



# Routing, Packets, and Protocols

# Recap of what we've learned so far

All devices on the Internet have a unique IP Address that can be for identification, so that information can be transmitted to them.

In the past, we've used IPv4, but since that can only support ~4 billion devices, we're upgrading to IPv6. *That* can support ~340 **undecillion** devices!

Since IP Addresses are not very human-readable, we need a system to translate between human-readable names and machine-readable IP Addresses. This is the Domain Name System, or DNS!




# Recap of what we've learned so far

All devices on the Internet have a unique IP Address that can be for identification, so that information can be transmitted to them.

In the past, we've used IPv4, but since that can only support ~4 billion devices, we're upgrading to IPv6. *That* can support ~340 **undecillion** devices!

Since IP Addresses are not very human-readable, we need a system to translate between human-readable names and machine-readable IP Addresses. This is the Domain Name System, or DNS!

We now know how to find the destination, but what does the journey look like?

A decorative graphic in the bottom right corner consisting of several overlapping green triangles and rectangles in various shades of green, creating a modern, abstract design.

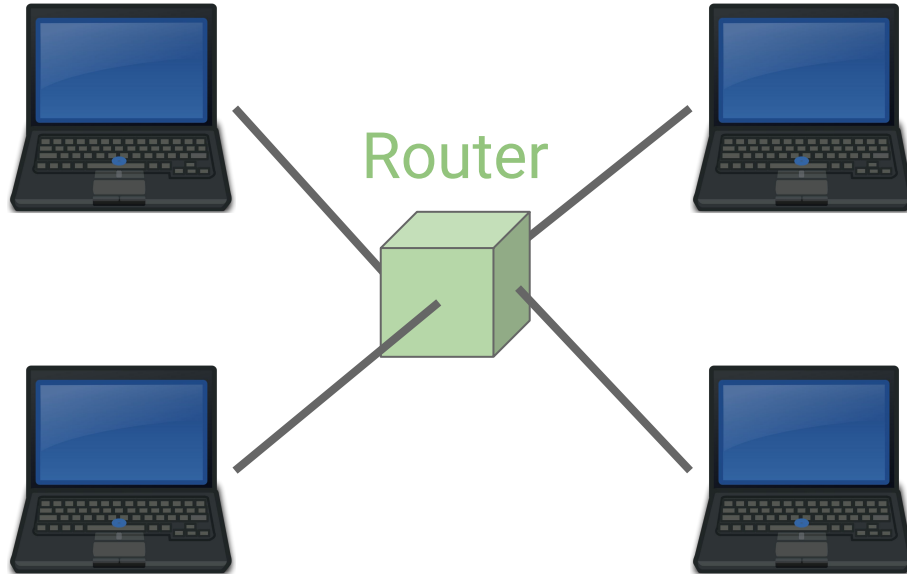
# Routing

**Routing** is the process of sending data between two computers on the internet. The data is sent through **routers** that determine the **route**.



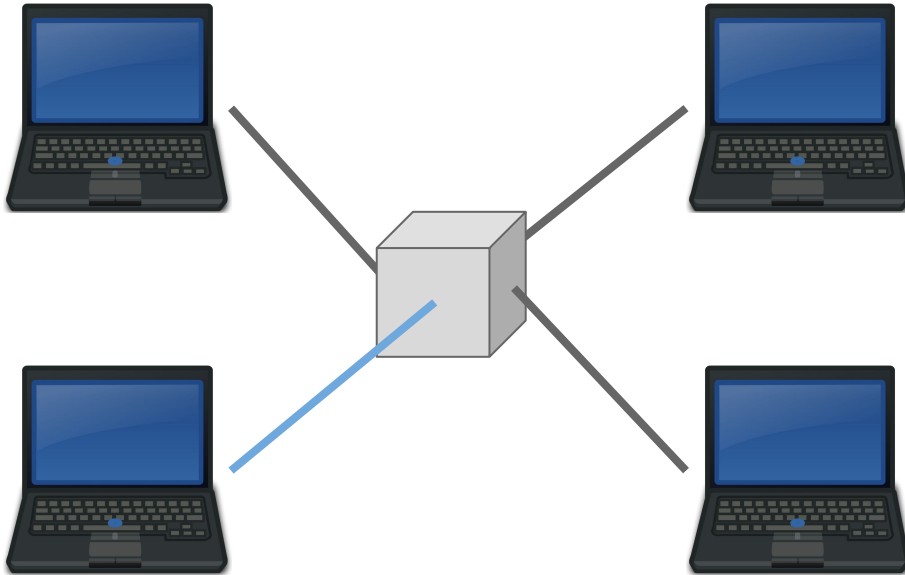
# Routing

**Routing** is the process of sending data between two computers on the internet. The data is sent through **routers** that determine the **route**.



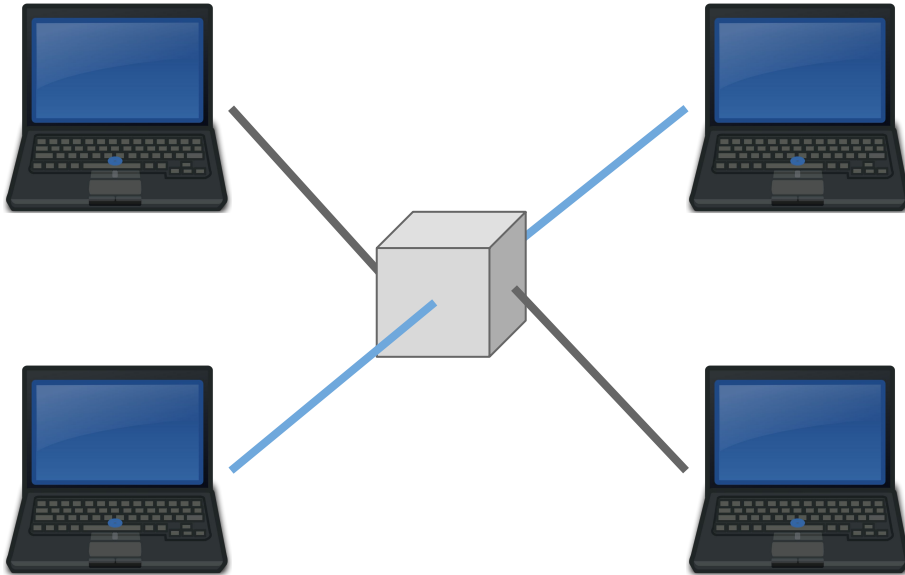
# Routing

One computer can send a message to another computer through a simple **router**!



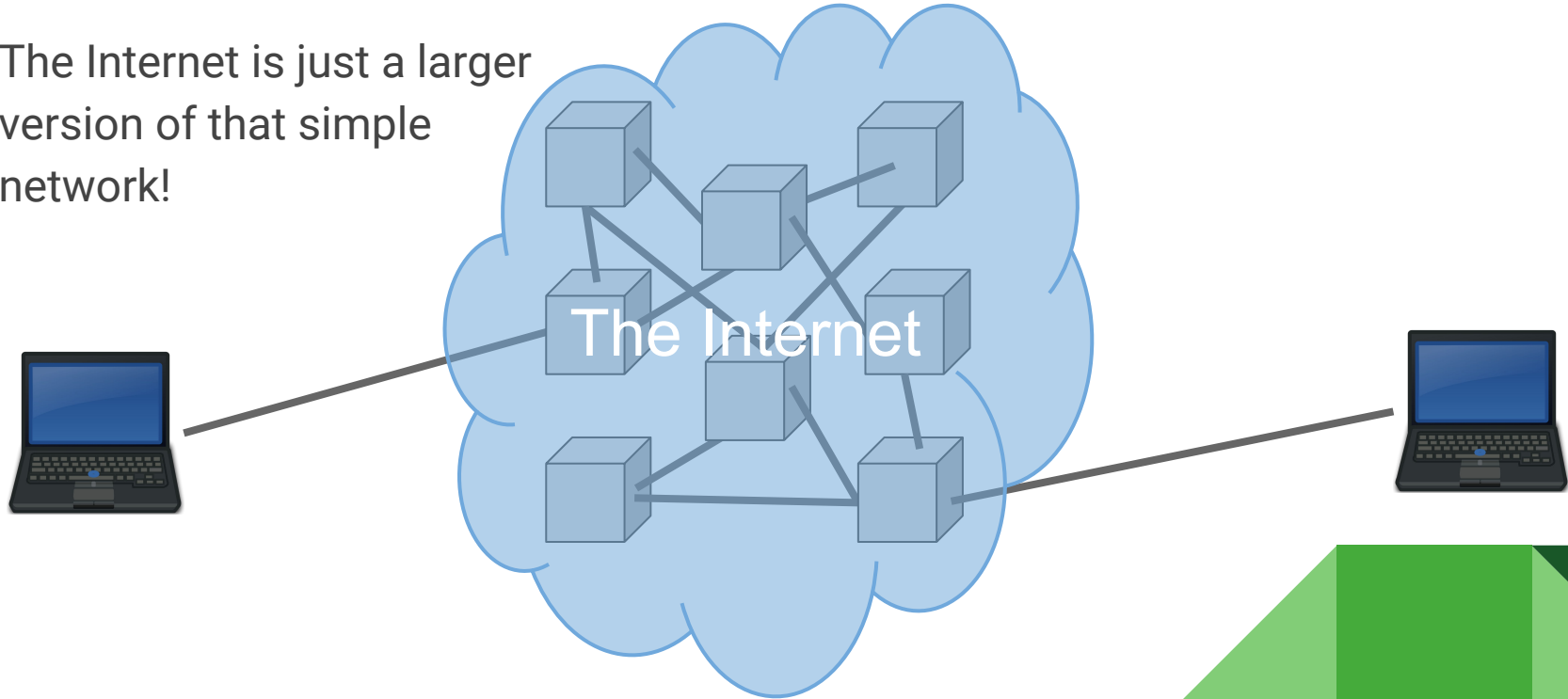
# Routing

One computer can send a message to another computer through a simple **router**!



# Routing

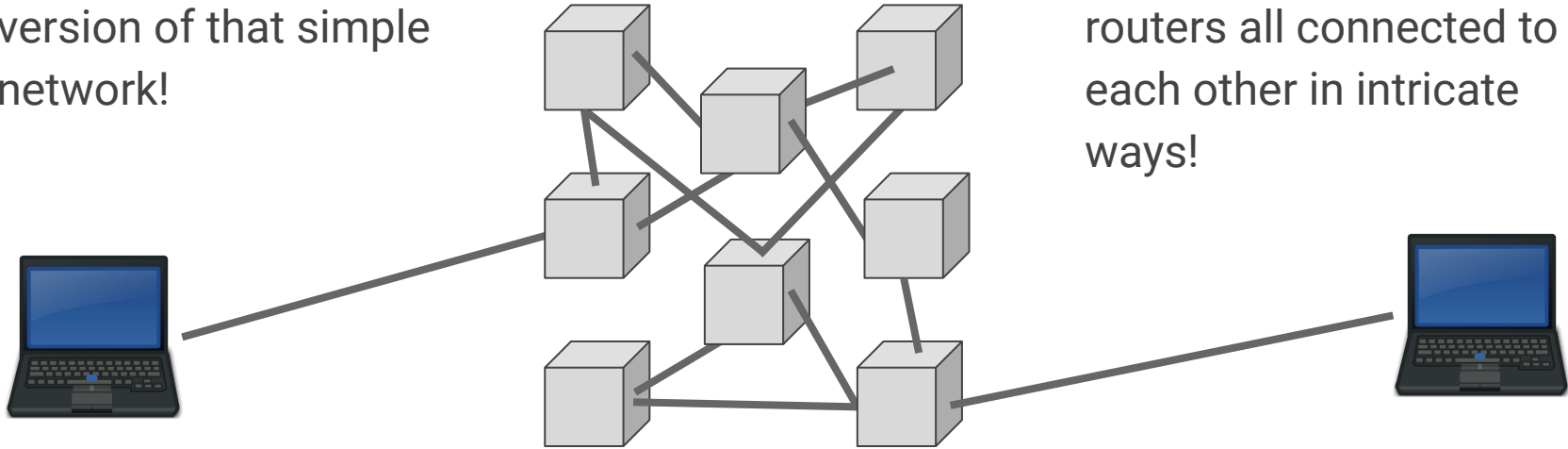
The Internet is just a larger version of that simple network!





# Routing

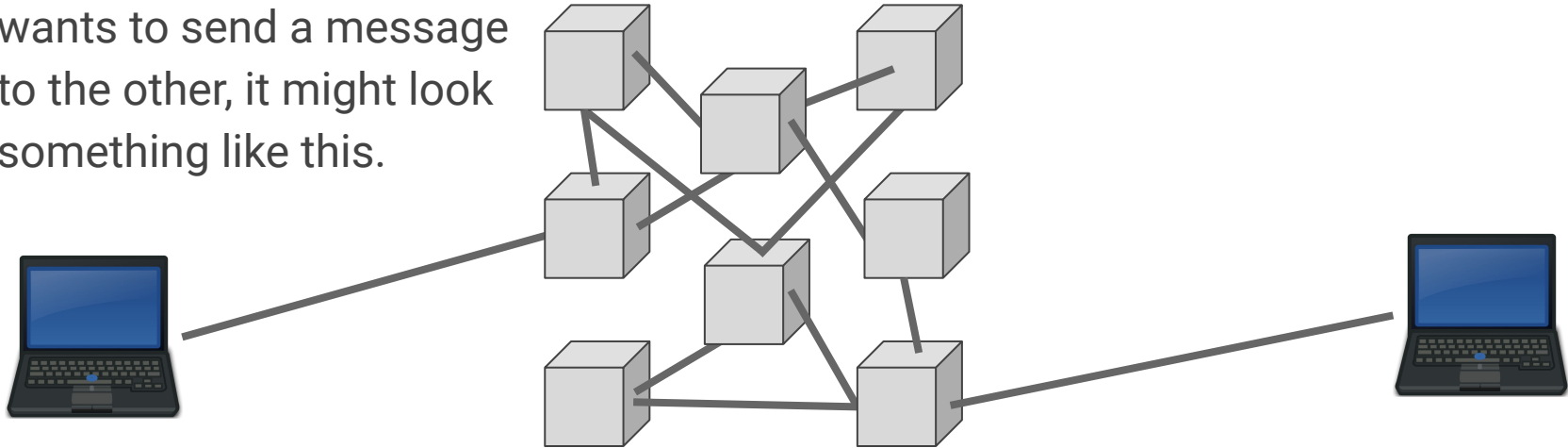
The Internet is just a larger version of that simple network!



It's just a **whole bunch** of routers all connected to each other in intricate ways!

# Routing

If one of these computers wants to send a message to the other, it might look something like this.

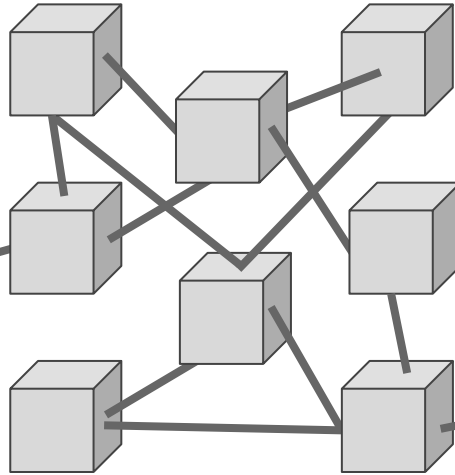


# Routing

The computer on the left writes up a message, then marks it with a **to** and **from** address, just like with postal mail!



2.2.2.2



9.9.9.9



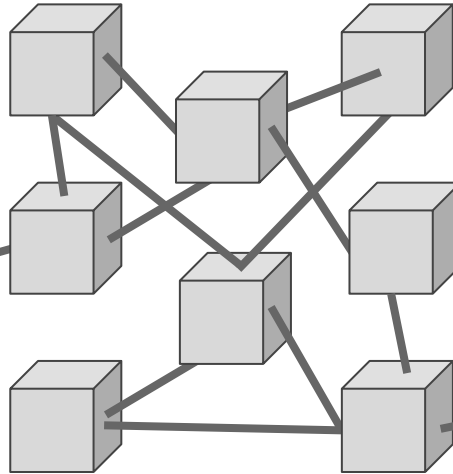
# Routing

The computer on the left writes up a message, then marks it with a **to** and **from** address, just like with postal mail!



2.2.2.2

**From:** 2.2.2.2  
**To:** 9.9.9.9  
**Message:** "hey"



9.9.9.9



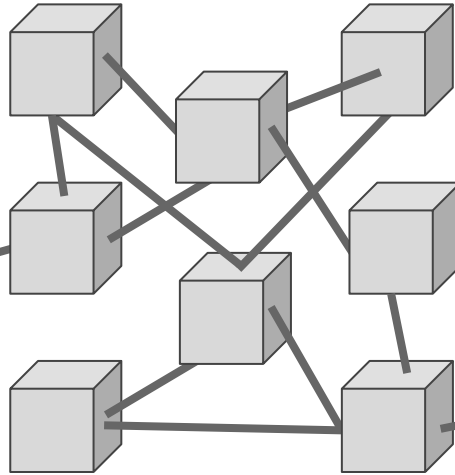
# Routing

The computer then forwards that message to the router with the lowest **cost**.



2.2.2.2

**From:** 2.2.2.2  
**To:** 9.9.9.9  
**Message:** "hey"



9.9.9.9



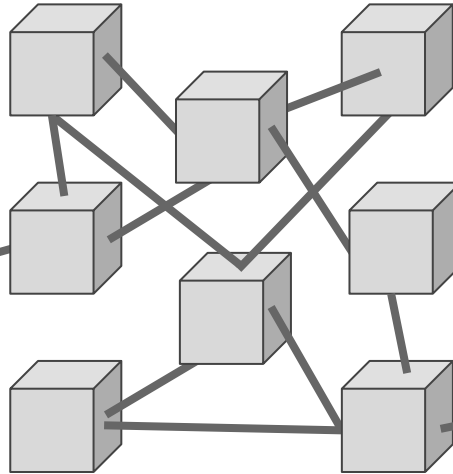
# Routing

The computer then forwards that message to the router with the lowest **cost**.



2.2.2.2

**From:** 2.2.2.2  
**To:** 9.9.9.9  
**Message:** "hey"



"Cost" can refer to a variety of different factors, including:

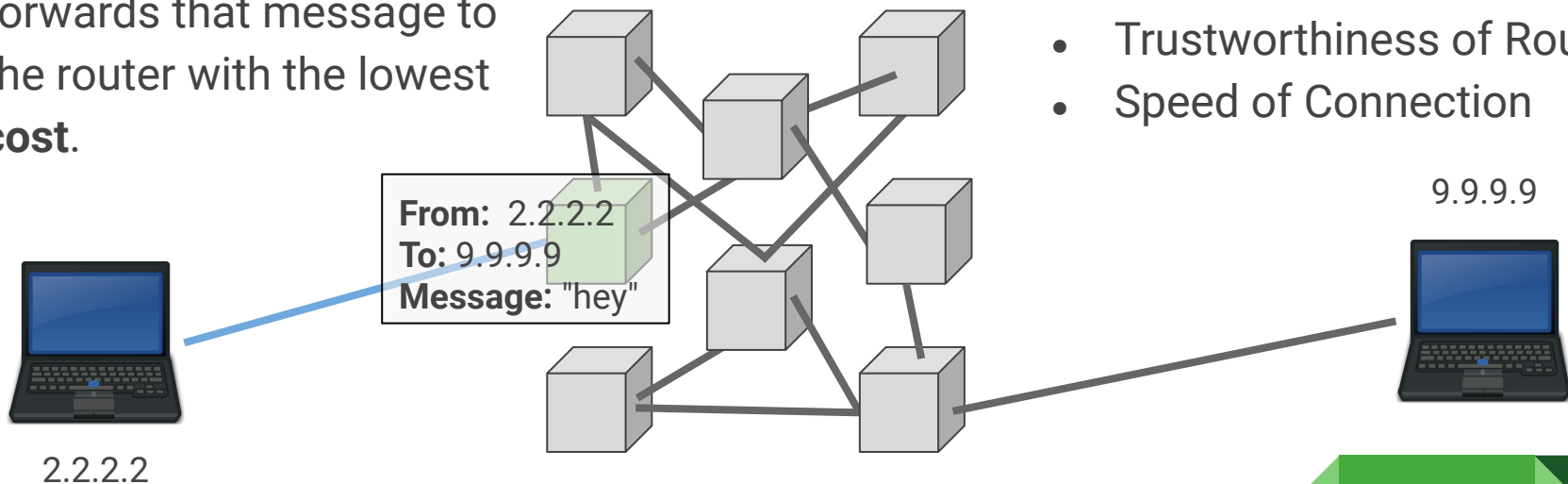
- Closeness of IP Address
- Trustworthiness of Router
- Speed of Connection

9.9.9.9



# Routing

The computer then forwards that message to the router with the lowest **cost**.

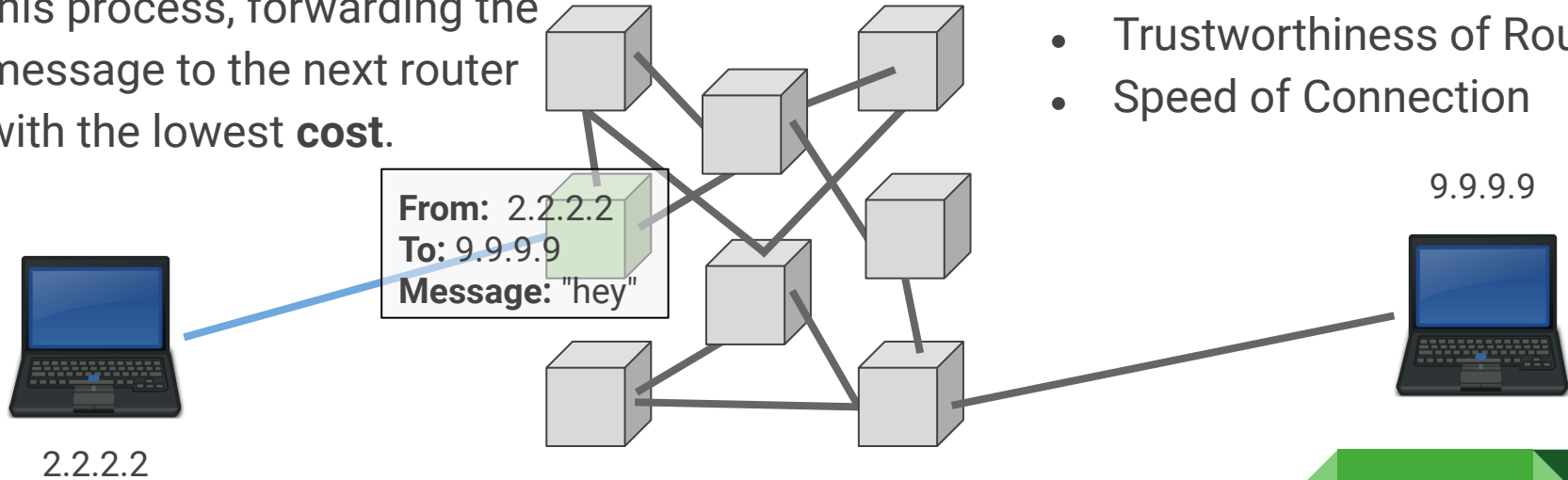


"Cost" can refer to a variety of different factors, including:

- Closeness of IP Address
- Trustworthiness of Router
- Speed of Connection

# Routing

That router then repeats this process, forwarding the message to the next router with the lowest **cost**.



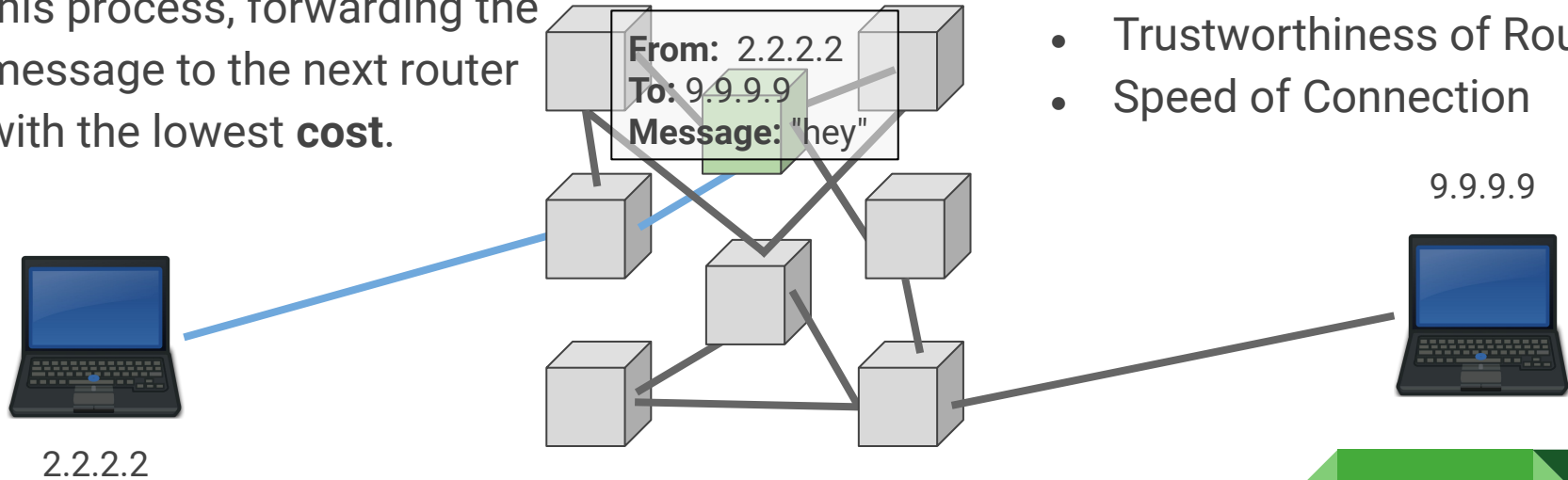
"Cost" can refer to a variety of different factors, including:

- Closeness of IP Address
- Trustworthiness of Router
- Speed of Connection



# Routing

That router then repeats this process, forwarding the message to the next router with the lowest **cost**.

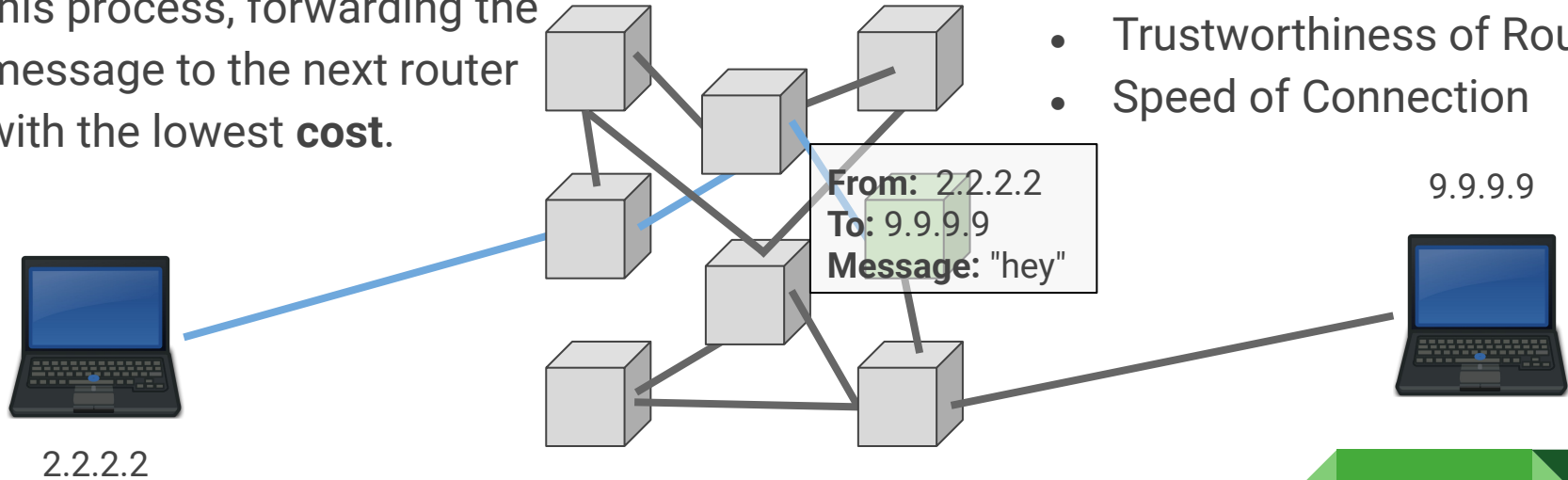


"Cost" can refer to a variety of different factors, including:

- Closeness of IP Address
- Trustworthiness of Router
- Speed of Connection

# Routing

That router then repeats this process, forwarding the message to the next router with the lowest **cost**.

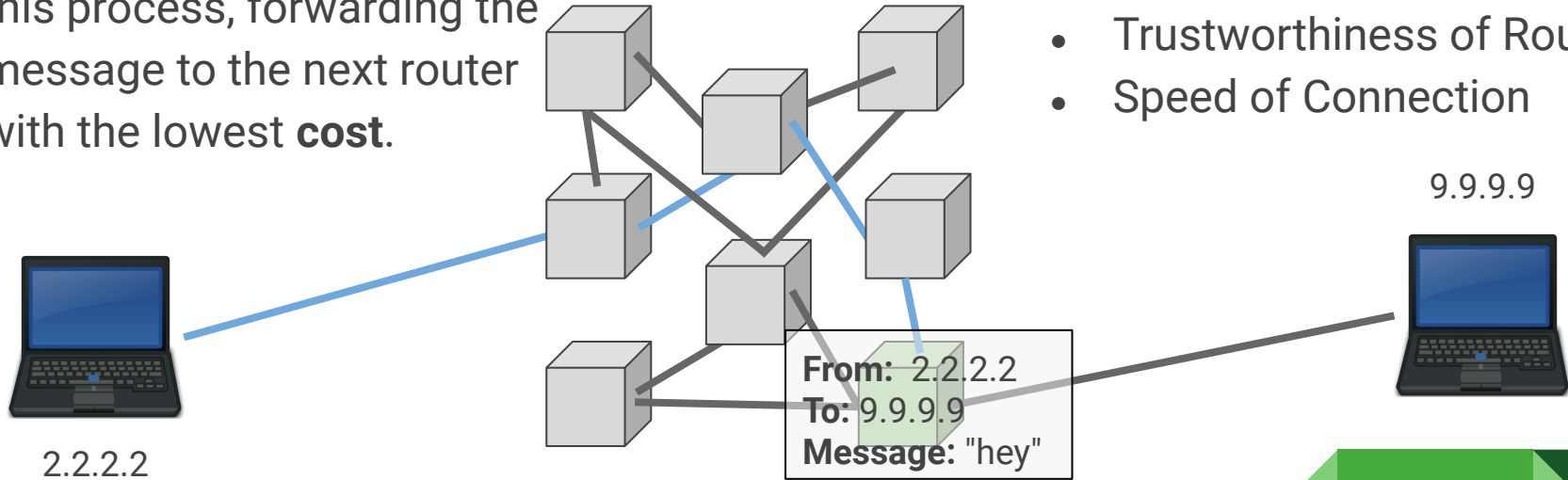


"Cost" can refer to a variety of different factors, including:

- Closeness of IP Address
- Trustworthiness of Router
- Speed of Connection

# Routing

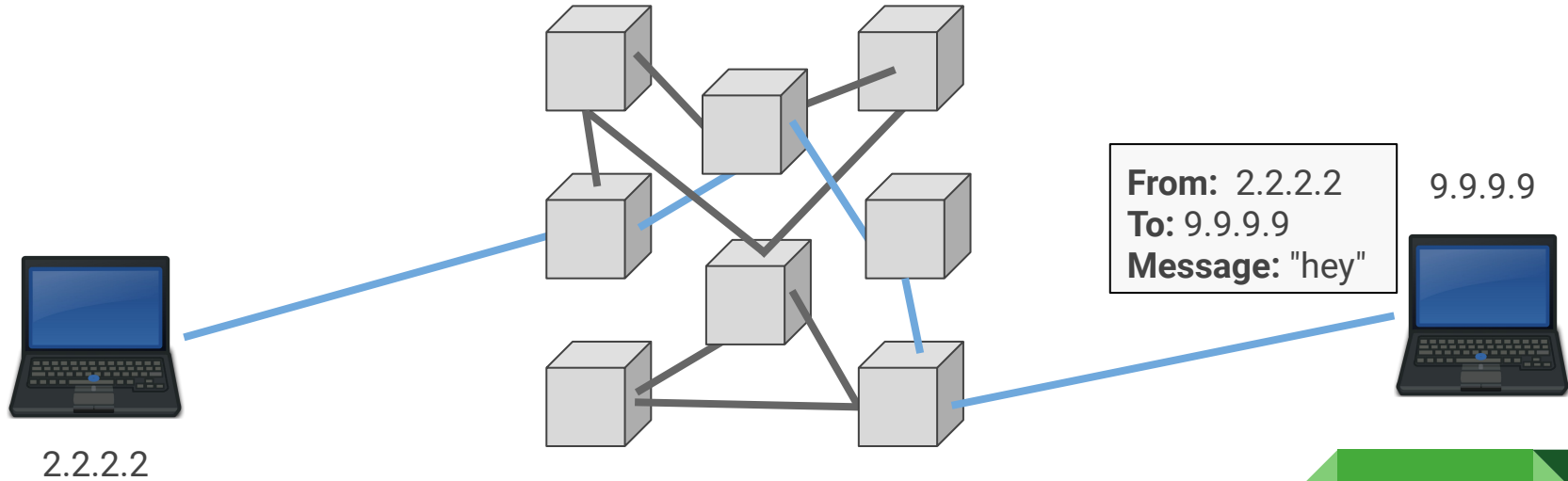
That router then repeats this process, forwarding the message to the next router with the lowest **cost**.



"Cost" can refer to a variety of different factors, including:

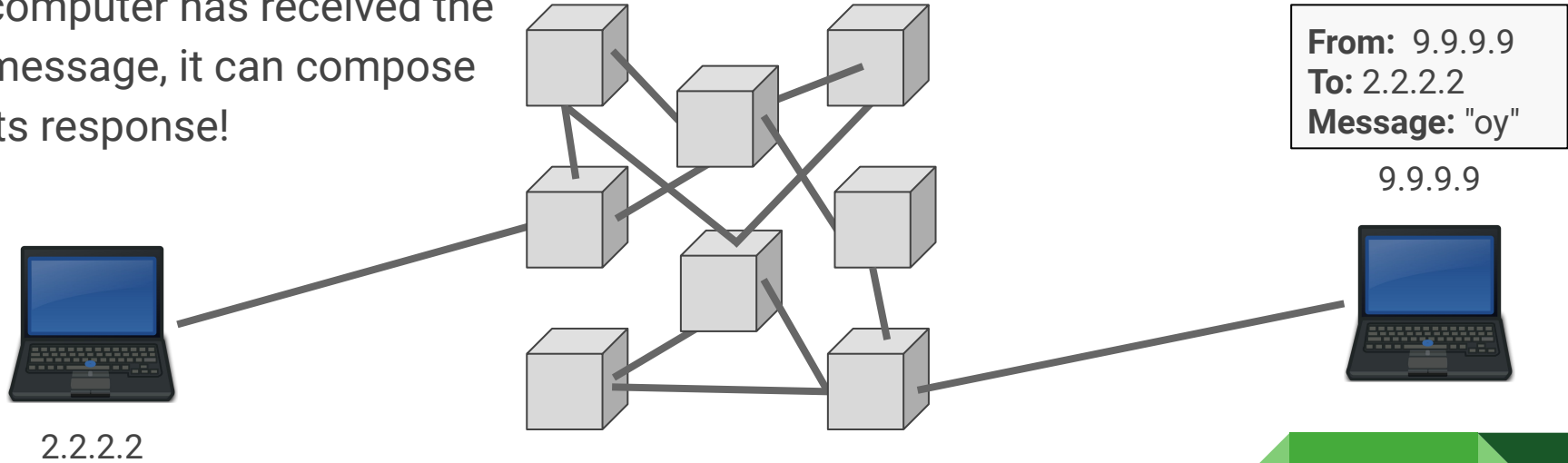
- Closeness of IP Address
- Trustworthiness of Router
- Speed of Connection

# Routing



# Routing

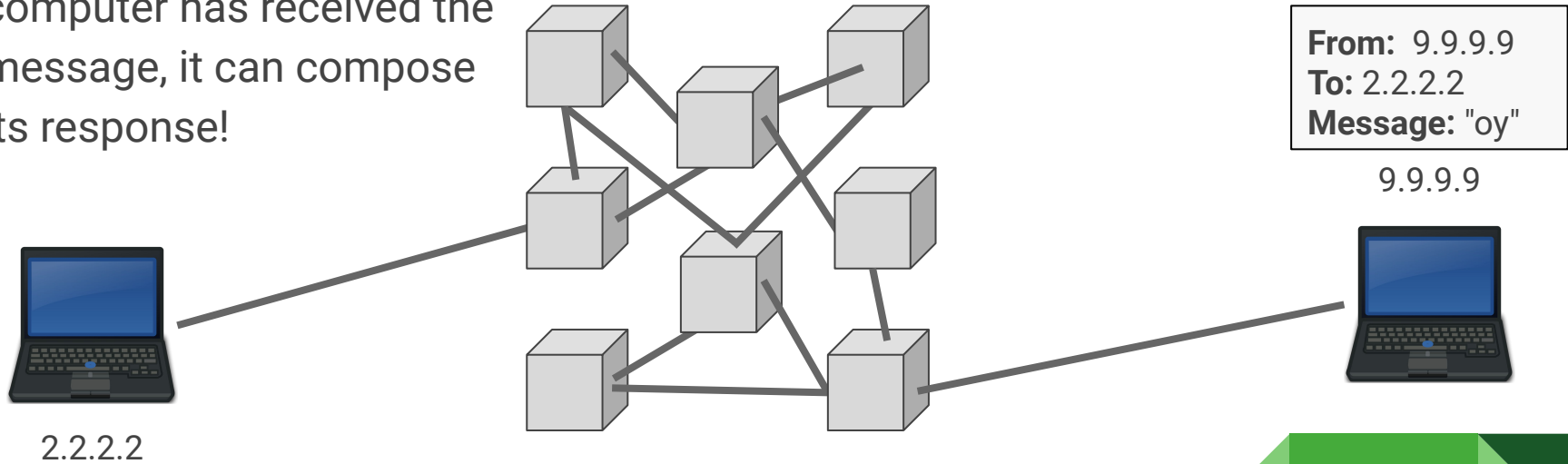
Now that the other computer has received the message, it can compose its response!



# Routing

Now that the other computer has received the message, it can compose its response!

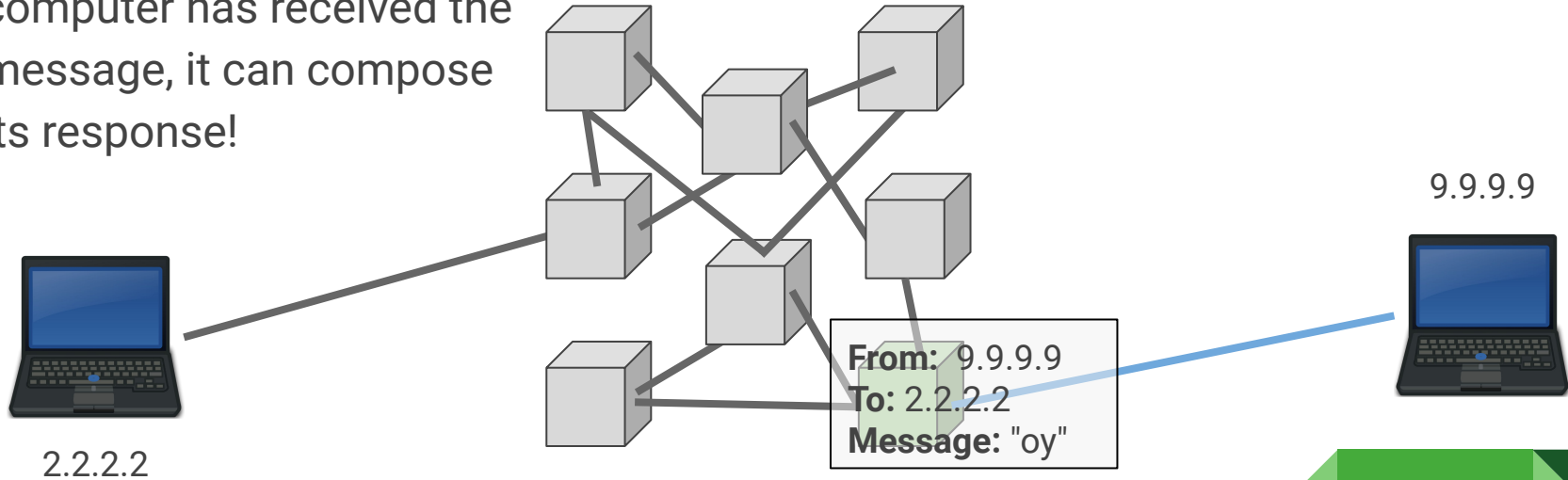
That computer then sends its message back using the same process!



# Routing

Now that the other computer has received the message, it can compose its response!

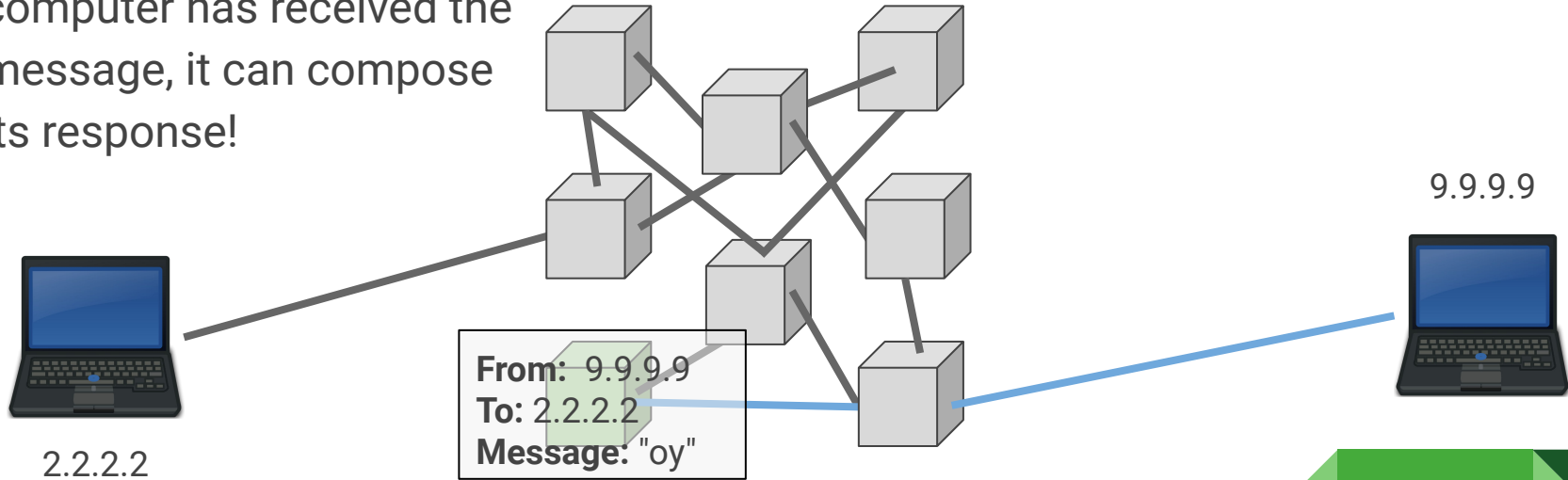
That computer then sends its message back using the same process!



# Routing

Now that the other computer has received the message, it can compose its response!

That computer then sends its message back using the same process!

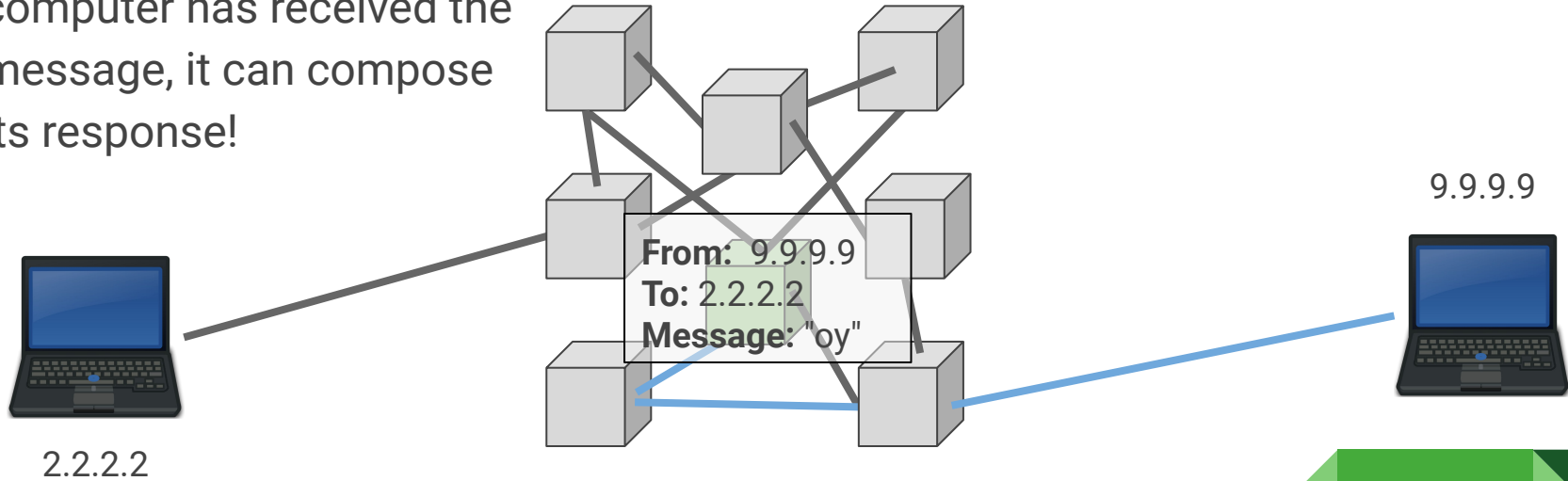




# Routing

Now that the other computer has received the message, it can compose its response!

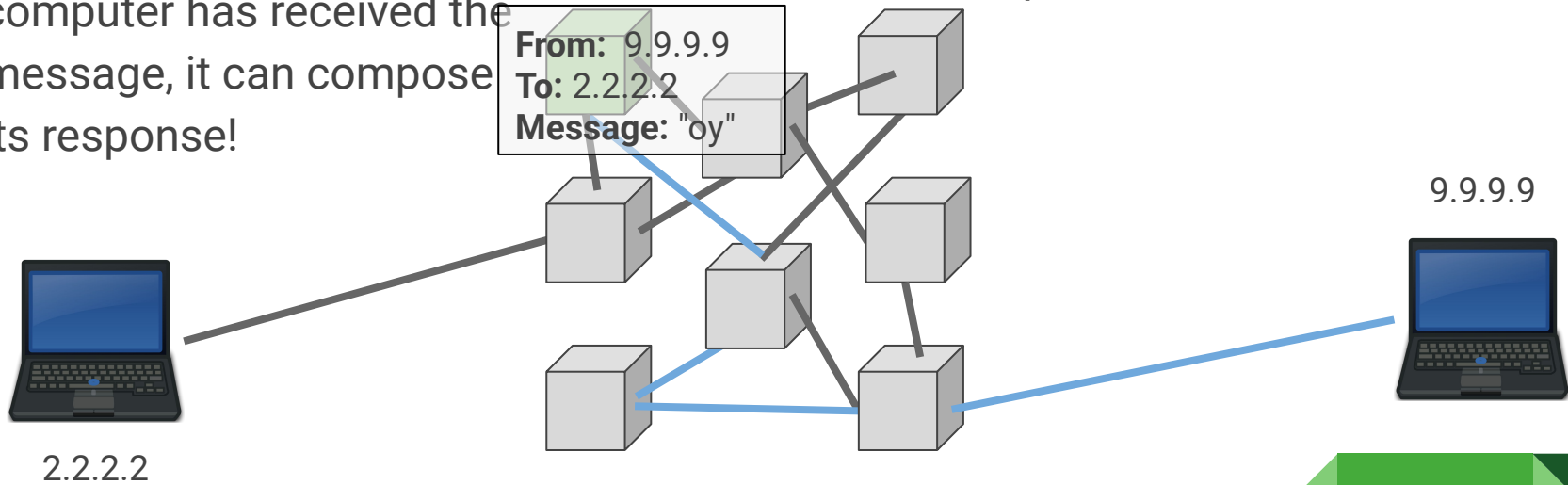
That computer then sends its message back using the same process!



# Routing

Now that the other computer has received the message, it can compose its response!

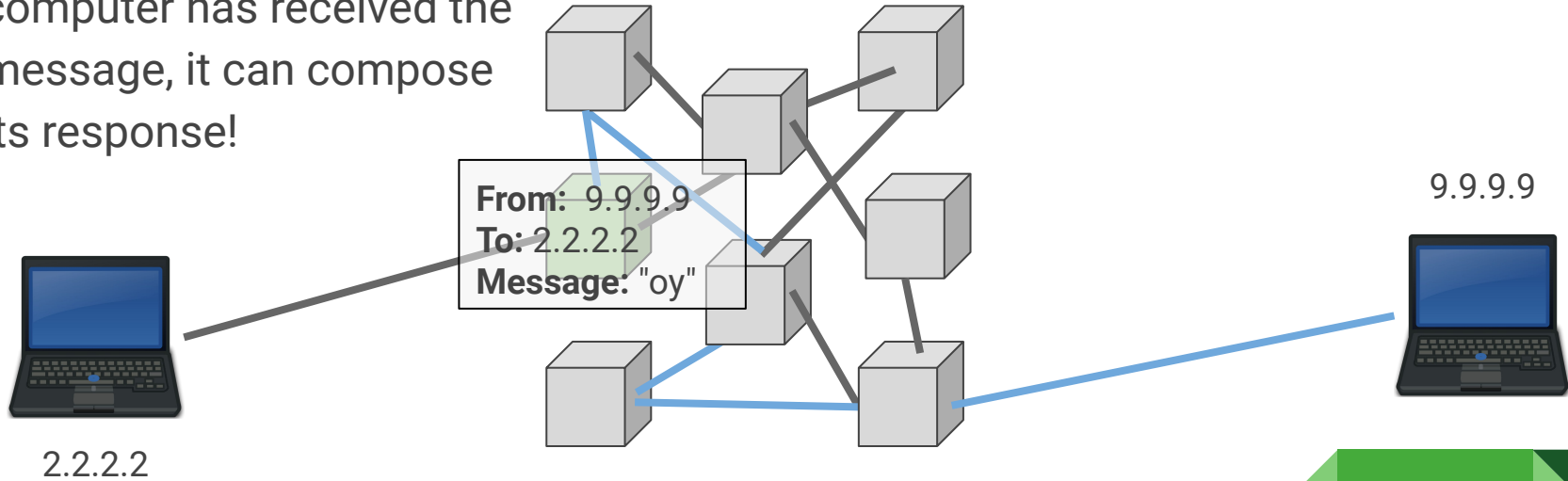
That computer then sends its message back using the same process!



# Routing

Now that the other computer has received the message, it can compose its response!

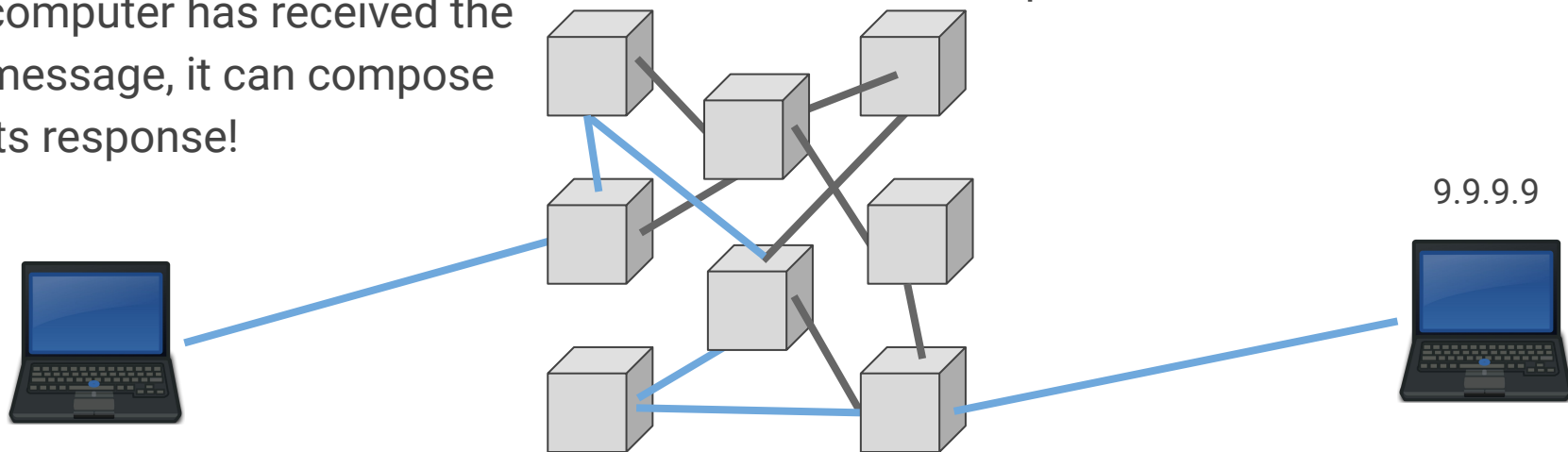
That computer then sends its message back using the same process!



# Routing

Now that the other computer has received the message, it can compose its response!

That computer then sends its message back using the same process!



2.2.2.2

**From:** 9.9.9.9  
**To:** 2.2.2.2  
**Message:** "oy"

9.9.9.9

# Redundancy

One great thing about the way the Internet is structured is that there's plenty of **redundancy** - there are many ways for information to travel in between two devices.



# Redundancy

One great thing about the way the Internet is structured is that there's plenty of **redundancy** - there are many ways for information to travel in between two devices.

This improves the overall reliability of the system! Since there are multiple different paths, the system is **fault tolerant** - that means it won't all come crashing down if a single piece breaks! Individual routers can (*and often do!*) break, but it doesn't mean that the whole Internet grinds to a halt.

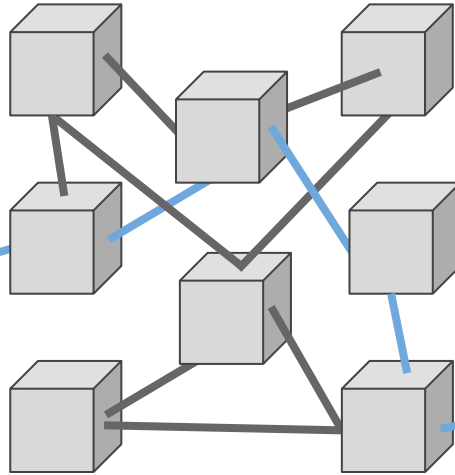


# Redundancy

If we look at this path, we can see that it's only **one** of many potential paths the data could take from one computer to the other!



2.2.2.2



9.9.9.9

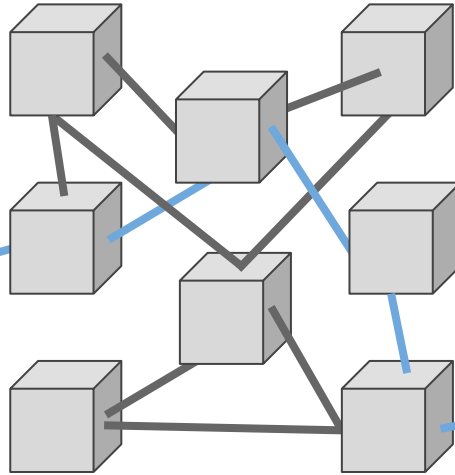


# Redundancy

If we look at this path, we can see that it's only **one** of many potential paths the data could take from one computer to the other!



2.2.2.2



If a router were to fail, we could simply find another way to get our information where it needs to go!

9.9.9.9



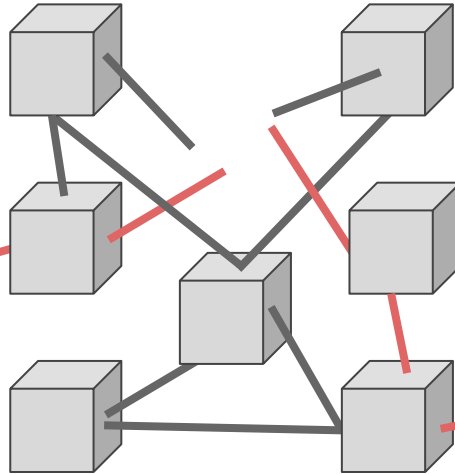


# Redundancy

If we look at this path, we can see that it's only **one** of many potential paths the data could take from one computer to the other!



2.2.2.2



If a router were to fail, we could simply find another way to get our information where it needs to go!

9.9.9.9

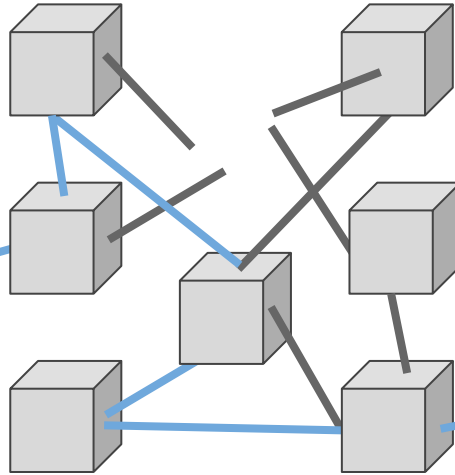


# Redundancy

If we look at this path, we can see that it's only **one** of many potential paths the data could take from one computer to the other!



2.2.2.2



If a router were to fail, we could simply find another way to get our information where it needs to go!

9.9.9.9

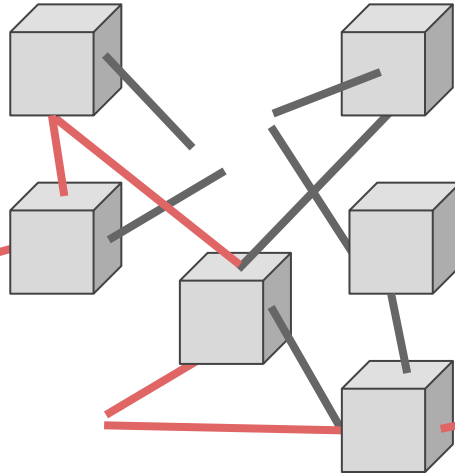


# Redundancy

If we look at this path, we can see that it's only **one** of many potential paths the data could take from one computer to the other!



2.2.2.2



If a router were to fail, we could simply find another way to get our information where it needs to go!

9.9.9.9

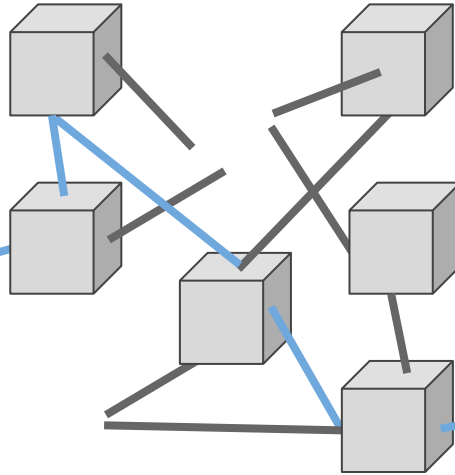


# Redundancy

If we look at this path, we can see that it's only **one** of many potential paths the data could take from one computer to the other!



2.2.2.2



If a router were to fail, we could simply find another way to get our information where it needs to go!

9.9.9.9



# Redundancy

The routing system is also wonderfully **scalable**. We've been using that word a lot in terms of good things about the Internet, because we want to futureproof our systems and ensure that the Internet will continue to run smoothly as it increases in size!

Unlike many other things, when we add more routers to the system, i.e. we increase the load, we get **better** performance! This is because we have even more redundancy!





# How Data Gets Transmitted

# What Do Our Messages Look Like?

Now that we have a good idea of **how** our messages are sent from one end of the Internet to the other, we're going to get a better idea of what the contents of each message looks like!



# Packets

Anytime data is sent over the Internet, it's broken down into small units of data called **packets**.

The reason we do this is to ensure that our data can transfer nicely - it's much easier to transfer many small pieces than one large whole!





# Packets

Anytime data is sent over the Internet, it's broken down into small units of data called **packets**.

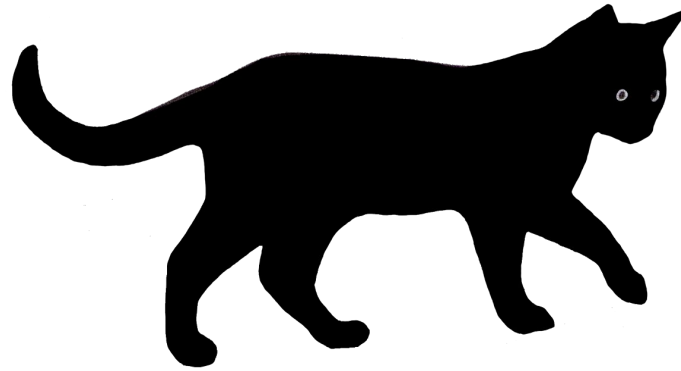
The reason we do this is to ensure that our data can transfer nicely - it's much easier to transfer many small pieces than one large whole!

Let's say you built a house in Minnesota, where it's cheap, but you want to move that house to California. It wouldn't be feasible to put that whole house on a truck and move it all at once! You'd want to break it down into its component pieces, then move each one individually. Each of those pieces could even take its own path to get to the destination!



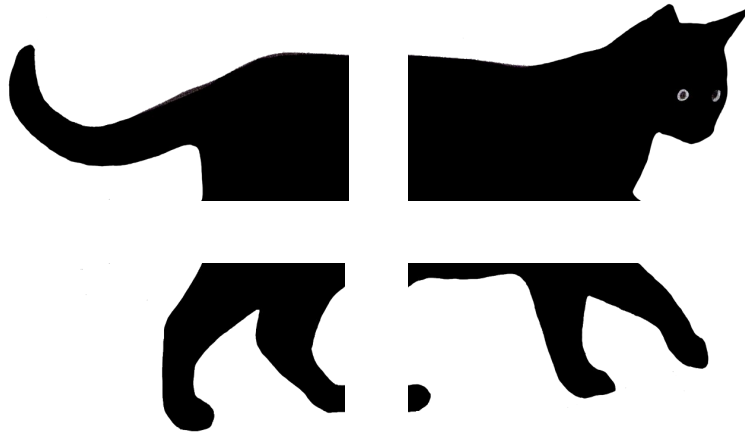
# Packets

Anytime data is sent over the Internet, it's broken down into small units of data called **packets**.



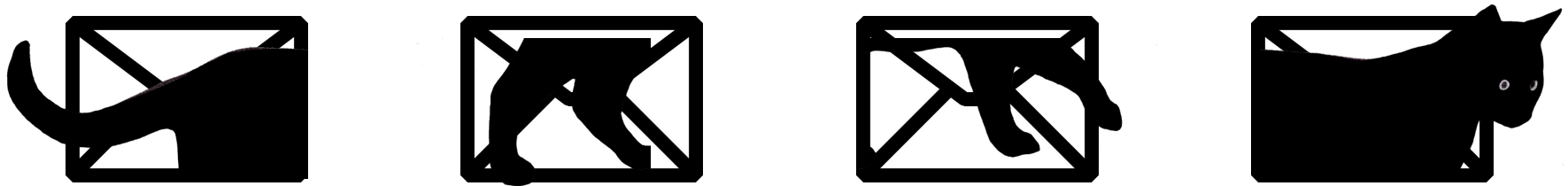
# Packets

Anytime data is sent over the Internet, it's broken down into small units of data called **packets**.



# Packets

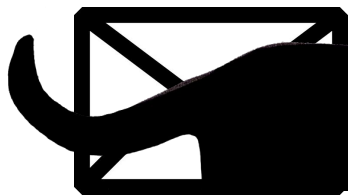
Anytime data is sent over the Internet, it's broken down into small units of data called **packets**.



# Packets

Anytime data is sent over the Internet, it's broken down into small units of data called **packets**.

Each of those packets is labeled with **metadata** - data *about* the data inside them!



To: 2.2.2.2  
From: 9.9.9.9  
Size: 5 KB



To: 2.2.2.2  
From: 9.9.9.9  
Size: 5 KB



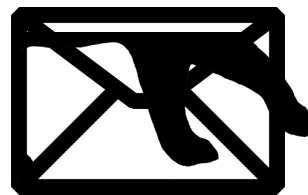
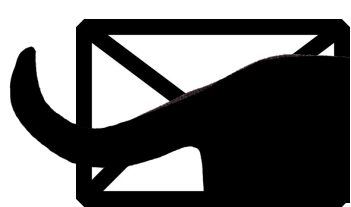
To: 2.2.2.2  
From: 9.9.9.9  
Size: 5 KB



To: 2.2.2.2  
From: 9.9.9.9  
Size: 5 KB

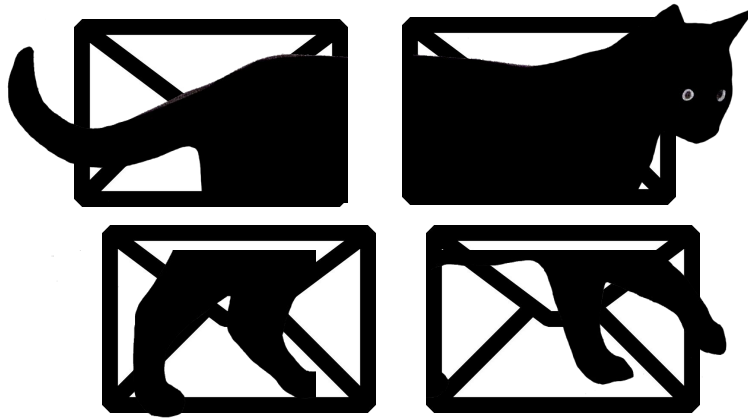
# Packets

Once those packets arrive on the other end, they can be re-assembled into an image!



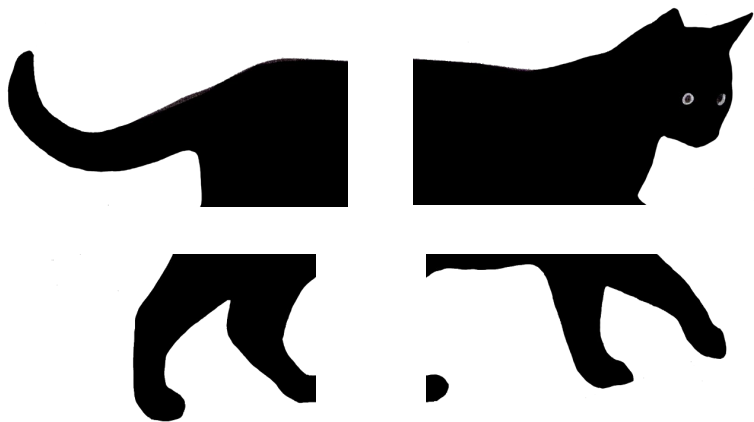
# Packets

Once those packets arrive on the other end, they can be re-assembled into an image!



# Packets

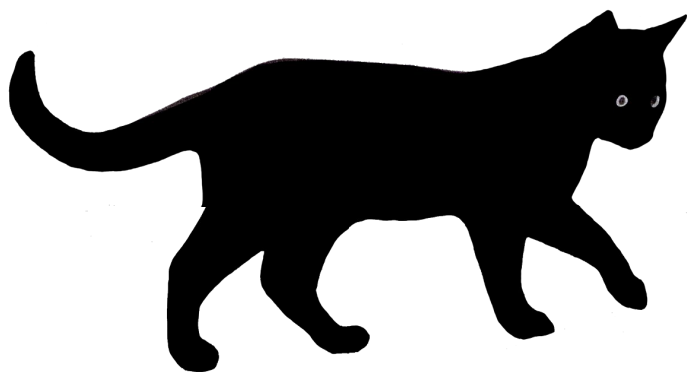
Once those packets arrive on the other end, they can be re-assembled into an image!





# Packets

Once those packets arrive on the other end, they can be re-assembled into an image!



# Packets

The layout of a packet is defined by a ***protocol***, as are many other things about how the Internet works!

This protocol is call the **Internet Protocol**, or **IP**! (Yes, it's the same protocol that dictates how computer addresses are structured!)



# Packets

The layout of a packet is defined by a ***protocol***, as are many other things about how the Internet works!

This protocol is call the **Internet Protocol**, or **IP**! (Yes, it's the same protocol that dictates how computer addresses are structured!)

There is a standard layout for every package using the Internet Protocol. All packets must have:

1. A Destination IP Address
2. A From IP Address
3. The actual data being transmitted



# Packets

The layout of a packet is defined by a ***protocol***, as are many other things about how the Internet works!

This protocol is called the **Internet Protocol**, or **IP**! (Yes, it's the same protocol that dictates how computer addresses are structured!)

There is a standard layout for every package using the Internet Protocol. All packets must have:

1. **A Destination IP Address**
2. **A From IP Address**
3. The actual data being transmitted

These two pieces of information are **metadata**!



# Packets

The Internet Protocol gives each device on a network an address, and defines the layout of a single packet. Using just this protocol, two computers on a network are able to send single packets to one another!



# Packets

The Internet Protocol gives each device on a network an address, and defines the layout of a single packet. Using just this protocol, two computers on a network are able to send single packets to one another!

But what if there are... *multiple* packets?



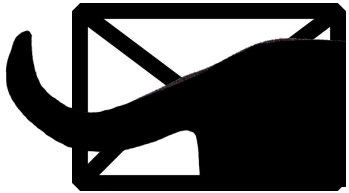
# Transmission Control Protocol

We need another **Protocol** to handle that! The **Transmission Control Protocol**, or TCP, is a protocol that allows for the sending of **multiple** packets between computers. TCP will check that all packets have arrived, and that they are able to be re-assembled in the proper order!



# IP Packets

Like I mentioned earlier, every packet is labeled with some **metadata** - information about the stuff held within the packet. With IP, the information is just the to address, the from address, and the size of the data being transmitted.



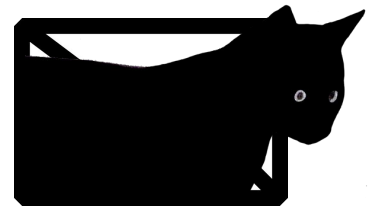
To: 2.2.2.2  
From: 9.9.9.9  
Size: 5 KB



To: 2.2.2.2  
From: 9.9.9.9  
Size: 5 KB



To: 2.2.2.2  
From: 9.9.9.9  
Size: 5 KB

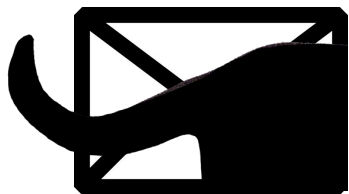


To: 2.2.2.2  
From: 9.9.9.9  
Size: 5 KB

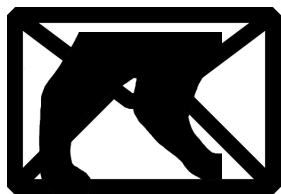


# TCP/IP Packets

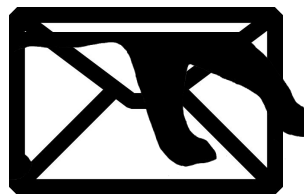
TCP adds a packet number to each packet, so that we can keep track of the order and quantity of our packets!



To: 2.2.2.2  
From: 9.9.9.9  
**Packet: 1/4**  
Size: 5 KB



To: 2.2.2.2  
From: 9.9.9.9  
**Packet: 2/4**  
Size: 5 KB



To: 2.2.2.2  
From: 9.9.9.9  
**Packet: 3/4**  
Size: 5 KB



To: 2.2.2.2  
From: 9.9.9.9  
**Packet 4/4**  
Size: 5 KB

# TCP/IP Packets

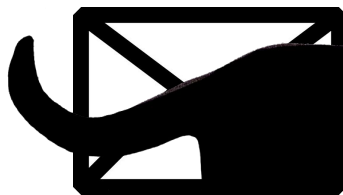
If our packets happen to arrive in the wrong order, this piece of metadata will allow them to be pieced back together correctly!



To: 2.2.2.2  
From: 9.9.9.9  
**Packet: 3/4**  
Size: 5 KB



To: 2.2.2.2  
From: 9.9.9.9  
**Packet: 2/4**  
Size: 5 KB



To: 2.2.2.2  
From: 9.9.9.9  
**Packet: 1/4**  
Size: 5 KB



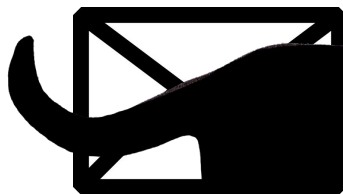
To: 2.2.2.2  
From: 9.9.9.9  
**Packet 4/4**  
Size: 5 KB

# TCP/IP Packets

If one of our packets is missing entirely, TCP will re-request that packet!



To: 2.2.2.2  
From: 9.9.9.9  
**Packet: 3/4**  
Size: 5 KB



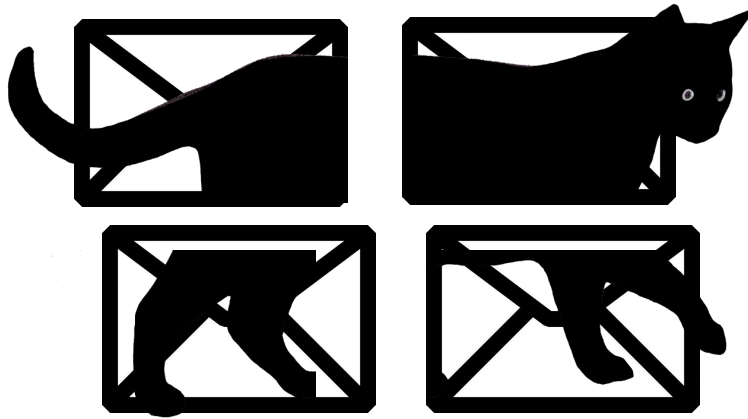
To: 2.2.2.2  
From: 9.9.9.9  
**Packet: 1/4**  
Size: 5 KB



To: 2.2.2.2  
From: 9.9.9.9  
**Packet 4/4**  
Size: 5 KB

# TCP/IP Packets

Once those packets arrive on the other end, they can be re-assembled in the correct order!



# TCP/IP Packets

We generally call packets that are following both the TCP **and** IP protocols TCP/IP packets. Each of these protocols defines some part of the **metadata** for each packet!

1. A Destination IP Address
2. A From IP Address
3. Size of message being sent
4. Order number




# TCP/IP Packets

We generally call packets that are following both the TCP **and** IP protocols TCP/IP packets. Each of these protocols defines some part of the **metadata** for each packet!

1. A Destination IP Address
2. A From IP Address
3. Size of message being sent
4. Order number

Next we're going to talk about the protocol that dictates how the data within our packets is actually stored!

A decorative graphic in the bottom right corner consisting of several overlapping green triangles and squares in different shades of green.

# HyperText Transfer Protocol

HyperText Transfer Protocol, or HTTP, is a protocol that standardizes the language that computers use to talk to web servers, as well as receiving web resources. It's a system that all computers use that standardizes communication between devices in how web resources are transmitted.



# HyperText Transfer Protocol

HyperText Transfer Protocol, or HTTP, is a protocol that standardizes the language that computers use to talk to web servers, as well as receiving web resources. It's a system that all computers use that standardizes communication between devices in how web resources are transmitted.

If you recognize the name HyperText, it's probably from HyperText Markup Language, or HTML - the language used to build websites!





# HyperText Transfer Protocol

HyperText Transfer Protocol, or HTTP, is a protocol that standardizes the language that computers use to talk to web servers, as well as receiving web resources. It's a system that all computers use that standardizes communication between devices in how web resources are transmitted.

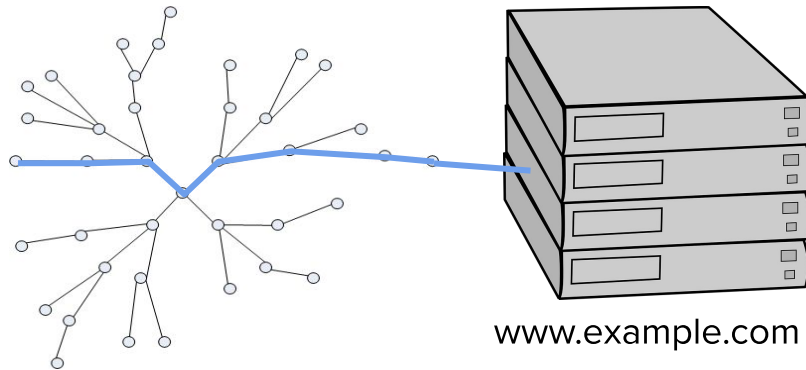
If you recognize the name HyperText, it's probably from HyperText Markup Language, or HTML - the language used to build websites!

HTTP defines how computers request and receive HyperText information.



# HTTP Request & Response

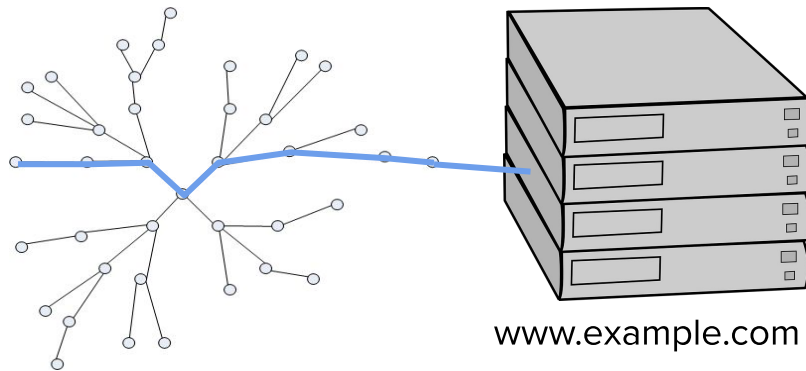
Anytime we send a message to a server asking for a web resource, there is both an HTTP **request** and an HTTP **response**.



"Hey, example.com! Please send me your homepage.html file!"

# HTTP Request & Response

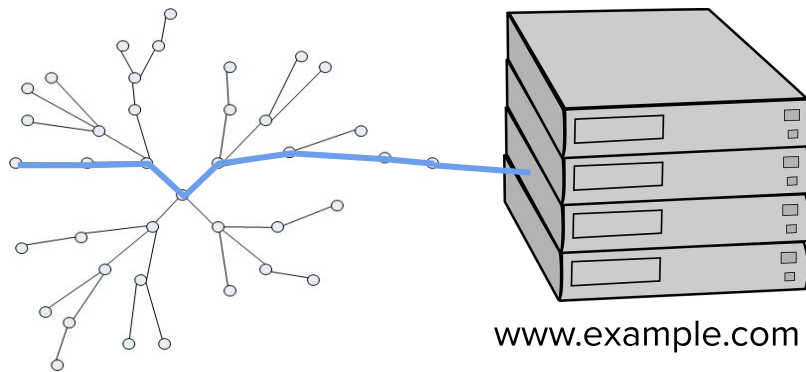
Anytime we send a message to a server asking for a web resource, there is both an HTTP **request** and an HTTP **response**.



```
GET /homepage.html HTTP/1.1
Host: www.example.com
Content-Type: text/html
Content-Language: en
...
```

# HTTP Request & Response

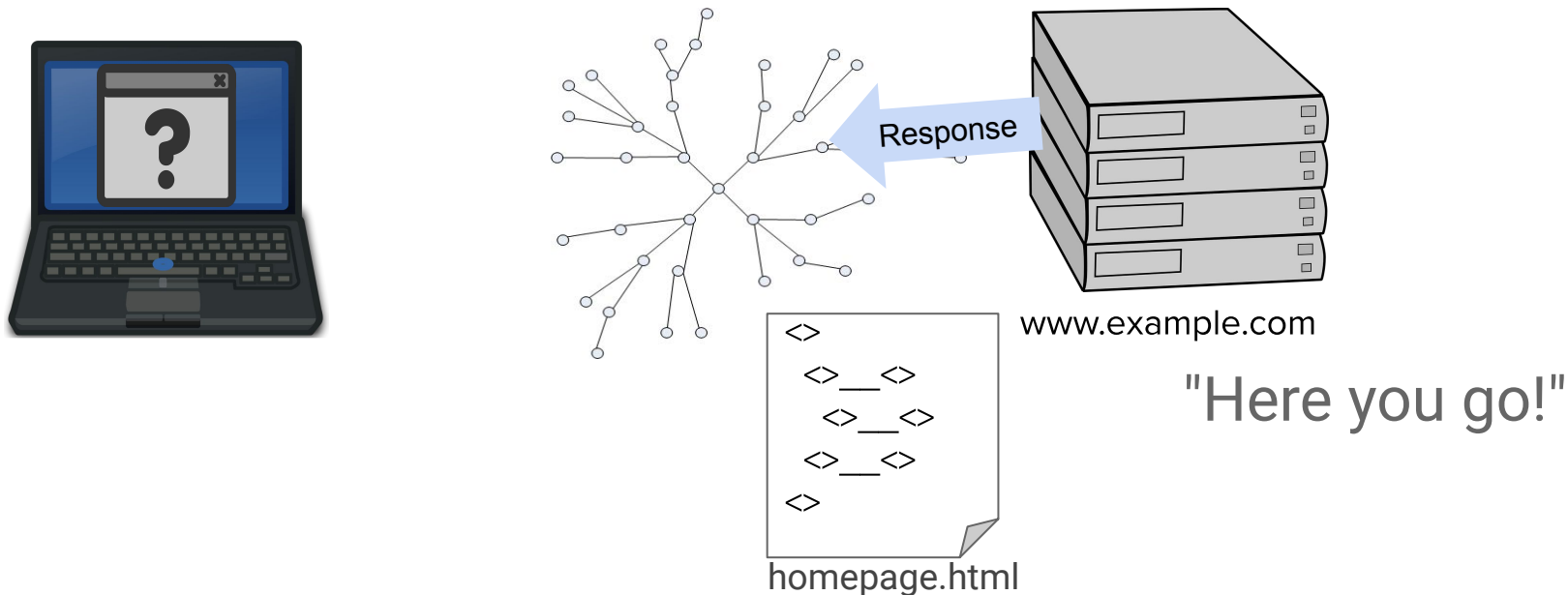
This request is received by the server, then a program reads it and determines how to respond!



```
GET /homepage.html HTTP/1.1
Host: www.example.com
Content-Type: text/html
Content-Language: en
...
```

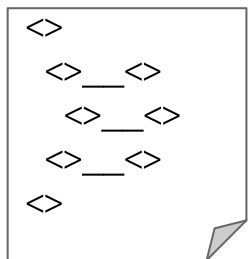
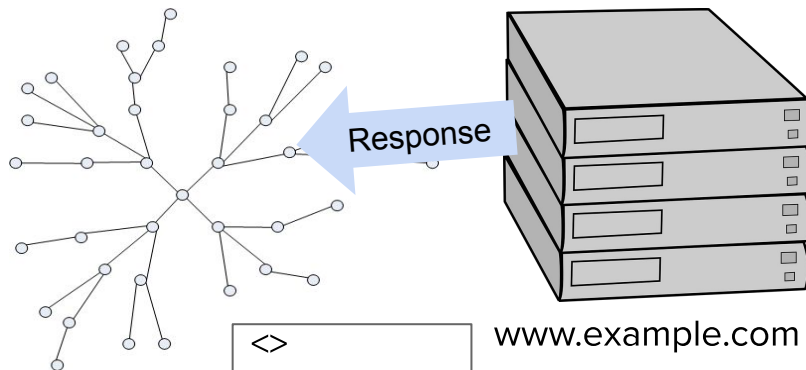
# HTTP Request & Response

This request is received by the server, then a program reads it and determines how to respond!



# HTTP Request & Response

This response will also be formatted using HTTP!



homepage.html

www.example.com

```
HTTP/1.1 200 OK
Server: LiteSpeed
Content-Type: text/html
Content-Language: en
...
```

# Internet Protocols

All of the protocols and systems we've learned about in this unit work together in tandem to send messages from one device to another!

TCP/IP, DNS, and routing all work together to send packets over the internet, and HTTP makes sure that the information within those packets can be understood on both ends!

Since all computers connected to the Internet follow these same rules, communication is standardized across devices! This makes transmission of data from one computer to another incredibly simple!



# Full Example Rundown

Alright, now that we have **ALL** the information about how data is transferred across the Internet, let's do a full rundown of what the process looks like!

How do all of these systems work together when I try to view a webpage?





# Step 1: The URL

The first step in viewing a webpage is building a URL to find that webpage!

[www.example.com/homepage.html](http://www.example.com/homepage.html)

**U**niform

**R**esource

**L**ocator

We are **locating** a **resource** that exists somewhere on the Internet!



# Step 1: The URL

The first step in viewing a webpage is building a URL to find that webpage!

[www.example.com/homepage.html](http://www.example.com/homepage.html)

## DOMAIN

**Where** on the Internet you  
need to look for this  
resource



# Step 1: The URL

The first step in viewing a webpage is building a URL to find that webpage!

[www.example.com/homepage.html](http://www.example.com/homepage.html)

## DOMAIN

**Where** on the Internet you need to look for this resource.

## PATH

**What** resource is being requested.



## Step 2: Create an HTTP Request

Once our browser knows which URL we want to visit, it will construct an HTTP request to fit those needs!

[www.example.com](http://www.example.com)

[/homepage.html](http://www.example.com/homepage.html)



```
GET /homepage.html HTTP/1.1  
Host: www.example.com  
Content-Type: text/html  
Content-Language: en  
...
```

## Step 3: Use DNS to find the IP Address

With our request complete, we need to know where we're going to send this request - if we don't have the IP Address in our cache, we'll need to use DNS!

[www.example.com](http://www.example.com)

[/homepage.html](http://www.example.com/homepage.html)



```
GET /homepage.html HTTP/1.1
Host: www.example.com
Content-Type: text/html
Content-Language: en
...
```

# Step 3: Use DNS to find the IP Address

DNS searches using increasingly specific servers to find the IP Address for the server with the domain you specified in your URL.

[www.example.com](http://www.example.com)

[/homepage.html](http://www.example.com/homepage.html)



```
GET /homepage.html HTTP/1.1  
Host: www.example.com  
Content-Type: text/html  
Content-Language: en  
...
```

## Step 3: Use DNS to find the IP Address

DNS searches using increasingly specific servers to find the IP Address for the server with the domain you specified in your URL.

www.example.com

/homepage.html



```
GET /homepage.html HTTP/1.1
Host: www.example.com
Content-Type: text/html
Content-Language: en
...
```

# Step 3: Use DNS to find the IP Address

DNS searches using increasingly specific servers to find the IP Address for the server with the domain you specified in your URL.

www.example.com

/homepage.html



```
GET /homepage.html HTTP/1.1  
Host: www.example.com  
Content-Type: text/html  
Content-Language: en  
...
```



## Step 3: Use DNS to find the IP Address

DNS searches using increasingly specific servers to find the IP Address for the server with the domain you specified in your URL.

[www.example.com](http://www.example.com)

[/homepage.html](#)



```
GET /homepage.html HTTP/1.1  
Host: www.example.com  
Content-Type: text/html  
Content-Language: en  
...
```

## Step 3: Use DNS to find the IP Address

DNS searches using increasingly specific servers to find the IP Address for the server with the domain you specified in your URL.

[www.example.com](http://www.example.com)

[/homepage.html](http://www.example.com/homepage.html)



```
GET /homepage.html HTTP/1.1  
Host: www.example.com  
Content-Type: text/html  
Content-Language: en  
...
```

IP Address Acquired!  
**5.5.5.5**

# Step 4: TCP/IP Breaks the Request into Packets

Using the TCP/IP protocols, we can then break our request down into several packets and label them with the appropriate metadata.

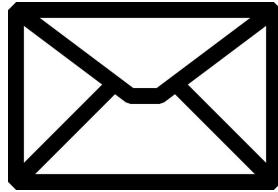


```
GET /homepage.html HTTP/1.1  
Host: www.example.com  
Content-Type: text/html  
Content-Language: en  
...
```

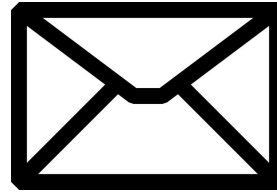
IP Address Acquired!  
**5.5.5.5**

# Step 4: TCP/IP Breaks the Request into Packets

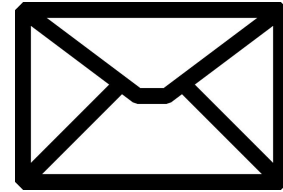
Using the TCP/IP protocols, we can then break our request down into several packets and label them with the appropriate metadata.



To: 5.5.5.5  
From: 9.9.9.9  
Packet: 1/3  
Size: 5 KB



To: 5.5.5.5  
From: 9.9.9.9  
Packet: 2/3  
Size: 5 KB



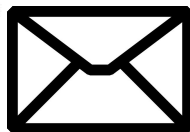
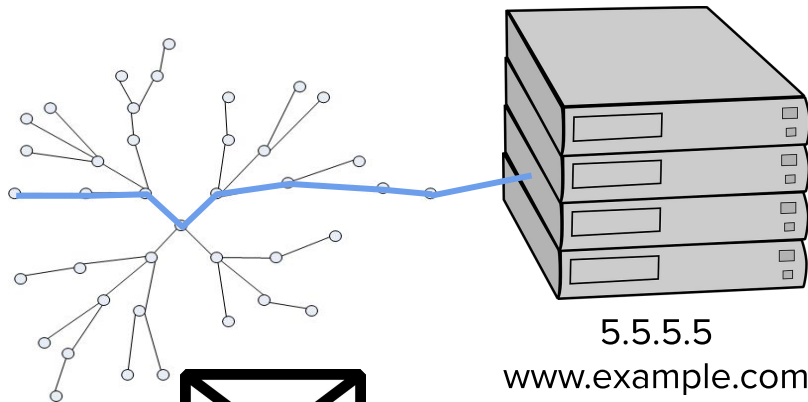
To: 5.5.5.5  
From: 9.9.9.9  
Packet: 3/3  
Size: 5 KB

# Step 5: Packets are Routed to the proper IP Address

We then transmit those packets across the Internet, through the vast network of routers!



9.9.9.9  
Our Computer



To: 5.5.5.5  
From: 9.9.9.9  
Packet: 3/3  
Size: 5 KB



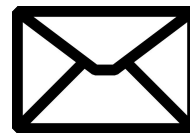
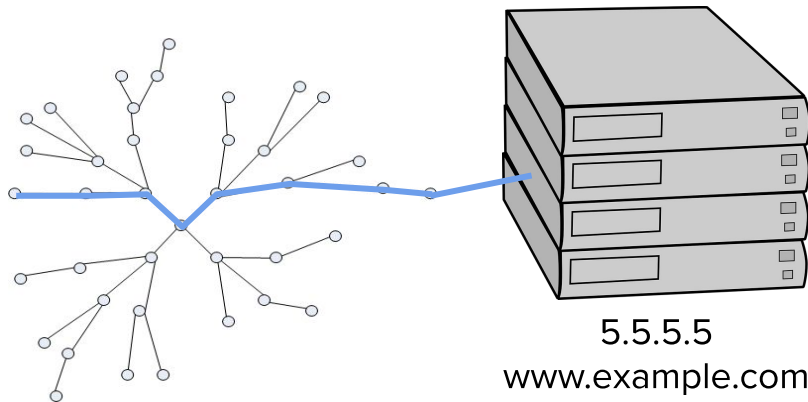
To: 5.5.5.5  
From: 9.9.9.9  
Packet: 2/3  
Size: 5 KB

# Step 5: Packets are Routed to the proper IP Address

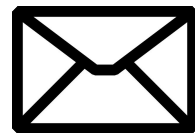
We then transmit those packets across the Internet, through the vast network of routers!



9.9.9.9  
Our Computer



To: 5.5.5.5  
From: 9.9.9.9  
Packet: 2/3  
Size: 5 KB



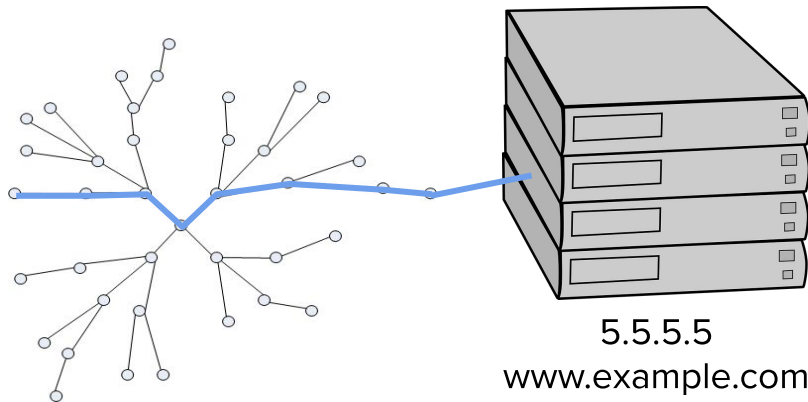
To: 5.5.5.5  
From: 9.9.9.9  
Packet: 3/3  
Size: 5 KB

# Step 5: Packets are Routed to the proper IP Address

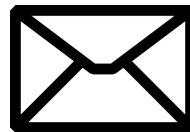
If any packets got lost along the way, TCP will go back to your computer and request that packet again!



9.9.9.9  
Our Computer



5.5.5.5  
www.example.com



To: 5.5.5.5  
From: 9.9.9.9  
Packet: 2/3  
Size: 5 KB



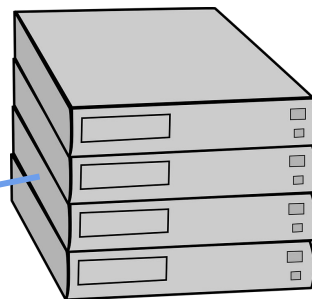
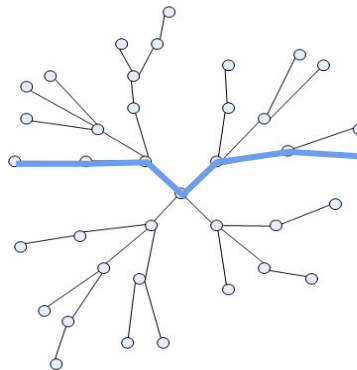
To: 5.5.5.5  
From: 9.9.9.9  
Packet: 3/3  
Size: 5 KB

# Step 5: Packets are Routed to the proper IP Address

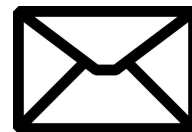
If any packets got lost along the way, TCP will go back to your computer and request that packet again!



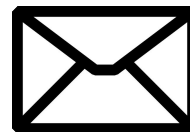
9.9.9.9  
Our Computer



5.5.5.5  
www.example.com



To: 5.5.5.5  
From: 9.9.9.9  
Packet: 1/3  
Size: 5 KB



To: 5.5.5.5  
From: 9.9.9.9  
Packet: 2/3  
Size: 5 KB



To: 5.5.5.5  
From: 9.9.9.9  
Packet: 3/3  
Size: 5 KB

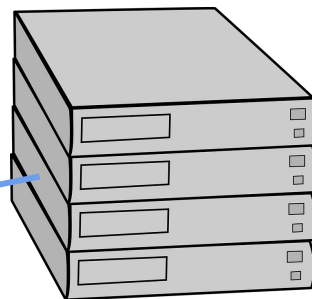
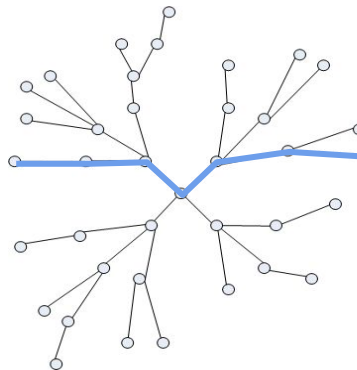


# Step 6: Original Request is Constructed from Packets

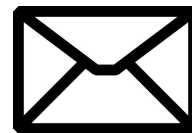
Once the recipient has received all the parts of the request, then can put them together in order to find the original request!



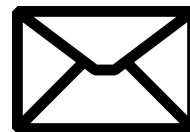
9.9.9.9  
Our Computer



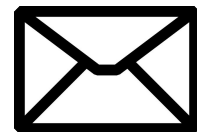
5.5.5.5  
www.example.com



To: 5.5.5.5  
From: 9.9.9.9  
Packet: 1/3  
Size: 5 KB



To: 5.5.5.5  
From: 9.9.9.9  
Packet: 2/3  
Size: 5 KB



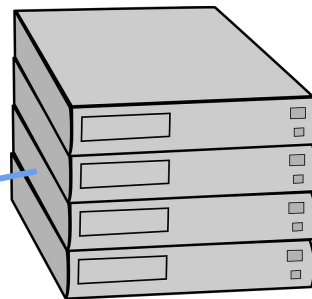
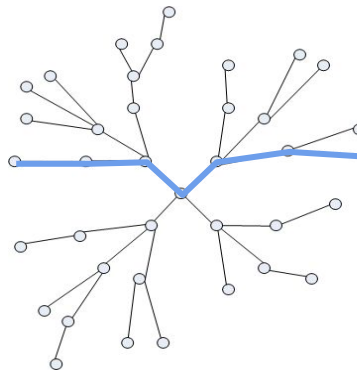
To: 5.5.5.5  
From: 9.9.9.9  
Packet: 3/3  
Size: 5 KB

## Step 6: Original Request is Constructed from Packets

Once the recipient has received all the parts of the request, then can put them together in order to find the original request!



9.9.9.9  
Our Computer



5.5.5.5  
www.example.com

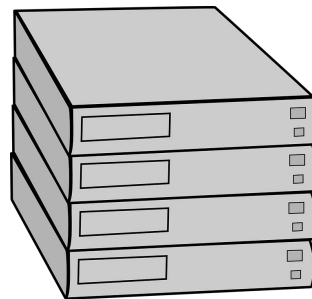
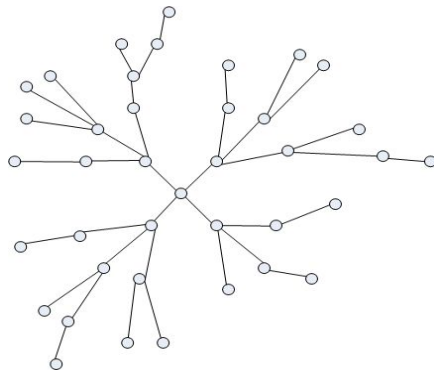
```
GET /homepage.html HTTP/1.1
Host: www.example.com
Content-Type: text/html
Content-Language: en
...
```

# Step 7: Server Creates an HTTP Response

After the server reads your request, it will compile any resources that were requested and create an HTTP response to send back!



9.9.9.9  
Our Computer



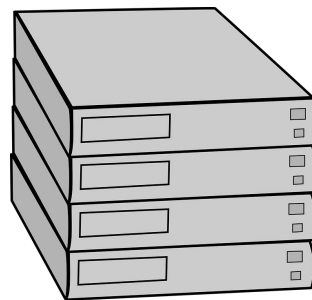
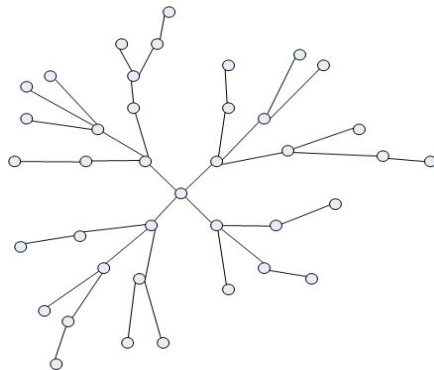
5.5.5.5  
www.example.com

# Step 7: Server Creates an HTTP Response

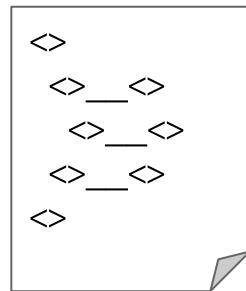
After the server reads your request, it will compile any resources that were requested and create an HTTP response to send back!



9.9.9.9  
Our Computer



5.5.5.5  
www.example.com



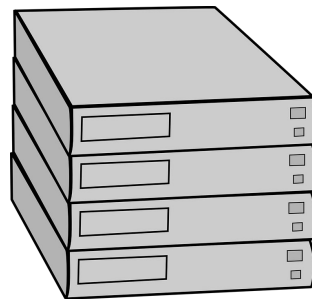
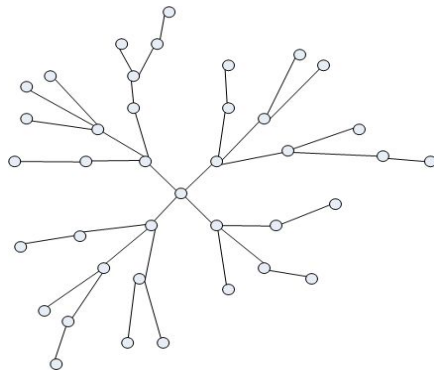
```
HTTP/1.1 200 OK    homepage.html
Server: LiteSpeed
Content-Type: text/html
Content-Language: en
...
```

# Step 7: Server Creates an HTTP Response

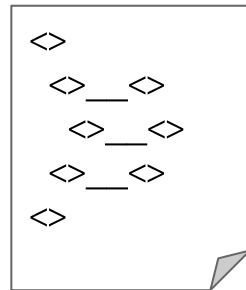
This HTTP response will include a status code (in this case 200 OK). The actual HTML file is included in the response!



9.9.9.9  
Our Computer



5.5.5.5  
www.example.com



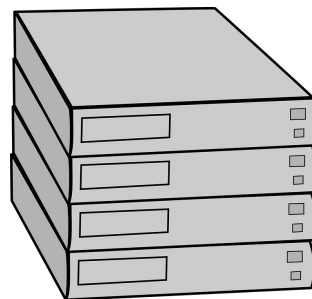
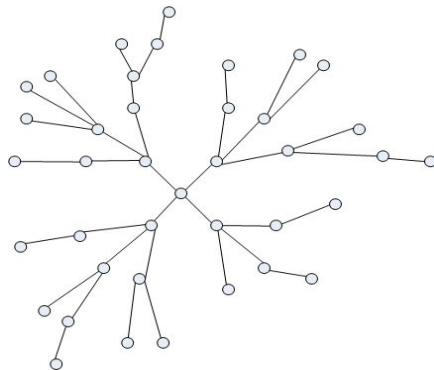
```
HTTP/1.1 200 OK    homepage.html
Server: LiteSpeed
Content-Type: text/html
Content-Language: en
...
```

# Step 8: TCP/IP Breaks the Response into Packets

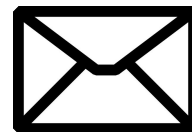
Just like when the request was sent, the response also needs to be broken down into packets!



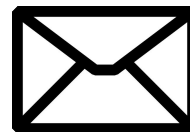
9.9.9.9  
Our Computer



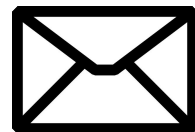
5.5.5.5  
www.example.com



To: 9.9.9.9  
From: 5.5.5.5  
Packet: 1/3  
Size: 5 KB



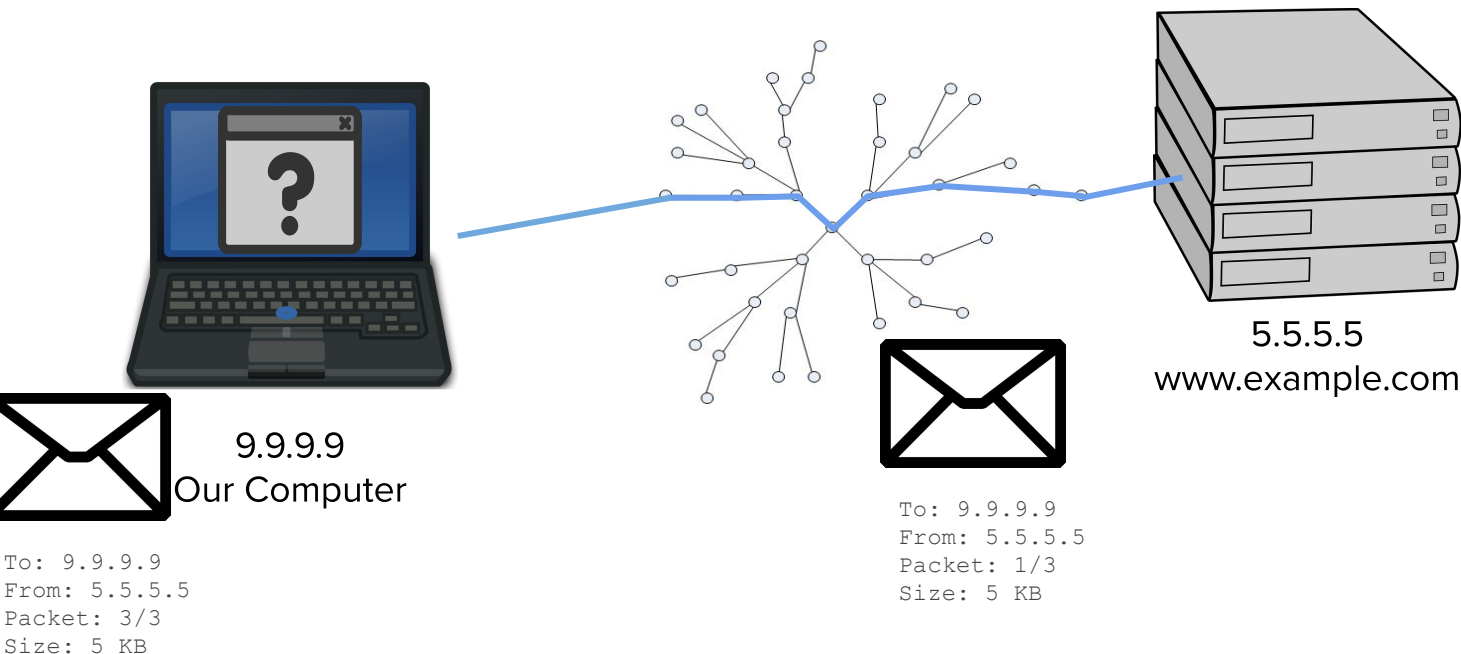
To: 9.9.9.9  
From: 5.5.5.5  
Packet: 2/3  
Size: 5 KB



To: 9.9.9.9  
From: 5.5.5.5  
Packet: 3/3  
Size: 5 KB

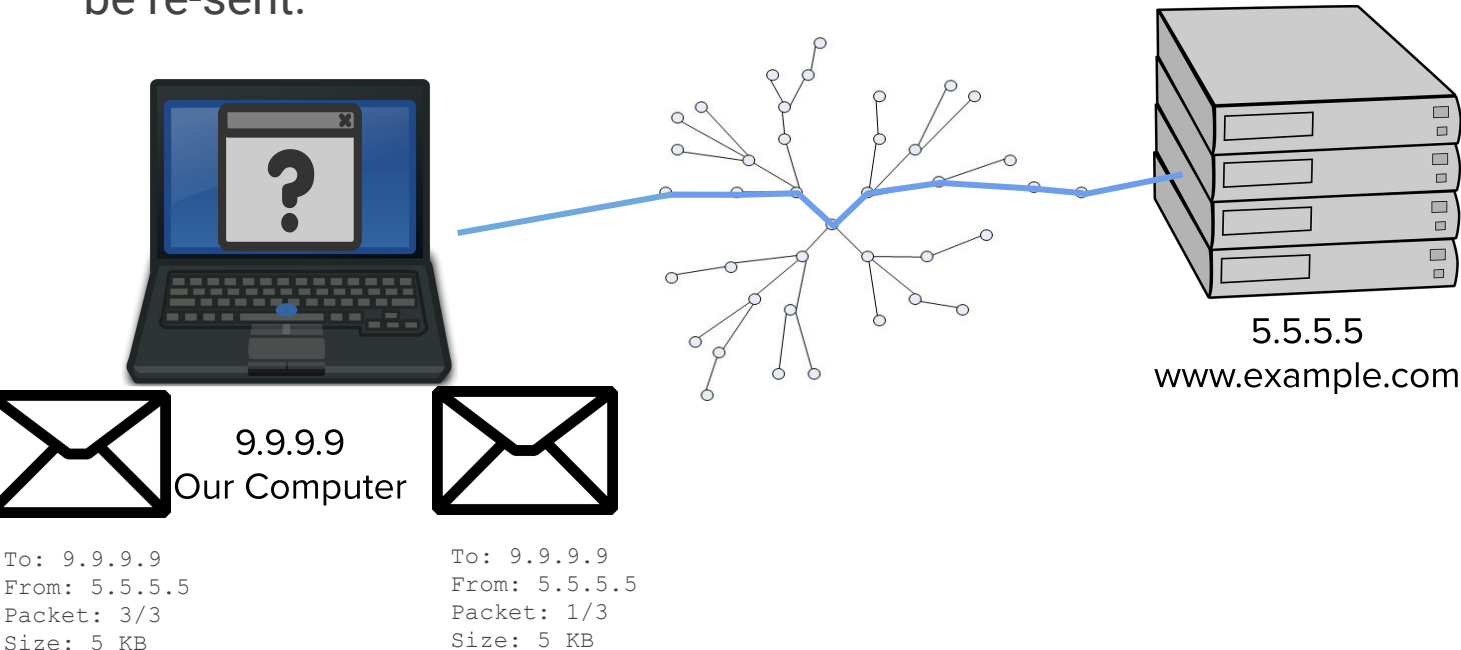
# Step 9: Response Packets are Routed Back to Computer

Each packet is routed through the Internet once again!



# Step 9: Response Packets are Routed Back to Computer

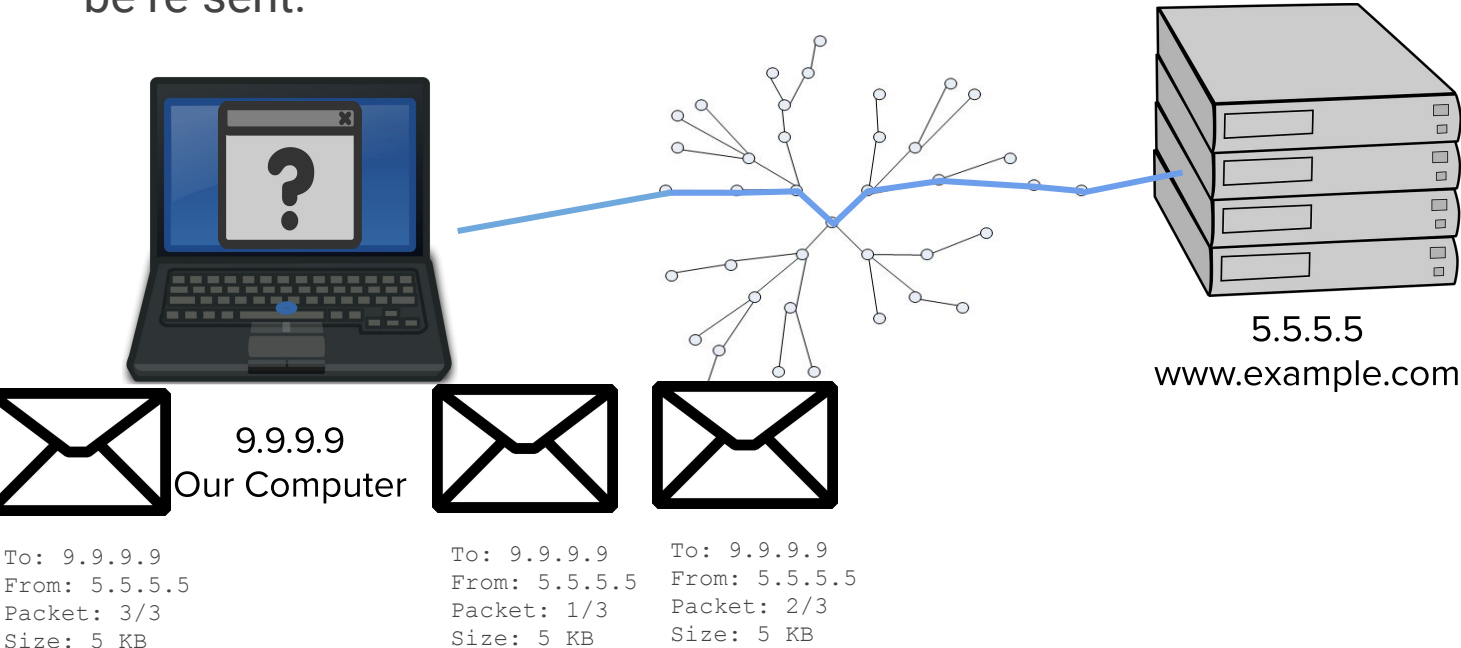
Once again, if any packets were lost on the way, TCP will request that those packets be re-sent.





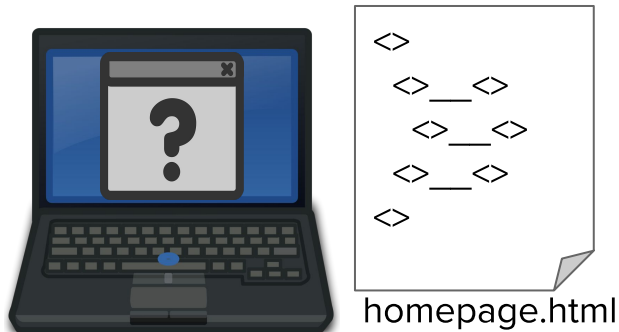
# Step 9: Response Packets are Routed Back to Computer

Once again, if any packets were lost on the way, TCP will request that those packets be re-sent.



# Step 10: Response is Reassembled from the Packets

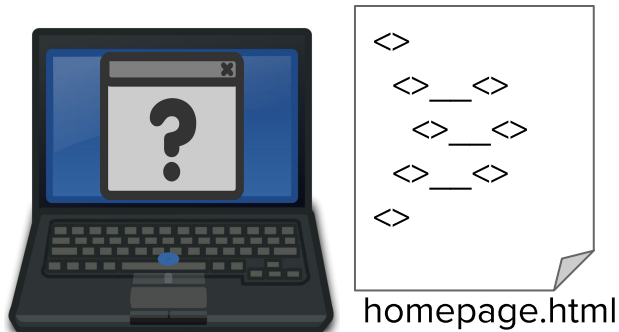
Our computer then re-assembles the packets into the original response! It's able to do this because the computer and the server are using the same exact **protocols**!



```
9.9.9.9    HTTP/1.1 200 OK
Our Computer Server: LiteSpeed
           Content-Type: text/html
           Content-Language: en
           ...
```

# Step 11: The Webpage is Rendered

The last step has our web browser do its thing with the HTML file we received back, rendering it into the beautiful webpage it was always meant to be!



9.9.9.9  
Our Computer

# Step 11: The Webpage is Rendered

The last step has our web browser do its thing with the HTML file we received back, rendering it into the beautiful webpage it was always meant to be!



# Summary

The Internet is a system of routers, though with digital data is sent by breaking it down into blocks of bits called **packets**.

Each packet contains both the **data being transmitted** and the **metadata** that helps **route** the data - it makes sure it gets where it's going!

This whole process works because every machine connected to the Internet has agreed to use the same protocols for creating, sending, and reading packets!

