

DEVELOPMENT OF HIGH SPEED HARDWARE ACCELERATED LAYER 7 ROUTER



By

Haider Khalid

M. Haseeb Javed

Abdul Moeed Rao

Rawal Khan Baloch

Submitted to the Faculty of Department of Electrical Engineering,
Military College of Signals, National University of Sciences and Technology,
Islamabad in partial fulfillment for the requirements of B.E Degree in
Electrical Engineering JUNE 2016

Abstract

DEVELOPMENT OF HIGH SPEED HARDWARE ACCELERATED LAYER 7 ROUTER

Our project is based on layer 7(application) routing technique. Improvement for Quality of Service (QoS) is of great necessity for the World Wide Web due to its increasing demand day by day . Through our router we have been able to generate congestion free and improved network availability for the end users along with onboard data processing for calculation of various valuable data statistics which if implemented onto the CPU would cause deterioration in the efficient memory usage . Present routing techniques depend upon IP checking and OSPF routing protocol which are unintelligent . Through the Layer 7 router we have been able to forward data packets based on their routing protocols for intelligent bandwidth allocation and also data processing to meet the data statistic needs of the day . The layer 7 router is made on ARDUINO board which allows Software Defined Networking (SDN) on it .

CERTIFICATE FOR CORRECTNESS AND APPROVAL

It is certified that the work contained in the thesis for DEVELOPMENT OF HIGH SPEED HARDWARE ACCELERATED LAYER 7 ROUTER Haider Khalid, M. Haseeb Javed, Abdul Moeed Rao and Rawal Khan under the supervision of Lec Abdul Moiz Ahmed Pirkani for partial fulfillment of Degree of Bachelor of Electrical Engineering is correct and approved.

Approved by

Lec Abdul Moiz Ahmed Pirkani

**Electrical Engineering Dept.
MCS**

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent.
To our parents, without whose unflinching support and unstinting cooperation, a work
of this magnitude would not have been possible.

ACKNOWLEDGEMENTS

We bow in gratitude to Allah Almighty for giving us strength and knowledge to accomplish this task as nothing happens without his will.

The group is indebted by the immense help and moral support given to us by our parents as it would not have been possible without their prayers.

The team also likes to thank our project supervisor, Lec Abdul Moiz Ahmed Pirkani, without his support and encouragement; it would not have been possible to complete this project.

Table of Contents

Chapter 1: Introduction	1
1.1 Background.....	1
1.1.1 Computer Network	1
1.1.2 Router	1
1.1.3 OSI Layer	2
1.2 Devices that make networking possible.....	2
1.2.1 Hub.....	2
1.2.2 Gateway	3
1.2.3 Router	3
1.2.4 Switch.....	3
1.2.5 Bridge.....	4
1.2.6 Repeater.....	4
1.2.7 Multiplexer.....	5
1.2.8 Modem (Modulator/Demodulator).....	5
1.3 Future of Networking	5
1.4 Problem Statement	7
1.5 Proposed Solution	8
Chapter 2: Background Study	9
2.1 Application Aware Routing.....	9
2.1.1 Why Application Aware Routing.....	10
2.2 Scope of Work	11
2.3 Software Defined Networks (SDN)	11
2.4 ARDUINO	12
2.4.1 Onboard Arduino Components.....	12
2.4.2 ARDUINO Mega.....	13
2.4.3 ARDUINO Mega PIN's.....	13
2.4.4 Ethernet Shield	15
2.4.5 SD Card Shield.....	17

2.5 Uniform Resource Locator.....	17
Chapter 3: Design.....	20
3.1 Working Points	20
3.2 Working	20
3.2.1 Device setup.....	20
3.3 Hardware Processes	22
3.3.1 Flow Chart.....	22
3.4 Software Processes.....	24
3.4.1 Flow Chart.....	24
3.5 Academic Objectives	25
3.6 Special Skills Required	26
3.7 Work Done.....	26
3.7.1 Hardware	26
3.7.2 Hardware PCB Design	27
3.7.3 Hardware Configuration	27
3.7.4 Software.....	30
3.7.5 Software Programs	30
3.7.6 Results.....	32
3.8 Visual Indications.....	34
3.8.1 Hardware LCD Display.....	34
3.8.2 Application Indications	36
3.9 Deliverables	37
3.9.1 Reflective Comments.....	37
Chapter 4: Applications.....	39
4.1 Applications	39
4.1.1 Server Load Balancing.....	39
4.1.2 Content Based Routing	39
4.1.3 URL Blocking.....	39
4.1.4 Multi-layer Switching	39
4.1.5 High Availability Networks	39
4.1.6 Network Monitoring Device	39
4.2 Conclusion	40

Chapter 5: Resources Required	41
Chapter 6: References	42
Chapter 7: Bibliography	43
Appendix A	44

List of Figures

Figure 1: Computer Network Diagram.....	1
Figure 2: Generic Router Symbol	1
Figure 3: OSI Layer Model.....	2
Figure 4: Growth of Network Bandwidth Compared to the Growth of the Processing Power	6
Figure 5: World Wide Data Growth over the Years	7
Figure 6: Hour Glass Shape figure of layered protocol stack.....	9
Figure 7: Bandwidth Depiction	11
Figure 8: ARDUINO Mega.....	15
Figure 9: ARDUINO Pin Configuration.....	15
Figure 10: Ethernet Shield	16
Figure 11 : Stacked Ethernet Shield on ARDUINO Board	16
Figure 12: SD Card Shield Module	17
Figure 13: URL Breakdown.....	19
Figure 14: Device Set-up and Working Block Diagram	20
Figure 15: Hardware Process Flow-chart.....	22
Figure 16: Software Process Flow-chart	24
Figure 17: PCB Schematic Diagram	27
Figure 18: ARDUINO Mega Pin Configuration.....	27
Figure 19: LCD Pin Configuration	28
Figure 20: ARDUINO Pro-Mini Pin Configuration.....	28
Figure 21: Ethernet Shield Pin Configuration	29
Figure 22: Card Module Pin Configuration	29
Figure 23: Sending Module	31
Figure 24: Receiving Module	31
Figure 25: Data Statistics Requested by the Server	32
Figure 26: Statistics Requested by the Server and their Plotting	33
Figure 27: Data Statistics at the Receiving End.....	33
Figure 28: Final Hardware	37

List of Tables

Table 1 : ARDUINO Mega Specifications	13
Table 2 : SD Card Shield Specifications	17
Table 3 : Hardware LCD Display Notations	36
Table 4 : Software Application Notations	36

Chapter 1: Introduction

1.1 Background

1.1.1 Computer Network

A **computer network** is a model which allows computers to exchange data among themselves. In computer networks, networked computing devices pass data to each other along network links through which data is transferred in the form of packets. The connections between nodes are established using either cable media or wireless media. The best form of extensive computer network is the Internet.



Figure 1: Computer Network Diagram

1.1.2 Router

A router is a networking device that forwards data packets between computer networks. A router is connected to two or more data lines from different networks. When a data packet comes in on one of the lines, the router reads the address information in the packet to determine its ultimate destination. Then, using information in its routing table or routing policy, it directs the packet to the next network on its journey.

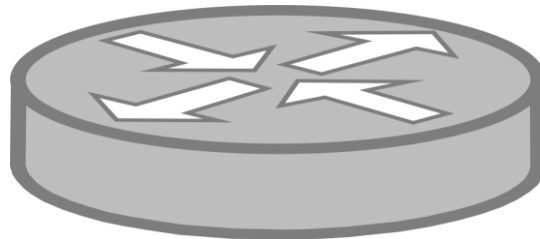


Figure 2: Generic Router Symbol

1.1.3 OSI Layer

The OSI, or Open System Interconnection, model defines a networking framework to implement protocols in seven layers. Control is passed from one layer to the next, starting at the application layer in one station, and proceeding to the bottom layer, over the channel to the next station and back up the hierarchy.

Application	<ul style="list-style-type: none">• End User layer• Applications present data to the user e.g. Web Browsers• URL, Protocol, Payload (Data)
Presentation	<ul style="list-style-type: none">• Formatting/Syntax• Encryption/Decryption• Compression/Decompression
Session	<ul style="list-style-type: none">• Session Establishment; Logging in/out• Session Security• Time slice to be given to an ongoing session
Transport	<ul style="list-style-type: none">• Segmentation• Acknowledgement• Handshaking (Matching protocols)
Network	<ul style="list-style-type: none">• IP Address
Data Link	<ul style="list-style-type: none">• MAC Address
Physical	<ul style="list-style-type: none">• Transmission media

Figure 3: OSI Layer Model

1.2 Devices that make networking possible

1.2.1 Hub

- OSI Layer 1 Device
- Provides a physical connection from one node to multiple nodes
- Broadcasts the received signal to all outgoing ports
- Active Hubs act as repeaters i.e. they strengthen signals passing through them
- Passive Hubs simply direct signals to multiple nodes without regeneration
- Originally this device was called a concentrator since it consolidated the cable runs from all network devices

1.2.2 Gateway

- OSI Layers 4 - 7 Device
- Used to interface one network node with another network node that uses different protocols
- In order to communicate with a host on another network, an IP host must be configured with a route to the destination network. If a configuration route is not found, the host uses the gateway (default IP router) to transmit the traffic to the destination host. The default gateway is where the IP host sends packets that are destined for remote networks. If no default gateway is specified, communication is limited to the local network. Gateways receive data from a network using one type of protocol stack and then remove that protocol stack and repackage it with the protocol stack that the other network uses

1.2.3 Router

- OSI Layer 3 Device
- Determines the next hop/network node towards which a network packet has to be directed to in order to reach its final destination
- Works on the basis of IP and MAC Addresses, Longest Prefix Matching, Address Resolution Protocol, OSPF (Open Shortest Path First) Protocol, etc. to figure out the shortest path to be taken for speedy packet delivery
- Maintains routing tables which contain a list of IP Addresses and MAC Addresses of the devices in nearby vicinity of the router
- In static routers, routing tables are to be maintained manually by the users
- In dynamic routers, routing tables are maintained at runtime by the router as it listens to network traffic (ARP Packets)

1.2.4 Switch

- OSI Layer 2 Device
- A device which maintains a table of MAC Addresses of network nodes connected to its outgoing ports

- It directs network packets to their destined network nodes only (deduced by MAC Address present in Data Link Layer Header) unlike hubs which direct packets to all outgoing ports (broadcasting)

1.2.5 Bridge

- OSI Layer 2 Device
- Used for connecting (creating a link/bridge) multiple network segments together
- It allows computers on either segment to access resources on the other. They can also be used to divide large networks into smaller segments. Bridges have all the features of repeaters, but can have more nodes, and since the network is divided, there is fewer computers competing for resources on each segment thus improving network performance.
- Bridges can also connect networks that run at different speeds, different topologies, or different protocols. But they cannot join networks on different networking standards. Bridges operate at both the Physical Layer and the MAC sub-layer of the Data Link layer. Bridges read the MAC header of each frame to determine on which side of the bridge the destination device is located, the bridge then repeats the transmission to the segment where the device is located.

1.2.6 Repeater

- OSI Layer 1 Device
- Regenerates/amplifies a digital signal while transferring it from one node to another
- Multilayer switch:
- Like OSI Layer 2 switch which uses MAC Addresses to direct packets to their destinations, multilayer switches work on different layers of the OSI model and use the features present in those layers to direct packets to their destinations
- Firewall:
- Hardware/Software form
- Used to prevent certain types of communication forbidden by network policy

1.2.7 Multiplexer

- A device which combines several electrical signals into a single signal
- Network interface controller (NIC):
- A device responsible for providing a connection between wire-based networks
- Wireless network interface controller:
- A device responsible for providing a connection between radio-based networks (WiFi)

1.2.8 Modem (Modulator/Demodulator)

- OSI Layer 1 Device
- A device responsible for modulating a carrier analog signal to encode digital information in transmission mode; While receiving, it demodulates the received analog carrier wave to decode/extract the digital signal from it
- Used in networks where the network nodes are communicating over telephone cables
- Wireless Access Point (WAP)
- A wireless network adapter card with a transceiver sometimes called an access point, broadcasts and receives signals to and from the surrounding computers and passes them back and forth between the wireless computers and the cabled network.
- Access points act as wireless hubs to link multiple wireless NICs into a single subnet. Access points also have at least one fixed Ethernet port to allow the wireless network to be bridged to a traditional wired Ethernet network (often, this is a link to the World Wide Web/Internet)

1.3 Future of Networking

- The future is never easy to predict
- Ever since the advent of the World Wide Web, finding better, more efficient, reliable, faster and more secure means of data transfer has been the major goal

- The amount of data worldwide has grown exponentially especially in the past few years mostly due to media sharing on social networking web sites and due to the fact that almost all conventional manual systems of data keeping have evolved to large digital data centers
- We get to hear new terminologies that were never heard of before: Big Data, Business Intelligence...
- Computer Networks should be ready to assure the speedy transfer of such huge amounts of data the world over
- We have evolved from 14.4kbps internet connection speeds to internet connection speeds in the range of hundreds of megabits
- We are looking forward to attaining speeds in the range of gigabits now
- We are also seeing a shift from hardware based network management to the software era of network management
- We need intelligent (intelligent in a sense that they optimize bandwidth at runtime while minimizing redundancy) networking devices that operate at extremely high speeds, but are yet cost effective

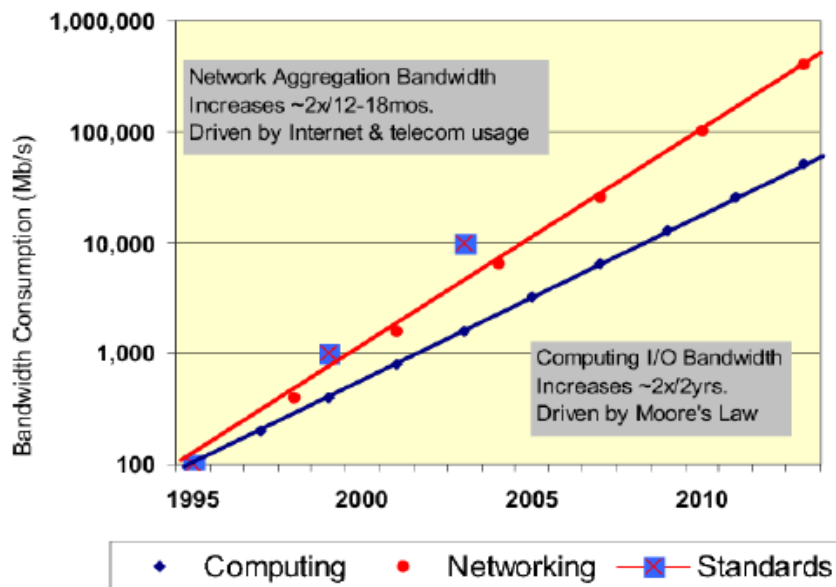


Figure 4: Growth of Network Bandwidth Compared to the Growth of the Processing Power

Worldwide Data Growth

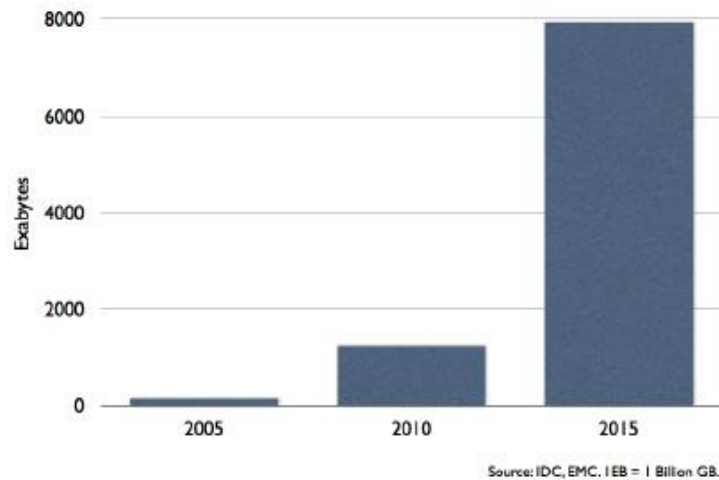


Figure 5: World Wide Data Growth over the Years

1.4 Problem Statement

Improved Quality of Service (QoS) experience for World Wide Web (WWW) can be provided to the end users by utilizing rich set of features and services of upper layers of the OSI architecture. In the current scenario, WWW is solely based on IP Address (IPv4) and Open Shortest Path First (OSPF) routing protocol and no onboard data processing features are available on the router. This renders it unintelligent because path bandwidth is not considered during routing also data processing unavailability renders it to abstain functionality from many desired needs required of today of data statistics.

1.5 Proposed Solution

QoS deterioration due to routing protocol failure for low bandwidth or high traffic paths is proposed to be improved by directing (routing) packets based on their application layer protocol and thereby improving the user experience.

- Layer 7 (application layer) router is proposed which would specify the application paths based on their usage and provide congestion control and improved network availability.
- A layer 7 router is sufficiently intelligent to analyze network layer packets and filter out packets containing the desired URL.
- After extraction of the desired URL, packet's path is allocated based on the required bandwidth.
- Browser statistics and application forwarding would also be achieved through calculation and proper assessment of the data from the data storage module.

Chapter 2: Background Study

2.1 Application Aware Routing

The network routers that we normally hear about and utilize (often, to act as a wireless access point) in our daily lives are OSI Layer 3 Routers i.e. they work at the third layer of the OSI model by routing packets based on the information attained from layer 3 and layer 2 headers. They route packets by using the destination IP Addresses and MAC Addresses and by using OSPF (Open Shortest Path First) Protocol, etc. to route packets to their destinations using the shortest available paths. These routers look up the next hop by running searches in the routing tables that they maintain. They perform longest prefix matching and address resolution protocol requests to figure out the next hops and to update their routing tables.

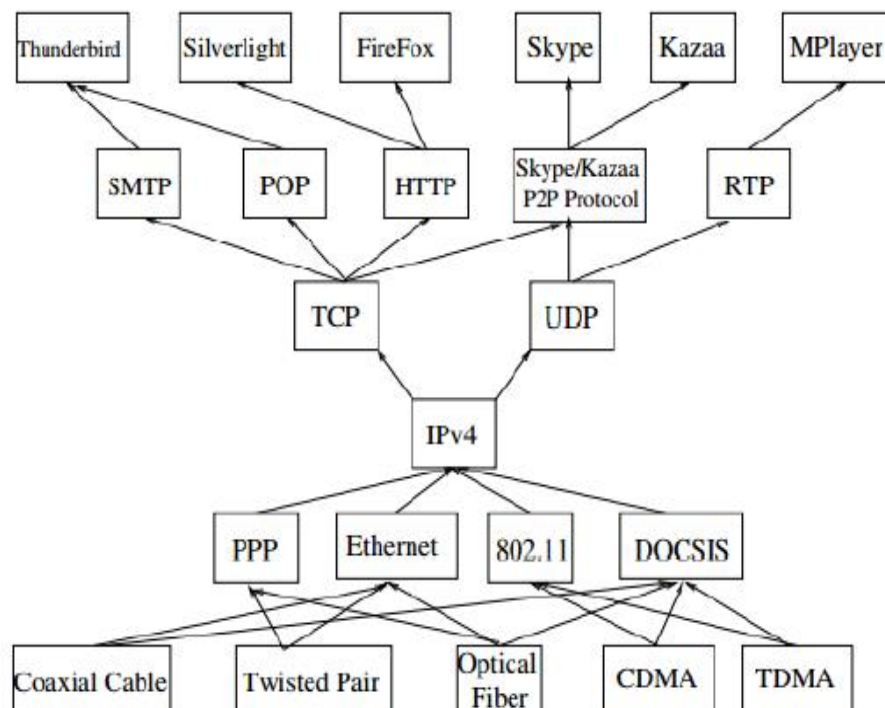


Figure 6: Hour Glass Shape figure of layered protocol stack

It has recently been in wide speculation that the protocol stack has acquired an hour-glass shaped figure i.e. more distinguishing features are saturated at the top and bottom layers whereas very few features distinguishing features are present at the middle layers.

Talking about the third layer at which conventional routers work, the IP address is the only distinguishing feature which makes routing possible from one hop to another. As we move to seventh layer, we get several features which can help in distinguishing packets and requests and subsequently the possibility of using a new approach towards routing. The application layer header provides us with the protocol and the URL of the packet. We believe that by extracting these features from packets, we can route received packets to servers that are specified to handle those application requests e.g. routing video streaming requests to a specific server which aims to provide high bandwidth, while simple HTML and CSS requests are routed to a server that provides low bandwidth. In networks, even Nano-seconds spent at each hop matter when network packets have long routes to travel. We aim to use bandwidth efficiently, achieve higher server cache hit rates and improve user QoS experience.

2.1.1 Why Application Aware Routing

Today's networks are forced to be application-aware, to improve user experience, provide service differentiation, and reduce operational costs. Application awareness is the result of gained intelligence about Layer 4 to Layer 7 protocol attributes and delivery requirements. The following are some use cases where the network has to be application-aware:

- A user watching videos requires more network bandwidth compared to a user browsing the Internet. The network has to dynamically adjust the bandwidth allocation to users based on the application, instead of statically allocating capacity.
- Gaming applications require lower network latency compared to applications transferring log files over the network. The network has to adapt to the application's latency needs.
- A router forwarding request to a content server (cache or streaming server) doesn't know whether the content resides in the content server. This increases the number of hops a request has to traverse before reaching the content server that can serve the request.

2.2 Scope of Work

The proposed project aims at making the router sufficiently intelligent to analyze network layer packets and filter out packets containing the desired URL (based on their application layer protocol). After extraction of the desired URL, packet's path is allocated based on the required bandwidth. Packets for low bandwidth requirement applications will be routed to low bandwidth providing servers while those for bandwidth hungry applications will be routed to high bandwidth providing servers. Figure below conceptualizes the model:

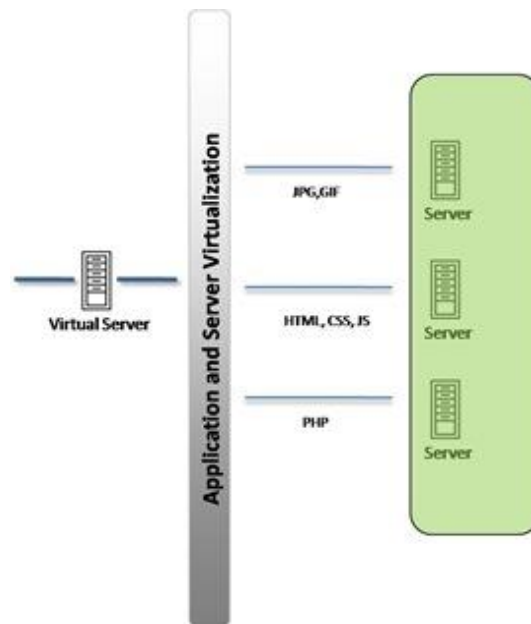


Figure 7: Bandwidth Depiction

2.3 Software Defined Networks (SDN)

Software Defined Networking is a new approach towards designing, implementing and managing networks. SDN basically is an abstraction between the control and forwarding planes of network devices. The control plane makes decision about where to send packets while the forwarding plane puts these packets on the physical media. SDN makes it easier to manage and optimize each of the control and forwarding planes. SDN acts as a controller which encompasses the entire network. SDN allows networking devices to be highly programmable and flexible to suit the needs of the environment in which the bare-

metal networking device is placed. Unlike purpose built ASIC Networking Equipment, SDN offers flexibility and modification in the controlling algorithms. The SDN controller can be a network operating system (NOS). SDN is often referred to as the software-era of networks.

2.4 ARDUINO

Arduino is an open-source computer hardware and software board. The board is based on a family of microcontroller board designs. These systems provide sets of digital and analog I/O pins that can be interfaced to various expansion boards ("shields") and other circuits. The boards feature serial communications interfaces, including USB on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino platform provides an integrated development environment (IDE) based on the Processing project, which includes support for C, C++ and Java programming languages.

2.4.1 Onboard Arduino Components

An Arduino board consists of an Atmel 8-, 16- or 32-bit AVR microcontroller with complementary components that facilitate programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which lets users connect the CPU board to a variety of interchangeable add-on modules known as shields. In our case we would be using the Ethernet shield. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an I²C serial bus—so many shields can be stacked and used in parallel.

At a conceptual level, when using the Arduino software stack, all boards are programmed over an RS-232 serial connection, but the way this is implemented varies by hardware version. Serial Arduino boards contain a level shifter circuit to convert between RS-232-level and TTL-level signals. Current Arduino boards are programmed via USB, implemented using USB-to-serial adapter chips such as the FTDI FT232RL. Some boards, such as later-model Uno's, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware (itself reprogrammable via its own ICSP header).

2.4.2 ARDUINO Mega

The Arduino Mega is a microcontroller board based on the ATmega1280 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	128 KB of which 4 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Table 1 : ARDUINO Mega Specifications

2.4.3 ARDUINO Mega PIN's

Each of the 54 digital pins on the Mega can be used as an input or output, using `pinMode()`, `digitalWrite()`, and `digitalRead()` functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor

(disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.
- External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the `attachInterrupt()` function for details.
- PWM: 2 to 13 and 44 to 46. Provide 8-bit PWM output with the `analogWrite()` function.
- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- I2C: 20 (SDA) and 21 (SCL). Support I2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I2C pins on the Duemilanove or Diecimila.

The Mega has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and `analogReference()` function.



Figure 8: ARDUINO Mega

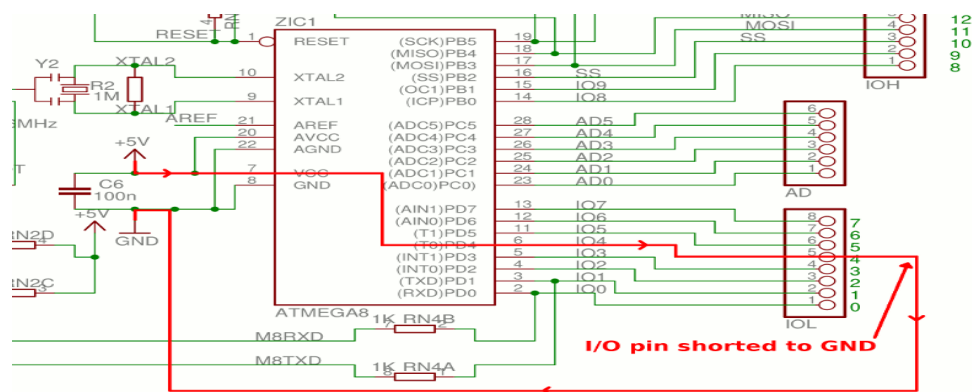


Figure 9: ARDUINO Pin Configuration

2.4.4 Ethernet Shield

The Arduino Ethernet Shield connects the Arduino to the internet in mere minutes.

Requirements:

- Operating voltage 5V (supplied from the Arduino Board)
- Ethernet Controller: W5100 with internal 16K buffer
- Connection speed: 10/100Mb
- Connection with Arduino on SPI port

The Arduino Ethernet Shield allows an Arduino board to connect to the internet. It is based on the Wiznet W5100 ethernet chip. The Wiznet W5100 provides a network (IP)

stack capable of both TCP and UDP. It supports up to four simultaneous socket connections. Use the Ethernet library to write sketches which connect to the internet using the shield. The Ethernet shield connects to an Arduino board using long wire-wrap headers which extend through the shield. This keeps the pin layout intact and allows another shield to be stacked on top.

The current shield has a Power over Ethernet (PoE) module designed to extract power from a conventional twisted pair Category 5 Ethernet cable.



Figure 10: Ethernet Shield



Figure 11 : Stacked Ethernet Shield on ARDUINO Board

2.4.5 SD Card Shield

SD card shield provides a storage space for the Arduino. Users can read/write SD card via Arduino's built-in SD library. It supports SD, SDHC and Micro SD cards. It will only occupy the SPI port of your Arduino. It combines the standard SD slot and the Micro SD slot into a standard one, the included adaptor enables using of Micro SD cards. You can stack on other shields that work with the unused pins. Additionally, the preformed I2C and UART port facilitates your connection with Grove modules.

Item	Min	Typical	Max	Unit
Voltage	3.5	5.0	5.5	V
Current	0.159	100	200	mA
Supported Card Type	SD card(<=32G); Micro SD card(<=32G); SDHC card(<=32G)			/
Dimension	68.7x53.5x19.00			mm
Net Weight	14.8			g

Table 2 : SD Card Shield Specifications



Figure 12: SD Card Shield Module

2.5 Uniform Resource Locator

A Uniform Resource Locator (URL), commonly informally termed a web address (which term is not defined identically) is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it. A URL is a specific type of Uniform Resource Identifier (URI). A URL implies the means to access an indicated resource, which is not true of every URI. URLs occur most commonly to reference web

pages (http), but are also used for file transfer (ftp), email (mailto), database access (JDBC), and many other applications.

It comprises:

- The scheme, consisting of a sequence of characters beginning with a letter and followed by any combination of letters, digits, plus (+), period (.), or hyphen (-). Although schemes are case-insensitive, the canonical form is lowercase and documents that specify schemes must do so with lowercase letters. It is followed by a colon (:). Examples of popular schemes include http, ftp, mailto, file, and data. URI schemes should be registered with the Internet Assigned Numbers Authority (IANA), although non-registered schemes are used in practice.
- Two slashes (/): This is required by some schemes and not required by some others. When the authority component (explained below) is absent, the path component cannot begin with two slashes.
- An authority part, comprising:
 - An optional authentication section of a user name and password, separated by a colon, followed by an at symbol (@)
 - A "host", consisting of either a registered name (including but not limited to a hostname), or an IP address. IPv4 addresses must be in dot-decimal notation, and IPv6 addresses must be enclosed in brackets ([]).
 - An optional port number, separated from the hostname by a colon
- A path, which contains data, usually organized in hierarchical form that appears as a sequence of segments separated by slashes. Such a sequence may resemble or map exactly to a file system path, but does not always imply a relation to one. The path must begin with a single slash (/) if an authority part was present, and may also if one was not, but must not begin with a double slash.

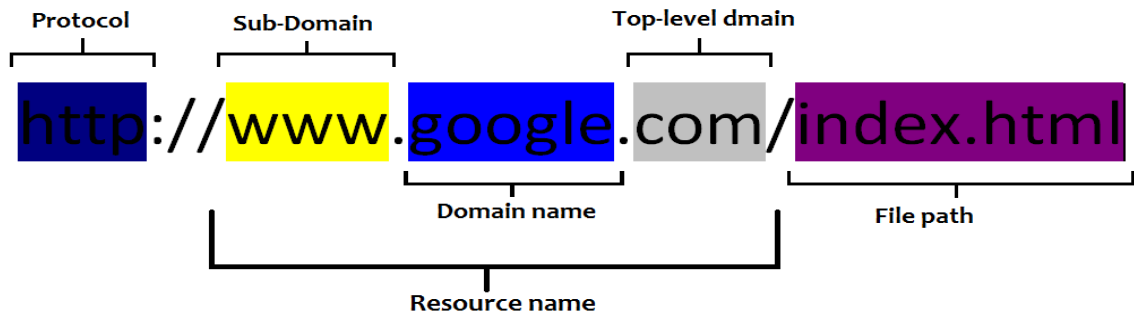


Figure 13: URL Breakdown

Chapter 3: Design

3.1 Working Points

The proposed project works on the following fronts:

- Prototyping of high-speed, hardware-accelerated networking systems by reviewing literature provided by various open source platforms for hardware and software development.
- Development and Integration of Hardware Accelerated Networking Systems (ARDUINO) with suitable server system
- Development of URL extractor and testing with necessary software

Integration of ARDUINO, URL Extractor on the board for the development of Layer 7 Application based router

The project developed works as follows.

3.2 Working

3.2.1 Device setup

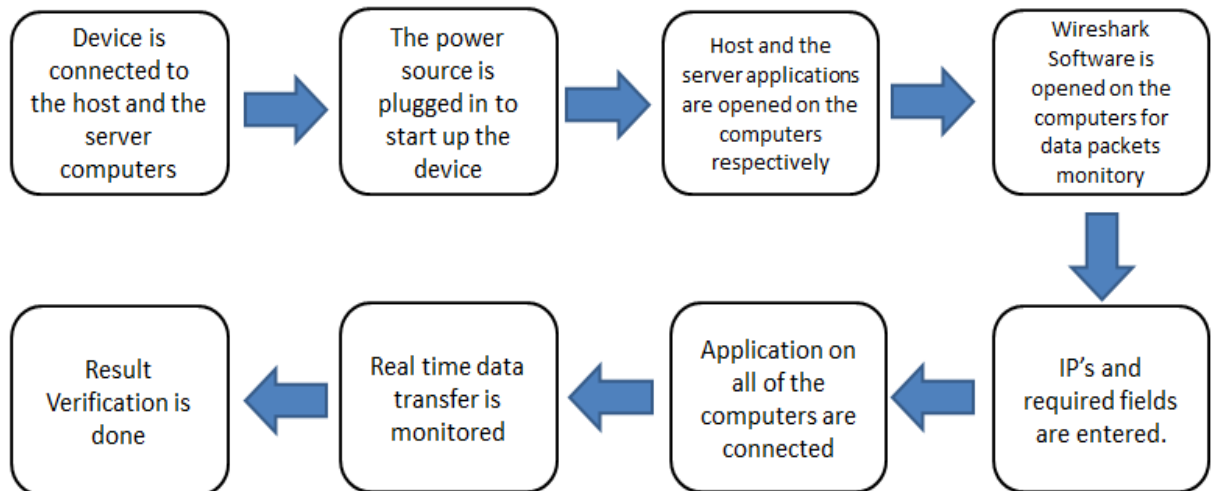


Figure 14: Device Set-up and Working Block Diagram

The developed device being a standalone device can work on any of the desired computer and provide the routing options efficiently. At present the device has the capability to be attached to a single server computer and 3 peripheral host computers where the device would send the packets differentiated on the basis of the protocol. All the computers are connected to the device using LAN cables which are connected through the Ethernet shields.

The connection process is as follows:

- The device is connected to the computers using the LAN cables.
- The device is connected to the power source.
- The server application is opened on the server computer to monitor the data in real time.
- The host application is opened on the host computer which is also a tool to monitor the real time data transfer.
- Wireshark software is opened on all of the computers for result verification.
- The IP's and other required fields are set on the computers.
- Applications on all of the computers are connected.
- Real time result verification is done.

3.3 Hardware Processes

The process followed by the data packets sent from the source computer is as follow:

3.3.1 Flow Chart

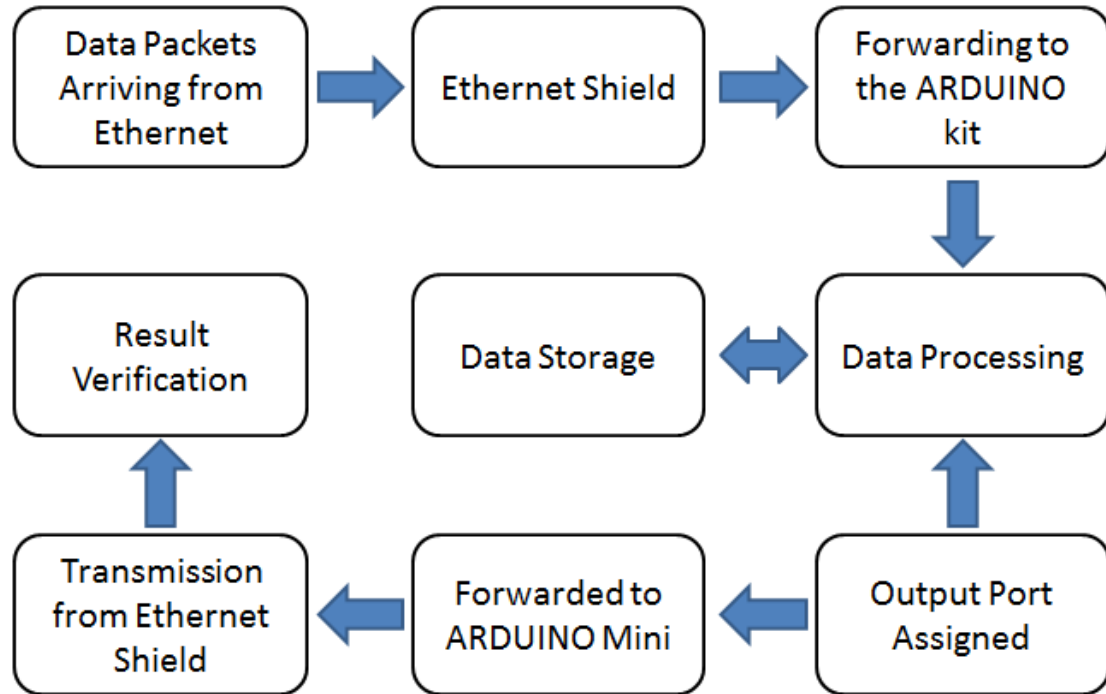


Figure 15: Hardware Process Flow-chart

- URL entered onto the address bar of the search engine would send data packets through the Ethernet cable to the designed device.
- The Ethernet shield stacked onto the Arduino board would be integrating the functionalities of the Ethernet onto the board.
- The data received from the Ethernet cable sent by the source computer would be received by the Ethernet shield and sent to the Arduino board for further processing.
- After the processing and the sorting out of the data, it is labeled to which port it would be sent to.

- The main ARDUINO is connected to three mini ARDUINO which are further connected to the Ethernet shields.
- The mini ARDUINO is added to the hardware as one mega ARDUINO cannot be directly to more than one same kind of the shield.
- The labeled data is forwarded to the mini ARDUINO from where to the respective shields and off to the destination.
- The labeled data is also sent to the SD card module for data storage.
- Received data would be tested on the Wireshark software to check the results.

The URL or the search term which entered in the address bar of the browser are formed in the form of a data packet and are sent to the Ethernet port through the Network interface card of the computer. The Ethernet port receives the data packets which are transmitted to the developed router device using the co-axial cable attached to the receiving side Ethernet module. The module receives the data packets and forwards them to the mega ARDUINO board for further processing. The board receives the data packets and implements the routing methodology on them. The data packets of the google, YouTube and Facebook are separated; labeled and sent to the specific ports for further data transmission. The labeled packets are also sent to the SD card module for data storage and this data can be called upon if requested by the server computer. The data calling capability is allowed to the server/administrator computer for security purposes. The Ethernet shields of the host computers are joined to the Mega ARDUINO board through three specific mini ARDUINO boards as a single board cannot be attached to more than two shields at one time. The Ethernet shields receive their specific data and transmit them to the destined computer through the coaxial cables attached.

3.4 Software Processes

Software processing done by the Arduino kit is as follows;

3.4.1 Flow Chart

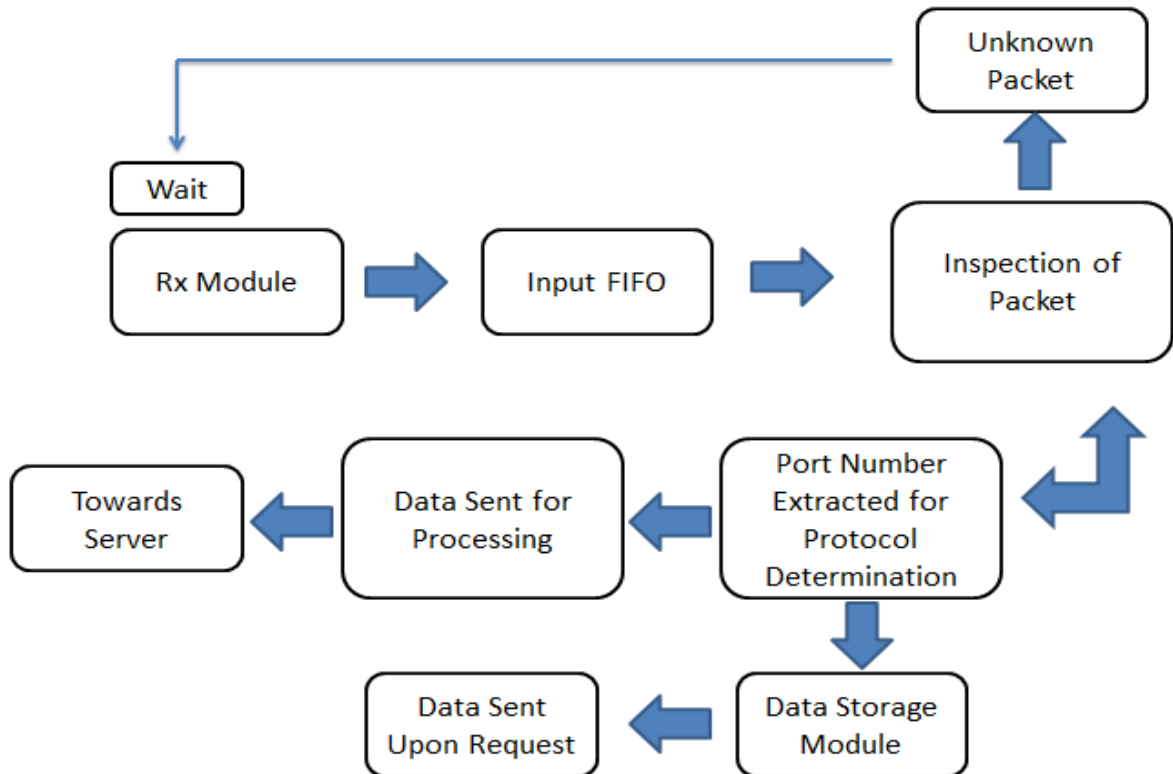


Figure 16: Software Process Flow-chart

- Received data packets from the source server/computer are received by the Ethernet shield.
- The Ethernet shield forwards the received data packets to the Arduino kit stacked on with the Ethernet shield.
- The data received is stored by the receiving module and processed by the input FIFO (First in First out). The input FIFO waits for the whole of the data packet to be received with the help of the receiving module. It does not forward the packet until the receiving is completed.

- The received data packet is forwarded to the data length inspection module which inspects the length of the data packet received to determine that it is a valid data packet and not an unverified data.
- The verified data will be sent to the inspection module.
- The data as processed by all of the OSI layers will be removed with the headers until the application layer data is achieved. The application layer data would be extracted with the help of the port numbers and the protocol would be determined by the help of the protocol numbers.
- The data extracted would also be stored in the data storage module for further calculation of data statistics i.e. throughput, browsing patterns etc.
- The output ports would be allotted on the basis of the port numbers detected.
- The processed data would be sent to the respective Ethernet shield through the mini ARDUINO's and off to the destination computer where further verification can be done through the Wireshark software.
- The data stored in the data storage module (SD card shield) can be called for to be displayed at any time when the router is attached to the computer.

Data received by the Mega ARDUINO board is received by the buffer module of the board which works on the FIFO methodology i.e. first in first out basis. The data packet which is received at first will be processed first and that received later would be done later. The received packets would be first inspected for length. If the length of the packet does not match the required fields it would be rejected. The accepted data packets would be sent to the next module where the data packets would be opened step by step through the OSI layers headers to reach the application layer data. The protocol of the packet would be checked out through the help of the port numbers as each of the protocol has its own protocol. The detected data packets are labeled and forwarded to their destined ports for further transmission. The data is also sent to the card module for storage.

3.5 Academic Objectives

During the course of this project we have developed sound knowledge and technical skills in the fields of:

- Data Networking and Embedded Systems
- Network Monitoring

- Network Security
- Software Defined Networks (SDN) and ARDUINO

The developed skills are both hardware and software in nature.

3.6 Special Skills Required

- ARDUINO
- Wireshark Expertise
- ARDUINO Programming

3.7 Work Done

3.7.1 Hardware

Hardware implementation of the project has been completed, which includes proper router assembly on PCB board. The PCB board was designed on the PROTEUS and AREAS software. The PCB board was etched and the ARDUINO boards and other components were soldered onto the board.

The Components used in the hardware include:

- ARDUINO Mega
- ARDUINO mini
- Ethernet shield
- SD card shield
- LCD display
- Timing IC
- Cooling Equipment (Fan's)

3.7.2 Hardware PCB Design

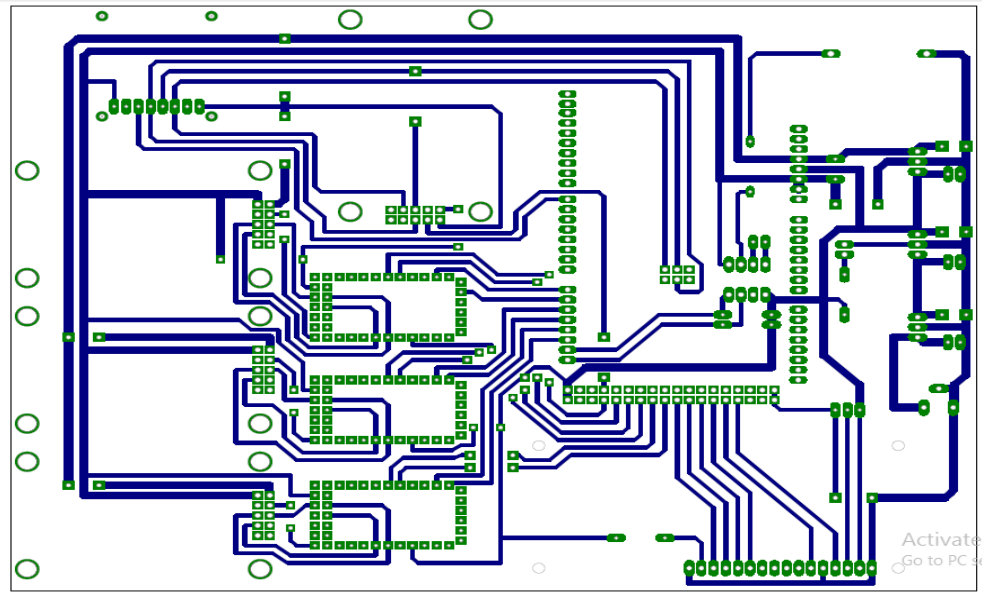


Figure 17: PCB Schematic Diagram

3.7.3 Hardware Configuration

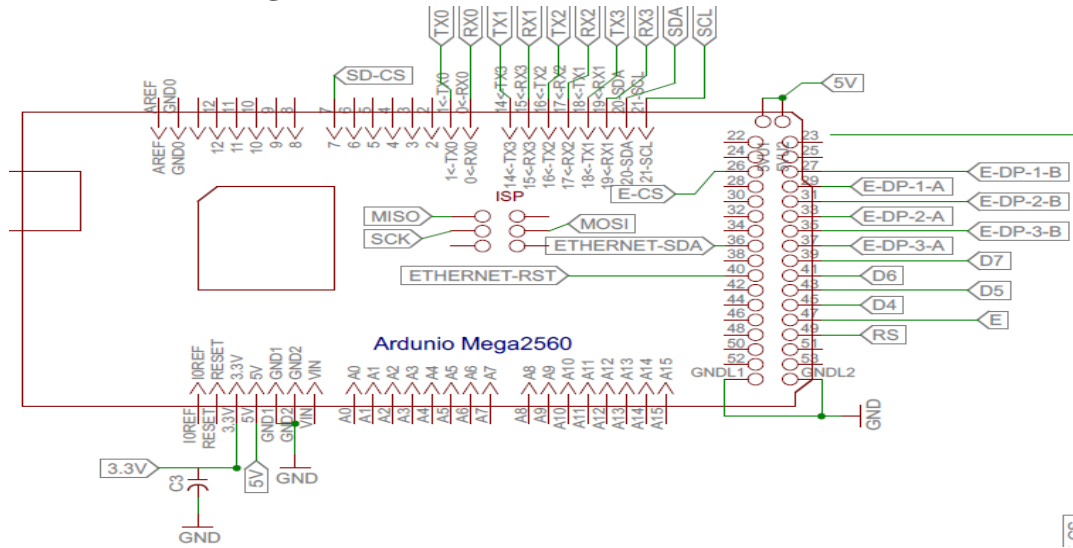


Figure 18: ARDUINO Mega Pin Configuration

Figure 18 shows the pin configuration of how the ARDUINO mega board is assembled onto the PCB board. The digital pins 1,2 and 14-21 have used for connecting the different mini ARDUINO pins. The pins 4-7 have been used to connect the mega board with the

LCD. The MOSI and MISO pins have been used to attach the ethernet shield modules with the board which are used for serial communication.

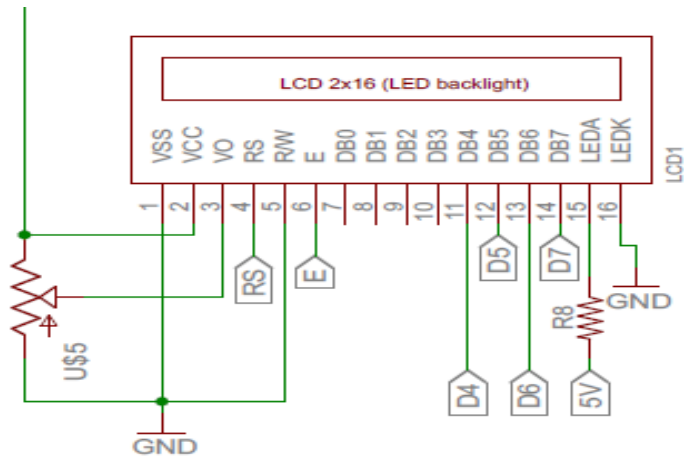


Figure 19: LCD Pin Configuration

Figure 19 shows the Pins which have been connected with the ARDUINO mega board on the PCB design. The LCD is 2x16 array LCD which takes a 3V input and 5 digital pins input.

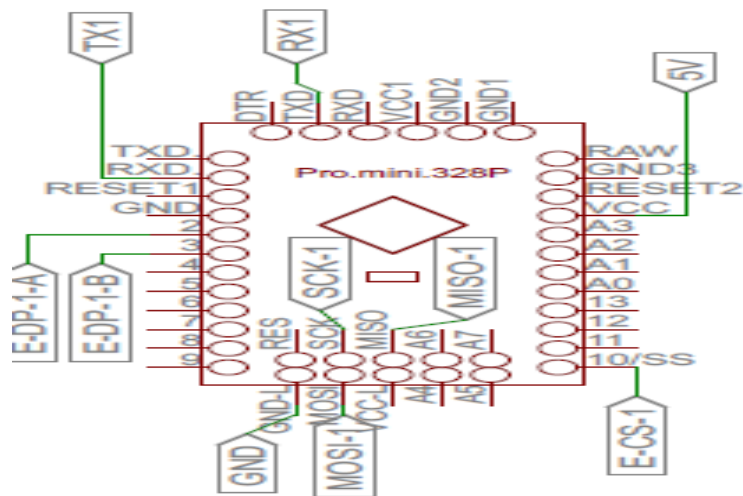


Figure 20: ARDUINO Pro-Mini Pin Configuration

Figure 20 shows the pin configuration of the ARDUINO mini pro of how it is attached to the mega ARDUINO board on the PCB board. It takes MOSI an MISO inputs for serial port communication as it acts as an interface between the Mega ARDUINO board and the ethernet shield module.

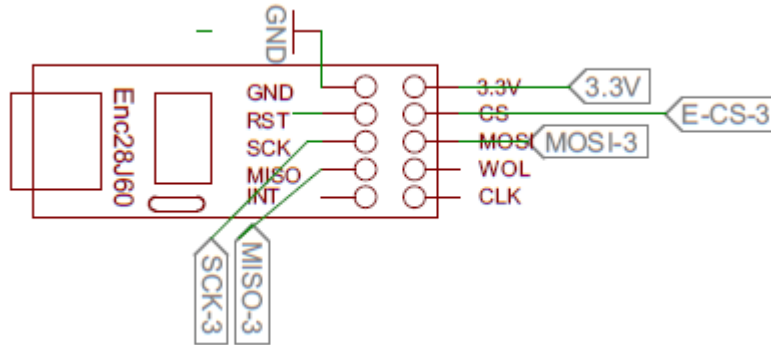


Figure 21: Ethernet Shield Pin Configuration

Figure 21 shows the pin configuration of the ethernet shield module of how it is attached to the mega ARDUINO and the mini ARDUINO board on the PCB board. The ethernet shield is used to connect the developed board with the specified computers.

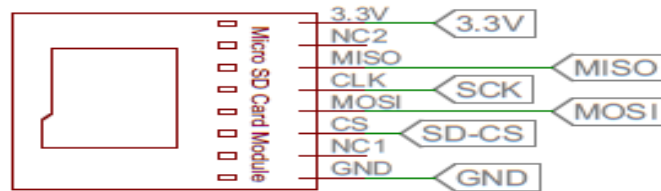


Figure 22: Card Module Pin Configuration

Figure 22 shows the pin configuration of the SD card module shield of how it is attached to the mega ARDUINO board on the PCB board. It takes a 3.3V input and other MOSI and MISO pins for serial communication. the ground is also provided.

3.7.4 Software

The software requirements of the project were completed through the

- Visual Studio
- Arduino 1.6.4
- Wireshark

Visual studio was used for the development of the applications which were to run onto the computers connected to the Router. Two applications have been developed. One is for the Server computer and one is developed for the Receiving/Host computer. The application on the Server would be responsible for extracting the URL's of the network application used and send to the ARDUINO board assembly for further processing.

The application on the receiving end would be responsible for collecting the data packets and information sent from the ARDUINO board through the Ethernet shields. The data received by the receiving application on the receiving server has already been sorted out on the basis of the protocols. The data is received and shown in a presentable manner using GUI. The data shown by the receiving application is

- URL
- IP
- Date and Time
- Subnet Mask
- Gateway IP

Arduino 1.6.4 programming software was used for the programming of the ARDUINO kits and shields. The programs were made using online help as the Arduino uses open source software.

3.7.5 Software Programs

Programs attached in Annex B of the document

The developed applications work efficiently on all of the computers attached. Some of the results are shown as follow:



Figure 23: Sending Module

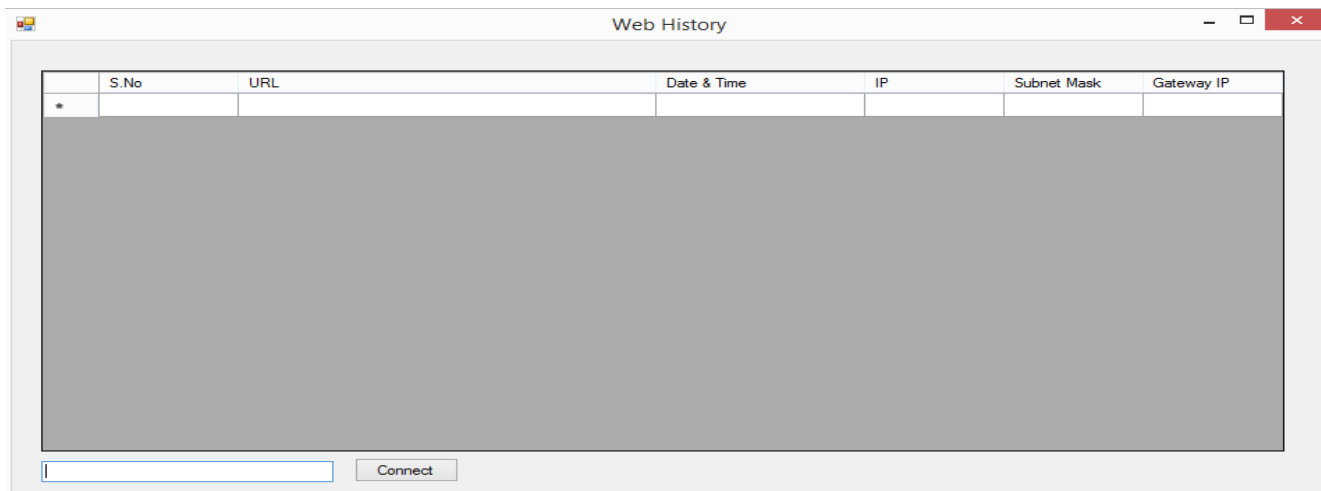


Figure 24: Receiving Module

These are the interfaces of the developed applications. The sending module or named “URL Extractor” would be working on the server computer. The receiving module would be working on the host computers attached to the hardware.

3.7.6 Results

The working can be seen by the following images

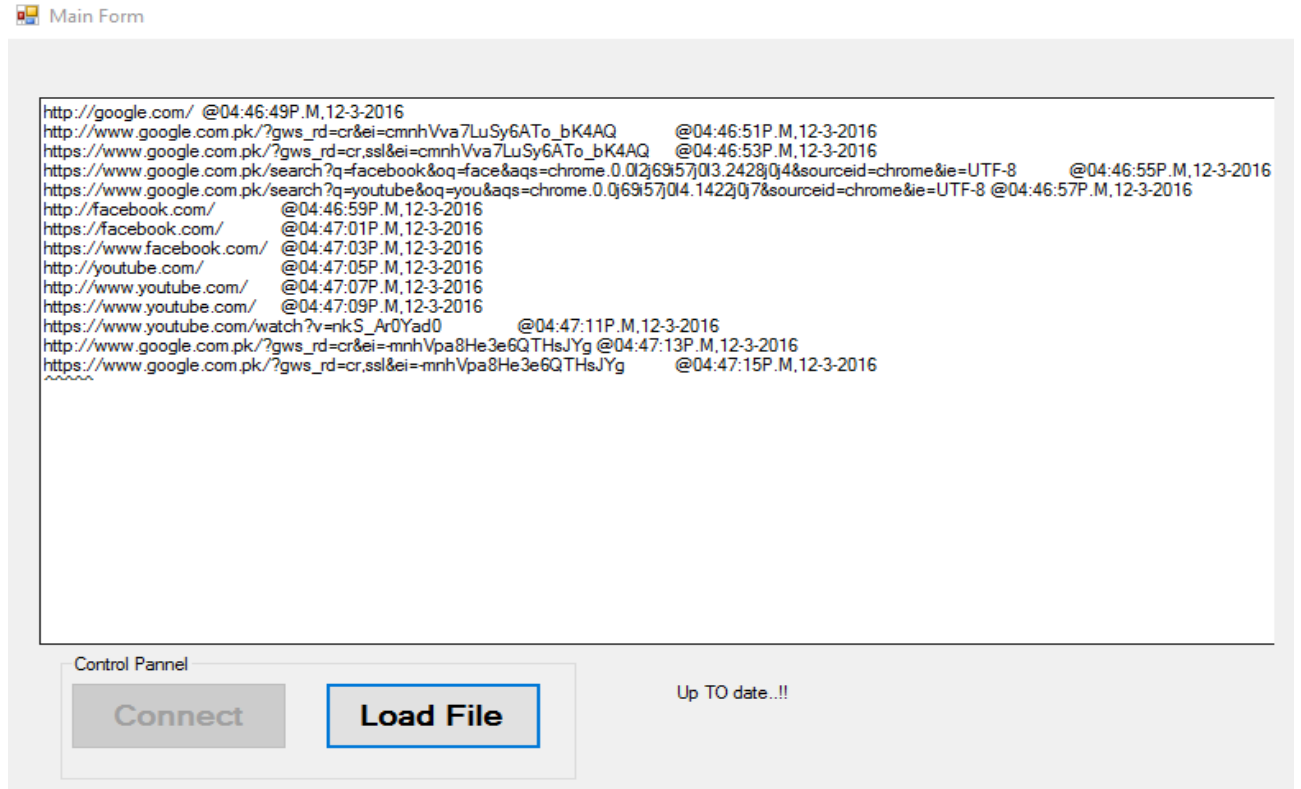


Figure 25: Data Statistics Requested by the Server

Figure 25 shows the data statistics shown at the server application. Once a call for the searched data is done. The ARDUINO extracts the data from the SD card Module and sends it to the server application where it is displayed. The file can be loaded using the LOAD FILE button.

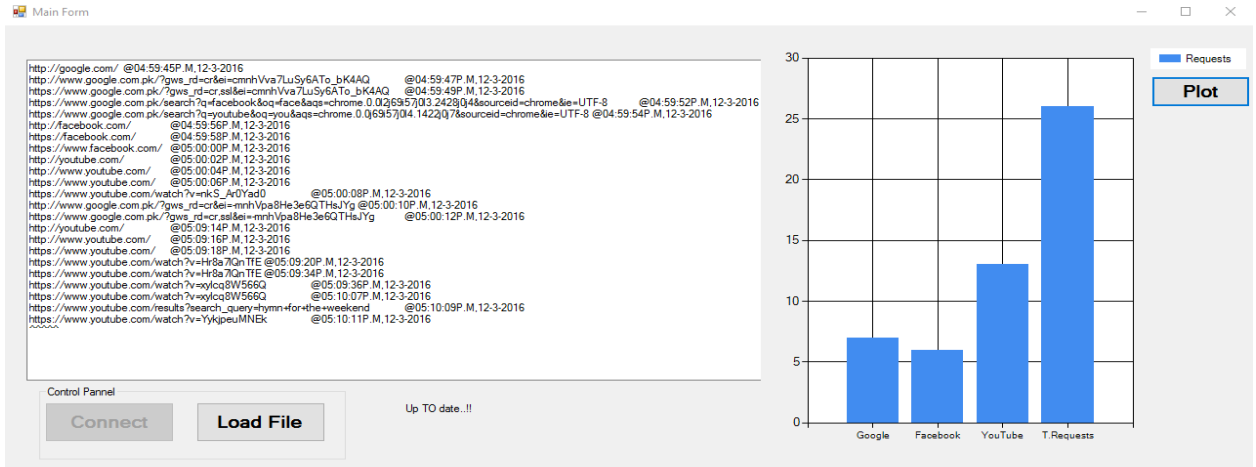


Figure 26: Statistics Requested by the Server and their Plotting

Figure 26 shows the data statistics shown at the server application. The server application has the capability to show the statistics of the data in the form of a plot/graph. The data is recalled and sent and shown by the server application. The plot is across (request, total searches). Each sites visiting statistics are shown and the overall queries are also shown.

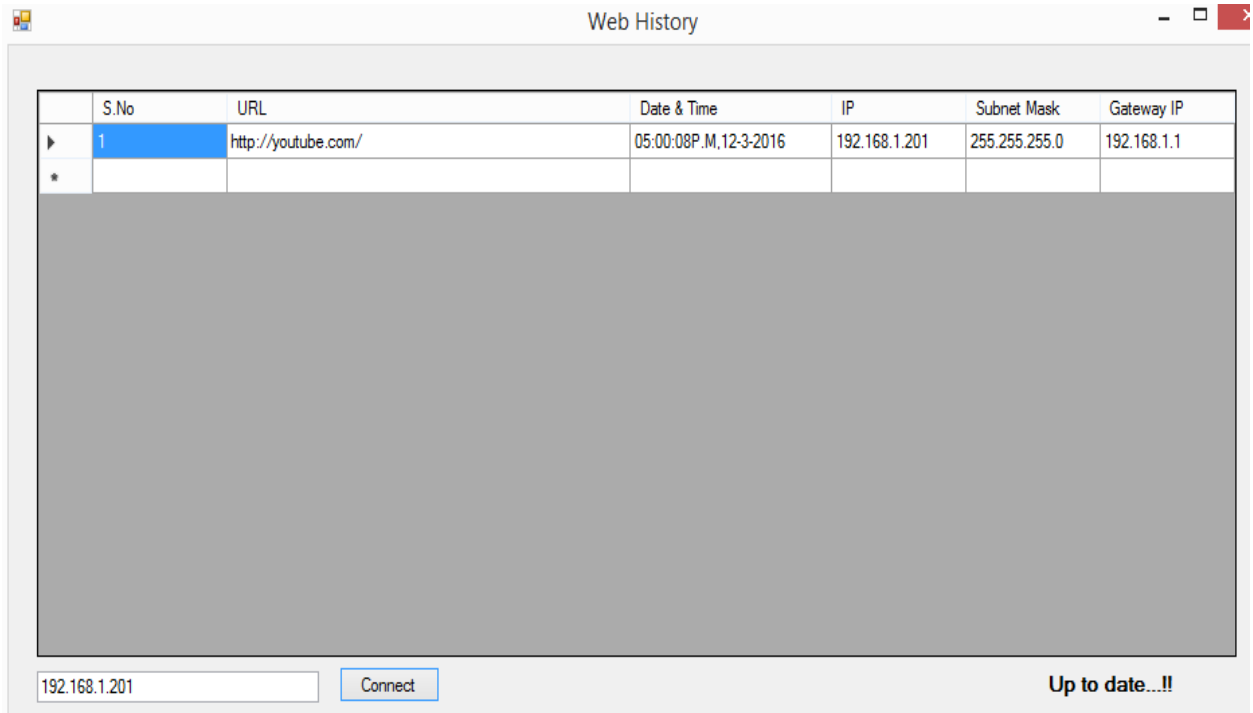






Figure 27: Data Statistics at the Receiving End

Figure 27 shows the data statistics shown at the host application. The server application has the capability to show the statistics of the data once it starts to receive the frames in real time. The data transmitted is shown by the host application alongwith its various other fields.

3.8 Visual Indications

3.8.1 Hardware LCD Display

Ser.	Indication	Implication	Display
1	Initializing	Basic boot setup of the ARDUINO board is running where the board boots up the programmed software and readies itself. "DONE" is shown when the initializing process is done	
2	Connecting	There has been a connection request from the server computer to connect itself with the prototype hardware.	
3	Connecting Connected..!!	"Connected" is shown when the handshake between the server and the hardware is done	
4	Welcome	The hardware is in the idle state. Meaning whereby no onboard processing is being done by the ARDUINO board. It can be afterwards <ul style="list-style-type: none"> • The hardware has no further data to send • The hardware 	

		<p>is awaiting for some request</p> <ul style="list-style-type: none"> The requested data has been sent 	
5	Google Request	A google request has been generated from the server computer which has arrived at the hardware and requires further processing.	
6	YouTube Request	A YouTube request has been generated from the server computer which has arrived at the hardware and requires further processing.	
7	Facebook Request	A Facebook request has been generated from the server computer which has arrived at the hardware and requires further processing.	
8	Sending File to Server	Data request has been generated by the server. The server wants to display the overall data that has been transmitted by the hardware. The data is compiled and sent back to the server.	


9	Plotting Data	Request to statistically plot the overall data which has passed through the hardware. The data is compiled and sent to the server for further processing.	
---	---------------	---	--

Table 3 : Hardware LCD Display Notations

3.8.2 Application Indications

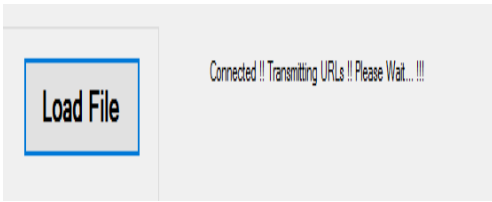
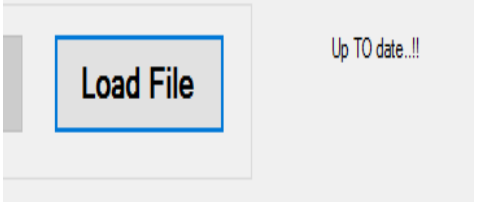
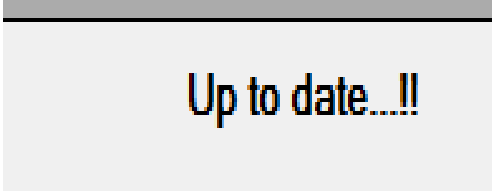
Ser.	Indication	Implication	End	Displayed
1	Connected Transmitting URLs	The server is connected with the ARDUINO board and the process of sending the generated requests(URL) is taking place	Server	
2	Upto Date	The data sending is done and no further requests are being generated.	Server	
3	Upto Date	The data receiving is done and no further data is to be shown until new data arrives	Host	

Table 4 : Software Application Notations

3.9 Deliverables

The layer 7 router has been implemented onto a collection of ARDUINO boards and various other necessary items. It is a totally separate unit and can standalone alone act as one. The voltage input of the device being 12V and a maximum of 1 x server and 3 x host computers can be attached.

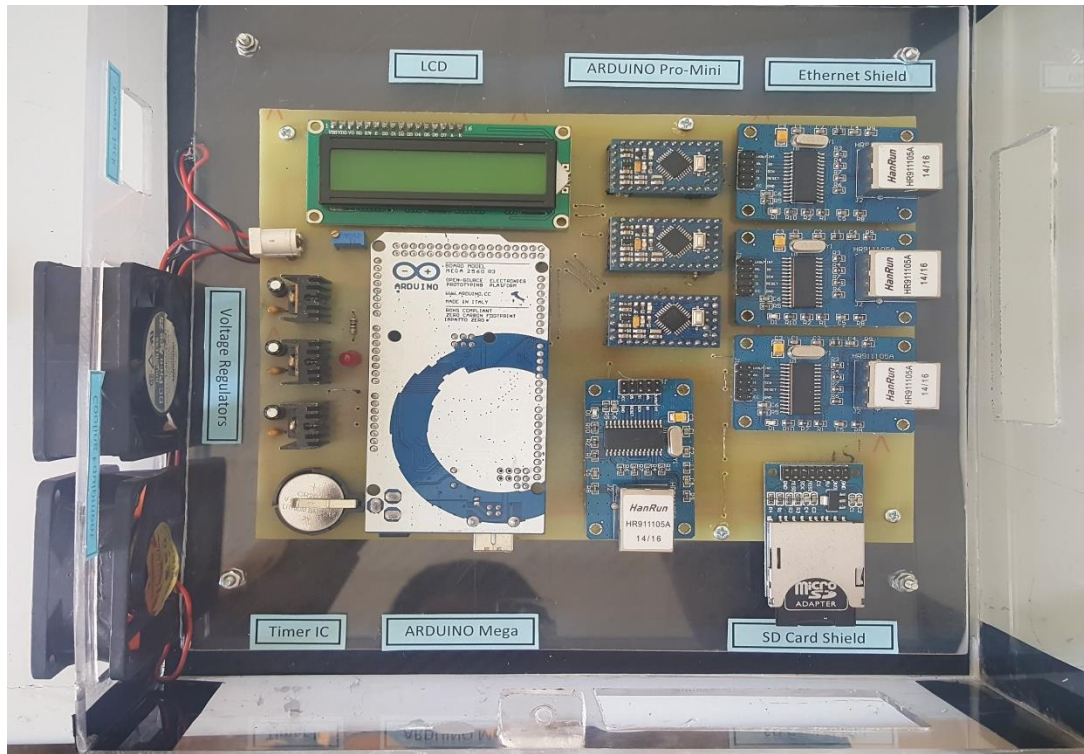


Figure 28: Final Hardware

3.9.1 Reflective Comments

- The completed project is able to work to its full capacity as a separate routing device as well as a network Monitoring device and can standalone act as it.
- The device is working at a minimum load level to the computer, requiring very less amount of data space
- The device is able to extract the URL packets from the server host to which it is connected and sort them out on the basis of the routing methodology adopted.

- The extracted URL's are sent to the ARDUINO kit in the form of an array where they are separated and forwarded to the destined ports passing through the ARDUINO pro-mini.
- At the same time real time data is being stored for later on data statistical work and calculation of the search terms etc.
- Data plotting and display capabilities are also available.
- The data statistics can only be accessed by the server computer for security purposes as no one would want their search history to be shared by anyone, thereby including security parameters as well.

Chapter 4: Applications

4.1 Applications

4.1.1 Server Load Balancing

Using Server load balancing, all possible routing links can be used at the same time to increase the available bandwidth. A router monitors the accessibility of all links and selects the path for sending packets.

4.1.2 Content Based Routing

Content-Based Router allows the router to examine message contents and then route the message onto a corresponding channel. The routing can be based on a number of criteria such as existence of fields, specific field values, content acceptability of the end router etc.

4.1.3 URL Blocking

URL blocking refers to allowing or denying the access to a certain websites or certain URL addresses for the web users either temporarily or permanently.

4.1.4 Multi-layer Switching

Multi-layer switching combines layer 2, 3 and 4 switching technologies and provides high-speed scalability with low inactivity. Multi-layer switching can move traffic effectively and also provide layer 3 routing that can eliminate the blockage from network routers.

4.1.5 High Availability Networks

Improvement towards bandwidth congestion and distribution of traffic over multiple paths can improve the availability of networks.

4.1.6 Network Monitoring Device

Network monitoring device would allow the constant and real time monitoring of the data being requested and processed by the router improving security and more control features

4.2 Conclusion

Layer 7(application) forms the base of our project. In order to achieve good quality along with reliable and protected use of World Wide Web we implemented the routing technique in the layer 7 where a user can catch the packets and can analyze and route the packets based according to their protocols. We have been able to overcome the congestion problem along with onboard processing and statistical calculation of packets with efficient use of memory. Present routing techniques depend upon IP checking and OSPF routing protocol which are unintelligent and time consuming. Layer 7 router is able to forward the data packets based on their routing protocols for intelligent bandwidth allocation and also data processing to meet the data statistical needs of the day.

Chapter 5: Resources Required

<u>Ser. No</u>	<u>Item Required</u>	<u>Quantity</u>	<u>Estimate Cost (Rs)</u>
1	ARDUINO Mega	1	5,000
2	ARDUINO mini	3	2,500
3	Ethernet Shield Module	4	5,000
4	SD Card Shield Module	1	1,500
5	Ethernet cables	4	500
6	Components (IC, Resistors etc)	-	500
7	PCB Board	1	200
8	Hardware Casing	1	2000
		Total	17,200

Chapter 6: References

1. <https://devcentral.f5.com/articles/layer-7-switching-load-balancing-layer-7-load-balancing>
2. <http://nginx.com/resources/glossary/layer-7-load-balancing/>
3. http://en.wikipedia.org/wiki/Routing_osilayer_history
4. <http://www.xilinx.com/training/Arduino/Arduino-field-programmable-gate-array.htm>
5. <http://www.xilinx.com/Arduino/>
6. <https://www.altera.com/products/Arduino/overview.html>

Chapter 7: Bibliography

1. SaroVelrajan, “Application Aware Routing in Software Defined Networks”, Aricent, 2013.
2. G. Antichi, S. Giordano, D. J. Miller, “Enabling Open-Source High Speed Network Monitoring on NetARDUINO”, Network Operations and Management Symposium, 2012.
3. J. Naous, G. Gibb, S, Bolouki N. McKeown, “NetARDUINO: Reusable Router Architecture for Experimental Research” Stanford University, 2009
4. M Ciesla, V Sivaraman, ASeneviratne, “URL Extraction on the NetARDUINO Reference Router”, The University of New South Wales.

Appendix A

Applications code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SQLite;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Threading;
//using finisar.sqlite;
namespace URL_Extractor
{public partial class Form1 : Form{
    long Myreader6 = 0;
    int file_count = 0;
    int ex_count = 0;
    string google;
    string Facebook;
    string YouTube;
    string TotalRequests;
    //Socket TCPCClient;
    //NetworkStream TCPStream;
    Socket tcpserver;
    public Form1()
    { InitializeComponent();}
```

```

private void button1_Click(object sender, EventArgs e) {
    label1.Text = "Connecting ... Please Wait...!!!";
    // TCPClient =new System.Net.Sockets.Socket("192.168.1.1",8080);
    // TCPClient = New Sockets.TcpClient(Text_ip.Text, 8080)
    TcpListener tcplist = new TcpListener(IPAddress.Any, 8080);
    tcplist.Start();
    tcpserver = tcplist.AcceptSocket();
    tcpserver.Blocking = false;
    label1.Text = "Connected !! Transmitting URLs !! Please Wait... !!!";
    button1.Enabled = false;
    timer1.Enabled = true;
    timer1.Interval = 10000;
    timer2.Enabled = true;
    timer2.Interval = 100;}

private void timer1_Tick(object sender, EventArgs e){Start...!!!
/[snip] - As C# is purely object-oriented the following lines must be put into a class:
// We use these three SQLite objects:
SQLiteConnection sqlite_conn;
SQLiteCommand sqlite_cmd;
SQLiteDataReader sqlite_datareader;
string google = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) +
@"\Google\Chrome\User Data\Default\History";
//string google = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData)
+ @"D:\test\History";if
(System.IO.File.Exists(Environment.GetFolderPath(Environment.SpecialFolder.Desktop) +
@"\abcc")){
System.IO.File.Delete(Environment.GetFolderPath(Environment.SpecialFolder.Desktop) +
@"\abcc");}
System.IO.File.Copy(google, Environment.GetFolderPath(Environment.SpecialFolder.Desktop)
+ @"abcc");
string google1 = Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + @"\abcc";
//SQLiteConnection cn = new SQLiteConnection("Data Source=" + google +
".db;Version=3;New=False;Compress=True;");
//cn.Open();
sqlite_conn = new SQLiteConnection("Data Source=" + google1 +
";Version=3;New=False;Compress=True;");
// open the connection:
sqlite_conn.Open();
sqlite_cmd = sqlite_conn.CreateCommand();
sqlite_cmd.CommandText = "CREATE TABLE urls (url varchar, title varchar, visit_count
varchar, last_visit_date varchar)";
// Now lets execute the SQL ;D
//sqlite_cmd.ExecuteNonQuery();
// Lets insert something into our new table:
//sqlite_cmd.CommandText = "INSERT INTO urls (url, title, visit_count, last_visit_time)
VALUES ('www.facebook.com', 'Facebook','1','12/15/2015)";
//sqlite_cmd.CommandText = "INSERT INTO test (id, text) VALUES (2, 'Test Text 2)";

```



```

// But how do we read something out of our table ?
// First lets build a SQL-Query again:
sqlite_cmd.CommandText = "SELECT * FROM urls";
//SQLiteDataAdapter sd = new SQLiteDataAdapter("select url, title,visit_count,last_visit_time
from urls order by last_visit_time asc", sqlite_conn);
//DataSet ds = new DataSet();
//sd.Fill(ds);
sqlite_datareader=sqlite_cmd.ExecuteReader();
string[] u_url = new string[300];
string[] u_title = new string[300] ;
long[] u_time = new long[300] ;
long count = 0;
long Myreader5 = 0;
// The SQLiteDataReader allows us to run through the result lines:
while (sqlite_datareader.Read()) // Read() returns true if there is still a result line to read{
// Print out the content of the text field:
//System.Console.WriteLine( sqlite_datareader["text"] );
    string Myreader = sqlite_datareader.GetString(1); // url
    string Myreader2 = sqlite_datareader.GetString(2); // title
long Myreader3 = sqlite_datareader.GetInt64(3);
long Myreader4 = sqlite_datareader.GetInt64(4);
    Myreader5 = sqlite_datareader.GetInt64(5); // time
    // string Myreader4 = sqlite_datareader.GetString(6);
//if (Myreader5 > 0)
if (Myreader5 > Myreader6){
    u_url[count] = Myreader;
    u_title[count] = Myreader2;
    u_time[count] = Myreader5;
    if (u_url[count].Contains("google.com") || u_url[count].Contains("youtube.com") ||
u_url[count].Contains("facebook.com")){ // MessageBox.Show(u_url[count].ToString());
        byte[] send = Encoding.ASCII.GetBytes(u_url[count]);
        Thread.Sleep(3000);
        tcpserver.Send(send); // MessageBox.Show(u_title[count].ToString());
    }count += 1;
    //MessageBox.Show(Myreader);
    //MessageBox.Show(Myreader2);
    /// MessageBox.Show(Myreader3);
    //MessageBox.Show(Myreader3.ToString());
    //MessageBox.Show(Myreader4.ToString());
    //MessageBox.Show(Myreader5.ToString());} // MessageBox.Show(Myreader4);
}if( count == 0) { // MessageBox.Show("No History Update...!!");
    label1.Text = " Up TO date..!! "; }
//if (Myreader6 >= Myreader5){
//    MessageBox.Show("No update in History
//if (Myreader6 > Myreader5)
    long maxVvalue = u_time.Max();

```

```

        long maxIndex = u_time.ToList().IndexOf(maxValue);
//MessageBox.Show(u_time[maxIndex].ToString());
//MessageBox.Show(u_url[maxIndex].ToString()); //MessageBox.Show(count.ToString());
        if (maxValue == 0)
            { } else
            { Myreader6 = maxIndex; }
//MessageBox.Show(u_time[2].ToString());
//MessageBox.Show(u_time[3].ToString());
//MessageBox.Show(u_time[4].ToString());
// We are ready, now lets cleanup and close our connection:
sqlite_conn.Close();
        //string google =
Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) +
@"\Google\Chrome\User Data\Default\History";
        //SQLiteConnection cn = new SQLiteConnection("Data Source="+ google
+".db;Version=3;New=False;Compress=True;");
        //cn.Open();
        //MessageBox.Show("ok");
//SQLiteDataAdapter sd = new SQLiteDataAdapter("select url, tilte,visit_count,last_vist_date
from urls order by last_vist_time desc", sqlite_conn);}
        private void Form1_Load(object sender, EventArgs e){ }
        private void timer2_Tick(object sender, EventArgs e)
        { string textbox_data = "";
try{ Thread.Sleep(3000);
        byte[] rcv = new byte[tcpserver.ReceiveBufferSize];
tcpserver.Receive(rcv);
        textbox_data = System.Text.Encoding.ASCII.GetString(rcv);
        if (file_count == 1){
            timer2.Enabled = false;
            // string textbox_data = System.Text.Encoding.ASCII.GetString(rcv);
            if (textbox_data.Length > 0)
                {textbox_data = tcpserver.Receive(rcv);
                    while (true) {
                        ex_count = 1;
                        tcpserver.Receive(rcv);
                        textbox_data += System.Text.Encoding.ASCII.GetString(rcv);
                        if (textbox_data.Contains("^^^^^")){
                            //tcpserver.Receive(rcv);
                            textbox_data += System.Text.Encoding.ASCII.GetString(rcv);
                            timer2.Enabled = true;
                            break; }
                        else{
                            //tcpserver.Receive(rcv);
                            textbox_data += System.Text.Encoding.ASCII.GetString(rcv);} }
                textbox1.AppendText(textbox_data);
                file_count = 0; }
            if (file_count == 2)
                { string chart_data = System.Text.Encoding.ASCII.GetString(rcv);

```

```

        google = chart_data.Split('&').First();
        YouTube = chart_data.Split('&').Last();
        YouTube = YouTube.Split('$').First();
        Facebook = chart_data.Split('$').Last();
        Facebook = Facebook.Split('%').First();
        TotalRequests = chart_data.Split('%').Last();
        TotalRequests = TotalRequests.Split('!').First();
        ex_count = 2;          this.chart1.Series["Requests"].Points.Clear();
this.chart1.Series["Requests"].Points.AddXY("Google", google);
this.chart1.Series["Requests"].Points.AddXY("Facebook", Facebook);
this.chart1.Series["Requests"].Points.AddXY("YouTube", YouTube);
this.chart1.Series["Requests"].Points.AddXY("T.Requests", TotalRequests);
        file_count = 0;}}
    catch (System.Exception ex)
    {if (ex_count == 1){
        textBox1.AppendText(textbox_data);
        timer2.Enabled = true;
        ex_count = 0;}if (ex_count == 2){
//this.chart1.Series["Requests"].Points.AddXY("Google", google);
//this.chart1.Series["Requests"].Points.AddXY("Facebook", Facebook);
//this.chart1.Series["Requests"].Points.AddXY("YouTube", YouTube);
//this.chart1.Series["Requests"].Points.AddXY("T.Requests", TotalRequests);
        //file_count = 0;
        //ex_count = 0;}}
private void chart1_Click(object sender, EventArgs e){ }
private void button2_Click(object sender, EventArgs e)
{ byte[] send2 = Encoding.ASCII.GetBytes("PLOT");
  //Thread.Sleep(3000);
  tcpserver.Send(send2);
  file_count = 2; }
private void button3_Click(object sender, EventArgs e)
{  textBox1.Text = "";
  label1.Text = "Recieving File... Please wait...!!";
  byte[] send1 = Encoding.ASCII.GetBytes("GET");
  //Thread.Sleep(3000);
  tcpserver.Send(send1);
  file_count = 1;}}

```

ARDUINO code (INPUT Port):

```
#include <UIPEthernet.h>
#include <Wire.h>
#include <Time.h>
#include <DS1307RTC.h>
#include <LiquidCrystal.h>
#include <SPI.h>
#include <SD.h>
LiquidCrystal lcd(49, 47, 45, 43, 41, 39);
int google = 0;
int youtube = 0;
int facebook = 0;
int value ;
EthernetClient client;
signed long next;
uint8_t mac[6] = {0x11, 0x12, 0x13, 0x04, 0x05, 0x06};
IPAddress myip(192, 168, 1, 199);
void setup() {
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Initalizing...!");
  delay(2000);
  Serial.begin(9600);
  Serial.println("Initalizing");
  Serial1.begin(9600);
  Serial2.begin(9600);
  Serial3.begin(9600);
  Ethernet.begin(mac, myip);
  Serial.print("localIP: ");
  Serial.println(Ethernet.localIP());
  Serial.print("subnetMask: ");
  Serial.println(Ethernet.subnetMask());
  Serial.print("gatewayIP: ");
  Serial.println(Ethernet.gatewayIP());
  Serial.println("Initalizing SD card!");
  if (!SD.begin(7)) {
    Serial.println("initialization failed!");
    return; }
  Serial.println("Initalizing SD card Done...!");
  lcd.setCursor(0, 1);
  lcd.print("Done..!");
  delay(3000); lcd.clear(); lcd.setCursor(0, 0);
  lcd.print("Connecting.....");
  while (value != 1)
  { value = client.connect(IPAddress(192, 168, 1, 3) , 8080);}
  next = 0;
```

```

Serial.println("Connection Stablished");
lcd.setCursor(0, 1);
lcd.print("Connected!");
delay(3000);
lcd.clear();
lcd.setCursor(4, 0);
lcd.print("Welcome");
SD.remove("Record.txt");}
void loop() {
  if (value) {
    int size;
    while ((size = client.available()) > 0)
    { String RecPacket = "";
      uint8_t* msg = (uint8_t*)malloc(size);
      size = client.read(msg, size);
      for (int i = 0; i < size ; i++){
        RecPacket += char(msg[i]); }
      RecPacket += "\t" ;
      RecPacket += "@";
      RecPacket += GetDateTime(1, 1);
      if (RecPacket.indexOf("google.com") > -1) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Google Request!");
        SaveData(RecPacket);
        google += 1;
        Serial.println("its a google Request");
        Serial1.print(RecPacket + '^'); }
      else if (RecPacket.indexOf("youtube.com") > -1) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Youtube Request!");
        SaveData(RecPacket);
        youtube += 1;
        Serial.println("Its A Youtube Request");
        Serial2.print(RecPacket + '^'); }
      else if (RecPacket.indexOf("facebook.com") > -1) {lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Facebook Request!");
        SaveData(RecPacket);
        facebook += 1;
        Serial.println("Its A Facebook Request");
        Serial3.print(RecPacket + '^'); }
      else if (RecPacket.indexOf("GET") > -1){
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Sending File..");
        lcd.setCursor(0, 1);

```

```

    lcd.print("To Server...");
    Serial.println("GET Request!!!!");
    Serial.println(ReadFile() + "^^^^^");
    client.print(ReadFile() + "^^^^^");}
else if (RecPacket.indexOf("PLOT") > -1){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Ploting Data..");
    String plot = "";
    plot += String(google);
    plot += "&";
    plot += String(youtube);
    plot += "$";
    plot += String(facebook);
    plot += "%";
    plot += String(google + youtube + facebook);
    plot += "!";
    client.print(plot);
    Serial.println(plot);
} else{
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("No Request..");
    Serial.println("No Request for (Google,Facebook,Youtube)");}
delay(2000);
    lcd.clear();
    lcd.setCursor(4, 0);
    lcd.print("Welcome!");
    //Serial.println(RecPacket);
    free(msg);} } }
String GetDateTime(boolean tme , boolean dat){
    String Result = "";
    tmElements_t tm;
    if (RTC.read(tm)) {
        if (tme){
            if (tm.Hour > 12) {
                if ((tm.Hour % 12) >= 0 && (tm.Hour % 12) < 10)Result += '0';
                Result += tm.Hour % 12;} else
            {if (tm.Hour >= 0 && tm.Hour < 10)
                Result += '0';
                Result += tm.Hour;}
            Result += ':';
            if (tm.Minute >= 0 && tm.Minute < 10)
                Result += '0';
            Result += tm.Minute;
            Result += ':';
            if (tm.Second >= 0 && tm.Second < 10)
                Result += '0';

```

```

    Result += tm.Second;
    if (tm.Hour < 12) Result += "A.M";
    else Result += "P.M";
    if (tme && !dat)
        return Result; } if (dat)
{ if (dat && tme)
    Result += ','; Result += tm.Day;
    Result += '-'; Result += tm.Month;
    Result += '-'; Result += tmYearToCalendar(tm.Year);
    return Result; }}void SaveData(String data)
{ File myFile = SD.open("Record.txt", FILE_WRITE);
  if (myFile) {
    myFile.println(data);
    myFile.close();}
String ReadFile(void){
  File myFile = SD.open("Record.txt");
  String readData = "";
  if (myFile) {
    while (myFile.available()){
      readData += char(myFile.read());}
    myFile.close();} return readData; }

```

ARDUINO code (OUTPUT Port):

```
#include <EtherCard.h>
byte Ethernet::buffer[500];
BufferFiller bfill;
boolean connection = false ;
String RecPacket = "";
boolean DataComplete = false;
void setup() {
  Serial.begin(9600);
  static byte mymac[] = { 0x74, 0x69, 0x69, 0x2D, 0x30, 0x31 };
  static byte myip[] = { 192, 168, 1, 202 };
  static byte subnet[] = { 255, 255, 255, 0 };
  static byte gwip[] = { 192, 168, 1, 1 };
  ether.begin(sizeof Ethernet::buffer, mymac, 10);
  ether.staticSetup(myip, gwip, 0, subnet);}
void loop() { word len = ether.packetReceive();
  word pos = ether.packetLoop(len);
  if (pos && DataComplete)
  { String ip = ""; String sbnet = ""; String gw = "";
  for (byte i = 0; i < 4 ; i++)
  { ip += String(ether.myip[i], DEC);
  if (i < 3)
  ip += ".";}for (byte i = 0; i < 4 ; i++){ sbnet += String(ether.netmask[i], DEC);if (i < 3)
  sbnet += ".";}
}for (byte i = 0; i < 4 ; i++)
  {gw += String(ether.gwip[i], DEC);
  if (i < 3)
  gw += ".";}
  RecPacket += ip;
  RecPacket += "%"; RecPacket += sbnet; RecPacket += "{";
  RecPacket += gw; RecPacket += "}";
  char senddatatoclient[RecPacket.length()];
  RecPacket.toCharArray(senddatatoclient, RecPacket.length() + 1);
  ether.httpServerReply(senddata(senddatatoclient));
  DataComplete = false;RecPacket = "";}
else if (pos && !DataComplete)
  { ether.httpServerReply(senddata("Null"));
  }}void serialEvent() {
  while (Serial.available())
  { char inChar = (char) Serial.read();
  RecPacket += inChar;
  if (inChar == '^')
  { DataComplete = true; }} }
static word senddata(char abc[]) {
  bfill = ether.tcpOffset();
  bfill.emit_p(PSTR("$S"),abc);
  return bfill.position();}
```