VirginiaTech
Invent the Future

# Raspberry Pi - Hardware Interface

## ECE 4564 - Network Application Design

**Dr. William O. Plymale**

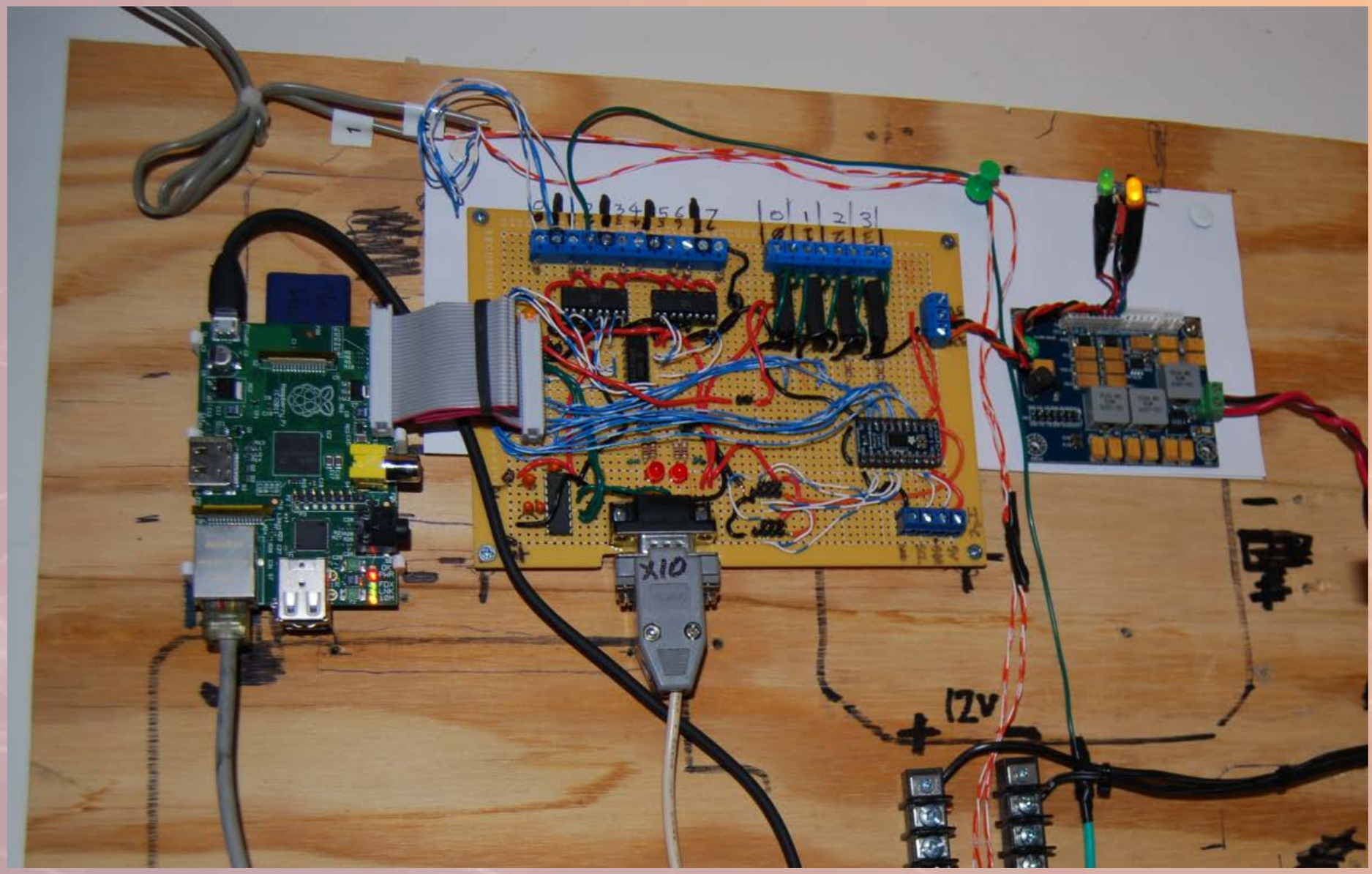
BRADLEY DEPARTMENT
of ELECTRICAL & COMPUTER ENGINEERING

# Topics

Interactive Hardware

- General-Purpose Input/Output
- Raspberry Pi – GPIO
- Sysfs
- Python Rpi.GPIO Module

# Interactive Hardware



# General-Purpose Input/Output

A generic pin on a microcontroller whose behavior, including whether it is an input or output pin, can be controlled by the user at run time.

GPIO capabilities may include:

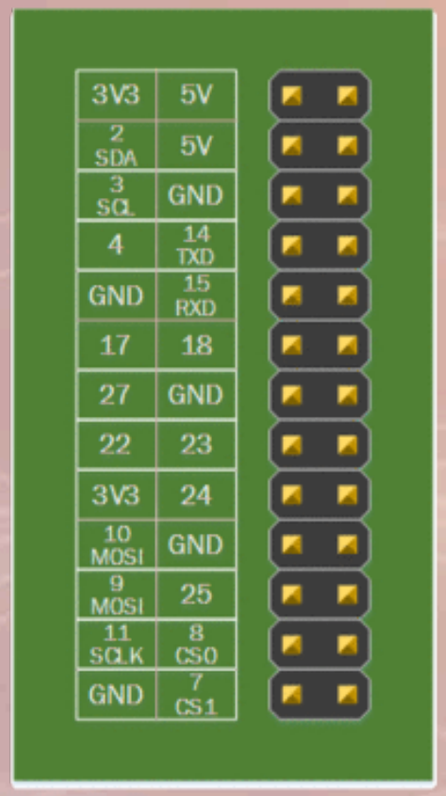
- GPIO pins can be configured to be input or output
- GPIO pins can be enabled/disabled
- Input values are readable (typically high=1, low=0)
- Output values are writable/readable
- Input values can often be used as IRQs (typically for wakeup events)

# Raspberry Pi GPIO

| | | | | | |
|---|---|---|---|---|---|
| 3.3V PWR | 1 | | 2 | 5V PWR |
| I2C1 SDA | 3 | | 4 | 5V PWR |
| I2C1 SCL | 5 | | 6 | GND |
| Reserved | 7 | | 8 | Reserved |
| GND | 9 | | 10 | Reserved |
| SPI1 CS0 | 11 | | 12 | GPIO 18 |
| GPIO 27 | 13 | | 14 | GND |
| GPIO 22 | 15 | | 16 | GPIO 23 |
| 3.3V PWR | 17 | | 18 | GPIO 24 |
| SPI0 MOSI | 19 | | 20 | GND |
| SPI0 MISO | 21 | | 22 | GPIO 25 |
| SPI0 SCLK | 23 | | 24 | SPI0 CS0 |
| GND | 25 | | 26 | SPI0 CS1 |
| Reserved | 27 | | 28 | Reserved |
| GPIO 5 | 29 | | 30 | GND |
| GPIO 6 | 31 | | 32 | GPIO 12 |
| GPIO 13 | 33 | | 34 | GND |
| SPI1 MISO | 35 | | 36 | GPIO 16 |
| GPIO 26 | 37 | | 38 | SPI1 MOSI |
| GND | 39 | | 40 | SPI1 SCLK |

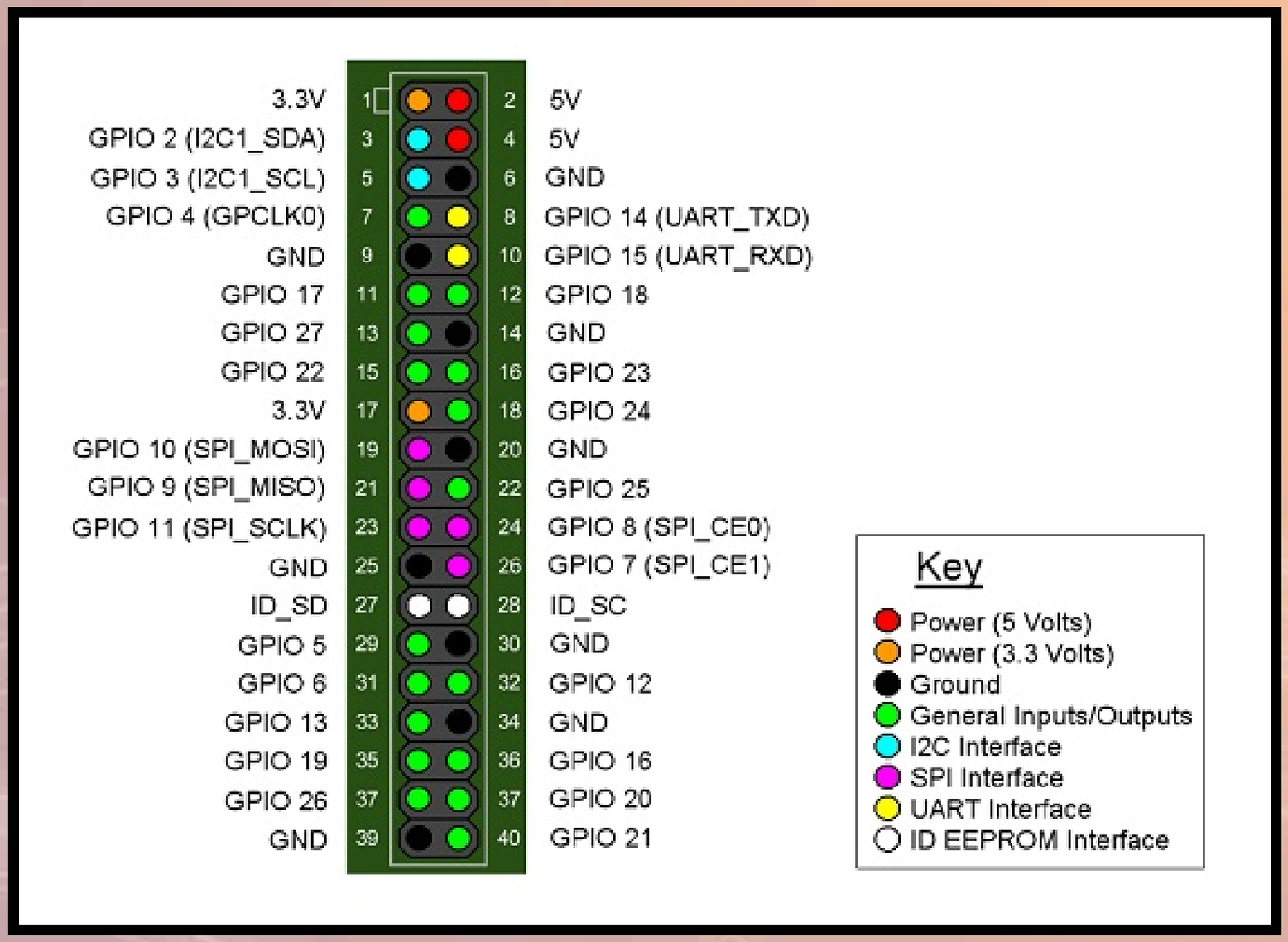
# Raspberry Pi

## Models A & B




## Models A+, B+ & Pi2

# Raspberry Pi




GPIO Reference

# More on Pin Numbering

The GPIO pins are sometimes renamed with another set of numbers.

In order to avoid damaging your Pi you need to be sure what pins you are connecting to other hardware and that your program is referring to the correct pins.

http://raspberrypi.stackexchange.com/questions/12966/what-is-the-difference-between-board-and-bcm-for-gpio-pin-numbering

http://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/

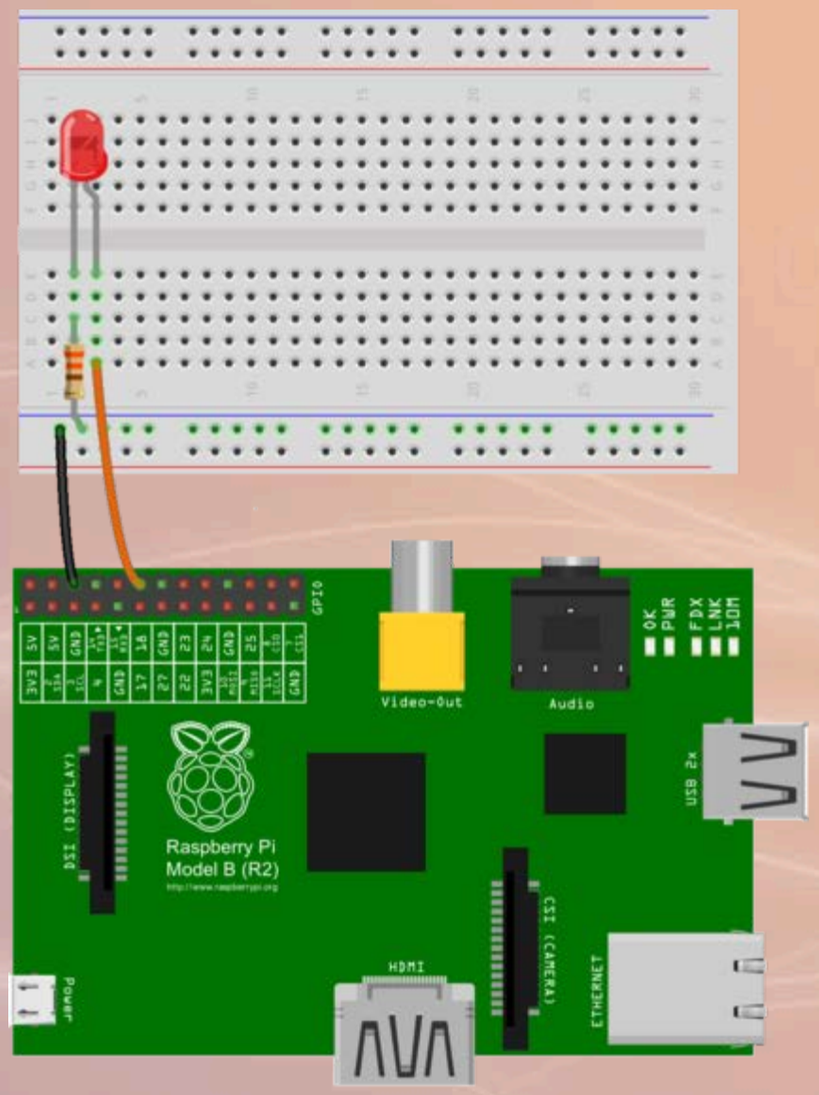
# GPIO Pins – Raspberry Pi

- GPIO voltage levels are 3.3 V and are not 5 V tolerant.
- There is no over-voltage protection on the board
  - the intention is that people interested in serious interfacing will use an external board with buffers, level conversion and analog I/O rather than soldering directly onto the main board.

**(Sending 5V to a pin may kill the Pi)**

**Rpi Low-level Peripherals**

# Raspberry Pi




Turning on an LED

# GPIO with sysfs on Raspberry Pi

- In Linux everything is a file: /dev/ttyUSB0, /sys/class/net/eth0/address, /dev/mmcblk0p2,…

- sysfs is a kernel module providing a virtual file system for device access at /sys/class
  - provides a way for users (or code in the user-space) to interact with devices at the system (kernel) level

- Advantages / Disadvantage
  - Allows conventional access to pins from userspace
  - Much slower the digitalWrite()/digitalRead() of Arduino

# /sys/class/gpio

- Explore this directory
- As *root,* cd /sys/class/gpio
- List files
  - export
  - gpiochip0 – sym link
  - unexport
- Create the sysfs alias for a pin by *exporting* the pin
  - echo 4 > export
- sysfs monitors these files, and updates the links between user-space and kernel-space when they're updated
- When finished, echo 4 > unexport

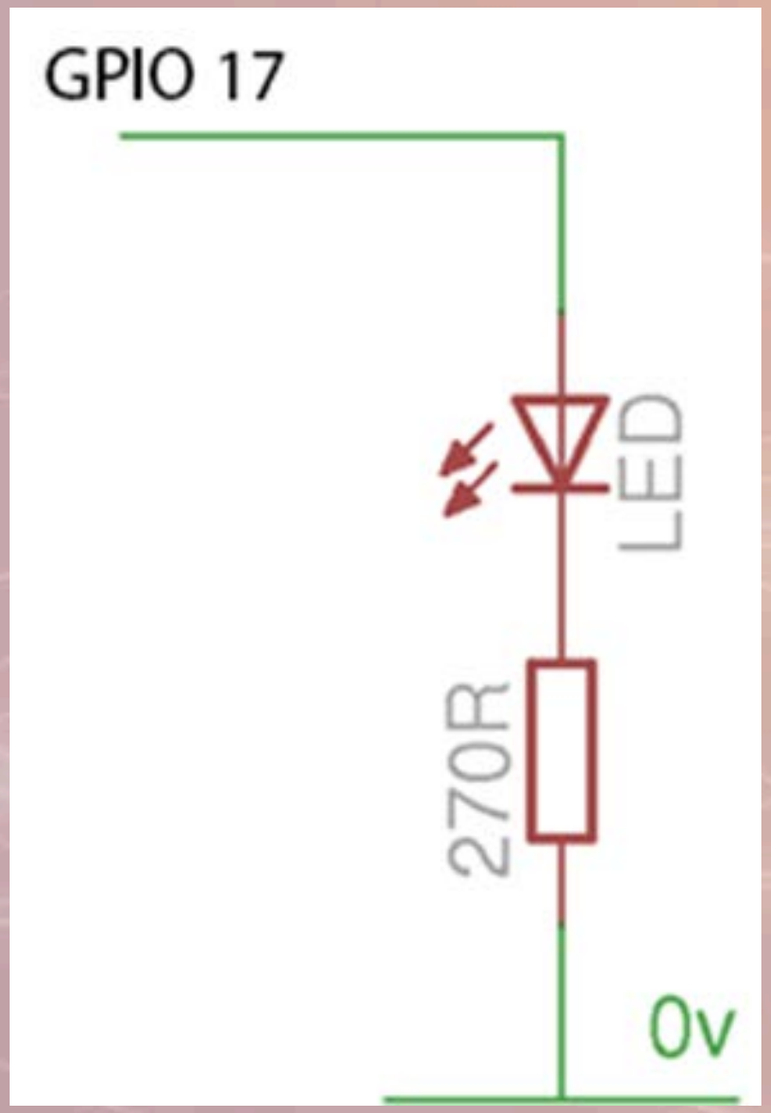
# /sys/class/gpio

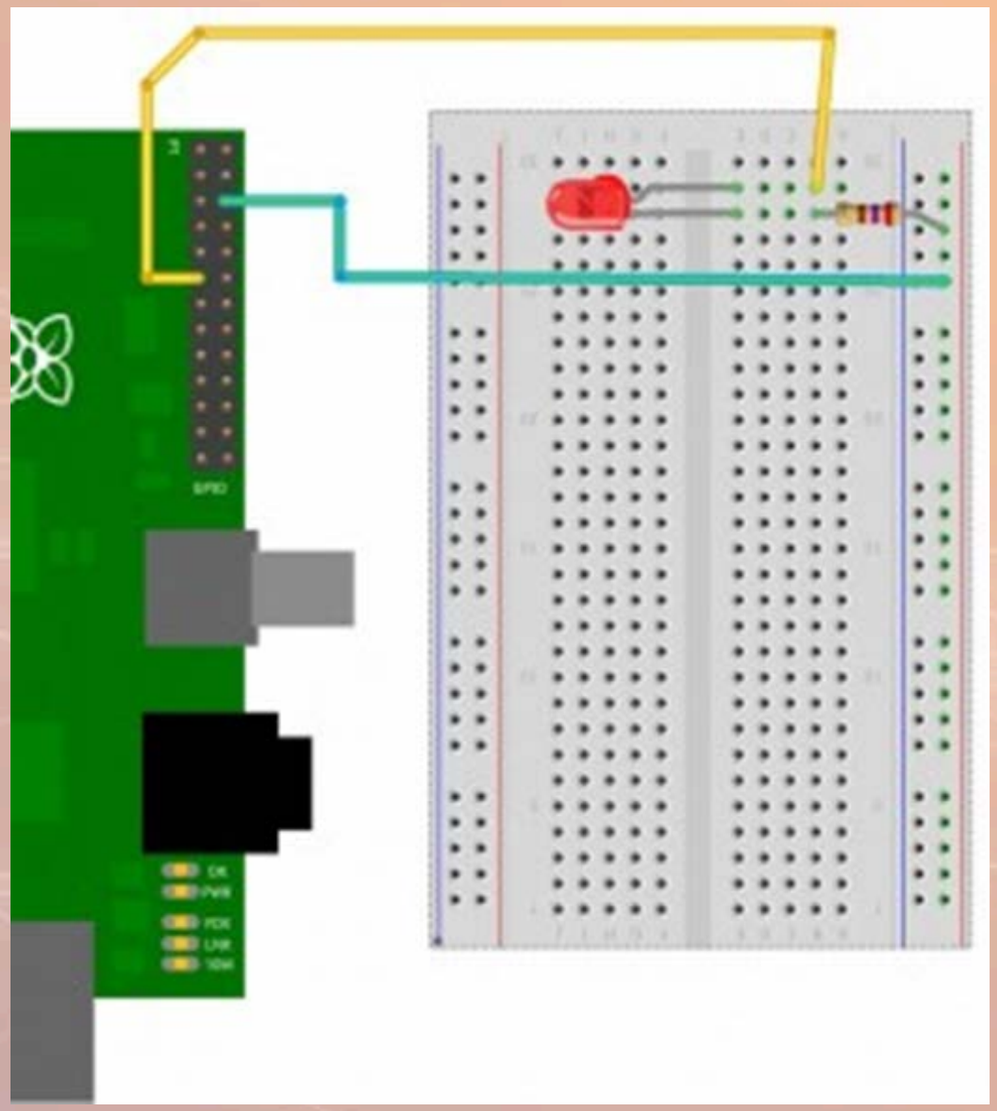
- Export the pin we want to use
  - Write the pin number to /sys/class/gpio/export
    - echo 17 > /sys/class/gpio/export

- Set the direction
  - Write "in" or "out" to /sys/class/gpio/gpio??/direction
    - echo out > /sys/class/gpio/gpio17/direction

- Set the value
  - Write "1" or "0" to /sys/class/gpio/gpio??/value
    - echo 1 > /sys/class/gpio/gpio17/value

# Connect an LED between GPIO 17 (P1-11) and GND



# blink.sh

```
#!/bin/sh
echo 17 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio17/direction
while true
do
        echo 1 > /sys/class/gpio/gpio17/value
        sleep 1
        echo 0 > /sys/class/gpio/gpio17/value
        sleep 1
done
```

# Rpi.GPIO

A module to control Raspberry Pi GPIO channels

RPi.GPIO 0.7.0

Installation

# Rpi.GPIO Demo

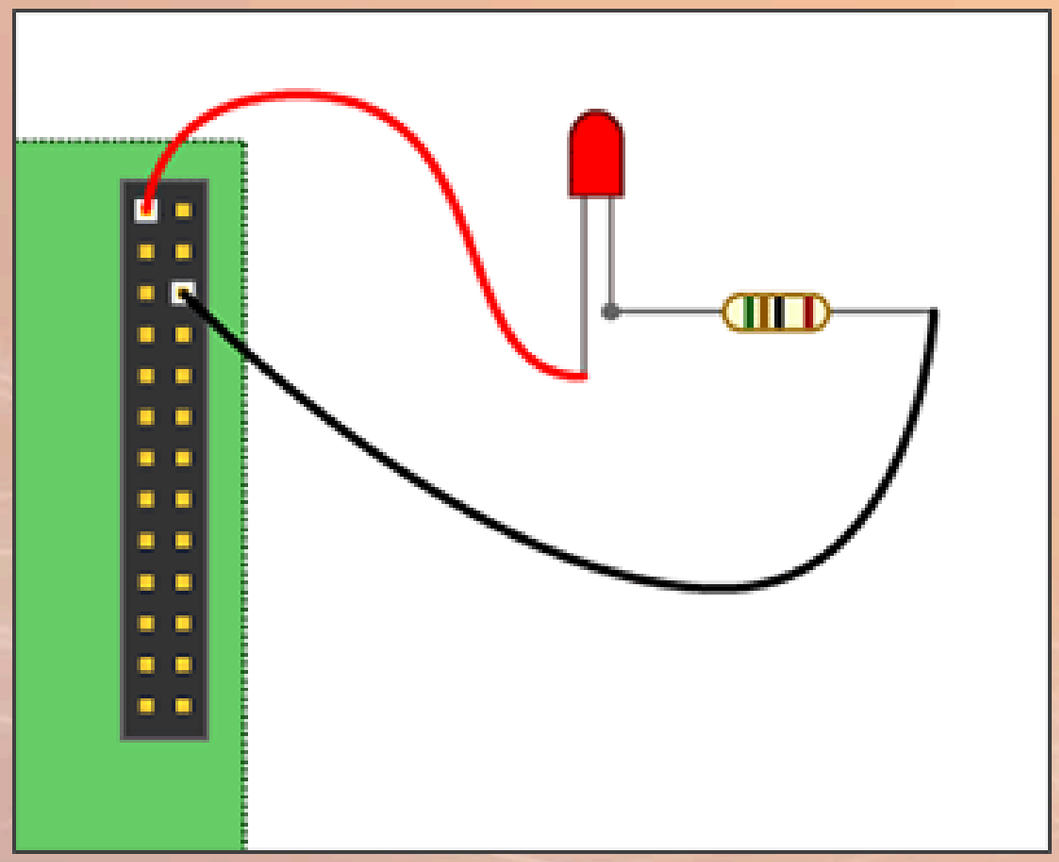
```
#!/usr/local/bin/python

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(17, GPIO.OUT)
GPIO.output(17, False)

while True:
        GPIO.output(17, True)
        time.sleep(2)
        GPIO.output(17, False)
        time.sleep(2)
```



Emulator

# PWM Control

Pulse width modulation (PWM) is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts.

PWM in Python

# Flex Sensor

Flex Sensor with Raspberry Pi

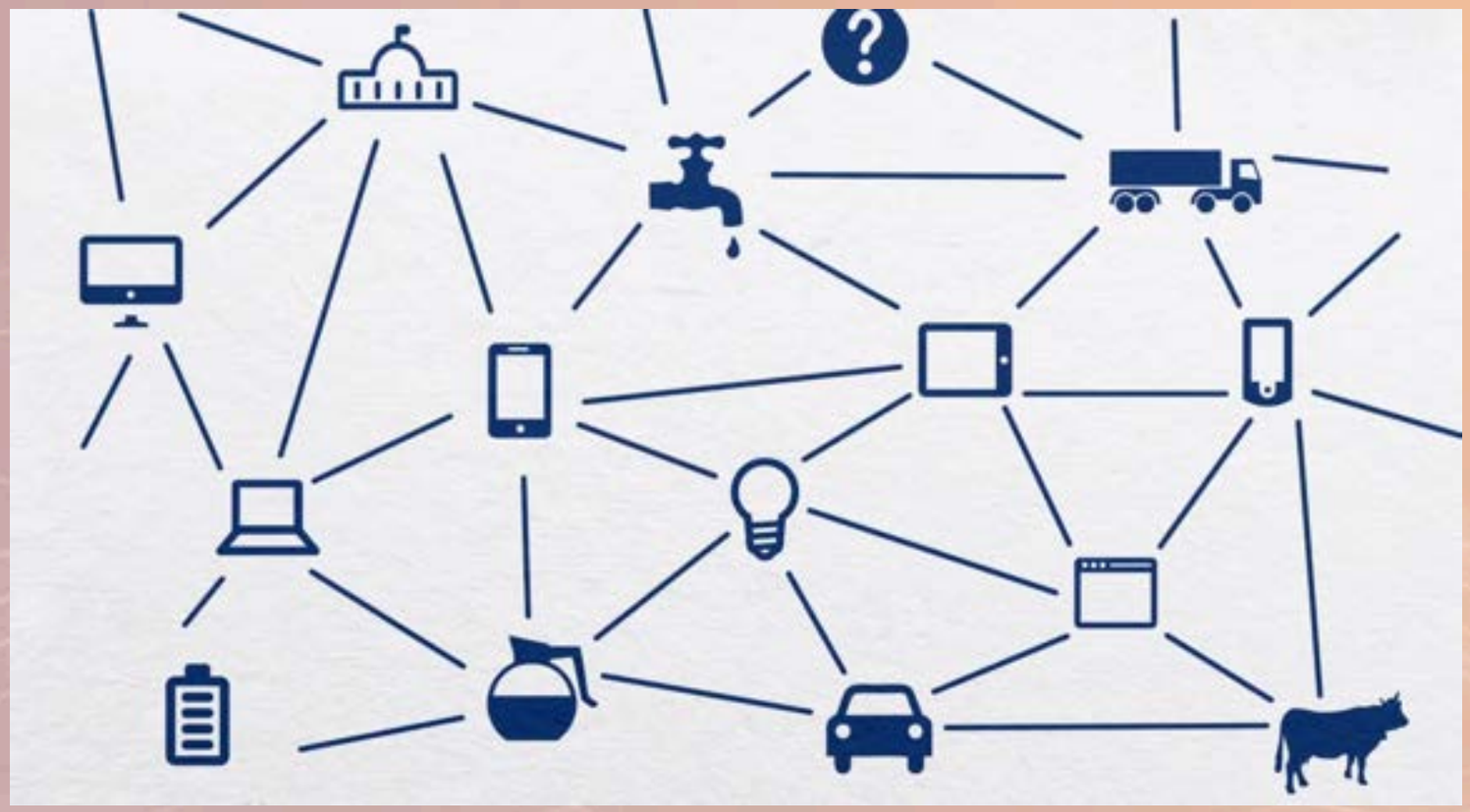
# Absolute Orientation Sensor

Absolute Orientation Sensor

# Stepper Motor

Stepper Motor

# Closing



# REpresentational State Transfer (REST)

# Acknowledgements

This lecture has been inspired by and based off the work of NetApps GTAs from the past:

| | |
|---|---|
| Kelvin Aviles | Fall 2017, Spring 2018 |
| Prakriti Gupta | Fall 2016, Spring 2017 |
| Gaurang Naik | Fall 2015 |
| Thaddeus Czauski | Fall 2014 |

# Outline

1. What is REST?
2. What is HTTP?
    a. Requests and Methods
    b. Response and Status Codes
    c. Examples
3. Multipart Request/Response
4. Caching
5. Cookies
6. REST Properties

# What is REST?

# REST

- Based off of Hypertext Transfer Protocol (HTTP).
- Often incorrectly used interchangeably with HTTP.
- Has sadly become somewhat of a buzzword as a result.
- NOT a protocol like HTTP.
- It describes how a protocol should be used, like a set of principles or rules.
- Can be used with protocols besides HTTP but is mostly only used for HTTP.

# Dr. Roy Fielding

- Described REST as part of his dissertation in 2000 at UC Irvine.
- Did this while working on HTTP 1.1
- Co-Founded Apache Server
- HTTP Work probably contributed to confusion with REST.

We will come back to REST...

# What is HTTP?

# HTTP

RFC2616:

"The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers."

# HTTP Request/Response Diagram

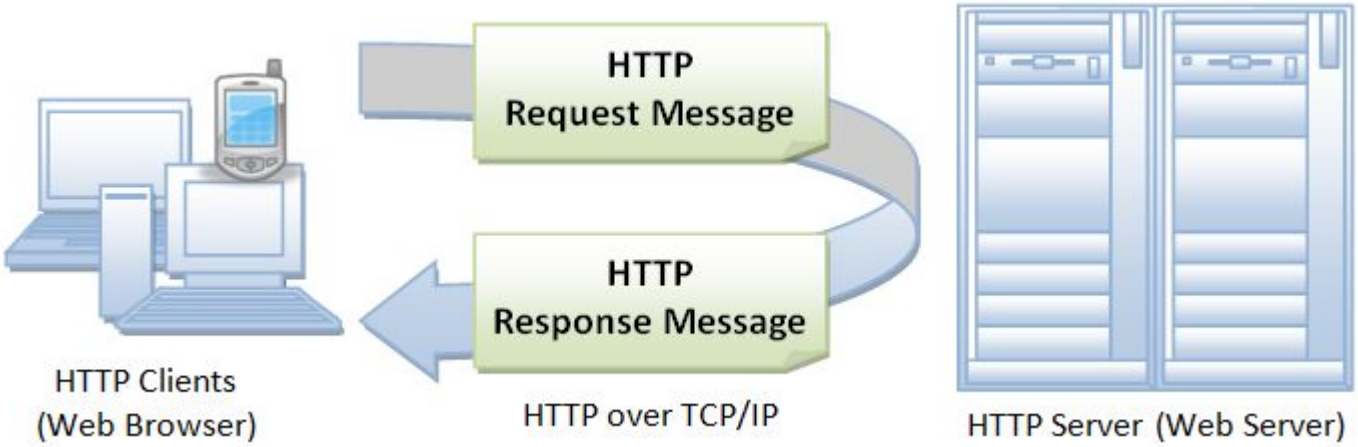
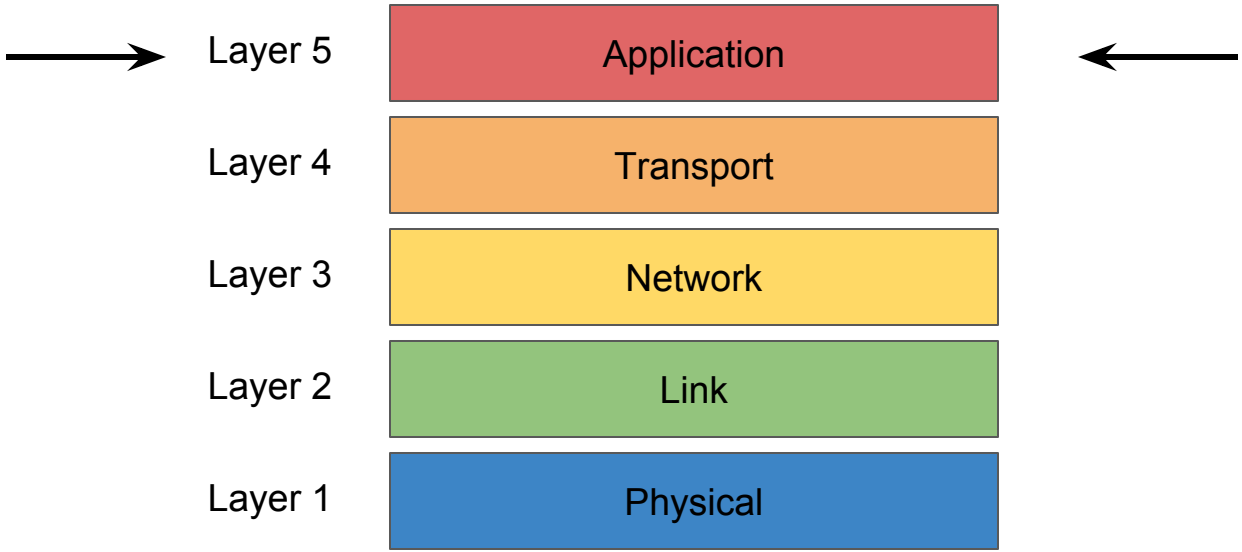



Image Ref: https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

# Internet Layers



# HTTP Requests and Responses

- google.com
- twitter.com
- wolframalpha.com
- vt.edu
- Each are part of an HTTP request.

# HTTP Request Format

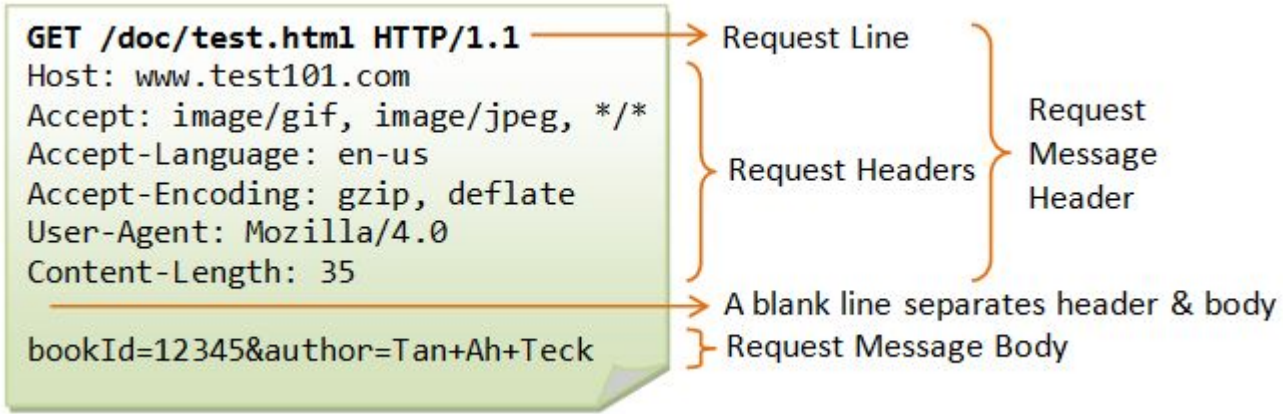



Image Ref: https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

# HTTP Request Line

request-line = method SP request-target SP HTTP-version CRLF

GET / HTTP/1.1

HTTP/1.1 200 OK, Resource Data...

# HTTP Request Line

request-line = method SP request-target SP HTTP-version CRLF

GET / HTTP/1.1

HTTP/1.1 200 OK, Resource Data...

# HTTP Request Line

request-line = method SP request-target SP HTTP-version CRLF

GET / HTTP/1.1

HTTP/1.1 200 OK, Resource Data...

# HTTP Request Line

request-line = method SP request-target SP HTTP-version CRLF

GET / HTTP/1.1

HTTP/1.1 200 OK, Resource Data...

# CRLF?

- CRLF is how the Server knows the request line has ended.
- What comes next?
  - Request Headers
  - Request Body

# HTTP Methods

- **GET** – request (or "get") for a piece of resource from a HTTP server.
- **POST** – used to "post" additional data up to the server (e.g., submitting HTML form data or uploading a file).
- **HEAD** – request only the response header.
- **PUT** – used to update a file/resource on the server.
- **DELETE** – delete a resource present on the server.
- **OPTIONS** – query the server for a list of supported HTTP methods.
- **CONNECT** – creates TCP/IP connection.
- **TRACE** – echo the received request so client knows changes that occurred on server.

# Note on REST and HTTP Request Methods

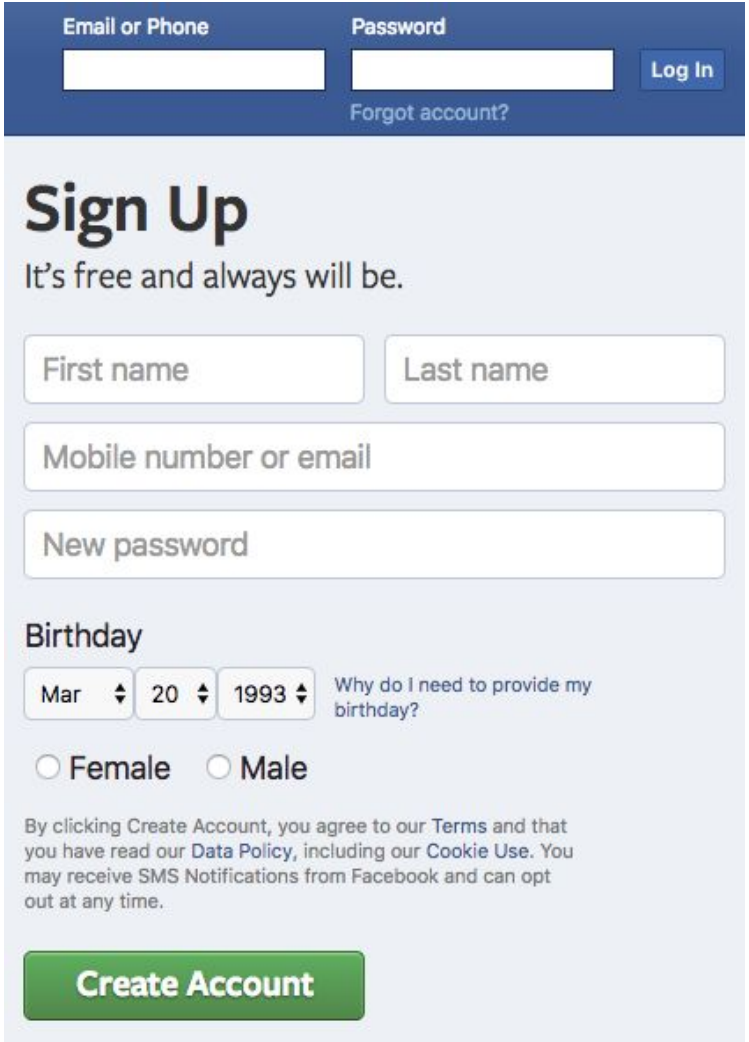
- REST rules dictate that you use GET, POST, PUT, PATCH, DELETE.
- In real world, for most part, only GET and POST are used.
- Sometimes you might see PUT and DELETE.
- Why? Historical design decisions related HTML Forms and Web Browsers.

# POST Method

- Used to modify data on a server. Used in conjunction with an HTTP Form.
- Think account login or filling out a survey.
- HTTP POST Requests are formatted almost the same as HTTP GET REquests.
- Parameters are not included in the URI though.
- Stored in the Request Body like GET, but formatted differently.

# Facebook Login/Account Creation

- When the “Log In” or “Create Account” button is clicked, the web browser will generate a POST request based on the form data and send it to Facebook for processing.
- Facebook then responds with an HTTP Response.



# Note on GET vs. POST methods

- If a server supports it, the client could send data through either method.
- This is strongly NOT encouraged.
- Convention is to stick with GET for getting a resource and POST for updating a resource.
- If you had the following keys-value pairs:
  - Name: Kelvin
  - Hair: Black
  - Eyes: Black
  - Feelings: None

# Note on GET vs. POST methods cont.

- For GET request, these could be found in the URI and/or Request Body:
  - Request Body format:

    Name:Kelvin&Hair:Black&Eyes:Black&Feelings:None

- For POST request, these would be found ONLY in the Request Body.
  - Request Body format:

    Name=Kelvin
    Hair=Black
    Eyes=Black
    Feelings=None

# HTTP Response Format

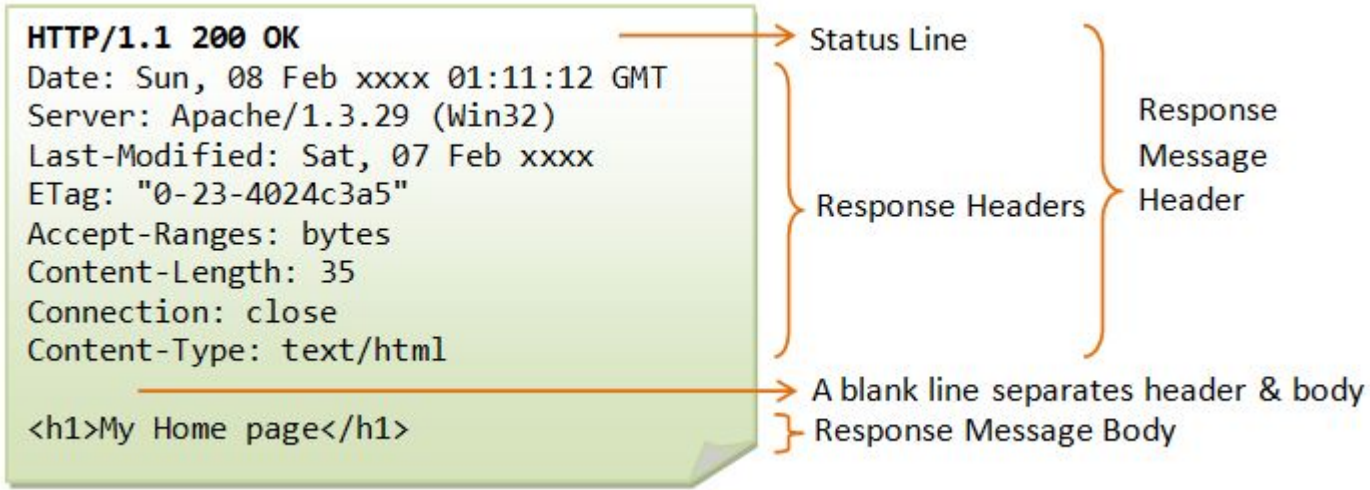



Image Ref: https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

# HTTP Status Line

status-line = HTTP-version SP status-code SP reason-phrase CRLF

GET / HTTP/1.1

HTTP/1.1 200 OK, Resource Data...

# HTTP Status Line

status-line = HTTP-version SP status-code SP reason-phrase CRLF

GET / HTTP/1.1

HTTP/1.1 200 OK, Resource Data...

# HTTP Status Line

status-line = HTTP-version SP status-code SP reason-phrase CRLF

GET / HTTP/1.1

HTTP/1.1 200 OK, Resource Data...

# HTTP Status Line

status-line = HTTP-version SP status-code SP reason-phrase CRLF

GET / HTTP/1.1

HTTP/1.1 200 OK, Resource Data...

# CRLF?

- CRLF is how the Client knows the response status line has ended.
- What comes next?
  - Response Headers
  - Response Body

# HTTP Status Codes

- 1xx – informational message
  - 100 - Continue.
- 2xx – success
  - 200 - OK.
- 3xx – redirect somewhere else
  - 304 - Moved Permanently.
- 4xx – client side error
  - 400 - Bad Request.
  - 403 - Forbidden.
  - 404 - Not Found.
- 5xx – server side error
  - 500 - Internal Server Error.

# Example GET Request

GET / HTTP/1.1
Host: www.vt.edu

# Example Response

HTTP/1.1 200 OK
Date: Sat, 15 Oct 2016 22:16:51 GMT
Server: Apache
X-RouteInfo: cmsw-prod-01
Cache-Control: max-age=60, public, must-revalidate
Vary: Accept-Encoding
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="created" content="2016-10-14T07:27:05Z"/>

# Example POST Request

POST /fake_login HTTP/1.1
Host: www.vt.edu

username:kaviles
password:iloveece123

# Example Response

HTTP/1.1 200 OK
Date: Sat, 15 Oct 2016 22:16:51 GMT
Server: Apache
X-RouteInfo: cmsw-prod-01
Cache-Control: max-age=60, public, must-revalidate
Vary: Accept-Encoding
Content-Type: text/html; charset=UTF-8

# Multipart Requests or Response

- A header that tells the client or server that the request or response will come in multiple parts instead of a single part.
- This is typically for large file upload or downloads.

# Caching

- Client can store server response to prevent from making requests again.
- Server response should have information about how caching can be done at the client.
- This is usually done to lighten server load or make client experience load faster.
- Cache control headers
  - **Public** - resource is cacheable by any component
  - **Private** - resource is cacheable by only client and
  - server
  - **No-cache/no-store** - resource is not cacheable
  - **Max-age** - valid up to max-age in seconds
  - **Must-revalidate** - revalidate resource if max-age has passed

# Cookies

- HTTP is stateless. But can maintain state by using cookies.
- A cookie is a small piece of data that the server sends back to the client as a result of a request that the client stores
- On subsequent requests to the server, the client automatically includes any cookies that it received from that server
- Example: Language preference for a website

# REST Properties

- All content (txt, html, jpg, mp4 etc.) is treated like a resource.
- Must use Universal Resource Identifiers (URIs).
  - Format:
  - Note: Anything in brackets is optional.

```
scheme:[//[user[:password]@]host[:port]][/path][?query][#fragment]
```

- Resources can be represented any way. Most commonly JSON and XML.

VirginiaTech
Invent the Future

# RESTful Tools

## ECE 4564 - Network Application Design

**Dr. William O. Plymale**

ece
BRADLEY DEPARTMENT
of ELECTRICAL & COMPUTER ENGINEERING

# Topics

- cURL
- Python Requests
- Web Microframeworks

# cURL

- The name "cURL" stands for "client URL“
- A tool that allows you to interact with URLs to get a variety of work tasks done

*$ curl http://www.vt.edu*

- show the web page at this address
  - makes a GET request, fetch the page and displays the corresponding HTML code

# cURL

- Download web pages and upload files
- Post data to web sites
- Converse with web sites using wide range of protocols
  - HTTP
  - DICT
  - FTP
  - SMTP

# cURL

- Interacts with web-based APIs like REST
- Ported to many operating systems including Linux, Mac OS and Windows.

cURL Tutorial

# cURL Examples

1. curl http://wttr.in/LOCATION
2. curl wttr.in/Moon
3. curl http://artscene.textfiles.com/asciiart/panda

Get cURLy

# Requests

- Requests is an Apache2 Licensed HTTP library, written in Python,
- Requests allow you to send HTTP/1.1 requests.
- You can add headers, form data, multipart files, and parameters with simple Python dictionaries, and access the response data in the same way.

Requests:HTTP for Humans

# HTTP Requests Types

```
r = requests.get('https://github.com/timeline.json')

r = requests.post("http://httpbin.org/post")

r = requests.put("http://httpbin.org/put")

r = requests.delete("http://httpbin.org/delete")

r = requests.head("http://httpbin.org/get")

r = requests.options("http://httpbin.org/get")
```

# Requests Example

```
>>> import requests
>>> r = requests.get('https://github.com/timeline.json')
>>> r.text
```

# URL Parameters

```
>>> import requests
>>> payload = {'key1': 'value1', 'key2': 'value2'}
>>> r = requests.get('http://httpbin.org/get', params=payload)


>>> print(r.url)
```

# Status Codes and Headers

```
>>> import requests
>>> r = requests.get('http://httpbin.org/get')
>>> r.status_code

>>>r.headers
>>> r.headers['Content-Type']
```

# Response Content

```
>>> import requests
>>> r = requests.get('https://api.github.com/events')
>>> r.text


>>>r.content       //for non-text requests

>>>r.json()        //builtin JSON decoder
```

# POST Requests

```
>>> import requests
>>> payload = {'key1': 'value1', 'key2': 'value2'}

>>> r = requests.post("http://httpbin.org/post", data=payload)
>>> print(r.text)
```

# Python Requests Tutorial

Python API Tutorial

How to Call a Weather API

# Web Framework

- Web Framework represents a collection of libraries and modules that enables a web application developer to write applications without having to bother about low-level details such as protocols, thread management etc.

- Common Web Framework Functionality:
  - URL routing
  - HTML, XML, JSON, and other output format templating
  - Database manipulation
  - Security Support
  - Session Storage and Retrieval

# Full-Stack vs Micro

Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer.

Full-Stack

- An enterprise grade business application
- Django

Micro

- An API focused application
- Flask

Best Python Microframeworks

# Flask

- Flask is a micro web development framework for Python.
- Developed by Armin Ronacher, who leads an international group of Python enthusiasts named Pocco.
- Based on
  - Werkzeug WSGI toolkit
    - Web Server Gateway Interface
    - Implements requests, response objects, and other utility functions
  - Jinja2 template engine
    - web templating system combines a template with a certain data source to render dynamic web pages

# Flask and REST

“The task of designing a web service or API that adheres to the REST guidelines is an exercise in identifying the resources that will be exposed and how they will be affected by the different request methods.”

Designing a RESTful API with Python and Flask

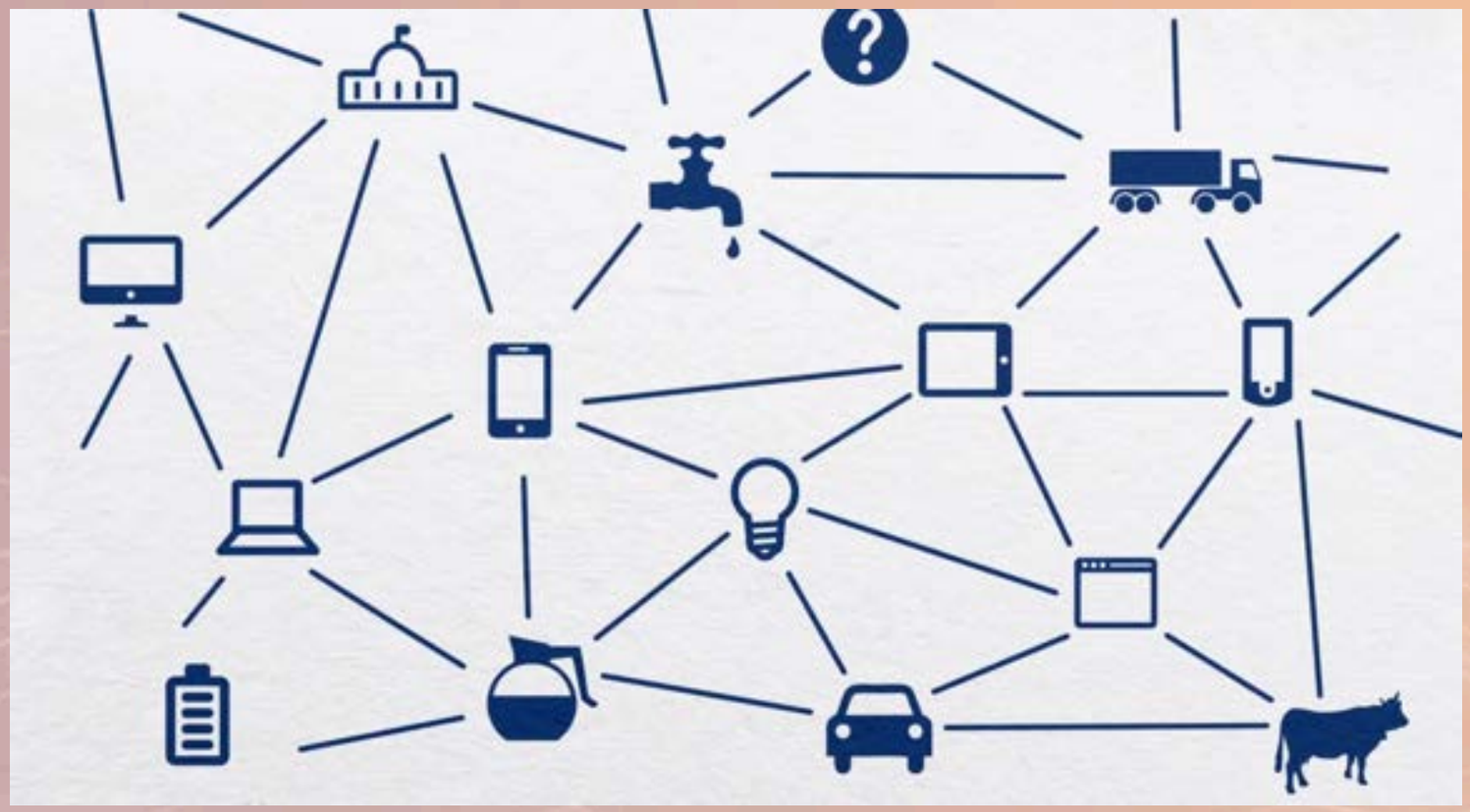
# Flask and MongoDB

Flask Rest API with MongoDB

# Flask and Authentication

HTTP Basic Authentication

# Closing



VirginiaTech
Invent the Future

# Networking Tools

## ECE 4564 - Network Application Design

**Dr. William O. Plymale**

ece
BRADLEY DEPARTMENT
of ELECTRICAL & COMPUTER ENGINEERING

# Topics

- Unix Network Commands
- Network Tools
- Python Network Code

# Unix Network Commands

- ping
- netstat
- nmap
- netdata
- tcpdump

# Linux Howto's

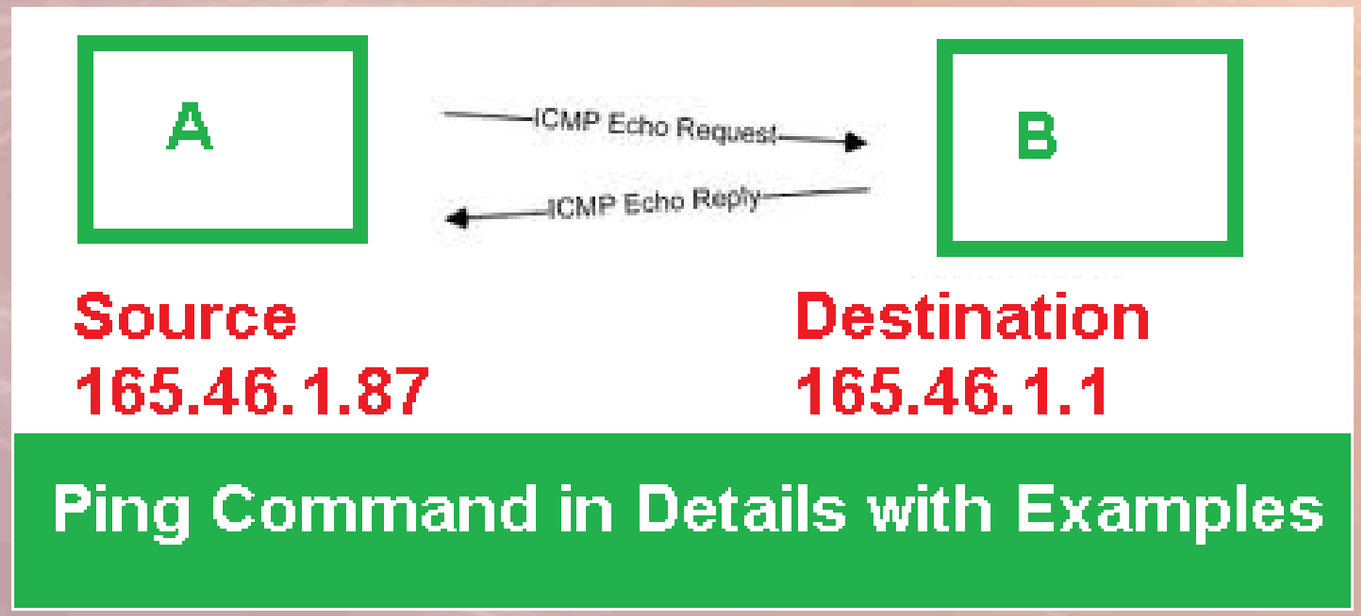
Tecmint

Linux Network Config and Troubleshooting

# ping

Ping is a computer network administration utility used

- To test the reachability of a host on an Internet Protocol (IP) network
- To measure the round-trip time for messages sent from the originating host to a destination computer
- Name comes from active sonar terminology which sends a pulse of sound and listens for the echo to detect objects underwater



# netstat

netstat (network statistics) is a command-line tool that

- displays network connections (both incoming and outgoing)
- routing tables
- network interfaces
- network protocol statistics

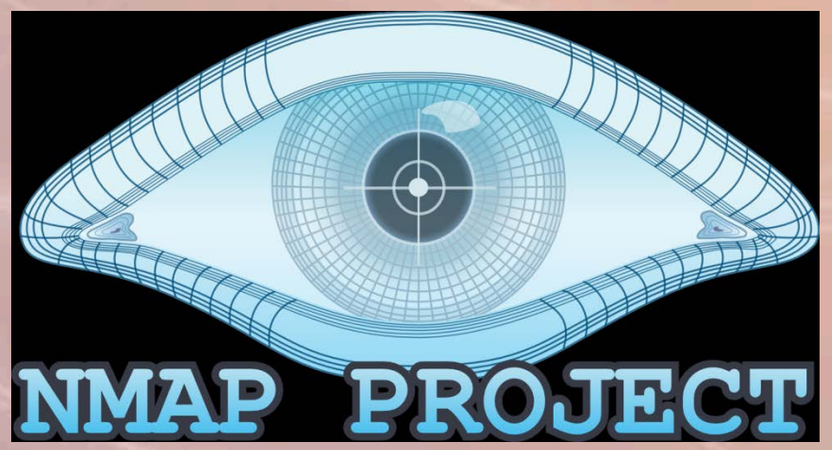

http://www.tecmint.com/20-netstat-commands-for-linux-network-management/

# nmap

Nmap ("Network Mapper") is a free and open source (license) utility for network discovery and security auditing.

Useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services those hosts are offering, what operating systems they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics




http://nmap.org/

VirginiaTech
Invent the Future

# netdata

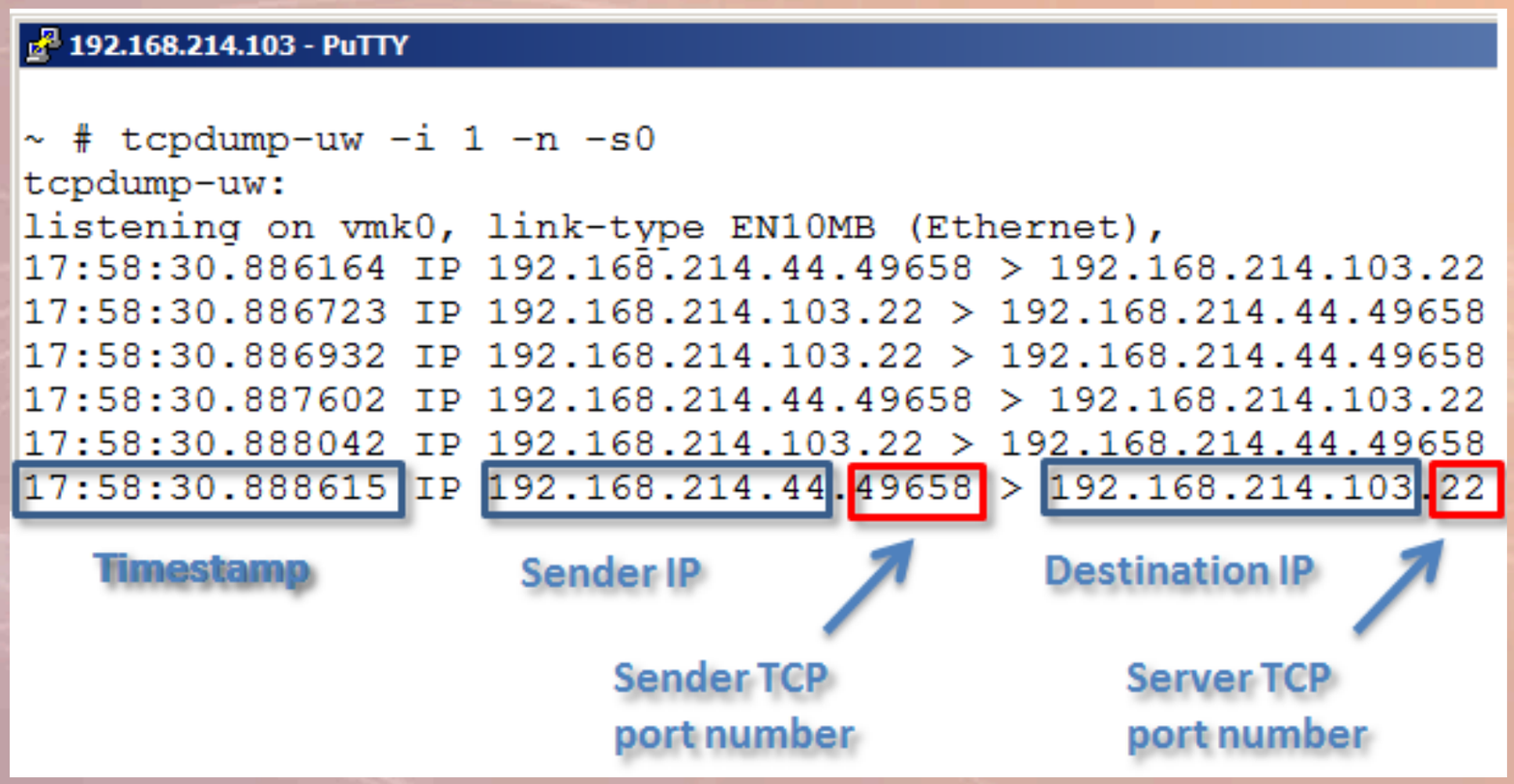
A Real-Time Performance Monitoring Tool for Linux Systems

http://www.tecmint.com/netdata-real-time-linux-performance-network-monitoring-tool/

ece
BRADLEY DEPARTMENT
of ELECTRICAL & COMPUTER ENGINEERING

# tcpdump

- tcpdump is a common packet analyzer that runs under the command line.
- It allows the user to intercept and display TCP/IP and other packets being transmitted or received over a network to which the computer is attached.




tcpdump

# tcpdump

tcpdump -s 0 port ftp or ssh -i eth0 -w mycap.pcap

In above command

- -s 0 will set the capture byte to its maximum i.e. 65535, after this capture file will not truncate.
- -i eth0 is using to give Ethernet interface, which you to capture. Default is eth0, if you not use this option.
- port ftp or ssh is the filter, which will capture only ftp and ssh packets.
- -w mypcap.pcap will create a pcap file

pcap files are data files created using the program and they contain the packet data of a network.

# Unix Network Tools

Wireshark

Top 20 Free Network Monitoring and Analysis Tools for Sys Admins

# Wireshark

Wireshark is a free and open-source packet/protocol analyzer.

https://www.wireshark.org/

It is used for network troubleshooting, analysis, software and communications protocol development, and education.

Wireshark is cross-platform, running on GNU/Linux, OS X, BSD, Solaris, some other Unix-like operating systems, and Microsoft Windows.

There is a terminal-based (non-GUI) version called TShark.

Wireshark is very similar to tcpdump, but has a graphical front-end, plus some integrated sorting and filtering options.

# Wireshark

Wireshark is software that "understands" the structure (encapsulation) of different networking protocols.

It can parse and display the fields, along with their meanings as specified by different networking protocols.

Wireshark uses *pcap* to capture packets, so it can only capture packets on the types of networks that *pcap* supports.

Data can be captured "from the wire" from a live network connection or read from a file of already-captured packets.

# Wireshark

Live data can be read from a number of types of network, including Ethernet, IEEE 802.11, PPP, and loopback.

Captured network data can be browsed via a GUI, or via the terminal (command line) version of the utility, TShark.

Captured files can be programmatically edited or converted via command-line switches to the "editcap" program.

Data display can be refined using a display filter.

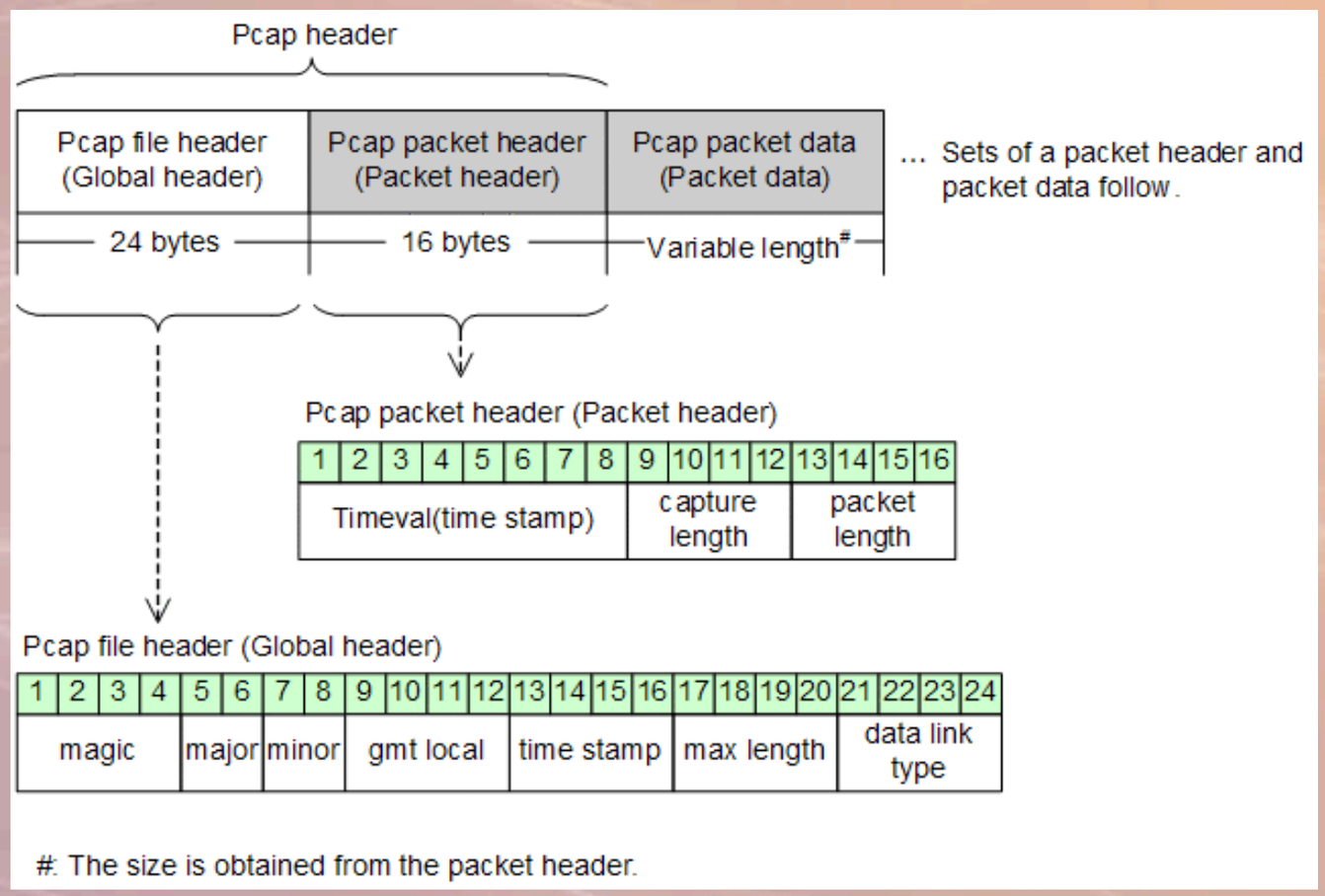
Plug-ins can be created for dissecting new protocols.

Wireshark is perhaps one of the best open source packet analyzers available today for UNIX and Windows.

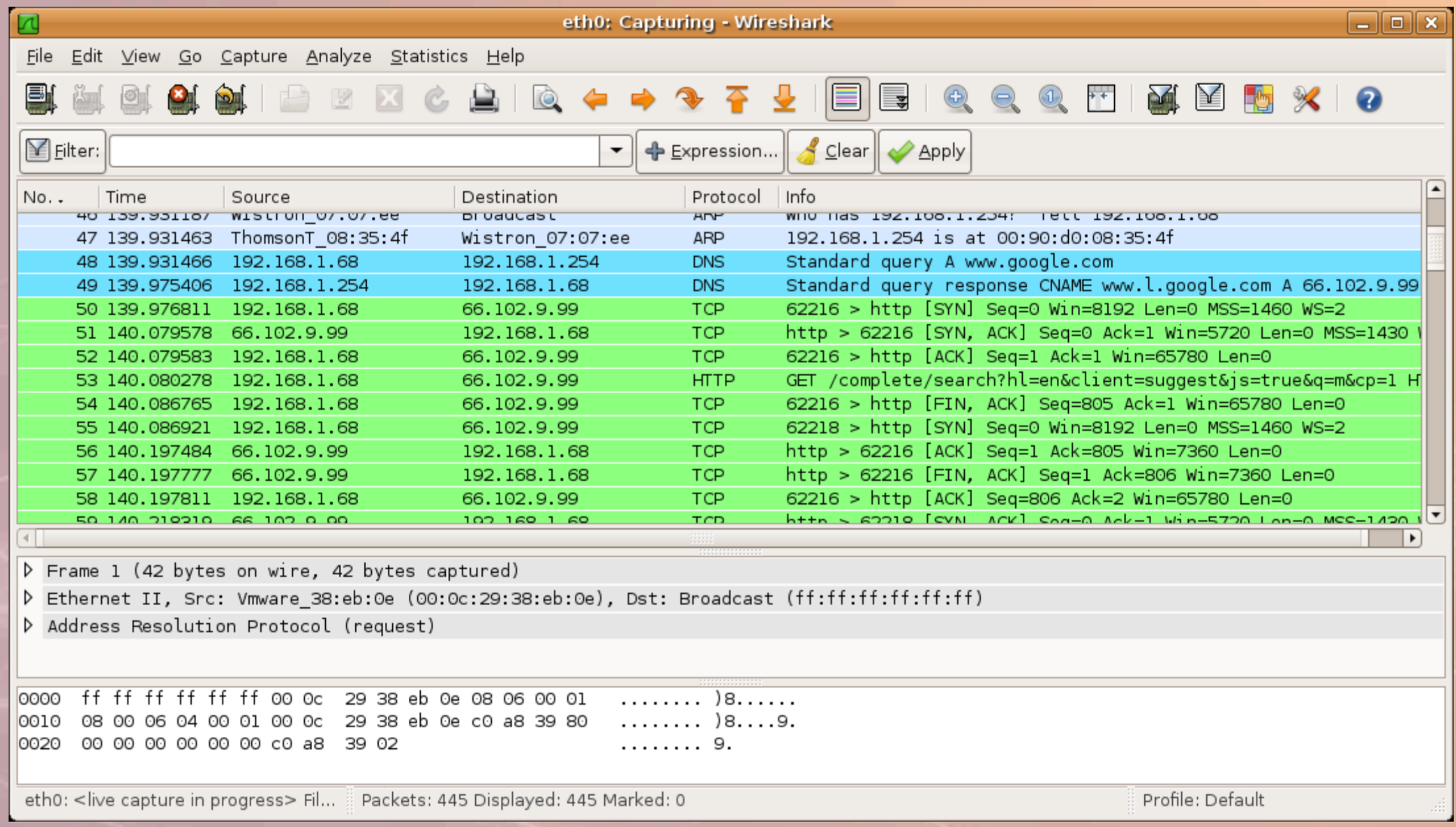
# pcap

pcap (packet capture) consists of an application programming interface (API) for capturing network traffic

Unix-like systems implement pcap in the libpcap library
Windows uses a port of libpcap known as WinPcap.

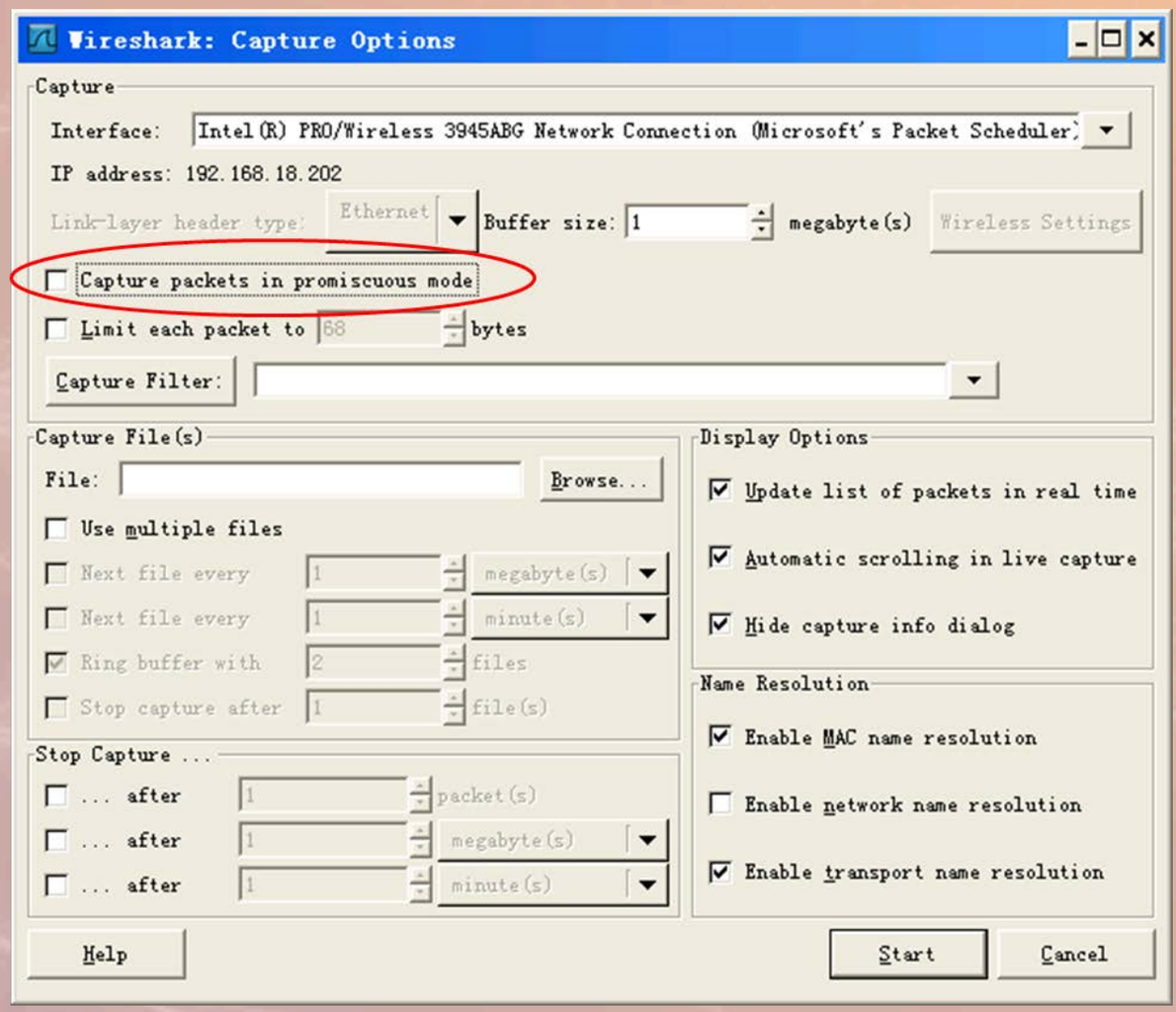


# Wireshark

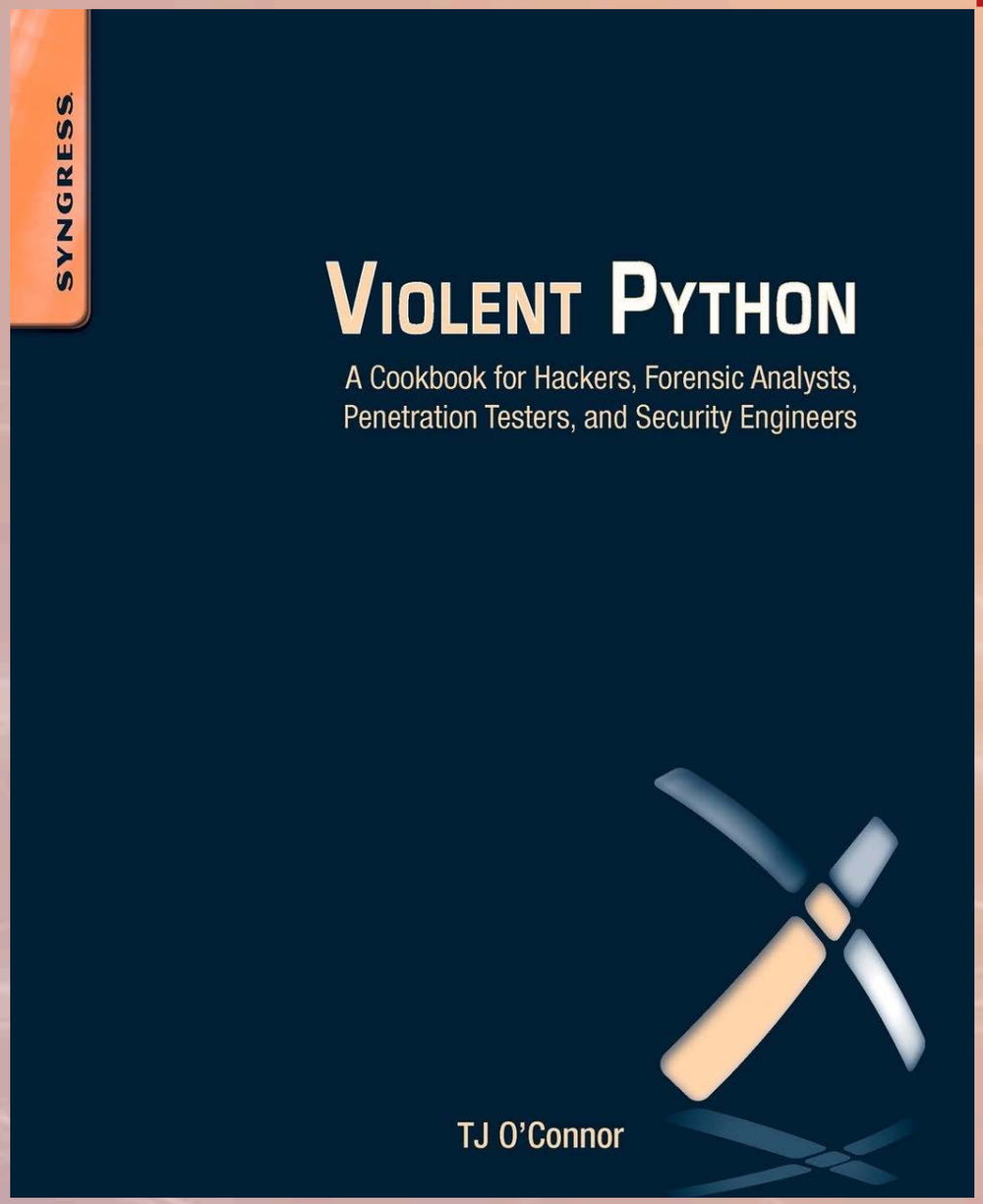

eth0: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Info
47 139.931463 ThomsonT_08:35:4f Wistron_07:07:ee ARP 192.168.1.254 is at 00:90:d0:08:35:4f
48 139.931466 192.168.1.68 192.168.1.254 DNS Standard query A www.google.com
49 139.975406 192.168.1.254 192.168.1.68 DNS Standard query response CNAME www.l.google.com A 66.102.9.99
50 139.976811 192.168.1.68 66.102.9.99 TCP 62216 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
51 140.079578 66.102.9.99 192.168.1.68 TCP http > 62216 [SYN, ACK] Seq=0 Ack=1 Win=5720 Len=0 MSS=1430
52 140.079583 192.168.1.68 66.102.9.99 TCP 62216 > http [ACK] Seq=1 Ack=1 Win=65780 Len=0
53 140.080278 192.168.1.68 66.102.9.99 HTTP GET /complete/search?hl=en&client=suggest&js=true&q=m&cp=1
54 140.086765 192.168.1.68 66.102.9.99 TCP 62216 > http [FIN, ACK] Seq=805 Ack=1 Win=65780 Len=0
55 140.086921 192.168.1.68 66.102.9.99 TCP 62218 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
56 140.197484 66.102.9.99 192.168.1.68 TCP http > 62216 [ACK] Seq=1 Ack=805 Win=7360 Len=0
57 140.197777 66.102.9.99 192.168.1.68 TCP http > 62216 [FIN, ACK] Seq=1 Ack=806 Win=7360 Len=0
58 140.197811 192.168.1.68 66.102.9.99 TCP 62216 > http [ACK] Seq=806 Ack=2 Win=65780 Len=0
Frame 1 (42 bytes on wire, 42 bytes captured)
Ethernet II, Src: Vmware_38:eb:0e (00:0c:29:38:eb:0e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)
0000 ff ff ff ff ff ff 00 0c 29 38 eb 0e 08 06 00 01 ........ )8......
0010 08 00 06 04 00 01 00 0c 29 38 eb 0e c0 a8 39 80 ........ )8....9.
0020 00 00 00 00 00 00 c0 a8 39 02 ........ 9.
eth0: <live capture in progress> Fil...
Packets: 445 Displayed: 445 Marked: 0
Profile: Default

# Configuration

This checkbox allows you to specify that Wireshark should put the interface in promiscuous mode when capturing. If you do not specify this, Wireshark will only capture the packets going to or from your computer (not all packets on your LAN segment).



# Python Network Apps




https://github.com/shadow-box/Violent-Python-Examples

# Scapy

Scapy is a powerful interactive packet manipulation program.

- able to forge or decode packets of a wide number of protocols
- send packets on the wire
- capture packets
- match requests and replies
- can handle most classical tasks like scanning, tracerouting, probing, unit tests, attacks or network discovery
- can also send invalid frames or inject your own 802.11 frames,

http://www.secdev.org/projects/scapy/

# Scapy

"The Very Unofficial Dummies Guide to Scapy"

Adam Maxwell

## Installation

1. Install Python 2.5+
2. Download and install Scapy

sudo apt-get install python-scapy

3. (Optional): Install additional software for special features.

apt-get install tcpdump graphviz imagemagick python-gnuplot python-crypto python-pyx

4. Run Scapy with root privileges.

https://theitgeekchronicles.files.wordpress.com/2012/05/scapyguide1.pdf

# Scapy

Welcome to Scapy (2.2.0)

>>> send(IP(dst="127.0.0.1")/ICMP()/"HelloWorld")

Sent 1 packets.

>>>

**send** - this tells Scapy that you want to send a packet (just a single packet)

**IP** - the type of packet you want to create, in this case an IP packet

**(dst=”127.0.0.1”)** - the destination to send the packet to (in this case my router)

**/ICMP()** - you want to create an ICMP packet with the default values provided by Scapy

**/”HelloWorld”)** - the payload to include in the ICMP packet (you don’t have to provide this in order

for it to work.

VirginiaTech
Invent the Future

# Scapy

Scapy Basics

# Scapy

"Packet Wizardry Ruling the Network with Python"
Rob Klein

Scan an entire C-Class network for all hosts running that have port 80 listening.

```
p=IP(dst="hackaholic.org/24")/TCP(dport=80, flags="S")
sr(p)

results = _[0]

for pout, pin in results:
...    if pin.flags == 2:
...        print (pout.dst)
```

# Scapy

Created a packet which was sent to the /24-subnet that hackaholic.org is connected to and set the TCP header to destination port 80 and the SYN flag.

The SYN flag is used to initiate a connection.

A reply of SA (SYN/ACK) means the port is listening, a RA (RESET/ACK) means it is closed, and finally no response means the host is down or filters packets.

After constructing the packet, Scapy emits the packets.

The results are then dissected in the for-loop and the destination IP addresses of hosts that replied SA are listed.

# Closing