VOTER Protocol – Voice Observing Time Extension (for) Radio

Jim Dixon, WB6NIL

Version 0.4 – Feb. 19, 2011

VOTER is a completely conectionless UDP-based protocol (using IANA-assigned well-known port 667, originally assigned to me for a project I was doing for the United States Federal Election Commission, which, of couse, deals mainly with an entirely different kind of voting). It provides for transmission of audio signals from remote radio receivers, along with ultra-precise (GPS-based) timing information and signal quality, allowing synchronization of multiple signals at the "head" end (hereinafter referred to as the "host") of the system and selection of which signal to use based upon the signal quality information and the consistency thereof. Audio and timing information may also optionally be sent back the remote location (hereinafter referred to as the "client") by the host to facilitate a multiple-site simulcast transmission system.

Security is provided by a Challenge/Response authentication technique using CRC-32 based digest information. Although it is by no means a terribly secure method (it is not even close to crytographically-strong), it at least it provides somwhat of a deturrent for those attempting to circumvent the security.

Both the client and the host use the same exact messages as follows:

The VOTER protocol packet consists of a 24 octet header and optionally a payload, which currently can either be 160 samples (20ms) of Mu-law encoded audio, or GPS location and elevation information (sent once per second).

The VOTER protocol packet header consists of 24 octets as follows:

All date and time information is in GMT.

All multi-byte numeric fields are sent standard network-order (most significant byte first).

Octets 0-3: Current accurate time in whole seconds, 32 bits (network order).
Octets 4-7: Current accurate fractional time in nanoseconds, 32 bits (network order).
Octets 8-17: Authentication challenge, ASCII string, null-terminated (up to 9 chars long + null).
Octets 18-21: Autheication reponse (digest), 32 bits (network order)..
Octets 22-23: Payload type, 16 bits (network order).

Payload types are as follows (header octets 22-23):

Playload type 0 – Authentication plus flags only if sent by host. Client does not send payload.

Octet 24: Flag bits, as follows (specified in bit value):

1 – Flat Audio (NOT de-emphasized) if flag set, Processed audio (de-emphasized) if not set
2 - Send audio always, regardless of whether or not signal is detected if set, do not send audio unless signal is detected..if not set.
4 – Do Not filter Sub-Audible tones from audio stream, if set. Filter Sub-Audible (<300Hz) tones from audio stream if set.
(8-128) - RFU

Payload type 1 – RSSI information + audio. Octet 24 is RSSI value 0-255 and Octets 25-184 are 160 samples of Mu-law encoded audio (20ms @ 8K samples/sec). Packets of this payload type are sent only when the receiver is receiving a valid signal (or the host is indication that a signal is to be transmitted).

Payload type 2 – GPS Information as follows:

Octets 24-32: Longitude value, in the format XXXX.DDY (eg 4807.038N), ASCII string, null-terminated.
Octets 33-42: Latitude value, in the format XXXXX.DDY (eg 01131.000E), ASCII string, null-terminated.
Octets 43-48: Elevation value in Meters (eg 545.4), ASCII string, null-terminated.

By definition, a digest value of 0 is an indication that the entity generating it has not received a validly recognizable digest from its peer. Therefore, a challenge must not be generated that would result in a digest value of 0.

The intended methodology for the use of the protocol is as follows:

Then the client starts operating, it starts perodically sending VOTER packets to the host containing a payload value of 0, the client's challenge string and an authentication response (digest) of 0 (to indicate that it has not as of yet gotten a valid response from the host). When such a packet is received by the host, it responds with a packet containing a payload value of 0, the hosts's challenge string, an authentication response (digest) based upon the calculated CRC-32 value of the challenge string it received from the client and the host's password., and option flags. If the client approves of the host's repsonse, it may then start sending packets with a payload type of 1, the client's challenge string, and an authentication response (digest) based upin the calculated CRC-32 value of the challenge string it received from the host and the client's password, along with the 1 byte of RSSI information and 160 bytes of mu-law audio (and continues doing this every 20ms) as long as there is signal being received by the receiver.. If the host approves of the client's response, it then uses the audio and/or GPS information provided by the client (and the flow continues).

At any pont, if either side does not approve of its peers authentication repsonse, it must reply to its peer with a packet with a payload type of 0 (indicating that authentication need to take place).

The connection-less nature and the flexibility of this protocol allow the client to continue operating even of the host stops or changes its challenge stirng in mid-stream.

Basically, when a client starts up, it just starts sending out packets, whether or not the host is there, or responding or repsonding positively. Once it receives valid authentication information from the host, it then is to send periodic GPS packets (once every second), and audio packets (once every 20ms) when there is audio to be sent (a signal is present on the receiver, or there is a signal to transmit). If, at any point the client gets reponses from the host that do no contain valid authentication, it merely goes back to sending authenticaiton packets once again, and the whole process repeats. Similarly, any time the host receives a packet from the client that does not contain valid authentication, it sends an authentication packet to the client.

Therefore, the protocol is completely tolerant of interruptions in network connectivity, restarts of either or both sides of the connection, and accomplishes this quite easily and with very little complicaiton and overhead.

In summary, there are 5 cases of packets:

24 octets, payload 0, digest 0 -- This is only sent by a client
24 octets, payload 0, digest (calculated) – This is only sent by a client
25 octets, payload 0, digest (calculated) ,flags – This is only sent by a host
185 octets, payload 1, digest (calculated)
50 octets, payload 2, digest (calculated) – This is only sent by client

A host will not send a packet to a client until the client has send a packet fo the host. Therefore the "24 octets, payload 0, digest 0" packets will only be sent by a client (since a digest of 0 indicates not yet hearing from a peer). A client may send as many payload 0 packets as is appropriate (there really is no limit). A host must respond to each of these packets.

Presence of payload 1 packets indicates valid signal received or to be transmitted.

For a "receive-only" implementation, the host may only send payload 0 packets in repsonse to packets from the client. It need not send any payload 1 packets if doing so is inappropriate.
A client need not keep intrinsic track of date information (such as a real-time clock, etc). The client may, if no other means is available, determine current date (not including time of day) from the time information in the authentication packet last received from the host. Therefore, the host must send time information that has an accurate date (accurate time is not required).