1. FTP can be implemented using either TCP or UDP. Discuss the advantages and disadvantages of each protocol when implementing FTP. Which protocol does FTP use?

To answer the second part first: FTP uses TCP because the spec says it should (File Transfer Functions, sec. 4, and Connection Establishment, sec. 8, in [File Transfer Protocol (FTP), RFC 959](http://www.ietf.org/rfc/rfc959.txt)).

The answers, approximately verbatim, and in no particular order:

* 1. The advantages of TCP when implementing FTP are,
     + Since the TCP is a connection-oriented protocol reliable data transformation takes place.1
     + TCP provides interoperability when implementing FTP.2
     + It operates independently3 and enables internetworking4
     + While transmitting if the dta is lost it is resent to the destination until it receives an acknowledgement
     + There is no duplication of data in TCP

The disadvantages of TCP when implementint FTP are,

* + - It is costly to set up a TCP connection
    - The time consumption for this protocol is more5
    - It is intricate to set up and manage6

The advantages of UDP wehen implementing FTP are,

* + - Since UDP is a connectionless oriented it has less frame work7
    - It uses less processing time when compared to TCP8
    - It has less latency9 and is application flexible10

The disadvantages of UDP when implementing FTP are,

* + - There is no reliability whether the data is tranmitted or not11
    - There is duplication of data in UDP
    - There is no retranmission of data if it is lost
    - There is lack of communication between the server and the client.12

Hence the FTP protocol uses the TCP protocol because there is reliability because the data sent gets delivered accurately at the destination.

*1: Transformation? Perhaps you meant “transmission”?*

*2: What does “interoperability” mean? Are you saying that UDP wouldn't provide interoperability?*

*3: Independently of what?*

*4: Doesn't UDP establish internetworking too?*

*5: More than what?*

*6: What are some of the intricacies? What kind of management? Is this at the end-points or in the network?*

*7: Less framework than what? Why is this important to FTP?*

*8: Good. You compared*X*to something, and included what*X*is being compared to.*

*9: Less latency than what?*

*10: What is application flexibility? Why is it important to FTP?*

*11: How do you have reliability when not transmitting data?*

*12: What kind of communication is lacking? Does this mean communication is unreliable?*

* 1. UDP is a connectionless protocol which means it sends data immediately. That makes it a fast protocol.13 So, it is used with applications that require fast transmission of data. For instance, it is used for DNS, DHCP, and real time communication. Some UDP disadvantages are reliability which means it is possible to not receive data. It also sends data not ordered and data reaches in random order. (Vivek Gite, *Cyberciti.biz*, 15 May 2007. Web. 18 Sep. 2012.)

On the other hand, TCP is a connection oriented protocol and that means it tests and checks connection14 then start sending data, and it guarantees that all data will reach the destination. It also sends data ordered. For that, TCP is a reliable protocol, and it corrects mistakes and re-sends corrupted data again. So it is used for HTTPS, SMTP, and FTP. However, that makes TCP slower than UDP when sending data. (Erik Rodriguez,*Skullbox.net*, 26 July 2011. Web. 18 Sep. 2012.)

FTP uses TCP protocol.

*13: Fast by what measure? Throughput? Delay? Low host overhead?*

*14: Test and checks the connection in what way? Does it do resource allocation?*

* + - TCP implementation:
      * Advantages:
        + Data is guaranteed to reach the end point, will reach in predicted time,15 lack of duplication.
        + All work needed to be done with sending the data is done for the user.16
        + Automatic breakdown of data into packets.
      * Disadvantages:
        + Any problems in the OS will cause problems while using the network.17
        + Cannot be used to broadcast and multicast connections18
    - UDP implementation:
      * Advantages:
        + Can broadcast and multicast connections
        + User is not restricted to the connection based communication model
        + Faster than TCP19
    - FTP only uses TCP.

Sources: laynetworks.com — *comparative analysis*

*15: Is this true? What kind of predictions are made? By what mechanism are predictions made?*

*16: Isn't that true for UDP too? Did you mean “reliably sending the data”?*

*17: Isn't that true for any network protocol?*

*18: Is multicast or broadcast for FTP?*

*19: Faster in what sense? Faster raw throughput? Faster usable throughput? Under what conditions?*

* 1. The main difference between TCP and UDP is that TCP connects and remains connected to the destination of the data while UDP does not; therefore, TCP is more reliable protocol while UDP is a faster one. Since transferring files is a function that ideally yields success on the first attempt, a more reliable protocol is typically used. FTP uses TCP to help ensure this success. Research Source:[http://www.bleepingcomputer.com/tutorials/TCP-and-UDP-ports-explained/](http://http/www.bleepingcomputer.com/tutorials/TCP-and-UDP-ports-explained/)

Section: “The two Internet workhorses: UDP and TCP”

TCP stands for Transmission Control Protocol. Using this method, the computer sending the data connects directly to the computer it is sending the data it to, and stays connected for the duration of the transfer. With this method, the two computers can guarantee that the data has arrived safely and correctly, and then they disconnect the connection. This method of transferring data tends to be quicker and more reliable, but puts a higher load on the computer as it has to monitor the connection and the data going across it. A real life comparison to this method would be to pick up the phone and call a friend. You have a conversation and when it is over, you both hang up, releasing the connection.

UDP stands for User Datagram Protocol. Using this method, the computer sending the data packages the information into a nice little package and releases it into the network with the hopes that it will get to the right place. What this means is that UDP does not connect directly to the receiving computer like TCP does, but rather sends the data out and relies on the devices in between the sending computer and the receiving computer to get the data where it is supposed to go properly. This method of transmission does not provide any guarantee that the data you send will ever reach its destination. On the other hand, this method of transmission has a very low overhead and is therefore very popular to use for services that are not that important to work on the first try. A comparison you can use for this method is the plain old US Postal Service. You place your mail in the mailbox and hope the Postal Service will get it to the proper location. Most of the time they do, but sometimes it gets lost along the way.

…FTP servers use TCP ports 20 and 21 to send and receive information, so you won't have any conflicts with the web server running on TCP port 80.20

*20: Ugh, don't do this. Cutting and pasting is complete waste of time: you learn nothing from it, and I can look it up from the citations if I need to.*

* 1. ## FTP / TCP
     + Advantages:
       - Connection-oriented protocol: any connection has to be set up before transferring data to the other side by using the 3-way handshake system, as a result, TCP has the ability to establish a connection ([HTTP://www.computer-networks.blurtit.com](http://http/www.computer-networks.blurtit.com), The Advantages Of Using TCP Over UDP).
       - Reliability: FTP uses the reliable TCP in order to guarantee that data (files) are sent and received without loss of data and without any corruption ([HTTP://tcpipguide.com](http://http/tcpipguide.com), FTP overview).
       - Flexibility by supporting multiple data types and file types21 ([HTTP://tcpipguide.com](http://http/tcpipguide.com), FTP overview).
       - Error correction by detecting errors and correcting them and resending segments when it's needed ([HTTP://www.en.wikipedia.org](http://http/www.en.wikipedia.org), TCP).
     + Disadvantages:
       - Slow speed ([www.diffen.com](http://www.diffen.com/), TCP vs. UDP).

## FTP / UDP

* + - Advantages:
      * Fast speed ([HTTP://www.diffen.com](http://http/www.diffen.com), TCP vs. UDP).
    - Disadvantages:
      * Connectionless protocol: UDP uses a simple transmission model, so UDP does not establish a connection, and it just sends the data (segments) directly without set up connection before transferring data to the other side ([HTTP:www.en.wikipedia.org](http://http:www.en.wikipedia.org/), UDP).
      * Unreliability.

## FTP primary uses TCP.

*21: Is it the case that UDP doesn't support multiple data types and file types? What are data and file types?*

* 1. Advantages of TCP when implementing FTP are:
     + It provides reliability, TCP gives the receiver a complete copy of the file. Data packets that are lost are resent again, if the connection fails then the data is re-requestred, thus making sure that data is received at the other end. Data arrives in order and that there are no duplicates.
     + It provides congestion control, the mechanism that throttles the sender when one or more links between sender and receiver becomes excessively congested.
     + It provides flow control.
     + This makes application developers work easier since everything is implemented at network level.

Disadvantages of TCP when implementing FTP are

* + - The extra overhead makes the file transmission slower.
    - It is expensive in terms of overhead at execution time, since it needs to provide reliability.
    - This does not support broadcast and multicast file transfers.22

pAdvantages of UDP when implementing FTP are

* + - It is much faster than TCP because it doesn't need to establish any connection between end points.
    - It is less expensive when compared to TCP in terms of overhead at execution time since it has only 8 bytes of overhead.
    - This supports broadcast and multicast file transfers.23

Disadvantages of UDP when implementing FTP are

* + - This doesn't provide reliability i.e. does not guarantee the data transfer, disordered packets, so this makes application developers work tedious to implement it in application level.
    - UDP's best effort service does not protect against datagram duplication, i.e., an application may receive multiple copies of the same UDP datagram.

FTP uses TCP protocol  
Reference: Textbooks and various sites.

*22: Is this something important to FTP?*

*23: If UDP doesn't establish connections between end-points, how can it support broadcast or multicast? What does “support” mean?*

* + - The advantage of using TCP when implementing FTP is that it provides a reliable connection guarantees correct sequencing of IP datagrams,24 guarantees delivery, guarantees no duplication. It can also establish connections between two different types of computers and servers25 and has scalable client/server architecture.26 The disadvantage of using TCP is it is complex to set up and manage.27
    - The advantages of using UDP when implementing FTP is that there is lower latency, application flexibility, and no connection state. The disadvantages are it does not maintain a reliable connection, does not preserve sequences,28 does not guarantee delivery or protect against duplication. Is best for minimum protocol intervention.
    - FTP uses TCP/IP

Source: Notes from Networking and Internet Technologies class at Rutgers.

*24: Is that what the application's sending, IP datagrams?*

*25: Can't UDP, or IP for that matter, do the same thing?*

*26: Scalable in what sense?*

*27: Complex for which side? Complex in what way?*

*28: Does not preserve sequences of what?*

* 1. The applications associated with FTP29 require all the data to be received in correct order. It is fairly simple that TCP provides this service and that's why FTP uses TCP, and not UDP.

In TCP lost packets are resent and thus it is reliable. It guarantees efficient delivery.30 TCP guarantees 3 things:

* + - The data gets there.
    - The data gets there in order.
    - Without duplication.

It has good throughput on a modem or LAN.31

Disadvantages of UDP

A packet may not be delivered in order or may be duplicated. And you get no indication that its been delivered or not unless the listening one says something.

UDP has no flow control.

No retransmission if data collides.

Disadvantages of TCP

* + - Extra overhead makes the transmission slower where in file transfer of large files transmission speed is important.
    - Its latency is its downside of TCP. It has to wait for acknowledgments.
    - TCP cannot be used for broadcast or multicast connections.32

Advantages of UDP while implementing FTP in it.

* + - Broadcast or multicast connections are available.33
    - Much faster than TCP.
    - Less expensive.

*29: FTP is the application being considered. What are the applications associated with FTP?*

*30: Efficient in what sense? Efficient at the end-points? In the operating system? Over the network?*

*31: Why is throughput across a modem interesting?*

*32: So? Why is that important to FTP?*

*33: Does FTP care at all about broadcast or multicast?*

1. How big would a 5,000 byte file be if it was encoded using base64? Assume 1) lines in the encoded file contains are 80 characters long except the last one which may be shorter, 2) the newline character ('\n') is one character and appears at the end of every line, and 3) the encoded file contains only lines from the original file, there is no extra header or footer information added.

Base64 encodes a source sequence of three bytes as a result sequence of four bytes by encoding successive groups of six bits from the source sequence as a byte (eight bits) in the result sequence. If necessary, the source sequence is padded (appended at the end) with one or two null (all zero) bytes to make the sequence size in bytes evenly divisible by three, and the result sequence is also padded with a distinguished byte value (which must exists because the source-sequence representation uses only 64 of the possible 256 values available) equal to the number of null bytes appended to the source sequence.

Source: [Base 64 Encoding](http://tools.ietf.org/html/rfc4648#section-4) (section 4) in [Base-N Encodings (rfc 4648)](http://www.ietf.org/rfc/rfc4648.txt).

A 5,000-byte sequence requires one pad byte to be divisible by three, and the Base64-encoded resulting sequence contains (5001/3)\*4 + 1 = 6668 + 1 = 6669 bytes. The result sequence divides into ceiling(6669/79) = 85 lines of at most 79 characters, and the total size of the result sequence is 6669 + 85 = 6754 bytes.

The answers, approximately verbatim, and in no particular order:

* 1. Input file size = 5000 bytes

|  |  |  |
| --- | --- | --- |
| Base64 | = | bytes + 21 - ((bytes + 2) MOD 3)/3 \* 4 |
|  | = | 5000 + 2 - ((5000 + 2) % 3) / 3 \* 4 |
|  | = | 6668 bytes = 6.51172 KB2 |

* 1. The output file size is **6668 bytes = 6.51172 KB.**3   
     ([HTTP://www.obviex.com](http://http/www.obviex.com), How to Calculate the Size of Encrypted Data).
  2. *1: Why “+ 2”? What does it represent? Could it be related to padding? How?*
  3. *2: Why translate from exact character units to approximate kilobyte units?*
  4. *3: Does this include the newlines added to the encoded result?*
  5. The formula for measuring the size of file encrypted4 with Base64 is:

Base64 = (File\_size + 25 - ((File\_size + 2) MOD 3))/36 \* 47

Base64 = (5000 + 2 - ((5000 + 2) MOD 3))/3 \* 4

The file size will be: 6668 Byte.8

*4: Encoded, not encrypted, although the information is obscured after being run through Base64.*

*5: Why 2? What does it represent?*

*6: Anything mod 3 will be less than 3, and dividing anything mod 3 by 3 makes it less than 1. What does this value represent?*

*7: Is padding included somewhere in this equation? Is that what the + 2 is for?*

*8: Where does this calculation account for the newlines added to the encoded result?*

* 1. Original file size is 5000 bytes.  
     So 5000 byte file approximately9 contains 5000 characters.  
     Base64 encoding generates a file which is 137%10 more than of original file.  
     So encoded file size is 5000\*1.37 = 6850 bytes  
     So it approximately contains 6850 characters.11   
     Number of lines in that encoded file = 6850/80 = 85 lines  
     Since, each line in encoded file has 80 characters.1213   
     References: multiple sites14

*9: Approximately? Why approximately? Why not exactly?*

*10: 137%? How did you get that number?*

*11: Again, why approximately?*

*12: Given assumption 1 in the question, is it always true that a line has 80 characters?*

*13: What's the answer? Is it 85 lines?*

*14: This is an unhelpful citation. Where would I go if I wanted to check your answer?*

|  |  |
| --- | --- |
| Output size | ((input\_size - 1)/3)\*4 + 415 |
|  | ((5000 - 1)/3)\*4 + 4 = 6669 |
| final size | output\_size + (output\_size/80)\*216 |
|  | 6669 + (6669/80)\*2 = 6835.725 bytes17 |

* 1. Source [http://stackoverflow.com/questions/1533113/calculate-the-size-to-a-base-64-encoded-message](http://http/stackoverflow.com/questions/1533113/calculate-the-size-to-a-base-64-encoded-message) ; answer by kanaka.
  2. *15: Why does this formula subtract one from the input size? And what does that final “+ 4” represent?*
  3. *16: What does multiplying by 2 represent?*
  4. *17: Fractional bytes? There are bits left over? (Note that 5/8 = 0.625 < 0.725 < 0.75 = 6/8, so there are fractional bits too.) Is that how Base64 encoding works?*
  5. 4[n/3]  
     (4[5000/3])/8018   
     (4[1666.67])/8019  
     6666.68/8020  
     83.3321

Sources: *Computer Networks*, pg 702

*18: Is there any padding being described by this equation? Does Base64 do any padding?*

*19: Are the added newlines accounted for by this equation?*

*20: What units are 6666.68 in?*

*21: Units?*

* 1. Base64 takes 3 bytes at one time and converts them to 4 Base64 characters. A 5,000 byte file would consist of 6666.67 characters after the conversion (5000 bytes/3 bytes per group = 1666.67 groups of 3 bytes.221666.67 groups \* 4 characters per group = 6666.67 characters). 6667 characters (rounded up)23 will require 84.39 lines in the file (6667/79 characters per line, \n is the 80th character on each line). Final answer: The file will be 85 lines long to sufficiently contain the 5,000 byte file endoded in Base64.24

Research Source: [http://email.about.com/cs/standards/a/base64\_encoding.htm](http://http/email.about.com/cs/standards/a/base64_encoding.htm), section Base64 to the rescue (explanation and example).

*22: How is it Base64 produces fractional characters?*

*23: How are characters rounded up?*

*24: But how long is a line?*

* 1. Total bytes of data = 5000 bytes.

The base64 converts 3 bytes of data into 4 characters. So 5000 bytes of data is converted into

\[{5000 \cross 4} \over 3 = 6666.66 = 6667\] characters25

Given that each line has 80 characters. So clearly number of lines = 6667/80 = 83.3 lines.

i.e. Total no. of lines = 8426

*25: How does the rounding up go? What extra data is added to produce an integer?*

*26: But how long is a line?*

* 1. Given a 5000-byte file, Base64 convertes 3 bytes of data into 4 characters i.e., 1 byte = (4/3)characters.27 So for 5000 bytes it is = 5000\*(4/3) = 6666.667 characters.28 Lines in the encoded file contains are 80 characters i.e., 1 line contains 80 characters.29 Hence, the no. of lines = 6666.667/80 = 83.333 lines.

*27: What kind of unit is characters? How many bits does it have?*

*28: Fractional characters? What do they look like?*

*29: Is that true? Is that what the problem states? (Hint: no)*

1. What is Zipf's law? How would a file server exploit Zipf's law to improve performance? What would be the expected benefits of exploiting Zpif's law?

Zipf's law was originally an observation about the relation between frequency and rank ordering in English words: the frequency of the *i*th most frequently occurring word is proportional to 1/*i*. If Zipf's law holds, the most frequent word (rank 1) has a frequency that's twice the frequency of the second most frequent word, three times the frequency of the third most frequent word, and so on.

A server offering a population of objects for which Zipf's law holds can exploit the law by creating a static, two-level cache. A two-level cache is sufficient because Zipf's law divides the population into popular (frequent) and unpopular (infrequent) objects, and the popular objects are much more popular than the unpopular objects. The cache can be static because accessing an unpopular object can be considered a rare event, and moving the object into the popular cache would be a waste of time because it's unlikely to be accessed again soon.

Source: [HTTP://xlinux.nist.gov/dads/HTML/zipfslaw.html](http://http/xlinux.nist.gov/dads/HTML/zipfslaw.html)

The answers, approximately verbatim, and in no particular order:

* 1. **## Zipfs law:'** “Zipfs law is the observation that frequency of occurrence of some evnet (P)”   
     ([HTTP://www.linkage.rockefeller.edu](http://http/www.linkage.rockefeller.edu), introduction to Zipf's law).

## A File server can exploit Zipf's law to improve performance by identifying the most popular requested files on itself, and it can also tell how many times these files have been downloaded, modified, or even opened.1 So, the file server can make a prior access to these files and put them on the top of its files’ list. Thus, this process will improve the file server performance because it is going to speed up the access speed to the requested file.

## As mentioned previously, Zipf's law will be beneficial to identfy what or where the popularity is and gives it the priority to be accessed; for example, Zipf's law can give a hand of help with search engines, etc.

*1: What in particular about Zipf's law improves performance over the usual cache operation?*

* 1. [ not answered ]2

*2: Always answer the question. Even a lame answer can earn some points, but no answer can only earn zero points.*

* 1. Zipf's law states that the probability of occurrence of words or other itmes starts from high occurrence and then reduces off. Thus, many items occur rarely while a few occur very often. In other words, the frequency of occurrence of any word is inversely proportional to its rank in the frequency occurrences table.

Formula: *Px* = 1/*xa*, where *Px* is the frequency of occurrence of the *x*th ranked item and *a* is close to 1.

A file server would exploit a Zipf's law to improve the performance by enhancing the performance of the cache i.e. this property leads to effective web caches, which contain the most popular objects and typically employ the least frequently used replacement policy due to which the server often achieves higher cache hit rates.

The expected benefits of exploiting the Zipf's law are:

* + - Since the requests are served immediately from the cache, the response time can be significantly faster than contacting the origin server.3
    - Catching conserves bandwidth by avoiding redundant transfers along remote internet links.4
    - The content reaches the users more quickly and they avoid being overloaded themselves by many direct requests.

*3: Is this a property of Zipf's law, or of caching?*

*4: Ok, but what does this have to do with Zipf's law?*

* 1. Zipf's Law (refers to George K. Zipf) describes the incidence of distinct objects5 in special sorts of collections.6 (Aaron Krowne, *Planetmath.org*. Version 4. Web. 18 Sep 2012.)

Server exploits Zipf's law can improve sorting and delivering data methods according to popularity and importantly of data.7 As a result, that can decrease access time to contents in the server which saves time and resources.

*5: What does the description say about the incidence of distinct objects? Is there any particular relation described?*

*6: What's special about the collections?*

*7: Ok, but how might that be possible? what is it about Zipf's law that makes this possible?*

* 1. Zipf's law: Zipf's law states that the relative probability of a request for the i'th most popular page is inversely proportional to ‘i’. It specifies that popularity objects are ranked according to their popularity, then the probability that the user chooses the ‘m’th item on the list is 1/m.

File server exploits Zipf's law by searching the request8 based on least occurrence of words by removing highly ranked words.9

*8: Perhaps that phrase should have read “searching for the request”?*

*9: If I'm understanding the answer correctly, it's suggestion that commonly occurring words occur in too many results to be useful and should be thrown away in favor of less commonly occurring words. Ok, but what does that have to do with Zipf's law? Zipf's law relates frequency and rank ordering in a particular way, and this answer doesn't exploit that relation.*

* 1. Zipf's law outlines how often individual objects in a set will occur. The frequency of an object in a set is inversely proportional to its overall frequency.10 A file server could use Zipf's law to stack the cache with more frequently used files.11 This act would increase the speed of transfers using this server.

Research Source: [HTTP://planetmath.org/ZipfsLaw.html](http://http/planetmath.org/ZipfsLaw.html), section: Zipf's law (explanation, formulae, graph).

*10: Isn't it more like the probability of occurrence is inversely proportional to the rank?*

*11: Doesn't replacement cache do this anyway? How does Zipf's law help make this better?*

* + - Zipf's law is the theory that the most common words in the English language start off with a high rate of occurrence which begins to drop as the words become less common.12 This can work with other types of data as well.
    - A file server could exploit Zipf's law by looking for the most common information13 and patterns14 by repeating how it handles the information15 which will make the process faster.
    - Exploiting Zipf's law would allow for smoother performance and move similar information faster through the system.

Sources: Planet math — *Zipfs law'*

*12: That's all Zipf's law states? That the most common words have a high rate of occurrence? Isn't that tautological?*

*13: What kind of information? Information (that is, queries) that comes from the clients, or information (that is files) that comes from the server? Or maybe both.*

*14: What kind of patterns?*

*15: What does repeating information handling involve?*

* 1. The most frequently occurring words are rankded in increasing order of their occurrence. So most frequently occuring word is ranked 1.16

These most frequently occurring words are non-content words and the least frequently occurring words i.e. with high rank have more content in it.17

The file server exploits Zipf's law to improve performance by “effective caching”. This property provides an important tool in designing architectures of web caching. Zipf's law helps in selecting which objects to cache.18 In this case it uses Zipf's law to select content objects to cache to improve its performance. Popularity of a video file can be calculated using Zipf's law.19

*16: Is this all Zipf's law says?*

*17: Is Zipf's law concerned with meaning (word content), or popularity?*

*18: How does Zipf's law help? How is it used to select objects for caching?*

*19: Is that what Zipf's law is concerned about? Isn't popularity determined by how many times an object is used?*

1. A video file contains 640 × 420 pixel frames with 16 bits of color per pixel and a 50 frames/sec display rate. How much bandwidth is required to stream this file?

A 640 × 420 frame has 268,800 pixels; at 16 bits of color per pixel, a frame has 4,300,800 bits. 50 frames per second is equivalent to 215,040,000 bits per second. Unencoded video transmission requires a little under a quarter gigabit per second.

The answers, approximately verbatim, and in no particular order:

* 1. The formula of measuring the required bandwidth is:

Required Bandwidth = Frames/sec × Resolution × Color Depth

1

Required Bandwidth = 50 × 640 × 420 × 16 = 215,040,000 Bits/second = 205.07 Megabits/second.

*1: Do the units carry through in this equation?*

* 1. Pixels/frame = 640\*420 = 268800  
     Bits/frame = 268800\*16 = 4300800  
     Bandwidth = 4300800\*50 = 215040000 = 215Mbps  
     Reference: Text (A Systems approach)
  2. Given

a video file contains 640\*420 pixel frames = 268800 pixels

Bits of color per pixel = 16

display rate = 50 frames/sec.

Now,

|  |  |  |
| --- | --- | --- |
| The data in one frame | no. of pixels \* no. of bits/pixel |  |
|  | 268800\*16 |  |
|  | 4300800 bits |  |

Hence,

|  |  |  |
| --- | --- | --- |
| The no of bits displayed in one second | display rate \* data in one frame |  |
|  | 50\*4300800 |  |
|  | 215040000 bits = 215 Mbps |  |

So 215 Mbps bandwidth is required to stream this video file.

* 1. A video file with 640 × 420 pixel frames would normally have a size of 268,800 bytes; however, since the file uses 16 bits of color per pixel, this number is doubled (8 bits per byte normally) to 537,600 bytes. The file for this problem requires a fram rate of 50 frames per second, so the bandwidth that is needed to stream this video is 537,600 bytes multipled by 50 frames/second, yielding a final answer of 26,800,000 bytes/sec.

Source: [HTTP://www.pk3.org/Astro/index.htm?astrophoto\_vesta\_pro.htm](http://http/www.pk3.org/Astro/index.htm?astrophoto_vesta_pro.htm)

(Example) The reason of above facts is limited throughput of USB. Vesta Pro is using YUV420 codec, which requires 12 bits per pixel. That means, that, for example, 640x480 pixels frame has size 460800 bytes. For 5 fps video stream it requires 2304000 bytes/s - it is more than throughput of USB. That's why there must be used some compression of video data, which are sent through USB. As my measurements confirmed, the compression is lossy.2

*2: Really, don't do any quoting; just the citation will be enough.*

420 × 16 = 6720 bits per frame3

640/50 = 12.8 seconds of video

6720 × 50 = 336000 Kilobit per second

It would take 322.560 Kilobytes, or 0.315 Mbytes

Sources: astream.com — *streaming bandwidth calculator*

*3: Ok, this was my mistake.*

15KB4 × 10 bits/byte5 = 150 ; 150 × 50 frames/sec = 7500 Kbits/sec

[http://www.imakenews.com/kin2/e\_article000345313.cfm?x=b11,0,2](http://http/www.imakenews.com/kin2/e_article000345313.cfm?x=b11,0,2)

*4: From where did 15KB come? What does it represent?*

*5: What does 10 bits/byte represent?*

6

|  |  |  |
| --- | --- | --- |
| **Bandwidth** \* (16 bits) \* (50 frames/sec) |  |  |
|  | 215 040 000 bits |  |

**The required bandwidth to stream this file is 215 040 000 bits.**

*6: Do the units carry through to bits?*

Number of bits per pixel = 16.

Total no. of pixels is 640 × 420 = 268,800.

Data in one frame is 268,800 × 16 = 4,300,800 bits.

Given display rate is 50 frames/sec.

So in one second no of bits displayed = 50 × 4,300,800 = 215,040,000 bits = 215 mbits.

So clearly bandwidth required to stream this video is 215 mbps.

An FTP does not keep the port 20 data connection open for multiple file transfers. What is the justification for breaking the data connection after every file transfer?

Some transfer modes used by FTP require that the server close the data connection to indicate end-of-file.

Reference: Transmission Modes (section 3.4) from [File Transfer Protocol (rfc 959)](http://www.ietf.org/rfc/rfc959.txt).

The answers, approximately verbatim, and in no particular order:

Closing port 20 after every file transfer is an indication that the current file or data which is being sent was completely transferred. For that, server or receiver can know that the amount of data was reached to its destination.

[picture]

In step 1 client command part 1026, in this case sends a request to server command port, i.e. port 21.

It sends an acknowledgment in step 2 to port 1026.

In step 3 the server initiates the data connection at data port.

Finally the client sends an acknowledgment back.1

Since FTP used TCP, it has to send an acknowledgment back after each file transfer. So every time each file transfer is made Step 3 and step 4 are repeated, i.e. after each file transfer the data connection is broke2 and new connection is established.3

*1: Is this an acknowledgement in response to the server making the data connection, or to something else?*

*2: But why is the connection broken?*

*3: Even if there is no other data being transferred?*

The FTP does not keep the port 20 data connection open for multiple file transfers because it breaks the data connection after every file transmission.

The FTP client initiates a connection4 with the FTP server on its port 21.

Port 21 is where the server is listening for commands issued to it, and in turn, which it will respond to. Hence here the TCP/IP handshake is complete.5

At this point, the client begins to listen on its ephemeral port + 1, and sends the PORT N + 1 command to the server on its port 21, i.e., if the ephemeral port in use by the client is 1026, then it would listen on port 1027.

Once this is done the data tranfer port (port 20) on the FTP server would initiate a conneciton to the FTP client's ephemeral port plus 1, as indicated above. This is how an active FTP session is conducted by both the client and server.

Hence the port 20 data connection for multiple file transfers is not open because it breaks for every single connection after every file is transferred.6

*4: Which connection? There are two of them: command and data.*

*5: Is this something that FTP does, or does TCP do the handshake?*

*6: That's true, but why does FTP break the connection? That's what the question's asking?*

Breaking the data connection after every transfer is necessary to avoid confusion between different connections.7

Source: Notes from Networking and Internet Technologies class at Rutgers.

*7: Which different connections? Between the control and data connections? How many other connections are there?*

An FTP closes the port 20 data connection after each file transfer because it avoids confusion between the data connections. It's possible that old data from previous transfers could still be present, and reestablishing a conneciton each time helps avoid one transfer absorbing the old data from a previous one. It is also possible that a system could “lock up” because it is waiting for an end-of-transfer message8 that it might not receive due to “time outs” on the firewall's side.9 It is safer and more acurate for each transfer to have freshly opened ports.

Source: [HTTP://msmvps.com/blogs/alunj/archive/2009/07/13/1300796.aspx](http://http/msmvps.com/blogs/alunj/archive/2009/07/13/1300796.aspx)

In typical Stream Mode operation, a new data connection is opened and closed for each data transfer, whether that’s an upload, a download, or a directory listing. To avoid confusion between different data connections, and as a recognition of the fact that networks may have old packets shuttling around for some time, these connections need to be distinguishable from one another.

Source: [HTTP://www.ncftp.com/ncftpd/doc/misc/ftp\_and\_firewalls.html](http://http/www.ncftp.com/ncftpd/doc/misc/ftp_and_firewalls.html)

Even if the client program is planning on ending the session, the FTP requires that the client program send a message ("QUIT") to the server indicating that the connection should be closed, and the server is then required to reply with another message indicating that the session is officially closed. The ramifications are that the client program could then lock up waiting for a reply to a "QUIT" message that the server will not receive since the firewall timed-out the session, unbeknownst to both client and server. The solution for this specific case, which some, but not all, FTP client programs do, is to either place a very short time-out on the reply to the "QUIT" message, or to simply close its end of the FTP session (which violates the FTP protocol, but is de facto behavior and is generally accepted).

*8: From where does the end-of-transfer message come?*

*9: Firewall? Why is there a firewall involved?*

The justification for the FTP breaking the data connection after every file transfer is that FTP uses port 21 for listening to commands, while port 20 I for receiving data files. The connection is broken so that the FTP can receive the commands about the data10 before it receives the data, so there is no need to keep it open.

Sources: windosnetworking.com — *understanding ftp protocol*

*10: What does breaking the data connection have to do with receiving commands? Aren't they separate connections?*

**## There are two FTP modes**, and when FTP uses either active or passive mode, there will be a justification for breaking the data connection after every file transfer.

**## The Active FTP mode** is inefficient way to deal with a multiuser system because if many users made a lot of FTP requests, the system wouldn't be capable to match the all incoming FTP data connections to right users. Consequently, FTP does not keep the port 20 data connection open for multiple file transfers to make sure about the matching the incoming FTP data connection.11 Also firewalls might block connections on this mode.12

**## The Passive FTP mode** use to solve the firewalls issues that might block some connections. On this mode, the client initiates the data connection from its data port to the specified server data port to avoid blocking connections; as a result, the client opens a new connection port for each connection or every file transfer, so the server cannot deal with all incoming connections. Consequently, FTP does not keep the port 20 data connections open for multiple file transfers to make sure about the matching the incoming FTP data connections.

*11: But doesn't the command connection have the same problem? And FTP doesn't close the command connection after each command.*

*12: If a firewall blocks connections, how can connections be broken?*

An FTP does not keep the port 20 data connection open for multiple file transfer because when the server has completed sending data in a transfer mode then server needs to close the connection to indicate the end of file because server is done transferring requested data.  
Reference: rfc.13

*13: Which rfc? There are over 6,000 of them.*

This page last modified on 2012 October 1.

**Computer Networking**

**Homework 2, 25 September 2012**

This homework assignment has five questions; answer at all of them.

1. The IPv4 address space has run out (let us assume): ICANN has no unallocated IPv4 addresses to hand out. There is another address space associated with IPv4 that could also run out; what is it? (Hint: think UDP or TCP.) Explain how likely it is it that this second address space would run out.

Ports are the other IPv4 address space. An IPv4 address combined with a port is a 32 + 16 = 48-bit address. It seems unlikely that a single host would exhaust the port address space, as that would require that it be running 65,000+ processes at one time, but note that the 48-bit address space has the same problem as the class-based IPv4 address space: it is inefficiently distributed in the network and difficult to share. Hosts that need only a few addresses still gets a full compliment of them, and can't easily make the extras available to hosts that may need more addresses.

The answers, approximately verbatim, and in no particular order:

* 1. Besides ICANN, network address translation (NATs) can be used to connect computers to the Internet.1 NATs modify a single, unique IP address such that a group of computers can connect to the Internet.2 NATs will help alleviate the IPv4 exhaustion crisis insofar as IP addresses are needed within home networks and other such setups that lend themselves to the NAT “Band-Aid.” If unique sources continue to arise and require unique IP addresses (which is likely in this day and age), then the NAT solution will not be able to facilitate, and the “address space” will run out. It is currently being used as a bridge to smooth the transition to IPv6, which will have a much larger address space, and it seems to be a patch to the problem that will allow humanity to limp into the realm of IPv6 where the Internet can grow and thrive successfully again.3 Source: http://computer.howstuffwords.com/nat.htm Source: http://www.pjsip.org/pjnath/docs/html/group\_\_nat\_\_intro.htm

*1: How can ICANN be used to connect computers to the Internet?*

*2: How do NATs do that?*

*3: This answer is too long.*

* 1. Given the address span has run out: ICANN has no unallocated IPv4 addresses to hand out and there is another address space that could also run out associated with the IPv4.4 Then it is the NAT device that is responsible for establishing services from internal[?] addresses to extend devices.5 Will not work then[?] the translation[?] session6 is less than 64511 then it is inadequate if the IPv4 pool run out.

*4: Are you restating the question here, or is this part of the answer?*

*5: How does a NAT device do this? And what are external addresses? And what does it mean to extend devices?*

*6: What is the translation session?*

* 1. The second address space that is associated with IPv4 is subnets. This allows the addresses to be split into multiple parts for internal use as multiple networks, and still acting as a single network to the outside networks. Subnet masks can run out, however there are two options if you run out of a subnet mask. You can change it to allow for more devices7 or add a router to increase the IP address range, which will basically allow the user to start over with a new subnet mask.8 These two options make it highly unlikely to run out of subnet masks. Source: Computer Networks --- Tennenbaum and http://www.russbelew.com/pages/networking\_subnets.aspx

*7: Change what to allow for more devices? And what are the changes?*

*8: Do routers do that? How?*

* 1. Another address space associated with IPv4 is Port addressing. A port is a 16-bit identifier assigned to processes. So there are 2^16 ports available. There are 65,535 ports available for TCP and UDP connections in TCP/IP. The first 1024 ports are reserved for specific services and protocols. So 65535 - 1024 is still 64511 ports. Dynamic communication is any sort of network communication that doesn't already have a port specifically reserved for sending or receiving it. There's a possibility of running out of ports, it happens because an application has been grabbing those ports and not releasing them properly. So, over time, it uses up more and more ports from the dynamic range until we run out. TCP/IP port exhaustion is more likely to occur under high load conditions than under normal load conditions. On Windows Server 2003 and Windows XP the default range of ephemeral ports used by client applications is from 1025 through 5000. Under certain conditions it is possible that the available ports in the default range will be exhausted. References: Class Notes & Textbook
  2. TCP or UDP if transmission session one less than 645119 then it is inadequate f10 if the IPv4 pool runs out. So NAT device which is responsible for establishing sessions from internal addresses11 to internal device doesn't work if transmission session one less than 64511.12

*9: What does it mean for a TCP or UDP transmission session to be one less than 64511? And what's so special about 64511?*

*10: What is inadequate?*

*11: What are internal addresses?*

*12: Did you argue somewhere in this answer why establishing sessions from internal addresses doesn't work?*

* 1. The second address is IPv6. IPv6 is a new version of the Internet protocol which was designed to replace the old protocol which is IPv4 since it ran out of addresses.13 IPv6 was designed by the Internet Engineering Task Force which is known as IETF. It is known that every technology has a life span and one technology will not last forever. When IPv4 was first designed, it was unlikely to think about a problem like running out of IPv4 addresses since IPv4 was offering addresses up to 4.3 billion addresses. In 2011, world had to accept the fact that 4.3 billion addresses were exhausted. Fortunately, the replacement version of IPv4 offers addresses up to 3.4 x 10 sup 38 addresses or in other words 430 times 10 to the 36th power.14 As we mentioned before, every technology has a life span, and has IPv4 ran out of addresses, IPv6 would too but not in the near future. IPv4 took 3 decades to run out with 3.4 billion addresses. Meanwhile, technology and devices 3 decades ago are not the same as today and more decades later. researches show that there were 10 billion devices connected to the Internet in 2011, and there will be 25 billion in 2020. (Dave Asprey cloud.trendmicro.com, 25 Sep. 2012. Web). That means there will be 3 devices for each person on Earth in 2020. What if we consider people and devices in 2030 or 2050? The number and ration will be increasing dramatically. However, if that ratio will not change, we will be using IPv6 for 2.55e31 more years.15

*13: But the question asks about IPv4.*

*14: Is this part of the sentence necessary?*

*15: This answer is way too long.*

* 1. The other address space associated with IPv4 is IPv6 (Internet Protocol version 6).16 IPv6 is the newest version of the Internet protocol, and it is growing to replace the IPv4 because IPv4 is exhausted of the Internet growth and it is not longer capable to deal with that. So, IPv6 was developed to deal with the running out of IPv4 addresses. IPv6 would be run out as same as IPv4, but by different reasons, which are: \* Most of ISPs around the world are not interested in IPv6 because there is not need to it as long as IPv4 can get the job done. \* Rare uses of IPv6 within places around the world such as Europe and Asia. \* Implementing IPv6 is difficult or even impossible sometimes because of some consequences such as changing the Network Layer protocols, changing Application layer protocols, and replacing the foundations. Reference is (Computer Networking, a Top-Down Approach, pages 320--326).

*16: In what way is IPv6 associated with IPv4?*

1. A gateway can processes a maximum of 2 million packets/sec. Assume the average offered load for gateway is 1.5 million packets/sec. A packet is sent from Seattle to Miami along a route passing through ten gateways. What fraction of the packet's total transit time for the packet is spent in gateways?

A gateway with a maximum throughput of 2 x 10 sup 6 packets/second handles 1.5 x 10 sup 6 packets in (1.5 x 10 sup 6)/(2 x 10 sup 6) = 0.75 sec. A packet arriving at a gateway under average offered load can expect to spend 0.75 sec reaching the other side of the router. Miami and Seattle are approximately 2,700 great-circle miles apart. Light travels through fiber at 124,000 miles/sec, which means a packet spends somewhere around 2,700/124,000 = 0.02 sec in the cable. The fraction of time a packet spends in gateways when going between Miami and Seattle is (0.75 times 10)/((0.75 times 10) + 0.02) = 99.7%

The answers, approximately verbatim, and in no particular order:

* 1. So p = 1.5/2 = 0.75 Each packet experiences a delay four times that in an idle system.1 So the time in idle system is 500 nsec or 0.5 usec. Here it is 4 x 0.5 = 2 usec. As these are 10 gateways, total time taken is 2 x 10 = 20 usec.

*1: Why is that true? Is there an idle system somewhere in this problem?*

* 1. A gateway can process a maximum of 2 million packets/sec, i.e. μ = 2 million. the average offered load for gateway is 1.5 million packets/sec i.e. λ = 1.5 million. So \[P = {\lambda \over \mu} = {1.5 \over 2} = 0.75\] From queuing theory, each packet experiences a delay 4 times in an idle system,2 i.e., the latency[?] in an idle system is 500 nsec.3 Hence here it is 500 nsec t × 4 = 2 usec. Now the packet is passing through 10 gateways so the total transit time sent through the gateways is 2 × 10 = 20 usec

*2: What? If the system is idle, why would a packet experience any delay? And is the system described in the question idle?*

*3: Why 500 nsec?*

* 1. Data we have: \* Maximum number of packets (M) : 2,000,000 packets/sec \* Average number of packets (A) : 1,500,000 packets/sec \* Number of gateways (NG) : 10 Formula: 1/(1 - (A/M))\*NG 1/(1 - (A/M))\*NG = 1/(1 - (1500000/2000000) = 4 microseconds4 4\*10 = 40 microseconds

*4: Why is this microseconds?*

* 1. 1 gateway, maximum of 2 million packets/second \ average offered load for one gateway, 1.5 million packets/second \ One packet passes through ten gateways
  2. If the packet arrives at a gateway that is beyond its 2 million packet/sec limit, it will be forced to wait until the gateway frees up space and is able to push the packet through to the next gateway. Since the average offered load for the gateways is 1.5 million packets/sec, this overload issue (theoretically) will not occur as frequently.5 If 1.5 million packets/sec is the average workload, it is possible to assume that maybe three of out the ten gateways are overloaded at the time of the packet's receipt.6 If this scenario is true, the packet will spend at least one second at each of the three gateways waiting to be processed.7 That time plus the standard time that it takes each packet to be processed by the gateway is the total amount of time that it spends within the gateways.8 This number divided by the total amount of time it takes the packet to travel from Seattle to Miami is the fraction of time it spends in the gateways. Source: Class lecture 10/03/2012.

*5: Or at all, so why bring it up?*

*6: Why is that assumption valid?*

*7: What happens to the packet at the other seven gateways?*

*8: How much time is that?*

* 1. As mentioned, the gateway processes 2 million packets/sec as a Maximum, and the average offered load for gateway is 1.5 million packets/sec as a Minimum. Also, there are 10 gateways. To calculate the packet's total transit time through one gateway, we should divide the Minimum process on Maximum process.9 Then, we should multiply the packet's total transit time through one gateway in how many gateways. The packet's total transit time through (n) gateways = (Min/Max)\*(n) Gateways = (1.5 million/2 million)\*10 = 7.5 seconds The total transit time for a packet through 10 gateways is 7.5 seconds.10

*9: Why? What are the units in the result? What does it represent?*

*10: Is that what the question asked for?*

* 1. Given a gateway can process 2 million packets/sec = u \ The load offered to it is 1.5 million packets/sec = lambda \ From queuing theory, p = u/lambda = 2 x 10^6/1.5 x 10^911 = 0.75 \ The time in an idle state12 is 500 nsec, here it is 2 usec.\ The route from source to destination contains 10 routers \ So, time spent by a packet in gateways is 10 x 2 usec = 20 usec. Reference: textbook

*11: 9 or 6?*

*12: What idle state?*

1. Suppose you assigned IPv6 addresses to grains of sand in the Gobi desert. Assuming that 1) you assigned every IPv6 address to the sand and 2) every grain of sand got an IPv6 address, how many addresses does each grain of sand get? The answer can be fractional; if every three grains gets two addresses, each grain gets 2/3 addresses, if every three grains get seven address, each grain gets 7/3 of an address.

The Gobi desert is 5 times 10 sup 5 square miles. A square mile is 27,878,400 square feet, and the Gobi desert is 5 times 10 sup 5 times 27,878,400 = 1.4 times 10 sup 13 square feet. Assuming the desert is uniformly 10 feet deep, the Gobi desert is 1.4 times 10 sup 14 cubic feet. A grain of sand is 3.5 times 10 sup -5 cubic feet, and there are (1.4 times 10 sup 14)/(3.5 times 10 sup -5) = 4 time 10 sup 18 grains of sand in the Gobi desert. There are 2 sup 128 ≈ 3.4 times 10 sup 38 IPv6 addresses, and each grain of sand gets (3.4 times 10 sup 38)/(4 times 10 sup 18) ≈ 2.5 times 10 sup 18 IPv6 addresses. Sources: Wolfram alpha - How big is the gobi desert? How big is a grain of sand?

The answers, approximately verbatim, and in no particular order:

* 1. Assuming that the Gobi desert covers 1.3 million square kilometers, the conversion to square meters yields 13 m^2. Desert sand heights range from 0 to 100 feet typically, so assuming the ceiling value of 100 feet (converted to 30.48 meters), the volume of the Gobi desert is (very roughly) 1,3000,000,000,000 m^2 \* 30.48 m, which equals 39,624,000,000,000 m^3. Assuming a 1mm diameter for an individual grain of sand, there would be approximately 1,3000,000,000 grains of sand in one cubic meter. Multiplying this number by the approximate volume of the Gobi desert. Given that there are 340,282,3666,920,938,463,374,607,431,768,211,456 addresses available in a 128-bit space, addresses divided by grains of sand results in 6.61 x 10^15 addresses per grain of sand. Source: http://www.why.is/svar.php?id=4803 Source: http://www.quora.com/How-deep-does-the-sand-go-in-a-desert Source: http://www.freebsd.org/doc/handbook/network-ipv6.html Source: http://gobidesert.org/content/facts
  2. Gobi area is 1,295,000 square kilometers. Average diameter of a grain of sand is 1mm therefore best estimate is 1645 million billion grains. IPv6 uses 1028-bit addresses, allowing 2^128, or approximately 3.4x10^38 addresses. Number of IPv6 addresses available are 3.4x10^38 addresses. So each grain gets (3.4x10^38)/(1645x10^15) addresses = 0.00206686 x 10^23 addresses. Reference: \ Wikipedia for Gobi desert area \ Text book for IPv6 addresses number
  3. [ unanswered ]
  4. No of addresses = 2128 No of grains = 10191 So each grain gets \[ \frac{2 ^ {128}}{(2)^{19}(10)\_^{19}} = \frac{2 ^ {119}}{(10)\_^{19}}. \]

*1: From where did this number come?*

* 1. \* First Step: The size of Gobi desert in cubic kilometer: Length x Width x Height 1500 x 800 x 1 = 1200000 km sup 3 \* Second Step: Convert the size of Gobi desert from km sup 3 to mm sup 3 Size = 1.2e24 mm sup 3 \* Third Step: In average, we have 20 grains per centimeter. (Since we have a minimum of 15 grains and a maximum of 25 grains) In cubic centimeter that equals: 20 x 20 x 20 = 8000 cm sup 3 In cubic millimeter: 8000 x 1000 = 8000000 mm sup 3 Estimated number of stand grains in Gobi desert is: 1.2e28 x 8000000 = 9.6e30 \* The Result: Number of IPv6 per grain = 3.4e38/9.6e30 We get: 35416666.6 IP per grain. (http://mathdude.quickanddirtytips.com. 20 July 2012. Web)
  2. Total number of addresses that IPv6 can hold is 2^128 = 3.4 x 10^38 Estimated number of grains on earth = 7.5 x 10^182 So each grain gets 3.4 x 10^38/7.5 x 10^18 = 34 x 10^39/75 x 10^19 = 0.5 x 10^20 = 5 x 10^19 IP addresses approximately.

*2: From where did this number come? And is that what the question asks about?*

* 1. Firstly, let's assume how many grains of sand in the Gobi desert: The Gobi desert area is about 1,295,000 square km, and my assumption for the diameter of a grain of sand is about 1 mm. Then, the grains of sand per km are about 1,000,000 grains in 1 km. Finally, to assume how many grains of sand in Gobi desert I should multiply the Gobi desert area in how many grains of sand per km. Grains of sand in Gobi desert = the Gobi desert area \* Grains of sand per km = 1,295,000 \* 1,000,000 = 1.285e+12 grains of sand in the Gobi desert Secondly, IPv6's address space size is about 3.402823669 x 10^38 = 3.4028237e+38. Lastly, to know how many IPv6's addresses that each grain of sand gets, I divided the IPv6 address space size on now many grains of sand in Gobi desert. Each grain of sand gets = IPv6's address space size/Grains of sand in Gobi desert = 3.4028237e+38/1.925e+12 = 2.6276631e+26 Each grain of sand gets 2.6276631e+26 IPv6 addresses. References is (www.en.wikipedia.org, IPv4 address exhaustion / IPv6 / Gobi desert).

1. The gateways *Gi* in the internet

an internet

drop packets with probability *p*; the networks *Ni* always deliver a packet from one edge to the other (they never lose packets). If host *S* sends a packet to host *D* and doesn't get a reply from *D* within some time, it resends the packet, and keeps resending the packet until it gets an reply. *D* only sends packets to *S* in reply to packets received from *S*; *D* never spontaneously sends a packet to *S*. Counting each network transit has a hop, what is the average (mean) number of

* 1. hops made by a packet sent by *S*?
  2. times a packet is sent (initial + retransmissions) by *S*?
  3. hops made by a packet received by *D*?

State your assumptions, and justify your answers.

As always, the trick to probability problems is to count the right things. In this case, if a packet drop is represented by 0 and a packet pass-through is represented by 1, then a successful packet delivery is a sequence of 0s and 1s such that the last two digits are 11 and every other 1 in the sequence appears between two 0s or starts the sequence. Given that every successful transmission ends with 11, we can ignore them and remember that they're there. A *n*-digit sequence with *i* 1s has probability (p sup i)((1 - p) sup (n - i)) assuming each gateway makes its drop-pass decision independently (which is a convenient but questionable assumption). However, not every sequence is legal, only those in which every 1 has 0s for neighbors. How many sequences is that? An *n*-digit sequence has at most ⌈*n*/2⌉ 1s because otherwise there wouldn't be enough 0s to separate adjacent 1s (the last digit in the sequence must be 0 because otherwise there would be three adjacent 1s). \* What is the mean number of hops made by a packet sent by *S*? The mean (expected) number of hops is the number of hops *i* times the probability that it takes *i* hops to go from *S* to *D*. For *i* less than 2, the probability is 0. For *i* = 2, the probability is (1 - *p*) sup 2, assuming each router makes its drop-no drop decision independently (which is an arguable assumption, but good enough for this problem).

The answers, approximately verbatim, and in no particular order:

* 1. Hops are number of routers/gateways a packet passes through to get from one computer to another. In computer networking, a hop represents one portion of the path between source and destination. When communicating over the Internet, for example, data passes through a number of intermediate devices (like routers) rather than flowing directly over a single wire. Each such device causes data to "hop" between one point-to-point network connection and another. (computernetworking.about.com) Based on that: Hops made by a packet sent by it(S) are 3 hops. Times a packet is sent by S: In regular cases, a packet is sent from it(S) to it(D) once, so initial time = 1. it(S) keeps sending packet to it(D) as long as it(D) is not replying. That means the times of retransmission depend on having replies from it(D) or not. If it(S) received a reply, times of retransmission would be 0. If it(S) didn't receive a reply, times of retransmission would be: n - 1. Where *n* is the number of sent packets. Hops made by a packet received by it(D) are 3 hops.
     + First hop probability is p. for second hop probability is p(1 - p). For third hop it is (1 - p)(1 - p). So1 p + 2 x (1 - p)p + 3(1 - p)(1 - p) = p + 2p - 2p^2 + 3(1 - p) - p + p^2) = p + 2p - 2p^2 + 3(p^2 - 2p + 1) = p + 2p - 2p^2 + 3p^2 - 6p + 3 = p^2 - 3p + 3
     + Probability of successful transmission is (1 - p)^2. The average number of transmission per packet is approximately 1/(1 - p)^2.
     + average hops per packet = average hops per transmission x average no. of transmissions = (p^2 - 3p + 3)/(1 - p)^2

*1: What happens if it takes more than three hops?*

* 1. \* Each packet may make 1, 2, or 3 hops. For 1 hop, the first router drops it and the probability is p. For 2 hops, it goes through first router but not the second and the probability is (1 - p)p. For 3 hops, it goes through both routers and the probability is (1 - p)(1 - p). Average number of hops made by a packet sent by it(S) is given by 1 x p + 2 x (1 - p)p + 3 x (1 - p)(1 - p) = p + 2p - 2p^2 + 3 - 6p + 3p^2 = p^2 - 3p + 3 \* The probability of successful transmission all the way is (1 - p)^2. Let us denote it by w. The average number of times a packet is sent is given by w + 2w(1 - w) + 3w(1 - w)^2 + \(cdots\) nw(1 - w)^(n - 1) + \(\cdots\) = 1/2w = 1/(1 - p)^2 \* Average number of hops made by a packet received by it(D) = Average hops made by a packet sent by it(S) x Average number of times a packet is sent transmissions is (p^2 - 3p + 3)/((1 - p) x (1 - p)) References: textbooks.
  2. If host it(S) sends a packet to host it(D) and doesn't get a reply from it(D) within some time, it resends the packet, and keeps resending the packet until it gets a reply. it(D) only sends packets to it(S) in reply to packets received from S; D never spontaneously resends a packet to S. Counting each network transit has a hop, what is the average (mean) number of \* Hops made by a packet sent by S? 3 (once through N1 to G1, once through N2 to G2, and once through N3 to D).2 \* Times a packet is sent (initial + retransmissions) by S? Assuming both G1 and G2 drop the packet on the first time 5 times. \* Hops made by a packet received by D? Assuming G1 and G2 do not drop the packet at all 3 hops; if G1 drops the packet and G2 does not then 4 hops. If both G1 and G2 drop the packet the first time then 5 hops.

*2: Does the packet ever get dropped?*

* + - Hops made by a packet sent by *S*? Each packet makes 1, 2, or 3 hops. For 1 hop, the probability is *p*. For 2 hops, the probability is (1 - *p*)*p*. For 3 hops, the probability is (1 - *p*)(1 - *p*) = (1 - *p*)2. The mean hops per transmission is3

|  |  |
| --- | --- |
| (1 x *p*) + 2 x *p*(1 - *p*) + 3(1 - *p*)2) | |
|  | = *p* + 2(*p* - *p*2) + 3(*p*2 - 2*p* + 1) |
|  | = *p* + 2*p* - 2*p*2 + 3*p*2 - 6*p* + 3 |
|  | = *p*2 -3*p* + 3 |

* 1. Times a packet is sent is i.e. (initial + retransmissions) by *S*. The probability of successful transmission all the way = (1 - p)^2. Let us say (1 - p)^2 is “*x*”. The expected no. of transmissions per packet is
  2. x + 2x + (1 - x) + 3 times (1 - x)^2 + \(\cdots\) + n times (1 - x)^(x - 1) + \(\cdots\)
  3. which will reduce to 1/*x*. Hence i.e.
  4. 1/x = 1/(1 - p)^2
  5. Hops made by a packet received by b is

|  |  |  |
| --- | --- | --- |
| mean hops | = | mean hops made by a packet sent by *S* \* mean number of transmissions |
|  | = | (p^2 - 3p + 3) times (1/(1 - p)^2) |
|  | = | (p^2 - 3p + 3)/(1 - p)^2 |

* 1. then the hops made by a packet received by is (p^2 - 3p + 3)/(1 - p)^2.
  2. *3: But what happens if the packet is dropped?*
  3. Since there are three networks with two gateways in between them, assume it takes two successful hops to reach it(D) from it(S) (one from N sub 1 to N sub 2 and one from N sub 2 and N sub 3) and two successful hops to reach it(S) from D (reverse order). Since p is the probability of a successful transfer from S to it(D) or from it(D) to S. Let h equal the number of hops allowed before a packet is dropped or lost. 1 The mean number of hops from it(S) to it(D) is equal to h\*(1 - p) sup 2 such that the mean is greater than or equal to 2. It takes a minimum of two hops to reach it(D) from S. The formula described above is the total number of possible hops multiplied by the probability of a successful transfer from S to D. 1 The mean number of times a packet is sent from *S* is equal to h(1 - p) sup 4 such that the mean is greater than or equal to 4. It takes at least two hops to reach *D* from it(S) and another two hops fro the receipt packet to reach it(S) from *D*. The formula described above is the total number of possible hops multiplied by the probability of both a successful transfer from it(S) to it(D) and a successful transfer from it(D) to it(S) (the receipt). 1 If the homework question is correct, then it appears to be the same as part 1 of this question. A packet that is received by it(D) will always be a packet that was sent by it(S) in this scenario. If it is an error, the assumption is that the question is looking for the mean number of hops made by a packet sent by D. The only time it(D) will “resend” a packet is if it(S) does not receive D's receipt packet, resends the initial packet for which the receipt packet was sent, it(D) receives this packet (again), and it(D) sends another receipt (the assumption being that the new receipt counts as a resend of the original receipt message sent from D). the probability then remains the same as part 1 anyway since it still takes two successive successful hops to reach it(S) from D; therefore, the probability would be h\*(1 - p) 2. Source: Statistics Course at SJU during undergrad.
  4. \* Hops made by a packet sent by *S*? A hop accrued each time that packets are passed to next router or gateway. So, there are 2 gateways (2Gs), also if each network transit has a hop, so; there are 3 networks (3Nets). Then, the average of numbers of hops made by a packet sent by it(S) is 2 + 3 = 5 hops. \* Times a packet is sent (initial + retransmissions) by S? it(S) will send a packet only one time is an initial, and it keeps retransmitting (n) times until it(D) gets that packet. Then it(S) will stop retransmitting to D. The average of numbers of times for a packet sent to it(D) by it(S) is = (initial) time \* (n) times of retransmissions. \* Hops made by a packet received by D? There are 2 gateways. Also, if each network transit has a hop, so, there are 3 networks. Then the average of numbers of hops made by a packet sent to it(D) is 2 + 3 = 5 hops. Reference is (www.en.wikipedia.org, Hop Count)

1. Design a reliable, pipelined, data transfer protocol that uses only negative acknowledgments. How quickly will your protocol respond to lost packets when the arrival rate of data to the sender is low? Is high? Justify your answers.

The difficulty with NACKs is they don't (shouldn't) occur often, and they don't mean much when they do occur. To provide a useful protocol, the receiver should NACK in such a way as to provide extra information. The sender can pipeline by sending packets in sequence at a regular interval. The usual reliability concerns can be dealt with using sequence numbers and time-outs. Because the problem's concerned with reliability and not efficiency, we can assume the transmission interval gets set and maintained using some mechanism, probably based on NACK arrival rates. The first problem is buffering at the sender. In the absence of other information, the sender has to buffer all the data sent because the receiver can NACK any packet. If the receiver adheres to the rule that it only NACKs the earliest (lowest sequence number) missing packet, then the sender can assume on receiving NACK *i* that all packets before *i* have been received. Although workable, this solution is not perfect because if the receiver never NACKs packets, the sender never clears the buffer. When the sender receives a NACK, it schedules a retransmission for the missing packet. In the presence of heavy packet loss, the protocol degenerates into stop-and-wait. Depending on packet-loss behavior, it might be useful to change NACKs to start plus run so groups of missing packets can be retransmitted on one NACK, but that's for efficiency, not reliability. The other problem is ending the transmission. Transmission end can be handled with NACKs (more or less) by having the sender end with an end-of-transmission (eot) packet. When the receiver gets an eot packet with sequence number *i*, it NACKs packet *i* + 1. When the sender gets NACK *i* + 1, it knows the receiver has all the data, and the transmission is at an end. The protocol response to lost packets depends entirely on rate at which NACKs arrive at the sender, it has nothing to do with the rate at which the sender receives data to send.

The answers, approximately verbatim, and in no particular order:

* 1. Since the protocol will only implement NACKs (negative acknowledgments), it will send packets into the pipeline until the receiver is nearly pushed to its capacity (as discussed in class 10/02/2012). It will then wait for an allotted amount of time (a grace period) before it continues to send packets into the pipeline. This allocated time is used to obtain and handle NACKs sent from the receiver should one of the packets not reach it (the receiver). If the sender receives a NACK, it will resend the corresponding packet. This system ensures reliability. This protocol will respond slowly if the arrival rate of data to the sender is low because it will take more time to receive the NACK and take the corresponding action; conversely, it will respond quickly if the arrival rate is high since the receiver will quickly issue a NACK that returns to the sender promptly within the retransmission time. [ picture ]
  2. The protocol response to the lost packet when the arrival [??] of the data to the server is lost is also lost and the protocol[?] response is the lost packet when[?] the arrival rate of the data to the sender is high is also high i.e. it is quick because when a sender sends a packet to the receiver and the packet is reached[?] then the receiver gets an acknowledgment and sends another packet and when the sender sends a packet and the packet is lost and the receiver sends a negative acknowledgment i.e. NACK the receiver waits the packet is obtained and when it is got then the [??] starts[?]. Hence the response rate is [??] as the arrival rate of the data to the sender.1

*1: What does this paragraph say? Is there a protocol described in here?*

* 1. In a protocol that using only negative acknowledgments, the loss of packets is only detected by the receiver when it receives it(x) - 1 and then it(x) + 1, only when it(x) + 1 is received makes the receiver realize that it(x) was lost. When the arrival rate of data to the sender is low then there will be a delay in receiving NAKs for lost packets by the sender. So, there will be a delay in sender's response, i.e., delay in retransmission. If there is a long delay between the transmission of it(x) and the transmission of it(x) + 1, then it will be a long time until it(x) can be recovered, under a NACK only protocol. On the other hand, if arrival rate of data is high, then recovery under a NACK-only protocol could happen quickly. Moreover, if errors are infrequent, then NAKs are only occasionally sent (when needed), and ACKs are never sent, a significant reduction in feedback in the NACK-only case over the AC-only case. Reference: \ Kurose and Ross textbook
  2. With a reliable data pipelined data transfer protocol that uses only negative acknowledgment it would take longer for the protocol to respond to lost packets when the arrival rate is low. The lost of packet *x* is only noticed by the receiver when packet *x* + 1 is received. However if data is sent more frequently then recovery could happen quickly.2

*2: What is the protocol being described?*

* 1. As known, in the pipelined reliable data transfer protocol the sender can start sending a second data packet before the sender receives the acknowledgment for the first data packet.3 Thus, if the sender needs to send many packets, then the time until the last of the packets is sent will be shorter with a pipelined protocol. Also, errors can occur when a pipelined protocol is used in two ways, which are losing or corrupting data and losing or corrupting acknowledgments. \* The pipelined reliable data transfer protocol will respond to lost packets by using the (Go Back N) which is the sender transmits ALL the data packets it had sent since the lost or corrupted data packets because when the arrival rate of data to the sender is low, that means there is much lost or corrupted data packets. Thus, there is a need to retransmits all lost or corrupted data packets to save time. [ picture ] \* The same idea for lost or corrupted ACKs.4 \* The pipelined reliable data transfer protocol will respond to lost packets by using the (Selective Repeat) which is the sender JUST retransmits the lost or corrupted data packet because when the arrival rate of data to the sender is high, that means there is not much lost or corrupted data packets. Thus, there is no need to retransmits all lost or corrupted data packets to save time. [ picture ] \* The same idea for lost or corrupted ACKs. Reference is (Computer Networking a Top Down Approach, pages 203--211).

*3: Is that pipelining, or using negative acknowledgments?*

*4: How are lost acks, or NACKs, detected?*

* 1. A pipelined data transfer protocols is based on the idea of sending multiple packets by the sender without waiting for acknowledgment after each packet. Instead, the sender is allowed to send number of packets before waiting for an acknowledgment. As a result, the range of sequence numbers must be increased and packets have to be buffered at least at the sender side. A sample of pipelined transfer protocol is presented in figure 1. Pipelined protocol provides reliability which means that there is no loss in data, corrupted data, or unordered data. (Kurose and Ross. Computer Network: A Top-Down Approach. Fifth edition) [ picture ] Steps of designing a pipelined data transfer protocol and they would be integrated with Figure 1. Version 1 Sender: [ picture ] packet = make(data) & send(packet) Receiver: [ picture ] data = extract(packet) & deliver(data) Version 2: Acknowledgment We implement techniques like checksum to ensure that the data were transmitted without error by comparing the umber of bits that were received by the receiver with the numb er of bits that were sent by the sender. Sender: [ picture ] If NACK? \ Send again Receiver: [ picture ] If packet is received & checksum = True \ Then, Extract data & deliver data & send ACK Pipelined protocol deals with lost packet with two approaches: 1 Go-Back\_N 1 Selective Repeat In Go-Back-N approach, the sender sends packets up to the lost packet again to the receiver. This makes the network suffers of performance problem when the network is slow. In selective repeat, the sender sends specific packets that were lost or corrupted. This reduces redundancy and fasts the data transform rate. When we have slow arrival rate of data, it is better to apply selective repeat approach to avoid overwhelming the network. Otherwise, using Go-Back-N is not a problem when we have fast data arrival.
  2. In this type of protocol,5 if the receiver doesn't receive the packet from sender, it sends a NACK, and when the sender receives it, it sends a new packet.6 If the arrival of NACK is slow the sender waits until it receives it. So when it is low, the protocol response is also low. When it is high, the sender also responds quickly.

*5: What type of protocol? Where is it described?*

*6: A new packet? What about the packet that wasn't received?*

This page last modified on 2012 October 15.

**Computer Networking**

**Homework 5, 20 November 2012**

This homework assignment has five questions; answer all of them. This assignment is due no later than 7:00 p.m. on Thursday, 29 November.

If you mail in your assignment, please submit a printable document — a PostScript .ps or PDF .pdf document, for example — and not a source document — a Word .docx or Latex .tex document, for example.

1. Radio antennas often work best when the diameter of the antenna is equal to the wavelength of the radio wave. Reasonable antennas range from 1 cm to 5 meters in diameter. What frequency range does this cover?

Re-arrange λ*f* = *c* to get *f* = *c*/λ. From *c* = 3×108 m/sec and λ = 3×10-2 m/cycle get

(3×108 m/sec)/(3×10-2 m/cycle) = 3 × 1010 m/cycle = 30 GHz.

From λ = 5 m/cycle get

(3×108 m/sec)/(5 m/cycle) = 60×106 m/cycle = 60 MHz.

The answers, approximately verbatim, and in no particular order:

* 1. Speed of Light = Wavelength\*Frequency

To solve for frequency: \ Frequency = Speed of Light/Wavelength (in nm) \ Speed of Light = 3 \* 10^8 m/s

1 cm = .01 m

(3\*10^8)/100000001 = 30 MHz \ (3\*10^8)/5 = 60000000 Hertz\*1.0 × 10^-6 = 60 MHz

60 MHz - 30 MHz

*1: This is not the right value for 1 cm.*

* 1. Frequency, f = c/l where c is speed of light, i.e., 3\*10^8 m/s \ Frequency of radio wave with 1 cm (i.e., 0.01m) diameter is f = c/l = 3\*10^8/0.01 = 3 × 10^10 = 30 Ghz \ Frequency of radio wave with 5m diameter is f = c/l = 3 × 10\*8/5 = 6 × 10^7 = 60 MHz \ The cover range is from 60 MHz to 30 GHz. \ Reference: Class Notes
  2. Frequency = the speed of light/wavelength. Based on that formula (and keeping wavelengths in meters), the minimum frequency would be 299,92,458 m/s / 5 m = 59,958,491.6 Hz, and the maximum frequency would be 299,792,458 m/s / .01 m = 29,979,245,800 Hz. The range is approximately 60.0 MHz to 30.0 GHz.

Source: HTTP://www.tomcarpenter.net/2009/11/21/rf-wavelength-calculations-for-wireless-networks/

* 1. Freq = C/W where C: speed of light. W: wavelength. \ Convert 1 cm to m → 1 cm = 0.01 m

For diameter of 1 cm: \ Freq = (3 × sup(10, 8)/0.01 \ Freq = 3 × sup(10, 10) = 30 GHz

For diameter of 5 m: \ Freq = (3 × sup(10, 8)/5 \ Freq = 6 × sup(10, 7) = 60 MHz

The cover range is from 60 MHz to 30 GHz.

* 1. Given, \ Diameter of the antenna = wavelength of the radio wave i.e. 1 cm to 5 m \ We know that,

Frequency(f) = speed of light(c)/wavelength(l)

Now for 1 cm, f = c/l = 3 × 10^8 / 0.01 = 3 × 10^10 = 30 GHz Now for 5 meters, f = c/l = 3 × 10^8 / 5 = 6 × 10^7 = 60 MHz Hence the frequency range that these radio antennas cover are from 60 MHz to 30 Ghz.

1. Give examples of non-electrical systems that can be considered a) simplex, b) half-duplex, c) full-duplex and d) none of a, b, or c. Justify your answers.

a) A river rapids is simplex because transport can only go downstream. b) A canal is half-duplex because ships can go up-grade or down-grade, but only one way at a time. c) A wide river is full-duplex because it supports simultaneous two-way traffic. d) A water fall is none of these because it supports no (useful) transportation at all.

The answers, approximately verbatim, and in no particular order:

* 1. Simplex: One-way streets are simplex because they allow traffic in only one direction with no change in direction allowed.

Half-duplex: Street Lights with delayed green lights (such that one side is green first, then it switches to red while the opposite side becomes green) is an example of a half-duplex system because it allows traffic to flow both ways (e.g. across the highway), but only one way at a time.

Full-duplex: Regular traffic lights are full-duplex because when the light is green, traffic can flow from both directions at the same time (e.g., crossing the highway at the same time).

None of the above: A traffic circle is none of the above because it allows for multiple directions to be traveled simultaneously with both inputs and outputs; alternatively, a road closed for maintenance is none of the above because there is no traffic flow in any direction.

* 1. a) A simplex network is a service that provides one-way communication. A non-electrical system for the simplex is the radio broadcasting1 where the radio device obtains the signals from the broadcasting source.

b) A half-duplex network is a two-way communication but cannot communicate simultaneously, they can communicate one at a time in one direction. An example for this is a walkie-talkie where when one person uses the device the other person has to just hear and vice versa.2

c) A full-duplex communication is a two-way communication achieved over a physical link that has the ability to communicate in both directions simultaneously. A example for this is a telephone line which has two lines one for transmitting and the other for receiving which means both the persons can talk and listen at the same time.

d) Examples where the systems don't use simplex half-duplex and full-duplex are repeaters and modems.

*1: How is radio broadcasting non-electrical?*

*2: Non-electrical?*

* 1. - Simplex Transmission

Data in a simplex channel is always one way. Simplex channels are not often used because it is not possible to send back error or control signals to the transmit end.

Examples of simplex channels: In tankless water heater, water goes in one direction inside the heater. In a one-way street, vehicles move in one direction.

- Half-Duplex Transmission

A half-duplex channel can send and receive, but not at the same time. Here we need to switch between transmit and receive mode. It is like a one-lane bridge.

Examples of half-duplex channels: In a single railroad track, trains can move either forward or backward at a time.

- Full-Duplex Transmission

Data can travel in both directions simultaneously. There is no need to switch from transmit to receive mode like in half-duplex. It is like a two-lane bridge.

Examples of full-duplex: In a two-lane road, two vehicles can move in both directions at the same time.

- Any other system goes under one of these categories.

(Tanenbaum, Computer Networks, chapter 2)

* 1. a) Simplex example is speech because the person who is giving speech is the only one who talks and all others listen to it.3

b) Half-duplex example is satellite phones example Iridium operates a one-way pager service as well as the call alert feature.

c) Duplex example is cellular phones because both sides can talk at the same time.

d) [ not answered ]

*3: How polite you are.*

* 1. A one-way street can be considered simplex because cars can only go one way on it, there is no chance of going the other way. A bus can be considered half-duplex transmission, people can get on or off the bus but not at the same time; people get off, and then people get on. A two-way street can be considered full duplex because cars can come and go at the same time. A closed street would be none of a, b, and c because no cars can come or go.

1. Three networks *A*, *B*, and *C* each have *n* nodes. The best, average, and worst case transmission paths in hops in each network is given by

|  |  |  |  |
| --- | --- | --- | --- |
| **network** | **best** | **average** | **worst** |
| *A* | 2 | 2 | 2 |
| *B* | 1 | *n*/2 | *n*/2 |
| *C* | 1 | 1 | 1 |

1. Describe the topology of each of the networks. Justify your answers.
2. Network *A* is a star network with one hop going to the hub and the other hop to the destination. Network B is a bi-directional ring network. Network C is a fully connected network.
3. The answers, approximately verbatim, and in no particular order:
   1. Network *A* could be described as a linear topology consisting of three nodes *X*, *Y*, and *Z*1 (and therefore, two hops from *X* to *Z* via *Y*) linked together in a line. It would always take two hops to travel from *X* to *Z* (or from *Z* to *X*) since y is the intermediate node.

Network *B* could be described as a tree where the best case is one hop to an ancestor/descendant and the worst case is traveling from the root to a leaf node (n/2 hops).2

Network C could be described as a central hub surrounded by nodes in a bicycle-wheel pattern where the nodes are linked with the central hub with spokes (the spokes being the network path). It would take one hp from the center to any node in the network.

*1: It could, but*n*is arbitrary.*

*2: But trees aren't linear; they're logarithmic.*

* 1. Three networks each contain ‘n’ nodes. \ The first is star topology with a central router because in a start topology, transmission path from one node to other node will be ‘2’ in all cases (through central router), the second is a bidirectional ring because in bidirectional ring topology, transmission path from one node to other node is ‘1’ if nodes are adjacent and ‘n/2’ if nodes are far away, and the third is fully interconnected (every node is connected to every other) because in a fully connected topology, the transmission path from one node to other is ‘1’ because all nodes are connected to all other nodes.

star - 2, 2, 2 hops (best, average, worst) \ ring - 1, n/2, n/2 \ full - 1, 1, 1

* 1. [ not answered ]
  2. First case, \ Now the first network is having a path equal to 2. Hence the packet goes from the originating computer to the switch, then from the switch to the destination computer. During this process 2 hops take place.3

Second case, \ The second will have a best path equal to 1 and a worst path equal to n/2 and an average of (n/2)/2. Hence the worst path is equal to n/2 because the farthest a packet will have to travel is equal to half the total nodes because it is a bidirectional ring. Hence we assume the average.

Third case, \ Now the third which is an interconnected network will have a best, average and worst case of 1. Because all of them are connected to each other, there is no hopping at all and each computer can connect to any other computer directly.

*3: What's the network topology?*

* 1. A 2 2 2

This represents a star topology.

In this network, each node is connected to other nodes via central node called hub or switch, and a node gets to any other node in this network via the hub or switch.

B 1 n/2 n/2

This represents a ring topology.

In this network, each node is connected to two other nodes in the same signal path, and the path forms a circle.

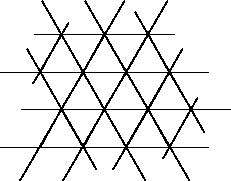
C 1 1 1

This represents a fully connected topology.

In this network, each node is directly connected to all other nodes.

(source: http://computer.howstuffworks.com)

1. A mobile phone system with equilateral triangular cells



cannot share a frequency band between edge-adjacent cells. Devise a scheme to allocate 840 frequency bands in such a system so that each cell has the maximum number of frequency bands available.

Consider a cell *C* and its three surrounding neighbors. Each frequency used in *C* is unavailable for its neighbors, and each frequency used by a neighbor is unavailable to *C*, but is available to *C*'s other neighbors. By this argument *C* gets half the frequencies and each of its neighbors get the other half. Because this argument doesn't distinguish among the neighbors, it can be repeated endlessly on the plane.

The answers, approximately verbatim, and in no particular order:

* 1. Take six triangular cells around a point as a hexagonal group of cells. Every other cell should contain a frequency band.1 Repeat this pattern for all neighboring groups of cells throughout the network. Once the network is populated with these neighboring groups, each empty cell will have access to three frequency bands.2

*1: The same frequency band?*

*2: Just three frequencies per cell? Is that the maximum?*

* 1. [ not answered ]
  2. In the following figure, each cell has three neighbors. When a cell uses frequency of group A, its two neighbors can use frequency of group B (as shown below). Consequently, only two group of frequencies are needed, and number of frequencies that can be used will be: \ frequency bands/number of group \ 840/2 = 420 frequencies (http://searchnetworking.techtarget.com)

[ diagram ]

* 1. Each cell has three other adjacent cells, in a triangular grid. \ So it has to avoid any frequency that any of these cells use. \ This pattern can repeat, and cover the plane endlessly, so these three other cells have to avoid each of the others, but no more. \ The above figure makes this clear, and shows that there are for “kinds” of cells. For each cell to have the same number of frequencies (so that a user has the same sort of experience no matter what cell they are in). \ Hence they all have to have the same number of frequencies, i.e. thus

840/4 = 210

The obtained answer is from the assumption that no two cells that are adjacent to the same cell can re-use frequencies.3

*3: Does this provide each cell with the maximum number of frequencies?*

* 1. There can be at least three frequency groups *A*, *B*, and *C*. Dividing the total available frequencies by three gives us 840/3 = 280 frequencies for each cell.4

*4: But is that a maximum for each cell?*

1. A PCM system has been set to sample at 125 μsec. Explain why; be specific as you can.

Because 125 usec is 1/8th of a msec, which allows a 1 byte/msec (or a 1k byte/sec) sample rate.

The answers, approximately verbatim, and in no particular order:

* 1. This corresponds to a sampling rate of 8000 samples/sec. According to the Nyquist theory, this is the sampling frequency needed to capture all the information in 4 kHz channel. (HTTP://searchnetworking.techtarget.com)
  2. A sampling time of 125 usec corresponds to 8000 samples per second. According to Nyquist theorem, this is the sampling frequency needed to capture all the information in a 4 kHz channel, such as a telephone signal. \ Reference: Tanenbaum and Wetherall
  3. A PCM system digitally represents analog signals. Analog signals are digitized by codecs and the codecs make 8000 samples per second which is equal to 125 usec. The Nyquist theorem states 125 usec is enough to capture all the information from the 4 kHz telephone bandwidth.

Source: Computer Networks - Tanenbaum and Wetherall

* 1. Given a sampling time of 125 usec,

which corresponds to 8000 samples per second.

Now according to the Nyquist theorem which states that,

IF a function *x*(*t*) contains no frequencies higher than *B* Hertz. It is completely determined by giving its ordinates at a series of points spaced 1/(2*B*) seconds apart.

Hence this is the sampling frequency that is needed to capture all the information in a 4-kHz channel, such as a telephone channel. Actually the nominal bandwidth is some what less, but the cutoff is not sharp.

* 1. The Nyquist Sampling theorem ("The minimum data rate D (in bps) of a communication channel (Channel capacity) with bandwidth B where we can recognize K levels in the signal is D= 2B log2 K) shows how to sample the analog signal in PCM. It says that it's possible to reconstruct a signal from 2B samples. If a telephone channel is used (4 kHz), then 8000 samples are needed (2\*4000 Hz). Applying the formula with 256 amplitude levels yields a data rate of one byte every 125 usec. HTTP:www.ics.temple.edu/~giorgio/cis307/readings/datatrans.html

This page last modified on 2012 December 3.