RTCM Channel Driver

From "AllStarLink Wiki"

The chan_voter channel driver is the interface between Asterisk and the RTCM/VOTER radio/repeater hardware interface. It allows receiver voting, simulcast transmitters, or just a plain repeater interface when used with the appropriate hardware.

This page will document configuring the channel driver, done through voter.conf, as well as some common issues and more obscure features that may not be widely known.

Contents

Basic Information On How chan_voter Works

Each node has a number of potential "clients" associated with it. In the voter.conf file, each stanza (category) is named by the node number that the clients specified within the stanza are to be associated with. Each entry consists of an arbitrary (relatively meaningless, just included for easy identification purposes within this channel driver, and has nothing to do with its operation) identifier equated to a unique password. This password is programmed into the client. **All clients must have unique passwords, as that is what is used by this channel driver to identify them.**

Each channel instance (as opened by app_rpt as a main radio channel, e.g. rxchannel=Voter/1999 in rpt.conf) and is directly associated with the node that opened it.

Each client has a pair of circular buffers, one for mu-law audio data, and one for RSSI value. The allocated buffer length in all clients is determined by the "buflen" parameter, which is specified in the "global" stanza in the voter.conf file in milliseconds, and represented in the channel driver as number of samples (actual buffer length, which is 8 * milliseconds).

Every channel instance has a index ("drainindex"), indicating the next position within the physical buffer(s) where the audio will be taken from the buffers and presented to the Asterisk channel stream as VOICE frames.

Therefore, there is an abstraction of a "buffer" that exists starting at drainindex and ending (modulo) at drainindex - 1, with length of buflen.

Buflen is selected so that there is enough time (delay) for any straggling packets to arrive before it is time to present the data to the Asterisk channel.

The idea is that the current audio being presented to Asterisk is from some time shortly in the past. Therefore, "Now" is the position in the abstracted buffer of 'bufdelay' (generally buflen - 160, you gotta at least leave room for an entire frame) and the data is being presented from the start of the abstracted buffer. As the physical buffer moves along, what was once "now" will eventually become far enough in the "past" to be presented to Asterisk (gosh, doesn't this sound like a scene from "Spaceballs"??.. I too always drink coffee while watching "Mr. Radar").

During the processing of an audio frame to be presented to Asterisk, all client's buffers that are associated with a channel instance (node) are examined by taking an average of the RSSI value for each sample in the associated time period (the first 160 samples of the abstracted buffer (which is the physical buffer from drainindex to drainindex + 159) and whichever one, if any that has the largest RSSI average greater then zero is selected as the audio source for that frame. The corresponding audio buffer's contents (in the corresponding offsets) are presented to Asterisk, then ALL the clients corresponding RSSI data is set to 0, ALL the clients corresponding audio is set to quiet (0x7f). The overwriting of the buffers after their use/examination is done so that the next time those positions in the physical buffer are examined, they will not contain any data that was not actually put there, since all client's buffers are significant regardless of whether they were populated or not. This allows for the true 'connectionless-ness' of this protocol implementation.

Asterisk CLI Commands

These are the CLI commands that Asterisk and chan/voter supports:

- voter debug level {0-7}
  - Prints debug information from the channel driver at increasing verbosity
- voter test instance_id [test value]
  - Specifies/Queries test mode for voter instance
  - Test Value can be:
    - 0 - Normal voting operation
    - 1 - Randomly pick which client of all that are receiving at the max RSSI value to use
    - >1 - Cycle thru all the clients that are receiving at the max RSSI value with a cycle time of (test mode - 1) frames. In other words, if you set it to 2, it will change every single time. If you set it to 11, it will change every 10 times. This is serious torture test.
- voter prio instance_id [client_id] [priority value]
  - Specifies/Queries priority value for voter client
    - *CLI> `voter prio 1999` *will query node 1999. Not overridden = the value from voter.conf is being used*
    - *CLI> `voter prio 1999 North` *will query node 1999 site North*
    - *CLI> `voter prio 1999 North 10` *Highest priority will always vote (value 0 - 100)*
    - *CLI> `voter prio 1999 North -1` *will stop node 1999 site North from voting. Shuts off receiver.*
    - *CLI> `voter prio 1999 North -2` *Returns site to normal. ie "prio: 0 (not overridden)". Can also use off or disable in place of -2."*

- voter record instance_id [record filename]
  - Enables/Specifies (or disables) recording file for chan/voter
- voter tone instance_id [new_tone_level(0-250)]
  - Sets/Queries Tx CTCSS level for specified chan/voter instance
- voter reload
  - Reload chan/voter parameters
- voter display [instance]
  - Display voter instance clients
- voter txlockout [instance] <client_list>
  - Set Tx Lockout for voter instance clients
    - *CLI> `voter txlockout 1999` *will show a list of locked-out and able to transmit sites*
    - *CLI> `voter txlockout 1999 +North` *will add lock-out the North site. Transmit disabled.*
    - *CLI> `voter txlockout 1999 -North` *will remove lock-out of the North site. Transmit enabled.*
- voter ping [client] <# pings, 0 to abort>
  - Ping (check connectivity) to client

voter.conf Variables

This is a dump of variables found in the source... they'll need to be fleshed out with some descriptions and applications:

```
linger (uses default if not specified)
plfilter <--this doesn't seem to work?
hostdeemp
duplex (defaults to 1)
mixminus (defaults to 0)
streams
txctcss
txctcssfreq
txctcsslevel (defaults to 62)
txtoctype (defaults to none?, options phase and notone)
primary
isprimary
thresholds
gtxgain (defaults to default_gtxgain)

password
sanity
puckit
dyntime
buflen
utos
port
bindaddr

These are options allowed for each site:
transmit
master
adpcm
nulaw
dynamic
gpsid
buflen
nodeemp <-- this works, bypasses the deemp filter (sets the switch H)
hostdeemp
noplfilter <-- this works, toggles the pl filter in the rtcm (sets the switch L, default seems to be H so the fi
prio
```

Less Known Facts/Features

ADPCM Functionality

The original intent was to change this driver to use signed linear internally, but after some thought, it was determined that it was prudent to continue using mulaw as the "standard" internal audio format (with the understanding of the slight degradation in dynamic range when using ADPCM resulting in doing so). This was done because existing external entities (such as the recording files and the streaming stuff) use mulaw as their transport, and changing all of that to signed linear would be cumbersome, inefficient and undesirable.

"Dynamic" client functionality

DON'T USE IT!!. It is intentionally **NOT** documented to encourage non-use of this feature. It is for demo purposes **ONLY**. The chan/voter driver will **NOT** properly perform reliably in a production environment if this option is used.

Duplex Mode 3/Hostdeemp

As of VOTER board firmware 1.19 (7/19/2013), there is a set of options in both the firmware ("Offline Menu Item 12, "DUPLEX3 support"), and the "hostdeemp" option (instance-wide) in the voter.conf file on the host.

Duplex Mode 3 in app_rpt allows for "in-cabinet" repeat audio (where the actual radio hardware supplies the repeated audio directly itself, and app_rpt simply "adds" all of the other audio as appropriate.

The RTCM/VOTER board now has an option to do the same functionality itself, for a case where local repeat audio is desired without the "network audio delay" normally associated with RTCM/VOTER board operation, and for a radio that doesn't have the option of providing "in cabinet" repeat audio (along with externally provided audio) itself.

Because of limitations with the RTCM/VOTER board hardware (being that there is only 1 audio path into the processor, and it either has de-emphasis in its "hardware path" or not), it is necessary if you:

1. Wish to have the "duplex=3" functionality in app_rpt
2. Have the "DUPLEX3" support enabled in the RTCM/VOTER board
3. Have a transmitter that you are "modulating directly" (with flat audio)

If all of the above is true, then you need to use the "hostdeemp" option in chan/voter (voter.conf), which basically "forces" the RTCM **NOT** to do de-emphasis in hardware (it will send the non-de-emphasized audio to the host), and have the host "do" the de-emphasis (in software) instead.

This will allow the RTCM/VOTER board to be able to "pass" the non-de-emphaszed audio back into the "direct modulation audio" stream, since that is what will be "presented" to the processor in the RTCM/VOTER board, as the hardware de-emphasis is disabled in this mode.

If you have a transmitter that you are "feeding" line-level (mic) audio, then this mode is not necessary, as the RTCM/VOTER board is fully capable of providing the functionality all by itself.

Obviously, it is not valid to use **ANY** of the duplex=3 modes in a voted and/or simulcasted system.

Redundant "Proxy" Mode

A "Redundant" (backup) server may be set up, so that if the "primary" server fails, clients can detect this failure, and connect to the designated "backup" (or "secondary") server.

Needless to say, since Internet connectivity is not by any means guaranteed to be consistent, it is possible for some clients to have working connectivity to the "primary" server and not others, even though the "primary" server is functional.

If this was to occur, actual voting and/or simulcast clients would have a "broken" system (being that all the clients need to be on the same server for any sort of functional operation).

To eliminate this possibility, functionality has been added so that a "secondary" server will "proxy" (forward) all of its VOTER packets to the "primary" (if the "primary" is on line), and the "primary" will generate all of the outbound VOTER packets, which (for clients "connected" to the "secondary" server) get sent to the "secondary" server to distribution to its clients.

This allows for a "unity" of all of the clients on a network, even though they may be connected to different servers.

In addition, it is assumed that "permanent linking" (at least of some sort) will be provided between the channel side of the chan/voter instances (presumably through a "perma-link" provided by app_rpt).

When the "secondary" is "proxying" (to the "primary") it does not provide direct connectivity to/from its locally-connected clients, thus allowing them to "connect" via the "primary" server instead. In "normal" mode, it works "normally".

The operation is performed by more-or-less "encapsulating" the VOTER packets received by the "secondary" server, and forwarding them on to the "primary" server, where they are "un-encapsulated" and appear to that serer to be coming from clients connected directly to it (and keeps track of which ones are connected in this manner, etc). When it needs to send VOTER packets to a client connected through the "secondary", it "encapsulates" them, and sends them to the "secondary", where they get "un-enacpsulated" and sent to their associated connected clients, based upon information in the "encapsulation".

If the "secondary" server loses (or does not make) connection to the "primary", it operates as normal, until such time as it can make the connection.

The server redundancy feature is local to each chan/voter instance.

For each chan_voter instance served by both the "primary" and "secondary" servers, the client list (parameters, etc) **MUST** be identical.

In addition, the following things must be added uniquely on each server:

- In the "primary" server, there needs to be a "primary connectivity" client specified for each "secondary" server for which it is "primary". Basically, this is a client that does NOTHING other then providing a means by which the "secondary" can determine whether the "primary" is on line.
- It is a standard chan/voter client, with nothing else specified other then its password. Again, although it is a "legitimate" client (technically), its only purpose **MUST** be to allow the secondary server to connect to it.
- The "primary" server also needs to have the following in all of its instances that require redundancy:

```
isprimary = y
```

- The "secondary" server needs to have the following in all of its instances that require redundancy:

```
primary = 12.34.56.78:667,mypswd
```

Where 12.34.56.78:667 is the IPADDDR:PORT of the "primary" server, and mypswd is the password of the "primary connectivity" client

Note: Master timing sources **MUST** be local to their associated server, and therefore, can not be operated in a redundant configuration. If a radio needs server redundancy, it **CAN NOT** be connected to a master timing source. Also, the master timing source **MUST** be associated with a chan/voter instance that **DOES NOT** have redundancy configured for it, even if a separate instance needs to be created just for this purpose.

Also, if non-GPS-based operation is all that is needed, just the use of redundancy within the clients is sufficient, and does not require any use of the server redundancy features.

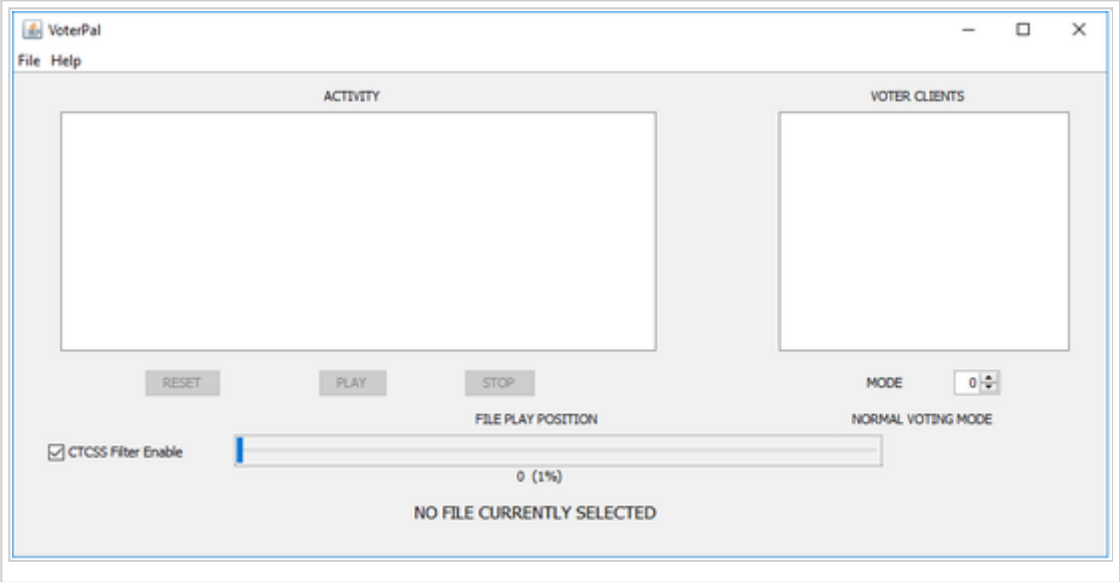
Voter Recording and Playback

chan_voter has the ability to record VOTER streams by the channel driver live on the air (using the 'voter record' command in Asterisk). This also records time-stamped data and the voted receiver at that particular time along with RSSI readings. This can then be played back using the VoterPal GUI JAVA applet.

How to setup playback

1. Download the VoterPal application.
2. Download the appframework-1.0.3.jar and swing-worker-1.1.jar files. Place these in a folder called lib in the same directory as the VoterPal application
3. Run VoterPal.jar. You may come across issues with running Java such as security permissions. Most of these can be allowed.
4. Once running, you will see the applications GUI. File > Open to select your voter data files. A bunch of test files have been made already to test here

Reset



Resets the file back to start

Play

Plays the recorded file

Stop

Stops the current playing file in it's current position

CTCSS Filter Enable

Removes CTCSS hum from any recorded audio stream

File Play Position

Drag this slider to skip

Activity

This will show time-stamped information including RSSI of the selected VOTER client within the recorded data stream

Voter Clients

This will show the real time voted client. This is also dependent of the Mode setting

Mode

Various modes can be selected in VoterPal

0 - Normal Voting Mode

1 - Randomly pick which client of all that are receiving at the max RSSI value to use

> 1 - Cycle thru all the clients that are receiving at the max RSSI value with a cycle time of (test mode - 1) frames. In other words, if you set it to 2, it will change every single frame. If you set it to 11, it will change every 10 frames. This is a serious torture test.

< 0 - Any value less than zero will force select the Voter Client (i.e. -1 is the first client, -2 is the 2nd client)

Node Configuration

As mentioned above the Asterisk CLI is as follows -

- voter record instance_id [record filename] - Enables/Specifies (or disables) recording file for chan/voter

Where

- instance_id = voter number instance

- record filename is the filename of the recording to be stored

A script can be made and placed in cron.hourly which will run every hour. Every time the above command is run in Asterisk, it creates a new file. An example of such a script is below

```
#!/bin/sh asterisk -r -x "voter record instance_id /tmp/voter-record"`date +%F-%H-%M` >> /var/log/voterrecordlog
```

the /tmp/ directory can be substituted for mounted USB HDD.

Categories: How to | Node Configuration
