A Simplified Example of TCP/IP Communication

Chuck Cusack

This is a simple example of how messages are sent from one machine to another (from a *source* to a *destination*) using the four-layer Internet software model. The four layers are application, transport, network, and link layers. In TCP/IP, the transport layer is TCP (or UDP, etc.), the network layer is IP, and the link layer is Ethernet. Although the layers used here are the same as the TCP/IP protocol, it is not exactly TCP/IP. Several simplifying assumptions were made and some of the details are not exactly how TCP/IP is implemented. This is done so we can focus on only a few of the important issues involved—specifically the jobs of each of the four layers.

The four layers

Here is a description of the tasks of each of the four layers. A brief example is given for each. Following this, a longer example is given to demonstrate the overall process.

APPLICATION

Sending: It creates a message, specifies the destination, and forwards it to the *transport layer*.
Receiving: It receives messages from the transport layer and does whatever it needs to with them.

A few important details:

- The source and destination need to be specified by the *IP address* of each machine. When sending, the machine in question is the source. When receiving (from the transport layer), the destination is the machine in question.
- Users don't like to remember IP addresses and instead use *URLs*. Thus, the application layer is responsible for determining the IP address based on the URL. Applications use *Domain Name System* (*DNS*) to look up IP addresses.
- Since there are many applications running on a computer there has to be a way to determine which message is for which application. For instance, if the homepage from cnn.com is returned, should it be sent to *FireFox* or *Chrome*? This is done by assigning a *port* to each application. A port is just a number—think of it as a room number or the number in a street address. As we will see, the transport layer forwards messages to applications based on the port number.
- Some port numbers are fixed and known to everyone. For instance, *HTTP* uses port 80, *SMTP* uses 25, *FTP* uses 21, *SSH* uses 22, *POP* uses 110, etc. This is important since it allows any computer in the world to request a webpage by sending the request to port 80 of the proper machine (e.g. cnn.com). Other ports are assigned as needed—for instance, your web browser (FireFox, Chrome, etc.) might be assigned different port number every time you run them.

Example 1: I want to see the latest XKCD comic, so I type http://xkcd.com into *Firefox* (my web browser). Firefox knows that this means that I want to view the webpage at that URL. Firefox noticed that I specified the address as "xkcd.com", which is not an IP address. It uses DNS and determines that the IP address is 67.215.65.132. Since it is an HTTP request, it also knows that port 80 should be used. Finally, Firefox tells the transport layer "Request the webpage at port 80 on IP Address 67.215.65.132 and have it sent back to me (IP address 209.140.209.32)."

Transport

Sending: When sending a message (from the application layer), the transport layer is responsible to chop it up into manageable sized *packets*, assign a sequence number to each packet (so they can be put back together in the right order), and forward the packets to the *network layer*.
Receiving: When receiving packets (from the network layer), it is responsible to reassemble packets into the original message (using the sequence numbers) and forward the message to the appropriate application based on the port number.

A few important details:

- The application layer will indicate the port on the destination machine (since generally it will be a well-known port number), but the transport layer must add to each packet the port number it assigned to the application that is sending the message. Again, it needs to do this so that when the response comes back the transport layer will know which application should receive the response.
- The transport layer sends the network layer a packet with source and destination IP addresses and port numbers, along with a portion of the original message (assuming it was split into multiple packets).

Example 2: The transport layer received the request from Example 1 and decided it was too long, so it chopped it into three parts, giving them sequence numbers 1, 2, and 3. It was told to use port 80 on the destination and has assigned Firefox port number 2143. For each of the three parts of the original message, it tells the network layer: "Please send this packet (with its contents) from port 2143 at IP address 209.140.209.32 to port 80 at IP address 67.215.65.132."

Network

The job of the network layer is to route packets to their destination. When the network layer receives a packet (from either the *link layer* or the *transport layer*) it needs to decide where to send it next. The network layer can do one of three things with a packet:

1. If it is from the link layer and is addressed to *this machine*, it forwards it to the transport layer.
2. If it is for *a machine in the same network*, it forwards it to that machine.
3. If it is for *a machine in a different network*, it forwards it to the appropriate router than can get it closer to its final destination. We won't worry about how it decides this.

Since the link layer in Ethernet relies on *MAC addresses* (*Media Access Control address*, also called *Ethernet addresses*) to send packets, the network layer needs to determine the source and destination MAC addresses. MAC addresses are usually represented with 12 hexadecimal characters, and when written, typically a colon is placed between every other character (e.g. 18:A9:05:B8:E3:7D). It is important to note that *the source and destination MAC addresses may or may not correspond to the source and destination IP addresses*. In particular, it always sets the source to its own MAC address since it is the one sending the packet right now. The destination is set to the MAC address of the next machine it is going to—either a router or the final destination. It determines MAC addresses using *ARP* (*Address Resolution Protocol*).

Example: The MAC address of the source is 18:A9:05:B8:E3:7D. The destination is not on this network, but the router with IP address 67.200.45.1 and MAC address 00:34:AB:86:7D:20 will get it closer. So the network layer tells the link layer to send the packet (including the contents, IP addresses, and port numbers) to MAC address 00:34:AB:86:7D:20 from MAC address 18:A9:05:B8:E3:7D. Notice that we never used the IP address of the router.

LINK

Sending: When the network layer gives a packet to the link layer, the link layer simply sends it on its way on the network.

Receiving: When the link layer receives a packet from the network it looks at the MAC address to see who it is addressed to.

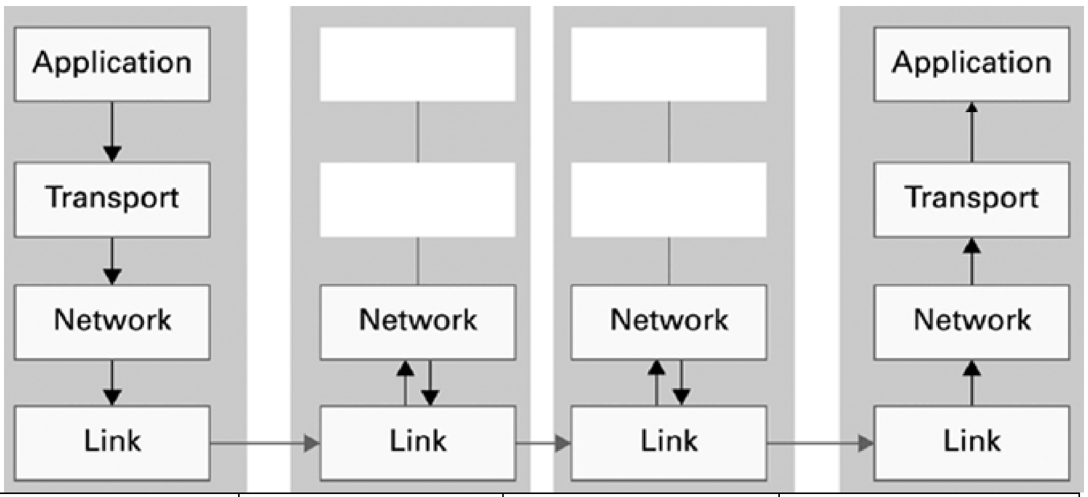
- If it is the MAC address of this computer, it forwards it to the network layer.
- Otherwise it ignores the packet.

Example: Given the above request, the link layer simply sends the message on its way.

The Big Example

Now we give a more complete example. For this example I am on *foo.cs.hope.edu* and want to see a webpage from *homestarrunner.com*. There are two routers between foo.cs.hope.edu and homestarrunner.com as seen in the diagram below. The example will follow the message from foo.cs.hope.edu to homestarrunner.com. We will focus on what happens at the various layers on the various machines involved.

The network in our example



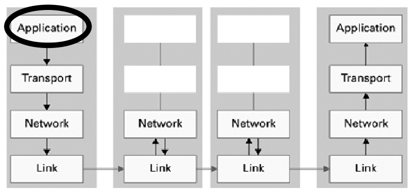
Machine	foo.cs.hope.edu	Router 1	Router 2	homestarrunner.com
IP Address	198.110.97.7	198.110.96.1	123.45.67.89	69.59.23.70
MAC Address	00:07:e9:e5:ca:43	00:08:e2:62:34:08	00:01:02:03:04:05	11:22:33:44:aa:bb

On foo.cs.hope.edu

User

1. I am working on foo.cs.hope.edu, and I want to load homestarrunner.com, so I put the URL http://homestarrunner.com into the address bar of *FireFox*.

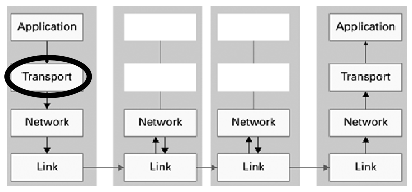
Application Layer (FireFox)



1. A user wants the contents from http://homestarrunner.com. I need to request this. In order to do so, I need to know the IP of the machine and the port number of the application on that machine that handles the request I am going to make.
2. Since the URL starts with "http", the request is using the HTTP protocol and I want to communicate with the web server. These requests always use port 80.
3. To get the IP address for homestarrunner.com, I use DNS. The IP is 69.59.23.70.
4. I also need to specify my IP address as the return address so the information can be sent back. Mine is 198.110.97.7.
5. Finally, I need to send the following to the **Transport Layer**:

   *Please send the message "*Send the contents of index.html*" to port 80 of 69.59.23.70, and return the response to me at IP 198.110.97.7.*

Transport Layer



1. An application (FireFox) asked me to send the message "Send the contents of index.html".
2. I need to pick a port number for FireFox so when I get a response I know where it goes. Port 2175 is available, so I will use that.
3. I don't know what this message means, and I don't care. But I do know that the message is too long, so I'll break it up into 2 packets: "Send the conten" and "ts of index.html".
4. Next I need to create two packets which specify the messages, ports, and sequence numbers so the packets can be reassembled. The packets are:

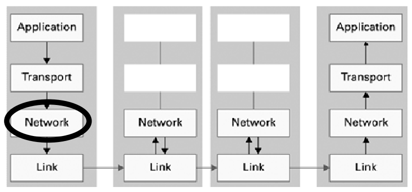
src port	dst port	Seq#	Message
2175	80	1	Send the conten

src port	dst port	Seq#	Message
2175	80	2	ts of index.html

5. Finally, I need to send the following two messages to the **Network Layer**:

   Please send the message "2175|80|1|Send the conten" *to 69.59.23.70, and set the return address to 198.110.97.7.*
   Please send the message "2175|80|2|ts of index.html" *to 69.59.23.70, and set the return address to 198.110.97.7.*

Network Layer



1. The transport layer gave me a few packets to send to 69.59.23.70. I don't know what the details in the *Message* from the transport layer mean. I just treat it as a blob of data. The transport layer on the receiving machine can worry about what it means.
2. I assemble two packets as follows:

src IP	dst IP	Message
198.110.97.7	69.59.23.70	2175|80|1|Send the conten

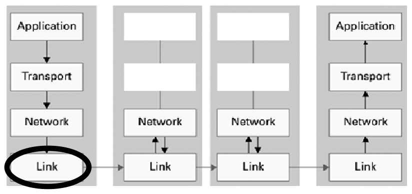
src IP	dst IP	Message
198.110.97.7	69.59.23.70	2175|80|2|ts of index.html

3. Now I need to determine if the destination is in my network. If so, I will send the packet directly. If not, I will send it to a router to forward. Unfortunately, this is not in my network. I'll send it to the router with IP 198.110.96.1 since it is the only one connected to my network.
4. Next, I need to determine the Ethernet addresses of the source and destination. I need to do this because the link layer does not use IP addresses—it uses Ethernet addresses.
5. The source is always the machine sending the packets, which is not necessarily the original source of the message. I can use the *arp* protocol to look up the addresses, and I see that
   a. My Ethernet address is 00:07:e9:e5:ca:43.
   b. The Ethernet address of 198.110.96.1 (the current destination) is 00:08:e2:62:34:08.

   Notice that the Ethernet address of the destination is not the final destination, but the intermediate step.
6. Finally, I send the following messages to the **Link Layer**:

   Please send the message "198.110.97.7|69.59.23.70|2175|80|1|Send the conten" *to* 00:08:e2:62:34:08*, and set the return address to* 00:07:e9:e5:ca:43.
   Please send the message "198.110.97.7|69.59.23.70|2175|80|2|ts of index.html" *to* 00:08:e2:62:34:08*, and set the return address to* 00:07:e9:e5:ca:43.

Link Layer



1. The network layer gave me some messages to send. I don't know what the details in the *Message* from the network layer mean. I just treat it as a blob of data. The network layer on the destination can worry about what it means.
2. I have to assemble and send the packets to the specified address. The packets are:

src Ethernet	dst Ethernet	Message
00:07:e9:e5:ca:43	00:08:e2:62:34:08	198.110.97.7|69.59.23.70|2175|80|1|Send the conten

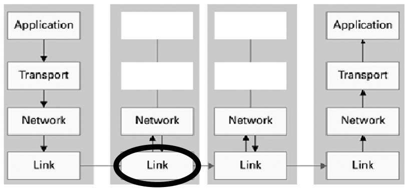
src Ethernet	dst Ethernet	Message
00:07:e9:e5:ca:43	00:08:e2:62:34:08	198.110.97.7|69.59.23.70|2175|80|2|ts of index.html

3. I send the packets to the machine with Ethernet address 00:08:e2:62:34:08. More accurately, I send the packets to the network card which will forward it to the network which will send it to all of the machines on that part of the network (since Ethernet uses a bus architecture).

We will ignore the second packet for a while, since exactly the same thing happens with each packet until they both get to the final destination.

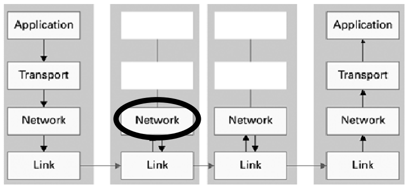
On Router 1 (IP 198.110.96.1 and Ethernet address 00:08:e2:62:34:08)

Link Layer



1. I received a packet from the network. It is addressed to my MAC address, so I'll extract the message and give it to the network layer since that is all I know how to do.
2. I send the following to the network layer:
   Received message:
   "198.110.97.7|69.59.23.70|2175|80|1|Send the conten".

Network Layer



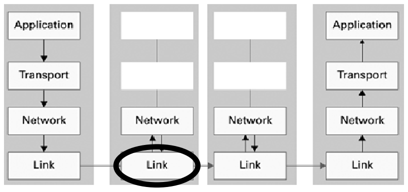
1. I was given a packet from the link layer. Here is what it is:

src IP	dst IP	Message
198.110.97.7	69.59.23.70	2175|80|1|Send the conten

2. It is not addressed to me, so I need to forward it.
3. It is headed to an IP address that is not on my network. I have 4 possible routers to forward it through. After looking a few things up, I determine that I should forward it to IP 123.45.67.89.
4. Next, I need to determine the Ethernet addresses of the source (me) and destination.
   a. My Ethernet address is 00:08:e2:62:34:08
   b. The Ethernet address of 123.45.67.89 (the current destination) is 00:01:02:03:04:05

   Notice that neither the source nor destination Ethernet address corresponds to the IP addresses of the original source or destination. The link layer, which uses these addresses, does not care about the original source or final destination. It only cares about where it is coming from and going to *right now*.
5. Finally, I send the following messages to the **Link Layer**:
   Please send the message "198.110.97.7|69.59.23.70|2175|80|1|Send the conten" *to* 00:01:02:03:04:05*, and set the return address to* 00:08:e2:62:34:08.

Link Layer



1. The network layer gave me a message to send.
2. I have to assemble and send the packet to the specified address. The packet is:

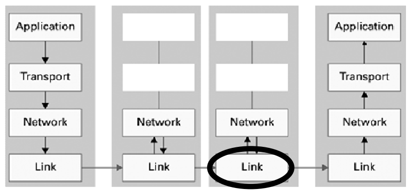
src Ethernet	dst Ethernet	Message
00:08:e2:62:34:08	00:01:02:03:04:05	198.110.97.7|69.59.23.70|2175|80|1|Send the conten

3. I send the packet to the machine with Ethernet address 00:01:02:03:04:05.

On Router 2 (IP 123.45.67.89 and Ethernet address 00:01:02:03:04:05)

Link Layer

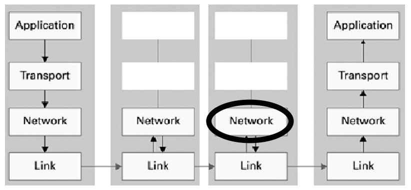
1. I received a packet from the network. It is addressed to my MAC address, so I'll extract the message and give it to the network layer since that is all I know how to do.
2. I send the following to the network layer:
   Received message:
   "198.110.97.7|69.59.23.70|2175|80|1|Send the conten".



Network Layer

1. I was given a packet from the link layer. Here is what it is:

src IP	dst IP	Message
198.110.97.7	69.59.23.70	2175|80|1|Send the conten

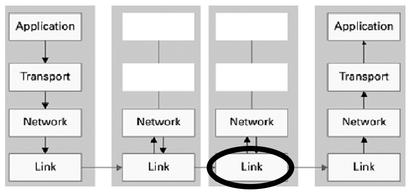


2. It is not addressed to me, so I need to forward it.
3. The destination IP is in my network, however, so I can send it to the final destination.
4. Next, I need to determine the Ethernet addresses of the source (me) and destination.
   a. My Ethernet address is 00:01:02:03:04:05.
   b. The Ethernet address of 69.59.23.70 (the current and, in this case, final destination) is 11:22:33:44:aa:bb.

   In this case, the Ethernet address and the IP address of the destination correspond, since this is the final destination.
5. Finally, I send the following messages to the **Link Layer**:
   Please send the message "198.110.97.7|69.59.23.70|2175|80|1|Send the conten" *to* 11:22:33:44:aa:bb, *and set the return address to* 00:01:02:03:04:05.

Link Layer

1. The network layer gave me a message to send.
2. I have to assemble and send the packet to the specified address. The packet is:



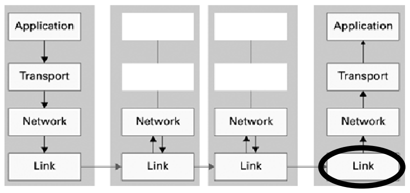
src Ethernet	dst Ethernet	Message
00:01:02:03:04:05	11:22:33:44:aa:bb	198.110.97.7|69.59.23.70|2175|80|1|Send the conten

3. I send the packet to the machine with Ethernet address 11:22:33:44:aa:bb.

On homestarrruner.com (IP 69.59.23.70, Ethernet address 11:22:33:44:aa:bb)

Link Layer

1. I received a packet from the network. All I know how to do is send and receive packets, so I'll extract the message and give it to the network layer.
2. I send the following to the network layer:
   Received message:
   "198.110.97.7|69.59.23.70|2175|80|1|Send the conten"

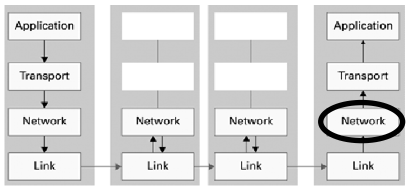


Network Layer

1. I was given a packet from the link layer. Here is what it is:

src IP	dst IP	Message
198.110.97.7	69.59.23.70	2175|80|1|Send the conten

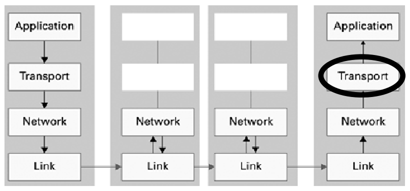
2. The packet is addressed to me, so I will extract the message and send it to the transport layer:
   Received message: "2175|80|1|Send the conten"



Transport Layer

1. A message was received from the network layer. I need to decode it. It is:

src port	dst port	Seq#	Message
2175	80	1	Send the conten



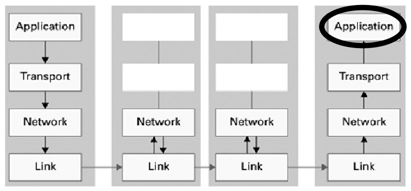
2. Good, but I need another packet. Oh, here it is:

src port	dst port	Seq#	Message
2175	80	2	ts of index.html

3. I combine the packets using the sequence numbers to get "Send the contents of index.html".
4. The message is addressed to port 80, which is the HTTP port. I will forward the message to the HTTP server application. The message is:
   Received message "Send the contents of index.html" *to port 2175 of 198.110.97.7.*

Application Layer (HTTP Server)

1. I received a request for the webpage. It came from port 2175 of the machine with IP address 198.110.97.7.
2. I need to prepare and send a response (the content of the webpage).
3. I send the following to the **Transport Layer**:
   Please send the message "Everybody! Everybody!" *to port* 2175 *of 198.110.97.7, and set the return IP address to 69.59.23.70.*



And the process continues the other way…