```yaml
id: 1715166999-LOLW
Author: Vortex
aliases:
  - CN-U2
tags: []
```

Domain Naming Service

- Mneumonic names are easier for humans to read. But routers and other network devices prefer fixed length numeric addresses.
- The service that maps these 2 togeather is called DNS.
- Serviced on port 53
- DNS consists of :
  - Bunch of DNS servers containing databases of mappings
  - Client side DNS service that sends the request to the DNS servers
- Other application protocols like HTTP and SMTP use DNS as a way to translate hostnames to IP addresses.
- The Client then receives the IP address corresponding to that hostname and can then proceed to establish a connection.
- Services :
  - Host Aliasing : canonical names can be long and complex. DNS can alias this into a simpler hostname
  - Mail server Alias : DNS can be used by mail applications to obtain the IP address of a hostname and also its canonical name. MX records permit mail and webserver to have same aliased hostnames.
  - Load Balancing : To reduce latency and traffic many webservers may have multiple server farms with different IP addresses. DNS can map all of these to a single hostname. When DNS is requested for this hostname, it returns a list of IP addresses.
    - DNS rotates this list after every response so that the subsequent requests can be routed to a different server and load is balanced equally.
- Reasons for having Distributed DNS:
  - Single point of failure
  - Traffic volume
  - Distance from servers and clients
  - Maintenence

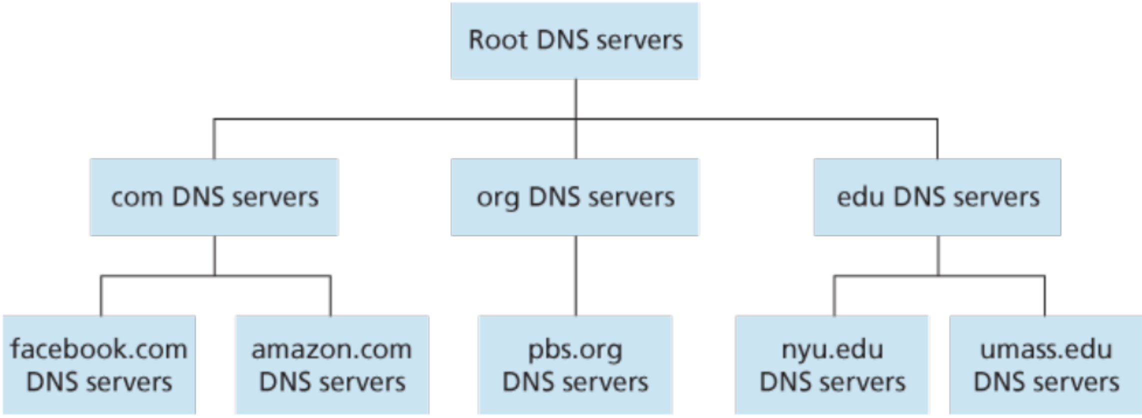
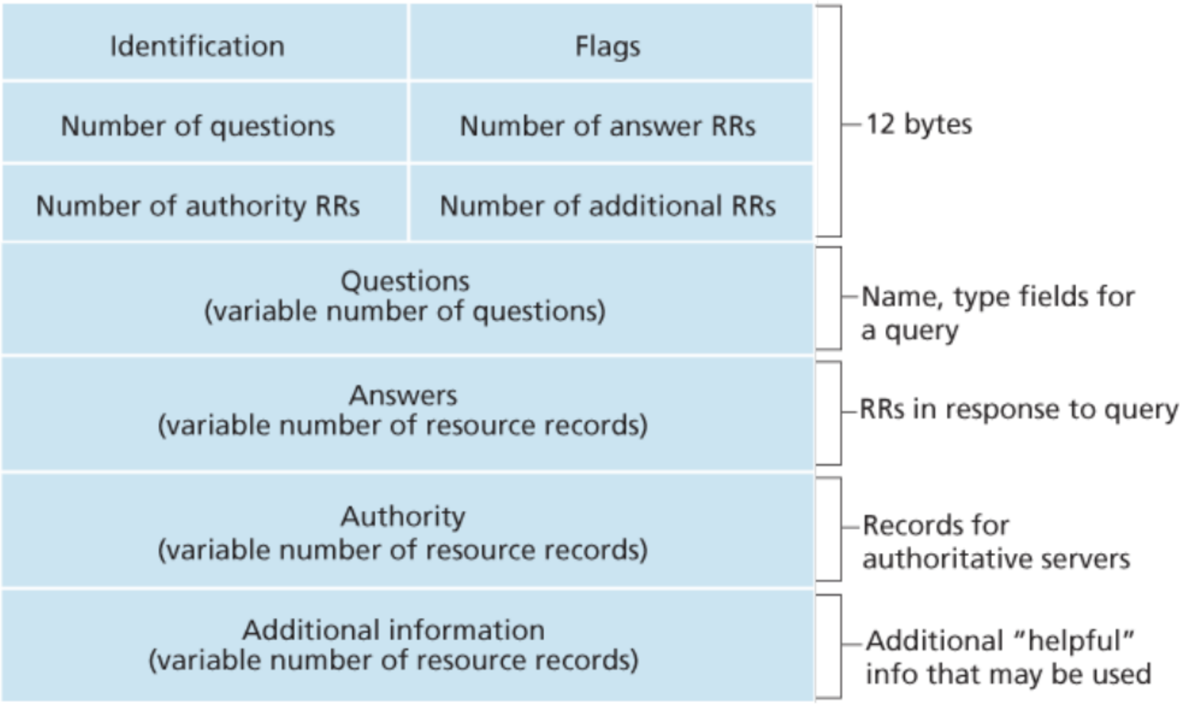


Figure 2.17 Portion of the hierarchy of DNS servers

- Root level : returns the IP address of the TLD.
- Top level : all top level domains like `.com`, `.org` and country level domains like `.in`, `.fr`
- Authoritative : Every organisation with a publicly accessible host must have this to map the host to IP address.
- Local level : not part of the heirarchy. Maintained by the ISP very close to the client.
  - Acts as a proxy and sends the request to the DNS heirarchy.
- Types:
  - Iterative : repeatedly exchanges req-res cycles with multiple levels of DNS servers.
  - Recursive : hands off responsibilities to the next DNS server
- DNS caching : DNS requests are cached in the server for some amount of time (TTL)
  - If request to this hostname is made in this time, it won't have to reference the heirarchy and can simply return the cached IP.
- Response Records(RR)
  - DNS servers store data as RRs.
  - response message from DNS server is a 4 tuple value
  - `(Name,Value,Type,TTL)`
  - Types:
    - `A`, then `Name` is the hostname and the `value` is the IP.
    - `CNAME`, then `Name` is the alias name and `value` is the canonical name.
    - `MX`, then `Name` is the mail server name and value is the canonical name
    - `NS`, then `Name` is domain and `value` is the hostname of an authoritative DNS server that knows how to get IP for hosts in that domain.
- DNS Message format



- Identification : to match requests and responses.
- Flags : multiple flags. One flag indicates whether this is a request or response message.
- Questions : Information about the query.
    - Name
    - Type
- Answer : Request Records
    - there may be multiple since one hostname may have multiple IPs
- Authority : records of other authoritive servers
- DNS records are inserted into the server by the DNS registrar.
- They verify the uniqueness of the domain.

P2P

- Minimal dependence on servers.
- $D_{C-S} = max(\frac{NF}{u_s}, \frac{F}{d_{min}})$
- $D_{P2P} = max(\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \Sigma_{i=0}^{N} u_i})$

BitTorrent

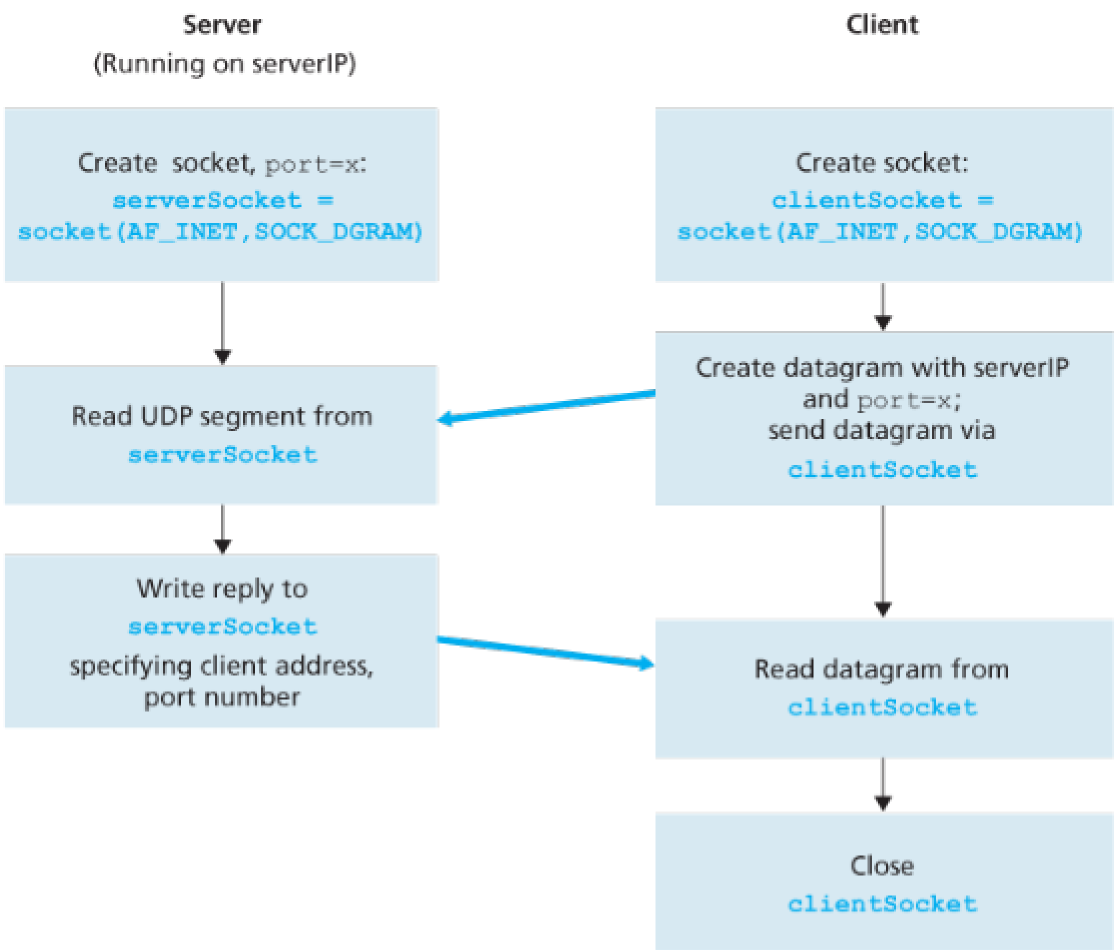
- Has a tracker which is a server keeping track of all the peers in the torrent.
- All the participating peers make up the `torrent`.
- When a new host joins, it registers itself with the tracker. Then periodically it pings the tracker to let it know that it is still in the network.
- Torrent networks exchange data in chunks(256Kb).
- Initially the peer doesn't have any chuncks.
- The tracker picks a list of 50 peers and gives it to the new machine.
- This new peer then tries to establish simultaneous TCP connections with these peers.

- These are called `neighbouring peers`.
- While a peer is receiving chunks it can also send chunks.
- When a peer has received all the chunks, it can either selfishly leave the torrent or alturistically continue sharing.
- The peers use a method of `rarest first` to get chunks.
    - The chunks that are rare in the neighbouring peers are demanded first so that these chuncks are distributed evenly throughout the network.
- A peer is always sharing with 4 peers who are transferring to it at the highest rate.
- Every 30 seconds, this peer picks another peer at random and starts sending data to it.
- This is now said to be optimistically unchoked
- Churn : when a peer leaves or joins the network.

Sockets

UDP

- No reliability
- Each packet contains :
  - IP address of destination
  - Destination Port
  - IP and port of source(added by the OS)



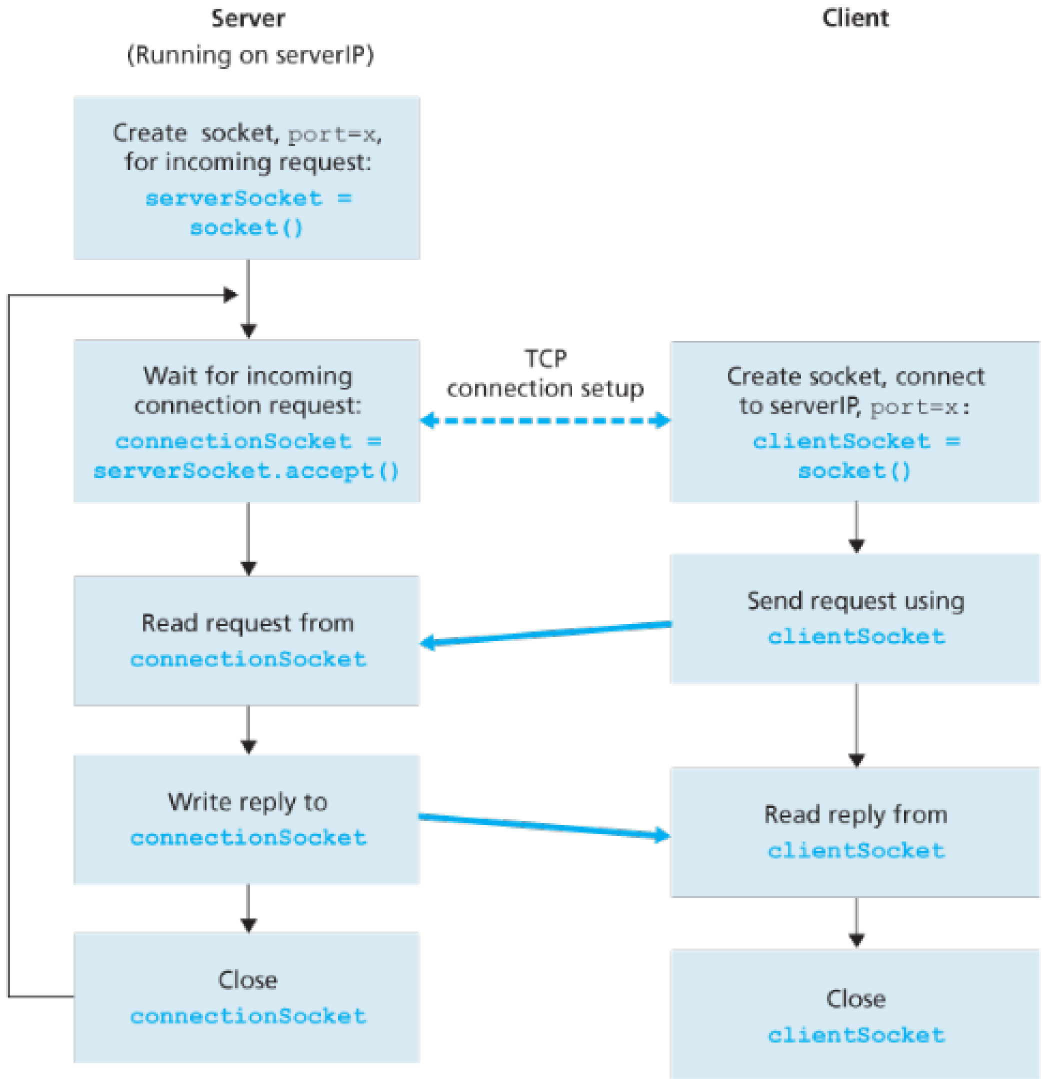
```
#Client
from socket import *
servername = <hostname>
serverport = 12000
newsock = socket(AF_INET,SOCK_DGRAM)
msg = raw_input("Enter something")
newsock.sendto(msg.encode(),(servername,serverport))
reply,serveraddress = newsock.recvfrom(2048)
print(reply.decode())
newsock.close()
```

```
#Server
from socket import *
serverport = 12000
serversocket = socket(AF_INET,SOCK_DGRAM)
serversocket.bind('',serverport)
while True:
        msg, clienaddress = serversocket.recvfrom(2048)
        x = msg.decode().upper()
        serversocket.sendto(x.encode(),clientaddress)
```

- Client port is assigned by OS and doesn't need to be explicitly specified.

TCP

- Connection oriented and reliable.
- Server must be running well before the client and ready to accept connections.
- No need to mention destination port and IP as data can just be dropped across the socket as connection is already established.
- For each client TCP duplicates the socket and makes one just dedicated to this one client.
- The 2 processes(from the perspective of the application layer are connected through a direct pipe)



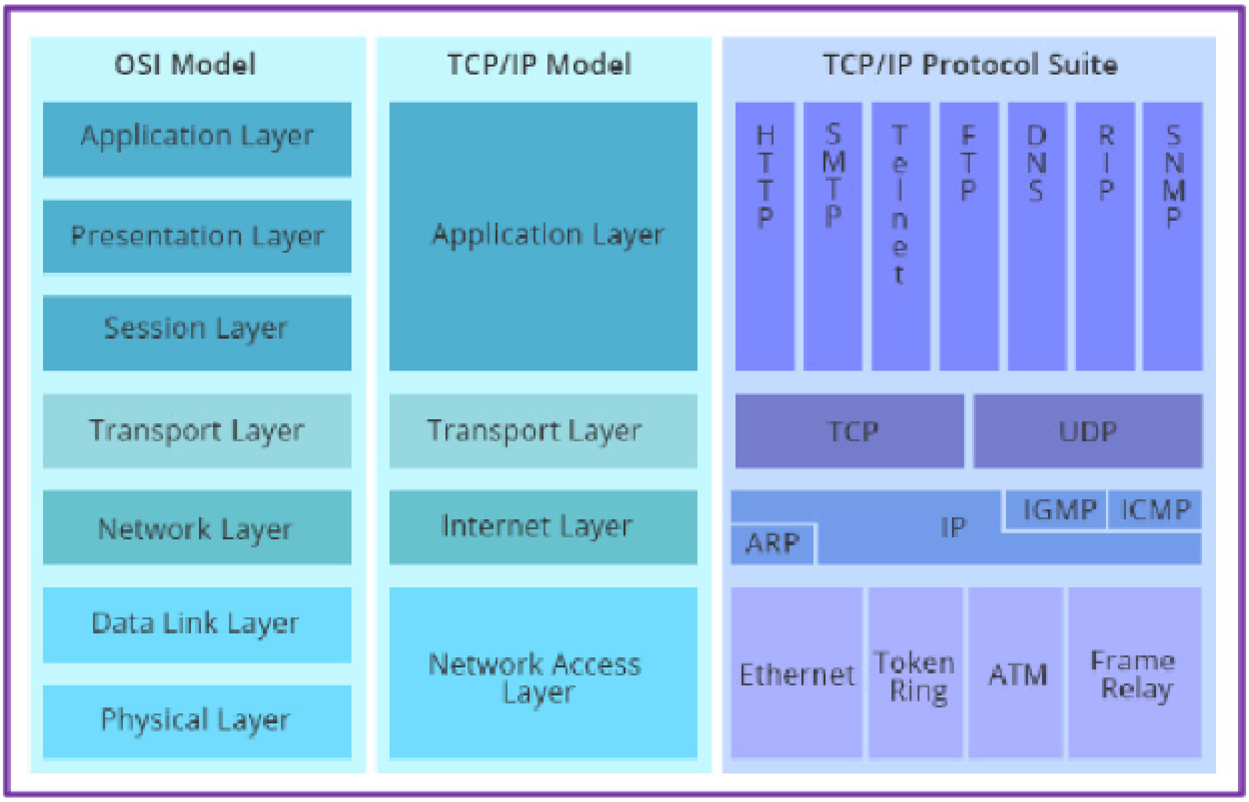

```
#Client
from socket import *
servername = <hostname>
serverport = 12000
newsock = socket(AF_INET,SOCK_STREAM)
newsock.connect((servername,serverport))
msg = raw_input("Enter something")
newsock.send(msg.encode)
reply = newsock.recv(1024)
print(reply)
newsock.close()
```

```
from socket import *
serverport = 12000
serversock = socket(AF_INET,serverport)
serversock.bind('',serverport)
serversock.listen(1)
while True:
        connectionSocket,addr = serversock.accept()
        msg = connectionSocket.recv(1024)
        x = msg.decode().upper()
        connectionSocket.send(x.encode)
        connectionSocket.close()
```

Application Layer Protocols

1. FTP
    - File Transfer Protocol
    - Exchange large files over TCP
    - Invoked from CLI or GUI
    - used to delete,rename,move or copy files in the FTP server
    - Port 20 - data connection ; Port 21 - Control Connection
2. SMTP
    - Simple Mail Transfer Protocol
    - standard way of exchanging emails over TCP
    - messages encrypted using SSL
    - messages are stored and then forwarded to destination
    - Port 25
3. DHCP
    - Dynamic Host Configuration Protocol
    - assigns IP addresses to hosts in a network dynamically
    - sometimes IP addresses may change even when computer is in network(DHCP leases)
    - based on discovery,offer,request,ACK
    - Port 67 - server ; Port 68 client
4. SNMP
    - Simple Network Management Protocol
    - exchanges management info between network devices
        - SNMP manager
        - Managed Devices
        - SNMP Agents
        - Management Information Base(MIB)
    - Port 161 & 162
5. Telnet & SSH
    - Remote access protocols
    - Port 22 - SSH ; Port 23 - Telnet

Transport Layer Services



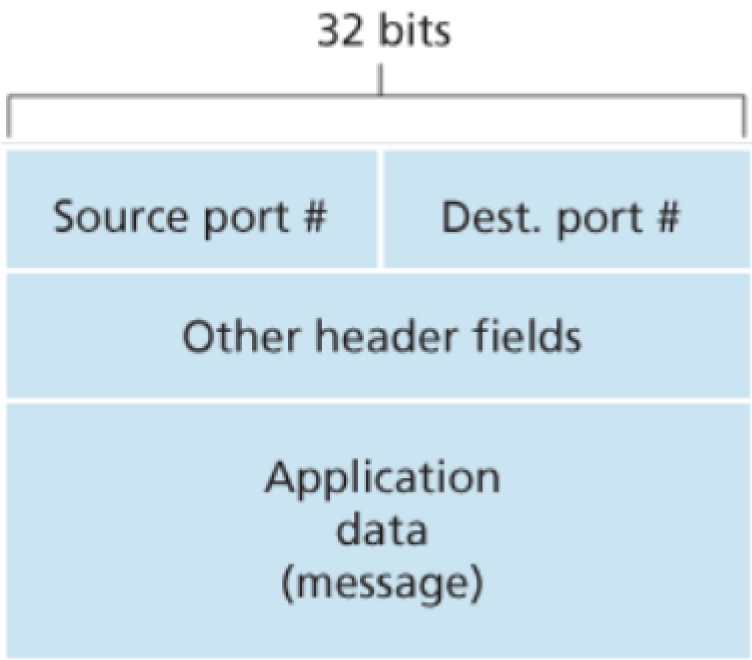
- Layer between the Application and Network layers
- Provides the illusion that hosts are directly connected to each other even when they are not
- Segment : data + headers
- Passed to network layer where it is encapsulated and sent to the network
- At the destination system it is passed to the transport layer where the segment is unpacked and data becomes available for application
- IP is unreliable by default. It makes a best ditch effort to pass data from source to destination but doesn't guarantee delivery intact.
- Transport layer protocols try to extend on IP functionality.
- UDP:
  - process to process delivery
  - error checking
- TCP:
  - Flow Control
  - Congestion Control
  - reliable connection

Multiplexing and Demultiplexing

- The action of receiving data from sockets and packaging that into a segment by attaching headers is called multiplexing.
- The action of taking the datagram from the network layer on the destination system and detaching the headers and passing the data to the application socket is called

demultiplexing.

- We need to be able to address each socket specifically as a system may have multiple open sockets to multiple processes.
- Requirements :
  - Unique identifier for sockets
  - Fields in header that dictate where to deliver data
- Source and Destination port numbers are used as identifiers.



- Port numbers are 16 bit values that range from 0-65535. But 0-1023 are reserved as well known ports and can't be assigned
- Connectionless Mux and DeMux : sockets are addressed as 2-tuples - (destination port,destination IP). Uses UDP.
  - Drawback : if 2 segments from 2 different sources have the same destination port and IP they go to the same socket.
- Connection Oriented Mux and DeMux : Overcomes the drawback as it's sockets are addressed using 4-tuple(source IP,source Port, Destination IP, Destination Port)

UDP

- Connectionless protocol
- No prior Handshaking
- Best effort protocol.
  - No guarantee of delivery to destination
- Advantages:
  - Low Latency - no handshaking so reduces delay
  - No congestion control - UDP can send data segments without waiting for decongestion

    - Simple - No need to store connection state
    - Smaller header than TCP
- Used for applications involving large number of hosts and where real time error correction isn't necessary.
    - DNS lookup
    - SNMP
    - Trivial FTP(TFTP)
    - Real time video streaming
- If needed congestion control,error correction and reliability can be added in the application level

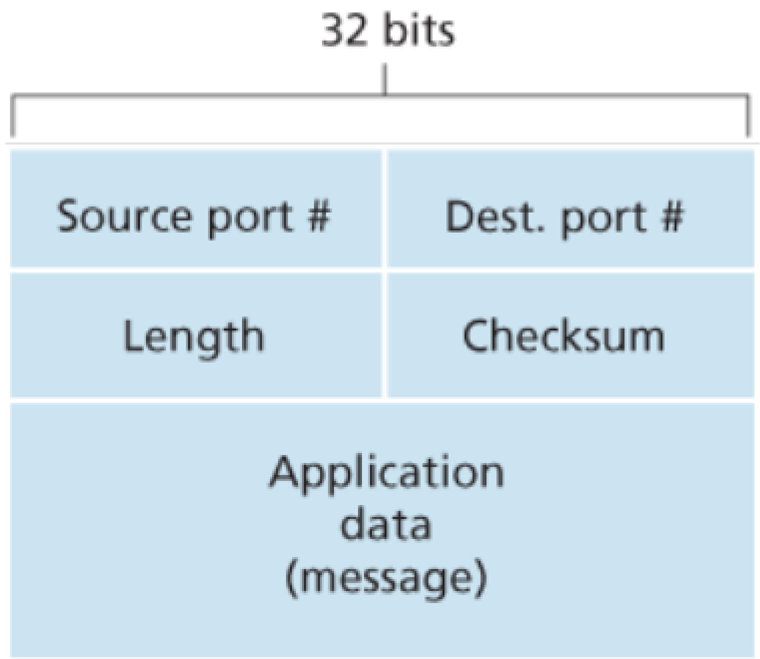


Figure 3.7 UDP segment structure

- each header field is 2 bytes long

UDP Checksum

- For error detection
- All 16-bit words added. If there is carry, add that to the sum.
- Then take 1's complement
- This is the checksum
- At the receiver we add all the words with the checksum. If there is no error, sum should be 11111111111111...