How Does It Work?

An overview of how the web allows communication between devices.

WE'LL COVER THE FOLLOWING

- Switches to connect devices
- Routers
- Data packets
- TCP
- HTTP & HTTPS
- Ports

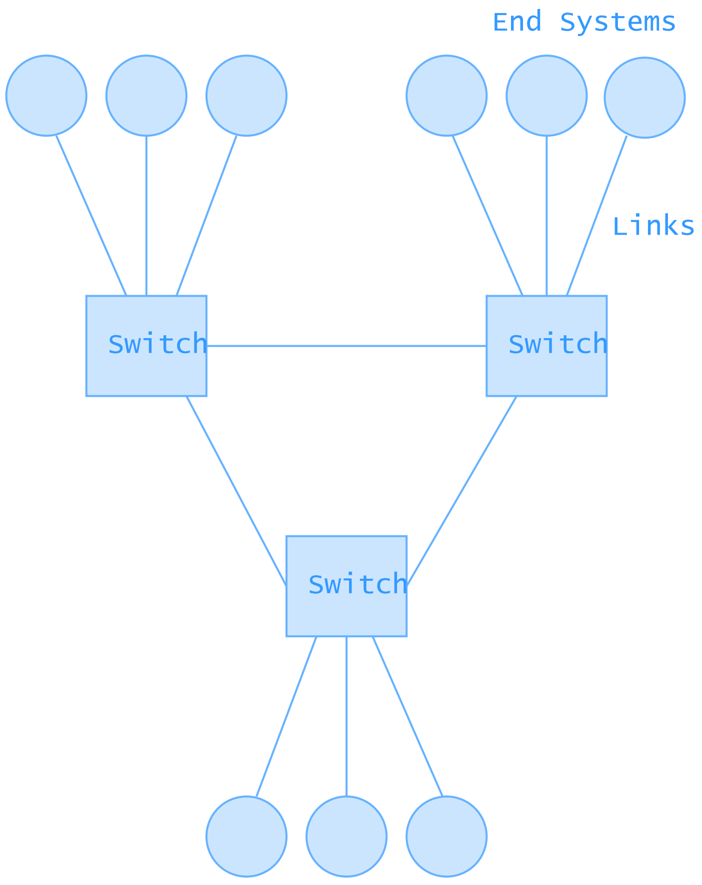
In the previous lesson, we talked about clients and servers and that the Internet allows them to communicate with one another. Now, we will look into exactly how this communication happens.

Switches to connect devices

In essence, the communication is pretty intuitive. Clients send messages to servers requesting data, and servers respond with the required data, but how is this data transferred? The answer requires defining the structure of the Internet first.

The Internet comprises of devices known as **switches** that facilitate the connection of each device to every other device on the network. The devices themselves are referred to as **end-systems**, and which is essentially just a fancy term for the computer you're using to access this webpage right now!

End-systems are connected to switches through **links** and all of the switches are, in turn, connected to each other, thus ensuring that every end-system on the Internet is implicitly connected to every other end-system.

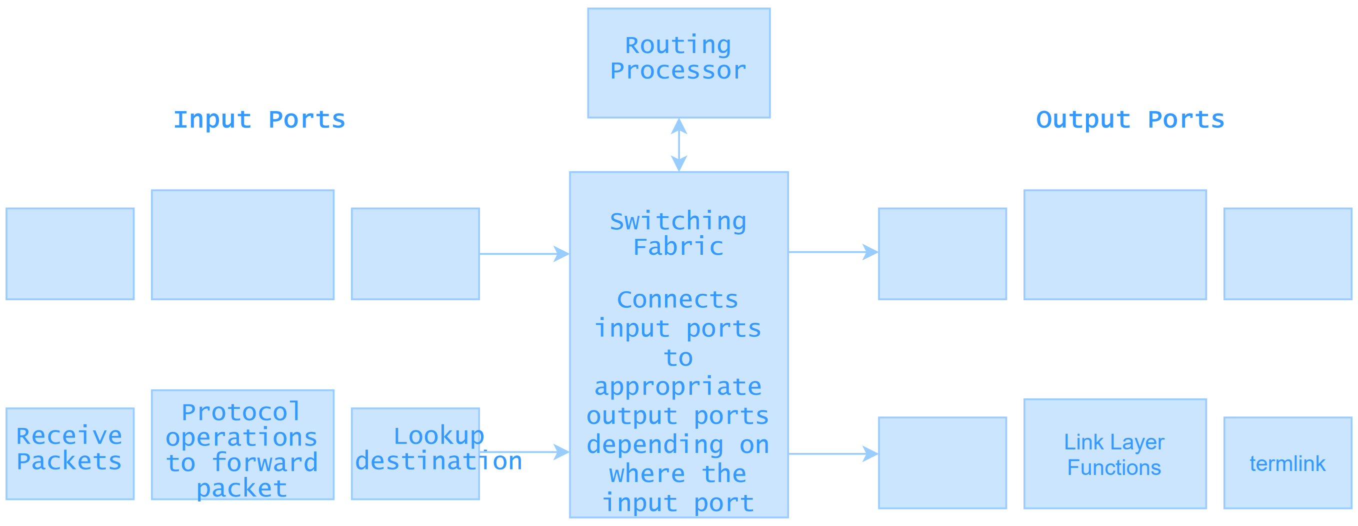


The Structure of the Web

In addition to connecting end-systems to one another, switches facilitate the communication between any two end-systems by forwarding packets along the path that they know exists between the packet source and destination. So basically, switches store pre-determined paths between end-systems and forwards packets among them.

Routers

A commonly heard term when it comes to networks is routers. *Routers* have the same function as switches per se in that they also connect end systems to the rest of the web. However, routers are actually very different from switches since they have the additional capability of allowing lookups for destination addresses and determining the shortest or the least busy path from the source of a packet to its destination. Routers are, therefore, a more powerful version of switches and the Internet comprises of a mix of both routers and switches that facilitate the transfer of data between end-systems.


Routing Processor
Input Ports
Output Ports
Switching Fabric
Connects input ports to appropriate output ports depending on where the input port
Receive Packets
Protocol operations to forward packet
Lookup destination
Link Layer Functions
termlink

Anatomy of a Router

Data packets

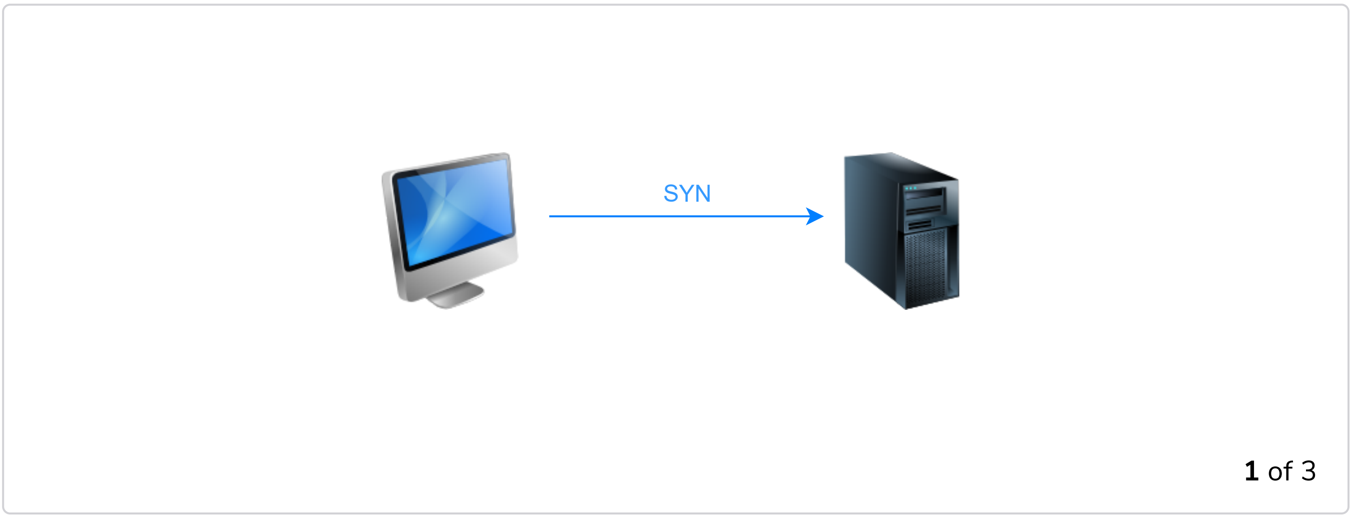
Now that we have established how the Internet is structured to ensure connectivity let’s talk about how data is actually transported across the network. This is done by dividing the data, which is just a set of bits, that needs to be transferred into several smaller chunks of bits known as **packets** and then sending each packet to its destination independently. The reason for this is that you cannot always send large amounts of data in a single packet because it is highly likely that data in large packets get corrupted on the path from its source to its destination due to a bit flip during transmissions. It is, therefore, more efficient and reliable to send multiple smaller packets.

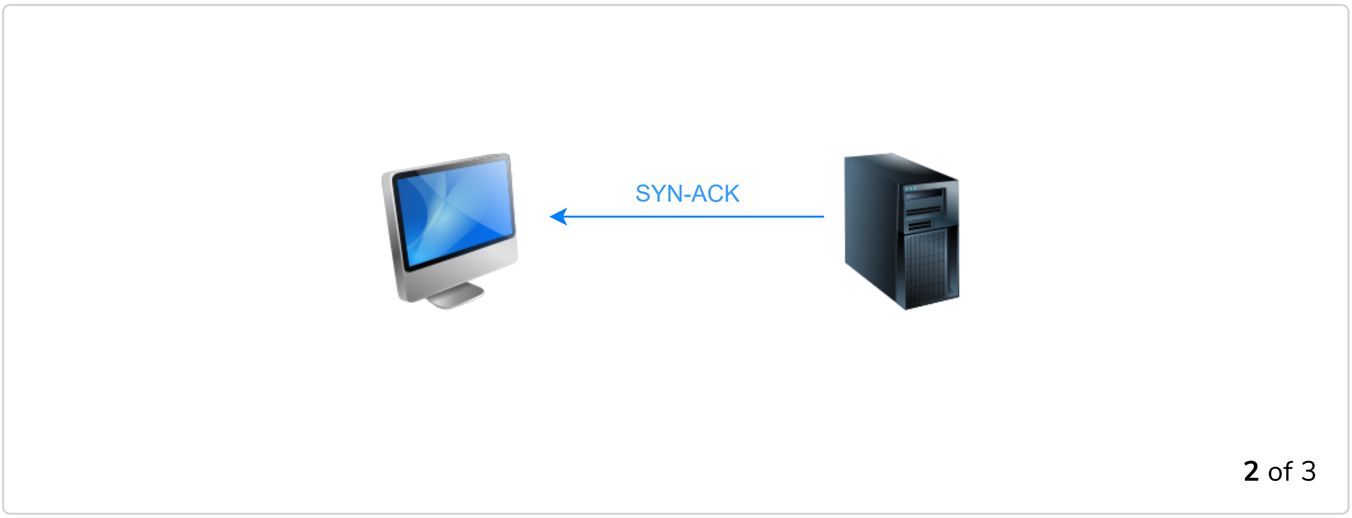
TCP

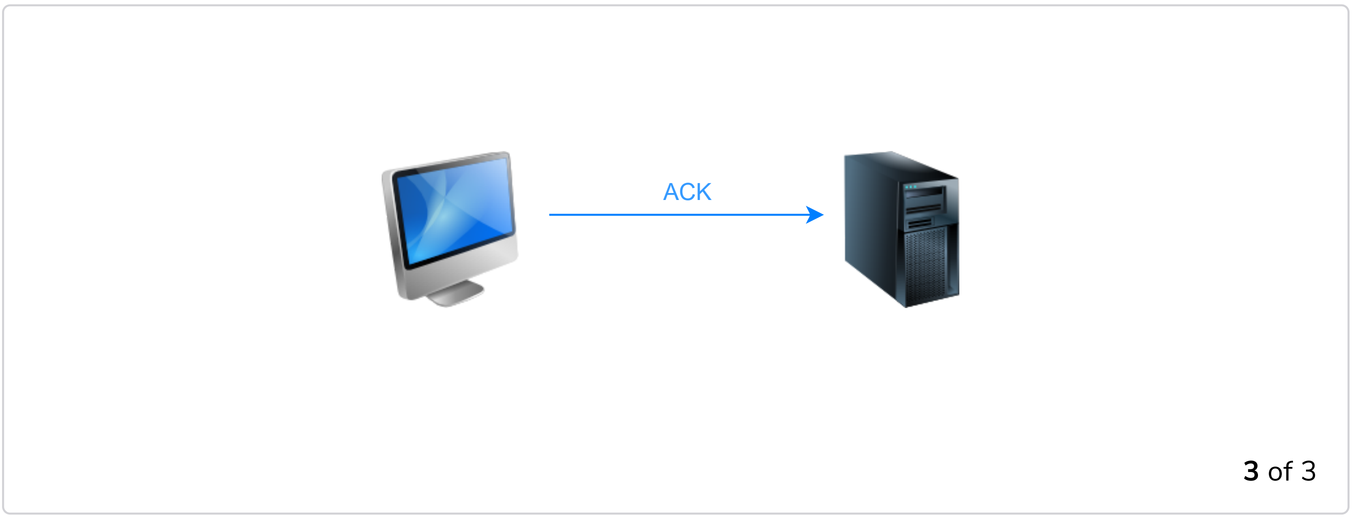
Since it is important for clients and servers across the network to be able to understand the same language, there are certain protocols that dictate the communication between devices on the Internet. The primarily used protocol for communication between a web application and a browser is referred to as the Transmission Control Protocol (TCP). TCP is a *transport layer* protocol that takes the responsibility of transmitting data and ensures reliable data transfer between clients and servers across the web. The way TCP does this is by adding additional information to data packets that allow for packet authentication and by allowing the exchange of acknowledgment messages between the client and server to confirm data transmissions.

The TCP protocol starts with a 3-way handshake. The handshake allows both ends (server and client) to initiate and maintain several TCP connections at

once. Have a look at the simplified diagram of the TCP 3-way handshake below.







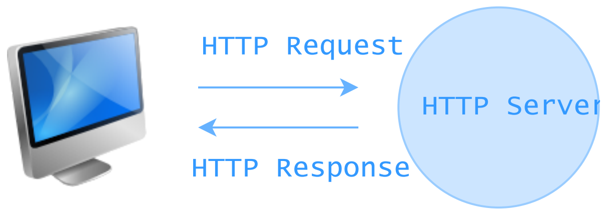
HTTP & HTTPS

We have just learned that clients and servers communicate with each other by initiating TCP connections and then sending messages to one another. Now, we will look into exactly how these messages are structured.

HyperText Transfer Protocol, commonly referred to as HTTP, is an application layer protocol that dictates how the messages a client and server exchange on the web are structured as well as how they are exchanged. The client program and server program talk to each other by exchanging HTTP messages, and the benefit of HTTP is that it ensures messages are being delivered intact and in time.

This may seem ambiguous, but the high-level idea is that HTTP is built on top of TCP and creating an HTTP server for your web application basically just means that you are creating a server that clients create TCP connections with for reliable data transfer. In simple terms, HTTP is the means through which you can make sure your application is using TCP to transmit messages from clients to the server and vice versa. So basically, when you enter a URL in your browser, what actually happens is that an HTTP command gets sent to the server hosting the application to fetch and transmit the requested web page through TCP.

The more common structure of URLs is **https**://www.educative.io/. So, what does HTTPS stand for? HTTPS is an acronym for HyperText Transfer Protocol Secure, and it is basically just the secure version of HTTP. What this means is that communications between the browser and the hosting server are encrypted so that no third parties on the network can access information that is not intended to be shared.



Communication Between a Browser and Web Application

Ports

So far, we have discussed both the transport layer protocol as well as the application layer protocol that ensures efficient communication between end-systems on the web. But, where exactly do the messages these protocols allow end-systems to go? **Ports** are the endpoints of the communication between clients and servers. That is to say: ports are where messages from the network clients and servers. That is to say, ports are where messages from the network arrive on an end-system. We briefly discussed sockets earlier and talked about

how they are the gateways to processes; sockets are opened on ports in order to allow processes to send and receive messages. Ports are designated by numbers, and all ports below 1024 are associated with a specific protocol each by default. The port number for HTTP, for instance, is 80, and what this means is that any messages you send or receive on the web come in to and leave your machine on a socket at port 80. Ports above 1024 are **open ports** available to programmers to use for any process they want to communicate with a network. They can build sockets on these ports and define the structure and type of messages this socket can cater to through *socket programming*. Socket programming is an aspect of Computer Networks that is beyond the scope of this discussion, but it is a highly useful skill to get under your belt and should definitely be looked into if you are interested in the field.

1 Which of the following is not a difference between switches and routers?

COMPLETED 0%

1 of 5

Now that we’ve given a high-level overview of the communication that occurs on the web starting from the protocols and devices that allow systems on the Internet to communicate all the way to the exact point at which the messages arrive, it is safe to assume that we understand how systems on the web are placed. In the next lesson, we will move on to look into how these systems actually find one another to communicate with to bring all the pieces together.