Advanced Telecommunications

by Ultan

Introduction

Notes from the course, Advanced Telecommunications I took. These are notes that are relevant for a general software engineering position.

Other courses, from my degree, that are relevant for a general, entry level software engineering position are Introduction to Programming, Algorithms and Data Structures, Information Management II and Software Engineering. Notes for these courses are also included in `../`.

These summary notes are based mainly on content from the course, Advanced Telecommunications, but also from the course Introduction to Computer Security at the University of Texas at Austin, linked here.

Open Systems Interconnection Model

A conceptual model.

1. Physical Layer
2. Data Link Layer
3. Network Layer
4. Transport Layer
5. Session Layer
6. Presentation Layer
7. Application Layer

Internet Protocol Suite

Another conceptual model.

1. Link Layer
2. Internet Layer
3. Transport Layer
4. Application Layer

Bits

Bytes, Halfwords and Words

8 bits = 1 byte

2 bytes = 16 bits = 1 halfword

4 bytes = 32 bits = 1 word

Units

Value	**ISO/IEC 80000**
1	B (byte)
1024B	KiB (kibibyte)
1024KiB	MiB (mebibyte)

- Bit rate: Number of bits that we can send over a given period of time (bits per second)
- Bandwidth: Maximum bit rate of a system
- Latency: Time it takes for a bit to travel from one place to another
- Example: 3MiB is 25,165,824 bits. To send in 3s the bit rate needs to be 25,175,824 / 3 which is 8,388,608 bits per second

Bitwise Logical Operations

A	**B**	**A AND B**	**A OR B**	**A EOR B**	**NOT A**
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Bit Manipulation

Clear Bits 3 and 4: Observe 0 & x = 0, 1 & x = x

Before	1	0	0	**1**	**0**	1	1	0
Mask	1	1	1	0	0	1	1	1
After	1	0	0	0	0	1	1	0

Set Bits 3 and 4: Observe 0 | x = x, 1 | x = 1

Before	1	0	0	**1**	**0**	1	1	0
Mask	0	0	0	1	1	0	0	0
After	1	0	0	1	1	1	1	0

Invert Bits 3 and 4: Observe 0 EOR x = x, 1 EOR x = NOT x

Before	1	0	0	**1**	**0**	1	1	0
Mask	0	0	0	1	1	0	0	0
After	1	0	0	0	1	1	1	0

Querying the Status of a Bit: Is bit 4 set? Clear all other bits using AND. If after is non-zero, then the bit is set.

Before	1	0	0	**1**	0	1	1	0
Mask	0	0	0	1	0	0	0	0
After	0	0	0	1	0	0	0	0

Hexadecimal Number System

One “hex” digit can represent the same sixteen values as four binary digits (bits)

UDP

- User Datagram Protocol: A transport layer protocol
- Connectionless protocol (no handshaking)
- Connection in message stream
- Supports broadcasting
- No error control or flow control (segments may be lost or delivered out of order)
- No full duplex
- Packet is called a user datagram

TCP

- Transmission Control Protocol: A transport layer protocol
- Connection orientated protocol (handshaking - initialise sender and receiver state before data exchange)
- Connection in byte stream
- Doesn’t support broadcasting (point-to-point - one sender and one receiver)
- Error control and flow control (sender will not overwhelm the receiver)
- Full duplex (bi-directional data flow)
- Packet is called a segment

HTTP

- Hypertext Transfer Protocol: An application layer protocol

Client-Server

- Server: Always-on, permanent IP address, data centre for scaling
- Client: Communicate with the server, may be intermittently connected, may have dynamic IP addresses, clients do not communicate directly with each other

Processes

- A program running within a host
- Within the same host, two processes communicate using inter-process communication determined by the OS
- Processes in different host communicate by exchanging messages
- Client process: A process that initiates communication
- Server process: A process that waits to be connected

Objects

- A web page consists of objects e.g. HTML file, JPEG image

- A web page consists of a base HTML file which includes several referenced objects
- An object is addressable by a uniform resource locator: www.site.edu (host name) and /main/pic.jpeg (path name)

Non-Persistent HTTP

- At most one object sent over a TCP connection. Connection then closed
- Requires two RTT per object (one RTT to initiate TCP connection and one RTT for HTTP request to return)
- OS overhead for each TCP connection
- Browsers often open parallel TCP connections to fetch referenced objects
- HTTP is stateless

1(a). HTTP client initiates TCP connection to HTTP server (process) at www.test.edu on port 80

1(b). HTTP server at host www.test.edu waiting for TCP connection at port 80. Accepts the connection and notifies the client

1. Client sends HTTP request message (containing URL) into TCP connection socket. Message indicates the object the client wants
2. Server receives request message, forms response message containing the requested object, and send message into its socket
3. Server closes TCP connection
4. Client receives response message containing HTML file. It displays the HTML file. Parsing the file it finds more referenced objects
5. Steps 1-4 repeated for each of the referenced objects

Persistent HTTP

- Multiple objects can be sent over a single TCP connection (server leaves connection open after sending a response)
- This is the default mode of HTTP and it uses pipelining

Request Message

- ASCII format
- GET: Used to retrieve an object
- GET: (Alternative) Form input is uploaded in the URL field of the request line
- POST: Form input is uploaded in entity body
- PUT: File is uploaded in entity body
- DELETE: Deletes specified file

Request Line Header Lines Blank Line Body

Response Message

- ASCII format

Status Line
Header Lines
Blank Lines
Body

Response Codes

- 1s: Informational responses
- 2s: Success
- 3s: Redirection
- 4s: Client-side error
- 5s: Server-side error

Common Response Codes

- 200 OK: Request succeeded, requested object later in this msg
- 301 Moved Permanently: Requested object moved, new location specified later in this msg (Location)
- 400 Bad Request: Request message not understood by server
- 404 Not Found: Requested document not found on this server
- 505 HTTP Version Not Supported

Cookies

- HTTP is stateless. Maintain state using cookies that are carried in HTTP messages
- Server creates an ID when it gets a request. It keeps a record of this ID and then sends the ID in the response. The client keeps a record of this ID. Future requests from the client will contain the ID. Thus, the server can identify a client who has been in touch before.

DNS

- Domain Name System: An application layer protocol
- A distributed hierarchical database of many name servers
- Purpose is for hostname to IP address translation
- Has *query* and *reply* messages, both with the same message format

Types of Name Servers

- Root: Returns list of IP addresses for responsible TLD servers
- Top-level domain: Responsible for com, org, edu, ie, jp etc.

- Authoritative: Organisation's own name servers e.g. amazon.com

Name Resolution Example

- Host makes a DNS query. Query is sent to the local name server
- Local name server is part of ISP (residential ISP, company, university). Also called default name server
- Local name server wants IP for www.amazon.com
- Local name server queries root server to find .com DNS server
- Local name server queries .com DNS server to get amazon.com DNS server
- Local name server queries amazon.com DNS server to get IP address for www.amazon.com

The above is an iterated query example. Alternatively, a recursive query can be used. Puts burden of name resolution on the contacted name server.

Caching

- Name servers cache mappings as they learn about them. Root name servers are not often visited
- Cached entries timeout after some time (TTL). IP address changes take time to update Internet-wide

Resource Records

- The database stores RR: (Name,Class,Type,TTL,Value)
- A record: IPv4 address record (IPv4 - 32 bits per address)
- AAAA record: IPv6 address record (IPv6 - 128 bits per address)
- NS record: Name is domain (e.g. example.com), value is hostname of authoritative server for this domain (e.g. ns.example.com)
- CNAME record: Name is alias (e.g. ftp.example.com), value is the canonical (e.g. www.example.com)
- MX record: Name (e.g. example.com), value is the mail server (e.g. mail1.example.com)

DNSSEC

- For authenticity and integrity of messages. Makes use of digital signatures. DNSSEC adds a number of new resource record types

WebSockets

- An application layer protocol
- Synchronous web communication: User must wait while new page loads
- Asynchronous web communication: User can keep interacting with the page while data loads
- WebSockets application layer protocol allow client-side to open (WebSocket handshake) and persist a connection with the server
- Full duplex: Both parties can send data at any time using messages (contrast with Ajax where server has no method for pushing messages to the client)
- Framing: Each message has frames which contain the payload
- Reduction in latency: Frame-based messaging reduces the amount of non-payload data that is transferred

Modular Arithmetic

Integer division

$$\frac{a}{m} = q \text{ and remainder } r$$

$$\lfloor \frac{a}{m} \rfloor = q$$

Always possible to write a such that:

$$a = q * m + r, 0 \leq r < m$$

Modulo operator

$$\frac{13}{5} = 2 \text{ and remainder } 3$$

$$13 mod 5 = 3$$

Congruence modulo

$$a \equiv r \pmod{m}$$

- $\equiv$ is the symbol for congruence, which means a and r are in the same equivalence class
- $\pmod{m}$ is the operation that was applied to a and r
- Thus, if a is congruent to $r \pmod{m}$ then $a \pmod{m} = r \pmod{m}$

Congruence modulo is an equivalence relation for $\pmod{m}$

Reflexive

$$a \equiv a \pmod{c}$$

Symmetric

$$\text{if } a \equiv b \pmod{c} \text{ then } b \equiv a \pmod{c}$$

Transitive

$$\text{if } a \equiv b \pmod{c} \text{ and } b \equiv d \pmod{c} \text{ then } a \equiv d \pmod{c}$$

Addition

$$(a+b) \bmod c = ((a) \bmod c + (b) \bmod c) \text{mod } c$$

Subtraction

$$(a-b) \bmod c = ((a) \bmod c - (b) \bmod c) \text{mod } c$$

Multiplication

$$(a * b) \text{ mod } c = ((a) \text{ mod } c * (b) \text{ mod } c) \text{mod } c$$

Exponentiation

$$a^b \text{ mod } c = ((a \text{ mod } c)^b) \text{ mod } c$$

Multiplicative Inverse

The multiplicative inverse of a modulo m is the x value that makes:

$$ax \equiv 1 \pmod{m}$$

The multiplicative inverse only exists for a if an only if a and m are co-prime/relatively prime/mutually prime (i.e. $gcd(a, m) = 1$)

Network Security

Aims

- Confidentiality: Only sender and intended receiver should "understand" the message contents
- Authentication: Sender and receiver want to confirm identity of each other
- Integrity: Sender and receiver want to ensure the message is not altered

Cryptography

- Decryption function, D. Encryption function, E. Plaintext, P. D(E(P)) = P.

Symmetric-Key Encryption

- Sender and receiver knowing and using the same secret key. The same crypto-system
- Sender uses the secret key to encrypt the message. Receiver uses the secret key to decrypt the message
- One example is Data Encryption Standard: A block cipher (works on a single chunk of data at a time). 64 bit block, 56 bit key
- Block Cipher Operation Modes: ECB, CBC, OFB, CFB
- Another example is Advanced Encryption Standard: A block cipher. Variable key and block size of 128, 192 and 256 bits. Most common is 128 key and block size

Asymmetric-Key Encryption

- In public-key cryptography, each person generates a public and private key. Public key is published and distributed. Private key is kept secret
- Must be computationally easy to encipher or decipher a message given the appropriate key. Must be computationally infeasible to derive the private key from the public key
- Alice takes plaintext and Bob's public key and creates the cipher text. Bob receive's the cipher text and decrypts it using his own private key
- One example is Rivest Shamir Adleman: Security based on the difficulty of factoring very large numbers

```
keygen() -> (public key, private key)  (this function is randomized)

encrypt(plaintext: array<byte>, public key) -> array<byte>  (the ciphertext)
decrypt(ciphertext: array<byte>, private key) -> array<byte>  (the plaintext)
```

Hybrid Schemes

- Asymmetric-key algorithms are not a replacement for symmetric-key algorithms. They supplement fast bulk encryption ciphers such as DES
- We use a public-key algorithm to securely transfer a session key. And, use the session key for bulk encryption and decryption

Authentication & Integrity

- Use a message digest/cryptographic hash
- A message digest is a strong digital fingerprint of a message
- Takes an input m, produces fixed length value, $H(m)$. 128/160/256 bits
- Computationally infeasible to find two different messages, x and y such that $H(x) = H(y)$. This means the hash is unique to a message
- Examples are MD2, MD4, MD5, SHA-1, SHA-2
- Hash functions are deterministic: The same input always generates the same output
- Hash functions are non-invertible: Hard to find an input m such that $hash(m) = h$ for some desired output h
- Hash functions are target collision resistant: Given an input m_1, it's hard to find a different input m_2 such that $hash(m_1) = hash(m_2)$
- Hash functions are collision resistant: It's hard to find two inputs m_1 and m_2 such that $hash(m_1) = hash(m_2)$

Digital Signatures

- Bob
  - Hash function: Large message m -> $H(m)$
  - Digital signature (encrypt): $H(m)$ plus Bob's private key -> encrypted message digest
  - Send large message m and digital signature
- Alice
  - Receive large message m and digital signature
  - Hash function: Large message m -> $H(m)$ (a)
  - Digital signature (decrypt): Encrypted message digest plus Bob's public key -> $H(m)$ (b)
  - and (b) should be equal and this proves that Bob and no one else signed the document (verifiable and non-forgeable). We are assuming the public key is good

```
sign(message: array<byte>, private key) -> array<byte>  (the signature)
verify(message: array<byte>, signature: array<byte>, public key) -> bool  (whether or no
```

Key Management

- User's on the network must be able to retrieve a public key and associate a user's identity with it
- Enlist the services of a trusted part (TPP)
- A TPP constructs a message referred to as a certificate
- Certs are digitally signed with the private key of the TPP. Every user in the system is equipped with the public-key of the TPP. Allows one to verify the digital signatures on the certificate guaranteeing that the public key on the cert belongs to the named user on the cert
- Example is X.509 certificates
- TPPs that issue certificates are referred to as certification authorities (CAs). The root CA issues certs only to other CAs. Each user on the network need only hold the public key of the root CA

Diffie-Hellman Key Exchange (DHKE)

- Allow establishing a shared symmetric key without having to meet
- Several classes of cryptographic schemes rely on g^k being easy to compute ($log(k)$ operations by using repeated squaring) and finding k from g^k being hard (known as the "discrete logarithm" assumption)

1. Alice and Bob agree on a large prime p and a generator g for p
2. Alice randomly generates a number a, Bob randomly generates a number b
3. Alice sends g^a to Bob, Bob sends g^b to Alice
4. Alice and Bob independently calculate g^{ab} and use this as the shared key

EIGamal Encryption Scheme

- The EIGamal Encryption Scheme is an extension of the DHKE. In contrast to DHKE, no TTP is needed to choose a prime and primitive element. One issue with the scheme is overhead, as the cipher text is twice as long as the message

Elliptic Curve Cryptography

- A more efficient scheme based on the discrete logarithm assumption (fewer bits needed for the same security, fewer computations needed)

Key Lengths and Efficiency

Algorithm Family	Crypto System	Security level 80b	Security level 128b	Security level 192b	Security level 256b
Int. factorisation	RSA	1024	3072	7680	15360
Discrete log	DH, EIGamal	1024	3072	7680	15360
Elliptic curves	ECDH, ECDSA	160	256	384	512
Symmetric key	AES, 3DES	80	128	192	256

- 256b ECC key provides the same security as a 3072b RSA key

Message Authentication Code from Hash Functions (HMAC)

- Bob
  - Hash function: Message m plus shared secret key -> MAC
  - Send message m and MAC
- Alice
  - Receive message m and MAC
  - Hash function: Message m plus shared secret key -> MAC (a)
  - MAC (a) same as MAC then accept message
- Like digital signatures they can be used to quickly verify data integrity and authenticity of a message

Authentication Using a Key Distribution Centre

- To talk to n people using a shared secret would require setting up a number of keys
- Alternatively, use a KDC
- Each user has a single shared key with the KDC. Authentication and session key management happens through the KDC

Needham-Schroder Authentication Protocol

- A shared-key authentication protocol designed to generate and propagate a session key, i.e. a shared key for subsequent symmetrically encrypted communication
- Note that there is no public key infrastructure in place
- There are three principles: A and B, two principles desiring mutual communication, and S, a trusted key server
- It is assumed that A and B already have secure symmetric communication with S using keys K_{as} and K_{bs}respectively
- Needham-Schroder uses nonces (short for "numbers used once"), randomly generated values included in messages
- If a nonce is generated and sent by A in one step and returned by B in a later step, A knows that B's message is fresh and not a replay from an earlier exchange
- Note that a nonce is not a timestamp. The only assumption is that it has not been used in any earlier interchange, with high probability

$$A \to S : A, B, N_a$$

$$S \to A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$$

$$A \to B : \{K_{ab}, A\}_{Kbs}$$

$$B \to A : \{N_b\}_{Kab}$$

$$A \to B : \{N_b - 1\}_{Kab}$$

- Here N_a and N_b are nonces
- Kerberos V5 is a widely used protocol that builds on the idea of Needham-Schroder. It includes an authentication server and a ticket granting server

Secure Sockets Layer (SSL)

- Provides transport layer security to any TCP-based application using SSL services. For example, between web browsers and servers for electronic commerce (in practice Transport Layer Security should be used)
- Provides server authentication, data encryption and client authentication (optional)
- Normal application

Application
TCP
IP

- Application with SSL

Application
SSL
TCP
IP

- Cipher suite: Public-key algorithm (e.g. RSA), symmetric encryption algorithm (e.g. DES), MAC algorithm

SSL Handshake

- Server authentication
- Negotiation - agree on crypto algorithms
- Establish keys
- Client authentication (optional)

Virtual Private Networks

- Institutions often want private networks for security. This is costly (separate routers, links, DNS infrastructure)
- With a VPN, institution's inter-office traffic is sent over public internet instead. But, inter-office traffic is encrypted before entering public internet
- IPsec is used

Passwords

- Entropy is a measure of randomness. It is measured in bits, log_2 of possibilities . A fair coin flip gives 1 bit of entropy. A dice roll has 2.58 bits of entropy. For online guessing, 40 bits is good for a password