01 - Lesson 3 Intro  
  
At its core, a network serves to route packets between machines on the network. Lets take a look at how packets are moved across networks. It's more complicated than it sounds at first, but quite fascinating. >> That's right. To even reach your screen, the packets that make up this video likely traveled across at least four or five networks, if not more. >> You'll learn how that works in the routing videos on BGP, a routing protocol.

02 - Switching and Bridging  
  
In this lesson, we will learn about switching and bridging. In particular, we will learn about how hosts find each other on a subnet and how subnets are interconnected. We will also learn about the difference between switches and hubs, and the difference between switches and routers. And we'll talk about the scaling problems with Ethernet. And mechanisms that can be used to allow it to scale better.

03 - Bootstrapping Networking Two Hosts  
  
To start lets talk about how you would network two machines, each with a single interface, to each other. So Host 1 and Host 2 would be connected by two Ethernet adapters or network interfaces and each of these would have a LAN or physical or MAC address. Now a host that wants to sent a datagram to another host can simply send that datagram via its Ethernet adapter with a destination MAC address of the other host that it wants to receive the frame. Frames can also be sent to a broadcast destination MAC address which would mean that the datagram would be sent to every host that it was connected to on the local area network. Now, of course, typically what happens is a host knows a DNS name or or an IP address of another host, but it may not know the hardware or MAC address of the adapter on the host that it wants to send it's datagram to. So we need to provide a way for a host to learn the MAC address of another host. The solution to this is a protocol called ARP or the address resolution protocol.

04 - ARP Address Resolution Protocol  
  
In ARP, a host queries with an IP address, broadcasting that query to every other node on the network. That query will be of a form, who has a particular IP address, such as 130.207.160.47, and that particular host. Who has that IP address on the LAN will respond with the appropriate MAC address. So the ARP query is a broadcast that goes to every host on the LAN from the host that wants the answer to the query and the response is a unicast response with the MAC address as the answer. That's returned to the host that issued the query. When the host that issues the query receives a reply, it starts to build what's called an ARP table. It's ARP table then maps each IP address on the local area network to the corresponding MAC address. Now, instead of broadcasting a ARP query to discover the MAC address corresponding with this IP address, the host can simply consult its local ARP table. Let's now take a look at what the host does with this information. When the host wants to send a packet to. The destination with a particular IP address. It takes that IP packet and encapsulates it in an Ethernet frame with the corresponding destination MAC address. Essentially, it puts that IP packet inside of an Ethernet frame. So before it sends the IP packet with that destination IP address it first puts the packet inside a larger Ethernet frame with its own source MAC address and the destination MAC address from its local ARP table.

05 - ARP Quiz  
  
So let's consider what we learned about ARP. So what are the formats of the queries and responses in ARP? Is the query a broadcast where a host is asking about an IP address, and the response is a Unicast with a MAC address? Is the query a Unicast message asking about an IP address and the response is broadcast with a MAC address? Or is the query a broadcast asking about a particular MAC address, where the response is a Unicast with the response of a particular IP address?

06 - ARP Solution  
  
The purpose of ARP is to allow a host to discover the Mac address corresponding to a particular IP address. And the host doesn't know which host on the LAN owns that particular Mac address. So, ARP allows the host to send a broadcast query asking about who owns a particular IP address. And the response comes from the owner of that particular IP address and the response is the Mac address.

07 - Interconnecting LANs with Hubs  
  
The simplest way that a LAN can be connected is with something called a hub. Hubs are the simplest form of interconnection and in some sense they don't even exist in networks anymore today, because you can build a switch for essentially the same price. But for the sake of example, let's just take a look at how a LAN would be connected with a Hub. Now, a hub essentially creates a broadcast medium. Among all of the connected hosts where all packets on the network are seen everywhere. So if a particular host sends a frame that's destined for some other host on the LAN, then a hub will simply broadcast that frame that it receives on an incoming port out every outgoing port. So all packets are seen everywhere. There is a lot of flooding and there are many chances for collision. The chance of collision of course, introduces additional latency in the network because collisions require other hosts or senders to back off and not send as soon as they see the other senders trying to send at the same time. LANs that are connected with hubs are also vulnerable to failures or misconfiguration. Because even one misconfigured device can cause problems for every other device on the LAN. Suppose that you had a misconfigured device that was sending a lot of rogue or unwanted traffic. Well, on a network that's connected with hubs, every other host on the network would see that unwanted traffic. So, we need a way to improve on this broadcast medium by imposing some amount of isolation.

08 - Switches Traffic Isolation  
  
To in contrast, switches perform some amount of traffic isolation so that the entire LAN doesn't become one broadcast medium. But instead, we can partition the LAN into separate broadcast domains or collision domains. Now a switch might break the subnet into multiple LAN segments. Typically a frame that is bound for a host in the same part or segment of the LAN is not forwarded to other segments. So, for example if we had a network with three hubs, all connected by a switch, then each of these would be its own broadcast domain. And if a host here wanted to send a frame to another destination in the same segment, well that frame would be broadcast within that domain. But the switch would recognize that the destination was in the same segment. And would not for the packet on output ports, destined for other LAN segments where the destination was not. Now enforcing this kind of isolation, requires constructing some kind of switch table, or state at the switch, which maps destination MAC addresses to output ports.

09 - Learning Switches  
  
Let's take a quick look at how learning switches work. A learning switch maintains a table between destination addresses and output ports on the switch, so that when it receives a frame destined for a particular place it knows what output port to forward the frame. Initially the forwarding table is empty, so if there's no entry in the forwarding table the switch will simply flood. Let's look at a quick example. If host A sends a frame destined for host C, then initially the switch has nothing in its table to determine where that frame should be sent, so it will flood the frame on all of its outgoing ports. On the other hand, because the frame has a source address of A, and arrived on input port one, the switch can now make an association between address A and port one. In other words, it knows that the host with address A is attached to port one. So that in future, when it sees frames destined for host A, it no longer needs to flood, but can instead send the frames directly to port one. So, for example, when C replies with a frame destined for A, the switch now has an entry that tells it that it doesn't need to flood that packet. But instead, can simply send the packet directly to the output port. Note also that when C replies, the switch learns another association between address C and port three. So future frames destined for host C, no longer need to be flooded, either. They can simply be forwarded to output port three. So, in summary, if a learning switch has no entry in the forwarding table, it must flood the frame on all outgoing ports. But otherwise, it can simply send that frame to the corresponding output port in the table. Note that learning switches do not eliminate all forms of flooding. The learning switch must still flood in cases where there is no corresponding entry in the forwarding table. And also, these switches must forward broadcast frames, such as ARP queries. Now because learning switches still sometimes need to flood, we still have to take care when the network topology has loops. Now most underlying physical topologies have loops for reasons of redundancy. If any particular link fails, you'd still like hosts on the LAN to remain connected. But let's see what happens when the underlying physical topology has a loop. Let's suppose a host on the upper LAN broadcasts a frame. Each learning switch will hear that frame and broadcast it on all of its outgoing ports. When that broadcast occurs, the other learning switches that are in the topology that contains a loop will hear the rebroadcast. They in turn will not know that they shouldn't rebroadcast the packet that they just heard. So each of those switches will in turn rebroadcast the packet on their outgoing ports. And, of course, this process will continue, creating both packet loops and what are known as broadcast storms. So, cycles in the underlying physical topology can create the potential for learning switches to introduce forwarding loops and broadcast storms. So we need some kind of solution to ensure that even if the underlying physical topology has cycles, which it often needs for redundancy, that the switches themselves don't always flood all packets on all outgoing ports. In other words, we need some kind of protocol to create a logical forwarding tree on top of the underlying physical topology.

10 - Learning Switches Quiz  
  
So, as a quick quiz about learning switches, let's suppose that initially the switch forwarding table is empty and host D sends a frame that is destined for host B. Fill out the entry in the switch forwarding table that is populated as a result of this message.

11 - Learning Switches Solution  
  
When the switch sees the frame from host D, destined for host B, it doesn't know what to do with the frame, so it forwards that frame on all of its output ports. However, because it sees a frame arrive from source D, it knows that future frames that are destined for source D should be output on port four.

12 - Spanning Tree  
  
The solution to this problem is to construct what's called a Spanning Tree, which is a loop-free typology that covers every node in the graph. The set of edges, shown in blue, constitutes what's known as a Spanning Tree. The collection of edges in the blue typology covers every node in the underlying physical typology. And yet, there are no loops in the blue topology. Now, instead of flooding a frame, a switch in this topology would simply forward packets along the spanning tree. So for example, this switch would only send a frame along the port corresponding to the blue edge. And would not forward the frame out any edges that were not part of the spanning tree. Other switches that receive the frame, would flood in the same fashion, along all edges that were part of the spanning tree. While omitting edges that were not members of the spanning tree. Let's take a look at how to construct the spanning tree. First the collection of switches must elect a root, since every tree must have a root. Typically this is the switch with the smallest ID. In this case, the switch at the top of the topology is the root. Then each switch must decide which of its links to include in the spanning tree. And it excludes any link if that link is determined to be not on the shortest path to the root. For example, lets consider the switch in the lower right. It has three lengths, this length takes it on a path that's three hops from the root. This length takes it on a path that's two hops to the root, and this length takes it on a path, that's one hop to the root. Any link that's not on a shortest path to the route is excluded and any link that's on a shortest path on a route is included. Simarly here, this edge is on a path that's one hop away from the route and this edge is on a path that's two hops away. So this node will include this link from the spanning tree. Now, each switch repeats this process to exclude links from the underlying topology. And ultimately, this yields a [UNKNOWN] topology that looks like the blue Graph. And of course there is an issue which is how do we determine the root in the first place? Well initially, every node think it's the root. And the switches run an election process to determine which switch has the smallest ID. And if they learn of a switch with a smaller ID, they update their view of the root, and they compute the distance to the new root. Whenever a switch updates its view of the root, it also determines how far it is from that root. So that when other neighboring nodes receive those updates. They can determine their distance to the new root simply by adding one to any message that they receive.

13 - Spanning Tree Example  
  
Let's take a quick example. Suppose the message forma is as follows. Y, d and x, where x is the origin of the message. Y is the node being claimed as root and do is the distance of the particular node sending this message, x from a claimed root. So, initially every switch in the network broadcast a message like x,0,x to indicate that the node that it think itself is the root. When other switches hear this method, they compare the ID of the sender to their owner ID, and they update their opinion of who the root is based on the comparison of these IDs. Let's suppose that we have the following graph and switch number 4 thinks it's the root. So we will send a message 4, 0, 4 to nodes 2 and 7. But 2 also thinks it is the root, so. 4 is going to receive the message 2, 0, 2 from node 2, and then it's going to realize that 4 is just one hop away from node 2. So node 4 will update its view of the root to be node 2, eventually 4 will also hear a message 2, 1, 7 from node 7; indicating that node 7 thinks it is one hop away. From its view of the root, which is node 2. It will realize that the path through node 7 is a longer path to the root, and it will remove the link 4, 7 from the tree. We can repeat this process and ultimately we will end up with a spanning tree.

14 - Switches vs Routers  
  
Let's do a quick comparison of switches and routers. Switches typically operate at layer two. A common protocol at layer two is ethernet. Switches are typically automatically configuring and forwarding tends to be quite fast, since packets only need to be processed through layer two on flat look ups. Routers, on the other hand, typically operate at layer three where IP is the common protocol. And router level topologies are not restricted to a spanning tree. One can even have multipath routing, where a single packet could be sent along one of multiple possible paths in the underlying router level topology. So, in many ways ethernet, or layer two switching, is a lot more convenient, but one of the major limitations is broadcast. The spanning tree protocol messages and ARP queries both impose a fairly high load on the network. So this raises the question of whether it's possible to get many of the benefits of the auto configuration and fast forwarding of layer two without facing these broadcast limitations. As it turns out, there are ways to strike this balance. And in the third part of the course, when we talk about network management, we will look at some ways to scale Ethernet to very large topologies. For example, in data center networks. We'll also explore how an emerging technology called Software Define Networking, or SDN, is effectively blurring the boundary between the layer two and layer three.

15 - Buffer Sizing  
  
So in this lesson, we'll look at an important question in switch design which is how much buffering do routers and switches need? It's fairly well known that routers and switches do need packet buffers to accommodate for statistical multiplexing. But it's less clear how much packet buffering is really necessary. Now given that queuing delay is really the only variable part of packet delay on the internet, you'd think we'd know the answer to this question already. And for quite some time there have been some well understood rules of thumb but it turns out that we've recently revisited this question and come up with some different answers. So let's first look at the universally applied rule of thumb. Now for the sake of the examples in this lesson, I'm going to use routers and switches interchangeably because it doesn't really matter. All that matters here is that we have a network device that's a store and forward packet device. That, has the capability of storing a frame or a packet and then later sending it on. So let's suppose that we have a path between a source and a destination, and the round-trip propagation delay is 2T and the capacity to bottleneck link is C. Now the commonly held view is that this router needs a buffer of 2T times C. It should be clear why this rule of thumb exists. C is the capacity to the bottleneck link in say, bits per second and T is the time of units second, so this works out to bits, and the meaning of this quantity is simply the number of bits that could be outstanding along this path at any given time. It effectively represents the maximum amount of outstanding data that could be on this path between the source and destination at any time. Now this rule of thumb guideline was mandated in many backbone and edge routers for many years. It appears in RFCs and ITF Architectural guidelines and it has major consequences for router design. Simply because this can be a lot of router memory and memory can be expensive. The other thing of course is that the bigger these buffers, not only the bigger the cost but also the bigger the queuing delay that could exist at any given router. And hence, the more delay the interactive traffic may experience and the more delay that feedback about congestion will experience. The longer these delays are, the longer it will take for the source to hear about congestion that might exist in the network. Now to understand why this guideline is incorrect, let's first re-derive the rule of thumb a bit more formally and then we'll understand why it does not always apply in practice.

16 - Buffer Sizing for a TCP Sender  
  
Let's suppose that we have a TCP sender that's sending packets, where the sending rate is controlled by the window W, and it's receiving ACKs. Now at any time if the window's W, only W unacknowledged packets may be outstanding. So the sender's sending rate, R, is simply the TCP window, W, divided by the round trip time of the path. So the rate is W over RTT. Now remember that TCP uses additive increase, multiplicative decrease, or AIMD, congestion control. So for every W ACKs received, we send W plus one packets, and our TCP sawtooth will look something like this. We'll start at a rate W max over 2, increase the window to W max and then when we see a drop we will apply multiplicative decrease and reduce the sender's sending rate to W max over 2 again. So here, right at the point of a packet drop, this represents the maximum number of packets that can be in flight. So again, the required buffer is the maximum number of packets that can be in flight, or simply the height of this TCP sawtooth. Now we know the rate is W over RTT, and we'd like the sender to send at a common rate, R. And if we'd like the sender to be sending at the same rate before and after it experiences a loss, then we know that the rate before the drop must equal the rate after the drop. So then we can set these two rates equal. We know that the RTT is part transmission delay T, and part queuing delay which is the maximum buffer size of the bottleneck link, divided by the capacity of the bottleneck link. We also know that after reducing the window, the queuing delay is zero. So we can replace the term on the left with W old over 2T plus B over C and we can replace the term on the right with W old over 2, because the congestion window has been reduced half divided by 2T. Simply the propagation delay with no queuing delay. Now if we solve this equation we find that the required buffering is simply 2T times C. Now the rule of thumb makes sense for a single flow, but a router in a typical backbone network has more than 20,000 flows. And it turns out that this rule of thumb only really holds if all of the those 20,000 flows are perfectly synchronized. If the flows are desynchronized, then it turns out that this router can get away, with much less buffering.

17 - If TCP Flows are Synchronized  
  
Now, TCP flows are synchronized, the dynamics of the aggregate window as shown in the upper part of the graph, would have the same dynamics as any individual flow. The quantities on the Y axis here would simply be different. Specifically, the number of pockets occupying the buffer would be the sum of all of the TCP flows windows, rather than the window of any individual flow. Now if there are only a small number of flows in the network then these flows may tend to stay synchronized, and the aggregate dynamics might mimic the dynamics of any single flow as shown. But as the network supports an increasingly large number of flows, these individual TCP flows become de-synchronized. So instead of all of the flows lining up with the saw tooth as shown in the bottom part, individual flows might see peaks at different times. As a result, instead of seeing a huge sawtooth that's the sum of a bunch of synchronized flows, the aggregate instead might look quite a bit more smooth, as a result of the individual flows being desynchronized. And we can represent this sum, which is the buffer occupancy, as a random variable. At any given time, it's going to take a particular range of values. The range of values that this buffer occupancy takes, can actually be analyzed in terms of the central limit theorem. The central limit theorem tells us that the more variables that we have, and, in this case the number of variables are the number of unique congestion windows of flows that we have. The narrowerer the Gaussian will be. In this case, the Gaussian is the fluctuation of the sum of all of the congestion windows. In fact the width decreases as 1/ root N, where N is the number of unique, congestion windows of flows that we have. And therefore, instead of the required buffering, needing to be 2T times C. We can get away, with much less buffering. In particular, 2 times C divided by the square root of n. Where N, is the number of flows, passing through the router.