Limiting the Number of Connects to a Node and Load Balancing

Many nodes are on limited bandwidth connections and might need connection limitations or you might need to load balance connections between nodes. The scripts below allow you to modify your extensions.conf file and add a small script to set connection limits for nodes on your Hamvoip server. This also gives you the opportunity to play a very short voice message to the connecting node when the limit is exceeded. Here is the example:

Note that the examples have changed when using Hamvoip update version at or greater than **RPi2-3-4 version 1.6-10 or later** the following script is NOT needed and the extensions file is handled differently. See the update extensions file below in that case.

Limiting the Number of Connects

Method for Hamvoip Versions BEFORE xxxxxxxx and legacy systems.

Create a script in **/var/lib/asterisk/agi-bin/count_connects.sh**

```
#!/bin/bash
Links=$(asterisk -rx "rpt xnode $1" | grep NUMALINKS | cut -d "=" -f2)
if [ "$Links" == "" ] ; then Links=0 ; fi
echo "SET VARIABLE result $Links"
echo $Links
```

Make this script executable - **chmod 755 count_connect.sh**

EXAMPLE for basic limits

Then edit the [radio-secure] stanza in **/etc/asterisk/extensions.conf**

[radio-secure]

```
exten => xxxxx,1,AGI(count_connects.sh, "xxxxx")
exten => xxxxx,n,Verbose(Number of direct connections is: ${result})
exten => xxxxx,n,Gotoif($[ ${result} >= Y ]? 80:4)
exten => xxxxx,4,rpt,xxxxx
exten => xxxxx,80,Playback(/root/queue_full,noanswer)
exten => aaaaa,1,rpt,aaaaa
exten => bbbbb,1,rpt,bbbbb
```

The playback line is optional. Replace with the following line if not used -

```
exten => xxxxx,80,Hangup
```

Add the first 3 lines, the 4th you would already have and change the xxxxx to the node number you want to limit on ALL lines. Add the fifth and sixth lines and add node there in place of xxxxx. Change the "Y" to the maximum number of nodes you will accept in line 3.

Line 5 above is optional. If you leave it out be sure to change line 6 from 81 to 80. Line 5 takes advantage of a feature of Asterisk called “early media” or “early audio.” There is a very small window between asking for a connect and getting connected or rejected that allows you to play an audio file to the connecting node. The window is about 5 seconds. In the example above the audio file was created in the /root directory and is called queue_full.ul. In this test case it says “Queue Full use node xxxxx” This message just fits in the available time slot. Users should customize a message that tells users where else to connect or you could say something like “No connections available.” **NOTE that this playback is experimental! The audio file should not exceed the 5 second window! If you use this method check it by connecting to yourself!**

You may have other nodes listed in [radio-secure] shown here as lines 6 and 7 with aaaaa and bbbbb as second and third node numbers. They could also have the same lines added as in the first node in the same way for each node if you wanted connection limits for them also.

When changing the extensions.conf file you must reload it in the Asterisk client - “extensions reload” or restart Asterisk.

If a node tries to connect and the connection limit is exceeded they will get thr playback file if it is used and a “connection failed” message. As soon as the number of connections is reduced below the limit they will connect normally.

For Hamvoip version RPi2-3-4 version 1.6-10 or greater use this method -

In this Hamvoip version the event method in rpt.conf has an added feature of being able to return variables that can be directly used by the extensions file. This eliminates the need for an external AGI script.

In this example we assume node 22900. You would use your own node number in its place.

In rpt.conf in the nodes stanza add the events= statement

[22900]

events=events22900

Then below the node stanzas make sure there is a matching events stanza -

[events22900]
LINKS0 = G|S|${RPT_NUMALINKS}

So LINKS0 now has the number of direct connects to node 22900. This is automatically updated on each connect/disconnect event. The G means set the assigned variable to a global and the S means set that variable to the value of the expression to the right of the second vertical bar.

And the extensions.conf file would now look like this -

[globals]
LINKS0 = 0

[radio-secure]

exten => 22900,1,Set(i0=${GLOBAL(LINKS0)
exten => 22900,n,Verbose(Number of direct connections is: ${result})
exten => 22900,n,Gotoif($[${i0} >= Y]? 80:4)
exten => 22900,4,rpt,22900
exten => 22900,80,rpt,1999
exten => aaaaa,1,rpt,aaaaa
exten => bbbbb,1,rpt,bbbbb

Change 22900 to your actual node.

Add to node 1999 in rpt.conf - **connpgm=/etc/asterisk/local/dump_node**

Add this script - /etc/asterisk/local/dump_node

#!/bin/bash
sleep 8
asterisk -rx "rpt playback 1999 /etc/asterisk/local/connection_limits"
sleep 10
asterisk -rx "rpt fun 1999 *76"

Timings before and after the playback will need to be adjusted to the length of the sound file. Typically the user will get the connect to 22900 message then the playback message will be heard and then the node will be disconnected.

Load Balancing Method

The Pi4 is a very powerful platform and tests have shown that it can handle as many as 200 direct connects when running in turbo mode and using Hamvoip code. In order to achieve this the load is spread out over four virtual nodes on the one server and a method of load balancing to ensure that any one node sees no more than 50 connects. One caveat is the use of Supermon and the accessing of the AMI can greatly effect processor loading. Code is being developed to reduce the load but until that happens the number of Supermon or Allmon users should be limited on the server. **Note that this method requires Hamvoip version RPi2-3-4 version 1.6-10 or later!** Here is the example programming to do this -

In rpt.conf you create six nodes. One is a Hamvoip registered node. The other five are what we call virtual nodes. These need to have node numbers out side of the entire Hamvoip range. In this example we add a “1” before the official node number and make five consecutive nodes. We will use registered node 68500 in this example and then 168501 through 168504 as our virtual nodes. You would of course replace the primary node with your node assignment and then use virtual number assignments of that nodes number with a “1” before it. Node 168505 is what we call a dump node. When limits are reach incoming connections go to that node where a message is played and the connection terminated.

Then in each node stanza create the following events=

[168501]
events=events168501

[168502]
events=events168502

[168503]
events=events168503

[168504]
events=events168504

Then after the nodes stanzas create the actual events stanzas.

[events168501]
LINKS1 = G|S|${RPT_NUMALINKS}

[events168502]
LINKS2 = G|S|${RPT_NUMALINKS}

[events168503]
LINKS3 = G|S|${RPT_NUMALINKS}

[events168504]
LINKS4 = G|S|${RPT_NUMALINKS}

Then in extensions.conf create the following -

radio-secure]
exten => 68500,1,NoOp(Date: ${STRFTIME(${EPOCH},,%m/%d/%Y-
exten => 68500,n,NoOp(Incoming node: ${CALLERID(number)})
exten => 68500,n,Set(i0=${GLOBAL(LINKS1)
exten => 68500,n,Set(i1=${GLOBAL(LINKS2)
exten => 68500,n,Set(i2=${GLOBAL(LINKS3)
exten => 68500,n,Set(i3=${GLOBAL(LINKS4)

;;; Calculate total connections.
exten => 68500,n,Set(Total=$[${i0} + ${i1} + ${i2} + ${i3}])
;;; Compare total to 204 (200 plus 4 local virtual nodes)
exten => 68500,n,GotoIf($[${total} < 204]?68500,100:68500,30)

exten => 68500,30,NoOp(Too many connections to node 68500!)
exten => 68500,n,168505|X

;;; Call dns-query to cache the node info, speeding performance.
exten => 68500,100,System(dns-query ${CALLERID(number)})
exten => 68500,n,Set(j0=${IF($[${i0} <= ${i1}] ? 1 : 0)})
exten => 68500,n,Set(j1=${IF($[${i0} <= ${i2}] ? $[${j0} + 1] : ${j0})})
exten => 68500,n,Set(j2=${IF($[${i0} <= ${i3}] ? $[${j1} + 1] : ${j1})})
exten => 68500,n,GotoIf($[${j2} = 3]?68500,200:68500,300)

exten => 68500,200,NoOp(connecting to node 168500)
exten => 68500,n,rpt,168501|X

exten => 68500,300,Set(j0=${IF($[${i1} <= ${i0}] ? 1 : 0)})
exten => 68500,n,Set(j1=${IF($[${i1} <= ${i2}] ? $[${j0} + 1] : ${j0})})
exten => 68500,n,Set(j2=${IF($[${i1} <= ${i3}] ? $[${j1} + 1] : ${j1})})
exten => 68500,n,GotoIf($[${j2} = 3]?68500,400:68500,500)

exten => 68500,400,NoOp(connecting to node 168501)
exten => 68500,n,rpt,168502|X

exten => 68500,500,Set(j0=${IF($[${i2} <= ${i0}] ? 1 : 0)})
exten => 68500,n,Set(j1=${IF($[${i2} <= ${i1}] ? $[${j0} + 1] : ${j0})})
exten => 68500,n,Set(j2=${IF($[${i2} <= ${i3}] ? $[${j1} + 1] : ${j1})})
exten => 68500,n,GotoIf($[${j2} = 3]?68500,600:68500,700)

exten => 68500,600,NoOp(connecting to node 168502)
exten => 68500,n,rpt,168503|X

exten => 68500,700,NoOp(connecting to node 168503)
exten => 68500,n,rpt,168504|X

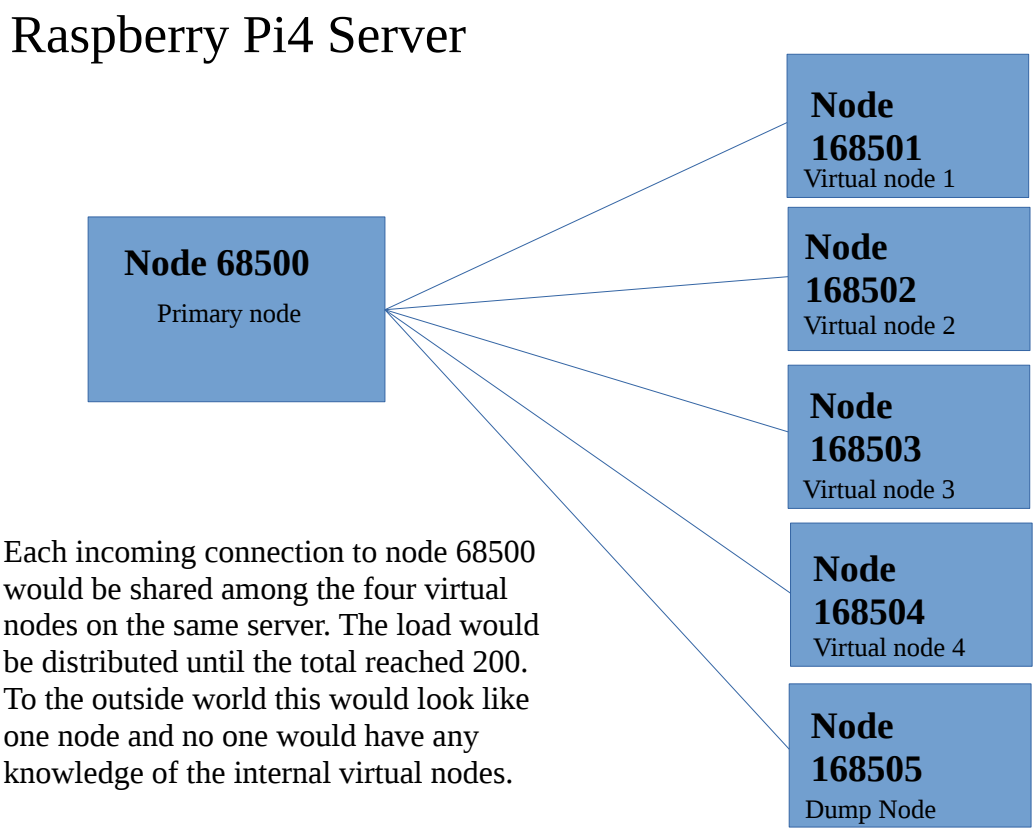
Add to node 168505 in rpt.conf - **connpgm=/etc/asterisk/local/dump_node**

Add this script - /etc/asterisk/local/dump_node

#!/bin/bash
sleep 8
asterisk -rx "rpt playback 168505 /etc/asterisk/local/connection_limits"
sleep 10
asterisk -rx "rpt fun 168505 *76"

Timings before and after the playback (the sleep values) will need to be adjusted based on the length of the sound file. Typically the user will get the connect to 68500 message then the playback message will be heard and then the node will be disconnected from 68500. Test this by connecting to the dump node directly. The recorded message can be anything you want. An example would be "This node has reached connection limits, please try node xxxxx or try back here later."

In practice the registered node, in this case 68500, is the node that all users would connect to. Behind the scenes this node is connected to each of the four virtual nodes. All nodes on the server are setup as pseudo nodes without any radios since this is a hub. As the users connect they are assigned to each virtual node in succession starting with 168501 balancing the load between the four virtual nodes. When the total connections on all four virtual nodes is 200 the user is connected to the dump node and optionally a message is sent back to the connecting node that the server is full or directing them to another node elsewhere and they are disconnected. A script will be called to aggregate the four virtual nodes status information and pass it back to the status server as the primary node 68500 in this case. All connections will appear as being connected to the primary node, 68500 in this case.



This graphic shows the primary nodes relationship to the virtual nodes. All nodes are on one server. The rpt.conf file contains the definitions for all the nodes. Nodes can be defined by include statements to make the file more manageable. The four virtual nodes 1-4 could have the same definitions. A typical rpt.conf file would look like this -

[68500]
includeifexists /etc/asterisk/rpt68500.conf

[168501]
includeifexists /etc/asterisk/rpt16850x.conf
[168502]
includeifexists /etc/asterisk/rpt16850x.conf
[168503]
includeifexists /etc/asterisk/rpt16850x.conf
[168504]
includeifexists /etc/asterisk/rpt16850x.conf

[168505]
includeifexists /etc/asterisk/rpt168505.conf

Rest of standard rpt.conf functions

Here is what the various rpt.conf files would look like. The commands shown are the required changes other commands are optional and if not specified would be at their defaults. The individual nodes could be in include files or this could all be in one file as shown here. This would be up to the user. Playback would be used on the main node to send messages to all connected users.

/etc/asterisk/rpt.conf for load balancing example.

```
[68500]
; Main node
rxchannel = dahdi/pseudo
startup_macro=*168501,*168502,*168503,*168504,*168505
statpost_program=/etc/asterisk/local/balance_node_stats.sh
statpost_url=Y

[168501]
; virtual node 1
rxchannel = dahdi/pseudo
events=events168501
statpost_program=/etc/asterisk/local/balance_node_stats.sh
statpost_url=N

[168502]
; virtual node 2
rxchannel = dahdi/pseudo
events=events168502
statpost_program=/etc/asterisk/local/balance_node_stats.sh
statpost_url=N

[168503]
; virtual node 3
rxchannel = dahdi/pseudo
events=events168503
statpost_program=/etc/asterisk/local/balance_node_stats.sh
statpost_url=N

[168504]
; virtual node 4
rxchannel = dahdi/pseudo
events=events168504
statpost_program=/etc/asterisk/local/balance_node_stats.sh
statpost_url=N

[168505]
; virtual node 5 – dump node
rxchannel = dahdi/pseudo
statpost_program=/etc/asterisk/local/balance_node_stats.sh
statpost_url=N
```

```
[events168501]
LINKS1 = G|S|${RPT_NUMALINKS}

[events168502]
LINKS2 = G|S|${RPT_NUMALINKS}

[events168503]
LINKS3 = G|S|${RPT_NUMALINKS}

[events168504]
LINKS4 = G|S|${RPT_NUMALINKS}

[nodes]
68500 = radio@127.0.0.1/68500,NONE
168501 = radio-internal@127.0.0.1/168501
168502 = radio-internal@127.0.0.1/168502
168503 = radio-internal@127.0.0.1/168503
168504 = radio-internal@127.0.0.1/168504
168505 = radio-internal@127.0.0.1/168505
```

In /etc/asterisk/iax.conf

```
[general]
;;;; other config
disallow=all
allow=slin
;;;; other config


[radio-internal]
type=friend
codecpriority=host
disallow=all
allow=slin
context=radio-internal
transfer=no
```