Application Layer

Arnav Gupta

December 2, 2024

Contents

1 Principles of Network Applications **1**

2 Socket Programming **3**
2.1 UDP 3
2.2 TCP 3

3 Web and HTTP **3**
3.1 QUIC: Quick UDP Internet Connections 5

4 Email, SMTP, IMAP **5**
4.1 SMTP 6

5 Domain Name System (DNS) **6**

1 Principles of Network Applications

Network applications are programs that run on (different) end systems and communicate over the network.

There is no need to write software for network-core devices since they do not run user applications. Also, applications on end systems allow for rapid app development and propagation.

In client server paradigm:

- server is an always-on host with a permanent IP address, often in data centers for scaling
- clients (HTTP, IMAP, FTP)

  - contact and communicate with the server
  - may be intermittently connected and have dynamic IP addresses
  - do not communicate directly with each other

In peer-peer architecture (like P2P file sharing):

- no always on server so end systems directly communicate
- peers request service from and provide service to other peers
  - **self scalability**: new peers bring new service capacity as well as new service demands
- peers are intermittently connected and change IP addresses (so management is complex)

Process: program running with a host

With the same host, two processes communicate using inter-process (defined by OS). In different hosts, processes communicate by exchanging messages.

Client process: process that initiates communication

Server process: process that waits to be contacted

Both are present in P2P architectures.

Application-layer protocol defines:

- types of message exchanged
- message syntax: what fields in messages and how fields are delineated
- message semantics: meaning of info in fields
- rules for when and how processes send and respond to messages

Protocols can be:

- open protocols: defined in RFCs, everyone has access to protocol definition, allows for interoperability
- proprietary protocols

To choose a transport service, consider:

- **data integrity**: can loss be tolerated or is reliability required
- **timing**: some apps require low delay to be effectve

- **throughput**: does the app require some minimum amount of throughput to be effective
- **security**: encryption and data integrity

2 Socket Programming

Socket: door between application process and end-to-end transport protocol

Sockets can be UDP (unreliable datagram) or TCP (reliable, byte stream-oriented).

2.1 UDP

No connection between client and server (no handshake).

Sender explicitly attaches IP destination address and port number to each packet. Receiver extracts sender IP address and port number from received packet.

Transmitted data may be lost or received out-of-order.

2.2 TCP

Client must contact running server, which must have created socket to welcome client's contact.

Client contacts server by creating TCP socket with IP address and port number specified.

When contacted by client, server TCP creates new socket for server process to communicate with that particular client:

- allows server to talk with multiple clients
- source port numbers used to distinguish clients

3 Web and HTTP

Web page consists of objects which can be stored on different web servers, like base HTML file which includes referenced objects, each addressable by a URL.

HTTP: HyperText Transfer Protocol is the web's application layer protocol:

- client: browser that requests and receives web objects
- server: web server sends objects in response to requests

HTTP uses TCP:

- client initiates TCP connection to server at port 80
- server accepts TCP connection from client
- HTTP messages exchanged between client and server
- TCP connection closed

Server maintains no information about past client requests. (protocols that maintain stat are complex)

Non-persistent HTTP:

1. TCP connection opened
2. at most 1 object sent over TCP connection
3. TCP connection closed

For non-persistent HTTP,

$$\text{Response Time} = 2\text{RTT} + \text{File transmission time}$$

This requires 2 RTTs per object and OS overhead for each TCP connection. Browsers often open multiple parallel TCP connections to fetch referenced objects in parallel.

Persistent HTTP:

1. TCP connection opened to a server
2. multiple objects can be sent over a single TCP connection between client and that server
3. TCP connection closed

For persistent HTTP, as little as 1 RTT for all referenced objects (cutting response time in half).

HTTP request message is ASCII (human readable format), with request line (GET, POST, HEAD, etc), header lines, carriage return and line feed at start of line to indicate end of header lines, and body.

HTTP request messages can be POST (often form input), GET (query for user data), HEAD (headers only returned), and PUT (upload new file or replace existing file).

HTTP response message is also ASCII with status line (protocol, status code, status phrase), header lines, and data (after `\r\n`).

HTTP response status codes include 200 OK, 301 Moved Permanently, 400 Bad Request, 404 Not Found, and 505 HTTP Version Not Supported.

3.1 QUIC: Quick UDP Internet Connections

Application-layer protocol on top of UDP that increases the performance of HTTP.

Adopts TCP approaches for connection establishment, error control, and congestion control.

Uses multiple application-level streams multiplexed over single QUIC connection, so separate reliable data transfer and security, as well as common congestion control.

QUIC does a single handshake for TCP and TLS (security/auth) together in one.

With parallelism from QUIC stream, there is no head-of-line blocking (no need to wait for earlier requests to go through since parallel).

4 Email, SMTP, IMAP

Major components are user agents, mail servers, and Single Mail Transfer Protocol (SMTP).

User agent is mail reader, for composing, editing, and reading mail messages.

Mail server has mailbox with incoming messages for user and message queue of outgoing mail messages.

SMTP used between mail servers to send email messages.

4.1 SMTP

Use TCP to reliably transfer email message from client (mail server) to server with port 25.

3 phases of transfer:

- SMTP handshaking
- SMTP transfer of messages
- SMTP closure

Command/response interaction with ASCII text commands and status code/phrase response.

As opposed to HTTP, which is client pull SMTP is client push. In HTTP, each object is encapsulated in its own response message. In SMTP, multiple objects are sent in multipart message.

SMTP uses persistent connections, requires message to be in 7-bit ASCII, and uses CRLF.CRLF to determine end of message.

SMTP email message contains header lines (to, from, subject) and body (message).

Internet Mail Access Protocol (IMAP) provides retrieval, deletion, and folders of stored messages on mail server (retrieval from mail server to user agent).

5 Domain Name System (DNS)

Distributed database implemented in hierarchy of many name servers. An application-layer protocol where hosts and DNS servers communicate to resolve names (translate from name to address). This is a core Internet function implemented as an application-layer protocol.

Hostname-to-IP address translation to assist with host aliasing (canonical alias names) and mail server aliasing. Also helps with load distribution through replicated web servers (many IP addresses correspond to one name).

Centralizing DNS would cause single point of failure, too high traffic volume, distant centralized DB, and difficult maintenance. Overall, wouldn't scale.

For DNS, billions of simple records and trillions of queries daily with many more reads than writes. Must be reliable and secure always, while being

organizationally and physically decentralized.

Hierarchy:

- root server for top level domain DNS server
  - official, contact of last resort by name servers that cannot resolve name
  - managed by ICANN
  - DNSSEC → provides security (auth and message integrity)
  - 13 globally
- top level domain DNS server for authoritative DNS servers
  - responsible for TLDs like .com, .org, etc
- authoritative DNS servers for IP address
  - organization's DNS server providing authoritative hostname to IP mappings for organization's named hosts
  - can be managed by organization or service provider

DNS query is sent to local DNS server, which returns reply answering from local cache or forwarding request into DNS hierarchy.

Each ISP has local DNS server (does not strictly belong to hierarchy).

Iterated query: contacted server replies with name of server to contact

Recursive query: puts burden of name resolution on contacted name server, can cause heavy load at upper levels of hierarchy

Once any name server learns mapping, this is cached and the cached mapping is returned. Improves response time and cache entries timeout after some TTL. TLD servers typically cached in local name servers.

Cached entries can be out of date, so DNS is best effort.

DNS records have format (`name`, `value`, `type`, `ttl`) and can have different types:

- type A: `name` is hostname, `value` is IP address
- type NS: `name` is domain, `value` is hostname of authoritative name server for domain

- type CNAME: `name` is alias name, `value` is canonical name
- type MX: `value` is name of SMTP mail server associated with `name`

DNS query and reply messages have the same format:

- message header: identification (16 bit number for query and reply) and flags (query or reply, recursion desired, recursion available, reply is authoritative)
- numbers of questions, answer records, authority records, additional records
- questions
- answers
- authority
- additional info

To get info into DNS, register name at DNS registrar by providing names and IP addresses of authoritative name server (primary and secondary). Then, create authoritative server locally with IP address.