The Complexity of Networking: An Example

- Complexity of networking: an example
- Review: Layered communication architecture
- The TCP/IP protocol suite

© Jörg Liebeherr, 1998-1999

An Illustration of the Complexity of Networking

- I was sitting at my office machine, with name ellington.cs.virginia.edu, and I typed:
  
  ftp neon.cs.virginia.edu

What did actually happen in the network?
Step 1: A “simple” file transfer

- The ftp program of ellington now tries to establish an ftp session with the ftp program on neon.
- This is done by establishing an TCP connection to the ftp server program at neon.

```
ellington.cs.virginia.edu          neon.cs.virginia.edu
FTP program                      FTP program
+-------------------+                       +-------------------+
| FTP session       |                       | FTP session       |
+-------------------+                       +-------------------+
| TCP connection    |                       | TCP connection    |
+-------------------+                       +-------------------+
TCP
```

Step 2: Establishing a TCP connection

- Since TCP does not work with symbolic names and also would not know how to find the ftp server program at neon, two things must happen:

1. The name “neon.cs.virginia.edu” must be translated into a 4-byte long IP address
2. The ftp server at neon must be identified by a 2-byte long port number.
Step 3: Translating a name into an IP address

- The translation of the hostname *neon.cs.virginia.edu* into an IP address is done via a database lookup.

  ![Diagram](image)

  - The distributed database used is called the *Domain Name System (DNS)*.
  - All machines on the Internet have an IP address:
    - `ellington.cs.virginia.edu` 128.143.137.144
    - `athena.cs.virginia.edu` 128.143.137.11

Step 4: Finding the port number

- This is simple: Ellington knows the port number of ftp server at a remote machine.
- **Note**: All ftp servers on the Internet can be reached at port number “21”. This is called a well-known port.

- On most Unix systems, the well-known ports are listed in a file with name `/etc/services`. The well-known port numbers of some of the most popular services are:
  - `ftp` 21
  - `telnet` 23
  - `smtp` 25
  - `finger` 79
  - `http` 80
  - `nntp` 119
Step 5: Requesting a TCP Connection

- The `ftp` program at `ellington.cs.virginia.edu` requests the TCP protocol to establish a connection to port 21 of the machine with address `128.141.71.21`

Step 6: Invoking the IP Protocol

- TCP protocol at `ellington` sends a request to establish a connection to port 21 at `neon`.
- This is done by asking its local IP protocol to send an IP datagram to `128.143.71.21`
- *(The data portion of the IP datagram contains the request to open a connection)*
Step 7: Sending the IP datagram to a gateway

- *ellington* (128.143.137.144) can transmit the IP datagram directly to *neon* (128.143.71.21), only if it is on the same local network.

- However, *ellington* and *iodine* are not on the same local network (Q: How does *ellington* know this?)

- So, *ellington* sends the IP datagram to a default gateway (router) which is responsible for forwarding traffic to remote machines.
  The default gateway for *ellington* is *router137.cs.virginia.edu* (128.143.137.1).

The “route” from *ellington* to *neon*

- Note that the gateway has a different name for each of its interfaces.
Step 8: Finding the MAC address of the gateway

- To send an IP datagram to router137, ellington sticks the IP datagram in an Ethernet frame, and transmits the frame.
- However, Ethernet uses different addresses, so-called Media Access Control (MAC) addresses.
- Therefore, ellington must first translate the IP address 128.143.137.1 into a MAC address.
- The translation of addressed is performed via the Address Resolution Protocol (ARP)

![Diagram showing ARP process]

Step 9: Invoking the device driver

- The IP protocol at ellington, tells its Ethernet device driver to send an Ethernet frame to address 00:e0:f9:23:a8:20

![Diagram showing IP Protocol to Device Driver]
Step 10: Sending an Ethernet frame

- The Ethernet device driver of *ellington* sends the Ethernet frame to the Ethernet card, which transmits the frame onto the wire.

Step 11: Forwarding the IP datagram

- The gateway receives the Ethernet frame at interface *router137*, recovers the IP datagram and determines that the IP datagram should be forwarded to the interface with name *router71*.

Note: *The gateway* performs a forwarding operation.
Step 12: Another lookup of a MAC address

- The gateway needs to determine the MAC address of neon. Again, ARP is invoked, to translate the IP address of neon (128.143.71.21) into the MAC address of neon (00:20:af:03:98:28).

Step 13: Invoking the device driver once more

- The IP protocol at router71, tells its Ethernet device driver to send an Ethernet frame to address 00:20:af:03:98:28
Step 14: Sending an Ethernet frame

- The Ethernet device driver of router71 sends the Ethernet frame to the Ethernet card, which transmits the frame onto the wire.

Step 15: Data has arrived at neon

- neon receives the Ethernet frame
- The payload of the Ethernet frame is an IP datagram which is passed to the IP protocol.
- The payload of the IP datagram is a TCP segment, which is passed to the TCP protocol.

Note: Since the TCP segment is a connection request (SYN), the TCP protocol does not pass data to the ftp program for this packet. Instead, the TCP protocol at neon will respond with a SYN segment to ellington.
Wrapping-up the example

• So far, neon has only obtained a single packet
• Much more work is required to establish an actual ftp connection (or even the transfer of data)

• The example was simplified in several ways:
  – No transmission errors
  – ellington and neon are close to another
  – ellington knew how to contact the DNS server (without routing or address resolution)
  – ….

How many packets were really sent when I typed “ftp neon.cs”?

tcpdump: listening on fxp0
16:54:51.341749 128.143.137.11.53 > 128.143.137.144.1555: 1 NXDomain* 0/1/0 (98) (DF)
16:54:51.342539 128.143.137.144.1555 > 128.143.137.11.53: 2+ (41)
16:54:51.343436 128.143.137.11.53 > 128.143.137.144.1556: 2 NXDomain* 0/1/0 (109) (DF)
16:54:51.344147 128.143.137.144.1557 > 128.143.137.11.53: 3+ (38)
16:54:51.345220 128.143.137.11.53 > 128.143.137.144.1556: 3* 1/1/2 (122) (DF)

16:54:51.345220 128.143.137.11.53 > 128.143.137.144.1556: 3* 1/1/2 (122) (DF)
16:54:51.350996 arp who-has 128.143.137.1 tell 128.143.137.144
16:54:51.351614 arp reply 128.143.137.1 is-at 0:e0:f9:23:a8:20/* I have "cheated" in last two lines. The actual trace said :
16:54:51.350996 arp who-has 128.143.71.21 tell 128.143.137.144
16:54:51.351614 arp reply 128.143.71.21 is-at 0:e0:f9:23:a8:20*/

16:54:51.351712 128.143.137.144.1558 > 128.143.71.21.21: S 607568:607568(0) win 8192
<mss 1460> (DF)
16:54:51.352895 128.143.71.21.21 > 128.143.137.144.1558: 3964010655:3964010655(0)
ack 607569 win 17520 <mss 1460> (DF)
16:54:51.353007 128.143.137.144.1558 > 128.143.71.21.21: . ack 1 win 8760 (DF)
16:54:51.365603 128.143.71.21.21 > 128.143.137.144.1558: P 1:60(59)
ack 1 win 17520 (DF) [tos 0x10]
16:54:51.507399 128.143.137.144.1555 > 128.143.71.21.21: . ack 60 win 8701 (DF)
Where did data really flow?

- Even though “ellington” and “neon” are in the same building (Olsson Hall), data is traveling all over UVA.

Summary

- In this course you will learn the details of all the protocols seen in the example
- You will also learn how to setup and debug a network
- You will also learn how to collect and interpret data on a real network
- In the VINTLab, you will experiment with these protocols