0:00:16 So just a brief announcement, on this Friday you guys have a 0:00:20.425 writing tutorial. It's at 2 p.m. 0:00:22.75 there's only one tutorial this week at 2 p.m., 0:00:26.125 and it's going to be in this room. 0:00:28.6 So make sure you come. We're going to talk about 0:00:32.125 issues involved with preparing and presenting your DP2. 0:00:37 And it's really important that you come to pay attention. 0:00:41.985 So today we're going to continue our discussion of 0:00:46.347 networking and network layering. If you remember last time, 0:00:51.51 we talked about the three layers that are in any typical 0:00:56.406 network stack. And these three layers we said 0:01:00.323 were the end-to-end layer, the network layer, 0:01:04.24 and the link layer. And we went through the example of 0:01:09.253 how these three layers interact with each other as, 0:01:13.139 say, a message is sent through a network. 0:01:16.248 So, on a typical sender node, we said there are these three 0:01:20.756 layers -- 0:01:22 0:01:29 And there might also be a receiver node. 0:01:31.79 And then there could be several. 0:01:34.008 Each time a message is sent through the network, 0:01:37.371 it might pass through any number of intermediate gateway 0:01:41.306 nodes, or intermediate switches. So when a packet gets sent in, 0:01:45.742 it gets sent through the end-to-end layer in through the 0:01:49.678 network layer, down into the link layer. 0:01:52.468 The link layer chooses the next link to send the packet out 0:01:56.618 over, sends it's one of these switches. 0:01:59.337 The switch looks at the packet, sends it up to its own network 0:02:03.701 layer, which is in charge of determining the next link that 0:02:07.851 the message will take. On the next hop, 0:02:11.974 the message goes up to the link layer. 0:02:14.228 The link layer determines yet another link for the message to 0:02:17.883 take. The network layer determines 0:02:19.893 yet another message, link, for the message to take. 0:02:22.939 And then finally, the message reaches the 0:02:25.375 receiver or the message propagates up through the link 0:02:28.604 layer into the network layer, and then finally to the 0:02:31.771 end-to-end layer, and out to the user. 0:02:35 So we talked a little bit last time about various things that 0:02:38.916 happen in this architecture. We said that there's this 0:02:42.375 process of encapsulation that happens at each step along the 0:02:46.227 way. So the end-to-end layer may 0:02:48.25 attach headers on to the packet, a header or trailer onto the 0:02:52.167 packet; the network layer may attach a header and trailer, 0:02:55.887 and the link layer may attach a header and trailer. 0:03:00 But at no point does any layer look at the data that a higher 0:03:03.521 layer sent. And you also notice that in 0:03:05.752 this architecture, what we've shown here is that 0:03:08.51 only the link layer and the network layer of the switches 0:03:11.797 that are forwarding packets are actually processing the packets. 0:03:15.495 So the end-to-end layer, by definition, 0:03:17.726 is not involved in the forwarding of the packet. 0:03:20.484 The end-to-end layer is only involved when one of the 0:03:23.536 endpoints of the communication is involved. 0:03:27 So what we're going to talk about today, we're going to 0:03:30.606 finish very briefly our discussion of the link layer. 0:03:34.08 And then we're going to turn and focused mostly on the 0:03:37.62 networking layer. So do you remember last time? 0:03:40.692 We got as far as saying that the link layer is in charge of a 0:03:44.7 number of sort of important issues with the transmission of 0:03:48.574 data across one link of the network. 0:03:50.912 And we talked for awhile at the end of class last time about 0:03:54.853 this analog, to digital, sorry, got that backwards 0:03:58.125 digital to analog to digital conversion. 0:04:02 We are going to talk about the other issue, though, 0:04:04.617 that I said we needed to talk about in the context of the 0:04:07.549 network layer or the link layer is the issue of framing. 0:04:10.429 So the idea with framing is when you're sending a message 0:04:13.361 out over a network link, the receiver on the other end 0:04:16.136 needs to have some way of knowing that a packet is 0:04:18.701 starting or a packet is ending, right? 0:04:20.638 So as we call these packets when they are at the link layer, 0:04:23.727 frames. So, the issue with framing is 0:04:25.612 to identify the sort of beginning and end of every one 0:04:28.387 of these frames as it transmits over the network. 0:04:32 And there is a sort of fairly obvious way to do this is, 0:04:35.691 well, attach some special symbol, put some special symbol 0:04:39.449 at the beginning and end of every packet. 0:04:42.134 So, for example, if we were looking at Ethernet, 0:04:45.288 the payload of an Ethernet packet might contain the 0:04:48.644 destination address, the source address, 0:04:51.261 the type. So we'll talk more about what 0:04:53.812 the type field means in a minute, the data, 0:04:56.63 and some checksum information that can be used to detect 0:05:00.322 errors. And the preamble is a special 0:05:03.715 code that is attached to the beginning of every one of these 0:05:07.205 messages. And this is used to make the 0:05:09.393 Manchester encoding, which you remember we talked 0:05:12.232 about last class. This is used, 0:05:14.007 sorry, to allow the phase lock loop to lock into the message, 0:05:17.556 which we talked about at last class. 0:05:19.626 And it might be, in the case of Ethernet, 0:05:21.992 it's a well-defined sequence: one, zero, one, 0:05:24.595 zero, one, zero, one, zero. 0:05:26.133 But remember that with Manchester encoding that the 0:05:29.09 data that's actually transmitted looks a little bit different. 0:05:34 And then following the preamble, there is this start of 0:05:37.345 frame symbol. And then at the end of the 0:05:39.761 message, there's this end symbol. 0:05:41.743 So, one thing we might be concerned about is, 0:05:44.469 say, for example what if the network layer tries to send a 0:05:48 message that contains the end symbol in it? 0:05:50.601 that would be a problem, right, because then the end 0:05:53.761 layer would have inadvertently terminated the message, 0:05:57.044 even though this wasn't really the end of the message, 0:06:00.327 this is just something that happened to be the same code as 0:06:03.92 whatever the link layer had chosen for its end of code 0:06:07.203 symbol. And the reason that this is a 0:06:10.798 concern is, remember, we don't want the network layer 0:06:14.021 to have to understand lots of details about how the link layer 0:06:17.803 operates underneath it, right? 0:06:19.601 The network layer shouldn't have to make any assumptions 0:06:23.012 about what our valid symbols for it to transmit, 0:06:25.925 and what are invalid symbols for it to transmit. 0:06:28.839 So the way that we're going to solve this is through one of two 0:06:32.684 techniques. One: so the first technique 0:06:35.04 that we'll talk about for a moment is this idea of bit 0:06:38.264 stuffing. I'll get to that in a minute. 0:06:41.611 Another simple thing that we could do would be to simply use 0:06:44.888 a code that can't possibly be generated by, 0:06:47.222 say for example, the

Manchester encoding scheme. 0:06:49.833 So if the network layer sends a message like one, 0:06:52.5 one, one, one, the link layer is going to 0:06:54.722 convert that into some sequence of ones and zeros once it 0:06:57.833 applies Manchester coding. So if the link layer on the 0:07:01.623 receiver sees a message like one, one, one, 0:07:04.149 one, one, that can't possibly be a valid code. 0:07:06.855 That can't possibly be a valid message that could have been 0:07:10.343 generated by the network layer. Only the link layer can 0:07:13.591 actually send that sequence of bits out. 0:07:15.936 So we could tell that would be a terminating symbol. 0:07:19.003 Another simple way, though, to send one of these 0:07:21.829 and codes is using this technique called bit stuffing. 0:07:25.017 And the idea is pretty simple, and it's kind of a neat 0:07:28.204 technique that can be used in general when you have to do this 0:07:31.872 kind of encoding at a lower layer. So the idea is, 0:07:36.043 suppose we set our end code was equal to some bit string like 0:07:40.215 one, one, one, one. 0:07:41.467 And, the solution in bit stuffing is very simple. 0:07:44.805 What it says is that when you receive a sequence of bytes or 0:07:48.908 sequence of bits from the network layer at the link layer, 0:07:52.872 it says that you should convert any sequence of bytes that look 0:07:57.184 like one, one, one, three ones in a row, 0:07:59.896 into three ones in a row followed by a zero. 0:08:04 So this happens at the sender. And then the receiver just 0:08:07.585 reverses this, says any sequence of one, 0:08:10.082 one, one, zero, gets converted to one, 0:08:12.451 one, one, one. OK, so you could see that this 0:08:15.269 means, and this transform is only applied to the payload of 0:08:18.983 data packets that it's coming from the network layer down into 0:08:22.888 the link layer. OK, so you can see that there 0:08:25.706 is no way for the link layer, now, to send a sequence, 0:08:29.099 or for the network layer to actually cause four one's in a 0:08:32.749 row to be sent out over the wire because of this transformation. 0:08:38 So if the network layer tries to send a message that contains 0:08:42.741 a sequence with four ones, what the link layer is going to 0:08:47.245 send is three ones followed by a zero followed by a one. 0:08:51.591 Notice, however, that this also means that if 0:08:55.068 the network layer tries to send a sequence of three ones 0:08:59.414 followed by a zero, the link layer will transmit 0:09:03.128 three ones followed by two zeros, OK? 0:09:07 And then at the receiving end, we can just trivially apply 0:09:10.744 this reverse transformation. So this idea of bit stuffing 0:09:14.423 allows us to guarantee that any time that the link layer on the 0:09:18.496 receiving side actually sees four ones in a row, 0:09:21.583 this is really the end symbol as opposed to the network layer 0:09:25.525 trying to transmit four ones in a row, OK? 0:09:28.218 So that's all I want to say about the network layer or about 0:09:32.094 the link layer. And what I want to do now is to 0:09:35.61 move on to a discussion of the network layer. 0:09:38 0:09:52 So the network layer -- 0:09:54 0:10:00 -- has two primary functions that we're going to talk about 0:10:05.031 today. The first is forwarding. 0:10:07.72 And the second is routing. So the idea with forwarding is 0:10:12.491 as follows. So the idea with forwarding is 0:10:15.961 basically to allow nodes to decide, given a particular 0:10:20.559 packet to allow them to decide what the next hop for that 0:10:25.416 packet should be by looking at the destination address that's 0:10:30.621 in the message that they are trying to transmit. 0:10:36 So they're going to look at the destination address and make 0:10:39.314 some decision about where to send this packet next. 0:10:42.123 And they're going to do that by keeping a table that basically 0:10:45.55 maps every address into the next link to send to. 0:10:48.247 So let's look at a really simple example. 0:10:51 0:10:58 Suppose we have five machines, A, B, C, D, and E -- 0:11:02 0:11:06 -- connected as follows. And let's number these links. 0:11:10.818 Let's number this link from E to B. 0:11:13.909 We're going to call it L1. We're going to letter this link 0:11:19.09 E to D L2. And then, I'm going to number 0:11:22.636 all the links sort of in the same way. 0:11:26 So, B's link to A. I'll number L1. 0:11:30 B's link to D I'll number L2, and B's link to E I'll number 0:11:34.328 L3. So, notice that this link, 0:11:36.492 I've given two names to this link depending on whether we're 0:11:40.895 talking about it from a perspective of B or E. 0:11:44.253 OK, so now we can do the same numbering for all of the other 0:11:48.656 links. So, call this L1. 0:11:50.373 Call this L2. Call this L3. 0:11:52.313 And, this one is L1, L2 from C's perspective. 0:11:55.597 And this is L1 and L2 from A's perspective. 0:12:00 So now we have this labeled graph here. 0:12:02.615 And now let's look and see what the forwarding table for one of 0:12:06.882 these nodes might look like. So, for example, 0:12:09.91 if we look at the forwarding table for node A, 0:12:13.008 what we'll see is one entry for each of the other nodes that are 0:12:17.344 in the network. And this entry will tell us, 0:12:20.303 given a message destined for this, whatever address is in, 0:12:24.226 this is the destination address. 0:12:26.36 So given a particular destination address, 0:12:29.182 it will tell us what the next link we should use is. 0:12:34 OK so, and this is the forwarding table for a 0:12:37.296 particular node. So in this case, 0:12:39.694 we're looking at the forwarding table for node A. 0:12:43.291 So, if A sees a node destined to node A, what should it do 0:12:47.561 with it? Well, it's destined to its 0:12:50.109 local mode. So it's going to do the obvious 0:12:53.256 thing, which is send it up to the end to end layer. 0:12:57.002 So I'll just write E to E. So if it receives a node 0:13:01.562 destined for B, it's going to, 0:13:03.449 presumably it wants to send it along whatever the shortest path 0:13:07.485 is. And we'll talk about how the 0:13:09.502 next step, routing is actually deciding which thing should be 0:13:13.408 there. But, for example, 0:13:14.905 it might have L2 as the next link. 0:13:17.053 OK, and if it wants to send to C, it might have L1 as the next 0:13:21.023 link. So, A to B is using, 0:13:22.65 it's using L2, and A to C it's using L1. 0:13:25.189 OK, now, node D is, say if it wants to route to D, 0:13:28.378 it's going to have to route through either B or C. 0:13:33 And so, for now, let's just say it routes to L1. 0:13:36.366 And so, it routes to C, and then it's going to allow C 0:13:40.161 to go ahead and forward the packet onto D. 0:13:43.098 If it wants to route to E, well, again, 0:13:45.819 it can either route through B or through C. 0:13:48.827 But let's say maybe it wants to route through B because it has a 0:13:53.339 shorter path. So, we'll write L2 here. 0:13:55.989 OK, so this is just a very simple example. 0:14:00 Now you can see that any time that A wants to send a packet, 0:14:03.554 it has a next hop that it should use for every destination 0:14:06.987 address within the network. So, this is very simple. 0:14:10.06 And this is sort of at a high level the way that forwarding 0:14:13.373 works. Of course, there are some other 0:14:15.602 details associated with getting forwarding to work properly. 0:14:19.156 In order to talk about sort of how forwarding actually works, 0:14:22.771 the interaction between these layers, what I want to do is 0:14:26.204

just give you some examples of what the packet headers look 0:14:29.698 like for different kinds of protocols that are used in the 0:14:33.132 real world. So we're talking about here, 0:14:37 this is the IPV4 header. So, remember, 0:14:39.642 IP is a protocol that runs at the network layer. 0:14:43 And this header has the following information in it. 0:14:46.642 So, what I've shown here down the left side is a sequence of 0:14:50.857 words. So these are 32-bit words. 0:14:53.142 And I'd just broken up the bits by number along the top. 0:14:57.071 So, the first sort of word has information about the version of 0:15:01.5 the protocol that's running in it. 0:15:05 So in this case the version would be four because this is 0:15:08.03 IPV4. There's a new IP protocol that 0:15:09.924 is in the process of being deployed called IPV6, 0:15:12.467 which you'll sometimes see referenced. 0:15:14.469 There's a header length field. It just specifies the length of 0:15:17.77 the header. There's a TOS field. 0:15:19.448 This is type of service. It's typically not used in most 0:15:22.424 packets. And then there's a packet 0:15:24.318 length which is the entire length of the whole packet, 0:15:27.186 not just the length of the header. 0:15:30 So this should be fairly straightforward except for the 0:15:34.281 type of service field which you guys don't need to pay any 0:15:38.801 attention to. So the next field, 0:15:41.259 so there's this identification. The next word has 0:15:45.303 identification, flags, and fragment offset. 0:15:48.634 Let's just look at the next one. 0:15:51.092 And the next one has time delay of end to end protocol and 0:15:55.612 checksum. So the important fields, 0:15:58.229 I'm blanking on what the identification field contains, 0:16:02.511 which is why I'm stalling. So the identification field, 0:16:08.077 so let's just talk about the other fields and we'll come back 0:16:12.528 to identification if I remember it. 0:16:15.05 So the flag's field simply contains some information about 0:16:19.278 whether this packet has been fragmented or should be 0:16:23.062 fragmented. So fragmentation is something 0:16:26.029 that this packet can be split into multiple sub packets. 0:16:31 You don't need to worry about it too much. 0:16:33.214 And the fragment offset as if this packet has been split into 0:16:36.455 these sub packets, these fragments. 0:16:38.291 Then this is the number of the fragment that's being 0:16:41.046 transmitted so, the interesting ones now come 0:16:43.422 in this next row. So, we have the time to live 0:16:45.853 flag, the end to end protocol, and then the checksum. 0:16:48.662 So, the time to live field is sometimes abbreviated TTL. 0:16:51.632 And time to live is a little bit of a strange name for it. 0:16:54.711 Basically what this says is that as this packet is being 0:16:57.682 forwarded through the network, we're going to decrement the 0:17:00.814 time to live by one on every step that it's forwarded through 0:17:04.055 the network. And if the time to live reaches 0:17:08.072 zero, we're going to stop forwarding this. 0:17:10.813 So the reason that we care about time to live is that there 0:17:14.69 can sometimes be loops and are forwarding routes. 0:17:17.699 So suppose that we had set this up so that A sends messages 0:17:21.576 destined for, say, A sent messages destined 0:17:24.384 for E through C. But C sent messages destined 0:17:27.325 through E through A, right? 0:17:30 So that would be a loop, and that would be a problem. 0:17:33.148 So, we're going to try and avoid forming these loops when 0:17:36.539 we do our routing protocol. But they're going to be certain 0:17:40.05 situations when those fail, for example, 0:17:42.412 that loops can occasionally occur. 0:17:44.41 And we're going to use the time to live field to eliminate 0:17:47.861 those. The next thing is the end to 0:17:49.919 end protocol. So the end to end protocol is 0:17:52.462 the specification of which end to end protocol we should 0:17:55.792 forward this message onto. So I'll talk about that more in 0:17:59.244 a second. And then there's this checksum 0:18:02.75 field, which can be used to determine whether the message 0:18:06.019 was fully formed, although it turns out that in 0:18:08.704 most cases in IP, this field probably isn't used. 0:18:11.505 So, and then there is the source address and a destination 0:18:14.832 address. So these are IP addresses that 0:18:17.05 identify the endpoints of the packet and then, 0:18:19.677 finally, there is some additional, optional information 0:18:22.828 which can specify a huge range of different things. 0:18:25.747 It can be up to 44 bytes. In this case, 0:18:27.964 you don't need to worry about it. 0:18:31 And then finally there's the payload, which is the actual 0:18:35.748 message that was sent from the end to end layer into the IP 0:18:40.666 layer. So what you see happening here, 0:18:43.804 if we look at a diagram of what the interface between the end to 0:18:49.146 end layer, what the interface between the network layer and 0:18:54.064 link layer is. So let's just look at our three 0:18:57.88 layers again. We have our end to end layer. 0:19:01.757 We have our network layer. And we have our link layer. 0:19:04.969 And one of the things that we can pick out here is that notice 0:19:08.666 that we have this end to end protocol that's actually in our 0:19:12.242 IP packet so what this means is that if I have a network 0:19:15.575 protocol like IP that's running here, when it receives a packet 0:19:19.333 destined for its local address, say, destined to itself where 0:19:22.969 it's supposed to forward it up to the end to end layer, 0:19:26.242 it can actually dispatch this packet to a number of different 0:19:29.878 end to end layers. So, for example, 0:19:33.033 we talked last time about the TCP protocol. 0:19:35.747 But there are a number of other end to end layers that can be 0:19:39.625 used in the Internet. So we'll talk in this class a 0:19:42.856 little bit about a different protocol called UDP. 0:19:45.957 We'll talk about this more. We'll talk about the end to end 0:19:49.705 layer more next time. But the point is that this up 0:19:52.936 call to the layer above us is controlled by the contents of 0:19:56.684 this header that we have. So similarly, 0:19:59.14 if we look an Ethernet header, what the Ethernet header has on 0:20:03.082 it is the destination address, which is 48 bits, 0:20:06.119 and the source address which are 48 bits, followed by a link 0:20:09.932 protocol. So the link protocol says, 0:20:14.034 and so the Ethernet, remember, is a link layer 0:20:17.556 protocol. And what happens is, 0:20:19.826 so if we have Ethernet here, it has a protocol ID that 0:20:23.973 specifies the protocol that it's supposed to send messages back 0:20:28.826 up to. OK, so these fields, 0:20:31.341 one question you might ask is, well, how does the Ethernet 0:20:35.165 layer know which link protocol it should send messages to? 0:20:38.989 Or how does the network protocol know which ends in 0:20:42.209 protocol to send its messages to? 0:20:44.355 So it's kind of interesting and it's worth looking at the 0:20:48.112 pseudocode for one of these protocols just quickly in order 0:20:52.003 to understand this a little bit better. 0:20:54.552 So let's look at the pseudocode for the function which we're 0:20:58.51 going to call Net Handle that accepts messages either from the 0:21:02.602 end to end layer to be sent out across the network, 0:21:05.956 or from the link layer when a message is sort of received as 0:21:09.914 it's forwarded through. So we can use this procedure 0:21:14.423 both for down calls and up calls, and it accepts a packet 0:21:17.271 which, say for example, has this format that I've shown 0:21:20.016 here, which is the same as the format that I've shown on the 0:21:23.016 other page. So the first thing this is 0:21:24.898 going to do is it's going to check and see if the destination 0:21:27.949 of the packet, for example, 0:21:29.271 is the local address. And if the destination of the 0:21:32.45 packet is the local address, then we know what we should do 0:21:35.349 with it, right? We're going to pass it up to 0:21:37.5 the end and layer. And we're going to send, 0:21:39.599 so what we're going to do is we're going to de-encapsulate 0:21:42.45 this packet, right? We are going to strip the 0:21:44.65 header off of it. So we're just going to send the 0:21:47.049 payload up to the layer above, but we're going to need to 0:21:49.849 dispatch to the appropriate layer, which we're going to do 0:21:52.7 by sort of saying and naming what the end to end protocol is 0:21:55.65 that we would like to dispatch to. 0:21:57.299 And there may be some other options that we pass along with 0:22:00.2 this as well. So that's what the three dots 0:22:03.631 there mean. So now, if this isn't for the 0:22:05.881 local packet, for the local node, 0:22:07.681 then we need to send this packet out again, 0:22:10.043 right? So what we're going to do is 0:22:11.956 we're going to check the time to live on this packet. 0:22:14.881 And if the time to live is greater than zero, 0:22:17.356 then we are going to decrement the time to live. 0:22:20 If the time to live is less than or equal to zero, 0:22:22.756 then we know that we're not going to forward this packet 0:22:25.849 anymore. We're just going to drop it. 0:22:29 So the time to live gets set initially by the first guy who 0:22:32.974 sends the packet out. He initializes it to some 0:22:36.125 value, which is the maximum number of hops that he wants 0:22:39.894 this message to be propagated for. 0:22:42.155 And so, after we decrement the time to live, 0:22:45.101 we're going to look up in our forwarding table the next link 0:22:49.144 that we want to use. OK, and what I've shown here is 0:22:52.638 that we're going to look up the sort of protocol to use and the 0:22:56.887 name of the next link. So, for example, 0:22:59.49 this IP layer might have Ethernet underneath it, 0:23:02.711 which might be one protocol. But there might also be a WiFi 0:23:08.068 layer that's here that we could also send messages out. 0:23:11.793 So we could send out through either one of these protocols. 0:23:15.793 So we look up the one we want to send out, and then we call 0:23:19.793 this link send routine, which actually does the 0:23:22.965 transmission out over the appropriate named link. 0:23:26.275 Notice now what I've specified, I have this NETPROT in all 0:23:30.413 caps here. So in this case, 0:23:32.961 this should be, for example, 0:23:34.585 IP. It's the name of the network 0:23:36.448 protocol. So when the network protocol 0:23:38.673 sends the message to the link layer below it, 0:23:41.318 it specifies what the network protocol is that wants to have 0:23:44.865 returned, that it wants to be called on return. 0:23:47.631 So it says, please link layer; when you have received a 0:23:50.877 message that needs to be sent up to the network layer, 0:23:54.064 dispatch it to the network layer named NETPROT. 0:23:56.829 OK, and then for example the TTL we can't transmit the 0:24:00.016 message out. TTL has expired. 0:24:02.897 If we've sent it too many hops already, then we're going to 0:24:06.153 need to do some kind of error handling. 0:24:08.285 In this case, error handling might be, 0:24:10.362 send a message back to the source address of this message, 0:24:13.561 and tell them that the TTL ran out on this message. 0:24:16.367 That will be one thing you could do. 0:24:18.331 Another thing you might do is simply drop the packet or ignore 0:24:21.755 the error. OK, so this is just a simple 0:24:23.887 example of how forwarding might work. 0:24:25.908 And what I wanted to use it to illustrate is the fact that 0:24:29.107 there can be multiple different link layers at the bottom that 0:24:32.53 can be dispatched to. And these link layers can, 0:24:36.709 in fact, dispatch up to multiple different network 0:24:39.93 layers as well. So in today's Internet, 0:24:42.428 it's uncommon to see, you'll rarely interact with 0:24:45.583 anything that's not using IP at some layer or some network 0:24:49.33 layer. But there have, 0:24:50.711 in the past, been a variety of different 0:24:53.274 network layers that have been proposed. 0:24:55.772 You guys may have seen an old protocol from Novell called IPX, 0:24:59.782 or you may have used AppleTalk, which is an Apple protocol that 0:25:03.858 in some varieties also runs at the IP layer. 0:25:08 Finally, I just want to point out that the last little 0:25:11.34 layering detail that I want to get to is that if you think about the 0:25:15.186 link layer from the perspective of the network layer, 0:25:18.464 the link layer is simply, the link just looks like one 0:25:21.805 hop to one more connection. But in fact, 0:25:24.263 these links can, themselves, be networks. 0:25:26.785 So it's just kind of a weird statement. 0:25:30 But here's a simple example of what I mean. 0:25:32.287 Suppose that this link layer, that one of the link layers 0:25:35.338 that the IP layers can send a message to is a modem 0:25:38.062 connection, OK? But if you think about what a 0:25:40.459 modem connection is, right, it's a connection out 0:25:43.073 over a phone line. And a phone line is, 0:25:45.143 itself, right, as we talked about at the 0:25:47.268 beginning of class, a phone line is, 0:25:49.175 itself, a kind of the network. Right, it's this kind of 0:25:52.116 circuit-switched thing that goes through multiple ones of these 0:25:55.494 circuit switches as opposed to going through these. 0:25:58.217 So underneath the modem, the modem layer, 0:26:00.396 which looks like a link to the network layer, 0:26:02.793 there's actually, these link layers can actually 0:26:05.354 themselves be composed of a number of links, 0:26:07.696 and they can, themselves, be a type of a 0:26:09.821 network. So it's sort of worth realizing 0:26:13.896 that there is a kind of hierarchical relationship, 0:26:16.892 one, but the link layer may actually itself be a network 0:26:20.256 consisting of multiple links, but from the point of view of 0:26:23.804 the higher-level link layer here, it's just a single link, 0:26:27.29 a modem connection to some endpoint that we can send a 0:26:30.532 message over. OK, so what I want to do now is 0:26:34.542 take you through, we're going to switch gears a 0:26:38.314 little bit and talk about the sort of next piece that I said 0:26:43.153 we needed to address, which is this issue of routing. 0:26:47.417 So I said we're going to talk about forwarding, 0:26:51.19 and then we're going to talk about something else called 0:26:55.701 routing. So routing is the process by 0:26:58.653 which we build up our forwarding tables. 0:27:03 OK, so what I showed you here was I just sort of told you what 0:27:06.122 the values that should go in these forwarding tables should 0:27:09.092 be. But I didn't tell you how they 0:27:10.781 were derived or where they came from. 0:27:12.624 So in this case of the simple network, it was relatively easy 0:27:15.696 for us to, by hand, sort of come up with a set of 0:27:18.153 possible forwarding paths that seem reasonable. 0:27:20.508 But you can imagine that if this network had scaled up to a 0:27:23.477 million nodes, that's not something that we 0:27:25.627 want any individual to have to do. 0:27:27.317 No person should have to configure all these routing 0:27:29.928 tables or all these forwarding tables. 0:27:33 So instead, what we're going to do is we're going to use this 0:27:36.299 process called routing in order to build these forwarding tables 0:27:39.765 automatically. And we really want our routing 0:27:42.184 protocol to be three things. First, we wanted to be 0:27:44.934 scalable. And the obvious way in which we 0:27:47.134 want it to be scalable is the way that I just said. 0:27:49.884 We don't want to have to have people, we want this thing to 0:27:53.075 scale up to, say, a million nodes or several 0:27:55.44 million nodes and be able to continue to work. 0:27:57.914 We shouldn't have to have people sort of involved with 0:28:00.829 configuring, building up these forwarding tables at every step 0:28:04.184 of the way. We also want it to be robust. 0:28:07.949 So by that, I mean it should be tolerant of faults. 0:28:11.31 If a node fails in the network, the network should eventually 0:28:15.344 discover that that node has failed and be able to forward 0:28:19.109 packets around the failure or be able to compensate for the 0:28:23.008 failure if at all possible. Finally, we want this routing 0:28:26.773 protocol hopefully to be distributed. 0:28:30 So we don't want to have to have one machine that's 0:28:32.733 responsible for setting up the forwarding table on all the 0:28:35.848 other machines. We don't want to have one 0:28:38.035 machine that needs to contact all of the other nodes and 0:28:41.041 collect information about what all of their links are, 0:28:43.938 and assemble all of those links together into one global 0:28:46.945 forwarding scheme. And this really gets at the 0:28:49.405 scalability again. So what I want to do is to take 0:28:52.083 you through the process of routing. 0:28:53.942 Before I do that, I just want to take a quick 0:28:56.347 digression. We're going to start off with 0:28:58.533 very tiny networks on this, and just talking about a very 0:29:01.594 simple baby network. It just so that you guys don't 0:29:05.788 feel like this is completely unrealistic, I want to show you 0:29:09.307 that in fact the Internet at some level started out like 0:29:12.587 this, too. What we're going to do it in 0:29:14.853 the course of this class is sort of build up from these very 0:29:18.371 simple networks that maybe look the way the Internet did at 0:29:21.83 first, the schemes that are more like what is actually used in 0:29:25.467 today's production Internet. So, sorry I went back on you 0:29:28.807 guys. OK, so this is a picture of 0:29:32.019 what the Internet look like in 1969. 0:29:34.637 So you may not be able to read the labels on these things, 0:29:38.901 but we're looking at three nodes here. 0:29:41.669 This is UC Santa Barbara on the coast of California. 0:29:45.484 It's southern central California. 0:29:47.877 This is Stanford Research Institute, SRI, 0:29:50.87 which is the Bay Area. This is Utah, 0:29:53.488 and this is UCLA. So, this was the ARPANET, 0:29:56.629 which was a precursor to the Internet, and was sort of one of 0:30:01.118 the very first of these large wide-area networks that was ever 0:30:05.681 developed. And each of these little square 0:30:10.232 nodes here is a machine that's on this. 0:30:12.969 So there were four machines on the ARPANET in 1969. 0:30:16.57 And there were these four routers that were being used. 0:30:20.459 So, this is 1971. So by 1971, there still is a 0:30:23.7 cluster of these machines in California. 0:30:26.509 But you notice that Illinois, Carnegie Mellon, 0:30:29.75 and Boston have suddenly appeared on this map. 0:30:34 So, MIT is here, Lincoln Labs is here, 0:30:36.051 Harvard is here, BBN, which is another large 0:30:38.434 company that does a lot of networking research in this area 0:30:41.65 is here. So, the network has started to 0:30:43.756 evolve. And in particular, 0:30:45.142 you notice that there are now these sort of two clusters, 0:30:48.247 these two regions, one on the West Coast and one 0:30:50.852 on the East Coast that have a number of nodes. 0:30:53.347 So, it just keeps growing and growing. 0:30:55.398 So, by 1980, you see the network has gotten 0:30:57.727 substantially larger. And one of the interesting 0:31:01.436 things about this is you are starting to see a diversity of 0:31:04.64 links. So notice that Hawaii is now 0:31:06.519 connected into California by way of a satellite connection, 0:31:09.723 as is London. So, we now have not only just 0:31:12.044 these wires that are running, but in fact we have wireless 0:31:15.193 links. But at the same time this was 0:31:17.127 happening, all the protocols we've been talking about were 0:31:20.276 being developed. And one of the whole goals of 0:31:22.762 this was to be able to support these multiple different kinds 0:31:26.077 of links on top of the standard networking protocol. 0:31:30 OK, so by 1987, this thing has really become, 0:31:32.575 turned into a number of decentralized networks. 0:31:35.267 There's this large network called the ARPANET. 0:31:37.901 There is something smaller called the NSF backbone, 0:31:40.827 and a number of other networks that are military networks, 0:31:44.163 and so on. These are all connected 0:31:46.095 together, and they're all being, by 1987 they were all running 0:31:49.665 this TCP/IP protocol that was being used to exchange 0:31:52.65 information between all of these things. 0:31:54.933 So roundabout a few years after this, right, the sort of World 0:31:58.503 Wide Web suddenly happened, and there became this huge 0:32:01.605 commercial interest in the Internet. 0:32:05 And that has really just sparked this explosion of nodes, 0:32:08.458 and made the network just huge and incredibly vast. 0:32:11.546 So it's hard to see this, but in the middle of this you 0:32:14.881 notice that there is a little orange node here. 0:32:17.722 This bar on the left side is showing the out degree of the 0:32:21.242 nodes in the network. So this number is 2,977. 0:32:24.021 So that means there's a node at the center of the Internet that 0:32:27.85 has 2,977 links to other nodes in it. 0:32:31 OK, so this is a really incredibly large network. 0:32:33.861 And you notice that all of these 20 or 30 bright pink red 0:32:37.2 nodes here have hundreds to thousands of outgoing 0:32:40.061 connections on each one of them. So there is this core of the 0:32:43.638 Internet that is very highly connected to a very large part 0:32:47.096 of the network. And now, down or out around the 0:32:49.838 edges are these much smaller networks that have much lower 0:32:53.236 collectivity. And these are things like 0:32:55.501 service providers, for example, 0:32:57.29 that consumers might pay some money to get a connection from. 0:33:02 So this is the sort of core of the Internet. 0:33:04.335 These are the people who are not so much the end-users on the 0:33:07.593 Internet, but the service providers that are providing 0:33:10.471 connectivity for the end-users. Each one of these little nodes 0:33:13.784 represents one of those service providers. 0:33:16.011 This was as of 2003. OK, so what I want to do now is 0:33:18.78 to start as I said. We're going to sort of start 0:33:21.333 off with a baby version of a network. 0:33:23.288 In particular, we're going to look how 0:33:25.297 forwarding, or how routing might work in this very simple network 0:33:28.773 that

I've shown here. So let's see what happens. 0:33:32.923 So we're going to use a protocol that we call path 0:33:36.548 vector routing. And the idea behind path vector 0:33:39.951 routing is that we're going to build up the paths from, 0:33:43.946 that is, the sequence of nodes that we should forward messages 0:33:48.459 through in order to reach a particular destination. 0:33:52.158 And the way that this is going to work is we're going to have two 0:33:56.522 steps. So, let's put it over here. 0:34:00 So routing is going to consist of two steps: 0:34:03.503 an advertisement phase or an advertisement step, 0:34:07.331 and an integration step. And the idea is that during the 0:34:11.812 advertisement step, each node is going to advertise 0:34:15.885 what other nodes it knows how to reach. 0:34:18.981 And then during the integration step, each node is going to take 0:34:24.114 all of the advertisements that heard during the previous 0:34:28.594 advertisement step, and integrate them into a new 0:34:32.505 set of routes that identifies the new set of nodes that this 0:34:37.311 node can reach. OK, so this'll be very clear 0:34:41.506 when I show you the example. So let's just look at the case 0:34:44.625 of all the nodes, figuring out how they can reach 0:34:47.207 node E. So what's going to happen is 0:34:49.09 that first node E is going to send out an advertisement that 0:34:52.264 says that node E knows how to reach node E, 0:34:54.523 right, which it obviously does. And the way that it reaches 0:34:57.643 node E is simply by forwarding a message up to its end to end 0:35:00.87 layer. So it says, to reach node E, 0:35:03.593 come this way. And it sends it out over the 0:35:05.983 two links that it has to node C and D. 0:35:08.089 So when node C and D hear this advertisement, 0:35:10.593 during this integration step what they are going to do is to 0:35:13.951 add this information about this connection to node E, 0:35:16.91 and they're going to store which like it is that they send 0:35:20.154 this message out over. They should send messages out 0:35:23.056 over in order to reach node E. So node C is going to store 0:35:26.3 link one, and node D is going to store link two. 0:35:30 But in addition to that link, they're going to store the path 0:35:32.752 that they use. So now what's going to happen 0:35:34.77 is that each of C and D, during the next advertisement 0:35:37.201 step, C and D are going to also advertise the information that 0:35:40 they have about the nodes that they can reach in the network. 0:35:42.752 So, I'm only showing the advertisements for node E here. 0:35:45.275 But of course, at the same time, 0:35:46.697 C and D are also advertising the fact that they can reach 0:35:49.266 themselves, and maybe their ability to reach other nodes in 0:35:51.926 the network that we haven't shown. 0:35:53.44 So we're just looking at E, but bear in mind that all the 0:35:56.009 advertisements are being done for all the nodes at the same 0:35:58.669 time. So, C sends out an 0:36:01.277 advertisement that says I can reach node E, 0:36:04.258 and the way to do it is to send, and I can reach node E via 0:36:08.374 C and then E. And similarly, 0:36:10.29 D sends out a message that says I can reach node E by a D and 0:36:14.548 then E. So, OK, up to this point we 0:36:16.961 haven't really seen anything very interesting. 0:36:20.154 But now during the next integration step, 0:36:22.993 we see that these two nodes, B and A, now have heard an 0:36:26.825 advertisement that gives them a path that allows them to reach 0:36:31.154 node E. So now, every node has a path 0:36:34.87 that allows them to reach node E. 0:36:36.935 But this advertisement and integrations process is going to 0:36:40.677 keep going on. So, for example, 0:36:42.612 nodes A and B are going to advertise that they can, 0:36:45.838 also, reach node E to their neighbors. 0:36:48.225 And in this case, it's probably unlikely that any 0:36:51.322 of the nodes would like to switch to a new route. 0:36:54.419 So, for example, it's not clear, 0:36:56.419 there's no reason, for example, 0:36:58.354 that A would want to forward its message. 0:37:00.935 It's probably unlikely that A will want to forward its 0:37:04.354 messages through B to reach E, right, because that's going to 0:37:08.225 be a longer path than for A to send its messages simply through 0:37:12.225 C and then to E. But B doesn't actually know 0:37:16.599 whether or not A knows about E yet or not. 0:37:19.027 So it needs to continue to broadcast this information out. 0:37:22.403 So this is pretty simple. It's pretty clear how this 0:37:25.423 works. And once this process has been 0:37:27.555 running for a while, you can see that the network is 0:37:30.576 going to converge into a state where every node has, 0:37:33.596 as long as the network is connected, it will converge into 0:37:36.972 a state where every node has a path to node E, 0:37:39.637 right? And the amount of time that 0:37:42.67 it'll take for that to happen is equal to the maximum number of 0:37:46.507 hops that any node is away from node E. 0:37:48.858 So, once this node has converged, now we can trivially 0:37:52.138 build up our forwarding table simply by pulling out the link 0:37:55.789 number from each one of these nodes. 0:37:57.955 So, for example, we can see that E's forwarding 0:38:00.801 table simply says: to reach node E, 0:38:02.905 you send it to my end to end layer. 0:38:06 D's forwarding table would say, to reach node E, 0:38:08.692 you send it over my link, L2. 0:38:10.295 C's forwarding table would say, to reach node E, 0:38:12.988 you send it over L1. B's forwarding table would say, 0:38:15.909 to reach node E, you send it out over my L1, 0:38:18.372 and so on, OK? So, once we've done this path 0:38:20.835 vector routing, at the end of this process we 0:38:23.355 will know which links, we'll have built up the 0:38:25.933 forwarding table that we can use for sending our links. 0:38:30 OK, so this is a very simple process. 0:38:31.93 But now, let's look at what happens when something, 0:38:34.611 so what we said here is that each advertisement step, 0:38:37.4 and after each advertisement, we're going to go ahead and do 0:38:40.564 integration. Integration, 0:38:41.851 basically what we're going to do is we're going to try and 0:38:44.908 pick the best path in some way. What we've shown here is simply 0:38:48.126 picking the shortest path. 0:38:50.003 So we've picked the shortest possible path for every node to 0:38:53.167 reach node E. And in case it wasn't clear, 0:38:55.365 I didn't explicitly state this, it's important to realize that 0:38:58.637 nodes are going to ignore advertisements with their own 0:39:01.533 address in the vector. OK so if, for example, 0:39:05.359 when node E hears node D advertising that it can reach 0:39:08.635 node E, node E is going to say, oh, well, I am node E. 0:39:11.911 I don't actually need to pick up this path. 0:39:14.507 I don't need to send my packets to myself through node D. 0:39:17.969 That would be a silly thing to do. 0:39:20.008 That would create a routing loop, which is something that we 0:39:23.655 presumably don't want to do. So, this is a simple way using 0:39:27.24 these path vectors we can use to avoid creating routing loops. 0:39:32 OK, so now let's look at what happens, something a little bit 0:39:35.386 more interesting. Let's look and see what happens 0:39:38.095 when, for example, there's a failure. 0:39:40.126 So this is just exactly the same network that I showed you. 0:39:43.4 But, suppose for example that the link between D and E fails. 0:39:46.786 OK, so now D no longer has a route to node E. 0:39:49.269 But remember, the way that I described this 0:39:51.64 is that this advertisement process is just going to 0:39:54.462 continue going on in the background, right? 0:39:56.832 So what's going to happen is that at some point node D is 0:39:59.992 going to realize that its link to node E went down. 0:40:04 And, node D is going to cross this table out of its entry. 0:40:07.106 So, node D is going to basically stop hearing 0:40:09.505 advertisements from node E. When it stops hearing 0:40:12.121 advertisements from node E, eventually it's going to do 0:40:15.064 something. It's basically going to expire 0:40:17.245 that entry from its link table. So after it hasn't heard 0:40:20.243 advertisements from node E for a while, it'll expire that entry. 0:40:23.677 And then, sometime later it will hear an advertisement from 0:40:26.838 node C saying I know how to get to node E. 0:40:30 And now, node D can go ahead and integrate this new path into 0:40:33.867 its routing table. Now, notice that this process 0:40:36.832 is just going to propagate through the network. 0:40:39.796 So, once node D stops hearing advertisements for how to reach 0:40:43.664 node E, it's going to stop sending advertisements for how 0:40:47.273 to reach node E. And so, similarly, 0:40:49.464 because B is also routing its information through, 0:40:52.623 had previously been routing its packets to reach node E by way 0:40:56.554 of D, it's going to say, oh, well, I stop hearing about 0:41:00.035 this route, DE, from node D. 0:41:03 So I'm going to stop using this. 0:41:04.795 And instead, then sometime later, 0:41:06.648 it's going to hear about this new route, DCE, 0:41:09.195 and it's going to integrate that into its table. 0:41:11.917 So, it's this process where there is this sort of process 0:41:15.16 whereby nodes continually advertise and integrate routes. 0:41:18.403 And there's this sort of interesting thing which happens, 0:41:21.645 which is that nodes forget about routes that they haven't 0:41:24.888 heard about for awhile when they miss an advertisement. 0:41:29 So this, forgetting about routes is an important sort of 0:41:32.631 principle that's often employed in networking called soft state. 0:41:36.79 And the idea with soft state is that you should only keep 0:41:40.487 information when that information gets refreshed. 0:41:43.656 So all the information that you have has some time limit on it. 0:41:47.749 And when you haven't heard that information refreshed after some 0:41:51.908 time limit, you throw it out. So that's what we're doing with 0:41:55.869 the routes here. And so we only keep routes that 0:41:58.972 we have heard advertised recently, basically. 0:42:03 And that has this nice property that it allows us to adapt to 0:42:06.615 faults within the network, right? 0:42:08.483 So we saw a failure happen. We saw that sometime after that 0:42:11.979 failure, this soft state property would cause the 0:42:14.871 information about the link between D and E to disappear 0:42:18.126 from the network, and then nodes would rediscover 0:42:21.018 their new links that allow them to connect to node E. 0:42:24.152 OK, so what I want to do now, so this is sort of the basic 0:42:27.587 process. And, path vector routing is 0:42:29.696 fairly similar to the way that routing in the Internet actually 0:42:33.433 works. For next time in recitation, 0:42:36.804 you're going to talk about a protocol called the border 0:42:40.284 gateway protocol. And you're going to actually 0:42:43.184 study in much more detail how Internet routing works. 0:42:46.535 But this is a nice simple model of how routing in a small 0:42:50.144 network might act. So the problem with what we've 0:42:53.237 discussed so far, while we are here, 0:42:55.493 is that if this is a very large network, these number of routes 0:42:59.488 that you're going to have to hear about is really huge. 0:43:04 So suppose that this, instead of being a five node 0:43:06.256 network was a million node network. 0:43:07.822 Well, now every node is going to have a table that's a million 0:43:10.631 entries long, right? 0:43:11.506 That's going to be really big, and it's going to have to hear 0:43:14.269 a million advertisements. And every time there's a 0:43:16.526 failure, well, that's going to be a pain 0:43:18.322 because we're going to have to wait for that information about 0:43:21.131 that failure to propagate through the whole network. 0:43:23.48 So in some sense, this simple path vector routing 0:43:25.69 protocol we have doesn't really meet the scalability goal that 0:43:28.5 we want to scale this network up to a very large size. 0:43:32 So we have an issue with path vector routing and its 0:43:35.804 scalability. OK, so the solution is a 0:43:38.49 solution that is often used when we have a scalability problem in 0:43:43.264 a computer system: its hierarchy, 0:43:45.651 OK? So let's see what I mean by 0:43:47.889 that. So, you remember when I was 0:43:50.277 showing those pictures of the Internet, there are these 0:43:54.305 different kind of subnetworks that were forming over time on 0:43:58.706 the West Coast, on the East Coast, 0:44:01.168 or there was the ARPANET, and the NSFNET, 0:44:04.152 and the MILNET that were these sort of different networks 0:44:08.777 that were all a part of the Internet as a whole. 0:44:14 But they were these sort of different regions that were 0:44:17.55 separately administered, and that often corresponded to 0:44:21.1 a specific organization like the NSF or like the military. 0:44:24.847 So, oftentimes these regions, so it's very common when you 0:44:28.595 look at any network to have these kinds of regions in them. 0:44:33 So, for example, clearly there's a network that 0:44:36.274 is MIT's network, right? 0:44:37.912 And Harvard, for example, 0:44:39.62 has a network that's Harvard's network, right? 0:44:42.824 And those two things are sort of logical regions that define 0:44:47.024 different parts or different groups within a network. 0:44:50.726 So if you were to look at any network, you would see that sort 0:44:55.069 of almost any large network is organized in this way into these 0:44:59.483 regions. In the paper next time, 0:45:01.69 we're going to call these regions autonomous systems or 0:45:05.534 AS's, OK, so autonomous as in sort of operating on its own, 0:45:09.663 operating without being dependent on the other parts of 0:45:13.508 the system. So for example, 0:45:16.74 if MIT's Internet connection went down, connection to the 0:45:19.503 outside world went down, that wouldn't stop you from 0:45:22.019 being able to connect to machines within MIT, 0:45:24.19 right? So MIT is autonomous in the 0:45:25.819 sense that it continues to operate in the absence of its 0:45:28.532 connection to the rest of the Internet. 0:45:31 So let's look at a simple example of a network that has 0:45:35.366 this hierarchy property, and see how we would modify the 0:45:39.814 routing algorithm that I just talked about. 0:45:43.21 So suppose I have two small networks, each with three nodes 0:45:47.9 in them. Let's call them A, 0:45:50.002 B, C, and D, E, and F, OK? 0:45:52.024 So, these are each going to be autonomous systems, 0:45:55.986 which I'll draw by drawing a circle around them. 0:46:01 OK, so one way we could do routing would be to do what I'd 0:46:04.107 shown before, which is that each node within, 0:46:06.505 say, this autonomous region which I'll label one, 0:46:09.122 or autonomous system one, and this one two, 0:46:11.411 would have information about all the other nodes everywhere 0:46:14.573 else in the network. But that has the scalability 0:46:17.189 problem that we mentioned. So instead, what we want to do 0:46:20.242 is we want to make it so that only a few of the nodes in here, 0:46:23.567 so that we don't have to have information about all of the 0:46:26.674 nodes that are anywhere within the network. 0:46:30 We only have to know about a few nodes we say are on the edge 0:46:33.447 of each one of these nodes. So the idea is as follows. 0:46:36.492 Suppose these are connected in this way. 0:46:38.732 And what we're going to do is we're going to appoint one node 0:46:42.18 within each one of these regions. 0:46:44.018 For example, it could be several modes. 0:46:46.201 One of these regions to be a so-called border node that sort 0:46:49.591 of sits on the edge of these two regions. 0:46:51.889 And we're going to connect just those two nodes together. 0:46:55.107 So we're only going to have a small number of links between 0:46:58.439 our two AS's. And now, if we look at the 0:47:00.68 forwarding table for one of these nodes within, 0:47:03.322 say, AS1, it's going to look as follows. 0:47:07 So what I'm going to do is I'm going to write the addresses for 0:47:12.381 these different AS's as hierarchical addresses. 0:47:16.375 So, we're going to call node A's address 1.A. 0:47:20.194 And, we'll call node E's address 2.E, OK? 0:47:23.666 So, what our forwarding table looks like is a list of 0:47:28.18 addresses, and then a link to use. 0:47:32 OK, so this is going to be the forwarding table, 0:47:35.145 again, for node A. So, A is going to have an 0:47:38.023 address. It's going to have an address 0:47:40.5 1.A in it. And, the link to use, 0:47:42.574 in that case, we've already said is 0:47:44.85 end-to-end. OK, it's going to have an 0:47:47.259 address 1.B. So, to get to B, 0:47:49.133 A is going to route by this link here, which let's call this 0:47:53.082 link number one, OK? 0:47:54.354 So, it's going to route to link one. 0:47:56.696 And it's going to have a connection to 1.C, 0:47:59.507 which is going to route via this link here, 0:48:02.318 which let's label that two, OK? 0:48:06 So now, we don't have any information about how to reach 0:48:09 network two. So what we're going to do is 0:48:11.181 rather than storing information about every machine in network 0:48:14.563 two, we're going to store just information about how to reach 0:48:17.836 the edge of network two, and then trust that network two 0:48:20.836 is going to be able to route to anybody whose address begins 0:48:24.054 with two. So what we're going to do is 0:48:26.072 we're going to say two dot star, right, two dot everybody. 0:48:30 So, we'll star the character that says anybody whose address 0:48:33.766 has this prefix two dot. So, anybody whose address has 0:48:36.985 prefix two dot, we are just going to send to 0:48:39.598 node C. So, we're just going to send 0:48:41.724 that over link two, OK? 0:48:43.06 And so, similarly on the other side, we're going to have a 0:48:46.523 table. Each of these nodes is going to 0:48:48.771 have an entry, one dot star, 0:48:50.411 that specifies that it should route messages for network one 0:48:53.995 through node D. OK, so you see in this way now 0:48:58.009 what we've been able to do is to make it so that each one of 0:49:01.961 these autonomous systems, sort of each node only knows 0:49:05.511 how to route to other nodes within its sort of group, 0:49:08.995 and that in order to cross groups, you route through one of 0:49:12.88 these special sort of border routers or border nodes, 0:49:16.363 OK? And, we're going to talk about 0:49:18.574 the way that this border routing protocol actually works in the 0:49:22.727 Internet in recitation. But you can see that what we've 0:49:26.344 done is we've accomplished this sort of hierarchy so we can make 0:49:30.564 this system, we make these tables much smaller by including 0:49:34.784 these star entries in them. It's worth just mentioning as a 0:49:40.093 final caveat that we have given up something for doing this. 0:49:44.077 OK, what we've done is we've forced every node that's within 0:49:48.062 one of these regions. Now, this node's address, 0:49:51.168 in order for this node 1.A to be able to continue to operate, 0:49:55.22 it can only be connected to somebody like C who can 0:49:58.597 advertise information about all the nodes whose names begin with 0:50:02.851 one. So, we've basically forced this 0:50:06.097 sort of network to have this. By imposing this kind of a 0:50:09.819 hierarchy on the network we've limited, to some extent, 0:50:13.473 the ability of these nodes to move between networks because 0:50:17.398 node A can only be advertised by way of node C. 0:50:20.511 So in the Internet today, there's actually multiple 0:50:23.894 layers of hierarchy. So you may have noticed that 0:50:27.142 Internet IP addresses have the form A.B.C.D. 0:50:31 So, it's a four layer hierarchy that we have in the Internet. 0:50:35.051 You guys will learn more about this on Thursday in recitation. 0:50:39.169 And what we'll do next time is talk about the end-to-end layer, 0:50:43.355 and eliminating some of the limitations of the best effort 0:50:47.204 network.