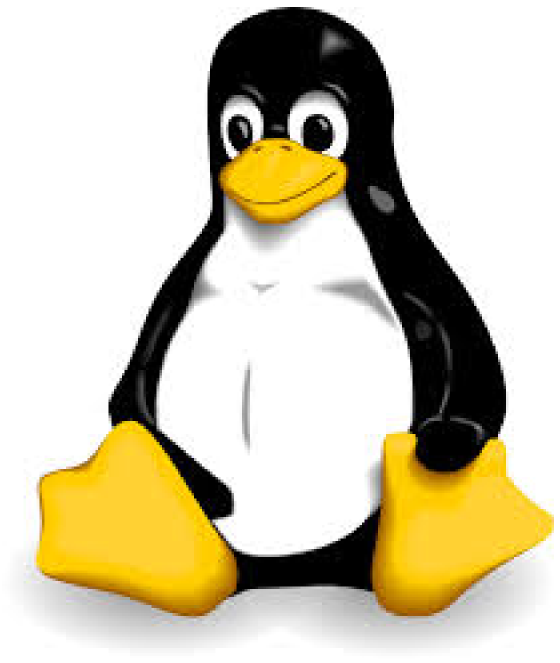
Engineering Notebook Documentation

Release 0.3.0

Kevin Walchko

October 25, 2016

Contents



These are some useful ramblings of Linux/Unix/OSX and general engineering. This is mainly targeted for me, so I can remember how to do all of this stuff without constantly asking Google. However, I hope others can find use for it too.

CHAPTER 1

Computers

1.1 SSH

Secure Shell (SSH) is a cryptographic network protocol for secure data communication, remote shell services or command execution and other secure network services between two networked computers that connects, via a secure channel over an insecure network, a server and a client (running SSH server and SSH client programs, respectively). The protocol specification distinguishes between two major versions that are referred to as SSH-1 and SSH-2.

The best-known application of the protocol is for access to shell accounts on Unix-like operating systems, but it can also be used in a similar fashion for accounts on Windows. It was designed as a replacement for Telnet and other insecure remote shell protocols such as the Berkeley rsh and rexec protocols, which send information, notably passwords, in plaintext, rendering them susceptible to interception and disclosure using packet analysis. The encryption used by SSH is intended to provide confidentiality and integrity of data over an unsecured network, such as the Internet. [1]

1.1.1 Summary of Useful Commands

Command	Example	Use
ssh	ssh kevin@thor.local	login to a computer
ssh-keygen	ssh-keygen	generate an ssh key
ssh-keygen	ssh-keygen -lvf	view the key finger print
ssh-copy-id	ssh-copy-id kevin@loki.local	copy key to remote server

1.1.2 Key Generation

To increase security, you can disable password logins and rely on ssh public keys. To do this, take a look here for details. Basic steps are:

1. Generate an ssh key pair using either RSA (2048-4096 bit) or DSA (1024 bit) both public and private keys. They will be stored in `~/.ssh` with the public key having .pub appended to the end. Two ways to generate keys are shown below (pick one).

```
ssh-keygen -t dsa -b 1024 -C "$(whoami)@$(hostname)-$(date)"
ssh-keygen -t rsa -b 4096 -C "$(whoami)@$(hostname)-$(date)"
```

   Note you can create a key for a different username if you change $(whoami) to the user name you want. If no type is specified, the default is RSA 2048 bits.

[1] Wikipedia entry source

TOP
SECRET

```
[kevin@Tardis ~]$ ssh-keygen -C "test@$(hostname)-$(date)"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/kevin/.ssh/id_rsa): test
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in test.
Your public key has been saved in test.pub.
The key fingerprint is:
00:04:27:9d:7e:33:6f:65:1c:a5:e0:c3:82:5d:7b:92 test@Tardis.local-Tue Apr 21 22:29:40 MDT 2015
The key's randomart image is:
+--[ RSA 2048]----+
|  o++.  o  ..    |
|   oo+ + +..     |
|   .. + E.o.     |
|    . +o ++      |
|     . +So       |
|        o        |
|       .         |
|                 |
|                 |
+-----------------+
```

Also note, it is advisable you create a strong pass phrase that you won't forget. However, I typically do not create one. But it does add an added level of protection.

2. Copy the public key (.pub) to the server you will connect to:

```
ssh-copy-id username@remote-server.org
```

This will update ~/.ssh/authorized_keys in the process. **Note:** `ssh-copy-id` may need to be installed. Most Linux/Unix systems should have this, but for OSX do `brew install ssh-copy-id`. Also ensure the correct protections are on the file by:

```
chmod 600 ~/.ssh/authorized_keys
```

3. Edit /etc/ssh/sshd_config to disable password logins.

```
PasswordAuthentication no
ChallengeResponseAuthentication no
```

1.1.3 SSH Key Finger Prints

To view the finger print of a key:

```
[kevin@Tardis ~]$ ssh-keygen -lvf ~/.ssh/id_rsa.pub
    2048 b1:58:41:c5:93:b3:bc:c7:34:5b:e8:be:bc:15:ff:55  kevin@tardis.local (RSA)
    +--[ RSA 2048]----+
    |       .oo..     |
    |         .=      |
    |        o. + .   |
    |       o oo + .  |
    |      . S  = +. E|
    |          . =  o.|
    |           o  . o|
    |           ...  o|
    |            +o  .|
    +-----------------+
```

This tells you the type of key (e.g., RSA or DSA), the bit size, what email/account it is tied to, and a graphical representation of the key. In this case, the 2048 bits of my public RSA key.

1.1.4 16 SSH Hacks

The original source for this work is here

So you think you know OpenSSH inside and out? Test your chops against this hit parade of 16 expert tips and tricks, from identifying monkey-in-the-middle attacks to road warrior security to attaching remote screen sessions. Follow the countdown to the all-time best OpenSSH command!

Running SSH on a non-standard port

SSH tips #16-14:Detecting MITM attacks

When you log into a remote computer for the first time, you are asked if you want to accept the remote host's public key. Well how in the heck do you know if you should or not? If someone perpetrated a successful monkey-in-the-middle attack, and is presenting you with a fake key so they can hijack your session and steal all your secrets, how are you supposed to know? You can know, because when new key pairs are created they also create a unique fingerprint and randomart image:

```
$ ssh-keygen -t rsa -C newserver -f .ssh/newkey

Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in .ssh/newkey.
Your public key has been saved in .ssh/newkey.pub.
The key fingerprint is:
44:90:8c:62:6e:53:3b:d8:1a:67:34:2f:94:02:e4:87 newserver
The key's randomart image is:
+--[ RSA 2048]----+
|oo   +.o.        |
|. = B o.         |
| E X +   .       |
|  B B ..         |
| . * o  S        |
|  .              |
|                 |
|                 |
|                 |
+-----------------+
```

SSH tip #16: Retrieve the fingerprint and randomart image of an SSH key

If you make a copy of this when you create new encryption keys, then you can fetch a key's fingerprint and randomart image anytime to compare and make sure they have not changed:

```
$ ssh-keygen -lvf  keyname
```

SSH tip #15: View all fingerprints and randomart images in known_hosts

And you can see all of them in your ~/.ssh/known_hosts file:

```
$ ssh-keygen -lvf ~/.ssh/known_hosts
```

SSH tip #14: Verify server keys

You can see the fingerprint and randomart for any computer you're logging into by configuring/etc/ssh/ssh_config on your client computer. Simply uncomment the VisualHostKey option and set it to yes:

```
VisualHostKey yes
```

Then login to any remote computer to test it:

```
$ ssh user@host2
Host key fingerprint is 66:a1:2a:23:4d:5c:8b:58:e7:ef:2f:e5:49:3b:3d:32
+--[ECDSA  256]---+
|                 |
|                 |
|  . o   .        |
| + = . . .       |
|. + o . S        |
| o   o oo        |
|. + . .+ +       |
| . o .. E o      |
|      .o.+ .     |
+-----------------+

user@host2's password:
```

Obviously you need a secure method of getting verified copies of the fingerprint and randomart images for the computers you want to log into. Like a hand-delivered printed copy, encrypted email, the scp command, secure ftp, read over the telephone...The risk of a successful MITM attack is small, but if you can figure out a relatively painless verification method it's cheap insurance.

SSH tip #13: Attach to a remote GNU screen session

You can attach a GNU screen session remotely over SSH; in this example we'll open a GNU screen session on host1, and connect to it from host2. First open and then detach a screen session on host1, named testscreen:

```
host1 ~ $ screen -S testscreen
```

Then detach from your screen session with the keyboard combination Ctrl+a+d:

```
[detached from 3829.testscreen]
```

You can verify that it's still there with this command:

```
host1 ~ $ screen -ls
```

There is a screen on:

```
3941.testscreen (03/18/2012 12:43:42 PM) (Detached)
1 Socket in /var/run/screen/S-host1.
```

Then re-attach to your screen session from host2:

```
host1 ~ $ ssh -t terry@uberpc screen -r testscreen
```

You don't have to name the screen session if there is only one.

vSSH tip #12: Launch a remote screen session

What if you don't have a running screen session? No worries, because you can launch one remotely:

```
host1 ~ $ ssh -t user@host2 /usr/bin/screen -xRR
```

SSH tip #11: SSHFS is better than NFS

sshfs is better than NFS for a single user with multiple machines. I keep a herd of computers running because it's part of my job to always be testing stuff. I like having nice friendly herds of computers. Some people collect Elvis plates, I gather computers. At any rate opening files one at a time over an SSH session for editing is slow; with sshfs you can mount entire directories from remote computers. First create a directory to mount your sshfs share in:

```
$ mkdir remote2
```

Then mount whatever remote directory you want like this:

```
$ sshfs user@remote2:/home/user/documents remote2/
```

Now you can browse the remote directory just as though it were local, and read, copy, move, and edit files all you want. The neat thing about sshfs is all you need is sshd running on your remote machines, and thesshfs command installed on your client PCs.

SSH tip #10: Log in and run a command in one step

You can log in and establish your SSH session and then run commands, but when you have a single command to run why not eliminate a step and do it with a single command? Suppose you want to power off a remote computer; you can log in and run the command in one step:

```
carla@local:~$ ssh user@remotehost sudo poweroff
```

This works for any command or script. (The example assumes you have a sudo user set up with appropriate restrictions, because allowing a root login over SSH is considered an unsafe practice.) What if you want to run a long complex command, and don't want to type it out every time? One way is to put it in a Bash alias and use that. Another way is to put your long complex command in a text file and run it according to tip #9.

SSH tip #9: Putting long commands in text files

Put your long command in a plain text file on your local PC, and then use it this way to log in and run it on the remote PC:

```
carla@local:~$ ssh user@remotehost "`cat filename.txt`"
```

Mind that you use straight quotations marks and not fancy ones copied from a Web page, and back-ticks, not single apostrophes.

vSSH tip #8: Copy public keys the easy way

The ssh-copy-id command is not as well-known as it should be, which is a shame because it is a great time-saver. This nifty command copies your public key to a remote host in the correct format, and to the correct directory. It even has a safety check that won't let you copy a private key by mistake. Specify which key you want to copy, like this:

```
$ ssh-copy-id -i .ssh/id_rsa.pub user@remote
```

SSH tip #7: Give SSH keys unique names

Speaking of key names, did you know you can name them anything you want? This helps when you're administering a number of remote computers, like this example which creates then private key web-admin and public key web-admin.pub:

```
$ ssh-keygen -t rsa -f .ssh/web-admin
```

SSH tip #6: Give SSH keys informative comments

Another useful way to label keys is with a comment:

```
$ ssh-keygen -t rsa -C "downtown lan webserver" -f .ssh/web-admin
```

Then you can read your comment which is appended to the end of the public key.

SSH tip #5: Read public key comments

```
$ less .ssh/web-admin.pub

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC1

[snip] KCLAqwTv8rhp downtown lan webserver
```

SSH tip #4: Logging in with server-specific keys

Then when you log in, specify which key to use with the -i switch:

```
$ ssh -i .ssh/web-admin.pub user@webserver
```

SSH tip #3: Fast easy known_hosts key management

I love this one because it's a nice time-saver, and it keeps my ~/.ssh/known_hosts files tidy: using ssh-keygen to remove host keys from the ~/.ssh/known_hosts file. When the remote machine gets new SSH keys you'll get a warning, when you try to log in, that the key has changed. Using this is much faster than manually editing the file and counting down to the correct line to delete:

```
$ ssh-keygen -R remote-hostname
```

Computers are supposed to make our lives easier, and it's ever so lovely when they do.

SSH tip #2: SSH tunnel for road warriors

When you're at the mercy of hotel and coffee shop Internet, a nice secure SSH tunnel makes your online adventures safer. To make this work you need a server that you control to act as a central node for escaping from hotspot follies. I have a server set up at home to accept remote SSH logins, and then use an SSH tunnel to route traffic through it. This is useful for a lot of different tasks. For example I can use my normal email client to send email, instead of hassling with Web mail or changing SMTP server configuration, and all traffic between my laptop and home server is encrypted. First create the tunnel to your personal server:

```
carla@hotel:~$ ssh -f carla@homeserver.com -L 9999:homeserver.com:25 -N
```

This binds port 9999 on your mobile machine to port 25 on your remote server. The remote port must be whatever you've configured your server to listen on. Then configure your mail client to use localhost:9999 as the SMTP server and you're in business. I use Kmail, which lets me configure multiple SMTP server accounts and then choose which one I want to use when I send messages, or simply change the default with a mouse click. You can adapt this for any kind of service that you normally use from your home base, and need access to when you're on the road.

1 Favorite SSH tip: Evading silly web restrictions

The wise assumption is that any public Internet is untrustworthy, so you can tunnel your Web surfing too. My #1 SSH tip gets you past untrustworthy networks that might have snoopers, and past any barriers to unfettered Web-surfing. Just like in tip #2 you need a server that you control to act as a secure relay; first setup an SSH tunnel to this server:

```
carla@hotel:~$ ssh -D 9999 -C carla@homeserver.com
```

Then configure your Web browser to use port 9999 as a SOCKS 5 proxy. Figure 1 shows how this looks in Firefox.

Figure 1: Configuring Firefox to use your SSH tunnel as a SOCKS proxy. An easy way to test this is on your home or business network. Set up the tunnel to a neighboring PC and surf some external Web sites. When this works go back and change the SOCKS port number to the wrong number. This should prevent your Web browser from connecting to any sites, and you'll know you set up your tunnel correctly. How do you know which port numbers to use? Port numbers above 1024 do not require root privileges, so use these on your laptop or whatever you're using in your travels. Always check /etc/services first to find unassigned ports. The remote port you're binding to must be a port a server is listening on, and there has to be a path through your firewall to get to it.

To learn more try the excellent [Pro OpenSSH by Michael Stahnke] (http://www.apress.com/networking/openssh/9781590594766), and my own Linux Networking Cookbook has more on secure remote administration including SSH, OpenVPN, and remote graphical sessions, and configuring firewalls.

1.2 Git

1.2.1 Git Cheat Sheet

Command	Example	Definition
Init	`git init`	Start a repository
Clone	`git clone git://somewhere.com/something.git [new_name]`	Create a copy of a repository
Remote	`git remote -v`	Display remote repository
Push	`git push origin [master]`	Share changes to an upstream remote (origin) for a branch
Commit	`git commit -m 'update'`	One step add/push upstream
Add	`git add *.c`	Add files to the repository or to stage files for commit
Fetch	`git fetch [origin]`	Pull all changes from a remote repository that have been pushed since you cloned it. Note this doesn't merge any of the changes.
Pull	`git pull`	Will automatically `fetch` and merge changes from upstream into current branch
Tag	`git tag -a v1.4 -m 'my Version 1.4'`	Create an annotated tag, note you may have to: `git push origin v1.4` or `git push origin --tags` to get tags push on upstream server.
Status	`git status`	Reports the status of untracked changes in your working repository
Diff	`git diff`	Display changes between working directory and repository
Rm	`git rm file.c`	Remove files from repository
Mv	`git mv from_file to_file`	Move files
Log	`git log`	View the commit log

1.2.2 Git Setup

Let's start off with making git look nice:

```
git config color.ui true
```

Since we will work remotely, we need to tell git who we are. Git stores that info in `~/.gitconfig`

```
git config --global user.name "walchko"
git config --global user.email kevin.walchko@hotmail.com
```

1.2.3 Working with Git

First clone a repository, make sure you use the `ssh` address and not the default https one

```
git clone git@github.com:walchko/soccer.git
```

Note: The https one has `https` in the address: `https://github.com/walchko/soccer2.git`

if you accidentally clone the `https` one, you can switch to `ssh` by

```
git remote set-url origin git@github.com:walchko/soccer2.git
```

Now create ssh keys following the github directions

basically:

1. create a key: ssh-keygen -t rsa -C "pi@bender.local":

```
pi@bender ~ $ eval "$(ssh-agent -s)"
Agent pid 12480
pi@bender ~ $ ssh-add ~/.ssh/id_rsa
Identity added: /home/pi/.ssh/id_rsa (/home/pi/.ssh/id_rsa)
```

2. Go to github and add a new ssh key under your profile. Copy/paste in the key (use `more ~/.ssh/id_rsa.pub`) making sure not to add or remove white space. You can use `pbcopy < ~/.ssh/id_rsa.pub` to copy it to your clip board.
3. Then try to ssh in:

```
pi@bender ~ $ ssh -T git@github.com
The authenticity of host 'github.com (192.30.252.128)' can't be established.
RSA key fingerprint is 1d:57:ac:a4:76:23:2d:34:63:1b:56:4d:74:7f:76:48.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.252.128' (RSA) to the list of known hosts.
Hi walchko! You've successfully authenticated, but GitHub does not provide shell access.
```

Success ... enjoy!

1.2.4 Git Workflow

Read this awesome guide

1. Make sure your current copy is up to date

```
git pull
```

2. Create a new branch to hold your new feature

```
git checkout -b my-cool-new-thing
```

3. Edit your code. To see status:

```
git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   docs/computers/git.rst

no changes added to commit (use "git add" and/or "git commit -a")
```

4. Mark files for change

```
git add *
```

5. Commit files (locally) to HEAD

```
git commit -m "what did you do?"
pi@bender ~/soccer/IMU $ git push origin master
Counting objects: 12, done.
```

```
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 736 bytes, done.
Total 8 (delta 6), reused 0 (delta 0)
To git@github.com:walchko/soccer.git
   8162ade..cd9a476  master -> master
```

6. Push changes upstream, back to the repository so everyone can use them

   ```
   git push origin master
   ```

   or `git push origin`

7. Create a tag

   ```
   git tag -a v0.5.3 -m "update"
   git push origin v0.5.3
   ```

To undo what you have committed already and basically create an anti-patch for each commit

```
git revert 0766c053 25eee4ca a867b4af
```

1.3 Curl

This cheatsheet came from mixu.

Simple GET request

```
curl -k "https://localhost/foo?bar=baz&amp;abc=def"
```

JSON POST or PUT request

```
curl -k -H "Content-Type: application/json" -X POST -d '{"accountName":"test","value":"hello"}' https
```

or

```
curl -X "PUT"
```

POST a file

```
curl ... --data-binary @filename
```

Fake a /etc/hosts entry and a Host: header with curl

```
curl -vvv --resolve 'book.mixu.net:80:123.145.167.189' http://book.mixu.net/
```

Make a request with basic auth enabled

```
curl -vvv -u name@foo.com:password http://www.example.com
```

or

```
curl --user name:password http://www.example.com
```

Set the Referer header

```
curl -e http://curl.haxx.se daniel.haxx.se
```

Set the User Agent header

```
curl -A "Mozilla/4.73"
```

or

```
curl --user-agent "Mozilla".
```

Set Cookies

```
curl -b "name=Daniel"
```

or

```
curl --cookie "name=Daniel"
```

Time a request (connect time + time to first byte + total tile)

```
curl -o /dev/null -w "Connect: %{time_connect} TTFB: %{time_starttransfer} Total time: %{time_total}
```

Downloading files from Github

```
curl -O  https://raw.github.com/username/reponame/master/filename
```

1.4 Python

1.4.1 Python Packages

Alot of very useful packages are available from PyPI and can be installed using `pip`.

You can use `pip` to install and keep python libraries up to date. Unfortunately `pip` isn't the best package manager, but it could be worse ... `apt-get` anyone? Some useful, undocumented commands:

Pip flag	Description
list	list installed packages
list –outdated	list packages that can be upgraded
install *pkg*	install a package
install -U *pkg*	upgrade a package

Why the people who run `pip` don't make useful commands like `pip upgrade` or `pip outdated` I don't know. Instead there are duplicate commands like `pip freeze` which is the same as `pip list` and adds no real value.

1.4.2 Python Modules

Simple structure:

```
module_name
-- CONTRIB
-- LICENSE
-- MANIFEST.in
-- setup.py
-- README.rst
-- module
   -- __init__.py
   -- script1.py
   -- script2.py
```

Install from source:

```
[kevin@Tardis media_server]$ sudo python setup.py develop
running develop
/usr/local/lib/python2.7/site-packages/pkg_resources/__init__.py:2510: PEP440Warning: 'pygame (1.9.1r
PEP440Warning,
running egg_info
writing requirements to media.egg-info/requires.txt
writing media.egg-info/PKG-INFO
writing top-level names to media.egg-info/top_level.txt
writing dependency_links to media.egg-info/dependency_links.txt
writing entry points to media.egg-info/entry_points.txt
reading manifest file 'media.egg-info/SOURCES.txt'
reading manifest template 'MANIFEST.in'
warning: no previously-included files matching '*.html' found under directory '*'
warning: no previously-included files matching '.AppleDouble' found under directory '*'
warning: no previously-included files matching '*.yaml' found under directory '*'
warning: no previously-included files matching '.gitignore' found under directory '*'
warning: no previously-included files matching '*.pyc' found under directory '*'
writing manifest file 'media.egg-info/SOURCES.txt'
running build_ext
Creating /usr/local/lib/python2.7/site-packages/media.egg-link (link to .)
Adding media 0.1.0 to easy-install.pth file
Installing media script to /usr/local/bin

Installed /Users/kevin/github/media_server
Processing dependencies for media==0.1.0
Searching for PyYAML==3.11
Best match: PyYAML 3.11
PyYAML 3.11 is already the active version in easy-install.pth

Using /Library/Python/2.7/site-packages
Searching for tmdb3==0.7.2
Best match: tmdb3 0.7.2
Adding tmdb3 0.7.2 to easy-install.pth file

Using /usr/local/lib/python2.7/site-packages
Searching for rottentomatoes==2.1
Best match: rottentomatoes 2.1
Adding rottentomatoes 2.1 to easy-install.pth file

Using /usr/local/lib/python2.7/site-packages
Finished processing dependencies for media==0.1.0
```

Uninstall

```
sudo pip uninstall module
```

Run command line program

```
python -m module.script
```

where `module` is your package name and `script` is the python script that does something.

1.4.3 PyPi

Some good resources are Python Packaging Guide and Tom Christie for more info.

1. Create an account at pypi.org
2. Create a package repository at pypi.org using the web form and uploading the PKG-INFO file
3. Run a test to ensure no problems `python setup.py test`
4. Create the package for upload `python setup.py sdist`
5. Upload package to pypi.org `twine upload dist/*`

Twine can be installed using `pip install twine` which will secure your upload and protect your password. Also the username and password are stored in a `.pypirc` in your home directory.

1.5 Arch Linux

1.5.1 Arch Linux

I like arch linux better than raspbian is a lot of ways, except:

- default is python 3
- not as fully supported (tutorials, software packages, etc) on RPi as raspbian is

Basic Install Process

Don't follow the install instructions, they suck. Instead, look at the Beginner's Guide on the wiki.

Virtualbox Guest

Follow these instructions for the install.

1. Install this package:

   pacman -S virtualbox-guest-utils

2. Create the module file /etc/modules-load/virtualbox.conf

   vboxguest vboxsf vboxvideo

Post-Install Packages

```
pacman -S package
```

Useful packages:

- distcc
- sudo - edit /etc/sudoers, uncomment wheel group
- avahi and nss-mdns
- openssh
- virtualbox-guest-utils

Update System

```
pacman -Syu
```

Fonts

Fontconfig configuration is done via `/etc/fonts/conf.avail` and `conf.d`. Read `/etc/fonts/conf.d/README` for more information.

Configuration via `/etc/fonts/local.conf` is still possible, but is no longer recommended for options available in conf.avail.

Main system wide configuration should be done by symlinks (especially for autohinting, sub-pixel and lcdfilter):

```
cd /etc/fonts/conf.d
ln -s ../conf.avail/XX-foo.conf
```

Check also https://wiki.archlinux.org/index.php/Font_Configuration and https://wiki.archlinux.org/index.php/Fonts.

Arch Networking

Avahi - Multicast

You can enable Avahi Daemon at startup with the following command:

```
systemctl enable avahi-daemon.service
```

SSH

For the client edit /etc/ssh/ssh_config remove protocol 1 since it is deemed insecure and only use 2:

```
Protocol 2
```

For the server edit /etc/ssh/sshd_config enable:

```
AllowUsers user1 user2 (change to appropriate user names)
PermitRooLogin no
Banner /etc/issue
```

Then add it to the DAEMONS list in /etc/rc.conf, so it starts on boot:

** Don't use rc.conf anymore!! **

```
DAEMONS=( .... sshd ....)
```

You can also start it immediately by:

```
sudo rc.d start sshd
```

To see if it worked, type:

```
ps -e | grep sshd
```

1.6 OS X

1.6.1 OSX

This is my main unix operating system I use daily.

Software Update

You can use the command line to update your OSX systems remotely:

```
softwareupdate --install
```

-l, --list	List packages available
-i, --install arg	Install packages where arg can be: **-a, --all** All packages; **-r, --recommended** Just the recommended packages
--schedule arg	Setup scheduling downloads, either on or off

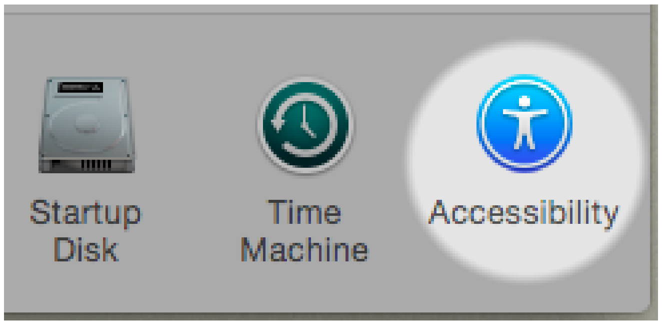
OSX Fixes

Enable One-Finger Tap & Drag

Step 1: On your Mac, first open the Preferences panel. There, head to the Accessibility option as shown below.

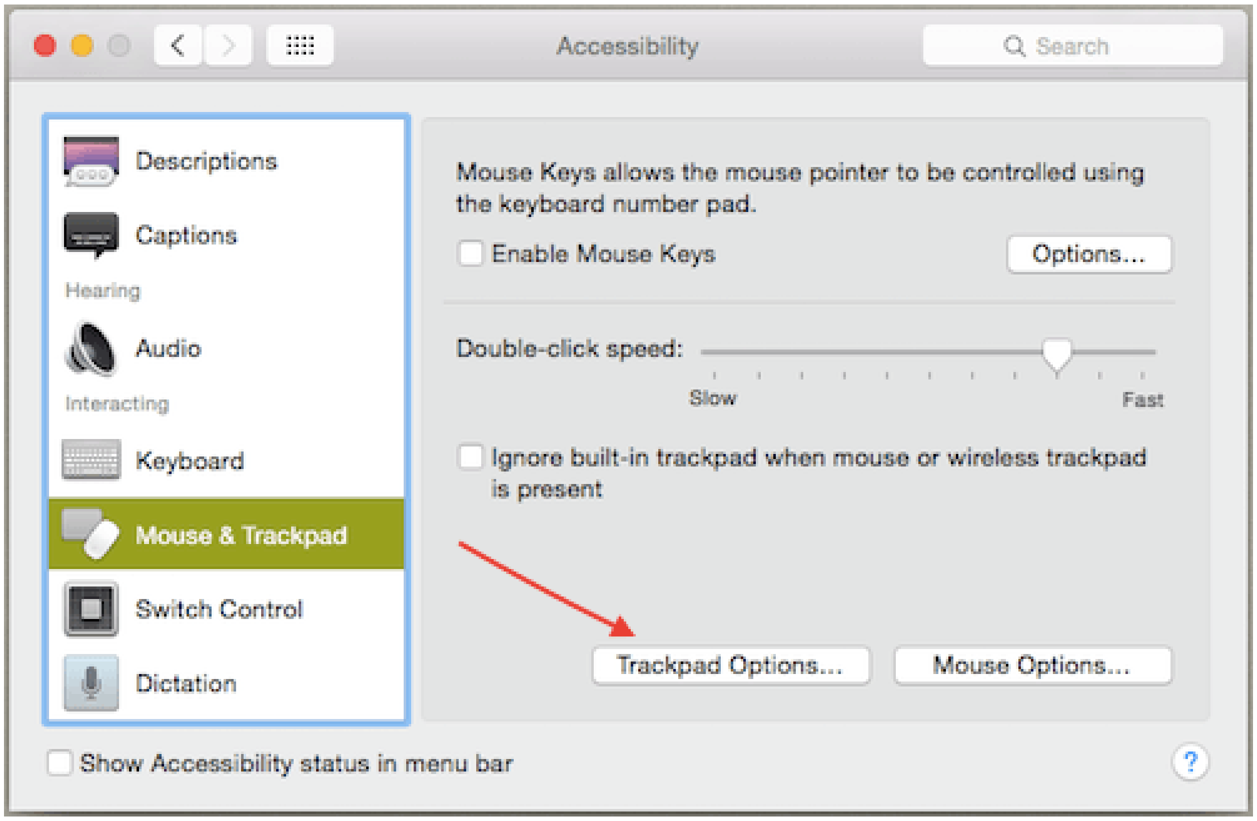
Preferences

Preferences Accessibility



Step 2: On the left panel of the next window, scroll down and select the Mouse & Trackpad option. There, click on the Trackpad Options… button.

Mouse and Trackpad



Mouse Keys allows the mouse pointer to be controlled using the keyboard number pad.

☐ Enable Mouse Keys

Double-click speed:

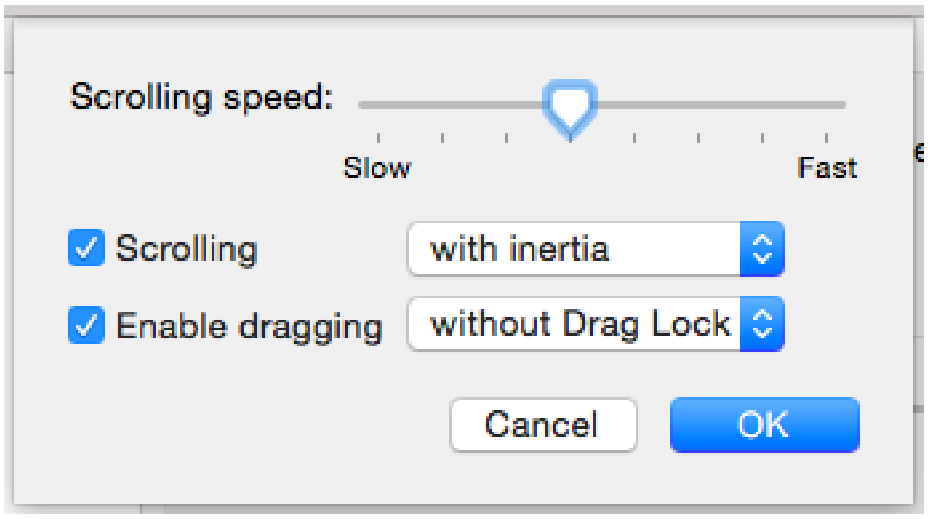
☐ Ignore built-in trackpad when mouse or wireless trackpad is present

☐ Show Accessibility status in menu bar

This will bring down an options panel.

Step 3: On this panel, you will notice an interesting option at the bottom named Enable dragging. Check it.

Preferences Enable Dragging



Scrolling speed:

☑ Scrolling: with inertia

☑ Enable dragging: without Drag Lock

This option not only allows you to enable dragging on your Mac’s trackpad, but also lets you activate or deactivate drag lock (when enabled, this means that when you drag a window, you will have to perform one more click in order to deactivate the drag function).

And done! Now whenever you use your Mac’s trackpad, just double-tap on any ‘draggable’ area of a window to move it around.

Stop Safari re-opening windows

Hold the shift key down and open Safari

```
defaults write com.apple.Safari ApplePersistenceIgnoreState YES
defaults write com.apple.Preview ApplePersistenceIgnoreState YES
```

Stopping App Store Downloads

Pausing downloads is easy, but if you start one and want to cancel it hold Option key and click on cancel.

Useful Software

My hard drive crashed (29 April 2015) and I had to re-install everything. Oh, a USB drive attached to Airport Extreme as your Time Machine backup doesn't work! When I tried to reinstall from that, it complained about corrupt Time Machine files or some such crap. Connect your drive directly to your computer.

So, here is all the software I had to re-install:

- Github
- BitTorrent Sync
- Chrome
- Atom
- Beamer
- Dropbox
- Google Drive

1.6.2 Virtualbox

1.6.3 Raspberry Pi VM Setup

This HOWTO follows the work by Russell Davis for setting up a cross compiling environment for the Raspberry Pi. I have modified his instructions to make them more ROS specific.

If you don't already have Virtualbox and the Extension pack installed then you'll need to download it from https://www.virtualbox.org/wiki/Downloads, choosing the correct one for your platform and also download the Extension Pack from the same page if you haven't already installed it. Once you have downloaded and installed Virtualbox+Extension pack for your platform move on to step 2.

1. You can download the Ubuntu iso from the Ubuntu website
2. Once you have the Ubuntu iso downloaded, start Virtualbox and create a new Virtual machine by clicking the New button. Virtualbox will then start the Virtual Machine Wizard. Give your VM a name. I suggest something like RaspberryPi Development. Choose Linux & Ubuntu from the dropdowns (or if you are not going to use Ubuntu as the guest os then select whichever distro you are going to use). It should look something like this
3. Choose the amount of memory to allocate to the VM, give it 1 or 2 GB of ram.
4. Create a virtual harddisk and choose VDI which will be dynamically allocated. Make it at least 8GB to hold everything. The harddisk will resize dynamically.
5. Click the Create button. You will then be returned to the main Virtualbox screen with the VM you have just created highlighted click the Settings button.

6. Now open up the setting for the VM you just created. The only setting you MUST change:

- storage
- networking
- video

7. Go to the storage tab and click the little CD icon. Add the iso you downloaded earlier from Ubuntu. This will only effect this boot cycle. After you install Linux, the disk image will automatically be removed.
8. Go to the networking tab and ensure NAT is selected so you can see the interweb. This will also allow you to install updates and 3rd pa
9. Go to the video tab and sensure 3D acceleration is turned off and there is enough video memory for your VM ... I selected 32 MB.

1.6.4 Install Linux

Now hit start on the Virtualbox screen and install Linux.

If you this error message:

```
piix4_smbus 0000.00.07.0: SMBus base address uninitialized - upgrade bios or use force_addr=0xaddr
```

Fix based on work by Karl Foley, in a terminal type:

```
sudo vi /etc/modprobe.d/blacklist.conf
```

Add the line blacklist i2c_piix4 to the end of the file and save

```
sudo update-initramfs -u -k all
sudo reboot
```

Also make sure you have the development tools installed

```
sudo apt-get install build-essential
```

Also to reduce the size of the install, I uninstalled office, game, and other unneeded software.

Bonjour

see kinect/README.md

SSH Server

see kinect/README.md

1.6.5 VM Commands

VBoxHeadless –startvm vb_ros

VBoxManage controlvm vb_ros poweroff | pause | reset

Networking

In order for your vm to see the internet and other attached vm and computers, you must use a bridged connection. However on **OSX** the bridged network doesn't work if you are using a wireless connection (airport). A real wired connection works fine in OSX (10.8.2 tested).

1.6.6 RPI Tool Chain

The Raspberry Pi Foundation is providing a ready-to-use toolchain on their github repository. You can use it to save yourself some time.

To do so, you need to have git installed and to clone the repository

```
> sudo apt-get install git-core
> git clone https://github.com/raspberrypi/tools.git --depth=1
> export PATH=$PATH:$HOME/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin
```

The "–depth=1" is here to tell git we only want the last revision, and not the whole history to be cloned.

Create a new file named test.cpp and copy/paste the following code:

```
#include <iostream>

int main(void)
{
    std::cout<<"Hello ARM world !\n";
    return 0;
}
```

Then, enter the following commands:

```
> arm-linux-gnueabihf-g++ test.cpp -o test
> file test
test: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked (uses shared libs), for GN
```

As you see, you can't execute this program on your PC. The file command tells you that this executable is built for ARM processors.

1.7 Raspbian

1.7.1 Raspberry Pi (RPi)

Where to buy?

I always buy mine from Adafruit, they have tons of other great stuff at great prices. They also make make lots of example code and drivers available for their products.

Install

Raspbian is a Raspberry optimized version of Debian. The version installed here is based on Debian Wheezy.

```
[kevin@raspberrypi ~]$ lscpu
Architecture:          armv6l
Byte Order:            Little Endian
```

```
CPU(s):                1
On-line CPU(s) list:   0
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             1
```

Note this output doesn't really tell you much other than it is ARMv6.

Copying an image to the SD Card in Mac OS X

These commands and actions need to be performed from an account that has administrator privileges.

1. Download the image from a mirror or torrent.
2. Verify if the the hash key is the same (optional), in the terminal run:

```
shasum ~/Downloads/debian6-19-04-2012.zip
```

3. Extract the image:

```
unzip ~/Downloads/debian6-19-04-2012.zip
```

4. Attach the SD Card to the computer and identify the mount point:

```
df -h
```

   Record the device name of the filesystem's partition, e.g. `/dev/disk3s1`

5. Unmount the partition so that you will be allowed to overwrite the disk, note that unmount is **NOT** the same as eject:

```
sudo diskutil unmount /dev/disk3s1
```

   Using the device name of the partition work out the raw device name for the entire disk, by omitting the final "s1" and replacing "disk" with "rdisk" (this is very important: you will lose all data on the hard drive on your computer if you get the wrong device name). Make sure the device name is the name of the whole SD card as described above, not just a partition of it (for example, rdisk3, not rdisk3s1. Similarly you might have another SD drive name/number like disk2 or disk4, etc. – recheck by using the df -h command both before & after you insert your SD card reader into your Mac if you have any doubts!): e.g. `/dev/disk3s1` => `/dev/disk3`

   Note: Using rdisk3 might be faster than using disk3, need to look into this.

6. Write the image to the card with this command, using the raw disk device name from above (read carefully the above step, to be sure you use the correct rdisk# here!):

```
sudo dd bs=1m if=~/rasbian.img of=/dev/rdisk3
```

   If the above command report an error(dd: bs: illegal numeric value), please change bs=1M to bs=1m.

   Note that dd will not feedback any information until there is an error or it is finished, information will show and disk will re-mount when complete. However if you are curious as to the progresss - ctrl-T (SIGINFO, the status argument of your tty) will display some en-route statistics.

7. After the dd command finishes, eject the card:

```
sudo diskutil eject /dev/disk3
```

8. Insert it in the raspberry pi, and have fun

Configuration

Once you download and install Raspbian you have to configure it for it to be useful.

1. **`sudo raspi-config` and change**
   (a) update `raspi-config` via the advanced option, update
   (b) hostname
   (c) memory split between GPU and RAM
   (d) resize the file system to the size of your disk
   (e) set correct timezone via the internationalization option
   (f) turn on I2C interface
2. `sudo apt-get update` and then `sudo apt-get upgrade`
3. `sudo apt-get install apt-show-versions`
4. `sudo easy_install pip` then `sudo pip install -U pip` to get the latest pip version
5. `sudo apt-get install rpi-update` and then `sudo rpi-update` to update the kernel

6. **Fix the pip paths so you don't have to use sudo (that is a security risk)**
   (a) `sudo chown -R pi /usr/local`
   (b) `sudo chown -R pi /usr/lib/python2.7/dist-packages`
   (c) `sudo chown -R pi /usr/share/pyshare`
7. **Fix the `pip` certificate warnings**
   (a) `sudo apt-get install python-dev libffi-dev`
   (b) `pip install -U urllib3 certifi pyopenssl`
8. Find outdated python libraries with `pip list --outdated` then update them with `pip install -U package_name`

SSH Login

To increase security, you can disable password logins and rely on ssh public keys. To do this, take a look here for details. Basic steps are:

1. Generate an ssh key pair using either RSA (2048-4096 bit) or DSA (1024 bit) both public and private keys. They will be stored in `~/.ssh` with the public key having .pub appended to the end:

```
ssh-keygen -C "$(whoami)@$(hostname)-$(date -I)"
```

   Note you can create a key for a different username if you change $(whoami) to the user name you want.

2. Copy the public key (.pub) to the server you will connect to:

```
ssh-copy-id username@remote-server.org
```

   This should update ~/.ssh/authorized_keys in the process. Also ensure the correct protections are on the file by:

```
chmod 600 ~/.ssh/authorized_keys
```

3. Edit /etc/ssh/sshd_config to disable password logins.

```
PasswordAuthentication no
ChallengeResponseAuthentication no
```

OSX

On OSX install `ssh-copy-id` via `brew` and in a terminal window on OSX:

```
ssh-copy-id pi@raspberry.local
```

Sound

Double check sound works:

```
aplay /usr/share/sounds/alsa/Front_Center.wav
```

/boot/config.txt

You can change the Pi's default settings for CPU MHz and memory split (between RAM and GPU) using `raspi-config`. An alternate way is to simply edit the `/boot/config.txt`.

```
[kevin@raspberrypi ~]$ more /proc/cpuinfo
Processor    : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS     : 795.44
Features     : swp half thumb fastmult vfp edsp java tls
CPU implementer : 0x41
CPU architecture: 7
CPU variant : 0x0
CPU part     : 0xb76
CPU revision     : 7

Hardware     : BCM2708
Revision     : 0002
Serial       : 000000008e0a5a17

[kevin@raspberrypi ~]$ free -h
             total       used       free     shared    buffers     cached
Mem:          232M        57M       174M         0B        11M        28M
-/+ buffers/cache:        18M       214M
Swap:          99M         0B        99M
```

The output here shows overclocked to 800 MHz and the GPU given only 16 MB of RAM. Now the CPU MHz will change dynamically based on load. So with no load, my 800 MHz system will default to the original 700 MHz system. If you want to always be running at max speed, put `force_turbo=1` in the `/boot/config.txt`:

```
[kevin@raspberrypi ~]$ more /boot/config.txt
#uncomment to overclock the arm. 700 MHz is the default.
arm_freq=800

# for more options see http://elinux.org/RPi_config.txt
gpu_mem=16     # can be 16, 64, 128 or 256
core_freq=250
sdram_freq=400
over_voltage=0
force_turbo=1
```

More info can be found here.

1.7.2 Bluetooth

Dongle

Make sure your bluetooth dongle is working:

```
pi@calculon ~ $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 0a12:0001 Cambridge Silicon Radio, Ltd Bluetooth Dongle (HCI mode)
Bus 001 Device 005: ID 0bc2:3312 Seagate RSS LLC
```

Device 004 is my bluetooth dongle.

Audio

References 1 2

Get the bluetooth software:

```
sudo apt-get install bluetooth bluez-utils bluez-alsa
```

Add `pi` to the user group:

```
sudo gpasswd -a pi bluetooth
```

Start up bluetooth dongle:

```
sudo hciconfig hci0 up
```

Scan for bluetooth devices

```
pi@calculon ~ $ hcitool scan
Scanning ...
    44:2A:60:D6:49:34   Dalek
```

If you need to figure out what your bluetooth hardware address is for `hci0`:

```
hcitool dev
```

Pair with device using `bluetooth-agent --adapter hci0 <pin> <hardware_id>`:

```
bluetooth-agent --adapter hci0 0000 00:11:67:8C:17:80
```

See if device is trusted or not:

```
bluez-test-device trusted <hardware_id>
```

If it returns a `0` then it is not trusted and a `1` if it is trusted. To make it trusted:

```
bluez-test-device trusted <hardware_id> yes
```

Edit/create `~/.asoundrc` with:

```
pcm.bluetooth {
                type bluetooth
                device <hardware_id>
}
```

Note: You may have to copy this file and rename it to: `/etc/asound.conf`

Edit `/etc/bluetooth/audio.conf` and add the following to the `[General]` section:

```
Disable=Media
Enable=Socket,Sink,Source
```

Restart bluetooth service with:

```
sudo /etc/init.d/bluetooth restart
```

Audio Programs

```
mplayer -ao alsa:device=bluetooth sound.mp3
mpg321 -a bluetooth -g 15 sound.mp3
```

1.7.3 iBeacon

I2C

Get software:

```
sudo apt-get install libusb-dev libdbus-1-dev libglib2.0-dev libudev-dev libical-dev libreadline-dev
```

Additionally, get:

```
sudo apt-get install bluez bluez-tools python-bluez
```

Check status of bluetooth

```
$ hciconfig
hci0:   Type: BR/EDR  Bus: USB
BD Address: 00:1A:7D:DA:71:13  ACL MTU: 310:10  SCO MTU: 64:8
UP RUNNING
RX bytes:2401 acl:0 sco:0 events:119 errors:0
TX bytes:2155 acl:0 sco:0 commands:119 errors:0
```

If it is already up and running, you can shut it down with:

```
sudo tools/hciconfig hci0 down
```

Now turn it on:

```
sudo tools/hciconfig hci0 up
sudo tools/hciconfig hci0 leadv
sudo tools/hciconfig hci0 noscan
```

Then send one of these commands:

```
sudo hcitool -i hci0 cmd 0x08 0x0008 1E 02 01 1A 1A FF 4C 00 02 15 E2 0A 39 F4 73 F5 4B C4 A1 2F 17 D
sudo hcitool -i hci0 cmd 0x08 0x0008 1E 02 01 1A 1A FF 4C 00 02 15 E2 C5 6D B5 DF FB 48 D2 B0 60 D0 F
```

Format

```
hcitool -i hci0 cmd 0x08 0x0008 1E 02 01 1A 1A FF 4C 00 02 15 [UUID] [Major] [Minor] [Power]
```

where, Manufacturers Specific Data starts with FF and 4C 00 is for Apple. The other stuff is:

```
UUID: E2C56DB5-DFFB-48D2-B060-D0F5A71096E0
Major: 00 00
Minor: 00 00
Power: C8
```

You should see it appear on a bluetooth finder Locate Beacon by Radius Networks.

To stop transmission:

```
sudo hciconfig hci0 noleadv
```

iBeacon Software

`git clone https://github.com/carsonmcdonald/bluez-ibeacon.git`

iBeacon-Scanner: `git clone https://github.com/switchdoclabs/iBeacon-Scanner-.git`

BeaconAirPython: `git clone https://github.com/switchdoclabs/BeaconAirPython.git`

1.7.4 I2C

`sudo apt-get install python-smbus sudo apt-get install i2c-tools`

```
pi@bender ~ $ sudo i2cdetect -y 1 0 1 2 3 4 5 6 7 8 9 a b c d e f 00: --
-- -- -- -- -- -- -- -- -- -- -- -- 10: -- -- -- -- -- -- -- -- 18 -- --
-- -- -- 1e -- 20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- 30:
-- -- -- -- -- -- -- -- -- -- -- UU -- -- -- -- 40: -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- 50: -- -- -- -- -- -- -- -- -- -- -- -- -- --
-- -- 60: -- -- -- -- -- -- -- -- -- 69 -- -- -- -- -- -- 70: -- -- --
-- -- -- -- --
```

This shows what things are on the I2C bus: 0x18 (accelerometers), 0x1e (forget?), and 0x69 (gyros).

Next, get Adafruit's python code:

`git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git`

1.7.5 BitTorrent Sync

Use BitTorrent's Sync program to keep files on different computers up to date. You need to add the repository for `apt-get` to get the package from, first create the file.

```
sudo nano /etc/apt/sources.list.d/btsync.list
```

Next, copy these lines into it:

```
deb http://debian.yeasoft.net/btsync wheezy main contrib non-free
deb-src http://debian.yeasoft.net/btsync wheezy main contrib non-free
```

Next, import the signing key:

```
sudo gpg --keyserver pgp.mit.edu --recv-keys 6BF18B15
sudo gpg --armor --export 6BF18B15 | sudo apt-key add -
```

Finally, update and install:

```
sudo apt-get update
sudo apt-get install btsync
```

Uninstall

To uninstall everything do:

```
sudo apt-get --purge remove btsync btsync-common
```

If you just do `btsync` then it will leave stuff behind.

Configure Sync

A configuration window will pop up during install, just take the defaults. Please **do not** select https or set a password for the web gui!!! You can re-run the installer anytime by:

```
sudo dpkg-reconfigure btsync
```

The check the `status`, `start`, or `stop` with:

```
sudo service btsync status
```

Finally, login into the webui at `computer_name:8888` to configure `btsync`.

1.7.6 Linux Kernel

Determine Kernel version and upgrade

You can determine the current linux kernel version by:

```
[kevin@raspberrypi tmp]$ more /proc/version
Linux version 3.2.27+ (dc4@dc4-arm-01) (gcc version 4.7.2 20120731 (prerelease)
(crosstool-NG linaro-1.13.1+bzr2458 - Linaro GCC 2012.08) ) #250 PREEMPT Thu Oct
 18 19:03:02 BST 2012
```

or

```
[kevin@raspberrypi tmp]$ uname -a
Linux raspberrypi 3.2.27+ #250 PREEMPT Thu Oct 18 19:03:02 BST 2012 armv6l GNU/Linux
```

In both cases, this is kernel version 3.2.27. Note that the *l* in `armv6l` refers to little endian.

Get and install rpi-update:

```
sudo apt-get install rpi-update
```

Now you can use rpi-update to get the current kernel:

```
sudo apt-get upgrade rpi-update
sudo rpi-update
```

1.7.7 Hardware

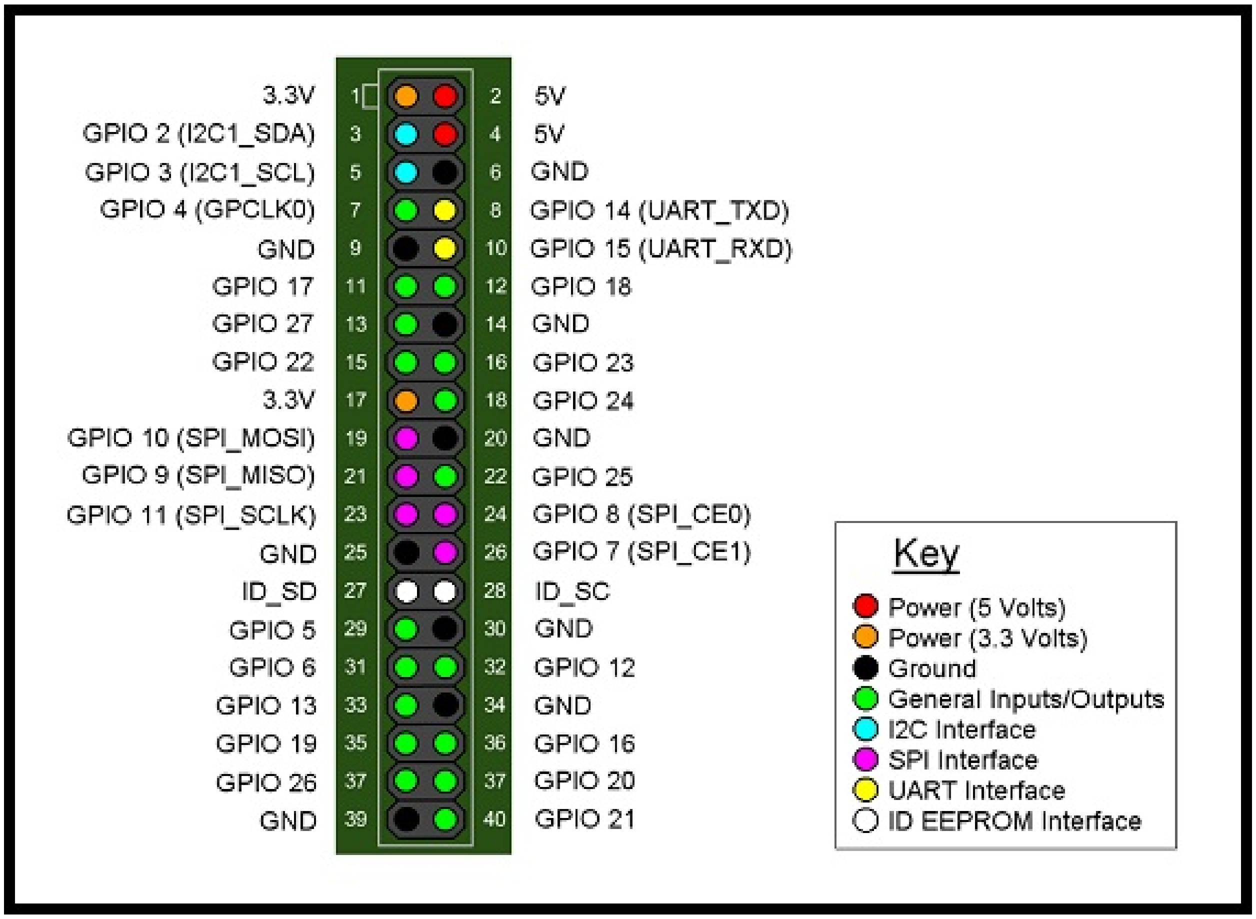
Pinouts

Depending on the version of the rpi you have, there are different pinouts for the different versions. A great resource is Pinout to figur out what pin is what.

Lights

The main indicators are the lights on the front corner of the board. These are:

```
OK (green): The board is active (blinks off when accessing the SD card)
PWR (red): The board is successfully powered from USB
FDX (green): Network is full-duplex
LNK (green): The network cable is connected (blinks off when transferring data to/from the network)
10M (yellow): Lit when the board is using a 100Mbps link, not lit when using a 10Mbps
```


3.3V
1
2
5V
GPIO 2 (I2C1_SDA)
3
4
5V
GPIO 3 (I2C1_SCL)
5
6
GND
GPIO 4 (GPCLK0)
7
8
GPIO 14 (UART_TXD)
GND
9
10
GPIO 15 (UART_RXD)
GPIO 17
11
12
GPIO 18
GPIO 27
13
14
GND
GPIO 22
15
16
GPIO 23
3.3V
17
18
GPIO 24
GPIO 10 (SPI_MOSI)
19
20
GND
GPIO 9 (SPI_MISO)
21
22
GPIO 25
GPIO 11 (SPI_SCLK)
23
24
GPIO 8 (SPI_CE0)
GND
25
26
GPIO 7 (SPI_CE1)
ID_SD
27
28
ID_SC
GPIO 5
29
30
GND
GPIO 6
31
32
GPIO 12
GPIO 13
33
34
GND
GPIO 19
35
36
GPIO 16
GPIO 26
37
37
GPIO 20
GND
39
40
GPIO 21
Key
Power (5 Volts)
Power (3.3 Volts)
Ground
General Inputs/Outputs
I2C Interface
SPI Interface
UART Interface
ID EEPROM Interface

i2c

First we need to load the drivers

```
sudo modprobe i2c-dev
sudo modprobe i2c-bcm2708
```

Now `/dev/i2c-0` and `/dev/i2c-1` should exist. Also, so see what is on the i2c bus, install the `i2c-tools` using:

```
sudo apt-get install i2c-tools
```

Now to explore the i2c bus try:

```
[kevin@raspberrypi ~]$ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

To have these load at boot, add them to `/etc/modules`.

USB Camera

To use the Logitech C270 camera you need to add your user (pi in this case) to the video group:

```
sudo usermod -a -G video pi
```

For other users, just change pi to the correct username. Then make sure the driver is loaded:

```
sudo modprobe uvcvideo
```

You can double check it works by grabbing an image:

```
sudo apt-get install fswebcam

fswebcam image.jpg
```

If an image appeared, then all is good.

1.7.8 Computer Monitoring

Program	Usefulness	Description
iftop	great	Monitor inbound and outbound IP traffic of the host computer
netstat	good	Monitor incoming/outgoing network packets, `netstat -a`
lsof	low	List of open files (e.g., disk files, network sockets, pipes, devices, and processes) and processes
htop	great	Like `top` but much more user friendly
iptraf	good	Network monitoring
btscanner	low	Bluetooth scanner
darkstat	great	Network traffic analyzer with web interface, `darkstat -i en0 -p 8080`, point browser to localhost:8080 to see results
nmap	good	Network mapper
p0f	low	Passive OS finger printing tool
wireshark	great	Network traffic analyzer

Other useful tools for your network.

Tools

Install:

```
    sudo apt-get nmap
sudo apt-get install tshark libcap-dev snmp-mibs-downloader
pip install scapy pcapy
```

1.7.9 Node.js

Installing Node.js

The version in apt-get is old (at the time of this writing), http://node-arm.herokuapp.com/ do:

```
wget http://node-arm.herokuapp.com/node_latest_armhf.deb
sudo dpkg -i node_latest_armhf.deb
# Check installation
node -v
npm -v
```

Packages

Nodejs uses Node Package Manager (npm) for add/removing packages. The best way is to build a package.json file in your project and run `npm install` to get what you need. See these **‘tutorials<https://docs.npmjs.com/>‘__** for more info.

1.7.10 External USB Drive

A goog resource is here

Setup

First connect the drive and find it:

```
pi@calculon ~ $ sudo fdisk -l

Disk /dev/mmcblk0: 15.9 GB, 15931539456 bytes
4 heads, 16 sectors/track, 486192 cylinders, total 31116288 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000981cb

        Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1            8192      122879       57344    c  W95 FAT32 (LBA)
/dev/mmcblk0p2          122880    31116287    15496704   83  Linux

WARNING: GPT (GUID Partition Table) detected on '/dev/sda'! The util fdisk doesn't support GPT. Use G


Disk /dev/sda: 640.1 GB, 640135028736 bytes
255 heads, 63 sectors/track, 77825 cylinders, total 1250263728 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1  1250263727   625131863+  ee  GPT
```

Make a mount point and mount and/or unmount the drive.

```
pi@calculon ~ $ sudo mkdir /mnt/usbdrive
pi@calculon ~ $ sudo mount /dev/sda1 /mnt/usbdrive
pi@calculon ~ $ sudo umount /dev/sda1
```

Formatting Disk

Starting with a USB hard drive 640GB formated in OSX’s HSF+, I needed to change it to ext4:

```
pi@calculon ~ $ sudo parted /dev/sda
GNU Parted 2.3
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) p
```

```
Model: Maxtor OneTouch (scsi)
Disk /dev/sda: 640GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number  Start  End  Size  File system  Name  Flags

(parted) mkpart primary ext4 1 640000
(parted) p
Model: Maxtor OneTouch (scsi)
Disk /dev/sda: 640GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number  Start   End    Size   File system  Name     Flags
 1      1049kB  640GB  640GB               primary

(parted) quit
Information: You may need to update /etc/fstab.
```

Note in the `mkpart` line, the start and stop locations are in MB.

```
pi@calculon ~ $ sudo mkfs.ext4 /dev/sda1 -L Calculon
mke2fs 1.42.5 (29-Jul-2012)
Filesystem label=Calculon
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
39067648 inodes, 156249856 blocks
7812492 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=0
4769 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872, 71663616, 78675968,
    102400000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

Now mount the drive and double check it:

```
pi@calculon ~ $ sudo mount /dev/sda1 /mnt/usbdrive
pi@calculon ~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs           15G  3.7G   11G  27% /
/dev/root        15G  3.7G   11G  27% /
devtmpfs        112M     0  112M   0% /dev
tmpfs            24M  240K   23M   2% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs            47M     0   47M   0% /run/shm
/dev/mmcblk0p1   56M  9.8M   47M  18% /boot
/dev/sda1       587G   70M  557G   1% /mnt/usbdrive
```

Fix permissions:

```
pi@calculon ~ $ sudo chown pi:pi /mnt/usbdrive
pi@calculon ~ $ sudo chmod 777 /mnt/usbdrive
pi@calculon ~ $ sudo ls /mnt/usbdrive -alh
total 24K
drwxrwxrwx 3 pi   pi   4.0K Dec 14 20:45 .
drwxr-xr-x 3 root root 4.0K Dec 14 18:24 ..
drwx------ 2 root root  16K Dec 14 20:45 lost+found
```

Automounting

Add the following to fstab so it mounts on boot.

```
sudo nano /etc/fstab
/dev/sda1 /mnt/usbdisk auto defaults,user 0 1
```

This sets the file system to `auto` and `user` enables write permissions for all users. The 0 is for debugging and 1 is for a file system check at boot. You can test this out by:

```
sudo mount -a
```

Swap Partition on Hard Drive

Don't ever make a swap partition on the sd card ... it is too slow and will reduce the card's life span.

1. Create a partition for swap on say `/dev/sda2` following the method above, then exit `parted`.
2. Use `mkswap /dev/sda2` to set it up.
3. Edit `/etc/fstab` and add the following line: `/dev/sda2 none swap sw 0 0`
4. Get rid of RPi's file base swap by removing the packages: `sudo apt-get remove dphy-swapfile`
5. Make sure swap is working: `swapon -s`

   pi@calculon ~ $ swapon -s Filename Type Size Used Priority /dev/sda2 partition 4295676 0 -1

Making available to OSX

Make the following changes to the config file, adding the drive to netatalk.

```
pi@calculon ~ $ sudo pico /etc/netatalk/AppleVolumes.default
# The line below sets some DEFAULT, starting with Netatalk 2.1.
:DEFAULT: options:upriv,usedots

# By default all users have access to their home directories.
~/            "Home Directory"
/mnt/usbdrive

# End of File
```

There are also options to enable Time Machine support (see tm).

Then restart `netatalk`:

```
pi@calculon ~ $ sudo /etc/init.d/netatalk stop
Stopping Netatalk Daemons: afpd cnid_metad papd timelord atalkd.
pi@calculon ~ $ sudo /etc/init.d/netatalk start
Starting Netatalk services (this will take a while):  cnid_metad afpd.
```

1.7.11 WiFi

D-Link wireless N 150 (DWA-121) Pico USB adaptor install.

```
[kevin@raspberrypi ~]$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 2001:3308 D-Link Corp. DWA-121 802.11n Wireless N 150 Pico Adapter [Realtek R
```

Note: If you don't see it, make sure it is the only USB device plugged in because it takes a lot of power. Otherwise attach to a powered USB hub and you should be fine.

WPA2

First create a file with the following information, but substitute in the correct ssid and psk (with quotes around them) for your network.

```
[kevin@raspberrypi ~]$ more /etc/wpa_supplicant/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
ap_scan=2

network={
    ssid="wireless access point name in quotes"
    key_mgmt=WPA-PSK
    proto=WPA2
    pairwise=CCMP TKIP
    group=CCMP TKIP
    psk="pass phrase in quotes"
}
```

Note: The ssid is case sensitive!!

Network Setup

Next you will need to change your network interface for a static IP to:

```
[kevin@raspberrypi ~]$ more /etc/network/interfaces
auto lo
iface lo inet loopback

# dynamic interface
#iface eth0 inet dhcp

# static interface
iface eth0 inet static
    address 192.168.1.120
    netmask 255.255.255.0
    gateway 192.168.1.1
```

```
auto wlan0
iface wlan0 inet static
    address 192.168.1.121
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
    netmask 255.255.255.0
    gateway 192.168.1.1
```

Note that the wifi interface (wlan0) points to the WPA config file from above. Also there is an example dynamic interface commented out (`iface eth0 inet dhcp`) to show you how to use DHCP. The `lo` is the loopback interface, eth0 is the wired interface, with the wlan0 being the wireless interface. Also, the lines with `auto` in them tell linux to automatically start those interfaces during the bootup process.

Or if you are fine with DHCP determining all your IP addresses:

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

Finally to get the wireless up and running, use `ifup` to get things started.

```
[kevin@raspberrypi ~]$ sudo ifup wlan0
ioctl[SIOCSIWAP]: Operation not permitted
ioctl[SIOCSIWENCODEEXT]: Invalid argument
ioctl[SIOCSIWENCODEEXT]: Invalid argument
```

These errors don't seem to effect the wifi adaptor. You can double check all is well by using `iwconfig`.

```
[kevin@raspberrypi ~]$ iwconfig
lo        no wireless extensions.

wlan0     IEEE 802.11bgn  ESSID:"GC9J2"  Nickname:"<WIFI@REALTEK>"
          Mode:Managed  Frequency:2.437 GHz  Access Point: 00:7F:28:05:4D:D9
          Bit Rate:150 Mb/s   Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=100/100  Signal level=76/100  Noise level=0/100
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

eth0      no wireless extensions.
```

Looking at the wlan0 interface, it has a 150 Mb/s data rate (802.11n), and sees a signal strength of 76/100.

```
[kevin@raspberrypi ~]$ ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr fc:75:16:04:96:5f
          inet addr:192.168.1.121  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:59222 errors:0 dropped:63403 overruns:0 frame:0
          TX packets:11365 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:92009000 (87.7 MiB)  TX bytes:1154992 (1.1 MiB)
```

Notice here a lot of dropped packets on the receive (RX).

1.7.12 Software

Debian Packaged Software

Why have one program that does a few common sense things well when you can have multiple programs that do one or two things really badly!

Program	Description
apt-get	install: `apt-get install <prgm>`
	remove: `apt-get remove <prgm>` add `--purge` to remove configs
	upgrade: `apt-get upgrade <prgm>` to get latest
	update: `apt-get update` updates package databases
apt-cache	search for programs you can install: `apt-cache search <prgm>`
	info: `apt-cache showpkg [packagename]` displays package info
dpkg	list programs you have installed: `dpkg -l`

Updates, Search, and List

`apt-get` is a horrible program and a beautiful example of how not to design software. So if you want to know what packages are outdated, then you need to install this package:

```
sudo apt-get install apt-show-versions
```

Now to figure out what is outdated, do:

```
apt-show-versions -u
```

Now some packages will get `kept back` which seems to be some strange apt-get issue. To update your system completely, do:

```
sudo apt-get dist-upgrade
```

Or list all packages installed on the computer by:

```
dpkg -l
```

Raspbian

There is a lot of junk automatically installed on Raspbian, use `dpkg -l` to see. Suggest removal via `sudo apt-get remove <pkg>`:

- isc-dhcp-server: Already have one on my network, don’t need another running (it is on by default)
- sonic-pi: a music programming environment aimed at new programmers
- printer-driver-*: don’t print anything
- hplip*: HP printing stuff
- cups cups-bsd cups-client cups-common cups-filters cups-ppdc: printing stuff
- supercollider*: real-time audio synthesis programming language
- samba-common: Windoze stuff
- sane-utils: scanner stuff
- penguinspuzzle: game

- ghostscript

1.7.13 SD Card

Backup and Restore

Use the `dd` command to make a full backup of the image:

```
dd if=/dev/sdx of=/path/to/image
```

or for compression:

```
dd if=/dev/sdx | gzip > /path/to/image.gz
```

Where sdx is your SD card and the target could be ~/raspbian_wheezy_`date "+%Y%m%d_%T"`. This will save it to your home directory and append the current date and time on the end of the filename.

To restore the backup you reverse the commands:

```
dd if=/path/to/image of=/dev/sdx
```

or when compressed:

```
gzip -dc /path/to/image.gz | dd of=/dev/sdx
```

1.7.14 Services

To setup daemons to run at boot time or be able to easily start/stop them, we need to the init system. Create a file in `/etc/init.d` like the one below:

```
# /etc/init.d/netscan
#

# Some things that run always
DAEMON_USER=root
DIR=/home/pi/github/netscan
DAEMON_NAME=netscan
SERVER_NAME=simple_server
DAEMON=$DIR/$DAEMON_NAME.py
SERVER=$DIR/$SERVER_NAME.py
PIDFILE=/var/run/$DAEMON_NAME.pid
SERVER_PIDFILE=/var/run/$SERVER_NAME.pid

. /lib/lsb/init-functions

# Carry out specific functions when asked to by the system
case "$1" in
  start)
    echo "Starting netscan"
    log_daemon_msg "Starting system $DAEMON_NAME daemon"
    start-stop-daemon --start --background --pidfile $PIDFILE --make-pidfile --u
ser $DAEMON_USER --chuid $DAEMON_USER --startas $DAEMON
    start-stop-daemon --start --background --pidfile $SERVER_PIDFILE --make-pidf
ile --user pi --chuid pi --startas $SERVER
    log_end_msg $?
    ;;
  stop)
```

```
    log_daemon_msg "Stopping system $DAEMON_NAME daemon"
    start-stop-daemon --stop --pidfile $PIDFILE --retry 10
    start-stop-daemon --stop --pidfile $SERVER_PIDFILE --retry 10
    log_end_msg $?
    ;;
  status)
    status_of_proc $SERVER_NAME $SERVER && status_of_proc $DAEMON_NAME $DAEMON &
& exit 0 || exit $?
    ;;
  *)
    echo "Usage: /etc/init.d/netscan {start|status|stop}"
    exit 1
    ;;
esac

exit 0
```

Another example:

```
# /etc/init.d/nodesjs
#

# Some things that run always
DAEMON_USER=root
DIR=/usr/local/bin
DAEMON_NAME=http-server
DAEMON=$DIR/$DAEMON_NAME
PIDFILE=/var/run/$DAEMON_NAME.pid
DAEMON_full="$DAEMON -- /mnt/usbdrive -p 9000 -s"

. /lib/lsb/init-functions

# Carry out specific functions when asked to by the system
case "$1" in
  start)
        echo "Starting Nodejs HTTP Server for movies"
        echo $DAEMON_full
        log_daemon_msg "Starting system $DAEMON_NAME daemon"
        start-stop-daemon --start --background --pidfile $PIDFILE --make-pidfile --user $DAEMON_USER
        log_end_msg $?
        ;;
  stop)
        log_daemon_msg "Stopping system $DAEMON_NAME daemon"
        start-stop-daemon --stop --pidfile $PIDFILE --retry 10
        log_end_msg $?
        ;;
  status)
        status_of_proc status_of_proc $DAEMON_NAME $DAEMON && exit 0 || exit $?
        ;;
  *)
        echo "Usage: /etc/init.d/nodejs-movies {start|status|stop}"
        exit 1
        ;;
esac

exit 0
```

Change the permissions with:

```
chmod 755 /etc/init.d/netscan
```

Add the service to the proper run levels:

```
update-rc.d netscan defaults
```

References

- thegeekstuff
- debian-administration.org

1.8 Engineering

1.8.1 Statistics

Expected Value

The predicted value of a variable, calculated as the sum of all possible values (x_i) each multiplied by the probability (p_i) of its occurrence.

$$E(x) = x_1p_1 + x_2p_2 + ... + x_np_n$$
$$E(x) = \mu$$

The expected value is the mean of a random variable (greek letter μ).

Example

You pay $2 to roll a six sided dice. If you roll a 6, you get $10. The expected value will tell you how much money you should make in the long run.

$$E(x) = -2\frac{5}{6} + (10-2)\frac{1}{6} = -0.33333334$$

So, 5 of the 6 faces is a loosing roll, so you loose $2. You have to pay $2 to play, so your next win is only $8 if you roll a 6. Therefore you expect to loose 33 cents in the long run if you play this game … house always wins!

Variance

Variance describes how far a set of numbers are spread out from each other from the mean. It is the average of the *squared* differences from the mean. It is also the covariance of x with it self.

$$var(x) = E[(x-\mu)^2] = (x_1-\mu)^2p_1 + ... + (x_n-\mu)^2p_n = cov(x,x) = E[(x-\mu)(x-\mu)]$$

Variance is also one of the moments of a distribution.

Example

A perfect die has an expected value and variance of:

$$E(x) = (1 + 2 + 3 + 4 + 5 + 6)\frac{1}{6} = 3.5$$

$$var(x) = [(1 - 3.5)^2 + (2 - 3.5)^2 + ...(6 - 3.5)^2]\frac{1}{6} = 2.9167$$

Note: The 1/6 above is the probability assuming a fair 6 sided dice.

Covariance

The measure of how two random variables change together.

$$cov(x, y) = E[(x - \mu)(y - \mu)]$$

For random variables which show similar behavior the sign of the cov is positive. If the random variables act differently, then the sign is negative. The magnitude is hard to interpret unless the cov is normalized, called the correlation coefficient.

Standard Deviation

The standard dev is the square root of the variance.

$$std = \sqrt{(}E[(x - \mu)^2])$$

Example

The standard dev of the above dice example is:

```
std(x) = 1.7078
```

Example: Octave

```
x = [ 1 2 3 4 5 6 ];
var(x,1) = 2.9167
std(x,1) = 1.7078
```

Note: The option 1 is passed because the variance and std are taken of the complete set of 6 (N=1). The complete set is also called the *Population*. The default is that this is a sample of a much larger distribution, so this is just 6 samples. When this is just a sample set, the devisor is now N-1 (i.e., N=5 for this example) and the results are different.

Example: Python

```
from numpy import *
x = array([1,2,3,4,5,6])
std(x) = 1.7078
var(x) = 2.9167
```

References

1. http://www.mathsisfun.com/data/index.html

1.8.2 Subnetting

IP subnetting made easy by George Ou has a good overview, here is a summary.

The IP address is a 32 bit number which provides 0 to 4,294,967,295 unique addresses on a network. This is chopped up into 4 8-bit octets separated by a period.

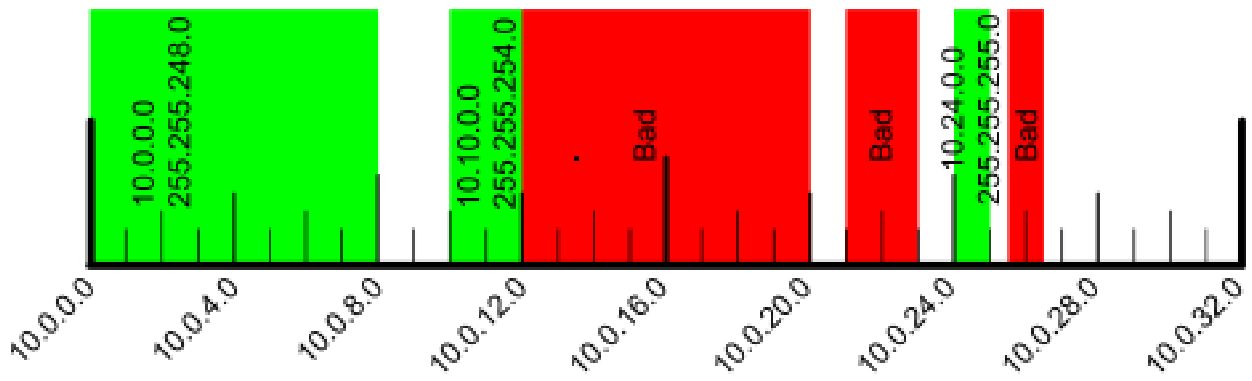
0.0.0.0 to 0.0.0.255 is one octet which provide 256 addresses.

Subnetwork

Chopping a large network up into smaller pieces is subnetworking.

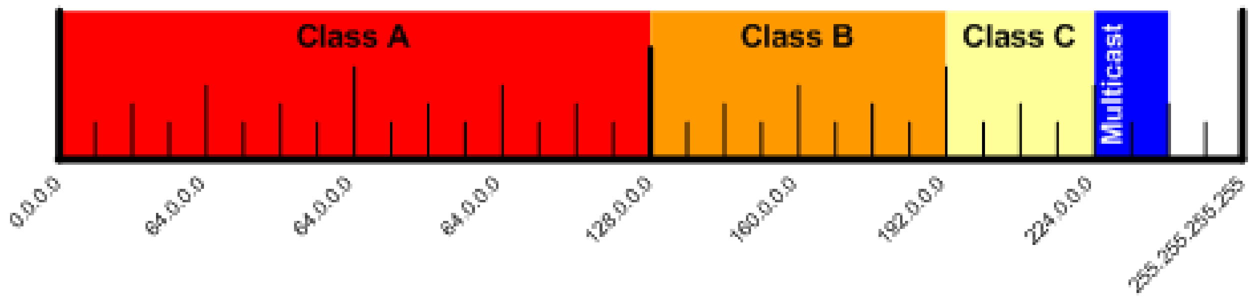
- beginning is always even, network id
- ending is always odd, broadcast id
- you can not use the network or broadcast id address, since they have special meaning
- 0.0.0.0/8 (0.0.0.0 to 0.255.255.255) not used
- 127.0.0.0/8 (127.0.0.0 to 127.255.255.255) used for loop back addresses
- Number of hosts: 2^bits - 2 where bits is the number of bits for the subnet and the -2 accounts for both the network and broadcast id's

You need to chop along clean binary division as shown below:



Network Classes

Class	IP Range	Subnet Bits	Mask Bits	Notes
A	0.0.0.0 to 127.255.255.255	24	8	the first half of the address space
B	128.0.0.0 to 191.255.255.255	16	16	half the remaining address space
C	192.0.0.0 to 223.255.255.255	8	24	half the remaining address space
D	224.0.0.0 to 239.255.255.255	undefined	undefined	half the remaining address space for multicast
E	240.0.0.0 to 255.255.255.255	undefined	undefined	everything remaining



Private Subnetworks

Class	Subnet Bits	Mask Bits	IP	IP Range	Number of Hosts
A	24	8	10.0.0.0/8	10.0.0.0 to 10.255.255.255	16,777,216
B	20	12	172.16.0.0/12	172.16.0.0 to 172.31.255.255	1,048,576
C	16	16	192.168.0.0/16	192.168.0.0 to 192.168.255.255	65,536
C	16	16	169.254.0.0/16	169.254.0.0 to 169.254.255.255	65,536

The 169.254.0.0 addresses are only used when DHCP server is not available.

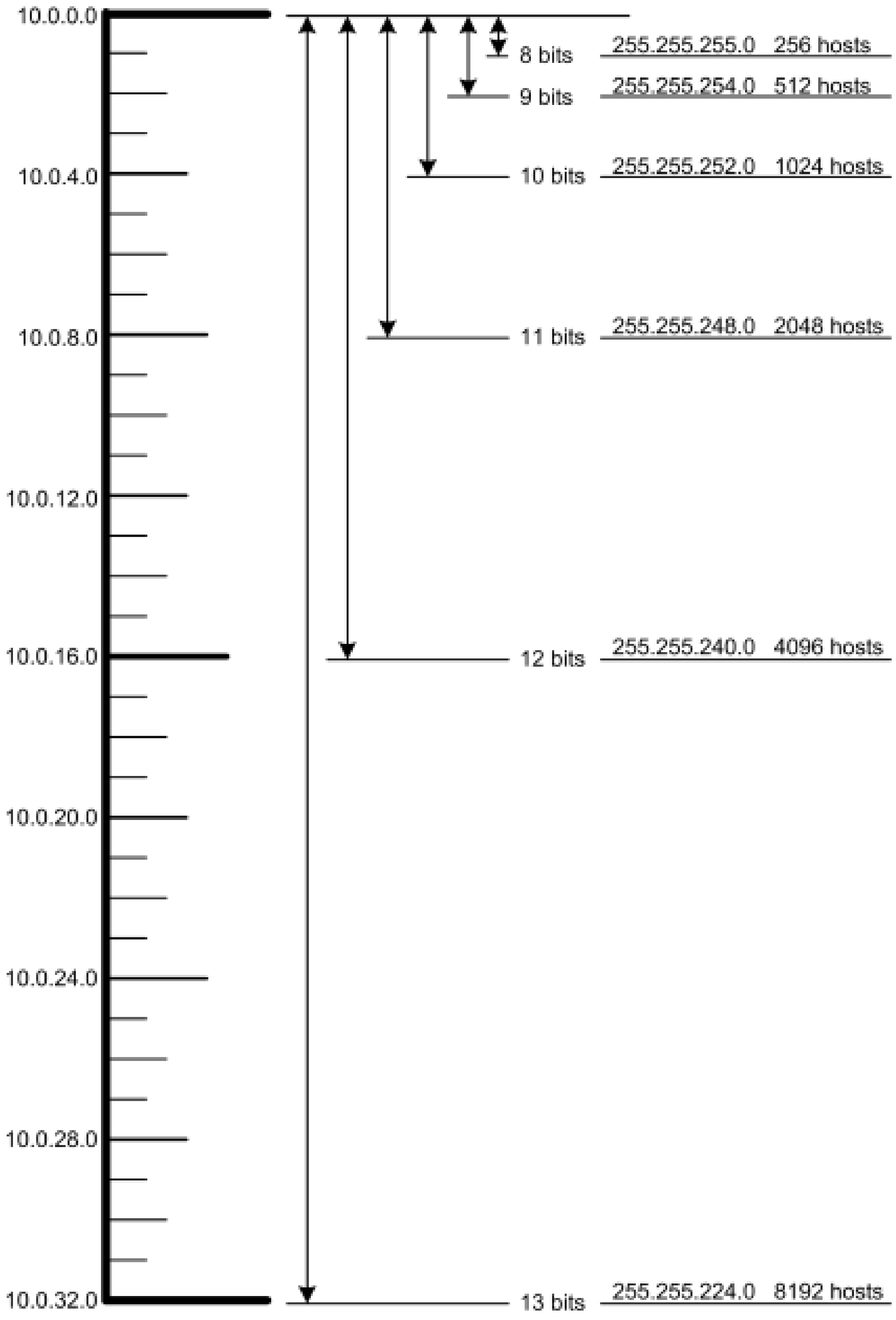
Address Bits

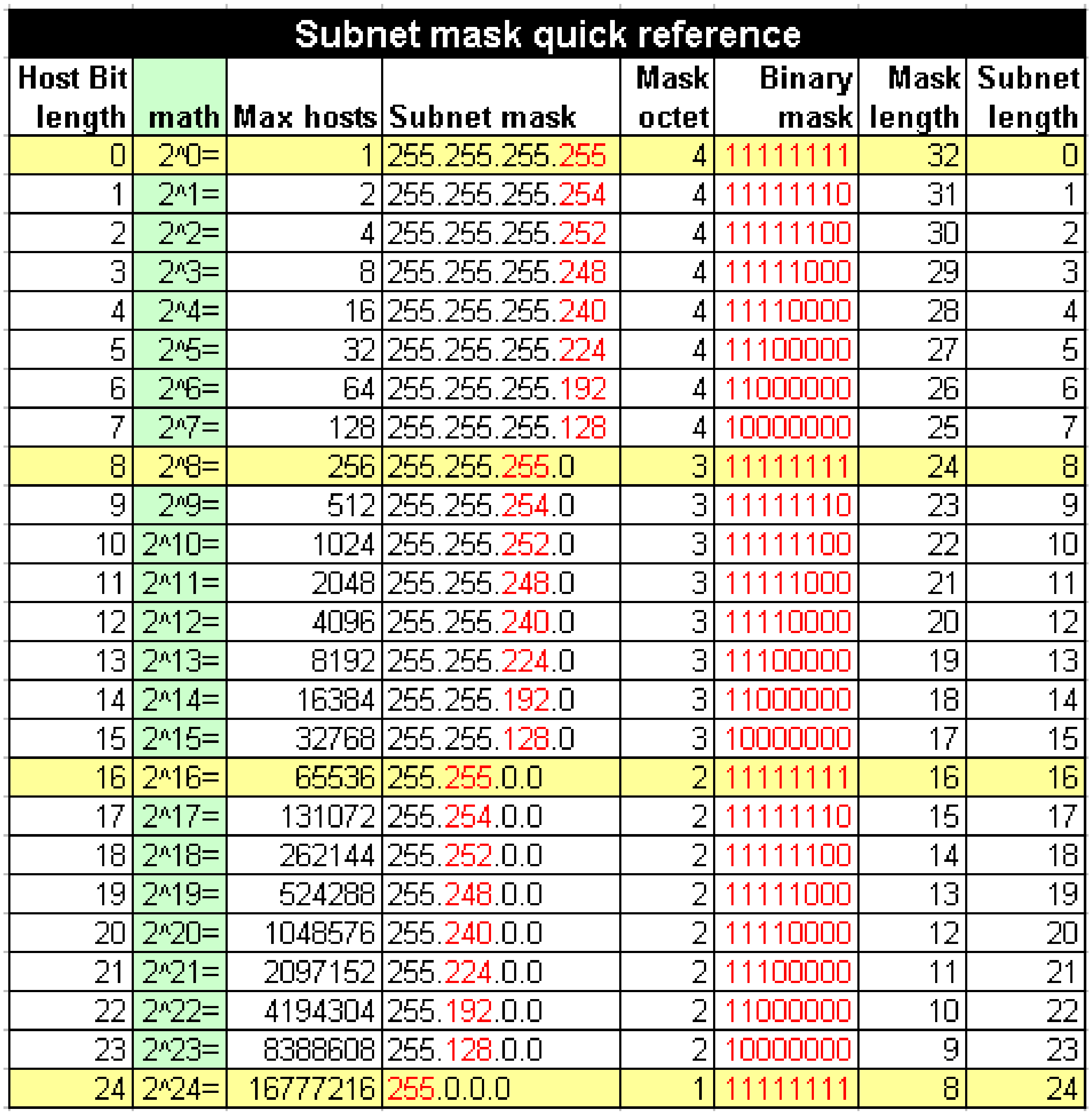
- 8 bits = 256 addresses (254 hosts, minus the network and broadcast id's)
- 9 bits = 512 addresses
- 10 bits = 1024 addresses
- 11 bits = 2048 addresses

Masks

Typical home networks use 192.168.0.0/16 where the 16 indicates a 16-bit mask giving addresses from 192.168.0.0 to 192.168.255.255. Since a complete mask is 32-bits and each octet is 8-bits each, 16-bits uses the bottom 2 octets or 192.168.x.x. An 8-bit mask would use the lower 3 octets or 192.x.x.x. A 24-bit mask would only use the bottom octet of the address range, or 192.168.1.x.

- 8-bit mask: 255.0.0.0
- 16-bit mask: 255.255.0.0


10.0.0.0
10.0.4.0
10.0.8.0
10.0.12.0
10.0.16.0
10.0.20.0
10.0.24.0
10.0.28.0
10.0.32.0
8 bits
9 bits
10 bits
11 bits
12 bits
13 bits
255.255.255.0 256 hosts
255.255.254.0 512 hosts
255.255.252.0 1024 hosts
255.255.248.0 2048 hosts
255.255.240.0 4096 hosts
255.255.224.0 8192 hosts



Subnet mask quick reference

Host Bit length	math	Max hosts	Subnet mask	Mask octet	Binary mask	Mask length	Subnet length
0	2^0=	1	255.255.255.255	4	11111111	32	0
1	2^1=	2	255.255.255.254	4	11111110	31	1
2	2^2=	4	255.255.255.252	4	11111100	30	2
3	2^3=	8	255.255.255.248	4	11111000	29	3
4	2^4=	16	255.255.255.240	4	11110000	28	4
5	2^5=	32	255.255.255.224	4	11100000	27	5
6	2^6=	64	255.255.255.192	4	11000000	26	6
7	2^7=	128	255.255.255.128	4	10000000	25	7
8	2^8=	256	255.255.255.0	3	11111111	24	8
9	2^9=	512	255.255.254.0	3	11111110	23	9
10	2^10=	1024	255.255.252.0	3	11111100	22	10
11	2^11=	2048	255.255.248.0	3	11111000	21	11
12	2^12=	4096	255.255.240.0	3	11110000	20	12
13	2^13=	8192	255.255.224.0	3	11100000	19	13
14	2^14=	16384	255.255.192.0	3	11000000	18	14
15	2^15=	32768	255.255.128.0	3	10000000	17	15
16	2^16=	65536	255.255.0.0	2	11111111	16	16
17	2^17=	131072	255.254.0.0	2	11111110	15	17
18	2^18=	262144	255.252.0.0	2	11111100	14	18
19	2^19=	524288	255.248.0.0	2	11111000	13	19
20	2^20=	1048576	255.240.0.0	2	11110000	12	20
21	2^21=	2097152	255.224.0.0	2	11100000	11	21
22	2^22=	4194304	255.192.0.0	2	11000000	10	22
23	2^23=	8388608	255.128.0.0	2	10000000	9	23
24	2^24=	16777216	255.0.0.0	1	11111111	8	24

- 24-bit mask: 255.255.255.0

Calculator

IP address calculator Another one

```
192.168.0.1/16:
Address:   192.168.0.1          11000000.10101000 .00000000.00000001
Netmask:   255.255.0.0 = 16     11111111.11111111 .00000000.00000000
Wildcard:  0.0.255.255          00000000.00000000 .11111111.11111111
=>
Network:   192.168.0.0/16       11000000.10101000 .00000000.00000000 (Class C)
Broadcast: 192.168.255.255      11000000.10101000 .11111111.11111111
HostMin:   192.168.0.1          11000000.10101000 .00000000.00000001
HostMax:   192.168.255.254      11000000.10101000 .11111111.11111110
Hosts/Net: 65534                (Private Internet)

192.168.1.1/24:
Address:   192.168.1.1          11000000.10101000.00000001 .00000001
Netmask:   255.255.255.0 = 24   11111111.11111111.11111111 .00000000
Wildcard:  0.0.0.255            00000000.00000000.00000000 .11111111
=>
Network:   192.168.1.0/24       11000000.10101000.00000001 .00000000 (Class C)
Broadcast: 192.168.1.255        11000000.10101000.00000001 .11111111
HostMin:   192.168.1.1          11000000.10101000.00000001 .00000001
HostMax:   192.168.1.254        11000000.10101000.00000001 .11111110
Hosts/Net: 254                  (Private Internet)
```

1.8.3 Sequences and Series

Sequence

A **sequence** is an ordered list of numbers (e.g., a_n); the numbers are called “elements” or “terms”. Every convergent sequence is bounded, thus an unbounded sequence is divergent.

Sequence Test	Converge	Notes
Squeeze Theorem	$\lim_{n\to\infty} a_n = \lim_{n\to\infty} c_n = L$ then $\lim_{n\to\infty} b_n = L$	$a_n \le b_n \le c_n$
Def 1, pg 692	$\lim_{n\to\infty} a_n = L$	
l'Hospital's Rule	$\lim_{n\to\infty} \frac{f(x)}{g(x)} \Rightarrow \lim_{n\to\infty} \frac{f'(x)}{g'(x)}$	where $f(x)$ = numerator and $g(x)$ = denominator
Theorem 3, pg 693	if $f(n) = a_n$ then $\lim_{n\to\infty} f(x) = L$	
Theorem 6, pg 694	$\lim_{n\to\infty} \lvert a_n\rvert = 0$ then a_n converges	
Theorem 9, pg 696	$\lim_{n\to\infty} r^n = \begin{cases} 0, & \text{if } -1 < r < 1 \\ 1, & \text{if } r = 1 \end{cases}$	Divergent for all other values of r
Theorem 12, pg 698	Every bounded ($m \le a_n \le M$), monotonic sequence is convergent	The bounds exists for $n \ge 1$, also see Theorem 10 and 11

Series

A **series** is the sum of the terms of a sequence: $\sum\limits_{n=1}^{\infty} a_n$.

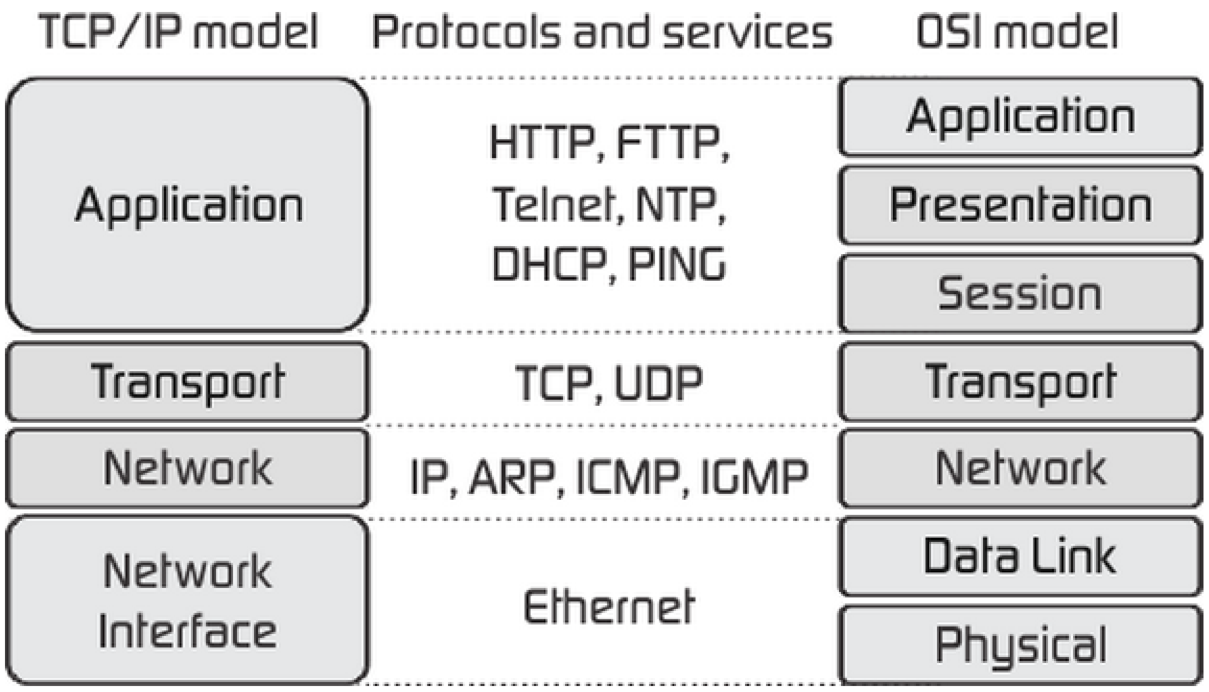
Series Test	Converge	Diverge	Notes
Divergence	N/A	$\lim\limits_{n\to\infty} a_n \neq 0$	Doesn't show convergence and the converse is not true
Integral	if $\int\limits_1^{\infty} f(x)dx$ converges	if $\int\limits_1^{\infty} f(x)dx$ diverges	$f(x)$ must be positive, decreasing, and continous, also $f(n) = a_n$ for all n
Root	$\lim\limits_{n\to\infty} \sqrt[n]{\lvert a_n\rvert} = L < 1$	$\lim\limits_{n\to\infty} \sqrt[n]{\lvert a_n\rvert} = L > 1$ or ∞	inconclusive if $L = 1$
Ratio	$\lim\limits_{n\to\infty} \left\lvert \frac{a_{n+1}}{a_n} \right\rvert = L < 1$	$\lim\limits_{n\to\infty} \left\lvert \frac{a_{n+1}}{a_n} \right\rvert = L > 1$ or ∞	inconclusive if $L = 1$
Direct Comparison	$0 \leq a_n \leq b_n$ for all n and $\sum\limits_{n=1}^{\infty} b_n$ converges	$0 \leq b_n \leq a_n$ for all n and $\sum\limits_{n=1}^{\infty} b_n$ diverges	$a_n, b_n > 0$
Limit Comparison	$\lim\limits_{n\to\infty} \frac{a_n}{b_n} = L$ and $\sum\limits_{n=1}^{\infty} b_n$ converges	$\lim\limits_{n\to\infty} \frac{a_n}{b_n} = L$ and $\sum\limits_{n=1}^{\infty} b_n$ diverges	$a_n, b_n > 0$ and L is a positive constant, if L is ∞ or 0, then pick a different b_n
Absolute	$\sum\limits_{n=1}^{\infty} \lvert a_n\rvert = 0$		Definition of absolutely convergent, the sum is independent of the order in which the terms are summed
Conditional	$\sum\limits_{n=1}^{\infty} \lvert a_n\rvert$ diverges but $\sum\limits_{n=1}^{\infty} a_n$ converges		The sum is dependent of the order in which the terms are summed

Common Series

Series Test	Formula	Converge	Diverge	Notes
Alternating	$\sum_{n=1}^{\infty}(-1)^{n-1}a_n$	$0 < a_{n+1} \leq a_n$ for all n and $\lim_{n\to\infty} a_n = 0$	N/A	
Geometric	$\sum_{n=1}^{\infty} ar^{n-1}$	$\|r\| < 1$ and converges to $\frac{a}{1-r}$	$\|r\| \geq 1$	finite sum of the first n terms: $= \frac{a(1-r^n)}{1-r}$
P-Series	$\sum_{n=1}^{\infty} \frac{1}{n^p}$	$p > 1$	$p \leq 1$	cannot calculate sum
Power	$\sum_{n=0}^{\infty} c_n(x-a)^n$	i, converge if $x = a$ ii, converge for all x iii, converge if $\|x-a\| < R$		R is the radius of convergence, you need to check the end points for convergence too. Typically use Ratio Test.
Taylor	$\sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n$	$\|x - a\| <$	$\|x - a\| >$	Taylor

1.8.4 dpkt Cheatsheet

dpkt is a python library for manipulating packets and although it is a good library it is very poorly documented.



Useful Ports

TCP

TCP Port	Service
21	IRC
22	SSH
25	STMP
80	Http
123	Network Time Server
443	Https
445	SMB
548	Apple File Protocol (AFP) over TCP
3689	iTunes using the iTunes Library Sharing feature
5009	Airport admin utility
9100	HP Jet Direct
10000	??

UDP

UDP Port	Service
67,68	DHCP
123	NTP
5353	mDNS, Bonjour
17500	Dropbox

Packet Types

There are many types of packets supported and several ways to test what they are:

```
if isinstance(packet,dpkt.ethernet.ETH_TYPE_IP)
if type(packet) == dpkt.ethernet.ETH_TYPE_IP
if packet.p == dpkt.ethernet.ETH_TYPE_IP
```

dpkt.tcp.* dpkt.udp.* dpkt.icmp.*

dpkt.ethernet.*	Val	dpkt.ip.*	Val
ETH_TYPE_ARP		IP_PROTO_ICMP	1
ETH_TYPE_IP	2048	IP_PROTO_TCP	6
ETH_TYPE_IP6	34525	IP_PROTO_UDP ICMP6 SCTP	17 58 132

Ethernet (Physical & Datalink Layers)

Ethernet is a family of computer networking technologies for local area networks (LANs).

```
ether = dpkt.ethernet.Ethernet(data)
mac = ...
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Ethernet destination address (first 32 bits)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Ethernet dest (last 16 bits)  |Ethernet source (first 16 bits)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Ethernet source address (last 32 bits)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Type code              |                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Addresses	Description
224.0.0.251	Multicast DNS (mDNS)

IP (Internet Protocol)

The Internet Protocol (IP) is the principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries. Its routing function enables internetworking, and essentially establishes the Internet.

```
ip = dpkt.ip.IP(data)
ip = ether.data
ip_addr = socket.inet_ntoa(ip.src|ip.dst)
```

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IPv6

:: $ host -t AAAA www.cyberciti.biz www.cyberciti.biz has IPv6 address 2607:f0d0:1002:51::4

$ ping6 2607:f0d0:1002:51::4 PING 2607:f0d0:1002:51::4(2607:f0d0:1002:51::4) 56 data bytes 64 bytes from 2607:f0d0:1002:51::4: icmp_seq=1 ttl=64 time=0.056 ms 64 bytes from 2607:f0d0:1002:51::4: icmp_seq=2 ttl=64 time=0.027 ms 64 bytes from 2607:f0d0:1002:51::4: icmp_seq=3 ttl=64 time=0.021 ms 64 bytes from 2607:f0d0:1002:51::4: icmp_seq=4 ttl=64 time=0.023 ms

^C

dpkt.ip6.IP6

::

class IP6(dpkt.Packet):

__hdr__ = (('_v_fc_flow', 'I', 0x60000000L), ('plen', 'H', 0), # payload length (not including header) ('nxt', 'B', 0), # next header protocol ('hlim', 'B', 0), # hop limit ('src', '16s', ''), ('dst', '16s', '')

)

nxt Next header type, typical values are 6 for TCP, 17 for UDP, 58 for ICMPv6, 132 for

SCTP.

```
socket.inet_ntop(AF_INET6, ip.dst)
socket.inet_pton(socket.AF_INET6, "2001:1938:26f:1:204:4bff:0:1")

ip = eth.data
if eth.type == dpkt.ethernet.ETH_TYPE_IP6 and ip.nxt == dpkt.ip.IP_PROTO_UDP:
```

UDP (User Datagram Protocol)

UDP uses a simple connectionless transmission model with a minimum of protocol mechanism. It has no handshaking dialogues, and thus exposes any unreliability of the underlying network protocol to the user's program. There is no guarantee of delivery, ordering, or duplicate protection. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram.

```
0        7 8     15 16    23 24    31
 +--------+--------+--------+--------+
 |     Source      |   Destination   |
 |      Port       |      Port       |
 +--------+--------+--------+--------+
 |                 |                 |
 |     Length      |    Checksum     |
 +--------+--------+--------+--------+
 |
 |          data octets ...
 +---------------- ...
```

TCP (Transmission Control Protocol)

The Transmission Control Protocol (TCP) is a core protocol of the Internet Protocol Suite. TCP provides reliable, ordered, and error-checked delivery of a stream of octets between applications running on hosts communicating over an IP network.

```
tcp = ip.data
port = tcp.sport|dport
```

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TCP Flags

```
fin_flag = ( tcp.flags & dpkt.tcp.TH_FIN ) != 0
syn_flag = ( tcp.flags & dpkt.tcp.TH_SYN ) != 0
rst_flag = ( tcp.flags & dpkt.tcp.TH_RST ) != 0
psh_flag = ( tcp.flags & dpkt.tcp.TH_PUSH) != 0
ack_flag = ( tcp.flags & dpkt.tcp.TH_ACK ) != 0
urg_flag = ( tcp.flags & dpkt.tcp.TH_URG ) != 0
ece_flag = ( tcp.flags & dpkt.tcp.TH_ECE ) != 0
cwr_flag = ( tcp.flags & dpkt.tcp.TH_CWR ) != 0
```

DNS (Domain Name System)

The Domain Name System (DNS) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. An often-used analogy to explain the Domain Name System is that it serves as the phone book for the Internet by translating human-friendly computer hostnames into IP addresses. For example, the domain name www.example.com translates to the addresses 93.184.216.119 (IPv4) and 2606:2800:220:6d:26bf:1447:1097:aa7 (IPv6).

A simple example of parsing a DNS packet

```
eth = dpkt.ethernet.Ethernet(buf)
ip = eth.data
udp = ip.data
# make the dns object out of the udp data and check for it being a RR (answer)
# and for opcode QUERY (I know, counter-intuitive)
if udp.dport != 53 and udp.sport != 53: continue
dns = dpkt.dns.DNS(udp.data)
if dns.qr != dpkt.dns.DNS_R: continue
if dns.opcode != dpkt.dns.DNS_QUERY: continue
if dns.rcode != dpkt.dns.DNS_RCODE_NOERR: continue
if len(dns.an) < 1: continue
# now we're going to process and spit out responses based on record type
# ref: http://en.wikipedia.org/wiki/List_of_DNS_record_types
for answer in dns.an:
        if answer.type == dpkt.dns.DNS_CNAME:
                print "CNAME request", answer.name, "\tresponse", answer.cname
        elif answer.type == dpkt.dns.DNS_A:
                print "A request", answer.name, "\tresponse", socket.inet_ntoa(answer.rdata)
        elif answer.type == dpkt.dns.DNS_PTR:
                print "PTR request", answer.name, "\tresponse", answer.ptrname
```

Make a DNS packet

```
dns = dpkt.dns.DNS(udp.data)
dns.op = dpkt.dns.DNS_RA
dns.rcode = dpkt.dns.DNS_RCODE_NOERR
dns.qr = dpkt.dns.DNS_R

# make a record
arr = dpkt.dns.DNS.RR()
arr.cls = dpkt.dns.DNS_IN
arr.type = dpkt.dns.DNS_A
arr.name = 'paypal.com'
arr.ip = dnet.addr('127.0.0.1').ip

dns.an.append(arr)
print dns
>> DNS(an=[RR(name='paypal.com')], qd=[Q(name='paypal.com')], id=21825, op=32896)
```

DNS Arrays

dpkt.dns.* arrays	Description
qd(name='',type='')	Question
qd.name	The name that was searched, such as 'www.google.com'
qd.cls	class, dpkt.dns.DNS_IN
qd.type	dpkt.dns.DNS_A, many more options
an(name='',type='')	Answer
ns	List of name servers for this domain. You can iterate over the list

DNS Codes

qr Type of message, either query or response, hence Q/R: dpkt.dns.DNS_Q (0), dpkt.dns.DNS_R (1)

opcode What type of query, usually standard query: dpkt.dns.DNS_QUERY (0)

rcode Errors: dpkt.dns.DNS_RCODE_NOERR (0), anything else is an error

op No clue what this is: dpkt.dns.DNS_RA, dpkt.dns.DNS_AA

ar Authority record or additional record

DNS Questions

dpkt.dns.DNS.Q(data)

name Domain name

type Type of query

cls Class of query

data Data ?

DNS Answer

dpkt keeps an array of responses in dpkt.dns.DNS.RR(data) with the fields:

name Name that was queried

type Type of response, see dpkt.dns.*.type table

cls Class of response, usually internet addr: dpkt.dns.DNS_IN (1)

ttl The number of seconds the result can be cached

rlen The length of the RDATA field

rdata The response data. The format is dependent on the TYPE field: A(1) is IPv4 addr, CNAME(5) then a name, NS(2) is name servers, etc

data Data?

dpkt.dns.*.type		Description, assume rr is the python dpkt.dns.DNS object
DNS_A	1	IPv4 address; data = socket.inet_ntoa(rr.ip)
DNS_AAAA	28	IPv6 address
DNS_CNAME	5	Conical name or alias
DNS_HINFO	13	OS info
DNS_MX	15	Mail server: an.mxname
DNS_NS	2	Name server info:
DNS_PTR	12	Map IP to hostname, data = rr.ptrname; ex. 10.27/1.168.192.in-addr.arpa. 1800 PTR mail.example.com.
DNS_SRV	33	Service locator; data = rr.srvname, rr.priority, rr.weight, rr.port
DNS_SOA	6	Start of Authorities, gives info about domain: admin, contact info, etc
DNS_TXT	16	Text field; data = tuple(rr.text) # Convert the list to a hashable tuple

::

class DNS(dpkt.Packet):

__hdr__ = (('id', 'H', 0), ('op', 'H', DNS_RD), # recursive query # XXX - lists of query, RR objects ('qd', 'H', []), ('an', 'H', []), ('ns', 'H', []), ('ar', 'H', []))

)

class RR(Q): "''"DNS resource record."''" __hdr__ = (

(‘name’, ‘1025s’, ‘’), (‘type’, ‘H’, DNS_A), (‘cls’, ‘H’, DNS_IN), (‘ttl’, ‘I’, 0), (‘rlen’, ‘H’, 4), (‘rdata’, ‘s’, ‘’)

)

class Q(dpkt.Packet): “”“DNS question.”“” __hdr__ = (

(‘name’, ‘1025s’, ‘’), (‘type’, ‘H’, DNS_A), (‘cls’, ‘H’, DNS_IN)

)

Some examples:

```
    DNS_CACHE_FLUSH = 0x8000
    answer = dpkt.dns.DNS.RR(
            type = dpkt.dns.DNS_TXT,
            cls = dpkt.dns.DNS_IN | DNS_CACHE_FLUSH,
            ttl = 200,
            name = 'www.hello.com',
            text = 'Some text')

    ans = dpkt.dns.DNS.RR(
            type = dpkt.dns.DNS_SRV,
            cls = dpkt.dns.DNS_IN | DNS_CACHE_FLUSH,
            ttl = self._response_ttl,
            name = q.name,
            srvname = full_hostname,
            priority = priority,
            weight = weight,
            port = port)
    # The target host (srvname) requires to send an A record with its IP
    # address. We do this as if a query for it was sent.
    q = dpkt.dns.DNS.Q(name=full_hostname, type=dpkt.dns.DNS_A)
    answers = []
    for ip_addr in self._a_records[q.name]:
            answers.append(dpkt.dns.DNS.RR(
                    type = dpkt.dns.DNS_A,
                    cls = dpkt.dns.DNS_IN | DNS_CACHE_FLUSH,
                    ttl = self._response_ttl,
                    name = q.name,
                    ip = ip_addr))
[ans] + answers

    MDNS_IP_ADDR = '224.0.0.251'
    MDNS_PORT = 5353
    resp_dns = dpkt.dns.DNS(
            op = dpkt.dns.DNS_AA, # Authoritative Answer.
            rcode = dpkt.dns.DNS_RCODE_NOERR,
            an = answers)
    # This property modifies the "op" field:
    resp_dns.qr = dpkt.dns.DNS_R, # Response.
    sock.send(str(resp_dns), MDNS_IP_ADDR, MDNS_PORT)
```

DNSLib

class RR :rclass: ? :rdlength: ? :rname: ? :rtype: ? :ttl: ?

ARP

The Address Resolution Protocol (ARP) is a telecommunication protocol used for resolution of network layer addresses into link layer addresses, a critical function in multiple-access networks. ARP is used to convert a network address (e.g. an IPv4 address) to a physical address such as an Ethernet address (also known as a MAC address). *wiki <http://en.wikipedia.org/wiki/Address_Resolution_Protocol>*__

If you have the IP address, you can get the MAC address by sending an ARP message with a broadcast MAC address (FF:FF:FF:FF:FF:FF or 00:00:00:00:00:00 (arping uses)) which every computer will read. Then the computer with the IP address will respond with its MAC address.

dpkt.arp.*	
op	dpkt.arp.ARP_OP_REQUEST,dpkt.arp.ARP_OP_REPLY
sha	Source hardware address
spa	Source protocol address
tha	Target hardware address
tpa	Target protocol address

This doesn't work on Windows and OSX becuase socket.PF_PACKET isn't defined, only on linux:

```
s = socket.socket(socket.PF_PACKET, socket.SOCK_RAW)
s.bind(('en1', ethernet.ETH_TYPE_ARP))
```

```
my_mac = commands.getoutput("ifconfig " + 'en1' + "| grep ether | awk '{ print $2 }'")
ans = commands.getoutput('arp -i en1 -l -n 192.168.1.13')

# bulid an ARP reply
arp_p = arp.ARP()
arp_p.sha = eth_aton(src_mac)
arp_p.spa = socket.inet_aton(src_ip)
arp_p.tha = eth_aton(dst_mac)
arp_p.tpa = socket.inet_aton(dst_ip)
arp_p.op = arp.ARP_OP_REPLY

packet = ethernet.Ethernet()
packet.src = eth_aton(so_mac)
packet.dst = eth_aton(to_mac)
packet.data = arp_p
packet.type = ethernet.ETH_TYPE_ARP
```

You can also use *arping* to find the MAC:

```
[kevin@Tardis docs]$ sudo arping -c 3 192.168.1.6
ARPING 192.168.1.6
Timeout
42 bytes from 40:30:04:f0:8c:50 (192.168.1.6): index=0 time=556.113 msec
42 bytes from 40:30:04:f0:8c:50 (192.168.1.6): index=1 time=164.716 msec

--- 192.168.1.6 statistics ---
3 packets transmitted, 2 packets received,  33% unanswered (0 extra)
rtt min/avg/max/std-dev = 164.716/360.415/556.113/195.698 ms
```

Or simplify it using other utils:

```
sudo arping -c 2 192.168.1.5 | grep bytes | awk '{ print $4 }'
```

Decoding an ARP Packet

```
import binascii
def add_colons_to_mac( mac_addr ) :
        """This function accepts a 12 hex digit string and converts it to a colon
separated string"""
        s = list()
        for i in range(12/2) :  # mac_addr should always be 12 chars, we work in groups of 2 chars
                s.append( mac_addr[i*2:i*2+2] )
        r = ":".join(s)
        return r

eth = dpkt.ethernet.Ethernet(pkt)

# should actually double check this is an ARP packet and not assume
arp = eth.arp
print "source protocol address", socket.inet_ntoa(arp.spa)
print "source hardware address", add_colons_to_mac( binascii.hexlify(arp.sha) )
print "Target protocol address", socket.inet_ntoa(arp.tpa)      #IPv4 address
print "target hardware address", add_colons_to_mac( binascii.hexlify(arp.tha) )
arp_cache[arp.spa] = arp.sha
add_colons_to_mac( binascii.hexlify(arp_cache[ip]))
```

ICMP

The Internet Control Message Protocol (ICMP) is one of the main protocols of the Internet Protocol Suite. It is used by network devices, like routers, to send error messages indicating, for example, that a requested service is not available or that a host or router could not be reached.

The variable size of the ICMP packet data section has been exploited. In the well-known “Ping of death,” large or fragmented ping packets are used for denial-of-service attacks. ICMP can also be used to create covert channels for communication, as with the LOKI exploit.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             unused                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Internet Header + 64 bits of Original Data Datagram      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

types

- echo reply 0
- destination unreachable 3
- echo request 8
- timestamp 13
- timestamp reply 14

ICMPv6 RFC4443

The ICMPv6 messages have the following general format:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                         Message Body                          +
|                                                               |
```

ICMPv6 messages are grouped into two classes: error messages and informational messages. Error messages are identified as such by a zero in the high-order bit of their message Type field values. Thus, error messages have message types from 0 to 127; informational messages have message types from 128 to 255.

ICMPv6 error message types:

1 Destination Unreachable 2 Packet Too Big 3 Time Exceeded 4 Parameter Problem

ICMPv6 informational message types:

128 Echo Request 129 Echo Reply

ICMPv6 Fields:

Type 1 Destination Unreachable

Code 0 - No route to destination

1 - Communication with destination administratively prohibited

2 - Beyond scope of source address 3 - Address unreachable 4 - Port unreachable 5 - Source address failed ingress/egress policy 6 - Reject route to destination

ICMPv6 Informational Messages

Echo Request Message

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |     Type      |     Code      |          Checksum             |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |           Identifier          |        Sequence Number        |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |     Data ...
     +-+-+-+-+-


:Type:            128
:Code:            0
:Identifier:      An identifier to aid in matching Echo Replies to this Echo Request.  May be zero.
:Sequence Number: A sequence number to aid in matching Echo Replies to this Echo Request.  May be zer
:Data:            Zero or more octets of arbitrary data.
```

Echo Reply Message

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Identifier          |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Data ...
+-+-+-+-+-
```

Type 129

Code 0

Identifier The identifier from the invoking Echo Request message.

Sequence Number The sequence number from the invoking Echo Request message.

Data The data from the invoking Echo Request message.

Neighbor Discovery (ND) protocol for Internet Protocol Version 6 (IPv6)

RFC4861

Nodes (hosts and routers) use Neighbor Discovery to determine the link-layer addresses for neighbors known to reside on attached links and to quickly purge cached values that become invalid. Hosts also use Neighbor Discovery to find neighboring routers that are willing to forward packets on their behalf. Finally, nodes use the protocol to actively keep track of which neighbors are reachable and which are not, and to detect changed link-layer addresses.

Neighbor Solicitation Message Format

Nodes send Neighbor Solicitations to request the link-layer address of a target node while also providing their own link-layer address to the target. Neighbor Solicitations are multicast when the node needs to resolve an address and unicast when the node seeks to verify the reachability of a neighbor.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |     Type      |     Code      |          Checksum             |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                           Reserved                            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                                                               |
     +                                                               +
     |                                                               |
     +                       Target Address                          +
     |                                                               |
     +                                                               +
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   Options ...
     +-+-+-+-+-+-+-+-+-+-+-+-

 IP Fields:

   Source Address
                  Either an address assigned to the interface from
                  which this message is sent or (if Duplicate Address
                  Detection is in progress [ADDRCONF]) the
                  unspecified address.
   Destination Address
                  Either the solicited-node multicast address
                  corresponding to the target address, or the target
                  address.
   Hop Limit      255

ICMP Fields:

   Type           135

   Code           0

   Checksum       The ICMP checksum.  See [ICMPv6].

   Reserved       This field is unused.  It MUST be initialized to
                  zero by the sender and MUST be ignored by the
                  receiver.

   Target Address The IP address of the target of the solicitation.
                  It MUST NOT be a multicast address.

Possible options:

   Source link-layer address
                  The link-layer address for the sender.  MUST NOT be
                  included when the source IP address is the
                  unspecified address.  Otherwise, on link layers
                  that have addresses this option MUST be included in
                  multicast solicitations and SHOULD be included in
                  unicast solicitations.
```

Neighbor Advertisement Message Format

A node sends Neighbor Advertisements in response to Neighbor Solicitations and sends unsolicited Neighbor Advertisements in order to (unreliably) propagate new information quickly.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|S|O|                     Reserved                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                       Target Address                          +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options ...
+-+-+-+-+-+-+-+-+-+-+-+-
```

IP Fields:

Source Address An address assigned to the interface from which the advertisement is sent.

Destination Address For solicited advertisements, the Source Address of an invoking

Neighbor Solicitation or, if the solicitation's Source Address is the unspecified address, the all-nodes multicast address. For unsolicited advertisements typically the all-nodes multicast address.

Hop Limit 255

ICMP Fields:

Type 136

Code 0

Checksum The ICMP checksum. See [ICMPv6].

R Router flag. When set, the R-bit indicates that the sender is a router. The R-bit is used by Neighbor Unreachability Detection to detect a router that changes to a host.

S Solicited flag. When set, the S-bit indicates that the advertisement was sent in response to a Neighbor Solicitation from the Destination address. The S-bit is used as a reachability confirmation for Neighbor Unreachability Detection. It MUST NOT be set in multicast advertisements or in unsolicited unicast advertisements.

O Override flag. When set, the O-bit indicates that the advertisement should override an existing cache entry and update the cached link-layer address. When it is not set the advertisement will not update a cached link-layer address though it will update an existing Neighbor Cache entry for which no link-layer address is known. It SHOULD NOT be set in solicited advertisements for anycast addresses and in solicited proxy advertisements. It SHOULD be set in other solicited advertisements and in unsolicited advertisements.

Reserved 29-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Target Address For solicited advertisements, the Target Address field in the Neighbor Solicitation message that prompted this advertisement. For an unsolicited advertisement, the address whose link-layer address has changed. The Target Address MUST NOT be a multicast address.

Possible options:

Target link-layer address The link-layer address for the target, i.e., the sender of the advertisement. This option MUST be included on link layers that have addresses when responding to multicast solicitations. When responding to a unicast Neighbor Solicitation this option SHOULD be included.

The option MUST be included for multicast solicitations in order to avoid infinite Neighbor Solicitation “recursion” when the peer node does not have a cache entry to return a Neighbor Advertisements message. When responding to unicast solicitations, the option can be omitted since the sender of the solicitation has the correct link- layer address; otherwise, it would not be able to send the unicast solicitation in the first place. However, including the link-layer address in this case adds little overhead and eliminates a potential race condition where the sender deletes the cached link-layer address prior to receiving a response to a previous solicitation.

Multicast DNS (mDNS)

The multicast Domain Name System (mDNS) resolves host names to IP addresses within small networks that do not include a local name server. It is a zero configuration service, using essentially the same programming interfaces, packet formats and operating semantics as the unicast Domain Name System (DNS). While it is designed to be stand-alone capable, it can work in concert with unicast DNS servers.

The mDNS Ethernet frame is a multicast UDP packet to:

- MAC address 01:00:5E:00:00:FB (for IPv4) or 33:33:00:00:00:FB (for IPv6)
- IPv4 address 224.0.0.251 or IPv6 address FF02::FB
- UDP port 5353

You can simulate mDNS request with *dig*:

```
[kevin@Tardis test]$ dig -p 5353 @224.0.0.251 calculon.local

; <<>> DiG 9.8.3-P1 <<>> -p 5353 @224.0.0.251 calculon.local
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53097
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;calculon.local.                        IN      A

;; ANSWER SECTION:
calculon.local.         10      IN      A       192.168.1.17

;; Query time: 59 msec
;; SERVER: 192.168.1.17#5353(224.0.0.251)
;; WHEN: Thu May 28 12:04:07 2015
;; MSG SIZE  rcvd: 48
```

Active Network Mapping

Fast determination if a host is up: UDP - low cost to send packets

1. Send UDP packets to a port on a remote machine

2. Listen for ICMP (error, type=code=3) responses back. An unreachable port error means the host is up

Scan a host for open ports: TCP

1. For each host that is up, start the handshake process:
2. Send a SYN packet to a port

3. Wait for the ACK (port is open) and then send a RST (reset) to close out the process and move on to the next port
4. If you don't get the ACK, then the port is closed

Passive Network Mapping

Listen for:

- DNS responses: DNS_A will match names to IPv4 addresses
- mDNS: same as above
- mDNS: DNS_SRV will match services to IPv4 addresses

Resources

- DNS record types
- dpkt doc
- dpkt-tutorial-1-icmp-echo
- dpkt-tutorial-2-parsing-a-pcap-file
- dpkt-tutorial-3-dns-spoofing

- IP Protocol headers
- ICMP message
- ARP message
- mDNS message
- unixwiz.net
- IANA DNS types
- Internet Engineering Task Force

Airplay

http://nto.github.io/AirPlay.html#audio-airportexpressauthentication

Misc

```
import AppKit
# Create instance of OS X notification center
notification_center = AppKit.NSUserNotificationCenter.defaultUserNotificationCenter()
# Create new notification instance
notification = AppKit.NSUserNotification.alloc().init()
notification.setTitle_(title)
notification.setSubtitle_(subtitle)
notification.setInformativeText_(content)
# Deliver OS notifications
notification_center.deliverNotification_(notification)
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search