0:00:00 Let's go ahead and get started, you guys. 0:00:05 0:00:12 OK, so before we begin, I just want a brief note about 0:00:17.088 the quiz. So you should get the quiz back 0:00:20.928 tomorrow in recitation. The mean and median on the quiz 0:00:26.112 were 69 out of 100, so look for it tomorrow. 0:00:30.24 And there should be solutions post it also tomorrow. 0:00:36 Does anybody or member with standard deviation was on the 0:00:40.896 quiz? OK, thanks. 0:00:42.295 OK, so the topic for today, we're going to continue talking 0:00:47.366 about networking. And just to recap where we left 0:00:51.562 off last time, I wanted to review a few of the 0:00:55.497 best effort network properties. So you remember at the end of 0:01:00.743 lecture last time we talked about this notion of a best 0:01:05.464 effort network. And we said that these packet 0:01:10.657 switch networks like the Internet are typically best 0:01:15.329 effort, which means that there are certain things in particular 0:01:21.01 that they don't guarantee. So one property of best effort 0:01:26.141 networks is that they're subject to delays. 0:01:29.989 So these are delays due to the propagation of messages down our 0:01:35.67 wire, the time it takes for the messages to propagate along the 0:01:41.35 pipe. Remember we talked about that 0:01:45.249 analogy last time? Time to transmit the message, 0:01:48.894 so transmission time is sort of the amount of time it takes 0:01:53.393 proportional to the length of the message and the bit rate at 0:01:58.047 the link. And then, there is additional 0:02:00.994 delays that are introduced by these properties that we talked 0:02:05.648 about at the end of the lecture last time such as queuing. 0:02:11 So the fact that remember we said that these networks are 0:02:14.32 congested and the load is somewhat unpredictable. 0:02:17.167 And because the load of the network is unpredictable, 0:02:20.25 that necessarily means that there are going to be queues in 0:02:23.69 between the links on our network. 0:02:25.587 And then finally, additional sorts of delay that 0:02:28.374 we didn't talk much about, but also can be an issue as the 0:02:31.161 processing delay. So this is the delay incurred 0:02:35.043 by sort of the overhead of processing messages between 0:02:38.776 switches, for example, in our network. 0:02:41.383 OK, so other interesting properties of best effort 0:02:44.835 networks, they incur some loss. OK, so there are going to be 0:02:48.992 messages that are corrupted as they are transmitted down these 0:02:53.289 links. There are going to be losses 0:02:55.684 that are caused by congestions. Remember we talked about how 0:02:59.841 one of the only responses, oftentimes the best response to 0:03:03.857 congestion is simply to drop packets to reduce overload. 0:03:09 And there are sometimes going to be failures that are going to 0:03:12.873 be switches within the network that will sometimes simply cease 0:03:16.809 to function. And of course you're going to 0:03:19.412 lose the data when that happens. And there can be reordering of 0:03:23.349 packets. So you didn't talk much about 0:03:25.698 this. But it's a fairly sort of obvious 0:03:27.984 idea that suppose an application has two different messages, 0:03:31.73 right? Well if you think about the way 0:03:34.811 that those messages may be routed along different paths 0:03:37.967 through the network. And we'll talk about this more 0:03:40.889 when we talk about how routing works next time. 0:03:43.577 But because they can be routed along different paths, 0:03:46.616 there's no guarantee that the messages, because message one 0:03:50.006 was sent before message two, that message one actually 0:03:53.103 arrives before message two at whatever endpoint you're sending 0:03:56.668 to. And finally, 0:03:57.545 there can be duplication. And duplication is often a side 0:04:01.631 effect of techniques that we'll again talk about next time for 0:04:04.842 dealing with this issue of loss. So if you have lost packets, 0:04:08 somewhere in the network you may want to try and retransmit 0:04:11.052 those packets. And because you're 0:04:12.736 retransmitting things, you're sending things multiple 0:04:15.473 times. There can be places in the 0:04:17.157 network where you actually see two copies of a message. 0:04:20 So these best effort networks have a number of sort of 0:04:22.789 properties that make using them somewhat complex. 0:04:25.315 And what we're going to talk about over the next two or three 0:04:28.473 lectures basically are ways of mitigating the complexity of 0:04:31.526 these best effort networks, of making the interface that 0:04:34.421 these best effort networks provide to applications a little 0:04:37.473 bit more usable in the sense that we're going to try and find 0:04:40.631 ways to reduce the effects of things like loss and reordering 0:04:43.789 or to make it so that applications don't necessarily 0:04:46.526 need to be aware of those things. 0:04:50 So in order to start getting at sort of dealing with complexity, 0:04:54.464 we're going to talk today about this notion of layering. 0:04:58.362 So -- 0:04:59 0:05:08 So when we talked generically about systems earlier in the 0:05:11.98 class, we said that a standard way that we're going to deal 0:05:16.031 with complexity is by introducing modularity. 0:05:19.104 And the specific kind of modularity that is widely used 0:05:22.876 within networks is layering. And that's what we're going to 0:05:26.926 talk about most of the time today. 0:05:30 The other way that we typically deal with, another common way of 0:05:34.883 dealing with complexity with the networks is by using protocols. 0:05:40 0:05:44 And a protocol is really just a set of rules for how two 0:05:47.676 machines on the network do communicate with each other. 0:05:51.286 So, for example, a protocol might say that if I 0:05:54.362 want to tell you about, if I want to send a message to 0:05:57.905 you that I will put this certain bit of information at the 0:06:01.715 beginning of it, I'll put some address 0:06:04.189 information at the beginning of that message. 0:06:08 And the message will be no more than a certain number of bytes 0:06:11.201 long. And I'll put that message on 0:06:12.932 the wire, and then part of the protocol is also that you will 0:06:16.081 send an acknowledgment back to me that tells me that you in 0:06:19.125 fact got the message. So these protocols are very 0:06:21.644 well defined sets of rules about how communication is supposed to 0:06:25.002 happen. These protocols are going to 0:06:27.343 help us to deal with complexity because they are going to tell 0:06:30.268 us what we expect to have happen within the network. 0:06:32.715 And when things don't follow the protocols, 0:06:34.729 then that's going to be an indication that something went 0:06:37.415 wrong, and that we should therefore try and recover from 0:06:40.053 it. So, for example, 0:06:40.965 if we lose a packet, the protocol is going to allow 0:06:43.363 us to detect that a packet was lost and request that that be 0:06:46.193 retransmitted. We'll talk much more about 0:06:48.111 protocols next time, but it's sort of worth bearing 0:06:50.51 in mind that almost at any time, two things are communicating 0:06:53.388 within a network they are using one of these protocols. 0:06:55.978 And a protocol is just a set of rules. 0:06:59 OK so I want to just quickly give you this list of five 0:07:02.075 questions. We're going to come back to 0:07:04.182 this list of five questions a little bit later. 0:07:06.802 But these are sort of five key questions that we need to sort 0:07:10.219 of figure out how we're going to actually resolve in networks. 0:07:13.693 And the first question is, how do we multiplex 0:07:16.256 conversations on a network? When we talked about this last 0:07:19.502 time, we have these two different alternatives, 0:07:22.122 time division multiplexing, which we said was used in 0:07:25.083 telephone networks. And we talked about packet 0:07:27.646 switching as a way in which we can have multiple people sharing 0:07:31.177 a network. But there is a bunch of other 0:07:34.515 questions that we haven't really answered yet. 0:07:36.866 And this is what we're going to get at over the next few 0:07:39.74 lectures. So how do we transmit bits on a 0:07:41.83 link? How do we actually, 0:07:43.084 given a wire, how do we actually modulate 0:07:45.174 that wire in the right way to transmit the message that we 0:07:48.153 want to transmit down it? How do we forward packets via 0:07:50.974 switches? So we sort of said that there 0:07:52.96 are these switches, and that along any 0:07:54.893 communication path, there may be sort of multiple 0:07:57.401 communication hops that have to be taken through multiple 0:08:00.327 switches. But we haven't yet talked about 0:08:03.544 what is actually going on inside those switches. 0:08:05.886 So we are going to look in particular at what's going on 0:08:08.626 inside switches in the Internet and see how that works. 0:08:11.316 Then there's this question about, how do we actually make 0:08:14.106 communication reliable? So we said that one of the 0:08:16.548 limitations of a best effort networks is that, 0:08:18.79 well, it introduces loss, and it can reorder packets. 0:08:21.38 Applications oftentimes don't want to have their packets 0:08:24.12 reordered or dropped, right, because they're trying 0:08:26.612 to actually exchange information with each other. 0:08:30 So we'd like to understand what kinds of techniques we can use 0:08:33.308 that can systematically allow us to avoid, or to make 0:08:36.128 communication reliable. And then, finally, 0:08:38.352 how do we manage congestion? So we said that one of the 0:08:41.281 fundamental problems with these packet switch networks is that 0:08:44.589 the amount of load may be unpredictable in the network. 0:08:47.518 So the question that we need to ask, then, is, 0:08:49.959 OK, how are we going to manage this congestion? 0:08:52.454 What are we going to do in response to congestion in order 0:08:55.545 to sort of minimize the effect that it has on the applications 0:08:58.854 that are trying to run on the network. 0:09:02 OK, so we'll come back to these questions in a minute, 0:09:05.38 and as we talk about what the layers of the network stack are, 0:09:09.271 we'll sort of see how these different questions fit in to 0:09:12.843 these different parts of the network stack. 0:09:15.522 But what I want to do now is really turn to this issue of 0:09:19.094 layering. OK, so in order to understand a 0:09:21.646 little bit about why there might be wires within a network, 0:09:25.345 or a why there might be layers within a network, 0:09:28.343 excuse me, let's look at a very simple kind of a network. 0:09:33 So suppose we have some client, a client, C, 0:09:36.463 OK, who has a number of different connections to the 0:09:40.572 outside world available to it. And, it can connect each of 0:09:45.163 those connections go, say, for example, 0:09:48.225 to some switch which may in turn connect to another switch 0:09:52.816 which may in turn connect to some N host S, 0:09:56.2 server S, that the client is trying to communicate with, 0:10:00.63 OK? So, suppose C is trying to send 0:10:04.645 a message to S. Let's think a little bit about 0:10:08.485 how this might actually work. So we have the application, 0:10:13.264 which is perhaps running on C. And, it might try and send a 0:10:18.213 message by calling some routine send S message. 0:10:22.138 OK, so it says send to endpoint S this message that I have. 0:10:27.087 So let's think about how we might actually go about 0:10:31.354 implementing this. So one way that you might have 0:10:35.687 the application send a message to S is that you might have this 0:10:39.174 sort of the application might understand everything about what 0:10:42.604 the whole network topology looks like, right? 0:10:45.079 So, it might understand, know about all the different 0:10:48.004 links that are available to it, and with each one of these 0:10:51.209 different links, it may understand what the 0:10:53.572 topology of devices that are out there. 0:10:55.709 So if I have a connection to the Internet, 0:10:58.015 that would mean that I have to understand all of the machines 0:11:01.389 that are connected to me via the Internet, right, 0:11:04.089 so I have this list of a million hosts, 0:11:06.226 and those are at every client, and I sort of just scan 0:11:09.432 down this list of a million machines that I can connect to 0:11:12.637 until I find S. And then I send a message out 0:11:16.342 over the next link or something, right? 0:11:18.101 So this sounds very complicated because it forces the 0:11:20.509 application to understand the sort of entire functionality of 0:11:23.287 the network underneath it, how the network is connected, 0:11:25.833 and how the nodes talk to each other. 0:11:27.5 So that doesn't seem like a very good idea. 0:11:30 Right, instead we'd like to application simply to be able to 0:11:34.3 just send its message out. And we would like some lower 0:11:38.236 layer service to take care of the details of figuring out 0:11:42.318 where to send the message next. So, in networks, 0:11:45.744 we talk about that sort of service that runs underneath the 0:11:49.972 application, that's in charge of figuring out what the next sort 0:11:54.564 of connection, the next top to use within the 0:11:57.772 network, we call that the network layer, 0:12:00.615 OK? So, I'm going to abbreviate as 0:12:03.81 NET. And so, what the network layer, 0:12:05.922 for example, suppose this client had three 0:12:08.396 connections available to it. It has a modem connection. 0:12:11.655 It has a WiFi connection. And maybe it has an Ethernet 0:12:14.853 connection. OK? What the network layer is going 0:12:17.387 to do is it's going to look at this name, S, 0:12:19.982 that the client has specified. And it's going to try and 0:12:23.301 decide which one of these links is the next link to use in order 0:12:27.103 to forward the message on to S. OK, so it's just going to make 0:12:32.064 a decision from amongst the available connections. 0:12:35.435 So, it's going to basically pick the next link, 0:12:38.6 OK? And if this is the Internet, 0:12:40.733 you're going to have these switches or these routers that 0:12:44.587 may have links to a bunch of other networks. 0:12:47.545 And so in the Internet, the network layer runs actually 0:12:51.261 on all of the routers within the Internet, and is making these 0:12:55.458 decisions about how to forward packet after packet. 0:13:00 So we'll see again in the next lecture and in recitation how 0:13:03.757 the Internet actually does packet forwarding. 0:13:06.56 But so we said there is this network layer. 0:13:09.235 But we have this thing that's picking the next link to send 0:13:12.929 the message out over, right? 0:13:14.649 But we don't necessarily want that thing to have to understand 0:13:18.535 the details of how you actually communicate over each of the 0:13:22.292 available physical connections, right? 0:13:24.649 So suppose that the only thing that was here was this network 0:13:28.471 layer. And the network layer had to 0:13:31.677 understand how to communicate over Ethernet and WiFi and the 0:13:35.213 modem. Right, well then the network 0:13:37.25 layer is going to be very complicated because, 0:13:39.947 of course, sending messages out over a wireless radio is very 0:13:43.543 different than sending messages out over an Ethernet. 0:13:46.659 So what we have is one layer that sits underneath the network 0:13:50.254 layer, which we typically call the link layer. 0:13:52.951 And the link layer is responsible for managing the 0:13:55.887 physical connection for the transmission of data from a long 0:13:59.423 just one of these wires. It's the thing that actually 0:14:03.95 moves bits from C to whatever the next top within the network 0:14:08.131 is, OK? So, these are these two layers. 0:14:10.778 You notice now, I've left this hole here. 0:14:13.565 You might be wondering what goes into that hole. 0:14:16.84 So what we said is so the network layer is responsible for 0:14:20.811 picking the links. But remember that what we 0:14:23.807 talked about so far in this best effort network abstraction is 0:14:28.057 sort of there's all these problems, decent sort of with 0:14:31.819 delays, reordering, duplication, 0:14:33.979 and so on. And we might be that neither of 0:14:37.746 these layers really is responsible for dealing with 0:14:40.563 those problems. What we said is, 0:14:42.309 what we want, to be able to provide an 0:14:44.394 abstraction for applications where some of these problems 0:14:47.549 with the best effort networks are hidden from the network. 0:14:50.76 OK, so typically networks introduce a third layer, 0:14:53.521 which in this class we call the end-to-end layer that's in 0:14:56.732 charge of addressing these kinds of issues. 0:15:00 The end-to-end layer may seem a little bit fuzzy, 0:15:02.745 the details of it when we talk about it because the end-to-end 0:15:06.234 layer can do lots of different things for different 0:15:09.094 applications. OK, so some applications may be 0:15:11.611 concerned about, for example, 0:15:13.213 the possibility of messages being lost, right, 0:15:15.786 whereas other applications may not be as concerned about 0:15:18.932 messages being lost, but may be very concerned about 0:15:21.85 delay. And we'll see as we talk to the 0:15:23.966 class that delay and loss trade off with each other which makes 0:15:27.512 sense. If I lose a message and I have 0:15:30.771 to retransmit it, that's going to increase the 0:15:33.428 delay on the network. But, so it's possible to sort 0:15:36.38 of trade these things off for each other. 0:15:38.742 So the end-to-end layer is the thing that's responsible for 0:15:42.167 trying to make the application environment sort of more 0:15:45.12 pleasant for the application. And that can mean dealing with 0:15:48.603 trying to eliminate loss or trying to minimize delay, 0:15:51.674 for example. We'll talk about different 0:15:53.918 kinds of end-to-end layers in just a few lectures. 0:15:56.811 OK, so these are kind of the three. 0:16:00 So we're going to decompose our network into these three layers. 0:16:03.644 And in order to sort illustrate this to you a little better, 0:16:07.057 what I want to do is just walk you through a simple example of 0:16:10.586 how the letters might look in the Internet with a simple web 0:16:14 application. And I'm going to use some of 0:16:16.314 that terminology from the Internet here, 0:16:18.57 and we're going to return to some of this terminology, 0:16:21.636 introduced its older more carefully in recitation. 0:16:24.471 But I'll try and explain it as much as we go. 0:16:28 So suppose we have a laptop, say my laptop here, 0:16:32.309 that wants to connect to a Web server, MIT.edu. 0:16:36.527 And, the way that the Web works is it uses a protocol called 0:16:41.936 HTTP, which is used for making requests for specific webpages, 0:16:47.53 and for returning the results of those webpages. 0:16:51.839 So these are typically called requests and responses in the 0:16:57.157 HTTP specification. Now, if you look at, 0:17:01.193 so this is sort of from the user's perspective; 0:17:03.938 the application is a browser that knows about the HTTP 0:17:07.1 protocol, or a Web server that knows about the HTTP protocol. 0:17:10.681 And it's running on these two remote machines. 0:17:13.366 Underneath each of these things, of course, 0:17:15.872 there is a set of layers. And each of these things has 0:17:19.034 layers underneath it. And these layers correspond to 0:17:22.078 these three things we just talked about, 0:17:24.405 the end-to-end layer, the network layer, 0:17:26.732 and the link layer. So one thing you may notice 0:17:30.667 here is that the link layer is different between these two 0:17:34.187 things. So the laptop is perhaps 0:17:36.101 communicating over WiFi, and the Web server is 0:17:38.88 communicating over the Ethernet. But otherwise these two things 0:17:42.709 have the same; their Ethernet layer and their 0:17:45.426 network layer are TCP and IP. So, TCP is going to be in 0:17:48.76 charge of basically providing this reliable abstraction for 0:17:52.342 us. We're not going to talk about 0:17:54.318 it anymore today, except to say that, 0:17:56.541 that it makes communication reliable. 0:18:00 We'll see how it does that later. 0:18:02.076 What IP is responsible for is choosing the sort of next top to 0:18:06.034 make along each connection of the way, each connection within 0:18:09.927 the Internet. So, IP is the protocol that 0:18:12.522 runs the Internet. And, you may have seen IP 0:18:15.312 addresses. So, IP addresses are the sort 0:18:17.843 of names for the endpoints in the Internet, 0:18:20.568 and takes an IP address and basically gives you this next 0:18:24.202 link that you should use in order to transmit a message out 0:18:27.965 over it. OK, so suppose that the browser 0:18:30.496 generates a request for some page. 0:18:34 What it's going to do is it's going to call send on the 0:18:37.06 end-to-end layer. OK, so I've just written this 0:18:39.668 as e2esend to make it clear that this is a send request to 0:18:43.125 the E to E layer. And it's going to pass some 0:18:45.449 message. In this case, 0:18:46.639 the message is simply going to be the contents of this request. 0:18:50.153 And it's going to specify the underlying protocol that it 0:18:53.327 should use as well as the destination address. 0:18:55.878 So this address is one of these Internet addresses and IP 0:18:59.052 address followed by this colon 80. 0:19:02 So, what colon 80 does is it identifies what's called a port 0:19:05.709 number. And it identifies the 0:19:07.469 application we're running on the remote server that we want to 0:19:11.304 communicate with. So, typically Web servers run 0:19:14.196 on port. Now, you say they run on port 0:19:16.522 number 80. That just means that the TCP 0:19:18.911 layer on the other side knows how to communicate, 0:19:21.929 knows that the Web server on the other end is connected on 0:19:25.512 this port number 80. So it gives us a way to 0:19:28.216 communicate, to identify applications that are running on 0:19:31.736 the remote host. So now what happens is this 0:19:35.672 request is going to be, the TCP layer is going to take 0:19:38.955 this request, which I've shown in blue, 0:19:41.309 and it's going to attach what are called headers and trailers 0:19:45.026 to it. So these headers and trailers 0:19:47.194 are the information that TCP needs, the TCP layer on the 0:19:50.601 other side is going to need in order to deliver this message to 0:19:54.442 the application. So in particular, 0:19:56.486 this thing is going to contain this port number that we already 0:20:00.327 mentioned. So this is going to be used on 0:20:03.878 the other end in order to send the message to the Web server. 0:20:07.636 It's also going to have a sequence number, 0:20:10.204 which as we will see later, we're going to use in order to, 0:20:13.837 say for example, reorder in order to detect 0:20:16.279 reordered messages or detect lost messages. 0:20:18.91 OK, so the TCP layer now is just going to repeat the same 0:20:22.417 thing. It's just going to take this 0:20:24.546 packet, and it's going to call some request, 0:20:27.24 say, netsend on the IP layer that sits underneath it. 0:20:32 And again, it's going to specify the set of data that it 0:20:35.177 built up, which we called a segment. 0:20:37.199 So that's what SEG is, which was the purple and the 0:20:40.087 blue blocks. And it's going to pass that on 0:20:42.513 to the IP header using this netsend message to the IP layer. 0:20:45.922 And it's going to tell the IP layer what IP address it wants 0:20:49.33 to send this message to. Now, the IP layer is going to 0:20:52.392 take this message, and it's going to put the 0:20:54.876 header on it. So IP doesn't use a trailer, 0:20:57.245 although it could, in principle, 0:20:59.035 use a trailer. And this IP header is going to 0:21:02.611 sort of just contain the IP address of the next top, 0:21:05.351 or of the destination. OK, now the IP layer is going 0:21:08.091 to do exactly what we said. The IP layer is our network 0:21:10.993 layer so it's going to do exactly what we said it does 0:21:13.84 before. It's going to look at all the 0:21:15.774 available links that it has to it, and it's going to send this 0:21:19.051 message out over one of those links that it believes is the 0:21:22.168 correct next top in forwarding. So it's going to call link 0:21:25.23 send. It's going to pass this packet 0:21:27.11 on, and it's going to specify the name of the link that it 0:21:30.173 wants to use. And it may have to specify some 0:21:33.905 address information, for example, 0:21:35.872 what I've shown here is just colon one, which is just to say 0:21:39.192 whatever machine is one the wireless network at wireless 0:21:42.573 address number one. So you guys saw sort of a 0:21:45.278 similar addressing scheme being used in Ethernet last time. 0:21:48.844 You can sort of think of that the same here. 0:21:51.487 So what happens now is of course the same process. 0:21:54.5 The link layer attaches its header and trailer. 0:21:57.327 So, I've called WH and WT for WiFi header and WiFi trailer. 0:22:02 And now at this point we called this thing a frame. 0:22:05.478 And this frame is now ready to be delivered out sort of along 0:22:09.653 the next top of the network. So suppose that the network 0:22:13.48 layer identified one particular switch as the next destination 0:22:17.724 of this packet. It's going to send this out 0:22:20.647 over the network to the WiFi interface of this switch. 0:22:24.334 The switch is going to receive this message. 0:22:27.326 What it's going to do is it's going to take the WiFi header 0:22:31.362 and the WiFi trailer, and it's going to peel them off 0:22:34.98 of the message, and it's going to pass the 0:22:37.833 message with just the network header, no more WiFi header on 0:22:41.938 it up to the IP layer. Now the IP layer now has an 0:22:47.076 exact copy of the sort of message including the IP header 0:22:51.384 from the laptop. And, so what the IP header 0:22:54.615 layer will do is look at its IP header labeled NH here, 0:22:58.769 and it'll decide what the next appropriate hop to use, 0:23:02.846 to send this message out, is. 0:23:06 So it's going to then pick the next link to send a message out 0:23:09.376 over and, say, in this case, 0:23:10.87 it decides to send the message over an Ethernet connection. 0:23:14.081 The Ethernet connection is going to receive the message. 0:23:17.125 It's going to attach its header and its trailer to it. 0:23:20.059 And then it's going to send it out to the next link, 0:23:22.881 OK? So this process just repeats. 0:23:24.653 The message gets sent to the Ethernet link on the other side. 0:23:27.974 The Ethernet link forwards the message onto the IP layer. 0:23:31.073 The IP layer decodes the message, decides what the next 0:23:34.062 link to use is. In this case, 0:23:36.674 it decides again to use an Ethernet link. 0:23:38.748 And it sends the message out over the Ethernet. 0:23:41.134 Finally we get to MIT.edu. And once we get to MIT.edu, 0:23:43.883 we just start forwarding this message up the layers, 0:23:46.529 OK? So, we do what are called 0:23:47.981 up-calls from the lower layers to the higher layers, 0:23:50.626 notifying them that a message has arrived. 0:23:52.753 So, the Ethernet layer peels its headers off, 0:23:55.035 sends them up to the IP layer. The IP layer peels its headers 0:23:58.147 off, sends them up to the TCP layer. 0:24:01 Then remember, we said the TCP layer, 0:24:03.082 so the TCP layer has a port number that's associated with 0:24:06.322 it. The port number is used to 0:24:08 identify the application that should receive this message. 0:24:11.297 So, the TCP layer pulls out the port and sends it up to the web 0:24:14.884 server, which finally receives our request, OK? 0:24:17.545 Now the Web server does whatever it does. 0:24:19.859 It chews on this request for a while, and say, 0:24:22.462 for example, generates a webpage, 0:24:24.314 generates some HTML that it's going to send back to the 0:24:27.438 client. And now this process just 0:24:30.71 repeats all over again. The MIT.edu sends a message 0:24:34.131 back down to TCP identifying the client as the endpoint that it 0:24:38.373 wants the message to reach, OK? 0:24:40.426 So this is sort of the basic way in which we use layering. 0:24:44.326 What I want to do is now kind of step back and look at, 0:24:48.021 I sort of presented this as a very quick example. 0:24:51.305 But what I want to do is step back and look at some of the 0:24:55.205 sort of rules that we are following as we use these 0:24:58.626 layers, and to sort of talk about why we have sort of 0:25:02.184 constructivist thing in the exact way that we have 0:25:05.536 constructed it. So the first rule that we are 0:25:11.782 following here is called encapsulation. 0:25:17 0:25:26 So what encapsulation is, is it's simply this way in 0:25:28.952 which you notice that when we send messages out, 0:25:31.673 each layer associated its own header and trailer with those 0:25:35.031 messages that were sent out. And it didn't modify anything 0:25:39.831 about any of this sort of data that wasn't associated with the 0:25:45.787 header and trailer for that layer, OK? 0:25:49.4 So, encapsulation says that each layer may add or remove 0:25:54.771 depending on whether we're sending a message down or are 0:26:00.141 sending a message up. Its own headers or trailers. 0:26:06 OK, but that layer doesn't touch, doesn't look at or use 0:26:14.125 the payload from higher layers. 0:26:19 0:26:26 OK, so the link layer doesn't look at anything that the end to 0:26:29.445 end layer sends it. It simply treats this as a 0:26:31.986 block of data that it has to transmit, and it doesn't 0:26:34.923 understand anything about what the contents of that block of 0:26:38.255 data are. It doesn't assume anything 0:26:40.232 about the contents of that block of data. 0:26:42.491 Similarly, the network layer doesn't assume anything about 0:26:45.71 the contents of the data that's received from the end to end layer. 0:26:49.212 And similarly, the end to end layer doesn't 0:26:51.584 assume anything about the format or the layout of the data that's 0:26:55.199 received from the application layer, OK, or from the 0:26:58.079 application. So what this layering 0:27:01.737 abstraction buys us is that it allows these things to sort of 0:27:06.702 coexist without any understanding of what the other 0:27:10.839 layers do. So in particular, 0:27:13.073 it means that we can, for example, 0:27:15.803 change something about the format of the data that the 0:27:20.189 network layer sends. And we continue to use the same 0:27:24.408 link layer. OK, so I can send data out over 0:27:27.718 an Ethernet that's not data that's been sort of packaged up 0:27:32.517 by IP, right? It doesn't necessarily have to 0:27:36.878 be an IP packet to send it out over Ethernet. 0:27:39.634 And so, this sort of separation between the layers is going to 0:27:43.455 be really critical for allowing us to sort of maintain and 0:27:47.025 develop new networking code over time. 0:27:49.342 And it also means that the people who provide these sort of 0:27:52.849 companies that build and sell software and hardware that works 0:27:56.67 at these different layers don't really have to assume very much 0:28:00.553 about what the other layers are going to provide, 0:28:03.559 right? So if I'm making an Ethernet 0:28:06.719 card, I don't have to understand anything about exactly what, 0:28:10.405 I hopefully won't have to understand anything about 0:28:13.476 exactly what's sort of going on up at the higher levels of the 0:28:17.223 network stack. You have to be a little bit 0:28:19.742 careful, though, because of course the 0:28:22.014 individual layers do have some protocol that there is some sort 0:28:25.823 of API that they're using to interface with each other. 0:28:30 So, API is an application programming interface. 0:28:32.729 You have some set of routines that they call on each other. 0:28:36.098 So for example, I showed in this example up 0:28:38.538 here that the networking layer is calling this link send 0:28:41.733 message on the link layer below it. 0:28:43.708 So the link layer has to provide this link send 0:28:46.379 interface. And similarly, 0:28:47.773 there's a comparable interface when the link layer receives a 0:28:51.259 message that it uses to send up to the network layer. 0:28:54.105 OK, but basically what this means is that we can develop the 0:28:57.532 software that runs at the different layers in isolation 0:29:00.668 without having to worry too much about what's going on at the 0:29:04.153 layers above or below us. OK, so -- 0:29:11 0:29:19 Let's return back to our set of questions now and talk a little 0:29:24.683 bit about how these questions map onto our three layers. 0:29:30 OK, so our first question is, so we said question one is, 0:29:34.107 we've sort of already addressed that. 0:29:36.748 But question two is, well, how do we transmit bits 0:29:40.342 on a link? OK, so clearly that's going to 0:29:43.276 be handled by layer two, OK, or by the link layer, 0:29:46.87 sorry. And, so the link layer is the 0:29:49.437 thing that's going to be in charge of actually pushing the 0:29:53.618 bits onto the link. And none of the other layers 0:29:57.066 need to know about this. Question three, 0:30:00.986 OK, how do we forward packets via switches? 0:30:03.748 Well, it seems that it's pretty clearly what's happening at the 0:30:07.825 network layer. OK, so the network layer is the 0:30:10.785 thing that's deciding sort of which packet, 0:30:13.547 what the next link that we should use is within a switch. 0:30:17.23 OK, and now questions four and five are these questions about 0:30:21.176 this kind of our best effort network properties. 0:30:23.806 How do we achieve reliable communication? 0:30:26.437 How we manage congestion? These are things that were 0:30:29.791 going to worry about at the end to end layer. 0:30:34 OK, so just to sort of illustrate a little bit more 0:30:37.174 about sort of how the commercial, and how these things 0:30:40.539 are separated in sort of a commercial world, 0:30:43.269 I thought I'd just show you the slide that sort of illustrates 0:30:47.142 that there are lots of different vendors, both hardware and 0:30:50.825 software, that run at each one of these different layers. 0:30:54.38 So, for example, at the end to end layer, 0:30:56.92 first there are clearly a bunch of applications that run up 0:31:00.603 there. And each of those applications 0:31:03.991 perhaps has some different sort of set of requirements about how 0:31:08.172 data is delivered. And those applications, 0:31:10.893 typically the sort of things like the TCP protocol are 0:31:14.411 provided by, say, the operating system vendor. 0:31:17.398 So Microsoft Windows provides an implementation of TCP; 0:31:20.982 similarly Mac OS and Linux also provide this. 0:31:23.902 At the network layer now we have sort of this huge variety 0:31:27.685 of people who are building these network switches. 0:31:32 And these network switches are the things that have sort of 0:31:35.212 coded into them the rules for how you should forward messages 0:31:38.534 around on the Internet. And so these are these 0:31:41.026 companies like Cisco and Alcatel, in sort of these big 0:31:43.962 companies that hear mentioned in the news all the time. 0:31:46.952 And then finally at the sort of lowest layer, 0:31:49.389 there are these sorts of link layer things. 0:31:51.715 And again, there are a number of link layer technologies like 0:31:55.037 WiFi and Ethernet. And those link layer 0:31:57.142 technologies are different than the technologies that are used 0:32:00.52 to actually decide which hop we should next use in order to 0:32:03.732 transmit data around in the Internet. 0:32:07 OK, so if there's a big diversity of applications. 0:32:10.665 And part of this diversity of applications is enabled by this 0:32:15.154 separation of the layers because Microsoft Windows doesn't really 0:32:19.942 have to know anything about how the network layer works. 0:32:24.057 It simply passes sort of messages on for the network to 0:32:28.097 transmit. OK, so and the point is that 0:32:30.865 the vendors are generally different at each of these 0:32:34.68 different layers. OK, so given this sort of 0:32:38.612 high-level overview of layering, what we're now going to do 0:32:42.354 throughout the next few lectures is to pay some attention to how 0:32:46.419 each of the different layers works. 0:32:48.612 So we're going to start off talking about this class about 0:32:52.29 the link layer. And then we'll move on to the 0:32:55.129 network and end to end layers in later lectures. 0:32:59 0:33:06 OK, so we've sort of seen some of the things that touched on 0:33:10.269 some of the things the link layer needs to provide at a high 0:33:14.539 level. But let's make a list and talk 0:33:17.144 about what these things are. So the first thing clearly that 0:33:21.414 the link layer does is manage the sort of transmission of bits 0:33:25.828 along this physical wire. OK, so there's going to be some 0:33:29.881 digital to analog to digital conversion that happens in sort 0:33:34.151 of as a part of using any one of these links and, 0:33:37.625 OK? And this is going to be one of 0:33:40.645 the main functions of the link layer is it's going to decide 0:33:43.881 how this sort of digital to analog to digital conversion is 0:33:47.063 done, OK? Another thing to link layer 0:33:49.037 does is framing. And so, framing, 0:33:50.793 well you remember we talked about at the link layer, 0:33:53.59 we sometimes call the messages that are transmitted around, 0:33:56.772 we call these things frames. Framing is simply separating 0:33:59.843 the frames, is deciding how we should separate the frames that 0:34:03.189 are on the wire. So it says, how does the 0:34:06.724 software that's running at the link layer decide that one frame 0:34:11.178 has ended and another frame has begun? 0:34:13.836 That's what framing is about. And we'll talk about these two 0:34:18.074 issues briefly today. The other kinds of things that 0:34:21.738 happen at the link layer we're not going to talk about as much 0:34:26.12 about today. One of them is channel access. 0:34:30 And so this is how somebody who wants to send a message actually 0:34:34.491 is able to physically use the wire or the air that it's 0:34:38.341 transmitting out of without interfering or stepping on top 0:34:42.475 of somebody else who's transmitting at the same time. 0:34:46.254 So you guys have read the Ethernet paper, 0:34:49.105 and you saw one way in which that's done and Ethernet, 0:34:52.884 which is basically by listening for a carrier, 0:34:56.092 right, which is called carrier sense, and only when the channel 0:35:00.512 is not used, there's not a carrier on the wire, 0:35:03.791 does somebody try and send. And then you use this notion of 0:35:08.851 collision detection and Ethernet in order to actually sort of 0:35:12.436 detect whether or not you are able to successfully transmit 0:35:15.901 your message. So these are the kinds of 0:35:18.171 things you worry about in channel access. 0:35:20.561 Last time with the phone network, we talked about time 0:35:23.727 division multiplexing as another way in which you can sort of 0:35:27.311 share access to a physical wire. You can carve it up into a 0:35:32.251 bunch of little units of time and assign each sender one unit 0:35:37.448 of time. OK, so now the last link layer 0:35:40.74 issue which sometimes is done in the link layer is error 0:35:45.503 detection and correction. And I don't want to talk at all 0:35:50.354 about really how error detection or correction works except to 0:35:55.637 say that some link layer is include it in some link layers 0:36:00.401 don't include it. The idea here is suppose we are 0:36:04.599 transmitting a message out over a wire. 0:36:06.559 Of course, there's some probability that that message 0:36:09.241 will become corrupted or garbled as it is being transmitted, 0:36:12.129 either because it interferes with somebody else who is 0:36:14.863 transmitting at the same time, or as the message propagates, 0:36:17.906 it decays somewhat and we can't decode it anymore. 0:36:20.434 So sometimes link layers include a facility for doing 0:36:23.116 this error detection and correction. 0:36:24.921 And error detection and correction is one of these 0:36:27.449 things that can sometimes be included at the link layer, 0:36:30.286 and is very often included at the end to end layer. 0:36:34 And so, as you read, the reason I mention this as 0:36:36.929 being a part of the link layer is that as you read the end to end 0:36:40.652 arguments paper for recitation next time, you should sort of 0:36:44.253 think about how the end-to-end argument relates to whether 0:36:47.732 error detection and correction should be within the link layer 0:36:51.455 or should be within the end to end layer. 0:36:53.896 OK, so let's talk about these sort of two issues that I said 0:36:57.497 we'll address briefly here. So the first issue I want to 0:37:02.774 talk about is how the conversion from digital to analog to 0:37:08.423 conversion works. So -- 0:37:11 0:37:17 So the way to think about, suppose that we have some 0:37:20.028 sender which has some sequence of bits, say, 0:37:22.581 one, zero, one, zero that they want to send out 0:37:25.312 over the radio channel. So if you think about what this 0:37:28.518 is going to look like in terms of an analog signal, 0:37:31.487 they are of course are many different ways you could 0:37:34.515 represent it. But a simple way to represent 0:37:38.175 it might be to say that this is a digital line, 0:37:41.625 and we'll make the line high when we are transmitting a one, 0:37:46.05 and we'll make the line low when we're transmitting a zero, 0:37:50.4 OK? So we would send this message 0:37:52.8 out as high followed by low followed by high followed by 0:37:56.925 low. So this would be one, 0:37:58.8 zero, one, zero. OK, you might ask the question, 0:38:02.324 OK, how do we decide when to sort of push the next bit out on 0:38:06.824 to the wire? Well, typically the way we do 0:38:10.671 is we have some clock signal that tells us when the next bit 0:38:14.072 should be, we should start sending the next bit on the 0:38:17.126 wire. So we might have some protocol 0:38:19.144 that says something like, every time there is a rising 0:38:22.198 edge in the clock signal, we'll start transmitting the 0:38:25.253 next bit. So, what does that mean? 0:38:27.155 So a rising edge -- 0:38:29 0:38:39 OK, so suppose this is our clock and this is our data. 0:38:42.037 And what I've just shown here is that every time the clock 0:38:45.189 signal goes high, we start sending in new bit. 0:38:47.768 OK, so every time we go from low to high in the clock signal, 0:38:51.207 we start putting a new bit on the wire. 0:38:53.385 So, now it's going to happen is oftentimes the way the network 0:38:56.881 connections are connected is what's a so-called serial 0:38:59.918 connection. So we have one wire that's 0:39:02.039 transmitting the data. And we don't have a separate 0:39:04.905 wire that includes the clock signal. 0:39:08 We just sort of transmit the data out over the wire. 0:39:11.814 So if you were to look at this data as it comes down the serial 0:39:16.45 connection, at the receiver, what it would look like is 0:39:20.489 something that was sort of, it's not going to look exactly 0:39:24.752 like these nice square pulse was that we have here. 0:39:28.491 It's going to have decayed somewhat. 0:39:31.108 So it might look something like each of these nice square 0:39:35.521 waves might have sort of become a somewhat decayed version of 0:39:40.008 their former selves, OK? 0:39:43 And now the question we have is, OK, how are we going to 0:39:46.81 decode this, right? So we don't have access to the 0:39:50.205 original clock signal that was used. 0:39:52.63 But we may know what frequency, would rate this data was 0:39:56.441 encoded at. So we may be able to generate a 0:39:59.351 comparable clock signal. So suppose we generate a clock 0:40:03.092 signal that's about the same frequency, and then try and use 0:40:07.18 that to decode the message, say, by looking again at the 0:40:10.99 rising edges. If we're not careful, 0:40:14.052 we're going to get something that's wrong. 0:40:16.107 So in this case, I transmitted one, 0:40:17.811 zero, one, zero, but now I'm decoding something 0:40:20.116 that's shifted from that somewhat. 0:40:21.77 So I'm decoding zero, one, zero, one. 0:40:23.575 So, I've sort of become offset because the two clock signals 0:40:26.532 aren't actually identical, right? 0:40:28.136 Even though they're at the same frequency, they are not 0:40:30.842 specifically lined up with each other in time. 0:40:34 So we say that those signals are out of phase. 0:40:37.208 So one signal is essentially a time shifted version of the 0:40:41.272 other signal. OK, and this time shifting 0:40:44.052 leads to these kinds of problems where we don't properly decode 0:40:48.472 the message because we're not sampling the channel at the 0:40:52.465 right point in time. We're not looking at the 0:40:55.602 channel to see whether it's high or low. 0:40:58.382 OK, so how are we going to fix this? 0:41:00.878 It turns out that there's sort of a simple and elegant way 0:41:04.941 that's often used to fix this. And it's what's called a phase 0:41:09.219 lock loop or a PLL. So a phase lock loop, 0:41:13.16 a very simple way that you can implement a phase lock loop is 0:41:16.477 as follows. So the idea here is that what 0:41:18.688 we want to do is we want to figure out, we want and make it 0:41:21.894 so that the receiver has essentially a lined up version 0:41:24.879 of the transmitters clock. So we need to figure out how 0:41:27.864 much we need to shift the receiver's clock in order to 0:41:30.793 make it line up with the transmitter's clock. 0:41:34 And once we do that, then hopefully we will properly 0:41:37.898 decode the message instead of being shifted when we decode the 0:41:42.56 message. So the idea with a phase lock 0:41:45.388 loop is kind of as follows. Suppose we have our signal like 0:41:49.821 this. A simple way to implement a 0:41:52.267 phase lock loop is to take this signal and to sample it not once 0:41:57.082 per clock period, but some multiple number of times per clock 0:42:01.133 period, OK? We call this oversampling. 0:42:03.961 We might, say for example, do eight times oversampling on 0:42:08.242 this. OK, so what that means is if we 0:42:12.125 were perfectly lined up in time on this oversampled signal, 0:42:16.101 and we were encoding a one as a high, and a zero as a low, 0:42:20.009 then what we would see if we were perfectly lined up is 0:42:23.711 alternating sequences of eight zeroes and eight ones, 0:42:27.276 right? So, if instead we start 0:42:29.264 decoding this and we see something that's not quite that, 0:42:33.104 suppose we see three ones followed by eight zeroes 0:42:36.463 followed by five ones. OK, so we decode two bits worth 0:42:40.726 of information off the wire, and we see that it's sort of 0:42:43.846 shifted in this way. OK, that suggests that I need 0:42:46.575 to shift the signal to the left some amount. 0:42:48.97 OK, and so this is exactly what the phase lock loop does is it 0:42:52.368 observed the signal as it's coming in, and it computes an 0:42:55.488 amount that we need to shift the signal in one direction or the 0:42:58.941 other. OK, so if you wanted to make a 0:43:02.269 sort of schematic for how a phase lock loop works, 0:43:05.976 we have our sender and our receiver. 0:43:08.624 The idea is simply as follows. You have some data at the 0:43:12.784 sender, and a clock at the sender. 0:43:15.281 Those go into some encoder box. They are transmitted out over a 0:43:19.971 line to the receiver, which has a decoder box. 0:43:23.375 OK, but the decoder box needs a clock signal that's been lined 0:43:27.99 up with the sender's clock signal, OK? 0:43:32 So we're going to do is use our phase lock loop to do that. 0:43:36.461 So the phase lock loop is going to take in the incoming signal 0:43:41.153 as well as the unaligned clock from the local machine. 0:43:45.23 And it's going to input into the decoder an aligned clock 0:43:49.538 signal. OK, so basically what's going 0:43:52.307 to happen is this PLL is going to be able to reconstruct the 0:43:56.846 clock signal from the sender. Of course, it takes a little 0:44:01.522 bit of time for the PLL to reconstruct the clock signal. 0:44:04.512 So usually what we do is every message that we transmit over 0:44:07.72 the link, we have some sort of set of synchronization bits at 0:44:10.982 the beginning of it that we use in order to allow the phase lock 0:44:14.407 loop to lock in to the phase of the signals being transmitted. 0:44:17.724 And that preamble doesn't carry any data bytes. 0:44:20.225 It's simply sort of overhead that's on every packet to 0:44:23.106 guarantee that the sender and receiver's clock synchronize 0:44:26.205 with each other. So there is one last little 0:44:29.518 detail associated with phase lock loops that you guys may 0:44:32.667 have noticed. So the issue is that the signal 0:44:35.142 that I've shown being transmitted here is a one, 0:44:37.785 zero, one, zero, one, zero signal, 0:44:39.641 right? But if you think about the way 0:44:41.666 that this phase lock loop works, what the phase lock loop does 0:44:45.096 is it looks for these transition points in the signal, 0:44:48.077 right? So it looks for points where 0:44:49.989 the signal changes from a one to a zero, right? 0:44:52.576 So suppose that instead of transmitting one, 0:44:54.994 zero, I transmitted zero, zero, zero, zero, 0:44:57.356 zero, zero, zero, right? 0:45:00 Then the problem I'm going to have is I'm just going to have a 0:45:02.911 very long sequence of zeroes even if I do this eight times 0:45:05.631 over sampling. I'm still just going to have a 0:45:07.731 long sequence of zeroes, and I'm not going to know how 0:45:10.261 much I need to shift the signal, right? 0:45:12.074 I'm going to have no way of computing how much I need to 0:45:14.699 shift the signal. So it turns out there's a very 0:45:16.943 simple and kind of elegant solution to this called 0:45:19.281 Manchester encoding. 0:45:21 0:45:27 And the idea with Manchester encoding is as follows. 0:45:30.352 It says will transmit a zero as a transition from a low to high, 0:45:34.492 and we'll transmit a one as a transition from a high to a low. 0:45:38.502 OK, so now if I transmit a signal like one, 0:45:41.262 zero, one, if I transmit the signal: zero, 0:45:43.957 zero, zero, it simply looks like this. 0:45:46.389 So, zero is a transition from a low to high followed by another 0:45:50.464 transition from a low to a high. OK, so now this is zero, 0:45:54.145 zero, zero, zero, right? 0:45:55.657 And if I transmit the signal, zero, one, zero, 0:45:58.615 this is a transition from a low to a high followed by a 0:46:02.164 transition from a high to a low followed by a transition from a 0:46:06.239 low to a high, OK? 0:46:09 So now we have a way that we can guarantee that every bit as 0:46:12.277 it least one transition in it. And that's going to allow our 0:46:15.555 phase lock loop to look into the face of the signal that's being 0:46:19.055 transmitted. Of course, the cost of this is 0:46:21.388 that we've now doubled the number, sort of, 0:46:23.722 we've doubled, every bit as a transition in 0:46:26.055 it, right? So we sort of have the number 0:46:28.222 of bits that we can send over the channel because instead of a 0:46:31.611 one being simply a low, or a one simply being a high 0:46:34.444 and a zero being a low, now everything is both a high 0:46:37.333 and a low. So we have the amount of data 0:46:40.563 that we can send on the channel. But we've sort of gotten this 0:46:43.852 wind that now we can have the phase lock loop actually work. 0:46:47.034 And so turns out Manchester encoding is actually commonly 0:46:50.053 used. There are other encoding 0:46:51.617 schemes that you can use that are sort of less wasteful of 0:46:54.691 channel bandwidth, but operate on the same 0:46:56.901 principle, trying to introduce extra transitions when possible. 0:47:01 So this basically wraps up our discussion of the link layer. 0:47:04.172 What we're going to talk about next time, are going to start 0:47:07.345 talking about how the network layer works. 0:47:09.55 And we're going to basically talk about how Internet routing 0:47:12.722 actually functions, and how these routers actually 0:47:15.357 decide which link they should use to transmit the next message 0:47:18.637 around in the network. So, that's it for this time, 0:47:21.326 and we'll see you on Wednesday.