

Mod 2.1

providing service)

internet to provide service to end user

TCP/IP makes it more flexible to do so

If protocol is added at layer it should be designed in a way to use protocols from lower levels.

If protocol is removed it should be taken care that higher levels don't use it

Application Layer - do not provide service to any other layer

Standard Application Layer Protocols

Standardised and documented by internet authority.

Each standard protocol is a pair of computer programs that interact with user and transport layer to give specific service

Non Standard

if two programs that provide service to user by interacting with transport layer then it is non standard protocol. Any programmer can create their own protocol and doesn't require approval if privately used.

Application Layer Paradigms

i) client server

ii) peer to peer paradigm

Client server -

service provide - server process

receiver + client process

server waits for client to make connection

server should be running always client only when required

Problem -

load on server so if there is heavy traffic it should be powerful

Service provider willing to bear cost of powerful computer

ex. HTTP

Peer to Peer -

A computer connected to internet can provide service at one time and received at another or at same time

Problem -

Security

applicability - many users not ready to be involved.

ex. Bit torrent

Mixed -

Mixture of two

For example light load server which connects one user to address of other that can offer the service.

* Client Server Paradigm

communication at application layer between two running application is called processes

Client is running program that initializes the communication by sending a request. server waits for request from client

Server program should be started before client program

Application Programming Interface

We need a process to be able to communicate with another process, we need set of instructions to tell lowest four layers of TCP/IP suite to open connection send and receive data and close connection. This is called as an API

Some APIs

- i) socket interface
- ii) Transport Layer interface
- iii) STREAM

Socket interface is a set of instruction that provide communication between application layer and OS

we can use socket as key

socket address (IP+PORT)

For two way communication we need address of sender and receiver

Find address

server

local socket address - give by OS - IP address

remote socket address - This comes from client in packet received

Client

Local socket address -

IP + Port number where program is listening

Using service of Transport Layer

- i) TCP
- ii) UDP
- iii) SCTP

UDP

unreliable

Provides connectionless datagram service.

UDP is message oriented.

UDP is best effort

It checks if data is corrupted but doesn't ask to resend the lost or corrupted datagram

Used where speed is priority

TCP

Connection oriented, reliable byte stream

First logical connection is established called handshaking
bytes are numbered so order is maintained

If bytes are lost receiver can request for those

UDP

It is connection oriented reliable service

Combination of TCP and UDP. It is message oriented

It provide multi stream

* PEER 2 PEER PARADIGM

P2P Networks

Internet user ready to share resources become peer
and join the network

Those peers who have the file make it available
for others to download

Centralized

hybrid P2P network. Information of peer is store on
central directory but the actual information sharing happens
bet P2P. Central server are vulnerable to attack

Decentralised

The Peers arranged into overlay network

P2P classified by the structure of overlay networks

- i) Unstructured network - nodes are linked randomly. To find a file a query is sent to all the nodes causing a lot of traffic
- ii) ~~It~~ Structured - It has predefined rules to link nodes so that query is efficiently resolved
Most common technique - Distributed hash table

Bit Torrent

A peer which has completed file - seed
A peer which needs file - leech
Set of peers which take part is swarm

BitTorrent with tracker

there is entity called tracker it tracks the operation of swarm

New peer gets metadata which contains information of pieces and address of tracker that handles that torrent. It joins torrent and gets address of neighbours. Now it can upload or download the file

To avoid overloading each peer can have 4 concurrent connection

A peer flags neighbour as choked unchoked interested or not interested
unchoked - list peers the current is connected to
choked - not connected

Every 10 secs a choked peer in interested group is tried for better data rate. If its better it becomes unchoked

every 30 sec a random peer is promoted from choked to unchoked. This is called optimistic unchoking

It also follows rarest first.

Trackerless Bittorrent

If tracker fails new ~~user~~^{peer} cannot join and updating is interrupted

In this job of tracking is distributed among nodes
DHT can be used

If we use hash function of metadata as key and hash function of list of ~~peers~~ peers in swarm as value

new peer joins and sends key of meta data to peer if key matches \rightarrow file present

The responsible node sends value, i.e. actually list of peers in corresponding torrent. Then peer can join