

Tech Newsletter

Golden Bits Software, Inc.
3525 Del Mar Heights Rd, Suite 158
San Diego, CA 92130
858.259.3870 phone
858.259.7655 fax

Golden Bits^(R) Software, Inc.

Volume 2, Issue 1, Summer 2009

In this issue:

- RTSP

Golden Bits^(R) is a software engineering firm providing consulting services in a wide range of diverse technologies:

- Windows, Linux
- Device Drivers, Embedded Systems

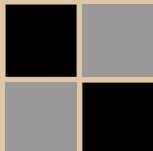
email: deang@goldenbits.com

See page 8 for past projects
www.goldenbits.com.

Technologies

- Fibre channel
- Device drivers
- Embedded
- Database, SQL
- C#, C/C++
- SCSI
- Windows
- Linux

Copyright (c) 2002-2009
Golden Bits Software, Inc.



RTSP - Real Time Streaming Protocol

Internet Standard used to control streaming media

Introduction

If you love watching sports no doubt you have a high definition TV connected to your local cable operator. However another option is using an IP based system, such as ATT U-Verse, and bypass your cable operator completely. These systems

bypass your cable operator by providing programs over a network connection.

So what does it mean when I say "IP based"?

This term generally refers to a set of network technologies which enable data to be transmitted over

the internet. For settop systems like ATT U-Verse, this means all the program data (audio and video) along with guide data is transferred using IP network data packets. These are the same IP network packets that are used to transfer web pages and email to your laptop. Instead of an email message, for settop systems the IP packet will contain part of the program's video; it is the same IP packet just different content. Streaming of programs is not limited to settop boxes, basically any smart device (phone, wristwatch, Kindle) with a display and sound can stream programs from a server.

So that's the high level view, but how this all works (the delivery of audio, video, and program guide information) is complex. To accomplish this task a set of network technologies and standards, loosely referred to as IPTV (Internet Protocol TV) is used. One of the more interesting protocols is RTSP - Real Time Streaming Protocol, which I will dive into detail in this article.

RTSP Overview

RTSP, Real Time Streaming Protocol, is used to control the actual streaming of programs from a server. A client, typically a media player of some sort, will send RSTP requests (referred to as methods in the standard document) to a server to get a program description, start and stop a program, and pause a program. Conceptually you can think of the RTSP as the remote control commands sent to a server. The protocol does not define the transmission of the actual program data (audio and video); this can be handled by other protocols such as RTP, Multicast, or a possibly a proprietary protocol.

The RTSP protocol is defined by the standard RFC 2326. The protocol is similar

continued on page 2

RTSP (continued)

to and borrows from the HTTP 1.1 protocol defined in RFC 2616; in fact the RTSP standard refers directly to sections in the HTTP 1.1 protocol. When reviewing the RTSP protocol, you should have a copy of the HTTP1.1 standard handy. The protocol is text based, so don't bother looking for bitfields and header checksums. Since this is an application level protocol, RTSP can use any underlying transport, however almost always the transport is either UDP or TCP. This also means that RTSP can take advantage of a secure transport provided by SSL. When using UDP, there are some special considerations for lost packets. Timeout values are used by the client and server to retransmit a potentially lost client command or server response.

The current standard defines RTSP version 1.0, however a new 2.0 version is in the Internet-Draft stages of the standards process. This is the early stages where the proposed draft is open for review and change. The Internet Engineering Task Force (www.ietf.org) is the organization which handles the development of Internet standards, referred to as RFCs. The 2.0 Internet-Draft can be found at <http://tools.ietf.org/html/draft-ietf-mmusic-rfc2326bis-22>

Unlike HTTP, RTSP does maintain some state information between commands. When the client sets up a media stream (via the SETUP method), the server responds with a session id which is used to identify the stream. Any command that operates on a stream must contain the stream's session id. For example, the command to pause a stream (PAUSE) requires the session id. The use of a session id enables a client to control

several media streams at once.

The two key pieces to RTSP messages are the set of methods and header fields. Methods are used to perform actions, such as PLAY or PAUSE, on the media identified by a URI or session id. Methods are also commonly referred to as commands, this article will use both terms, so Methods == Commands. Header fields contain additional information to support or facilitate the command. For example, the CSeq field specifies a sequence number which is used to identify a specific command and response. Not all field headers are used for every command, the Transport header is used by the client to specify the lower layer transport details (unicast, multicast, port numbers, etc.) for the actual steaming of the data.

Responses from the media server contain a status code and the response content. The first digit of the status code defines the class of response with the remaining

continued on page 3

Fishing Vacation

An engineer and a lawyer were recently fishing in the Caribbean. The fishing was outstanding and they got to talking about their vacations.

The lawyer said, "I'm here because my house burned down, and everything I owned was destroyed by the blazing fire. The insurance company paid for everything."

That's quite a coincidence," said the engineer. "I'm here because my house and all my belongings were destroyed by a raging flood, and my insurance company also paid for everything."

The puzzled lawyer asked, "How DO you start a flood?"

Tech Newsletter published by Golden Bits Software, Inc. Copyright (c) 2002-9. All rights reserved.

Disclaimer: All material is presented "as is" without warranty of any kind, either expressed or implied, including, without limitation, the implied warranties of merchantability or fitness for a particular purpose. Golden Bits shall not be liable for any damages whatsoever related to the use of any information presented in these materials. The sample code provided is just that, samples and is not intended for any commercial use. The information presented is as accurate as possible, however mistakes can and do happen. Please inform Golden Bits by email with any errors. Golden Bits shall not be responsible for any damages owing to editorial errors.

RSTP (continued)

digits defining the specific status. A leading digit of 1 indicates an informational response and a 5 indicates a server error. For example, a status code of 551 would indicate that an option is not supported on the server. Using a class of status enables status codes to be extensible; the client does not have to know every status code, just the status class.

The overall operation of the RTSP protocol is straight forward, the client will make a request to a media server and the media server responds. Depending on the application different protocols can be used in conjunction with RTSP. For example, there is no RTSP command to list all of the available media on a particular server. To accomplish this, another protocol such as HTTP (or even a set of private messages) can be used to get this list, the RTSP DESCRIBE method can then be used to get the details for a particular media file.

Command Format

The format of a request command is:

```
Request-line + "\r\n" + one or more headers +
"\r\n" + optional message body + "\r\n"
```

The CR LF characters "\r\n" are used to terminate a request line or header line. Since the protocol is text based the '+' symbols indicates the various fields are concatenated together. The Request-line is defined as:

```
Method + SP + Request-URI + SP + RTSP-Version
"\r\n"
```

Where 'SP' denotes a space character (ASCII 32). The key fields in Request-line are the Method and Request-URI. The Method is the actual command to run against the Request-URI. The RTSP standard defines the following commands: DESCRIBE, ANNOUNCE, GET_PARAMETER, OPTIONS, PAUSE, PLAY, RECORD, REDIRECT, SETUP, SET_PARAMETER, and TEARDOWN. The other key field in the Request-line is the Request-URI which is the URI (Uniform Resource Identifier) for media file on the server. Keep in mind that the file path in the URI is based on the server's root media directory. Specifically there is some directory on the server system that the media server program considers as the root, all of the URIs are under that root directory. This is important for the client applications, they do not have to know the details of the sever file system; however this also means that the client has to keep separate lists of media URIs for each media server.

Figure 1 shows an example of a DESCRIBE request captured by WireShark.

Notice the other header fields, CSeq, Accept, and User-Agent, each separated by a CR LF. Depending on the Method, some of these header fields are required and some are optional.

You can extend RTSP (see section 1.5 of the standard) by adding new parameters to existing methods, and/or adding a new method. However both the

continued on page 4

```

Real Time Streaming Protocol
Request: DESCRIBE rtsp://snorkel/samplevideo.ts RTSP/1.0\r\n ← Request Line
CSeq: 8\r\n
Accept: application/sdp\r\n
User-Agent: VLC media player (LIVE555 Streaming Media v2009.04.20)\r\n
\r\n

```

Figure 1

RTSP (continued)

server and client need to understand the command. For a closed system this maybe acceptable, but if your application is intended to work with different vendors then you will have to stick with commands defined in the RTSP standard.

Response format

The format of the response is defined as:

```
Status-line + "\r\n" + one or more headers +
"\r\n\r\n" + message body "\r\n"
```

The headers and message body vary depending on the original request; however the Status-line is always

present in the response message. The Status-line contains the status code and is defined as:

```
RTSP-Version + SP + Status-Code + SP + Reason-
Phrase + "\r\n"
```

The key field in the Status-line response is the Status-Code and Reason-Phase. As mentioned earlier in this article, the first digit of the status code is the status class. The Reason-Phrase is a short text description of the status code.

Figure #2 shows an example of a response message captured using WireShark. Notice the CSeq response

continued on page 5

```
Real Time Streaming Protocol
  Response: RTSP/1.0 200 OK\r\n ← Response-Line
    CSeq: 8\r\n
    Date: Sat, Jul 25 2009 14:29:21 GMT\r\n
    Content-Base: rtsp://172.24.40.142/samplevideo.ts/\r\n
    Content-type: application/sdp
    Content-length: 396
    \r\n
```

Figure 2

```
Session Description Protocol
  Session Description Protocol Version (v): 0
  Owner/Creator, Session Id (o): - 1248532101264930 1 IN IP4 172.24.40.142
  Session Name (s): MPEG Transport Stream, streamed by the LIVE555 Media Server
  Session Information (i): samplevideo.ts
  Time Description, active time (t): 0 0
  Session Attribute (a): tool:LIVE555 Streaming Media v2009.02.25
  Session Attribute (a): type:broadcast
  Session Attribute (a): control:*
  Session Attribute (a): range:npt=0-
  Session Attribute (a): x-qt-text-nam:MPEG Transport Stream, streamed by the LI
  Session Attribute (a): x-qt-text-inf:samplevideo.ts
  Media Description, name and address (m): video 0 RTP/AVP 33
  Connection Information (c): IN IP4 0.0.0.0
  Media Attribute (a): control:track1
```

Figure 3

RTSP (continued)

header containing the sequence number, this number corresponds to the CSeq in the original request.

The response to the DESCRIBE request is interesting in that the response uses the Session Description Protocol to describe the media file. The SDP (Session Description Protocol) is defined by RFC 4566. The SDP uses a set of name value pairs to describe the various attributes of the media file, each name is defined by a single character followed by the an equal character '=' and then the value. The 'a' character is used to describe different attributes, where each line contains the attribute name followed by a semicolon and then the attribute value. For example: "a=tool:LIVE55" describes the tool used to create the session description.

forward, just a simple sequence of methods to start streaming a media file. The commands (or methods as the RTSP standard states) are DESCRIBE, SETUP, PLAY, and TEARDOWN as shown in Figure 4.

Setup and playing a streaming session

The SETUP request is used by the client to setup a streaming connection with the server. The SETUP command contains the URI to the media file and the transport details to be used to stream the media. This is how the client can ask the server to use RTP and what client ports to direct the streaming data to. This enables a client that can play media in multiple windows to use different ports. Imagine a security surveillance system where each camera streams images to a client application, the application in turns displays each camera view in a separate window. To avoid port conflicts, the application would tell the server to use different ports. Figure #5 (on next page) shows a WireShark trace of a SETUP command.

The response to the SETUP command contains the session id for this media file. The session id is used to identify this particular streaming media in the PLAY, PAUSE, and TEARDOWN commands.

After the SETUP request has completed, the client can then start playing the media using the PLAY command. One of the interesting parameters to the PLAY command is the Range parameter, Range tells the server to position the play time to a specific spot and play a certain time range. The Range value can be an absolute UTC time, NPT (Normal Play Time), or SMPTE relative time stamps. Both NPT and SMPTE times are relative to the beginning of the media whereas UTC is an absolute time. When the server has completed playing a range, the server will pause automatically, no PAUSE command is required from the client. If no Range is specified, the server will start at the beginning of the media and start streaming until the end.

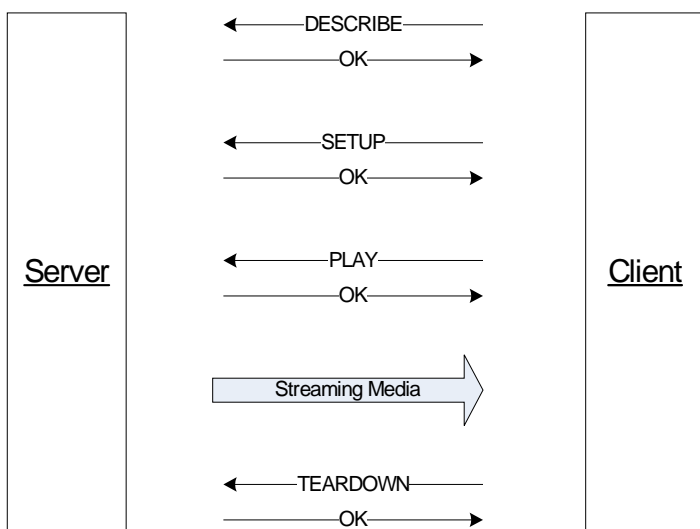


Figure 4

Figure #3 shows the response to a DESCRIBE command, note the SDP is contained in the message body of the response.

So what is involved in setting up and streaming a media file with RTSP? The command sequence is straight

RTSP (continued)

```
Real Time Streaming Protocol
Request: SETUP rtsp://172.24.40.142/samplevideo.ts/track1 RTSP/1.0\r\n
CSeq: 9\r\n
Transport: RTP/AVP;unicast;client_port=1728-1729 ← Port Information
User-Agent: VLC media player (LIVE555 Streaming Media v2009.04.20)\r\n
\r\n
```

Figure 5

What is very cool is multiple PLAY commands with different time ranges can be pipe lined to the server. The server will queue these requests and play them sequentially. Note there is no simple way to flush a particular PLAY command; the client must use the TEARDOWN command.

Once a client is done playing a media file, then the client should send a TEARDOWN command to the server. This enables the server to free any resources allocated for the session. The TEARDOWN is not implicit when the media is done streaming, the client must explicitly send the command.

One of the common features of most VOD systems is the ability to pause, rewind, and fast forward through a program. This is typically referred to as trick play, which is different from just skipping over part of the program. As you are fast forwarding, the images and sound are displayed. This is accomplished in the RTSP protocol by the 'Scale' header field. A scale value of 2 tells the server to play the medial at twice the normal speed, a negative value indicates to play in reverse.

RTSP Usage, Getting Started

So how is the RTSP used in commercial applications? Whenever you need to control streaming media over a network connection, you should think about using RTSP. Some of the commercial applications which use RTSP include Apple's Quick Time Streaming Server and Windows Media Server. RTSP is also use in a lot of VOD applications.

By now you probably can't wait to get your hands on some RTSP commands and dive deeper. The best place to start is with reading the actual RTSP standard (RFC 2326); as far as standards go it isn't terribly long or difficult to understand. I also recommend downloading and trying out several open source packages that support RTSP client and server implementations. From this you should be able to setup a server and client actually stream media files across a network connection.

continued on page 7

RTSP (continued)Resources

www.live555.com - Open source client and server code including C++ library for streaming media files.

www.videolan.org - Open source client which uses RTSP to communicate with a server, can also be used as a server.

http://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol - Wiki article.

<http://tools.ietf.org/html/rfc2326> - RFC Standard

<http://dss.macosforge.org> - Darwin, an open source version of Apple's QuickTime streaming server.

www.wireshark.org - WireShark web site. Very nice open source network trace/capture program.

Unicoi - www.unicoi.com/fusion_net/fusion_rtsp.htm. A commercial package that provides RTSP support for embedded devices.

32° 49' 7.16" N 118° 20' 57.01" W - One of my favorite places to scuba dive.

The End

So there you have it, a quick run down on RTSP. If you have any questions or comments, send me an email at deang@goldenbits.com. I'd love to hear from you.

Notes:**Efficiency Expert**

The efficiency expert concluded his lecture with a note of caution. "You don't want to try these techniques at home." "Why not?" asked someone from the back of the audience. "I watched my wife's routine at breakfast for years," the expert explained. "She made lots of trips to the refrigerator, stove, table and cabinets, often carrying just a single item at a time. 'Hon,' I suggested, 'Why don't you try carrying several things at once?'" The voice from the back asked, "Did it save time?" The expert replied, "Actually, yes. It used to take her 20 minutes to get breakfast ready. Now I do it in seven."

Project Experience - Projects which Golden Bits has been involved with

SCSI Port driver for Fibre Channel. Designed the operating system layer for a SCSI storage driver (XP, Win2K, NT, Linux) for a fibre channel HBA (host bus adapter – PCI/SBUS card).

MPEG Multiplexer. Developed core multiplexing algorithm used to multiplex several transport streams into a single output stream.

Embedded Devices. Developed an embedded monitoring device for web sites and/or other data center systems. The device uses uC/OS real time kernel running on Motorola ColdFire processor (MCF5206e). Developed management architecture for an embedded device running ARM 729 processor and uClinux.

WDM, NDIS Device Drivers. Developed a WDM and NDIS device driver for a prototype wireless system. Also developed USB wireless driver which presented the USB device as a network adapter to the host.

Device Drivers, Windows & Linux. Developed a variety of Windows and Linux drivers to handle network packet inspection, USB devices, SCSI Fibre channel adapters, ethernet adapters, and job scheduling to a device.

Parallel Search Engine. Developed a search engine that distributes database query to other systems; the search runs in parallel on the supporting systems and the results are written (through bulk inserts) back into the database.

Set Top Box. Helped port a STB to new Broadcom 7405 chip. Developed a script language and compiler used to code the television UI (guide menus, channel select).

Notes: