DMRGateway 2 Way

We are pleased to announce the open beta test of DMRGateway 2 Way. This software & hardware package will allow a AllStar node to connect to a IPSC network for full transceive. The software will run on Linux x86, x64 and the Raspberry Pi2. The hardware required is a DV3000 Vocoder from NW Digital Radio. You can use either the ThumbDV (USB) or PiDV (GPIO).

Two software packages are required to build the DMR <---> AllStar Gateway, DMRGateway and DMRlink. Most communications between modules, ASL, DMRGateway, AMBEServerGPIO and ambe_audio are done using UDP packets. This allows for modules to be hosted on different machines (if desired). You could for example, run ambe_audio, AMBEServerGPIO and DMRGateway on one machine and ASL on another.

DMRGateway is the audio bridge between AMBE (DMR) and PCM (Analog). Its main purpose is to encode PCM to AMBE and to decode AMBE to PCM. It uses UDP to communicate between several endpoints in the system. DMRGateway when in DMR receive mode, listens for AMBE frames. For each frame it will format and transmit it to the Vocoder. The resulting PCM audio is forwarded to AllStar via the chan_usrp channel driver. In the DMR transmit mode, DMRGateway listens for PCM from chan_usrp, formats and transmits them to the Vocoder. The resulting AMBE frames are packaged into IPSC frames and sent to the DMR network. The chan_usrp channel driver is included with the DIAL distributions and possibly some of the lesser distributions. In DIAL, chan_usrp must be enabled in /etc/asterisk/modules.conf.

The Second required package is ambe_audio.py and its configuration files. This Python script is part of the larger DMRlink package. ambe_audio receives AMBE frames from the IPSC network, and transmits them to DMRGateway. In the other direction, ambe_audio listens for AMBE from DMRGateway and sends them to the IPSC network. Since the analog source does not contain Metadata (peer and subscriber radio IDs) , ambe_audio will provide Metadata in the constructed stream.

Some general notes about usage are in order here. First, this should NEVER be used to connect to a DMR network that does not allow this type of traffic. Specifically, this should not be used on the DMR-MARC networks as they explicitly forbid it. Cort, N0MJS had provided the option of testing this on the SANDBOX network. This network is designed for experimentation and is perfect for ASL to DMR gateways. ASL node 2100 is already using this (at times) and it works very well. Contact Steve, N4IRS for IP address and authorization keys to access the SANDBOX. In addition, BrandMeister also is very experimentation friendly. You can connect to the BM network, and with the help of one of the administrators, have access to several TGs on their system.

- Components
    - AMBEServer and AMBEServerGPIO (Vocoder)
        - If using the ThumbDV plug it into any available USB slot. Verify it

exists in /dev/USBx (probably 0) Then edit the DV3000 section of DMRGateway.ini to point to the /dev/USBx port
        - If using a DV3000, compile/install AMBEServerGPIO (currently done by install script, should be optional) and edit DMRGateway.ini to point to the IP address of the server.
- DMRGateway
    - Place the DMRGateway in your path /usr/local/bin (done by install script)
    - Copy the DMRGateway.ini file in /etc (done by install script)
    - Adjust any parameters necessary in the DMRGateway.ini file (IP address, etc). The DMRGateway.ini file is very well documented internally. We suggest you read and understand all of the comments in the doc even if you are not changing them.
- DMRlink
    - Make sure the prerequisites are installed (python, twisted, bit string, etc) The complete DMRlink package is installed in /opt/dmrlink by the install script.
    - Although DMRlink bridge is not required, you may need this to route DMR packets where you want them
    - ambe_audio.py
        - Adjust ambe_audio.cfg - REQUIRED
        - **gatewayDmrId** must be set to a valid DMR ID. This is the User ID the traffic will appear to come from.
        - **section** should be set to reference a named section with network specific values for talk groups
        - Within a section the **tgFilter** value defines which talk groups are passed on to the DMRGateway. Set this appropriately.
    - Adjust dmrlink.cfg as needed
        - **RADIO**_ID will be used when generating DMR from the gateway. This is the Repeater ID the traffic will appear to come from.
- Analog
    - AllStar
        - Given a running ASL node, enable the chan_usrp channel driver in /etc/asterisk/modules.conf
        - Create a private node in ASL using the chan_usrp channel driver
    - Dongle Mode (experimental)
        - Dongle mode is used to communicate on a DMR network with only the use of a DV3000, sound fob, speaker and microphone.
        - Connect up your sound in/out device (CM1xx is perfect)
        - Make sure OSS sound module is loaded (modprobe snd_pcm_oss)
        - Edit DMRGateway.ini to set useMicrophone to true

- PTT is either VOX or toggled by the keyboard (any key on/off)
- Use alsamixer to adjust speaker and mic audio levels

- Execution
  - The packages are designed to run in the background, and no GUI is required. First time installers may want to create multiple ssh windows and run each component in the foreground to watch for errors and to verify proper operation. Once all is well, then proceed to run each component in the background.
  - Make sure ASL is running (ps -ef)
  - Connect your public ASL node to the private node number
  - Start AMBEServer (if needed)
  - Start DMRGateway
  - Start ambe_audio.py (python ambe_audio.py)
  - DMR to ASL
    - For each transmission you will see logging from ambe_audio with slot, TG and ID or call
    - DMRGateway is silent for this stage unless an error occurs
    - ASL should receive the audio and pass it on to all connected nodes
  - ASL to DMR
    - chan_usrp sends each PCM frame to DMRGateway (no logging)
    - DMRGateway will output logging on each start and end of transmission
    - ambe_audio will log each transmission
    - ambe_audio will pass the DMR on to IPSC on the assigned slot and TG with the proper DMR IDs
  - Remote control
    - In order to change the execution without restarting the ambe_audio server you may send it remote control commands. The commands can be generated any number of ways but an easy interface is to use netcat. Send the commands to the IP address and port specified in the ambe_audio.cfg file. A guru can hook this up to an ASL script command.
    - Change talk groups to monitor:
      - echo -n "tgs=x,y,z" | nc 127.0.0.1 31002
      - where x,y,z are the new tg numbers. Remember this is in memory only and will reset if the server is restarted.
    - Reread the subscribers (use the get_ids.sh script)
      - echo -n "reread_subscribers" | nc 127.0.0.1 31002