LIT Documentation

Release 1.1

Weqaar Janjua, Ahmer Malik

February 05, 2016

CHAPTER

ONE

WEEK-1 LINUX IOT TRAINING

1.1 Task - 1: Battery Calculation

Minimum components required to run a MCU board like Raspberry Pi.

Power consumption for of each component in full working mode.

We have 2500 mAh battery. How many days/hours the battery will take to discharge in full working mode of board.

How to optimize the battery.

How we can convert the 5V into 3.3V. Name that component or IC.

Name a communication device that can be used on the design board.

1.2 Task - 2: IEEE-754 Floating Point Standard (Refer: Number_Systems_Exercise.pdf -Question 4)

Learn the algorithm that how a floating point is saved.

Code: write a C/C++ code that can present the whole algorithm of IEEE-754.

Code for IEEE-754 can be seen or downloaded from github

1.3 Task - 3: CRC & Checksum

Read and Implement CRC and checksums. Why they are used.

1.4 Task - 4: Vim Editor

Full command on the Vim Editor. Be full aware of all the utilities that we usually use in MS Word in order to manipulate data in a MS file (i.e copy, cut, paste, select, find etc)

1.5 Task - 5: Signal Generator & Oscilloscope

Use siggen and xoscope for waveform generation using the sound cards and display it on the screen in bash.

1.6 Task - 6: Special Characters

We should be well aware of meanings of special characters in bash/Linux in order of scripting.

1.7 Task - 7: BASH HOW TO

How to use bash. Bash full command is necessary to move forward in embedded systems.

You can follow Bash How to recent pdf.

1. Redirection (STDIN, STDOUT, STDERR) [0 1 2]
2. Make your own output file descriptor with fd=3. It is used when 1 process writes to a pipe and another process reads to it later on.
3. Pipes are important.
4. Conditional Theory
5. Loops

CHAPTER

TWO

WEEK - 2 LINUX IOT TRAINING

2.1 Task - 1: Signals and Traps

Learn different types of signals in Linux. Figure out which signals cannot be trapped. The trap-able signals were caught and performed a certain action on trapping.

Code: Write a shell script that can continuously printing a "Hello-World" while it has not been caught or trapped by any or the terminate signal.

check shell script click here

2.2 Task - 2: Process Scheduling

How to schedule a process that it can run on specific times and days. Learn the man page of Crontab command in Linux.

Code: Write a shell script that prints your name 100 times in .txt file specifically on every Friday. The process should execute in background.

Note: Read about special character "&" in bash

2.3 Task - 3: IPC (Pipes and FIFO's)

Read about Pipes and FIFO, how they work in IPC's. The basic commands in bash for using them.

Code: Write a shell script to make your own output file descriptor with fd=3. It is used when 1 process writes to a pipe and another process reads to it later on. Check shell script click here

2.4 Task - 4: Job Control Commands

What is job. What is the difference between job and process. Learn the basic commands in bash that are used for job control.

Practice these commands in various scenarios.

2.5 Task - 5: Message Queues

Read about the message queues that for what purpose they are use. Read the man pages for the different commands that are used for handling messages like mq_send, mq_receive.

2.6 Task - 6: Shared Memories

Learn about the benefits of shared memory and the what logical errors and flaws might cause when different processes are using same shared memory simultaneously. What techniques are used to avoid the these errors.

Code: Write a code to create a shared memory. A process should write a complete alphabets from A – Z on this shared memory and waits for the other process to write '*' on its 1st entry to complete its execution. Like a server is waiting for a message from client to perform a certain task.

Check shell script click here

2.7 Task - 7: Process & Process Management

Learn about the process what it is? How an OS identify a process. How different processes are handled by OS ? What is priority of a process? How CPU deals with the process based on their priorities. Learn all the concepts related to process in OS. (Recommended books by instructor).

Practice: All the commands related to process and priorities under different scenarios.

2.8 Task - 8: Fork and Exec

How a process can start a new process. This new process is called the child process. What is the difference between exec and fork command.

Code: Write a code that uses fork and exec commands. It clears the main difference between both. Sketch down the situations which exec command is useful and where fork is better to use.

2.9 Task - 9: Semaphores

What are semaphores. What is their advantage and why they are used for process handling in the scenarios where different process has access to the shared resources.

Code: Write a code that if a process P1 is using the shared resource, no other process or P2 could not use that resource until P1 has not allowed.

2.10 Task - 10: Case Study (Mars Rover)

What happened to the MARS ROVER on the mars in its early stages. What was the fault occurred that forces engineers to rethink. State that fault. How they have fixed it out?

Code:

Part - 1: (Priority Inversion): Mimic the same situation myself that how I can code that faulty program to see, can it really malfunctioned a system and force it to restart. How that unexpected restart would have lost a lot of useful data or information.

Part - 2: (Priority Inheritance): Now solve the problem as it was solved for Mars Rover.

To check c codes for Semaphores, Priority settings, Priority Inversion, Priority Inheritance click here

2.11 Task - 11: Virtual Memory Management

Learn what is the benefit of virtual memory. Why the OS's use virtual memory. What is Paging? How virtual address is mapped into physical address. Learn about the page fault, page tables. What is the disadvantage of long page tables. How this issue can be resolved up to reasonable extent.

2.12 Useful Resources:

Message-Queues

http://menehune.opt.wfu.edu/

http://linux.die.net/man/7/mq

https://www.cs.cf.ac.uk/Dave/

FIFO/Pipes

https://linuxprograms.

Process Scheduling (Crontab)

http://www.cyberciti.biz/faq/

Process and Process Management

Read chapter 1.1 of Linux Performance and Tuning Guidelines

Chapter 5 - Process Management of Book: linux-system-

/proc (Virtual file system):

http://www.tldp.org/LDP/Linux-

Difference between bin and sbin

http://blog.taylormcgann.com/

MARS PathFinder

http://research.microsoft.com/

http://research.microsoft.com/

Shared Memory

https://www.cs.cf.ac.uk/Dave/

Fork()

http://www.csl.mtu.edu/cs4411.

Dont use printf for fork(). use write instead.

SSH

http://www.linuxhowtos.org/

http://www.slashroot.in/

Virtual Address to Physical Address

http://www.cs.umd.edu/class/

CHAPTER

THREE

WEEK - 3 LINUX IOT TRAINING

3.1 Task - 1: SSH and SFTP

Learn why and how to use ssh and sftp in Linux. What is the difference between ssh and sftp. How to manipulate data between host and client. After remotely login into the server, how we can do download data from there using get command.

3.2 Task - 2: Kernel Virtual Machines (KVM's)

What is a KVM and how virtual machines are beneficial. Build a KVM using Python VM builder and Virt-Mnager. The guest should be on the same OS as host is (in my case Ubuntu 14.04.3) .

How to make virtual machines follow this

3.3 Task - 3: IP Addresses and Subnets

What is an IP address. What is difference between IPV4 and IPV6. Why IPV6 was necessary to introduced. What each octet represents in an IP address. What are subnets and learn about the VLSM and CIDR and present back to the instructor.

3.4 Task - 4: IP Addressing Case

Let's assume that IoT Systems has 10 departments. The list of departments are:

DEPARTMENTS	NO. OF EMPLOYEES
Software	112
Hardware	60
IT	40
Finance	06
Public Relations	02
Marketing	12
Business development	25
Quality Insurance	20
House Keeping	4
Admin	3

Each department should have its own subnet. The design of sub-netting should be wise, as the company is well developed and there will be chances to extend it up to little extent only (in term of employees).

Techknox Systems

IP Addresses

Department	Total Employees	IP Addresses	Subnetwork ID's (0xd)	Subnetwork ID's (0xb)	Total Hosts	Reserve Hosts	
			192.168.5.0/24				
Software	112	192.168.5.0	255.255.255.128	11111111.11111111.11111111.10000000	126	14	Host the IT (10)
Hardware	60	192.168.5.128	255.255.255.192	11111111.11111111.11111111.11000000	62	2	
IT	40	192.168.5.192	255.255.255.224	11111111.11111111.11111111.11100000	30	-10	Host by Software (10)
Finance	6	192.168.5.224	255.255.255.248	11111111.11111111.11111111.11111000	6	0	
Public Relations	2	192.168.5.232	255.255.255.252	11111111.11111111.11111111.11111100	2	0	
Marketing	12	192.168.5.236	255.255.255.240	11111111.11111111.11111111.11110000	14	2	
			192.168.6.0/24				
Business Development	25	192.168.6.0	255.255.255.224	11111111.11111111.11111111.11100000	30	5	
Quality Insurance	20	192.168.6.32	255.255.255.224	11111111.11111111.11111111.11100000	30	10	
House Keeping	4	192.168.6.64	255.255.255.248	11111111.11111111.11111111.11111000	6	2	Host the Admin
Admin	3	192.168.6.72	255.255.255.248	11111111.11111111.11111111.11111000	6	3	

Fig. 3.1: IP Addressing for IoT Systems.

3.5 Task - 5: Software Defined Network (SDN)

Now, we have a proper design of subnets for a company. Now, its time to simulate it on our PC. This task is divided into different subtasks:

1. Install 20 KVM's on your host (2 KVM's/Department). I have used different embedded .iso's (tiny core linux, snappy etc), but snappy suits best. Build 20 KVM having 128M RAM, 3.5 G Disk, 1 CPU.
2. Change their IP's according to sub-netting exercise that we have done in previous task. This directory (/etc/network/interfaces.d/eth0) is important for changing IP's.
3. Reboot all KVM's and check using ifconfig that all the desired changes have been saved or not. If not then might be some issue.
4. Now, install 21st KVM that will be act as Router. It should be running Ubuntu 14.04.3. Its RAM should be around 1-2 GB.
5. The router should have 10 ports for the communication through a switch to the 10 departments of company. Hence, here is the need to add 10 network interfaces in the Router VM that should be act as 10 ports of the router.

   Shutdown the router VM —> virsh edit router —> add 10 network interfaces here (.xml file) change: mac = 0x_____ , slot: each should have different slot.
6. After adding 10 NIC's, start the router domain. Check in host machine using ifconfig whether 10 macs are added or not.
7. **Now, its time to connect each router port to its corresponding subnet through a switch.** Learn brctl using man page. P1 (router) connects to N1 (Software –> Employ 1 (KVM-1) Employ 2 (KVM -2))
8. Learn the ping command. And check whether each guest of a Network can communicate to or listen to its corresponding Port on router.

In router: ping _IP (N1: E1)____

3.6 Task - 6: Bash Scripting of SDN Lab

All that has been done above should be scripted now. Think how we can efficiently start all the KVM's, check each KVM status. After successful booting of all VM's add the router ports and corresponding network VMS to same

switch (bridge).

CHAPTER

FOUR

WEEK - 4 LINUX IOT TRAINING

4.1 Task - 1: SDN - Scripting

As we have concluded our week-3 by establishing the IoT Systems Networks through commands and have to think how we can implement it using bash scripts. Why Bash Scripts?

Now, we have got that how the things are working, but did not done in professional way. Secondly, if we have to work on some different machine, having the VMS, then we have to perform all this procedure again. It sounds not good.

We have performed a task 1 time, then why we do the same again and waste time. Lets do it in professional way.

1. Create a .conf file that contains the names of domains (VM machines). Because every user or client doesn't have the same name as we used. The .conf should look like this

#VM-Domains.conf

```
HOSTS = Router_name,H1,H2, H3,H4,.  .  .
```

What is the benefit of using Router name at 1st position, we will discussed it later.

1. Read the all domain names and check its status whether it is ON or OFF? You can use (Hint: man virsh dominfo). Based on the state, you should take step to ON it.

#Boot.sh

- Start all VMS by reading names from .conf.
- Add the hosts and ports of router to corresponding switch/bridge.

For scripting you should have solid grip on the **"Regular Expressions"**. Because you have to extract out useful information from files, have to edit it and have to perform different functions.

Hint-1: have excessive use of "grep", "cut", "sed" commands. Do read them in detail.

Hint-2: /etc/network/ . . . This directory will be used many times.

1. Instead of giving full path to .xml files, use dumpxml (read it carefully, what it do) . Because, in different computer the path to .xml files might be different. And your script will not work properly.

You should write such script that can run on any machine. Not just on your own local.

1. Create the required switches and add corresponding VMS or host to it. For Example:

```
SW-1 = [H1,H2 and eth0 of router]
```

Hint: Read man *brctl* because you have create/delete bridges, add/delete interfaces.

1. Write shutoff.sh to off all the domains when need. Otherwise you will have to shutdown 20 VMS manually. (Its the power of scripting).
2. Now we have to test whether domains under same switch can communicate or not.

Hint: Ping command will help you a lot. Read it.

1. Using ping, the message received properly to the destination or not. It should work fine.

Debugging challenge: Now the Challenge begins, they are not communicating within the same bridge. Why? As everything is just looking fine from frontal view

4.2 Task - 2: Figure Out the Problem

Figure out, what is going wrong and where? Look into each KVM, their Macs, IP's (broadcast, net masks etc), vnets. Use ifconfig, read xml's. . . Its too irritating but we have to do it.

What was the problem —> After 1 day of debugging, we have found that when a VM starts it goes into default virbr0. We know it from the first very first day. But, we were doing was that we deleted it from virbr0 and add it in SW-1 / SW-2 etc, their macs somehow changes. And the change was very minor, only 1st octet, whose consequence was that, machine pings to some unknown mac address. Hence, host became unreachable.

4.3 Task - 3: Fix It Now

I think its obvious now, what will be the next task. "How to Fix it". The solution might be that somehow we edit the .XMLs of VMS in a way that our domain automatically added to bridges (SW-1, 2) etc. despite of adding to virbr0.

We edit the XMLs in following fashion:

</controller>

<interface type='network'> <mac address='52:54:00:0c:cf:7c'/> <source network='default'/>

</controller>

<interface type='bridge'> <mac address='52:54:00:0c:cf:7c'/> <source bridge='SW-1'/>

It works fine, now each domain/NIC automatically added to its corresponding switch/bridge. We also configure the IP's and net masks of switches to an un-useable IP addresses in order to ssh from host working machine (Ubuntu 14.04.3). Yes, all the things were working properly.

4.4 Task - 4: Inter-subnet communication

Uptil now, we have checked the the intra-subnet communication. Now check, is our script is working fine for inter-subnet communication.

Things to learn here: - ARP cache http://www.thegeekstuff.com/2012/01/arp-cache-poisoning/

- What is function of router http://www.thegeekstuff.com/2012/04/route-examples/
- What is routing table.
- What steps are involved for inter-subnet communication
- What are the chances of hacking

Hint: Before moving onward just ON the ip-forwarding on router VM.

```
Echo 1 > /proc/sys/net/ipv4/ip-forwarding
```

Check, now all subnets can communicate with each other subnet.

4.5 Task - 5: Different possibilities for communication in sub-netted network

4.5.1 Case - I: Intra-Subnet

- H1 (subnet-1) —> H2 (subnet-1) Communication through bridge
- Broadcast ARP request
- ARP reply by the required IP host H2 having its MAC address.

4.5.2 Case - II: Inter-Subnet But Different Switches

- H1 (subnet-1) SW-1 —> H2 (subnet-2) SW-2 Communication through Router
- Broadcast ARP request
- No reply

Handed Over to router (Gateway) ——> subnet mask & IP (See Routing Table) —-> Get network interface (eth-1) —-> SW2 (Broadcast and get reply)

4.5.3 Case - III: Inter-Subnet But Single switch

- H1 (subnet-1) SW-1 —> H2 (subnet-2) SW-1 Communication not possible
- Broadcast ARP request (127) but didn't reach t0 H2 (130)
- No reply

4.5.4 Case - IV: Inter-Subnet But Single switch (possible by hacker)

- H1 (subnet-1) SW-1 —> H2 (subnet-2) SW-1 Communication possible
- Hacker change the IP and net mask as that of subnet-1 (say 124 – free host IP)
- Broadcast ARP request (127) but reach to H2 (124)
- Reply

4.6 Task - 6: Inter-Router Communication

Uptil now, we have only 1 router with 12 interfaces, means 10 departments or 20 hosts are handled by only one router.

Now in order to learn the inter-router communication, split the network into 2 parts.

1. 5 switches should be handled by Router-1 on first PC. 5 switches should be handled by Router-2 on 2nd PC.
2. **Both the PC's are interconnected using LAN cable.**

   The hierarchy will be like: Router-1 —-> EDGE_SWITCH_1 —-> eth0 —-> eth0—-> EDGE_SWITCH_2 —-> Router- 2 (192.168.7.1) (192.168.7.2) (192.168.7.3) (192.168.7.4)
3. Now copy the desired KVM's .xml's and .img's from PC-1 to PC-2 using SCP command.

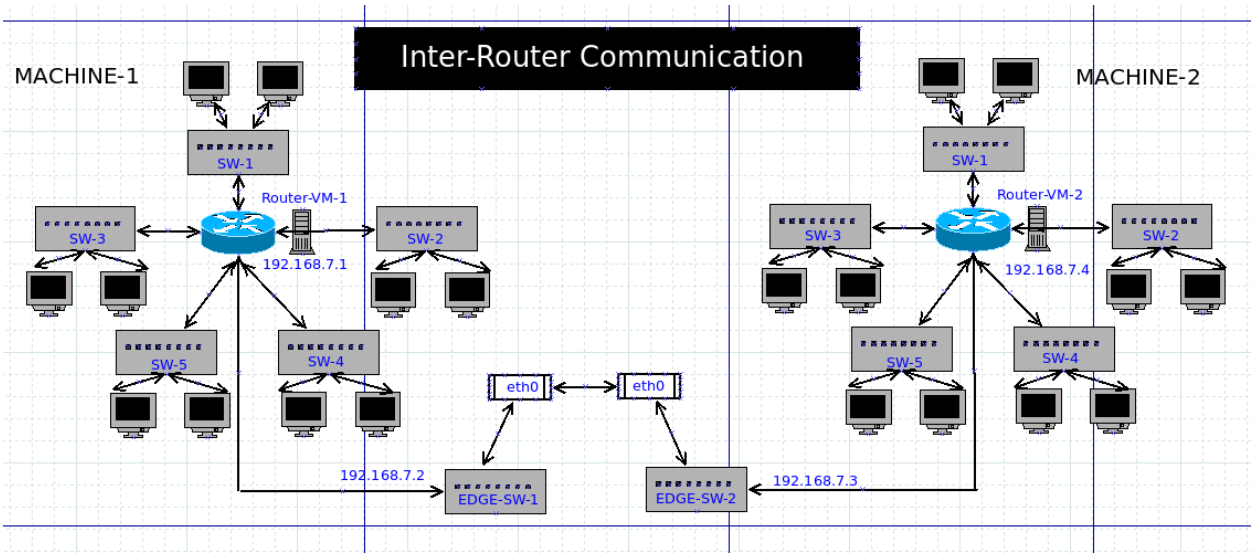


Fig. 4.1: **Inter-Router Communication Architecture**

SCP -C _____ (because -c will increase the speed up to a reasonable extent).

4. Now, we have to work carefully. Because, the enviroment in the .xlm's was according to the PC-1. Now, we have to edit its path of .img (where domain.img is copied).

Steps to run a KVM that was running in 1 PC to run it on another PC:

- virsh define domain.xml The line that can cause error are most likely to be following in .xml file. Do concentrate on it.

```
<type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type>
```

Just check that PC-2 should have same architecture. Machine variable should be supported otherwise replaced by a supported one. In my case supported machine was not present in PC-2.

I got this error:

error: internal error: process exited while connecting to monitor: qemu-system-x86_64: -machine pc-i440fx-trusty,accel=kvm,

Use -machine help to list supported machines!

Therefore, I replaced the variable with `machine='pc-i440fx-2.0'`

```
<emulator>/usr/bin/kvm-spice</emulator>
```

Must check the kvm-spice binary should installed on PC-2. Otherwise install it or replace it with installed one.

I have replaced the binary

```
<emulator>/usr/bin/kvm</emulator>
```

```
<source file='/home/ahmer/VMS/web-server/ubuntu-kvm/tmpgStITZ.qcow2'/>
```

Must check your path for the desired .qcow2 or .img

- Virsh create domain.img

Now, you are able to use your KVM.

1. **Now, SW-1 ——— SW-5 are handled by Router-1 and SW-6 ——- SW10 are handled by Router-2.** You can easily ping between the hosts of same router and but not to the host of other router.

2. **How we can make this communication possible? Hint:** Learn routing tables and route -n

   As communication between subnets are done based on the routing table. If you check the routing table in either machine, you will not find the IP addresses of the other router subnets of course. You have to add them manually.

Let's move step by step.

- Check that can you ping 192.168.7.1, 2, 3, 4 from either side? If yes then nice, because you have a successful connection between both the PC's. If not, then problem is that your Ethernet ports are interconnected, hence, no path to access the hosts of other router.
- Now, learn this concept that in order to access PC-2 from PC-1 your gateway is 192.168.7.4, and 192.168.7.1 vice versa.

1. Add 5 routes of Router -2 on Router-1 as follows:

---

route add -net 192.168.5.236 netmask 255.255.255.240 gw 192.168.7.4

route add -net 192.168.6.0 netmask 255.255.255.128 gw 192.168.7.4

route add -net 192.168.6.32 netmask 255.255.255.128 gw 192.168.7.4

route add -net 192.168.6.64 netmask 255.255.255.128 gw 192.168.7.4

route add -net 192.168.5.72 netmask 255.255.255.128 gw 192.168.7.4

Have you checked that all have same gateway. Because, gateway address must present in the broadcast of router-1

2. Similarly, add the routes on the Router-2, all have the gw=192.168.7.1

Now, check that we should be able to communicate over the LAN. I have first time, because after copying the router KVM, I have note change the MACs of the NIC's of router-2. Hence both of the routers were working with same MACs that results in no communication. One more thing, we should kept in mind, IP forwarding variable should be 1, because it can change, when you reboot the Router.

The solution was that change the MACs of Router 2 and must be unique. Check IP forwarding and setting it permanently 1 even after rebooting the KVM.

```
Sysctl -a | grep forward
```

```
vim /etc/sysctl.conf & set IP-forward =1
```

Now, you are able to communicate.

Practice: Figure out all the variables that are changed automatically when you reboot the system. Add them to rc_local or some other script that runs automatically when a system is booted.

CHAPTER

FIVE

WEEK - 5 LINUX-IOT TRAINING

5.1 Task - 1: Physical Layer

Learn Physical layer (L-1) of OSI model In detail.

- What services are provided by physical layer.
- What protocols are used by physical layer. (Ethernet, Blue-tooth, USB, Controlled Area Network etc)

5.2 Task - 2: Ethernet

- Most used topologies: Bus, Star
- Ethernet is passive (What means by passive).
- Ethernet used Broadcast topology with baseband signaling.
- Control methods used by Ethernet is CSMA/CD. Learn about CSMA/CD.
- What is back-off mode.
- What is broadband. What is Passive Hub. Jam Signal.
- Why CSMA/CD is not used in 802.11 standard? (Hint: half duplex mode, Hidden Nodes)

5.3 Task - 3: Ethernet Frame Format – 802.3 Standard

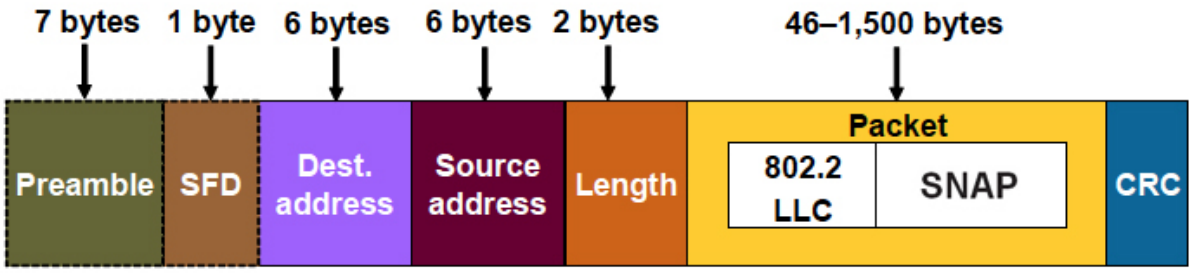


Fig. 5.1: Ethernet Frame 802.3 Standard

What is the function of each field in the frame .

Preamble	Start of new frame & Synchronization conditions.
Start Frame Delimiter	Always 10101011 (171).
Length	Total number of bytes in data field
Data	3-bytes = 802.2 = LLC = DSAP (Destination Service Access Protocol):SSAP:Control (Type of LLC frame it is.)
CRC	Error Detection function

5.4 Task - 4: Link Layer

Learn Link Layer (L-2) of OSI model in detail.

Services

Framing	The structure of frame depends on the link layer protocols. (i-e Ethernet)
Link Access	MAC protocol specifies the rules by which frame is transmitted onto the link. Sender sends frame whenever the link is idle.
Reliability	
Error Detection and Correction	CRC & Checksum

Where is the link layer implemented?

Link layer is implemented in link layer controller of NIC (Network Interface Card), a special purpose chip - hardware that performs the desired services as we discussed above.

Parity Checks

What are 2-dimensional parity checks. Why it is useful over the 1-D parity checks.

Checksum & CRC

We have done it in already previous weeks.

Types of Network Link Protocols

PPP – Point-to-Point Protocol

Protocol for communication between only 2 hosts over a common link.

Broadcast Link Protocol

Protocol for communication between different hosts over a shared link. Multiple Access problem mightbe caused.

Channel Partitioning protocols

- FDMA
- TDMA
- CDMA

Random Access Protocols

- Aloha
- Slotted Aloha

Reference: Computer Networking A Top Down Approach

CHAPTER

SIX

WEEK - 6 LINUX IOT TRAINING

For better understanding OSI model, read the OSI Model Analogy

6.1 Task - 1: Basic Overview on TCP-IP Protocol Suite

TCP-IP is considered as 4 layer system.

Application	Telnet, FTP, e-mail, etc.
Transport	TCP, UDP
Network	IP, ICMP, IGMP
Link	device driver and interface card

Link Layer: Media Details

Hardware details for physically connecting or interfacing with cable or any other media.

Network Layer

Routing or forwarding of packets intra or inter networks.

Transport Layer

Control flow of data between two hosts. The major difference between network layer and transport layer is that transport layer decide how and what data has to be transmitted (connection or connectionless) while network layer decide which route should be taken to deliver the message to destination.

What is Internet?

Internet is collection of different networks that uses the same protocol suite. Router is a special purpose hardware device that is used to connect two or more networks. Hence, Internet is totally inter-connected using routers.

What is the worth of Internet? Why a whole country uses the same network doesn't make sense?

If there was only one network in Pakistan or any other country, then all the companies will have a central administration that can make too mess and security problems. Now, each company has its own network, and has administration rights as well.

Routers

One special thing about routers that they provide connections to many different type of physical networks e.g. Token Ring, Ethernet etc. The Figure below explains it up to reasonable extent.

One of the goal of Internet is to hide the all the details of physical layout of Internet from applications.

Bridges

Another way to connection is bridge. Bridge is link-layer device while the router is network-layer device. Bridges make multiple LAN's appear to the upper layer as single LAN.

Encapsulation

An application sends data using TCP, the data is sent down the protocol stack. Each layers adds its information to the data in the form of headers. The unit of data that TCP sends to IP, is called TCP segment. IP sends to network interface (link layer), is called IP datagram. The stream of bits that flow across the Ethernet, is called frame.

A physical property of Ethernet frame is that its data size must between 46 – 1500 bytes.

DATA-LINK LAYER

Purpose

1. Send or receive IP datagrams for IP module.
2. ARP request and reply
3. RARP requests and reply

ARP and RARP is basically the map between the 32-bit IP address and 48-bit MAC address.

6.2 Task - 2: NETWORK LAYER

Every device that communicate over a network has an logical address. It is called IP address or layer-3 address. This address should be unique because it is the identity of a user in a particular network. Hence Network layer has following jobs:

1. Logical Addressing
2. Forwarding and Routing
3. Datagram Encapsulation
4. Fragmentation and Reassembling

The normal size of IP header is 20 bytes.

6.2.1 IP Header (Big Endian Ordering)

Note: For machines that use little endian format, should convert headers in network byte order (command: htons in c).

4B - 0		
IP version	0:3	4
Header Length	4:7	maximum (4*15 = 60 bytes) Normal (4*5 = 20)
Type of Service	8:15	0x00 (Delay, Throughput , Reliability Services)
Total Datagram Lentgh	16:31	Data + Header
4B - 1		
Identification	0:15	Number each datagram, usually increments during fragmentation
Flag	16:18	0 normally
Flag Offset	19:31	0
4B - 3		
Time To Live	0:7	Set upper limit on routers through which datagram can pass. It decrements at every router and datagram thrown away when counter reaches to 0 and sender is notified by an ICMP message.
Protocol	8:15	For demultiplexing at destination, what type of data
Header Checksum	16:31	Checksum calculated for the Header, stored here
4B - 4		
Source IP	0:31	
4B – 5		
Destination IP	0:31	
4B – 6		
Options		

i. Options

Security and Handling Restrictions – Military Applications

ii. Record Route – Each router record its IP address

Ping gives us an opportunity to look at the IP record route (RR) option. Most versions of ping have -R option to enable the record feature. This causes every router that handles the IP datagram (contains ICMP echo request message) to add its IP address to a list in the options. When datagram reaches to final destination, a list IP addresses should be copied into the outgoing ICMP echo reply, and all routers on the return path add their IP address into the list. When PING receives echo reply, prints the list of IP addresses.

Problem

The biggest problem in using this feature is that we have limited room in IP header for list of IP addresses. A maximum header length can be (4 * 15 = 60 bytes). 20 bytes are reserved for normal IP header, 3 bytes for RR option overhead. Hence, 60 - 20 - 3 = 37 bytes remaining. Therefore, only 37/4 = 9 IP addresses can be stored in the IP list.

Code:

Option: Type. For RR code = 7

Len: Total number of bytes for RR option. In this case len = 39.

ptr: pointer → IP address (4,8,12, . . .)

iii. Time Stamp

Before going to next option. We should know what is source routing?

What is Source Routing?

Normally, IP routing is dynamic with each router making decisions about next hop, where to send the IP datagram. Applications have no control on this and are normally not concerned with it. The idea behind the source routing is, sender specifies the route.

iv. Loose Source Routing

IP datagram should at least traverse from the IP addresses specifies by the sender. But, it can also pass through other router between two specified IP addresses in the list.

v. Strict Source Routing

IP datagram could not pass through other routers.

Code

0x83 = loose source routing 0x87 = strict source routing

6.2.2 IP Routing

It is simple for a host, when destination is directly connected to the host (point-to-point link) or on shared network (Ethernet, Token ring etc). Datagram is sent directly to the destination. Otherwise, the host sends the datagram to default router, and router delivers the datagram to its destination.

In general, IP receives a datagram form TCP, UDP etc to send or from network interface (to forward). IP layer has a routing table in memory, that it searches each time it receives a datagram to send. When a datagram received from network interface, IP first checks, destination IP is its own IP address or IP broadcast address.

IP routing performs the following actions:

1. Search the routing table = = Network ID and Host ID (Complete destination IP). If found send packet directly to the next hop or directly connected interface. PP link are found here.
2. Search the routing table = = Network ID (Ethernet / Token ring).
3. Routing table = = Default.

6.2.3 ARP – Address Resolution Protocol

APP Cache: IP address → MAC Address

Ethernet frame (ARP) Type → 0x0806

6.2.4 Understanding: Network Layer

We have ended our last by developing strong concepts on physical and Link Layer. Now, learn the Network Layer in detail.

1. Basic jobs of network Layer
2. IP Datagram Layout
3. Strong Concepts on each field in IP header.
4. Normal Header Length of IP datagram
5. Options that can be used if desirable.
    - Record Route (with header formats)
    - Time Stamp

- Loose Source routing
- Strict Source Routing

6. What is difference between Forwarding and Routing.
7. Routing Tables and actions they perform
8. ARP – Address Resolution Protocol

6.3 Task - 3: Raw Sockets

What are raw sockets. Why they are used?

Raw sockets are low level sockets to send one packet at a time. All the protocol headers are filled by the user, instead of kernel, in order to get strong grip on each protocol header.

Write a raw socket program that fills the UDP/TCP segment and IP datagram according to our will and transmit over the network to a specific destination. After transmission, sniff that packet using Wire-shark, that same packet is received that we have transmitted over the network.

Helpful link: http://www.tenouk.com/Module43a.html

6.4 Task - 4: Transport Layer

Transport layer provides the logical communication. Logical communication means process to process as IP layer do communication host to host.

Analogy

Lets consider a company "IoT" Head Office with 100 employees in Dublin. A sub-office with 100 employees in Cape Town. Its time, when there was only post service for sending messages in form of letters. Suppose, every day each individual writes a message to an individual in other office. In both office Ali and Akbar respectively has responsibility to receive and send letters from courier service, handed it over to each individual and receive theirs to post.

TCP/IP Stack	Real-Life Stack
Application	Letters in envelopes
Processes	Each individual is a process
Hosts	Offices
Transport Layer Protocol	Ali & Akbar
Network Layer Protocol	Post Service

6.4.1 Transport Control Protocol - TCP

Analogy

Ali and Akbar are reliable resource, because they take time to analyze how to deliver the letters to the respective individuals without any mistake. Here, time is the constraint.

Networking

Reference: The protocols TCP/IP Volume-1 by Stevens – Chapter # 17, 18]

6.4.2 User Datagram Protocol - UDP

Analogy

Ahmad or Akram works in the absence of Ali and Akbar. They always remain in haste, and delivers letters without any analysis or plan. Hence, individuals get their messages instantly. But the drawback is that in haste they do mistakes in proper delivery.

Networking

UDP is connectionless and unreliable transport layer protocol. Its sends the datagram that application writes to IP layer, with no guarantee that it reaches the destination or not.

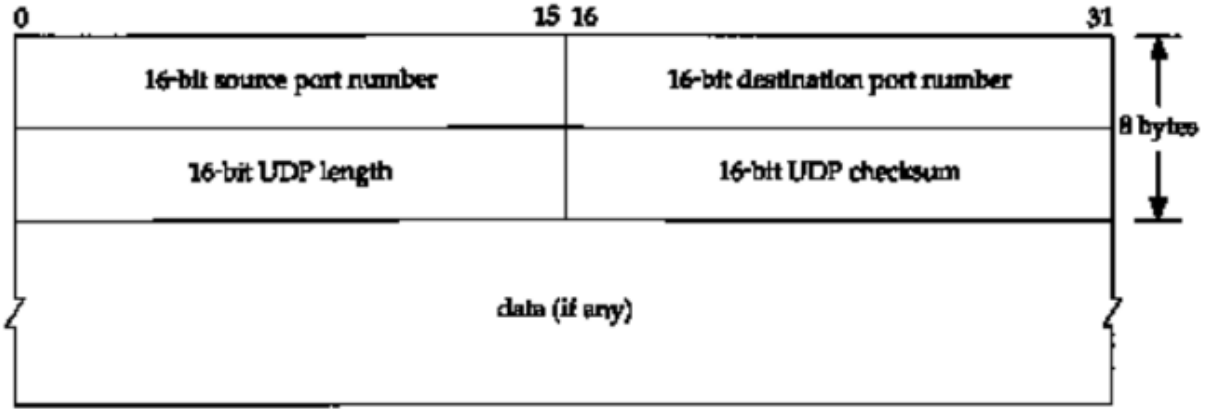


Figure 11.2 UDP header.

Fig. 6.1: UDP Header Format

Port numbers

Identify the sending and receiving processes. The TCP port numbers are independent of UDP port numbers. If a service can provide both TCP and UDP, port numbers can be same.

Length

UDP header length. The minimum value of this field is 8-bytes.

Checksum

UDP checksum field covers both UDP header and data unlike the checksum in IP datagram that only covers datagram header.

With UDP checksum is optional while with TCP checksum is mandatory.

Length of UDP datagram can be odd number of bytes. The solution is to append a pad byte of 0's to the end. For 16 bit word checksum calculation.

Both TCP and UDP include 12 byte pseudo header with UDP datagram, just for checksum calculation.

6.4.3 Understanding: Transport Layer

1. Relationship between Transport Layer and Network Layer.

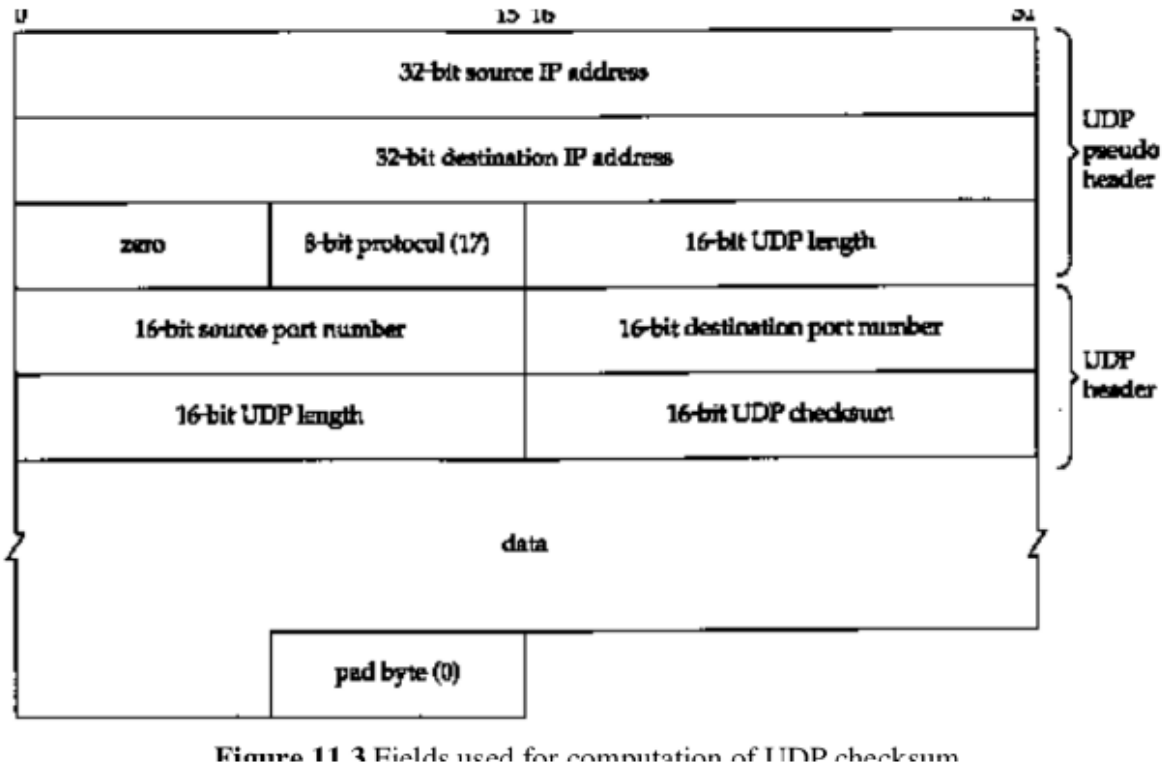


Fig. 6.2: UDP Header For Checksum Calculation

2. Difference between them.
3. What is difference between UDP and TCP.
4. UDP Header Format
5. What is the difference between the checksum of IP layer and UDP/TCP layer.
6. What are port numbers and IANA.
7. Basic services of TCP
8. TCP Header
9. What is ISN
10. Sequence number and acknowledgement number.
11. Flags: SYN, FIN, RST, PSH, URG, ACK
12. Connection Establishment Protocols (3 way handshake)
13. Termination Protocol

6.5 Task – 5: Socket Programs (TCP – UDP Clients and Servers)

Write a UDP client-server that client sends a string to server and server convert it into capital letters and send back to the client. After successful implementation check the packet in Wire-shark and check your understanding. If server is off, still packet can be sniffed or not by wire-shark that is sent by client. (YES)

Write same utility using TCP and check your understanding again about.

Note down the following

- flags that we have learn.
- Sniff packet when client just started. (See is there 3 way handshake)
- Clear the results, now sniff when client try to use server utility by sending data.
- Clear the results, shutdown the server and sniff again. (Terminating protocol should be noticed)

CHAPTER

SEVEN

WEEK - 7 LINUX IOT TRAINING

7.1 Task - 1: TCP Congestion Control

Learn TCP Congestion control in detail. What is Bandwidth, Round Trip Time, Packets Loss? What is tuning of TCP? What is window? What is slow start, congestion avoidance and different congestion control algorithms and their difference.

7.1.1 Compare different congestion control algorithms by developing a network

Develop a network that PC-2 acts as router, PC-1 and PC-2 acts as source or destinations, all are connected through Ethernet cables. By using different congestion control algorithms, copy a file of 1GB on either computer and store the data-rate in a text file. Based on the data rate stored, plot the graphs using GNU-plot in Linux. Explain your understandings.

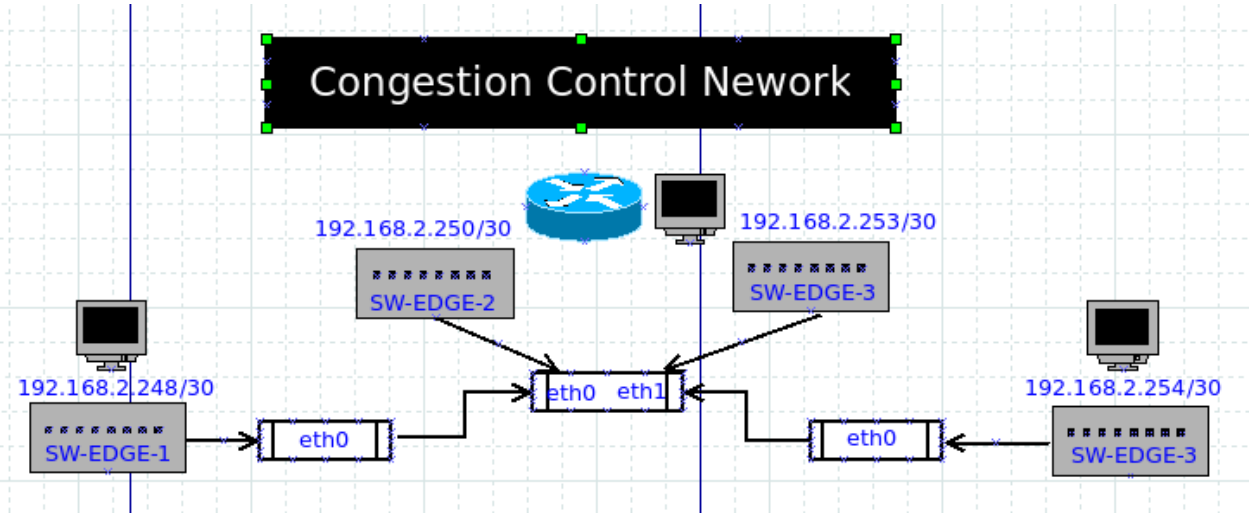


Fig. 7.1: Congestion Control Network Architecture

PC - 1

1. Add eth0 → EDGE_1 → EDGE_1 IP = 192.168.2.249, netmask = 255.255.255.252
2. Add route for eth0 (PC-2)
3. sudo route add -net 192.168.2.252 netmask 255.255.255.252 gw 192.168.2.250

PC - 2

1. It should have two Ethernet ports. Attach an Ethernet adapter.
2. Add eth2 → EDGE_2 → EDGE_2 IP = 192.168.2.250, netmask = 255.255.255.252 (PC - 1 & PC – 2 adapter using LAN cable).
3. Add eth0 → EDGE_3 → EDGE_3 IP = 192.168.2.253, netmask = 255.255.255.252 (PC - 2 & PC – 3 using LAN cable).

PC - 3

1. Add eth0 → EDGE_4 → EDGE_4 IP = 192.168.2.254, netmask = 255.255.255.252
2. Add route for eth0 (PC-2)
3. sudo route add -net 192.168.2.248 netmask 255.255.255.252 gw 192.168.2.253

Now, check using Ping that PC-1 should be able to communicate uptil eth-0 (SW-4) of PC-3 and vice-versa.

7.2 Task – 2: Compare different congestion control algorithms by developing a network

Develop a network that PC-2 acts as router, PC-1 and PC-2 acts as source or destinations, all are connected through Ethernet cables. By using different congestion control algorithms, copy a file of 1GB on either computer and store the data-rate in a text file. Based on the data rate stored, plot the graphs using GNU-plot in Linux. Explain your understandings.

7.3 Task – 3: NETEM Driver – Network Emulation

Learn how to add latency, noise and packet loss into a network. Useful link: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

iproute2

Linux package contains a collection of utilities for controlling TCP/IP networking and traffic control in Linux.

The current version of netem driver emulates variables delay, loss, duplication and re-ordering.

How to add fixed delay to all packets.

7.3.1 Latency

```
tc qdisc add dev eth0 root netem delay 100ms
```

7.3.2 Packet Loss

```
tc qdisc add dev eth0 root netem loss 0.1%
```

7.3.3 Packet Corruption – Noise

tc qdisc add dev eth0 root netem corrupt 0.1% (it introduces single bit in error).

Linux TCP Tunning helpful link: http://kaivanov.blogspot.com/2010/09/linux-tcp-tuning.html

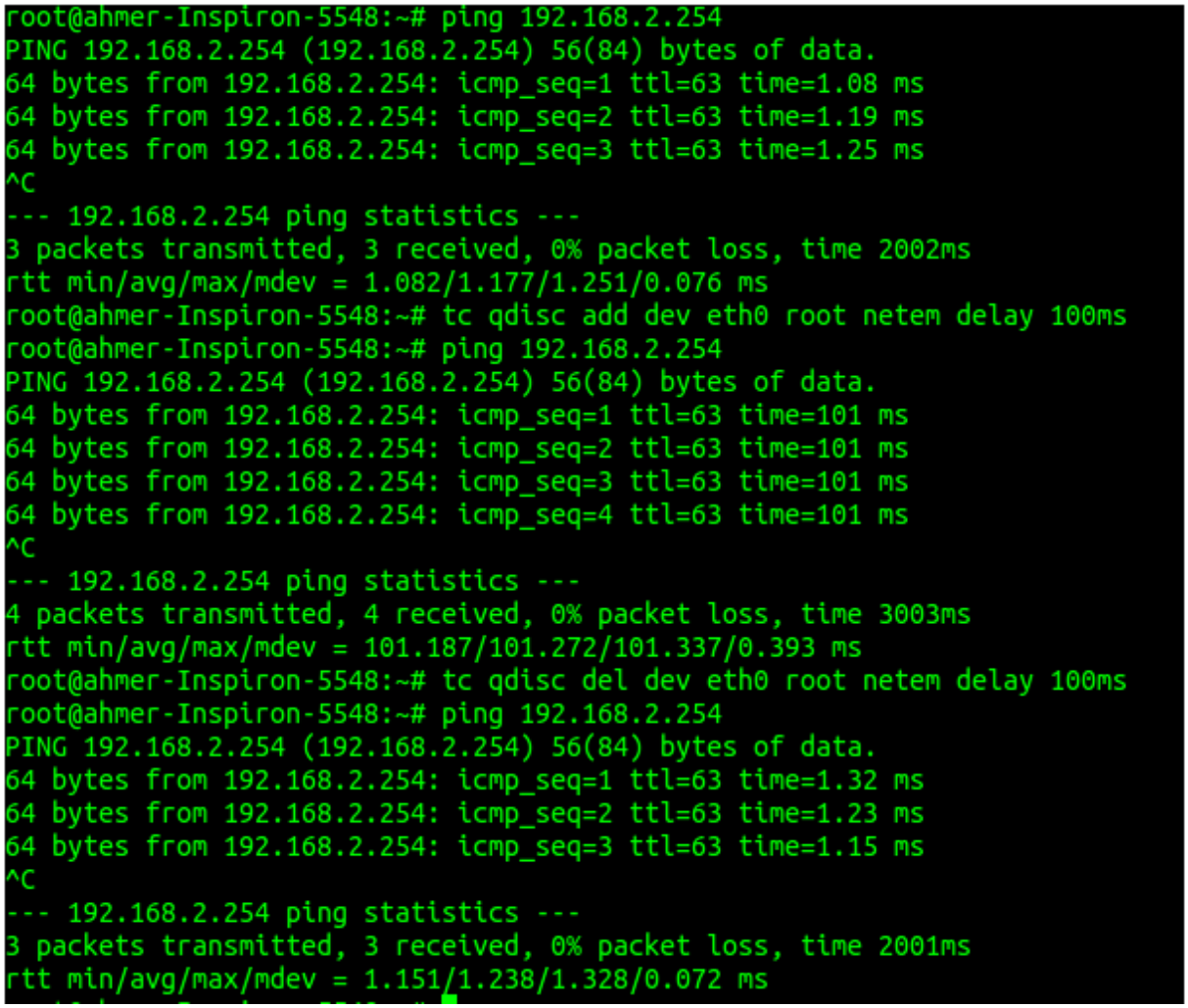


Fig. 7.2: Latency Results

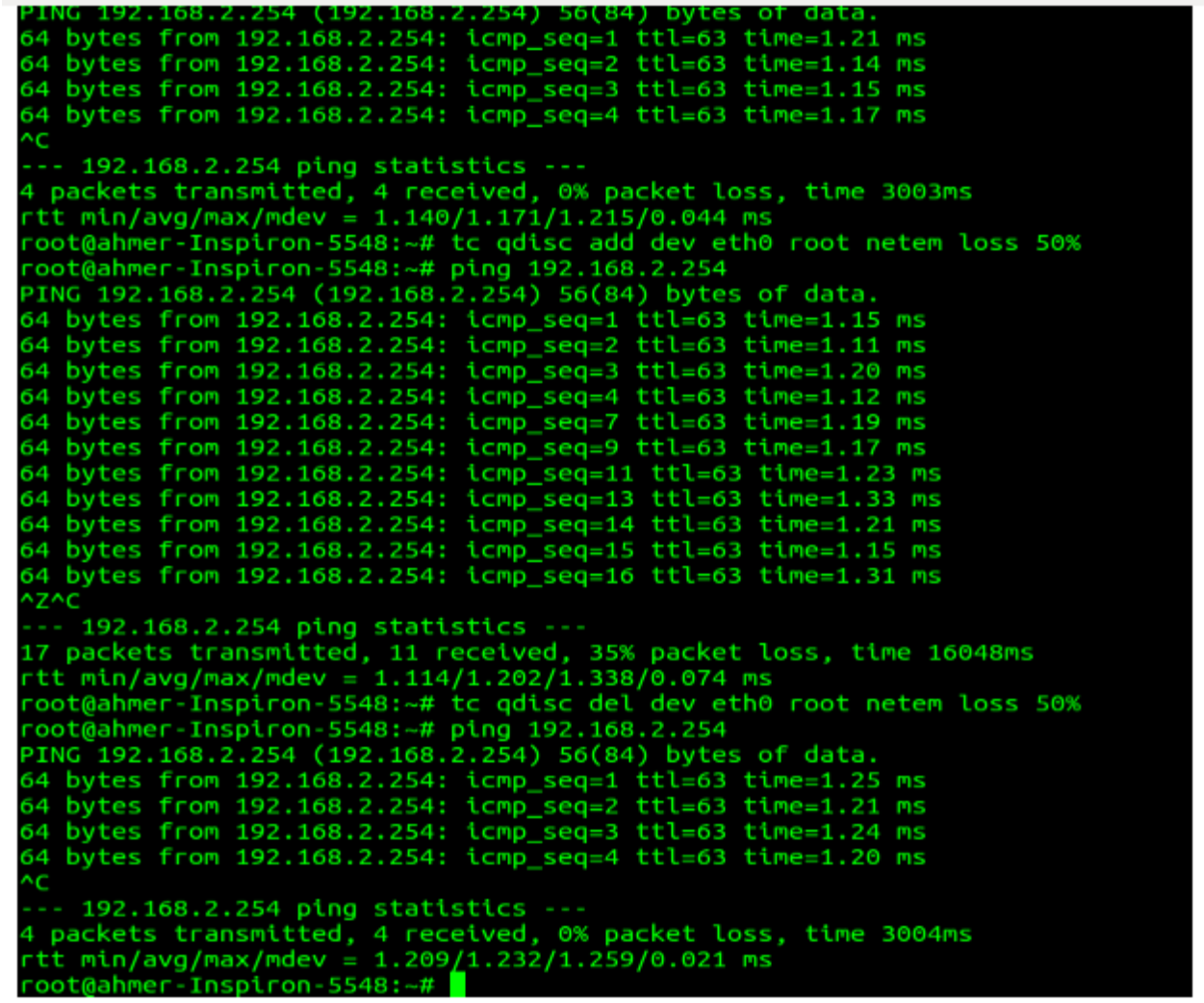


Fig. 7.3: Pscket Loss Results

7.3.4 Create 300 MB Raw Data File

dd if=/dev/sda1 of=/home/ahmer/TCP_data.img bs=1G count=10

```
script -q -c "scp -rp /home/ahmer/TCP_data_300MB.img ahmer@192.168.2.254:/home/ahmer/"
> /home/ahmer/TCP_reno
```

cut desired data into TCP_reno_2

Filter: `cat TCP_reno_2 | cut -d'%' -f2 | cut -d" " -f5 > TCP_reno_3`

How to check the available Congestion Control algorithms modules.ko in your kernel.

```
ls /lib/modules/`uname -r`/kernel/net/ipv4/
```

How to find running CC modules on your kernel?

```
root@ahmer-Inspiron-5548:/proc/sys/net/ipv4# sysctl net.ipv4.tcp_available_congestion_contr
net.ipv4.tcp_available_congestion_control = cubic reno
```

What if your desired CC modules are not running?

```
/sbin/modprobe tcp_bic /sbin/modprobe tcp_highspeed
```

How to change /set the TCP CC algorithm?

```
root@ahmer-Inspiron-5548:/proc/sys/net/ipv4# echo reno > tcp_allowed_congestion_control
```

7.4 Task - 4: Make Your Own Web-Server

How to Download a file from a web-server?

Apache: Hypertext Transfer Protocol Service

Curl: Transfer a URL

7.4.1 Steps

1. Install apache2 on server.
2. Copy the desired files in directory /var/www/html.
3. curl -o filename http://<IP-Address>/filename_wanted_to_download
4. You can Download the file using web-browser as well.

How to change the directory of web-server?

sudo vim /etc/apache2/sites-available/000-default.conf

Change: DocumentRoot /var/www/html → DocumentRoot /your/directory/

sudo vim /etc/apache2/apache2.conf

change: <Directory /var/www/html/> –> <Directory /your/directory/>

Options Indexes FollowSymLinks

AllowOverride None

Require all granted

</Directory>

7.5 Task - 5: GNU-Plot

Learn how to use GNU-Plot in Linux for elaborated graphing such as tiltle, X, and Y axis legends, sub-plotting, reading data from a file etc.

How to install?

Sudo apt-get install gnuplot-qt rlwrap

7.5.1 Settings

For doing diffrent settings for gnu-plot and help *Settings GNU-Plot*

7.6 Task - 6: Git-Hub

Why Git-Hub is used? Make an account on Git-Hub and how to setup git on you Linux machine. Upload some of your codes on the git-hub using command line.

1. Make an account on www.github.com.
2. Install git on your linux machine sudo apt-get install git.
3. ssh-keygen -t rsa -b 4096 -C "ahmer.malik@linuxiot.org".
4. ls -l .ssh → Check the list
5. mv linuxiot* .ssh → move linuxiot (with whatever extension) to .ssh folder.
6. chmod 0600 .ssh/linuxiot* → Change the ownership of the files only to the user (read & write).
7. cat linuxiot.pub → Copy all the contents of file to website of github.

   `https://github.com/settings/ssh → Add SSH Key` copy all the contents here.
8. Make your git repository folder_name = git-repo. Cd into your repository.
9. git init.
10. git config - -global user.email "ahmer.malik@linuxiot.org".
11. git config - -global user.name "Ahmer Malik".
12. Go into your browser → github → select ssh and copy its content (A path git@/..../)
13. git remote add origin (paste the copied path) git@github.com:LinuxIoT/training.git.
14. ssh-add /home/ahmer/.ssh/linuxiot
15. vim test → Make afile with whatever data.
16. git add test
17. git commit -m "test"
18. git push –set-upstream origin master or git push origin master
19. git pull → git push
20. git config –global push.default simple

How to make a new branch?

Only 1 master branch, we are working with till now.

To create new branch: git checkout -b develop

To delete a branch: git branch -d name_branch

To switch to that branch: git checkout develop

Back: git checkout master

To view branches: git branch -a

7.7 Task - 7: Basic Interaction with Python

Learn basic interaction with python environment using Idle-2.7. What are Lists and how variables, functions and modules are defined in python. Try to explore, what is the benefit of choosing python over other languages.

CHAPTER

EIGHT

WEEK - 8 LINUX IOT TRAINING

For learning Python or OOP follow the link

8.1 Task – 1: Practice of writing code in Python

Write a python code that should do following steps. #. Get the file-system list and store it in a file fs.txt #. Get size in KB for each element (i-e /opt, /etc) in the file-system. #. Store the size in dictionary across the name of directory. { i-e /opt: 4KB } #. Print that dictionary in a dictionary.txt file in readable form.

8.1.1 How to Proceed

1. Import os (Read about OS class, what type of methods it provides and those that might help).
2. Learn how to read and write a file in python.

   fd = open('/opt/pylabs/fs_list', 'w')

   for each_item in fs:

   fd.write(each_item)
3. Read about the data-types lists and dictionary. What are these and how they are used in python.

   Dictionary {'Ali' : 0 , 'Akram' : 1}

   List ['Ali' , 1 , 'Ahmer' , 2]
4. What is the benefit of using os.stat method of OS class.

   See the stats of the file or directory.
5. How to access a particular value like size from lists of different stat information.

   a = os.stat(line)

   print (a[6])
6. How to remove the last character from a string line.

   line[:-1]
7. How to concatenate different strings and how to covert an integer into string.

   line + ":" + str(A) + "n"

8.2 Task – 2: Object Oriented Programming in Python

Write the above program again, but now using OOP concepts.

8.2.1 How to Proceed

1. How to declare a class in python.

   class Class_Name : (": at the end")

2. What is class variable.

   A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

3. Constructor Declaration

   def __init__(self, name, salary):

   self.name = name

   self.salary = salary

   Employee.empCount += 1

4. Define methods or functions

   def Function_Name(self): ("self" is compulsory)

   Implementation

8.2.2 Code

The see code click here.

8.3 Task – 3: Importing a class in another File

Repeat the same task, but now the class should be declared in one python file named as class.py. While the object of that class should be instantiated in another python file, named as class_main.py.

How to Proceed:

1. Class definition in a separate file : class.py

   class name_class:

   def __init__(self, other arguments) .

   .

   .

2. How to import a class : class_main.py

   Three different ways: – import file_name_that_have_class_definition —> `import class (no need of extension .py)`

   – from file_name_that_have_class_definition import * (All classes in that file) `from class_name import *`

– from file_name_that_have_class_definition import class_name `from class_name import class_name`

3. Class object instantiated in another file.

– X = file_name.class_name("Constructor arguments initiallization")

X.Methods()

– X = classname ("Constructor arguments initiallization")

X.Methods()

8.3.1 Code

The see code click here.

8.4 Task - 4: Reading Configuration Files (name.conf)

What are configuration files and how to write it in python? How to read it and extract the information to be processed by accessing different OPTIONS and its variables.

Write a .py code that reads STATS.conf file that contains two options [SYSTEM, DIRS]. I have to extract out the STATS variable of SYSTEM and write the information in a text file based on the parameters stored in variable (i.e memory, cpu and processes etc.).

In another text file store the disk information of all the directories mention under the [DIRS] option.

8.4.1 Code

The see code click here.

8.4.2 INHERITANCE

Children inherit some of their characteristics from their parents. Same concept is used in OOP, in order to avoid the re-creation of same piece of code in another class for writing more functional classes. The more functional class inherits the functionality of previously written class and can use it for some other software creation.

Super / Parent Class

The class that provides its functionality to another class.

Sub / Child Class

The class that inherits the functionality parent.

In Python, all classes are sub classes technically.

– When we declare a class as class name_of_class : → class name_of_class (object).

– Those class class do not inherited from some other class are inherited from special class "object".

Example:: #!/usr/bin/python class contact:

```
all_contacts = []

def __init__ (self, name, email):
        self.name = name
        self.email = email
        contact.all_contacts.append(self)
```

class supplier(contact):

```
def order(self):
        print ('Hello I have shared the contacts of suppliers\n')
```

if __name__ == '__main__':

```
s1 = supplier("Ahmer","ahmer.malik@linuxiot.org")
print ( s1.name + ':' + s1.email + '\n')
s2 = supplier("Weqaar","weqaar.janjua@linuxiot.org")
print ( s2.name + ':' + s2.email + '\n')
print s1.all_contacts
```

8.5 Task – 5: Multi-Processing

What is multi-processing? What is the difference between a process and a thread? Learn the multiprocessing [Process] module in Python.

Write a code that can familiarize you how to do multi-processing and differentiate between the results of multiple process program and sequential program as we do traditionally.

CODE:

```
from multiprocessing import Process
import time
```

def f1():

```
print' P-1 Started'
for i in xrange (0,10000):
print 'NUM = %d\n' % i
print  ' P-1 ENDED'
```

def f2():

```
print 'P2 - Started'
for i in xrange (10000,20000):
print 'NUM = %d\n' % i
print 'P2 - ENDED'
```

def without_multi_process():

```
f1()
f2()
```

def multi_process():

```
p1 = Process(name ='Function - 1' , target=f1)
p2 = Process(name ='Function - 2' , target=f2)
p1.start()
p2.start()
```

```
p1.join()
p2.join()
```

if __name__ == '__main__':

```
_start = time.time()
multi_process()
_finish = time.time()
t1 = _finish - _start
_start = time.time()
without_multi_process()
_finish = time.time()
t2 = _finish - _start
print 'Time with multi_processes = %.8f seconds\n' % t1
print 'Time without multi_processes = %.8f seconds\n' % t2
```

8.6 Task - 6: Queues

What are queues and how to make a queue in python. Learn how to use [Queue class of multiprocessing].

Write a code that make a server.py that makes a queue of some friends. A client name "writer.py" can write on that queue and another client "reader.py" can read that queue.

8.6.1 Code

The see code click here.

Learn about some other modules in Python.

1. Sockets
2. Serial - - sudo setserial -g /dev/ttyS[0123]
3. Daemon

8.7 Task - 7: Setup QEMU for ARM Cortex M4 32bit CPU

8.7.1 What is QEMU?

It is used for emulation or virtualization. Emulation means we can write c codes for some other processor family. Instead of purchasing that processor to check, we can simply check our code performance and debugging using emulator. We can say that it virtualize a board.

Link: http://gnuarmeclipse.github.

What is mBedOS?

MbedOS is online IDE that can compile (cross-compilation) codes for different development boards using its C++ API. Compile a binary using their online IDE, for the board/chip supported by Qemu and run on it to verify.

*** Link - Learn mBed

https://drive.google.com/folderview id=0B9Whv5cCE_iEfldHdkp0RWtJbVhYeUFxWFcxNVQxaEpkeVZudS1YTUJtOGFOZVUzdlRl

I compile a binary for NUCLEO_F411RE board and run using QEMU as follows.

```
ahmer@ahmer-Inspiron-5548:/opt/gnuarmeclipse/qemu/2.4.50-201510290935-dev/bin$ .
/qemu-system-gnuarmeclipse --verbose --verbose --board STM32F4-Discovery --mcu S
TM32F411RE --nographic --image /home/ahmer/Downloads/mbed_blinky_NUCLEO_F411RE.b
in --semihosting-config enable=on

GNU ARM Eclipse 64-bits QEMU v2.4.50 (qemu-system-gnuarmeclipse).
Board: 'STM32F4-Discovery' (ST Discovery kit for STM32F407/417 lines).
Device: 'STM32F411RE' (Cortex-M4 r0p0, MPU), Flash: 512 kB, RAM: 128 kB.
Image: '/home/ahmer/Downloads/mbed_blinky_NUCLEO_F411RE.bin'.
Command line: (none).
Cortex-M4 r0p0 core initialised.
LED: Green active high 8*10 @(258,218) /machine/mcu/stm32/gpio[d],12
LED: Orange active high 8*10 @(287,246) /machine/mcu/stm32/gpio[d],13
LED: Red active high 8*10 @(258,274) /machine/mcu/stm32/gpio[d],14
LED: Blue active high 8*10 @(230,246) /machine/mcu/stm32/gpio[d],15
QEMU 2.4.50 monitor - type 'help' for more information
(qemu) QEMU 2.4.50 monitor - type 'help' for more information
(qemu) Cortex-M4 r0p0 core reset.
qemu-system-gnuarmeclipse: Attempt to set CP10/11 in SCB->CPACR, but FP is not s
upported yet.
```

Fig. 8.1: Output of File Compile by mbed on Qemu

What is the meaning of __main__ in Python?

8.8 Useful Information

Differentiate between Mutable and Immutable Data types?

https://lukewickstead.wordpress.com/2015/06/12/mutable-vs-immutable/

Install psutil module: `sudo pip install psutil`

Error:

compilation terminated.

error: command 'x86_64-linux-gnu-gcc' failed with exit status 1

Solution:

```
sudo apt-get install python-dev

sudo pip install psutil
```

```
>>> psutil.phymem_usage()
>>> usage(total=4153868288, used=2854199296, free=1299668992, percent=34.6)
``cat meminfo | grep "MemTotal:" | cut -d' ' -f9``
```

```
# pt.py
class Rectangle(object):
        def __init__(self, w, h):
                self.width = w
                self.height = h
        def area(self):
                return self.width * self.height

r1 = Rectangle(10,20)
print(r1.width, r1.height)
```

It is good if it stays as a standalone code. But we may not want to print out the test case when we want use it as a imported module. So, we need to block it from printing the output. That's what the module **__name__** check is doing in the code below:

```
# pt2.py
class Rectangle(object):
        def __init__(self, w, h):
                self.width = w
                self.height = h
        def area(self):
                return self.width * self.height

if __name__ == '__main__':
        r1 = Rectangle(10,20)
        print(r1.width, r1.height)
```

Fig. 8.2: Proof that getting rich is mostly luck.

CHAPTER

NINE

WEEK - 9 LINUX IOT PROJECT

FIT IoT Lab link

9.1 Project Kickstart

<TBD>

CHAPTER

TEN

WEEK - 10 IOT PROTOCOLS: RESTFUL APIS AND COAP

10.1 REST: Representational State Transfer

To see REST in detail follow the link

It is stateless, client server, cacheable communnication protocol. e.g HTTP

RESTful applications use HTTP requests to post data (create and/or update), read data (e.g., make queries), and delete data. Thus, REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations.

10.1.1 REST as Light Weight Web-Service

As a programming approach, REST is a lightweight alternative to Web Services and RPC.

Much like Web Services, a REST service is:

- Platform-independent (you don't care if the server is Unix, the client is a Mac, or anything else).
- Language-independent (C# can talk to Java, etc.),
- Standards-based (runs on top of HTTP)
- Can easily be used in the presence of firewalls.

Like Web Services, REST offers no built-in security features, encryption, session management, QoS guarantees, etc. But also as with Web Services, these can be added by building on top of HTTP:

For security, username/password tokens are often used.

For encryption, REST can be used on top of HTTPS (secure sockets). ... etc.

10.2 Web-Service

A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response.

10.3 Hypertext Transfer Protocol (HTTP)

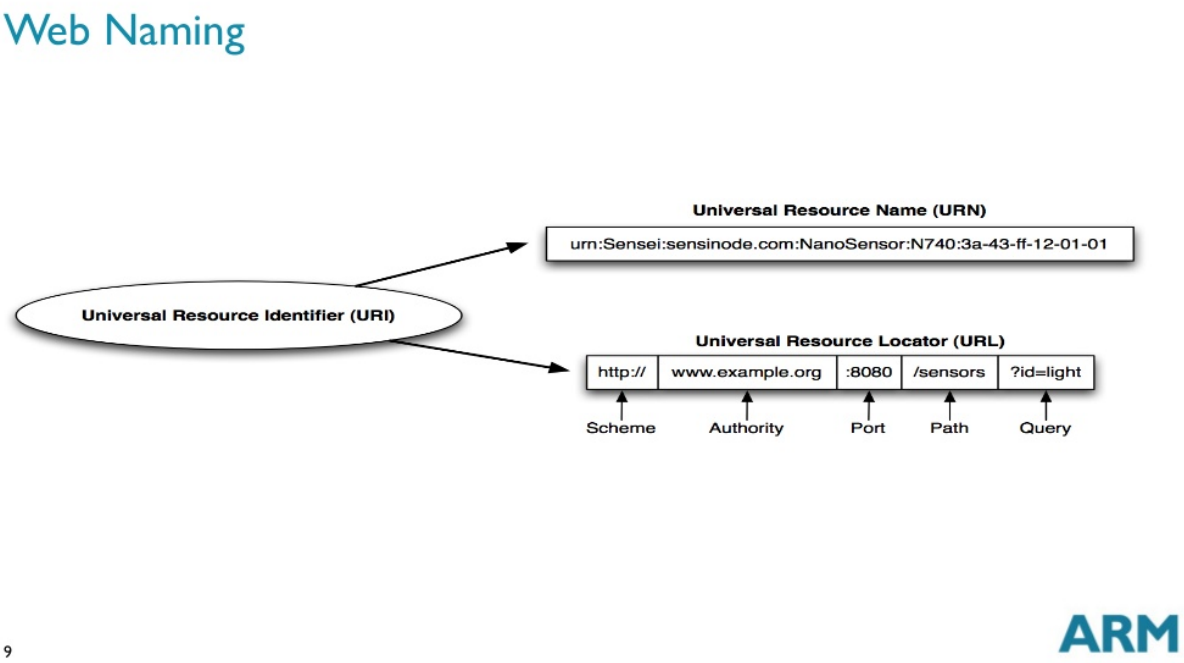
HTTP is a method for encoding and transporting information between a client (such as a web browser) and a web server. HTTP is the primary protocol for transmission of information across the Internet.

Follow the link for better understanding

HTTP follows a request-response paradigm in which the client makes a request and the server issues a response that includes not only the requested content, but also relevant status information about the request.

HTTP is an application layer protocol and relies on an underlying network-level protocol such as Transmission Control Protocol (TCP) to function.

HTTP resources such as web servers are identified across the Internet using unique identifiers known as Uniform Resource Locaters (URLs).



Follow the link for more understanding: https://en.wikibooks.org/wiki/Communication_Networks/HTTP_Protocol

There are two types of messages that HTTP uses

1. Request message
2. Response message

10.3.1 Request Message

The request line has three parts, separated by spaces: a method name, the local path of the requested resource and version of HTTP being used. The message format is in ASCII so that it can be read by the humans. For e.g.:

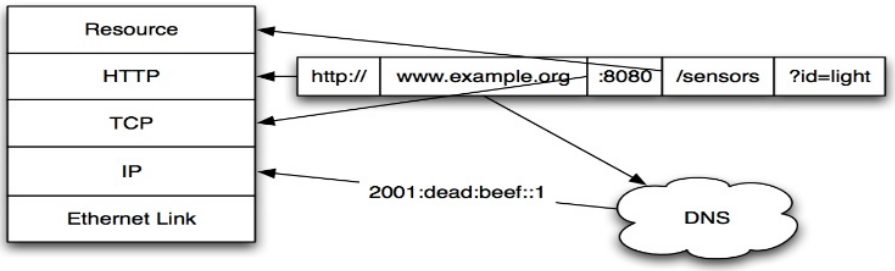
```
GET /path/to/the/file.html HTTP/1.0
```

The GET method is the most commonly used. It states that "give me this resource". The part of the URL is also called as Request URL. The HTTP is to be in uppercase and the next part denotes the version of HTTP. HTTP Request Message: General Format

The HTTP Request message format is shown below:

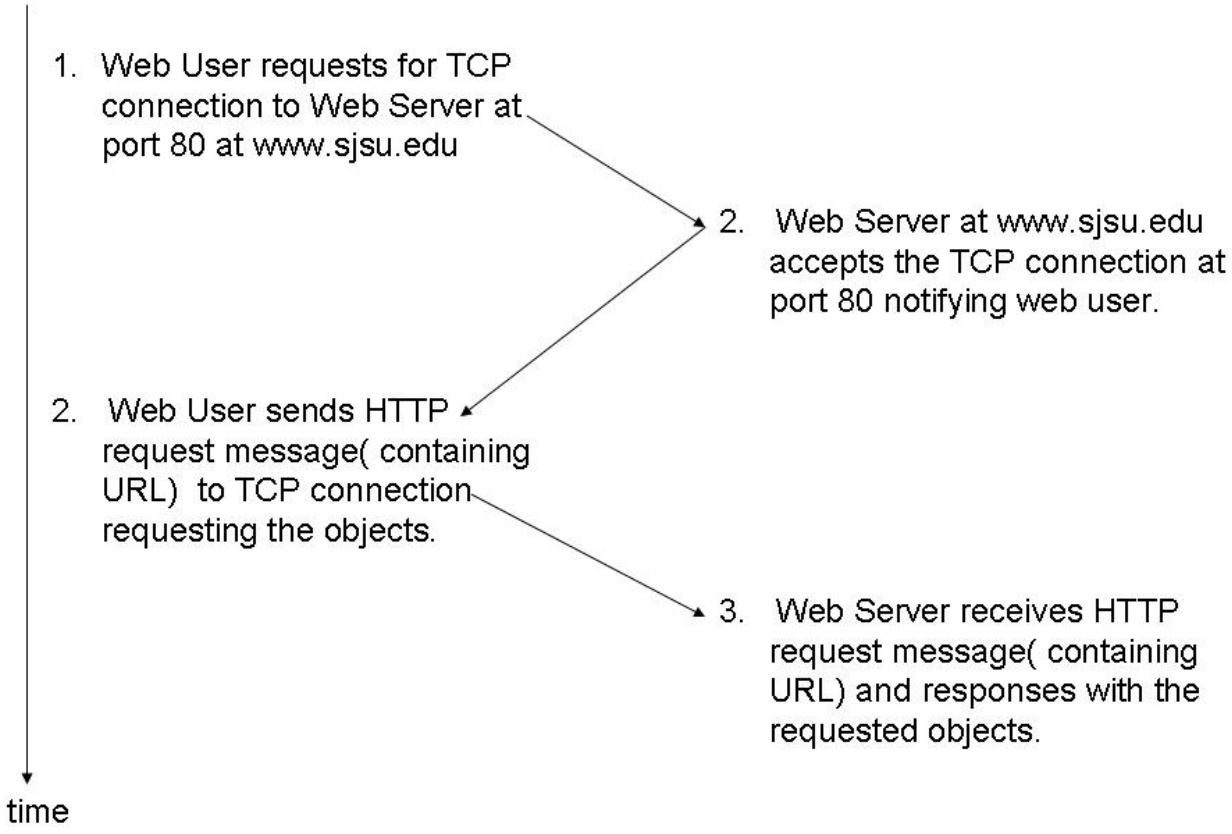
The method is the type of method used to request the URL. Like GET, POST or HEAD. The URL block contains the requested URL. Version denotes the HTTP version. Either HTTP/1.0 or HTTP/1.1. The header lines include the browser type, host , number of objects and file name and the type of language in the requested page. For e.g.:

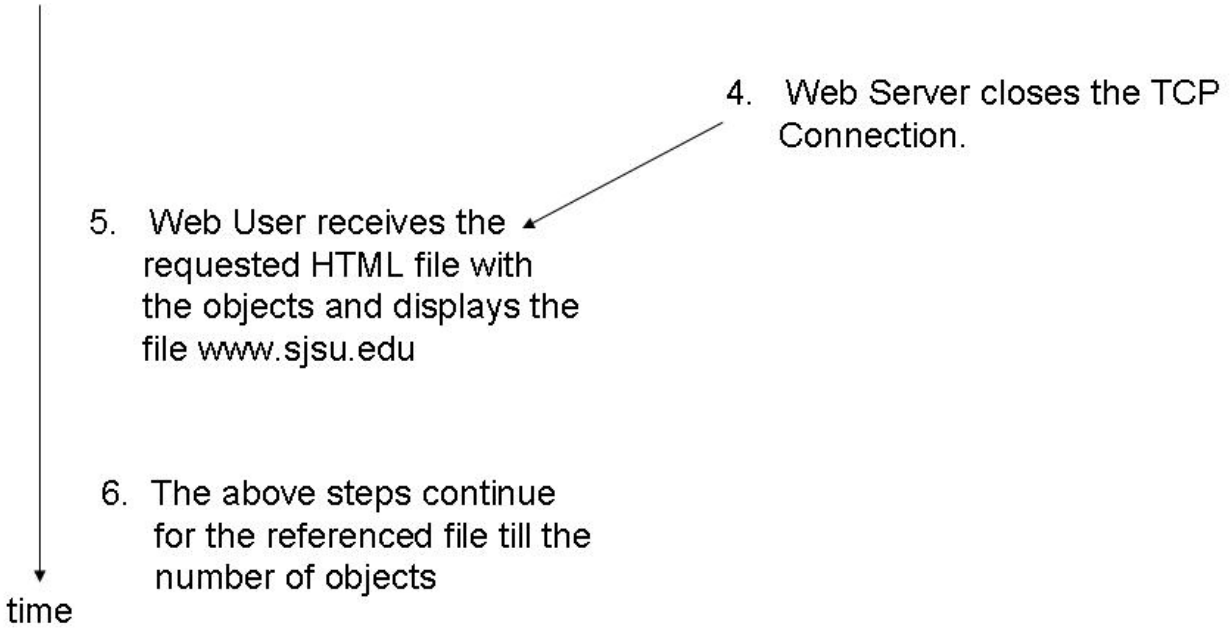
URL Resolution

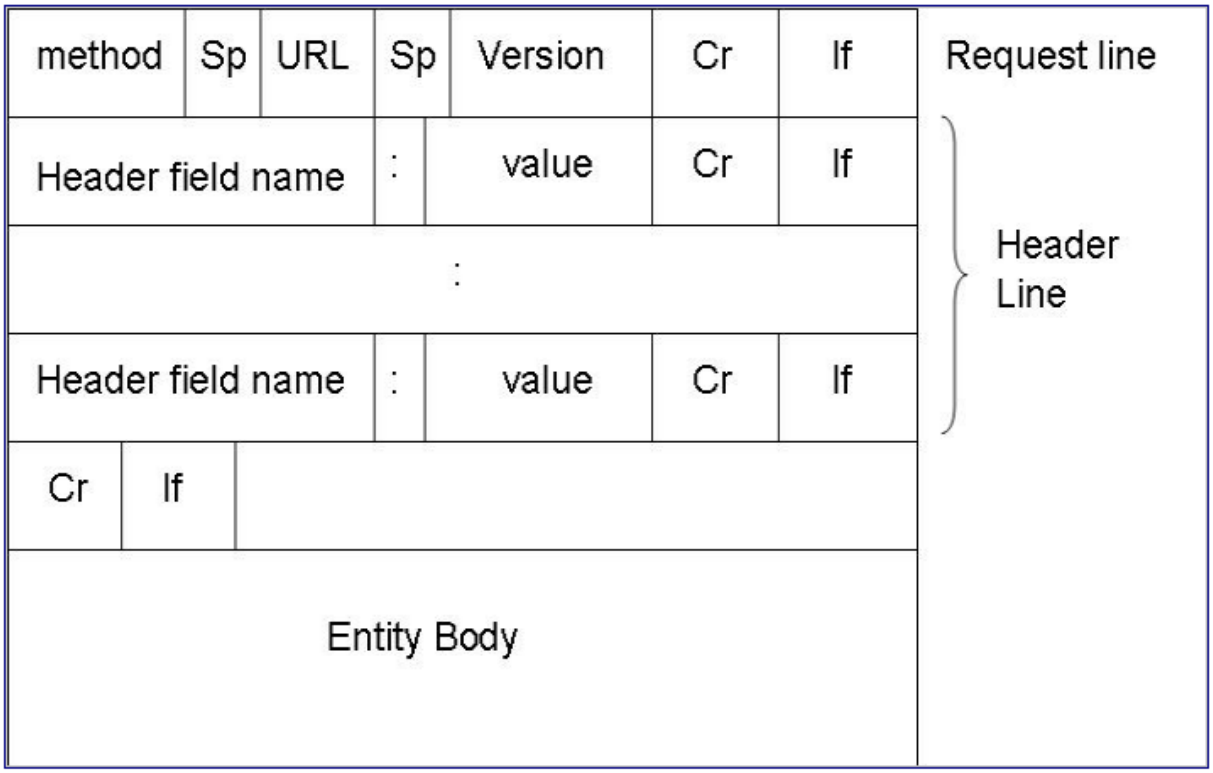


10

ARM




4. Web Server closes the TCP Connection.
5. Web User receives the requested HTML file with the objects and displays the file www.sjsu.edu
6. The above steps continue for the referenced file till the number of objects
time


method
Sp
URL
Sp
Version
Cr
lf
Request line
Header field name
:
value
Cr
lf
:
Header field name
:
value
Cr
lf
Header Line
Cr
lf
Entity Body

The entity body is used by the POST method. When user enters information on to the page, the entity body contains that information.

The HTTP 1.0 has GET, POST and HEAD methods. While the HTTP 1.1 has along with GET, POST and HEAD, PUT and DELETE.

Uploading the information in Web pages

The POST method

The Web pages which ask for input from user uses the POST method. The information filled by the web user is uploaded in server's entity body. The typical form submission in POST method. The content type is usually the application/x-www-form-urlencoded and the content-type is the length of the URL encoded form data.

The URL method

The URL method uses the GET method to get user input from the user. It appends the user information to be uploaded to server to the URL field.

10.3.2 Response Message

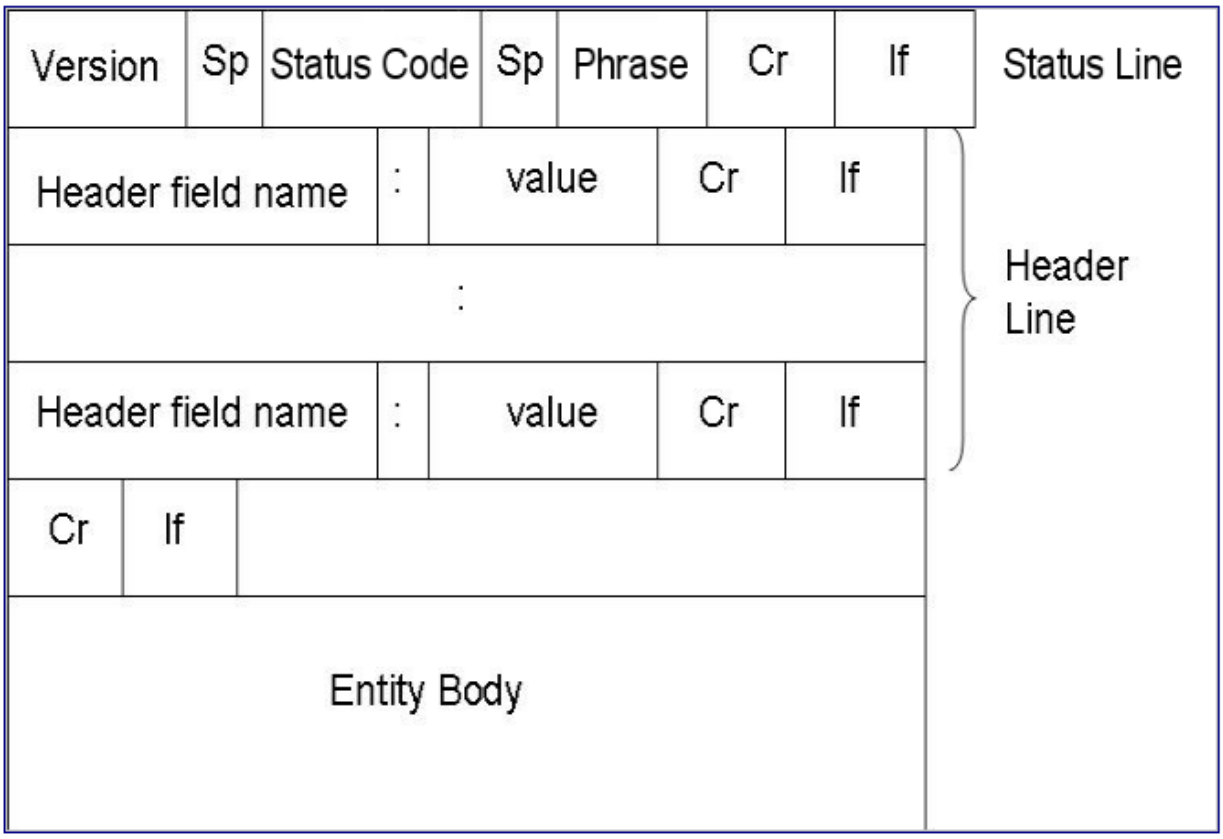
The HTTP message response line also has three parts separated by spaces: the HTTP version, a response status code giving result of the request and English phrase of the status code. This first line is also called as Status line. The HTTP Response message format is shown below:

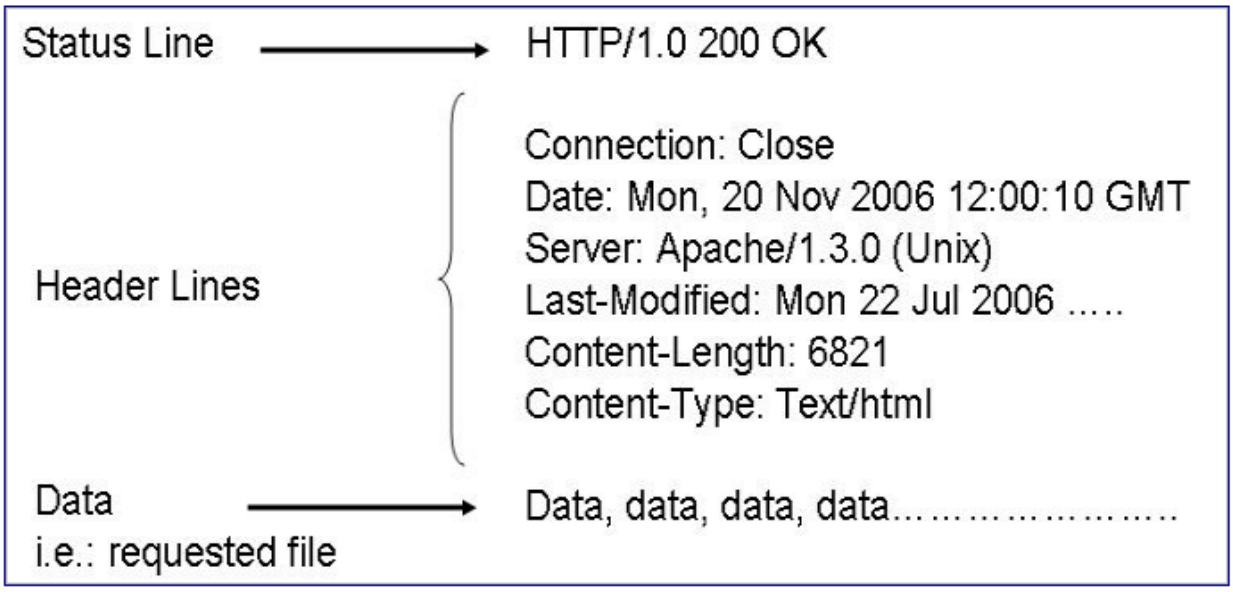
Below are the some HTTP response status codes: - 200 OK The request succeeded, and the resulting resource (e.g. file or script output) is returned in the message body. - 404 Not Found The requested resource doesn't exist. - 301 Moved Permanently - 302 Moved Temporarily - 303 See Other (HTTP 1.1 only)

The resource has moved to another URL (given by the Location: response header), and should be automatically retrieved by the client. This is often used by a CGI script to redirect the browser to an existing file.

- 500 Server Error

An unexpected server error. The most common cause is a server-side script that has bad syntax, fails, or otherwise can't run correctly.


Version
Sp
Status Code
Sp
Phrase
Cr
lf
Status Line
Header field name
:
value
Cr
lf
:
Header field name
:
value
Cr
lf
Header Line
Cr
lf
Entity Body


Status Line
HTTP/1.0 200 OK
Header Lines
Connection: Close
Date: Mon, 20 Nov 2006 12:00:10 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon 22 Jul 2006 .....
Content-Length: 6821
Content-Type: Text/html
Data
i.e.: requested file
Data, data, data, data……

10.4 Proxy Server

A proxy server is a computer that offers a computer network service to allow clients to make indirect network connections to other network services. A client connects to the proxy server, then requests a connection, file, or other resource available on a different server. The proxy provides the resource either by connecting to the specified server or by serving it from a cache. In some cases, the proxy may alter the client's request or the server's response for various purposes.

Useful Link: http://whatismyipaddress.com/proxy-server

10.4.1 Web Proxies

A common proxy application is a caching Web proxy. This provides a nearby cache of Web pages and files available on remote Web servers, allowing local network clients to access them more quickly or reliably.

When it receives a request for a Web resource (specified by a URL), a caching proxy looks for the resulting URL in its local cache. If found, it returns the document immediately. Otherwise it fetches it from the remote server, returns it to the requester and saves a copy in the cache. The cache usually uses an expiry algorithm to remove documents from the cache, according to their age, size, and access history. Two simple cache algorithms are Least Recently Used (LRU) and Least Frequently Used (LFU). LRU removes the least-recently used documents, and LFU removes the least-frequently used documents.

10.5 COAP – Constrained Application Protocol

To learn COAP basic follow

In the PC world, the message exchange is over TCP and application protocol is HTTP. However, for small devices, implementing the TCP stack and then HTTP protocol is really too much to ask for. That is where CoAP was devised. It's a tiny protocol at the application layer (an alternative to HTTP) that is envisaged to fit in tiny form factors. It runs on UDP and not on HTTP. The smallest message size that CoAP mandates is 4-bytes!

SO IS COAP A REPLACEMENT FOR HTTP

CoAP is not a replacement for HTTP. It's an alternative suggestion for tiny devices that have severe resource constraints (e.g. 256 KB memory, 32 KB RAM, 20 MHz speed).

WHAT ARE THE DIFFERENT TYPES OF COAP MESSAGE

CoAP messages follow the same request/response paradigm (largely) that is prevelant in HTTP world. There are 4 types of CoAP messages

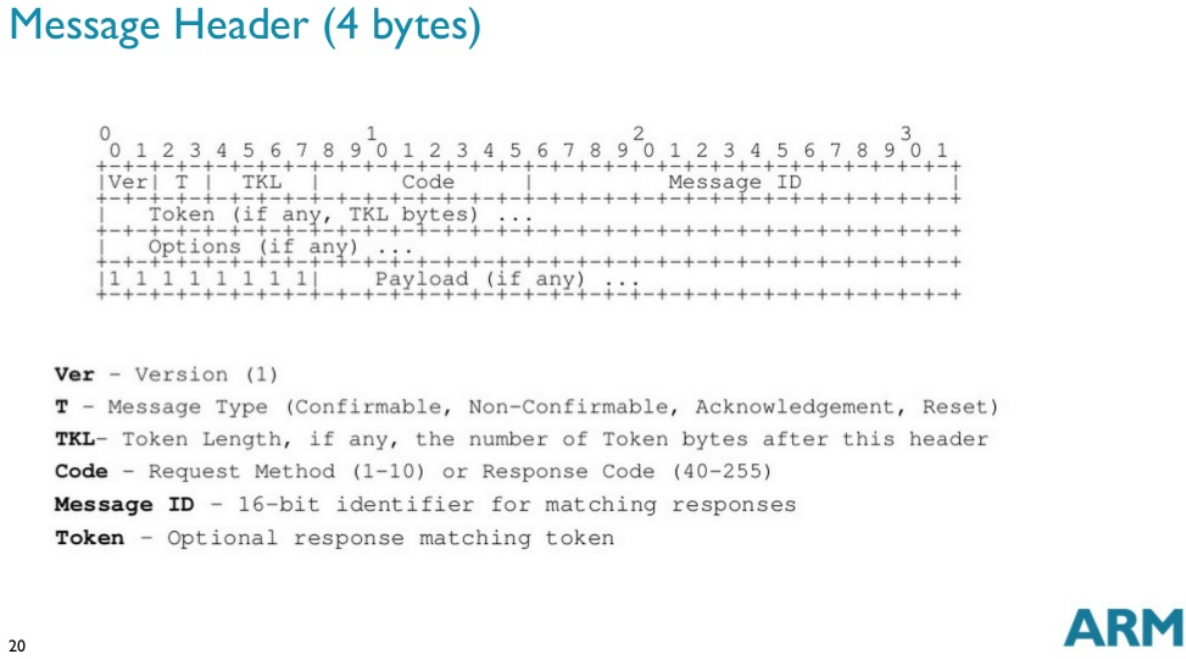
1. CON – This is called "Confirmable" request. When a CON request is sent, the recipient must respond.
2. NON – This is called "Non-Confirmable" request. When a NON request is sent, the recipient is not required to respond back.
3. ACK – This is called "Acknowledgement". This is a response to a CON message. When processing succeeds, the recipient of a CON message should respond back with an ACK. The ACK message can also contain result of the processing alongwith.
4. RST – This is called "Reset" message. When the recipient of a message encounters an error, does not understand the message or is no longer interested in the message sender, this response is sent bac

For detailed understanding follow

10.5.1 COAP Header Format

WHAT ARE THE DIFFERENT PARTS OF A COAP MESSAGE

A CoAP message, at minimum, must be 4 bytes long. The different parts, that can make up a CoAP message are: - Version – What is the CoAP version that this message follows (as of now its V 1) - The message type (CON, NON, ACK, RST) - The message code (Like PUT, GET, UNAUTHORIZED) - The message id. Each CoAP message has an ID associated with it. The ID is generally unique for a conversation for some period of time, after which, it can be recycled. - The token. A token is another value that may be attached to the message for matching. Messages can be sent out of order and at that time, token becomes important. - A set of options. Options in CoAP are like HTTP request headers. They contain metadata about the CoAP message itself (e.g. the CoAP port, the CoAP host, the CoAP querystring) - The payload. This is the actual message that is being exchanged.



WHAT IS OBSERVE IN COAP

Sometimes, you want to monitor a value, let's say, temperature. In that case, the client does not need to continuously poll the server. The client can send one "Observable" request to the server. From that point onwards, the server will remember the client connection details, and on every change in temperature, will send the new value to the client.If client no longer wishes to receive the temperature values, it can respond back with an RST message and server will clear it's memory. More on this in examples.

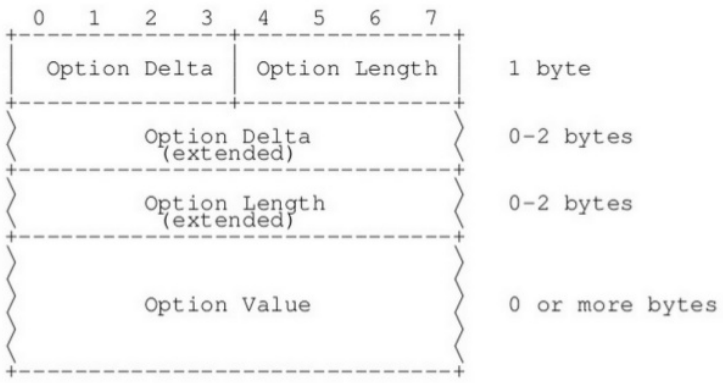
WHAT IS BLOCK TRANSFER IN COAP

Given that CoAP messages should be key very small in size, there will be cases when you need to exchange large data streams, either from client to server (like a PUT) or from server to client (like a GET). In that case, the CoAP message can contain options associated with block transfer and then the client and server can exchange multiple messages with blocks of data in each. More on this later on, in tutorials.

10.5.2 Logging in Python

Logging is way of printing information while running a module. It is just like "printf" but have many advantages over it. import logging

Option Format

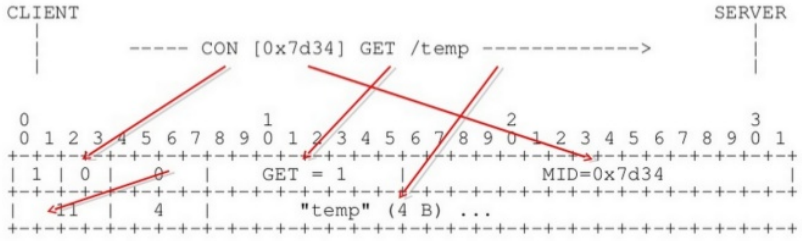


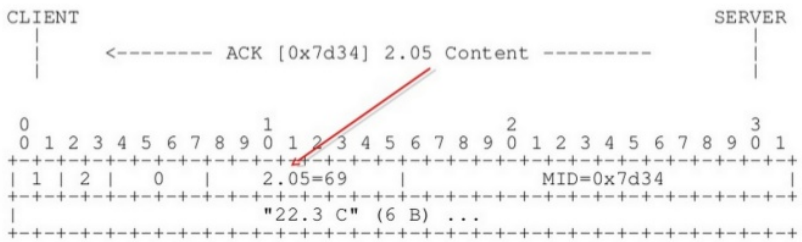
Option Delta - Difference between this option type and the previous
Length - Length of the option value
Value - The value of Length bytes immediately follows Length

21

ARM

Bits and bytes...

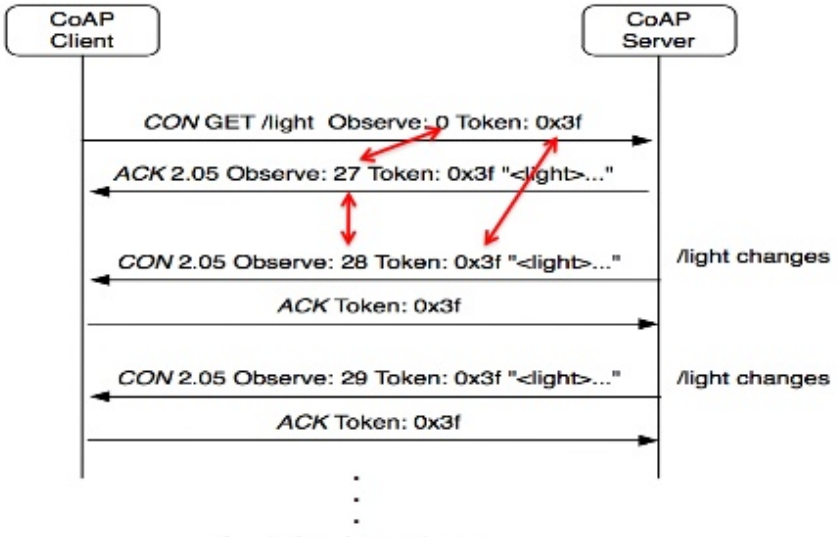




26

ARM

Observation

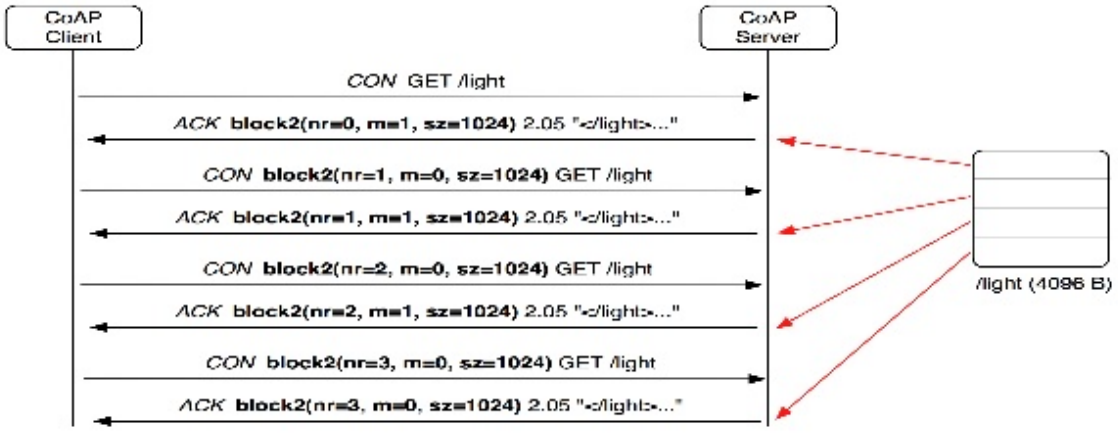


See draft-ietf-core-observe

29

ARM

Block transfer



See draft-ietf-core-block

30

ARM

Another useful feature of the logging API is the ability to produce different messages at different log levels. Let's suppose, I configure the level INFO, then all the log levels above it will be get printed [20 - 50], while lower will not be printed [0 – 10].

Level Value

CRITICAL	50
ERROR	40
WARNING	30
INFO	20
DEBUG	10
UNSET	0

Follow the links for better understanding the logging.

http://victorlin.me/posts/2012/08/26/good-logging-practice-in-python

https://pymotw.com/2/logging/

What is the difference between functions, sub-routines and co-routines?

Function

A function is basically same just like as it is we use in maths.

```
F(X) = 2X + 1
```

It has some input and based on the expression, it has specific output. Similarly, used in C/C++

int F (int X):

```
{       return  ((2*X)+1;
}
```

Sub-Routine

A sub-routine and a function is very much similar except one difference that is, sub-routines does not return anything. It is simply a set of instructions in order to perform a certain task. We can say that, a void function is a sub-routine.

Both functions and sub-routines have single entry and ending point, and executed completely.

Co-Routine

A co-routine is different from a sub-routine in a way that it has single entry point, multiple ending and re-entry points. That's why we use "resume" co-routine instead of "call" a subroutine.

Co-routine:

```
coroutine foo {
 yield 1;
 yield 2;
 yield 3; }
 print foo();
 print foo();
 print foo();
 Prints: 1 2 3
```

Sub-routine:

```
coroutine foo {
return 1;
return 2; //Dead code
return 3;}
print foo();
print foo();
```

```
print foo();
Prints: 1 1 1
```

Note: Coroutines may use a return, and behave just like a subroutine

For co-routines in Python (generators, yield) follow the link: import asyncio (python library) http://wla.berkeley.edu/~cs61a/fa11/lectures/streams.html#coroutines https://www.jeffknupp.com/blog/2013/04/07/improve-your-python-yield-and-generators-explained/

10.5.3 Coap Implementation in Python – aiocoap

https://aiocoap.readthedocs.org/en/latest/examples.html

you will want to create a single Context instance that binds to the network. The Context.create_client_context() and Context.create_server_context() coroutines give you a readily connected context.

On the client side, you can request resources by assembling a Message.

https://aiocoap.readthedocs.org/en/latest/aiocoap.message.html#aiocoap.message.Message

For COAP communication, we will use aiocoap module in Python 3.0+. aicoap module is not yet supported for Python 2.7. Let's install Python 3.0+.

Check the using version of Python.

```
python -V
```

To change Python2.7 → Python3.0+

```
alias python=python3
```

https://aiocoap.readthedocs.org/en/latest/examples.html

10.5.4 COAP Communication in C

libcoap is the COAP communication library for C. Download libcoap: https://libcoap.net/ How to use: https://gitlab.informatik.uni-bremen.de/bergmann/libcoap/tree/3e9afb146043a94da25e0e8fc4bc6c03ebc7a2f1/examples

CHAPTER

ELEVEN

SETTINGS GNU-PLOT

```
#
# Plot trigonometric functions with Gnuplot.
#
# AUTHOR: Hagen Wierstorf
reset
# wxt
set terminal wxt size 350,262 enhanced font 'Verdana,10' persist
# png
#set terminal pngcairo size 350,262 enhanced font 'Verdana,10'
#set output 'plotting_functions.png'
# svg
#set terminal svg size 350,262 fname 'Verdana, Helvetica, Arial, sans-serif'
#fsize '10'
#set output 'plotting_functions.svg'
# Line styles
set border linewidth 1.5
set style line 1 linecolor rgb '#0060ad' linetype 1 linewidth 2 # blue
set style line 2 linecolor rgb '#dd181f' linetype 1 linewidth 2 # red
# Legend
set key at 6.1,1.3
# Axes label
set xlabel 'x'
set ylabel 'y'
# Axis ranges
set xrange[-2*pi:2*pi]
set yrange[-1.5:1.5]
# Axis labels
```

set xtics (“-2Ï€” -2*pi, “-Ï€” -pi, 0, “Ï€” pi, “2Ï€” 2*pi)

set ytics 1

set tics scale 0.75

Functions to plot

a = 0.9

f(x) = a * sin(x)

g(x) = a * cos(x)

Plot

plot f(x) title ‘sin(x)’ with lines ls 1,

g(x) notitle with lines ls 2

Now we save our file as introduction.gnu and execute it by running the following command in BASH under Linux.

$ gnuplot introduction.gnu

Syntax: plot {[ranges]}

{[function] | {“[datafile]” {datafile-modifiers}}}

{axes [axes] } { [title-spec] } {with [style] }

{, {definitions,} [function] ...}

where either a [function] or the name of a data file enclosed in quotes is supplied. For more complete descriptions, type: help plot help plot with help plot using or help plot smooth .

11.1 Customization

Create a title: > set title “Force-Deflection Data”

Put a label on the x-axis: > set xlabel “Deflection (meters)”

Put a label on the y-axis: > set ylabel “Force (kN)”

Change the x-axis range: > set xrange [0.001:0.005]

Change the y-axis range: > set yrange [20:500]

Have Gnuplot determine ranges: > set autoscale

Move the key: > set key 0.01,100

Delete the key: > unset key

Put a label on the plot: > set label “yield point” at 0.003, 260

Remove all labels: > unset label

Plot using log-axes: > set logscale

Plot using log-axes on y-axis: > unset logscale; set logscale y

Change the tic-marks: > set xtics (0.002,0.004,0.006,0.008)

Return to the default tics: > unset xtics; set xtics auto

11.2 MULTI-PLOT

Gnuplot can plot more than one figure in a frame (like subplot in matlab) i.e., try:

set multiplot; # get into multiplot mode

set size 1,0.5;

set origin 0.0,0.5; plot sin(x);

set origin 0.0,0.0; plot cos(x)

unset multiplot # exit multiplot mode

In gnuplot, exponentiation uses `**, not ^.  x^2 = x**2.`

Variable define : plot [t=-4:4] exp(-t**2 / 2), t**2 / 16

11.3 Plotting Data

Discrete data contained in a file can be displayed by specifying the name of the data file (enclosed in quotes) on the plot or splot command line. Data files should have the data arranged in columns of numbers. Columns should be separated by white space (tabs or spaces) only, (no commas). Lines beginning with a # character are treated as comments and are ignored by Gnuplot. A blank line in the data file results in a break in the line connecting data points.

For example your data file, force.dat , might look like:

This file is called force.dat

Force-Deflection data for a beam and a bar

Deflection	Col-Force	Beam-Force
0.000	0	0
0.001	104	51
0.002	202	101
0.003	298	148
0.0031	290	149
0.004	289	201
0.0041	291	209
0.005	310	250
0.010	311	260
0.020	280	240

You can display your data by typing:

gnuplot> plot “force.dat” using 1:2 title ‘Column’,

“force.dat” using 1:3 title ‘Beam’

Do not type blank space after the line continuation character, “” .

Your data may be in multiple data files. In this case you may make your plot by using a command like:

gnuplot> plot “fileA.dat” using 1:2 title ‘data A’,

“fileB.dat” using 1:3 title ‘data B’

For information on plotting 3-D data, type:

gnuplot> help splot datafile