Network Working Group M. Spencer
Internet-Draft Digium, Inc.
Expires: July 5, 2005 F. Miller
Cornfed Systems, LLC
January 2005

Inter-Asterisk EXchange (IAX) Version 2
draft-mspencer-iax2-01

Status of this Memo

Abstract

The Inter-Asterisk EXchange (IAX) protocol provides control and transmission of streaming media over Internet Protocol (IP) networks. IAX can be used with any type of streaming media including video but is targeted primarily at the control of IP voice calls.

The protocol described in the document is actually Version 2 of the IAX protocol, commonly referred to as IAX2. IAX2 is intended to replace the original IAX Version 1 protocol and as such, this document simply refers to IAX. It is understood that subsequent references to IAX refer to Version 2.

Table of Contents

1. Introduction

The Inter-Asterisk EXchange (IAX) protocol provides control and transmission of streaming media over Internet Protocol (IP) networks. IAX can be used with any type of streaming media including video but is targeted primarily at the control of IP voice calls.

The design goals for IAX derive from experience with existing Voice-over-IP (VoIP) protocols such as the Session Initiation Protocol (SIP) and the Media Gateway Control Protocol (MGCP) for control and the Real-Time Transfer Protocol (RTP) for streaming media transmission.

The primary design goals for the IAX protocol are: 1) minimize bandwidth usage for both control and media with specific emphasis on individual voice calls and 2) provide native support for Network Address Translation (NAT) transparency

2. IAX Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119

3. Protocol Overview

IAX is a peer-to-peer media and signaling protocol. Peer-to-Peer means that the endpoints maintain state machines associated with the protocol operations. The signaling component of the IAX protocol more analgous to the Session Initiation Protocol (SIP) than to the Media Gateway Control Protocol (MGCP), which is a Master-Slave call control protocol. With respect to media, sequencing and timing information is included in IAX frames. The transport of media does not use the Real-Time Transport Protocol (RTP).

The basic design approach for IAX multiplexes signaling and multiple media streams over a single User Datagram Protocol (UDP) association between two Internet hosts. In this facet of its design, it is actually two protocols in one, a protocol for signaling sessions and a protocol for transporting the actual media streams themselves. This approach differs from the overall architecuture of other IETF-based protocols that separate the control (MGCP and SIP) and media stream (RTP/RTCP) components using different protocols. Because signaling and media share the same UDP port number, IAX does not suffer from the NAT traversal problems associated with SIP.

Figure 1 illustrates the basic relationship between two Internet hosts. Each host uses the ``well-known'' UDP port 4569 to communicate all Internet packets. IAX then uses a 15-bit Call Number to multiplex multiple streams over the UDP port number.

```
    +-----------------+                    +------------------+
    |  Call           |                    |            Call  |
    | Number          |                    |           Number |
    |  +-+            |                    |            +-+   |
    |  | |---+   UDP  |                    |  UDP   +---| |   |
    |  +-+    \  Port |                    | Port  /    +-+   |
    |  +-+     \ 4569 |     +---------+    | 4569 /     +-+   |
    |  | |---+  \+-+  |     |   IP    |    |  +-+/  +---| |   |
    |  +-+    \_ | |<------>| Network |<----->| | _/    +-+   |
    |  ...     _ +-+  |     +---------+    |  +-+ _     ...   |
    |  +-+    /       |                    |       \    +-+   |
    |  | |---+        |                    |        +---| |   |
    |  +-+            |                    |            +-+   |
    +-----------------+                    +------------------+
```

Figure 1: Multiplexing Muliple Streams Using a Single UDP Port

The value of zero is a special call number reserved on each host. When attempting to setup a call, the Call Number of the destination host is not yet known. A zero destination call number is used in this situation.

IAX is a binary protocol. This design choice was made for bandwidth efficiency. Further, the protocol is specifically optimized to make very efficient use of bandwidth for individual voice calls. The bandwidth efficiency for other stream types is sacrificed for the sake of individual voice calls.

3.1 Call Setup

Figure 2 illustrates the basic message flow used to setup a voice call. In this example, Host A initiates the call by sending a NEW message to Host B. Host B immediately sends back an ACCEPT message, indicating to Host A that it has received the request and is beginning to service it. Host A sends an ACK message to Host B indicating receipt of the ACCEPT message. Once Host B begins to ring the phone on its side, it sends back to Host A a RINGING message. Host A sends an ACK message back to Host B indicating receipt of the RINGING message. Finally, when the phone is picked up, Host B sends an ANSWER message to Host A and the call setup is complete. At this point full-duplex voice passes between Host A and Host B.

```
Host A          Host B
  |               |
  |      NEW      |
  |-------------->|
  |     ACCEPT    |
  |<--------------|
  |      ACK      |
  |-------------->|
  |               |
  |    RINGING    |
  |<--------------|
  |      ACK      |
  |-------------->|
  |               |
  |     ANSWER    |
  |<--------------|
  |      ACK      |
  |-------------->|
  |               |
```

Figure 2: A Typical Call Setup Scenario

3.2 Call Teardown

Figure 3 illustrates the message flow for a voice call teardown. In this example, Host A initiates the call teardown by sending a HANGUP message to Host B. Host B is expected to immediately send back an ACK message indicating the receipt of the teardown request and that the call has been torn down on the Host B side.

```
                      Host A          Host B
                        |                |
                        |     HANGUP     |
                        |--------------->|
                        |      ACK       |
                        |<---------------|
                        |                |
```

Figure 3: A Typical Call Teardown Scenario

3.3 Media Flow

Figure 4 illustrates a simple, one-way IAX Media Flow. For a nominal voice call, there would be two of these flows, one flowing in either direction. Each flow is comprised mostly of IAX Mini Frames (labeled M in Figure 4) which contain a simple 4-byte header that targets bandwidth efficiency. The flow is supplemented by periodic Full Frames (labeled F in Figure 4) that include synchronization information.

```
        +---+     +---+ +---+ +---+ +---+     +---+ +---+ +---+
Host A  | M | ... | M | | M | | F | | M | ... | M | | M | | F |  Host B
        +---+     +---+ +---+ +---+ +---+     +---+ +---+ +---+
          ---------------- Frames Transmitted ------------->
```

Figure 4: A Typical Media Flow Scenario

Note that the Mini Frames are sent unreliably. That is, the Full Frames that are part of the stream are acknowledged by Host B but the Mini Frames are not.

4. Frame Definitions

IAX messages are called frames. There are several basic frame types and each of these frame types is described in detail in this section. There a number of common fields within these frames that are explained here for consistency and brevity.

An F bit is used in indicate whether a frame is a Full Frame or not. A value of 1 in this field indicates the frame is a Full Frame and a value 0 indicates the frame is something other than a Full Frame.

A Call Number is a 15-bit unsigned integer that is used to track a media stream endpoint on a host. The value zero is a special Call Number that indicates the Call Number is unknown. A phone call actually has two Call Numbers associated with it, one for either direction.

A Timestamp can be a full 32- or an abridged 16-bit value. In the case of a 16-bit field, the value is actually the lower 16 bits of a full 32-bit timestamp that is maintained by the endpoint host.

4.1 Full Frame

A Full Frame can be used to send signaling, audio, or video information reliably. Full Frames are the only frame type are transmitted reliably. This means that the recipient host must return some type of message back to the sending host immediately upon reception. In some cases, the protocol may require a particular message be sent back immediately but if not the recipient must send an explicit acknowledgement. Figure 2 shows both cases. After receiving a NEW message, the recipient host must return an ACCEPT message immediately. In this case, no explicit ACK is required. Later, when a RINGING message is sent to the caller, Host A must send back an explicit ACK message since the IAX protocol does not require any other message to be returned at that time.

```
                     1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|     Source Call Number      |R|  Destination Call Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Timestamp                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     OSeqno    |    ISeqno     | Frame Type   |C|  Subclass    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                              Data                             |
|                                                               |
```

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Full Frame Binary Format

Figure 5 illistrates the binary format of a Full Frame. Table 1 describes each of the fields in Figure 5. The R bit is set to 1 if the frame is being retransmitted. Retransmission occurs after some timeout period and retransmissions are retried several times, depending on the context. The outbound stream sequence number, OSeqno, always begins with 0 and increases monotonically. OSeqno is used by the recipient to track the ordering of media frames. ISeqno is similar to OSeqno, except that it is used to track the ordering of inbound media frames. Specifically, ISeqno is the next expected inbound stream sequence number for the incoming media frames. Frame Type identifes the class of message as defined in Table 2. The C bit determines how the Subclass value should be interpreted. If C is set to 1, the Subclass value is interpreted as a power of two. If C is set to 0, Subclass is interpreted as a simple 7-bit unsigned integer value.

Table 1: Full Frame Field Descriptions

FIELD	DESCRIPTION
F	Set to the value 1 indicating that this is a Full Frame
Source Call Number	Call number of the transmitting side of the Full Frame
R	Set to the value 1 if this frame is being retransmitted and the value 0 for the initial transmission
Destination Call Number	Call number of the receiving side of the Full Frame
Timestamp	Full 32-bit timestamp
OSeqno	Outbound stream sequence number
ISeqno	Inbound stream sequence number
Frame Type	Frame type
C	Subclass value format

Subclass	Subclass

Table 2 lists the values defined for the Frame Type field. This table contains the value associated with a Full Frame type, its description, a brief description of how the Subclass field in the Full Frame are used for the Full Frame type, and a brief description of the format of the data field. If a cell in Table 2 is left blank, the correspoding field in the Full Frame is not used.

Table 2: Frame Type Values

TYPE	DESCRIPTION	SUBCLASS DESCRIPTION	DATA DESCRIPTION
0x01	DTMF	0-9, A-D, *, #	
0x02	Voice Data	Audio Compression Format	Raw Voice Data
0x03	Video	Video Compression Format	Raw Video Data
0x04	Control	See Control Frame Types	
0x05	Null		
0x06	IAX Control	See IAX Protocol Messages	Information Elements
0x07	Text		Raw Text
0x08	Image	Image Compression Format	Raw Image Data
0x09	HTML	See HTML Frame Types	Message Specific

When a Full Frame is used to transport DTMF digits, the Subclass contains the actual digit being transported. For voice, video, or image streams, the Subclass field specifies the compression format and the data portion of the Full Frame contains a packet of raw voice or video data. The compression formats for voice are given in Table 3, for video in Table 4, and for images in Table 5.

Table 3: Voice Data Subclass Audio Compression Format Values

SUBCLASS	DESCRIPTION	LENGTH CALCULATION
0x0001	G.723.1	4, 20, and 24 byte frames of 240 samples
0x0002	GSM Full Rate	33 byte chunks of 160 samples or 65 byte chunks of 320 samples
0x0004	G.711 mu-law	1 byte per sample
0x0008	G.711 a-law	1 byte per sample
0x0010	MP3 (deprecated)	
0x0020	IMA ADPCM	1 byte per 2 samples
0x0040	16-bit linear little-endian	2 bytes per sample
0x0080	LPC10	Variable size frame of 172 samples
0x0100	G.729	20 bytes chunks of 172 samples
0x0200	Speex	Variable
0x0400	ILBC	50 bytes per 240 samples

Table 4: Video Subclass Video Compression Format Values

SUBCLASS	DESCRIPTION
0x00010000	JPEG
0x00020000	PNG
0x00040000	H.261
0x00080000	H.263

Table 5: Image Subclass Image Format Values

SUBCLASS	DESCRIPTION

There are two types of control information that are passed between peers using Full Frames, Control Frames and IAX Control Frames. Control Frames provide session control, i.e. they refer to control of the devices connected to the IAX endpoint. IAX Control Frames provide IAX protocol specific endpoint management, i.e. they are used to manage IAX protocol interactions that are generally independent of the type of endpoints. Table 6 lists the Full Frame Subclass values for Control Frames and Table 7 lists the Subclass values for IAX Control Frames.

Table 6: Control Frame Subclass Values

SUBCLASS	DESCRIPTION
0x01	Hangup
0x02	Ring
0x03	Ringing (ringback)
0x04	Answer
0x05	Busy Condition
0x08	Congestion Condition
0x09	Flash Hook
0x0a	Wink
0x0b	Option
0x0c	Key Radio
0x0d	Unkey Radio
0x0e	Call Progress

Table 7: IAX Control Frame Subclass Values

SUBCLASS	MNEMONIC	DESCRIPTION
0x01	NEW	Initiate a new call

0x02	PING	Ping request
0x03	Reserved	
0x04	ACK	Acknowledgement
0x05	HANGUP	Initiate call teardown
0x06	REJECT	Reject
0x07	ACCEPT	Accepted
0x08	AUTHREQ	Authentication request
0x09	AUTHREP	Authentication reply
0x0a	INVAL	Invalid call
0x0b	LAGRQ	Lag request
0x0c	LAGRP	Lag reply
0x0d	REGREQ	Registration request
0x0e	REGAUTH	Registration authenticate
0x0f	REGACK	Registration acknowledgement
0x10	REGREJ	Registration reject
0x11	REGREL	Registration release
0x12	VNAK	Video/Voice retransmit request
0x13	DPREQ	Dialplan request
0x14	DPREP	Dialplan response
0x15	DIAL	Dial
0x16	TXREQ	Transfer request
0x17	TXCNT	Transfer connect
0x18	TXACC	Transfer accept
0x19	TXREADY	Transfer ready

0x1a	TXREL	Transfer release
0x1b	TXREJ	Transfer reject
0x1c	QUELCH	Halt audio/video transmission
0x1d	UNQUELCH	Resume audio/video transmission
0x1e	POKE	Poke request
0x1f	PAGE	Paging call description
0x20	MWI	Message waiting indication
0x21	UNSUPPORT	Unsupported message
0x22	TRANSFER	Remote transfer request

4.2 Mini Frame

A Mini Frame is used to send media with a minimal protocol overhead. Figure 6 illustrates the binary format of a Mini Frame and Table 7 describes the fields present in Figure 6.

```
                     1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|     Source Call Number      |          Timestamp            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                             Data                              |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Mini Frame Binary Format

The Timestamp in the Mini Frame is truncated. The client generally maintains a 32-bit full timestamp. When sending Mini Frames, the low-order 16 bits of the timestamp are sent in the Timestamp field. When the 16-bit timestamp wraps around, a Full Frame is sent to allow the other end to synchronize its full 32-bit timestamp counter.

Table 7: Mini Frame Field Descriptions

FIELD	DESCRIPTION
F	Set to the value 0 indicating that this is not a Full Frame
Source Call Number	Call number of the transmitting side of the Mini Frame
Timestamp	16-bit timestamp

4.3 Information Element

```
                     1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      IE       |  Data Length  |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
|                              Data                             |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Information Element Binary Format

Table 8: Information Element Field Descriptions

FIELD	DESCRIPTION
IE	
Data Length	

5. Protocol State Machines

This section describes a series of state machines that are implemented by the IAX endpoints.

5.1 Reliable Transmission of Full Frames

The most basic state machine is associated with the reliable transmission of Full Frames. Full Frames are the only frame type that is transmitted reliably.

Figure 8 illustrates the general state machine used by the sender of a Full Frame to implement reliablity. Some State in the diagram is used to represent any state in any subsequent state diagram that Full Frames are sent from.

```
+--------+                 +--------+                 +----------+
|        | send:Full Frame | Full   | recv:ACK        |          |
| Some   |---------------->| Frame  |---------------->| Another  |
| State  |                 | Sent   | or Full Frame   |  State   |
+--------+                 +--------+                 +----------+
                             ^   |
                             |   | Timeout
                             +---+ send:Full Frame
```

Figure 8: Reliable Transmission of Full Frames

Once an endpoint has transmitted the Full Frame, it enters a logical Full Frame Sent state and sets a timer. If an ACK or another Full Frame is received from the intended recepient of the original Full Frame before a timeout occurs, the original Full Frame is considered delivered and the endpoint moves to Another State.

This state diagram is logically present in all of the remaining state diagrams. Anytime a Full Frame is transmitted in a subsequent diagrams, there this mechanism is present in the subsequent states and transitions. Its presentation here as a separate state diagram is provided for clarity.

5.1.1 Estimating Round-Trip Delay

The Estimated Round-Trip Delay (ERTD) is a value that is continuously maintained between two IAX endpoints. The ERTD estimates the latency required to send a Full Frame and receive an ACK from the other endpoint.

The ERTD is updated each time a Full Frame is sent. When the Full Frame is either transmitted or retransmitted, the sending endpoint places a timestamp in the Timestamp field of the Full Frame. When either an ACK or another Full Frame is received that serves as an ACK, the sending endpoint records the time that ACK or secondary Full Frame is received. The ERTD is then updated as the difference between the recorded time of reception and the timestamp contained in the received ACK or secondary Full Frame.

5.1.2 Exponential Timer Backoff

The timer durations associated with each transmission and retransmission of a Full Frame follow an exponential backoff. When a Full Frame is first transmitted, the initial timer value is chosen as twice the current ERTD value. Every retransmission causes the timer last timer value to be doubled.

For example, assume that an initial Full Frame transmission occurs and the original timer value is taken from the current ERTD value of 50 milliseconds. If the Full Frame requires retransmission, the next timer value would be 100 milliseconds. The next retransmission would set a timer value of 200 milliseconds, and so on. This backoff continues until the maximum number of retries is reached.

5.1.3 Maximum Retries

The Maximum Number of Retries (MNR) is has two factors associated with it. There is a total number of retries and there is a maximum duration that retries can occur within. Whichever value is reached first causes retries to stop.

The total number of retries is 10 and the maximum duration is set at 60 seconds. If 10 retries with exponential backoff occur in less than 60 seconds, retrying is stopped. If however, less than 10 retries occurs after 60 seconds, retrying is also stopped.

5.2 Heartbeats

Two endpoint interactions form heartbeats functions. The first uses a PING message between two endpoints that have a connection established. The second uses a POKE message to determine whether a endpoint will respond when no connection exists between two endpoints. Each of these interactions has a similar state diagram.

The first state machine describes a PING message. When two endpoints are connected but no Full Frames have been exchanged for 15 seconds a PING is sent to make sure the other side is still connected.

Figure 9 illustrates the PING state machine. Since PING is sent in a Full Frame, the state machine in Figure 9 uses the retransmission described in Figure 8. When the sender send the initial PING Full Frame, it starts a timer and enters the Ping Sent state. If the timer expires, the PING is resent. If the PING is retransmitted the maximum number of times, the sender will teardown the connection.

```
            +-----------+  send:PING   +-----------+
            |           |------------->|           |---+
            | Connected |              | Ping Sent |   | Timeout
            |           |<-------------|           |<--+ send:PING
            +-----------+   recv:ACK   +-----------+
```

Figure 9: PING State Machine

The second state machine describes a POKE request. When one endpoint wants to probe another, it can send a POKE message to determine if the endpoint will respond.

Figure 10 illustrates the POKE state machine. Since POKE is sent in a Full Frame, the state machine in Figure 10 uses the retransmission described in Figure 8. When the sender send the initial POKE Full Frame, it starts a timer and enters the Poke Sent state. If the timer expires, the POKE is resent. If the POKE is retransmitted the maximum number of times, the sender assumes the target endpoint will not respond.

```
            +-----------+  send:POKE   +-----------+
            |           |------------->|           |---+
            |    Some   |              | Poke Sent |   | Timeout
            |   State   |<-------------|           |<--+ send:POKE
            +-----------+   recv:ACK   +-----------+
```

Figure 10: POKE State Machine

5.3 Call Setup Client Side

Figure 11 illustrates the client side of a call setup. By client side, we mean the side of the call that initiates the call setup. This state machine includes the timeouts associated with the sending of Full Frames illustrated in Figure 8.

```
                       recv:REJECT
          +--------------------------------+
         /             send:ACK             \
        /    +---------+                     +---------+
       /     |         |       Timeout       |         |
```

```
         /      | Final  |<--------------------|  Auth  |
        /       |        |     send:HANGUP     |        |
       /        +--------+                     +--------+
      /         /      |\                      /|      \
     /         /          \                    /         \
    |         /    Timeout \                  /           \ recv:ACCEPT
    |        /  send:HANGUP \                /             \ send:ACK
    |       /                \              /               \
    |      /recv:ACK          \            / recv:AUTHREQ    \
    |     /or Timeout          \          / send:AUTHREP      \
    |    /                      \        /                     \
   \|  |/                        \      /                       \|
 +--------+                     +--------+                   +--------+
 |        |       send:NEW      | Accept |     recv:ACCEPT   |        |
 |  Null  |-------------------->|  Wait  |------------------>|Accepted|
 |        |                     |        |      send:ACK     |        |
 +--------+                     +--------+                   +--------+
   ^  ^  ^                          |                        /   /   /
   |  |  |        recv:REJECT       |                       /   /   /
   |  |  +--------------------------+                      /   /   /
   |  |         send:ACK                                 /   /   /
   |  |                       recv:REJECT               /   /   /
   |  +-------------------------------------------------+   /   /
   |                          send:ACK                 /   /   /
   |                                                  /   /
   |                                  recv:ANSWER    /   /
   |                          +-------------------------+   /recv:
   |                         /                              |RINGING
   |                        /          send:ACK            / send:ACK
    \                      /                              /
     \                   |/                             |/
      \        +---------+                     +--------+
       \       |         |     recv:ANSWER     |        |
        \      |Connected|<--------------------| Ringing|
         \     |         |       send:ACK      |        |
          \    +---------+                     +--------+
           \             recv:REJECT          /
            +--------------------------------+
                          send:ACK
```

Figure 11: Client Side Call Setup State Machine

The client begins a call setup by sending a NEW Full Frame to the server side of the call. The client moves from the Null to the Accept Wait state when this happens. The sending of the NEW Full Frame triggers the state machine associated with the sending of a Full Frame as shown in Figure 8. The NEW Full Frame is retransmitted some number of times. Only when the final Timeout occurs does the client state machine move to the Final State shown in Figure 11 and send a HANGUP Full Frame to the server (that has not responded). When an ACK is received for the HANGUP Full Frame or a Timeout occurs, the client returns to the Null state.

Three events can cause a transition when the client is in the Accept Wait state. In the nominal case, an ACCEPT Full Frame is received that indicates the call setup request has been accepted and the server side is ringing the phone. The client sends an ACK to the ACCEPT Full Frame, moves to the Accepted state and waits for the call to be answered on the server side. The server side can also ask for authentication be responding with a AUTHREQ Full Frame. In this case, an AUTHREP is generated and the and the client moves to the Auth state. The server side can also respond with a REJECT Full Frame to indicate that the call cannot be completed. In this case, the client responds with the ACK of the REJECT Full Frame and returns to the Null state.

When the client is in the Auth state, one of three events can cause a transition. The server may return a REJECT Full Frame. The client then sends an ACK to the server and returns to the Null state. The client may Timeout waiting for a response from the server. The client would send a HANGUP to the server and enter the Final state. Finally, the client can receive an ACCEPT Full Frame from the server. The client then returns an ACK for the Full Frame and enters the Accept state to wait for the call to be answered.

When the client has reached the Accepted state, three events can cause a transition to another state. The server may return a REJECT Full Frame. The client sends an ACK to the server and returns to the Null state. The client may receive a RINGING Full Frame from the server. The client returns an ACK for the RINGING Full Frame and moves to the Ringing state. Finally, the client may receive an ANSWER Full Frame from the server. In this case, the client returns an ACK for the ANSWER Full Frame and moves to the Connected state.

Only two transitions can occur when the client is in the Ringing state. The server may return a REJECT Full Frame. The client would then return an ACK and move to the Null state. The server can also return an ANSWER Full Frame. The client returns an ACK for the ANSWER Full Frame and moves to the Connected state. Note that there are no Timeouts specified for the Accepted and Ringing states.

5.4 Call Setup Server Side

Figure 12 illustrates the server side of a call setup. By server side, we mean the side of the call that receives the initial call setup request. Figure 12 is divided into two parts, the state machine prior to a call setup request being accepted by the server and the state machine after the call setup request has been accepted by the server. The first part of the state machine has two paths, one that requires authentication of the client making the request and one that does not require authentication. The second part of the state machine also has two paths. The first answers the call without returning a RINGING Full Frame to the client a the second that returns a RINGING Full Frame prior to answering the call.

```
                                +------------+
                                |            |
                +--------------->|    Null    |---+  recv:NEW
      recv:ACK /                |            |    \| send:AUTHREQ
              /                 +------------+     +------------+
             /       recv:NEW  /                   |            |
            |    send:ACCEPT  /                    |    Auth    |
            |               |/   to Reject Sent <--|            |
            |   +------------+        recv:AUTHREP +------------+
            |   |            |        send:REJECT        | recv:AUTHREP
            |   |   No Auth  |                           v send:ACCEPT
            |   |            |                     +------------+
            |   +------------+                     |    Auth    |
            |                 \                    |   Reply    |
            |         recv:ACK \|                  |    Rcvd    |
     +------------+             +------------+     +------------+
     |   Reject   | send:REJECT |            |    /
     |    Sent    |<------------| Accept Rcvd|<--+ recv:ACK
     |            |             |            |
     +------------+             +------------+
            ^     send:RINGING /       |
            |                |/        |
            |   +------------+         |
            |   |   Ringing  |         |
            |   |    Sent    |         |
            |   |            |         |
            |   +------------+         |
            |          | recv:ACK      | send:ANSWER
            |          v               |
            |   +------------+         |
            |   |   Ringing  |         |
            +---|    Rcvd    |         |
    send:REJECT |            |         |
                +------------+         |
                              \        |
                   send:ANSWER \|      v
                                +------------+
                                |   Answer   |
                                |    Sent    |
                                |            |
```

```
+------------+
      |
      v recv: ACK
+------------+
|            |
|  Connected |
|            |
+------------+
```

Figure 12: Server Side Call Setup State Machine

A call setup request is initiated by a client when the server receives a NEW Full Frame. The server can either return an ACCEPT Full Frame immediately or send an AUTHREQ if authentication of the client is required. If an ACCEPT Full Frame is sent, the server moves to the No Auth state. If an AUTHREQ is send, the server moves to the Auth state. When the server receives an AUTHREP for authentication of the client, it sends an ACCEPT and moves to the Auth Reply Rcvd state. In both the No Auth and the Auth Reply Rcvd states, the server transitions to the Accept Rcvd state when the Full Frame ACK is received.

While in the Accept Rcvd state, the server can send either a RINGING or an ANSWER Full Frame to the client. If a RINGING Full Frame is sent, the server moves to the Ringing Sent state. When an ACK for the RINGING Full Frame is received, the server then moves to the Ringing Rcvd state. The server can then send an ANSWER Full Frame to the client. Whether the server sends an ANSWER Full Frame while in the Accept Rcvd or the Ringing Rcvd states, it moves to the Answer Sent state to await an ACK. Once the ACK is received, the server moves to the Connected state.

The server can reject the call in any of the Auth, Accept Rcvd, or Ringing Rcvd states. When the call is rejected, the server sends a REJECT Full Frame to the client and moves to the Reject Sent state. The server returns to the Null state when the ACK for the REJECT Full Frame is received.

5.5 Call Teardown Client Side

5.6 Call Teardown Server Side