Module 3- Network Systems Notes

Puru Vaish
Department Of Technical Computer Science

August 31, 2020

1 Chapter 1: Foundation

1.1 Applications

1. VOIP- Voice Over Internet Protocol
2. Streaming
3. HTML pages, i.e. Surfing the world wide web
   Downloading pages from the internet.
4. Video Calls

1.2 Requirements of the Applications

This section is about why are networks made the way they are, and not why they should be done in only one particular way. As time and needs progress the way networks work may change.

1.2.1 Shareholders

First, we shall look at the requirements ”from” a network from three different perspectives of shareholders.

1. Application Programmer: the services that his or her application needs: for example, a guarantee that each message the application sends will be delivered without error within a certain amount of time or the ability to switch gracefully among different connections to the network as the user moves around.
2. Network Operator (person who is in charge f debugging a network, configuring a network): These people in real life are likely IT managers in businesses who make sure the ’system network’ is not crashing. Hence, they would want a network which is easy to isolate faults in, add/remove systems, account for usage (all the people who use streaming services can be listed).
3. Network designer: Efficiency, usability, performance are the key topics here.

1.2.2 Scalability

Networks in general should be scalable, i.e. it should be possible to add more nodes to the network without affecting the network, for example, having to close down the network temporarily, or even without a drop of performance, and not effecting other users while removal or addition.

1. BAD:
   (a) Bus Networks- Too usch stress on the main line, when more users are added
   (b) Ring- Can not add more users without disturbing other users, or taking down the system, unnecesary traffic onto adjacent connections.
   (c) Mesh- Too many connections, tht are likely not being used, too much cabling required, not possible to connect 1000 nodes like this, simply not enough ports if you think about it.
2. GOOD:
   (a) Star- at least as a component.
   (b) P2P- not without it's own problems.

Notice in the GOOD section, we have network topologies that do have some problems as well, but are more easily scaled than the previous ones.

General Note: Two nodes need not have a direct link between them, but even an indirect link through several nodes. In such a scenario it is easy to see how using existing connections in start network or P2P, outweigh something like Mesh, or Bus.

1.2.3 Cost Effective Resource Sharing

Hosts on a network have to share its resources, the resource we are most interested in is the data transmission medium (usually a copper cable, optic fibre or radio waves), this sharing is called multiplexing, which means that a system resource is shared among multiple users. At an intuitive level, multiplexing can be explained by analogy to a timesharing computer system, where a single physical processor is shared (multiplexed) among multiple jobs, each of which believes it has its own private processor. Similarly, data being sent by multiple users can be multiplexed over the physical links that make up a network.

There are different ways of sharing resources under two different schemes:

1. Circuit Switching
   (a) What happens in this scheme:
      i. One node pings the destination node, possibly using some hello message.
      ii. If no response, waits, else proceeds to establish a connection according to the protocol AND (MORE IMP) fixes a route between the two nodes by fixing hops between the routers of the two nodes. (Since this path is fixed it can NOT be shared by other users of the network).
      iii. The users transmit data along this dedicated route.
      iv. packet receival is guaranteed unless a calamity happens (routers between hops gets disconnected, or the users themselves are disconnected).
      v. the connection is torn down, once 'bye' is done.
   (b) Why Bad?
   In reality, the two computers may not be communicating all the time. So, say they are connected, and so the router had to fix this route which takes 25% of the bandwidth of the path. If they do not communicate this is all wasted.
   The way the resource is allocated in this scheme are:

i. STDM: Synchronous Time Disvision multiplexing.
   The name is self explanatory, according to the number of users who use the link, it divides the time axis by the number of connections and then sends the data over those time periods. This is similar to Round-Robin approach where each connection between the nodes needs to wait it's turn.

ii. FDM: Frequency Division Multiplexing.
   The name is again self explanatory. Instead of diving the time axis, the frequency axis is divided (think of radio waves of different frequencies). Each connection is then given a frequency through which the data is then sent which is distinct from the other frequencies (so that they do not interfere in terms of data being carried) and then the data is then delivered.

(c) Nether of schemes solve the problem of the fact the path is held busy even when there are no transmissions and hence leads to an upper bound of the number of connections the connection can handle before people have to start waiting for their turn. For example: if each connection 'requires' (application requirement alert) to send at a speed of 1Mbps on a 10Mbps link but only transmits at a probability of 10% the connection can handle at best 10 nodes at the same time due to the link capacity, otherwise it may have to drop packets, store them, etc.

2. Packet Switching

   (a) What happens in this scheme (Also the description of statistical multiplexing):
      i. The packet is prepared by the computer system.
      ii. The packet route is then simply resolved, which means the path it needs to take to the destination is determined.
      iii. each packet can take a different route.
      iv. It then reaches the destination after getting routed by each router.
      v. If the destination is unavailable, that is okay, it can try again, else the message id delivered, and the destination computer takes care of all the packets arriving.
      vi. notice, the route taken is never fixed, and never kept busy for this particular connections.

   (b) Why Good?
   It allows sharing of resources. Consider the above example again, this time 35 users can access the same system the entire time, since the probability that more than 10 users are active at the same time is actually less than 0.004. Allowing more efficient data bandwidth usage, as it supports greater variety of connections. (Notice: we do not say greater number of maximum connections, as that is still 10, but in circuit switching only 10 different permutations of connections are possible in a given time frame, while in packet switching 35 choose 10 connections are possible in a time frame (not instantaneously) as the connection is blocked for the first 10 only).

   Statistical Multiplexing is similar to STDM, only the time slots are not predefined, but on-demand. Also it determines who to transmit from similarly to STDM in a round-robin manner, such allocation of bandwidth is used to call such networks having Quality of Service (QoS).

Even in switching there are two types on hardware level:

1. Store-and-Forward: Store-and-Forward switchingwill wait until the entire frame has arrived prior to forwardingit. This method stores the entire frame in memory. Once the frameis in memory, the switch checks the destination address, sourceaddress, and the

CRC. If no errors are present, the frame isforwarded to the appropriate port. This process ensures that thedestination network is not affected by corrupted or truncatedframes.

2. Cut-Through: Cut-Through switching will beginforwarding the frame as soon as the destination address isidentified. The difference between this and Store-and-Forward isthat Store-and-Forward receives the whole frame before forwarding.Since frame errors cannot be detected by reading only thedestination address, Cut-Through may impact network performance byforwarding corrupted or truncated frames. These bad frames cancreate broadcast storms wherein several devices on the networkrespond to the corrupted frames simultaneously.

The network obviously needs to ensure the proper packetisation of data, to ensure just one connections hogs the entire bandwidth and the rest of the connections end up getting nothing.

This does not mean that Circuit Switching is useless though, in some instances it may be more essential to have a dedicated connection rather than efficiency of resource sharing, but then networks likely need to be catered for such applications.

1.2.4 Support for Common Services

Think of WhatsApp, Messenger, Viber(?). They are all text messaging applications, or at least have that as one of their functions. The network needs to ensure that one instance of any of those application can communicate with another instance of the application. THis could be implmented all in the application itself, from protocol to creation of bits, but if they are all going to be doing the same tasks, (ask for authentication, generate bits using the same scheme, have IP addresses and so on) it makes sense to put it rather in the network implmentation. This is where layering comes in. The challenge is to find those abstractions which allow the greatest range of applications to be implemented, without constraining the Application Developer (If they had ocntrol of all parts they could do anything they wanted, the fact you take things away from their control may restrict them).
For example, does the application require a guarantee that messages sent over the channel are delivered, or is it acceptable if some messages fail to arrive? Is it necessary that messages arrive at the recipient process in the same order in which they are sent, or does the recipient not care about the order in which messages arrive? Does the network need to ensure that no third parties are able to eavesdrop on the channel, or is privacy not a concern? In general, a network provides a variety of different types of channels, with each application selecting the type that best meets its needs. The rest of this section illustrates the thinking involved in defining useful channels.

1.2.5 Identify Common Communication Patterns

How different are two applications. FTP/NFS they both transfer files but just in different ways, the entire file or in blocks. The underlying technology of sending the file (data) is not different. The network need to be concerned with the specifics but only the similarity.

1.2.6 Reliable Message Delivery

The causes why the message may not be delivered as sent/requested (apart from hacker activity)

1. Packet may switch bits over the link: noise, radiation. How can flip the bit maybe? Does it need to be discarded.
2. Packet itself may not arrive: queue, dropped. How to determine packet is lost and not just late? Sent to wrong connection.
3. The node and link layer may lose connection: Power Outage, mis-configuration of the network, reckless operator.

1.2.7 Manageability

Network system need to be manageable. This includes:

1. How to keep track of addresses?
2. Track of usages?
3. Troubleshoot in case of problems
4. Change to new equipment, add functionality.

A general ideology is used to keep away when it works, which promotes a slow progress in features available as stability is more focused upon.

1.3 Architecture

1.3.1 Layering and Protocols

Layering provides two benefits:

1. First, it decomposes the problem of building a network into more manageable components. Rather than implementing a monolithic piece of software that does everything you will ever want, you can implement several layers, each of which solves one part of the problem.
2. Second, it provides a more modular design. If you decide that you want to add some new service, you may only need to modify the functionality at one layer, reusing the functions provided at all the other layers.

Layering allows for different methods of communication. For example:

1. Request/Reply Protocol
2. Message Stream Protocol

Why? Because protocol is nothing but abstracted objects that make up the layers, which tell the higher layers how the data is to be handled, how the data was formatted, and not what the actual data is. But, how is it seen computationally by a a device? Each protocol defines two different interfaces:

1. First, it defines a service interface to the other objects on the same computer that want to use its communication services. This service interface defines the operations that local objects can perform on the protocol. For example, a request/reply protocol would support operations by which an application can send and receive messages. An implementation of the HTTP protocol could support an operation to fetch a page of hypertext from a remote server. An application such as a web browser would invoke such an operation whenever the browser needs to obtain a new page (e.g., when the user clicks on a link in the currently displayed page).
2. Second, a protocol defines a peer interface to its counterpart (peer) on another machine. This second interface defines the form and meaning of messages exchanged between protocol peers to implement the communication service. This would determine the way in which a request/reply protocol on one machine communicates with its peer on another machine. In the case of HTTP, for example, the protocol specification defines in detail how a GET command is formatted, what arguments can be used with the command, and how a web server should respond when it receives such a command.

But these details are not important, what is important is to understand that in either of these two methods of sending data, they are not any different for lower protocols, only on higher protocol on both receiving and sending side. Diving up the stack in these abstract entities allows other protocols like HHP (Host to Host protocol) which acquires connection for the hosts, or SIP (Session initiation Protocol) which acquires the knowledge of the contacted peer, Geographic, capability, availability, session setup and session management to be achieved.

To further understand how layering is actually implemented despite so many protocols, we look at encapsulation.

1.3.2 Encapsulation

The only thing to get in this subsubsection is what are headers. Application Layer collects data to be sent. Transport Layer packetises everything and adds a header $\rightarrow$ giving each packet a dest. and src. port packet number, checksum and relevant data in the header. The network protocol adds IP addresses, tolive, IP version, frgamentation limit and so on it its header, and so on for each protocol that is encountered. This adding of headers upon headers is the essence and what encapsulation is. The peer need not read the entire thing, strip of the header and pass along if the packet is of the node on the network.

1.3.3 Multiplexing and Demultiplexing

Since we are considering archtecture it is appropriate to think about how a network handles Multiplexing and Demultiplexing. Simple: Using keys. Each application would generate jeys, or even multiple keys either specific for that connection or some other arbitrary way and pass that along in the protocol being used (RRP or MSP), according to the application's implementation of the protocol (as applications care only about intra application communication, yes there is PC XBOX cross-platform gaming but those are special cases where they may use some sort of translators) and then communicate. These are called demux keys.

1.3.4 Internet Architecture

(We will not discuss the 5 layers of internet here: Application, Transport, Network, Link, Physical. These are talked about all the time anyway.)

Since internet is a part of the network, we shall give a brief of description of that as well. Internet as we know and use, has three different attributes:

1. First, as best illustrated by Figure 15, the Internet architecture does not imply strict layering. The application is free to bypass the defined transport layers and to directly use IP or one of the underlying networks. In fact, programmers are free to define new channel abstractions or applications that run on top of any of the existing protocols.
2. Second, if you look closely at the protocol graph in Figure 14, you will notice an hourglass shape—wide at the top, narrow in the middle, and wide at the bottom. This shape actually reflects the central philosophy of the architecture. That is, IP serves as the focal point for the architecture—it defines a common method for exchanging packets among a wide collection of networks. Above IP there can be arbitrarily many transport protocols, each offering a different channel abstraction to application programs. Thus, the issue of delivering messages from host to host is completely separated from the issue of providing a useful process-to-process communication service. Below IP, the architecture allows for arbitrarily many different network technologies, ranging from Ethernet to wireless to single point-to-point links.
3. A final attribute of the Internet architecture (or more accurately, of the IETF culture) is that in order for a new protocol to be officially included in the architecture, there must be both a protocol specification and at least one (and preferably two) representative implementations of the specification. The existence of working implementations is required for standards to be adopted by the IETF. This cultural assumption of the design community helps to ensure that the architecture's protocols can be efficiently implemented.

1.4 Software

1.5 Performance

Time for some calculation now.

1.5.1 Bandwidth and Latency

What is quick?

1. I can get big files quick- High Bandwidth (throughput)
   Depended on:
   (a) Size of the 'pipes' we use to send data.
       This is basically the transmission rate of the cable.
2. I can get a response quickly- Low Latency (delay)
   Depended on:
   (a) The physical distance between two points
   (b) the length of cabling

(c) the transmission method being used.
(All these things effectively correspond to the RTT (Round Trip Time))

Note: RTT not same as 2 x One way trip, we will see why. Although in many instances it can give a good approximation.

1.5.2 DBP- Delay Bandwidth Product

(Personal Note- In middle school we had this question which revolved around thinking of time as a length and the rate of flow as the width, so their product gave us the volume. If we talk about the units $m^3 s^{-1} * s = m^3$, i.e. the volume)

Using my personal note, I hope it is clear then why I say this Delay Bandwidth Product is actually the number of bits this pipe is carrying (amount of water in the pipe). The bandwidth is the flow rate, and the delay is the time the bits stay in the pipe for, hence you can not add more bits to the pipe in that frame since we are travelling at the max flow rate. Using this analogy we understand why the delay product is a good measure of performance.// This should not be confused with the DBP giving an indication on how good a particular network is in only one of the metrics. High BDP could simply mean a very high delay time, similarly a low DBP could mean a very low delay time. This serves kind of like a general view on how good something is, overall.

Let us look at some examples:

1.5.3 Q1

Handshake = 0.1s B = 1.5MBps L = 1000Kb RTT = 50ms Packet Size = 1Kb

Find total delay if sent continuously.

1. One Pipe = B*(RTT/2) 37500bits
2. number = L/One Pipe 218.45333 pipes
3. total Time = number*(RTT/2) 218.4533*25ms = 5.46133s
4. add = handshake time 0.1s
5. add the length of the last pipe, since the length is time 25ms
6. total delay = 5.586s

(Note we talking about bits, do not forget to convert from bytes to bits)

1.5.4 Q2

after every packet is sent we will wait for 1RTT. this is where we must understand the distinction between different delays, namely, Transmission Delay and Propagation delay.

1. how long does it take to transmit a packet: 1kb/B
2. how long does it take for the packet to propogate: RTT/2

3. how long do we wait: 1RTT
4. but since in the same time the packet has travelled to the other side we can say the RTT/2 is contained in the 1RTT.

Calc:

1. handshake = 0.1s
2. one packet sent = RTT + TD
3. 1000packets sent = 1000*one packet - RTT (since we do not wait for last packet ack)
4. pipe length = 0.025s (we need this even if only one bit is sent)
5. total = 55.536s

(No shortcut please, add everything slowly slowly).

1.5.5 Throughput calculations.

Throughput = TransferSize / TransferTime
TransferTime = RTT + TransferSize/Bandwidth

These ones can be a bit weird depending on the answer, you may not see why they chose to ignore some bits.

Q3) 12000bit packets are sent only after getting a 50byte ack from receiver on a link with three switches which add 10us for every pass and the link itself has a B of 100MBps.

1. transmission rate = 12000/100*10^6 = 120us
2. there are three switches so total 4 times on switch then on link (draw, please): 4*120us = 480us
3. each switch adds 10us, but don't forget on receiver we have to read too so +40us = 520us.
4. next packet not sent until 50byte ack is received. each transmission of ack takes 50*8/100*10^6 = 4us. Notice, this is done 4 times, and also it follows the same 10us 4 times thing since it goes through the same switches. So, +4*(10 + 4) = 56us
5. total = 576us.
6. Now how much data actually transferred = 12000bits (why not +50 bytes, cause that is ack).
7. How much time taken = 576us.
8. So actual throughput = 12000bits/576us = 20.833Mbps.

(this is an okay question once you know what is happening, you need to have an image in your head, and see all components work, links switches end receiver. the question itself is never clear sadly).

(no propagation delay specified for this question, but feel free to add that every time a packet travels over the Ethernet, which makes throughput even less.)

One last thing to consider is there may be times when transmission times are actually so low, that RTT ends up determining the delay entirely (especially when very high transmission cables are used).

2 Chapter 2: Direct Links

2.1 Encoding

Things we are worried about.

1. baseline wander: he receiver keeps an average of the signal it has seen so far and then uses this average to distinguish between low and high signals. Whenever the signal is significantly lower than this average, the receiver concludes that it has just seen a 0; likewise, a signal that is significantly higher than the average is interpreted to be a 1. The problem, of course, is that too many consecutive 1s or 0s cause this average to change, making it more difficult to detect a significant change in the signal.
2. clock recovery: The receiver derives the clock from the received signal—the clock recovery process. Whenever the signal changes, such as on a transition from 1 to 0 or from 0 to 1, then the receiver knows it is at a clock cycle boundary, and it can re-synchronize itself. However, a long period of time without such a transition leads to clock drift.

2.1.1 NRZ

Non-Return to Zero: basically high is 1, low is zero, nothing special. Solves nothing: long length of 0/1 possible.

2.1.2 NRZI

non-return to zero inverted: toggle if 1 on the clock, and if 0 then just there. Solves long length of 1's. Problem: what if long length of zeroes. (Solved using 4B/5B).

2.1.3 Manchester

Clock XOR NRZ data.
It doubles the transition rate of $1 \to 0$ and $0 \to 1$. The rate of transition is called baud rate. The bit rate is half of this. So in essence, the efficiency is half since the receiver is working twice the amount to get half the data. (This is the case only if you have 2 ”symbols” high and low. Having more allows more data to be transmitted, which allows bit rate to be even higher than baud rate, so efficiency is 2x or more. Look up QAM).

2.1.4 4B/5B and 8B/10B

Use 5 bits to represent a data of 4 bits, and 10 for 8. The mapping is used to eliminate the long lengths of 0's and 1's which result in baseline wander. Three consecutive zero bits only appear in normal data when a code ending with two 0 bits (2, E) is followed by a code beginning with a 0 bit (1, 4, 5, 6), so will always appear separated by multiples of the 5-bit encoded symbol length (and never separated by a single symbol) i.e. no code has more than 2 zero at the start and more than 1 zero at beginning. Then transmit using NRZI. Since only 1 extra bit sent, the efficiency is 80%.

2.2 Framing

How to know the bits collected by the adapter form a complete frame? Remember, the networks communicate with frames not series of bits.

2.2.1 Byte Oriented Protocol

Recent example of this is Point-to-Point Protocol (PPP). Two approaches to implement this.

1. Sentinel Characters: STX, ACK, ETX stuff like this are put for beginning of frame, ending of frame. What if they appear in text. We use Data Link Escape (DLE) character. In LaTeXwe preceed comment or mathematical operator using backslash to ensure not to go to math mode□,%. Called character stufing.
2. Include Count of number of bits, size of frame in the header. One danger with this approach is that a transmission error could corrupt the count field, in which case the end of the frame would not be correctly detected, leading to a framing error (more or less bytes accumulated).

The internet protocol uses Sentinel Characters with character stuffing. The PPP frame is negotiated using Link Control Protocol (LCP) which is a frame itself sent over in PPP frame.

2.2.2 Bit Oriented Protocol

The Synchronous Data Link Control (SDLC) protocol developed by IBM is an example of a bit-oriented protocol; SDLC was later standardized by the ISO as the High-Level Data Link Control (HDLC) protocol.

HDLC denotes both the beginning and the end of a frame with the distinguished bit sequence 01111110. This sequence is also transmitted during any times that the link is idle so that the sender and receiver can keep their clocks synchronized. Because this sequence might appear anywhere in the body of the frame—in fact, the bits 01111110 might cross byte boundaries—bit-oriented protocols use the analog of the DLE character, a technique known as bit stuffing:

1. 011111 is seen in text.
2. add 0: 01111100 $\rightarrow$ is the bit stuffing
3. on receivers side, when 5 bits are seen, check next bit. if 1 then end of frame, if 0 then stuffing so remove this, if 1 and next is also 1: 01111111: this is error so discard frame.

Both Bit/Byte Oriented Protocol have variable frame length due to their being stuffing. Next one is not variable length.

2.2.3 Clock Based Framing

A third approach to framing is exemplified by the Synchronous Optical Network (SONET) standard. SONET is very bid and can be covered in a book of itself. Also, SONET addresses both the framing problem and the encoding problem. It also addresses a problem that is very important for phone companies—the multiplexing of several low-speed links onto one high-speed link. (In fact, much of SONET's design reflects the fact that phone companies have to be concerned with multiplexing large numbers of the 64-kbps channels that traditionally are used for telephone calls).

read implementation from the book. Key Points:

1. Easy to synchronize,
2. allows multiple data rate, for each a different sized frame, better reception
3. N STS-1 make up STS-Nc frame where c stands for concatenated, done by interleaving N STS-1 frames to create a bigger frame if link allows. So precisely- STS Nc is N*STS-1 frames, it is quantized.
4. has overhead, and points to starting of the other, so in case frames are shifted for whatever reason, the receiver can resynchronize.

- All frames (STS formats) are 125 µsec long
- Problem: how to recover frame synchronization
  - 2-byte synchronization pattern starts each frame (unlikely in data)
  - Wait until pattern appears in same place repeatedly
- Problem: how to maintain clock synchronization
  - NRZ encoding, data scrambled (XOR'd) with 127-bit pattern
  - Creates transitions
  - Also reduces chance of finding false sync. pattern

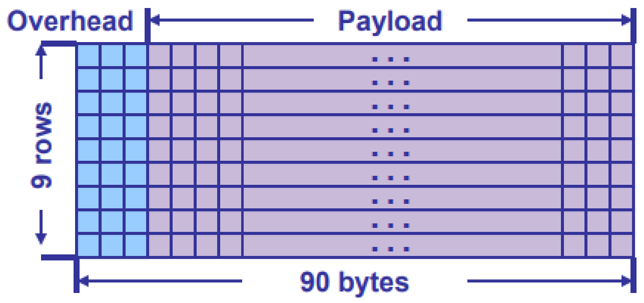
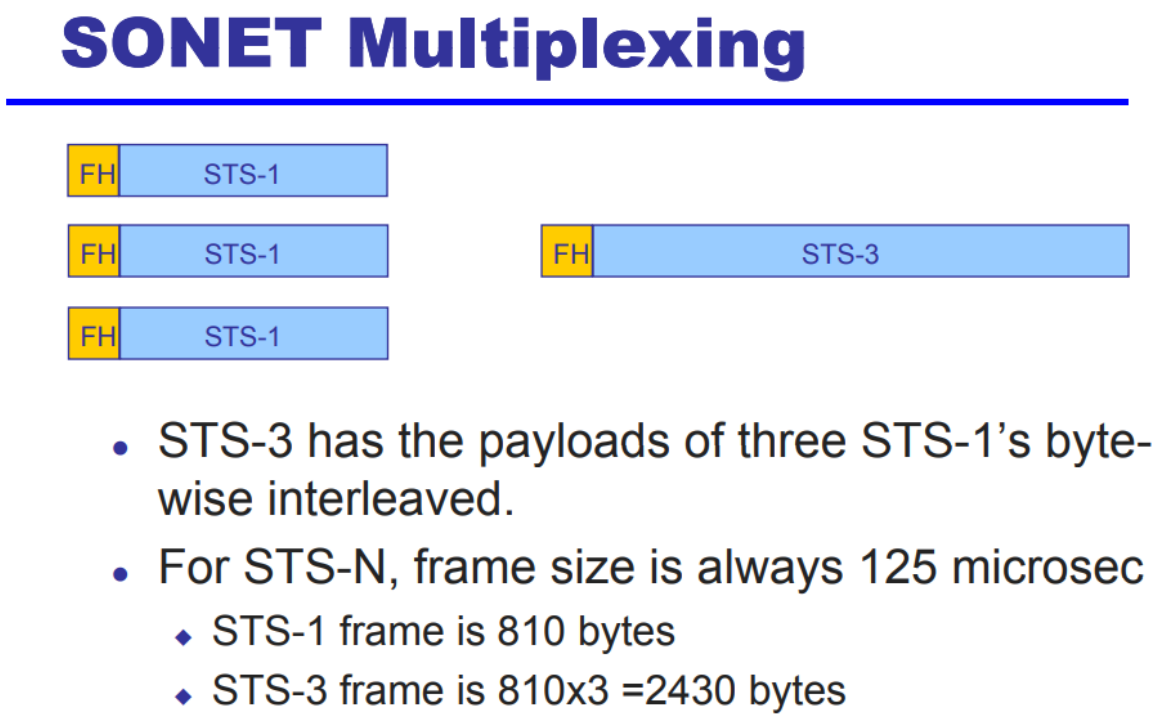


Figure 1: Caption



SONET Multiplexing

FH STS-1

FH STS-1

FH STS-1

FH STS-3

- STS-3 has the payloads of three STS-1's byte-wise interleaved.
- For STS-N, frame size is always 125 microsec
  - STS-1 frame is 810 bytes
  - STS-3 frame is 810x3 =2430 bytes

Figure 2: Caption

- STS-1 merged bytewise round-robin into STS-3
- Unmerged (single-source) format called STS-3c
- Problem: simultaneous synchronization of many distributed clocks

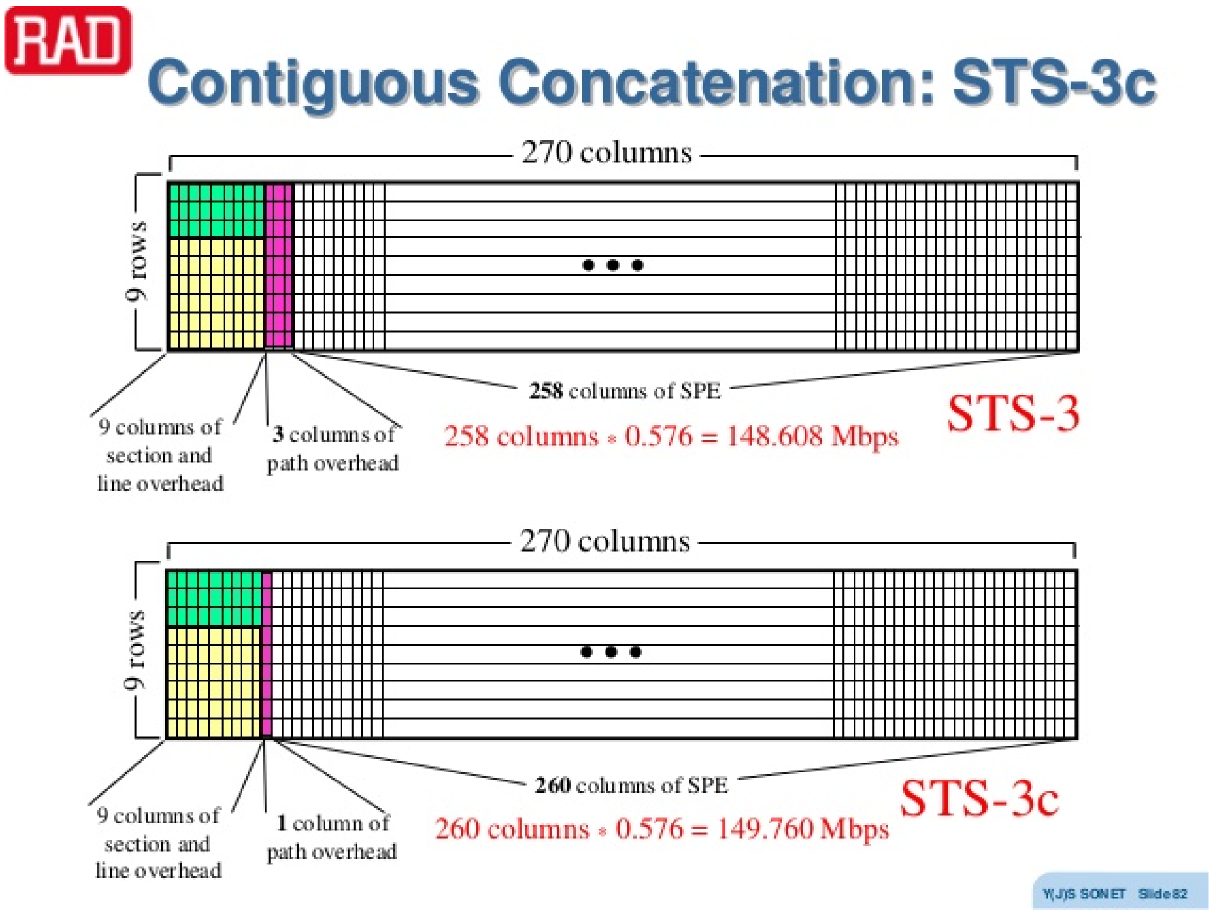

RAD
Contiguous Concatenation: STS-3c
270 columns
9 rows
9 columns of section and line overhead
3 columns of path overhead
258 columns of SPE
258 columns * 0.576 = 148.608 Mbps
STS-3
270 columns
9 rows
9 columns of section and line overhead
1 column of path overhead
260 columns of SPE
260 columns * 0.576 = 149.760 Mbps
STS-3c
Y(J)S SONET Slide82

Figure 3: Caption

2.3 Error Detection

2.3.1 Checksum

Sum up the messages, discard overhead, transmit with block. Nothing to write about. This is a simple error detection technique, which is not even the most efficient.

2.3.2 Parity Check

Look up in the reader. Should be familiar with the process of doing this, nothing to write about. Look up in the reader.

2.3.3 CRC- Circular Redundancy Check

The polynomial arithmetic, long division, XOR-ing. Look up in th reader. Process summarized below.

1. M(x) = message polynomial
2. C(X) = generated polynomial, look up in reference, not calculated

3. R(x) = calculated using the two above, M(x)*x^r where r is the order of the polynomial of C(x), the calculation is then M(X)*x^r/C(x) = R(X), this is the long division step done using XOR operation.
4. P(x) = polynomial to send = M(x) XOR R(x).
5. guarantees remainder is zero when divided by the polynomial C(x) as we just subtracted the remainder(when we XOR that is what happens since we are using modulo 2 arithmetic).
6. send
7. recalculate $\rightarrow$ if not remainder 0, then problem.

2.3.4 Shannon's Mathematical Theory of Communication

Only main topics are noted, most of this is intuition based and all formulae, nothing to write, other than reminding that these topics exist.

1. Channel Capacity
2. Amount of information in data - Huffman Coding
3. Entropy
4. Average number of bits.
5. Error Detection vs Error Correction (i.e. which has more overhead).
6. Entropy of a message * number of messages/s = information in message * messages/s = avg. no of bits per message * messages / s = avg number of bits / s.

2.4 Reliable Transmission

Two things to be concerned about: Reliable delivery is usually accomplished using a combination of two fundamental mechanisms—acknowledgments and timeouts. An acknowledgment (ACK for short) is a small control frame that a protocol sends back to its peer saying that it has received an earlier frame. By control frame we mean a header without any data, although a protocol can piggyback an ACK on a data frame it just happens to be sending in the opposite direction. The receipt of an acknowledgment indicates to the sender of the original frame that its frame was successfully delivered. If the sender does not receive an acknowledgment after a reasonable amount of time, then it retransmits the original frame. This action of waiting a reasonable amount of time is called a timeout.

1. Acknowledgements= Did you receive what I sent?
2. Timeout= Is it time for me to resend what I sent?

Rest of the section discusses the algorithms for ack/timeout.

1. Alternating Bit protocol
2. Stop-Wait
3. Sliding Window (please be familiar with all the acronyms- LAR, LFS, SWS – LAS, LAF, RWS)

2.5 Multi-Access Networks

2.5.1 Physical Properties of Ethernet

1. CSMA\CD Carrier Sense Multi Access, Collision Detection
2. Repeaters
3. length-maximal 2500m
4. single length is only 500m, so up to 4 repeaters max.

2.5.2 Access Protocol

1. Frame Format:
   (a) Premeable- 64bits: to synchronize the signal. Similar to how SONET was described.
   (b) Addresses- 48bits each Dest, Src.
   (c) Type- Demux key, used for seeing if needs to be passed to higher protocol.
   (d) Body
   (e) CRC- 32bit
2. Addresses:
   (a) MAC address. e.g. 8:0:2b:e4:b1:2.

2.5.3 Transmitter Algorithm

1. if idle send. Maximal length of frame = 1500 bytes (for fairness).
2. if busy, calc random time, wait random time, try again. Uses Exponential Backoff.
3. At the moment an adaptor detects that its frame is colliding with another, it first makes sure to transmit a 32-bit jamming sequence and then stops the transmission. Thus, a transmitter will minimally send 96 bits in the case of a collision: 64-bit preamble plus 32-bit jamming sequence.
4. Typically Ethernet for given setting has a minimum of 512 bits frame. This allows for a collision to be detected, otherwise the computer may go away and not know that a collision happened, as illustrated below. Use padding as required.

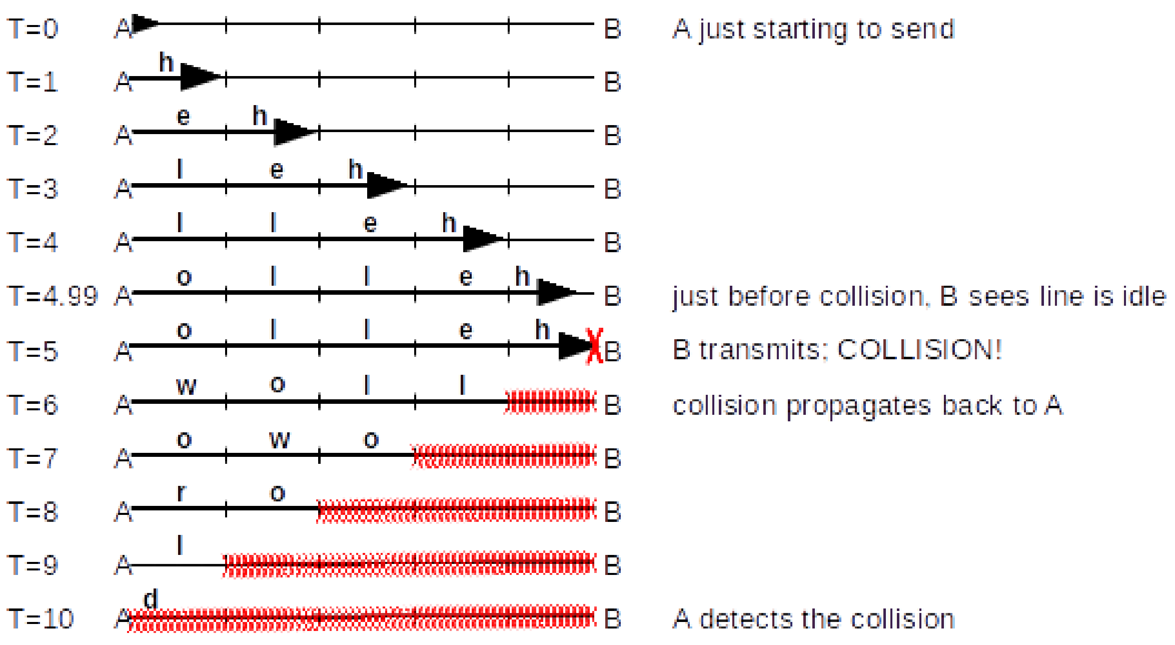


Figure 4: Caption

2.6 Wireless Networks

2.6.1 Medium

read from reader, and slide mostly.

2.6.2 Spread Spectrum

1. Frequency Hopping: Using a pseudorandom number generator (prng), jump to different frequency. The reciever has the same algorithm for the prng and so can follow the signal. Originally, Wifi typically uses 79- 1 MHZ bandwidth signals to jump across, all in 2.4GHz range.

2. Direct Sequence Spread Spectrum: use a prng to create a chipping code. Use this code to encode the data to send. Originally, Wifi typically uses 11 bit chip code. If reciever does not have the same prng, the recieval is seen as random. This is called scrambling. Typically 1 bit is encoded to a length of 4, and then chip code is used to encode these 4 repated bits. Having more bits also increases tolerance to interference.

2.6.3 Different Methods of making connection

1. Base Station: Router, nodes can not communicate directly. The base station as it were, has no mobility, it meant to stay fixed. All communication occurs through it.

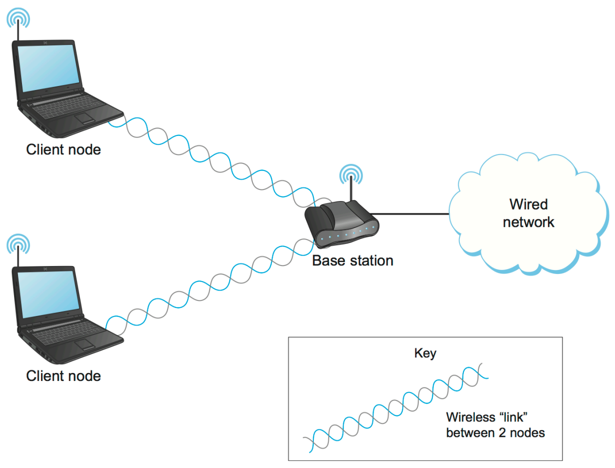


Figure 5: Caption

2. Access Point: Nodes form adhoc links between themselves, allowing direct communication.

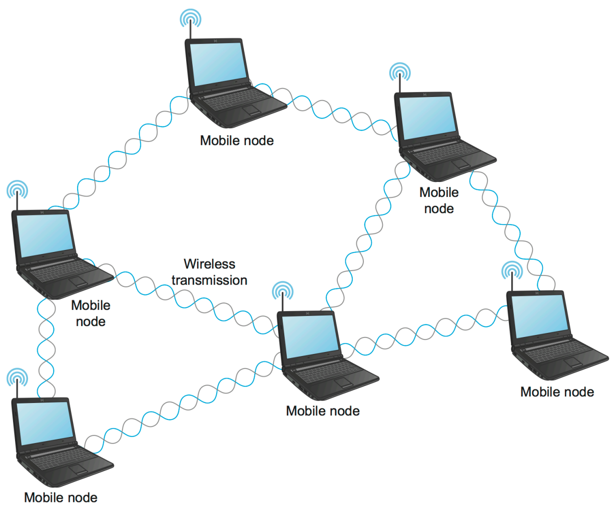


Figure 6: Caption

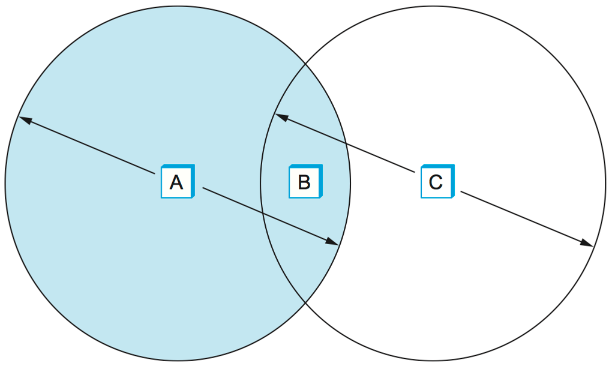
2.6.4 Current Wifi

The technology today allows for greater speed and bandwidths. It is worth noting that while all the 802.11 standards define a maximum bit rate that can be supported, they mostly support lower bit rates as well (e.g., 802.11a allows for bit rates of 6, 9, 12, 18, 24, 36, 48, and 54 Mbps). At lower bit rates, it is easier to decode transmitted signals in the presence of noise. Different modulation schemes are used to achieve the various bit rates. In addition, the amount of redundant information in the form of error-correcting codes is varied. More redundant information means higher resilience to bit errors at the cost of lowering the effective data rate (since more of the transmitted bits are redundant). Since this is factual, read from the book.

2.6.5 Collision Avoidance

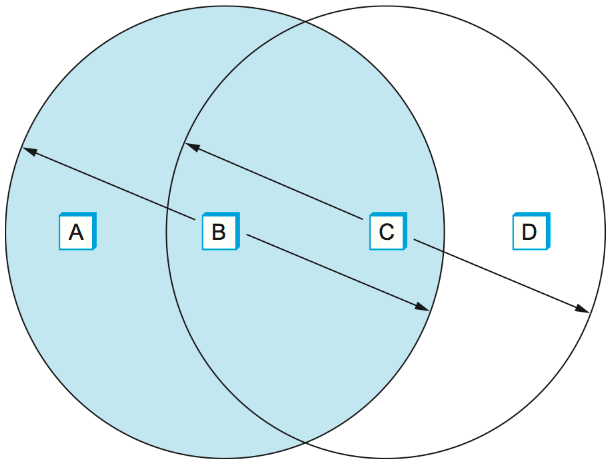
Two problems:

1. Hidden Node Problem:

   A hidden from C. A and C both want to communicate with B, the message collides, but unlike Ethernet where jamming sequence could be send, A and C are not within each other's range, so they cannot communicate each other's collision to each other.

2. Exposed Node Problem:

   B wants to communicate with A and C wants to commuincate with D. They should be able to, but since B is in range of C or vice versa, if C is sending messsage to D, B thinks it can not send a message without interfering C,D is out of range of B, hence this is not an issue.

FIX:

1. CSMA: The Carrier Sense part seems simple enough: Before sending a packet, the transmitter checks if it can hear any other transmissions; if not, it sends. However, because of the hidden node problem, just waiting for the absence of signals from other transmitters does not guarantee that a collision will not occur from the perspective of the receiver. For this reason, one part of CSMA/CA is an explicit ACK from the receiver to the sender. If the packet was successfully decoded and passed its CRC at the receiver, the receiver sends an ACK back to the sender.

2. RTS-CTS. Ready to send, Clear to send. B sends message that it wants to send message to A, RTS. A receives it. A sends CTS to all in range. Those who are not A, then wait and don't send anything for the time allowed by the Wifi for one node to occupy the signal. If it does not receive CTS, but perhaps receives the RTS, then this node is free to talk

to some other node, in problem of Exposed Node, D never receives a CTS from A, so it sends it's CTS to C, in response to C's RTS (which could have been sent before B started transmitting to A), and C sends message to D successfully.

3. Obviously Collisions of RTS are possible, but if CTS is not received after a while, the nodes can always try again later on, using exponential backoff.

2.6.6 Distribution System

These are used by AP's as mentioned above.

1. Nodes can talk to nodes in range. What about outside? They talk to Distribution network, and node is sent to the node which far, but connected to the Wifi.
2. **What if the node is moving?**
   (a) The node sends a Probe frame.
   (b) All APs within reach reply with a Probe Response frame.
   (c) The node selects one of the access points and sends that AP an Association Request frame.
   (d) The AP replies with an Association Response frame.

   A node engages this protocol whenever it joins the network, as well as when it becomes unhappy with its current AP. This might happen, for example, because the signal from its current AP has weakened due to the node moving away from it. Whenever a node acquires a new AP, the new AP notifies the old AP of the change (this happens in step 4) via the distribution system.

2.6.7 Wifi Frame Format

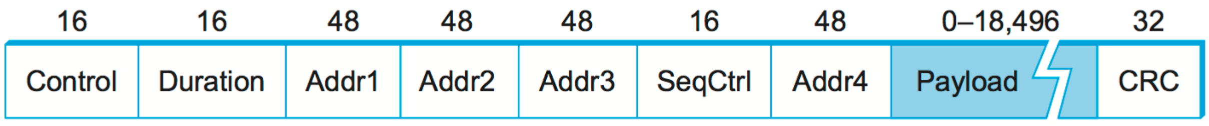


Figure 7: Caption

The peculiar thing about the 802.11 frame format is that it contains four, rather than two, addresses. How these addresses are interpreted depends on the settings of the ToDS and FromDS bits in the frame's Control field. This is to account for the possibility that the frame had to be forwarded across the distribution system, which would mean that the original sender is not necessarily the same as the most recent transmitting node. Similar reasoning applies to the destination address. In the simplest case, when one node is sending directly to another, both the DS bits are 0, Addr1 identifies the target node, and Addr2 identifies the source node. In the most complex case, both DS bits are set to 1, indicating that the message went from a wireless node onto the distribution system, and then from the distribution system to another wireless node. With both bits set, Addr1 identifies the ultimate destination, Addr2 identifies the immediate sender (the one that forwarded the frame from the distribution system to the ultimate destination), Addr3 identifies the intermediate destination (the one that accepted the frame from a wireless node and forwarded it across the distribution system), and Addr4 identifies the original source. Like a traceback.

2.6.8 Bluetooth

Slave Devices, Master Devices. Slave Devices can not talk to each other. Bluetooth operates in the license-exempt band at 2.45 GHz. Bluetooth links have typical bandwidths around 1 to 3 Mbps and a range of about 10 m. For this reason, and because the communicating devices typically belong to one individual or group, Bluetooth is sometimes categorized as a Personal Area Network (PAN). Parked devices, and active devices. Once parked can only reactivated by the master device.

3 Chapter 3: Internetworking

3.1 Switching Basics

How to interconnect nodes on different networks?

1. Connect the switch of the two networks. This is the most usual practice.

What do we need for networks to communicate?

1. Every node needs an address that can at the very least be uniquely identified on the network itself.
2. In reality, the Ethernet addresses are distributed to companies, so the Ethernet address are globally unique. So, the hardware is uniquely identifiable.
3. We need to be able to identify who is connected where, and when we send data, through which it is the best idea to send it through. This is the concept of forwarding table.

3.1.1 Datagrams

1. Called connection-less way of making networks.
2. Each router has a forwarding table (also called routing table), and when a packet comes to it, it just checks where it needs to go, and sends it there.
3. This requires the data-gram to have the entire address details to be in the packet, otherwise some node may send it to the wrong place.
4. This is good when topology is shifting too much and pre planning route is very hard. The responsibility is shifted to the router(switch) to find the path.
5. Since there is no pre-planning, no way to know if packet can be delivered. This is similar to how packet switching works.

3.1.2 Virtual Circuit Switching

1. This is called a connection oriented connection.
2. The network administrator, or the nodes themselves communicate to fix connection routes.
3. One RTT is spent making sure connection is possible.

4. Each switch has a table for every connection it has seen or is pre loaded. The table looks like this:

Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
2	5	1	11

Table 1: VCI

5. Every connection is identified using the VCI. So every switch maps the receiving VCI from an interface to an outgoing one and then sends it to the corresponding port. This is repeated until the last node. Usually numbering starts from 0 at each switch, if 0 is taken then we increment and define a map using the received VCI and port (interface) to the new incremented VCI and interface pair.
6. This is good when topology does not change a lot, path stay there. Little overhead, as complete address not needed.
7. ATM (Asynchrous Transfer Mode) is a nice example.

3.1.3 Source Routing

1. One last way that networks can communicate is that the source node tells where the packet needs to go, dictating each hop from the source itself at each switch.
2. It is obvious the problems we can see. Topology shifts, the node needs to have the entire knowledge if the network, that is possible only if everyone is talking to each other ALL the time. Inefficient.
3. This can be implemented by reshuffling, stripping and pointers.
4. Source routes are sometimes categorized as strict or loose.
   (a) In a strict source route, every node along the path must be specified as mentioned above, whereas
   (b) a loose source route only specifies a set of nodes to be traversed, without saying exactly how to get from one node to the next. A loose source route can be thought of as a set of waypoints rather than a completely specified route. The loose option can be helpful to limit the amount of information that a source must obtain to create a source route.

3.2 Switched Ethernet

3.2.1 Learning Bridge

As someone sends to me, I learn where the guy is connected. No need to flood.

3.2.2 Spanning Tree

The network can be seen as a planar graph, from which a spanning tree can be determined. The algorithm can be summarised as:

1. Functional switch with lowest ID is the root.

2. Every node which sees all offers form other nodes, chooses the strictly smallest one.
3. Every node which sees two switches offering same distance from the root, it chooses the switch with lower ID.
4. If two switches see they have same distance from the root, the close connection between themselves.
5. If creation of path creates a cycle, don't create connection.
6. If all nodes covered, stop.

3.2.3 VLAN

These things are not exactly scalable. Imagine flooding a really big network. The solution to do this is augment the network into smaller virtual networks, Instead of the entire network knowing everything about the entire network, it knows more about the small partitions. Then whenever the packet wants to talk to other nodes, in these other virtual networks, the packet is passed and then the that network takes care of it. Obviously more information is added into the header. Uses learning bridges. Packet forwarding does not alter IP or MAC, nothing.

3.3 Internet IP

3.3.1 Service Model

Internet protocol is connection less service. This is called best-effort as well. Since connection less, we have datagrams.

3.3.2 IPv4 Packet Format

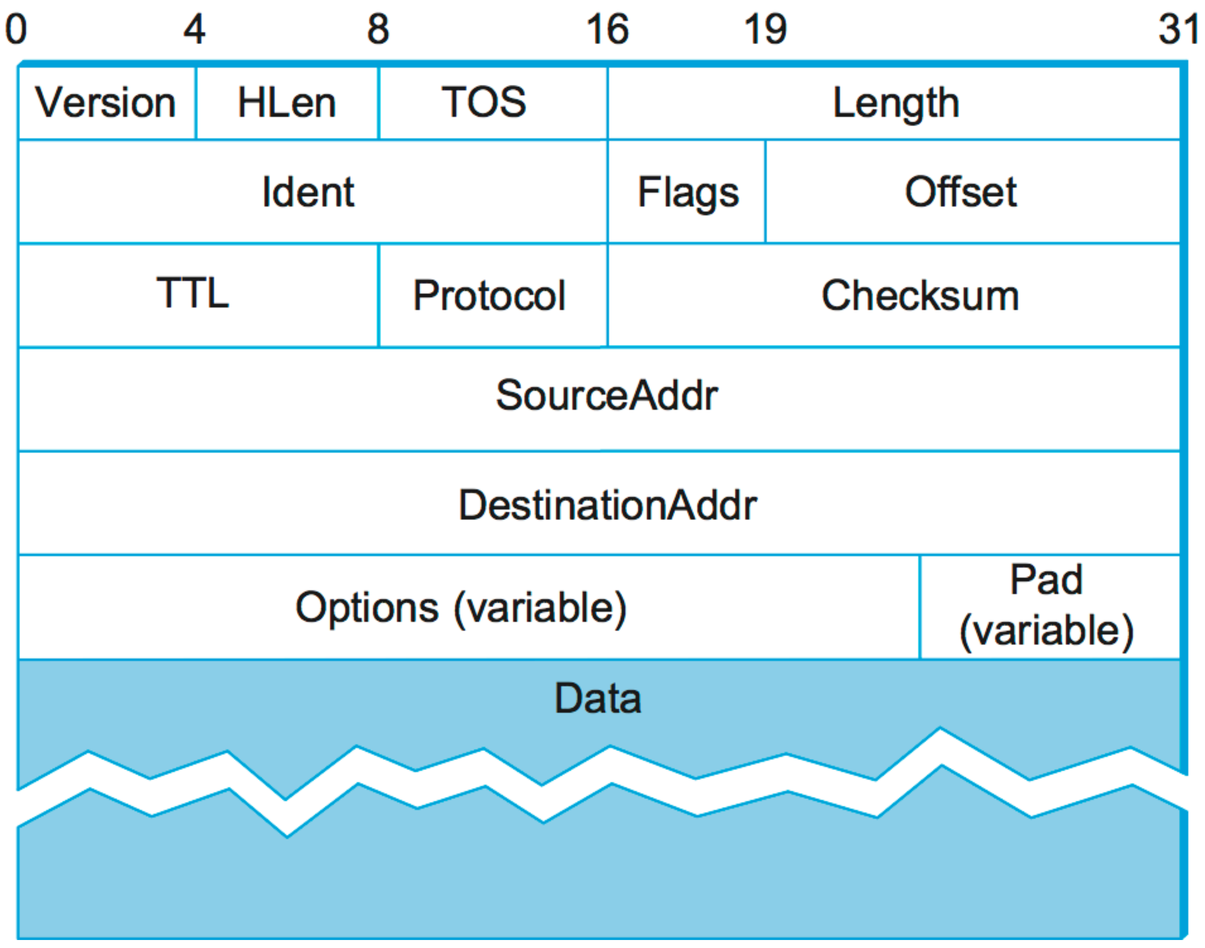
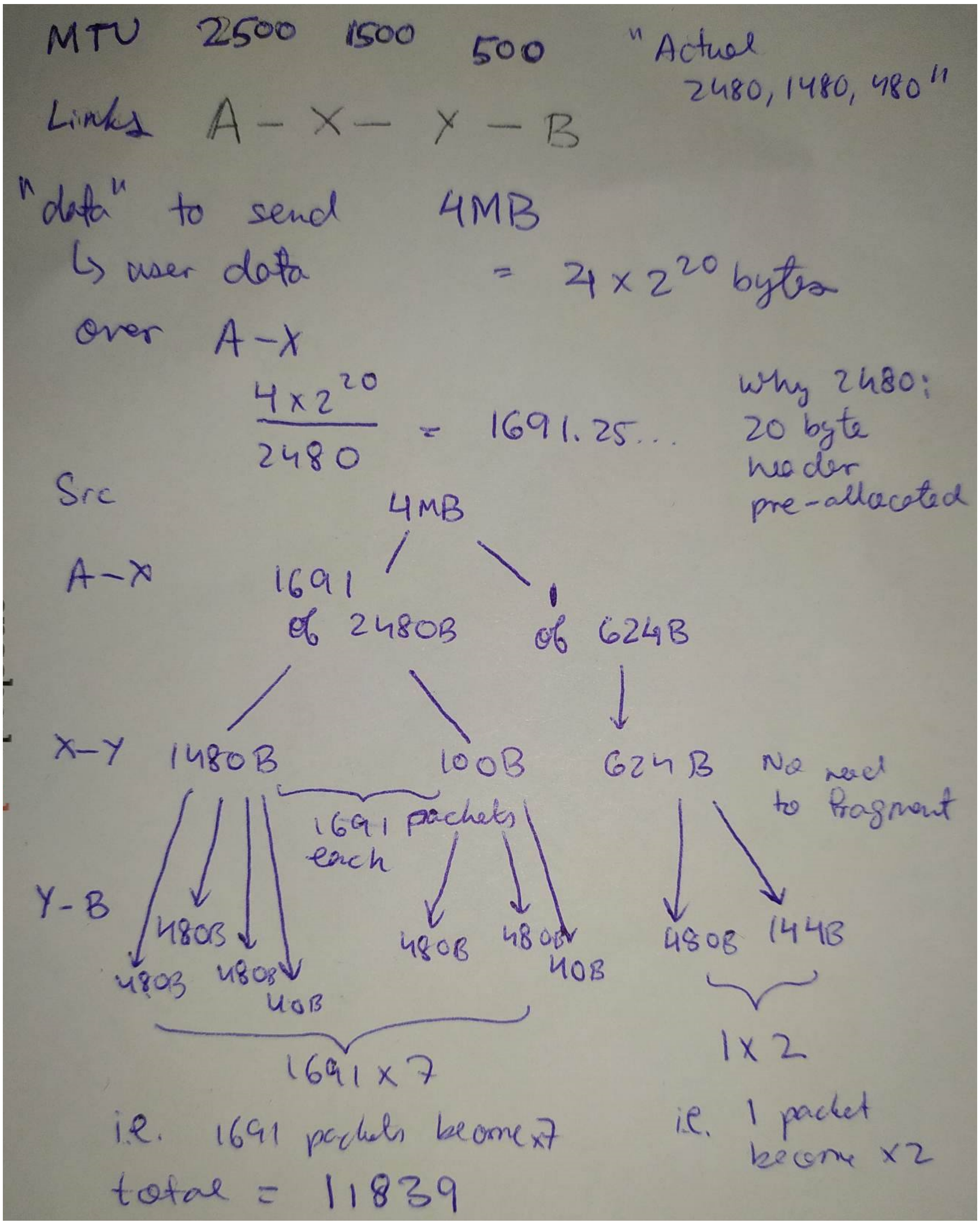


Figure 8: Packet Format

3.3.3 Fragmentation and Reassembly

Never reassemble at the intermediate switch.



MTU 2500 1500 500 "Actual 2480, 1480, 480"

Links A – X – Y – B

"data" to send 4MB

↳ user data = 4×2^{20} bytes

over A–X

$$\frac{4 \times 2^{20}}{2480} = 1691.25\ldots$$

why 2480: 20 byte header pre-allocated

Src 4MB

A–X 1691 of 2480B, 1 of 624B

X–Y 1480B, 100B, 624B

1691 packets each

No need to fragment

Y-B 480B 480B 480B 40B, 480B 480B 40B, 480B 144B

1691 x 7

1 x 2

i.e. 1691 packets become x7

i.e. 1 packet become x2

total = 11839

Figure 9: Example - all you need

3.3.4 Global Addressing

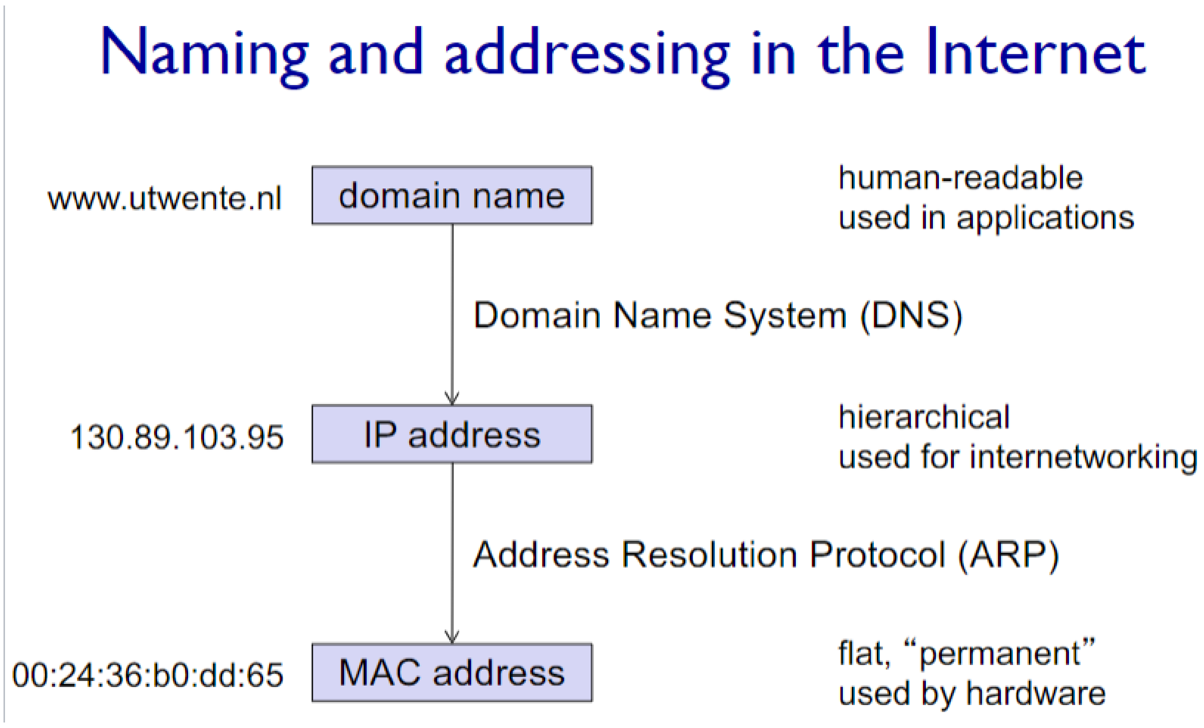


Figure 10: Addressing

3.3.5 Datagram forwarding in IP

1. You have the datagram.
2. it has an address.
3. compare in the routing table, do I know how to get there?
   (a) Yes it on my network: Send to that node
   (b) No: Send to best router, the next hop.
      i. in case many routers, choose best one.
   (c) in case no best router, absolutely no idea, give to default router.

3.3.6 Classful Addressing, Classless Addressing, Subnetting

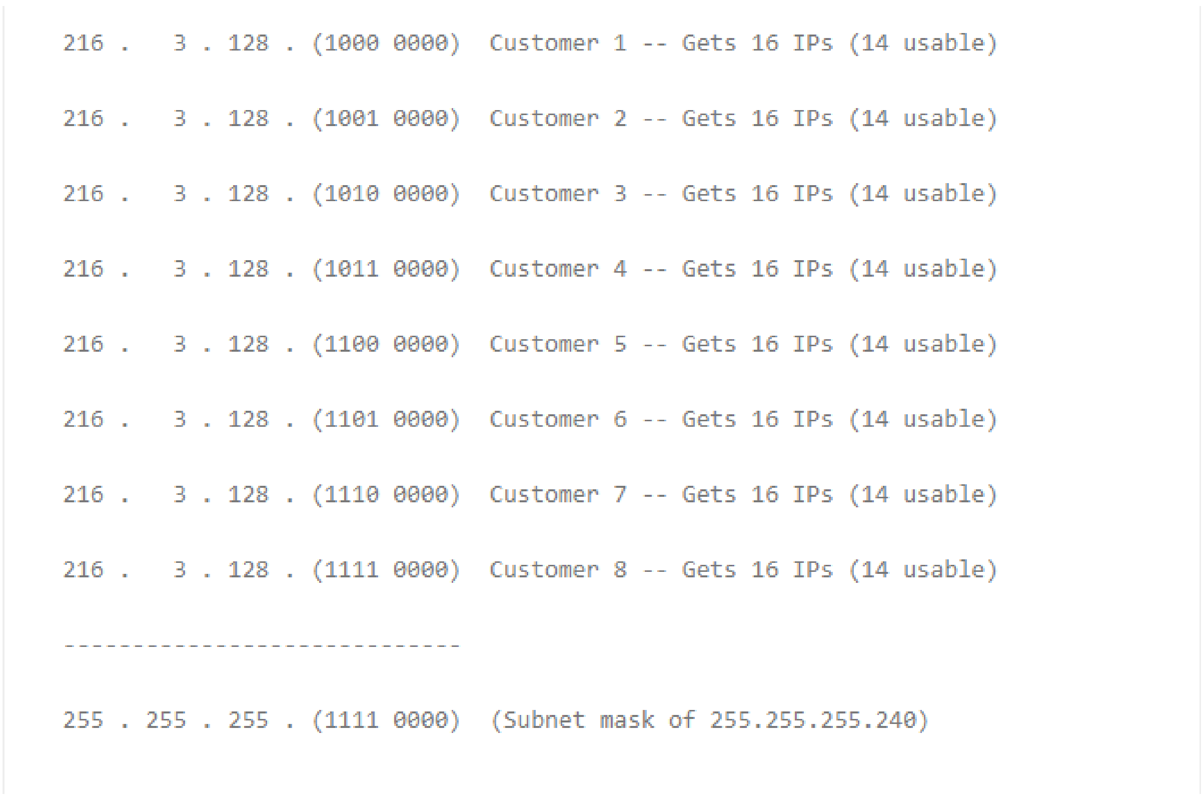
1. Subnetting:
   ex: 216.3.128.128 (/25)– this means the 25 MSB are network address. The rest is host address.// The mask is: 255.255.255.240 come from binary (11111111111111111111111110000000)
   (a) ex. 216 . 3 . 128 . (1000 0010)
   (b) To get network address: received address (AND) subnet mask
   (c) compare with the first 25 bits. If they same, then on this network, so extract host address (XOR) do whatevs. this case it is 2.
   (d) if not, then we are in the wrong place send it away.

2. Classful:

    (a) Class A, B, C: basically subnets with /8, /16, /24.

3. CIDR: Classless Interdomain Routing.
   Think of Classful Addressing as a special case CIDR. The example of sub netting explains what happens.

4. Last topic, Network Migration. Using a mask, we can add another 1, i.e. /25 to /26. Using this we can define sub networks, like a tree hierarchy. But if one of them wants to leave, they take their address, and then the switch broadcasts that specific chunk of address. When packets are being sent, the longest prefix matching is used first, since it more specific.



```
216 .   3 . 128 . (1000 0000)  Customer 1 -- Gets 16 IPs (14 usable)

216 .   3 . 128 . (1001 0000)  Customer 2 -- Gets 16 IPs (14 usable)

216 .   3 . 128 . (1010 0000)  Customer 3 -- Gets 16 IPs (14 usable)

216 .   3 . 128 . (1011 0000)  Customer 4 -- Gets 16 IPs (14 usable)

216 .   3 . 128 . (1100 0000)  Customer 5 -- Gets 16 IPs (14 usable)

216 .   3 . 128 . (1101 0000)  Customer 6 -- Gets 16 IPs (14 usable)

216 .   3 . 128 . (1110 0000)  Customer 7 -- Gets 16 IPs (14 usable)

216 .   3 . 128 . (1111 0000)  Customer 8 -- Gets 16 IPs (14 usable)

-----------------------------

255 . 255 . 255 . (1111 0000)  (Subnet mask of 255.255.255.240)
```

Figure 11: Subnet: The first address would then have a mask of /28 since the first 4 0's are fixed. If that network migrates this address is removed, but the switch still broadcasts the previous address. No change there. If someone wants to send a packet, they send to the one with the highest prefix /28, hence that's where the packet is sent first.

3.3.7 DHCP

Give the host, an address. Dynamic Host Resolution Protocol.

3.3.8 ARP

Address Resolution Protocol. Get the MAC to know where to send the data, so the actual party can identify on the Data link layer it needs to compare the MAC. In this we look at the IP address and check to which router or the node itself I need to send the packet to. So when packets travel over many routers, of different LANs, the MAC addresses change since they are travelling between routers and then finally the node. In case bridges are used, nothing changes, as the the same network is simply broken into many parts and not entirely different networks exist. In the same network the bridges know where to send each MAC.

3.3.9 Routing

1. RIP: Routing Information Vector. IE Distance vector- Bellman Ford algorithm.
2. OSPF: Open Shortest Path First. Link State Routing. IE Dijkstra algorithm.
3. AH OD VR: Ad hoc on demand vector routing. Mobile phones. Dynamic networks- too many changes, too much movement. The routes are made, fixed only when needed as the name suggests, so the orutes are not presistent for too long, which is good for the shifting topology.

4 Chapter 4: Advanced Internetworking

4.1 Interdomain routing

4.1.1 Border Gateway Protocol (BGP)

1. Problem: I do not want to provide a route through me, if I am paying. I want to be able to choose a route which is cheaper, maybe faster, maybe more secure. How can this be done?
2. How it is done: **Autonomous Systems**: These are ”systems” which represent a network or access to a network. Every ISP can have many (AS) and every AS can have many access points. AS can even connect to one another.
   (a) Stub AS: Only one access point AS
   (b) Multihomed-AS: Multiple AP’s. Helps to have redundancy, have connections in many geographic areas, need to connect them.
   (c) Transit AS: their main purpose is ti send data across two or more AS, or happen to provide this service along with other services.
3. **The purpose of BGP**: First, it is necessary to find some path to the intended destination that is loop free. Second, paths must be compliant with the policies of the various autonomous systems along the path—and, as we have already seen, those policies might be almost arbitrarily complex. Thus, while intradomain focuses on a well-defined problem of optimizing the scalar cost of the path, interdomain focuses on finding a non-looping, policy-compliant path—a much more complex optimization problem.
4. **How does all this help us to build scalable networks?**
   (a) First, the number of nodes participating in BGP is on the order of the number of autonomous systems, which is much smaller than the number of networks.
   (b) Second, finding a good interdomain route is only a matter of finding a path to the right border router, of which there are only a few per AS. Thus, we have neatly subdivided the routing problem into manageable parts, once again using a new level of hierarchy to increase scalability.

5. The complexity of interdomain routing is now on the order of the number of autonomous systems, and the complexity of intradomain routing is on the order of the number of networks in a single AS.

6. **How can the nodes inside a AS use the links?**

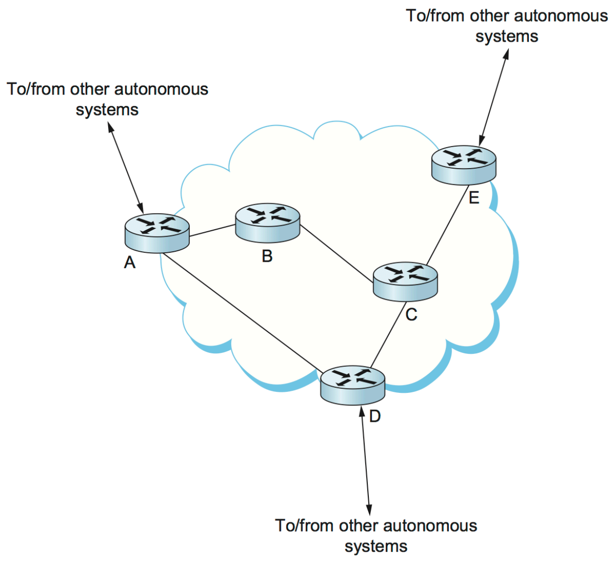

To/from other autonomous systems
To/from other autonomous systems
A
B
E
C
D
To/from other autonomous systems

Figure 12: Example of interdomain and intradomain routing. All routers run **i**BGP (OSPF or DV) and an intradomain routing protocol **e**BGP. Border routers A, D, and E also run **e**BGP to other autonomous systems.

   (a) **interior** BGP: OSPF or DV
   (b) **exterior** BGP: the actual BGP

4.2 IPv6

IPv6 provides a 128-bit address space, as opposed to the 32 bits of version 4. Thus, while version 4 can potentially address 4 billion nodes if address assignment efficiency reaches 100%, IPv6 can address 3.4 × 1038 nodes, again assuming 100% efficiency.

4.2.1 Host Density

Max threshold for host density is 87.5%. This is where it starts getting problematic.

$$\text{HD:} = \frac{\log(\text{number of allocated objects})}{\log(\text{maximum number of allocatable objects})}$$

Figure 13: Host Density Formula

4.2.2 Desirable Features of IPv6

1. Support for real-time services

2. Security support
3. Autoconfiguration (i.e., the ability of hosts to automatically configure themselves with such information as their own IP address and domain name)
4. Enhanced routing functionality, including support for mobile hosts

4.2.3 Address Notation

1. 8 pieces of 16 bits represented by 4 Hexadecimals.
2. Just as with IPv4, there is some special notation for writing down IPv6 addresses. The standard representation is x:x:x:x:x:x:x:x, where each x is a **hexadecimal** representation of a **16-bit** piece of the address. An example would be: 47CD:1234:4422:ACO2:0022:1234:A456:0124.
3. Any IPv6 address can be written using this notation. Since there are a few special types of IPv6 addresses, there are some special notations that may be helpful in certain circumstances. For example, an address with a large number of contiguous 0s can be written more compactly by omitting all the 0 fields. Thus, 47CD:**0000:0000:0000:0000:0000**:A456:0124 could be written 47CD**::**A456:0124
4. Clearly, this form of shorthand can only be used for **one** set of contiguous 0s in an address to avoid ambiguity.

4.2.4 Route Aggregation

IPv6 is classless, so implements CIDR as well. But, since there are so many addresses we have more hierarchy in the way the addresses are assigned. At the moment 001 prefix address are only assigned, whch means we still have 87.5% addresses left that have not yet even been touched.

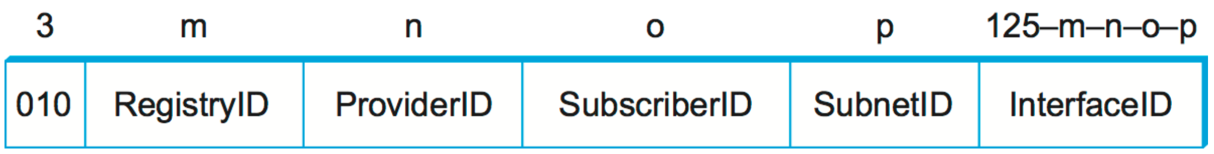


Figure 14: An IPv6 provider-based unicast address.

Problem may occur when two people connected tot he same router, but a go around is to broadcast both addresses to this and that part of the world, to create redundancy, and reduce load even.

4.2.5 Packet Format

Even though ipv6 is more complex the header is simpler. It has a header length of 40bytes, despite the fact the addresses are 4 times as long. This is pretty nice, and is possible since people took notice of what protocol were quite redundant and not needed in ipv6.

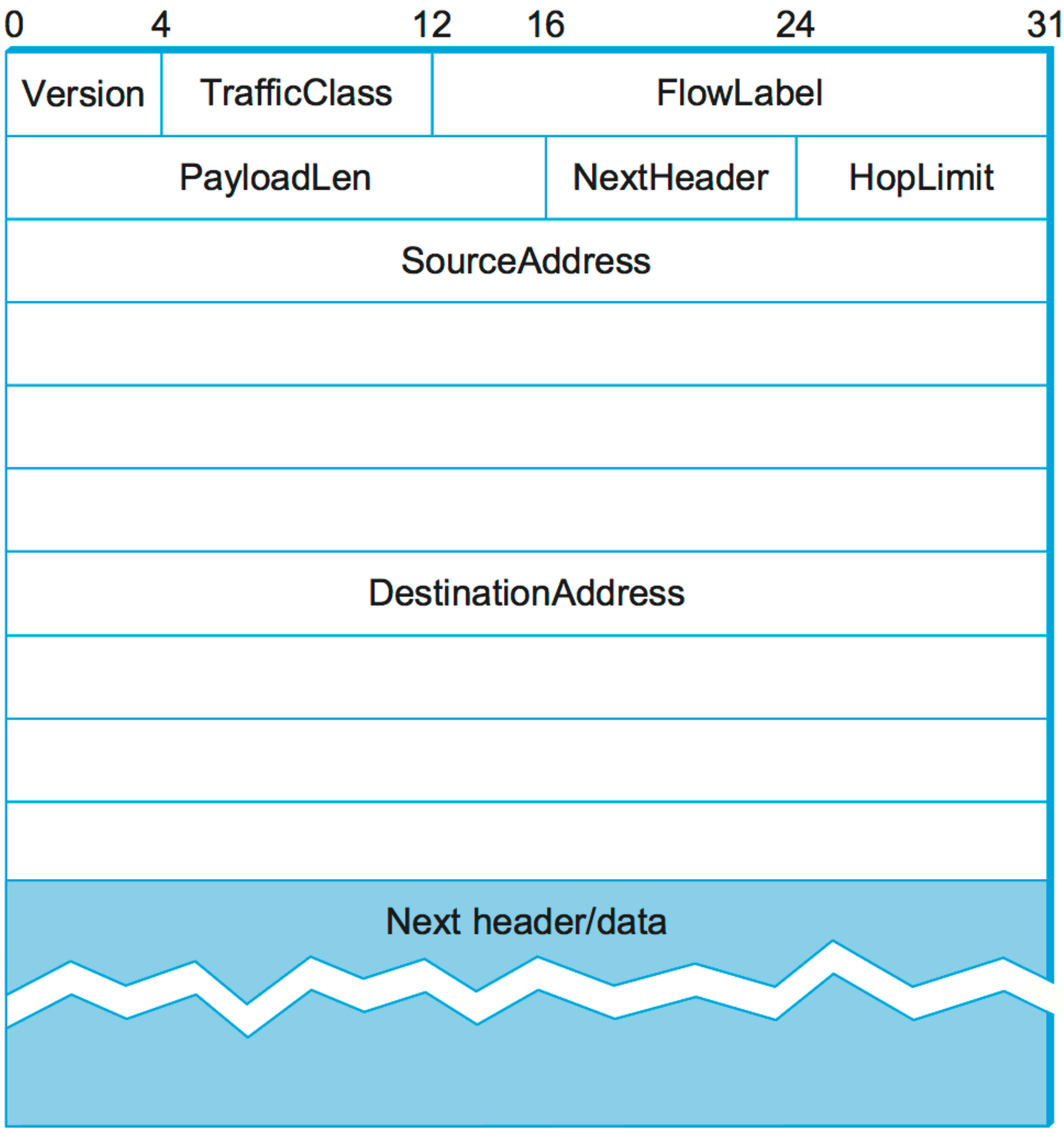


Figure 15: ipv6 packet format

4.2.6 Advanced Capabilities

1. Autoconfiguration: for IPv4 we always needed a DHCP. for IPv6 since the address sapce is so big we can any of these two:
   (a) Put your unique ethernet 48 bits into the interfaceID.
   (b) choose one at random, with a space so big, it is likely unused.
2. Source-Directed Routing: Another of IPv6's extension headers is the routing header. In the absence of this header, routing for IPv6 differs very little from that of IPv4 under CIDR. The routing header contains a list of IPv6 addresses that represent nodes or topological areas that the packet should visit en route to its destination. The reason are same as discussed in BGP.

4.3 Multicast

Normal IP communication, in which each packet must be addressed and sent to a single host, is not well suited to such applications. If an application has data to send to a group, it would have to send a separate packet with the identical data to each member of the group. This redundancy consumes more bandwidth than necessary. IP multicast is more scalable because it eliminates the redundant traffic (packets) that would have been sent many times over the same links, especially those near to the sending host.

Types of Multicast

1. **Source Specific Multicast (SSM):** a receiving host specifies both a multicast group and a specific sending host. The receiving host would then receive multicasts addressed to the specified group, but only if they are from the specified sender.
2. **Any Source Multicast (ASM):** many IP to many IP, so part of group but any sender.

How to join a group?

1. IPv4, that protocol is the Internet Group Management Protocol (IGMP)
2. IPv6, it is Multicast Listener Discovery (MLD).

For both they need to communicate with the router to become a part of the group. The router then does polling periodically to check is the node is still interested in the group, or has crashed, left correctly, etc.

As said before in IPv6 we have multicast addresses so deos IPv4.

4.4 Multicast Routing

As all thing uni and packets, Multicast needs routing too.

1. **Distance Vector Multicast Routing Protocol (DVMRP):** If I get packet from Source S, and I get it from the best path, I send to all. This floods the network and this is bad. If the packet then comes from not the bes tnode, sine flooding is happening, I do not forward that. Another problem people who are not part of the group G, need to send I don't part of this. For this we have the next one, **Sparse Mode**.
2. **Protocol Independent Multicast - Sparse Mode (PIM-SM):** Basically, I ask to join using a Randevouz Point, then a specific tree is created for this group for the router.

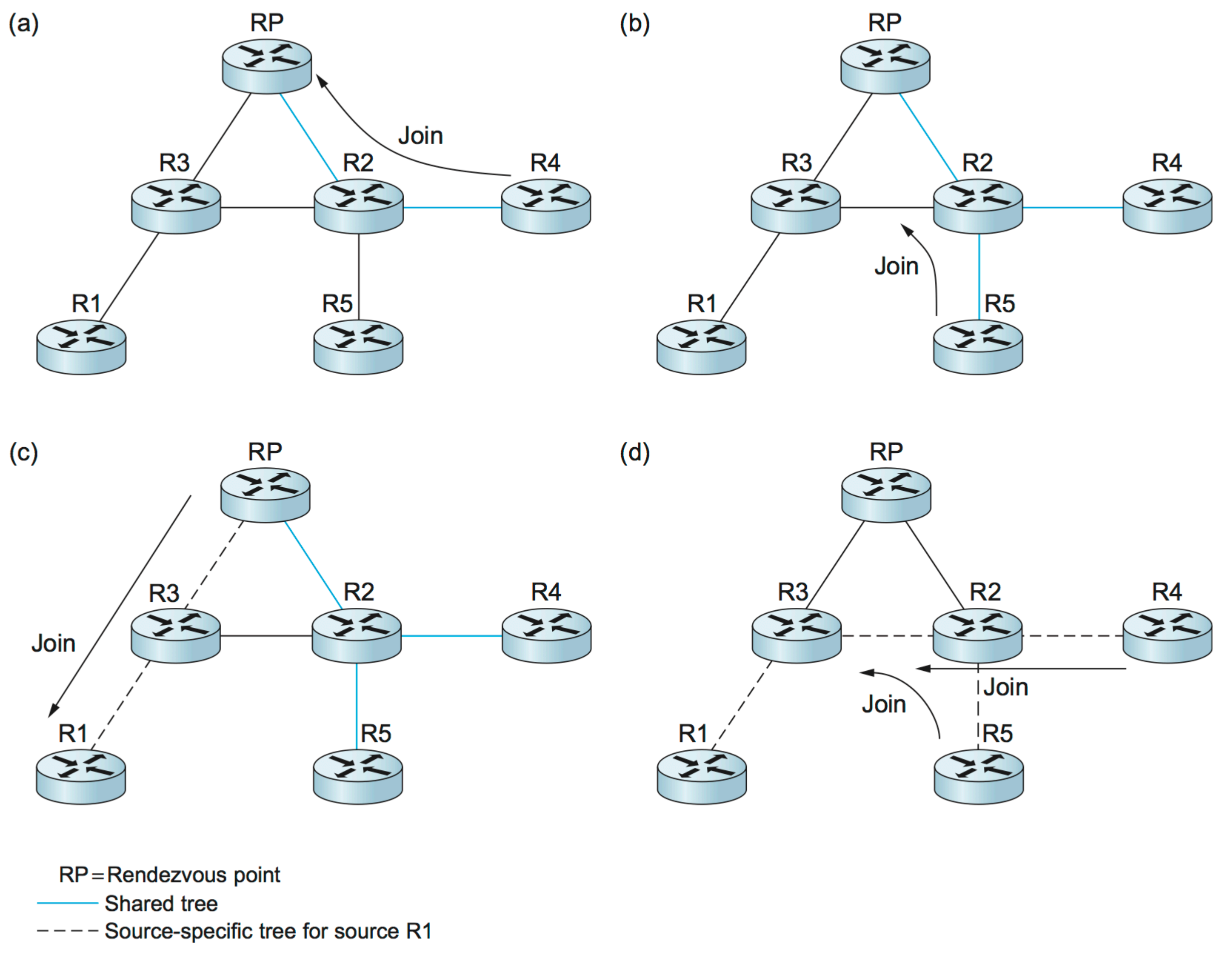


Figure 16: PIM operation: (a) R4 sends a Join message to RP and joins shared tree; (b) R5 joins shared tree; (c) RP builds source-specific tree to R1 by sending a Join message to R1; (d) R4 and R5 build source-specific tree to R1 by sending Join messages to R1.

If node wants to send to RP, the connecting router may choose to tunnel if it does not want others to know. Creating source specific tree when all nodes have joined and all.// **What if someone wants to send packet that is not even in the same network? Go to Interdomain Multicast**

3. **Interdomain Multicast (MSDP):** To extend multicast across domains using PIM-SM, the Multicast Source Discovery Protocol (MSDP) was devised. Each RP has one or more MSDP peer RPs in other domains. Each pair of MSDP peers is connected by a TCP connection over which the MSDP protocol runs. Send message to RP to register as source. Handle intra domain requests using reverse path broadcasts. If on another netowrk, send message to RP on that network, which should have registered before, and then that node sends join. Accept and all of that thing. Meanwhile, the source keeps sending Source Active messages to tell others who would like to join. One all joined up create tree and send data.

4. **Source-Specific Multicast (PIM-SSM):** The introduction of PIM-SSM has provided some significant benefits, particularly since there is relatively high demand for one-to-many multicasting:

   (a) Multicasts travel more directly to receivers.

   (b) The address of a channel is effectively a multicast group address plus a source address. Therefore, given that a certain range of multicast group addresses will be used for SSM exclusively, multiple domains can use the same multicast group address independently and without conflict, as long as they use it only with sources in their own domains.

(c) Because only the specified source can send to an SSM group, there is less risk of attacks based on malicious hosts overwhelming the routers or receivers with bogus multicast traffic.

(d) PIM-SSM can be used across domains exactly as it is used within a domain, without reliance on anything like MSDP.

(e) SSM, therefore, is quite a useful addition to the multicast service model.

5. **Bidirectional Trees (BIDIR-PIM):** I only they are the same thing as the question that question which was weird and going both directions, i.e. bidirectional trees.

4.5 MobileIP

4.5.1 Definition

1. **Home network** The home network of a mobile device is the network within which the device receives its identifying IP address (home address).

2. **Home address** The home address of a mobile device is the IP address assigned to the device within its home network.

3. **Foreign Network** A foreign network is the network in which a mobile node is operating when away from its home network.

4. **Care-of-Address** The care-of address of a mobile device is the network-native IP address of the device when operating in a foreign network.

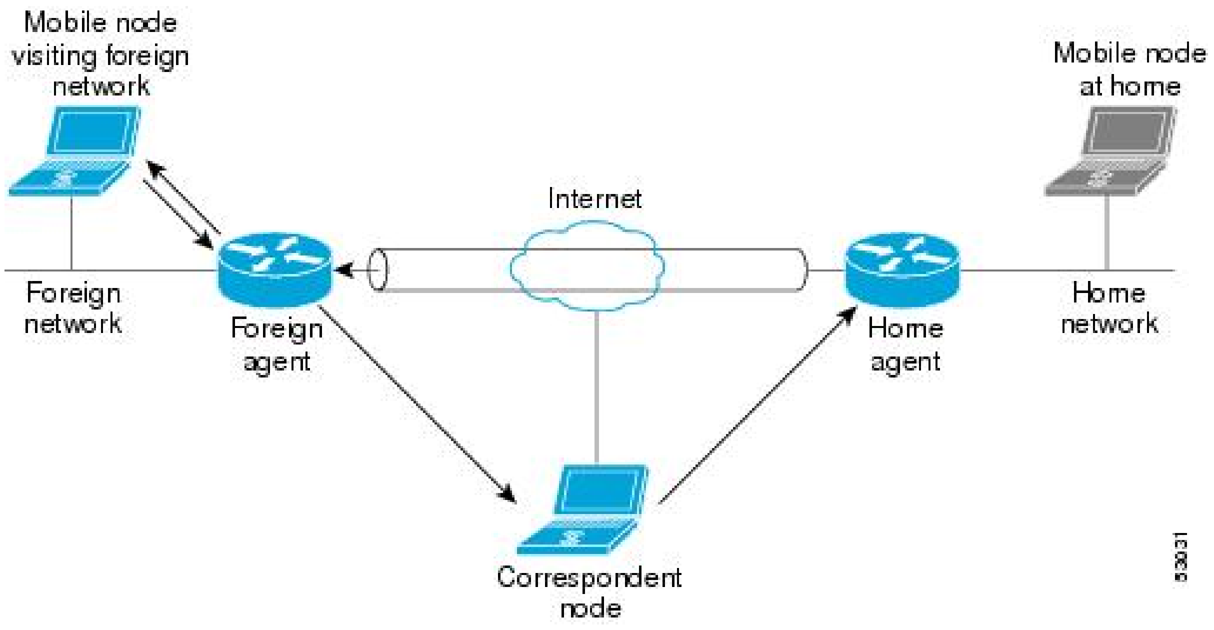
5. **Home Agent** A home agent is a router on a mobile node's home network which tunnels datagrams for delivery to the mobile node when it is away from home. It maintains current location (IP address) information for the mobile node. It is used with one or more foreign agents.

6. **Foreign Agent** A foreign agent is a router that stores information about mobile nodes visiting its network. Foreign agents also advertise care-of-addresses which are used by Mobile IP.

7. **Binding** A binding is the association of the home address with a care-of address.

4.5.2 How work?

So if I am sending a packet and I am not home, I send it to my agent, and he sends it to the node I wanted to send to, often using tunneling, i.e. it encapsulates my packet and sends it. If someone tries to send a packet to me, they send it to my home, then the home agent there tunnels the packet to my foreign agent who sends it to me, when I am away. This obviously requires a lot of registration.



When a packet finally arrives at the foreign agent, it strips the extra IP header and finds inside an IP packet destined for the home address of the mobile node.

What about traffic in the other direction (i.e., from mobile node to fixed node)? This turns out to be much easier. The mobile node just puts the IP address of the fixed node in the destination field of its IP packets while putting its permanent address in the source field, and the packets are forwarded to the fixed node using normal means. Of course, if both nodes in a conversation are mobile, then the procedures described above are used in each direction.

5 Chapter 5: End-to-End protocols

5.1 UDP

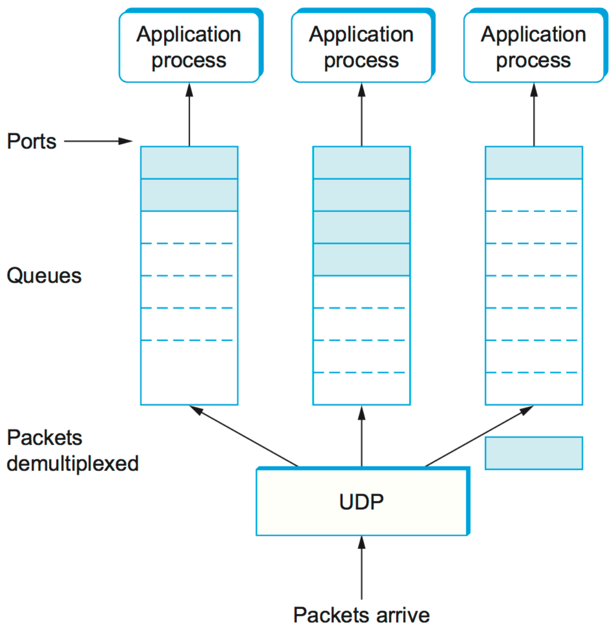
Simple Demultiplexor - user Datagram Protocol. There are likely to be many processes running on any given host, so the protocol needs to add a level of demultiplexing, thereby allowing multiple application processes on each host to share the network.

How to know demultiplex?

1. **OS assigned process Id (pid):** Typically not used since it is practical if a centralised system was in charge of all requests.
2. **Port numbers:** More typically used.
   (a) There is a demultiplexing function contains an identifier port for both sender and receiver. Ususally 16 bits long each. This allows for 2^{16} port numbers, enough for a single node. Hence the **port**, **host** pair is used to identify this connection.
   (b) **How to know to which port to send to?**
      i. Agree before hand. Called the well-known port. For HTML requests port 80 is generally used, for DNS requests port 53, and so on. If busy, retry later or send to some secondary agreed port. They can different from OS to OS. They can be changed during the conversation, for example- initiate using a well known one, and then switch to someother to allow others to talk on the well known one.

   ii. Use a port mapper. Think of hash tables, for asking to which port should I send a message. The advantage is that we don;t have to rely on port 80 to be the fixed one forever, can change anytime without changing the protocol.

3. **How demultiplexed?:** No flow control, if there is no space in the queue, just throw away the packet.



4. **UDP checksum:** UDP does not provide any reordering, flow control but does provide a checksum optional in IPv4, but mandatory in IPv6. This checksum to make sure the correct port is being sent to. The checksum takes in input the source IP, destination IP, protocol number, and UDP length. This forms what is called the psuedo header,

5.2 TCP

5.2.1 End -To-End-Issues

UDP was simple. Here are things make TCp harder.

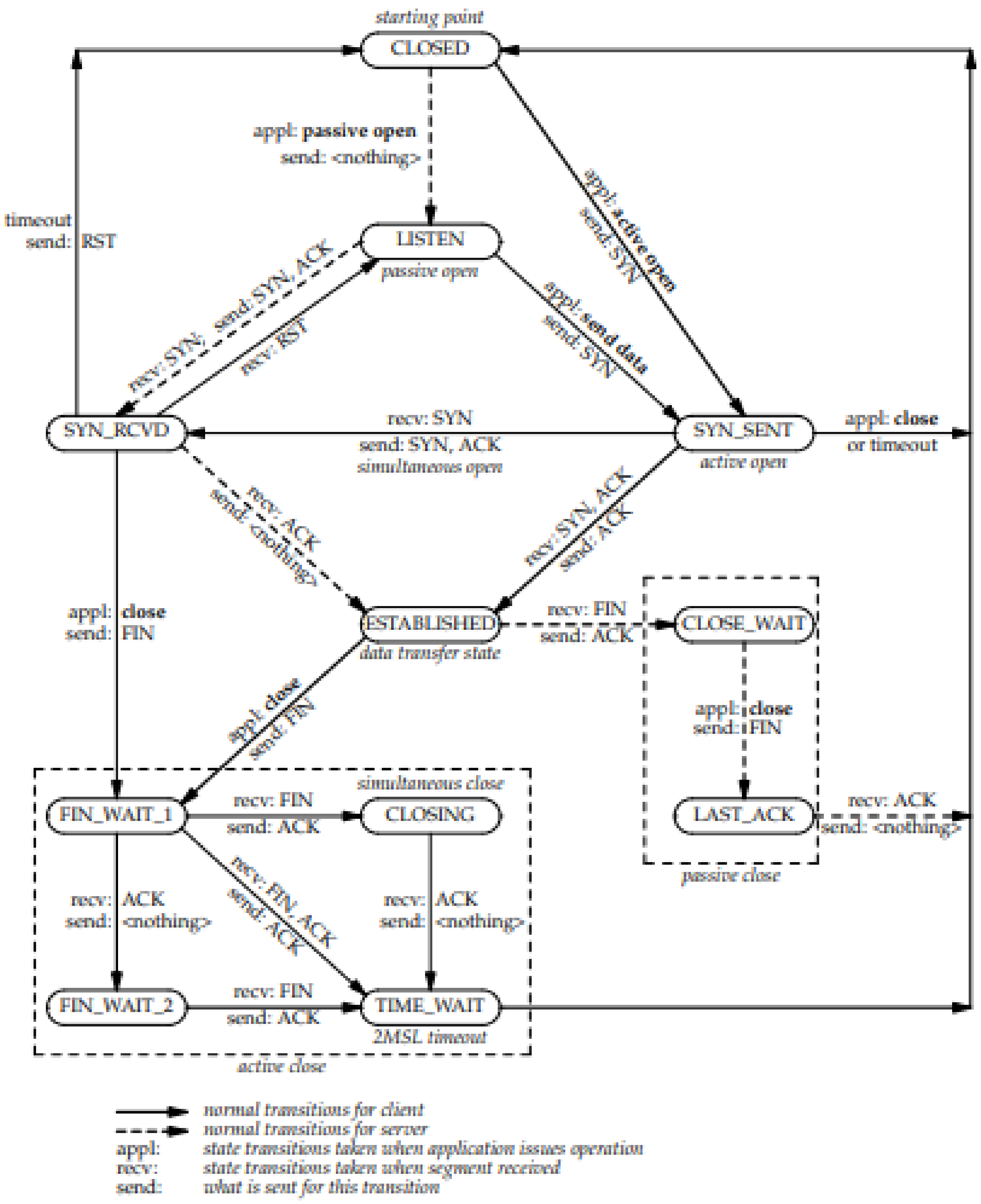
1. Need to start up and tear down the connection. since the resources need to be freed up.
2. Variable RTT, so can not hard code delay time like ethernet stuff, which guarantees reception for 2500m long ethernet packets and all.
3. out of order packets, as they travel in the internet. But we know packets defintely die out in 120 seconds, so we cna put this into a field called TTL, and take care of wandering packets. This can be changed by changing something called Maximum Segment Lifetime (MSL).
4. We set up a connection depending on how much we can send, to fill up all our pipes. But what if the reciever can not handle that. We need a mechanism to learn the resources.
5. network congestion (discussion later).

6. TCP uses the sliding window algorithm on an end-to-end basis to provide reliable/ordered delivery. Another network only took care of hop to hop. That was bad since packets may not be delivered to intended place and extra work done by mid way nodes,. TCP packets only travel to network layer, but X.25 needed to travel to transport layer.

5.2.2 Segment Format

1. TCP is a byte-oriented protocol. Which means the sequcne numbers and acks are actually increments of bytes not packets.
2. TCP's demux key is given by the 4-tuple: (SrcPort, SrcIPAddr, DstPort, DstIPAddr)
3. read actual segment from book. Copy paste stuff.
4. Checksum field is used in exactly the same way as for UDP—it is computed over the TCP header, the TCP data, and the pseudoheader, which is made up of the source address, destination address, and length fields from the IP header.

5.2.3 Connection establishment and Termination



TCP state transition diagram.

Learn how to read this, and the three way handshake.

5.2.4 Sliding Window Revisited

1. Need to know advertised window only.

2. Once buffer starts filling up on receiving end, it starts to tell the sender, that I do not have x space left but x - unprocessed bytes I have, where x is the original space it has. This way the sender does not overwhelm the receiver. This is called Control Flow. The receiver then uses this advertised window to calculate the effective windwow.

3. Once the receiver says I have no space, the sender starts Zero Frame Probes, these are frames of single byte of actual data that are sent on some time interval between 5 and 60s. This is used by the receiver to reply to the receiver, since the receiver never talks by itself.

   (a) Silly Window Syndrome- When a free window is advertised, does the sender jum in and send that much data or send the entire MSS (Maximum Segment Size). It needs to know when can it, or if ever can, since a smaller window size leads to it aggressivley filling up the pipe and not allowing bigger chunks to be sent, since the computer processes data segment wise, it would go byte by byte which can slow stuff down. Ususally the sender waits until it has one full MSS of free space. The question then comes how long should it wait? answered later.

4. protecting against wrap around. Use time stamps. Need to learn this through examples, hard to explain in words.

5. Keeping pipe full. with 16bits of sepeunce space we can only 64kb of data. How to resolve this? In TCP there is options called extended options. This allows a computer to shift the window size by an agreed amount to allow more sequence numbers.

5.2.5 To answer how to know when should I send data?

1. Nagle's: says wait till I have full MSS. Option of shutting it off, and allow SWS.

2. Original: Running average of sorts with weighting to understand rate of change. Then calculate o a timeout as twice the calculated RTT.

3. Karn/Patridge: modifies original, to not record RTT's during re-transmission. Original can take that into account and record a very small RTT the packet may simply be late. But also it just increases timeout by times two every time it needs to re transmit, exponential backoff.

4. Jacobson/Karels:

   (a) Difference = SampleRTT - EstimatedRTT

   (b) EstimatedRTT = EstimatedRTT + (delta x Difference)

   (c) Deviation = Deviation + delta (|Difference| - Deviation) where delta and delta are fractions between 0 and 1. That is, we calculate both the mean RTT and the variation in that mean.

   (d) TCP then computes the timeout value as a function of both EstimatedRTT and Deviation as follows:

   (e) TimeOut = mu x EstimatedRTT + phi x Deviation where based on experience, mu is typically set to 1 and phi is set to 4.

5.2.6 TCP extensions

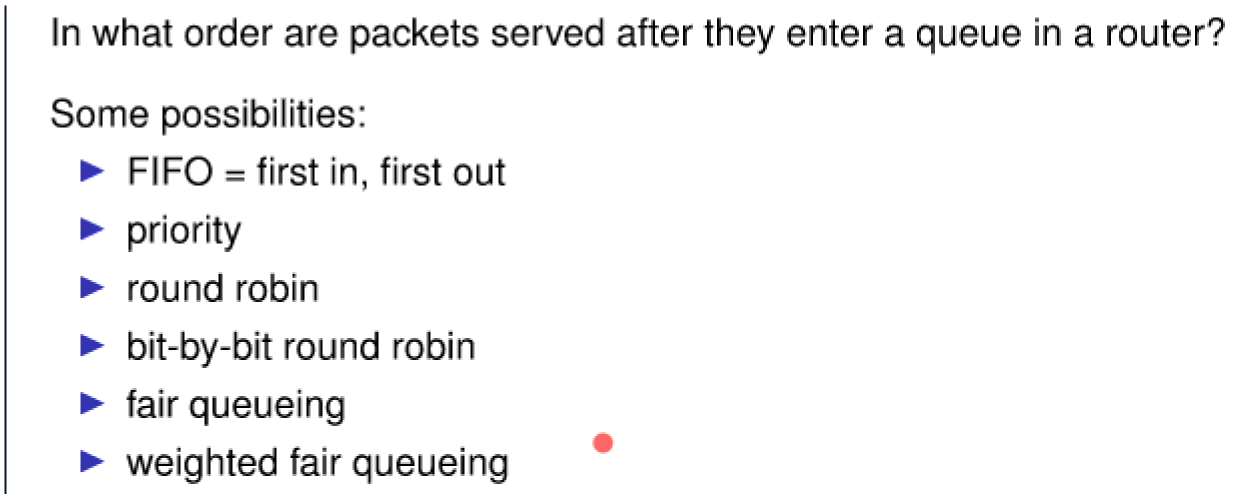
1. Scaling Factor: increase the size of the window.
2. Timestamp: TCP can read the actual system clock when it is about to send a segment, and put this time—think of it as a 32-bit timestamp—in the segment's header. The receiver then echoes this timestamp back to the sender in its acknowledgment, and the sender subtracts this timestamp from the current time to measure the RTT.
3. Timestamp: helps to figure when the packet got lsot and arrived too late. This will be discarded, buchecking with other timstamps of recent segments and seeing g this is too old and not in the current window.
4. SACK: Selective ACK, since cumulative ACKS are generally used.

5.3 RTP

Basically, it is a protocol bult on top of UDP, since the real time applications should not be tied down to what is provided, since packet losses may want to be treated differently, like in VOIP, just ignore packet loss since re-transmission is not fast enough.

6 Chapter 6: Congestion Control

6.0.1 Queing Disciplines


In what order are packets served after they enter a queue in a router?

Some possibilities:
- FIFO = first in, first out
- priority
- round robin
- bit-by-bit round robin
- fair queueing
- weighted fair queueing

Figure 17: Types of Servicing: FIFO: self, Priority: Two different Queues, obvious problem of senders sending self priority

1. FIFO: First in First Out, self explanatory. It is not very fair however. Whenever a person wants all the bandwidth they only need to send packets at higher rates. Can have priority, but hard to assign, priority is used for error messages, link updates however.
2. Fair Queuing: Trying to share the bandwidth, using almost bit-wise round robin like spending with additional possible priority queue, or just priority set for every connection and the packets in those queues. Things like failure, or change in routes is most important.

(a) Almost bit-wise, since packets are dealt according to size, and bits are not interleaved with one another. Also, this is used to decide which packets will be transmitted first, say, we have a long packet, and many short packets in another queue. The short packets will be scheduled first, then the long packets as the shorter packets take smaller time to send, but when the cumulative of shorter packets in one queue is longer, the longer packet instead will be sent. This way no one network is getting more service.

(b) Also note adding a smaller packet when transmitting a big packet does not stop the big packet transmission, in bit-wise round robin it would have been pre-empted.

3. Weighted Fairness Queue, sort of like priorty to a certain queue.

4. Fairness Index:

$$\mathcal{J}(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} {x_i}^2}$$

Figure 18: Fairness index: x_i is the bandwidth of i^{th} connection

6.0.2 TCP Congestion Control

1. Additive Increase, Multiplicative Decrease: for every ack until start threshold add 1 or some agrred number to congestion window, after that add 1 or some agreed number for every RTT if there is any ack. If packet loss, by timeout or triple dup ack, multiply the current congestion window by 0.5 or some number between 0 and 1, and set start threshold to this value, changing congestion window to 1, restart start threshold.

2. Fast Transmit, Fast Recovery:

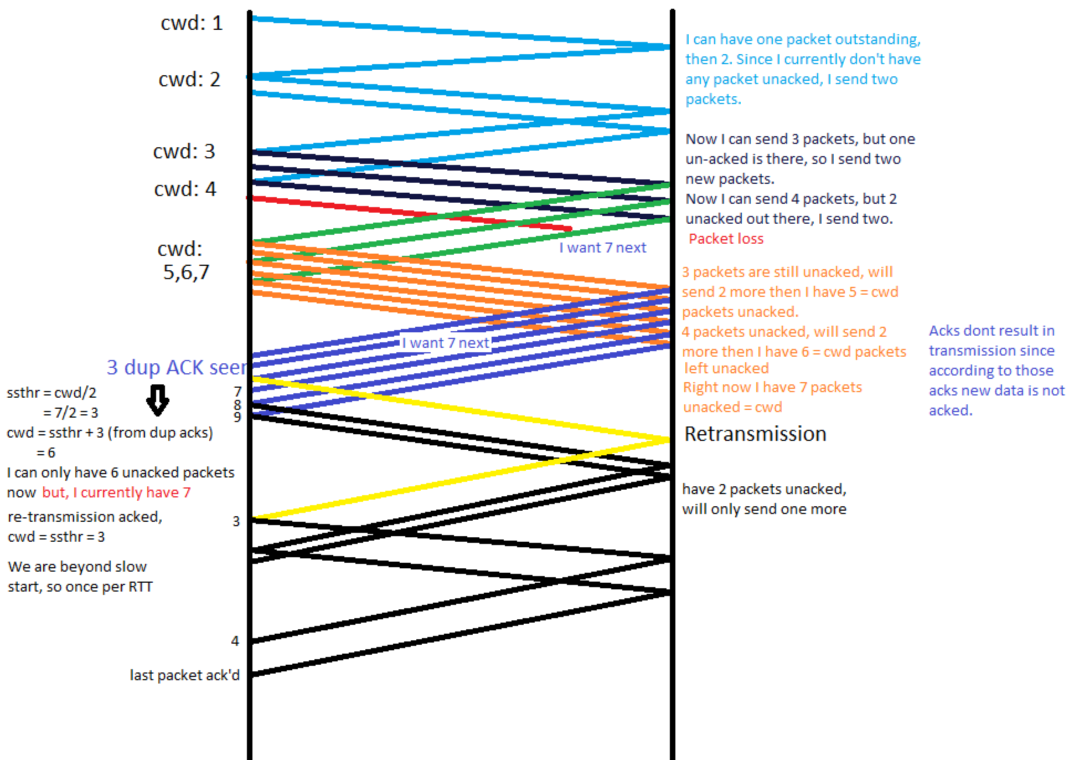

cwd: 1
cwd: 2
cwd: 3
cwd: 4
cwd: 5,6,7
I can have one packet outstanding, then 2. Since I currently don't have any packet unacked, I send two packets.
Now I can send 3 packets, but one un-acked is there, so I send two new packets.
Now I can send 4 packets, but 2 unacked out there, I send two.
Packet loss
I want 7 next
3 packets are still unacked, will send 2 more then I have 5 = cwd packets unacked.
4 packets unacked, will send 2 more then I have 6 = cwd packets left unacked
Right now I have 7 packets unacked = cwd
Acks dont result in transmission since according to those acks new data is not acked.
I want 7 next
3 dup ACK seen
7
8
9
ssthr = cwd/2 = 7/2 = 3
cwd = ssthr + 3 (from dup acks) = 6
I can only have 6 unacked packets now but, I currently have 7
Retransmission
re-transmission acked, cwd = ssthr = 3
3
have 2 packets unacked, will only send one more
We are beyond slow start, so once per RTT
4
last packet ack'd

Figure 19: FRFT

3. TCP CUBIC: follow a cubic curve centered around last ssthr, this way I approach the ssthr carefully, but find the next ssthr quickly by exponential like growth.

6.0.3 Advanced Congestion Control

1. DECBit: This is an active Queue Management Algorithm, since action taken actively. router puts a bit into a header, which then is echoed by the peers acking the fact router queue is about to start droping some packets if the rate of sending packets is this high. A router sets this bit in a packet if its average queue length is greater than or equal to 1 at the time the packet arrives. This average queue length is measured over a time interval that spans the last busy+idle cycle, plus the current busy cycle. (The router is busy when it is transmitting and idle when it is not.)

2. Random Early Detection: Router is starts dropping packet at a probability p, which increases as queue length reaches carrying capacity, in order to force the peers to have triple dup ack situation which automatically reduces traffic. Relies on peers employing TCP.

   (a) AvgLen = (1 - Weight) x AvgLen + Weight x SampleLen
   (b) if AvgLen <= MinThreshold
       i. queue the packet
   (c) if MinThreshold < AvgLen < MaxThreshold
       i. calculate probability P
       ii. drop the arriving packet with probability P
   (d) if MaxThreshold <= AvgLen

i. drop the arriving packet

Time for AvgLen is taken every RTT, since actions are onyl reflected after that time. Higher the bandwidth of a link, it will provide more candidates for drops hence in that sense RED is pretty fair.

3. Explicit Congestion Notification: ECT bit (ECN-Capable Transport). The other bit is set by routers along the end-to-end path when congestion is encountered, as computed by whatever AQM algorithm it is running. This is called the CE bit (Congestion Encountered).

   In addition to these two bits in the IP header (which are transport-agnostic), ECN also includes the addition of two optional flags to the TCP header. The first, ECE (ECN-Echo), communicates from the receiver to the sender that it has received a packet with the CE bit set. The second, CWR (Congestion Window Reduced) communicates from the sender to the receiver that it has reduced the congestion window.

4. TCP Vegas: Use the sampled RTT bas base RTT to calculate expected and actual throughout, then depending on some thresholds increase, decrease the congestion window, or remain the same. This uses the idea that actual rate may be falling if congestion too much, and close to actual rate if congestion low. So if diff of actual and expected is close to 0 $\rightarrow$ congestion low, send more, and if diff too much $\rightarrow$ congestion too much, send less. The congestion window is increased/decreased linearly.

   Reaction to ECE bit should be same as actual packet loss.

6.0.4 Quality of Service

Used for real time applications, where we need guarantee of service. In elastic applications delays are manageable, but for VOIP, etc. it is not. Real networks jitters and packets drops.

1. IP and UDP are best effort protocol: **NO** guarantees about delivery or in time.
2. TCP: solves mostly the delivery guarantee using transmissions, but not in time, only packets are eventually delivered. Goof for file transfer, file will arrive is guaranteed.
3. TCP is not good enough for timely delivery due to jitters, hence appilcations protocol (RTP as before) is based on UDP. There is a delay distribution on the internet for packets, like a gamma distribution.

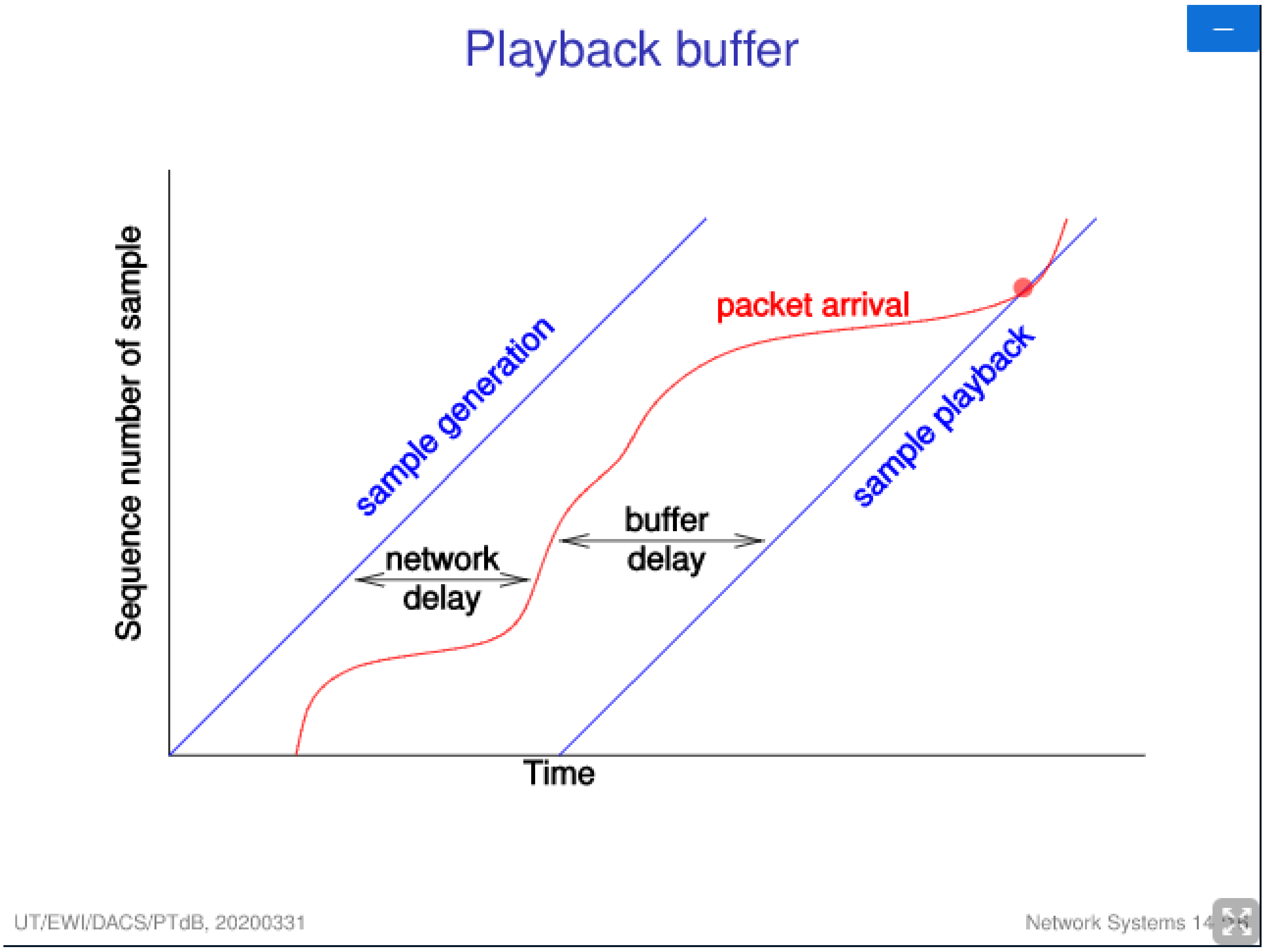


Figure 20: So we can use a threshold like 90% to say we shift the buffer playback by the time by which 90% packets arrive, the rest will be thrown out. So all packets in between the two oblique lines, at left will be the earliest possible, and the right be the latest we can wait. We collect all the packets in between for generaitng the playback at that left line times.

Types of applications

1. Elastic: File Transfers
2. Real Time: VOIP
3. Tolerant: VOIP
4. Intolerant: Surgery Robots
5. Adaptive: VOIP
6. Non-Adaptive: THose who can't handle jitters
7. Rate-Adaptive: VOIP $\rightarrow$ decrease quality
8. Delay Adaptive: VOIP $\rightarrow$ tearing.

Components needed to provide QOS:

1. **Queing discipline**

2. **Connection Treatment**: Priority.

3. **Admission Control**:

   (a) Is it technically possible to provide QoS for both existing with new connection.

      i. Need to know what connection **service** required.

      ii. Need to know **traffic** characteristic of the new flow.

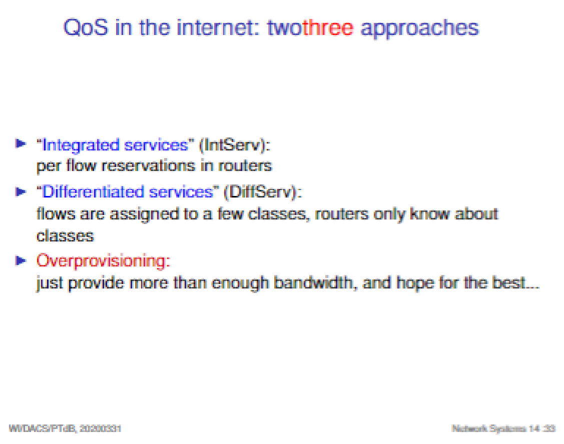
4. **Policing**: Are the existing connections actuallly conforming to the traffic characteristic.

1. Traffic Specification:

   (a) This can be done using the concept of the leaky bucket analogy, r is rate of token filling, B is the bucket size, and R is the rate at which the router promises the provided rate. The rate at which service provided is at best B/R. The average rate is B/2R.

   (b) *It is only a model: it only describes how many packets a system may send when, but the system may use a different method to guarantee that it never transmits more than allowed by this model.*

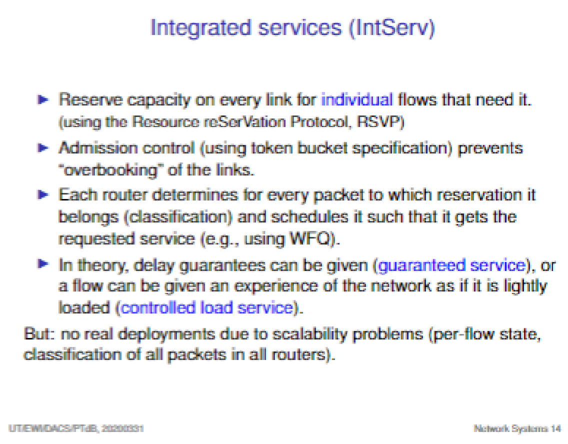
6.0.5 IntServ, DiffServ



QoS in the internet: twothree approaches

- "Integrated services" (IntServ): per flow reservations in routers
- "Differentiated services" (DiffServ): flows are assigned to a few classes, routers only know about classes
- Overprovisioning: just provide more than enough bandwidth, and hope for the best...

W/DACS/PTdB, 20200331 Network Systems 14 :33

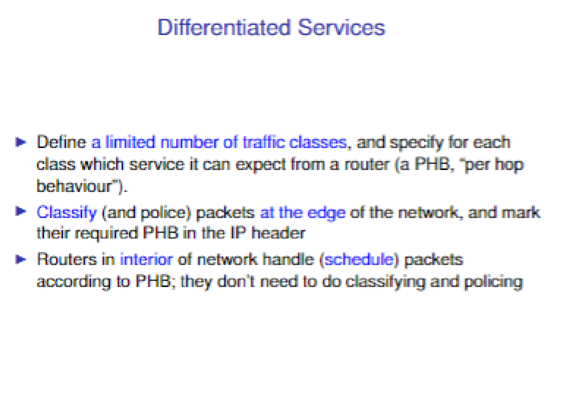


Integrated services (IntServ)

- Reserve capacity on every link for individual flows that need it. (using the Resource reSerVation Protocol, RSVP)
- Admission control (using token bucket specification) prevents "overbooking" of the links.
- Each router determines for every packet to which reservation it belongs (classification) and schedules it such that it gets the requested service (e.g., using WFQ).
- In theory, delay guarantees can be given (guaranteed service), or a flow can be given an experience of the network as if it is lightly loaded (controlled load service).

But: no real deployments due to scalability problems (per-flow state, classification of all packets in all routers).

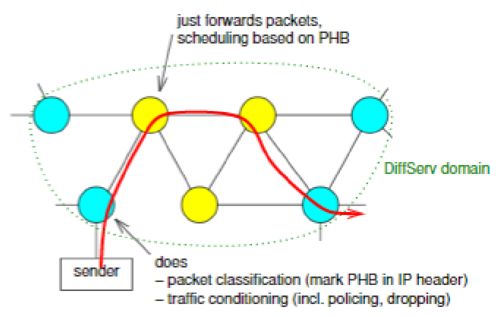
UT/EWI/DACS/PTdB, 20200331 Network Systems 14

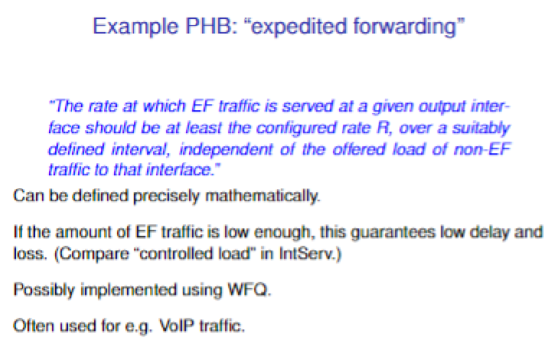


Differentiated Services

- Define a limited number of traffic classes, and specify for each class which service it can expect from a router (a PHB, "per hop behaviour").
- Classify (and police) packets at the edge of the network, and mark their required PHB in the IP header
- Routers in interior of network handle (schedule) packets according to PHB; they don't need to do classifying and policing

Differentiated services – example


just forwards packets, scheduling based on PHB
DiffServ domain
sender
does
– packet classification (mark PHB in IP header)
– traffic conditioning (incl. policing, dropping)



Example PHB: "expedited forwarding"

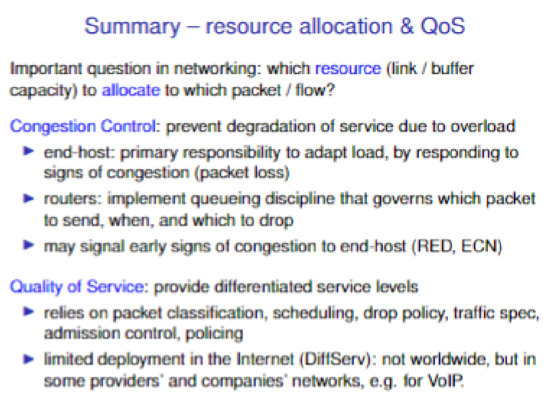
"The rate at which EF traffic is served at a given output interface should be at least the configured rate R, over a suitably defined interval, independent of the offered load of non-EF traffic to that interface."

Can be defined precisely mathematically.

If the amount of EF traffic is low enough, this guarantees low delay and loss. (Compare "controlled load" in IntServ.)

Possibly implemented using WFQ.

Often used for e.g. VoIP traffic.



Summary – resource allocation & QoS

Important question in networking: which resource (link / buffer capacity) to allocate to which packet / flow?

Congestion Control: prevent degradation of service due to overload

- end-host: primary responsibility to adapt load, by responding to signs of congestion (packet loss)
- routers: implement queueing discipline that governs which packet to send, when, and which to drop
- may signal early signs of congestion to end-host (RED, ECN)

Quality of Service: provide differentiated service levels

- relies on packet classification, scheduling, drop policy, traffic spec, admission control, policing
- limited deployment in the Internet (DiffServ): not worldwide, but in some providers' and companies' networks, e.g. for VoIP.

7 Chapter 7: End-to-End data

8 Chapter 8: Network Security

http://www.unixwiz.net/techtips/iguide-ipsec.html

Encrypted using the private key of the sender. So if wrong, then not accepted, if yes considering a trust, we get the data.

WEP- IV, too small, communicated in plain text. Easy to brute force if enough data.

WPA- Authentication Server(Radius), get key from AP

SSH tunnelling only works if you have a device behind the firewall.

(A,0,-) always in Djikstra

ARP is on the user computer.

SeqNum for Link State packets

Unless FIN received remote sender can still send data even if I sent FIN.

WFQ needs Shecduler to be installed too.

Link State packets have sequence numbers if the same sequence number is seen they do not forward it.

Learning Bridges use spanning tree algorithm, so no loops, the packet reaches only the end host.

cwnd += mss^2/cwnd only if ACK is mss, not a smaller byte, Instead use the formula cwnd += mss(n/cwnd) where n is the number of byte ackd in each ack.

-Bridges store all MAC for every computer (40 from that question) -Routers store the range for every LAN, (4 in that question) given route aggregation not possible. - ARP of every computer stores the MAC of every computer on the LAN and all the routers the LAN is connected to. (9+2 = 11 in that question).

The actual window size will be 280*16 = 4480 bytes, and thus include all sequence numbers up to 4599. Which comes from effective window being 120bytes. So if we say window scaling is used, and scale 2^4 = 16 * advertised window, we mean we have space for 280*16 bytes.

Careful with the dvr, difference between getting to know in that update and sending in the same update.

fast transmit at smaller windows and more retransmissions due to re-ordering.

PAWS is WS not used for short fat pipes, large bandwidth and small RTT. wraparound propotional to bandwidth not RTT. On small RTT no window scaling needed but wrap around possible.

window scaling fulfills lack of bits in window header.

ECN reduces packet loss, by telling people of congestion. start 50 / 50 when FQ (not WFQ). then fulfill smaller one and rest for larger one.

Allow packets destination port >= 1024 or use Statefull Firewall. Webcam can be hacked using TCP/IP header, basically if device and is connected to net, we can use firewall to block it.

When pakets are tunneled they get added header bytes, hence more fragmentation therefore MTU is smaller for IPv6 tunneled as IPv4, since less data more header.

LS does not have count to infinity problem period. PIM-SM do not who sender intitially so everyone joins a designated shared tree then connected to sender if there is one.

sending FIN incrementns ACK that is sent.

WS	PAWS	use
0	0	short slim
0	1	short fat
1	0	long slim
1	1	long fat

Table 2: Caption

for the TCP Vegas question, make a line equation between those which have delay. The queuing delay only occurs between them as. Then the gradient is the link speed. By extrapolating to the point where delay is 0, that will give you how many packets are not in queue, so 20-5 = packets in queue.

Difference between updates made and number of iterations done. Large bandwidth delay means convergence to best cwnd is slow. If additive increase parameter increased but not for others, then higher throughput, but not starving.

When senders increased, token not necessarily increased. If total 20 + 70 = 90 < 100. Then 10% idle already implied no need to take ratio.

sometimes only split horizon, and not poison reverse.

RQRQRQRQRQRQRQRQRQRQRQRQRQRQRQRQRQRQRQ

9 Chapter 9: Applications

HTTP vers: 1.0, 1.1

1.0 and 1.0 with parallel connections 1.0 does not have connection persistence, Handshake made, request forwarded, session closed.

1.1 and 1.1 with pipelining. while 1.1 has persistent connections. handshake made, request forwarded, session stays active.