CNT5106C Homework 1 Solutions

Maksim Levental

October 9, 2014

1. A flat network is one where all devices are on the same segment and in the same broadcast domain. This is accomplished by connecting all devices to several hubs or one small switch. The effect is that all edges can reach all other edges without transiting through intermediary bridges or routers. The advantages of a flat network topology is that it is adequate for up to a few hundred devices (hence cheaper than if many switches and routers were purchased), it's easy to implement, and easy to manage. Its disadvantage is that it's less fault tolerant (if the single switch fails the network becomes a completely disconnected topology/graph), poor security (all nodes accessible from all other nodes, easier to hack), congestion due to all traffic flowing through one switch, and concomittantly poor scalability because of congestion.

   A hierachical network is one where the network topology is divided into discrete layers, and each layer has a specific function. Typically there is a core layer of routers and switches optimized to forward traffic as quickly as possible, a distribution layer that defines network boundaries and broadcast domains and where MAC filtering is done, and finally the access layer where users actually access the network. The advantages are that this structure isolates subsets of the network making it more robust as a whole, more scalable through more efficient routing tables, and allows for specialization of particular subdomains of a network. The disadvantges are operating overhead (expensive equipment) and sub-optimal routing of packets.

2. The internet architecture administratively speaking is a loose hierarchy of administrative providers scoped by region. At the center there is a small number of well-connected large Tier-1 providers that provide national and international coverage. Then there are Tier-2 local providers that provide service to regions, and peer to each other to bypass Tier-1 providers. Finally there are the "last-hop" local ISPs that provide direct access to users. In terms of routing the internet is a layered protocol architecture. The advantages, from a routing point of view, are **1**. the structure directly reflects the relationships between edges and nodes, providing an adequate reference model for study and theory **2**. it's modular and hence robust to upgrades and changes in protocols and layers in the stack **3**. it's flexible enough to accommodate future innovations for things such as mobile network or other novel forms of connectivity. The disadvantages are **1**. overhead of headers being concatenated with all packets in order to facilitate commuting through all of the layers **2**. redundancy of some functionality, such as reliability/retransmission being implemented in the transport layer and link layer and routing being implemented in the network layer and link layer.

3. For circuit-switched networks the transmission time is a function of whether (TDM) time-domain multiplexing is used or (FDM) frequency-domain multiplexing is used. If TDM is used then the total transmission time is a function of the L_s link speed, T_{TDM} number of slots per second dictated by the TDM, C time delay to establish a circuit, and P propagation delay. L_s, T_{TDM}, C are all variable while P is a matter of speed of light in medium and hence fixed. Therefore in circuit switched networks the time to establish a circuit and propagation delay are major determinants of overall delay. As already mentioned propagation delay is a function of distance between edges ($100\mu s/20km$) and therefore in a sense dynamic. C, the time to establish a circuit is also dynamic in the sense that if the network is heavily loaded one might

have to wait a long time for an open-circuit.

For packet-switched networks in general the "nodal" delay (delay per node) is a function of d_{proc} processing delay (the time taken by the router to process the packet, error checking, determining output link), d_{queue} the delay due to being queued in the buffer of a node, d_{trans} the transmission delay (the actual amount of time it takes to push all of the bits into the pipe), d_{prop} the same propagation delay as for circuit-switched networks. Hence

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

and in total if $N = \{nodes\}$ and D is the total delay then

$$D = \sum_{n \in N} d_n$$

At each node d_{proc} is dynamic in the sense that it's a function of how fast the router is and how much loss due to error is incurred from the last hop, d_{queue} is definitely dynamic in the sense that it's longer when congestion is higher, d_{trans} is fixed as a function of the packet size, d_{prop} is only dynamic in the sense that nodes could be separated by different distances.

4. For circuit-switched networks, guarantees can be made about quality if it's taken for granted that you either have a circuit reserved or you can obtain a circuit reservation. Since reserved circuits have dedicated throughput there is no loss of quality for customers already being served when load is high, but customers attempting to obtain circuits during times of heavy load will be denied entry into the network, and in that sense quality degrades.

   For packet-switched networks it is the case that only statistical guarantees can be made about quality and that during times of heavy congestion quality will degrade.

5. The four design criteria of utmost importance during the formative years of the internet were scalability and economic access, robustness, reliability, and evolvability. To make the internet scalable and economic it was built using infrastructure that shared resources in order to reduce reservation time/need - the packet-switching model instead of the circuit-switching model. To make the internet more robust it was designed to reroute upon failure, use stateless connections (so that functionality didn't hinge on some resource which stored part of the state, being persistent). To make the internet more reliable lost packets can be retransmitted. Finally to make the internet evolvable all of the complexity was kept out of the core and pushed to the edges, so that drastic changes wouldn't necessitate changing core functionality.

   If I were redesigning the internet I would make security and privacy primary concerns. Security should be a concern since so much of the internet's functionality is being used for commerce and banking. Privacy concerns, in light of recent revelations about certain government organizations, are obviously important.

6. Because even if the physical layer is secured attackers can attack the application and network layers by exploiting vulnerabilities therein. For example securing the physical layer against something like passive optical taps or interception of wireless signals does nothing to prevent things like SQL injection, cross-site scripting, or PHP remote file inclusions on servers.

   Remote control of a computer, for example in a botnet, by the user downloading a trojan virus, cannot be prevented by use of encryption schemes. Furthermore once control of such a computer is gotten, by use of a trojan, encryption of packets sent from that computer is useless since the attacker can simply monitor exactly what the user is doing.

7. The first type of attack is a bandwidth flooding attack, where an attacker uses a botnet to send packets to a root server to overload it. To thwart such an attack root servers typically use

packet filters to block internet control message protocol (ICMP) messages. The second type of attack is a distributed denial of service attack on top-level domain servers, overwhelming them by requesting domain name resolution from many sources simultaneously. The effects of this are mitigated by using local caches, so that if in fact a TLD is overwhelmed, DNS resolution can still happen. The third type of attack is a man-in-the-middle attack, wherein DNS requests are intercepted in transit and then responded to by untrustworthy agents. A possible way to thwart this kind of attack would be to cryptographically sign all of DNS queries and responses but that would incur too much overhead. The fourth type of attack is using the DNS itself to attack a user, simulating many requests from a user so that many DNS servers will flood the user with responses. This is difficult to do and hence typically not actively planned against.

8. TCP is a connection-oriented protocol, which means a connection is established end-to-end and during that connection data is guaranteed to be delivered from the sender to the receiver. A connection is established using a 3-way handshake and this connection is persistent throughout the exchange. To guarantee delivery a combination of timers and ACKs (acknowledgements) is used to determine whether to retransmit. So for example a movie streaming application (Netflix) would use TCP since the order of data would be very important (frames in the movie). UDP (user datagram protocol) functions by sending only packets called datagrams with source and destination information (network participants, IP endpoints, DNS queries for both ends). No error mitigation or retransmission is provided but no latency is incurred either. The advantages of UDP over TCP are low latency, while the advantages of TCP over UDP is guaranteed delivery and guaranteed ordering for the “payload”. For example DNS operates over UDP because latency is of primary importance, not guaranteed transmission nor ordering, and similarly video chat applications like Skype use UDP because latency is of utmost importance.

9. An RFC is a “request for comments”, a publication put out by engineers and computer scientists at the Internet Engineering Task Force in order to solicit feedback on proposed changes to the internet. For example potential protocol implementations are distributed as RFCs and therefore contain the full specifications of those protocols. In the specifications of the protocols implementation details are coded as “MUST” for necessary for agreement with the protocol and “MUST NOT” for the converse, “SHOULD” for best practices but not necessary and similarly “SHOULD NOT”. They’re important for having a universal standard for the purposes of interoperability on the the internet.

10. A client-server architecture consists typically of a permanent “always-on” host designated the server, with a static IP address, which intermittently connected clients, with dynamic IP addresses, can access. Typically clients do not communicate with each other, while servers can in fact be server farms in order to distribute traffic and perform load-balancing. A peer-to-peer network is where there are no true servers acting as focal points of traffic and all peers communicate with each other, fetching data from other peers and sending data to other peers that request it. Peers are intermittently connected as mentioned before and have dynamic IP addresses making management of such a network difficult. The advantage of a client-server model for a network is that it’s easier to manage but its disadvantage is that it’s expensive to scale, requiring literally more servers. The advantage of peer-to-peer is that it scales very well since peers can fill requests for other peers (functioning as local micro-servers). This allows distribution of files to be much speedier since all peers that have chunks of a file can send it to a requesting user. One disadvantage of this of course is figuring out which users have chunks of a file.

11. The thin-waist of the internet is alludes to the fact that all of the diversity in technology is at the bottom layer (physical) and at the top (application), and there are few choices for protocol in the transport, network, and data link layers. The advantage of this phenomenon is that the most ubiquitous parts of the internet (most often used/universal to all services) are inter-operable among services since they’re all essentially running on one of two or three protocols. Furthermore having lots choice in the application layer and the physical layer enables

experimentation in those domains. The disadvantages are that aging protocols like TCP/IP are difficult to renovate/replace since so many applications are tailored to them.

12. The end-to-end principle is the idea that complexity should be minimal in the network and kept in the edges. Application specific functions should be kept in the end hosts rather than exist in the intermediary nodes. The reason being that implementing functionality in the network itself is much more difficult than in the edges. For example meeting reliability requirements by leveraging the network is much more difficult than expecting applications to retransmit, hence the error correcting (and congestion control) TCP is implemented on top of the connectionless IP. The principle has been violated in the implementation of DNS, a key piece of network functionality, that exists in the application layer.

13. Let $p = .1$ be the probability a user is active. Then the probability that at least 10 out of 35 users is active can be computed using the binomial distribution

$$P(X > 10) = \sum_{i=11}^{35} \binom{35}{i} \left(\frac{1}{10}\right)^i \left(\frac{9}{10}\right)^{35-i} = 0.000424298$$

14. Napster, Gnutella, Bittorent, and Skype are examples of hybrid peer-to-peer networks. They have centralized servers that keep track of users coming on and off line and assist in locating addresses of remote parties. But communication between parties happens directly - voice or data chunks are exchanged directly between users. The advantage of these networks is that they allow for higher throughput, in the cases of the file sharing networks, since users download from many servers simultaneously and are cost efficient for the service providers, in the case of skype, since data doesn't transit through Microsoft's servers. The disadvantages, in the cases of the file-sharing networks is that corralling users in order to make their collections available to the rest of the network is difficult. Bittorrent solves this problem using trackers and Gnutella solves this problem using peer group leaders. Napster had centralized servers, which led to suits being brought directly against it.

15. HTTP is a stateless protocol and yes it can (and has been made) stateful using cookies. The purpose is to store information about users on things like e-commerce sites in order to either enhance the user experience or monetize user data. Cookies store client info clientside that is then managed by the browser. A cookie stores a key in a backend database that keeps information about the user on the server's host. Cookies are set using HTTP response message headers and then queries are returned using regular HTTP response message headers. Then cookies are queried using HTTP request headers for purposes of indexing into the backend database.

16. A push architecture in peer-to-peer networks is the case when users are spontaneously sent data, rather than making requests for it. The users are passive in the sense that they idle while waiting for data to be sent to them. A pull architecture in peer-to-peer networks is the case when users request data directly from other nodes and actively discriminate amongst nodes, choosing the best (fastest) source. Push networks have to often reconfigure themselves when dealing with nodes entering and exiting the network. Pull networks make it possible for nodes with good throughput to be overwhelmed by other nodes.

    An alternative would be a hybrid of the two: a push-pull network. A node would pull until they could no longer and then they would idle waiting for a push from other node. This would resolve the problem of high-throughput nodes being overwhelming, but would do nothing for the reconfiguration problems.

17. A UDP server needs only, in theory, 1 socket to support n connections because the socket API allows one socket to receive from many endpoints and send to many endpoints. Since most often UDP implements very simple applications (request and reply) it's sufficient to take note of the source address and reply (no long-lived connection necessary). A TCP connection on

the other hand needs $n + 1$ sockets in order to maintain n connections: 1 for the handshake operation and 1 per connection, since TCP connection are long-lived.

18. A DHT is a distributed hash table and it's used in P2P networks so that information about the collection of data being stored in the P2P network is deterministically queriable. When new data is added to the network a hash is calculated deterministically and a message is sent out, and propagates, until the the user responsible for knowledge of the range that the hash key is in, receives it. Then when a query against the content stored in the network needs to be made, a requesting user hashes the same key and the query again propagates to the user responsible for the key range. It's advantage over peer leaders is scalability.

19. When peer 5 leaves peer 4 is alerted of this by the fact that its consistent pings on peer 5 are no longer answered. So it updates its successor to what used to be its second successor and then queries its successor for that peers successor. It then alerts peer 3, which updates its second successor to be peer 4's successor, peer 8. So peer 3's successor is peer 4 and second successor is peer 8.

20. It would query peer 15 for its successor, which is 1, then 1 for its successor, which is 2 and on until it queries for 5's successor which is 8. It then declares 8 its own successor, 10 to be its own second successor. Finally 5 updates its successor to be 6, and its second successor to be 8.