APPENDIX A

Inspecting WebSocket Traffic

When experimenting and building applications with WebSockets, occasionally you may need to take a closer look at what exactly is happening under the covers. Throughout this book, we've used some of these tools to examine WebSocket traffic. In this appendix, we review three handy tools:

- Google Chrome Developer Tools: a set of HTML5 applications that ships with Chrome and allows you to inspect, debug, and optimize Web applications
- Google Chrome Network Internals (or "net-internals"): a set of tools that allows you to inspect network behavior including DNS lookups, SPDY, HTTP caching, as well as WebSocket
- Wireshark: a tool that enables you to analyze network protocol traffic

WebSocket Frame Inspection with Google Chrome Developer Tools

Google Chrome Developer Tools offer a wide range of features to help web developers. Here we focus on how it helps you learn about and debug WebSockets. If you're interested in learning more about Google Chrome Developer Tools in general, there's plenty of information available online.

To access the Developer Tools, open Google Chrome, then click the Customize and Control Google Chrome icon, located to the right of the address bar. Select **Tools ➤ Developer Tools**, as shown in Figure A-1. Most developers who use this tool frequently prefer the keyboard shortcut to the menu selection.

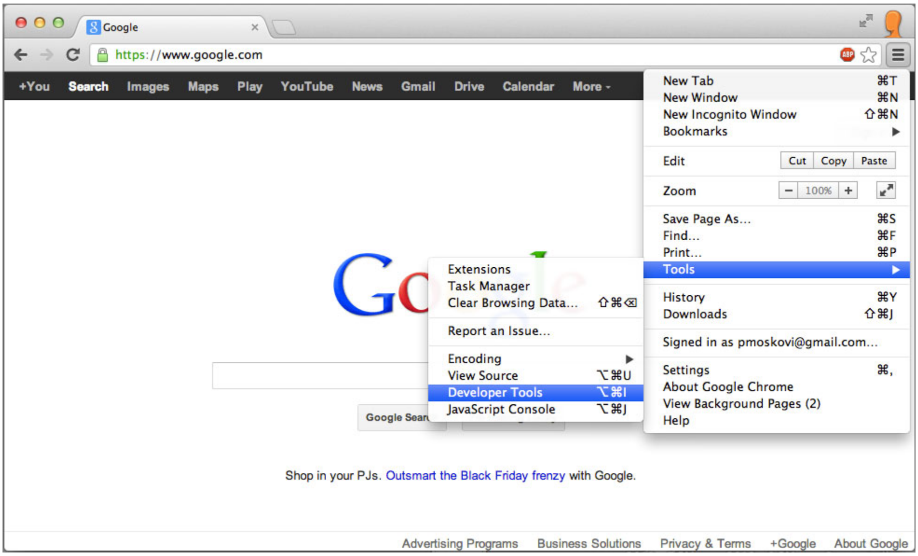

***Figure A-1.** Opening Google Chrome Developer Tools*

Google Chrome Developer Tools provide you with detailed information about your page or application through eight panels, allowing you to perform the following tasks:

- Elements panel: inspect and modify the DOM tree
- Resources panel: inspect resources loaded
- Network panel: inspect network communication; this is the panel you'll use the most while building WebSocket-enabled applications.
- Sources panel: inspect source files and debug JavaScript
- Timeline panel: analyze where time is spent when loading or interacting with your page
- Profiles panel: profile the time and memory usage
- Audits panel: analyze the page as it loads and makes suggestions to improve it.
- Console: display error messages and execute commands. The console can be used along with any of the above panels. Press the Esc key on your keyboard to open and close the console. Along with the Network panel, the Console is the Web and WebSocket developer's best friend.

First, let's take a closer look at the Network panel. Open Chrome and navigate to `http://www.websocket.org`. We will use the Echo Test on `websocket.org` to learn about

the WebSocket Frame inspection that Google Chrome Developer Tools provide.
To access the Echo demo, click the Echo Test link on the page, which will take you to `http://www.websocket.org/echo.html`. Open Google Chrome Developer Tools if you haven't opened it yet, and click the Network panel. Make sure your Network panel is empty. If it is not empty, click the Clean icon at the bottom of the Chrome Window, the sixth icon from the left in Figure A-2.

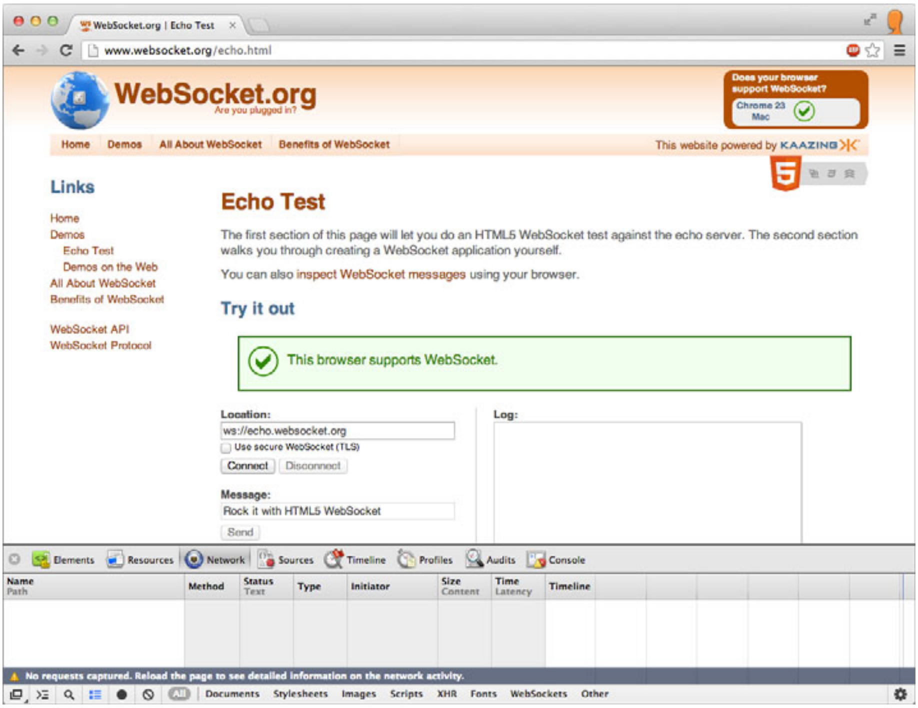

***Figure A-2.** Examining the creation of a WebSocket connection with Google Chrome Developer Tools*

Notice that the location field contains a WebSocket URL that we'll connect to: `ws://echo.websocket.org`. Click the Connect button to create the connection. Notice that the WebSocket connection displays in your Network panel. Click the name, `echo.websocket.org`, which is under the Headers tab; doing so allows you to look at the WebSocket handshake (Figure A-3). Listing A-1 shows the entire WebSocket handshake.

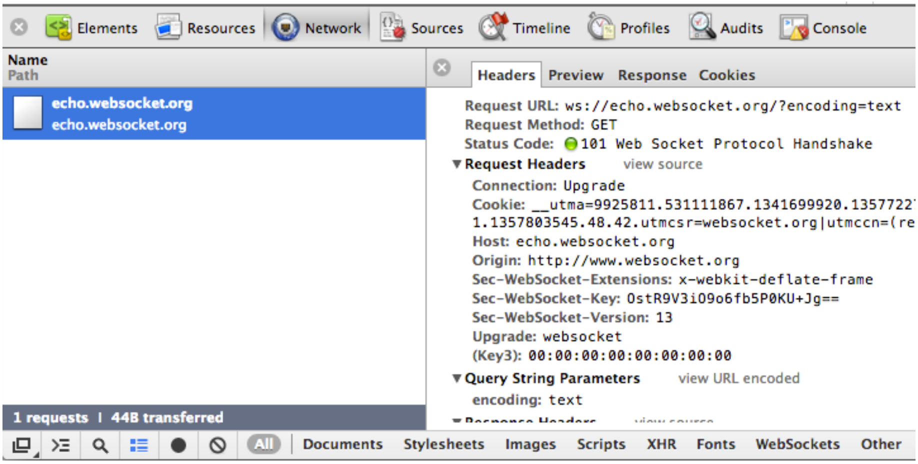

Figure A-3. *Inspecting the WebSocket handshake*

Listing A-1. The WebSocket Handshake

```
Request URL:ws://echo.websocket.org/?encoding=text
Request Method:GET
Status Code:101 Web Socket Protocol Handshake
Request Headers
Connection:Upgrade
Cookie:__utma=9925811.531111867.1341699920.1353720500.135372
5565.33; __utmb=9925811.4.10.1353725565; __utmc=9925811; __
utmz=9925811.1353725565.33.30.utmcsr=websocket.org|utmccn=(referral)|
utmcmd=referral|utmcct=/
Host:echo.websocket.org
Origin:http://www.websocket.org
Sec-WebSocket-Extensions:x-webkit-deflate-frame
Sec-WebSocket-Key:JfyxfhR8QIm3BSb0q/Tw5w==
Sec-WebSocket-Version:13
Upgrade:websocket
(Key3):00:00:00:00:00:00:00:00
Query String Parameters
encoding:text
Response Headers
Access-Control-Allow-Credentials:true
Access-Control-Allow-Headers:content-type
Access-Control-Allow-Origin:http://www.websocket.org
Connection:Upgrade
Date:Sat, 24 Nov 2012 03:08:27 GMT
Sec-WebSocket-Accept:Yr3WGnQMtPOktDVP1aBU3l5DfFI=
Server:Kaazing Gateway
Upgrade:WebSocket
(Challenge Response):00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
```

Now, feel free to change the contents of the Message field and click the Send button. To inspect the WebSocket frames, you'll need to click on the Name on the far left again, which will refresh the panel on the right, adding the Frames tab, as shown in Figure A-4.

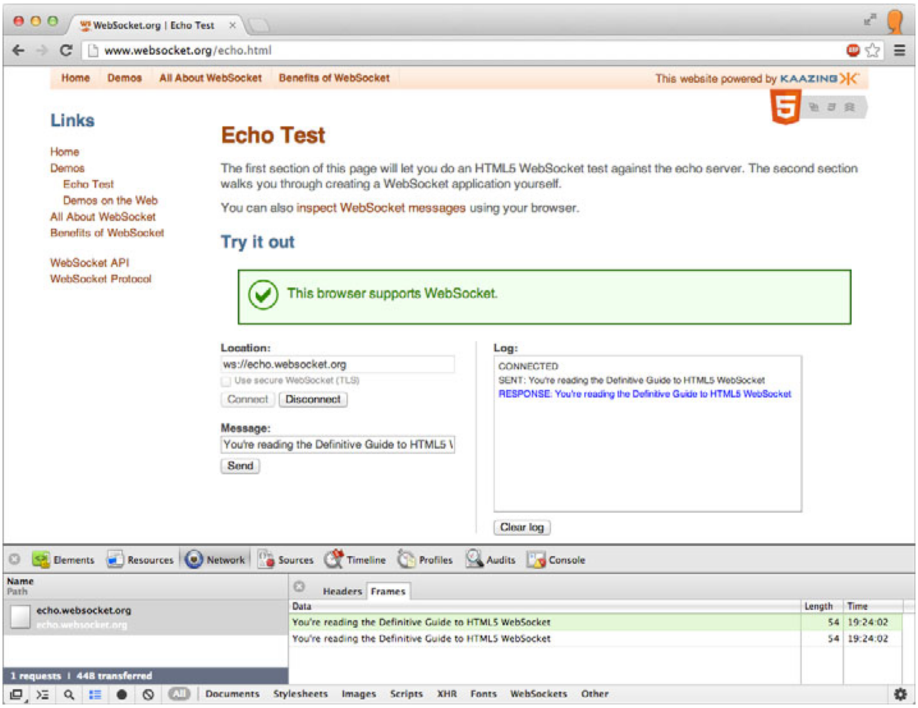

Figure A-4. *Inspecting WebSocket frames*

The WebSocket Frame inspector shows the data (which is text in this example), the length of the data, the time it was sent, as well as the direction of the data: a light green background indicates traffic from the browser to the WebSocket server (upload), and white indicates traffic from the server to the browser (download).

■ **Note** As you're sending WebSocket messages, be sure to always click the Name column to trigger the refresh of the Frames tab.

As you navigate to the Sources tab, and locate the `echo.js` file, you see a variable called `"websocket"` that represents our WebSocket connection. By displaying the Console, you can simply send a message to the WebSocket server, using the `send()` function, as shown in Listing A-2.

Listing A-2. Sending a WebSocket Message Using the Chrome Console

```
websocket.send("Hello World!");
```

In Figure A-5 we sent a `Hello World!` message from the console, and you can see that in the Log window, the Echo service sent us a response. If you display your Network tab, you can also see the corresponding WebSocket frames.

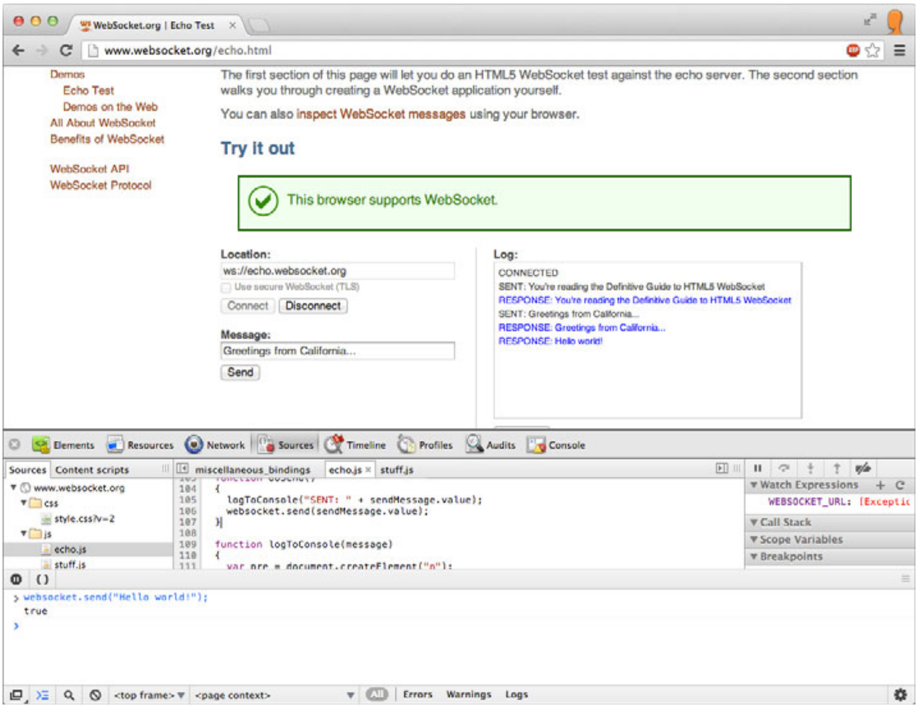

Figure A-5. *Sending WebSocket messages from the Chrome Console*

As demonstrated, the Chrome Developer Tools offer web developers a simple and effective way to "look under the hood" of their applications. Chrome's Network tab provides unique insight not only into the WebSocket handshake but also allows you to easily inspect the WebSocket frames.

Google Chrome Network Internals

Most of the time, Chrome Developer Tools display more than enough information to productively develop and debug web applications. Sometimes, however, lower-level details can help diagnose unusual connection failures or provide otherwise inaccessible information when investigating the behavior of the browser itself. Chrome has internal diagnostic pages that are extremely valuable in those rare situations in which you would like to observe the internal state of the browser. Chrome's internal tools expose events related to DNS requests, SPDY sessions, TCP timeouts, proxies, and other internal workings of the browser.

Google Chrome includes several additional utilities. For a list of them, type `chrome://about` in the browser's address bar.

■ **Note** In Google Chrome, the URL `about:about` redirects to `chrome://about`. Other browsers, such as Mozilla Firefox, have useful URLs listed on their `about:about` pages.

The page displays the following list of useful internal Chrome utilities:

- `chrome://appcache-internals`
- `chrome://blob-internals`
- `chrome://bookmarks`
- `chrome://cache`
- `chrome://chrome-urls`
- `chrome://crashes`
- `chrome://credits`
- `chrome://dns`
- `chrome://downloads`
- `chrome://extensions`
- `chrome://flags`
- `chrome://flash`
- `chrome://gpu-internals`
- `chrome://history`
- `chrome://ipc`
- `chrome://inspect`
- `chrome://media-internals`
- `chrome://memory`
- `chrome://nacl`
- `chrome://net-internals`
- `chrome://view-http-cache`
- `chrome://newtab`

- chrome://omnibox
- chrome://plugins
- chrome://policy
- chrome://predictors
- chrome://profiler
- chrome://quota-internals
- chrome://settings
- chrome://stats
- chrome://sync-internals
- chrome://terms
- chrome://tracing
- chrome://version
- chrome://print

In the address bar, type `chrome://net-internals`. One use of net-internals is to inspect TCP socket events. These TCP sockets are used to transport WebSocket and other protocols used by the browser for communication. When you click Sockets on the left, Chrome displays the socket pools. What we're interested in is the currently active, live sockets, so click the View live sockets link. In a separate window or tab, open the WebSocket Echo test at http://www.websocket.org/echo.html, and click Connect. A new entry shows up right away, along with the following URL: ws://echo.websocket.org/?encoding=text. Click the entry, and on the right, you'll see the network internals, as shown in Listing A-4.

Listing A-4. Network Internals of a WebSocket Handshake

```
830: SOCKET
ws://echo.websocket.org/?encoding=text
Start Time: 2012-11-23 20:08:27.489

t=1353730107489 [st=  0] +SOCKET_ALIVE  [dt=?]
                          --> source_dependency = 828 (SOCKET_STREAM)
t=1353730107489 [st=  0]   +TCP_CONNECT  [dt=91]
                            --> address_list = ["174.129.224.73:80"]
t=1353730107489 [st=  0]      TCP_CONNECT_ATTEMPT  [dt=91]
                              --> address = "174.129.224.73:80"
t=1353730107580 [st= 91]   -TCP_CONNECT
                            --> source_address = "10.0.1.5:57878"
t=1353730107582 [st= 93]    SOCKET_BYTES_SENT
                            --> byte_count = 470
t=1353730107677 [st=188]    SOCKET_BYTES_RECEIVED
                            --> byte_count = 542
```

Now, from the window that displays `websocket.org`, let's send a message. The net-internals panel refreshes, and shows the number of bytes sent (see Figure A-6).

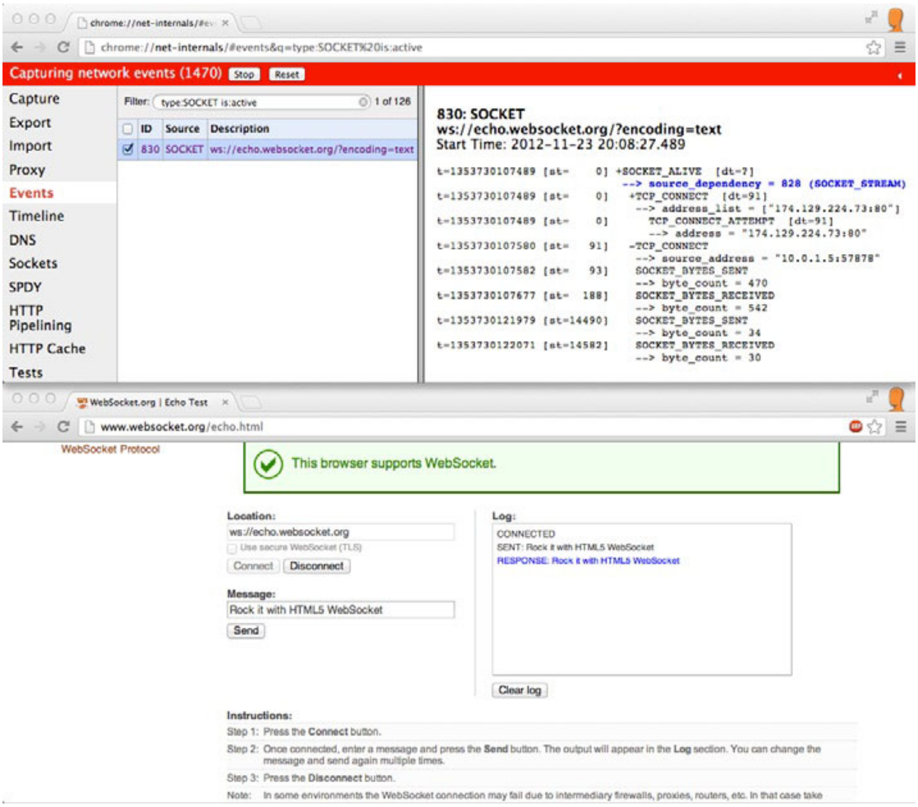

Figure A-6. *Google Chrome net-internals utility*

Much like the Google Developer Tools, net-internals is packaged and shipped with Google Chrome. Net-internals is a very handy tool if deeper, lower-level network diagnostics are required.

Analyzing Network Packets with Wireshark

Wireshark is a very powerful, free, and open source tool (available for download at `http://www.wireshark.org`) that provides detailed insight into network interfaces, allowing you to see and analyze what's traveling on the wire. Wireshark is a useful tool in WebSocket developers' hands but is widely used by network administrators, as well. Wireshark can capture live network data through the network interface that you can then export/import, filter, color code, and search.

Figure A-7 shows the Wireshark UI as it captures network packets. Under the menu bar and the main toolbar you see the Filter tool bar, which is used to filter the collected data. This data displays in a tabular format in the packet list pane. The packet details pane shows information about the packet selected in the packet list pane. The packet bytes pane, just above the status bar, displays the packet data, selected in the packet list pane.

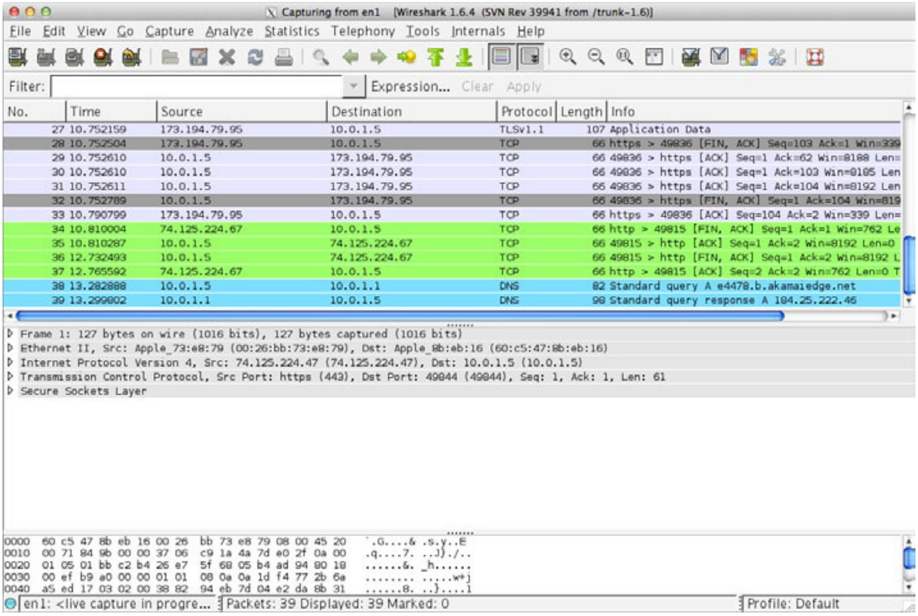

***Figure A-7.** Wireshark capturing network packets*

Start Wireshark and select the network adapter you're using: if you're hard-wired to the network, your adapter will be different than when you use WiFi. In our experiment with Wireshark, we'll inspect the WebSocket traffic between a browser and a WebSocket server, running on `websocket.org`. To get started, navigate with your browser to `http://www.websocket.org`. Then, click the Echo Test link. You can alternatively point your browser directly at `http://www.websocket.org/echo`. Now, you're ready to establish a WebSocket connection. Click the Connect button.

Since there tends to be quite a bit of traffic on the network, the traffic between your browser and websocket.org quickly scrolls out of view. To ensure we see some useful data, we'll filter for traffic going to `www.websocket.org`.

Figure A-8 shows how you can filter out packets with a specific IP address: ip.dst_host==174.129.224.73. Wireshark supports the double-equal sign in the condition, as well as the eq operator. In this figure, also notice the WebSocket handshake in the packet details page.

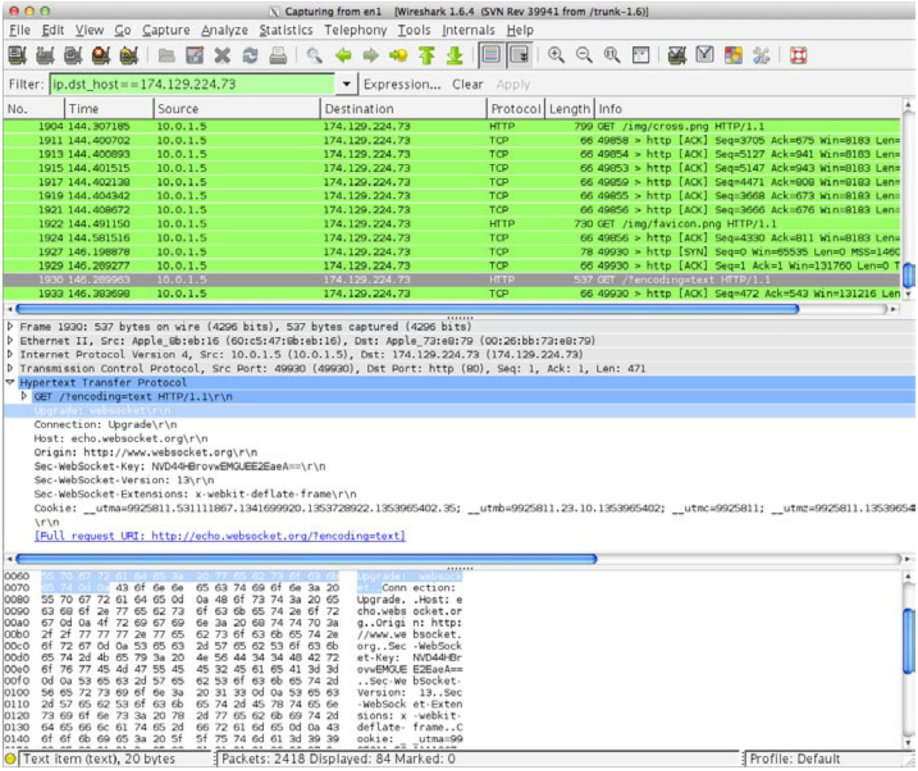

Figure A-8. *Filtering network packetsnetwork packets*

Another great feature of Wireshark is that it can follow various protocol streams. In Figure A-9 you can see how it follows a TCP stream. It displays the TCP segments that are on the same TCP connection as the selected packet. You can follow a protocol stream by right-mouse clicking on a packet in the packet list pane and choosing Follow from the context menu.

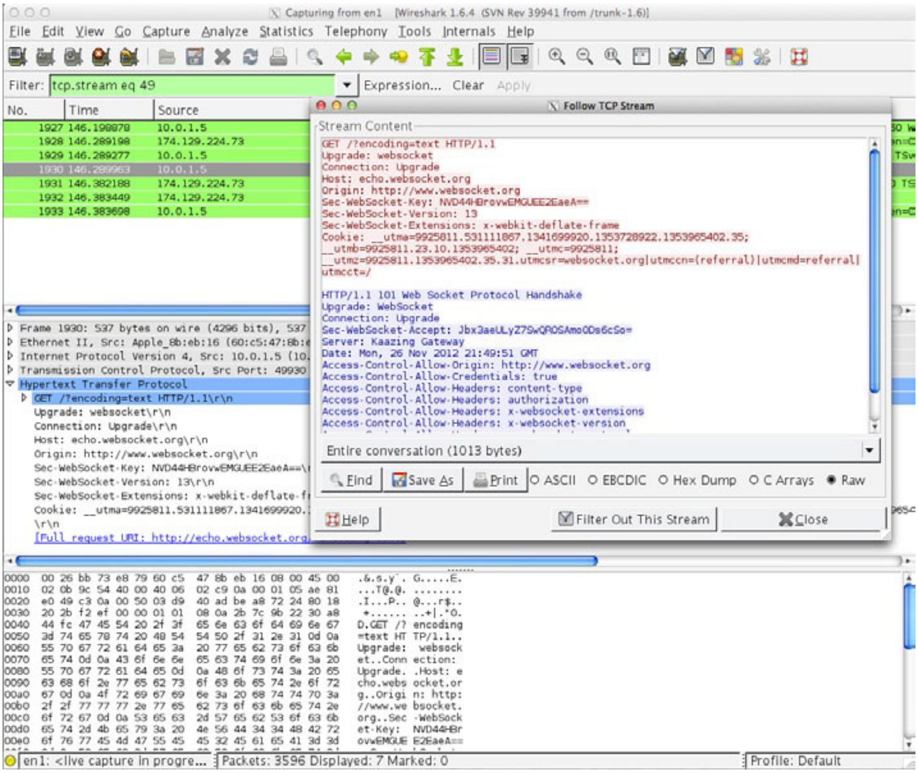

Figure A-9. *Following a TCP stream*

To see how Wireshark updates the packet list live, submit a WebSocket message in your browser. Figure A-10 shows how submitting the text, *Rock it with WebSocket*, to the Echo service appears in Wireshark.

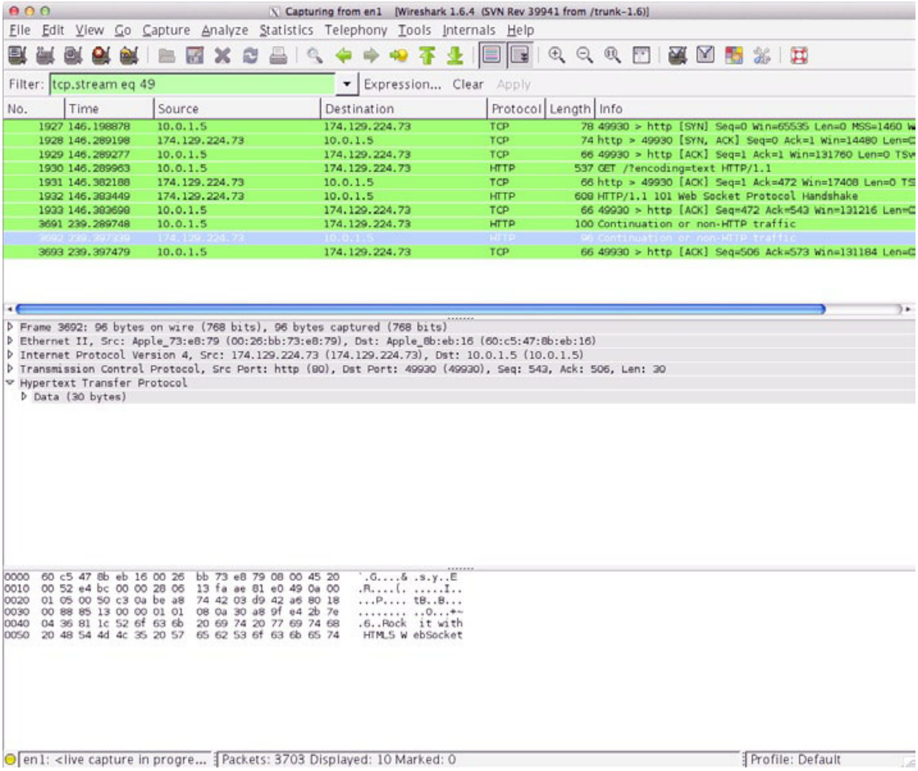

Figure A-10. *Wireshark updates live*

Summary

In this appendix, we explained some useful tools for inspecting, dissecting, and debugging WebSocket traffic. These tools will help you when building your WebSocket-enabled applications. The next appendix discusses the Virtual Machine (VM) we provide, which includes the open source code (libraries, tools, and servers) we used to build the examples in this book.

APPENDIX B

WebSocket Resources

Throughout this book, we've used a number of resources that help us build WebSocket applications every day. In this appendix, we walk through how to use the VM (virtual machine) that contains all the code and software pre-installed that you need to build or follow the examples in this book. We also summarize where to get all the libraries, servers, and other technologies we used in this book. Finally, we include a list of WebSocket servers and clients that are available at the time of writing this book.

Using the Virtual Machine

The VM accompanied by this book can be downloaded from the publisher's web site. Simply navigate to `http://apress.com` and search for this book's title (or go directly to `www.apress.com/9781430247401`). Click the Source Code/Downloads tab and click Download Now. After downloading it, you can start the VM using VirtualBox. VirtualBox is available as a free download from `http://virtualbox.org` for Windows, Mac, Linux, and Solaris host operating systems.

To open the VM, extract it, and double-click the `WebSocketBook.ova` file, or choose **File ➤ Import Appliance** from the menu of VirtualBox, and select the `WebSocketBook.vbox` file. The operating system of the VM is Ubuntu.

Once you've downloaded and installed the VM, you'll notice a few items on the desktop:

- Icons for Chapters 2–6
- A `README.txt` file

First, open and read the `README.txt`, which explains the servers and services that are automatically started for you when you install the VM. To build the examples described in Chapters 2–6, you can simply start building against the servers and libraries provided in the VM, which are described in the relevant chapter.

Tables B-1 and B-2 describe the servers and libraries that we use throughout the book and whether they are included in the VM.

Table B-1. *Servers Used in this Guide*

Server	Description	Where you can get it	Used in Chapters
Apache ActiveMQ	A popular open source message broker with support for messaging APIs and protocols, like JMS (Java Message Service) and STOMP (Simple or Streaming Text Oriented Messaging Protocol).	`http://activemq.apache.org`	5 and 7
node-xmpp-bosh	An open source server written by Dhruv Matani that enables XMPP connections over BOSH and WebSocket to any XMPP server. The server is implemented in JavaScript using Node.js.	`http://github.com/dhruvbird/node-xmpp-bosh`	4
Openfire	An open source RTC (real-time collaboration) server with support for XMPP (Extensible Messaging and Presence Protocol).	`http://www.igniterealtime.org/projects/openfire`	4
TightVNC	TightVNC is a cross-platform, open source VNC server.	`http://tightvnc.com`	**6**
`Websocket.org`	A publicly hosted WebSocket server with a simple Echo Service for testing and learning about WebSocket.	`http://www.websocket.org`	1, 3, and 7

Table B-2. *Libraries and Other Tools Used in this Guide*

Library/Tool	Description	Where you can get it	Used in Chapters
jQuery 1.8.2	A widely popular and commonly used open source JavaScript library simplifying cross-browser web development.	`http://jquery.com`	5
Node.js	A popular open source server for writing applications in JavaScript. Node.js is based on Google Chrome's performant open source V8 JavaScript with support for event-driven asynchronous I/O operations.	`http://nodejs.org`	3 and 6

(*continued*)

Table B-2. (*continued*)

Library/Tool	Description	Where you can get it	Used in Chapters
Node Package Manager (npm)	A Node.js package manager, allowing easy installation of Node.js packages.	http://npmjs.org	None (included in the VM)
Strophe.js	An open source XMPP library for JavaScript, originally created by Jeff Moffitt.	http://strophe.im/strophejs	4
VirtualBox	An open source virtualization product supporting Windows, Mac, Linux, and Solaris as the host operating system, and a significantly larger number of guest operating systems.	http://virtualbox.org	None (used to start the VM)

WebSocket Servers

While you can enable a server to accept WebSocket connections or indeed write your own WebSocket server, there are a few existing implementations that might make your life easier when developing your own WebSocket applications. At the time this book was written, the following are some of the WebSocket servers that are available (list provided by http://refcardz.dzone.com/refcardz/html5-websocket):

- Alchemy-Websockets (.NET): http://alchemywebsockets.net/
- Apache ActiveMQ: http://activemq.apache.org/
- apache-websocket (Apache module): http://github.com/disconnect/apache-websocket#readme
- APE Project (C): http://www.ape-project.org/
- Autobahn (virtual appliance): http://autobahn.ws/
- Caucho Resin (Java): http://www.caucho.com/
- Cowboy: http://github.com/extend/cowboy
- Cramp (Ruby): http://cramp.in/
- Diffusion (Commercial product): http://www.pushtechnology.com/home
- EM-WebSocket (Ruby): http://github.com/igrigorik/em-websocket
- Extendible WebSocket Server (PHP): http://github.com/wkjagt/Extendible-Web-Socket-Server

- gevent-websocket (Python): `http://www.gelens.org/code/gevent-websocket/`
- GlassFish (Java): `http://glassfish.java.net/`
- Goliath (Ruby): `http://github.com/postrank-labs/goliath`
- Jetty (Java): `http://jetty.codehaus.org/jetty/`
- jWebsocket (Java): `http://jwebsocket.org/`
- Kaazing WebSocket Gateway (Commercial product): `http://kaazing.com/`
- libwebsockets (C): `http://git.warmcat.com/cgi-bin/cgit/libwebsockets/`
- Misultin (Erlang): `http://github.com/ostinelli/misultin`
- net.websocket (Go): `http://code.google.com/p/go.net/websocket`
- Netty (Java): `http://netty.io/`
- Nugget (.NET): `http://nugget.codeplex.com/`
- phpdaemon (PHP): `http://phpdaemon.net/`
- Pusher (cloud service): `http://pusher.com/`
- pywebsockets (Python): `http://code.google.com/p/pywebsocket/`
- RabbitMQ (Erlang): `http://github.com/videlalvaro/rabbitmq-websockets`
- Socket.io (Node.js): `http://socket.io/`
- SockJS-node (Node): `http://github.com/sockjs/sockjs-node`
- SuperWebSocket (.NET): `http://superwebsocket.codeplex.com/`
- Tomcat (Java): `http://tomcat.apache.org/`
- Tornado (python): `http://www.tornadoweb.org/`
- txWebSocket (Python/Twisted): `http://github.com/rlotun/txWebSocket`
- vert.x (Java): `http://vertx.io/`
- Watersprout (PHP): `http://github.com/chrisnetonline/WaterSpout-Server/blob/master/server.php`
- web-socket-ruby (Ruby): `http://github.com/gimite/web-socket-ruby`

- Webbit (Java): `http://github.com/webbit/webbit`
- WebSocket-Node (Node.js): `http://github.com/Worlize/WebSocket-Node`
- websockify (Python): `http://github.com/kanaka/websockify`
- XSockets (.NET): `http://xsockets.net/`
- Yaws (Erlang): `http://yaws.hyber.org/websockets.yaws`

Index

S, T, U

V

W, X, Y, Z