Clojure | IPv6 | DNSSEC | Routing | Tomcat | Spyder

LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

Create Web Applications with CLOJURE

Develop Scientific Python Code

JULY 2013 | ISSUE 231 | www.linuxjournal.com

NETWORKING

BUILD COMPLEX LINUX-BASED ROUTERS

NETWORKING ISSUE SPONSORED BY

Microway®
Technology you can count on℠

intel® inside™ Xeon Phi™

TOMCAT ADMINISTRATION AND SETUP

REVIEW: SAMSUNG ARM CHROMEBOOK 3G

Clojure | IPv6 | DNSSEC | Routing | Tomcat | Spyder

LINUX JOURNAL™

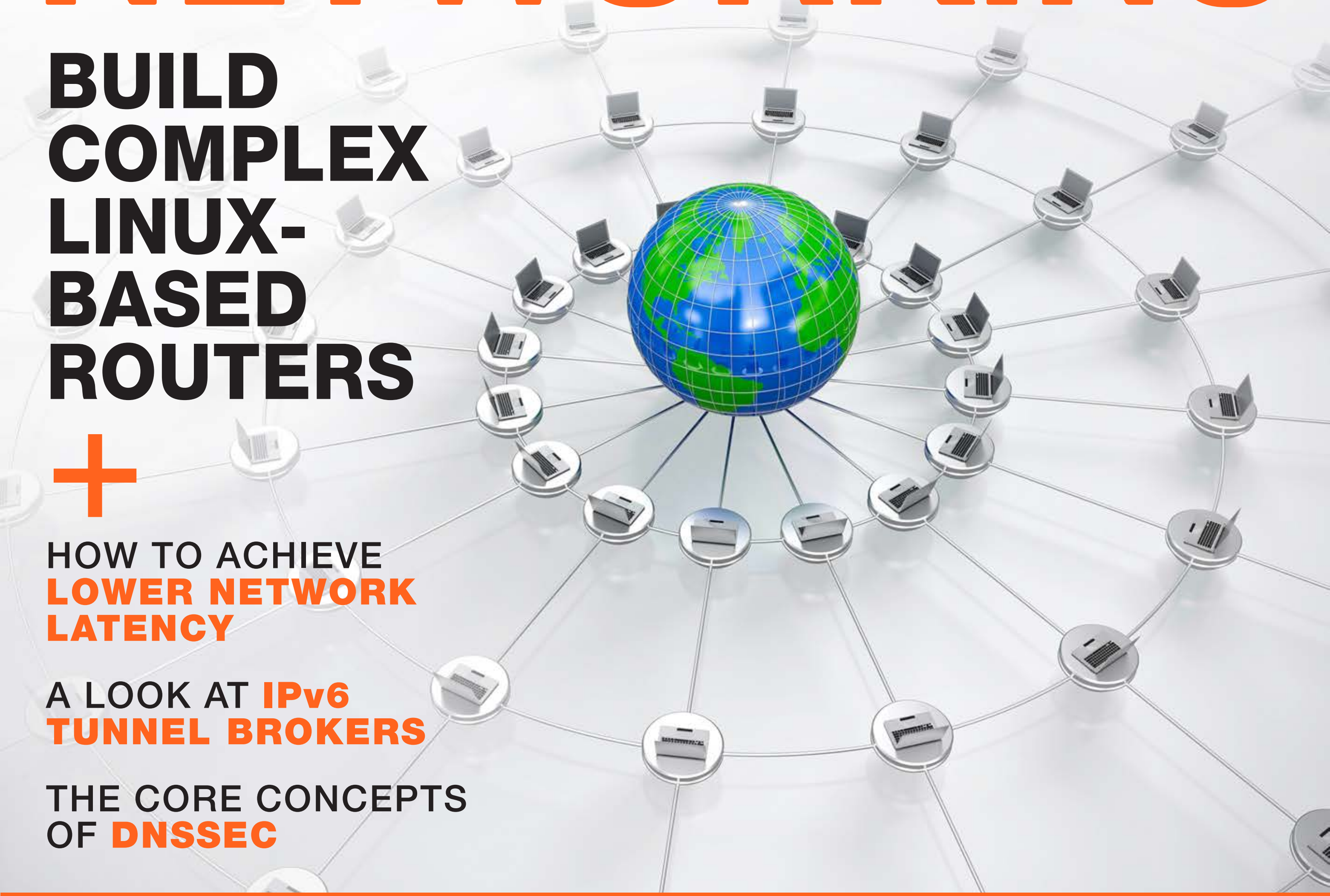
Since 1994: The Original Magazine of the Linux Community

Create Web Applications with **CLOJURE**

Develop Scientific Python Code

JULY 2013 | ISSUE 231 | www.linuxjournal.com

NETWORKING



BUILD COMPLEX LINUX-BASED ROUTERS

+

HOW TO ACHIEVE **LOWER NETWORK LATENCY**

A LOOK AT **IPv6 TUNNEL BROKERS**

THE CORE CONCEPTS OF **DNSSEC**

TOMCAT ADMINISTRATION AND SETUP

REVIEW: SAMSUNG ARM CHROMEBOOK 3G

Accelerate Your App

Are you tired of watching a sad workstation chug through your data?

Sign up for a test drive on one of our GPU server solutions today. See how you can accelerate your code or applications with parallel processing on NVIDIA® Tesla® K20 GPUs.

http://www.siliconmechanics.com/testdrive

NVIDIA.

SILICON MECHANICS

Expert included.

CONTENTS

JULY 2013
ISSUE 231

NETWORKING

FEATURES

80 Popular IPv6 Tunnel Brokers
A comprehensive review of various IPv6 tunnel providers and protocols as well as general advice on which protocols to choose and which to avoid.
Igor Partola

92 Queueing in the Linux Network Stack
Ever wondered where packets are queued within the kernel or how to achieve lower network latency? Read this article to find out.
Dan Siemon

106 Linux Advanced Routing Tutorial
Learn to build moderately complex Linux-based routers.
Michal Ludvig

COLUMNS

36 Reuven M. Lerner's At the Forge
Intro to Clojure on the Web

44 Dave Taylor's Work the Shell
Cribbage: Closing Things Up

48 Kyle Rankin's Hack and /
DNSSEC Part I: the Concepts

58 Shawn Powers' The Open-Source Classroom
Taming Tomcat

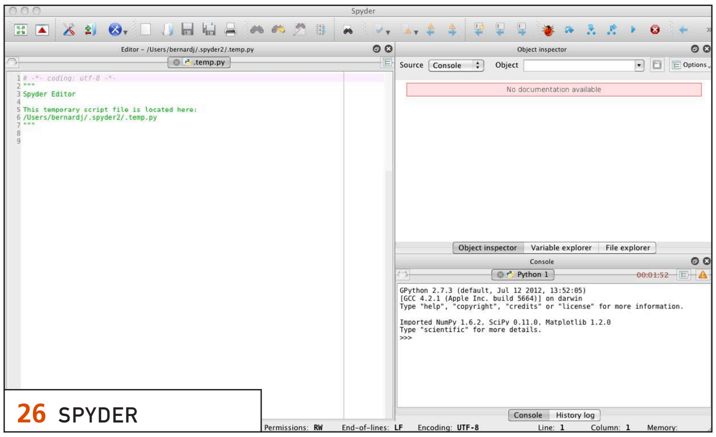
122 Doc Searls' EOF
Setting TV Free

REVIEW

70 Exploring the Samsung ARM Chromebook 3G
Bill Childers

IN EVERY ISSUE

8 Current_Issue.tar.gz
12 Letters
22 UPFRONT
34 Editors' Choice
66 New Products
127 Advertisers Index

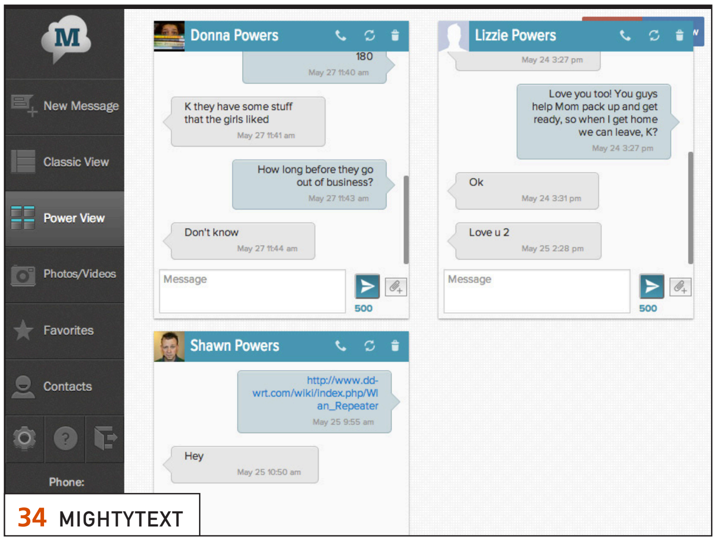


```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This temporary script file is located here:
/Users/bernardj/.spyder2/.temp.py
"""
```

```
GPython 2.7.3 (default, Jul 12 2012, 13:52:05)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.

Imported NumPy 1.6.2, SciPy 0.11.0, Matplotlib 1.2.0
Type "scientific" for more details.
>>>
```

Spyder
Editor - /Users/bernardj/.spyder2/.temp.py
Object inspector
Source Console Object
No documentation available
Object inspector Variable explorer File explorer
Console
Python 1
Console History log
Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 1 Column: 1 Memory:

26 SPYDER



Donna Powers
180
May 27 11:40 am
K they have some stuff that the girls liked
May 27 11:41 am
How long before they go out of business?
May 27 11:43 am
Don't know
May 27 11:44 am

Lizzie Powers
May 24 3:27 pm
Love you too! You guys help Mom pack up and get ready, so when I get home we can leave, K?
May 24 3:27 pm
Ok
May 24 3:31 pm
Love u 2
May 25 2:28 pm

Shawn Powers
http://www.dd-wrt.com/wiki/index.php/Wi an_Repeater
May 25 9:55 am
Hey
May 25 10:50 am

34 MIGHTYTEXT

70 SAMSUNG ARM CHROMEBOOK 3G

ON THE COVER

- Create Web Applications with Clojure, p. 36
- Develop Scientific Python Code, p. 26
- Build Complex Linux-Based Routers, p. 106
- How to Achieve Lower Network Latency, p. 92
- A Look at IPv6 Tunnel Brokers, p. 80
- The Core Concepts of DNSSEC, p. 48
- Tomcat Administration and Setup, p. 58
- Review: Samsung ARM Chromebook 3G, p. 70

LINUX JOURNAL (ISSN 1075-3583) is published monthly by Belltown Media, Inc., 2121 Sage Road, Ste. 310, Houston, TX 77056 USA. Subscription rate is $29.50/year. Subscriptions start with the next issue.

Executive Editor	Jill Franklin jill@linuxjournal.com
Senior Editor	Doc Searls doc@linuxjournal.com
Associate Editor	Shawn Powers shawn@linuxjournal.com
Art Director	Garrick Antikajian garrick@linuxjournal.com
Products Editor	James Gray newproducts@linuxjournal.com
Editor Emeritus	Don Marti dmarti@linuxjournal.com
Technical Editor	Michael Baxter mab@cruzio.com
Senior Columnist	Reuven Lerner reuven@lerner.co.il
Security Editor	Mick Bauer mick@visi.com
Hack Editor	Kyle Rankin lj@greenfly.net
Virtual Editor	Bill Childers bill.childers@linuxjournal.com

Contributing Editors
Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips • Marco Fioretti • Ludovic Marcotte
Paul Barry • Paul McKenney • Dave Taylor • Dirk Elmendorf • Justin Ryan • Adam Monsen

Publisher	Carlie Fairchild publisher@linuxjournal.com
Director of Sales	John Grogan john@linuxjournal.com
Associate Publisher	Mark Irgang mark@linuxjournal.com
Webmistress	Katherine Druckman webmistress@linuxjournal.com
Accountant	Candy Beauchamp acct@linuxjournal.com

***Linux Journal* is published by, and is a registered trade name of, Belltown Media, Inc.**
PO Box 980985, Houston, TX 77098 USA

Editorial Advisory Panel
Brad Abram Baillio • Nick Baronian • Hari Boukis • Steve Case
Kalyana Krishna Chadalavada • Brian Conner • Caleb S. Cullen • Keir Davis
Michael Eager • Nick Faltys • Dennis Franklin Frey • Alicia Gibb
Victor Gregorio • Philip Jacob • Jay Kruizenga • David A. Lane
Steve Marquez • Dave McAllister • Carson McDonald • Craig Oda
Jeffrey D. Parent • Charnell Pugsley • Thomas Quinlan • Mike Roberts
Kristin Shoemaker • Chris D. Stark • Patrick Swartz • James Walker

Advertising
E-MAIL: ads@linuxjournal.com
URL: www.linuxjournal.com/advertising
PHONE: +1 713-344-1956 ext. 2

Subscriptions
E-MAIL: subs@linuxjournal.com
URL: www.linuxjournal.com/subscribe
MAIL: PO Box 980985, Houston, TX 77098 USA

High Performance, High Density Servers for Data Center, Virtualization, & HPC

On-board 10 Gigabit Ethernet and Infiniband for greater throughput in less rack space

The Intel® Xeon® Processor E5-2600 family powers the highest-density servers iXsystems has to offer. The iXR-1204+10G features dual onboard 10GigE + dual onboard 1GigE network controllers, up to 768GB of RAM and dual Intel® Xeon® E5-2600 family processors, freeing up critical expansion card space for application-specific hardware. The uncompromised performance and flexibility of the iXR-1204+10G makes it suitable for clustering, high-traffic webservers, virtualization, and cloud computing applications - anywhere you need the most resources available.

For even greater performance density, the iXR-22X4IB squeezes four server nodes into two units of rack space, each with dual Intel® Xeon® E5-2600 Family Processors, up to 256GB of RAM, and an on-board Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector. The iXR-22X4IB is perfect for high-powered computing, virtualization, or business intelligence applications that require the computing power of the Intel® Xeon® Processor E5-2600 family and the high throughput of Infiniband.

iXR-1204+10G

- Dual Intel® Xeon® Processors E5-2600 Family
- Intel® X540 Dual-Port 10 Gigabit Ethernet Controllers
- Up to 16 Cores and 32 process threads
- Up to 768GB Main Memory
- 700W Redundant high-efficiency power supply

iXR-22X4IB

- Dual Intel® Xeon® Processors E5-2600 Family per node
- Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector per node
- Four server nodes in 2U of rack space
- Up to 256GB Main Memory per server node
- Shared 1620W Redundant high-efficiency Platinum level (91%+) power supply

E5-2600

HIGH Throughput & INCREDIBLE Performance Density

IXR-1204+10G: **10GbE On-Board**

4 Server Nodes in 2U

IXR-22X4IB

iXsystems®

intel® inside™ Xeon®

Call iXsystems toll free or visit our website today! **1-855-GREP-4-IX | www.iXsystems.com**

SHAWN POWERS

Networking, or Notworking?

When our house was built a few years ago (after our house fire), one of the things I wanted was a house wired for Ethernet connectivity. Unfortunately, the contractor must not have realized that running CAT5e isn't the same as running power lines. Although my house did in fact have RJ-45 jacks in a few rooms, opening the wall panel exposed approximately three inches of untwisted wire crammed into the wall jack. Even worse, they stapled, yes *stapled*, the *CAT3* cable to the wall studs.

When asked about the CAT3 wiring, mess of untwisted wire and stapling, the contractor assured me that CAT3 and CAT5e are the exact same thing, and that he'd been installing Ethernet cable for longer than I've been alive. Although it was flattering that he must have assumed I was born after 1990, I decided it wasn't a battle worth fighting. I've since run Ethernet cabling throughout my house, and terminated it properly. Thankfully, while this issue of *Linux Journal* is focused on networking, all the articles are written by experts, and none of them are stapled to the wall.

Reuven M. Lerner starts things off with a bit of Clojure. Although that may sound like a counterintuitive way to start an issue, in this case, Clojure is a way to make Web applications. It's a modern variant of Lisp, and whether you love or hate Lisp, it's worth checking out Clojure. And, Dave Taylor provides us with a real bit of closure this month, as he finishes his series on *Cribbage*. Whether you love the game, or just love the scripting process, it's been an educational series.

We get launched directly into the networking issue with Kyle Rankin's Hack and / column. Kyle starts a series on DNSSEC, which adds a much-needed layer of security to DNS. In a perfect world, DNSSEC would be unnecessary, but unfortunately, we don't live in a perfect world. I follow Kyle with my

Open-Source Classroom column. If you've never been faced with deploying WAR files, or haven't ever needed to install any Java applications, Tomcat might be confusing and foreign to you. I hope to clear some of that up this month with a primer on deploying Java on your server.

We dive in to hard-core networking next with Igor Partola's article on IPv6 tunnel brokers. If you participated in the IPv6 poll, you might be interested in setting up IPv6, and Igor explains how to do that, whether your ISP supports it or not. Speaking of ISPs, Dan Siemon shows how to look at network latency as opposed to bandwidth for measuring network performance. ISPs offer higher and higher bandwidth, but network engineers (and on-line gamers) know that there's more to the network than just speed. Dan describes how to deal with latency in our Linux systems.

Next, Michal Ludvig takes us on a trip to his Pacific island, where bandwidth comes either slow and expensive, or fast and insanely expensive. Thankfully with Linux, he can do some advanced routing to leverage multiple ISPs for maximum savings. If you need to load balance WAN connections, or if you just want to utilize your multiple connections better, Michal's article will be extremely useful.

Bill Childers is back this month with his review of the Samsung ARM-based Chromebook. I just bought the same model notebook, and I learned a lot from Bill's article. I wasn't sure if I'd love or hate a Chromebook, but Bill shows how to do a little more than Google intended. It's pretty awesome. Doc Searls finishes our issue off with a look at television. Are we approaching a major departure from TV as we know it? Looking at my house with its XBMC devices, Internet TV and streaming-only entertainment, I'd say yes. Doc agrees, and he goes into detail on what's to come.

I have a passion for networking, and this issue really piques my interest. It's hard to pick up the networking issue of *Linux Journal* and not learn something, but if nothing else, hopefully you've learned never to have a contractor install your Ethernet cabling—unless your contractor has a *Linux Journal* bumper sticker. Then he or she might be okay! ■

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via e-mail at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

OVH.COM

Dedicated server KS1

$39.00/month

free setup

Processor
Intel Core i3 2130 (Sandy Bridge)
2 Cores (4 Threads)
3.4 GHz+
RAM
8 GB DDR3
Hard Drive
2 x 1 TB SATA2
RAID SOFT (0/1)
Guaranteed Bandwidth
100 Mbps
5TB traffic/month

Dedicated server SP 2

$89.00/month

free setup

Processor
Intel E3 - 1245v2 (Ivy Bridge)
4 Cores (8 Threads)
3.4 GHz+ (3.8 GHz Turbo boost)
RAM
32 GB DDR3
Hard Drive
2x 2TB SATA3
RAID SOFT (0/1)
Guaranteed Bandwidth
100 Mbps Bandwidth included
Unlimited traffic

Dedicated server EG 3

$179.00/month

free setup

Processor
Intel E5 - 1620 (Sandy Bridge)
4 Cores (8 Threads)
3.6 GHz+ (3.8 GHz Turbo boost)
RAM
32 GB DDR3 ECC
Hard Drive
2x 3TB SATA3 + 80 GB SSD
MegaRAID 9271 6Gbps - 1 GB Cache - CacheVault - CacheCade
Guaranteed Bandwidth
200 Mbps up to 1 Gbps
Unlimited traffic

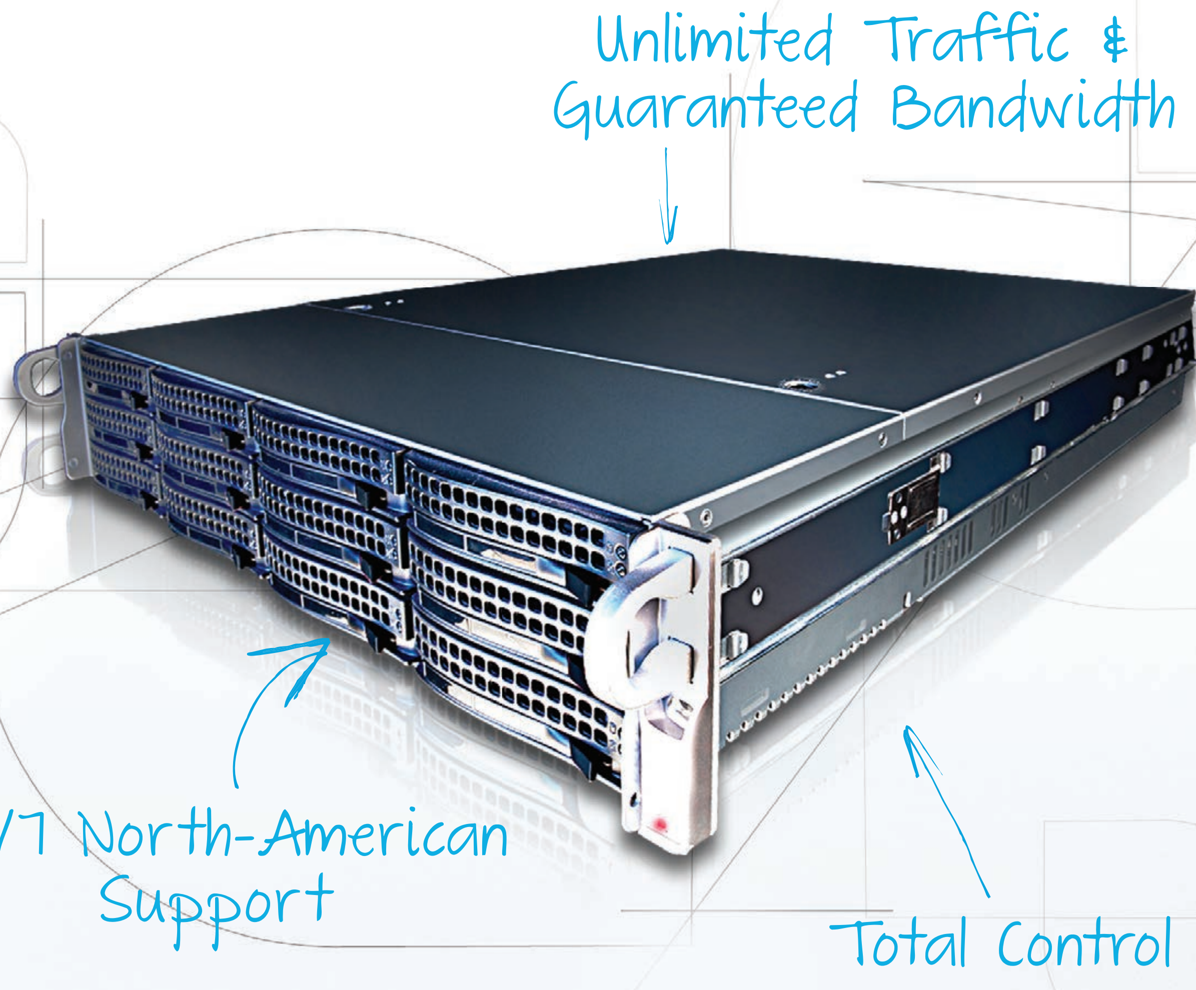
Dedicated server mHG1

$259.00/month

free setup

Processor
2x Intel Xeon E5606
2x 4 Cores
2x 2.13GHz
RAM
128 GB DDR3 ECC
Hard Drive
2x 600GB SAS 15k rpm
MegaRAID 6Gbps - Battery Kit
Guaranteed Bandwidth
300 Mbps up to 3 Gbps
Unlimited traffic

Dedicated Infrastructure for your Business



Private Cloud

Dedicated Servers

Dedicated Cloud

For more details:

OVH.COM
Dedicated Infrastructures For Your Business

or contact us: **1-855-684-5463** (toll free)

letters

Hacking Contest on a Live CD

I made something recently that you might find interesting: a hacking contest embedded on an ultra-light live CD (less than 30MB).

Just pop the ISO in VirtualBox or your favorite virtualization tool, and enjoy (ahem, struggle?) crunching through the levels. See the following links:

- My blog post: **http://www.janosgyerik.com/hacking-contest-on-a-live-cd**
- ISO file: **http://sourceforge.net/projects/ctfomatic/files**
- Source code: **https://github.com/janosgyerik/ctf-o-matic**

I hope you like it!
—Janos

I really love the idea of a bootable ISO with a built-in hacking contest. However, it really made me realize just how terrible I am at hacking! I don't want to mention exactly how sad my hacking skills are, but let's just say I don't know what it looks like when you get past the first level.—Shawn Powers

IPv6

We asked readers to write in and let us know whether they are using IPv6, and if so, what they do with it. Read on to see what they had to say.

Thomas2:
For one, I did the IPv6 certificate with Hurricane Electric, and you need one running. You also get a very geeky T-shirt when you succeed. I am (one of many hats) a network admin and want to stay up to date. I actually had to use IPv6 at work once, just for a PoC. This was just a local internal setup. At home, I have a permanent IPv6 setup via Hurricane Electric tunnel. I also had IPv6 a while ago on my server on the Internet, but I moved and didn't get a chance to set it up again. My hoster provides IPv6 via a tunnel (tunnel endpoint on their end), so I do send e-mail right now via IPv6.

Unfortunately, my ISP does not support IPv6 natively. I am in the UK, and native IPv6 is very, very rare here (there is one provider, but for businesses). So, I am running a tunnel with HE, and it works just fine.

Even though I am "certified" IPv6 with Hurricane Electric and do have some experience, it still is different in an enterprise. It is new, and people need to get to know it first. I have found that there are a lot of misunderstandings about IPv6. I like it though, and with my /48 and /64 address ranges, I have enough IP addresses, which is good—you can have never enough.

I wish I could deploy/migrate an IPv6 network in an enterprise soon—I really would love that.

Sending e-mail via IPv6 counts as something more than, "just to see if I can do it"—thanks!—Shawn Powers

Scott Gilbert:
All of my hosted VPSes have native IPv6. On some of them, I've removed the IPv4 addresses, so they are *only* accessible via IPv6.

Alas, at home, my ISP (US/Texas) does not provide IPv6, so I use a tunnel from he.net to my router (running OpenWRT). When I set this up two or three years ago, I was really disappointed in the level of IPv6 support from consumer wireless routers. It looks like things have improved somewhat since then, but v6 support still seems like an afterthought for most products.

7" Panel PC

PPC-E7+

- ARM9 400Mhz Fanless Processor
- Up to 1 GB Flash & 256 MB RAM
- 7" 800 x 480 TFT LED Backlit LCD
- Analog Resistive Touchscreen
- 10/100 Base-T Ethernet
- 3 RS232 & 1 RS232/422/485 Port
- 1 USB 2.0 (High Speed) Host port
- 1 USB 2.0 (High Speed) OTG port
- 2 Micro SD Flash Card Sockets
- SPI & I2C ports
- I2S Audio Interface w/ Line-in/out
- Operating Voltage of 12 to 26 Vdc
- Optional 2D Accelerated Video & Decoder
- Pricing start at $550 for Qty 1

2.6 KERNEL

Designed and Manufactured in the USA the PPC-E7+ Compact Panel PC comes ready to run with the Operating System installed on Flash. Apply power and watch the Linux X Windows User Interface appear on the vivid 7" color LCD. Interact with the PPC-E7+ using the responsive integrated touch-screen. Everything works out of the box, allowing you to concentrate on your application, rather than building and configuring device drivers. Just Write-It and Run-It.

www.emacinc.com/panel_pc/ppc_e7+.htm

Since 1985
OVER
28
YEARS OF
SINGLE BOARD
SOLUTIONS

EMAC, inc.
EQUIPMENT MONITOR AND CONTROL

Phone: (618) 529-4525 • Fax: (618) 457-0110 • Web: www.emacinc.com

What do I use it for? Well, everything that traditionally used IPv4, of course! (SSH, HTTP, IMAP, SMTP, BT and so on.)

As I see it, we will all migrate to IPv6 eventually (although I expect v4 and v6 will coexist for a very long time), so why not start using IPv6 now and start reaping the benefits of IPv6 as soon as possible? While working with IPv6 feels very foreign at first, one quickly finds that it is just much easier to manage than IPv4 (using v6, subnetting is trivial, there are plenty of available addresses in even a "small" /96 network, there's no need for NAT, and so on).

Scott, I think having your VPSes on both IPv4 and IPv6 (for Web sites and such) will be the key to an eventual migration. Unfortunately, since NATing works so well now, the stress over running out of IPv4 addresses has lessened. I think we'll see a hybrid world for a very long time.—Shawn Powers

Rob Hooft:
I'm in The Netherlands, and I'm using IPv6 where possible. xs4all gave me a static /48 net, which gives me enough address space to fill my whole house with 400 m3 of sand and address each of the grains individually.

At work, I am trying to convince people that all services should be hosted on both address spaces, but some hosters really make this unnecessarily complicated.

Rob, keep pushing! Until there's a saturation of services hosted on parallel address spaces, I don't think the migration will really go anywhere.—Shawn Powers

Sander Steffann:
I have been using IPv6 for almost ten years now. I was responsible for the technical department of a small ISP in The Netherlands that I cofounded, and I implemented IPv6 everywhere as soon as I could. The routers, firewalls, DNS servers, mail servers and the ISP's own Web servers all have IPv6. I left the ISP five years ago, and I am now a freelance consultant specializing in IPv6. I also run my own LISP (RFC6830)-based ISP, which, of course, has full IPv6 support. I helped several ISPs implement IPv6 in their back-bones, data centers and access (DSL, fiber) networks, did some IPv6

consultancy for a Dutch bank, and I am giving regular IPv6 training courses in the Middle East.

What do I do with it? Use the Internet. Seriously—using IPv6 should be invisible. It just works, behind the scenes. IPv6 is not something you consciously use. It is the technology that lets us keep and increase the flexibility that the Internet gives us today.

IPv4 addresses have run out. The last IPv4 addresses are in the distribution chain and are being handed out to users. Some parts of the world, like the US, still have some IPv4 addresses in their Regional Internet Registry (in the US that is ARIN). They distribute the right to use blocks of IP addresses to ISPs and companies, until they run out. ARIN is expected to run out approximately one year from now. In other parts of the world, the RIRs already have run out, and the last IPv4 addresses are being used by ISPs. Enterprises in those regions cannot get their own provider-independent IPv4 addresses anymore.

Because of the shortage of IPv4 addresses, it won't be possible to give every connected subscriber his or her own IPv4 address anymore. IPv4 addresses have to be shared. That means that connected subscribers lose control over their Internet connections. Certain applications (mostly VPNs to the office) will not work properly. Running your own servers for Web, mail and other applications will become impossible and so forth. For on-line applications (like banking), it will become almost impossible to distinguish between different subscribers based on their IPv4 address, so if one subscriber tries to attack the on-line service they will have no choice but to block all subscribers sharing the same IPv4 address—and that could be hundreds or thousands of subscribers.

IPv6 provides enough addresses to give all users more than they will ever need. This sounds like a "640K ought to be enough for anybody" statement, until you do the math. Every LAN in IPv6 gets 2^{64} = 18,446,744,073,709,551,616 addresses, and every subscriber gets a block big enough for multiple LANs (in Europe usually 65,536 subnets per subscriber, sometimes residential subscribers get only 256 subnets).

Using IPv6 means that we don't do address-sharing tricks; subscribers keep full control over their Internet connection, and security can be as fine-grained (or even better) than it is today.

As far as whether my ISP supports it natively, it's as native as you can get with LISP.

Sander, you're absolutely correct that using IPv6 should be seamless. I think we're at the place now that IPv6 should be provided to everyone by all ISPs, so that the transition/ migration can happen. I hope that is soon!—Shawn Powers

Anonymous:
My ISP is Internode in Australia, and it offers a /64 static IPv6 natively. Obviously, I use it whenever a server I connect to offers an IPv6 address. My VPS provider (Reliable Hosting) and dedicated server provider (Wholesale Internet, Inc.) both offer IPv6 on their servers.

Jonathan Guthrie:
I've had my networks on IPv6 for a long time now, as my original tunnel to the 6bone was over my Sprint T1—that was like 15 years ago. I originally connected to the 6bone to learn how this IPv6 stuff worked, and I'm still waiting for the rest of the world to figure out it's useful. My tunnel now is through he.net, because it's the best solution I could find. (I have Comcast business-class as my provider now, and it swears that it'll roll out native IPv6 "real soon now".) My VPS provider was chosen specifically because it offered native IPv6, among other criteria.

What I do with it is, well, stuff. All my computers have IPv6 addresses, and I SSH to and from them, do Web stuff and whatnot. Mr Graber's experience notwithstanding [see below], there seems to be a vanishingly small number of sites that are available over IPv6 and a vanishingly small number of people using IPv6, most of which (at least in my Apache logs) appear to be running Mac OS X. The last time I looked, about a year ago, I was getting about one IPv6 connection every other month, while my normal traffic is maybe 200–300 IPv4 connections a day (that's after filtering out my home addresses, of course).

Perhaps answering this survey will

change that. Perhaps not. I'll keep an eye on my logs.

One thing that I find interesting is that my Verizon 4G service is often IPv6. When the 4G works, that is.

I have a similar business account with an ISP, and I'm expecting IPv6 "real soon now" too. I didn't know about Verizon 4G/IPv6—that's fascinating!—Shawn Powers

Peter Nunn:
I'm using it at my home office, and my ISP, Internode, is one of the very few in oz to offer it natively.

I intend to use it to access my servers running in my network from outside, but I have to say that so far I've struggled to find an IPv6 connection anywhere else even to see if I can get back in.

Aaron Ogle:
I use IPv6 to access IPv6 Interwebs. My VPSes have IPv6, so I can set a different address for each service and so on. Being allocated a /64 and having that many routable addresses is a no-brainer. Supposedly IPv6 is coming to crapcast. I have yet to see it.

Kevin Otte:
I have an ASUS N66R router that has firmware based on DD-WRT. I have a Hurricane Electric IPv6 tunnel set up through it. It's very handy to be able to have a publicly accessible IPv6 address for every device. I am having to beef up on iptables in my firewall box though.

My home ISP does not yet have IPv6 support. It gave me the "We have enough IPv4, so we don't care" line. This made me sad. My VPS provider does offer IPv6, which has come in very handy for VPN access from places that don't.

I am testing the IPv6 on everything I can get my hands on—I've got to do it early to avoid the panic when IPv4 fully exhausts and the laggards finally freak out.

My main focus of late has been trying to get the word out. Here's a little bit about those efforts: **http://teamarin.net/2013/05/29/the-internet-its-not-commodity-its-community-guest-blog**.

I share in the disappointment that LJ.com isn't reachable over IPv6. I see from the BGP announcements that the data center is advertising a v6 prefix. Perhaps y'all would like a hand?

Although many of us are sysadmins in our day jobs, the magazine and Web site are clients in the hosting world along with every other business. It will be pretty cool if we can work with our hosting provider(s) to get native IPv6 going, but as you note, we haven't started that process yet.—Shawn Powers

Thomas Schafer:
I am using IPv6 to have Internet. As far as whether my ISP supports it natively, it depends on the situation. At work, yes—Leibniz Supercomputing Centre supports native IPv6. At home, in theory, the Deutsche Telekom could give IPv6 addresses, but in practice, I am still forced to use a tunnel broker. On the way: I am a lucky UMTS/IPV6-tester—native IPv6 (with NAT64).

Stephane Graber:
Not too surprisingly as the networking guy for Ubuntu, I've been using IPv6 for years now. My home ISP (teksavvy) supports it natively. I have separate IPv4 and IPv6 PPP sessions with a /29 IPv4 subnet and a /56 IPv6 subnet statically assigned to me.

During a normal month, IPv6 typically represents 75% of my traffic (around 800GB), although that's mostly explained because of backups of other IPv6-capable machines and because of Google who's been supporting IPv6 on all its services for a while now.

I've occasionally had issues with my IPv4 connectivity ending up in a few hours of surfing in the IPv6-only world, which is surprisingly usable so long as you mostly use Google, Wikipedia and things like the Debian servers that are all dual-stack.

I also have native IPv6 for my hosted server in Germany (hetzner), so all my server services are dual-stack.

For some other networks I managed (family setup) where native IPv6 isn't available yet, I'm using HE.net IPv6 tunnels. Those are very reliable, free and often provide a shorter (number of hops and latency) path between two hosts than going through the standard IPv4 route.

So in short, I've got IPv6 everywhere and try to make sure I never put something new on-line that's not IPv6-capable and that I port or retire anything that's already on-line and that's IPv4-only.

There are a few things that people will need to get used to to get good IPv6 connectivity though, like making sure ICMP packets aren't blocked in or out and that the MTU/PMTU is set properly on all machines. But that's really where most of the pain has been for me. The rest tends to just work. Now it's just a matter of waiting for the rest of the world to catch up!

PS: When will we have **http://www.linuxjournal.com** on IPv6?

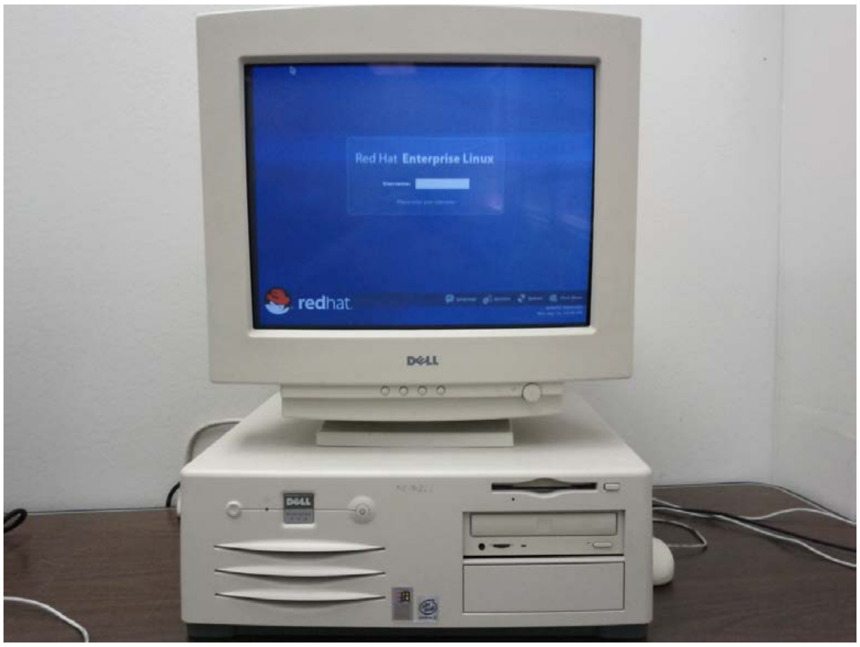
Stephane, it's awesome to hear how much you're able to use IPv6 on a daily basis! Hopefully we'll be able to get our Web site on IPv6 soon. We'll be an interesting litmus test, as Linux Journal *is staffed by brilliant folks, but is hosted just like any other company. Once it's easy for a nerdy company like us to provide IPv6 connectivity, other companies should be able to follow fairly quickly! (PS: On a personal note, thank you for all you've done for LTSP over the years—I was one of the sysadmins in the trenches for years making LTSP work for students, and you're one of my heroes!)—Shawn Powers*

Tiny Tina:
I've been a longtime user of IPv6 via Hurricane Electric's TunnelBroker.net. I use multiple subnets (from a /48) for my internal networks. Unfortunately, my ISP (Suddenlink) is still in the testing phase and has not yet rolled out native dual-stack. My previous ISP (Comcast) supported IPv6 natively, and it was enjoyable.

Tiny Tina, it's interesting that Comcast provided you native IPv6. I recently had it as an ISP and was told very clearly IPv6 was not available. Perhaps it's a regional thing.—Shawn Powers

Photo of the Month

This Web server was retired last year. In the picture is a Dell Precision based on PIII running Red Hat. It hosted several Web sites at the Texas A&M University for years, and has been replaced by a new server-grade Dell running Ubuntu 12.04 LTS.
—Tomo Popovic



WRITE *LJ* A LETTER We love hearing from our readers. Please send us your comments and feedback via **http://www.linuxjournal.com/contact**.

PHOTO OF THE MONTH
Remember, send your Linux-related photos to ljeditor@linuxjournal.com!

LINUX JOURNAL

At Your Service

SUBSCRIPTIONS: *Linux Journal* is available in a variety of digital formats, including PDF, .epub, .mobi and an on-line digital edition, as well as apps for iOS and Android devices. Renewing your subscription, changing your e-mail address for issue delivery, paying your invoice, viewing your account details or other subscription inquiries can be done instantly on-line: **http://www.linuxjournal.com/subs**. E-mail us at subs@linuxjournal.com or reach us via postal mail at *Linux Journal*, PO Box 980985, Houston, TX 77098 USA. Please remember to include your complete name and address when contacting us.

ACCESSING THE DIGITAL ARCHIVE: Your monthly download notifications will have links to the various formats and to the digital archive. To access the digital archive at any time, log in at **http://www.linuxjournal.com/digital**.

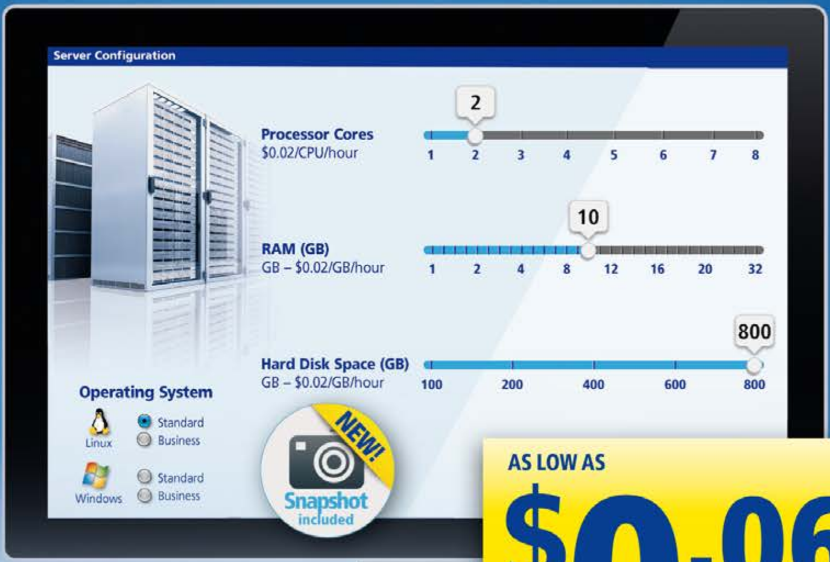
LETTERS TO THE EDITOR: We welcome your letters and encourage you to submit them at **http://www.linuxjournal.com/contact** or mail them to *Linux Journal, PO Box 980985,* Houston, TX 77098 USA. Letters may be edited for space and clarity.

WRITING FOR US: We always are looking for contributed articles, tutorials and real-world stories for the magazine. An author's guide, a list of topics and due dates can be found on-line: **http://www.linuxjournal.com/author**.

FREE e-NEWSLETTERS: *Linux Journal* editors publish newsletters on both a weekly and monthly basis. Receive late-breaking news, technical tips and tricks, an inside look at upcoming issues and links to in-depth stories featured on **http://www.linuxjournal.com**. Subscribe for free today: **http://www.linuxjournal.com/enewsletters**.

ADVERTISING: *Linux Journal* is a great resource for readers and advertisers alike. Request a media kit, view our current editorial calendar and advertising due dates, or learn more about other advertising and marketing opportunities by visiting us on-line: **http://ww.linuxjournal.com/advertising**. Contact us directly for further information: ads@linuxjournal.com or +1 713-344-1956 ext. 2.

1&1 DYNAMIC CLOUD SERVER
PRICE CONTROL



Server Configuration

Processor Cores $0.02/CPU/hour	1	2	3	4	5	6	7	8
RAM (GB) GB – $0.02/GB/hour	1	2	4	8	12	16	20	32
Hard Disk Space (GB) GB – $0.02/GB/hour	100	200	400	600	800			

Processor Cores: 2
RAM (GB): 10
Hard Disk Space (GB): 800

Operating System

Linux: [✓ Standard] Business

Windows: Standard Business

AS LOW AS **$0.06** PER HOUR* PLUS $60 OFF FIRST MONTH

COMPLETE COST CONTROL

- **Full transparency** with accurate hourly billing
- **Parallels® Plesk Panel 11 included,** with unlimited domains

FULL ROOT ACCESS

- The complete functionality of a root server with dedicated resources

MAXIMUM FLEXIBILITY

- Configure the Processor Cores, RAM and Hard Disk Space
- Add up to 99 virtual machines

FAIL-SAFE SECURITY

- Redundant storage and mirrored processing units reliably protect your server

OPTERON PROCESSOR AMD | SUSE | Parallels Plesk Panel

1&1

MEMBER OF united internet

DOMAINS | E-MAIL | WEB HOSTING | eCOMMERCE | SERVERS

Call **1 (877) 461-2631** or buy online

1and1.com

* Other terms and conditions may apply. Visit www.1and1.com for full promotional offer details. Customer is billed monthly for minimum configuration ($0.06/ hour * 720 hours = $43.20/ month minimum). A $60 credit (valid only for the first month) will be applied to your first month of service. Program and pricing specifications and availability subject to change without notice. 1&1 and the 1&1 logo are trademarks of 1&1 Internet, all other trademarks are the property of their respective owners. © 2013 1&1 Internet. All rights reserved.

diff -u
WHAT'S NEW IN KERNEL DEVELOPMENT

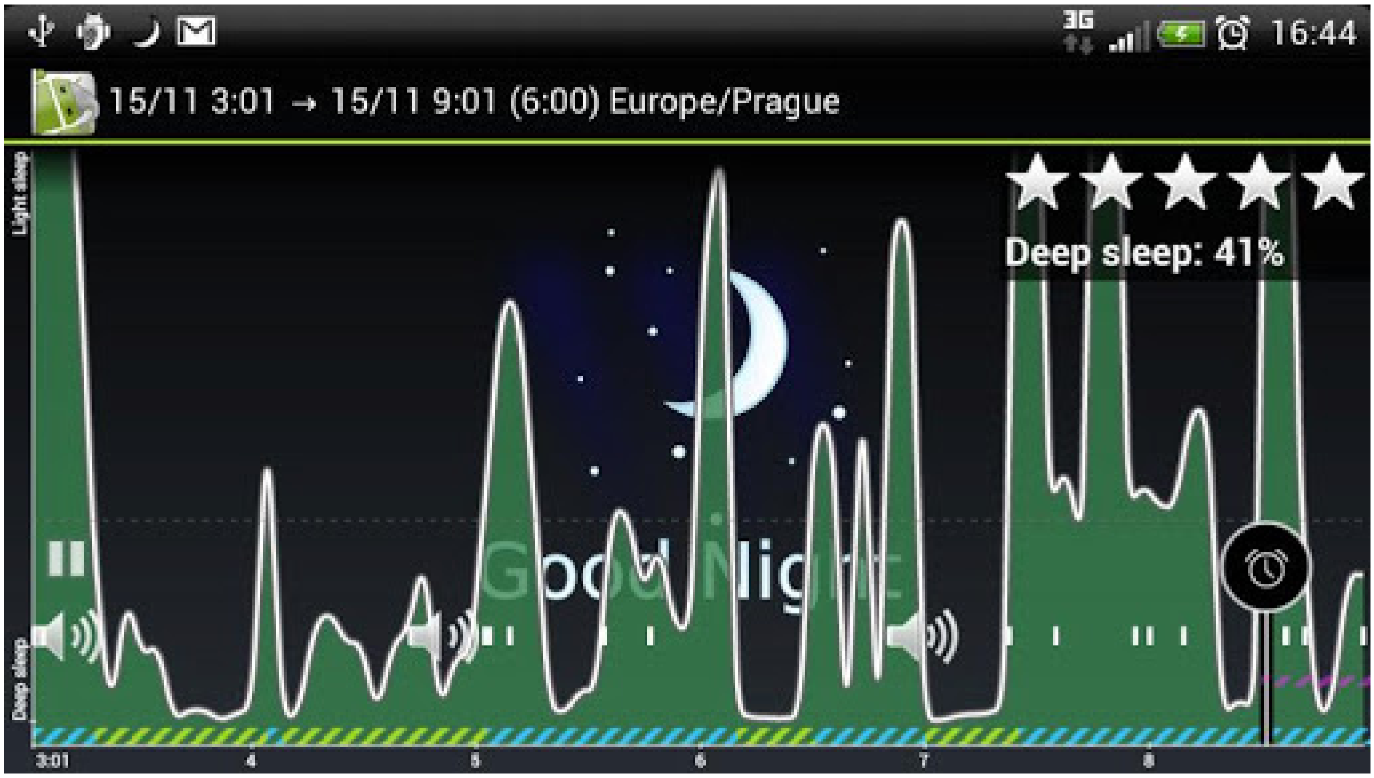
Tejun Heo has replaced **Jeff Garzik** as the official **SATA** maintainer. Jeff, who's been contributing to Linux since the early days, announced that he's moving away from kernel development altogether and is going to work on other open-source projects. I'm sure they'll be better for his interest.

Sarah Sharp has made a number of updates to the **REPORTING-BUGS** document, cleaning it up and making it easier for frustrated and stressed bug victims to follow. Along with other details, the new text clarifies exactly what information to include, whom to contact and when, and how long to wait before pinging the person again.

Ahmad Sharif, one of the **ChromeOS** developers, has announced **Quipper**, a new open-source library that creates **perf** reports. In fact, in addition to presenting information, it converts perf data to **Google's protobuf** format, which Ahmad argues is more stable and flexible than perf's native data format. The whole point of Quipper, and perf, is to help developers figure out how to make their code run faster.

Tim Bird recently asked about the best way for a **GPS** receiver to interface with the kernel. His company, **Sony**, was interested in writing up some Linux support for its GPS devices. However, as **Greg Kroah-Hartman** pointed out, this is not an easy question to answer, thanks to the diversity of hardware implementations and the tendency of some implementations to masquerade as others. In Sony's case, the best solution would depend on the hardware involved. So the real answer would require Sony to divulge the details of its hardware, which it may not be ready to do.

Simon Glass has submitted some patches to implement the **ChromeOS Embedded Controller**. This code would allow **Chromebooks** to charge their batteries, read keyboard input and do other tasks related to making the thing go. **Samuel Ortiz** accepted the patches without needing too many changes. ChromeOS, like **Android**, seems well accepted into kernel architecture.—**ZACK BROWN**



Screenshot from the Google Play Store

Sleep as Android

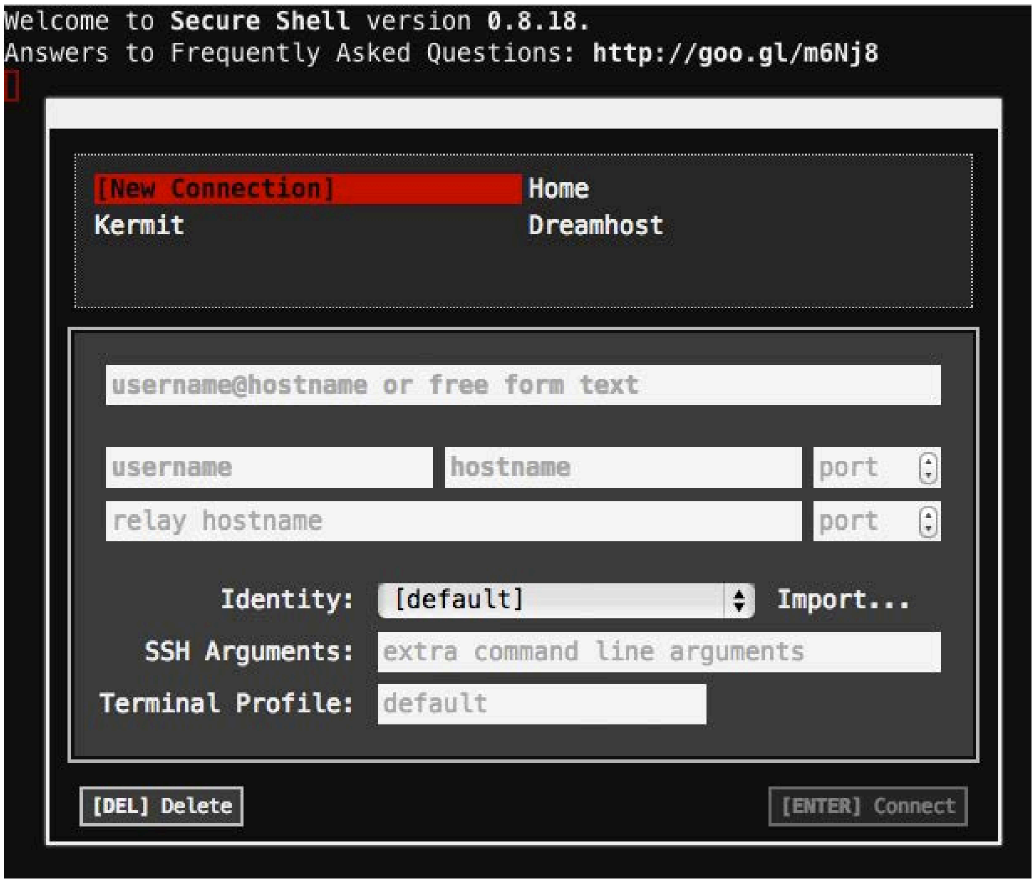
I sleep poorly. In fact, insomnia has plagued me for years. As it turns out, even when I think I'm sleeping well, I'm usually not. There's nothing worse than a shoddy night's sleep followed by an abrupt alarm going off when you've finally settled into a deep slumber.

Enter: Sleep as Android.

By using the accelerometer in your phone, Sleep as Android not only tracks how restful your sleep was during the night, but it also determines the best moment to wake you up. Rather than setting an exact time to wake up, you enter a "time range" in which you'd like to be woken. Sleep as Android looks for a time during that range where you're sleeping lightly (or not at all) and goes off. More times than not, it means waking up feeling refreshed, as opposed to my normal routine: groggy and coffee-starved. If you have trouble sleeping, or just prefer to wake up feeling refreshed, give Sleep as Android a try. It's free for two weeks, then only $2.99 if you want to keep it. I've listed only some of the features here, but check it out, it's truly an awesome program: **https://sites.google.com/site/sleepasandroid/home**.

—SHAWN POWERS

There's Browser in My SSH



```
Welcome to Secure Shell version 0.8.18.
Answers to Frequently Asked Questions: http://goo.gl/m6Nj8
```

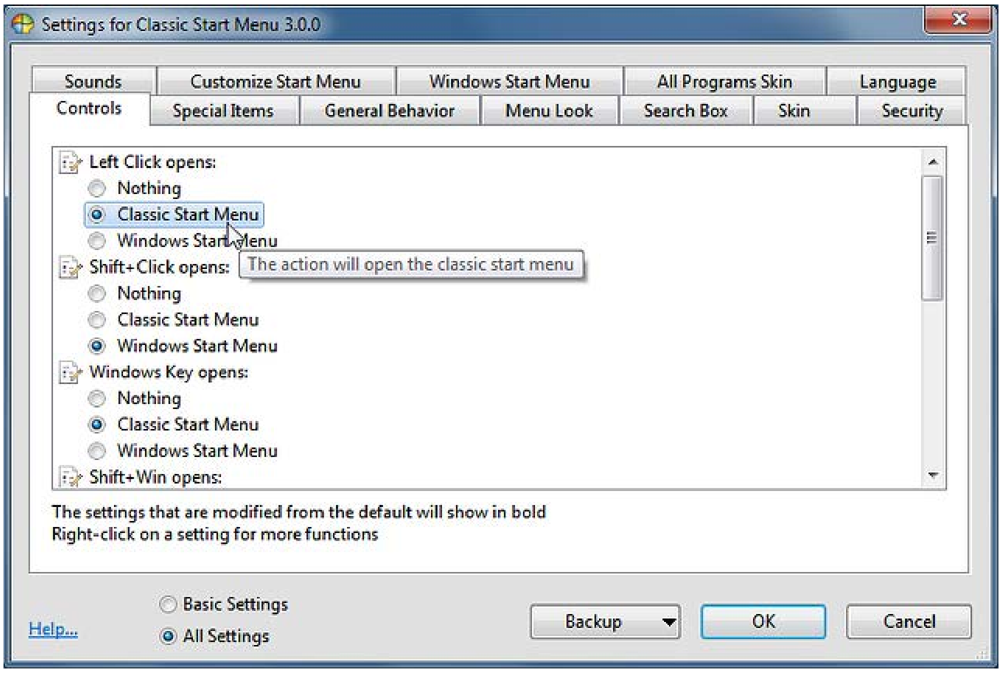
Figure 1. It's simple. It's texty. It's awesome.

No, there's SSH in my browser! Although it may not be as logical of a combination as chocolate and peanut butter, for Chromebook users, an HTML5 SSH client is pretty amazing. Granted, Google's "crosh" shell has SSH abilities, but it's a very limited implementation. With the Chrome extension "Secure Shell", it's easy to SSH in to remote servers and interact like a traditional terminal window—mostly.

Secure Shell is getting better all the time, and at the time of this writing, it supports port forwarding, logging in with keys, socks proxying and even many keyboard shortcuts for programs like Irssi. The keyboard shortcut support isn't perfect, but for me at least, it's manageable.

If you're a Chromebook user and want a real SSH client, give the "Secure Shell" extension a try. Heck, regardless of the OS you're using (I'm looking at you, Windows), it's a fast way to get a secure connection. It's being developed by Google, and it's free via the Play Store: http://goo.gl/irXYG.—**SHAWN POWERS**

Non-Linux FOSS: Classic Shell



Left Click opens: Nothing [✓ Classic Start Menu] Windows Start Menu

The action will open the classic start menu

Shift+Click opens: Nothing Classic Start Menu [✓ Windows Start Menu]

Windows Key opens: Nothing [✓ Classic Start Menu] Windows Start Menu

Shift+Win opens:

The settings that are modified from the default will show in bold
Right-click on a setting for more functions

Basic Settings [✓ All Settings]

Figure 1. The configuration options offer more features than Microsoft ever offered in its Start menu (screenshot from http://www.classicshell.net).

Even those of us on the Linux side of the fence have been watching Microsoft's Windows 8 roll-out—albeit for us, it has been with morbid fascination. Granted, we're not without our drastic changes (ahem, Unity), but the new interface Microsoft has chosen for version 8 is seemingly unusable for most people. The iconic Start menu has been taken away without a clear replacement.

If you're stuck on a Windows 8 machine and wish you had a Start menu, fear not; the Open Source community is here to help. With Classic Shell, not only can you get a Start menu back, but you also can choose what it looks like and configure every aspect of the menu system you can imagine. Prefer the Windows XP look? No problem. Is Windows 7 your idea of the perfect Windows experience? No problem there either.

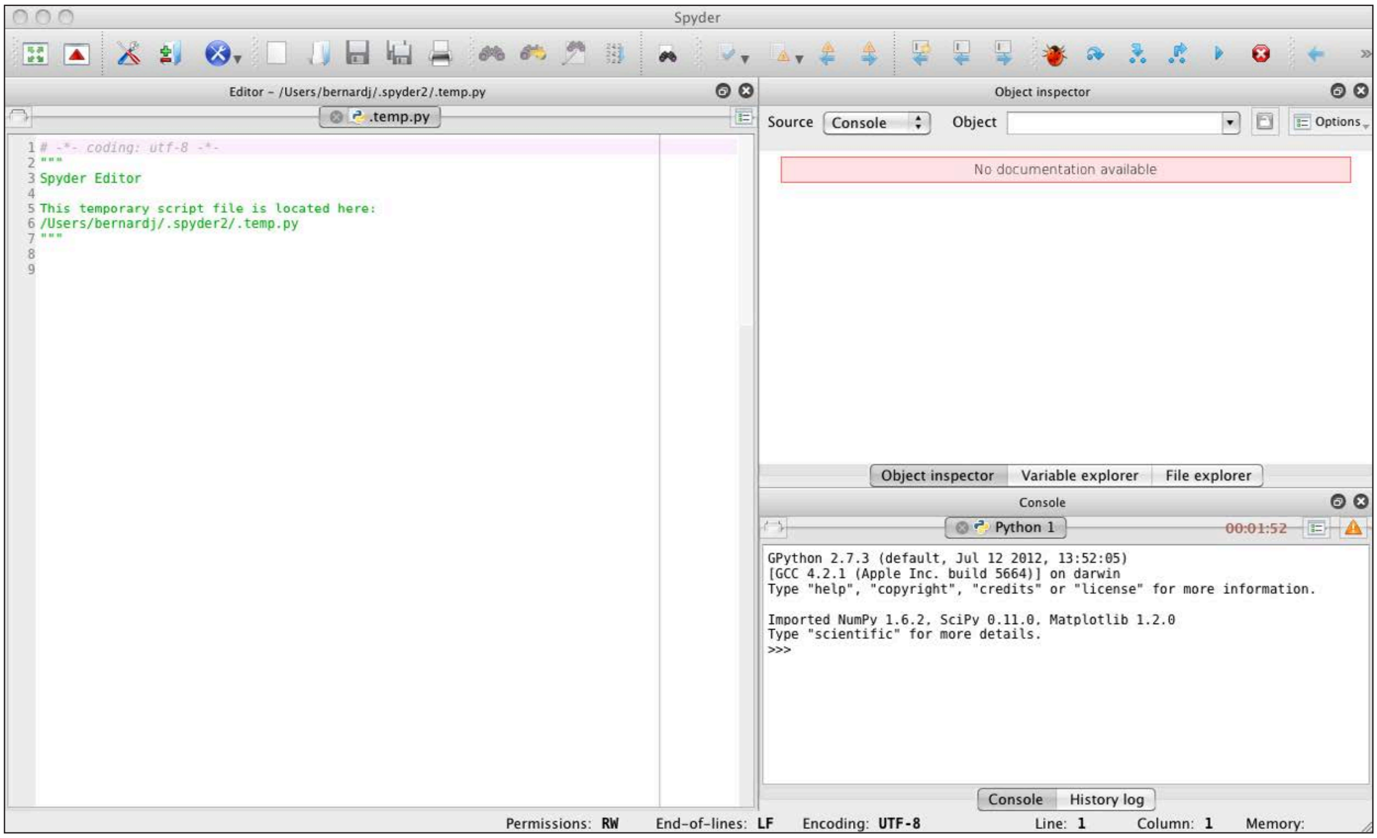
Classic Shell is an open-source project originally developed back in the Windows Vista days. Thankfully, it's been updated for the Windows 8 platform, and thanks to Microsoft's design decisions, it's more useful than ever! Grab yourself a copy today at **http://www.classicshell.net**.

—**SHAWN POWERS**

Developing Your Own Scientific Python Code

In many cases, scientific research takes you into totally new areas of knowledge, never before explored by others. This means the computational work you need to do may be totally new as well. Although typically such code development still happens in C or FORTRAN, Python is growing in popularity. This is especially true in physics.

If you are just working on a small code base, a basic text editor is fine; however, once your project reaches a certain size, using a proper IDE is a huge advantage. Luckily, an open-source project seeks to fill this exact niche: Spyder. Spyder is available for Windows, Mac OS X and Linux. You should be able to find a package for your



```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This temporary script file is located here:
/Users/bernardj/.spyder2/.temp.py
"""
```

No documentation available

```
GPython 2.7.3 (default, Jul 12 2012, 13:52:05)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.

Imported NumPy 1.6.2, SciPy 0.11.0, Matplotlib 1.2.0
Type "scientific" for more details.
>>>
```

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 1 Column: 1 Memory:

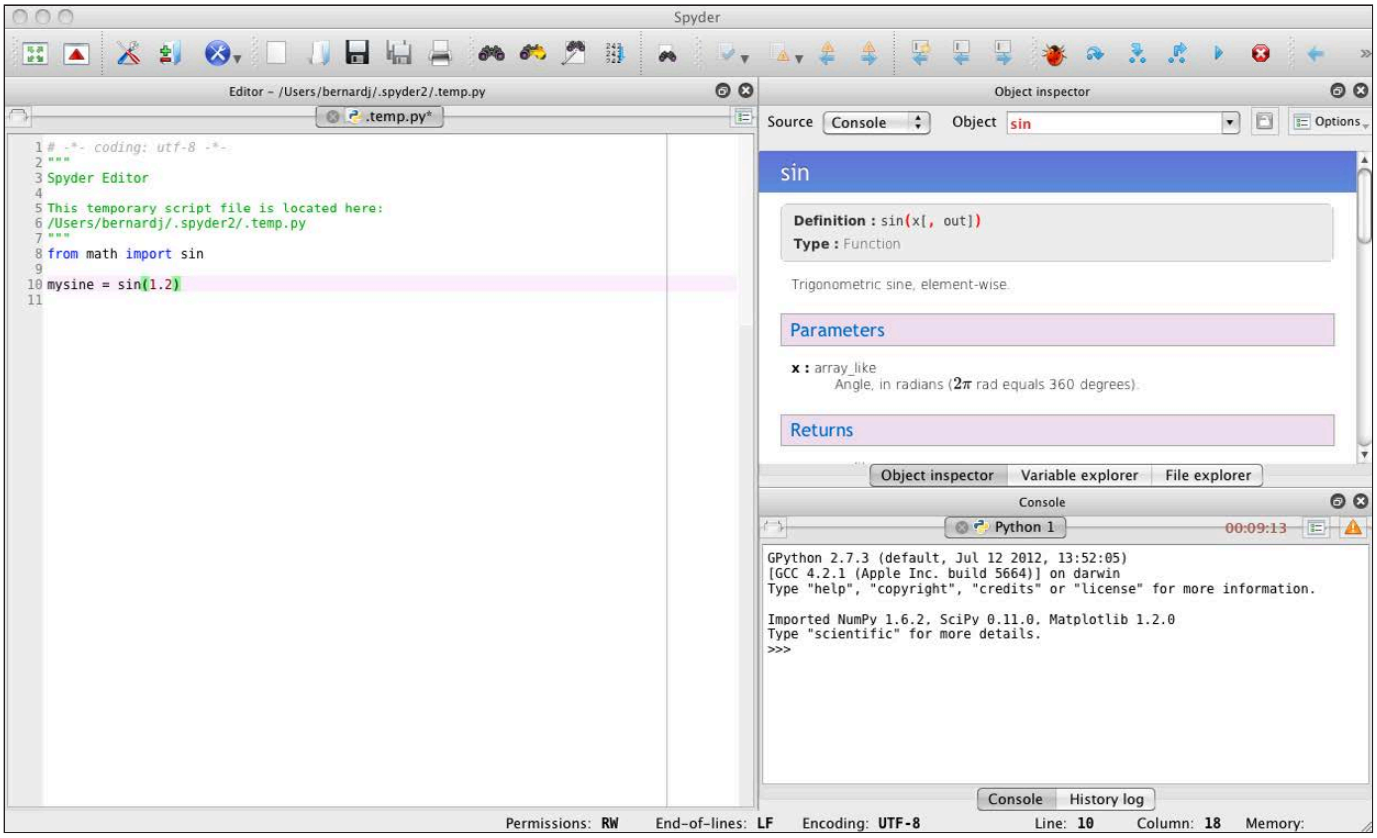
Figure 1. When you start Spyder, several information panes along with the main editor pane appear.

distribution, but if not, you always can grab the binaries or source code at the main Spyder Web site (**http://code.google.com/p/spyderlib**).

Spyder actually is written in Python, and it has been designed with a plugin architecture. This means you can add extra functionality by installing plugins. If you can't find a plugin for the functionality you need, you always can write your own.

When you start Spyder, several panes open and a temporary Python script that you can use to start editing appears. The main pane is the editor, where the temporary script is loaded and ready for you to start working. The right-hand side of the window is broken into two more panes. The bottom pane is the console to a running Python interpreter. Here you can see that Spyder automatically loads NumPy, SciPy and matplotlib on startup, so you already have most of the tools you likely will need ready to go.

You can use this console just as you would any other Python interpreter. The top pane has multiple tabs. The first tab that opens at start up is an



```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This temporary script file is located here:
/Users/bernardj/.spyder2/.temp.py
"""
from math import sin

mysine = sin(1.2)
```

sin

Definition : sin(x[, out])
Type : Function

Trigonometric sine, element-wise.

Parameters

x : array_like
Angle, in radians (2π rad equals 360 degrees).

Returns

```
GPython 2.7.3 (default, Jul 12 2012, 13:52:05)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.

Imported NumPy 1.6.2, SciPy 0.11.0, Matplotlib 1.2.0
Type "scientific" for more details.
>>>
```

Permissions: **RW** End-of-lines: **LF** Encoding: **UTF-8** Line: **10** Column: **18** Memory:

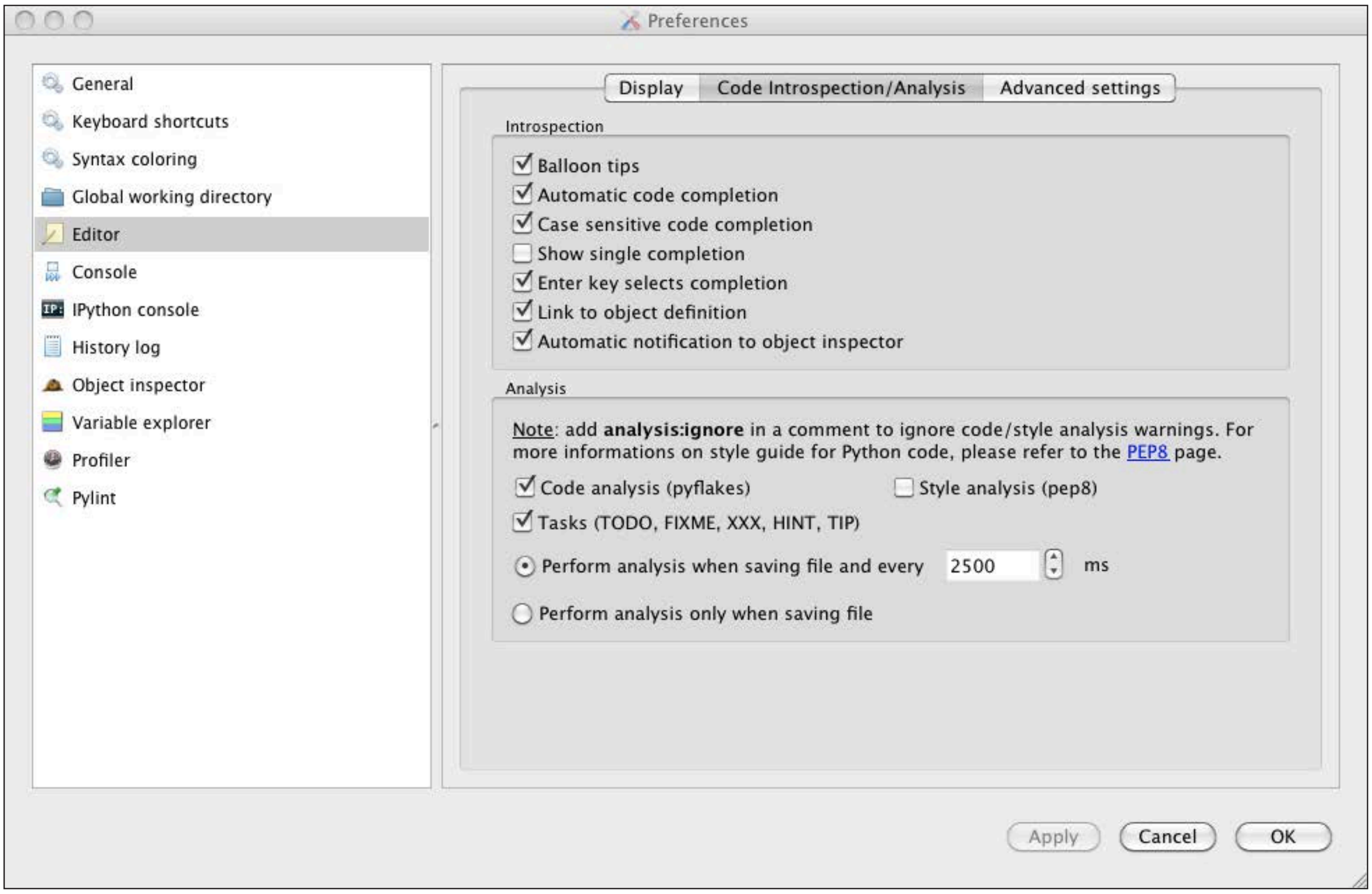
Figure 2. The object inspector lets you look at the details of any objects you may want to use.

object inspector. This tab lets you look at the details of any objects being used in your code. The other tabs available are a variable inspector and a file explorer.

Let's begin by looking at the main editor pane. Like any other programming editor, Spyder provides full-color highlighting of Python syntax. Rope is used to provide code introspection capabilities to the editor. If you start typing a function call, Spyder makes suggestions for code completion.

Pyflakes provides on-the-fly code analysis. Any errors in your code are highlighted right away with a triangle symbol in the margin. When you hover over it, details for the error are displayed in a pop-up window. You also can set breakpoints in the editor that are used by the Python debugger. This way, you have a bit of control over how your code runs when you run it under the pdb debugger.

The console pane provides a full set of tools for controlling multiple interpreters. When you run a script from the editor, you have the option of starting up a new interpreter



Introspection

- [✓] Balloon tips
- [✓] Automatic code completion
- [✓] Case sensitive code completion
- [ ] Show single completion
- [✓] Enter key selects completion
- [✓] Link to object definition
- [✓] Automatic notification to object inspector

Analysis

Note: add **analysis:ignore** in a comment to ignore code/style analysis warnings. For more informations on style guide for Python code, please refer to the PEP8 page.

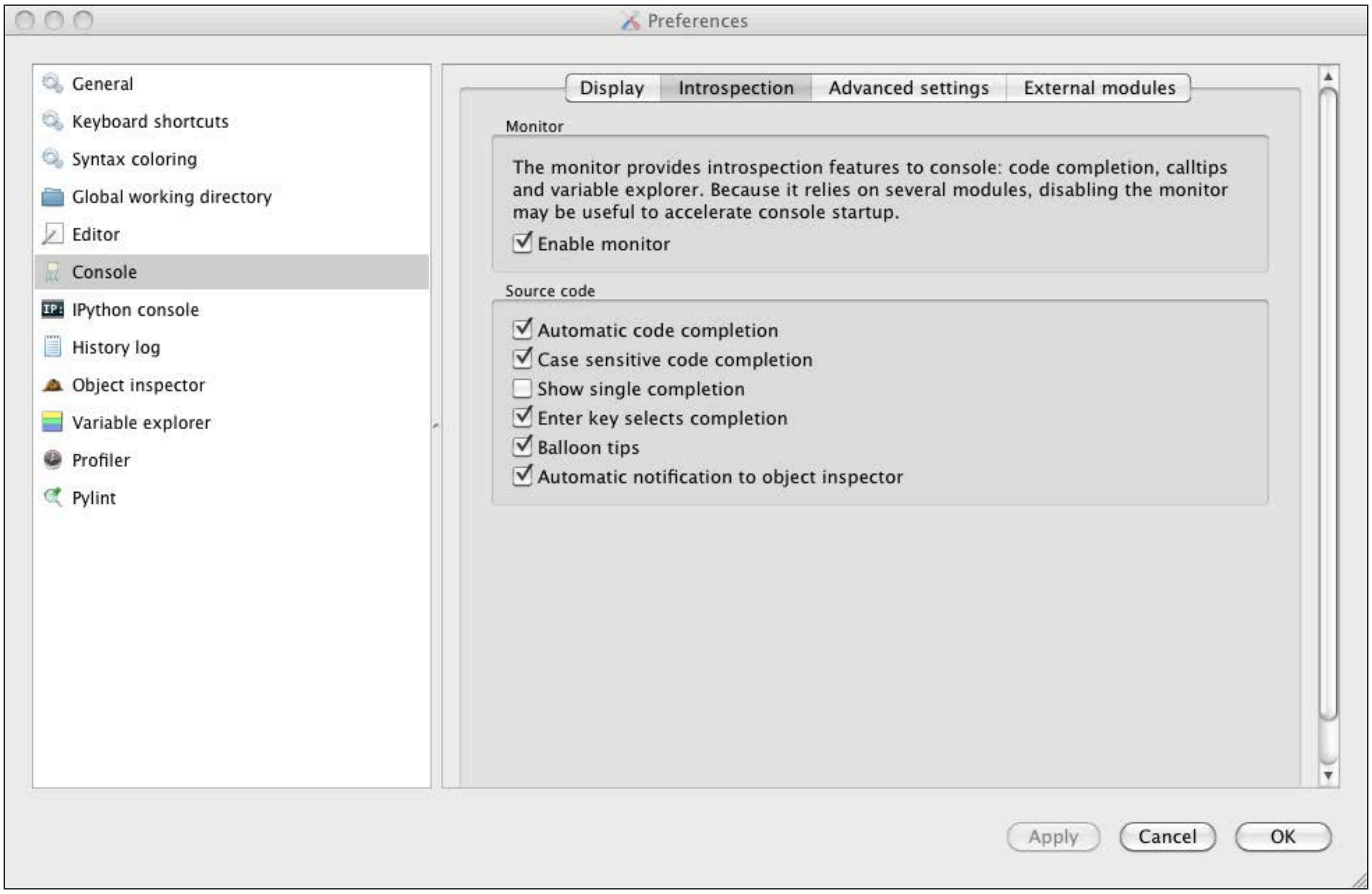
- [✓] Code analysis (pyflakes)
- [ ] Style analysis (pep8)
- [✓] Tasks (TODO, FIXME, XXX, HINT, TIP)
- (●) Perform analysis when saving file and every 2500 ms
- ( ) Perform analysis only when saving file

Figure 3. You can set all of the options for the editor window in the preferences section.

in which to run it. Or, you can run it in an existing interpreter. You can set this behavior in Spyder's preferences. You also simply can create a new interpreter from the console pane directly.

Also interesting is that you can create IPython interpreters in the console. Then, you have all of the extra functionality provided by IPython at your disposal. One of these advanced features is the ability to use multiple IPython engines in parallel. From the console in Spyder, you can create these IPython engines that will run in the background, ready to be used for whatever parallel processing you may have in mind. So, not only can you develop your new scientific code as a parallel program, you also can work with it directly from Spyder, in parallel. All of these extra interpreters and engines run as separate processes, which means they will not affect Spyder itself or cause it to hang if something bad happens within one of the Python interpreters.

One of the extra tabs in the


Preferences

General, Keyboard shortcuts, Syntax coloring, Global working directory, Editor, Console, IPython console, History log, Object inspector, Variable explorer, Profiler, Pylint

Display | Introspection | Advanced settings | External modules

Monitor

The monitor provides introspection features to console: code completion, calltips and variable explorer. Because it relies on several modules, disabling the monitor may be useful to accelerate console startup.

☑ Enable monitor

Source code

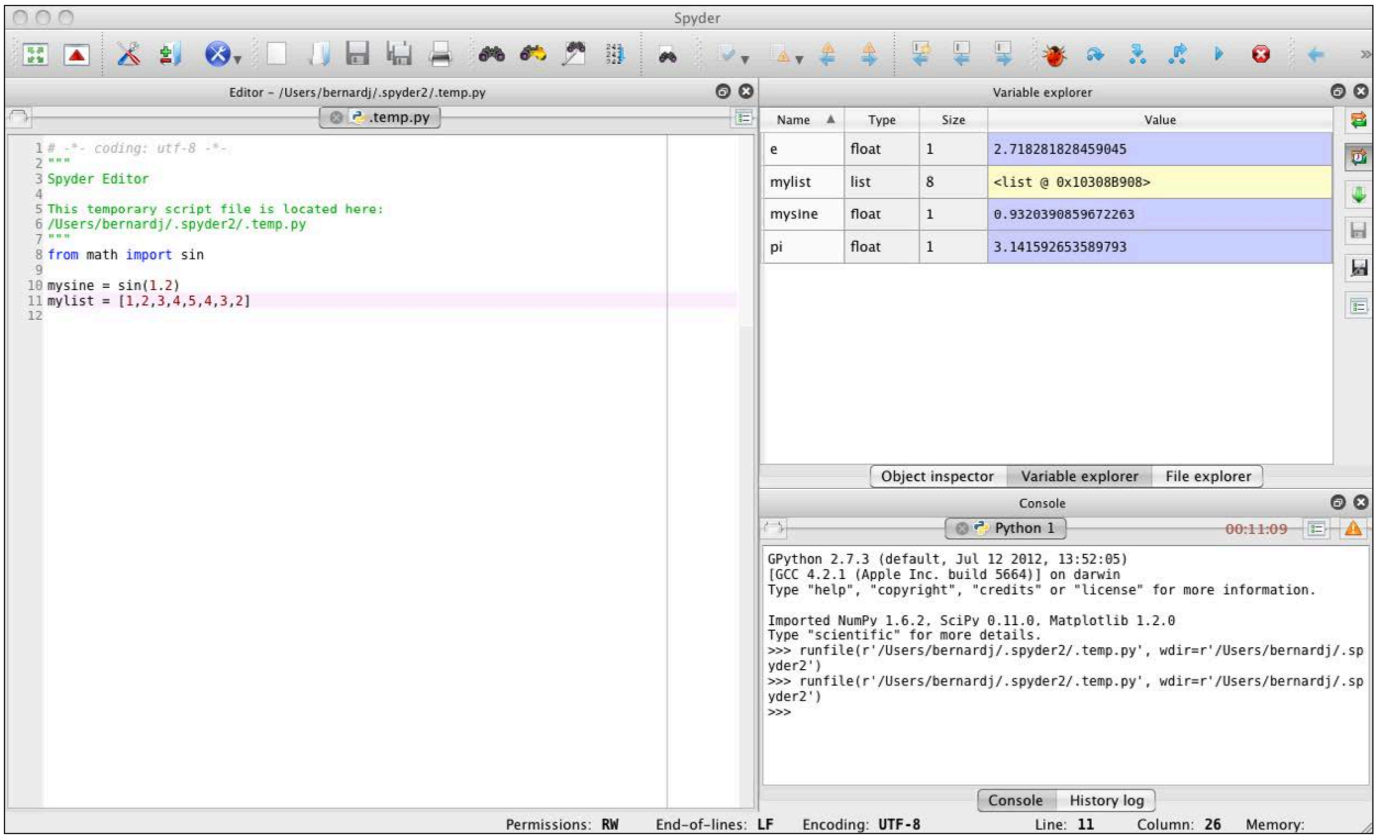
☑ Automatic code completion
☑ Case sensitive code completion
☐ Show single completion
☑ Enter key selects completion
☑ Balloon tips
☑ Automatic notification to object inspector

Apply | Cancel | OK

Figure 4. The preferences for the console section control things like whether it is monitored and so forth.

The variable explorer can handle all of the standard data types, like strings, integers and floats. It also includes an array editor that can be used to edit lists and tuples.

top-right pane brings up the variable explorer. This pane gives you a list of all the variables currently active in the memory space within Spyder. It shows the name, type, size and value for each of the variables accessible in the global namespace. This applies to both the internal Python interpreter and any external ones. The variable explorer can handle all of the standard data types, like strings, integers and floats. It also includes an array editor that can be used to edit lists and tuples. The array editor provides a nice environment for editing complex data types.



```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This temporary script file is located here:
/Users/bernardj/.spyder2/.temp.py
"""
from math import sin

mysine = sin(1.2)
mylist = [1,2,3,4,5,4,3,2]
```

Name	Type	Size	Value
e	float	1	2.718281828459045
mylist	list	8	<list @ 0x10308B908>
mysine	float	1	0.9320390859672263
pi	float	1	3.141592653589793

```
GPython 2.7.3 (default, Jul 12 2012, 13:52:05)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.

Imported NumPy 1.6.2, SciPy 0.11.0, Matplotlib 1.2.0
Type "scientific" for more details.
>>> runfile(r'/Users/bernardj/.spyder2/.temp.py', wdir=r'/Users/bernardj/.sp
yder2')
>>> runfile(r'/Users/bernardj/.spyder2/.temp.py', wdir=r'/Users/bernardj/.sp
yder2')
>>>
```

Figure 5. The variable explorer gives you a list of everything within the global namespace.

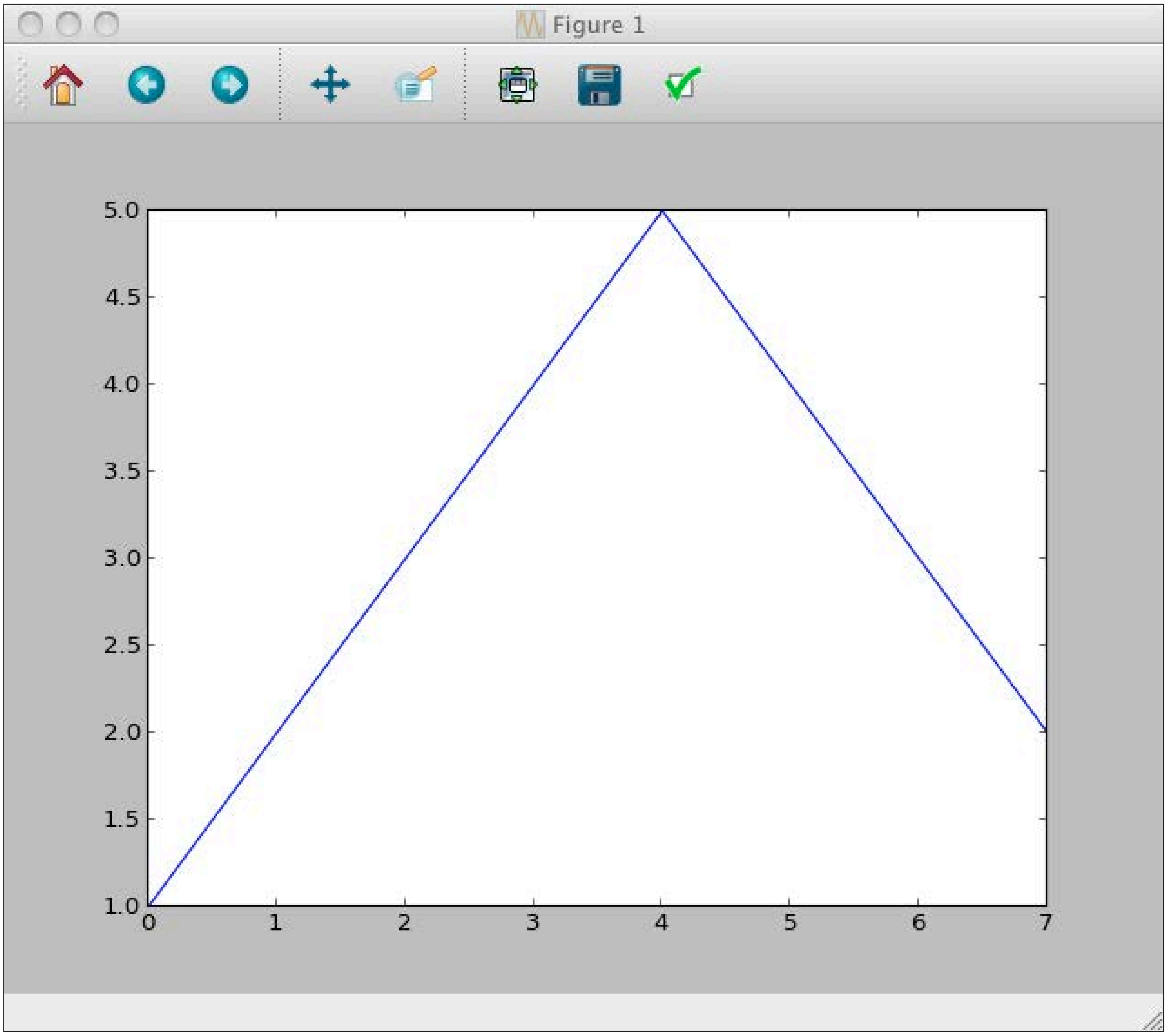
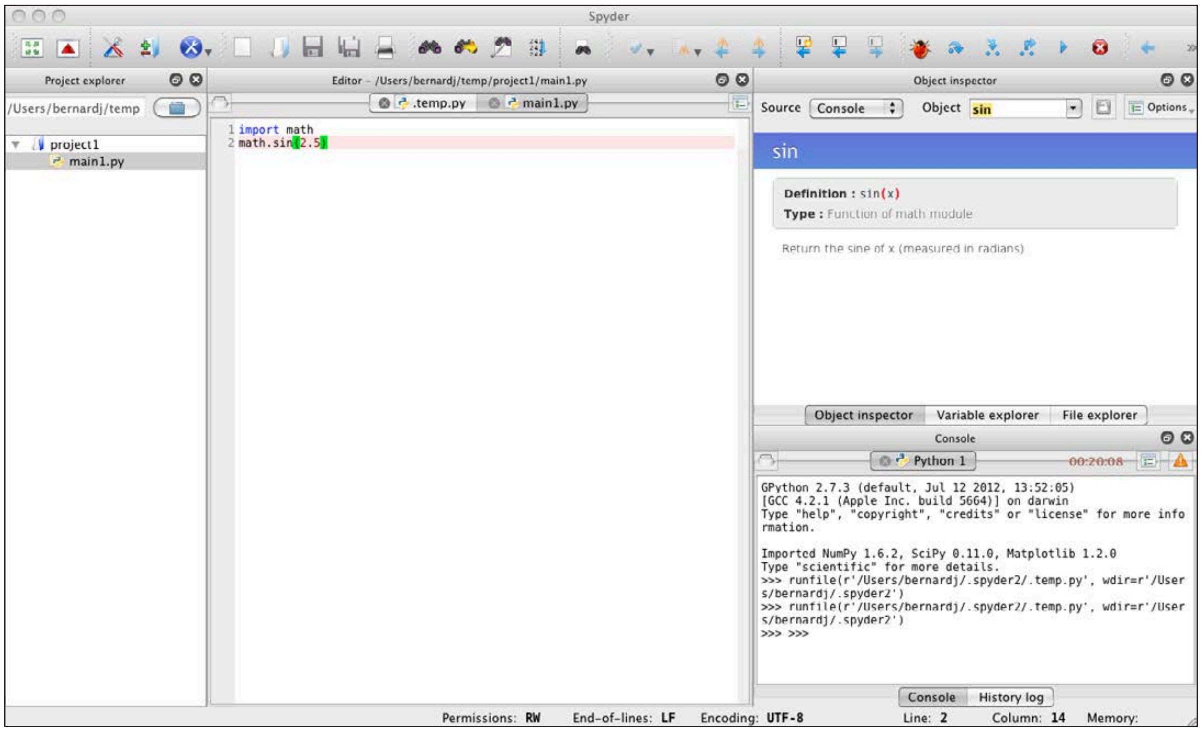


Figure 6. Spyder lets you plot the data within a list or tuple.

Spyder provides even more tools though. By right-clicking on a list or tuple, you can do some basic data analysis. Spyder allows you to plot the values in the data object to see what it looks like. Or, you can look at a histogram of the values if a statistical analysis would make more sense.

All of these tools are handy, but by themselves, they aren't enough if you are developing a large code base. In such cases, you will need some kind of project-level organization. Spyder can help in that situation too. You can create a project in order to encapsulate a group of files as a



```
import math
math.sin(2.5)
```

sin

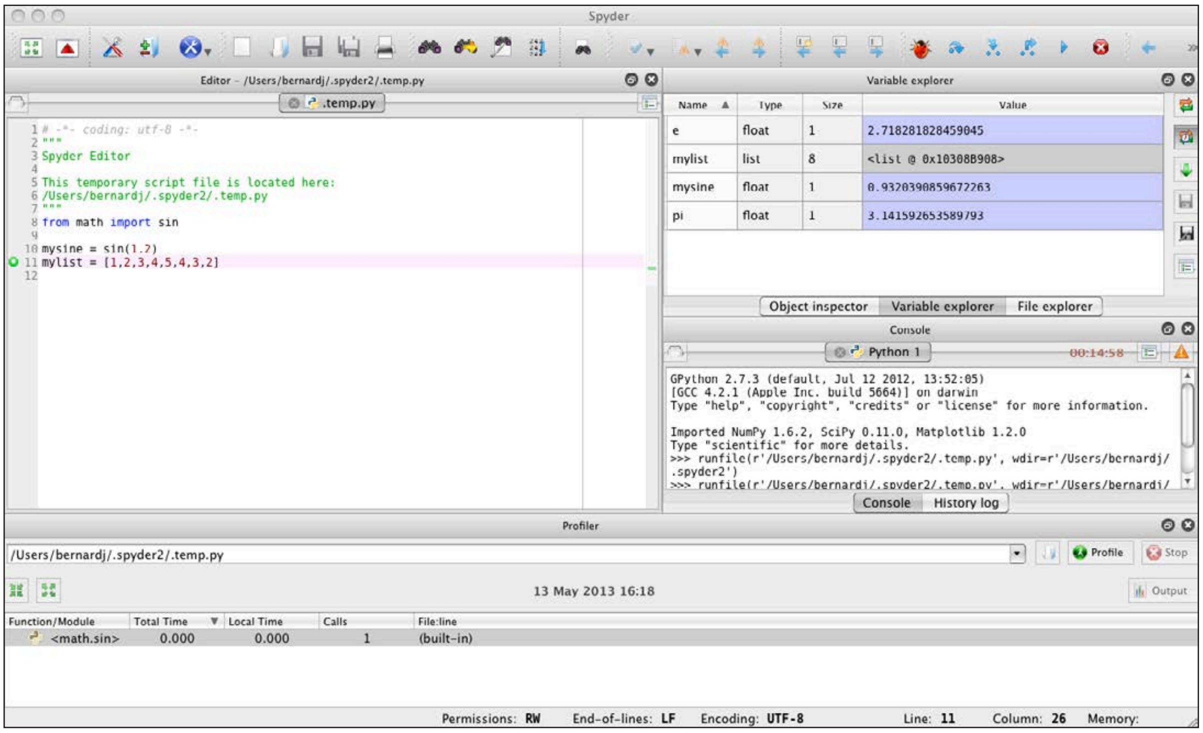
Definition : sin(x)
Type : Function of math module

Return the sine of x (measured in radians).

```
GPython 2.7.3 (default, Jul 12 2012, 13:52:05)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more info
rmation.

Imported NumPy 1.6.2, SciPy 0.11.0, Matplotlib 1.2.0
Type "scientific" for more details.
>>> runfile(r'/Users/bernardj/.spyder2/.temp.py', wdir=r'/User
s/bernardj/.spyder2')
>>> runfile(r'/Users/bernardj/.spyder2/.temp.py', wdir=r'/User
s/bernardj/.spyder2')
>>> >>>
```

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 2 Column: 14 Memory:

Figure 7. If you are working on a larger code base, the project explorer will come in very handy.



```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This temporary script file is located here:
/Users/bernardj/.spyder2/.temp.py
"""
from math import sin

mysine = sin(1.2)
mylist = [1,2,3,4,5,4,3,2]

```

Name	Type	Size	Value
e	float	1	2.718281828459045
mylist	list	8	<list @ 0x10308B908>
mysine	float	1	0.9320390859672263
pi	float	1	3.141592653589793

```
GPython 2.7.3 (default, Jul 12 2012, 13:52:05)
[GCC 4.2.1 (Apple Inc. build 5664)] on darwin
Type "help", "copyright", "credits" or "license" for more information.

Imported NumPy 1.6.2, SciPy 0.11.0, Matplotlib 1.2.0
Type "scientific" for more details.
>>> runfile(r'/Users/bernardj/.spyder2/.temp.py', wdir=r'/Users/bernardj/
.spyder2')
>>> runfile(r'/Users/bernardj/.spyder2/.temp.py', wdir=r'/Users/bernardj/
```

/Users/bernardj/.spyder2/.temp.py

13 May 2013 16:18

Function/Module	Total Time	Local Time	Calls	File:line
<math.sin>	0.000	0.000	1	(built-in)

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 11 Column: 26 Memory:

Figure 8. The profiler opens a new pane at the bottom of the window, showing you numbers of calls and time for each function.

single unit. Creating a new project makes a new folder to store all of the associated files. Opening this new project creates a new pane where you can work with the project files. By right-clicking on the project, you can create a new file, folder, module or package. When you create a new file, Spyder opens it up in the editor, ready for you to start working on it.

The last tool you should be aware of is the profiler. The first step in program development is writing code that works. After that, your job is to write code that is as efficient as possible. The rule of thumb is to optimize last, and only what needs to be optimized. But, what part of your program is that? Without reliable measurements, you won't know what needs optimizing. In Spyder, you can click on the menu item Run→Profile or press F10. This runs your code under the Python profiler, giving you a breakdown of where all the time is being spent. Once you have this information, you can focus your energies where they will do the most good.

Hopefully, you can take Spyder and run with it while developing your own scientific Python code. Although lots of IDEs exist for developing code, there aren't very many setups geared toward developing scientific code. With Spyder, you should have a head start in developing your new breakthrough code, solving the problem that could win you the next Nobel Prize.

—JOEY BERNARD

They Said It

Electricity is really just organized lighting.
—George Carlin

I am somewhat exhausted; I wonder how a battery feels when it pours electricity into a non-conductor?
—Arthur Conan Doyle

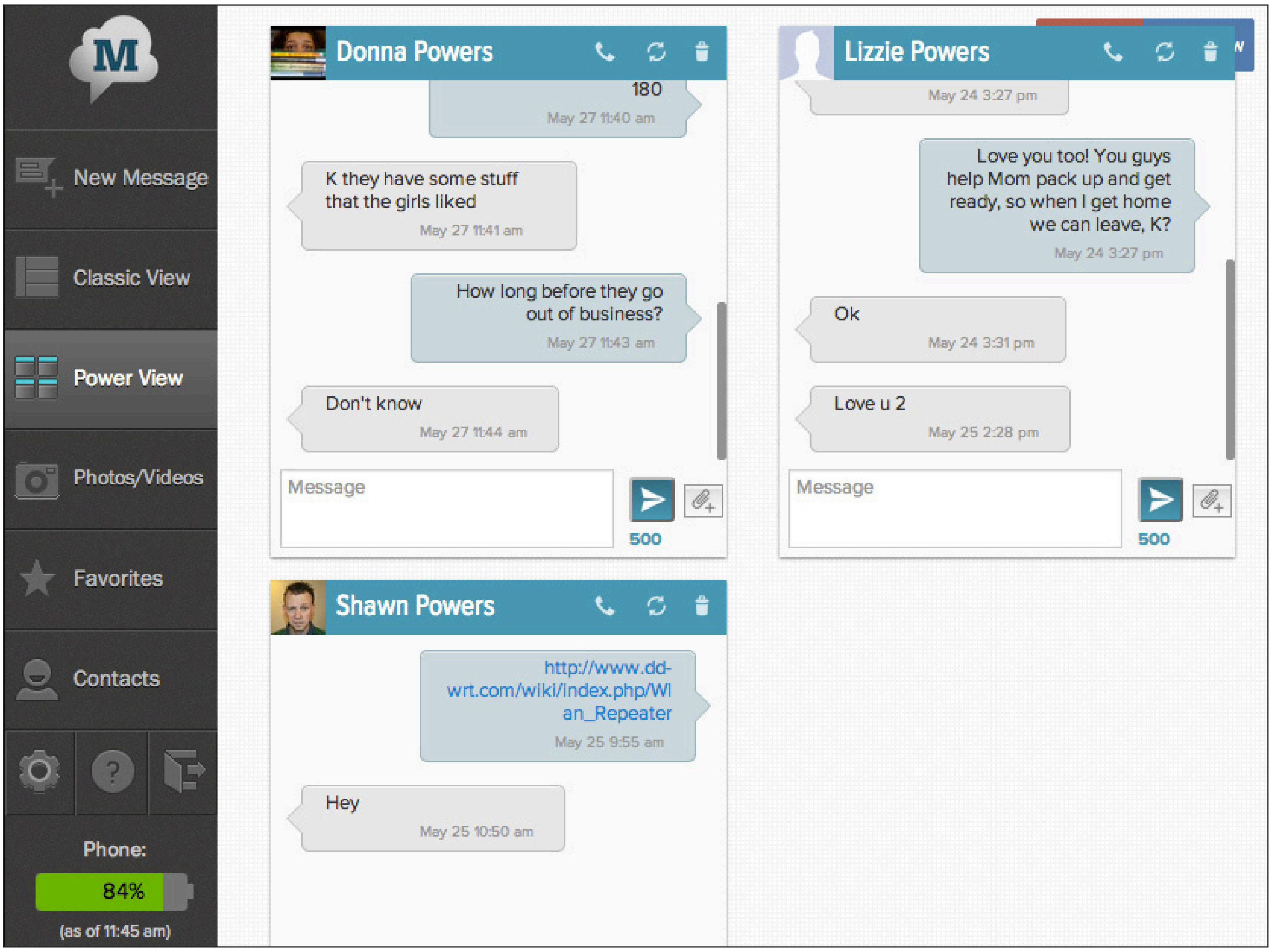
Enthusiasm is the electricity of life. How do you get it? You act enthusiastic until you make it a habit.
—Gordon Parks

There is a force more powerful than steam and electricity: the will.
—Fernán Caballero

Soon now, the faint tinkling of a broken filament will become another sound of another century.
—Jane Brox

Android Candy: MightyText, Mighty Awesome

I'll admit, I've always been impressed with Apple's iMessage program. With its integration into texting, it seamlessly combines instant messaging and SMS into a single communication stream. Whether on an iPhone, iPod, iPad or Macintosh, the messages



Donna Powers:

180 — May 27 11:40 am

K they have some stuff that the girls liked — May 27 11:41 am

How long before they go out of business? — May 27 11:43 am

Don't know — May 27 11:44 am

Lizzie Powers:

May 24 3:27 pm

Love you too! You guys help Mom pack up and get ready, so when I get home we can leave, K? — May 24 3:27 pm

Ok — May 24 3:31 pm

Love u 2 — May 25 2:28 pm

Shawn Powers:

http://www.dd-wrt.com/wiki/index.php/Wian_Repeater — May 25 9:55 am

Hey — May 25 10:50 am

The Web app syncs up when you log in from a computer. It doesn't matter where the discussion started, it's always there!

can be seen and sent to other Apple devices. The only downfall is that you can send only SMS messages to non-Apple phones from your actual iPhone with a texting plan.

MightyText takes the idea of integration in a slightly different, but also slightly more awesome, direction. If you have an Android phone, once you install the Free MightyText application, you can text back and forth from *any* browser on *any* computer, regardless of the operating system! It includes MMS pictures, group texting and all the other cool features that traditionally are available only from a phone.

If you've been wavering back and forth about getting an iPhone or an Android device, MightyText might be the application that seals the deal for you. It was for me. In fact, MightyText is so incredibly useful, we've given it the Editors' Choice award this month. Check it out now at **http://mightytext.net**.

—SHAWN POWERS

Powerful: Rhino

Rhino M4700/M6700
- Dell Precision M4700/M6700 w/ Core i7 Quad (8 core)
- 15.6"-17.3" FHD LED w/ X@1920x1080
- NVidia Quadro K5000M
- 750 GB - 1 TB hard drive
- Up to 32 GB RAM (1866 MHz)
- DVD±RW or Blu-ray
- 802.11a/b/g/n
- Starts at $1190
- E6230, E6330, E6430, E6530 also available

- High performance NVidia 3-D on an FHD RGB/LED
- High performance Core i7 Quad CPUs, 32 GB RAM
- Ultimate configurability — choose your laptop's features
- One year Linux tech support — phone and email
- Three year manufacturer's on-site warranty
- Choice of pre-installed Linux distribution:

Tablet: Raven

Raven X230/X230 Tablet
- ThinkPad X230/X230 tablet by Lenovo
- 12.1" HD LED w/ X@1366x768
- 2.6-2.9 GHz Core i7
- Up to 16 GB RAM
- 750 GB hard drive / 180 GB SSD
- Pen/finger input to screen, rotation
- Starts at $2050
- T430, T530, W530 also available

Rugged: Tarantula

Tarantula CF-31
- Panasonic Toughbook CF-31
- Fully rugged MIL-SPEC-810G tested: drops, dust, moisture & more
- 13.1" XGA TouchScreen
- 2.4-2.53 GHz Core i5
- Up to 8 GB RAM
- 320-750 GB hard drive / 512 GB SSD
- CF-19, CF-52, CF-H2 also available

EmperorLinux
...where Linux & laptops converge

www.EmperorLinux.com
1-888-651-6686

Model specifications and availability may vary.

REUVEN M. LERNER

Intro to Clojure on the Web

Create Web applications simply with Clojure, a modern Lisp.

Lisp is one of those languages that people either love or hate. Count me among the Lisp lovers. I was brainwashed during my undergraduate studies at MIT to believe that Lisp is the only "real" programming language out there, and that anything else is a pale imitation. True, I use Python and Ruby in my day-to-day work, but I often wish I had the chance to work with Lisp on a regular basis.

One window of opportunity to do exactly that has opened in the past few years. Clojure, a modern variant of Lisp that runs on the Java virtual machine (JVM), has been taking the programming world by storm. It's a real Lisp, which means that it has all of the goodness you would want and expect: functional programming paradigms, easy use of complex data structures and even such advanced facilities as macros. Unlike other Lisps, and contributing in no small part to its success, Clojure sits on top of the JVM, meaning that it can interoperate with Java objects and also work in many existing environments.

In this article, I want to share some of my experiences with starting to experiment with Clojure for Web development. Although I don't foresee using Clojure in much of my professional work, I do believe it's useful and important always to be trying new languages, frameworks and paradigms. Clojure combines Lisp and the JVM in just the right quantities to make it somewhat mainstream, which makes it more interesting than just a cool language that no one is really using for anything practical.

Clojure Basics

Clojure, as I mentioned above, is a version of Lisp that's based on the JVM. This means if you're going to run Clojure programs, you're also going to need a copy of Java. Fortunately, that's not much of an issue nowadays, given Java's popularity. Clojure itself comes as a Java archive (JAR) file, which you then can execute.

But, given the number of Clojure packages and libraries you'll likely want to use, you would be better off using Leiningen, a package manager for installing Clojure and Clojure-related packages. (The name is from a story, "Leiningen and the Ants", and is an indication of how the Clojure community doesn't want to use the established dependency-management system, Ant.) You definitely will want to install Leiningen. If your Linux distribution doesn't include a modern copy already, you can download the shell script from **https://raw.github.com/technomancy/leiningen/stable/bin/lein**.

Execute this shell script, putting it in your PATH. After you download the Leiningen jarfile, it will download and install Leiningen in your ~/.lein directory (also known as LEIN_HOME). That's all you need in order to start creating a Clojure Web application.

With Leiningen installed, you can create a Web application. But in order to do that, you'll need to decide which framework to use. Typically, you create a new Clojure project with `lein new`, either naming the project on which you want to work (`lein new myproject`), or by naming the template you wish to copy and then the name of the project (`lein new mytemplate myproject`). You can get a list of existing templates by executing `lein help new` or by looking at the **https://clojars.org** site, a repository for Clojure jarfiles and libraries.

You also can open an REPL (read-eval-print loop) in order to communicate directly with Clojure. I'm not going to go into all the details here, but Clojure supports all the basic data types you would expect, some of which are mapped to Java classes. Clojure supports integers and strings, lists and vectors, maps (that is, dictionaries or hashes) and sets. And like all Lisps, Clojure indicates that you want to evaluate (that is, run) code by putting it inside parentheses and putting the function name first. Thus, you can say:

```
(println "Hello")
(println (str "Hello," " " "Reuven"))
(println (str (+ 3 5)))
```

You also can assign variables in Clojure. One of the important things to know about Clojure is that all data is immutable. This is somewhat familiar territory for Python programmers, who are used to having some immutable data types (for example, strings and tuples) in the language. In Clojure, all data is immutable, which means that in order to "change" a string, list, vector or any other data type, you really must reassign the same variable to a new piece of data. For example:

```
user=> (def person "Reuven")
#'user/person

user=> (def person (str person " Lerner"))
#'user/person

user=> person
"Reuven Lerner"
```

Although it might seem strange to have all data be immutable, this tends to reduce or remove a large number of concurrency problems. It also is surprisingly natural to work with given the number of functions in Clojure that transform existing data and the ability to use "def" to define things.

You also can create maps, which are Clojure's implementation of hashes or dictionaries:

```
user=> (def m {:a 1 :b 2})
#'user/m

user=> (get m :a)
1

user=> (get m :x)
nil
```

You can get the value associated with the key "x", or a default if you prefer not to get nil back:

```
user=> (get m :x "None")
"None"
```

Remember, you can't change your map, because data in Clojure is immutable. However, you can add it to another map, the values of which will override yours:

```
user=> (assoc m :a 100)
{:a 100, :b 2}
```

One final thing I should point out before diving in is that you can (of course) create functions in Clojure. You can create an anonymous function with `fn`:

```
(fn [first second] (+ first second))
```

The above defines a new function

Now, although Clojure is built on top of the JVM and uses some of the primitive Java classes as its fundamental data types, it is not an object-oriented language.

that takes two parameters and adds them together. However, it's usually nice to put these into a named function, which you can do with `def`:

```
user=> (def add (fn [first second] (+ first second)))
#'user/add

user=> (add 5 3)
8
```

Because this is common, you also can use the `defn` macro, which combines `def` and `fn` together:

```
user=> (add 5 3)
8

user=> (defn  add [first second] (+ first second))
#'user/add

user=> (add 5 3)
8
```

Web Development

Now that you have seen the basics of Clojure, let's consider what is involved in Web development. You need to have an HTTP server that will accept requests, as well as a program (typically known as a "router") that decides which functions will be executed for each requested URL. Then, you want to return data to the user, typically in the form of HTML files.

Now, various Web application frameworks approach this problem differently. In the most primitive ones (for example, CGI programs), the "application" really is invoked only on a single program. Large frameworks, such as Ruby on Rails, handle all of these parts and even allow you to swap parts out—and each of those parts is done using instances of different classes that handle the appropriate information.

Now, although Clojure is built on top of the JVM and uses some of the primitive Java classes as its fundamental data types, it is not an object-oriented language. Clojure is a functional language, which means that all of the aforementioned steps will be handled using functions—either those that you define or those that have been defined for you by a Web framework.

For the purposes of this article, I'm going to use Compojure, a simple Web framework for Clojure. To create a basic Compojure project, you can use Leiningen:

```
lein new compojure cjtest
```

This potentially will download a number of libraries from clojars (the Clojure library repository), then the new Clojure project ("cjtest"). Once that is done, you can run it by going into the cjtest directory and executing:

```
lein ring server
```

This will download and install whatever dependencies exist, and it then will start a simple HTTP server, by default on port 3000. You then can visit this simple application at http://localhost:3000, where you'll be greeted by "Hello, world" from Compojure.

Now, how does this work? How does Compojure know what to do?

The answer is in the Clojure project file (project.clj). The default, generic Compojure project is defined using (of course!) Lisp code. Indeed, it's pretty standard in the Lisp world to use Lisp code as data. The default project created for me by Leiningen looks like this:

```
(defproject cjtest "0.1.0-SNAPSHOT"
  :description "FIXME: write description"
  :url "http://example.com/FIXME"
  :dependencies [[org.clojure/clojure "1.5.1"]
                 [compojure "1.1.5"]]
  :plugins [[lein-ring "0.8.5"]]
  :ring {:handler cjtest.handler/app}
  :profiles
  {:dev {:dependencies [[ring-mock "0.1.5"]]}})
```

This uses Clojure's `defproject` macro, giving it a name "cjtest" and a version number. Then there are several keywords, a Clojure data type used as (among other things) the keys in maps. Thus, the above example shows Clojure that the program has the default description, and it depends on both Clojure and Compojure (and specifies the versions of each).

A second configuration file, also written in Lisp code, sits within the src/cjtest directory and is called handler.clj. The default looks like this:

```
(ns cjtest.handler
  (:use compojure.core)
  (:require [compojure.handler :as handler]
            [compojure.route :as route]))

(defroutes app-routes
  (GET "/" [] "Hello World")
  (route/resources "/")
  (route/not-found "Not Found"))

(def app
  (handler/site app-routes))
```

In other words, if someone requests "/", it will return "Hello World". You then add a route for any static resources you might want to server, which should be in the resources directory of your Clojure project. You then add a not-found route for handling 404 errors.

You can run this amazing new Web application with:

```
lein ring server
```

If you're wondering what "ring" is doing there, suffice it to say that Ring is the Clojure library that handles just about all Web-related activity. It's a low-level library, however, and it doesn't function as an actual Web application framework. It uses the standard Jetty system for Java Web applications, although you don't need to worry about that at this stage, given the automation that Leiningen provides; any missing libraries or dependencies will be downloaded when you run Ring.

Sure enough, after running the above, if you go to /, you get a "Hello world" message back from the server, because you had mapped `GET /` to a string. If you want, you instead can map it to a function by having square brackets following the route name, and then adding a function body:

```
(defroutes app-routes
  (GET "/" [] "Hello World")
  (GET "/fancy/:name" [name]
       (str "Hello, " name))
  (route/resources "/")
  (route/not-found "Not Found"))
```

You even can add HTML tags for some formatting:

```
(defroutes app-routes
  (GET "/" [] "Hello World")
  (GET "/fancy/:name" [name]
       (str "<p><b>Hello</b>, " name "<p>"))
  (route/resources "/")
  (route/not-found "Not Found"))
```

Notice how in Lisp, because of the prefix syntax, you don't need to use + or any other operator to create a string from three (or any number) parts. You just add additional parameters, separated by spaces, and they are added to the new string that `str` returns.

If you don't want to have your

function inline, you always can define it elsewhere and then pass it to `defroutes`:

```
(defn say-hello
  [req]
  (str "<p><b>Hello</b>, "
  ➥(get (get req :route-params) :name) "<p>"))

(defroutes app-routes
  (GET "/" [] "Hello World")
  (GET "/fancy/:name" [name] say-hello)
  (route/resources "/")
  (route/not-found "Not Found"))
```

Notice how this function, `say-hello`, takes a single argument (called `req`), a map of maps that contains all the data having to do with the HTTP request you received. If you want to grab the `name` parameter that was defined in the route, you have to get it from the `route-params` map inside the `req` map that your function is passed. You probably will want to explore the `req` object to understand just what Compojure is passing and how to work with such data.

Getting Fancier

Now, Compojure advertises itself as a routing mechanism for Web applications. This raises the question of what sorts of templates Compojure brings to the table. And, the answer is none. That's right, Compojure itself expects you to return strings containing HTML, but how you make those strings is up to you.

Thus, you can create strings, as shown above, in order to return a value. Or, you can use a templating engine, which in the Clojure world, simply means including another dependency and using the functions defined by that dependency:

```
(defproject cjtest "0.1.0-SNAPSHOT"
  :description "FIXME: write description"
  :url "http://example.com/FIXME"
  :dependencies [[org.clojure/clojure "1.5.1"]
                 [compojure "1.1.5"]
                 [hiccup "1.0.3"]]
  :plugins [[lein-ring "0.8.5"]]
  :ring {:handler cjtest.handler/app}
  :profiles
  {:dev {:dependencies [[ring-mock "0.1.5"]]}})
```

Now, within the handler.clj file, you need to make another change, so that you can have access to the "hiccup" HTML-generation system:

```
(ns cjtest.handler
  (:use compojure.core hiccup.core)
  (:require [compojure.handler :as handler]
            [compojure.route :as route]))
```

With these in place, you now can

create HTML by using a function and Clojure's vector syntax:

```
(defn say-hello
  [req]
  (html [:p [:b "Hello, "
 ➥(get (get req :route-params) :name) ]]))
```

Now, if you look at the parentheses and say, "Yikes, that's what I dislike about Lisp", I'll understand. But if you look at this as Lisp code and realize that this means your templates automatically are valid HTML if they are valid Lisp, you're thinking in the terms that Clojure wants to encourage—that is, using Lisp data structures as much as possible and feeding them to functions that know how to work with them.

Conclusion

As you can see, Clojure provides the simplicity of Lisp's structure and syntax while staying within the world of Java and the JVM. My next article will look a bit deeper at Compojure, investigating forms, database connectivity and other issues that modern Web applications want to use.■

Web developer, trainer and consultant Reuven M. Lerner is finishing his PhD in Learning Sciences at Northwestern University. He lives in Modi'in, Israel, with his wife and three children. You can read more about him at http://lerner.co.il, or contact him at reuven@lerner.co.il.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

Resources

The home page for the Clojure language is **http://clojure.org** and includes a great deal of documentation.

You can read more about Leiningen at its home page: **http://leiningen.org**. Similarly, documentation for Compojure is at its home page, **http://compojure.org**, and Hiccup is at **https://github.com/weavejester/hiccup**.

Two good books about Clojure are *Programming Clojure* by Stuart Halloway and Aaron Bedra (published by the Pragmatic Programmers) and *Clojure Programming* by Chas Emerick, Brian Carper and Christophe Grand (published by O'Reilly). I've read both during the past year or two, and I enjoyed both of them for different reasons, without a clear preference.

DAVE TAYLOR

Cribbage: Closing Things Up

Dave wraps up his *Cribbage* script by explaining how to code a hand-optimizing routine.

After months of work, we're almost done with the *Cribbage* shell script. As you'll recall, we're focused on writing just the portion of the game that evaluates the best four cards out of the initial six dealt in a two-player game.

This is, of course, a complex task, because there are a lot of ways you can pick four from six, and a lot of ways that any two, three or four cards can produce points. And we haven't even considered "His Nobs", the two points you can get simply by having a jack of the same suit as the shared extra card turned up after both players discard.

At this point in the evolution of the script, it can extract all six-choose-four subhand combinations, calculate the point value of pairs, three of a kinds, combinations that add up to 15 and three-card runs. There's not much left to do—just making sure it'll properly calculate the rare instance when you have four of a kind and when you might have a four-card run.

The code has lots of debugging output, as befits a script in process, and it's very close. Here's an example:

```
Subhand 14:  4H  4S  5H  6H
a pair 4 and 4 for two
three card fifteen for two
4 + 5 + 6 run for three
three card fifteen for two
4 + 5 + 6 run for three
total point value of that hand: 12
```

That's correct. Those four cards combine to be worth a total of 12 points.

But, what about those edge cases? Let's find out what happens if we have four cards in the same

Let's find out what happens if we have four cards in the same suit: does the script recognize it's worth four additional points as a flush?

suit: does the script recognize it's worth four additional points as a flush? Let's see:

```
Subhand 6:  2H  3H  4H  5H
2 + 3 + 4 run for three
3 + 4 + 5 run for three
  total point value of that hand: 6
```

Nope. This is easily done, but it's more code—another edge case. It works by having the calling function give the function `calc4cardvalue`, not just the point rank of each card and the numeric rank of each card, but also the suit. That's 12 parameters (which is a bit clumsy, admittedly). Each of the suit parameters is assigned a local variable for convenience, then the test is straightforward:

```
if [ $cs0 -eq $cs1 -a $cs1 -eq $cs2 -a $cs2 -eq $cs3 ] ; then
  echo "four cards flush for four."
  points=$(( $points + 4 ))
fi
```

Now we get this sort of result:

```
Subhand 10:  3H  4H  5H  6H
3 + 4 + 5 run for three
three card fifteen for two
4 + 5 + 6 run for three
four card flush for four.
Total point value of that hand: 12
```

It's close, but there's a problem with the calculations, because runs aren't counted as 3+4+5 and 4+5+6 (twice three points) but instead as a single four-card run (3+4+5+6), which is worth only four points.

The only time that this is an issue, however, is when there are two runs of three, so the easy way to identify this situation is simply to keep a count of how many three-runs are encountered.

Ready? The conditional is pretty gnarly:

```
if [ $(( ${cardrankfull[${combo:0:1}]} + 1 )) -eq
➥${cardrankfull[${combo:2:1}]} -a
    $(( ${cardrankfull[${combo:2:1}]} + 1 )) -eq
     ➥${cardrankfull[${combo:4:1}]} ] ; then
  echo "run for three for two"
  points=$(( $points + 3 ))
  runof3=$(( $runof3 + 1 ))
fi
```

The challenge we're faced with now is perhaps the easiest one: how to calculate the optimal subhand.

Now at the end of the routine we check if `$runof3` is 2, then we need to subtract two points to correct the counting (remember, a run of four is worth four points, not six).

Here's the same hand, with the new code in place:

```
Subhand 10:  3H  4H  5H  6H
3 + 4 + 5 run for three
three card fifteen for two
4 + 5 + 6 run for three
four card flush for four.
two runs of three = a run of four. minus two
Total point value of that hand: 10
```

By George, I think we have it!

Calculating the Optimal Subhand

The challenge we're faced with now is perhaps the easiest one: how to calculate the optimal subhand. This output snippet demonstrates the dramatic variance in point value of different hand subsets:

```
Subhand 9:  3C  5S  6H  JS
fifteen for two
  total point value of that hand: 2

Subhand 10:  4H  5H  5S  6H
a pair 5 and 5 for two
three card fifteen for two
4 + 5 + 6 run for three
three card fifteen for two
4 + 5 + 6 run for three
  total point value of that hand: 12

Subhand 11:  4H  5H  5S  JS
a pair 5 and 5 for two
fifteen for two
fifteen for two
  total point value of that hand: 6

Subhand 12:  4H  5H  6H  JS
fifteen for two
three card fifteen for two
4 + 5 + 6 run for three
  total point value of that hand: 7
```

Choose one subset of four cards, and you have 12 points in your hand. Pick a different combination, however, and you have two points. That's why it's important to be able to do this correctly!

Calculating the best hand out of the set is easy, fortunately. We step through all possible four-card combinations and compare the value to the best previously seen. The code looks like this:

```
if [ $points -gt $handvalue ] ; then
  besthand=$subhand
  handvalue=$points
fi
```

And at the end, the output is done simply:

```
echo "Done with calculations. Determined that hand
$besthand is best, with $handvalue points"
```

That's really all there is other than some fancy output cleanup. Rather than seeing the subhands and point value of each, let's just have it show all six cards and which four it recommends you keep as your own hand:

```
Hand: 2S, 5H, 6C, 10S, JD, KD.
Your best subhand is worth 6 points:   5H  10S  JD  KD
$ sh cribbage.sh
Hand: 2S, 3C, 5C, 5D, 6S, 9H.
Your best subhand is worth 4 points:   2S  3C  5C  5D
$ sh cribbage.sh
Hand: AH, 7D, 8D, 9C, 9S, JC.
Your best subhand is worth 8 points:   7D  8D  9C  9S
$ sh cribbage.sh
Hand: 2C, 4D, 5C, 5H, 6C, 8H.
Your best subhand is worth 10 points:   4D  5C  5H  6C
$ sh cribbage.sh
Hand: 4H, 5S, 6C, 8S, JD, QC.
Your best subhand is worth 7 points:   4H  5S  6C  JD
```

The code underlying it ties in to the conditional statement shown earlier:

```
/bin/echo -n "Your best subhand is worth $handvalue points: "
for thecard in ${sixfour[$besthand]}
do
  showcard ${hand[$thecard]}
  /bin/echo -n "  $showcardvalue"
done

  echo ""
```

That's it. Mission accomplished, after months of working on this script. You can find the final script, debug, comments and warts and all, on the *LJ* Web site at **http://www.linuxjournal.com/231/wts**.

Next month, we'll switch gears and look at some administrative tasks for Webmasters who have config and admin scripts that need an occasional verification performed. And, of course, if you have ideas for scripts, don't be shy, send them along!■

Dave Taylor has been hacking shell scripts for more than 30 years. Really. He's the author of the popular *Wicked Cool Shell Scripts* and can be found on Twitter as @DaveTaylor and more generally at http://www.DaveTaylorOnline.com.

KYLE RANKIN

DNSSEC Part I: the Concepts

Before you deploy DNSSEC for your zones, it helps to understand some of the concepts and reasoning behind DNSSEC.

Like IPv6, DNSSEC is one of those great forward-looking protocols that unfortunately hasn't seen wide adoption yet. Before I implemented it myself, I could see why. Although some people think BIND itself is difficult to set up, DNSSEC adds an extra layer of keys, key management and a slew of additional DNS records. One day I decided to set up DNSSEC on a personal zone to familiarize myself with the concepts and process, and it turns out that the implementation isn't all that bad once you grasp a few concepts. In this article, I cover some of the general concepts, and in my next article, I'll describe the steps for using DNSSEC on your own zone.

How DNS Works

It can be difficult to understand how DNSSEC works if you don't completely grasp how DNS itself works. Really, I could spend a whole article just talking about how DNS works, but for the purposes of this article, here's a very high-level trace of a typical uncached DNS query that resolves a domain of mine: www.greenfly.org. When you type a URL in a Web browser and press Enter, behind the scenes, the OS starts a process to convert that hostname into an IP address. Although some people run DNS caching services on their personal computers, for the most part, you rely on an external DNS server you've either configured by hand or via DHCP. When your OS needs to convert a hostname into an IP, it sends a DNS query to a name server defined in /etc/resolv.conf (these days, if this file is managed by resolvconf, the real name servers can be trickier to track down). This starts what's known as a recursive query, as this remote DNS server acts on your behalf to talk to any other DNS servers it needs to contact to resolve your hostname to an IP.

In the case of resolving www.greenfly.org, the recursive DNS server starts by sending a query to one of the 13 root DNS servers on the Internet (192.33.4.12), asking for the IP for www.greenfly.org. The root name servers reply that they don't know that information, but the name servers for .org might know, and here are their names and IP addresses. Next, the recursive DNS server sends a query to one of the .org name servers (199.19.54.1), asking the same question. The .org name server replies that it also doesn't know the answer, but the name servers for greenfly.org might know, and here are their names and IP addresses. Finally, the recursive DNS server asks one of the greenfly.org name servers (75.101.46.232) and gets back the answer that www.greenfly.org is at 64.142.56.172.

If you are curious how this might work for a domain you own, just use the `dig` command with the `+trace` option. Here's example output for www.greenfly.org:

```
$ dig www.greenfly.org +trace


; <<>> DiG 9.8.1-P1 <<>> www.greenfly.org +trace
;; global options: +cmd
.                       498369  IN   NS    j.root-servers.net.
.                       498369  IN   NS    k.root-servers.net.
.                       498369  IN   NS    e.root-servers.net.
.                       498369  IN   NS    m.root-servers.net.
.                       498369  IN   NS    c.root-servers.net.
.                       498369  IN   NS    d.root-servers.net.
.                       498369  IN   NS    l.root-servers.net.
.                       498369  IN   NS    a.root-servers.net.
.                       498369  IN   NS    h.root-servers.net.
.                       498369  IN   NS    i.root-servers.net.
.                       498369  IN   NS    g.root-servers.net.
.                       498369  IN   NS    b.root-servers.net.
.                       498369  IN   NS    f.root-servers.net.
;; Received 436 bytes from 127.0.0.1#53(127.0.0.1) in 60 ms


org.                    172800  IN   NS    b2.org.afilias-nst.org.
org.                    172800  IN   NS    b0.org.afilias-nst.org.
org.                    172800  IN   NS    c0.org.afilias-nst.info.
org.                    172800  IN   NS    a0.org.afilias-nst.info.
org.                    172800  IN   NS    d0.org.afilias-nst.org.
org.                    172800  IN   NS    a2.org.afilias-nst.info.
;; Received 436 bytes from 192.33.4.12#53(192.33.4.12) in 129 ms


greenfly.org.          86400   IN   NS    ns2.greenfly.org.
greenfly.org.          86400   IN   NS    ns1.greenfly.org.
;; Received 102 bytes from 199.19.54.1#53(199.19.54.1) in 195 ms


www.greenfly.org.      900     IN   A     64.142.56.172
greenfly.org.          900     IN   NS    ns1.greenfly.org.
greenfly.org.          900     IN   NS    ns2.greenfly.org.
;; Received 118 bytes from 75.101.46.232#53(75.101.46.232) in 2 ms
```

Although this may seem like a lot of steps, in practice, name servers cache answers for a period of time known as the TTL, or Time To Live, that's assigned to every record. That way, a DNS resolver has to look up only any records that have expired.

DNS Security Issues

DNS has been around for quite a while, and it has had its share of security issues over time. DNS is designed to be an open, friendly service. Although some administrators might treat DNS records as secrets, generally speaking, a DNS record's purpose is to be looked up by anyone who requests it, so DNS records are not encrypted, and DNS queries generally occur over plain text. Here are a few DNS security issues facing us today:

- Domain names sometimes look alike (google.com vs. google.com), which an attacker can take advantage of to encourage you to click on a legitimate-looking link.
- Companies can't always register their name on all TLDs (.com vs. .biz vs. .net), so an attacker might register mybank.biz, which a victim may think is legitimate.
- Many DNS servers (known as open resolvers) will perform recursive queries for anyone who asks.
- Open resolvers commonly are used in modern DNS amplification DDOS attacks (an attack where a relatively small DNS query results in an orders of magnitude larger response that, due to DNS queries occurring over UDP, can be redirected to a different target than the host who initiated the request). With a DNS amplification attack, it takes much less bandwidth from an attacking machine to generate large amounts of traffic for a target.
- DNS is subject to MitM (Man in the Middle) attacks where DNS records can be rewritten before they get back to the victim. This lets an attacker, for instance, change the IP of yourbank.com in a DNS request to point to the Web site the attacker controls instead.
- DNS spoofing/cache poisoning attacks (this class of attacks was covered by a series of Paranoid Penguin columns in 2011) essentially allow an attacker to inject fake DNS records into a DNS resolver's cache to point victims, again, at an attacker's site instead

of the site they intend to visit.

Of all of these different DNS security issues, DNSSEC attempts to address the last two, MitM attacks and DNS cache poisoning, by signing every DNS reply with a signature, much like a PGP signature in an e-mail. The DNSSEC signature verifies that the DNS result you see came from the authoritative DNS server for that domain and that it wasn't tampered with in any way in transit.

How DNSSEC Works

If you are somewhat familiar with the CA (Certificate Authority) system or with how public-key cryptography works with PGP-signed e-mails, understanding DNSSEC will be a bit simpler, as it has some similarities. With DNSSEC, a domain creates a set of public and private keys that it uses to sign every record set in its zone. The domain then publishes the public keys in the zone as a record of its own along with the signatures. With these public keys and signatures, anyone performing a DNS query against that domain can use the public key to validate the signature for a particular record. Because only someone with the private key could sign the record, you can be assured the result was signed by someone who controls the domain. If someone tampered with the record along the way, the signature no longer would match.

Like with PGP-signed e-mail, having cryptographic signatures attached to a document isn't a sufficient reason to trust the document. After all, attackers simply could generate a different key pair, change the record and attach their public key and updated signature instead. With DNSSEC, you need an outside mechanism to know you can trust that the public key you are getting truly came from the domain. With PGP-signed e-mail, you validate the public key with outside mechanisms, such as key-signing parties, with the hope that if you receive an e-mail from someone for which you don't immediately have a public key signature, someone you already trust does, and you can use that chain of trust to validate the signature. I don't know of any DNSSEC key-signing parties; instead, the chain of trust is built much like how it is with the CA system.

When you visit a site that's protected by HTTPS, the site will present you with a copy of its public key (here called a certificate), so you can establish a secure, encrypted communication channel with the site, but equally important, you also can validate that you are in fact communicating with, for instance, mail.google.com and not some attacker. Because you probably didn't go to a Google key-signing party either, how can you trust that certificate? It turns out that each certificate is signed by a CA like Verisign, Thawte or a large number of others. This signature is attached to the certificate you receive, and your browser itself has public keys for each of the CAs built in to it. The browser implicitly trusts these CA certificates, so if you receive a certificate from a site that has been signed by any of these CAs, you will trust it as valid. This trust, by the way, is why it is such a problem when a CA gets hacked. Attackers then can use the private keys from that CA to generate new certificates for any site they want to impersonate and browsers will trust them automatically.

DNSSEC signatures follow a similar chain of trust to PGP keys and CAs. In this case, the root DNS servers act as the trust anchor, and DNSSEC resolvers implicitly trust what the root DNS servers sign, much like browsers trust CAs. When a TLD (Top Level Domain) wants to implement DNSSEC, it submits a special DS record to the root DNS servers to sign. Those DS records contain a signature for the subdomain generated by the private key. The root DNS server hosts that DS record and signs it with its private key. In that way, because you trust root, you can trust that the signature for org has not been modified; therefore, you can trust it as well in the same way you would trust a certificate signed by a CA. Then if you want to enable DNSSEC for a .org domain, for instance, you would submit a DS record for each key through your registrar if it supports DNSSEC. Each DS record contains a key's signature for your domain that the org name servers then would sign and host.

In this model, the chain of trust basically follows the same order that a recursive DNS query like I outlined above would follow. A DNSSEC query adds an extra validation step to each part of the process. For instance, a query for www.isc.org starts at the root, uses the DS record for org to validate com signatures, then uses the DS record for isc.org to validate the isc.org signature attached to www.isc.org. You can add the `+dnssec` option to `dig +trace` to see the full transaction:

```
$ dig +trace +dnssec www.isc.org

; <<>> DiG 9.8.1-P1 <<>> +trace +dnssec www.isc.org
;; global options: +cmd
```

```
.                 492727  IN      NS      g.root-servers.net.
.                 492727  IN      NS      m.root-servers.net.
.                 492727  IN      NS      i.root-servers.net.
.                 492727  IN      NS      b.root-servers.net.
.                 492727  IN      NS      f.root-servers.net.
.                 492727  IN      NS      a.root-servers.net.
.                 492727  IN      NS      k.root-servers.net.
.                 492727  IN      NS      h.root-servers.net.
.                 492727  IN      NS      l.root-servers.net.
.                 492727  IN      NS      e.root-servers.net.
.                 492727  IN      NS      c.root-servers.net.
.                 492727  IN      NS      d.root-servers.net.
.                 492727  IN      NS      j.root-servers.net.
.                 518346  IN      RRSIG   NS 8 0 518400 20130517000000
20130509230000 20580 . M8pQTohc9iGqDHWfnACnBGDwPhFs7G/nqqOcZ4OobVxW8l
KIWa1Z3vho56IwomeVgYdj+LNX4Znp1hpb3up9Hif1bCASk+z3pUC4xMt7No179Ied
DsNz5iKfdNLJsMbG2PsKxv/C2fQTC5lRn6QwO4Ml09PAvktQ9F9z7IqS kUs=
;; Received 589 bytes from 127.0.0.1#53(127.0.0.1) in 31 ms

org.              172800  IN      NS      d0.org.afilias-nst.org.
org.              172800  IN      NS      b0.org.afilias-nst.org.
org.              172800  IN      NS      a2.org.afilias-nst.info.
org.              172800  IN      NS      b2.org.afilias-nst.org.
org.              172800  IN      NS      c0.org.afilias-nst.info.
org.              172800  IN      NS      a0.org.afilias-nst.info.
org.              86400   IN      DS      21366 7 1
E6C1716CFB6BDC84E84CE1AB5510DAC69173B5B2
org.              86400   IN      DS      21366 7 2
96EEB2FFD9B00CD4694E78278B5EFDAB0A80446567B69F634DA078F0 D90F01BA
org.              86400   IN      RRSIG   DS 8 1 86400 20130517000000
20130509230000 20580 . kirNDFgQeTmi0o5mxG4bduPm0y8LNo0YG9NgNgZIbYdz8
gdMK8tvSneJUGtJca5bIJyVGcOKxV3aqg/r5VThvz8its50tiF4l5lt+22n/AGnNRxv
onMl/NA5rt0K2vXtdskMbIRBLVUBoa5MprPDwEzwGg2xRSvJryxQEYcT 80Y=
;; Received 685 bytes from 192.203.230.10#53(192.203.230.10) in 362 ms

isc.org.          86400   IN      NS      ns.isc.afilias-nst.info.
isc.org.          86400   IN      NS      ams.sns-pb.isc.org.
isc.org.          86400   IN      NS      sfba.sns-pb.isc.org.
isc.org.          86400   IN      NS      ord.sns-pb.isc.org.
isc.org.          86400   IN      DS      12892 5 2
F1E184C0E1D615D20EB3C223ACED3B03C773DD952D5F0EB5C777586D E18DA6B5
isc.org.          86400   IN      DS      12892 5 1
982113D08B4C6A1D9F6AEE1E2237AEF69F3F9759
isc.org.          86400   IN      RRSIG   DS 7 2 86400 20130530155458
20130509145458 42353 org.
Qp7TVCt8qH74RyddE21a+OIBUhd6zyzAgSB1Qykl2NSkkebtJ1QeE5C5
R8eblh8XvmQXjqN7zwcj7sDaaHXBFXGZ2EeVT5nwJ1Iu4EGH2WK3L7To
BDjR+8wNofZqbd7kX/LOSvNu9jdikb4Brw9/qjkLk1XaOPgl/23WkIfp zn8=
;; Received 518 bytes from 199.19.56.1#53(199.19.56.1) in 400 ms

www.isc.org.      600     IN      A       149.20.64.42
www.isc.org.      600     IN      RRSIG   A 5 3 600 20130609211557
20130510211557 50012 isc.org.
tNE0KPAh/PUDWYumJ353BV6KmHl1nDdTEEDS7KuW8MVVMxJ6ZB+UTnUn
bzWC+kNZ/IbhYSD1mDhPeWvy5OGC5TNGpiaaKZ0/+OhFCSABmA3+Od3S
fTLSGt3p7HpdUZaC9qlwkTlKckDZ7OQPw5s0G7nFInfT0S+nKFUkZyuB OYA=
isc.org.          7200    IN      NS      ord.sns-pb.isc.org.
isc.org.          7200    IN      NS      sfba.sns-pb.isc.org.
isc.org.          7200    IN      NS      ns.isc.afilias-nst.info.
isc.org.          7200    IN      NS      ams.sns-pb.isc.org.
isc.org.          7200    IN      RRSIG   NS 5 2 7200 20130609211557
20130510211557 50012 isc.org.
SdMCLPfLXiyl8zrfbFpFDz22OiYQSPNXK18gsGRzTT2JgZkLZYZW9gyB
vPTzm8L+aunkMDInQwFmRPqvHcbO+5yS98IlW6FbQXZF0/D3Y9J2i0Hp
ylHzm306QNtquxM9vop1GOWvgLcc239Y2G5SaH6ojvx5ajKmr7QYHLrA 8l8=
;; Received 1623 bytes from 199.6.0.30#53(199.6.0.30) in 60 ms
```

You'll see a number of new record types in this response, but don't worry, I go over all of the new DNSSEC record types next.

DNSSEC Terminology

A lot of different acronyms and new terminology comes up when you read DNSSEC documentation, so here are a few common terms you'll want to be acquainted with as you use DNSSEC:

- RR (Resource Record): this is the smallest unit of data in a zone, such as a single A record, NS record or MX record.
- RRSET: a complete set of Resource Records. For instance, an RRSET might be all NS records or A records for a particular name.
- KSK (Key-Signing Key): signs DNSKEY records in a zone.
- ZSK (Zone-Signing Key): signs all of the other records in a zone.
- SEP (Secure Entry Point): a flag set in a key to denote it as a KSK.

While best practice dictates a separate KSK and ZSK, it isn't an actual requirement. In my next article on DNSSEC implementation, I will discuss the main differences between the two key types and why separate keys is considered a best practice.

New DNSSEC Record Types

DNSSEC also has introduced a number of new DNS record types into the mix. These records are published with the zone along with the rest of your DNS records and are pulled down as needed by any DNSSEC-enabled query:

- DNSKEY: this is a public key for the zone and can either be a KSK or ZSK.
- RRSIG (Resource Record Signature): this record contains a signature for an RRSET created with a particular ZSK.
- NSEC (Next Secure record): these records are used in "negative answers" to prove whether a name exists.
- NSEC3 (Next Secure version 3): these records are like NSEC, but protect against "zone walking" where an outside user could use NSEC records to walk down the zone and discover all of the records in the zone (much like being able to perform a zone transfer).
- DS (Delegation Signer): this record contains a KSK signature

ManageEngine
Powering IT ahead

What is a **Superhero** without the right tools?

No, we're not referring to costumed crime fighters who come to the rescue of others in trouble. We're talking about IT operations personnel - people with complex skills and a daunting job, who fight downtime, performance slowdowns, and other evils to keep your apps running 24x7. They have to be ready to take action if there is trouble brewing in the system – and being ready involves having the right tools.

ManageEngine provides the right set of monitoring tools for your IT operations team, enabling them to keep track of the performance of their complex apps from both within and outside their data center.

www.manageengine.com/apm www.site24x7.com

- Application Performance Monitoring
- End User Experience Monitoring
- Anomaly Detection
- Automated Dependency Mapping
- Deep Transaction Monitoring
- Integrated Management Console

Zoho Corporation, 4900 Hopyard Rd., Suite 310 Pleasanton, CA 94588, USA
Phone: +1 925 924 9500 Email: sales@manageengine.com

and is submitted to the zone's parent where it is signed and is used as part of a chain of trust.

- DLV (DNSSEC Look-aside Validation): much like DS records, but are used when DS records are not supported by a zone, or as an alternate trust anchor if your registrar doesn't support DNSSEC.

DNSSEC Look-Aside Validation

DNSSEC has a sort of chicken-and-egg problem. If your TLD does not support DNSSEC, any outside resolvers won't have a complete chain of trust from root, through the TLD, to your zone. There also may be a case where your TLD does support DNSSEC, but your registrar doesn't provide a mechanism to upload a DS record to the TLD (most registrars sadly don't). In either case, DNSSEC Look-aside Validation (DLV) has been created to provide an alternate trust anchor.

You can find more details on DLV at **http://dlv.isc.org** (one of the main DLV providers), but essentially, instead of generating a DS record to submit to a TLD, you generate a special DLV record and submit it to a DLV server. As long as a DNS resolver is configured to trust, for instance, dlv.isc.org, it can use that to anchor the chain of trust and then trust your signed records.

It turns out DNSSEC has a lot of new concepts to understand before you even get to implementing it; however, the implementation isn't all that bad once you see the steps laid out, so be sure to follow up with my article next month when I talk about how to implement DNSSEC for a zone using BIND. I'll even cover how to set up DLV for the zone.■

Kyle Rankin is a Sr. Systems Administrator in the San Francisco Bay Area and the author of a number of books, including *The Official Ubuntu Server Book*, *Knoppix Hacks* and *Ubuntu Hacks*. He is currently the president of the North Bay Linux Users' Group.

Resources

A Collection of Links to DNSSEC Information: **http://dnssec.net**

ISC's DLV Documentation: **https://dlv.isc.org**

DNSSEC Chain of Trust Visualizer: **http://dnsviz.net**

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

O'REILLY®

oscon

open source convention

JULY 22–26, 2013

PORTLAND, OREGON

oscon.com

Build your open source expertise

OSCON is the must-attend gathering of the best and brightest minds in technology, an opportunity to challenge your assumptions and spark your imagination. Join us for five immersive days of all things open source — new and innovative projects, major enterprise-wide deployments, and — from icons of the open source movement — deep perspective on where we've been and where we're headed.

OSCON 2013 Tracks

- Business
- Cloud
- Community
- Data
- Education
- Geek Lifestyle
- Java and JVM
- JavaScript and HTML5
- Mobile
- Open Hardware
- Operations
- Perl
- PHP
- Programming
- Python
- Software Architecture
- Tools and Techniques
- User Experience (UX)

SAVE 15% ON ANY PASS

USE CODE **LNXJRN**

O'REILLY®

SHAWN POWERS

Taming Tomcat

Apache does more than serve Web pages, and now so can you.

I recently received a panicked call from a coworker who was doing a program demo for a client. The program was Java-based, so it needed to be deployed to a Java servlet engine. This coworker is a brilliant sysadmin and trainer, but was completely befuddled by Tomcat. So for this article, I'm giving a crash course on Tomcat administration and setup. If you've set up Java applications before, this article might be mundane, but if you don't know a JBoss from a Glassfish, read on.

Why Tomcat?

A person easily could write a book on the various ways to host Java applications. There are tons of options, ranging from free and open to commercial and expensive. Usually, when someone talks about serving up a Java application, Tomcat is the servlet engine of choice. Why? Maybe it's the cool name. Maybe it's because it's from the Apache folks, so there's a level of trust. Maybe it's just because it works well. Whatever the reason, it's a very common way to deploy Java apps, so that's what I focus on here.

Like most open-source programs, Tomcat will run on multiple platforms. There is a Windows version, which actually is rather difficult to install properly. (Running Tomcat as a service in Windows has some weird issues that must be addressed.) Thankfully, running Tomcat under Linux is fairly simple, and it runs like any other service. Linux, we love you.

Installation

Installing Tomcat in Linux is so simple that I'm really only going to mention it. If you need a specific version of Tomcat, or need some specialty compilation options, it can be a little more complex. If you have needs that particular, however, chances are you know far more about Tomcat and Java deployment than this article covers anyway.

Basically, look in your distro's software repository for "Tomcat". Note that Tomcat is a product from the fine folks at Apache. It's an entirely different product from the Apache Web server though. It would be a little less confusing if the Web server had a cool name, like "Apache Bearclaw", to differentiate it from other Apache products, but sadly, we need to deal with the naming conventions as they are. Most people use the term "Apache" for the Web server dæmon and are confused with the idea of "Apache Tomcat" as a completely separate application.

Anyway, find the Tomcat version your distro provides and install it. It can be from the software center, apt-get, yum or however you install packages for your distro. Once it's installed, it should start automatically and be happily running on your system. By default, Tomcat runs on port 8080. (See, I told you it was different from the Apache Web server!)

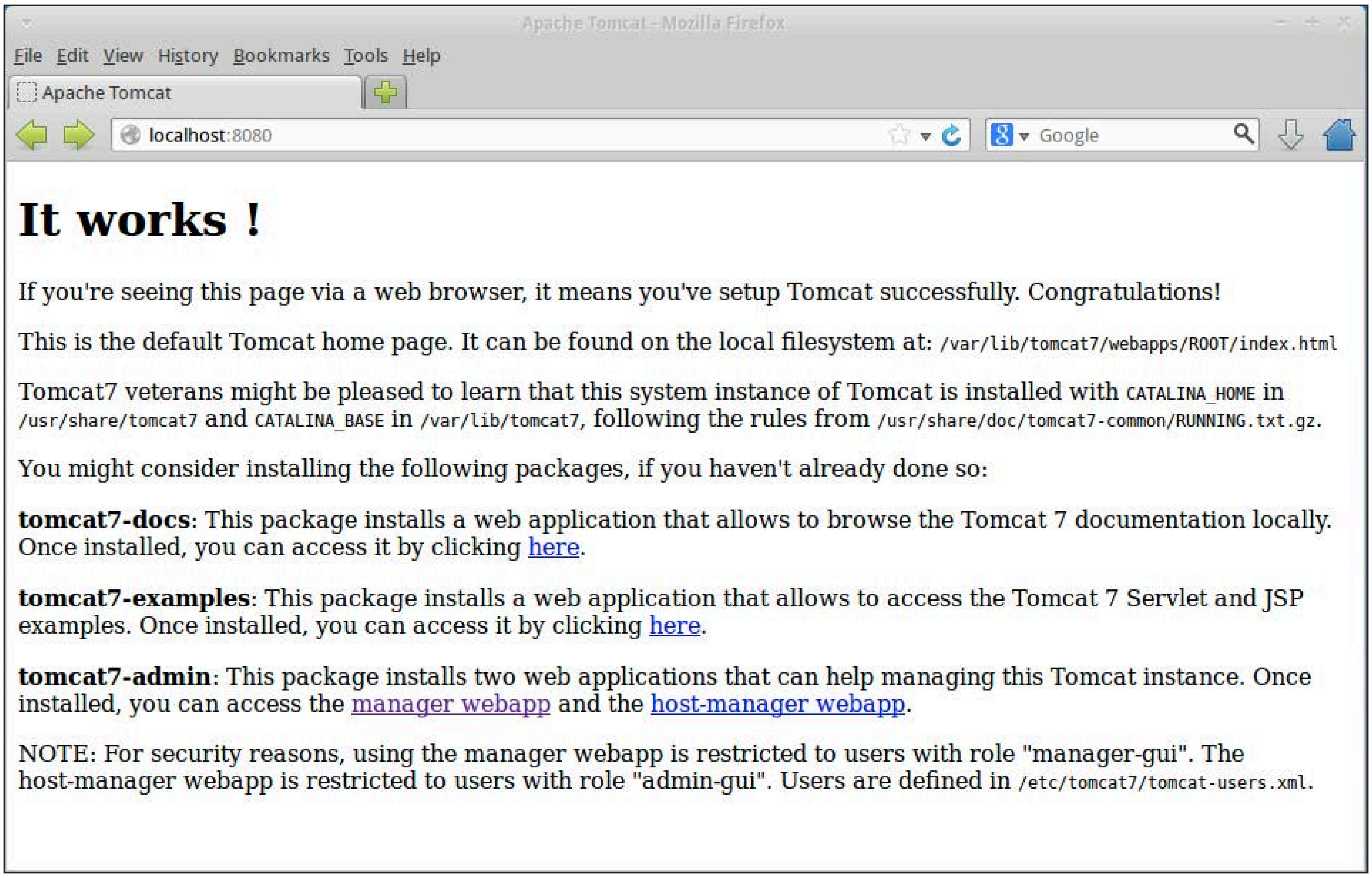
Testing

Before you deploy any Java applications, you should make sure things are working. The first thing to test is to go to http://localhost:8080 from your machine. (If it's a headless server, use the server's IP address from your workstation.) You should see a message similar to that shown in Figure 1. Of particular note, it mentions a few additional packages for installing. My example is in Ubuntu, so your distribution may include the management files with the program installation. Otherwise, like me, you'll need to install tomcat7-admin. I'm not going to do much with the management program here, but I want to install it, and point you in that direction for future exploration!

Unfortunately, the Internet has some non-trustworthy folks attached to it. By default, Tomcat's admin package disables all access. If you try to click on that admin link (even after installing the package), you won't be able to log in. You should get a login prompt, but there aren't any logins enabled. For that, you need to edit /etc/tomcat7/tomcat-users.xml, like it says at the bottom of the "It Works!" page shown in Figure 1. Basically, you need to add a line like this to the file:

```
<user name="admin" password="password" roles="manager-gui" />
```

Note that using admin/password as your login and password is a horrible, horrible idea. But this is a magazine article, not a computer. It's very unlikely my imaginary server will get hacked. Yours, however, isn't safe. So use a good user/password combination!


Apache Tomcat - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Apache Tomcat
localhost:8080
Google

It works !

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

This is the default Tomcat home page. It can be found on the local filesystem at: `/var/lib/tomcat7/webapps/ROOT/index.html`

Tomcat7 veterans might be pleased to learn that this system instance of Tomcat is installed with `CATALINA_HOME` in `/usr/share/tomcat7` and `CATALINA_BASE` in `/var/lib/tomcat7`, following the rules from `/usr/share/doc/tomcat7-common/RUNNING.txt.gz`.

You might consider installing the following packages, if you haven't already done so:

tomcat7-docs: This package installs a web application that allows to browse the Tomcat 7 documentation locally. Once installed, you can access it by clicking here.

tomcat7-examples: This package installs a web application that allows to access the Tomcat 7 Servlet and JSP examples. Once installed, you can access it by clicking here.

tomcat7-admin: This package installs two web applications that can help managing this Tomcat instance. Once installed, you can access the manager webapp and the host-manager webapp.

NOTE: For security reasons, using the manager webapp is restricted to users with role "manager-gui". The host-manager webapp is restricted to users with role "admin-gui". Users are defined in `/etc/tomcat7/tomcat-users.xml`.

Figure 1. This looks similar to the Apache Web server's test page—and for good reason, it's from the same people!

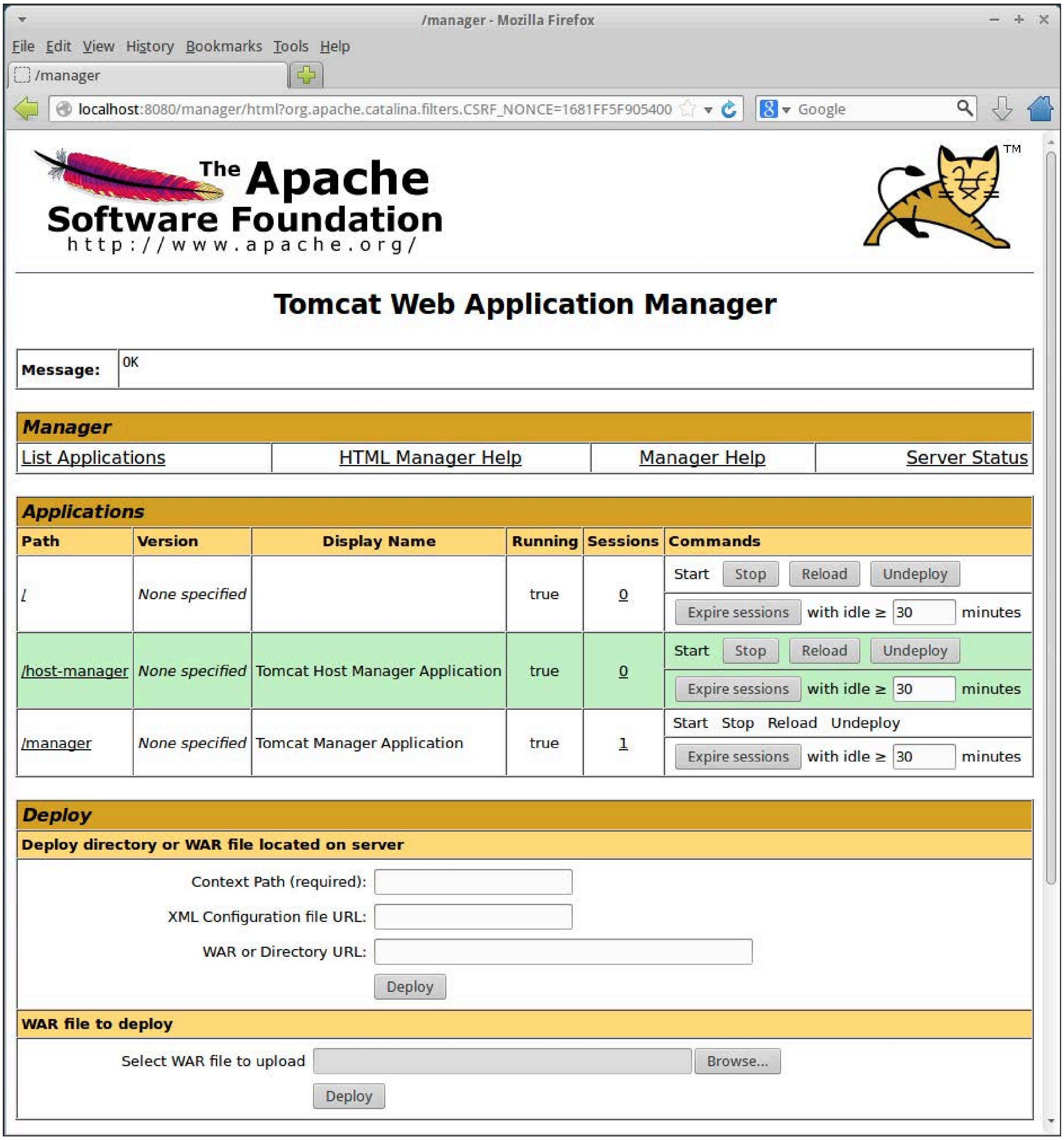
Restart the Tomcat process. In Ubuntu, that means:

```
sudo service tomcat7 restart
```

Then, try to log in to the management app, located at http://localhost:8080/manager/. Obviously, replace localhost with your server name if you're not on the same machine. After entering your credentials, you should see something like Figure 2. It gives you some information about the server and also allows you to deploy WAR files directly via the Web interface. (More about that in a bit.) If you see the management console, you're in pretty good shape and have a functional Java Servlet Engine.

Let's Start a WAR

If anyone ever asked you to set up a Tomcat server, it's probably because he or she needs to deploy a WAR file. A WAR file (Web Application aRchive) is a set of Java files bundled up into a single file, ready to deploy onto


/manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/manager
localhost:8080/manager/html?org.apache.catalina.filters.CSRF_NONCE=1681FF5F905400
Google
The Apache Software Foundation
http://www.apache.org/
TM

Tomcat Web Application Manager

Message:	OK

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path (required):	
XML Configuration file URL:	
WAR or Directory URL:	

WAR file to deploy

Select WAR file to upload

Figure 2. The Application Manger makes deploying and undeploying apps simple. It's important to know what's going on behind the scenes, however.

a Servlet Engine. If Apache Tomcat were a Nintendo Entertainment System, the WAR file would be the game cartridge. Once the WAR file is deployed, it's available to use via the Web browser.

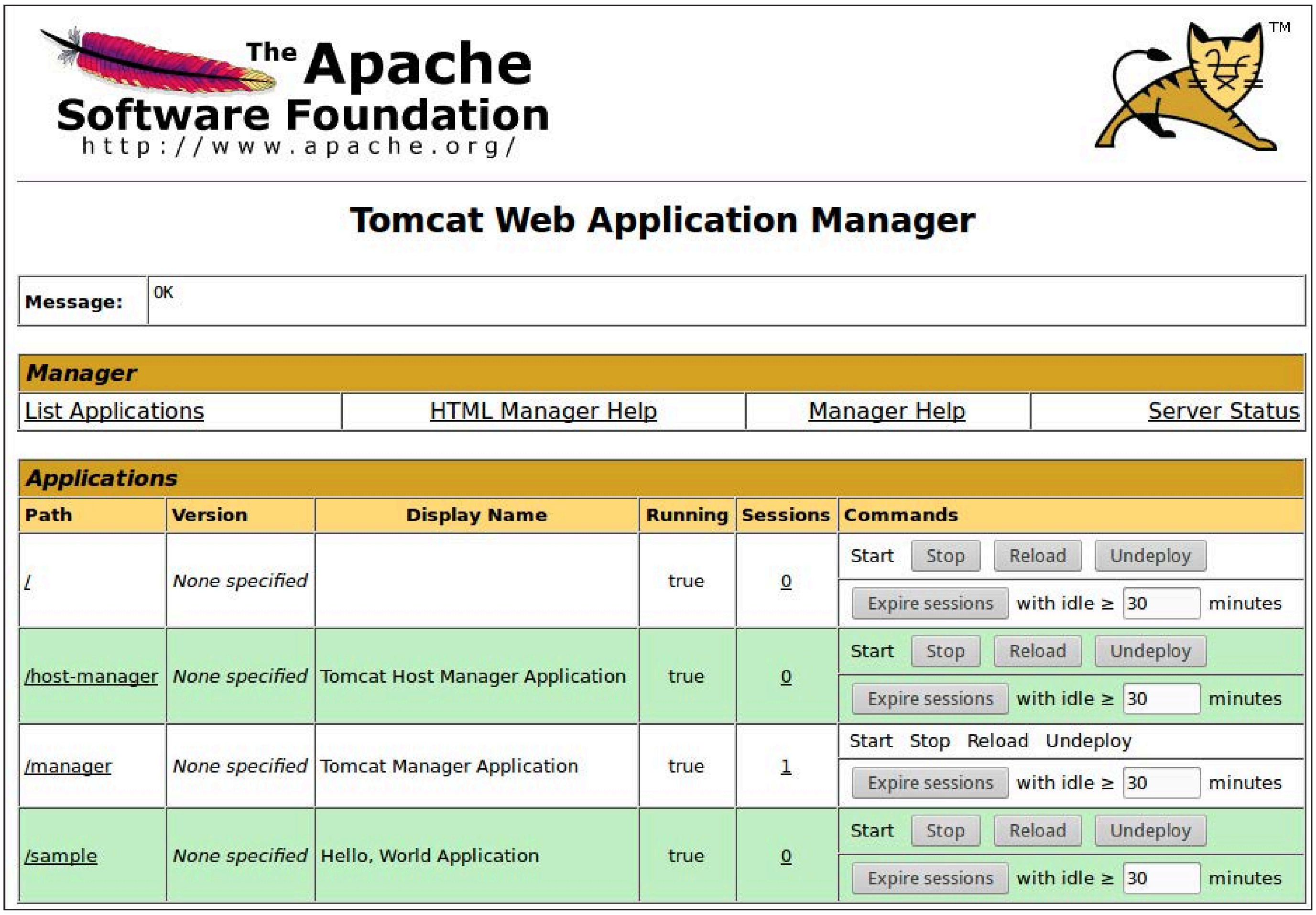
Let's deploy a WAR file to see how it works and see what actually happens. For my sample, I downloaded a sample WAR file from the Apache Foundation. If you don't have a favorite WAR file you want to deploy, you can play along at home by getting the sample WAR file from Apache. You can find one at **http://tomcat.apache.org/tomcat-6.0-doc/appdev/sample/sample.war**.

In order to deploy a WAR file, you need to copy it to the webapps folder. Its location may vary based on your distribution (or operating system, if you're a Windows user), but a quick search of your system should turn it up. In my Ubuntu server, the webapps folder is located at /var/lib/tomcat7/webapps/.

Tomcat will "hot deploy" a WAR file if it's copied to that folder, so simply do:

```
sudo cp sample.war /var/lib/tomcat7/webapps/
```

and you should be done. Nothing will be



Tomcat Web Application Manager

Message:	OK

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified		true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/sample	None specified	Hello, World Application	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes

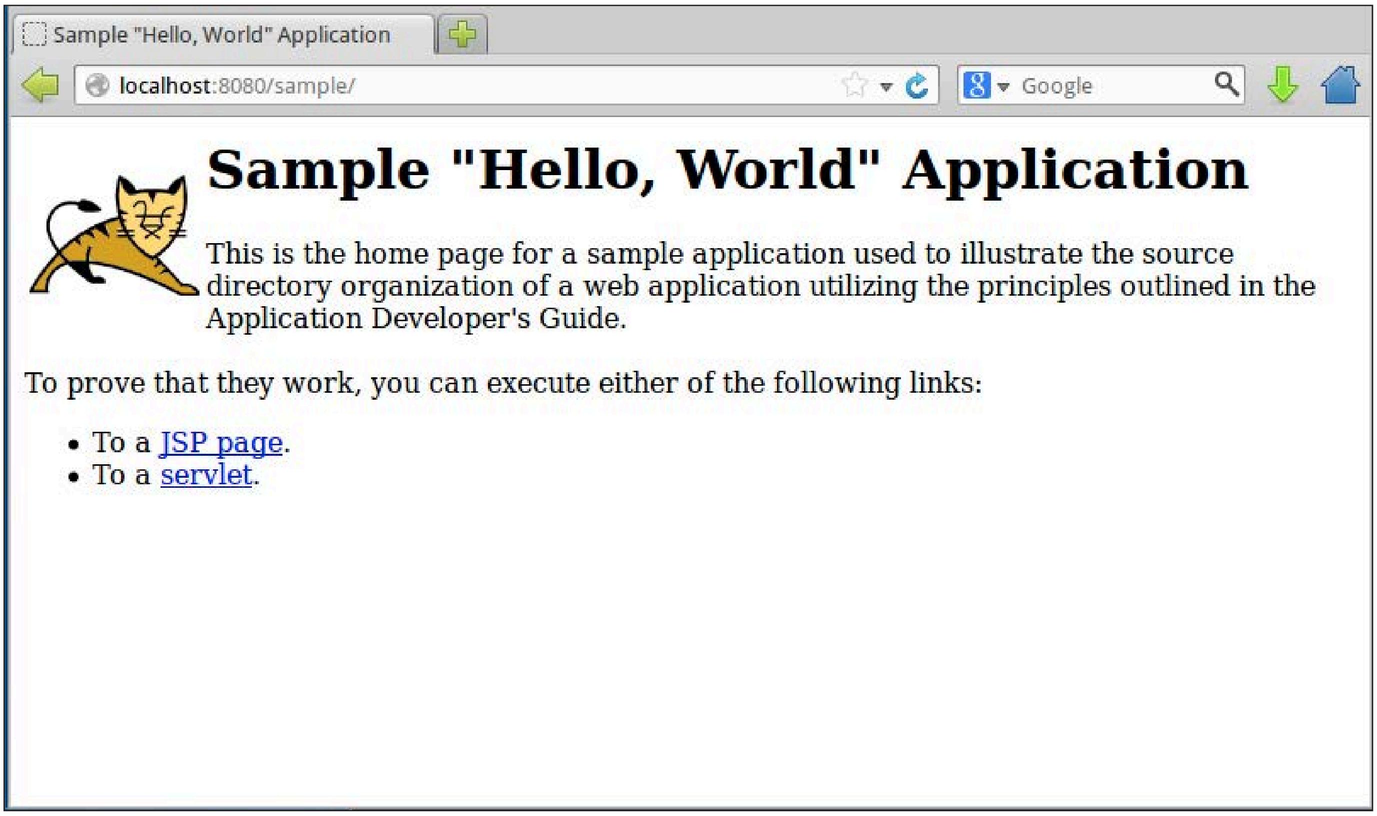
Figure 3. The new app shows up in the Application Manager. Success!

returned on the command line, but the Tomcat logs should show the application being deployed. Going back to the Web-based management app from Figure 2, you'd now see something like Figure 3, which shows the sample.war file fully deployed and active.

Now, if you go to http://localhost:8080/sample/, you should see the sample application fully deployed and working (Figure 4). Deploying other WAR files is basically the same process. Undeploying them can be done simply by deleting the WAR file from the webapps folder or clicking "Undeploy" in the Web-based management software. In fact, if you remember from my initial look at the management software, you actually can use the Web interface to deploy a WAR file, without ever needing to find the webapps folder on your own. This doesn't work for really large WAR files, as uploading via Web browser does have size limitations, but for most deployments, the management GUI works just fine.

A Few More Tips

Out of the box, Tomcat works pretty well and is fairly straightforward. Two



Sample "Hello, World" Application

This is the home page for a sample application used to illustrate the source directory organization of a web application utilizing the principles outlined in the Application Developer's Guide.

To prove that they work, you can execute either of the following links:

- To a JSP page.
- To a servlet.

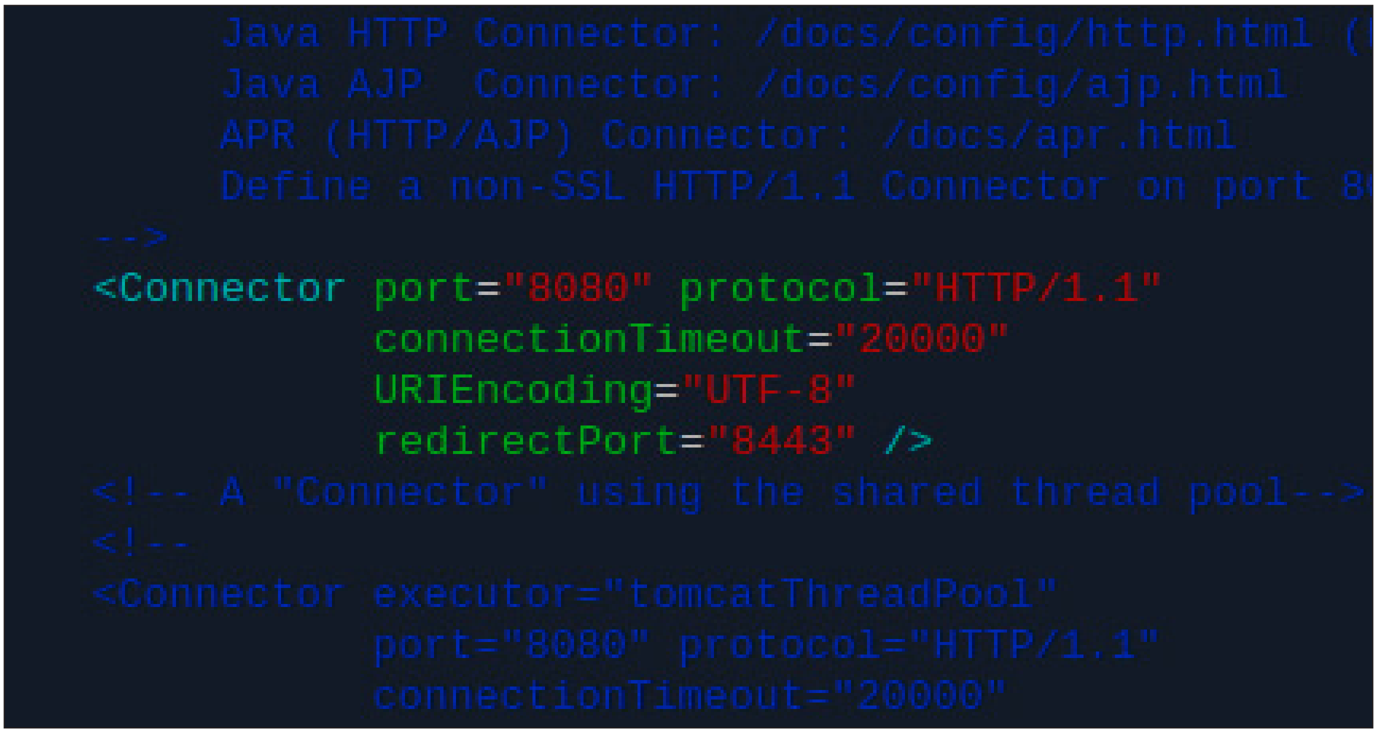
Figure 4. I was going to deploy a game, but I figured a sample application was more appropriate.

of the most common requests are to change the port to 80 so that it functions without clients needing to enter port numbers into their browser. The other common request I get is to have an application deployed at the root level, so no folder name (like the /sample above) is required to access a single-purpose server. Thankfully, both are pretty simple.

Changing Ports: On your server, locate the server.xml file for your Tomcat install. On my Ubuntu box, that's in /etc/tomcat7/. In the server.xml file, you'll find a section that looks like Figure 5. If you change the port "8080" to "80", it will tell the Tomcat server to listen on port 80 instead of the default 8080.

Unfortunately, because Tomcat runs as a nonroot user, telling it to listen on port 80 doesn't give the dæmon permissions to listen on the privileged port. For that, you need to grant permission. It may be a slightly different procedure on various distributions, but for Ubuntu, it's pretty straightforward. First, make sure the authbind package is installed. It most likely was installed with Tomcat, but just be sure it's installed and working. Then, edit /etc/default/tomcat7 and look for the line referencing `AUTHBIND`. (It's probably on the bottom of the file.) Uncomment the line, and change it to:

```
AUTHBIND=yes
```



```
        Java HTTP Connector: /docs/config/http.html (
        Java AJP  Connector: /docs/config/ajp.html
        APR (HTTP/AJP) Connector: /docs/apr.html
        Define a non-SSL HTTP/1.1 Connector on port 8
    -->
    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               URIEncoding="UTF-8"
               redirectPort="8443" />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
               port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
```

Figure 5. The syntax is confusing, but thankfully, you just need to change "8080" to "80".

Then, restart Tomcat:

```
sudo service tomcat7 restart
```

You should see no errors, and when you try to go to http://localhost/sample, you should see the same sample you deployed earlier, but this time, served from port 80. Note that there can be only a single dæmon listening on port 80, so you can't have the Apache Web server listening on port 80 (its default), and then expect Tomcat to work on the same port. Changing Tomcat to port 80 makes sense only if it's the only server trying to serve content on port 80. Hopefully, that makes sense.

Deploying an Application at the Root Level: This last tidbit actually is quite simple, but it wasn't obvious to me at first. If you look in your webapps folder, you'll see there is a ROOT folder. That ROOT folder is what holds the "It Works!" page Tomcat delivers by default. If you want to put your application at the root level, it's just a matter of deleting that ROOT folder:

```
sudo rm -rf /var/lib/tomcat7/webapps/ROOT
```

(Note: be careful with `rm -rf`; that's some powerful stuff.)

Then, change the name of your WAR file to ROOT.war.

After that, simply deploy the WAR file like you did with the sample.war file, and it will be accessible at the root level of the Tomcat server—pretty simple!

There's More, So Much More

This article is intended to get you started with Apache's Tomcat dæmon. There are plenty of tricks you can do in order to server Java applications alongside standard Web sites. You also can secure Tomcat with an SSL certificate just like a Web server. Once you understand how Tomcat works and how it's set up, it's far less scary to deploy Java apps in your production environment.

Next month, I plan to go further into setting up a heterogeneous environment that incorporates Java and standard Web apps. Plus, I want to give some tips on how to do all that from a single IP address. If you found this article interesting, you should be excited for next month!■

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via e-mail at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

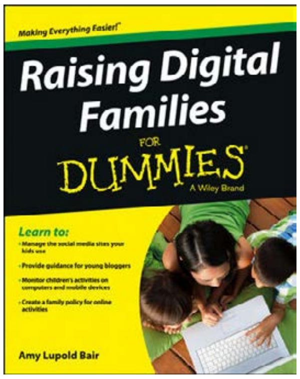
Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

Gumstix Inc.'s Overo and DuoVero COMs

Gumstix Inc. is encouraging embedded designers seeking a speedier way to bring advanced products to market to explore two newly upgraded computers-on-modules (COMs): the DuoVero Zephyr and the Overo TidalSTORM. The DuoVero COM now features wireless functionality on the Zephyr product series by way of a Wi2Wi module for 802.11 b/g/n Wi-Fi and Bluetooth. The DuoVero is based on the Texas Instruments OMAP4430 and gives electronic designers the same core technology that powers today's smartphones, tablets and other media-rich mobile devices. Meanwhile, the new TidalSTORM COM upgrades the Overo series to 1GB of RAM—twice the memory enables significantly better performance and results in an effective Linux graphical device. Rounding out the upgrade is the DaVinci Digital Video processor, which provides enhanced capabilities for memory-intensive graphics applications.
http://www.gumstix.com

CloudAccess's CloudIDM/AM

The new developments in the CloudIDM/AM solution by CloudAccess are intended to help IT administration regain and maintain control of every single user's unique relationship to the IT environment. CloudIDM/AM's function is to provide seamless integration between enterprise identity management and access control (single sign-on/access management) from the cloud. In this new release, CloudAccess has taken the next evolutionary step to integrate the key capabilities of provisioning, multifactor authentication and role-based workflow management with an advanced single sign-on for SaaS and legacy applications—and manage it all from the cloud. New so-called "game-changing" developments include multi-directional password synchronization between the identity management solution, Active Directory and/or LDAP, a federated gateway to control access to any application or Web site, dynamic self-service portal and centralized GUI-based workflow automation. CloudAccess says that its solution currently is the only product on the market that presents these features from a multi-tenant cloud.
http://www.cloudaccess.com

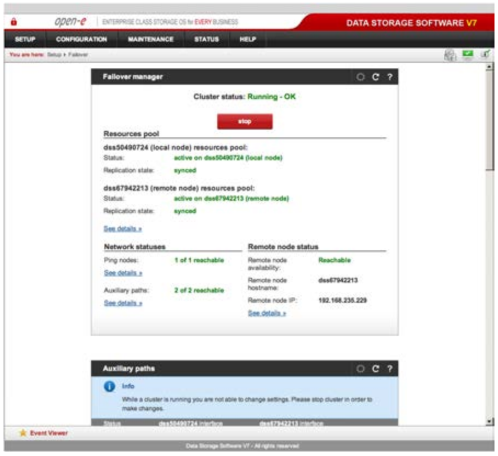

Making Everything Easier!
Raising Digital Families for Dummies
A Wiley Brand
Learn to:
Manage the social media sites your kids use
Provide guidance for young bloggers
Monitor children's activities on computers and mobile devices
Create a family policy for online activities
Amy Lupold Bair

Amy Lupold Bair's *Raising Digital Families For Dummies* (Wiley)

Go ahead, overlook the technical content in *Raising Digital Families For Dummies* by Amy Lupold Bair, because it's Computing 101. What tech-savvy parent couldn't use the Dr Phil-like parenting advice contained in its pages? Although most *Linux Journal* readers had PCs as youngsters—TI-99/4As, Commodore 64s and the like—this youngest generation never will know life without smartphones, Facebook and today's "always-connected" lifestyle. Bair's book is an easy-to-understand guide that helps parents get up to speed on everything they need to know in order to manage their children's on-line and gadget activity. The book offers guidance for managing mobile devices, social media and the Internet before it manages you! Also featured are tips and advice for establishing family rules for technology use and how to best handle situations when rules are broken. Other topics include monitoring software for computers and mobile devices, advice for handling cyberbullies, safe social networks for children, a guide to blogging and podcasting and useful on-line resources.

http://www.wiley.com

Open-E Data Storage Software



Cluster status: Running - OK

Resources pool

dss50490724 (local node) resources pool:
Status: active on dss50490724 (local node)
Replication state: synced

dss67942213 (remote node) resources pool:
Status: active on dss67942213 (remote node)
Replication state: synced

Network statuses
Ping nodes: 1 of 1 reachable
Auxiliary paths: 2 of 2 reachable

Remote node status
Remote node availability: Reachable
Remote node hostname: dss67942213
Remote node IP: 192.168.235.229

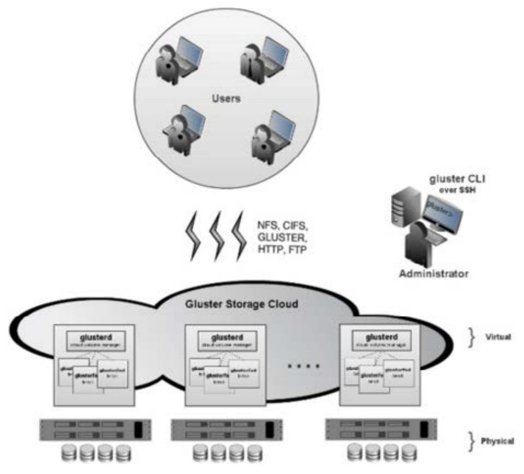
While a cluster is running you are not able to change settings. Please stop cluster in order to make changes.

The Open-E Data Storage Software (DSS), now a more feature-rich v7, is a Linux-based data storage software used for building and managing centralized data storage servers—NAS and SAN. Open-E's philosophy with DSS is to offer an all-in-one universal storage system that allows businesses of all sizes to turn any commodity server into an enterprise data-storage appliance for their big data, cloud and other enterprise applications. This new DSS v7 adds Hyper-V Cluster support for Windows Server 2008 R2 and 2012, which provides optimization of virtual storage environments, thus enabling customers to set up a virtualized storage environment with no single point of failure. This support adds to DSS's Active-Active Failover for iSCSI Volumes feature, which offers a self-validation functionality that checks all critical settings on each node, provides enhanced cluster security and fully utilizes all processing power of the data storage servers.

http://www.open-e.com

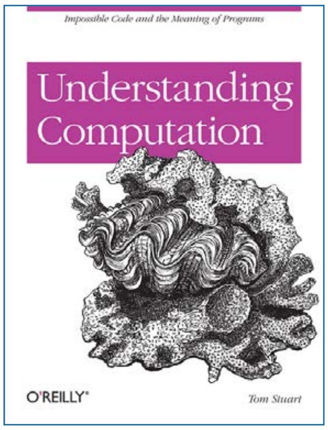
Logic Supply's LGX MK100 1U Rackmount Case

Specialized systems provider Logic Supply refers to its new MK100 1U Rackmount Case as "the most versatile 1U rackmount case we've seen" and "the last 1U rackmount case you'll ever need". Designed by LGX Systems, Logic Supply's engineering arm, the MK100 case offers mounting flexibility, world power compatibility and customizable faceplate options. The MK100 is constructed of high-quality, nickel-plated steel that protects against electromagnetic and radio frequency interference. To enhance longevity, the case comes equipped with dual long-life MagLev fans that provide plenty of airflow across even high-power Core i solutions. LGX's own fan mufflers ensure quiet operation and optimize airflow while also improving reliability by reducing strain on the fans. These features are a part of MK100's modular design, which allows for easy customization, with standard faceplate options including any combination of the following: four 2.5" drive hot-swap bays, an LCD panel with multifunction buttons and a slimline optical drive.
http://www.logicsupply.com and http://www.lgxsystems.com



Red Hat's GlusterFS

Our sources at Red Hat informed us of an update to GlusterFS, the open-source, distributed filesystem capable of scaling to several petabytes (or 72 brontobytes, to be exact) while handling thousands of clients. The new GlusterFS 3.4 release adds features and enhancements in cloud, virtualization and performance to the filesystem's existing high reliability, scalability and data mobility for users and application developers. New capabilities include QEMU integration with libgfapi, performance improvements for VM image storage for KVM, oVirt virtualization management GUI support, support for SSL-based encryption of "in-flight" data, multithreaded glusterd for better performance and NFSv3 ACL support, among others.
http://www.gluster.org


Impossible Code and the Meaning of Programs
Understanding Computation
O'REILLY
Tom Stuart

Tom Stuart's *Understanding Computation* (O'Reilly)

Programmers with little or no formal training in computer science who want to understand computation theory and programming language design can crack open *Understanding Computation* by Tom Stuart. In his book, Stuart tackles theoretical computer science in an engaging, practical way and in a context programmers will recognize, helping them to appreciate why these ideas matter and how they can inform one's day-to-day programming. Rather than use mathematical notation or an unfamiliar academic programming language like Haskell or Lisp, this book uses Ruby in a reductionist manner to present formal semantics, automata theory and functional programming with the lambda calculus. Also included is an exploration of the universal Turing machines that led to today's general-purpose computers.
http://www.oreilly.com

Linux Professional Institute's Jobs Initiative

Think of Linux Professional Institute's (LPI's) Jobs Initiative on oDesk as a "SaaS for job seekers". LPI is a well-known Linux certification organization; oDesk is a well-known on-line marketplace for contract employment. The goal of this workforce development initiative is to help Linux and open-source professionals connect with work opportunities via oDesk, a process that is aided by the opportunity for LPI-certified professionals to display their badges on their oDesk profiles. LPI notes that oDesk originally approached LPI after seeing growth in demand for certified Linux and open-source professionals. The initiative allows for free registration of LPI alumni in various groups corresponding to specific certification levels LPIC-1, LPIC-2 and LPIC-3, which enables businesses to hire IT professionals by level of LPI certification. LPI alumni who register and work on oDesk also will benefit from a payment guarantee on hourly jobs.
http://www.lpi.org and http://www.odesk.com

Please send information about releases of Linux-related products to newproducts@linuxjournal.com or New Products c/o *Linux Journal*, PO Box 980985, Houston, TX 77098. Submissions are edited for length and content.

HARDWARE

Exploring the Samsung ARM Chromebook 3G

Chromebook: frivolous toy or useful tool? Bill breaks it down for you. BILL CHILDERS

Back in late 2010, Google announced a "Chromebook"—a low-cost, entry-level netbook that would run Google's own operating system, ChromeOS. Google's vision of ChromeOS, although based on Linux, basically would be a giant Web browser, with all the apps on the machine running in the browser. ChromeOS would be a nearly stateless computer, with all the user's apps based in Google's cloud, running the Google Apps suite.

Google's first stab at this was the CR-48: an Intel Atom-powered netbook with 2GB of RAM and 16GB of Flash. The CR-48 wasn't a powerhouse by any means, but it had a couple cool things going for it. First, it came with 100MB of free 3G service a month. Second, it had a "developer mode" that allowed users to break free of the strict Chrome-based browser jail and expose the chewy Linux center. A CR-48 in developer mode became a usable machine for a lot of people, because the machine pretty much became a small Linux laptop.

Today—the Samsung ARM Chromebook

Fast-forward a couple years, and the first real Chromebook products are hitting the market. Quite a few Chromebooks exist today, but all of them are Intel-based (either Atom or Celeron). In late 2012, however, Samsung released an ARM-based Chromebook. This little guy is

Figure 1. The Samsung ARM Chromebook, atop an iPad for Scale

different in lots of ways—primarily, it beats its bigger brothers in size and battery life, without compromising much on performance. Speaking of performance, let's go over the specifications of the XE303—the first non-Intel powered Chromebook:

- Dual-Core, Samsung Exynos 5 ARM CPU (Cortex A15, 1.7GHz).
- 2GB of RAM (not upgradable, soldered to the mainboard).
- 16GB SSD/Flash-based disk (also not upgradable, soldered to the mainboard).
- ARM Mali T-604 Quad-Core GPU.
- Wi-Fi (802.11 a/b/g/n).
- Bluetooth 3.0 (sadly, no Bluetooth 4.0 here).

This is a functioning computer—with a keyboard—that is playing in the tablet price-point and space. The keyboard is actually quite good for a machine of this size.

- 11" LCD screen at 1366x768 resolution.
- One USB 2.0 port, one USB 3.0 port, one HDMI-out port and one SD card slot.
- Optional 3G modem (CDMA, on the Verizon network in the US, equipped on this model).
- 6.5 hours of (rated) battery life.
- Dimensions: 2.4 lbs (1.10 kg), 11.4" x 8.2" x 0.68" (289.6 x 208.5 x 17.5mm).
- Bonus: 100GB of Google Drive included for two years.
- Bonus: 100MB/month of Verizon 3G service included for two years (3G model only).

Looking at the specifications individually, they aren't mind-blowing. However, when you put them together in a $329 computer ($249 if you get the non-3G version), that's something different altogether. This is a functioning computer—with a keyboard—that is playing in the tablet price-point and space. The keyboard is actually quite good for a machine of this size. Compared to other netbook-class machines I've owned, this keyboard is far and away the best I've used amongst low-end hardware. The display isn't outstanding, but it's serviceable and does the job. The build quality is quite good, although the unit is built from plastic—there's no metal in it—keeping the price point down where it needs to be. The unit doesn't creak, pop or feel cheap, despite the fact that it's one of the least expensive computers I've owned.

Surprisingly enough, the little dual-core ARM CPU does a great job of keeping things running—and running silently, as there's no hard disk or fans to make noise in this little guy. The Chromebook had no problems playing 720p video

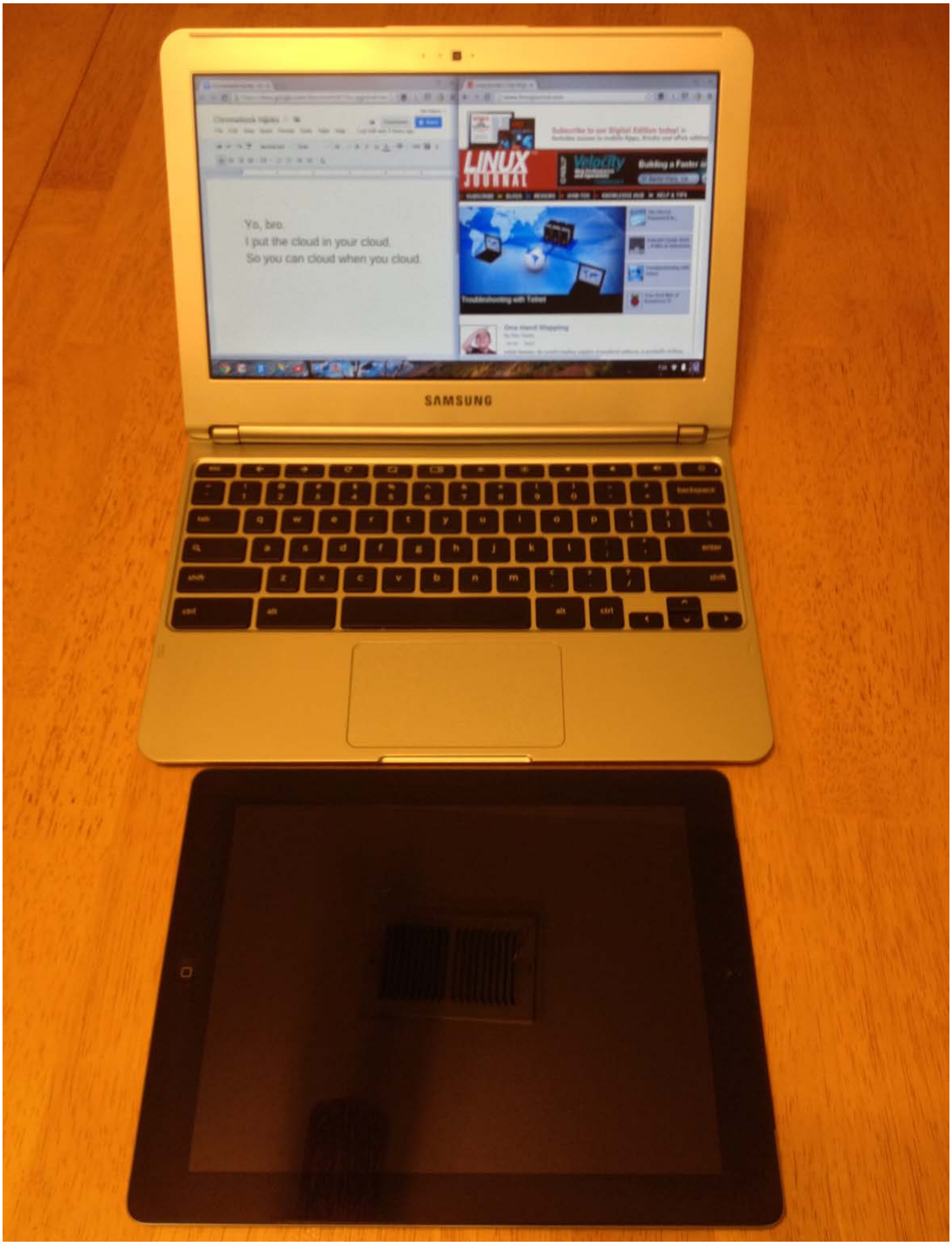

Yo, bro.
I put the cloud in your cloud.
So you can cloud when you cloud.
LINUX JOURNAL
SAMSUNG

Figure 2. The Chromebook in Use, iPad Provided for Scale

streams from Netflix (where the Google and Netflix wizards have HTML5 streaming running for the Chromebook without needing anything but the Chrome browser). The Chromebook also performed just fine while watching H264-encoded video streaming via Apache off a machine on my local network, and it did it while maintaining its advertised battery life of 6.5 hours. But, it's running something that's not a "real OS"—ChromeOS. Is ChromeOS usable?

Getting Down and Dirty with ChromeOS

From a basic user standpoint, ChromeOS is actually a good thing. It comes with an "office suite" included in the price of the machine, is virus-/malware-resistant out of the box and is easy to update. ChromeOS on the Chromebook boots to a user-login screen in about eight seconds, which is plenty fast for the average user. It's basically a Google Apps appliance, and when you think of the Chromebook within that frame of reference, it's a really nice little machine.

I loaned the Chromebook to my ten-year old son, who knows just enough about computers to be dangerous. In no time flat, he was surfing the Web, and he discovered you didn't need a Netflix "app" to watch Netflix movies on the Chromebook. If you're looking for a Web-surfing machine, the Chromebook more than fills that particular need.

In all, for most of the tasks an "average" user would do, this little Chromebook is a decent portable machine, so long as that user has a full-fledged desktop somewhere—and is comfortable with the Google Apps suite. However, power users (the readers of this magazine) will find the "jail" of ChromeOS to be limiting. However, that jail can be broken.

Power-User Time: Hacking the Chromebook

As mentioned previously, it was possible to put the old CR-48 Chromebook prototype into "developer mode" and extend its capabilities through various hacks. Thanks to Google and Samsung for retaining this ability on the new ARM-based Chromebook, as it's necessary to hack the unit.

There are a couple different ways to extend the Chromebook to run a full Linux environment, but I think the most flexible and easiest to install is an Ubuntu chroot environment, using the Crouton script by David Schneider (an engineer at Google).

Crouton is the best thing since sliced bread—it makes it easy to install Ubuntu's ARM port in a chroot on the Chromebook's filesystem. Although it's possible to repartition and reformat the Chromebook's disk (see the Another Way to Hack the Chromebook sidebar for more information), doing that brings its own set of risks and challenges. By keeping ChromeOS on the machine, and installing Ubuntu in a chroot, you can let ChromeOS handle the power management, networking and other hardware-related tasks. Going with a chroot-based install also lets you install multiple chroots side by side, play with them and delete them with impunity if you don't like the way they're behaving.

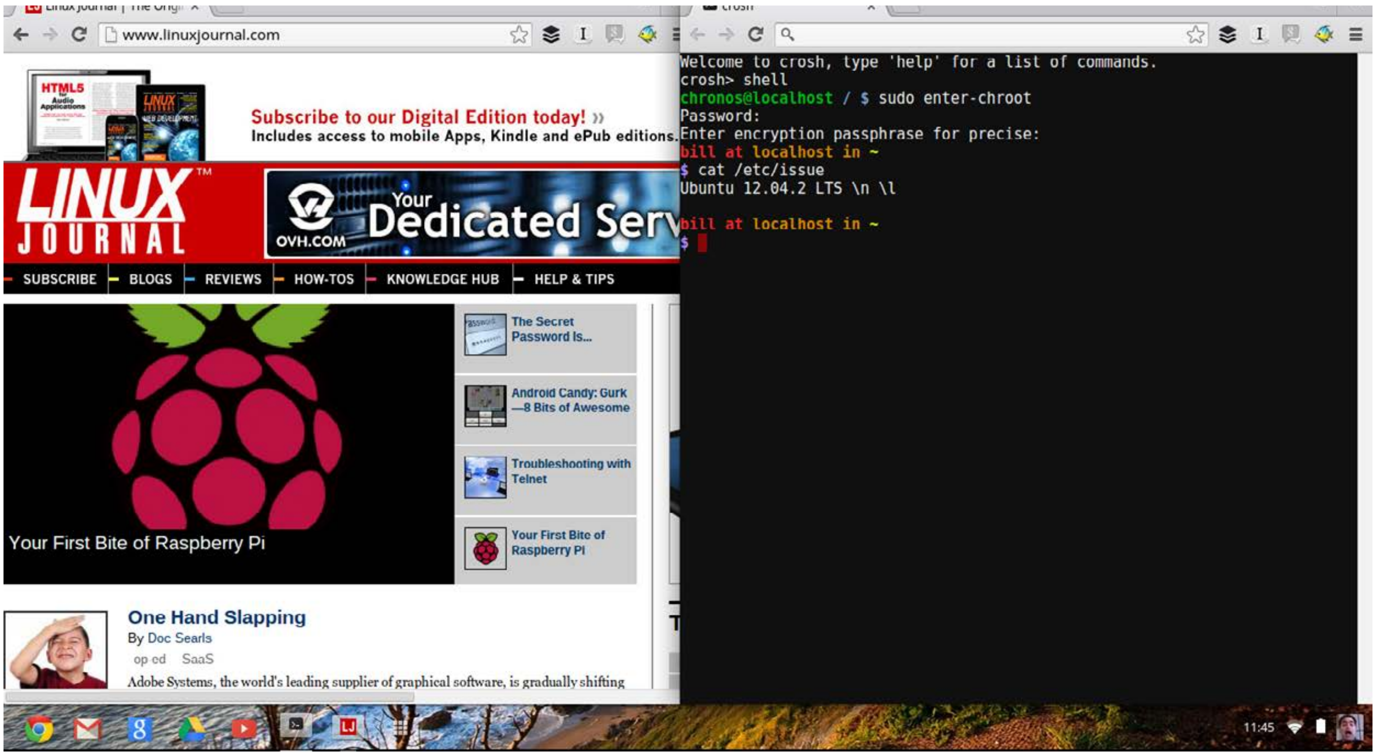
Getting Started with Crouton

First, please refer to the links provided in the Resources section of this article, and back up any data you have on your Chromebook. The tools and methods described here may change between the time of this writing and press time, so please, ensure that you're following the latest set of instructions.

Now that you're all caught up, start by putting your Chromebook into developer mode. This is done by starting the Chromebook in "Recovery" mode first: hold the Esc and Refresh keys down while turning on the system. Once the "Recovery" screen appears, invoke developer mode by pushing Ctrl-D (there's no prompt, you have to know the magic sequence). The system will ask you to confirm, and then it will wipe its internal disk, finally rebooting into developer mode. As a consequence, the system always will show an "OS verification is off" screen on power-up, and you'll need to press Ctrl-D to clear that boot screen.

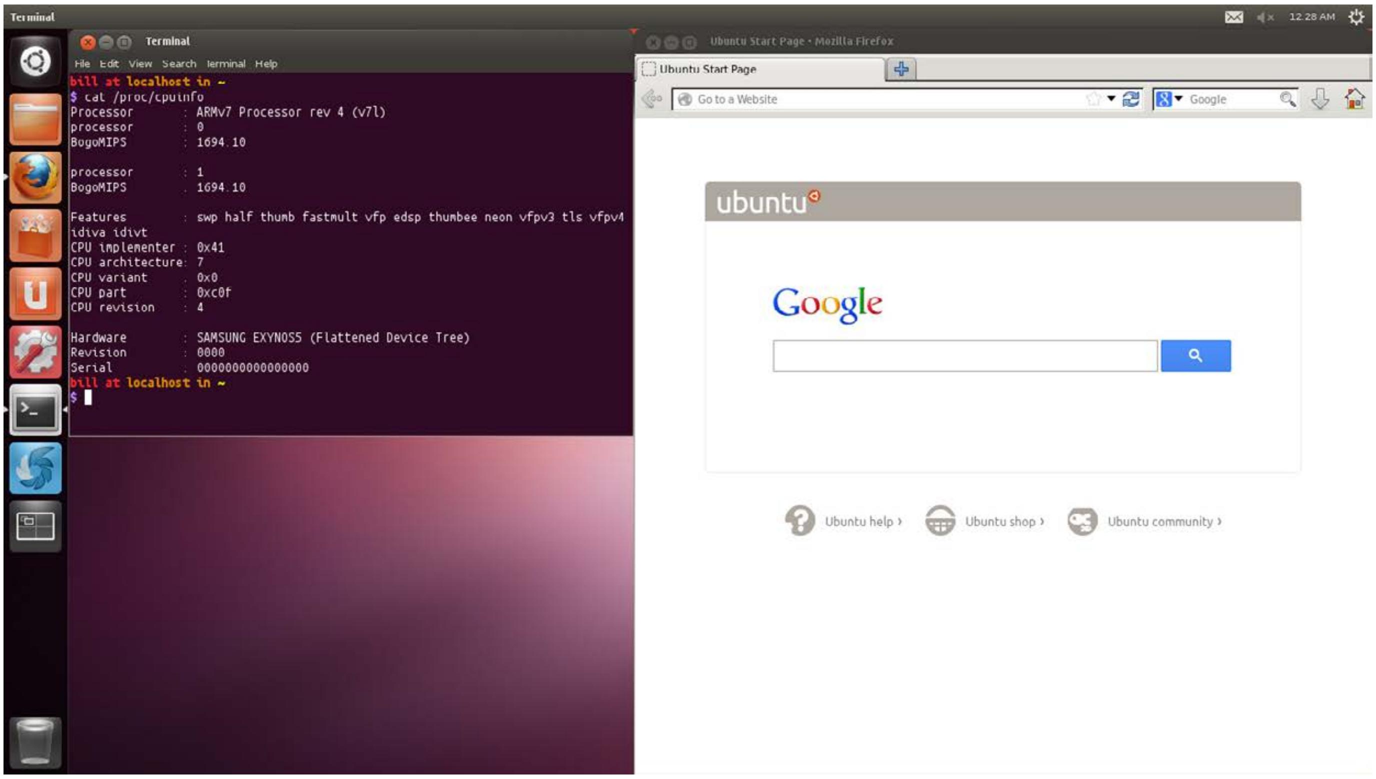
Once the Chromebook is booted in developer mode, open the Chrome browser and go to the Crouton download URL: http://goo.gl/fd3zc. The Crouton script will download automatically in your Downloads directory on the Chromebook.

Now that you've got the Crouton script on your Chromebook, you need to launch it. Start a crosh shell (Chrome Shell) in the Chrome browser by pushing Ctrl-Alt-T, then type `shell` and press Enter. You're almost ready to make your first chroot. Crouton has the concept of "targets" as install definitions. If you want an Xfce-equipped system, you can install with the `-t xfce` target. Because you're installing Ubuntu here, you may as well try out a full-blown Ubuntu—let's



```
Welcome to crosh, type 'help' for a list of commands.
crosh> shell
chronos@localhost / $ sudo enter-chroot
Password:
Enter encryption passphrase for precise:
bill at localhost in ~
$ cat /etc/issue
Ubuntu 12.04.2 LTS \n \l

bill at localhost in ~
$
```

Figure 3. Running a Ubuntu chroot in a Shell alongside ChromeOS



```
bill at localhost in ~
$ cat /proc/cpuinfo
Processor       : ARMv7 Processor rev 4 (v7l)
processor       : 0
BogoMIPS        : 1694.10

processor       : 1
BogoMIPS        : 1694.10

Features        : swp half thumb fastmult vfp edsp thumbee neon vfpv3 tls vfpv4
idiva idivt
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc0f
CPU revision    : 4

Hardware        : SAMSUNG EXYNOS5 (Flattened Device Tree)
Revision        : 0000
Serial          : 0000000000000000
bill at localhost in ~
$
```

Figure 4. Ubuntu Running on the Chromebook

install Unity. Do that by running Crouton with the following command from the crosh shell:

```
sudo sh -e ~/Downloads/crouton -t unity
```

Crouton will prompt you along the way, asking questions, such as user name and password, and it'll download all the necessary files to build the chroot. Then, it'll go and do the heavy lifting of actually building and installing the chroot.

Once it's all done (how long it takes depends on your Internet download speed, because Crouton downloads all its bits from the Ubuntu mirrors), you can start your Ubuntu session from crosh simply by running `sudo enter-chroot`. If you want to start Unity, once you're in your chroot, just run `startunity`, and Unity will start on a second virtual terminal. You then can flip between Unity and ChromeOS by pressing Ctrl-Alt-Shift-Forward or Ctrl-Alt-Shift-Backward. (The "Forward" and "Backward" keys are special Chromebook keys, about where F1 and F2 would be on a normal PC's keyboard.) You also can install any other software *inside* your new chroot, simply by using Ubuntu's Software Manager, apt-get or your favorite Debian package installer.

There are other versions of Crouton, in the event that Ubuntu's not your speed (see the Resources section for a port of Crouton that installs Arch Linux). Don't forget, if you want to experiment, or have more than one chroot for different reasons, you can. Simply install with your desired target(s) and name them differently with Crouton's `-n` option.

Another Way to Hack the Chromebook

If a Linux chroot doesn't appeal to you, and you want to excise all signs of ChromeOS from your Chromebook, there's a way to do that. The ChrUbuntu project (linked to in the Resources section) has a script that will allow you to repartition your Chromebook to install and run Ubuntu natively in a dual-boot configuration—or wipe your Chromebook and run Ubuntu as the sole OS. There's also an option to run Ubuntu off of an external disk (SD card or USB stick), if you choose. However you wish to extend your Chromebook, you have many options.

Conclusion

The Chromebook, out of the box, isn't exactly the most usable of machines for most *Linux Journal* readers. It's geared more for the basic or novice user, particularly one who's already using the Google Apps suite of products. However, once the Chromebook is extended in developer mode and has a full Linux chroot on it, it's an extremely good mobile companion—particularly for the price. Its long battery life, light weight and silent operation lend itself well to working just about anywhere. If you decide to put up the money and get the 3G model, you should factor the two years of free 3G and Google Drive into the equation too. Google Drive and 100MB of Verizon 3G would be priced at about $15 a month, or $360 for two years. With the 3G model going for $329, this is indeed a bargain.

If you're looking for an inexpensive, highly portable, Linux computer—and you don't mind hacking on it a little bit—check out the Samsung ARM Chromebook.■

Bill Childers is an IT Manager in Silicon Valley, where he lives with his wife and two children. He enjoys Linux far too much, and probably should get more sun from time to time.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

Resources

Google CR-48 Wikispace: **http://cr-48.wikispaces.com**

Developer Mode on the ARM Chromebook: **http://www.chromium.org/chromium-os/developer-information-for-chrome-os-devices/samsung-arm-chromebook**

Crouton—Ubuntu Chroot Installer for Chromebook: **http://github.com/dnschneid/crouton**

Crouton Download URL: **http://goo.gl/fd3zc**

ChrUbuntu—Dual-Booting the Chromebook: **http://chromeos-cr48.blogspot.com**

Chroagh—Crouton Ported to Arch Linux: **http://github.com/drinkcat/chroagh**

22nd USENIX SECURITY SYMPOSIUM

WASHINGTON, D.C. • AUGUST 14–16, 2013

The USENIX Security Symposium brings together researchers, practitioners, system administrators, system programmers, and others interested in the latest advances in the security of computer systems and networks.

USENIX Security '13 will feature:

Keynote Address

"Dr. Felten Goes To Washington: Lessons from 18 Months in Government"
Edward W. Felten, *Director, Center for Information Technology Policy, and Professor of Computer Science and Public Affairs, Princeton University; former Chief Technologist, U.S. Federal Trade Commission*

A 3-day Technical Program including:

- Paper presentations on large-scale systems security, attacks, mobile security, and applied crypto
- Invited talks by industry leaders
- Poster session
- Rump session
- Birds-of-a-Feather sessions (BoFs)

Register by July 22 and Save!
www.usenix.org/sec13

Co-Located Workshops Include:

EVT/WOTE '13: 2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections
August 12–13, 2013

CSET '13: 6th Workshop on Cyber Security Experimentation and Test
August 12, 2013

HealthTech '13: 2013 USENIX Workshop on Health Information Technologies
Safety, Security, Privacy, and Interoperability of Health Information Technologies
August 12, 2013

LEET '13: 6th USENIX Workshop on Large-Scale Exploits and Emergent Threats
August 12, 2013

FOCI '13: 3rd USENIX Workshop on Free and Open Communications on the Internet
August 13, 2013

HotSec '13: 2013 USENIX Summit on Hot Topics in Security
August 13, 2013

WOOT '13: 7th USENIX Workshop on Offensive Technologies
August 13, 2013

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Stay Connected...

twitter.com/usenixsecurity
www.usenix.org/youtube
www.usenix.org/gplus
www.usenix.org/facebook
www.usenix.org/linkedin
www.usenix.org/blog

Popular IPv6 Tunnel Brokers

This article reviews a number of popular IPv6 tunneling protocols to help you decide which one to use when getting your LAN IPv6-connected. Read about the strengths and weaknesses of each protocol, and learn how to transition your IPv4 LAN to a dual-stack IPv4/IPv6 network.

IGOR PARTOLA

IPv6 is the next-generation version of the IP protocol, designed to replace IPv4 and solve the problem of IPv4 address exhaustion. While IPv4 designates 4 bytes per IP address (roughly 4 billion addresses), IPv6 designates 16 addresses (4 billion to the fourth power). Indeed, there are so many addresses that a typical home LAN would be allocated a /64 network, or 4 billion squared.

IPv6 already is available through some ISPs, such as AT&T and Comcast, in select areas. However, not everyone is lucky enough to have the connectivity. For starters, wired copper connections require DOCSIS 3 modems, which not all ISPs provision. Second, your home or small-office router must support IPv6 as well. Finally, although most desktop/laptop OSes do support IPv6, certain devices, such as Microsoft's Xbox, do not. On the mobile-phone and tablet front, things are mixed as well. For example, T-Mobile has support only for certain phones (**http://support.t-mobile.com/message/140209#140209**), and the setup process is not straightforward.

On the positive side, core Internet services, such as DNS and large data centers and networks, are much closer to IPv6 deployment. Companies like Google, Facebook and WikiMedia have enabled IPv6 across their servers and are processing a fraction of their traffic over the new protocol. As IPv4 addresses become more and more expensive, IPv6 is becoming more and more attractive to networks and Web site operators alike.

All in all, IPv6 deployment is a chicken-and-egg problem. The more consumers there are with IPv6-enabled, or even IPv6-only connectivity, the more that Web site operators will support IPv6. On the other hand, as more IPv6-accessible content becomes available, IPv6 connectivity will become a priority and a competitive point for ISPs. Google has been collecting statistics about IPv6 connectivity, both overall and by country (**http://www.google.com/ipv6/statistics.html**), and it estimates that as of March 2013, more than 1% of its traffic comes in the form of IPv6. Although the figure may seem low, that number grew 500% since March 2011 and shows no signs of slowing down.

Amidst this grand shift of underlying protocols on the Internet, you can help by becoming one more Internet denizen with IPv6 connectivity, even if your ISP does not support IPv6 natively. This is accomplished by setting up a tunnel between your IPv6-only network and a tunnel broker. A tunnel broker is an entity that provides you with an IPv6 address set (a subnet) and a way to communicate to one of its "points of presence", or PoPs, via a special tunneling protocol.

In this article, I review my experiences and observations of several of these tunneling protocols. I also look at their advantages and disadvantages, and provide basic instructions on how to get started with them.

Equipment You Will Need

Generally, the most important bit of equipment you will need is an IPv6-capable router. Some consumer-grade routers now come with IPv6 support, but you also may be able to "upgrade" your existing router by using third-party firmware, such as Tomato, DD-WRT, OpenWRT and so on. When selecting an IPv6-capable router, it is important to make sure that it also comes with an IPv6 firewall. A major change from IPv4 to IPv6 for consumer LANs is that with IPv4, your NAT used to act like a firewall. Indeed, by default, none of the hosts behind your NAT were accessible from the outside world. If you wanted to have access to a Web server, a game server or NAS from the Internet, you had to "forward ports" in your router's settings.

This isn't the case with IPv6. In fact, the whole point of IPv6 is to provide at least one globally unique address to every host on your network. This opens up your printer to everyone in the world who has IPv6 connectivity. Although this isn't the most likely scenario, others might connect to it and use it as their own. Things like computers with Webcams, NAS devices and so on are all susceptible to this issue.

Despite this grim outlook, remember that NAT never was meant to be a security system. That is the job of your firewall. It is simple to disallow forwarding of any incoming connections by default and then enable specific ports to be allowed using a good firewall. Thus, when looking for a new IPv6-capable router, make sure to get one with a built-in firewall control panel. This will make it a lot easier than dealing with ip6tables and the like on the command line.

IPv6 Addresses

IPv6 addresses are composed of 128 bits. A typical IPv6 address looks like 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Note that leading zeros may be omitted like so: 2001:db8:85a3:0:0:8a2e:370:7334. Further, groups of zeros may be skipped once in the address by using the double : symbol, like so: 2001:db8:85a3::8a2e:370:7334. In this manner, an address like 2001:0db8:85a3:0000:0000:0000:0000:0001 could be written as 2001:db8:85a3::1.

An IPv6 also is logically divided into the network prefix and subnet parts, with the first and second 64 bits devoted to each, respectively. Therefore, in the example above, the network prefix is 2001:db8:85a3:0, and the local part is :0:8a2e:370:7334. Currently, an IPv6 address cannot be routed to more than one subnet using a /64 network prefix. If you are interested in that type of functionality, you need to secure a larger network prefix, such as a /48 (where only the first 48 bits are the network prefix).

To learn more about IPv6 addressing, read the IPv6 address page on Wikipedia (**http://en.wikipedia.org/wiki/IPv6_address**).

In a lot of cases, a modern operating system can use what is called "stateless autoconfiguration", where given a subnet prefix and the MAC address of the network interface, a globally unique IPv6 address, may be constructed. This is done by combining those two pieces of data using a predictable algorithm. Although convenient, this may create privacy concerns, as a Web site operator will know your computer's globally unique MAC address. To address this, some operating systems implement what is called IPv6 privacy extensions, where in addition to the static IPv6 address, a dynamic one is generated periodically (typically every hour) and used for making outgoing connections. This does not guarantee privacy (after all, your network/subnet prefix is always known), but it does let you hide your MAC address. This functionality typically is disabled on servers and routers, and enabled on end-user hosts.

Note: if you are used to remembering IPv4 addresses of certain hosts, your life will become a lot more difficult when using IPv6. Instead, you should use DNS to set up easy-to-remember hostnames. Using DNS with IPv6 is fairly straightforward, but an in-depth discussion of it warrants its own article.

6in4, aka Static Tunnel, aka Protocol 41

If you are lucky enough to have a stable, or at least infrequently changing IPv4 address from your ISP, and you have full control of your router, this method is likely to be the most stable option. Static 6in4 tunnels typically are established with one of the popular tunnel brokers, such as Hurricane Electric's Tunnel Broker (**https://tunnelbroker.net**), SixXS (**https://sixxs.net**, pronounced "Six Access") or one of many others. When you sign up for an account with one of the tunnel brokers, you are given the opportunity to create a tunnel and a subnet that you then can use.

Now, some terminology is in order:

- Tunnel ID: typically, a tunnel will have a unique identifier that you may use in various configuration files.

- Server IPv4 address: the IPv4 endpoint of the tunnel broker's PoP. Most tunnel brokers let you find a PoP that is closest to you geographically, in order to reduce latency.

- Server IPv6 address: the IPv6 address of the server to which you will be connecting. This address is unique to your tunnel.

- Client IPv4 address: your router's IPv4 address.

- Client IPv6 address: your router's IPv6 address.

- Routed /64 or routed /48: this is the subnet within which you will assign addresses to all your hosts. Somewhat confusingly, your router also may have an address in this subnet.

Note: the routed /64 and the client/server IPv6 address usually will differ by 1 in one of the digits. Do not confuse them in your configuration files, as using one in place of the other will not work.

Once you set up your account, you will be given the values for all of the above-mentioned items. You then will be able to put them into the router settings for IPv6. Once everything is plugged in, your router should have IPv6 connectivity. This means you can use an on-line ping6 tool (such as **http://www.subnetonline.com/pages/ipv6-network-tools/online-ipv6-ping.php**) to test that your client IPv6 address is accessible. Similarly, if you have command-line access to your router, you can run the following to test that your router can access IPv6-enabled hosts:

```
$ ping6 google.com
```

The biggest consideration for choosing a tunnel broker is whether it has a PoP fairly close to you.

The next step is to assign addresses to the rest of the hosts on your LAN. Under the hood, this typically is done using a tool called radvd, which stands for router advertisement dæmon. This dæmon periodically will send out ICMPv6 packets indicating which IPv6 network prefix may be used by the hosts on the LAN. Each IPv6-capable host on your LAN then will use this prefix, along with its MAC address to determine a static, globally accessible IPv6 address for itself, and will set up a proper routing table.

Note that you likely also will want to enable the IPv6 firewall on your router prior to enabling address assignment for the rest of your LAN. Typically, you will want to allow all connections that originate on your routed subnet and disallow all others. You also may want to enable SSH access if that is something you use, for either specific hosts or all the hosts on the network, by allowing connections to port 22.

If your router does not natively support IPv6, you can use a different computer as your IPv6 router. There are more detailed guides elsewhere that will walk you through the necessary steps for each OS and distribution, but the steps roughly are as follows:

1. Set the computer as the DMZ machine in your router (necessary for it to be reachable via protocol 41).

2. Enable the IPv6 firewall on the computer.

3. Enable IPv6 routing.

4. Set up the IPv6 tunnel.

5. Set up LAN address assignment to give IPv6 addresses to the rest of your LAN.

Small computers with low power consumption, such as Raspberry Pis, can be used for this purpose if you don't want to replace your current router.

Which tunnel broker should you choose? That question requires you to do some homework. The biggest consideration for choosing a tunnel broker is whether it has a PoP fairly close to you. If you live in Los Angeles, but your PoP is in Amsterdam, your latency will suffer.

Hurricane Electric's tunnel broker and SixXS both have a large number of PoPs in the US, Europe and Asia. Wikipedia has a larger list of tunnel brokers (**http://en.wikipedia.org/wiki/List_of_IPv6_tunnel_brokers**). SixXS is the largest, judging by the PoP size (52 as of March 2013).

There is also a difference in the style of service provided by these operators. Hurricane Electric's tunnel broker is a free, no-strings-attached service with good support from Hurricane Electric. The company provides many other paid services, and tunnelbroker.net is, if anything, a way for people to be aware of its other offerings.

SixXS, on the other hand, is a nonprofit service directed at end users. It comes with some strings attached, the biggest of which is that all requests for tunnels, subnets and so on must be reviewed by a SixXS volunteer before being approved. There also is a point system, where you may lose points, and therefore privileges, if you do not keep your static tunnel running. Do not let this discourage you. Although it seems like there are more hoops to jump through, there also is a helpful community behind this project, and in some cases, it is easier to set up the tunnel and keep it running, because SixXS provides its AICCU software—a dæmon that will keep all your configuration up to date at all times.

6in4 Heartbeat

The 6in4 heartbeat tunnel is a variant of the static 6in4 tunnel, where a

A Note about Protocol 41

Protocol 41 is an IP-level protocol where an IPv6 header follows an IPv4 header immediately. It is on the same level in the OSI model as TCP and UDP. The upside of it is that it minimizes protocol overhead, thereby potentially improving performance. The downside is that it is a somewhat unusual protocol that may be blocked by your ISP or a network operator that is between your ISP and the PoP you are trying to use. Typically, ISPs do not make a habit of blocking protocol 41, but it is possible that they might. If you can confirm that this is the case, you may want to ask your ISP to unblock it, or look into a tunnel type that is not based on protocol 41.

heartbeat allows for dynamically changing IPv4 address to be updated with the server. If you do not have a static IPv4 address from your ISP, this may be a very good choice for you, because it will mean uninterrupted service, even if your IPv4 address gets updated. Currently, only SixXS offers this type of tunnel; however, Hurricane Electric's tunnel broker provides a way to update your IPv4 address periodically over HTTP to achieve a similar effect. The beauty of the 6in4 heartbeat protocol is that when your IPv4 address is not changing, you are using a static IPv6 tunnel with all of its benefits, such as having a permanent IPv6 address for all your hosts, low protocol overhead and a stable PoP. In many cases, if your router supports it, the 6in4 heartbeat from SixXS is the best choice.

Anything-in-Anything, aka AYIYA

AYIYA is different from 6in4 in that it does not use protocol 41. Instead, IPv6 packets are encapsulated in UDP/IPv4 packets. This introduces more overhead, as there are more packet headers. It does provide some very nice benefits, such as increased security and being able to traverse NAT boundaries, and it is harder for your ISP to block.

Currently, SixXS is the only tunnel broker that provides AYIYA tunnels, but because it is the largest tunnel broker, that should not deter you. AYIYA with SixXS is also exempt from losing credits with SixXS if they are not up and running, because they are specifically designed to be dynamic.

Unlike 6in4, each packet over AYIYA is signed using a shared secret, allowing for greater security. As with other tunnel types from SixXS, its official client, AICCU, automatically brings up this type of tunnel for you.

Because AYIYA uses UDP, it is able to traverse most typical types of NAT without any special cooperation from the router. This means that if your router does not support IPv6 tunneling natively, or if you do not have control over your router, you still may be able to get your entire LAN IPv6-connected using this protocol. To do so, you would designate a computer inside your LAN/NAT as the IPv6 router and firewall, bring up an AYIYA tunnel on it, and then use radvd to give addresses to the rest of the hosts on your network.

Another use case for AYIYA is when you want IPv6-connectivity on the go. A typical coffee shop likely will not provide you with IPv6 access. If you want to connect to IPv6 resources from a coffee shop, Internet café or the like, you could set up an AYIYA tunnel on your laptop and turn it on only when you are at one of these locations. Once again, because AYIYA looks like UDP/IPv4, the network operator is not likely to block it. In this use case, you would be able to connect to your hosts on your home or office network, because they will have static IPv6 addresses. Indeed, this is how I typically work remotely: `ssh` to my workstation at home or at the office from whatever network I happen to be on.

AYIYA does have some disadvantages. Due to its nature, it has more protocol overhead. Additionally, when traversing a NAT, your router will treat this traffic like any other UDP traffic, so if are using some type of QoS rule set, you may want to adjust it to allow AYIYA to have a sufficiently high priority (since it can carry all types of traffic of its own, from DNS requests to streaming video). Generally, a static tunnel is preferred vs. AYIYA, but certain distinct use cases make AYIYA more appealing.

6to4 Tunneling

6to4 tunnels (note that 6to4 is different from 6in4, although the two are related) are based on a protocol that is meant to be used during the transitional period, when both IPv4 and IPv6 are enabled. A 6in4 tunnel uses protocol 41, just like 6in4. The difference is that instead of having an explicit tunnel provider with an account, a routed subnet and so on, your host's address is determined purely by its public IPv4 address. Each host on the public IPv4 Internet is given an IPv6 address that starts with 2002. For example, if your public IPv4 address was 192.0.2.4, your IPv6 address would be 2002:c000:0204::1/48 (192 becomes c0, 0 becomes 00, 2 and 4 become 02 and 04, respectively). You can choose a suffix other than ::1. Indeed, using this method, you have an entire /48 network, giving you an 80-bit address space to play with. You can use this address space to allocate addresses to the rest of your LAN.

On the surface, it would seem that a 6to4 tunnel is preferable to a static 6in4 tunnel. Indeed, it is easier to configure, requiring commands using /sbin/ip on Linux, for example. It does, however, have several downsides that generally would disqualify it from being used in practice. First, your IPv6 address would depend on your IPv4 address. This means that if your IPv4 address changes, so does the IPv6 one. If you are using a static 6in4 tunnel, your IPv6 address always is the same, which can be a big deal if you want to access hosts inside your LAN while on the go.

Additionally, when using 6to4, you are going to route your traffic through an anycasted IPv4 address, which may or may not be ideally located compared to where you are. With a static 6in4 tunnel, you get to choose your PoP, and your network becomes predictable.

All in all, 6to4 is best avoided, unless other methods of obtaining IPv6 connectivity are not available.

Teredo

Teredo is a last-resort protocol. It should not be used in any but the most dire circumstances. However, it is important to cover it here, because it does provide distinct benefits in some limited circumstances. Teredo's claim to fame is its ease of use. There are no accounts to set up, no tunnels to configure, and no firewall rules to write. Teredo is an IPv6 in UDP/IPv4 protocol, similar to AYIYA. The difference is that it also includes a clever address assignment scheme whereby your public IPv4 address and your private IPv4 address on your LAN are combined and used as a part of the IPv6 address. Theoretically, even if your computer is behind several layers of NAT, you still can get IPv6 connectivity using Teredo.

Although the above upside to Teredo is a big win for this tunnel type, it has several downsides. First, not all types of NAT are supported. Specifically, a Teredo tunnel cannot traverse a symmetric NAT. While this NAT type typically is not found on consumer LANs, it is possible to encounter it on networks you do not control.

Second, only a single address is assigned to each tunnel endpoint. Therefore, it is impossible to connect several hosts to the IPv6 Internet using a single Teredo tunnel. Instead, each host would have to run its own Teredo tunnel. Although most desktop/laptop operating systems are capable of this, other devices, such as mobile phones and tablets, are not able to do this, even though they are perfectly capable of connecting to an IPv6-enabled LAN.

Third, the IPv6 address you get is not static. In essence, you can connect to IPv6-enabled resources, but others cannot connect to you. This point alone may override the usefulness of Teredo for most LAN operators.

Finally, I have seen several coffee-shop networks where Teredo did not work, while AYIYA did. This may be an issue if this is your primary reason for using Teredo.

A typical use case for Teredo is when you do not already have an AYIYA tunnel set up with SixXS, are on the go and are trying to access hosts on your home/office LAN over IPv6. In that case, simply fire up a Teredo tunnel, and if there is no filtering on your on-the-go LAN, you will get IPv6 connectivity. On Linux, Teredo is implemented with a software package called miredo. Simply installing it on one of the popular distributions will bring up a Teredo tunnel.

Conclusion

If you have an interest in trying out IPv6 on your workstation or LAN in your home or office, you have many options. When choosing the tunneling method, your best bet is going to be a static 6in4 tunnel from Hurricane Electric's tunnel broker or a 6in4 heartbeat tunnel from SixXS. Another option you may consider will include AYIYA if you do not control your router, or if the router is incapable of IPv6 routing or if you are frequently on the go. Finally, 6to4 tunneling and Teredo are available as last-resort options when for whatever reason nothing else will work.

Happy IPv6 networking! ■

Igor Partola is an independent software developer specializing in scalable, distributed Internet applications. He has a particular interest in free and open-source software, as well as networking. He is often described by his friends as an "IPv6 nut", because he constantly is advocating that people get IPv6 access.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

Resources

After choosing your method of tunneling and getting accounts set up, you likely are going to need to read about setting up firewall rules. SixXS has a wiki page dedicated to ip6tables, the Linux IPv6 firewall (**http://www.sixxs.net/wiki/IPv6_Firewalling**). Note that if you are familiar with the IPv4 firewall, iptables, ip6tables will feel very familiar.

Additionally, you may want to learn more about radvd, the router advertisement dæmon. Most UNIX-like operating systems have radvd in their repositories. A good starting place for it is "Linux IPv6 HOWTO—Hints for IPv6-Enabled Dæmons" (**http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/hints-daemons-radvd.html**).

drupalize.me

Instant Access to Premium Online Drupal Training

- ✓ *Instant access to hundreds of hours of Drupal training with new videos added every week!*
- ✓ *Learn from industry experts with real world experience building high profile sites*
- ✓ *Learn on the go wherever you are with apps for iOS, Android & Roku*
- ✓ *We also offer group accounts. Give your whole team access at a discounted rate!*

Learn about our latest video releases and offers first by following us on Facebook and Twitter (@drupalizeme)!

Go to http://drupalize.me and get Drupalized today!

```
 *
 * Usage: $(obj).insertAtCursor(text);
 *   obj = a textarea or textfield
 *   text = a string to insert
 */
$.fn.extend({
  insertAtCaret: function(myValue){
    return $(this).each(function(){
      // If target element is hidden, don't do it.
      if ($(this).is(':hidden') || $(this).parents(':hidden'
        return;
      }
      if (document.selection) {
        this.fo
        sel = c
        sel.tex
        this.fo
      }
      else if (
        var sta
        var end
        var scr
        this.va
        this.fo
        // High
```

QUEUEING IN THE LINUX NETWORK STACK

Although most people focus on bandwidth as the main network performance metric, latency is just as important to perceived performance. In this article, Dan explains how overbuffering can cause high latencies and where this can occur in the Linux network stack.

DAN SIEMON

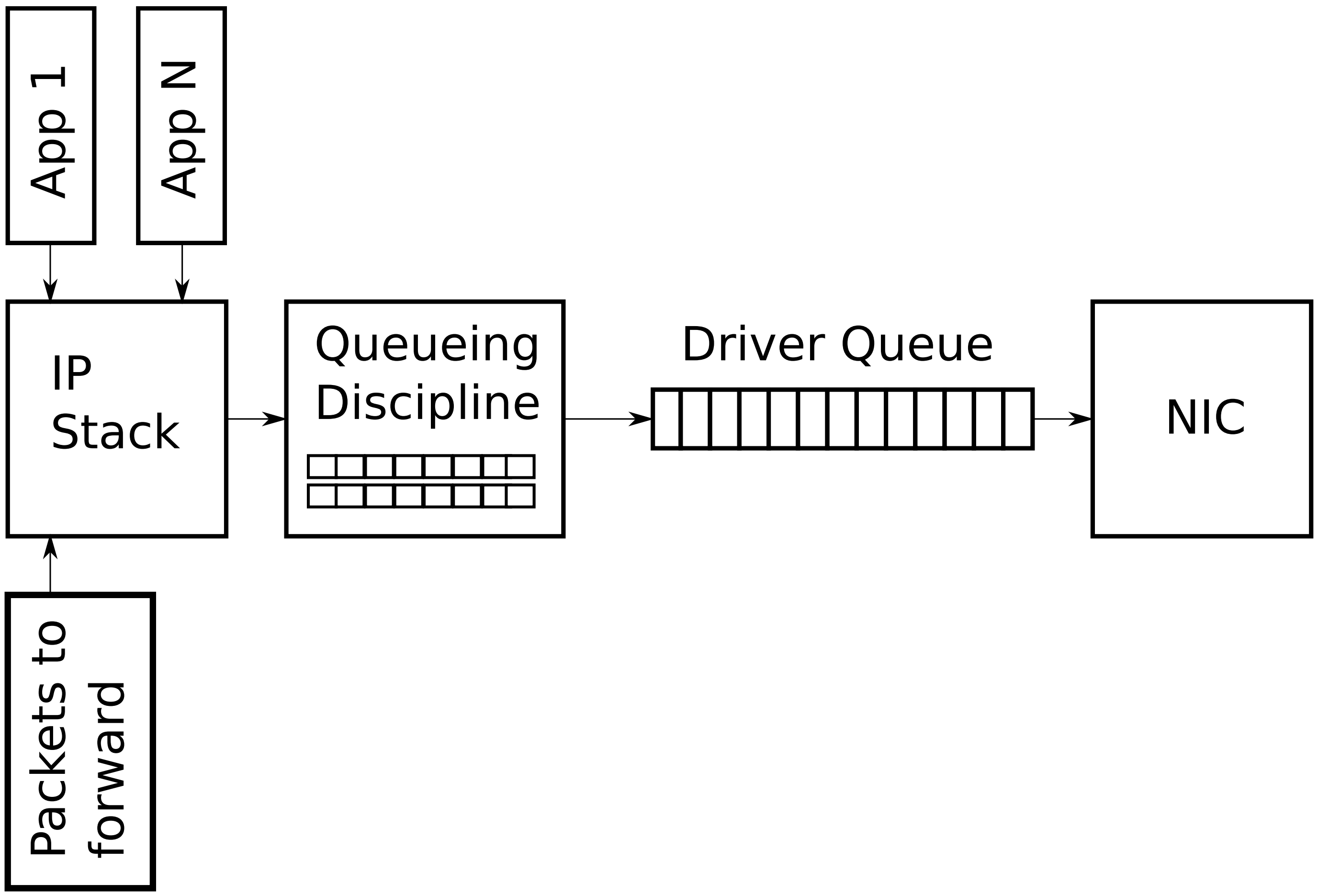


Figure 1. Simplified High-Level Overview of the Queues on the Transmit Path of the Linux Network Stack

Packet queues are a core component of any network stack or device. They allow for asynchronous modules to communicate, increase performance and have the side effect of impacting latency. This article aims to explain where IP packets are queued on the transmit path of the Linux network stack, how interesting new latency-reducing features, such as BQL, operate and how to control buffering for reduced latency.

Driver Queue (aka Ring Buffer)

Between the IP stack and the network interface controller (NIC) lies the driver queue. This queue typically is implemented as a first-in, first-out (FIFO) ring buffer (**http://en.wikipedia.org/wiki/Circular_buffer**)—just think of it as a fixed-sized buffer. The driver queue does not contain the packet data. Instead, it consists of descriptors that point to other data structures called socket kernel buffers (SKBs, **http://vger.kernel.org/%7Edavem/skb.html**), which hold the packet data and are used throughout the kernel.

The input source for the driver queue is the IP stack that queues IP packets. The packets may be generated locally or received on one NIC to be routed out another when the device is functioning as an IP router. Packets added to the driver queue by the IP stack are dequeued by the hardware driver and sent across a data bus to the NIC hardware for transmission.

The reason the driver queue exists is to ensure that whenever the system has data to transmit it is available to the NIC for immediate transmission. That is, the driver queue gives the IP stack a location to queue data asynchronously from the operation of the hardware. An alternative design would be for the NIC to ask the IP stack for data whenever the physical medium is ready to transmit. Because responding to this request cannot be instantaneous, this design wastes

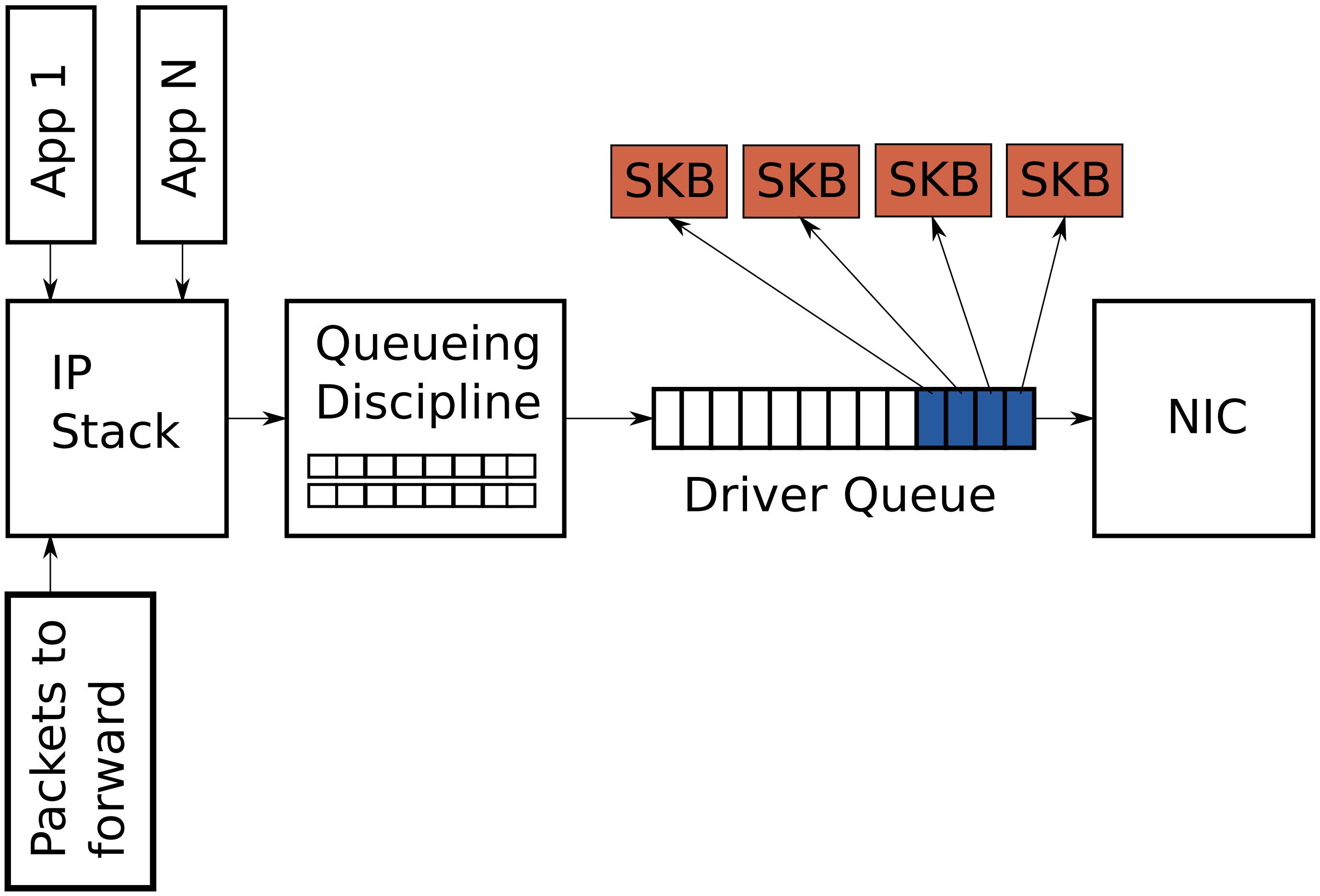


Figure 2. Partially Full Driver Queue with Descriptors Pointing to SKBs

valuable transmission opportunities resulting in lower throughput. The opposite of this design approach would be for the IP stack to wait after a packet is created until the hardware is ready to transmit. This also is not ideal, because the IP stack cannot move on to other work.

Huge Packets from the Stack

Most NICs have a fixed maximum transmission unit (MTU), which is the biggest frame that can be transmitted by the physical media. For Ethernet, the default MTU is 1,500 bytes, but some Ethernet networks support Jumbo Frames (**http://en.wikipedia.org/wiki/Jumbo_frame**) of up to 9,000 bytes. Inside the IP network stack, the MTU can manifest as a limit on the size of the packets that are sent to the device for transmission. For example, if an application writes 2,000 bytes to a TCP socket, the IP stack needs to create two IP packets to keep the packet size less than or equal to a 1,500 MTU. For large data transfers, the comparably small MTU causes a large number of small packets to be created and transferred through the driver queue.

In order to avoid the overhead associated with a large number of packets on the transmit path, the Linux kernel implements several optimizations: TCP segmentation offload (TSO), UDP fragmentation offload (UFO) and generic segmentation offload (GSO). All of these optimizations allow the IP stack to create packets that are larger than the MTU of the outgoing NIC. For IPv4, packets as large as the IPv4 maximum of 65,536 bytes can be created and queued to the driver queue. In the case of TSO and UFO, the NIC hardware takes responsibility for breaking the single large packet into packets small enough to be transmitted on the physical interface. For NICs without hardware support, GSO performs the same operation in software immediately before queueing to the driver queue.

Recall from earlier that the driver queue contains a fixed number of descriptors that each point to packets of varying sizes. Since TSO, UFO and GSO allow for much larger packets, these optimizations have the side effect of greatly increasing the number of bytes that can be queued in the driver queue. Figure 3 illustrates this concept in contrast with Figure 2.

Although the focus of this article is the transmit path, it is worth noting that Linux has receive-side optimizations that operate similarly to TSO, UFO and GSO and share the goal of reducing per-packet overhead.

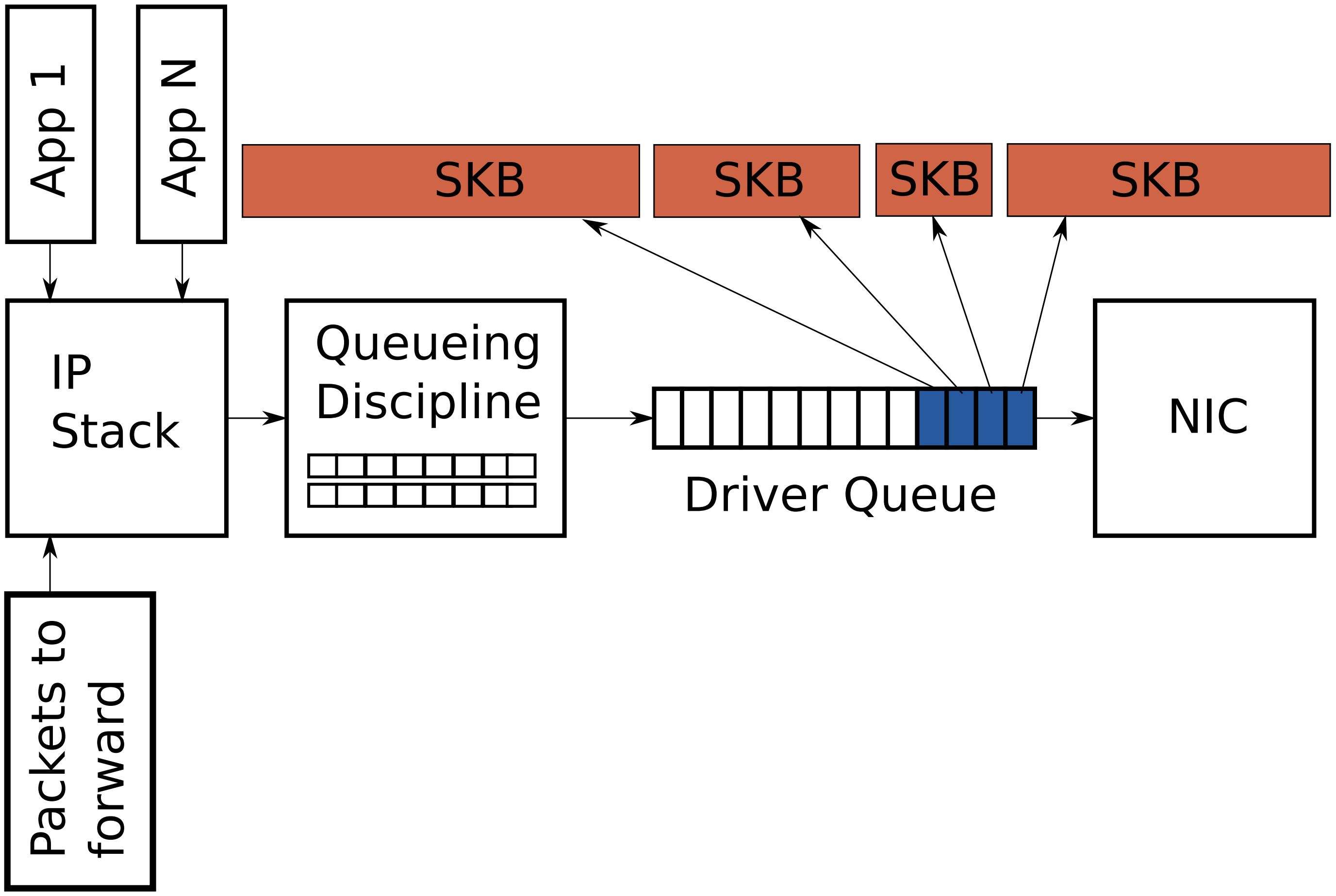


Figure 3. Large packets can be sent to the NIC when TSO, UFO or GSO are enabled. This can greatly increase the number of bytes in the driver queue.

Specifically, generic receive offload (GRO, **http://vger.kernel.org/%7Edavem/cgi-bin/blog.cgi/2010/08/30**) allows the NIC driver to combine received packets into a single large packet that is then passed to the IP stack. When the device forwards these large packets, GRO allows the original packets to be reconstructed, which is necessary to maintain the end-to-end nature of the IP packet flow. However, there is one side effect: when the large packet is broken up, it results in several packets for the flow being queued at once. This "micro-burst" of packets can negatively impact inter-flow latency.

Starvation and Latency

Despite its necessity and benefits, the queue between the IP stack and the hardware introduces two problems: starvation and latency.

If the NIC driver wakes to pull packets off of the queue for transmission and the queue is empty, the hardware will miss a transmission opportunity, thereby reducing the throughput of the system. This is referred to as starvation. Note that an empty queue when the system does not have anything to transmit is not starvation—this is normal. The complication associated with avoiding starvation is that the IP stack that is filling the queue and the hardware driver draining the queue run asynchronously. Worse, the duration between fill or drain events varies with the load on the system and external conditions, such as the network interface's physical medium. For example, on a busy system, the IP stack will get fewer opportunities to add packets to the queue, which increases the chances that the hardware will drain the queue before more packets are queued. For this reason, it is advantageous to have a very large queue to reduce the probability of starvation and ensure high throughput.

Although a large queue is necessary

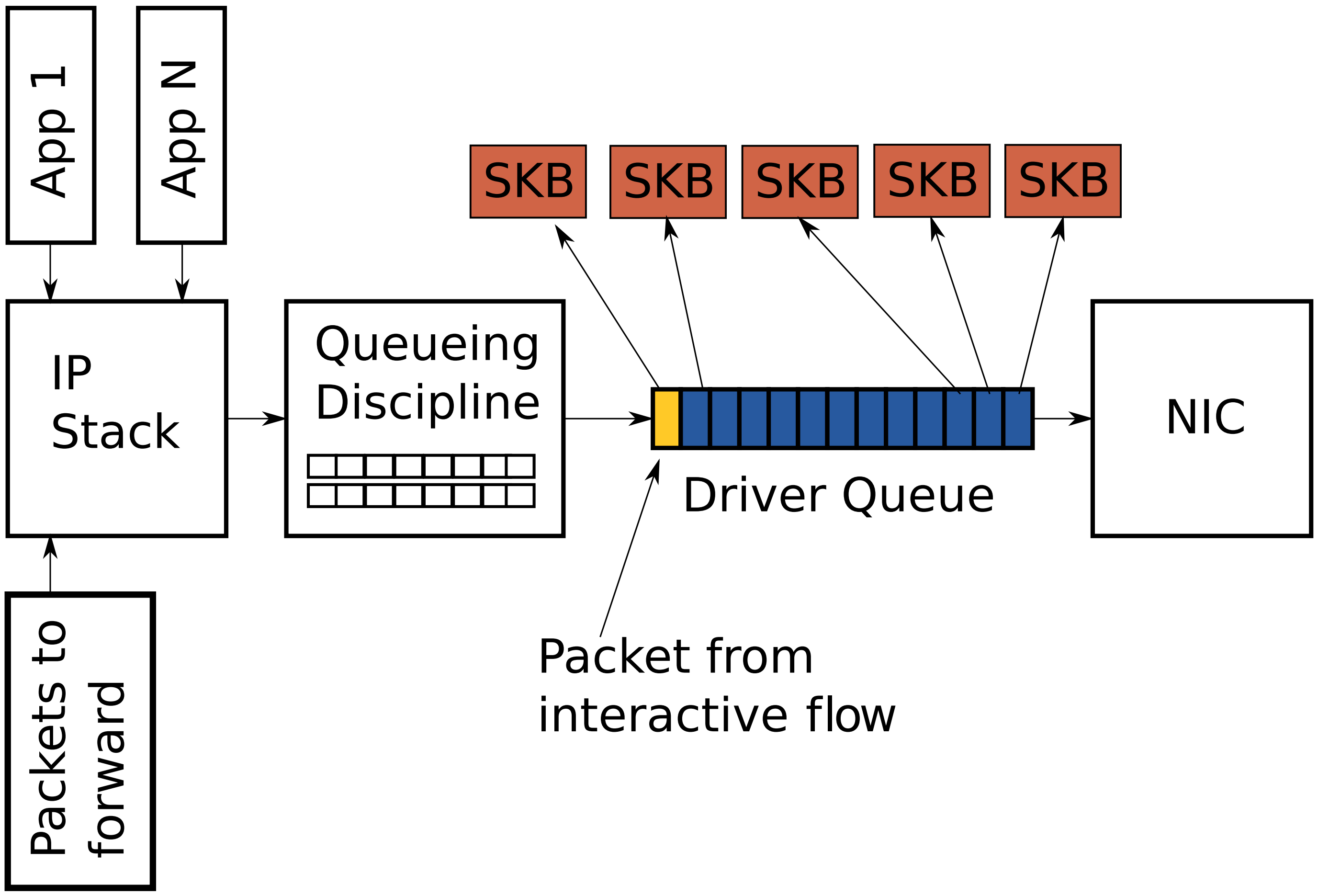


Figure 4. Interactive Packet (Yellow) behind Bulk Flow Packets (Blue)

for a busy system to maintain high throughput, it has the downside of allowing for the introduction of a large amount of latency.

Figure 4 shows a driver queue that is almost full with TCP segments for a single high-bandwidth, bulk traffic flow (blue). Queued last is a packet from a VoIP or gaming flow (yellow). Interactive applications like VoIP or gaming typically emit small packets at fixed intervals that are latency-sensitive, while a high-bandwidth data transfer generates a higher packet rate and larger packets. This higher packet rate can fill the queue between interactive packets, causing the transmission of the interactive packet to be delayed.

To illustrate this behaviour further, consider a scenario based on the following assumptions:

- A network interface that is capable of transmitting at 5 Mbit/sec or 5,000,000 bits/sec.
- Each packet from the bulk flow is 1,500 bytes or 12,000 bits.
- Each packet from the interactive flow is 500 bytes.
- The depth of the queue is 128 descriptors.
- There are 127 bulk data packets and one interactive packet queued last.

Given the above assumptions, the time required to drain the 127 bulk packets and create a transmission opportunity for the interactive packet is (127 * 12,000) / 5,000,000 = 0.304 seconds (304 milliseconds for those who think of latency in terms of ping results). This amount of latency is well beyond what is acceptable for interactive applications, and this does not even represent the complete round-trip time—it is only the time required to transmit the packets queued before the interactive one. As described earlier, the size of the packets in the driver queue can be larger than 1,500 bytes, if TSO, UFO or GSO are enabled. This makes the latency problem correspondingly worse.

Large latencies introduced by over-sized, unmanaged queues is known as Bufferbloat (**http://en.wikipedia.org/wiki/Bufferbloat**). For a more detailed explanation of this phenomenon, see the Resources for this article.

As the above discussion illustrates, choosing the correct size for the driver queue is a Goldilocks problem—it can't be too small, or throughput suffers; it can't be too big, or latency suffers.

Byte Queue Limits (BQL)

Byte Queue Limits (BQL) is a new feature in recent Linux kernels (> 3.3.0) that attempts to solve the problem of driver queue sizing automatically. This is accomplished by adding a layer that enables and disables queueing to the driver queue based on calculating the minimum queue size required to avoid starvation under the current system conditions. Recall from earlier that the smaller the amount of queued data, the lower the maximum latency experienced by queued packets.

It is key to understand that the actual size of the driver queue is not changed by BQL. Rather, BQL calculates a limit of how much data (in bytes) can be queued at the current time. Any bytes over this limit must be held or dropped by the layers above the driver queue.

A real-world example may help provide a sense of how much BQL affects the amount of data that can be queued. On one of the author's servers, the driver queue size defaults to 256 descriptors. Since the Ethernet MTU is 1,500 bytes, this means up to 256 * 1,500 = 384,000 bytes can be queued to the driver queue (TSO, GSO and so forth are disabled, or this would be much higher). However, the limit value calculated by BQL is 3,012 bytes. As you can see, BQL greatly constrains the amount of data that can be queued.

BQL reduces network latency by limiting the amount of data in the driver queue to the minimum required to avoid starvation. It also has the important side effect of moving the point where most packets are queued from the driver queue, which is a simple FIFO, to the queueing discipline (QDisc) layer, which is capable of implementing much more complicated queueing strategies.

Queueing Disciplines (QDisc)

The driver queue is a simple first-in, first-out (FIFO) queue. It treats all packets equally and has no capabilities for distinguishing between packets of different flows. This design keeps the NIC driver software simple and fast. Note that more advanced Ethernet and most wireless NICs support multiple independent transmission queues, but similarly, each of these queues is typically a FIFO. A higher layer is responsible for choosing which transmission queue to use.

Sandwiched between the IP stack and the driver queue is the queueing discipline (QDisc) layer (Figure 1).

This layer implements the traffic management capabilities of the Linux kernel, which include traffic classification, prioritization and rate shaping. The QDisc layer is configured through the somewhat opaque tc command. There are three key concepts to understand in the QDisc layer: QDiscs, classes and filters.

The QDisc is the Linux abstraction for traffic queues, which are more complex than the standard FIFO queue. This interface allows the QDisc to carry out complex queue management behaviors without requiring the IP stack or the NIC driver to be modified. By default, every network interface is assigned a pfifo_fast QDisc (**http://lartc.org/howto/lartc.qdisc.classless.html**), which implements a simple three-band prioritization scheme based on the TOS bits. Despite being the default, the pfifo_fast QDisc is far from the best choice, because it defaults to having very deep queues (see txqueuelen below) and is not flow aware.

The second concept, which is closely related to the QDisc, is the class. Individual QDiscs may implement classes in order to handle subsets of the traffic differently—for example, the Hierarchical Token Bucket (HTB, **http://lartc.org/manpages/tc-htb.html**). QDisc allows the user to configure multiple classes, each with a different bitrate, and direct traffic to each as desired. Not all QDiscs have support for multiple classes. Those that do are referred to as classful QDiscs, and those that do not are referred to as classless QDiscs.

Filters (also called classifiers) are the mechanism used to direct traffic to a particular QDisc or class. There are many different filters of varying complexity. The u32 filter (**http://www.lartc.org/lartc.html#LARTC.ADV-FILTER.U32**) is the most generic, and the flow filter is the easiest to use.

Buffering between the Transport Layer and the Queueing Disciplines

In looking at the figures for this article, you may have noticed that there are no packet queues above the QDisc layer. The network stack places packets directly into the QDisc or else pushes back on the upper layers (for example, socket buffer) if the queue is full. The obvious question that follows is what happens when the stack has a lot of data to send? This can occur as the result of a TCP connection with a large congestion window or, even worse, an application sending UDP packets as fast as it can. The answer is that for a QDisc with a single queue, the same problem outlined in Figure 4 for the driver queue occurs. That is, the high-bandwidth or high-packet rate flow can consume all of the space in the queue causing packet loss and adding significant latency to other flows. Because Linux defaults to the pfifo_fast QDisc, which effectively has a single queue (most traffic is marked with TOS=0), this phenomenon is not uncommon.

As of Linux 3.6.0, the Linux kernel has a feature called TCP Small Queues that aims to solve this problem for TCP. TCP Small Queues adds a per-TCP-flow limit on the number of bytes that can be queued in the QDisc and driver queue at any one time. This has the interesting side effect of causing the kernel to push back on the application earlier, which allows the application to prioritize writes to the socket more effectively. At the time of this writing, it is still possible for single flows from other transport protocols to flood the QDisc layer.

Another partial solution to the transport layer flood problem, which is transport-layer-agnostic, is to use a QDisc that has many queues, ideally one per network flow. Both the Stochastic Fairness Queueing (SFQ, **http://crpppc19.epfl.ch/cgi-bin/man/man2html?8+tc-sfq**) and Fair Queueing with Controlled Delay (fq_codel, **http://linuxmanpages.net/manpages/fedora18/man8/tc-fq_codel.8.html**) QDiscs fit this problem nicely, as they effectively have a queue-per-network flow.

How to Manipulate the Queue Sizes in Linux

Driver Queue:

The ethtool command (**http://linuxmanpages.net/manpages/fedora12/man8/ethtool.8.html**) is used to control the driver queue size for Ethernet devices. ethtool also provides low-level interface statistics as well as the ability to enable and disable IP stack and driver features.

The `-g` flag to `ethtool` displays the driver queue (ring) parameters:

```
# ethtool -g eth0
Ring parameters for eth0:
Pre-set maximums:
RX:        16384
RX Mini:     0
RX Jumbo:     0
TX:        16384
Current hardware settings:
RX:        512
RX Mini:     0
RX Jumbo:     0
TX:     256
```

You can see from the above

output that the driver for this NIC defaults to 256 descriptors in the transmission queue. Early in the Bufferbloat investigation, it often was recommended to reduce the size of the driver queue in order to reduce latency. With the introduction of BQL (assuming your NIC driver supports it), there no longer is any reason to modify the driver queue size (see below for how to configure BQL).

ethtool also allows you to view and manage optimization features, such as TSO, GSO, UFO and GRO, via the `-k` and `-K` flags. The `-k` flag displays the current offload settings and `-K` modifies them.

As discussed above, some optimization features greatly increase the number of bytes that can be queued in the driver queue. You should disable these optimizations if you want to optimize for latency over throughput. It's doubtful you will notice any CPU impact or throughput decrease when disabling these features unless the system is handling very high data rates.

Byte Queue Limits (BQL):

The BQL algorithm is self-tuning, so you probably don't need to modify its configuration. BQL state and configuration can be found in a /sys directory based on the location and name of the NIC. For example: /sys/devices/pci0000:00/0000:00:14.0/net/eth0/queues/tx-0/byte_queue_limits.

To place a hard upper limit on the number of bytes that can be queued, write the new value to the limit_max file:

```
echo "3000" > limit_max
```

What Is txqueuelen?

Often in early Bufferbloat discussions, the idea of statically reducing the NIC transmission queue was mentioned. The txqueuelen field in the ifconfig command's output or the qlen field in the ip command's output show the current size of the transmission queue:

```
$ ifconfig eth0
eth0    Link encap:Ethernet  HWaddr 00:18:F3:51:44:10
       inet addr:69.41.199.58  Bcast:69.41.199.63  Mask:255.255.255.248
       inet6 addr: fe80::218:f3ff:fe51:4410/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:435033 errors:0 dropped:0 overruns:0 frame:0
       TX packets:429919 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:65651219 (62.6 MiB)  TX bytes:132143593 (126.0 MiB)
       Interrupt:23

$ ip link
1: lo:  mtu 16436 qdisc noqueue state UNKNOWN
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0:  mtu 1500 qdisc pfifo_fast state UP qlen 1000
   link/ether 00:18:f3:51:44:10 brd ff:ff:ff:ff:ff:ff
```

Unfortunately, not all of the over-sized queues that will affect your Internet performance are under your control.

The length of the transmission queue in Linux defaults to 1,000 packets, which is a large amount of buffering, especially at low bandwidths.

The interesting question is what queue does this value control? One might guess that it controls the driver queue size, but in reality, it serves as a default queue length for some of the QDiscs. Most important, it is the default queue length for the pfifo_fast QDisc, which is the default. The "limit" argument on the tc command line can be used to ignore the txqueuelen default.

The length of the transmission queue is configured with the ip or ifconfig commands:

```
ip link set txqueuelen 500 dev eth0
```

Queueing Disciplines:
As introduced earlier, the Linux kernel has a large number of queueing disciplines (QDiscs), each of which implements its own packet queues and behaviour. Describing the details of how to configure each of the QDiscs is beyond the scope of this article. For full details, see the tc man page (`man tc`). You can find details for each QDisc in `man tc qdisc-name` (for example, `man tc htb` or `man tc fq_codel`).

TCP Small Queues:
The per-socket TCP queue limit can be viewed and controlled with the following /proc file: /proc/sys/net/ipv4/tcp_limit_output_bytes.

You should not need to modify this value in any normal situation.

Oversized Queues Outside Your Control

Unfortunately, not all of the over-sized queues that will affect your Internet performance are under your control. Most commonly, the problem will lie in the device that attaches to your service provider (such as DSL or cable modem) or in the service provider's equipment itself. In the latter case, there isn't much you can do, because it is difficult to control the traffic that is sent toward you. However, in the upstream direction, you can shape the traffic to slightly below the link rate. This will stop the queue in the device from having more than a few packets.

Many residential home routers have a rate limit setting that can be used to shape below the link rate. Of course, if you use Linux on your home gateway, you can take advantage of the QDisc features to optimize further. There are many examples of tc scripts on-line to help get you started.

Summary

Queueing in packet buffers is a necessary component of any packet network, both within a device and across network elements. Properly managing the size of these buffers is critical to achieving good network latency, especially under load. Although static queue sizing can play a role in decreasing latency, the real solution is intelligent management of the amount of queued data. This is best accomplished through dynamic schemes, such as BQL and active queue management (AQM, **http://en.wikipedia.org/wiki/Active_queue_management**) techniques like Codel. This article outlines where packets are queued in the Linux network stack, how features related to queueing are configured and provides some guidance on how to achieve low latency.

Acknowledgements

Thanks to Kevin Mason, Simon Barber, Lucas Fontes and Rami Rosen for reviewing this article and providing helpful feedback.■

Dan Siemon is a longtime Linux user and former network admin who now spends most of his time doing business stuff. He welcomes your comments sent to dan@coverfire.com.

Resources

Controlling Queue Delay: **http://queue.acm.org/detail.cfm?id=2209336**

Bufferbloat: Dark Buffers in the Internet: **http://cacm.acm.org/magazines/2012/1/144810-bufferbloat/fulltext**

Bufferbloat Project: **http://www.bufferbloat.net**

Linux Advanced Routing and Traffic Control How-To (LARTC): **http://www.lartc.org/howto**

Cutting Edge Linux Platforms with Intel® Xeon® Phi™ or NVIDIA® Tesla® K20

WhisperStation™ – Quiet HPC at Your Desk

- 1 to 4 NVIDIA Tesla K20 or Quadro GPUs
- NVIDIA Quadro® GPUs for visualization + NVIDIA Tesla GPUs for lightning-fast computation
- Up to 3 Intel® Xeon® Phi™ Coprocessors
- Up to 32 x86 Cores and 512GB Memory
- High clock speed Intel® CPUs; SSDs and/or RAID for fast I/O; up to 8 drives; PCI-E Generation 3.0
- 80 PLUS™ certified power supplies, ultra-quiet fans, sound proofing materials and high-efficiency components

Increase performance with CUDA C/C++, PGI CUDA FORTRAN, or these GPU-enabled applications:

Design	Simulation		BioTech
3ds Max	ANSYS CFD (Fluent)	MATLAB	AMBER
Adobe CS6	ANSYS Mechanical	ACUSIM AcuSolve	GROMACS
SolidWorks	Autodesk Moldflow	Tech-X GPULib	NAMD
Bunkspeed Shot	Abaqus	Seismic City RTM	VMD
	SIMULIA®	VoxxelGeo	TeraChem

Scalable, Fully Integrated Clusters

- Designed with Dense 1U/2U Servers and PCI-E Generation 3.0
- 8-core Xeon E5-2600 Series CPUs, DDR3-1600 Memory
- NVIDIA Tesla K10, K20 or M2090 GPUs
- Up to 4 Intel Xeon Phi Coprocessors per Node
- Mellanox ConnectX-3 FDR InfiniBand

Benchmark Remotely on 8 Xeon E5-2600s and 4 Tesla K20s.

Microway Puts YOU on the Cutting Edge

Design your next custom configuration with techs who speak HPC. Rely on our integration expertise for complete and thorough testing of your workstations, turnkey clusters and servers. Whether you require Linux or Windows, CUDA® or OpenCL, Tesla K20 or Intel Xeon Phi we've been resolving the complicated issues – so you don't have to – since 1982.

Get the Best Performance, Call Us First!

508-746-7341 or microway.com

Sign up for technical newsletters at microway.com/newsletter

Read our blog at microway.com/hpc-tech-tips

NVIDIA. TESLA

GSA Schedule Contract Number: GS-35F-0431N

intel inside Xeon Phi

Microway® *Technology you can count on since 1982*

LINUX ADVANCED ROUTING Tutorial

The Internet is a scarce resource on the remote Pacific island where I live. We can get slow, expensive connectivity, or fast bloody-expensive connectivity. Fast and cheap doesn't exist over here.

MICHAL LUDVIG

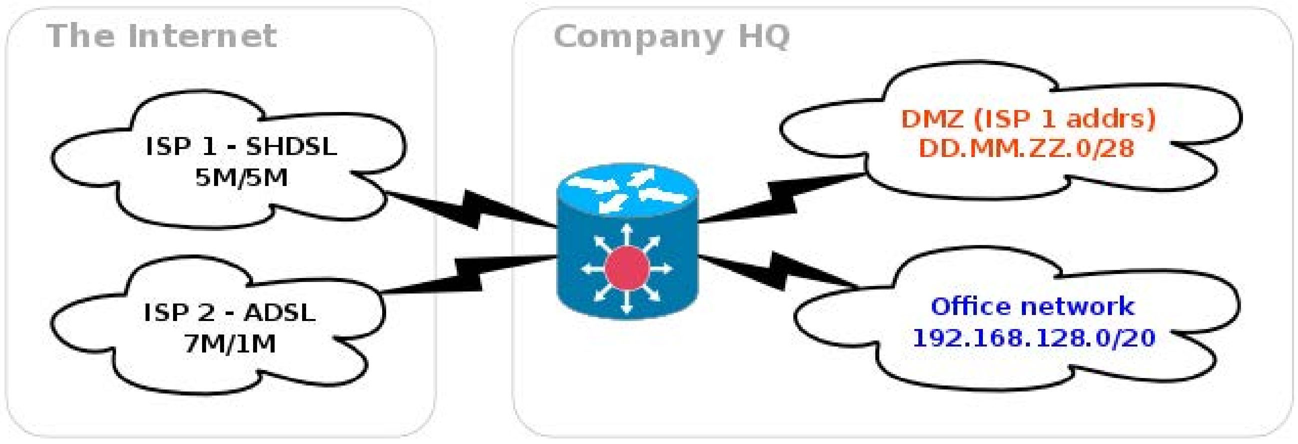


Figure 1. Network Overview

For years, we used to have a plain-old ADSL in the office—fast download speeds, slow upload, high latency—all that at the cost of $1/GB. We have had so many problems with performance and reliability that after a few years of struggling, we decided to get a second upstream link—SHDSL 5M/5M symmetric link—low latency, consistent speed during the day. It's simply awesome.

However, SHDSL is expensive! Although national traffic is free of charge, international traffic is $5/GB. Compare that with ADSL at $1/GB for both national and international (Table 1).

There are clear pros and cons to both. That made me think (and I don't do that often), what if we kept ADSL for international traffic that's not that critical for us, and use SHDSL primarily for national traffic—fast and cheap?

We also got a /28 subnet with the SHDSL to use for a DMZ, and obviously, we want all the traffic

Table 1. ADSL vs. SHDSL

	ADSL	SHDSL
Speed	7M down/1M up	5M down/5M up
Price national	$1/GB	Free
Price international	$1/GB	$5/GB
Public IP addresses	1	1 + /28

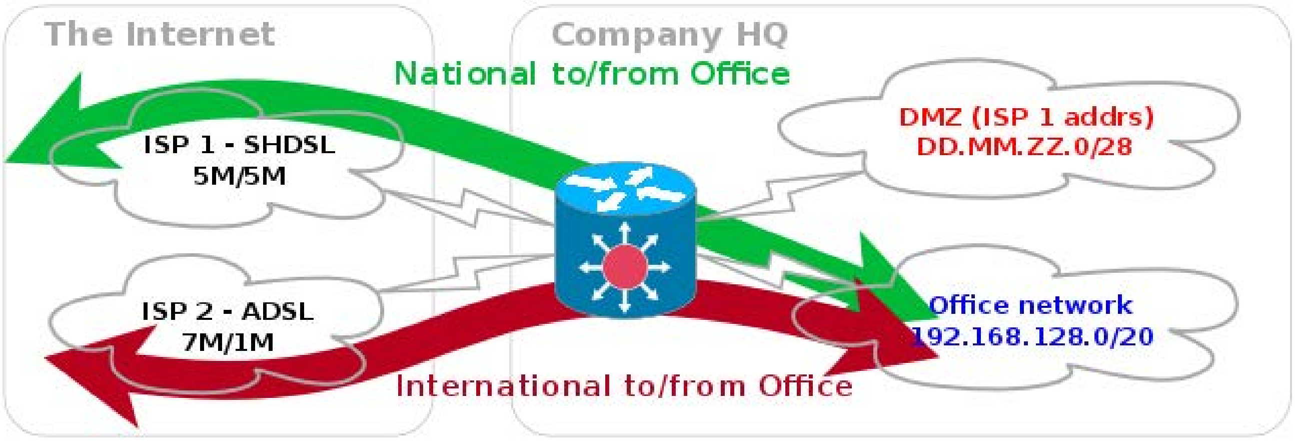


Figure 2. Split Traffic between International and National

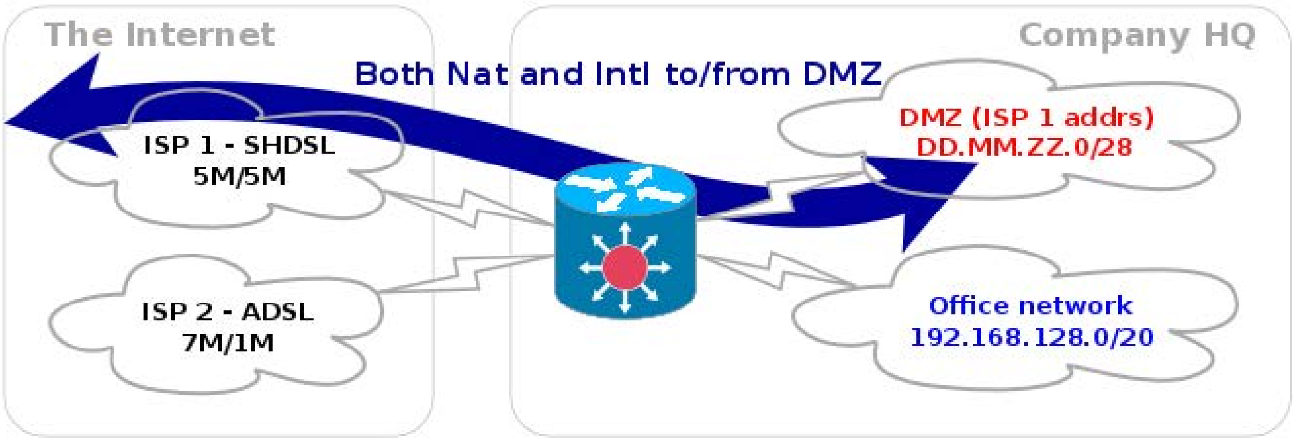


Figure 3. Traffic to/from the DMZ

to and from the DMZ to go via the SHDSL link, regardless of whether it's national or international.

Routing 101

For my plan to work, the company core router needs some way to tell whether a packet from a workstation in the office to an IP.AD.DR.ES in the wild open Internet is international or national and send it accordingly through ADSL or SHDSL. Routers use their routing tables for deciding the fates and paths of the packets they forward. A simple routing table for an office workstation with an address of 192.168.130.100 may look like this:

```
[workstation] ~ # ip route show
192.168.130.0/24 dev eth0 proto kernel  scope link
 ➥src 192.168.130.100
default via 192.168.130.1 dev eth0
```

This says that all traffic to all other addresses in the 192.168.130.0/24 range is to be sent straight to the local network segment attached to interface eth0. All other traffic follows the "default" route and is handed over to the router with IP 192.168.130.1 to handle it. Let's assume we're sending a packet to 8.8.4.4, Google's public DNS server. For starters, we're trying to "ping 8.8.4.4" and are observing the traffic on the workstation's eth0 interface with tcpdump:

```
[workstation] ~ # tcpdump -i eth0 -n -s0 -e
listening on eth0, link-type EN10MB (Ethernet),
 ➥capture size 65535 bytes
[1] 17:53:59.615650 00:16:17:ec:5c:6c > ff:ff:ff:ff:ff:ff,
 ➥ethertype ARP (0x0806), length 42:
        Request who-has 192.168.130.1 tell 192.168.130.100,
        ➥length 28
[2] 17:53:59.615775 00:14:c2:5b:4f:2c > 00:16:17:ec:5c:6c,
 ➥ethertype ARP (0x0806), length 60:
        Reply 192.168.130.1 is-at 00:14:c2:5b:4f:2c, length 46
[3] 17:53:59.615796 00:16:17:ec:5c:6c > 00:14:c2:5b:4f:2c,
 ➥ethertype IPv4 (0x0800), length 98:
        192.168.130.100 > 8.8.4.4: ICMP echo request,
        ➥id 3082, seq 1, length 64
```

My workstation consulted the routing table for the destination IP 8.8.4.4 and realized it should send the packet to the default router 192.168.130.1. For that, it needs the router's low-level Ethernet address (also known as a MAC address), and the first packet in the above tcpdump output, marked [1], is doing exactly that—asking for the MAC address of IP 192.168.130.1 by "broadcasting" the request to all nodes on the network segment. The second packet, marked [2], is a reply—IP 192.168.130.1 is at MAC address 00:14:c2:5b:4f:2c. Finally, the PING packet can be dispatched, with the destination IP 8.8.4.4 to the router's MAC address 00:14:c2:5b:4f:2c (see the line marked [3]).

All good, so now we can assume our router got the packet and will forward it further toward the destination. Let's see what happens on the router.

We've got both the ADSL and the SHDSL links configured, but all traffic is, by default, sent through ADSL. The ADSL modem is at 192.168.128.254. For now, the SHDSL link 203.0.113.36/30 sits idle. Here is the routing table:

```
[router] ~ # ip route show
[1] 203.0.113.36/30 dev vlan-shdsl  proto kernel
 ➥scope link  src 203.0.113.38
```

```
[2] 192.168.128.0/24 dev vlan-adsl  proto kernel
 ➥scope link  src 192.168.128.1
[3] 192.168.130.0/24 dev vlan-office  proto kernel
 ➥scope link  src 192.168.130.1
[4] default via 192.168.128.254 dev vlan-adsl
```

The first line [1] is the SHDSL link—our router's IP on that link is 203.0.113.38. The second line is the link to the ADSL modem; the third line [3] is the network segment with my workstation, and finally, the fourth line [4] is the default route. All packets that don't match any of the local subnets on lines 1, 2 or 3 are sent down to the ADSL modem at 192.168.128.254 that then will forward them to the ISP 2. That's also the fate of our packet to 8.8.4.4. Let's quickly verify what is going to happen to it by calling `ip route get`:

```
[router] ~ # ip route get 8.8.4.4 from 192.168.130.100
 ➥iif vlan-office
8.8.4.4 from 192.168.130.100 via 192.168.128.254 dev vlan-adsl
```

As you can see, it will be sent "via 192.168.128.254", which is the ADSL modem. A simple way to check the full network path from my workstation to any given destination address is the traceroute command. It shows all the routers ("hops") along the way to the destination:

```
[workstation] ~ $ /usr/sbin/traceroute 8.8.4.4
traceroute to 8.8.4.4 (8.8.4.4), 30 hops max,
 ➥40 byte packets using UDP
 1  gw-vlan-office.e-it.co.nz (192.168.130.1)
    ➥0.156 ms  0.126 ms  0.124 ms
 2  gw-vlan-adsl.e-it.co.nz (192.168.128.254)
    ➥0.853 ms  0.831 ms  0.830 ms
 3  core-adsl.isp2 (218.101.x.y)
    ➥11.765 ms  19.173 ms  19.066 ms
 4  core-xyz.isp2 (203.98.x.y)
    ➥16.052 ms  15.515 ms  17.153 ms
[... some more hops ...]
13  64.233.x.y (64.233.x.y)  193.826 ms  194.230 ms  194.412 ms
14  * * *
15  google-public-dns-b.google.com (8.8.4.4)
    ➥196.086 ms  195.909 ms  195.816 ms
```

As you can see, the first hop is our router 192.168.130.1. The second hop is the ADSL modem 192.168.128.254. The third hop is one of the ISP2's core routers, and so on and on, passing 11 more routers before the packet finally reaches the destination 8.8.4.4, aka google-public-dns-b.google.com.

National vs. International Traffic

Google's public DNS server B is clearly an off-shore international destination. However, one of the root DNS servers, namely f.root-servers.net with IP 192.5.5.241, is present here in New Zealand. At the moment, the traceroute to f.root-servers.net still shows the ADSL path:

```
[workstation] ~ $ /usr/sbin/traceroute f.root-servers.net
traceroute to f.root-servers.net (192.5.5.241), 30 hops max,
 ➥40 byte packets using UDP
 1  gw-vlan-office.e-it.co.nz (192.168.130.1)
    ➥0.175 ms  0.126 ms  0.125 ms
 2  gw-vlan-adsl.e-it.co.nz (192.168.128.254)
    ➥0.861 ms  0.840 ms  0.825 ms
 3  core-adsl.isp2 (218.101.x.y)  22.456 ms  22.298 ms  23.501 ms
 4  isp2.ape.net.nz (192.203.154.x)  21.035 ms  20.928 ms  21.268 ms
 5  isc2.ape.net.nz (192.203.154.y)  20.689 ms  21.724 ms  24.187 ms
 6  f.root-servers.net (192.5.5.241)  26.680 ms  26.059 ms  25.427 ms
```

This is clearly national traffic, and it should go out via the SHDSL link as per our plan. We can do that manually and set the appropriate route on our core router:

```
[router] ~ # ip route add 192.5.5.241 via 203.0.113.37
 ➥dev vlan-shdsl
```

A side note about NAT'ing: we also need to translate (or "masquerade" or NAT) the office address range 192.168.130.0/24 to our SHDSL public IP 203.0.113.38. If we didn't, the packet's source IP would be 192.168.130.100, and the replies would never find their way back to my workstation, as this address range is not routable in the public Internet. Discussing firewalls and NATs is out of scope of this article, but to get you going, here's the simplest magic command:

```
[router] ~ # iptables -t nat -A POSTROUTING -s 192.168.128.0/20
 ➥-o vlan-shdsl -j DNAT --to 203.0.113.38
```

All packets with a source address from the 192.168.128.0/20 range leaving through the interface vlan-shdsl will get the source rewritten to 203.0.113.38. We didn't need to set up any masquerading for the ADSL path, because there the ADSL router NATs all our outgoing traffic.

Now, let's check the new network path:

```
[workstation] ~ $ /usr/sbin/traceroute f.root-servers.net
traceroute to f.root-servers.net (192.5.5.241), 30 hops max,
 ➥40 byte packets using UDP
 1  gw-vlan-office.e-it.co.nz (192.168.130.1)
    ➥0.190 ms  0.129 ms  0.125 ms
 2  rt-shdsl.isp1 (203.0.113.37)  2.676 ms  2.599 ms  2.632 ms
 3  rt3.isp1 (121.98.ab.cd)  2.715 ms  2.680 ms  2.591 ms
 4  isc2.ape.net.nz (192.203.154.z)  2.919 ms  3.033 ms  3.088 ms
 5  f.root-servers.net (192.5.5.241)  3.007 ms  2.670 ms  2.864 ms
```

That's lot better. The first hop remains the same, that's my workstation's router, but the second hop is no longer the ADSL modem. Instead, it's the SHDSL ISP1's core router. It also clearly shows the latency improvement—from 26ms over ADSL to 3ms over SHDSL.

Let's remove the manual route again to avoid any confusion down the road

So all we need now is a list of all national IP addresses, put them in the routing table, and we're done. But how? Manually?

and query the routing table:

```
[router] ~ # ip route get 192.5.5.241 from
➥192.168.130.100 iif vlan-office
192.5.5.241 from 192.168.130.100 via 203.0.113.37
➥dev vlan-shdsl        # SHDSL

[router] ~ # ip route delete 192.5.5.241 via 203.0.113.37
➥dev vlan-shdsl

[router] ~ # ip route get 192.5.5.241 from
➥192.168.130.100 iif vlan-office
192.5.5.241 from 192.168.130.100 via 192.168.128.254
➥dev vlan-adsl      # ADSL
```

So all we need now is a list of all national IP addresses, put them in the routing table, and we're done. But how? Manually? Even in a small country like New Zealand there are hundreds of local IP addresses and prefixes, and the list is dynamic and changes daily. There is no way such a list could be managed manually. We need a better tool.

Feeding the Bird

BGP is *the* routing protocol of today's Internet. Each and every ISP on the planet tells all other ISPs on the planet what IP addresses it has in its network and what other IP addresses can be reached through that ISP's network. The protocol to exchange this kind of information is called BGP (Border Gateway Protocol).

I'm not going to dive in to the details of BGP. All we need to know is that our ISP1 can use BGP to send us the list of all the national prefixes. Instead of entering them manually, we will listen to the "BGP feed" and update our routing table from it. This part needs coordination with the Internet provider, so I asked our ISP1 to "please assign us a private ASN and send us a BGP feed with national prefixes". BGP usually runs between Autonomous Systems identified by ASN (Autonomous System Number). Any big enough organization can apply and pay for its own official ASN, but for our purpose, it's enough to use an ASN from a private range assigned by the ISP. Table 2 shows the information I received from them.

Table 2. Info from ISP1

	ISP	COMPANY
ASN	177XY	6452X
Router IP	203.0.113.37	203.0.113.38
Prefixes advertised	all national ones	none

Now we have the routers' IPs on both sides of the link, both ASNs and confirmation of what prefixes they will send us ("all national ones"). That's all the info we need.

A number of BGP dæmons are available for Linux. I chose Bird (at the time, the most recent version was 1.3.4) and installed it from an RPM.

Bird's configuration is very simple. The config file is at /etc/bird/bird.conf and may look like this:

```
log syslog all;
protocol kernel {
        import none;
        export all;
}
protocol device {
        scan time 10;
}
protocol bgp {
        description "ISP1 National Routes";
        local as 64526;
        neighbor 203.0.113.37 as 17746;
        source address 203.0.113.38;
        import all;     # Accept all prefixes from our neighbor
        export none;    # Don't send the neighbor any prefixes
}
```

Essentially, the Bird process talks to the BGP neighbor and imports all the advertised prefixes into its internal routes list. It also talks with the kernel and "exports to the kernel" all the routes it knows—that is, all the ones learned over BGP. That way, it feeds the kernel routing table from the BGP.

With the BGP routes imported, the kernel routing table now has more than 4,000 records:

```
[router] ~ # ip route show
   [1] 203.0.113.36/30 dev vlan-shdsl  proto kernel
   ➥scope link  src 203.0.113.38
   [2] 192.168.128.0/24 dev vlan-adsl  proto kernel
   ➥scope link  src 192.168.128.1
   [3] 192.168.130.0/24 dev vlan-office  proto kernel
   ➥scope link  src 192.168.130.1
   [4] default via 192.168.128.254 dev vlan-adsl
   [5] 14.1.32.0/19 via 203.0.113.37 dev vlan-shdsl  proto bird
   [6] 27.252.0.0/16 via 203.0.113.37 dev vlan-shdsl  proto bird
   [7] 58.28.0.0/16 via 203.0.113.37 dev vlan-shdsl  proto bird
 ....
[4509] 203.97.0.0/17 via 203.0.113.37 dev vlan-shdsl  proto bird
[4510] 222.153.128.0/18 via 203.0.113.37 dev vlan-shdsl  proto bird
[4511] 222.155.96.0/19 via 203.0.113.37 dev vlan-shdsl  proto bird
```

Lines [1] to [4] are the same that you've seen before—all of the directly connected subnets and the default route. Lines [5] to [4511] are the routes received over BGP.

Now, let's query the newly populated routing for a path to our favourite f.root-servers.net:

```
[router] ~ # ip route get 192.5.5.241 from
➥192.168.130.100 iif vlan-office
192.5.5.241 from 192.168.130.100 via
➥203.0.113.37 dev vlan-shdsl        # SHDSL
```

Excellent, that's what we wanted! A couple traceroute runs from the workstation to both national and international destinations verifies that the new path is used as expected.

The DMZ Challenge

One of the requirements for our second Internet link was the possibility to get a block of public IP addresses for our DMZ (DeMilitarized Zone) where we plan to host some externally accessible services. From the ISP, we got block 203.0.113.208/28, and we want all the traffic to and from the DMZ to use the SHDSL link, regardless of whether it's national or international traffic. Routing the inbound traffic is the ISP's responsibility, but outbound is our job.

Let's see what happens with our current setup. On a test DMZ box, 203.0.113.222, the routing table looks like this:

```
[dmz-box] ~ # ip route show
203.0.113.208/28 dev eth0  proto kernel
➥scope link  src 203.0.113.222
default via 203.0.113.209 dev eth0
```

All the traffic goes to the router's IP 203.0.113.209, and the router then decides what to do. Let's traceroute to the Google's public DNS server again and see how it goes:

```
[dmz-box] ~ # traceroute 8.8.4.4
traceroute to 8.8.4.4 (8.8.4.4), 30 hops max, 60 byte packets
 1  gw-dmz.e-it.co.nz (203.0.113.209)  0.175 ms  0.110 ms  0.110 ms
 2  gw-vlan-adsl.e-it.co.nz (192.168.128.254)
    ➥8.707 ms  9.080 ms  9.522 ms
 3  core-adsl.isp2 (218.101.x.y)  11.899 ms  13.555 ms  15.585 ms
 4  core-xyz.isp2 (203.98.x.y)  15.114 ms  16.332 ms  17.142 ms
[... some more hops ...]
13  64.233.x.y (64.233.x.y) 195.061 ms  198.098 ms  197.589 ms
14  * * *
15  google-public-dns-b.google.com (8.8.4.4)
    ➥203.621 ms  204.588 ms  205.792 ms
```

That's not quite what we wanted. The traffic flows through the ADSL link instead of the SHDSL link. But, it's understandable. Our router selects the network paths

Fortunately, the Linux kernel can have up to 255 independent routing tables and up to 32,768 different rules specifying which routing table to look up for each packet.

based only on the destination IP, disregarding the source IP. As soon as it sees a packet for 8.8.4.4, it sends it down the ADSL path, regardless of whether it comes from my workstation at 192.168.130.100 or from the DMZ box at 203.0.113.222.

Fortunately, the Linux kernel can have up to 255 independent routing tables and up to 32,768 different rules specifying which routing table to look up for each packet. The standard ruleset on a recent Linux machine looks like this:

```
[router] ~ # ip rule show
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default
```

There are no constraints for the rules. All three rules may be consulted for packets "from all" addresses until a matching route is found. According to the rules, the "local" table is looked up first. The "local" table is maintained by the kernel and contains rules for broadcast addresses and similar special destinations. You can list its contents, but unless you know exactly what you're doing, you better not change it:

```
[router] ~ # ip route show table local
broadcast 192.168.130.0 dev vlan-office  proto kernel
➥scope link  src 192.168.130.1
local 192.168.130.1 dev vlan-office  proto kernel
➥scope host  src 192.168.130.1
local 203.0.113.38 dev vlan-shdsl  proto kernel
➥scope host  src 203.0.113.38
broadcast 203.0.113.39 dev vlan-shdsl  proto kernel
➥scope link  src 203.0.113.38
broadcast 127.0.0.0 dev lo  proto kernel
➥scope link  src 127.0.0.1
local 127.0.0.1 dev lo  proto kernel  scope host  src 127.0.0.1
[... a lot more routes ...]
```

If no match is found in the

"local" table, the kernel moves on the the next rule that says "lookup main". The "main" table is the standard table where all the "normal" routes end up—all the routes for directly connected subnets, manually added static routes and also the routes from BGP. The command `ip route show` works by default with the "main" table and is equivalent to `ip route show table main`. If still no match is found, the last rule says "lookup default". The "default" table is seldom used and is usually empty.

In fact, the kernel tables have numerical IDs, and the words "main", "local" and "default" are just convenient verbose names defined in /etc/iproute2/rt_tables:

```
[router] ~ # cat /etc/iproute2/rt_tables
#
# reserved values
#
255     local
254     main
253     default
0       unspec
```

So, to recap our current situation, we've got the "main" table with a list of all national routes for SHDSL and one default route for ADSL for all international traffic. And, we've got a default rule that says "for all source addresses consult the main table".

Solving the DMZ Challenge

What we need now is to set up a special routing table for all DMZ traffic and create a rule that will plug it in to the routing decision process.

The first step is to create and populate the new table and name it "dmz". We can choose any non-reserved table ID and add one line into the rt_table file to map the ID to our desired name:

```
[router] ~ # echo "121 dmz" >> /etc/iproute2/rt_tables
[router] ~ # ip route show table dmz
[router] ~ #
```

The dmz table is empty for now. Let's populate it. Apparently, we need just the default route—everything should be sent to ISP1's gateway:

```
[router] ~ # ip route add default via 203.0.113.37
➥dev vlan-shdsl table dmz
[router] ~ # ip route show table dmz
default via 203.0.113.37 dev vlan-shdsl
```

All we need to do now is activate the table by adding a lookup rule to the routing decision process. The lookup rules can be based on many different parameters—the source and/or destination address or address range, the incoming interface, the TOS (Type Of Service) or an arbitrary "fwmark" value that is used in conjunction with iptables rules. For our purpose, the source IP and incoming network interface are the "selectors" that we need:

```
[router] ~ # ip rule add pri 1000 from 203.0.113.208/28
 ➥iif vlan-dmz lookup dmz
[router] ~ # ip rule show
0:      from all lookup local
1000:   from 203.0.113.208/28 iif vlan-dmz lookup dmz
32766:  from all lookup main
32767:  from all lookup default
```

All the packets coming from interface vlan-dmz with source address 203.0.113.208/28 will consult the dmz table. And the dmz table says to send them all to the SHDSL link by default. Let's see if it works:

```
[dmz-box] ~ # traceroute 8.8.4.4
traceroute to 8.8.4.4 (8.8.4.4), 30 hops max, 60 byte packets
 1  gw-dmz.e-it.co.nz (203.0.113.209)
    ➥0.200 ms  0.124 ms  0.110 ms
 2  rt-shdsl.isp1 (203.0.113.37)  2.676 ms  2.599 ms  2.632 ms
 3  rt3.isp1 (121.98.ab.cd)  2.715 ms  2.680 ms  2.591 ms
[... some more hops ...]
10  216.239.xx.yy (216.239.xx.yy)
    ➥172.955 ms  172.398 ms  172.339 ms
11  * * *
12  google-public-dns-b.google.com (8.8.4.4)
    ➥171.703 ms  171.621 ms  170.554 ms
```

That looks good. The international traffic from the box in DMZ goes over the SHDSL link, while at the same time, we can re-check that from the workstation in the internal

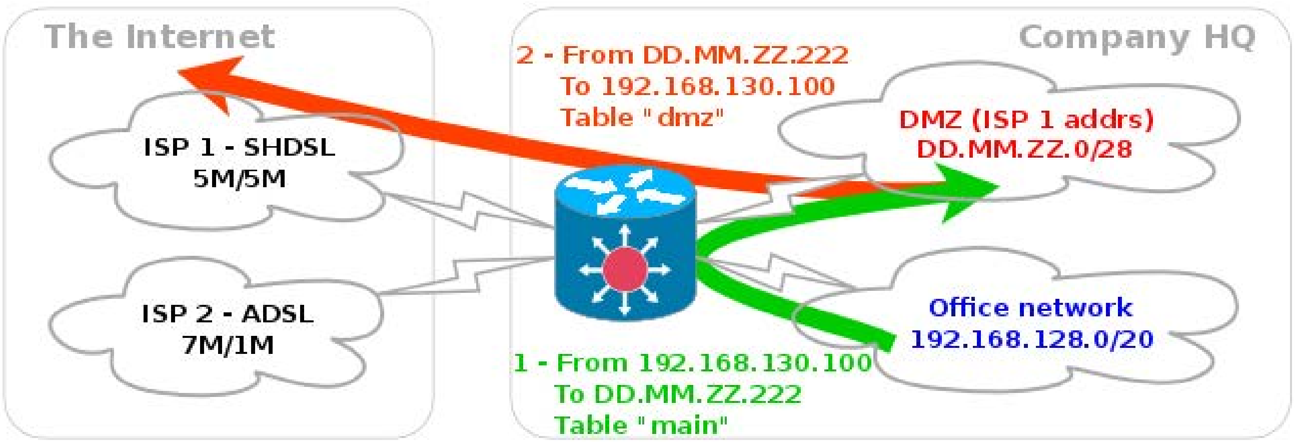


Figure 4. Broken Traffic between DMZ and the Office

LAN, it still goes over ADSL:

```
[workstation] ~ # traceroute 8.8.4.4
traceroute to 8.8.4.4 (8.8.4.4), 30 hops max,
 ➥40 byte packets using UDP
 1  gw-vlan-office.e-it.co.nz (192.168.130.1)
    ➥0.170 ms  0.128 ms  0.125 ms
 2  gw-vlan-adsl.e-it.co.nz (192.168.128.254)
    ➥0.861 ms  0.846 ms  0.825 ms
 3  core-adsl.isp2 (218.101.x.y)  22.718 ms  22.109 ms  21.931 ms
 4  core-xyz.isp2 (203.98.x.y)  20.417 ms  20.222 ms  19.769 ms
[... some more hops ...]
13  64.233.x.y (64.233.x.y)  198.456 ms  197.824 ms  197.189 ms
14  * * *
15  google-public-dns-b.google.com (8.8.4.4)
    ➥194.318 ms  194.153 ms  194.159 ms
```

Magic, isn't it! It all looks good, and we're done, right? Well, not quite.

More Work for DMZ Routing

What happens when pinging the DMZ box from my workstation?

First, the ping packet goes from 192.168.130.100 to the router. The router consults the "main" routing table, sees that the destination 203.0.113.222 is on a directly connected vlan-dmz and sends the ping request there. The DMZ box replies with a packet from 203.0.113.222 to 192.168.130.100—that's my workstation on vlan-office—and sends it to the router. The router sees that it arrived from the DMZ address range and consults the "dmz" table! The only record in there is the default route, so it takes it and dispatches the packet with destination IP 192.168.130.100 to the ISP where it eventually gets discarded. Oops!

There are a number of solutions. Either we can copy all the non-BGP routes from the "main" table to the "dmz" table—that's indeed possible but not very elegant. A better option is to add a new

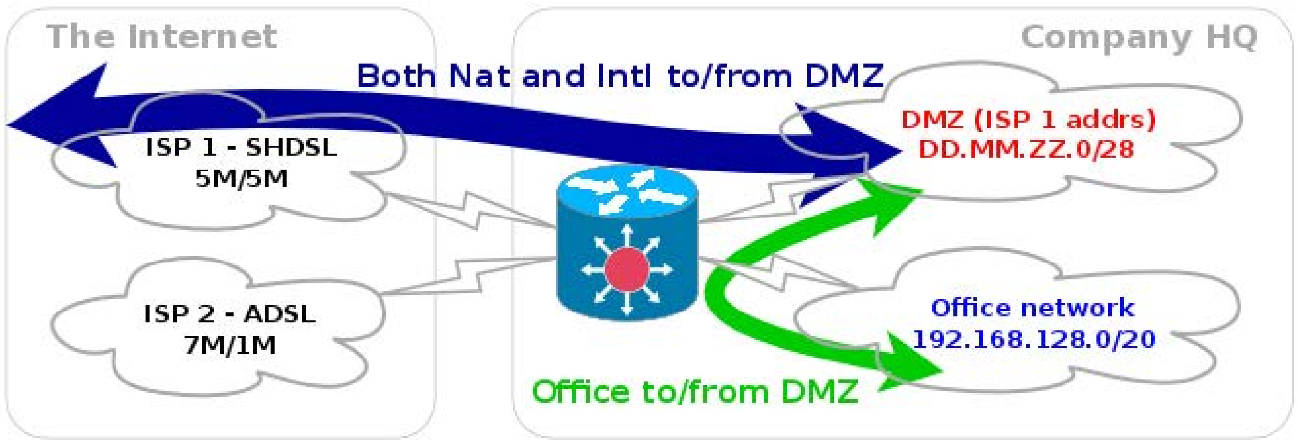


Figure 5. DMZ traffic works as expected.

routing entry to the "dmz" table that, when matched, will make the kernel move on to the next rule on the list:

```
[router] ~ # ip rule show
0:      from all lookup local
1000:   from 203.0.113.208/28 iif vlan-dmz lookup dmz
32766:  from all lookup main
32767:  from all lookup default

[router] ~ # ip route add throw 192.168.128.0/20 table dmz

[router] ~ # ip route show table dmz
throw 192.168.128.0/20
default via 203.0.113.37 dev vlan-shdsl
```

What happens now with the ping reply packet is this: the kernel sees a matching rule 1000 and looks up the "dmz" table. There it finds that the best matching route for a destination of 192.168.130.100 is "throw". That tells it to choose the next available rule—in our case, "32766: from all lookup main" and looks up the "main" table. Here it finds a path to the 192.168.130.0/24 subnet and sends the ping reply to vlan-office. Problem fixed.

Another way, equally as good if not even better, is to go straight to the "main" table for all packets that arrive from the DMZ and are destined for our company's internal address range. Let's add a rule for it:

```
[router] ~ # ip rule add pri 999 from 203.0.113.208/28
 ➥iif vlan-dmz to 192.168.128.0/20 table main

[router] ~ # ip rule show
0:      from all lookup local
999:    from 203.0.113.208/28 to 192.168.128.0/20
        ➥iif vlan-dmz lookup main
1000:   from 203.0.113.208/28 iif vlan-dmz lookup dmz
32766:  from all lookup main
32767:  from all lookup default
```

With this rule in place, the router will see that the ping reply packet matches the rule "from 203.0.113.208/28 to 192.168.128.0/20" and will look up the "main" table instead of the "dmz" table right away. Personally, I like this solution the most as it explicitly says "lookup main" instead of a more vague "go to the next rule", whatever it may be, that the former solution does.

Making the Changes Permanent

Many of the changes we have done so far are only in the router's memory and will disappear after a reboot. Let's make them permanent. Our core router is CentOS 6.2, and the following steps should work on any recent Red Hat Enterprise Linux or Fedora-based system.

A permanent name already has been given to the "dmz" table:

```
[router] ~ # cat /etc/iproute2/rt_tables
# reserved values
255     local
254     main
253     default
0       unspec
# local
121     dmz
```

To add the new ip rules, edit the file /etc/sysconfig/network-scripts/rule-<interface>. As soon as the given interface is configured, the rules will be loaded. The logical place is to load our rules once the DMZ interface vlan-dmz is configured:

```
[router] ~ # cat /etc/sysconfig/network-scripts/rule-vlan-dmz
# ip rule add ...
pri 999 from 203.0.113.208/28 to 192.168.128.0/20
 ➥iif vlan-dmz lookup main
pri 1000 from 203.0.113.208/28 iif vlan-dmz lookup dmz
```

Finally, populate the "dmz" table on reboot. This is done in the route-<interface> file in the same directory:

```
[router] ~ # cat /etc/sysconfig/network-scripts/route-vlan-dmz
# ip route add ...
default via 203.0.113.37 dev vlan-shdsl table dmz
# throw 192.168.128.0/20 table dmz ## unless the rule 999 is in place
```

That's it. The routing now works as designed and survives a reboot.

Networking 201?

The topics I explained in this article—multiple routing tables, rules for selecting them and a hint of BGP—should give you enough knowledge to build moderately complex Linux-based routers. In this trade, you will come across a lot more concepts—from OSPF through firewalls and NATs to VPNs and high-availability networks. None of these topics are within the scope of this article; however, in the end, the core thing that makes all this networking fun possible is routing. If you learn the few simple concepts of how IP routing works, you will know half of what networking is all about. Happy experimenting!■

Michal Ludvig works for Enterprise IT Ltd in New Zealand as a senior Linux engineer. He's got root access to some of the largest New Zealand corporations, but since he tends to forget his passwords, he is, in general, pretty harmless. Michal welcomes your comments or questions at mludvig@logix.net.nz.

DOC SEARLS

Setting TV Free

Version 3.x is as far as TV gets. There won't be a 4.0.

My 2006-vintage Sony Bravia flat-screen "Full HD" TV has Linux inside. I can tell because it comes with a two-page printout of the GPL, included almost as a warning. "Watch out", it seems to say. "This TV comes infected with freedom." Not that it's worth hacking: you can make breakfast in the time that passes between a click on the remote and a change on the screen. (I'm barely exaggerating here. Switching between the TV's eight HDMI inputs is amazingly slow.) But being a Linux device says volumes about what has happened to TV already, because the freedom it contains at the device level also ranges outward from the operating system to the network on which that operating system was born and grew up. That network was, and remains, the Internet.

TV wasn't designed to run on the Internet, and the TV industry continues to fight the drift of video production and consumption in a direction it calls "over the top", or OTT. The bottom is TV's distribution system, better known as cable. That fight is a movie we've seen before, and we know how it ends.

What we call cable today began as CATV, for Community Antenna Television, in the 1960s. CATV was invented to meet the needs of TV viewers who couldn't get signals. Hollywood didn't have a problem with that, but it was freaked out by what it saw as a new distribution channel for entertainment. At the time, Hollywood had just two distribution channels for what we now call "content". One was movie theaters and the other was commercial TV, dominated by networks and their programs. Back then, there were three major TV networks: ABC, CBS and NBC. The Fox network came along later. PBS was noncommercial and mattered less, but for distribution, it was well aligned with its commercial companions on what was then still TV's "dial". Hollywood correctly saw cable as a threat to that system, because it turned consumers of television into customers for distribution of anything, since there was hardly any limit to what could be distributed on cable's system, especially on channels other than the ones already occupied by TV stations.

So Hollywood fought CATV. It did this by, among other things, forcing employees of movie theaters to wear buttons that said "Fight Pay TV". Joining Hollywood's cause was the Television Accessory Manufacturers Institute, which misspelled itself TAME. *In Blue skies: A History of Cable Television* (Temple University Press, 2008), Patrick R. Parsons writes:

> Cable, clearly, had no friends in the TV antenna business. TAME widely distributed flyers and press releases in communities considering CATV systems. TAME newspaper advertisements warned, "Here's Why Cable TV Is Bad for Our Community", and "How to Fight Community Antenna Systems".

At the heart of broadcasting from the start has been a scarcity imperative, based on the very fact of limited RF (radio frequency) spectrum. This started with AM radio, which operated on Long Wave (LW) and Middle Wave (MW) in Europe and other parts of the world, and in MW alone in the US, where we call it AM. Here we had channels every 10KHz from 540–1600KHz (later pointlessly extended to 1700KHz). The waves in that band—the 1MHz band—are hundreds of feet long, and are radiated by whole towers. When you see a set of free-standing or insulated guyed towers, you're probably looking at an AM station's transmitter, still operating inside conceptual and legal frameworks developed in the 1920s.

As transmitters and receivers became capable of using higher frequency bands, shortwave was added. Then, as the capacity to operate at still higher frequencies developed, the new technologies called TV and FM radio appeared on what came to be called "low-band" VHF (Very High Frequency) bands: roughly 54–88MHz for TV channels 2 to 6, and 88–108MHz for FM radio. (TV audio also used FM, so channel 6 audio could be heard below the bottom of the FM dial, at 87.75MHz.) When the first TV stations filled up low-band VHF, channels 7 to 13 were added in the high-band VHF range, which runs from 174–216MHz.

Adding to the scarcity of these channels was the inability of signals to radiate on adjacent channels without interfering with each other. (Channels 4 and 5, used in New York, Chicago, Los Angeles and other major cities, were next to each other numerically, but had an open guard band between them.) So, in the 1950s, channels 14–83 were added in the UHF (Ultra High Frequency) band, which ranged from 470–862MHz.

Delivering television signals to viewers' antennae was a brute-force undertaking. The maximum permitted ERP (effective radiated power) of stations in the US was 100,000 watts on low-band VHF, 316,000 watts on high-band VHF, and 5,000,000 watts on UHF. That was for analog TV, which ended with the "digital transition" in 2008. Digital transmission continues today on high-band VHF and UHF channels, at lower powers that are still quite high. For example, the limit is still 1,000,000 watts on UHF. Higher frequencies require higher powers because their shorter waves propagate less well across terrain and through structures. This is why height matters even more than power for over-the-air TV and FM. When you see freakishly tall towers standing out in the middle of nowhere, and the tops of high buildings and mountains bristling with antennae, you're looking at TV and FM transmitters.

Cable obsoleted all these concerns for most TV viewers, especially because cable added many more channels than were possible over the air. So did satellite TV, which boasted even more channels than were possible through cable. Between cable and satellite, over-the-air (OTA) TV became almost completely obsolete, even with digital transmission. Today, OTA serves less than 10% of the viewing population. One reason is that there is far more to see on cable and satellite. Another is that digital transmission tends to fail without clear line-of-sight signal paths between receivers and transmitters. Digital TV (DTV) also isn't designed for mobile use, so having a TV receiver in your phone or pad isn't a happening thing—especially since you're already getting video over the Internet, and from infinitely more sources. And that's what Hollywood and cable are fighting today.

You need to understand Hollywood and cable as one thing. Comcast's purchase of NBC a couple years back simply clarified what was already a fact. Hollywood and cable today are also of one mind in wanting to control the distribution of video content, and of the viewing "experience", which consists, as it has since the 1920s, of "tuning" among a finite number of "channels". The Internet threatens the old channel-based TV scarcity model, and the brute-force imperatives of its distribution system, by offering *à la carte* choices beyond calculation. Against this fate, Hollywood and cable together are no less opposed than Hollywood and the antenna makers were to CATV's rise in the 1960s. The difference is that cable is also in the business of providing Internet access as a service, which puts it on both sides of a battle between the past and the future.

On the side of the future, the biggest video player is Google, through YouTube. And, since mobile video comes to you mostly over the Internet, Google is the biggest player there as well. But the Net doesn't have channels in the old-fashioned radio and TV sense. Even the term "content" fails to contain the scope of what the Net supports for communications and the sharing of data. This means Google doesn't control video on the Net. Nor does anybody.

TV over IP—the Internet Protocol—is also called IPTV. That's what you get "over the top" of cable. With IPTV, some sources look like TV channels, and some even carry old analog-era channel numbers. But they are TV only to the degree that they carry forward legacy branding and business models, such as advertising and subscription fees. Their new context, however, is the uncontained abundance opened by the Internet Protocol, and by the capability of devices running Linux and other abundance-loving operating systems. This abundance is terminal for television.

So, if we look back on TV's history, we see three eras:

1. Antenna
2. Cable
3. IP

We are in the third of those now: version 3.x. And, because TV v3 lives inside Internet v1, that will be as far as TV gets. There will be no TV 4.0.

In early May 2013, the *Wall Street Journal* carried a story titled "ESPN Eyes Subsidizing Wireless Data-Plans" (**http://online.wsj.com/article/SB10001424127887324059704578473400083982568.html**). The gist: "Under one potential scenario, the company would pay a carrier to guarantee that people viewing ESPN mobile content wouldn't have that usage counted toward their monthly data caps." I can't help but wonder what ESPN's actual agenda here might be. Such a plan would not only violate the principle of network neutrality—that the network itself should not favor one kind of data, or data producer, over another—but turn all competing content producers into instant network neutrality activists. More likely ESPN, the most powerful program source in the cable world, senses which way the wind is blowing, and it's against broadcasting's old models. So it wants to shake things up a bit, and hasten history along.

It is not a coincidence that Senator John McCain introduced the TV Consumer Freedom Act (**http://www.scribd.com/doc/140433670/TV-Consumer-Freedom-Act**) at around the same time: a move *Business Insider* said "would dismantle cable as it's currently constructed" (**http://www.businessinsider.com/john-mccain-wants-to-blow-up-the-cable-industry-as-we-currently-know-it-2013-5**). In a statement, McCain put his case this way:

> This legislation has three principal objectives: (1) encourage the wholesale and retail "unbundling" of programming by distributors and programmers; (2) establish consequences if broadcasters choose to "downgrade" their over-the-air service; and (3) eliminate the sports blackout rule for events held in publicly-financed stadiums.
>
> For over 15 years I have supported giving consumers the ability to buy cable channels individually, also known as "à la carte", to provide consumers more control over viewing options in their home and, as a result, their monthly cable bill.
>
> The video industry, principally cable companies and satellite companies and the programmers that sell channels, like NBC and Disney-ABC, continue to give consumers two options when buying TV programming: first, to purchase a package of channels whether you watch them all or not; or, second, not purchase any cable programming at all.
>
> This is unfair and wrong—especially when you consider how the regulatory deck is stacked in favor of industry and against the American consumer.

Like ESPN, McCain is watching the shifting tide of consumer sentiment. Cable's customers hate bundling, and cable itself. They get *à la carte* on the Net, and on devices native to the Net, and they want that for their TV channels and shows as well.

Once you unbundle TV and make it *à la carte*, you have nothing more than subscription video on the Net. There will still be lots of "channels" and "shows", and you will still pay for them. But the context will be IP, not TV.

This means there won't be a TV 4.0 because TV 3.0—TV over IP—will be the end of TV's line. And, after that happens, Hollywood and its buddies will discover, once again, that there is a lot more opportunity in a big new world it doesn't control than in a small old world it used to control.

In its locked-up little heart, my old flat-screen knows it carries the DNA of an IP TV. That DNA long ago decided the direction in which TV would evolve, whether Hollywood and the cable industry want it that way or not.■

Doc Searls is Senior Editor of *Linux Journal*. He is also a fellow with the Berkman Center for Internet and Society at Harvard University and the Center for Information Technology and Society at UC Santa Barbara.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

Advertiser Index

Thank you as always for supporting our advertisers by buying their products!

ADVERTISER	URL	PAGE #
1&1	http://www.1and1.com	21
Drupalize.me	http://drupalize.me	91
Emac, Inc.	http://www.emacinc.com	13
EmperorLinux	http://www.emperorlinux.com	35
iXsystems	http://www.ixsystems.com	7
Manage Engine	http://www.manageengine.com	55
Microway	http://www.microway.com	105
OSCON	http://www.oscon.com/oscon2013	57
OVH	http://www.ovh.com	10, 11
Silicon Mechanics	http://www.siliconmechanics.com	3
USENIX Security	https://www.usenix.org/conference/usenixsecurity13	79

ATTENTION ADVERTISERS

The *Linux Journal* brand's following has grown to a monthly readership nearly one million strong. Encompassing the magazine, Web site, newsletters and much more, *Linux Journal* offers the ideal content environment to help you reach your marketing objectives. For more information, please visit http://www.linuxjournal.com/advertising.