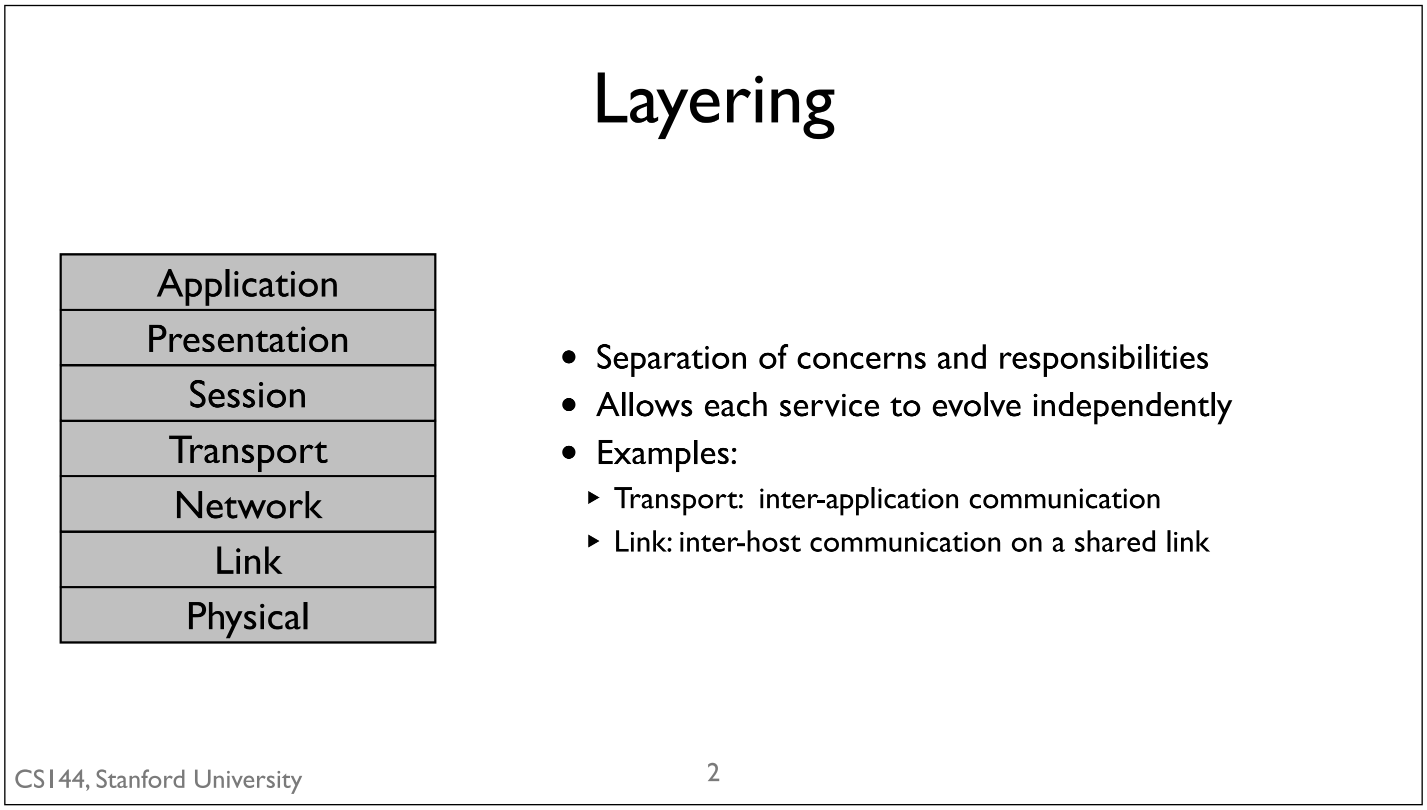
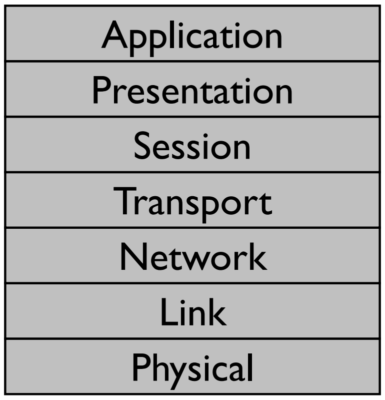
Principle: Encapsulation

Nick: This video is about the architectural principle we call encapsulation. Encapsulation is the result of what happens when you combine layers and packet switching. We want to break up our data into discrete units, called packets. However, each packet contains data from multiple layers. When you send a TCP segment, for example, it's inside an IP packet, which is inside an Ethernet frame. Encapsulation is how this works. Encapsulation is the principle by which you organize information in packets so that you can maintain layers, yet let them share the contents of your packets.



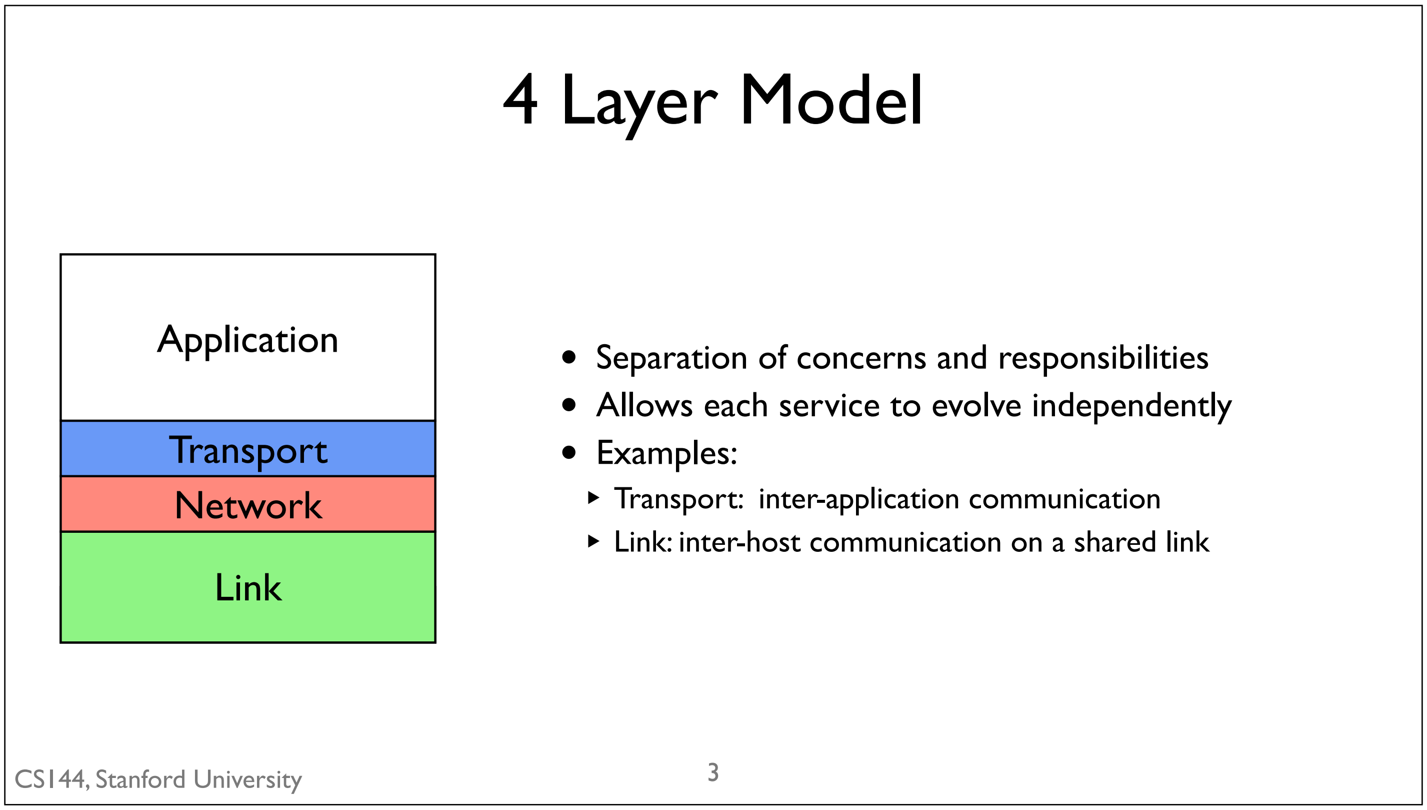
Layering



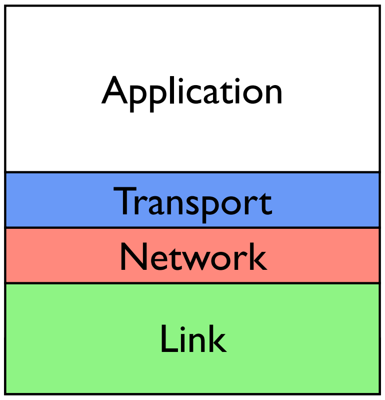
- Separation of concerns and responsibilities
- Allows each service to evolve independently
- Examples:
  - Transport: inter-application communication
  - Link: inter-host communication on a shared link

Phil: Recall that layering lets you take a complex system and break it into smaller parts. Each layer provides a service, an abstraction of the network to the layers above it. It provides this abstraction by using the layer below it. Each layer is self-contained, so as long as it provides the service expected of it, layers above don't need to worry about how. This separation of concerns means each layer can evolve independently. Just as IP, at the network layer, doesn't need to have to worry about changes to TCP at the transport layer, application layers such as HTTP don't have to worry about changes to TCP. For example, in the past few years most operating systems have changed the exact TCP algorithms they used, to better handle increasing network speeds. But a web browser works fine using both the old algorithms and the new ones.

Note that this layer picture has the 7 Layer OSI model.



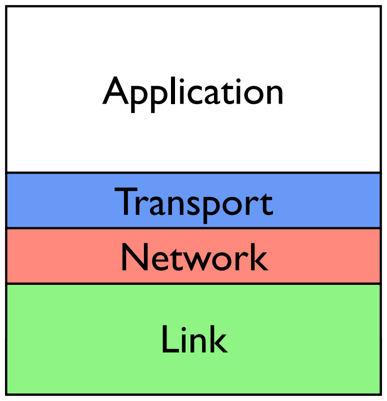
4 Layer Model



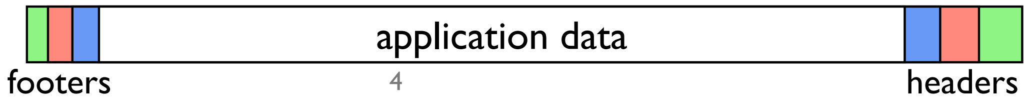
- Separation of concerns and responsibilities
- Allows each service to evolve independently
- Examples:
  - Transport: inter-application communication
  - Link: inter-host communication on a shared link

Nick: So let’s scrunch back down to the 4 layer model.

Encapsulation



- How layering manifests in data representation
- Layer N data is payload to layer N-1
- Example:
  - HTTP (web) application payload in
  - a TCP transport segment in
  - an IP network packet in
  - a WiFi link frame.

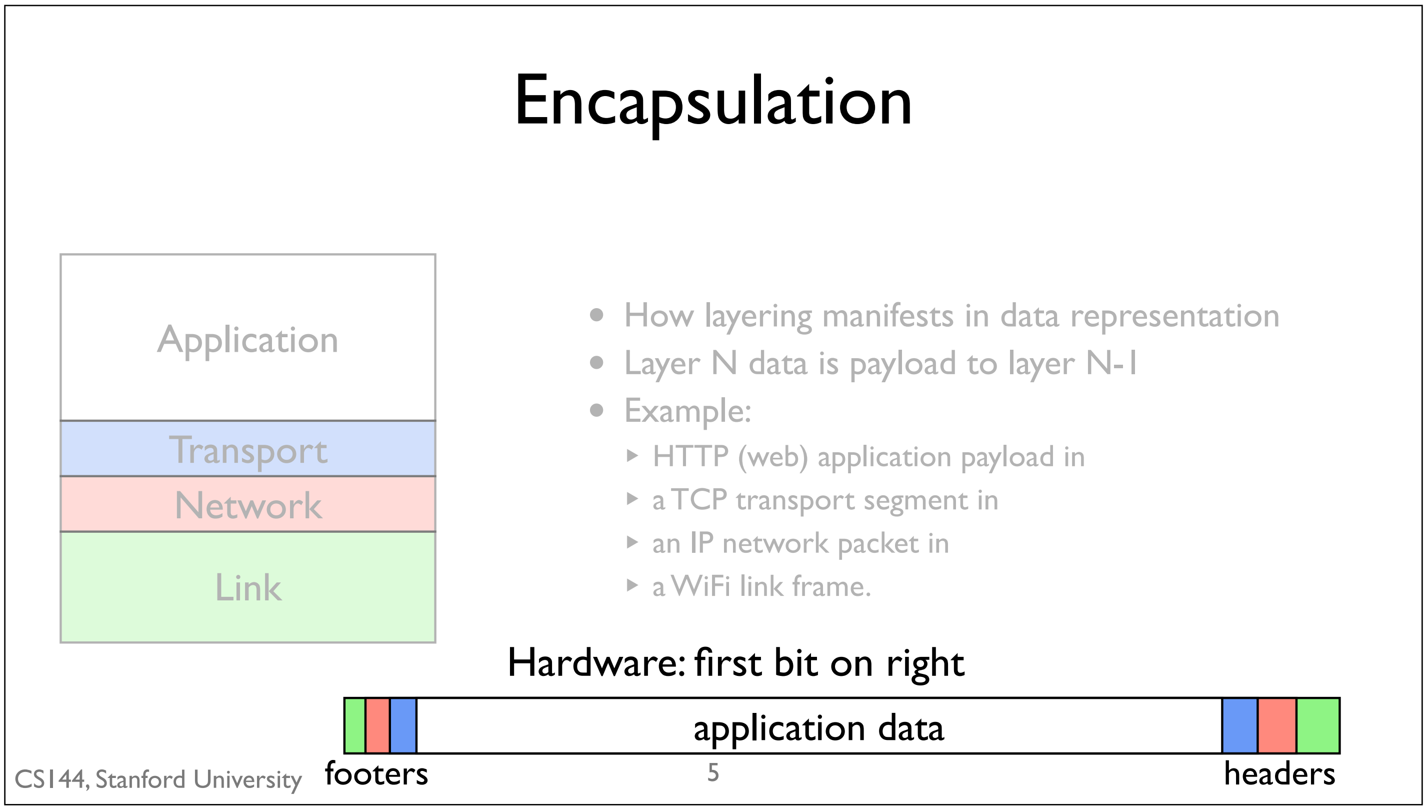


Nick: Encapsulation is the principle that lets us take protocol layers and let them easily share the storage within a packet. It's how layering manifests in the actual data representation. The way this works is each protocol layer has some headers, followed by its payload, followed by some footers. For example, an IP packet header has a source address and a destination address. To send a TCP segment with IP, we make the TCP format the payload of the IP packet. In this way, the IP packet "encapsulates" the TCP segment. IP doesn't know or care what its payload is, it just delivers packets to an end host. When the packet arrives, the host looks inside the payload, sees that it's a TCP segment, and processes it accordingly.

Nick: so here's a more complete example. Let's say that you're browsing the web using a computer connected through WiFi, wireless Ethernet. Your web browser generates an HTTP GET request. This HTTP GET request is the payload of a TCP segment. The TCP segment, encapsulating the HTTP GET, is the payload of an IP packet. This IP packet, encapsulating the TCP segment and the HTTP GET, is the payload of a WiFi frame. If you were to look at the bytes your computer sends, they'd look like this. The outermost encapsulating format is the WiFi frame, inside of which is an IP packet, inside of which is a TCP segment, inside of which is an HTTP GET.

Phil: How Nick has drawn this packet brings up something you might find confusing! It turns out there are two ways to draw packets. The difference comes from background and what part of the system you work on. Nick has drawn the packet here where the headers are on the right. The first bit of the packet is on the right and the last bit of the packet is on the left.

Nick: This makes total sense! As a router or switch sends a packet, we draw the packet moving from left to right. So the first bit to leave the router or switch is at the far right.



Encapsulation

Application
Transport
Network
Link

- How layering manifests in data representation
- Layer N data is payload to layer N-1
- Example:
  - HTTP (web) application payload in
  - a TCP transport segment in
  - an IP network packet in
  - a WiFi link frame.

Hardware: first bit on right

application data

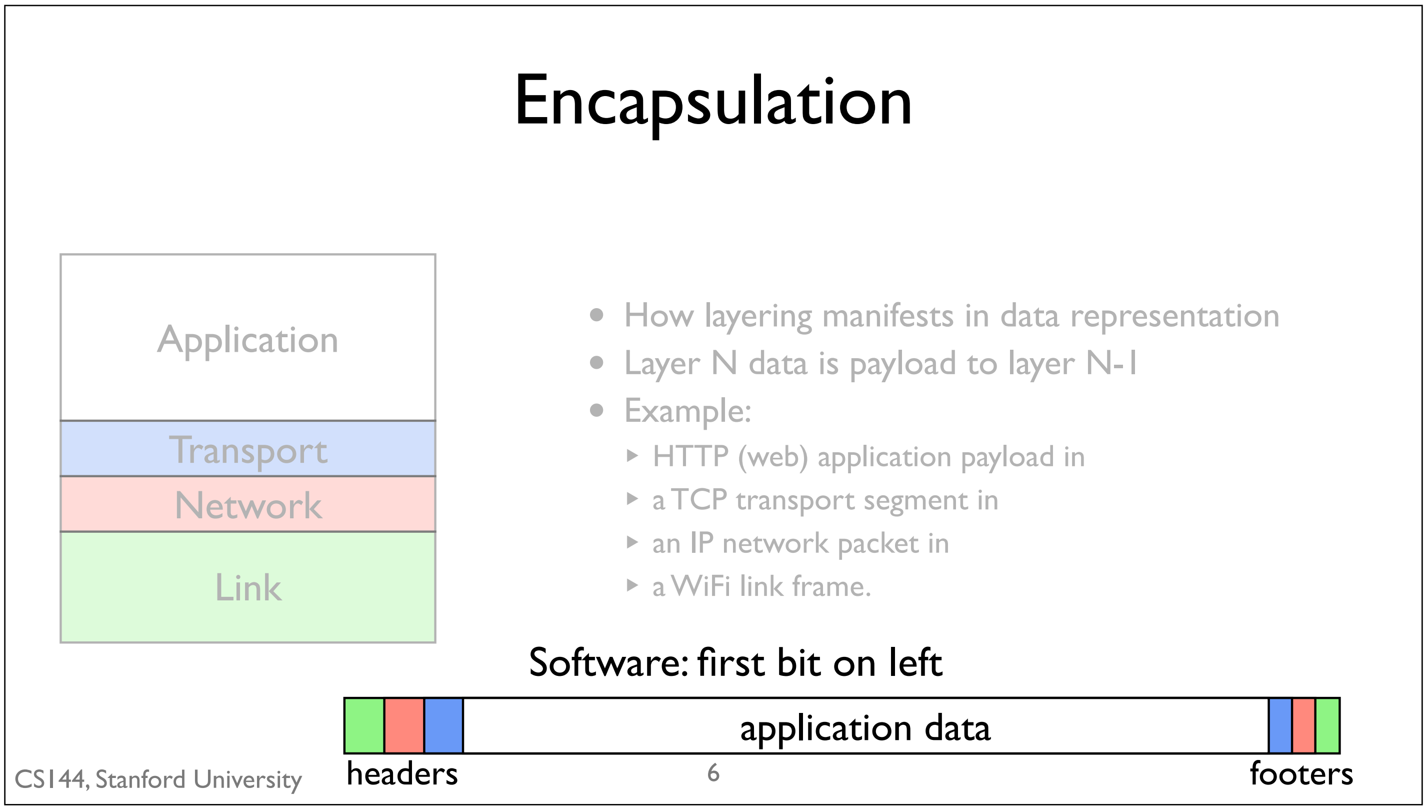
footers

headers

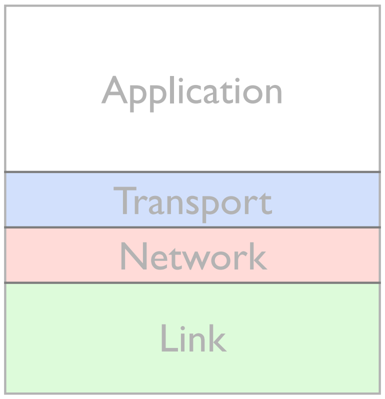
CS144, Stanford University

5

Phil:but I draw packets the other way, where the headers are on the left and the footers on the right. Like this.

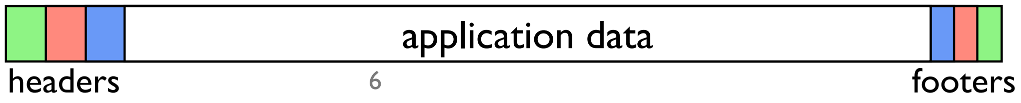


Encapsulation



- How layering manifests in data representation
- Layer N data is payload to layer N-1
- Example:
  - HTTP (web) application payload in
  - a TCP transport segment in
  - an IP network packet in
  - a WiFi link frame.

Software: first bit on left

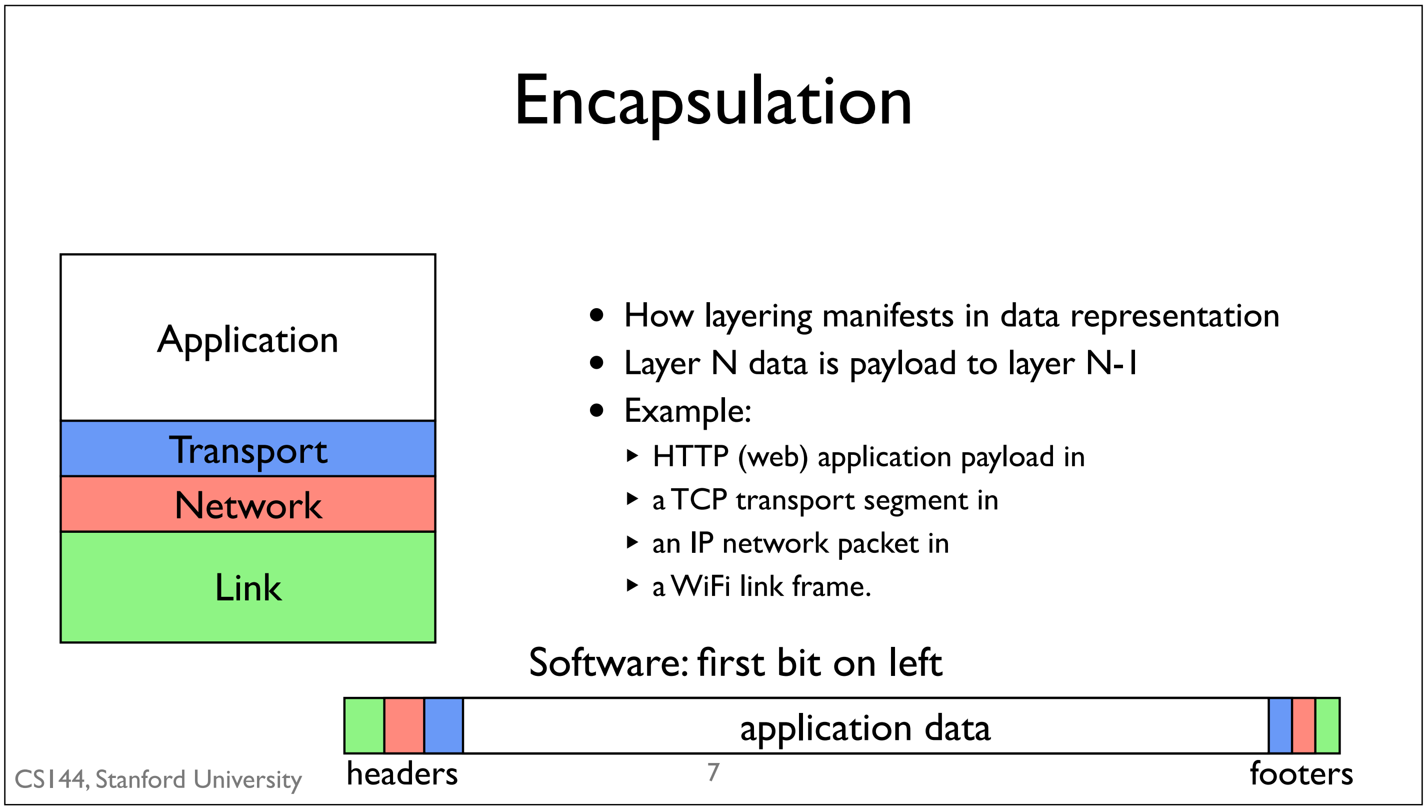


Phil: This approach comes from software. It's what you'll see when you read IETF documents and many other protocol specifications. The idea is that the beginning of the packet comes at address zero. So the first byte of a header is at address zero. Since addresses increase from left to right, this means the beginning of the packet is on the left and the end of the packet is on the right.

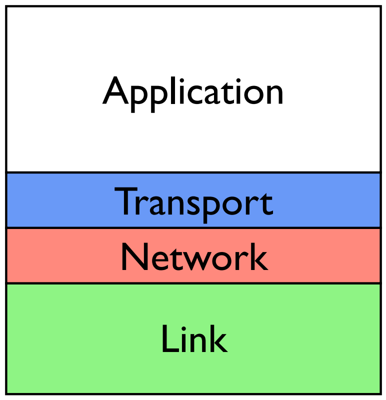
Nick: There isn't a right way or a wrong way here. Both ways of drawing packets are valuable, and depend on what you're using the drawing for. You should be comfortable with both. I'll generally draw headers on the right.

Phil: And I'll generally draw them on the left. Nick's background is electrical engineering and switch design, mine is computer science and protocol software.

=

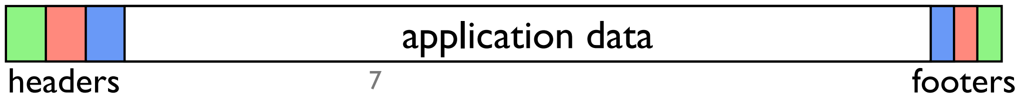


Encapsulation



- How layering manifests in data representation
- Layer N data is payload to layer N-1
- Example:
  - HTTP (web) application payload in
  - a TCP transport segment in
  - an IP network packet in
  - a WiFi link frame.

Software: first bit on left



Phil: Now let's go back to Nick's example, of an HTTP GET inside a TCP segment inside an IP packet inside a WiFi frame. Let's see what this looks like in an actual network with Wireshark. Before we started recording, I turned on Wireshark and recorded a packet trace of a web request. Let's just look at one packet. Here, we can see how Wireshark tells us that it's an Ethernet frame, inside which is an IP packet, inside which is a TCP segment, inside which is an HTTP GET. If I click on each of these protocol headers, then wireshark actually highlights where they are in the packet bytes, this gobbledigook below. WiFi comes first. Inside WiFi is IP. Inside IP is TCP. And inside TCP we can see the text of our HTTP GET!

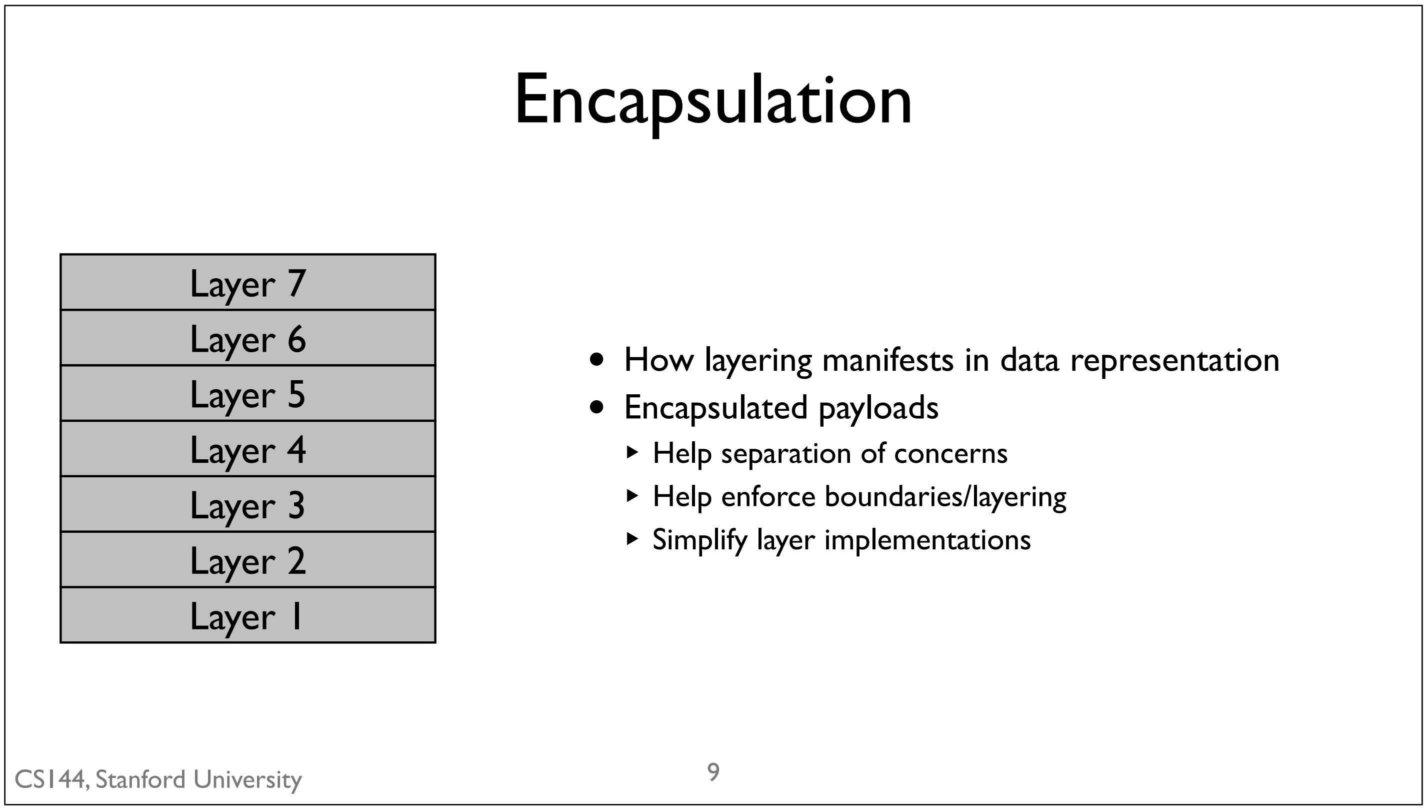
Encapsulation Flexibility

- Encapsulation allows you to layer recursively
- Example: Virtual Private Network (VPN):
  - HTTP (web) application payload in
  - a TCP transport segment in
  - an IP network packet in
  - a secured TLS presentation message in
  - a TCP transport segment in
  - an IP network packet in
  - an Ethernet link frame.

Phil: This very simple approach of encapsulating protocols within each other gives you tremendous flexibility. So far we've been talking about the 4 layer model as something completely static and inflexible. In practice, it's not like that. You can actually use encapsulation to recursively layer protocols. For example, something that's very commonly used today in offices and businesses is something called a Virtual Private Network. With a Virtual Private Network, you open a secure connection to a network you trust, such as your office, for example using Transport Layer Security (TLS). When you communicate with the Internet and send IP packets, rather than send them normally, you send them inside this VPN connection. So the IP packets go to your office network. At that point the office network can route them normally. This lets you do things like access private, protected network resources in your office. Rather than sprinkle network protections everywhere, you just have to be careful with one service, the service that lets people log into the network over the virtual private network. You do this with a gateway, a computer that accepts connections from permitted VPN clients and forwards their traffic into the private network.

So what does that look like? Let's say I'm accessing my internal company website. Well, my web browser generates an HTTP GET. Like usual, it puts this inside a TCP segment, which it puts inside an IP packet destined to the company's internal web server. But rather than put this IP packet inside a link layer frame -- I can't directly communicate with the internal web server -- my computer puts the IP packet inside a TLS segment. TLS protects the message and keeps it secret. This TLS session is inside a TCP stream that terminates at the virtual private network gateway. So *outer* TCP segment is inside an IP packet destined to the virtual private network gateway. We put this outer IP packet inside a link frame and send it to the next hop. So it looks like this:

HTTP inside TCP inside IP inside TLS inside TCP inside IP inside Ethernet.



Encapsulation

Layer 7
Layer 6
Layer 5
Layer 4
Layer 3
Layer 2
Layer 1

- How layering manifests in data representation
- Encapsulated payloads
  - Help separation of concerns
  - Help enforce boundaries/layering
  - Simplify layer implementations

Nick: Now you've heard about encapsulation, the principle that unifies layering and packet switching. Encapsulation is how we take protocol layers and assemble them into packets in a way that's flexible and maintains their separation of concerns. You saw an example of a computer can encapsulate a web request, as well as an example how one can use encapsulation in a more complex way for something like a virtual private network.