

50.005 Computer System Engineering

Shaun Toh, Joel Huang

April 24, 2018

1 Networking Basics

1.1 How to share the Internet

Packets are a unit of data made out of a header and a payload.

The internet is a shared resource, so how do we allocate resources, given data traffic is bursty by nature? Packet Switching and Statistical Multiplexing are essentially carried out at the same time on networks. Packets enable accountability, statistical multiplexing allows multiple channels.

1.1.1 Packet Switching

Packet Switching is breaking data into blocks instead of a stream.

Connectionless Packet Switching is when each packet contains the complete address and/or routing information, opening the possibility of an out-of-order delivery and different paths of transmission, while also possibly increasing the speed due to the ability to route packets to network nodes with lower loads.

Packets contain the following information:

- Destination address
- Source Address
- Total Piece number
- Sequence number, for reassembly and accounting purposes.

Packets are rearranged to form the original message after receiving. This allows for checking of the various packets and requests for re-sending on data corruption or packet loss.

Connection Oriented Packet Switching is very similar, the only difference being the route is pre-defined, so address information is not sent with the packets. This is also known as virtual circuit switching.

1.1.2 Multiplexing

Statistical Multiplexing

Statistical Multiplexing is dynamically allocating bandwidth to a channel on a as-needed basis.

Bandwidth is only allocated to channels that are currently broadcasting as a result, we are able to use multiple communications on the same channel at the same time. Keep in mind that this is a dynamic method, as opposed to the two methods listed below. The main difference is that the sharing is adapted to the current demand, as opposed to a static link sharing method like the next two methods.

Circuit Switching

- **Frequency Dimension Multiplexing**

Frequency Dimension Multiplexing is essentially dividing up total bandwidth in a communication medium into non-overlapping frequency bands. This allows one channel of communication through each band of frequency.

NOTE: Circuit Switching is a mode of communication where two clients have to establish a dedicated communications channel (a circuit) through the network before any communication can take place. As a result, the full bandwidth of the particular medium is available.

- **Time Dimension Multiplexing**

Time dimension multiplexing is transmitting and receiving independent signals over the same signal path. While it guarantees complete usage of the entire bandwidth during the period on which this signal is using the line, other signals cannot share the line in the meantime. Instead, all signals take turns using the entire bandwidth. The high speed at which it occurs creates an illusion of concurrency. (While it is effectively concurrent to us).

1.2 Layering

The same way that making your code in terms of modules and functions enables modularity and easier debugging (you can narrow down the source), Networking is also layered. With each layer implementing only a single service, and only interacting with the layer above and below it, we can narrow down the problem with greater ease. Of note is that Network Layering results in cascaded headers. i.e. Every layer just adds their own header to the packet that is being sent.

1.2.1 The Internet Protocol Stack

The Internet protocols are layered as follows:

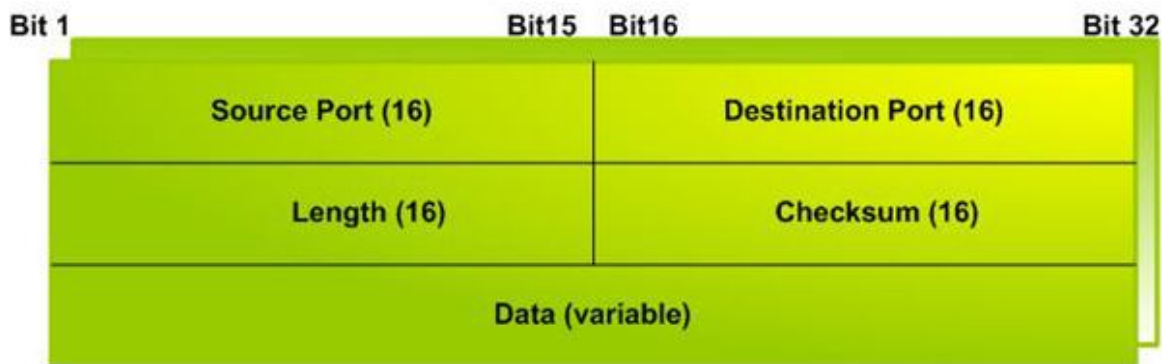
- **Application Layer Protocols:** Supporting network Applications: FTP, SMTP, HTTP
- **Transport: Process-process data transfer:** TCP, UDP
- **Network:** Routing of datagrams from Source to destinations -IP, Routing protocols
- **Link:** Data transfer between neighbouring network elements: Ethernet, 802.11 (WiFi), PPP
- **Physical:** Bits "On the Wire"

1.3 Host to Host Layer Protocols

1.3.1 UDP

The User Datagram Protocol is lightweight compared to TCP. It doesn't require reliable delivery and extra overhead, so reliability is outsourced to the application layer. **Real time traffic** utilizes this protocol as it cares only about fast transmission. UDP packets have a header of 8 bytes (64 bits):

The UDP Segment Format

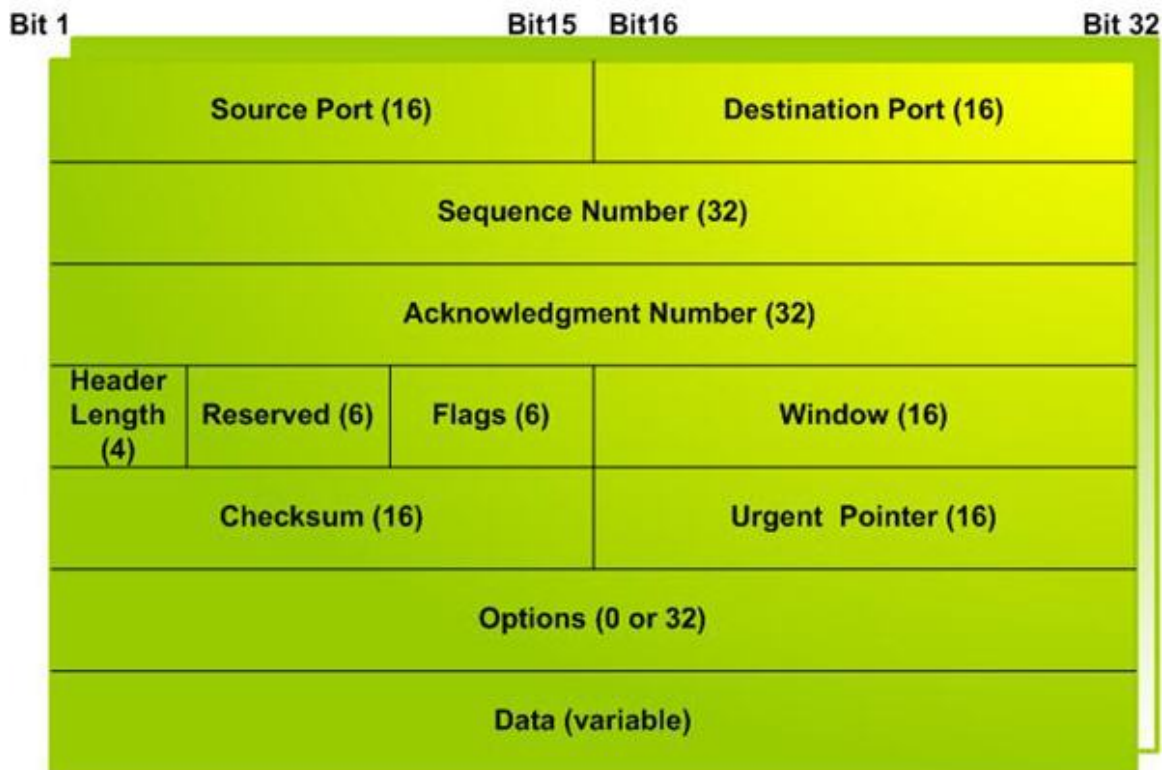


<<https://www.pluralsight.com/blog/it-ops/networking-basics-tcp-udp-tcpip-osi-models>>

1.3.2 TCP

TCP stands for Transmission Control Protocol. It is known to be connection oriented. This is because both parties have to establish a connection before beginning to transmit data. TCP communicates via segments of data, consisting of a fixed 20 byte header and a variable size data field at the end. Of note is the **constant back and forth of acknowledgement** requests from both machines. This protocol is hence able to maintain and ensure integrity of data sent and received.

The TCP Segment Format



<<https://www.pluralsight.com/blog/it-ops/networking-basics-tcp-udp-tcpip-osi-models>>

- **Source Port** and **Destination Port** fields together identify the two local end points of the particular connection. A port plus its hosts' IP address forms a unique end point. Ports are used to communicate with the upper layer and distinguish different application sessions on the host.
- The **Sequence Number** and **Acknowledgment Number** fields specify bytes in the byte stream. The sequence number is used for segment differentiation and is useful for reordering or re-transmitting lost segments. The Acknowledgment number is set to the next segment expected.
- **Data offset** or TCP header length indicates how many 4-byte words are contained in the TCP header.
- The **Window** field indicates how many bytes can be transmitted before an acknowledgment is received.

- The **Checksum** field is used to provide extra reliability and security to the TCP segment by detecting changes in the header and data content. IPv4 and IPv6 use different methods to compute the checksum.
- The actual **Data** is included after the end of the header.

1.3.3 Internet Control Message Protocol

The Internet Control Message Protocol (ICMP) is a supporting protocol, generally used for sending error messages or operational information. It is not typically used to transmit data and is utilised by **Ping** and **Traceroute**. Both end hosts and routers utilise this protocol. It is not the same as other transport protocols such as UDP or TCP.

1.3.4 General comparison between TCP and UDP

TCP	UDP
Reliable	Unreliable
Connection-oriented	Connectionless
Segment retransmission and flow control through windowing	No windowing or retransmission
Segment sequencing	No sequencing
Acknowledge segments	No acknowledgement

<<https://www.pluralsight.com/blog/it-ops/networking-basics-tcp-udp-tcpip-osi-models>>

1.4 Application Layer Protocols

Application layer protocols define how applications running on different end systems, process and pass messages to each other. The Application Layer Protocols determine the following aspects:

- The types of messages, e.g., request messages and response messages.
- The syntax of the various message types, i.e., the fields in the message and how the fields are delineated.
- The semantics of the fields, i.e., the meaning of the information that the field is supposed to contain.
- Rules for determining when and how a process sends messages and responds to messages.

Applications adhere by these rules allowing for a universal standard of communication. There are a few application protocols that are highlighted in this course.

1.4.1 HTTP (TCP)

HyperText Transfer Protocol (HTTP) defaults to port 80. It utilizes TCP connections to send client requests and server replies.

1.4.2 FTP (TCP/UDP)

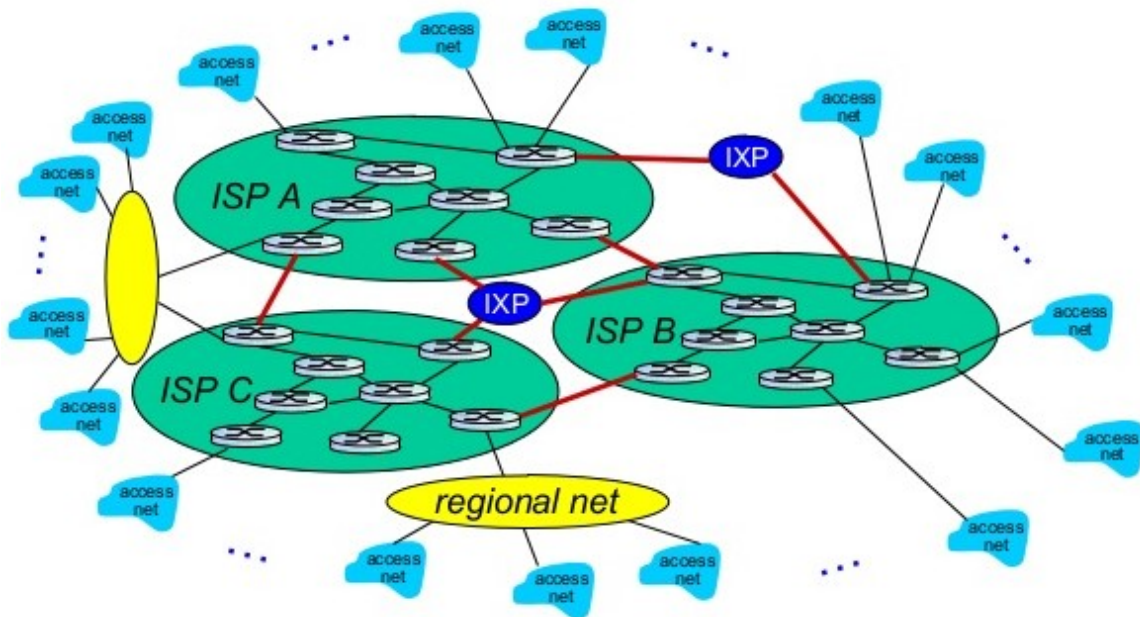
File Transfer Protocol establishes 2 connections between hosts. It uses one connection for control, and another for data. As a result of the absolute separation, it is able to transfer data more reliably and efficiently. FTP is able to handle binary and .txt formats.

1.4.3 SMTP (TCP)

Simple Mail Transfer Protocol is used for emails. It provides services for mail exchange between users on the same or different computers. It is based off a client-server model. Both clients and servers will run this protocol. If you send mail, you act as a client. Receiving mail means you act as a server.

1.5 Internet Structure

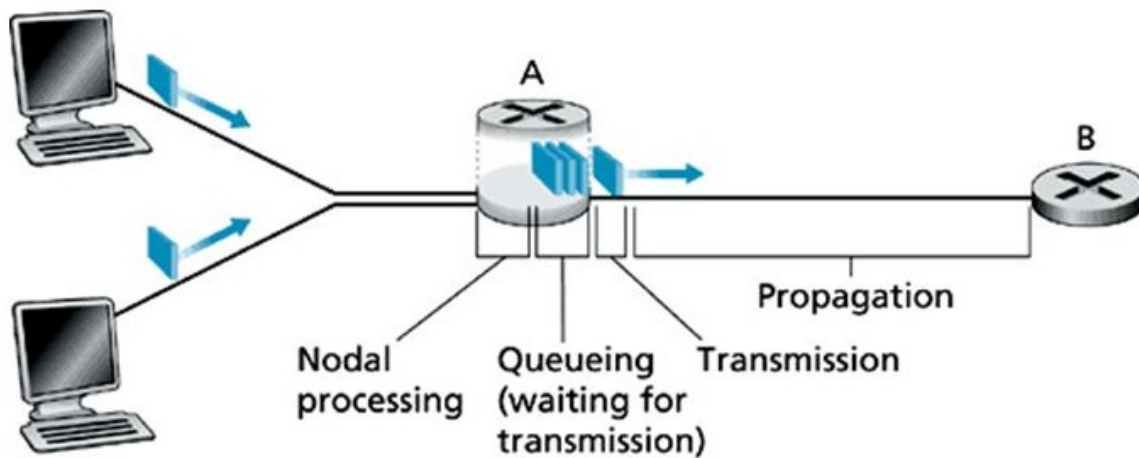
The internet is essentially a network of networks. Every internet service provider (ISP) is essentially linked together by Internet Exchange Points (IXP). Regional networks might also arise in order to connect access nets to ISPs, while content providers also can create their own networks allowing them to pass data to the internet while bypassing regional ISPs or even international ISPs. To make things clear:



- Tier 1: Commercial ISPs (Starhub, AT&T etc.), Content provider Networks (Google). Essentially the largest global ISPs, as they represent national and international coverage.
- Tier 2: IXPs. I.e. Connectors between the Tier 1 ISPs.
- Tier 3: Regional ISPs: Local internet service providers.
- Tier 4: Local Access Networks (your wires)

2 Network Performance

2.1 Sources of delay



2.1.1 Transmission delay

Transmission delay is the **amount of time it takes for a packet to be sent to the router's outward link**, or back.

2.1.2 Propagation delay

Propagation delay is the **total time the packet spends in transit**. It is influenced by distance from the target destination (or next where the next hop is) and hardware along the link.

2.1.3 Processing delay

Processing delay occurs as a result of **devices processing/examining the packets**. This is unavoidable and can only be reduced by getting better processing components on those devices.

2.1.4 Queuing delay

Queuing delay is the **waiting time for a packet to get to the front of the queue** for the output link. Depending on current traffic and the congestion level of the router, the queuing delay might be very high.

Calculating queuing delay R : Link bandwidth (KB/s)

L : Packet Length (bits)

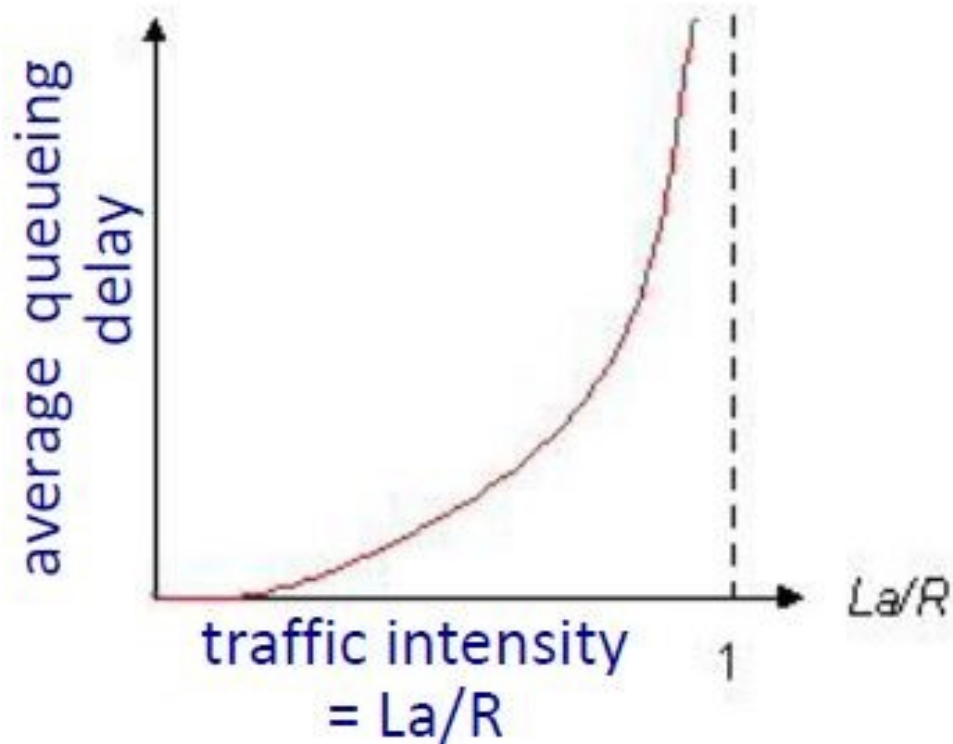
A : Average packet arrival rate (packets/s)

LA/R is the average link utilization and ranges from $[0,1]$

$LA/R \rightarrow 0$, there is a small average queue delay.

$LA/R \rightarrow 1$: There is a large average queue delay

$LA/R \gg 1$: There is more work arriving to the server, skewing the calculations.



The queuing delay is a convex increasing function. This means overloading the router will drastically increase queuing delay.

2.2 Traceroute, Ping, Internet Control Message Protocol

2.2.1 Ping

Ping is a network administration utility tool used to **test the reachability** of a host on an Internet Protocol (IP) Network. Ping uses the ICMP echo request/reply to messages to check for errors, packet loss, and the maximum and mean roundtrip times (RTT).

```

C:\Users\User>tracert google.com

Tracing route to google.com [172.217.26.78]
over a maximum of 30 hops:

  1      2 ms      2 ms      4 ms  10.12.0.1
  2      3 ms      2 ms      1 ms  172.16.1.106
  3     16 ms      2 ms      2 ms  172.16.1.210
  4      5 ms      2 ms      3 ms  103.24.77.1
  5     14 ms     10 ms      9 ms  203.116.245.177
  6      6 ms      5 ms      3 ms  203.118.12.17
  7      9 ms      6 ms     30 ms  203.118.2.30
  8     13 ms      9 ms      8 ms  203.118.12.10
  9     36 ms     99 ms      9 ms  108.170.240.225
 10      6 ms      7 ms      6 ms  108.170.240.161
 11      5 ms      3 ms      4 ms  72.14.234.177
 12      4 ms      3 ms      5 ms  sin10s02-in-f14.1e100.net [172.217.26.78]

Trace complete.

```

Example of a traceroute (tracert) run on a windows OS.

2.2.2 Traceroute (tracert on Windows)

Traceroute is a network diagnostic tool which **displays the path that a packet takes** across an Internet Protocol Network. It also records transit delays of packets. The round-trip times of the packets is received from each successive host in the route and the sum of the mean times in each hop is used to measure the complete time spent.

```

C:\Users\User>ping google.com

Pinging google.com [172.217.160.14] with 32 bytes of data:
Reply from 172.217.160.14: bytes=32 time=8ms TTL=51
Reply from 172.217.160.14: bytes=32 time=22ms TTL=51
Reply from 172.217.160.14: bytes=32 time=7ms TTL=51
Reply from 172.217.160.14: bytes=32 time=20ms TTL=51

Ping statistics for 172.217.160.14:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 7ms, Maximum = 22ms, Average = 14ms

```

Example of a ping command run on a windows OS.

2.3 Packet Loss and Handling

The queue within a router has a finite capacity. **A packet is dropped upon arriving to a full queue, resulting in loss.** Loss can also occur as a result of the link between devices. Loss is

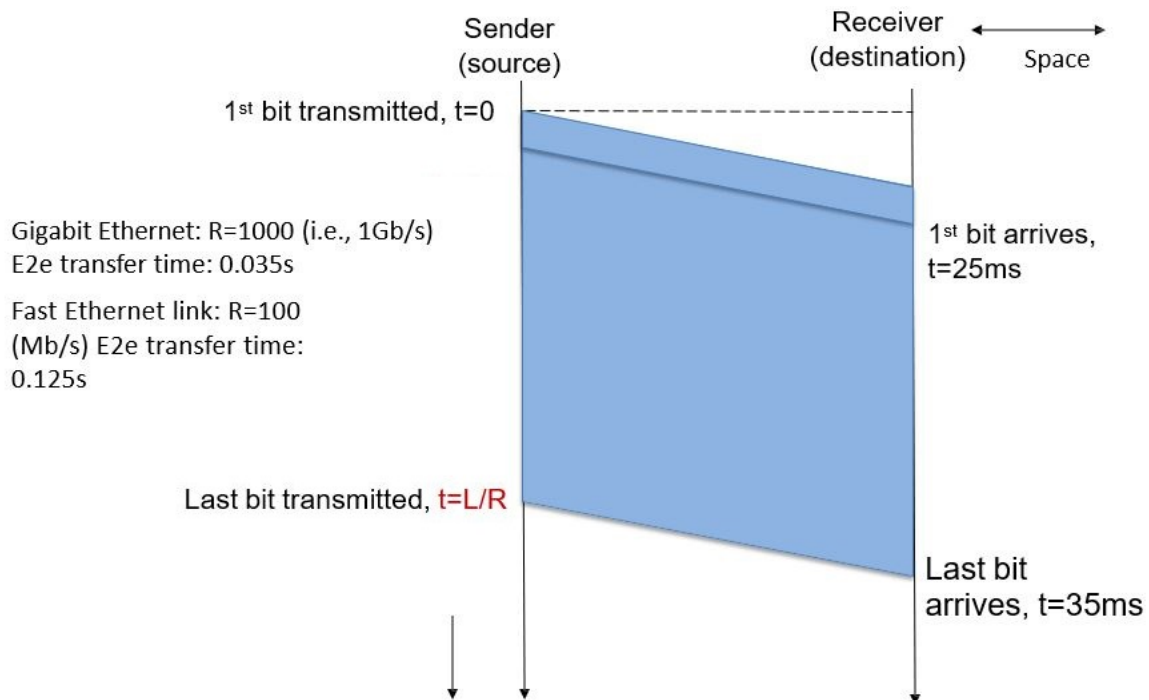
ultimately random, and different types of links and congestion situations affect the probability for each link/router. Packet loss is handled by re-transmission from the source end system (TCP), or not at all (UDP). In your calculation of total loss rate, it is important to remember that every pipe works with a different percentage of the packet, i.e., If Pipe 1 loses 5% of the packet, then Pipe 2, which follows after will only have 95% of the packets to work with. Therefore, if Pipe 2 had a loss rate of 10%, it would end up losing 9.5% of the total packets that were transmitted from your original machine. So the receiving end after pipe 2 would have $0.95 * 0.9 = 85.5\%$ of the packet. The loss rate can interact with the delay. Depending on whether re-transmission takes place, you can reduce the loss rate, but increase the delay, or do the exact opposite.

2.4 Throughput

Throughput (usually KB/s) is the rate at which bits are transferred between senders and receivers. It is important to recognize how throughput is organized throughout a system to identify bottlenecks and optimal paths, as links with lower throughput rate will act as an artificial limit to connection speed.

2.4.1 Space Time Diagram

Just for you to know as you may be required to draw it. No Google won't let you find this easily. Again, L is the **packet length (bits)**, and R is the **link bandwidth (bits/s)**.



3 Network Security

General goals in security:

1. confidentiality: Limited to intended recipient

2. authentication: Verification of identities
3. message integrity: Ensure message is not altered without detection
4. access and availability: accessible and available to all users

Possible attacks by intruders

1. eavesdrop on messages sent
2. insertion of messages into connection
3. impersonation of either party (IP Spoofing)
4. hijacking ongoing connection, taking place of a party
5. denial of service (DDOS, resource overloading)

3.1 Cryptography basics

Cryptography is the practice or study of techniques for secure communication. It aims to fulfill the goals of security above while preventing attacks. In network security, the basic idea is a function (mechanism) and a number (key) that allows for the obfuscation of information as required. The obfuscated information can then be sent over multiple networks before reaching its intended destination and decrypted. The system relies on the fact that knowledge of the system itself is insufficient to decrypt messages. One also needs to know a unique "key". There are two methods of doing this.

3.1.1 One-way Functions

A one way function is also known as a trapdoor function. In order to utilise this kind of method, two numbers are required for the trapdoor function to operate. The first number is used in encryption, but cannot be used for decryption. The second number is the exact opposite, designed only for use in decryption.

3.1.2 Symmetric Key

A Symmetric key is based on the idea of a typical mechanical lock. The same key is used both in decryption and encryption.

3.2 Symmetric Key Encryption

Symmetric Key encryption involves the following:

Where

m = message

c = Ciphertext

$KS()$ = Encryption/Decryption function

$KS(m) = c$

$m = KS(c)$

Therefore, before sending confidential information, both parties have to encrypt the message using the encryption function, and then decrypt the message using the same function. However, this raises the problem of sharing the key to begin with.

3.2.1 Simple Encryption Schemes

A few examples of simple encryption schemes are:

1. ROT1 (Replace every letter with the letter that follows it)
Basically replace A with B, B with C and so on. Prevents prying eyes, but is easily decrypted.
2. Transposition ciphers (replace every letter with another agreed letter.)
More difficult to decrypt but still easily brute forced, especially with the introduction of computers. Of note is how the encryption key only maps from a set of 26 letters to 26 letters. This means that knowledge of a word such as "POP" will result in an easier time decrypting the message via brute force.
3. DES (Data Encryption Standard) DES is known as a block cipher. It takes a fixed-length string and transforms it through a series of complicated operations into ciphertext of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key is composed of 64 bits, 8 of which are used solely for

checking parity. Hence the effective key length is 56 bits. It isn't used widely anymore due to its key length being too short.

3.2.2 Advanced Encryption Schemes

Advanced Encryption Standard

The advanced encryption standard has replaced DES since 2001, processing data in blocks of 16 bytes. Most implementations will chop blocks into sizes of 16 bytes for AES to process. If the block is not sufficient in length to reach 16 bytes, it is padded depending on the chosen padding scheme. Otherwise, a block of 16 bytes is added to the end of the entire cipher text.

Public Key Cryptography (RSA)

RSA is a system which utilises one-way functions. By using two prime numbers and an auxiliary value, one can create two keys, one of which is known as the public key, and the other, the private key. The public key can be sent to any party on the internet who wishes to send the owner of the keys anything confidential. The system works on the premise that given one of the keys, it should be very difficult for anyone to compute the other key. Some example uses:

- Basic Message sending
 1. One of the keys is used to encrypt a message, the ciphertext is sent to the intended recipient.
 2. The intended recipient receives the message, and uses the other key to decrypt the message, successfully receiving it.

Of note is that this method is used mainly by other recipients who wish to send private items over to the key owner. They will use the public key to encrypt the message, while the owner uses a private key.

- Authentication
 1. In order to authenticate the other user, the server can send an encrypted message using the public key to the user. The message is normally an arbitrary number, known as a nonce.
 2. The user receives the message, and uses the other key to decrypt the message, successfully receiving it, and echoes the encrypted message back to the server, proving that he is indeed the owner of the private key/public key key set that was used.

3.3 Creating an RSA Key pair

Just some basic math in case they need it.

1. choose two large prime numbers
2. compute a number $N = p \cdot q$, and another number $k = (p-1) \cdot (q-1)$
3. choose e (with e lesser than n) that has no common factors with k .
i.e. relatively prime numbers
4. choose one more number, C , such that $(e \cdot C) - 1$ is exactly divisible by k
5. the public key is hence (N, e) , and private key, (N, C)

3.4 RSA Encryption and Decryption

Given the public and private keys,

$$EncryptedMessage = m^e \bmod N$$

$$DecryptedMessage = (EncryptedMessage)^C \bmod N$$

Of note:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

use public key first,
followed by
private key

use private key
first, followed by
public key

result is the same!

3.5 RSA and AES used in tandem

RSA is relatively secure, but is extremely slow compared to AES. This is especially due to the computational cost of exponentiation. Therefore, RSA is normally first used for verification purposes, then an AES symmetric key is generated and sent over, allowing for faster and secure communication between both computers.

3.6 Non-repudiation, and digital signatures

Digital signatures serve the same function as signatures we make when cashing in our non-existent cheques. Essentially, the signatures are done by encrypting the message with the person's private key. The sender who signs a document establishes that he is the sole creator of the message. A simple example is using RSA encryption to "sign" a message. Because the public key can be used to verify that the message is decryptable, we can verify the owner of the message is the owner of the public key.

3.7 Message digests

As it is computationally expensive to public-key-encrypt long messages, hash functions are used to get a fixed, easy to compute digital fingerprint. Collisions are possible in the hash function, but are relatively rare. The point of the message digest is to allow for an easily identifiable fingerprint,

allowing for checking if the message was tampered with in transit. Examples of hash functions are SHA-1 or MD5.

3.8 Certification Authorities

Given that everyone can generate a certificate, there is nothing stopping anyone from simply stating that they are someone else, while proving that they are owners of the certificate sent over. As a result, certifying authorities serve to verify that public keys are bound to particular entity. Certifying authorities can be government backed or well known service provider.

4 Internet Naming and Addressing

4.1 Further on the DNS: Domain Name System

The Domain System is a distributed database, implemented in hierarchy of many name servers. Hosts and name servers communicate to resolve names and IP addresses.

4.1.1 DNS Services

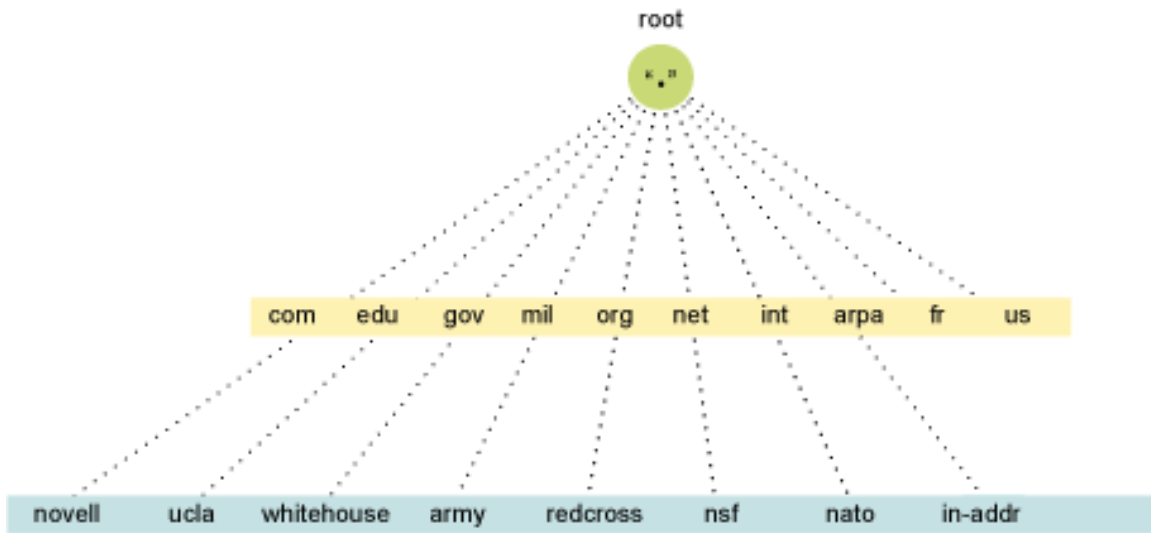
- hostname to IP address translation
- host aliasing - canonical, alias names
- mail serving aliasing
- load distribution (replicated webservers can be used, meaning many IP addresses correspond to one name)

4.1.2 reasons against DNS centralisation

- Single failure point
- No way to effectively scale the structure
- Traffic Volume (bottleneck)
- distant centralised database can result in increased latency
- maintenance is difficult

4.1.3 DNS-(UDP type connection)

Domain Name System is a UDP connection. DNS servers (since they use UDP) don't have keep alive connections, and requests are small and fit well within UDP segments. The TCP/IP protocol uses the IP address of a machine, allowing for unique identification of the connection of a host to the internet. Using a hierarchy of name servers, we are able to resolve Internet host names into the correct IP addresses by issuing a DNS query to the right name server. DNS also allows for mapping of the names to the addresses, allowing humans to navigate the internet using URLs. The DNS hierarchy is structured like this:



- Root.
- Generic top-level domains (gTLDs), which defines registered hosts according to their generic behaviour, e.g. {com, edu, gov, mil, net, org}. Also includes country-code top-level domains (ccTLDs), which define registered hosts according to their country, using 2 character country abbreviations instead of 3 like the generic domain, e.g. {us, sg, ai}.
- The inverse domain, which is used to map an address to a name.

4.2 Basic DNS mechanism

1. Clients requesting web pages query their local DNS server for the appropriate IP address.
2. If the call was **recursive** and the server does not have the appropriate answer, the server queries the root DNS servers to find an appropriate DNS server, and continuously makes calls until it manages to obtain the mapping for the requested address. At which point the server then returns the answer to the computer that requested it.
3. If the call was **iterative**, the server returns that it has no answer, and the local DNS server then queries the root DNS server, who further directs them to the next correct DNS server in the hierarchy to get the appropriate IP address for the website. (Root servers do not offer recursive call support!)
4. Upon obtaining the correct IP address, the response is returned to the original requester.

Of Note is that recursive queries are *are not always supported by the server*. This means that it is up to the DNS server to decide if it wishes to pursue the recursive query. This is because recursive queries are burdensome on name servers. If not, the server will only return a reference to another server that the local DNS server/computer can check with to answer the query.

4.3 Local DNS Servers

- Caches information for local clients requests after retrieving the appropriate answer from an authoritative name server.
- Speeds up queries by preventing repeat queries from being requested from the server.

- Past query results are stored for a limited amount of time, known as the TTL (Time to live).
- Of note is that cached entries might be out of date. So TTL is important in ensuring a balance between updating cached answers and maintaining efficiency.

4.4 Authoritative servers

- Organization's own DNS server(s), providing authoritative host name to IP for organisation's named hosts.
- Is generally maintained by either the organisation or the service provider.

4.5 DNS Server response format

DNS responses come in a predetermined format with several types of information, depending on the type of information required about a domain.

4.5.1 Format field

DNS Servers will reply to queries in a predetermined format.

- Domain Name: The domain name for which the query was sent
- Specifies the type of data included in the record
- Class: Specifies the data's class. This will be elaborated below
- Time to live the number of seconds this record will still be cached. It will be deleted upon expiry
- Data Length: Specifies the count of octets in the resource data field
- Resource Data: Contains the results of the binding data

This format is repeated for all 3 sections of the response. The answer section, Authoritative Name servers section, and the Additional Records Sections.

4.5.2 Field Types

1. A: this refers to the host address
2. CNAME: Canonical Name (alias) refers to an alias for eg. www
3. HINFO: Name of the CPU and Operating System
4. MINFO: Information on a mailing list or mailbox.
5. MX: 16-bit preference and name of the host that acts as a mail exchange server for a domain
6. NS: The authoritative name server for the domain, and its hostname
7. PTR: Symbolic link for a domain. Example: net.firewall.cx points to www.firewall.cx
8. SOA: Multiple fields that specify which parts of the naming hierarchy a server implements
9. TXT: just text. an uninterpreted string of ASCII.5b

4.5.3 DNS Record insertion

Creation of a DNS record requires a few steps.

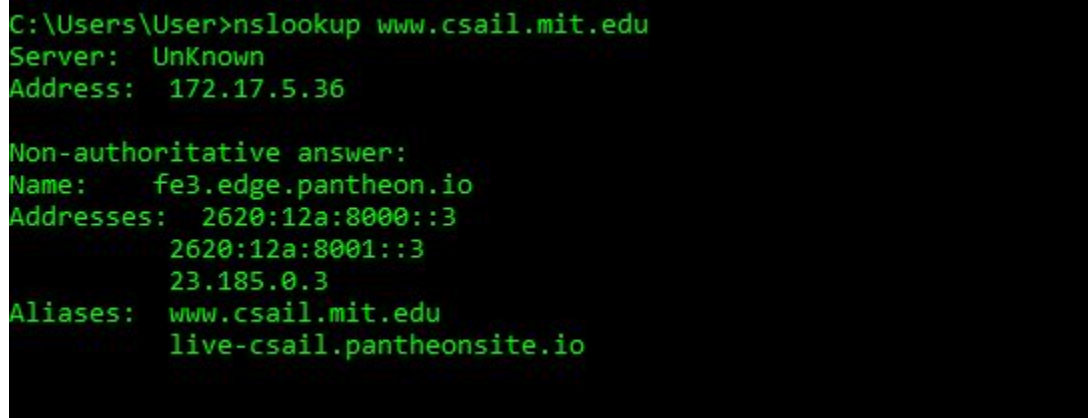
- Registration of website name/address at a DNS registrar. (need IP address of authoritative name servers too [primary and secondary]) The DNS registrar then inserts two resource records into the appropriate TLD (Top level domain) server. The first points to your Name server, and the second is your host address.
- Creation of authoritative server type A record and type MX record for the website.

4.5.4 DNS Attacks

- DDOS Attacks
By Bombarding the chosen servers with requests and traffic, prevent the proper usage of the DNS server by machines.
Not effective on root servers (It's been tried), but relatively effective on TLD/local DNS servers.
- Redirect Attacks
Man in the middle attacks (by intercepting queries)
DNS Poisoning (send bogus replies to the DNS servers, caching the wrong addresses)
- DNS DDOS exploitation
Given that SOME but not all DNS servers allow for recursive queries, by getting a lot of DNS servers to ask the same question from various servers, you can potentially DDOS something, even if they have traffic filtering/blacklisting. The issue being that amplification of such an attack is risky and due to the scale, easier to detect.

4.5.5 Reading DNS lookups via NSlookup

In this portion, you'll look at a nslookup of `www.csail.mit.edu`.



```
C:\Users\User>nslookup www.csail.mit.edu
Server:  UnKnown
Address:  172.17.5.36

Non-authoritative answer:
Name:     fe3.edge.pantheon.io
Addresses: 2620:12a:8000::3
           2620:12a:8001::3
           23.185.0.3
Aliases:  www.csail.mit.edu
           live-csail.pantheonsite.io
```

1. The request is for a DNS lookup of `www.csail.mit.edu` The address is `23.185.0.3`
Of note is the previous 2 addresses, which are `2620:12a:8000::3` and `2620:12a:8001::3`
Those are actually IPV6 addresses, which is a different protocol from what you're currently learning. Nevertheless it's an address too.
2. The Address provider, is listed at the top. While it says `Server: UnKnown`, the IP address of your DNS server is clearly written there, which is `172.17.5.36`

3. Noteworthy is the answer is non-authoritative, because the answer is from your DNS server, not directly from the organisation's DNS server. Your DNS server is simply echoing it to you while caching it.

4.5.6 Mail server interaction

If one wants to send a mail to someone say, bob@mit.edu, which IP address should he send mail to? The answer is really to get a MX reply from the DNS of that organisation. To refresh your memory, the MX reply from a DNS server tells you two things: **16-bit preference and name of the host that acts as a mail exchange server for a domain**. This implies that we should send an inquiry to the mit DNS server for their mail servers.

However, to do this we cannot use nslookup. This is because nslookup is to look up a name server. Not to inquire something from the name server. To do this, we use a different tool, known as **dig**.

4.5.6.1 Dig tool

The dig tool was developed for a different reason from nslookup. It is designed to send inquiries to DNS servers. In this case, we use it in the following fashion:

dig address type

this allows us to send a query of the MX type. to obtain the required mail servers. An example command is shown below:

```
root@DESKTOP-DU2PR80:~# dig mit.edu MX

; <<>> DiG 9.10.3-P4-Ubuntu <<>> mit.edu MX
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57579
;; flags: qr rd ra; QUERY: 1, ANSWER: 8, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:;; udp: 512
;; QUESTION SECTION:
;mit.edu.                IN      MX

;; ANSWER SECTION:
mit.edu.                  120     IN      MX      100 dmz-mailsec-scanner-1.mit.edu.
mit.edu.                  120     IN      MX      100 dmz-mailsec-scanner-3.mit.edu.
mit.edu.                  120     IN      MX      100 dmz-mailsec-scanner-8.mit.edu.
mit.edu.                  120     IN      MX      100 dmz-mailsec-scanner-2.mit.edu.
mit.edu.                  120     IN      MX      100 dmz-mailsec-scanner-4.mit.edu.
mit.edu.                  120     IN      MX      100 dmz-mailsec-scanner-5.mit.edu.
mit.edu.                  120     IN      MX      100 dmz-mailsec-scanner-7.mit.edu.
mit.edu.                  120     IN      MX      100 dmz-mailsec-scanner-6.mit.edu.

;; Query time: 133 msec
;; SERVER: 172.17.5.36#53(172.17.5.36)
;; WHEN: Mon Apr 23 00:33:26 DST 2018
;; MSG SIZE rcvd: 340
```

As seen, there are 8 mail servers that are available. This is due to the need to split the load between several servers and to maintain system integrity.

However, this does not include any authoritative answers.

So the difficulty in actually obtaining a authoritative answer is sometimes due to the intermediate DNS server/local DNS server caching the query, then echoing it back to you. As a result, we cannot always obtain an authoritative answer. Should you wish to **FORCE** an authoritative answer, the first thing you can do is use:

dig site you wish to look up query type +trace.

the +trace command forces the log of recursive commands to also be echoed to you, allowing you to find a possible authoritative name server. One such example is shown below:

```
root@DESKTOP-DU2PR80:~# dig mit.edu MX +trace
```

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> mit.edu MX +trace
```

```
;; global options: +cmd
```

```
.          144803 IN      NS      j.root-servers.net.
.          144803 IN      NS      g.root-servers.net.
.          144803 IN      NS      d.root-servers.net.
.          144803 IN      NS      i.root-servers.net.
.          144803 IN      NS      k.root-servers.net.
.          144803 IN      NS      e.root-servers.net.
.          144803 IN      NS      c.root-servers.net.
.          144803 IN      NS      h.root-servers.net.
.          144803 IN      NS      l.root-servers.net.
.          144803 IN      NS      b.root-servers.net.
.          144803 IN      NS      m.root-servers.net.
.          144803 IN      NS      a.root-servers.net.
.          144803 IN      NS      f.root-servers.net.
```

```
;; Received 379 bytes from 172.17.5.36#53(172.17.5.36) in 9 ms
```

```
edu.       172800 IN      NS      a.edu-servers.net.
edu.       172800 IN      NS      c.edu-servers.net.
edu.       172800 IN      NS      d.edu-servers.net.
edu.       172800 IN      NS      f.edu-servers.net.
edu.       172800 IN      NS      g.edu-servers.net.
edu.       172800 IN      NS      l.edu-servers.net.
edu.       86400  IN      DS      28065 8 2 4172496CDE85534E51129040355BD04B1FCFEBAE996DFDD
edu.       86400  IN      RRSIG  DS 8 1 86400 20180505050000 20180422040000 39570 . DIImVVO
g1CjSh1u2JENq57C/+4JB1WpN5yeb/Ac EklJ6bnyqlQQ+Lsu0ecItXjTKTdTkR0KrK+9K7Y19i15T1uJMexq3Vix yHJ4epghv8jQVVi
;; Received 606 bytes from 199.7.83.42#53(l.root-servers.net) in 198 ms
```

```
mit.edu.   172800 IN      NS      usw2.akam.net.
mit.edu.   172800 IN      NS      asia1.akam.net.
mit.edu.   172800 IN      NS      asia2.akam.net.
mit.edu.   172800 IN      NS      use2.akam.net.
mit.edu.   172800 IN      NS      ns1-37.akam.net.
mit.edu.   172800 IN      NS      ns1-173.akam.net.
mit.edu.   172800 IN      NS      eur5.akam.net.
mit.edu.   172800 IN      NS      use5.akam.net.
```

```
9DHS4EP5G85PF9NUFK06HEK0048QGK77.edu. 86400 IN NSEC3 1 1 0 - 9UJ88SQBJN4LMMPPUD327MGPLKU7ANV0 NS SOA RRSIG
9DHS4EP5G85PF9NUFK06HEK0048QGK77.edu. 86400 IN RRSIG NSEC3 8 2 86400 20180429125940 20180422114940 63863
xfeeCEBtzCKgiPn20TOvZovAS/mr/vWoV+nJfGI/pHq 5p8=
H1SPUQIV7KAEGO7MNVFS0014TGESK44N.edu. 86400 IN NSEC3 1 1 0 - I4667HA7DROBISP0J03FLRA51T795C7K NS DS RRSIG
H1SPUQIV7KAEGO7MNVFS0014TGESK44N.edu. 86400 IN RRSIG NSEC3 8 2 86400 20180429161826 20180422150826 63863
zIT0Wp/kv6YHMPv8ksjTvDAUNxrRdfoaUnOesF5wAiF 9+k=
```

```
;; Received 900 bytes from 192.35.51.30#53(f.edu-servers.net) in 91 ms
```

```
mit.edu.   120     IN      MX      100 dmz-mailsec-scanner-4.mit.edu.
mit.edu.   120     IN      MX      100 dmz-mailsec-scanner-5.mit.edu.
mit.edu.   120     IN      MX      100 dmz-mailsec-scanner-1.mit.edu.
mit.edu.   120     IN      MX      100 dmz-mailsec-scanner-8.mit.edu.
mit.edu.   120     IN      MX      100 dmz-mailsec-scanner-2.mit.edu.
mit.edu.   120     IN      MX      100 dmz-mailsec-scanner-6.mit.edu.
mit.edu.   120     IN      MX      100 dmz-mailsec-scanner-7.mit.edu.
mit.edu.   120     IN      MX      100 dmz-mailsec-scanner-3.mit.edu.
```

```
;; Received 468 bytes from 95.100.175.64#53(asia1.akam.net) in 243 ms
```


Then looking at the final response, which states which DNS server it came from (in this case, asia1.akam.net), we are able to find a server that can possibly give authoritative answers.

After this, simply run another dig command:

dig website query @Name server address

this will enable you forcibly obtain an authoritative answer straight from the authoritative DNS server. Of note is that @name server allows you to basically query any dns server. An example output is below.

```
root@DESKTOP-DU2PR80:~# dig mit.edu MX {asia1.akam.net}

; <<>> DiG 9.10.3-P4-Ubuntu <<>> mit.edu MX {asia1.akam.net}
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 716
;; flags: qr rd ra; QUERY: 1, ANSWER: 8, AUTHORITY: 0, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;mit.edu.                IN      MX

;; ANSWER SECTION:
mit.edu.                120     IN      MX      100 dmz-mailsec-scanner-8.mit.edu.
mit.edu.                120     IN      MX      100 dmz-mailsec-scanner-3.mit.edu.
mit.edu.                120     IN      MX      100 dmz-mailsec-scanner-2.mit.edu.
mit.edu.                120     IN      MX      100 dmz-mailsec-scanner-1.mit.edu.
mit.edu.                120     IN      MX      100 dmz-mailsec-scanner-6.mit.edu.
mit.edu.                120     IN      MX      100 dmz-mailsec-scanner-5.mit.edu.
mit.edu.                120     IN      MX      100 dmz-mailsec-scanner-4.mit.edu.
mit.edu.                120     IN      MX      100 dmz-mailsec-scanner-7.mit.edu.

;; ADDITIONAL SECTION:
dmz-mailsec-scanner-5.mit.edu. 1133 IN  A      18.7.68.34

;; Query time: 378 msec
;; SERVER: 172.17.5.36#53(172.17.5.36)
;; WHEN: Mon Apr 23 00:54:23 DST 2018
;; MSG SIZE rcvd: 356

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 49457
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;{asia1.akam.net}.      IN      A

;; AUTHORITY SECTION:
.                900     IN      SOA     a.root-servers.net. nstld.verisign-grs.com. 2018042200 1800 900 604800 86400

;; Query time: 1013 msec
;; SERVER: 172.17.5.36#53(172.17.5.36)
;; WHEN: Mon Apr 23 00:54:24 DST 2018
;; MSG SIZE rcvd: 120
```

Dig Answer Flag Examples:

- QR specifies whether this message is a query (0), or a response (1)
- OPCODE A four bit field, only valid values: 0,1,2
- AA Authoritative Answer

- TC TrunCation (truncated due to length greater than that permitted on the transmission channel)
- RD Recursion Desired
- RA Recursion Available
- Z Reserved for future use. Must be zero

4.5.7 whois command

The whois command is a protocol that is used for querying databases that store the registered users or assignees of an internet resource. This includes domain name, an IP address or other relevant information.

If you need to search for someone's IP, whois command can be used, if you have their namespace address.

4.5.8 DNS Issues

There is a cost on the indirection of DNS in general. Due to the spreading of servers, while we obtain general scalability portability and the ability to perform many-to-one mappings (aliasing), we compromise the security of the servers due to the possibility of poisoning/intercepted mappings.

4.6 Client-server and the Web: The application layer

4.6.1 Sockets

While an IP address points you to the machine, but does not tell you the process you should be looking for.

Due to process ID being volatile and not meant for external usage to point to the correct process, we need another pointer, the socket.

Therefore, packet headers contain port numbers in order to tell the other computer which process it should look for.

4.6.2 Multiplexing/Demultiplexing

Multiplexing is the process of combining several analog or digital signals to be sent via one signal. Demultiplexing is the opposite process, where we split the multiplexed signal into several different signals again. It requires a multiplexer on one side and a demultiplexer on the receiving end. There are several ways to achieve this, and two of the covered ways are frequency division multiplexing where the spectrum of each input signal is shifted to different bands, and time division multiplexing, where signals take turns.

4.6.2.1 Connection-oriented (TCP) demultiplexing Essentially, in order to split the signals among one TCP connection, the packets sent contain their source and destination IP address and port numbers, allowing for communication between two computer processes.

4.6.2.2 TCP Socket Programming

- The server has to be already running before the client contacts it.
- The server must have a socket that welcomes client contact.
- The client contacts by creating a TCP socket with the IP address and port number of the server process.
Socket creation will establish a connection to the server.
- Servers will create a new socket for the server process to communicate with the client upon contact. Keep in mind that the server still has a separate server socket.

- The Source port numbers are used to distinguish the clients.

4.6.2.3 UDP Socket programming UDP has no "connection between the client and server. There is no handshake before sending data. The sender simply attaches the IP destination and port number. Of note is that it may be lost or received out of order. It provides unreliable transfer of groups of bytes.

4.6.3 Web/HTTP Pages

HTTP stands for hypertext transfer protocol.

HTTP itself is stateless and maintains no information about past client requests, unless cookies are used.

The HTTP page itself that is transferred might contain further references to other servers to obtain more resources that the current services might not have.

Currently there are 2 HTTP connection types. There is HTTP/1.0, non-persistent HTTP, and HTTP/1.1, persistent HTTP.

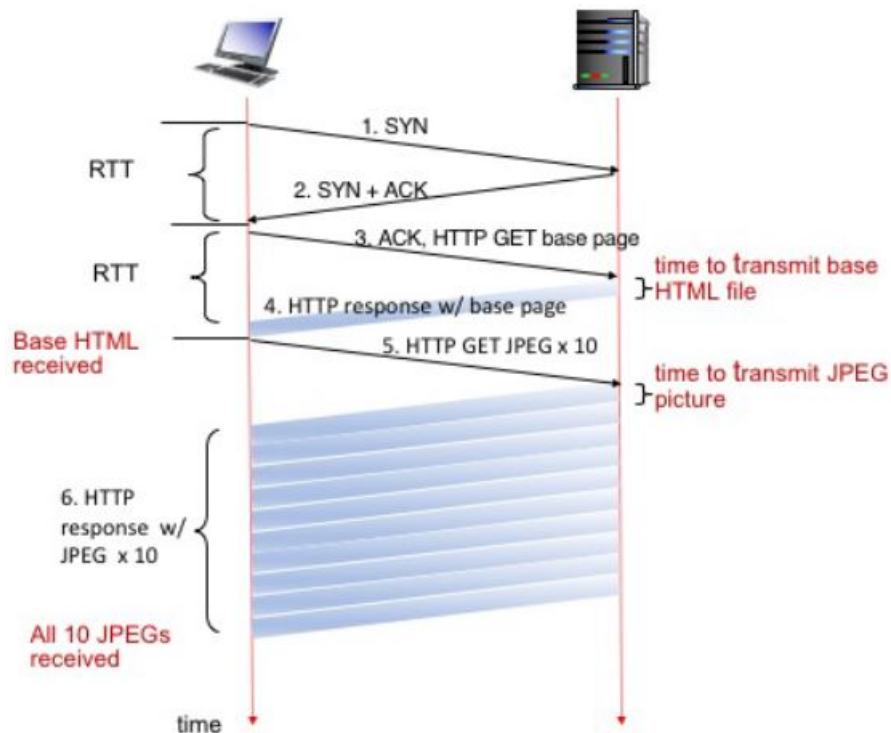
- non persistent HTTP, HTTP 1.0 allows only one object sent over TCP connections
- persistent HTTP, HTTP 1.1, the server leaves the connection open after sending the response, allowing for multiple objects to be sent over a single TCP connection.

The reason for using HTTP 1.1 connections as opposed to that of HTTP 1.0 is due to the number of round trip time packets take, the number of connections, and the allowance for instant requesting for objects.

When using non-persistent HTTP, browsers will often parallel TCP connections to fetch the referenced objects.

4.6.4 Spacetime diagram of sending 10 jpegs using persistent connection

Use the following diagram as a guide on how to draw what they call a space-time diagram in this course. This image is that of a HTTP request scenario assuming persistent connection(HTTP 1.1)



and pipelining.

4.6.5 HTTP Request message format

Depending on the type of request that is sent over, the actual contents and structure of the message will vary somewhat.

The general structure of the HTTP message is as follows:

- **Start line**
The message start line contains the Request and the Status Line. The request line is the request method that the client is using, and the status line is only included in the message if the sender was from a server.
- **Header Fields**
The header fields will contain one of two things, depending if it is a message response or a request.
 - If it was a request:
It will contain fields that allow the client to pass additional information about the request and the client itself to the server. Example fields include: Accepted Charset, Accepted Encoding, Accepted Language, Expect, Authorization, From, Host, If-Match, If-Modified-Since, Range, Referer.
 - If it was a response:
It will contain response headers, such as: Accept-ranges, Age, Location, Server, WWW-authenticate.
- **The final part is the Message body.**
It is an optional part, and carries the request/response data that is required for the message.

Here are those HTTP requests that you'll have to know

4.6.5.1 HTTP 1.0 (non persistent) Request types:

1. GET
Request information from the server for retrieval.
2. POST
Request the server accept the fields of data stored inside the message body, possibly for processing or storage.
3. HEAD
Ask the server to leave requested object out of response

4.6.5.2 HTTP 1.1 (persistent) Request types:

1. GET
2. POST
3. HEAD
4. PUT
Upload the file in entity body to the path specified in the URL field.
5. DELETE
Deletes the file specified in the URL Field.

4.6.6 Telnet (TCP)

Telnet is a client-server protocol. Telnet enables you to open communications between a server and your computer (the client). As a result, you can also send HTTP POST and GET requests.

4.6.7 HTTPS

HTTPS Runs at port 443.

Uses (SSL) encrypted communications

4.6.8 HTTP Response status codes

The more well known codes you'll need to know are below:

1. 200 OK - Request succeeded, requested object is in the message
2. 301 Moved Permanently - Requested object is in a new location: and specified later in the message.
3. 400 Bad Request - Request message is not understood by the server
4. 404 Not found - the requested document is not found on this server
5. 505 HTTP Version not supported

4.6.9 Cookies

Cookies are arbitrary pieces of data, usually chosen and first sent by the web server, and stored on the client computer by the web browser. The browser then sends them back to the server with every request. They aid in maintaining state requests.

Cookies are composed of four components:

1. Domain
2. Path
3. Expires
4. Max-Age

There are also two additional components: Secure and HttpOnly. However, they are optional fields and their presence simply indicates that they should only be treated as such.

Cookies can be used for authorization, Shopping carts, Recommendations, User Session State.

4.6.10 Proxy Servers

Proxy Servers are similar to a DNS server. The only difference is that they act as both client and server. Upon requesting a web page from a computer, the proxy server will check its cache for the webpage. Upon finding a hit, it immediately returns the webpage. But if it misses, it queries the relevant DNS servers, gets the webpage, stores it, and returns a copy to the original requester. Depending on the hit rate of the cache, this can heavily reduce the load on the external network link. This is known as a conditional GET. - If modified since - then send me a new one. else send me my 304 not modified header.