The HPSDR server handles communications with the HPSDR hardware and is designed to work with multiple receivers either on a single Mercury card or multiple Mercury cards. It demultiplexes the I/Q samples into separate streams for each receiver.

The hardware server listens on port 11000 for TCP connections from clients. The client sends commands to the server over the TCP connection. IQ data is sent over UDP.

**attach <rx>**

attach client to receiver.
example: attach 0

response: OK <samplerate>
ERROR <message>

**detach <rx>**

detach client from receiver
example: detach 0

response: OK
ERROR <message>

**frequency <f>**

set attached receiver to the specified frequency (in Hz)
example: frequency 7056000

response: OK
ERROR <message>

**start iq <port>**

start send I/Q samples for the attached receiver to the UDP port specified.
example: start iq 12000

response: OK
ERROR <message>

**start bandscope <port>**

start sending bandscope I/Q samples to the UDP port specified.
example: start bandscope 12500

response: OK
ERROR <message>

**stop iq**

stop sending I/Q data from the attached receiver.
stop bandscope
stop sending bandscope I/Q data.

response: OK
ERROR <message>

A typical exchange would be:

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>attach 0</td>
<td>OK 96000</td>
</tr>
<tr>
<td>set frequency 7056000</td>
<td>OK</td>
</tr>
<tr>
<td>start iq 12000</td>
<td>OK</td>
</tr>
<tr>
<td>set frequency 7056100</td>
<td>OK</td>
</tr>
<tr>
<td>stop iq</td>
<td>OK</td>
</tr>
<tr>
<td>detach 0</td>
<td>OK</td>
</tr>
</tbody>
</table>

The I/Q samples are accumulated in the server until there are 1024 I/Q samples to send. The samples are sent in 512 byte UDP packets, to the port specified in the 'start iq' message, with 12 byte header and up to 500 bytes of I/Q samples.

The UDP packet header contains:

- 8 byte sequence number
- 2 byte offset within this set of samples
- 2 byte length of samples

Note that there are currently no commands to control setting the number of receivers, preamp, dither, random, samplerate or clock sources. As this was developed to support multiple receivers, I decided to not allow clients to change these as they would be changed for all receivers. They are all controlled by command line arguments when running the server.

```
--receivers <1..4>
--samplerate [48000|96000|192000]
--dither [off|on]
--random [off|on]
--preamp [off|on]
--10mhzsource [atlas|penelope|mercury]
--122.88mhzsource [penelope|mercury]
--micsource [janus|penelope]
--class [E|other]
```

element:

```
```
**Note:**

As an experiment, there is also a softrockserver that supports the same protocol with the restriction that there is only one receiver and it uses the USB/si570 interface to control the frequency. The I/Q samples are taken from an Audio device specified on the command line.
**dspserver**

dspserver is a client to the hpsdr server. It provides a dsp processing service using DttSP. Typically one dspserver is started for each receiver.

The dsp server listens for client TCP connections on port 8000+rx. i.e. rx0 listens on 8000, rx1 listens on 8001, etc. The client sends commands to the server over the TCP connection:

**getSpectrum** `<points>`

get spectrum samples for the specified number of points.
The response is a packet containing `<points>` samples as 8 bit unsigned values.
The actual value is -sample. i.e. sample is has a value of 120, so the value is -120.
This allows values of 0 to -255.

example: getSpectrum 480

**setFrequency** `<frequency>`

set the frequency of the receiver to `<frequency>` in Hz.

example: setFrequency 7056000

**setMode** `<mode>`

set the mode:

0 – LSB
1 – USB
2 – DSB
3 – CWL
4 – CWH
5 – FM
6 – AM
7 - DIGU
8 - SPEC
9 - DIGL
10 - SAM
11 – DRM

example: setMode 0

**setFilter** `<low>` `<high>`

set the filter.

example: setFilter 150 3450
setAGC <agc>
set the AGC.
1 – LONG
2 - SLOW
3 – MEDIUM
4 – FAST

example: setAGC 3

setNR <nr>
set noise reduction.
0 – off
1 – on

example: setNR 1

setNB <nb>
set noise blanker.
0 – off
1 – on

example: setNB 1

setANF <anf>
set automatic noise filter.
0 – off
1 – on

example: setANF 1

setRXOutputGain <gain>
set the receiver audio output gain to the value of <gain>.

<gain> has a value 0 to 1

example: setRXOutputGain 0.5

startAudioStream [<buffer size>]
start sending audio buffers. <buffersize> is optional, by default it is 480 bytes.

example: startAudioStream 480
stopAudioStream

stop sending the audio stream.

The dspserver send 2 types of packets to the client, either spectrum samples as a result of a 
getSpectrum request, or audio samples after startAudioStream request. Each packet has a 48
byte header.

<table>
<thead>
<tr>
<th>Byte</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Packet type: 0=spectrum data, 1=audio data</td>
</tr>
<tr>
<td>1-31</td>
<td>Reserved</td>
</tr>
<tr>
<td>32-39</td>
<td>Sample rate in ASCII characters</td>
</tr>
<tr>
<td>40-47</td>
<td>Meter reading in ASCII characters</td>
</tr>
<tr>
<td>48-...</td>
<td>spectrum or audio data</td>
</tr>
</tbody>
</table>

Audio data is sent as 1 channel 8bit aLaw samples at 8000 samples per second.