"Designing a Wireless Sensor Network using RaspberryPi"

A

Project Report

submitted in partial fulfillment of the

requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

Name	Roll No.
Suhaib Yousuf	R610214051
Himanshu Sharma	R610214014
Aditi Thakur	R610214050
Shreya Singh	R610214038

under the guidance of

Ms. Sheetal Bisht

Asst. Professor

CIT,UPES

Department of Computer Science & Engineering

Centre for Information Technology

University of Petroleum & Energy Studies

Bidholi, Via Prem Nagar, Dehradun, UK

December – 2016

CANDIDATE'S DECLARATION

I/We hereby certify that the project work entitled **" Designing a Wireless Sensor Network using RaspberryPi"** in partial fulfilment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING with specialization in Mainframe Technology and submitted to the Department of Computer Science & Engineering at Center for Information Technology, University of Petroleum & Energy Studies, Dehradun, is an authentic record of my/ our work carried out during a period from **August**, **2016** to **November**, **2016** under the supervision of **Ms. Sheetal Bisht, Asst. Professor,CIT,UPES**.

The matter presented in this project has not been submitted by me/ us for the award of any other degree of this or any other University.

Suhaib Yousuf	**Himanshu Sharma**	**Aditi Thakur**	**Shreya Singh**
R610214051	**R610214014**	**R610214050**	**R610214038**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 10 December,2016

Ms. Sheetal Bisht
Asst. Professor
Project Guide

Dr. Hanumat Shashtry
Program Head – Mainframe Technology
Center for Information Technology
University of Petroleum & Energy Studies
Dehradun – 248 001 (Uttarakhand)

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **Ms. Sheetal Bisht**, for all advice, encouragement and constant support he has given us through out our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our respected Program Head of the Department, **Dr. Hanumat Shashtry**, for his great support in doing our project in **Network** at **CIT**.

We are also grateful to **Dr. Manish Prateek, Associate Dean** and **Dr. Kamal Bansal,** Dean CoES, UPES for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

Name	**Suhaib Yousuf**	**HimanshuSharma**	**Aditi Thakur**	**Shreya Singh**
Roll No.	**R610214051**	**R610214014**	**R610214050**	**R610214038**

ABSTRACT

with over a decade of intensive research and development, wireless sensor network technology has been emerging as a viable solution to many innovative applications. In a computer network, the transmission of data is based on the routing protocol which selects the best routes between any two nodes. Wireless Sensor Networks (WSNs) consist of small nodes with sensing, computation, and wireless communications capabilities. Many routing, power management, and data dissemination protocols have been specifically designed for WSNs where energy awareness is an essential design issue. The focus, however, has been given to the routing protocols which might differ depending on the application and network architecture. Different types of routing protocols are applied to specific network environment. In our project we will be implementing a routing protocol on an open-source hardware platform Raspberry Pi for a wireless sensor network. The system developed will be highly cost efficient and scalable.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Due to recent technological advances, the manufacturing of small and low cost sensors became technically and economically feasible. The sensing electronics measure ambient conditions related to the environment surrounding the sensor and transforms them into an electric signal. Processing such a signal reveals some properties about objects located and/or events happening in the vicinity of the sensor. Networking unattended sensor nodes may have profound effect on the efficiency of many military and civil applications such as target field imaging, intrusion detection, weather monitoring, security and tactical surveillance, distributed computing, detecting ambient conditions such as temperature, movement, sound, light, or the presence of certain objects, inventory control, and disaster management. Deployment of a sensor network in these applications can be in random fashion (e.g., dropped from an airplane) or can be planted manually (e.g., fire alarm sensors in a facility). For example, in a disaster management application, a large number of sensors can be dropped from a helicopter. Networking these sensors can assist rescue operations by locating survivors, identifying risky areas, and making the rescue team more aware of the overall situation in the disaster area.

In the past few years, an intensive research that addresses the potential of collaboration among sensors in data gathering and processing and in the coordination and management of the sensing activity were conducted. However, sensor nodes are constrained in energy supply and bandwidth. Thus, innovative techniques that eliminate energy inefficiencies that would shorten the lifetime of the network are highly required. Such constraints combined with a typical deployment of large number of sensor nodes pose many challenges to the design and management of WSNs and necessitate energy-awareness at all layers of the networking protocol stack. For example, at the network layer, it is highly desirable to find methods for Energy efficient route discovery and relaying of data from the sensor nodes to the BS so that the lifetime of the network is maximized.

Routing in WSNs is very challenging due to the inherent characteristics that distinguish these networks from other wireless networks like mobile ad hoc networks or cellular networks. First, due to the relatively large number of sensor nodes, it is not possible to build a global addressing scheme for the deployment of a large number of sensor nodes as the overhead of ID maintenance is high. Thus, traditional IP-based protocols may not be applied to WSNs. Furthermore, sensor nodes that are deployed in an ad hoc manner need to be self-organizing as the ad hoc deployment of these nodes requires the system to form connections and cope with the resultant nodal distribution especially that the operation of the sensor networks is un-attended.

1.1 ***<u>History</u>***

Early works on sensor networks and cyber physical systems have been focused on the development of enabling Technologies by addressing a myriad of technical challenges such as multihop routing, communication abstractions, middleware and operating systems (OS), and semantic abstractions and sharing of data. Most of the early testbed systems have been built using early stage sensor network research platforms such as CROSSBOW (now MEMSIC) motes and TinyOS software framework.

RIP is a dynamic routing protocol used in local and wide area networks. The routing algorithm used in RIP, the Bellman-Ford algorithm, was first deployed in a computer network in 1967, as the initial routing algorithm of the ARPANET.
The earliest version of the specific protocol that became RIP was the Gateway Information Protocol, part of the PARC Universal Packet internetworking protocol suite, developed at Xerox Parc. A later version, named the Routing Information Protocol, was part of Xerox Network Systems.
A version of RIP which supported the Internet Protocol (IP) was later included in the Berkeley Software Distribution (BSD) of the Unix operating system. It was known as the routed daemon. Various other vendors would create their own implementations of the routing protocol. Eventually, RFC 1058 unified the various implementations under a single standard.

1.2 Requirement Analysis

1.2.1 System Requirements

Language : C

Compiler : GCC

OS : Linux

Hardware Requirements : Minimum 512MB RAM and 128 GB HDD Machine running on intel i3.
Raspberry Pi.
2GB SD Card.

1.3 Main Objective

To deploy highly energy and cost efficient communication network using Raspberry Pi

1.4 Sub Objective

1. To implement socket programming using C on Linux platform
2. To implement the routing protocol for wireless network.

1.5 Pert Chart

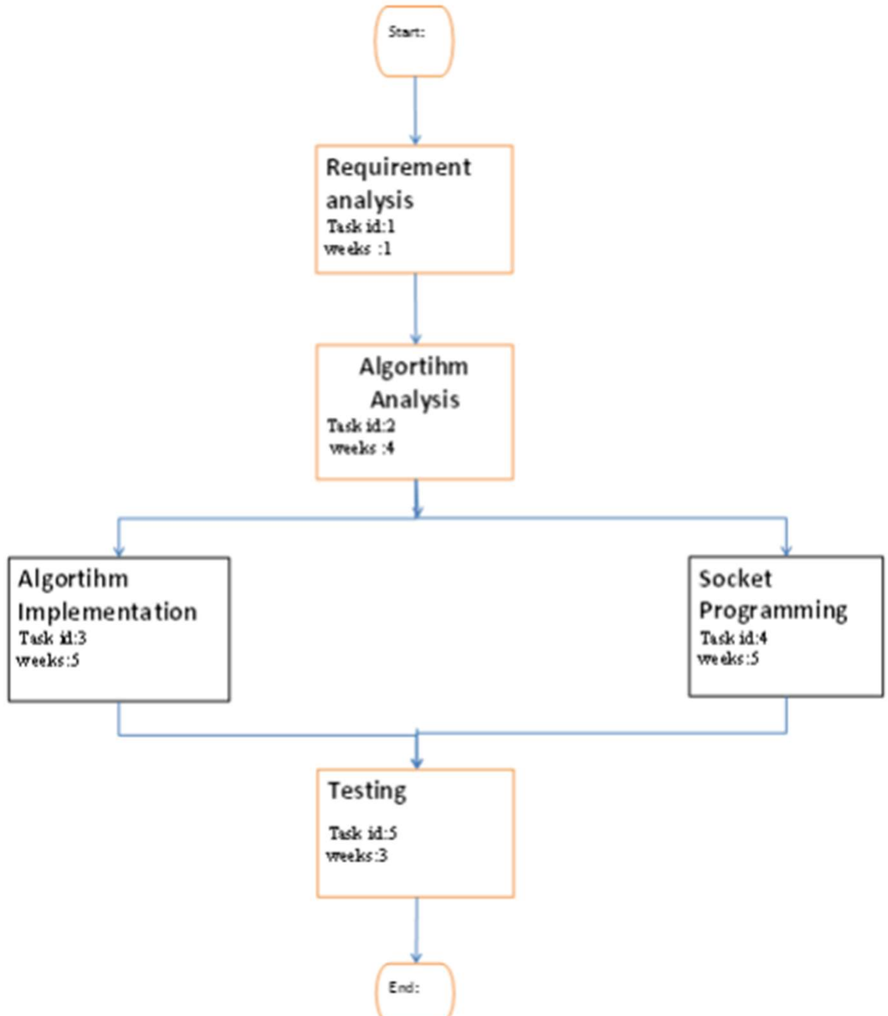


Fig 1.1

2. SYSTEM ANALYSIS

2.1 Motivations

Existing routers available in the market are very costly and energy inefficient.
So, looking at the current energy crisis in the world we thought of designing a energy efficient router at a low cost.

2.2 Proposed System

Implementing Routing protocol on Raspberry Pi device

2.3 Modules

Sending module

Receiving module

Routing Module

3. Design

3.1 Component Diagram

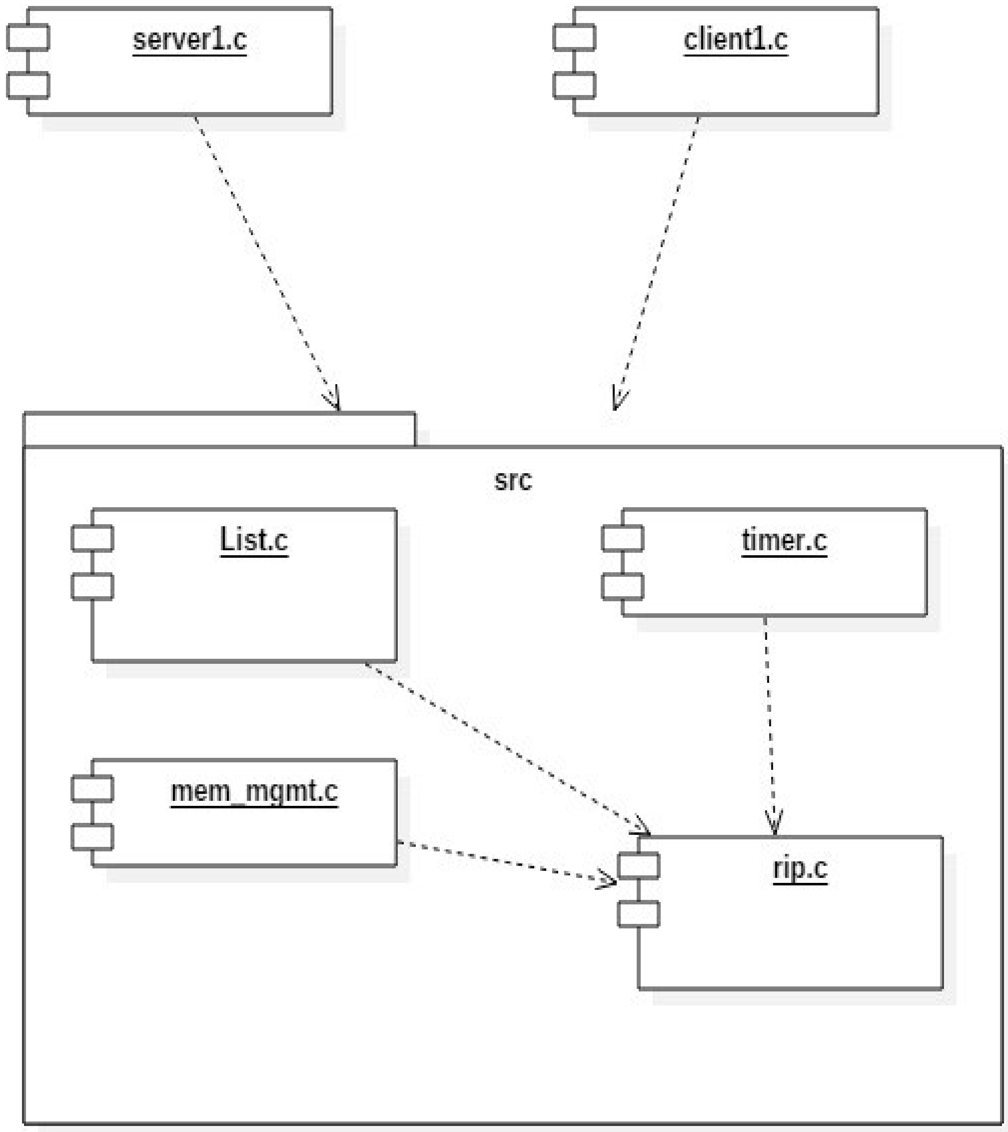


Fig 3.1

<u>3.3</u> ***Data Flow Diagram***

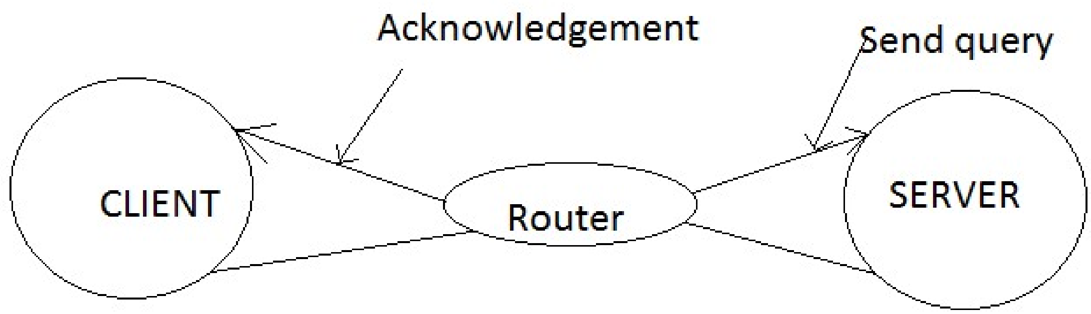


Fig 3.2

4. Implementation

Assumptions:

a) Each link has a cost of 1
b) IP address of neighbours in neighbour file

1)Data Structures:

1.1) Routing Table
Routing Table is a linked list with one route entry per node. Each entry will be of following type

```
struct route_entry {
        listnode *lnode
        struct in_addr destination;
        uint32_t metric;
        struct in_addr nexthop;
        byte routechanged;              //route change flag
        void *timeout_tmr;
        void *garbage_tmr
}
```

Lnode: generic linked list pointer
Destination: IP address of the destination
Metric: Cost to reach destination
Nexthop: IP address of the next router on the path to destination
Routechanged: Will be set when route is updated
Timeout_tmr: Stores timer object of the timeout timer
Garbage_tmr: Stores timer object of the garbage timer

1.2) RIP message
A RIP message will have the header and variable numbers of rip_entries

```
struct rip_header {
        byte command;
        byte version;
        short int _zero;
}
```

Command: Either Request or Response
Version: IP version- IPv4 or IPv6

```
struct rip_entry {
        short int addrfamily;
        short int _zero;
        struct in_addr destination;
        uint32_t _zero1;
        uint32_t _zero2;
        uint32_t metric;
}
```

Addrfamily: currently only AF_INET is supported
Destination: IP address of destination
Metric: cost to reach destination

1.3) Neighbours File

IP address of neighbours must be written in a file. This file name is passed as the first parameter to the executable.

Format of the file: IP address terminated by a $
Example :
192.168.4.1$192.168.1.5$

1.4) Neighbour List

Neighbours of current node will be placed in a list of neighbour

```
struct neighbour {
        struct in_addr addr;
}
```

2) Modules

2.1) Init :
This module reads the Neighbours file and populates the neighbour linked list.
It also sends a request message to all its neighbours.
It also send a message with it’s own ip to it’s neighbours. This is needed when a node was down and then comes up.

2.2) Input –

Module to do input processing.
Receive data from the RIP port and call appropriate processing function

There can be 2 types of messages:- Request and Response

A] Response message will handled by :
int decode_response(struct in_addr from, char *data, int len)

Arguments:
a) from – address from whom the response message has come
b) data- data buffer received
c) len- length of data received

Returns:
0 : on success -1: error

This function will take the data buffer received on the port, the address of the sender and the length of the data received.

This function will make and update entries in routing table. Timers for each route entry will be set/reset as maybe the case. In case of changes to route table, send_update will be called to send triggered update.

B] Request message will be handled by :
int decode_request(struct in_addr from, char *data, int len)

Arguments:
a) from – address from whom the response message has come
b) data- data buffer received
c) len- length of data received

Returns:
0 : on success -1: error

This function will process a request message and call send_update to send an update message to the requester.

2.3) Output
A timer will timeout after every 30 seconds and send_update will be called to send regular updates.

int send_update(int update_type)

Argument:
a) update_type- either triggered update or regular update –TRIGUPDATE/REGUPDATE

Returns:
0: success, -1: error

This function will generate the appropriate type of update – either triggered or regular. It will send this update to all the addresses in the neighbour list. TO generate regular update message , construct_regular_update function is called.

3) Timer Description:

When an entry is updated or newly added in route table, a timer with interval 30 sec(timeout_tmr) is started. If the timer times out, that entry is made invalid and another timer is started for 120 sec(garbage_tmr). If this timer also times out, that entry is deleted.
Callback functions for both timers are implemented.

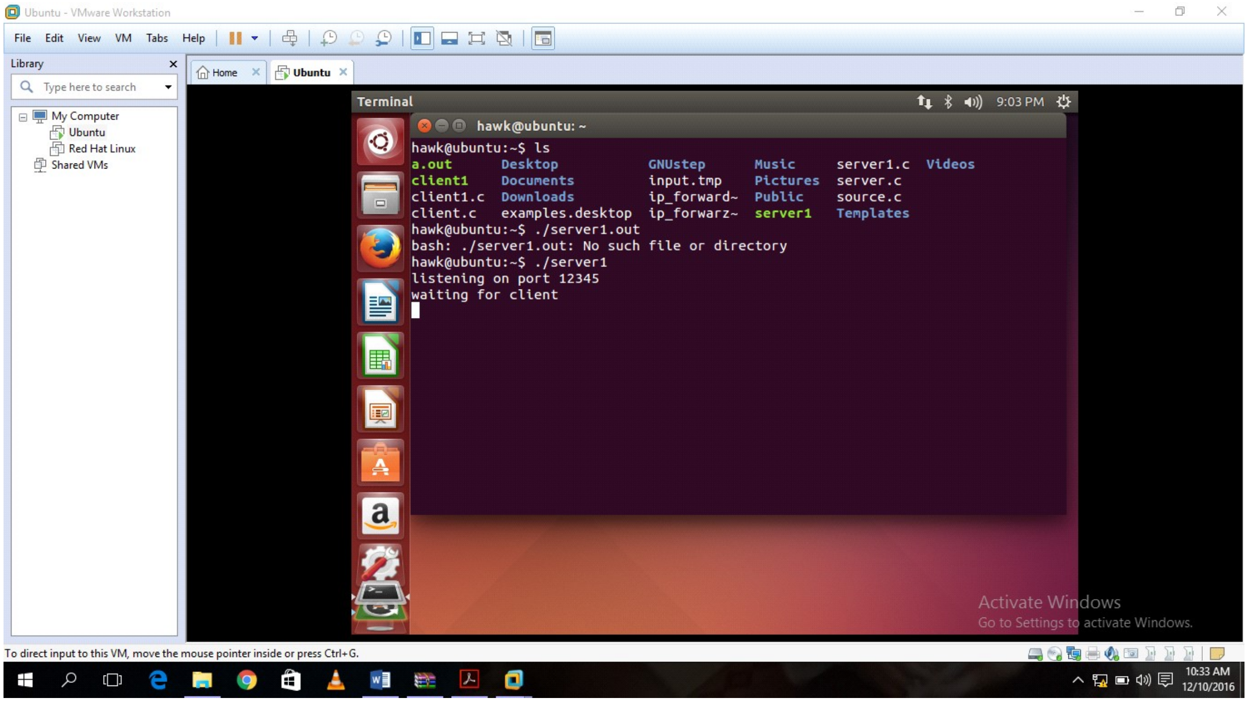
4)CLI Description:
CLI currently provides option for add route and display route table.
CLI runs in a separate thread.

1]add route: It will ask for following from the user
Destination:
Metric:
Nexthop:

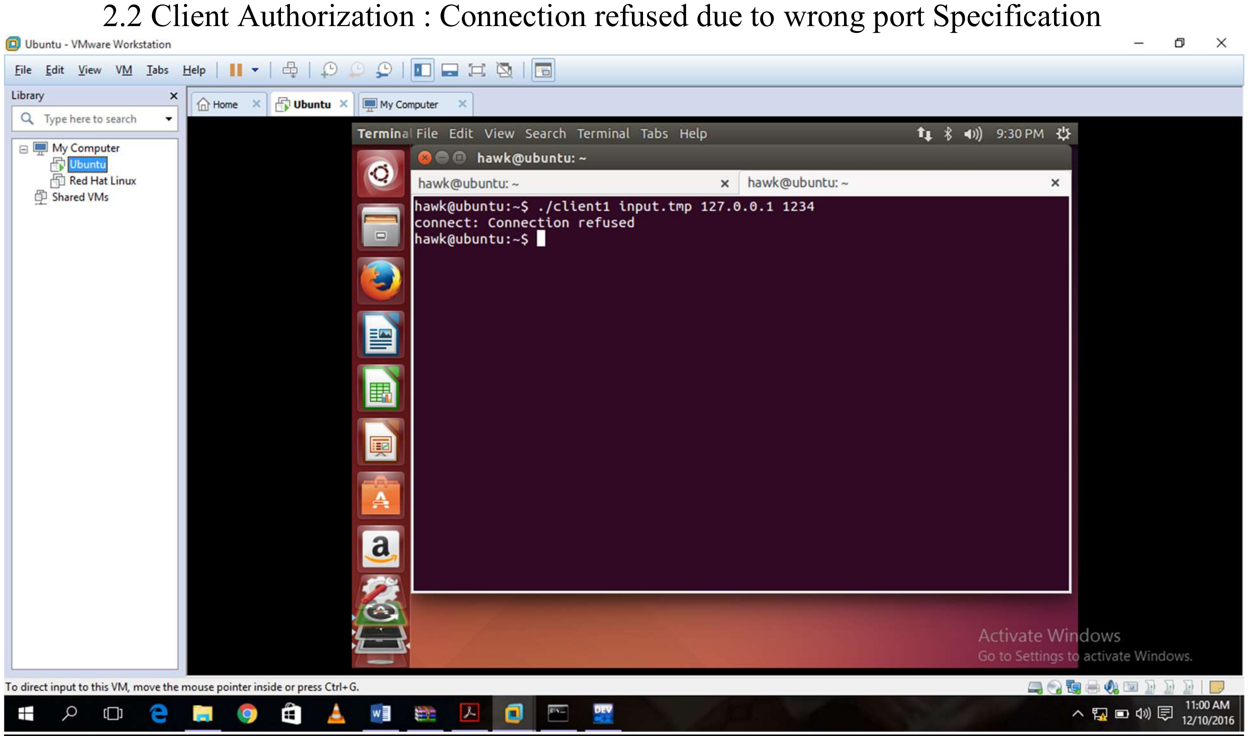
2]show route table

5. Output Screens

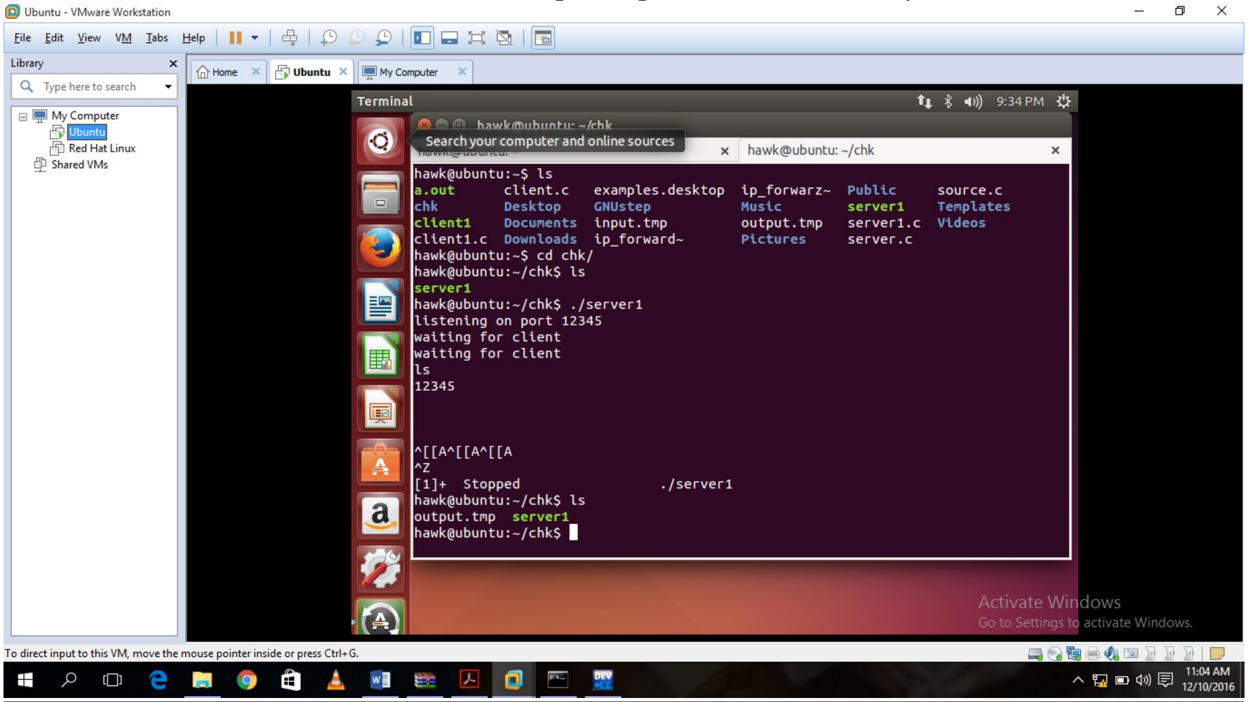
2.1 Server Establishment



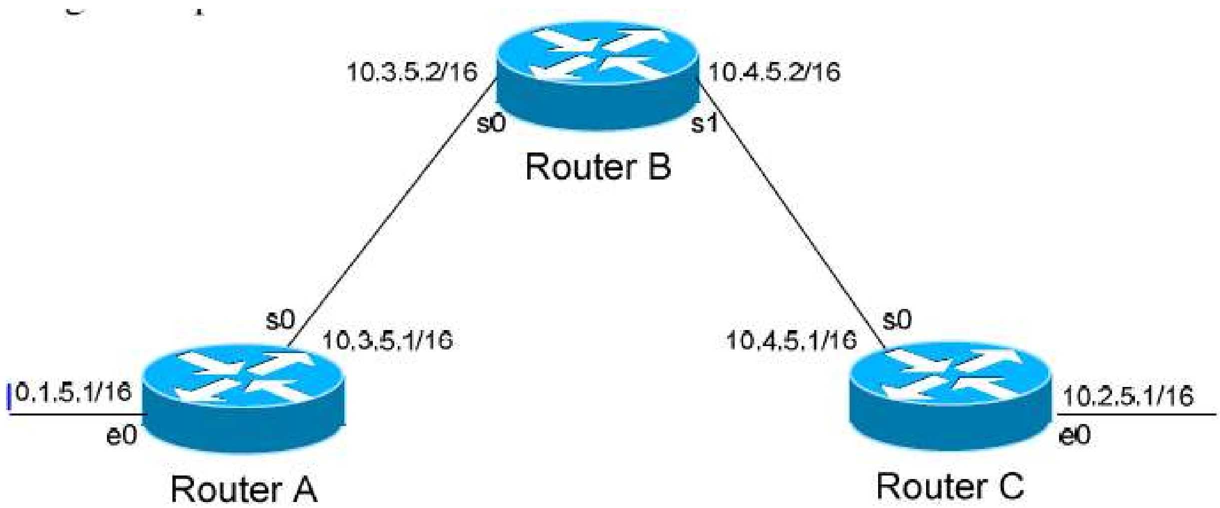
2.2 Client Authorization : Connection refused due to wrong port Specification



5.3 File creation on Server : file name Output.tmp created on server by client



Limitations



This particular scenario will *still* work when using RIPv1, despite the fact that we've subnetted the major 10.0.0.0 network. Notice that the subnets are contiguous (that is, they belong to the same major network), and use the same subnet mask.
When Router A sends a RIPv1 update to Router B via Serial0, it will not include the subnet mask for the 10.1.0.0 network. However, because the 10.3.0.0 network is in the same major network as the 10.1.0.0 network, it will **not summarize** the address. The route entry in the update will simply state "10.1.0.0".

Conclusions

- Already implemented routers are very costly and energy inefficient, by implementing our proposed system we will be able to overcome the energy inefficiencies .
- Our router is cost efficient because we will be using an open source hardware platform ,Raspberry Pi and open source OS Linux on top of which our algorithm will be implemented.

References

- [1] J.N.Al-Kalkari and A.E.Kamal, *Routing Techniques in Wireless Sensor Networks: A Survey.*
- [2] S.Ferdoush and X.li*, Wireless Sensor Network System Design using Raspberry Pi and Arduino for Environmental Monitoring Applications.*
- [3] R.A.Santos,A.Gonzalez,L.villasenor,M.G.Ruiz,V.Rangel and A.Edwards.*Analysis of Topological and Geographical Multicast Routing Algorithms on Wireless Ad Hoc Networks.*
- [4] R. A. Santos, A. González, L. Villaseñor, M. García-Ruíz, V. Rangel ,A. Edwars1 Analysis of Topological and Geographical Multicast Routing Algorithms onWireless Ad Hoc Networks
- [5]Mesut G¨unes¸, Udo Sorges, Imed Bouazizi, ARA – The Ant-Colony Based Routing Algorithm for MANETs
- [6]Routing Information Protocol,A. Balchunas
- [7]http://firebug.sourceforge.net/surgepics.html
- [8]http://www.binarytides.com/socket-programming-c-linux-tutorial/
- [9]http://www.tenouk.com/cnlinuxsockettutorials.html
- [10]http://stackoverflow.com/questions/11408609/linuxbluetoothprogramming-in-c
- [11]http://www.cs.unh.edu/cnrg/people/gherrin/linuxnet.html#tth_sEc7.2.5
- [12]http://raspberrypihq.com/how-to-turn-a-raspberry-pi-into-a-wifirouter/
- [13]http://www.computernetworkingnotes.com/ccna-study-guide/rip-tutorial-basic-operation-of-rip-protocol.html