CHAPTER 22

Practical: A Shoutcast Server

Shoutcast is a product from Nullsoft (which is also the creator of the WinAMP MP3 player for Windows) to enable audio broadcasting over a network. Nullsoft's Shoutcast server enables clients to stream MP3 data from one client to a server, which is the server that clients connect to in order to listen to that data stream. It is, in effect, a multiplexer of audio streams. Shoutcast was one of the first applications of its kind to become popular. Although streaming network audio predates Shoutcast, it was Nullsoft's product that had the power and flexibility that enabled it to take off.

Shoutcast is also the protocol used by Shoutcast servers to stream data to the client. This protocol defines the information about the data being streamed, as well as the stream itself and how client requests are handled.

The Shoutcast protocol is, in some ways, similar to HTTP. The client request is much like an HTTP GET request (using `Icy` headers instead of URLs).

Caution A protocol called Icecast has features similar to Shoutcast, but with a completely different implementation. This chapter covers only Shoutcast.

Shoutcast Protocol

When a client makes a request to a Shoutcast server, it sends a specially formatted request to the server that looks like this:

```
GET path/to/the/file HTTP/1.0 <CRLF>
Icy-MetaData:1 <CRLF>
<CRLF>
```

This is pretty much a normal HTTP GET request, except for the inclusion of `Icy-MetaData:1`, which tells the server that it should send metadata with the stream. The server defined in this chapter always sends the metadata and ignores the specifics of the request.

You can ignore the specifics of the request because the server has only one stream. Many full-featured Shoutcast servers enable you to create multiple streams from the same server. If you want multiple streams from the server, you have to run them on different ports.

After getting a request, the server sends the header information followed by the stream data. The headers will look something like this:

```
ICY 200 OK <CRLF>
icy-name:Ocaml Rocks! <CRLF>
icy-metaint:1024 <CRLF>
Content-Type:audio/mpeg <CRLF>
icy-pub:1 <CRLF>
```

These headers tell the client some very important things. They tell the client that the request was successful; they also tell the client what the name of the stream is (`icy-name`), how many bytes will pass between metadata updates (`icy-metaint`), what content type is being streamed (in this case, mpeg audio data), and whether the server is public or private (`icy-pub`).

The metadata updates in the stream are very important. One major shortcoming of the MP3 standard is that it does not include any way of encoding information about the data (for example, the name of the song being played, the artist, and so on).

Tip The creators of the Shoutcast protocol decided that the metadata would simply be transmitted along with the stream, so the client is responsible for figuring out what part of the stream is metadata and what part of the stream is data-data. To enable the client to do this, the server tells the client how many bytes will pass before the next metadata block. In this case, the client is being told that there will be a metadata block every 1024 bytes of stream data.

A metadata block is a length byte followed by the metadata itself. The length byte is a single byte that represents the length of the metadata divided by 16. Because you know that a byte has a maximum value of 255, you know that the maximum size of the metadata block is 4 KB (or 4096 bytes). The metadata itself is (most often) the title of the current stream, which is sent as the string `StreamTitle='ACTUALSTREAMTITLE';` with `ACTUALSTREAMTITLE` as the name of the current streaming audio. This string must not contain any single quotes (`'`). The entire length of the metadata block must be at least 16 bytes long (because that is the smallest non-zero value the length byte can hold). The length can also be zero, which also means that the metadata should be zero length as well.

This is important: Metadata must show up where you say it will. You can send a zero, which means a zero length metadata block. If you do not, the client will try to read metadata where there is only real data, which can cause your playback to be choppy.

A big problem occurs when the client mishandles the metadata—the audio will skip. There is also the problem of bandwidth usage. Because you are sending 16 bytes with each metadata block in which you have data, the metadata blocks then occur along with the stream. You can waste a lot of bandwidth with metadata if your update frequency is high.

Note Why is the minimum size for the metadata block 16 bytes? Because the size of the block is specified in multiples of 16.
