAIRMAN may sometimes want to retract the floor from a PARTICIPANT, warning him in advance (if he has been too verbose, or silent for too long); conversely, a PARTICIPANT must be able to inform the CHAIRMAN when he is finished being SPEAKER:

FINISH. References relinquishment of the SPEAKER FUNCTION. A TIME is supplied as an ADJECTIVE; it indicates the time at which the FINISH will take effect.

To make a PARTICIPANT summarize, the CHAIRMAN will send him a 'DO (FINISH,TIME)'. This message will be displayed on the user's interface device, and after the specified interval, his VCS will 'QUIET' the VCT.

A PARTICIPANT can voluntarily relinquish the floor with an 'AM [FINISH,0]' (i.e., finished now). If desired, 'FINISH' messages can be exchanged to interlock relinquishment of the floor before a new SPEAKER is assigned.

In a PARLIAMENTARY conference, floor HAND-OFF is largely determined by PARTICIPANT PRIORITY (i.e., pecking order). PRIORITY is implicit in the order of the conference PLIST, which is determined by the CHAIRMAN just prior to issuing the 'DO [MEMBERS,<PLIST>]'. PARTICIPANTS not satisfied with this assignment may request another via a 'WANT [MEMBERS,<PLIST>]', wherein they have specified a new PRIORITY ordering. The CHAIRMAN rejects the request with the usual 'DONT', while to accept it, he broadcasts a new 'DO [MEMBERS,<PLIST>]' to all (note that 'DO [MEMBERS,<PLIST>]' is also dispatched to evoke reconfiguration of everyone's VOICENET whenever the PLIST changes through PARTICIPANT addition, deletion, or loss).

If a PARTICIPANT places his own 'ID' at the head of the PLIST in a PRIORITY request, then he is asking for the CHAIR; a 'DO [MEMBERS]' with his 'ID' in this position is a directive to take the CHAIR. CHAIR HANDOFF may occur at any time, either by such request or at the discretion of the CHAIRMAN; it is achieved by the same mechanism that is used to establish a conference. To initiate a CHAIR HANDOFF, the CHAIRMAN sends 'DO [MEMBERS,<PLIST>]' to the potential CHAIRMAN, who then (if agreeable) broadcasts to all PARTICIPANTS, 'DO [MEMBERS,C-HANDOFF,<NEW CHAIR ID>]'; exchanges now occur as outlined in Section

VIII. When a PARTICIPANT becomes fully acquired by the new CHAIRMAN, he will terminate CONTROL communication with the old CHAIRMAN with 'AM [QUIT,C-HANDOFF,<NEW CHAIR ID>]'. Finally, the new CHAIRMAN will broadcast a 'DO [MEMBERS,<PLIST>]'; if all original PARTICIPANTS are not on the PLIST, then the old CHAIRMAN will have to 'DO [QUIT,FAIL]' them from the conference.

For security reasons in some instances, a PARTICIPANT may not trust the sender of a C-HANDOFF 'DO'. If so, he will hold his reply in abeyance and query the current CHAIRMAN with 'WANT [QUIT,C-HANDOFF,<NEW CHAIR ID>]'; a 'DO' will give him the go-ahead, and a 'WONT' will prove that you can only fool some of the people some of the time.

CHAIR HANDOFF does not require interruption of any conference FUNCTIONS; however, to be certain that the new CHAIRMAN's information is up to date, PARTICIPANTS with assigned FUNCTIONS (such as SPEAKER) should send him an 'AM [FUNCTION]'. Copies of 'WANTS' not granted by the old CHAIRMAN should also be sent to the new CHAIRMAN by all PARTICIPANTS.

Unstructured subconferencing (SESSIONING) is achieved through the normal CHAIR HANDOFF mechanism, except that:

(1) The OBJECT 'SESSION,<PLIST>' is used to request/grant a SESSION SUBCHAIRMANSHIP, as opposed to the 'MEMBERS,<PLIST>' for CHAIR HANDOFF.

(2) The ADJECTIVE for 'CONFER' and 'QUIT' is 'SUBCONF' instead of 'C-HANDOFF'.

(3) Instead of sending the final 'IAO DEPARTS,<PLIST>]' to the old CHAIRMAN, the SUBCHAIRMAN issues him an 'ALL [WAIT, SUBCONF,<SUBCHAIR IDs>]'. Note that the 'IAO DEPARTS,<PLIST>]' is still broadcast to the acquired PARTICIPANTS, as in regular CHAIR HANDOFF.

The effect is therefore to fragment a conference into fully independent entities.

When a subconference ends, PARTICIPANTS wishing to rejoin the main conference will have to re-connect to the old CHAIRMAN via the standard 'WANT [CONFER]' passive call. Of course, the CHAIR may have shifted in the interim, in which case the reply will be 'WENT [CONFER,IN-CONF,<CHAIR IDs>]'; the requester is thereby referred to the current CHAIRMAN.

In a natural conference, participants can whisper among themselves at will; such a side-commenting ability can be very useful, and is provided through a special passive conference call from PARTICIPANT to PARTICIPANT: 'WANT [CONFER,SIDE-COM,<CALLER IDs>]'. Naturally, a 'IAO' accepts the side-comment request, and the affected user VCT's are directed to switch their 'LISTEN' streams and 'TALK [<FOREIGN SIDE-COMMENTER IDs>]'. The two PARTICIPANTS may now conduct an independent (full-duplex) conversation; in effect, they have formed a two-party subconference, with the receiver of the side-comment call as CHAIRMAN. However, both PARTICIPANTS will still receive messages from the actual CHAIRMAN, as well as feedback streams from the conference 'VOICEFILE'.

Side comment sessions are terminated via the normal 'QUIT' mechanism, but with 'SIDE-COM' as the ADJECTIVE. The VCI's of both side-commenters will be reset back to 'QUIET' and 'LISTENING' to the SPEAKER stream; any parameters that were renegotiated for the side-comment session will revert to current conference values. Note that a 'SIDE-COM' request to the SPEAKER will normally be refused (unless he wishes to relinquish the floor first), while acquisition of the floor by a side-commenter will usually terminate the session (unless he wants to pass).

Akin to side-commenting is the ability to interrupt the SPEAKER for making a global comment, or even for providing a bit of auditory feedback (i.e., short noises from the listeners, such as 'hmm' or 'uh-huh'). A small amount of interruption (limited feedback, Section XII) could be provided by the VOICE protocol automatically for the latter purpose if 'FEEDBACK' is globally implemented (Section VII), but for interjections lasting more than a second or so, the CHAIRMAN must grant a PARTICIPANT permission to interrupt:

> INTERRUPT. References interruption of the SPEAKER by a user whose ID is supplied as one ADJECTIVE, for a time specified by a second ADJECTIVE.

A user will request an INTERRUPT through his VCS interface device, and the VCS will dispatch a 'WANT [INTERRUPT,<REQUESTER ID>,TIME]'. The CHAIRMAN may now either make the decision independently, or consult the SPEAKER by relaying the 'WANT' to him; the SPEAKER can accept by

returning a 'DO'. To achieve the interruption, the CHAIRMAN simply broadcasts 'DO [INTERRUPT,<REQUESTER ID>,TIME]' to all PARTICIPANTS; VCT's will be switched to the interrupter's VOICE stream for the specified period, and will then revert back to the SPEAKER's. This is effectively a very limited floor HANDOFF, with automatic switch back, and should provide for effective real-time interruption (but only in a network with small CONTROL delays).

Conferences can be characterized by frequent voting, awkward in a teleconferencing environment. To simplify this process, a special OBJECT is provided:

VOTE. References voting, and takes 'AYE', 'NAY', or 'ABSTAIN' as ADJECTIVES.

When the CHAIRMAN calls for a vote, a 'DO [VOTE]' is broadcast, which causes each VCS to display a 'VOTE' message to the local user(s) (ensuring that side-commenters can VOTE). There will be user device codes to elicit an 'AM [VOTE]' with 'AYE', 'NAY', or 'ABSTAIN' as an ADJECTIVE; the CHAIRMAN VCS will tabulate these responses, counting any too tardy as 'ABSTAIN', and display the results to the CHAIRMAN user. If a user generates more than one VOTE response, only his last will be counted.

X Failure Recovery

Since packet switched networks are very dynamic and often overburdened, crashes occur unpredictably. In a conference, terminal node failure will usually isolate a single PARTICIPANT, who is sometimes the CHAIRMAN; more seriously, a subnet problem could partition the communication network, separating the conference into several pieces. The VCS provides for automatic recovery from these conditions through the normal conference generation mechanism. To detect their occurrence, the following OBJECT is employed:

RESPOND. References the act of responding. A datum accompanies the RESPOND as an ADJECTIVE, and serves to tag it.

The CHAIRMAN and PARTICIPANTS exchange 'RESPOND' messages when there has been no traffic between them for some time. The originator will dispatch a 'DO [RESPOND,<DATA>]', and the responder will reply with an 'AM', returning the DATA supplied in the 'DO'; failure to do so after a specified timeout implies that the responder has crashed. In command 'ACK' mode, 'AM [RESPOND]' is also used to acknowledge receipt of a message that will not otherwise evoke an immediate reply (e.g., some 'WANTS' and 'CANS', Section VII). The OBJECT LIST of the acknowledged message is returned as 'DATA' in the 'AM [RESPOND]'.

While involved in side-comments, a pair of PARTICIPANTS may have to exchange 'RESPOND' directives, just as they do with the CHAIRMAN. Should one lose the other, his VCT will simply be 'DELETED' and switched back to the SPEAKER's VOICE stream.

When the CHAIRMAN discovers that a PARTICIPANT has died, he will broadcast a standard 'DO [MEMBERS,<PLIST>]', wherefrom the lost PARTICIPANT'S 'ID' has been deleted. Should the PARTICIPANT come alive at some future time, he will passively call in to the CHAIRMAN to rejoin; if the CHAIR has moved, of course, the 'WONT' he gets will refer him to the new CHAIRMAN.

If a PARTICIPANT finds that the CHAIRMAN has crashed, he will invoke the RECOVERY procedure. RECOVERY utilizes the conference generation mechanism, with intentional multiple callers (see Section VIII); in RECOVERY, however, PLIST PRIORITY (rather than internet ID's) is used to resolve 'DO' collisions. To initiate the process, PARTICIPANTS who lose contact with the CHAIRMAN issue a 'DO [CONFER,RECOVER,<OLD CHAIR ID>]' to those on the PLIST of lower PRIORITY. The following rules then apply:

(1) If the receiver of such a 'DO' is unaware of the crash, he will query the CHAIRMAN with 'DO [RESPOND,<DATA>]'. Should he get a response, the 'CONFER' request will be refused. Otherwise, he will enter the RECOVERY procedure; if the requester is of higher PRIORITY he will accept, else he will broadcast RECOVERY 'DOS' to those of lower PRIORITY than himself.

(2) When an active PARTICIPANT receives a RECOVERY 'DO' from a higher PRIORITY PARTICIPANT, he will accept and dismiss all whom he has acquired (Section VIII).

(3) When a PARTICIPANT thusly acquired receives a RECOVERY 'DQ' from one of higher PRIORITY than his current superior, he will 'AR DQBIT,PCALL,<ID>', and accept the 'DQ'.

(4) If a 'DQ' collision is not resolved by the PLIST, then it is resolved by internet ID'S (the PLIST's at different sites may not agree since the CHAIRMAN may have been updating them at the time of the crash. In this instance, two PARTICIPANTS may each believe that they are of higher PRIORITY than the other; 'DQS' will be exchanged, and each will await a submissive acceptance from the other. When the expected reply fails to arrive, despite successful 'RESPONDS', the PARTICIPANTS will fall back to the internet 'ID' comparison).

(5) When an active PARTICIPANT has received responses to all of his 'DQS', or has deduced that unavailability accounts for those missing, he will begin conducting the recovered conference by broadcasting the usual 'DO MEMBERS,<PLIST>'; however, rules two through four will still apply for a specified time interval to allow for delayed RECOVERY messages from higher PRIORITY PARTICIPANTS. Subsequently, such former PARTICIPANTS will have to passively join the conference, and negotiate for the CHAIR.

(6) As in CHAIR HANDOFF, PARTICIPANTS will inform the new CHAIRMAN of any assigned FUNCTIONS and ungranted 'WANTS'. Note that FUNCTIONS (e.g., SPEAKER) need not be terminated during RECOVERY, so the process is potentially transparent.

The effect of this RECOVERY procedure is to relocate the CHAIRMANSHIP to the highest PRIORITY surviving PARTICIPANT. If the network partitions, a conference will reform in each.

XI Protocol Scenarios

To facilitate its implementation as a finite state machine, the VCS will maintain five logical PATHS:

(1) CALL. The global listen PATH on which calls are received.

(2) ACQUIRE. During conference establishment, the PATH whereon potential PARTICIPANTS are placed until acquired (i.e., after a 'WANT [CONFER]' has been received from them, but before the final 'DO [CONFER]' has been sent).

(3) CONF. The resident PATH for acquired conference PARTICIPANTS.

(4) SIDE. The resident PATH for a side-commenting PARTICIPANT.

(5) QUIT. The PATH on which non-fully acquired PARTICIPANTS are placed when a higher priority call (PCALL) is received during active conference establishment. Upon receipt of his next message, each PARTICIPANT on the 'QUIT' PATH will be appropriately dismissed:

(a) If the message is 'WONT [CONFER]' or 'AU [QUIT]', no dismissal is needed.

(b) If the message is 'AU [CONFER]', the dismissal is 'DO [QUIT,<REASON>]'.

(c) Otherwise, the dismissal is 'WONT [CONFER,<REASON>]'

In command 'ACK' mode, PARTICIPANTS that have been issued a 'DO [QUIT]' are also placed on the 'QUIT' PATH, to await their 'AU [QUIT]' and to keep these replies separate from 'CALLS'.

During initialization, all VCS's will 'DEFINE CALL'. Example sessions:

(1) Passive two-party call. Scenario: Command 'ACK' is the system default. User P1 will call user P2, specifying a 'WANT LIST' ('WLIST') that P2 finds acceptable (it might contain CAPABILITY OBJECTS such as 'VOCODER CVSD', 'DATA-RATE 16 KBITS', 'MESSAGE-LENGTH 1024 BITS', 'C-VERSION NOLOCK'); the 'WLIST' will be returned unaltered as the 'DO LIST' ('DLIST').

P1. TO P2: WANT [CONFER,UNSCHED,P1; <WLIST>]

P2. FROM CALL (P1): WANT [CONFER,UNSCHED,P1; <WLIST>]
    DEFINE ACQUIRE (P1)
 => ACQUIRE HAS [P1]
    TO ACQUIRE: DO [CONFER,UNSCHED,P2; <DLIST>]

P1. FROM CALL (P2): DO [CONFER,UNSCHED,P2; <DLIST>]
    DEFINE CONF (P2)
 => CONF HAS [P2]
    TO CONF: AM [CONFER,UNSCHED,P2; <DLIST>];
             WANT [SPEAKER,P1]

P2. FROM ACQUIRE (P1): (the 'AM' and 'WANT')
    DEFINE CONF (P1)
 => ACQUIRE HAS [] (ACQUIRE PATH is undefined)
 => CONF HAS [P1]
    TO CONF: DO [MEMBERS,P2,P1; SPEAKER,P1]

P1. FROM CONF (P2): (the 'DO')
    TO CONF: AM [MEMBERS,P2,P1; SPEAKER,P1]

P2 is now the CHAIRMAN, and since this is only a two-party call, his VCI will be enabled to 'TALK' to P1, establishing full-duplex VOICE communication (Section IX); there never need be a floor HANDOFF.

(2) Two-party passive call. Scenario: P1 passively calling P2. The system default CAPABILITIES are 'VOCODER CVSD; DATA RATE 16 KBITS; MESSAGE-LENGTH 1024 BITS; C-VERSION NOLOCK, NOACK'. This time, P1 will not supply a 'WLIST' in his 'WANT' call, implicitly requesting the system default parameters. However, P2 can only support up to 12 KBITS CVSD, and prefers 8 KBITS, so negotiation will be triggered.

P1. TO P2: WANT [CONFER,UNSCHED,P1]

```
P2. FROM CALL (P1): (the 'WANT')
    DEFINE ACQUIRE (P1)
    TO ACQUIRE: DO (DATA-RATE)

P1. FROM CALL (P2): (the 'DO')
    TO P2: CAN (DATA-RATE,8 to 24 KBITS)

P2. FROM ACQUIRE (P1): (the 'CAN')
    DEFINE CONF (P1)
    TO CONF: DO (CONFER,UNSCHED,P2; DATA-RATE,8 KBITS;
                 MEMBERS,P2,P1)

P1. FROM CALL (P2): (the 'DO')
    DEFINE CONF (P2)
    TO CONF: WANT (SPEAKER,P1)

P2. FROM CONF (P1): (the 'WANT')
    TO CONF: DO (SPEAKER,P1)
```

(3) Addition of a PARTICIPANT. Scenario: During the conversation established in scenario two, a user P3 tries to call P1. He is refused and referred to P2, who adds him to the conference. When this happens, P2 will close his special VOICE channel to P1 (since this is no longer a two party conversation), and normal floor HANDOFF will ensue. Notice that P3 is requesting system default configuration parameters since he includes no 'WLIST' in his 'WANT'; negotiation is thereby triggered since the current 'DATA-RATE' is the non-default 8 KBITS.

```
P3. TO P1: WANT (CONFER,UNSCHED,P3)

P1. FROM CALL (P3): (the 'WANT')
    TO P1: WONT (CONFER,IN-CONF,P2)

P3. FROM CALL (P1): (the 'WONT')
    TO P2: WANT (CONFER,UNSCHED,P3)

P2. FROM CALL (P3): (the 'WANT')
    DEFINE ACQUIRE (P3)
    TO ACQUIRE: DO (DATA-RATE)

P3. FROM CALL (P3): (the 'DO')
    TO P2: CAN (DATA-RATE,12 to 18 KBITS)

P2. FROM ACQUIRE (P3): (the 'CAN')
    TO ACQUIRE: DO (CONFER,UNSCHED,P2)
    ADD CONF (P3)
    TO CONF: DO (DATA-RATE,12 KBITS;
                 MEMBERS,P2,P1,P3)
```

P2 and P1 will now add P3 to their 'VOICENETS', while P3 will create a new 'VOICENET' including P1 and P2.

(4) Active conference formation. Scenario: System defaults as per scenario two. User P1 actively establishes a scheduled conference assigned the identity 'CID'. At the same time, one of the scheduled PARTICIPANTS (P2) passively calls in to P1. All but P2 implicitly request system defaults.

```
P1. DEFINE ACQUIRE [P2,P3,P4]
    TO ACQUIRE: DO [CONFER,SCHED,CID]

P2. TO P1: WANT [CONFER,SCHED,CID; VOCODER LPC]
    FROM CALL (P1): (the 'DO'--since the 'WANT' just
                     went out, this 'DO' is ignored)

P3,P4. FROM CALL (P1): (the 'DO')
    TO P1: WANT [CONFER,SCHED,CID] (defaults wanted)

P1. FROM ACQUIRE (P2,P3,P4): (the 'WANTS')
    TO ACQUIRE: DO [VOCODER]
          :
    FROM ACQUIRE (P3,P4): CAN [VOCODER,CVSD]
    FROM ACQUIRE (P2): CAN [VOCODER;CVSD,LPC]
    DEFINE CONF [P2,P3,P4]
    TO CONF: DO [CONFER,SCHED,CID; VOCODER,CVSD;
                 MEMBERS,P1,P2,P3,P4]
```

(5) Conference establishment with two active callers. Scenario: A conference containing P1, P2, P3, P4, and P5 is scheduled. P1 is already in communication with P2 (as CHAIRMAN) at the scheduled conference time, and tries to actively call in the other PARTICIPANTS. At the same time, P5 tries to establish the conference; he will eventually become the CHAIRMAN since his internet 'ID' is smaller than P1's. However, P3 will already be negotiating with P1 before P1 or P3 receive P5's call. Results: P1 dismisses P2 and P3; P2 refuses P5, but then receives P1's dismissal (with a referral back to P5). For simplicity, all PARTICIPANTS request the same conference parameters (i.e., identical WLISIS).

```
P1. DEFINE ACQUIRE [P3,P4,P5]
    TO ACQUIRE: DO [CONFER,SCHED,CID]

P3. FROM CALL (P1): (the 'DO')
    TO P1: WANT [CONFER,SCHED,CID; <WLIST>]

P5. DEFINE ACQUIRE [P1,P2,P3,P4]
    TO ACQUIRE: DO [CONFER,SCHED,CID]
```

```
P2. FROM CALL (P5): (the 'DO [CONFER]')
    TO P5: WONT [CONFER,IN-CONF,P1]

P4. FROM CALL (P5): (the 'DO [CONFER]')
    TO P5: WANT [CONFER,SCHED,CID; <WLIST>]

P5. FROM ACQUIRE (P1): (the 'DO [CONFER]'.  Ignored.)
    FROM ACQUIRE (P4): (the 'WANT')
    FROM ACQUIRE (P2): (the 'WONT')
    REMOVE ACQUIRE [P2]

P1. FROM CALL (P5): (the 'DO')
    DEFINE QUIT [P3]
    TO P2: DO [QUIT,PCALL,P5]
    REMOVE CONF [P2]
    TO P5: WANT [CONFER,SCHED,CID; <WLIST>]
           :
    FROM QUIT (P3): WANT [CONFER,SCHED,CID; <WLIST>]
    TO P3: WONT [CONFER,PCALL,P5]
    REMOVE QUIT [P3]

P3. FROM CALL (P1): (the 'WONT [CONFER]')
    FROM CALL (P5): (the 'DO')
    TO P5: WANT [CONFER,SCHED,CID; <WLIST>]

P5. FROM ACQUIRE (P1,P3): (the 'WANTS')
    DEFINE CONF [P1,P3,P4]
    TO CONF: DO [CONFER,SCHED,CID; <DLIST>;
                 MEMBERS,P5,P4,P1,P3]

P1,P3,P4. FROM CALL (P5): (the 'DO')
    DEFINE CONF [P5]

P2. FROM CONF (P1): DO [QUIT,PCALL,P5]
    REMOVE CONF [P1]
    TO P5: WANT [CONFER,SCHED,CID; <WLIST>]

P5. FROM CALL (P2): (the 'WANT')
    TO P2: DO [CONFER,SCHED,CID; <DLIST>]
    ADD CONF [P2]
    TO CONF: DO [MEMBERS,P5,P4,P1,P3,P2]
```

Notice that P1 put P3 on the 'QUIT' PATH prior to dismissing him. This was done since P1 had sent P3 a 'DO [CONFIRM]' but had not yet received a reply, so he was unsure of P3's state (i.e., whether a 'WANT', 'WONT', or 'DO' was inbound). The 'WANT' that came shortly thereafter evoked a 'WONT', terminating communications. Had P3 sent a 'WONT' instead, P1 would not have needed to issue a dismissal.

(6) Conference floor management. Scenario: Users P1, P2, P3, and P4 are in a conference CHAIRED by user P5, with command 'NOJACK, NOLOCK' in effect. P1 has the floor, and P2 makes a floor request; P5 waits for P1 to finish talking, then P2 gets the floor. Shortly after P2 becomes SPEAKER, P1 INTERRUPTS, and P3 makes a side-comment call to P4. Finally, P1 requests the floor, along with a change to 'LOCK' mode (which is already known by P5 to be within everyone's CAPABILITIES). Notice how the 'LOCK' HANDOFF sequence differs from the 'NOLOCK' sequence preceding it.

```
P5. FROM CONF (P2): WANT [SPEAKER,P2]
          :
    FROM CONF (P1): AM [FINISH,0]
    TO CONF: DO [SPEAKER,P2]
          :
    FROM CONF (P1): WANT [INTERRUPT,P1,30 SEC]
    TO P2: WANT [INTERRUPT,P1,30 SEC]
          :
    FROM CONF (P2): DO [INTERRUPT,P1,20 SEC]
    TO CONF: DO [INTERRUPT,P1,20 SEC]
          :
    (20 seconds later, the floor reverts back to P2)

P4. FROM CALL (P3): WANT [CONFER,SIDE-COM,P3; <WLIST>]
    TO P3: DO [CONFER,SIDE-COM,P4; <WLIST>]
    DEFINE SIDE (P3)
          :
    (P3 and P4 talk over a full-duplex VOICE stream)
          :
    TO SIDE: DO [RESPOND,FOO]
    FROM SIDE (P3): AM [RESPOND,FOO]
          :
    TO SIDE: DO [QUIT,SIDE-COM,P4]
    REMOVE SIDE (P3)

P5. FROM CONF (P1): WANT [C-VERSION,LOCK; SPEAKER,P1]
    TO CONF: DO [C-VERSION,ACK,LOCK]
    TO P1: AM [RESPOND,SPEAKER,P1]
    TO P2: DO [FINISH,1 MIN]
          :
    FROM CONF (P1,P2,P3,P4): AM [C-VERSION,ACK,LOCK]
          :
    FROM CONF (P2): AM [FINISH,0]
    TO P2,P3,P4: DO [SPEAKER,P1]
          :
    FROM CONF (P2,P3,P4): AM [SPEAKER,P1]
    TO P1: DO [SPEAKER,P1]
          :
    FROM CONF (P1): AM [SPEAKER,P1]
```

(7) CHAIR HANDOFF. Scenario: P1, P2, and P3 are in conference with CHAIRMAN P4, 'NOACK, NOLOCK' as usual, P2 is SPEAKER. P3 requests and is granted the CHAIR. P3's VCS is not trusting, and will ask P4 to validate the HANDOFF before accepting P3's call. Both P2 and P3 request current conference parameter values in the 'WLISTS' sent to P1, so there will be no renegotiation. When the HANDOFF is complete, P2 will identify himself to P3 as SPEAKER, and P3 will repeat the (ungranted) 'WANT' that begins this scenario.

```
P3. TO CONF: WANT [SPEAKER,P3]

P4. FROM CONF (P1): WANT [MEMBERS,P1,P4,P2,P3]
    FROM CONF (P3): (the 'WANT')
    TO P3: DO [MEMBERS,P1,P4,P2,P3]

P1. FROM CONF (P4): (the 'DO')
    DEFINE ACQUIRE [P2,P3]
    TO ACQUIRE: DO [CONFER,C-HANDOFF,P1]

P2. FROM CALL (P1): (the 'DO')
    TO CONF: ALL [QUIT,C-HANDOFF,P1]
    REMOVE CONF [P4]
    TO P1: WANT [CONFER,C-HANDOFF,P2; <WLIST>]

P3. FROM CALL (P1): (the 'DO')
    TO CONF: WANT [QUIT,C-HANDOFF,P1]

P4. FROM CONF (P2): (the 'ALL [QUIT]')
    REMOVE CONF [P2]
    FROM CONF (P3): (the 'WANT [QUIT]')
    TO P3: DO [QUIT,C-HANDOFF,P1]
    REMOVE CONF [P3]

P3. FROM CONF (P4): (the 'DO [QUIT]')
    REMOVE CONF [P4]
    TO P1: WANT [CONFER,C-HANDOFF,P3; <WLIST>]

P1. FROM ACQUIRE (P2,P3): (the 'WANTS')
    TO ACQUIRE: DO [CONFER,C-HANDOFF,P1; <DLIST>]
    ADD CONF [P2,P3]
    TO CONF: DO [MEMBERS,P1,P4,P2,P3]

P2. FROM CALL (P1): (the 'DOS')
    DEFINE CONF [P1]
    TO CONF: ALL [SPEAKER,P2]

P3. FROM CALL (P1): (the 'DOS')
    DEFINE CONF [P1]
    TO CONF: WANT [SPEAKER,P3]
```

(8) Failure recovery (simple). Users P1, P2, P3, and P4 are in conference in that order of PRIORITY, with P1 as CHAIRMAN and P2 as SPEAKER. 'NOLOCK, NOACK' as before. With 'RESPOND' messages, the PARTICIPANTS discover that P1 has crashed. P2 and P3 therefore initiate RECOVERY, while P4 sits idle since he is at the bottom of the PLIST. All 'WLISTS' specify current conference parameters, so there is no renegotiation.

```
P1,P2,P3,P4. TO CONF: DO [RESPOND,FOO]
                :
     (no response)
                :
P2. DEFINE ACQUIRE [P3,P4]
     TO ACQUIRE: DO [CONFER,RECOVER,P1]

P3. DEFINE ACQUIRE [P4]
     TO ACQUIRE: DO [CONFER,RECOVER,P1]

P4. FROM CALL (P2,P3): (the 'DOS')
     TO P2: WANT [CONFER,RECOVER,P1; <WLIST>]
     TO P3: WONT [CONFER,PCALL,P2]

P3. FROM ACQUIRE (P4): (the 'WONT')
     REMOVE ACQUIRE [P4]
     FROM CALL (P2): (the 'DO [CONFER]')
     TO P2: WANT [CONFER,RECOVER,P1; <WLIST>]

P2. FROM ACQUIRE (P3,P4): (the 'WANTS')
     DEFINE CONF [P3,P4]
     TO CONF: DO [CONFER,RECOVER,P1; <DLIST>;
                  MEMBERS,P2,P3,P4]
```

(9) Failure recovery (complex). Scenario: Users P1, P2, P3, P4, P5, P6, and P7 are in conference with P7 as CHAIRMAN, and P6 the SPEAKER. P1 will crash unexpectedly, and P7 will delete him from the conference. P7 will then fail, and only P3, P4, and P5 will discover this and initiate RECOVERY. P6 gets P5's RECOVERY call, prompting him to check P7's status with a 'DO [RESPOND]'. P5 acquires P6, but then gets a call from P3 and P4; P3 is accepted, P6 is dismissed, and P4 is refused. Shortly thereafter, P6 answers the call from P4, but before he is fully acquired, they both receive P3's call; P4 and P6 swap rejections and submit to P3. P2 finally realizes that P7 has crashed and sends out RECOVERY 'DOS', but they are refused since the conference has already reformed with P3 as CHAIRMAN; P2 passively calls back in. This example is intentionally complex, and illustrates most of the possible RECOVERY exchange sequences. However, even with the

unfortunate (and unlikely) network timing implicit in the following sequence of events, this RECOVERY example could be made to occur with simple interchanges as in the previous scenario if a time delay in answering RECOVERY 'DIE' is introduced (Section XII).

```
P7. TO P1: DO [RESPOND,RUMBLE]
           :
    REMOVE CONF [P1]
    TO CONF: DO [MEMBERS,P7,P2,P3,P4,P5,P6]
           :
    (P7 crashes.)
           :
P3,P4,P5. TO CONF: DO [RESPOND,FOO]
           :
    REMOVE CONF [P7]

P3. DEFINE ACQUIRE [P4,P5,P6]
    TO ACQUIRE: DO [CONFER,RECOVER,P7]

P4. DEFINE ACQUIRE [P5,P6]
    TO ACQUIRE: DO [CONFER,RECOVER,P7]

P5. DEFINE ACQUIRE [P6]
    TO ACQUIRE: DO [CONFER,RECOVER,P7]

P6. FROM CALL (P5): (the 'DO')
    TO CONF: DO [RESPOND,RUMBLEFOO]
           :
    REMOVE CONF [P7]
    TO P5: WANT [CONFER,RECOVER,P7; <WLIST>]

P5. FROM ACQUIRE (P6): (the 'WANT')
    DEFINE CONF [P6]
    TO CONF: DO [CONFER,RECOVER,P7; <DLIST>;
                 MEMBERS,P5,P6]
           :
    FROM CALL (P3,P4): (the 'DO [CONFER]')
    TO P3: WANT [CONFER,RECOVER,P7; <WLIST>]
    TO P4: WANT [CONFER,PCALL,P3]
    TO CONF: DO [QUIT,PCALL,P3]
    REMOVE CONF [P6]

P6. FROM CALL (P5): (the 'DO [CONFER]')
    DEFINE CONF [P5]
    TO CONF: AT [SPEAKER,P6]
           :
    FROM CONF (P5): (the 'DO [QUIT]')
    REMOVE CONF [P5]
    FROM CALL (P4): (the 'DO [CONFER]')
    TO P4: WANT [CONFER,RECOVER,P7; <WLIST>]
```

```
P4. FROM ACQUIRE (P5): (the 'WANT')
    REMOVE ACQUIRE (P5)
    FROM CALL (P3): (the 'DO [CONFER]')
    TO P3: WANT [CONFER,RECOVER,P7; <WLIST>]
    DEFINE QUIT (P6) (ACQUIRE is now UNDEFINED)
          :
    FROM QUIT (P6): (the 'WANT [CONFER]')
    TO QUIT: DONT [CONFER,PCALL,P3]
    REMOVE QUIT (P6)

P6. FROM CALL (P3): (the 'DO [CONFER]')
    TO P3: WANT [CONFER,RECOVER,P7; <WLIST>]
    TO P4: DONT [CONFER,PCALL,P3]

P3. FROM ACQUIRE (P4,P5,P6): (the 'WANTS')
    DEFINE CONF [P4,P5,P6]
    TO CONF: DO [CONFER,RECOVER,P7; <DLIST>;
                 REMBERS,P3,P4,P5,P6]

P6. FROM CALL (P3): (the 'DO')
    DEFINE CONF (P3)
    TO CONF: ALL [SPEAKER,P6]

P2. TO CONF: DO [RESPOND,FOOMUMBLE]
          :
    REMOVE CONF (P7)
    DEFINE ACQUIRE (P3,P4,P5,P6)
    TO ACQUIRE: DO [CONFER,RECOVER,P7]
          :
    FROM ACQUIRE (P3,P4,P5,P6): DONT [CONFER,IN-CONF,P3]
    REMOVE ACQUIRE (P3,P4,P5,P6)
    TO P3: WANT [CONFER,SCHED,CID; <WLIST>]
```

XII Discussion

This section addresses the motivations for some of the fundamental VCS design decisions. This will hopefully spur an analysis of these decisions, and suggestions for improvement.

The VCT VOICE network was chosen to be fully connected since SPEAKER HANDOFF is the most frequent conference FUNCTION; the overhead and delay incurred by reconfiguration of the VOICE network for floor HANDOFF would be excessive. Fully-connected sets are also needed for most side comment/interruption facilities, and will make the best use of broadcast addressing when available, while maintaining compatibility with single-destination protocols.

Fully distributed CONTROL has been proposed as a robust conferencing strategy [4]. Such a strategy unfortunately requires that distributed copies of a very dynamic data base be maintained in an identical state; there is currently no way to achieve this in real-time with adequate reliability, requiring elaborate timeout recovery mechanisms. In fact, the proposed distributed mechanism simply involves centralized CONTROL that moves with every SPEAKER HANDOFF; such extra delay and CONTROL overhead degrades system robustness and performance, and is best avoided. A centralized CONTROL mechanism with movable

CONTROL was therefore designed. This approach is fully flexible, and allowed optimization of certain functions for speed (Floor HANDOFF), others for reliability (CHAIR HANDOFF and RECOVERY). Since CONTROL traffic is purely radial, full-connection was unnecessary, and the CONTROL network was made a star.

There has been too little emphasis in the past on designing communication protocols to be network-independent, which is understandable owing to the varied nature of packet-switched networks. The result, however, has been an imbedding of very network dependent actions in the operation of higher level protocols. For instance, the NVP's requirement that initial connection take place on a special link [1] makes establishing an internet NVP connection very problematical; the GATEWAY can not use the required link. The motivation for the GNI was to avoid similar trappings by isolating network specificity in a separate, non-protocol module. This is practical since the logical functions implemented by higher-level protocols like the VCS are NOT network dependent; they only require that the host network be capable of meeting their implicit performance demands (e.g. low loss, low delay, dynamic broadcast addressing).

Of course, the communication facilities provided by the GNI can be achieved in many ways, but an actual implementation of the GNI as a separate module is recommended. This would render VCS implementations network independent in a way that would be impossible if specific network management routines were imbedded within the VCS implementation

itself. The specification of the GNI may not be complete enough for implementation in any network, nor was it necessarily intended to be so at this stage; some important details may have been excluded, and certain extensions to the interface may be desirable. Fleshing out the GNI specification will only be worthwhile, however, if there is interest in implementing it.

Although VCS/VCI interactions are not a part of the conference CONTROL protocol per se, the VCI interface section was included since the interface between the two modules is important to the implementation of the VCS. Because the signals are not VOICE protocol specific, the VCI could well be an NVP DATA protocol in initial implementations. However, extensions to the NVP at some point would be very desirable. For example, the VOICE protocol could provide a limited feedback facility in addition to the VCS interruption mechanism (Section IX); with it, responses from the listening PARTICIPANTS, such as 'hmm' and 'ah-ha', could be automatically broadcast and heard by all others, without the need for a manual request or CONTROL protocol intervention. When enabled by the 'FDON' signal, 'QUIET' VCI's would allow a controlled amount of speech to be broadcast to 'VOICENET'; an average of one second per minute might be reasonable. Additionally, 'LISTENING' VCI's would be enabled to accept such speech from PARTICIPANTS outside of the 'LISTEN' stream in regulated amounts; these feedback streams would be kept on separate small queues for strategic insertion into the main 'LISTEN' stream, by interleaving or mixing (depending upon available hardware).

Two other useful VOICE protocol capabilities would be multiple SPEAKER's and dynamic flow control. A multiple SPEAKER conference would be most appropriate for arguments and debates; the VCS could set up this structure by attaching an 'ID' list to the SPEAKER OBJECT, instead of a single 'ID' (for generality, the VCI interface already permits a 'LISTEN' stream to contain more than one packet source, but to utilize this feature, multiple speech synthesizers or some sort of digital mixing strategy would be needed). Dynamic flow control could facilitate optimization of intelligibility vis-a-vis current network performance, and is possible if the speech synthesizer sampling rate is under software control. The encoding rate would vary at packet boundaries to reflect network load (send queue length), and the rate chosen for each packet would be included in the VOICE protocol header. This approach is probably most useful for high bandwidth vocoding systems such as CVSD.

There has been recent discussion of a new VOICE transmission technique, imbedded vocoding [5]. This system adds a low bandwidth background VOICE stream (e.g., 2.4 kbit LPC) to a better quality, high bandwidth encoding of the same speech. In this way, a VOICE network can contain multiple vocoder types. To service imbedded vocoding, extensions to the VOICE and CONTROL protocols would be necessary. Although not included in this version of the VCS specification, the modifications should be simple and present no problems.

The user interface section outlines the more important functions and messages available to the user. This is intended only as a

guideline and not as a specification, since the actual appearance of a user interface and the facilities it provides are too much a matter of taste. However, it is important to notice that the full spectrum of user information is available without recourse to special 'user information' CONTROL messages between VCS's; instead, normal CONTROL sequences affecting status implicitly evoke user messages based on their content. In this way for instance, users will be informed appropriately whenever a new '[0 [MEMBERS,<PLIST>]', '[0 [FINISH,<TIME>]', or '[0 [SPEAKER,<ID>]' is received.

The VCS is unusual in that its protocol is defined in terms of a grammar. This structuring facilitates modularization of protocol functions, as well as rendering them easily expandable. The 'language' approach will also aid implementation of the VCS, since familiar table-driven compiler control structures can be utilized.

To adequately service users, a flexible and efficient negotiation mechanism is essential. Previous strategies have focused primarily on simple two-party interactions; when applied to a conferencing system, they inevitably lead to highly iterative sessions of message exchanges which rarely achieve settings of negotiable parameters optimized for the PARTICIPANT set. To sidestep this problem, other conferencing systems [2] [4] do not really provide for dynamic negotiation; parameters are normally set in advance, or by the person who first establishes the conference (this is understandable in the case of NVCP, which was meant to be fully compatible with NVP).

To facilitate fully pipelined negotiation with several PARTICIPANTS simultaneously, a 'CAPABILITIES' philosophy was adopted. Since it is usually impossible to please every member of a bargaining group, it seemed reasonable that the CHAIRMAN should choose what options to set, given full information about the PARTICIPANTS' abilities; it is assumed that he will implement an equitable algorithm that will optimize parameters for the PARTICIPANT set as a whole. With this scheme, flexibility is only limited by the number of decision (e.g., floor HANDOFF) algorithms that are available at user request.

The negotiation procedure also minimizes CONTROL traffic; requiring that the CHAIRMAN never supply a directive without being sure that the PARTICIPANT can follow it avoids all sorts of otherwise common: do-this--I-can't--will-you-do-this?--no-but-I-can-do-this, etc. When fully reliable (e.g., TCP) CONTROL connections are available, further savings may be realized since command acknowledgments are not necessary for VCS operation.

The main design criteria for the conference establishment procedure were that:

(1) It be capable of fully automatic, pipelined operation.

(2) It not require that the conference be scheduled in advance, or that users cooperate carefully in setting it up.

(3) It serve as the basis for the CHAIR HANDOFF, RECOVERY, and subconference formation mechanisms.

The pipelined, simultaneous nature of this fundamental mechanism renders it very fast and efficient, as well difficult to analyze for faults. However, should any very sticky deadlocks arise, it would be a simple matter to unpipeline the defective aspect of the protocol by requiring interlocking via command acknowledgment. Interlocking of FUNCTION HANDOFFs ('LOCK') may also be necessary as a default in networks with large, very variable delays to avoid audible timing glitches, but conferencing performance would be degraded in such an environment anyway.

A multiple initiative RECOVERY strategy was adopted to promote rapid conference restoration. When the CHAIRMAN crashes, PARTICIPANTS can not always know the extent of the network failure; by becoming active, they avoid waiting in vain for external calls. Of course, if network timing is unfavorable, several shifts of CONTROL may thereby occur (Section XI). A possible way to reduce such overhead at the expense of extra delay is to require that all VCS's wait for a standard timeout (after broadcasting RECOVERY 'DOS') before answering RECOVERY calls; the network would thusly clear of these calls, so PARTICIPANTS could examine them all and submit directly to the one of highest PRIORITY.

It is possible to trim the CHAIR HANDOFF and RECOVERY mechanisms if the initial conference negotiation exchanges are not used; the new CHAIRMAN need simply send 'DO (CONFER,<REASON>,<NEW CHAIR ID>; <DO LIST>)', wherein the 'DO LIST' specifies the parameter values currently

in effect. This effectively short-circuits most of the conference generation message sequence, and allows a PARTICIPANT to be acquired with at most a simple 'AH' reply. The advantage of this alternate scheme is greater HANDOFF robustness (with smaller delay) owing to simpler exchanges. However, it is less general, and leads to extra exchanges if a PARTICIPANT loses a CAPABILITY required by the HO LIST just prior to the HANDOFF.

Beyond this point, designing the rest of the VCS was simply a matter of identifying FUNCTIONS desirable for conducting a conference. To add a FUNCTION, its relevant OBJECT and ADJECTIVES need only be defined and associated with VCT and user interface operations; the protocol grammar and negotiation mechanism determine how VCS's will manipulate the OBJECT. Currently being designed is a hierarchical conferencing facility that would be useful in military environments.

REFERENCES

1. D. Cohen, "Specifications for the Network Voice Protocol", Arpa Order No. 2223, NSC Note No. 68, Information Sciences Institute, University of Southern California, Marina Del Rey, Calif., March, 1976.

2. D. Cohen, "The Network Voice Conference Protocol (NVCP)", Information Sciences Institute, University of Southern California, Marina Del Rey, Calif., February, 1976.

3. Y. Dalal, "Broadcast Protocols in Packet Switched Computer Networks", ISL Technical Note No. 128, Stanford University, Stanford, Calif., April, 1977 (Ph.D. Thesis).

4. J. W. Forgie, and C. W. McElwain, "A Proposal for SATNET Voice Conferencing", NSC Note No. 104, PSCN No. 48, Massachusetts Institute of Technology, Lincoln Laboratory, Lexington, Mass., January, 1977.

5. C. K. Un, and D. T. Magill, "Development of an Imbedded Vocoder", Internal Stanford Research Institute Memo, Menlo Park, Calif., August, 1976.