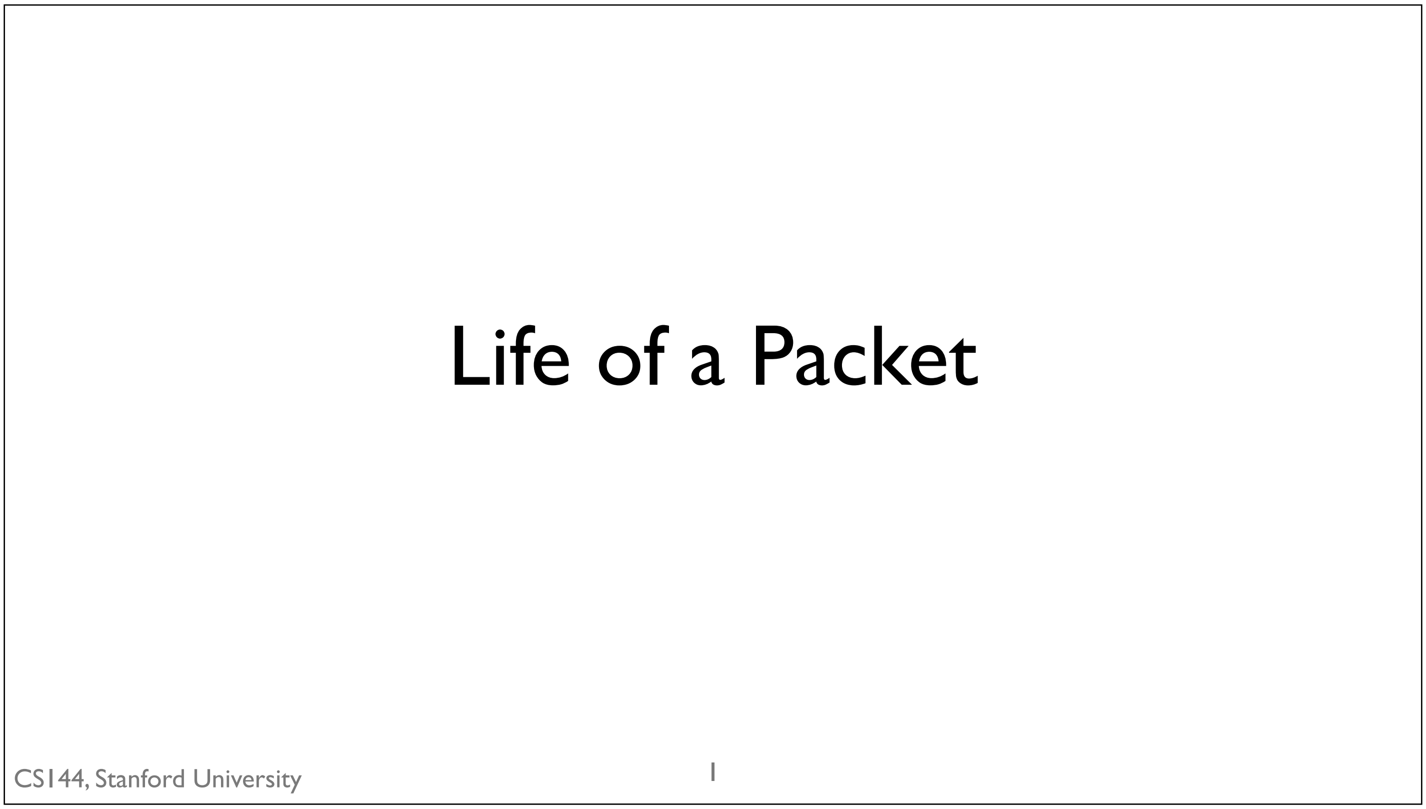


Life of a Packet

The 4-layer model of the Internet takes a stream of data from the application layer. The transport layer breaks this stream into segments of data that it reliably delivers to an application running on another computer. The transport layer sends these segments as network layer packets, which the network layer delivers to the other computer. Let’s see what that looks like in practice, the actual packets that a web browser and server send and receive.

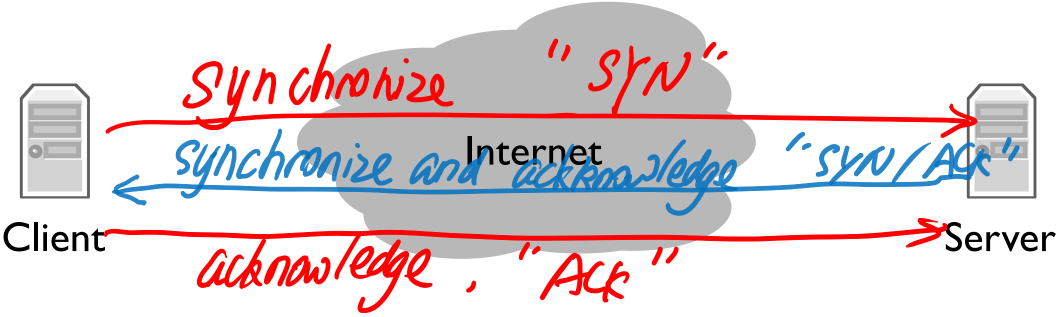
Transport layer

TCP Byte Stream

"3-way handshake"

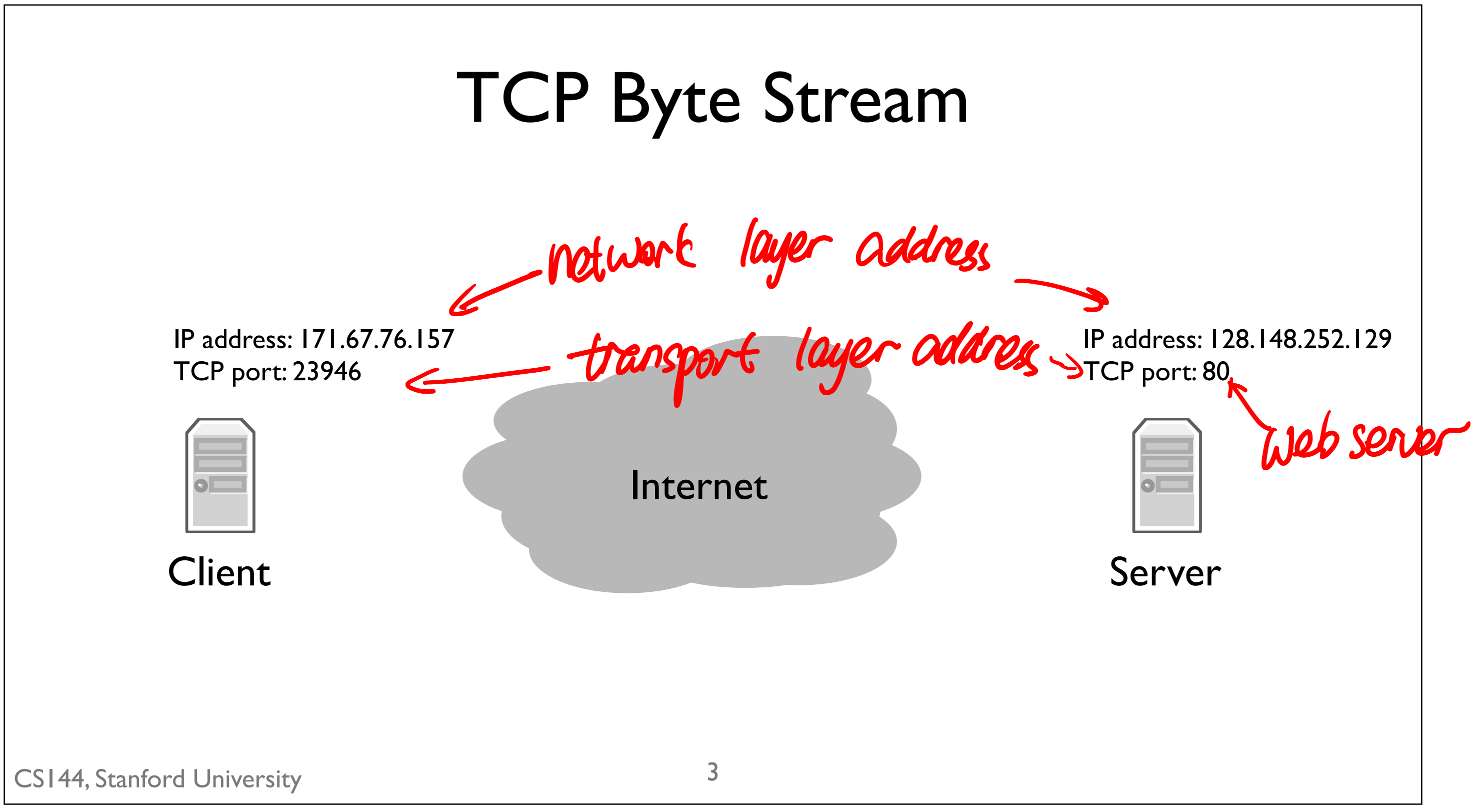
SYN: synchronize Sequence Numbers 同步序列编号 "SYN · SYN/ACK. ACK"

ACK: acknowledge character 确认字符



First, let's look at the transport layer. Almost all web traffic is over TCP, the Transport Control Protocol. In its typical operation, there's a client and a server. A server listens for connection requests. To open a connection, a client issues a connection request, which the server responds to. I won't go into the details of exactly how this works, but it turns out this exchange takes three messages, something called the "three way handshake."

The first step of handshake is when the client sends a "synchronize" message to the server, often called a SYN. The second step is when the server responds with a "synchronize" message that also acknowledges the clients "synchronize", or a "synchronize and acknowledge message", often called a SYN-ACK. The third and final step is when the client responds by acknowledging the server's synchronize, often called an ACK. So often the three way handshake is described as "synchronize, synchronize and acknowledge, acknowledge", or "SYN, SYN-ACK, ACK".



TCP Byte Stream

network layer address

IP address: 171.67.76.157
TCP port: 23946

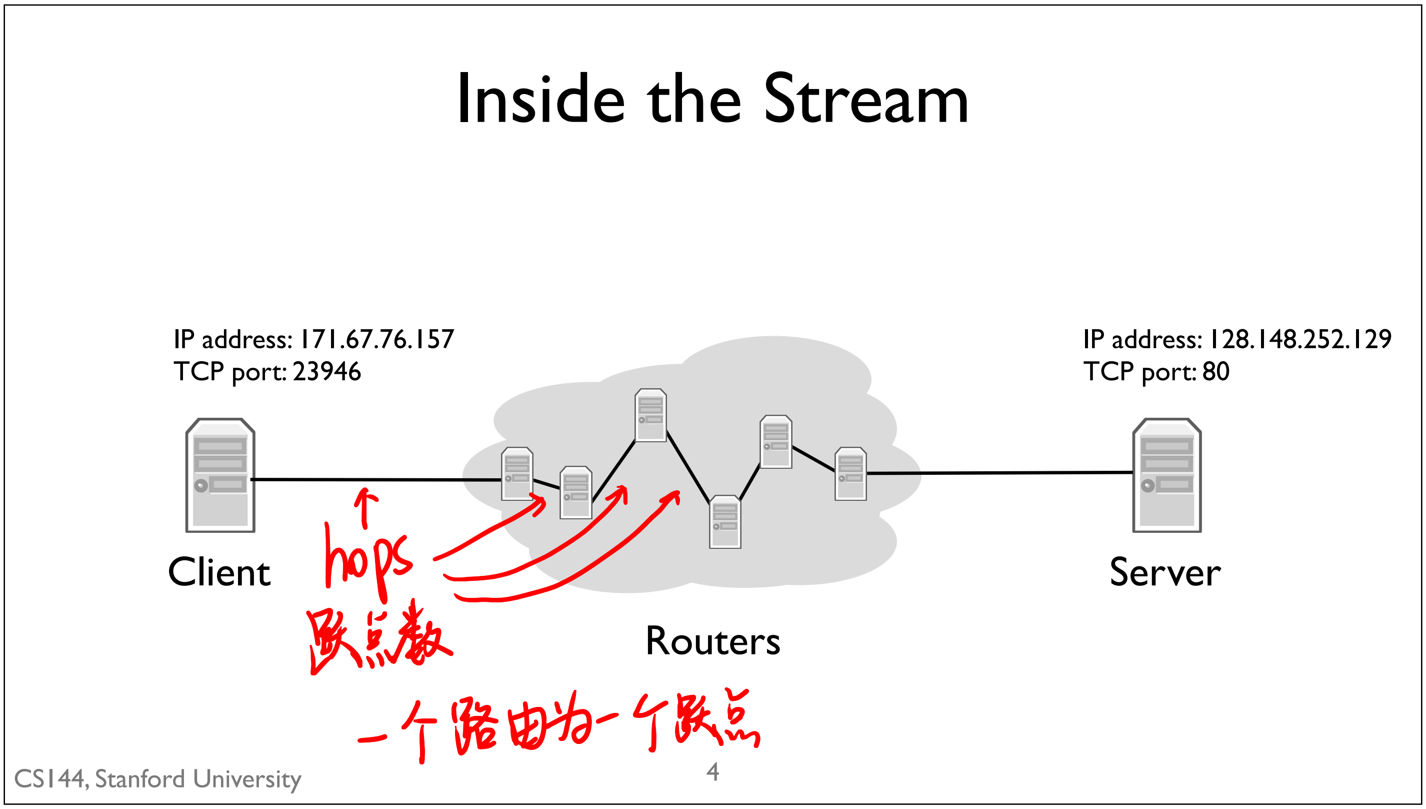
transport layer address

IP address: 128.148.252.129
TCP port: 80

web server

Recall that the network layer is responsible for delivering packets to computers, but the transport layer is responsible for delivering data to applications. From the perspective of the network layer, packets sent to different applications on the same computer look the same. This means that to open a TCP stream to another program, we need two addresses. The first, an Internet Protocol address, is the address the network layer uses to deliver packets to the computer. The second, the TCP port, tells the computer's software which application to deliver data to. Web servers usually run on TCP port 80. So when we open a connection to a web server, we send IP packets to the computer running the web server whose destination address is that computer's IP address. Those IP packets have TCP segments whose destination port is 80.

段 目地



Inside the Stream

IP address: 171.67.76.157
TCP port: 23946

Client

IP address: 128.148.252.129
TCP port: 80

Server

Routers

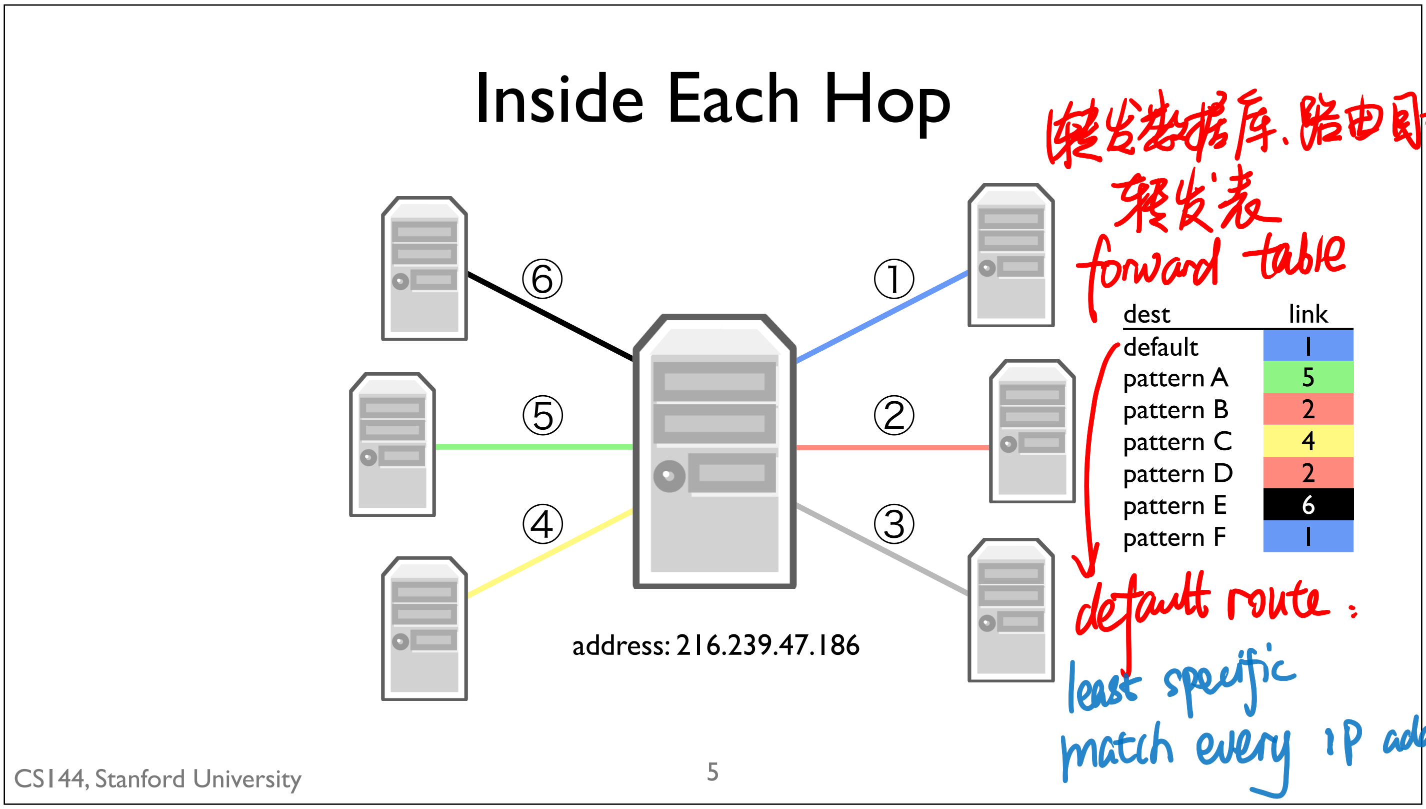
hops
跳点数

一个路由为一个跳点

But how do those IP packets get to their destination? We don't have a direct wire connecting my client to the server. Instead, my client is connected to an intermediate computer, a router. This router is itself connected to other routers. IP packets between the client and server take many "hops," where a hop is a link connecting two routers. For example, since my client is on a WiFi network, the first hop is wireless to the WiFi access point. The access point has a wired connection to the broader Internet, so it forwards my client's packets along this wired hop.

中间的

A router can have many links connecting it. As each packet arrives, a router decides which of its links to send it out on. Routers have IP addresses, so it's also the case that it might not forward a packet but rather deliver it to its own software. For example, when you log into a router using TCP, the IP packets are destined to the router's own IP address.



Inside Each Hop

address: 216.239.47.186

(转发数据库、路由目录)
转发表
forward table

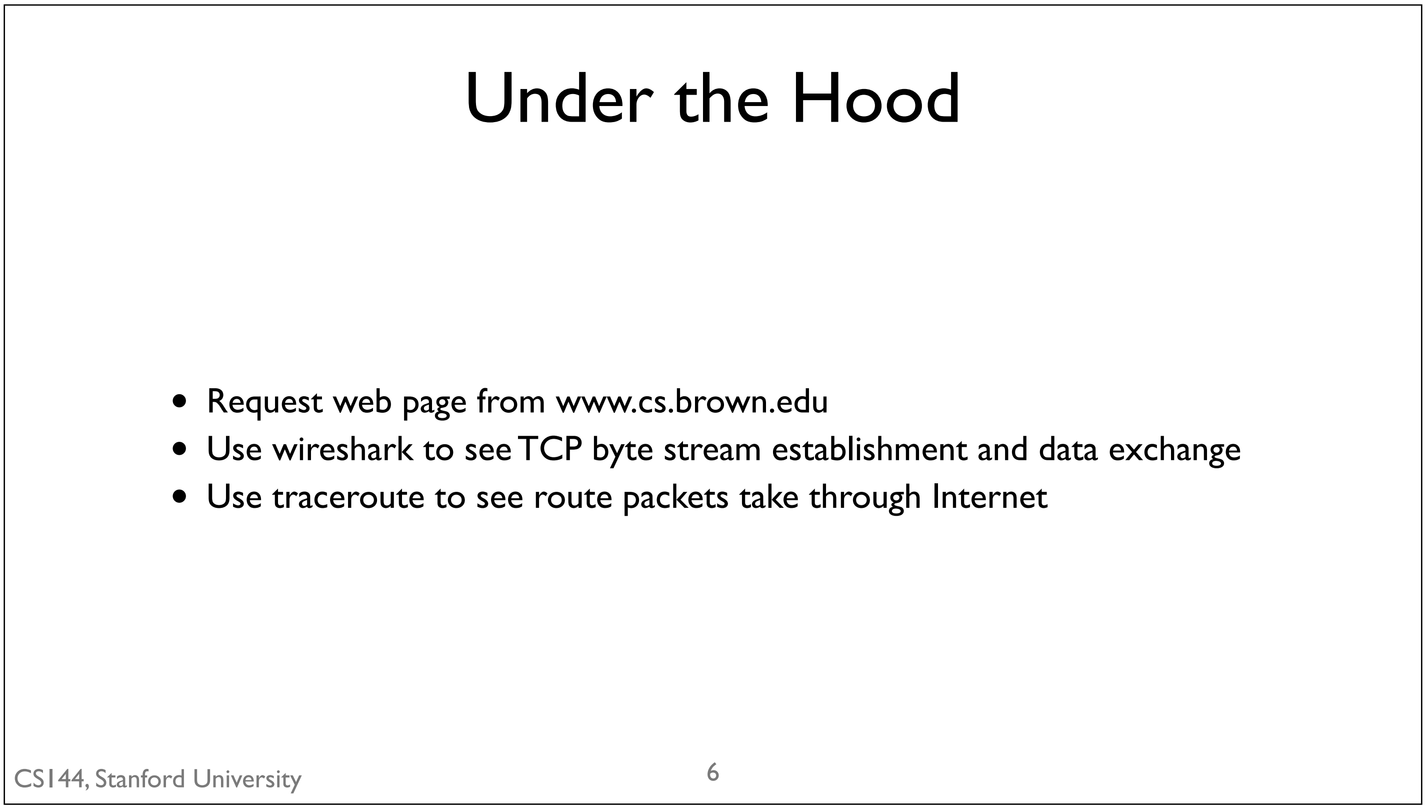
dest	link
default	1
pattern A	5
pattern B	2
pattern C	4
pattern D	2
pattern E	6
pattern F	1

default route:
least specific
match every IP add.

How does a router make this decision? It does so through something called a forwarding table, shown here on the right. A forwarding table consists of a set of IP address patterns and the link to send across for each pattern.

When a packet arrives, the router checks which forwarding table entry's pattern best matches the packet. It forwards the packet along that entry's link. Generally, "best" means the most specific match. I'll describe how this matching works in more detail in the video on longest prefix match. But in this simple example let's just consider the default route, the first entry in the table above. The default route is the least specific route -- it matches every IP address. If, when a packet arrives, there isn't a more specific route than the default route, the router will just use the default one.

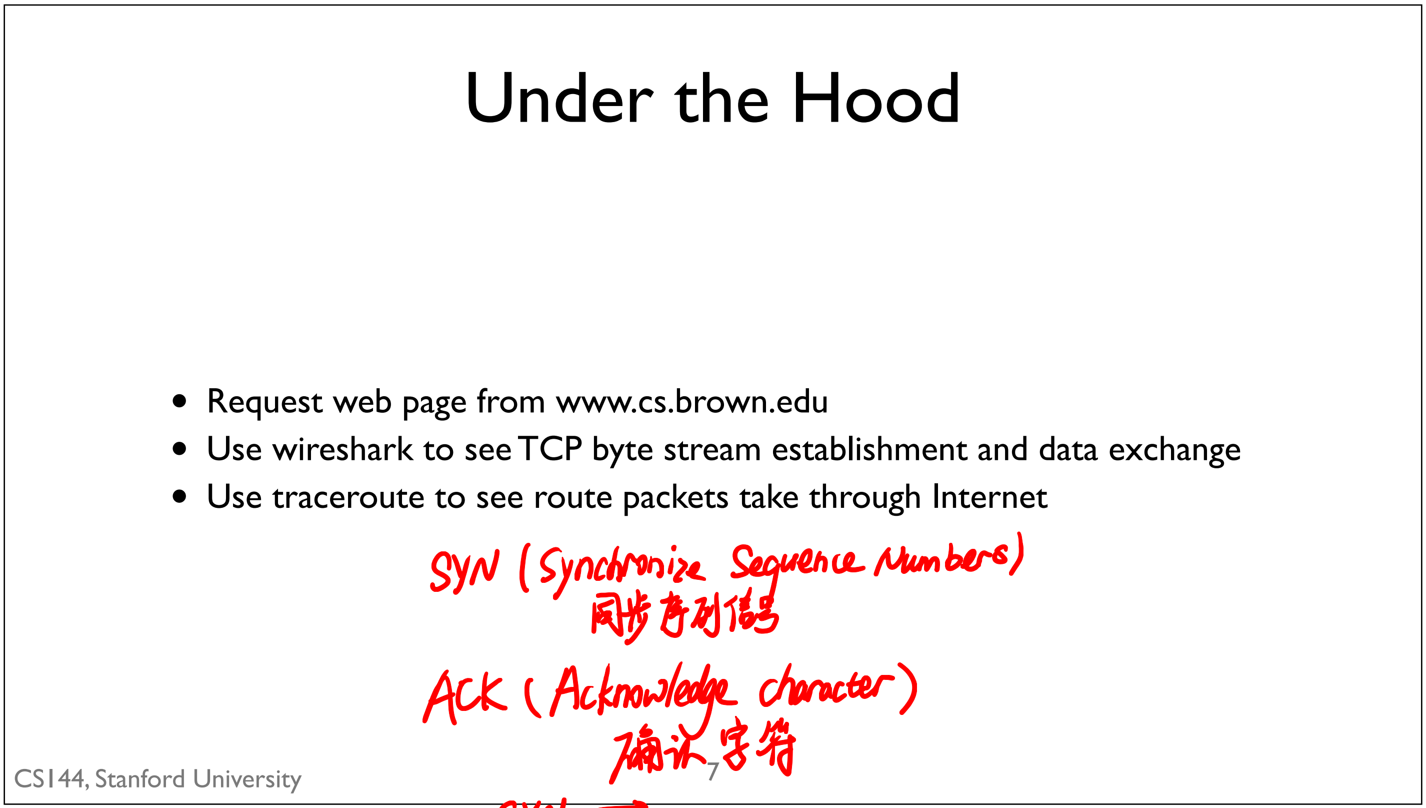
The default route is especially useful in edge networks. Say, for example, you're Stanford University and have a router connecting you to the larger Internet. Your router will have many specific routes for the IP addresses of Stanford's network: "send packets to the engineering school over this link", "send packets to the library over that link." But if the destination IP address isn't Stanford's, then the router should send it out to the larger Internet.



Under the Hood

- Request web page from www.cs.brown.edu
- Use wireshark to see TCP byte stream establishment and data exchange
- Use traceroute to see route packets take through Internet

So now let’s look at some IP packets in a real network. I’m going to request a web page from www.brown.edu and use a tool called Wireshark to show you all of the packets. We’ll see how my web browser opens a TCP connection to the Brown web server using a three way handshake of SYN, SYN-ACK, ACK, then starts issuing HTTP GET requests which the server responds to with data. Once we’ve seen the exchange of packets between my client and the Brown University web server, I’ll use another tool, called traceroute, to observe the path that these packets take through the Internet.



Under the Hood

- Request web page from www.cs.brown.edu
- Use wireshark to see TCP byte stream establishment and data exchange
- Use traceroute to see route packets take through Internet

SYN (Synchronize Sequence Numbers)
同步序列信号

ACK (Acknowledge character)
确认字符

SYN →
SYN, ACK ←
ACK →

Three-way handshake

数据段

So first I'll start up wireshark. Because my computer is using many network applications and sending lots of different packets, I'm going to tell wireshark to only display packets that are TCP segments to the Brown server using port 80. This way we'll only see the web traffic I'm generating. I'm also going to tell Wireshark to listen on en1, which is the name my Mac gives to my WiFi link layer. As you can see, I have many link layers available, but let's just look at en1 since that's how I'm connected to the Internet.

时间戳

Next I'll open my web browser and request the web page for Brown University's Computer Science department. This is where I went as an undergraduate and so I like to keep up with news on the department. You can see in wireshark that loading this page involved sending and receiving a lot of packets! Wireshark shows me the timestamp of each packet, the source IP address, the destination IP address, what protocol it uses, its length, and further information. Look at this first packet. It's from my computer, whose address is 192.168.0.106, to the Brown CS web server, whose address is 128.148.32.12. It's going to TCP port 80 -- the HyperText Transport Protocol port on the server, which we can see from the > http field in the Info column. The packets is SYN packets -- the first step of the three way handshake.

Look at the first three packets. The first is a SYN packet from my computer to the web server. The second is a SYN-ACK packet from the web server back to my computer. The third is an ACK from my computer back to the web server. This is the three way handshake! Now the two computers can exchange data, and you can see that the first data packet is an HTTP request -- my computer sends a GET request to the web server. The response to this GET request is three packets -- wireshark shows the response when it receives the third one, shown in the line whose info is HTTP/1.1 200 OK. So here we can see how my requesting a web page from Brown's Computer Science server creates a TCP connection through the three IP packets for the 3-way handshake, then more packets for the HTTP request and response.

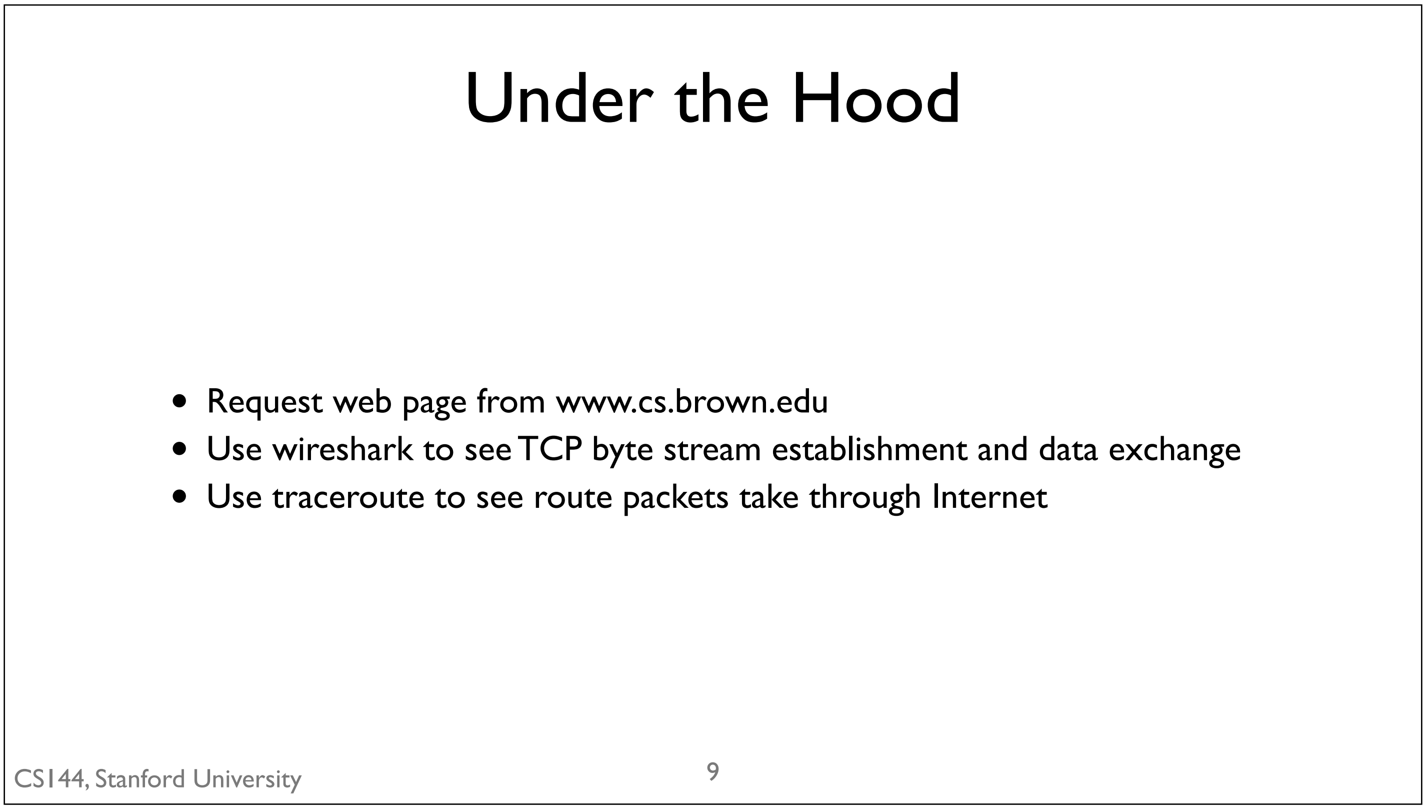
Under the Hood

- Request web page from www.cs.brown.edu
- Use wireshark to see TCP byte stream establishment and data exchange
- Use traceroute to see route packets take through Internet

This is how the network looks like to the end hosts, the computers, as they exchange packets at the network layer. But what does it look like inside the network layer? What hops do these packets take? To see this, I'm going to use a second tool, traceroute. Traceroute shows you the hops that packets to a destination take. So we can type traceroute www.cs.brown.edu to see the path through the Internet. I'll add the -w flag, which specifies a timeout, with a timeout of 1 second.

The first hop the packets take is to my wireless router, whose IP address is 192.168.0.1. As you can see from the next hop, I'm at home -- I have a cable modem and my Internet provider is Astound. After this packets take another hop to a router with IP address 74.14.0.3. The hop after that is a router in San Francisco, California, then several routers in San Jose, sjc for above.net and sanjose1 for level3.net. After sanjose1.level3.net, the packets are across the United States in New York! They go through a series of routers in New York -- ebr, csw, ebr, then, on hop 13, to Boston. Boston is very close to Providence, where Brown is. After oshean.org that we see three stars -- this means there's a router that won't tell traceroute about itself. The stars are traceroute's way to show it waited for a reply but the replied timed out. On hop 20 we see a router in Brown's CS department. After that, everything is hidden -- Brown's CS department doesn't want you to be able to see what the inside of its network looks like.

Because we couldn't see the path end with Brown's web server, let's try another one: the Computer Science and Artificial Intelligence Lab (CSAIL) at MIT. We can see that packets take the same path to Boston, until hop 15. The path to Brown goes to oshean at hop 15, while the path to MIT continues in level3's network. On the path to www.csail.mit.edu only two routers are hidden, the 13th and 19th hops. We can see that www.csail.mit.edu is also named akron.csail.mit.edu and, after 22 hops, packets from my computer reach MIT's web server. Look at the time values -- the time for my packet to reach the MIT web server and its response to return to me -- there and back, the round-trip time -- is under 90 milliseconds, or less than an eye blink.



Under the Hood

- Request web page from www.cs.brown.edu
- Use wireshark to see TCP byte stream establishment and data exchange
- Use traceroute to see route packets take through Internet

We’ve now seen the life of a packet, starting as an application-level client web request and taking nearly 20 hops through the Internet to reach its destination. For me, this is one of the best things about teaching this course. Everything we present is something that you and I interact with every day -- even just in the space of watching one video! It’s easy to see the principles and ideas in practice, and with a few simple tools you can inspect the Internet in operation, in real time!