CS144
An Introduction to Computer Networks

What the Internet is

The 4 Layer Internet Model

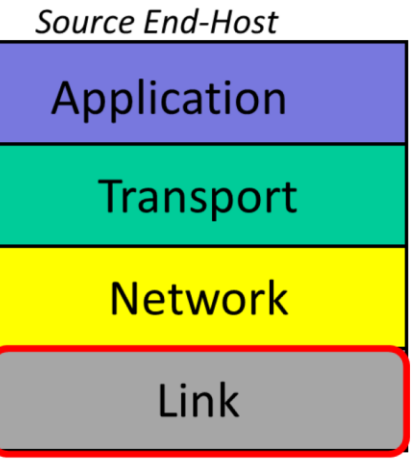
Nick McKeown
Professor of Electrical Engineering
and Computer Science, Stanford University

In the last video you learned how different applications -- such as BitTorrent, Skype and the World Wide Web – communicate over the Internet using a bi-directional reliable byte stream.

It takes a lots of different pieces working together to create this reliable communication model for our applications. But even though we use a huge variety of different Internet applications, sending many kinds of data at very different speeds, there are surprisingly

strong similarities in the way applications send and receive data. For example, applications want to send and receive data without having to worry about the path, or route, that the data takes across the Internet. And almost all applications want to be confident that their data is correctly delivered, with any lost or corrupted data automatically retransmitted until it is received correctly.

The 4 Layer Internet Model



The early Internet pioneers created the "4 Layer Internet Model" to describe the hierarchy of operations that make up the Internet, so that applications can reuse the same building blocks over and over again, without

having to create them from scratch for every application. Layering is an important and frequently used concept in networking and we'll be seeing it many times throughout this course. There is even a video devoted just to the concept of layering.

Let's take a look at what each layer of the 4 Layer Internet model does. It helps to remember that all four layers are there to enable applications

in the end-hosts communicate reliably.

To explain how it works, I’m going to start at the bottom layer. We’ll see that each layer has a different responsibility, with each layer building a service on top of the one below, all the way to the top where we have the bi-directional reliable byte stream communication between applications.

Let’s start with the Link Layer.

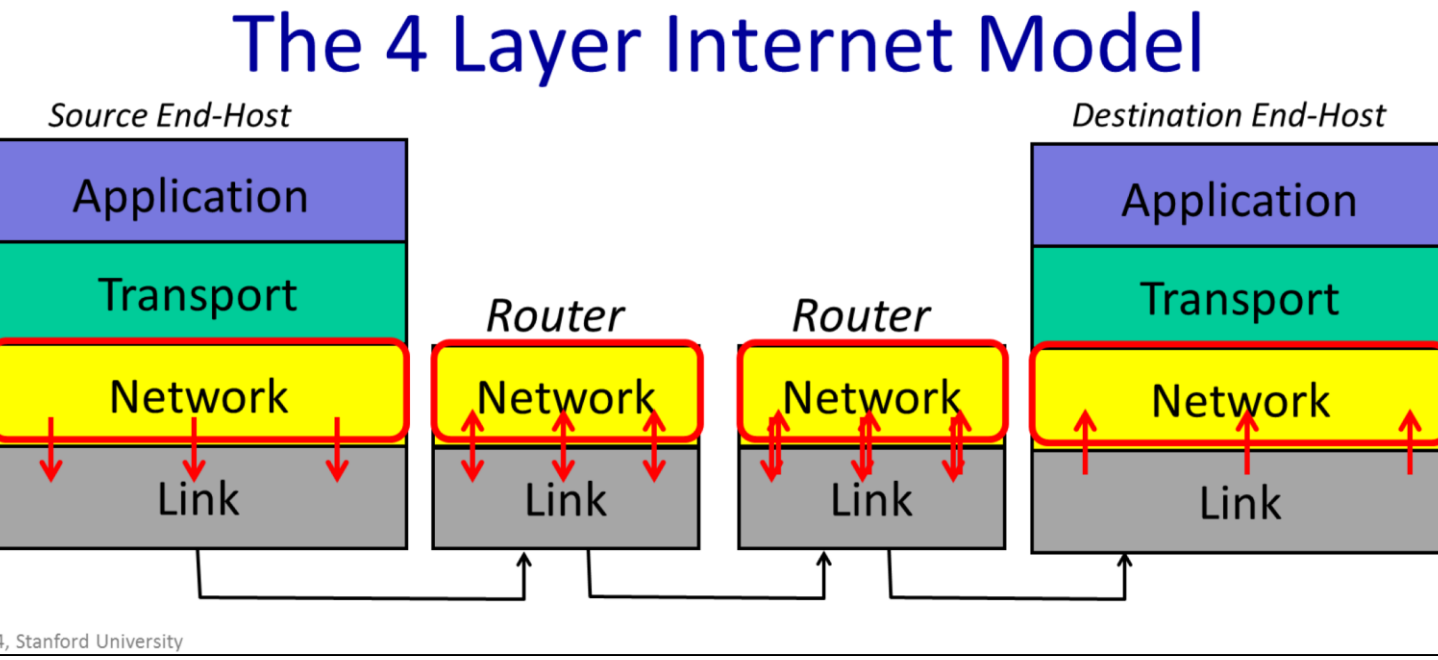
<click>

The Internet is made up of end-hosts, links and routers. Data is delivered hop-by-hop over each link in turn. Data is delivered in packets. A packet consists of the data we want to be delivered, along with a header that tells the network where the packet is to be delivered, where it came from and so on.

<Draw topology and hop by hop delivery of data>

The Link Layer’s job is to carry the data over one link at a time.

You’ve probably heard of Ethernet and WiFi – these are two examples of different Link layers.



The next layer up is, for us, the most important layer: the Network layer.

The network layer's job is to deliver packets end-to-end across the Internet from the

source to the destination.

A packet is an important basic building block in networks. A packet is the name we give to a self-contained collection of data, plus a header that describes what the data is, where it is going and where it came from. You will often see packets drawn like this: <draw a packet with header and data>

Network layer packets are called datagrams. They consist

of some data and a head containing the “To” and “From” addresses – just like we put the “To:” and “From” addresses on a letter. <Draw a datagram with To/From addresses> .

The Network hands the datagram to the Link Layer below <click to wipe arrows down>, telling it to send the datagram over the first link. In other words, the Link Layer is providing a *service* to the Network Layer. Essentially, the

Link Layer says: “if you give me a datagram to send, I will transmit it over one link for you”.

At the other end of the link is a router. The Link Layer of the router accepts the datagram from the link, and hands it up to the Network Layer in the router. The Network Layer on the router examines the destination address of the datagram, and is responsible for routing the datagram one hop at a time

towards its eventual destination. It does this by sending to the Link Layer again, to carry it over the next link. And so on until it reaches the Network Layer at the destination. <sequence of clicks shows the steps>

Notice that the Network Layer does not need to concern itself with *how* the Link Layer sends the datagram over the link. In fact, different Link Layers work in very different

ways; Ethernet and WiFi are clearly very different. This separation of concerns between the Network Layer and the Link Layer allows each to focus on its job, without worrying about how the other layer works. It also means that a single Network Layer has a common way to talk to many different Link Layers by simply handing them datagrams to send. This separation of concerns is made possibly by the modularity of each layer and a common well-defined API to the layer below.

The network layer is “special”

We must use the Internet Protocol (IP)

- IP makes a best-effort attempt to deliver our datagrams to the other end. But it makes no promises.
- IP datagrams can get lost, can be delivered out of order, and can be corrupted. There are no guarantees.

In the internet, the network layer is special: When we send packets into the Internet, we must use the Internet Protocol. It is the Internet Protocol, or IP, that holds the Internet together. We’ll learn more about the details of IP in later videos. But for now it’s good to know some basic facts about IP.

<bulleted list to appear on screen>

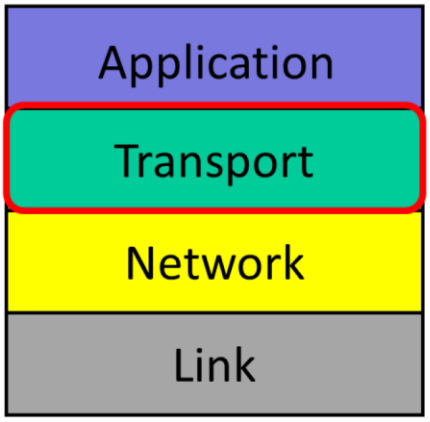
•IP makes a best-effort attempt to deliver our packets to the other end. But it makes no promises.

•IP packets can get lost, can be delivered out of order, and can be corrupted. There are no

guarantees.

This may come as a surprise. How can the Internet work at all when the packets are not guaranteed to be delivered? Well, if an application wants a guarantee that its data will be retransmitted when necessary and will be delivered to the application in order and without corruption then it needs another protocol running on top of IP. This is the job of the Transport Layer….

The 4 Layer Internet Model



The most common Transport Layer is TCP <draw TCP: Transmission Control Protocol>, or the Tranmission Control Protocol. (You have probably heard of TCP/IP, which is when an application uses

both TCP and IP together).

TCP makes sure that data sent by an application at one end of the Internet is correctly delivered – in the right order - to the application at the other end of the Internet. If the Network Layers drops some datagrams, TCP will retransmit them, multiple times if need-be. If the Network Layer delivers them out of order – perhaps because two packets follow a different path to their

destination – TCP will put the data back into the right order again. In later videos you will learn a lot about TCP and how it works. For now, the main thing to remember is that TCP provides a service to an application guaranteeing correct in-order delivery of data, running on top of the Network Layer service, which provides an unreliable datagram delivery service.

As I’m sure you can imagine,

applications such as a web client, or an email client, find TCP very useful indeed. By employing TCP to make sure data is delivered correctly, they don’t have to worry about implementing all of the mechanisms inside the application. They can take advantage of the huge effort that developers put into correctly implementing TCP, and reuse it to deliver data correctly. Reuse is another big advantage of layering.

But not all applications need data to be delivered correctly. For example, if a video conference application is sending a snippet of video in a packet, there may be no point waiting for the packet to be retransmitted multiple times; better to just move on. Some applications just don’t need the TCP service.

If an application doesn’t need reliable delivery, it can use the

much simpler UDP – or user datagram protocol – instead <draw: UDP: User Datagram Protocol>. UDP just bundles up application data and hands it to the Network Layer for delivery to the other end. UDP offers no delivery guarantees.

In other words, an Application has the choice of at least two different Transport Layer services: TCP and UDP. There are in fact many other choices too, but these are the most

commonly used transport layer services.

The 4 Layer Internet Model

Finally we have the Application Layer at the top of the 4 Layer Model. There are of course many thousands of applications that use the Internet. While each application is different, it can reuse the Transport Layer by

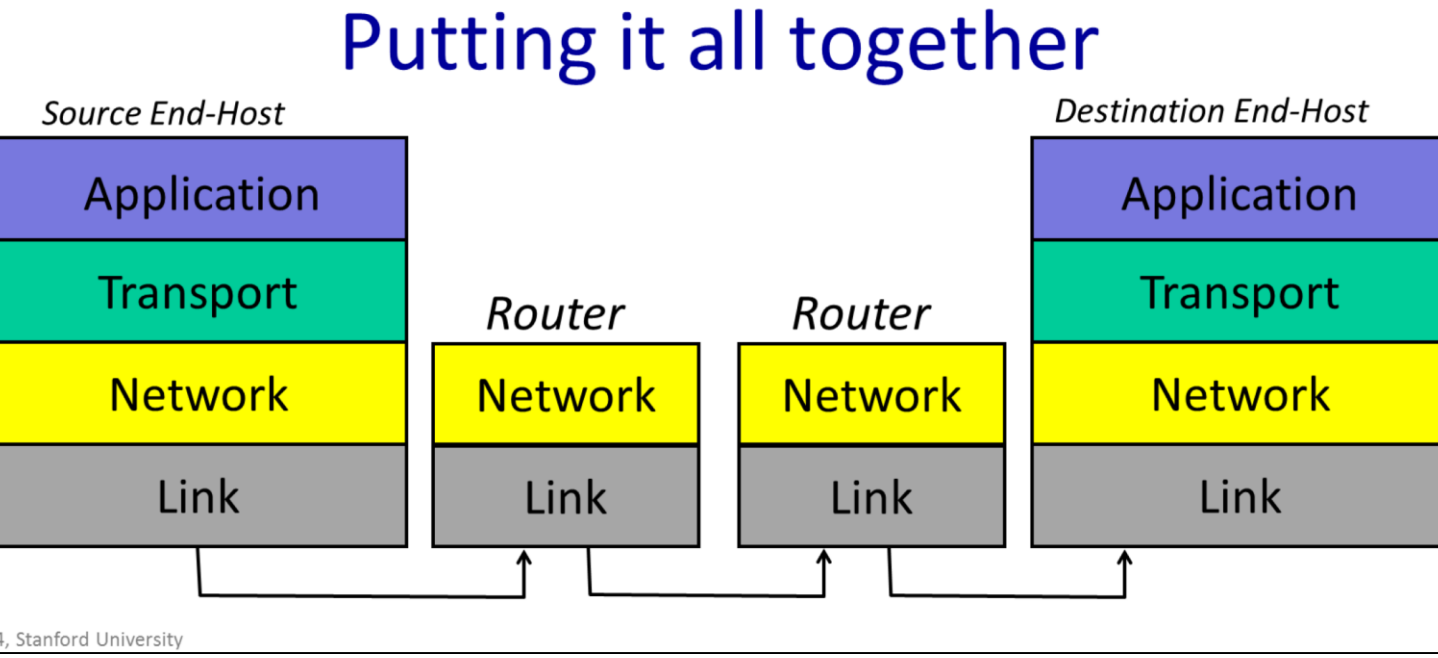
using the well-defined API from the Application Layer to the TCP or UDP service beneath.

As we saw in the last video, applications typically want a bi-directional reliable byte stream between two end points. They can send whatever byte-stream they want, and Applications have a protocol of their own that defines the syntax and semantics of data flowing between the two end points. <click to make second 4layer

stack appear> For example, as we saw in the last video, when a web client requests a page from a web server <draw them on Application layer>, the web client sends a GET request. This is one of the commands of the hypertext transfer protocol, or http. http dictates that the GET command is sent as an ASCII string, along with the URL of the page being requested. As far as the Application Layer is concerned, the GET request is sent directly to its peer at the other end – the web server

Application <draw dotted lines to show peer communication of GET request>. The Application doesn’t need to know how it got there, or how many times it needed to be retransmitted. At the web client, the Application Layer hands the GET request to the TCP layer, which provides the service of making sure it is reliably delivered. It does this using the services of the Network layer, which in turn uses the services of the Link Layer.

We say that each layer communicates with its peer layer <draw dotted lines>. It’s as if each layer is only communicating with the same layer at the other end of the link or Internet, without regard for how the layer below gets the data there.



Putting it all together then….

Network engineers find it convenient to arrange all the functions that make up the Internet into *Layers*.

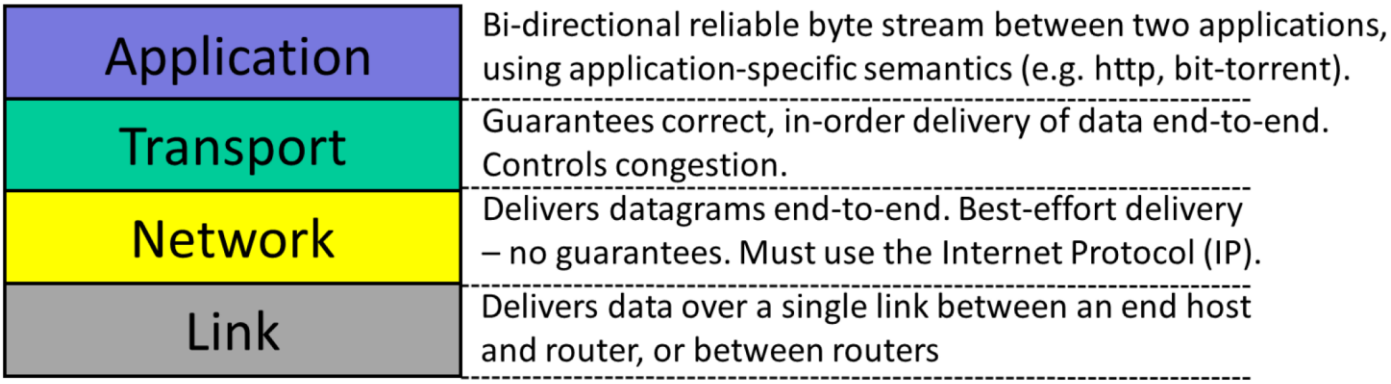
At the top is the Application,

such as BitTorrent or Skype or the world wide web, which talks to its peer layer at the destination <draw dotted line>. When the application has data to send, it hands the data to the Transport layer <draw red arrows down>, which has the job of delivering the data reliably (or not) to the other end. The Transport Layer sends data to the other end by handing it to the Network Layer <draw red arrow down>, which has the job of breaking the data into packets, each with the correct

destination address. Finally, the packets are handed to the Link Layer, which has the responsibility of delivering the packet from one hop to the next along its path.

The data makes its way, hop by hop, from one router to the next. The Network Layer forwards it to the next router, one at a time, until it reaches the destination. There, the data is passed up the layers, until it reaches the Application.

Summary of 4 Layer Model

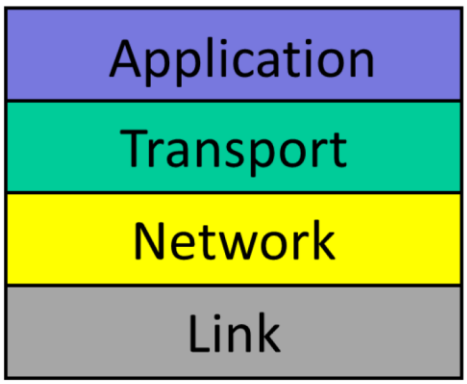


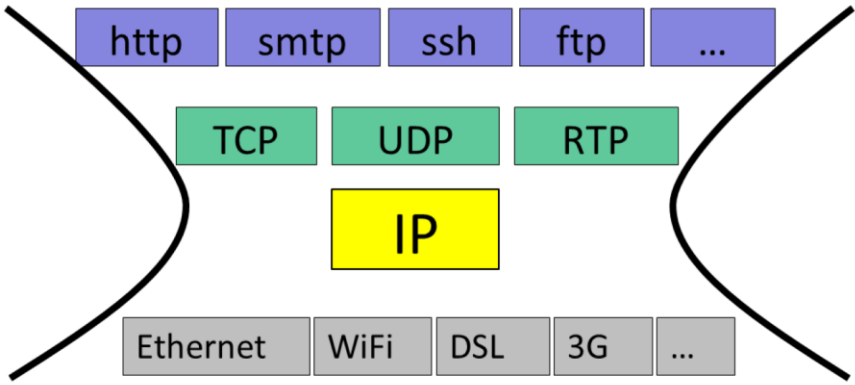
<read 4 pieces of text on slide. Click to make each appear in turn>

Two extra things you need to know…

Now we've seen what the 4 layers are, I'll finish with two extra things you need to know….

IP is the “thin waist”

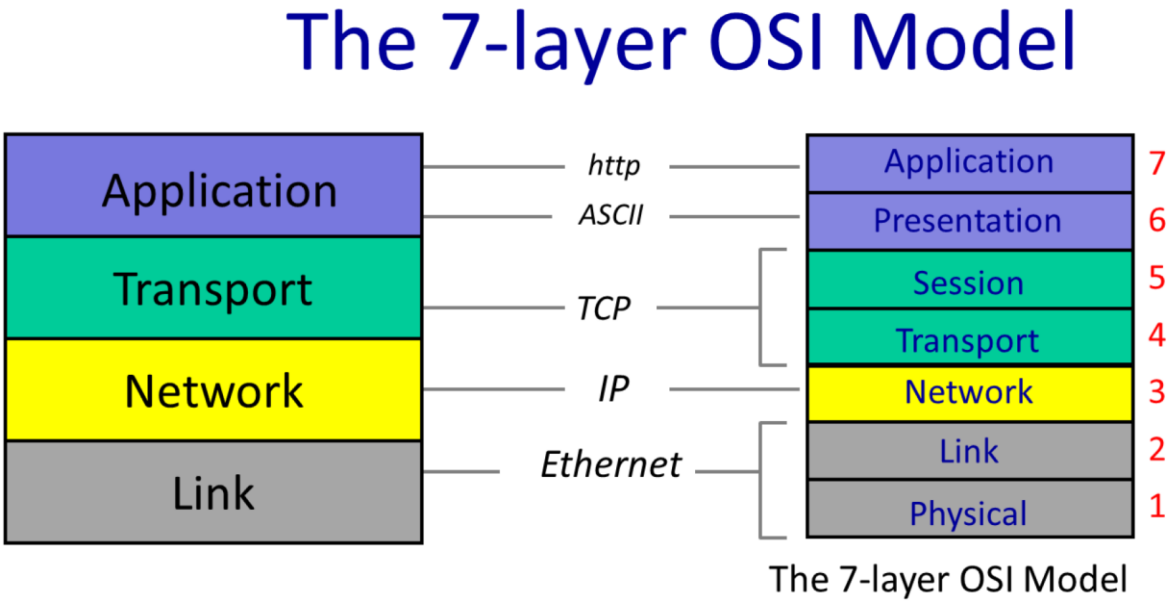




The first is that IP is often referred to as “the thin waist” of the Internet <click>. This is because if we want to use the Internet, we have to use the Internet Protocol, or IP. We have no choice.

But we have lots of choices for Link Layers: IP runs over many different Link Layers, such as <click> Ethernet, WiFi, DSL, 3G cellular, and so on.

On top of the unreliable IP layer, we can choose between many different transport layers <click>. We already saw TCP and UDP. There is RTP for real time data and many others too. And of course there are tens of thousands of different applications. <click>.



The second thing you should know is that in the 1980s the International Standards Organization, or ISO created a 7-layer model to represent any type of network <click>. It was called the 7-layer Open Systems

Interconnection or OSI model. We don’t spend any time on it in this course because it has been replaced by the 4-layer Internet model. If you’re interested, you’ll find any networking textbook and Wikipedia describes the 7 layers in lots of detail. The 7 layer model defined layers that were combined in the 4 layer Internet model. For example, what we call the Link Layer today <click> was separated into the Link Layer – that defined the framing format – and the

Physical Layer that defined things like the voltage levels on a cable, or the physical dimensions of a connector. <click> The Network Layers are pretty much the same. <click> The Transport and <click> Application Layers are each represented by 2 layers in the OSI model. These <annotate the text in the middle> are examples of commonly used Internet protocols, and how they map to the OSI numbering scheme.

Today, the only real legacy of the 7-layer OSI model is the numbering system. You'll often hear network engineers refer to the Network Layer as "Layer 3", <circle it> even though it is the 2nd layer up from the bottom in the Internet Layer. Similarly, you'll hear people refer to Ethernet as a Layer 2 protocol <circle it>, and the Application as Layer 7 <circle it>.

<The End>