Networks - Top Down approach - Application and Transport Layer

Saurabh Barjatiya

2012-03-20 Tue

Contents

1 Introduction

Computer networks are build using layered approach. There are different types of layers each with its unique function which together work to make networks work as we use them in every day life. There are different models which are used to describe these layers. Two popular models are:

- OSI (Open Systems Interconnection)
- TCP/IP model

1.1 OSI model

As per OSI model there are seven different layers in a computer network:

- Layer 1 - Physical layer
- Layer 2 - Data link layer

- Layer 3 - Network layer
- Layer 4 - Transport layer
- Layer 5 - Session layer
- Layer 6 - Presentation layer
- Layer 7 - Application layer

1.2 TCP/IP model

TCP/IP model divides same computer network into following four layers:

- Layer 4 - Application layer
- Layer 3 - Transport layer
- Layer 2 - Internet layer
- Layer 1 - Link layer

Layers 5, 6 and 7 of OSI model are combined together as single Layer 4 in TCP/IP model. Layers 1 and 2 of OSI model are combined together as single Layer 1 in TCP/IP model. Hence there are three less layers in TCP/IP model when it is compared with OSI model.

Rest of this lecture and next few lectures will be in terms of TCP/IP model which makes more sense from practical perspective. Study and understanding of OSI model will be left for more formal course in networks.

1.3 Studying various layers

To understand how computer networks work, one needs to understand details about each layer and how each of them work. There are two famous approaches in studying networks:

- Top-Down
- Bottom-Up

where in top-down approach one starts with application layer and goes all the way down to link layer and in bottom-up approach one starts with link layer and goes all the way up to application layer.

We would be using top-down approach for this small series of lecture on networks as application layer is more interesting for most users in comparison to link layer and starting with application layer generates enough interest for students to pursue learning and understanding of other lower layers.

2 Application Layer

At application layer we have various application protocols like

- HTTP (HyperText Transfer Protocol)
- SMTP (Simple Mail Transfer Protocol)
- DNS (Domain Name Service)
- DHCP (Dynamic Host Configuration Protocol)
- FTP (File Transfer Protocol)
- SSH (Secure Shell)
- IMAP (Internet Message Access Protocol)
- POP3 (Post Office Protocol 3)
- Secure versions of most protocols HTTPS, SMTPS, SFTP, IMAPS, POP3S, etc.

2.1 HTTP and SMTP protocol

HTTP protocol is used while browsing websites both Internet (facebook) and Intranet (courses). SMTP protocol is used by mail servers in sending emails to each other. Most of the time beginner email users use browser based email clients which work using HTTP protocol. With help of HTTP protocol users communicate with web server (and in turn email server) and read / send emails. But mail servers when they actually send emails to other mail servers (Gmail) they use only SMTP protocol and not HTTP.

2.2 DNS service

DNS servers are used to convert FQDN (Fully Qualified Domain Names) or just domain names (www.google.co.in) to IP addresses (173.194.36.63). Human beings prefer easily remembered English names compared to IP addresses which are very hard to remember. DNS allows one to use majority services available on LAN / Internet without remembering (in fact many times without evening seeing) the IP addresses.

2.3 DHCP service

DHCP servers help in assigning IPs to clients. DHCP servers help in achieving many different goals:

- Users do not need to know which IP range/sub-net mask, gateway combination will work on a given port or in a given area
- Users do not need to spend effort in manually determining which IPs are free and which are in use
- A large number of non-parallel users can be accommodated in less number of IPs with help of DHCP servers. With static IP addressing we would need to reserve one IP per user. This feature of DHCP server has helped a lot in extending life of IPv4 protocol.

2.4 FTP service

FTP servers are meant for file transfers. FTP is not very firewall or NAT friendly and hence is not used very often in campus scenarios where large number of users connect to Internet via few proxy servers. FTP is older than HTTP and SMTP protocol and was used a lot in transferring files in earlier days of Internet. But with wide-spread use HTTP use of ftp now-a-days is very limited.

2.5 SSH service

Running commands on remote machine or managing devices remotely has been one of the main objectives of LAN / Internet. telnet was the first protocol which helped in achieving this objective and surprisingly is still the most common protocol while connecting to hardware devices (at least for first time). telnet was replaced by rsh (Remote shell) which had almost same features and capabilities as simple ssh has today with just one limitation of security. In rsh passwords were being transmitted in plain-text which was not secure. Hence ssh was invented so that not just passwords, but entire communication between two machines get encrypted.

Now a days SSH supports many other facilities apart from providing secure remote shell. Some of which are:

- Local port forwarding (Tunneling)
- Remote port forwarding (Tunneling)

- Dynamic port forwarding (Socks proxy)
- X11 or Graphics forwarding (GUI remote login)
- SFTP sub-system (File transfer)

2.6 IMAP protocol

IMAP protocol is used by email clients to connect to mail servers for synchronizing email information. Unlike its famous counterpart POP3 IMAP does not downloads all emails from server, but instead just gets header information (Subject, Date/Time, Size, Sender, Recipient, etc.) from all emails from server and displays it to user. Only when a user chooses to read an email, then IMAP clients send request for content of that particular email to mail servers. Due to this users have much less total time and bandwidth spent while checking emails using IMAP. Even when email is being downloaded from server, many IMAP clients do not download attachments of email from server unless user explicitly asks them to be displayed / saved locally again saving considerable time / bandwidth.

IMAP protocol also allows synchronization of folders between email clients and email server. We can use many different IMAP clients to connect to same server for same email account and all clients display similar email information. Thus IMAP allows users not to be dependent on any particular email client or machine. Since copy of all email data is always available on server, failure of email client does not leads to loss of email information. This also allows IMAP clients to store very less information on local hard-disk as they can fetch required information from server as and when requested.

2.7 POP3 protocol

POP3 protocol can also be used by email clients to access emails. Usually POP3 clients download all emails from server and store them on local disk. Hence total time and bandwidth required for checking email with POP3 is more when compared with IMAP. But users can do other things while POP3 client is downloading emails and check email once download is finished. In such cases however total time is more, response time is considerably less as all emails are available locally and no further Internet communication is required for reading emails or viewing attachments.

Since POP3 usually stores emails on local machine, users get tied to particular machine when using POP3. Also if local machine fails users loose access to their emails (at least temporarily) till emails are restored on some

other working client. POP3 also downloads all emails into INBOX and all organization of email has to be done on local folders.

Q. What protocol squirrel-mail must be using to get emails from IMAP server?

3 Capturing packets

We can capture packets traversing through network interface of our machine (provided we have root or administrative access) to see exactly what is being transmitted by our network card and what is being received by it. By studying information traveling in wire we can learn various protocols by observing them in action. One of the tools which provides very intuitive and easy to use GUI interface for capturing and analyzing packets is `wireshark`. We will use `wireshark` in next few lectures on networks to observe various protocols, learn them and then try them with our own hand (or code).

4 Protocol usage examples

4.1 Telnet and nc demonstration

- Use telnet or nc client to connect to nc server

– Starting nc server

```
nc -l <port>
```

For example, `nc -l 9000`

To terminate nc connection use `Ctrl+C` short-cut.

– Connecting to nc server via telnet client

```
telnet <IP address> <port>
```

For example, `telnet localhost 9000`

To terminate telnet connection use `Ctrl+]` short-cut to get `telnet>` prompt. Then use `quit` command to end connection.

– Connecting to nc server via nc client

```
nc <IP address> <port>
```

For example, `nc 127.0.0.1 9000`

To terminate nc connection use `Ctrl+C` short-cut.

4.1.1 Difference between TCP and UDP

`telnet` and `nc` use TCP (Transmission Control Protocol) sockets. TCP is connection oriented transport protocol. Following features are provided by TCP:

- In order delivery of packets / information
- Re-transmission of lost packets / garbled packets
- Rate control based on receivers capability to receive information
- Rate limiting based on network congestion
- Acknowledgment of receipt of data at other end

Because of this one needs to close only client or server side connection in nc / telnet setup. The other side is automatically notified that connection is closed and the programs on both side terminate successfully.

The other most common transport layer protocol after TCP is UDP (User Datagram Protocol). UDP is connection-less protocol. Hence following things are **not** provided by UDP:

- In order delivery of packets / information
- Re-transmission of lost packets / garbled packets
- Rate control based on receivers capability to receive information
- Rate limiting based on network congestion
- Acknowledgment of receipt of data at other end

4.1.2 Advantages of UDP over TCP

This may make it look like UDP is not very useful. But UDP has its uses. UDP is used in following scenarios:

- Delay sensitive or throughput sensitive applications UDP is used in almost all voice / video transmission protocols because it does not slows down because of congestion. Hence UDP gets unfair amount of bandwidth in comparison to TCP making it ideal for voice / video traffic where unfair amount of bandwidth is desired for communication. UDP is also preferred transport protocol for LAN / Internet games

because of same reason. (Counter strike very often uses UDP port 27015 for its communication)

It should be noted that both in games and audio/video application loss of packets does not degrade the quality / user experience by noticeable amount. The same kind of loss cannot be tolerated in many other applications like file transfer, SSH.

- Multicast applications work only with UDP. TCP does not (and in some sense cannot) support multicast

  Multicast allows sender to send data only once for multiple recipients thus saving considerable bandwidth and processing at sender and even on in-network devices. Multicasts are used very often for Internet TV / Radio applications where same video / audio may have to be delivered to millions of users.

- Small amount of data (<=1000 bytes) needs to be transferred

  Most of the time DNS, DHCP, NTP (Network Time Protocol) transfer very small amount of data between clients and server. In such cases uses full connection establishment phase of TCP is very expensive and re-transmission of lost packets after time-out (or for efficiency / redundancy sometimes even before that) makes more sense.

You have already used or will use UDP in IIIT for following purposes:

- DNS
- DHCP
- Voice chat
- LAN games
- VPN (Virtual Private Network) - IIIT openvpn uses UDP port 1194
- DC++ - By default DC++ search operates over UDP port 411

4.1.3 Significance of port numbers (1 to 65, 535)

Since many applications work in parallel and many connections are made from same system to many remote systems there is requirement of distinguishing one connection from another. Also there is requirement of supporting multiple connections from same client to same server without causing any confusion. These requirements are met with the help of port numbers.

A pair of:

- Client IP address
- Server IP address
- Client port
- Server port

is called a socket / connection and can be used to distinguish between any two connections. For example if same client (Firefox browser) on machine (say 10.0.0.1) connects to proxy server (say 10.4.3.204) at proxy port (say 8080). Then the client will use different client ports, usually in range 27,000 to 65,535 (sometimes in range 1,025 to 65,535) for each of the parallel connection and that client port will be useful in distinguishing between different connections.

Once an application uses a port number for server / client use then other applications cannot use the same port for server / client use unless first application closes the connection and frees the port. Hence we cannot have two different browsers originating connections from same client port (say 9987) to two different servers. Similarly we cannot have two different servers on same machine listening on same port (say 80) for client connections.

4.1.4 Ports of common services (`/etc/services` file)

We can find out ports used by common services using `/etc/services` file. For example to find out port used by SSH we can use:

```
grep -i ssh /etc/services
```

4.2 HTTP protocol

4.2.1 Observe HTTP protocol through wireshark and Firefox

Start wireshark and capture packets (on most machines on interface `eth0`). Then start Firefox and try to open some website (http://intranet.iiit.ac.in) and observe packets sent and received in wireshark. Stop packet capture on wireshark after intranet site is opened successfully so that wireshark does not continues to capture packets while we are trying to analyze packets transmitted during browsing.

4.2.2 Observe HTTP protocol through wireshark and wget

Again restart wireshark packet capturing on interface (mostly `eth0`) and now try to download some web page using `wget`. Stop packet capture after `wget` has successfully downloaded a page and observe the protocol followed in wireshark. Notice that both Firefox and `wget` use same protocol while talking to different web servers. This example should help in understanding what **protocol** actually means.

4.2.3 Try HTTP protocol using telnet

Can only `wget` and Firefox speak HTTP protocol? Lets find out by trying to send HTTP requests by hand using `telnet` to http://intranet.iiit.ac.in to see whether intranet server replies to our requests same as it replies to requests from Firefox and wget.

```
[saurabh@labpc networks]$ telnet intranet.iiit.ac.in 80
Trying 10.4.2.208...
Connected to intranet.iiit.ac.in (10.4.2.208).
Escape character is '^]'.
GET http://intranet.iiit.ac.in/ HTTP/1.0
Host: intranet.iiit.ac.in

HTTP/1.1 200 OK
Date: Wed, 21 Mar 2012 06:24:29 GMT
Server: Apache/2.2.3 (CentOS)
X-Powered-By: PHP/5.1.6
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from url=(0022)/ -->

<HTML><HEAD><TITLE>IIIT Intranet</TITLE>
....
<CENTER></CENTER></TR></TBODY></TABLE></CENTER></BODY></HTML>


Connection closed by foreign host.
```

The actual lines typed by us are only three:

```
[saurabh@labpc networks]$ telnet intranet.iiit.ac.in 80
GET http://intranet.iiit.ac.in/ HTTP/1.0
Host: intranet.iiit.ac.in
```

rest is output displayed by telnet application when it starts or response from web-server because of our request.

4.3 SMTP protocol

4.3.1 Observe SMTP protocol through wireshark and evolution / thunderbird

Configure evolution or thunderbird as IMAP (avoid POP3) and SMTP clients for students.iiit.ac.in server. Prefer use of plain-text (insecure) protocols over secure counterparts for this exercise. Try to check / send email using evolution / thunderbird and see the language (protocol) used by email clients when they talk with email servers.

4.3.2 Try SMTP protocol using telnet

Having learned the protocol (SMTP) used by email clients while sending emails lets try to send email by hand using telnet to check whether any application can send email or is there something special about evolution / thunderbird etc. that only they can send email.

```
[saurabh@labpc networks]$ telnet students.iiit.ac.in 25
Trying 10.4.2.200...
Connected to students.iiit.ac.in (10.4.2.200).
Escape character is '^]'.
HELO Saurabh
220 students.iiit.ac.in ESMTP Postfix (Debian/GNU)
250 students.iiit.ac.in
MAIL FROM: saurabh.barjatiya@students.iiit.ac.in
250 2.1.0 Ok
RCPT TO: saurabh.barjatiya@students.iiit.ac.in
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: Trying to send email by hand
From: magic_user@students.iiit.ac.in
To: why_you_got_this@students.iiit.ac.in
```

```
Hi Saurabh,

If you look at sender and recipient addresses in your email client, you may wonder why

Hope this helps.

Regards,
Saurabh

.
250 2.0.0 Ok: queued as EF6703182A6
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

Again actual lines typed by user are:

```
[saurabh@labpc networks]$ telnet students.iiit.ac.in 25
HELO Saurabh
MAIL FROM: saurabh.barjatiya@students.iiit.ac.in
RCPT TO: saurabh.barjatiya@students.iiit.ac.in
DATA
Subject: Trying to send email by hand
From: magic_user@students.iiit.ac.in
To: why_you_got_this@students.iiit.ac.in

Hi Saurabh,

If you look at sender and recipient addresses in your email client, you may wonder why

Hope this helps.

Regards,
Saurabh

.
QUIT
```

and rest is information displayed by email server.

WARNING - A careful observer would realize that we did not supply any password for saurabh.barjatiya@students.iiit.ac.in while sending this email. Hence anyone can send email from any email ID to any person. To make things worse one can intentionally put different address on envelope and different in email headers. It is expected that this information would not be misused by students to send spurious (SPAM) emails to each other. Tracking emails sent in this manner back to sender is not difficult, so do not attempt anything foolish assuming such techniques to be anonymous.

4.4 DNS protocol

4.4.1 Observe use of UDP instead of TCP at transport layer

Again start wireshark and try to do DNS queries with below mentioned commands:

```
nslookup intranet.iiit.ac.in
nslookup courses.iiit.ac.in
nslookup www.google.co.in
```

Look at query packets in wireshark to understand various values sent by DNS client (resolver) to DNS server. Similarly look at responses sent by DNS server to DNS client to understand various values and parameters sent by DNS server to DNS client. Try to understand significance of as many values and parameters as you can.

Notice use of UDP and various fields in UDP header in contrast to TCP.

4.4.2 Different types of queries (MX, A, PTR, AXFR)

Try different queries as suggested below and observe responses as sent by DNS server:

```
dig -t MX iiit.ac.in
dig -t A iiit.ac.in
dig -t PTR -x 74.125.236.159
dig -t AXFR iiit.ac.in
```