

*Computer
Networks—Internet
Protocols and Routing*

Mark A. Wicks

July 6, 2000

Chapter 1
Introduction

4

1 Goals

At the completion of this course you should

- have a general understanding of the role of several layers of the OSI model,
- have a good understanding of the Internet Protocol (IP) layer,
- be able to setup simple routing configurations.
- be familiar with terminology used in internet networking.

2 Overview

This course will proceed as follows:

- We will discuss the organization of network software and the role of each software layer of a network
- We will examine the Internet Protocol layer and routing in some detail.

6

3 How is network software organized?

Network software is organized in layers with each layer playing a very specific role.

Each layer has

1. A well-defined protocol used to communicate with its peer layer.
2. A well-defined programming interface providing access to the lower layer and from the higher layer.

3. *HOW IS NETWORK SOFTWARE ORGANIZED?*

7

3.1 The OSI Model

The standard model for discussing network software is the OSI model. The OSI model has seven layers:

1. Physical Layer
2. Data Link Layer
3. Network Layer
4. Transport Layer
5. Session Layer
6. Presentation Layer
7. Application Layer

3.2 The TCP/IP Model

There's a TCP/IP model which doesn't strictly follow the OSI model. TCP and IP are two important protocols used on the Internet. Roughly speaking, the Internet protocols uses 5 layers similar to those in the OSI model. The session layer and presentation layer do not appear as distinct layers in the internet model.

The organization of these layers along with sample protocols and interfaces is shown in the following figure:

3. HOW IS NETWORK SOFTWARE ORGANIZED?

9

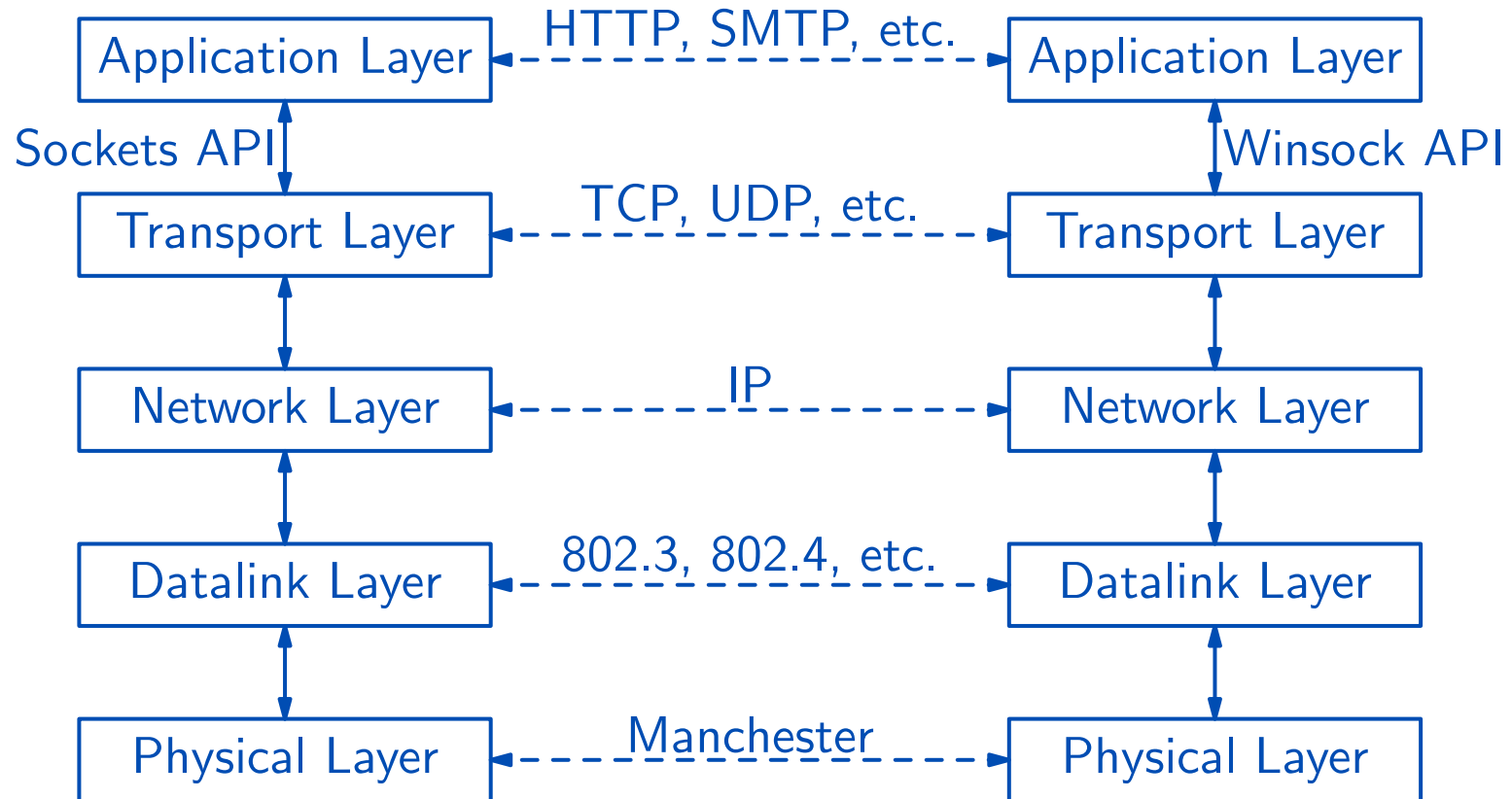


Figure 1.1: Reference Model for Software Layers used on the Internet

Keep in mind the following points:

- Horizontal communication is “virtual.” Each layer *appears* to talk to its peer entity in the corresponding layer at the remote end.
- True communication is between adjacent vertical layers.

For example:

- A Web browser at the application layer appears to talk to a Web server also running at the application layer, but in reality it talks to the transport layer below it.
- A person on the phone appears to talk to the person at the other end of the line, but in reality he’s talking to telephone.

3. *HOW IS NETWORK SOFTWARE ORGANIZED?* 11

In the previous example, the telephone system is analogous to the “datalink layer”. It has a well-defined “interface”. The operator supplies an address (the telephone number) to the physical layer. The communication between speakers is horizontal and occurs a higher layer. The speakers use a protocol to speak to each other. For example, the person receiving the call usually says “hello” as part of the protocol.

3.3 Why Use Layers?

If the interface to a layer is well-defined, that layer (and its protocols) can be replaced at any time without breaking higher layers. The protocols may change, but the services offered should not. For example, PPP and Ethernet (802.3) are both datalink protocols. Either may be interchanged without breaking the IP layer. As long as the IP interface can instruct a datalink layer to send a frame over a physical interface, the physical nature of that interface is irrelevant. It could be serial, parallel, broadcast, point-to-point, electrical, or optical.

3. HOW IS NETWORK SOFTWARE ORGANIZED?

13

3.4 Summary

- *Network software is organized in layers.*
- *Horizontal communication uses “protocols.”*
- *Each layer offers well-defined services to the layer above it via well-defined interfaces.*
- *Services have addresses in each layer.*

3.5 Questions

1. In the telephone analogy, the telephone number is a layer 2 address. It connects you to the right household. What “address” might you need at the next “layer?”
2. Continuing the telephone analogy, is a person-to-person “service” at a higher layer than the layer having phone numbers as addresses? Why? What datalink layer address do you use for a person-to-person call?
3. Discuss how touchtone phone service represents a change in the physical layer protocol that does not alter the services or interface to the higher layers.

4 What does each layer do?

4.1 The Physical Layer

The *physical layer* is the interface to the physical medium. As part of the protocol both sides agree on what signal represents a 0 or 1. In some systems the amplitude of a voltage distinguishes 0 from 1. A 0 could be 0 V and a 1 could be 5 V. In other systems, the phase shift of a sinewave may distinguish a 0 from a 1. The physical layer is responsible for interpreting physical phenomena (e.g., voltage fluctuations) as 0s and 1s. The higher layers don't know and don't care how a 0 and how a 1 are transmitted.

16

Note: Physical layers come in two major varieties:

- Broadcast (shared)
- Point-to-point

Some physical layer protocols: Manchester encoding, RS-232, RS-422, V.34.

4.2 The Datalink Layer

One purpose of the datalink layer is to provide framing for the physical layer. In the case of broadcast media, the *datalink layer* controls media access control (MAC) and may provide error detection or correction. Why is MAC necessary? Broadcast media are shared, so there must be some protocol to provide orderly transmission of data. A datalink protocol for a broadcast medium must

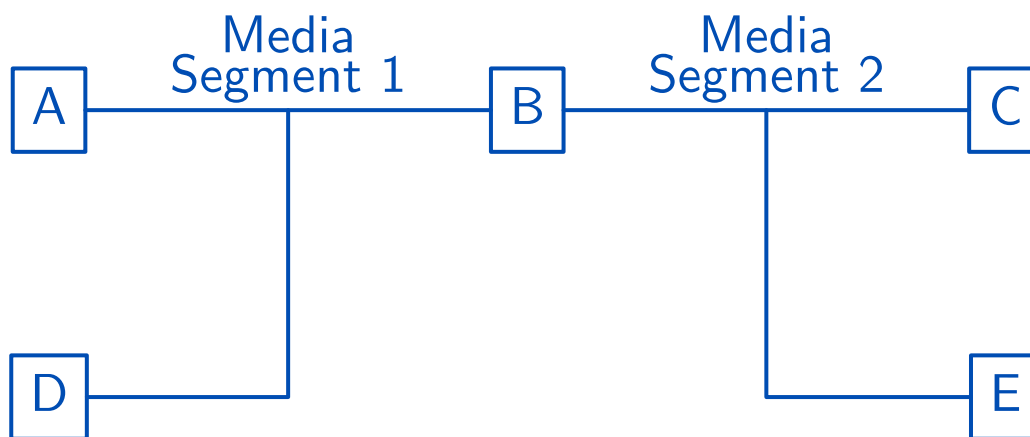
- grant permission for each node to speak at an appropriate time, and
- identify the intended recipient of a message.

Some datalink protocols: PPP, Ethernet (802.3), Token Ring (802.4)

4.3 The Network/Internet Layer

The *network layer* is responsible for getting data from one media segment to another, and eventually for getting data to its final destination (that's routing!)

Consider the following situation:



In this configuration, A, B, and D share a single medium. If any of A, B, or D begins transmitting, all three will hear the transmission. Likewise, B, C, and E share a different medium. A protocol is needed

4. *WHAT DOES EACH LAYER DO?*

19

that allows **B** to relay messages on behalf of **A**, **C**, **D**, and **E**. That is what routers do.

Examples of network protocols: IP, IPX.

Note. We don't necessarily need a new protocol to speak to nodes on a different segment. **B** could simply forward every frame it sees on every interface (flooding) or it could try to learn the location of each MAC address and forward only when necessary (bridging/switching). What are the advantages and disadvantages of these two approaches?

4.4 The Transport Layer

The *transport layer* is responsible for providing *reliable* connections, and optionally *connection-oriented* services. The network layer may or may not be reliable and may or may not be connection-oriented. For a connection-oriented service, the transport layer will

- break streams of data into packets which can be handled by the network layer,
- reassemble and reorder packets that arrive out of order, and
- request retransmission of missing packets or packets received with errors,
- provide flow control to prevent overloading slow receivers.

Transport layer protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

Some terminology

What is a connection-oriented service? A connection-oriented service appears to be a dedicated connection once it's been established. It's as if a wire (which may be virtual) has been strung between the two endpoints of the connection. The destination address is used to setup the connection, but is not once the connection is established. In a connection-oriented service, data may be expected to arrive in the same order it was sent (or not at all). The telephone system is a connection-oriented service.

What is a connectionless service? A connectionless service requires no initial setup. Each packet of data is independent of other data packets sent to the same recipient. Packets must be addressed individually. The packets may arrive out of order. The US mail system is a connectionless service.

What is an acknowledged service? An acknowledge service is one where receipt of data is acknowledged. The US mail system can provide either acknowledged or unacknowledged service. An unacknowledged connection-oriented service would not be of much user, so the words “acknowledged” or “unacknowledged” usually pertain to connectionless services.

Note: The TCP protocol provides a connection-oriented service in the transport layer. The UDP protocol provides a connectionless service in the transport layer. The IP protocol and the Ethernet protocol are both unacknowledged and connectionless.

4.5 The Application Layer

The *application layer* provides applications to the user.

Sample application protocols: Simple Mail Transport Protocol (SMTP), Simple Network Management Protocol (SNMP), HyperText Transport Protocol (HTTP), Telnet, Network News Transport Protocol (NNTP).

Chapter 2
A Brief Discussion of the
Datalink Layer

1 Common Frame Formats

The datalink layer provides framing and media access control (if needed) for the physical layer. Typical datalink protocol frames have headers identifying the sender and recipient of the frames. For example, PPP and Ethernet (802.3) both have the following frame format:



1. The header and trailer depend on the protocol.
2. The payload is anything the network layer chooses to put there.

1.1 PPP Frame Format

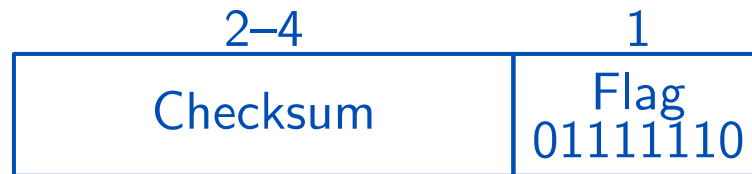
In the case of PPP, the format of the header is

1	1	1	1-2
Flag 01111110	Addr 11111111	Control 00000011	Protocol

- The Flag field is for framing.
- The Address field is set to ones indicating that anybody may receive the packet (PPP isn't for broadcast or shared media).
- The control field shown is the default for unacknowledged, connectionless service.
- The protocol identifies the particular network layer protocol contained in the payload area of the frame. PPP can support IP, IPX, and AppleTalk, among others.

The format of the trailer is

28

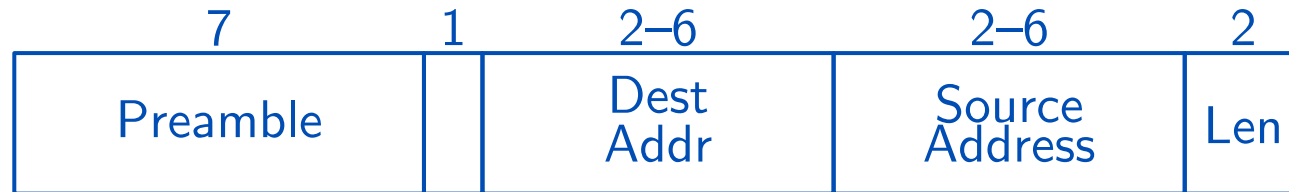


Again, the Flag is for framing. The checksum is for error detection.

1. COMMON FRAME FORMATS

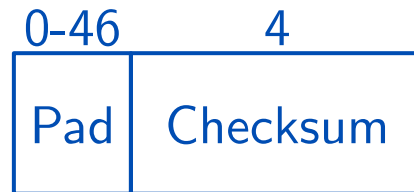
1.2 Ethernet (802.3) Frame Format

In the case of Ethernet, the format of the header is



The preamble and delimiter fields are for framing. The destination and source fields identify the intended recipient and the originator. A source address that is to all ones (**FF:FF:FF:FF:FF:FF**) indicates that anybody may receive the packet. A polite ethernet interface honors the destination field and will not pass the frame to the host CPU unless the destination field matches the programmed MAC address for that Ethernet adaptor. The Length field identifies the length of the payload to be expected.

The format of the trailer is



The minimum Ethernet frame length is 64 bytes, so the pad field is used if the payload is smaller than 46 bytes.

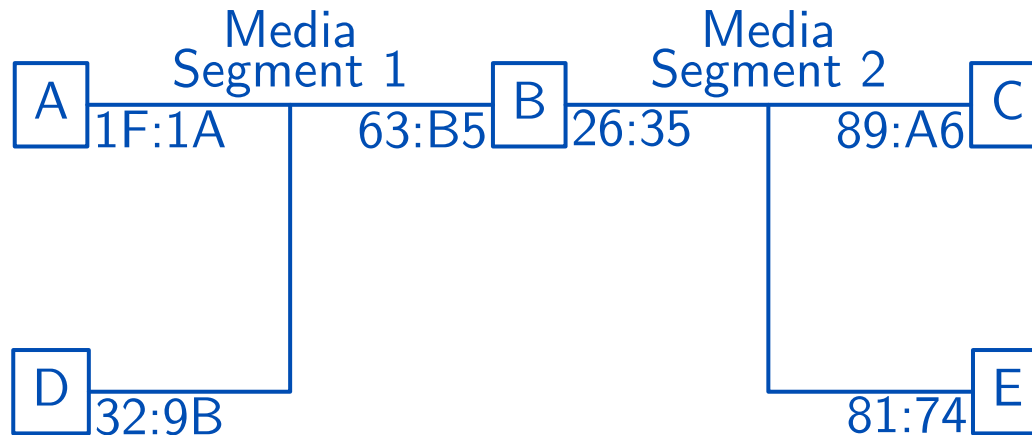
The maximum payload length is 1500 bytes.

An ethernet interface that ignores the destination field and received all frames is said to be *promiscuous*. Promiscuous interfaces are used for in sniffers because they observe all frames whether intended for them or not.

Chapter 3
Basic Network Layer
Operation

1 How does the network layer work?

Consider the following simple network:



The MAC addresses have been shortened to two bytes for convenience. Let's consider several scenarios:

Remember. Two nodes can communicate using the physical layer and the datalink layer only if they are on the same physical media segment.

Scenario 1—The datalink layer on **A** wants to send a message to the datalink layer on **D**. In principle, this presents no difficulty. If A knows the MAC address of D, A simply sends the message using the datalink layer.

Scenario 2—The datalink layer **A** wants to send a message to the datalink layer on **B**. Again, this presents no difficulty. If A knows the MAC address of the left interface of **B**, A simply sends the message using the datalink layer.

Scenario 3—The datalink layer on **A** wants to send a message to the datalink layer on **E**. *This is impossible! The datalink layer protocols used on the internet are not routable!*

Scenario 4—The IP layer on **A** wants to send a message to the network (IP) layer on **E**. This is possible. For this to

happen, **A**, **E**, and **B** must cooperate:

- **A** must know that **E** can hear **B**.
- **A** must *address a message to B*, and put the message intended for **E** *inside* the message it sends to **B**.
 - The message from **A** to **B** is transmitted using the datalink layer, probably with MAC addresses.
 - The enclosed message from **A** intended for **E** is a network layer message (an IP message), with its own different addressing scheme.

1. HOW DOES THE NETWORK LAYER WORK?

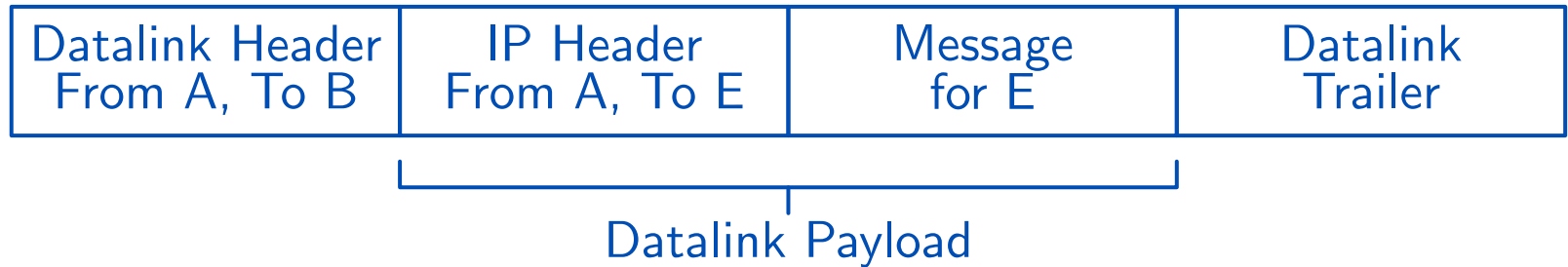
35

Here's how the different software layers view this transaction:

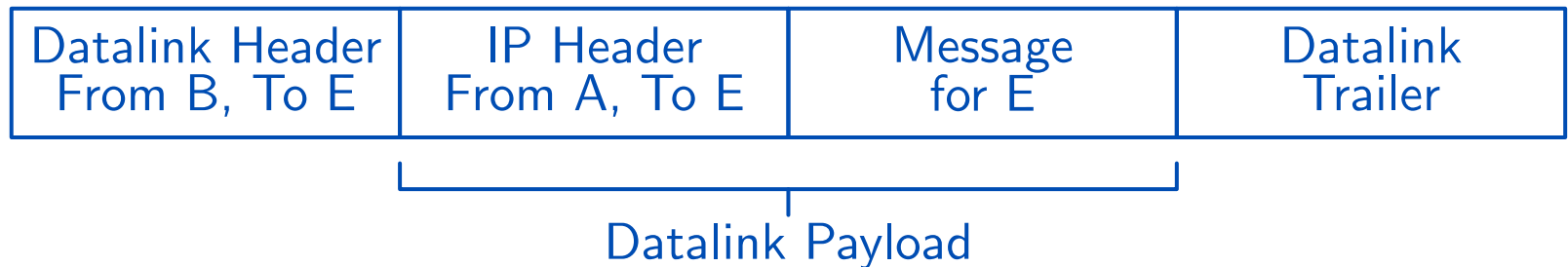
- The network layer software on **A** believes it sent a network layer message to **E** over a virtual channel using the IP protocol. It used a service from the datalink layer to do so. It put its own header and addressing information on the message before passing it to the datalink layer:



- The datalink layer on **A** doesn't know anything about IP. It simply forwards a message without knowledge of the contents to **B** because it knows how to do that. It added a datalink header to the message:



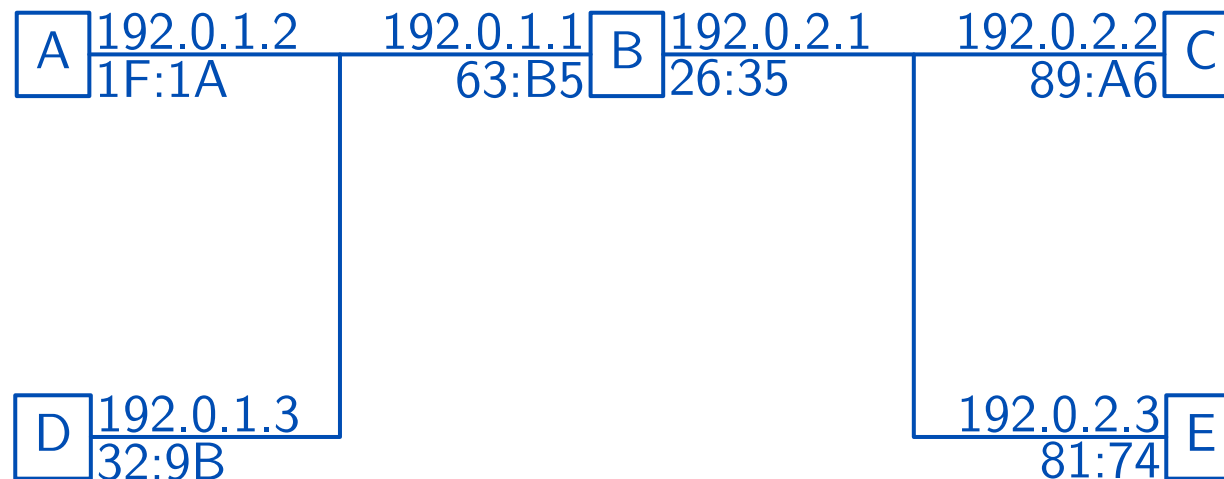
- When **B** receives the frame, the datalink layer on **B** strips off the datalink header and passes the IP packet to the network layer. The IP layer sends it right back to the datalink layer which adds a datalink header addressed to **E**:



2 Address Resolution

In the discussion of Scenario 4 in the previous section, we omitted one very important detail. The IP layer on **A** has to provide the MAC address of **B** to the datalink layer on **A**. The IP layer certainly knows the IP address of **E**, and possibly the IP address of **B**, but how does it determine the datalink (MAC) address of **B**?

Let's consider the same network with IP addresses and MAC addresses shown:



38

Scenario 5—The IP layer on **A** wants to send a packet to the IP layer on **D**. While this message can be sent using the datalink layer, **A** must first identify the MAC address of **D**.

How does **A** determine the MAC address of **D**?

- The IP layer on **A** tells the datalink layer on **A** to send a message addressed to *all machines* on the shared medium. All machines can be addressed with **FF:FF:FF:FF:FF:FF**).
- That packet is an Address Resolution Protocol (ARP) request packet containing the IP address of **D**—**192.0.1.3**.
- Any host that received the packet that is using the IP address in the ARP packet, in this case **D**, is supposed to send an ARP response packet to the originator of the ARP request, informing the requester of the MAC address, in this case **32:9B**.

- At this point, **A** knows what MAC address to use to send packets to **192.0.1.3**.
- Host **A** tries to remember the MAC address that goes with **192.0.1.3** by placing **D**'s MAC address in its own *ARP cache*. After a while, if it doesn't need to send anything to **D**, it will remove the entry from the cache.

2. ADDRESS RESOLUTION

41

Back to Scenario 4. Let's revisit Scenario 4, with the IP addresses filled in. The IP layer at address 192.0.1.2 wants to talk to the IP layer at address 192.0.2.3. For the moment, assume that **A** knows that **B** can reach **E**. (For the moment, don't worry about how it knows.) How does **A** get a packet to **E**?

1. **A** knows it must send its IP layer packet destined for **E** in a datalink frame addressed to **B**. If **A** doesn't already have an ARP entry for **B**, it will send an ARP request for the address of **B** to the MAC broadcast address and wait for an ARP response from **B**.
2. Once the ARP entry has been created, **A** simply sends the packet to **B**. The packet headers have the following contents:

MAC Header:	Source	1F:1A
	Destination	63:B5
IP Header:	Source	192.0.1.2
	Destination	192.0.2.3

3. **B** will receive the frame, since the frame is addressed to **B**. **B** will strip off the MAC header and examine the IP destination, which is 192.0.2.3. **B** will examine its ARP tables to determine if there is an entry for **E**. If the ARP entry doesn't exist, **B** will send an ARP request and wait for a response.
4. Once the ARP entry has been created, **B** simply sends the packet to **E**. The packet headers have the following contents:

MAC Header:	Source	26:35
	Destination	81:74
IP Header:	Source	192.0.1.2
	Destination	192.0.2.3

2. ADDRESS RESOLUTION

43

ARP in action

```
14:01:36.934534 0:c0:4f:d:68:7d ff:ff:ff:ff:ff:ff 0806 60: arp who-has 198.110.4.1 tell 198.110.4.97
14:01:36.944535 0:0:a2:c3:12:e0 0:c0:4f:d:68:7d 0806 60: arp reply 198.110.4.1 is-at 0:0:a2:c3:12:e0
14:01:36.944535 0:c0:4f:d:68:7d 0:0:a2:c3:12:e0 0800 98: 198.110.4.97 > 192.138.137.2: icmp: echo request
```

44

Who's been pinging me?

Anybody who pings you generates an ARP entry in your arp cache, if they are on the same media segment:

```
[mwicks@gaspra mwicks]$ /sbin/arp -n
Address          HWtype  HWaddress          Flags   Mask    Iface
198.110.4.84     ether   00:60:08:C6:3D:5E   C           eth0
198.110.4.1      ether   00:00:A2:C3:12:E0   C           eth0
198.110.4.78     ether   00:00:C0:76:45:77   C           eth0
```

Questions to think about

- Suppose that **A** lies about its IP address and puts 192.0.1.3 in the source field of the IP header. Do you think the packet will still arrive at **E**? Can **B** detect the deception. Assuming **B** passes the packet, can **E** detect the deception?

46

- Reconsider the previous question. How do things change if **A** puts 10.0.0.1 in the source field of the IP header? Can **E** detect the deception. Can **B** detect the deception?

3 Network Addressing

How does **A** know that **B** can get packets to **E**? The answer is that **A** has a routing table that tells it which machines can get to which addresses.

We will examine routing tables shortly, but prior to that discussion, we need to have a good understanding of terminology and notation used for specifying IP addressing.

IP addresses

- IP addresses are 4-byte numbers.
- A byte can store the numbers 0 through 255 inclusive.
- An IP address is written as four numbers (each representing one of the bytes) separated by periods. Each number is between 0 and 255 inclusive.

3. NETWORK ADDRESSING

49

Questions

1. Which of the following are possible IP addresses?
 - 10.0.0.1
 - 101.202.303.010
 - 192.255.255.100
2. How many possible IP addresses are there?

Binary number review

- Convert the following numbers to binary: 192, 138, 137, and 2.

Answers: **11000000, 10001010, 10001001, 00000010**

In other words, the IP address 192.138.137.2 can be thought of as

11000000.10001010.10001001.00000010

- Convert the number 255 to binary:

Answer: **11111111.**

- So what's 255.255.255.0?

Answer: **11111111.11111111.11111111.00000000.**

ANDs and ORs

The AND of two binary digits is 1 if and only if *both* binary digits is 1.

The OR of two binary digits is 1 if and only if *either* binary digit is 1.

- What is 1100000000 AND 11111111?

Answer: 1100000000.

- What is 10001010 AND 10101010?

Answer:

- What is 192 AND 255?

Answer: 192.

- What is n AND 255?

Answer:

- What is 192.138.137.2 AND 255.255.255.0?

52

Answer:

What is 198.110.1.35 AND 255.255.255.224?

Answer:

Network addresses and netmasks

A network address/netmask pair is a way of addressing a large number of IP addresses (often representing a single media segment) simultaneously. A network is completely specified by *two* numbers that resemble IP addresses. The first address is called the network address. The second address is called the netmask.

Here's the key point to remember:

Two IP addresses belong to the same network (which means they can reach each other using datalink protocols) if and only if the bitwise AND of the IP address and the netmask is equal to the network address.

Another way to say this is:

An IP address belongs to a network's address space (which usually means that other hosts on that network can reach it using datalink protocols) if and only if the bitwise AND of the

IP address and the netmask is equal to the network address.

Note. The first and last addresses normally are not be used by hosts. Usually the last address is used as the IP broadcast address (this is different from the MAC broadcast address).

Problems.

1. For each of the following pairs of IP addresses, state whether the two addresses are on the same network:
 - 192.138.137.2 and 192.138.137.6 with netmask 255.255.255.248
 - 192.138.137.2 and 192.138.137.129 with netmask 255.255.255.248
 - 192.138.137.255 and 198.110.4.255 with netmask 0.0.0.255

2. For each of the following network address and netmask pairs, what range of IP addresses belong to the network's address space?
- Network address 198.110.4.0 and netmask 255.255.255.0.
 - Network address 198.110.4.32 and netmask 255.255.255.224.
 - Network address 0.110.5.198 and netmask 0.255.255.255
 - Network address 0.0.0.255 and netmask 0.0.0.255
 - Network address 198.110.4.32 and netmask 255.255.255.252

Default netmasks

The old style on the internet was to *infer* the netmask based on the first byte of the IP address. This is done as follows:

First Byte	Default Netmask	Class	Number	Size
1–127	255.0.0.0	A	126	16777214
128–191	255.255.0.0	B	16382	65534
192–223	255.255.255.0	C	2097152	254

Classless routing

The internet has moved toward classless routing. In classless routing, the first byte of the IP address provides no information about the netmask. Using classless routing, it is possible to define subnets of varying sizes. The size could be between that of a class C network and a class B network, between that of a class A and B, or smaller than a class C network.

Note. Some older routers and older routing protocols may not understand classless routing.

Netmask shorthand

Since netmasks almost always start with a long string of ones, when written as binary numbers, a shorthand has evolved for specifying the netmask. This notation uses the network address followed by a slash (/), followed by a number between 0 and 32. The number following the slash indicated the number of ones in the netmask, when written as a binary number.

Examples

- Network 10.1.2.0 with netmask 255.0.0.0 may be written as 10.1.2.0/8.
- Network 192.10.15.0 with netmask 255.255.255.0 may be written as 192.10.15.0/24.

In other words, class A equates to “/8”; class B equate to “/16”, and class C to “/24.”

Questions

- How many nodes should be put on a network with the network address 198.110.0.0/21? What should be the first address available for hosts on the network? What should be the last address available for hosts on the network?
- How many nodes can be put on a network with the address 198.110.4.0/27? What is the address of the first addressable host? What is the address of the last addressable host? What is the netmask?
- Repeat for the address 212.108.10.32/30. What is the netmask?

62

- Repeat for 0.0.0.0/0?
- Repeat for 212.108.10.37/32.
- What is wrong with the network address 212.108.10.56/28?

4 Routing

Let's return to the question, "How does **A** know that **B** can get packets to **E**?"

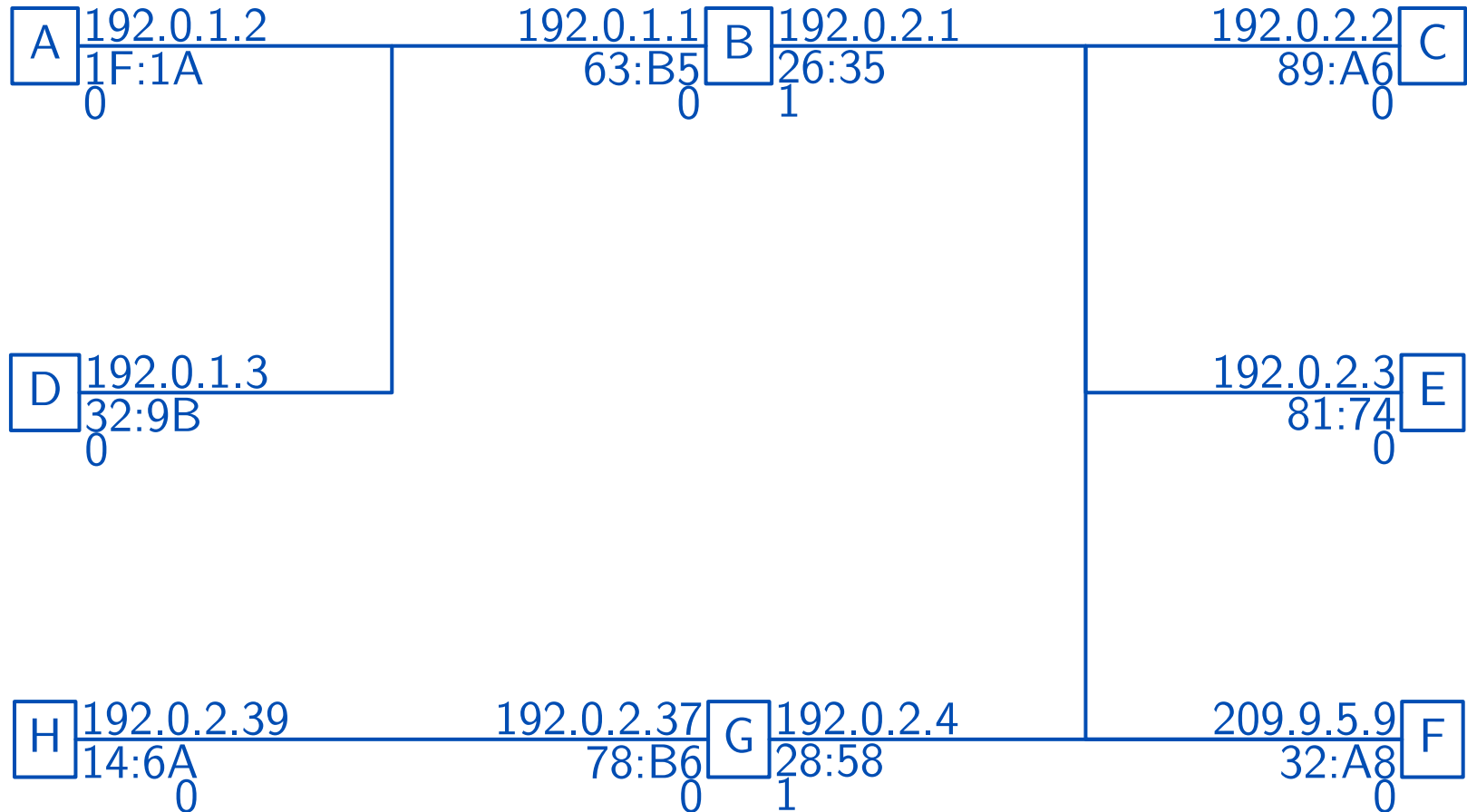
We are now equipped to discuss the routing tables that allow **A** to make this determination.

Every host has a routing table. The routing table tells that host how to get packets that are addressed to other hosts on various networks. Each entry in the routing table usually contains:

- a network address,
- a *gateway* or "next hop" required to reach that network address, and
- the interface on which to send the frame.

64

Let's build routing tables for each host on the following network:



4. ROUTING

65

Let's start with A's and D's routing table:

Network/Netmask	Gateway (Next hop)	Interface
192.0.1.0/255.255.255.0	–	0
192.0.2.0/		
209.9.5.9/		

66

Here's B's routing table:

Network/Netmask	Gateway (Next hop)	Interface
192.0.1.0/255.255.255.0	-	0
192.0.2. /		
192.0.2. /		
209.9.5.9/		

4. ROUTING

67

C, E, and F will have the same routing table:

Network/Netmask	Gateway (Next hop)	Interface
192.0.1.0/255.255.255.0		0
192.0.2. /		
192.0.2. /		
209.9.5.9/		

68

G's routing table:

Network/Netmask	Gateway (Next hop)	Interface
192.0.1.0/255.255.255.0		
192.0.2. /		
192.0.2. /		
209.9.5.9/		

4. ROUTING

69

And finally H's routing table:

Network/Netmask	Gateway (Next hop)	Interface
192.0.1.0/255.255.255.0		
192.0.2. /		
192.0.2. /		
209.9.5.9/		

Default routes

Default routes greatly simplify routing tables in most cases. A default route matches *all* network addresses. Let's redo the previous routing tables using default routes:

A's and D's routing table:

Network/Netmask	Gateway (Next hop)	Interface
192.0.1.0/255.255.255.0 0.0.0.0/0.0.0.0		

4. *ROUTING*

71

B's routing table:

Network/Netmask	Gateway (Next hop)	Interface
192.0.1.0/255.255.255.0		

72

Are C, E and F's routing tables any simpler using default routes?

4. *ROUTING*

73

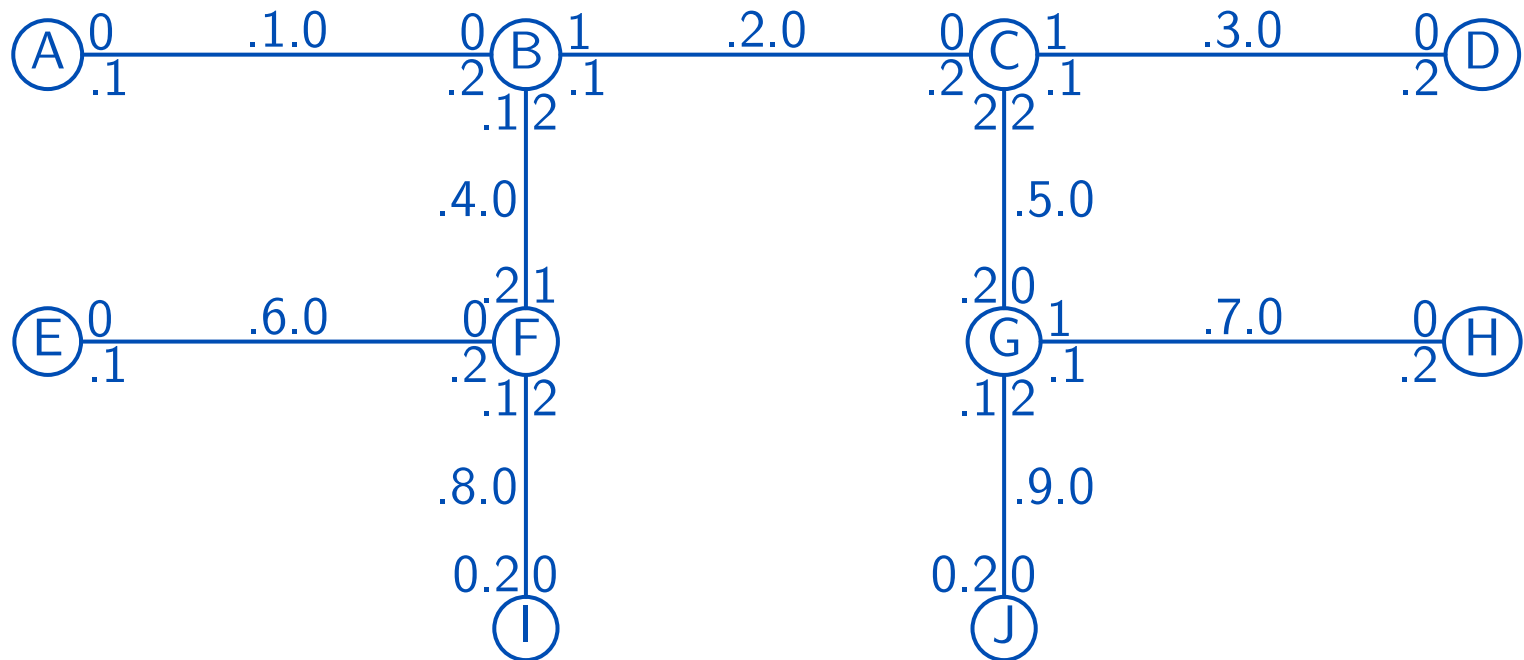
What about H's routing table?

Network/Netmask	Gateway (Next hop)	Interface
192.0.2. / 0.0.0.0/0.0.0.0		

Chapter 4
More about routing

1 An Example

Consider the following network, where the leading three digits of the IP addresses and the leading two digits of the network address have been omitted to avoid cluttering the diagram.



1. AN EXAMPLE

77

Again, let's build routing tables for the hosts on the left half of the network:

Let's start with **A**'s routing table:

Network/Netmask	Gateway (Next hop)	Interface
.1.0/.255.0	–	0
default	.1.2	0

E's routing table is easy:

Network/Netmask	Gateway (Next hop)	Interface
.6.0/.255.0	–	0
default	.6.2	0

And so is **J**'s:

Network/Netmask	Gateway (Next hop)	Interface
.8.0/.255.0	–	0
default	.8.1	0

78

B's routing table:

Network/Netmask	Gateway (Next hop)	Interface
.1.0/.255.0	—	0
.2.0/.255.0	—	1
.4.0/.255.0	—	2
.6.0/.255.0	.4.1	1
.8.0/.255.0	.4.1	1
default	.2.2	1

1. *AN EXAMPLE*

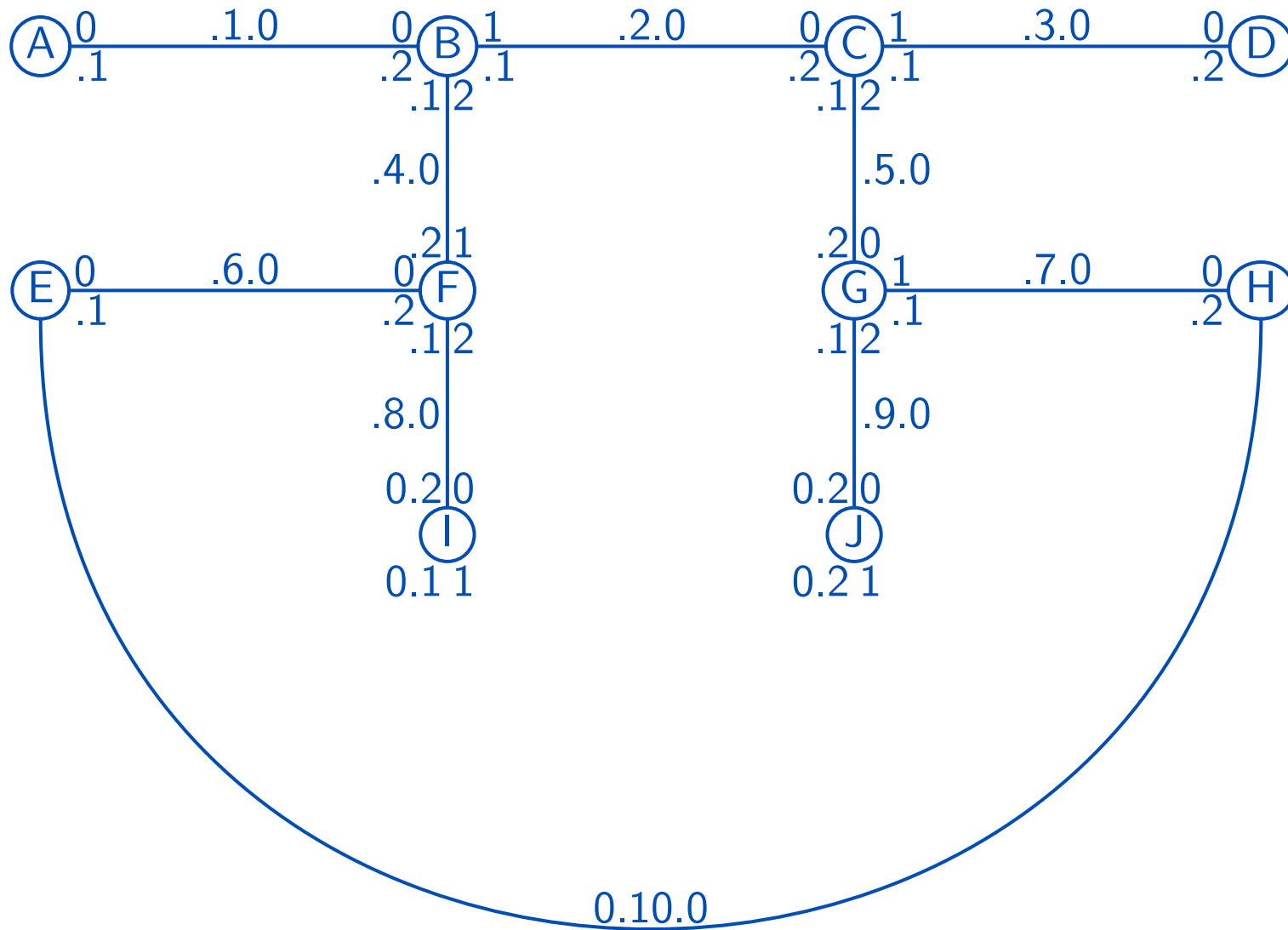
F's routing table:

Network/Netmask	Gateway (Next hop)	Interface
.6.0/.255.0	—	0
.4.0/.255.0	—	1
.8.0/.255.0	—	2
default	.4.1	1

80

Let's add an additional link to the network to see how the routing changes?

1. AN EXAMPLE



82

A's routing table presents no trouble:

Network/Netmask	Gateway (Next hop)	Interface
.1.0/.255.0	–	0
default	.1.2	0

Neither does J's:

Network/Netmask	Gateway (Next hop)	Interface
.8.0/.255.0	–	0
default	.8.1	0

1. *AN EXAMPLE*

83

E's routing is trickier. We will add an additional column called the "metric," which, for the purpose of this example, indicates the number of routers that must be crossed to get to the target network:

Network/Netmask	Gateway (Next hop)	Interface	Metric
.6.0/.255.0	—	0	0
.10.0/.255.0	—	1	0
.1.0/.255.0	.6.2	0	2
.1.0/.255.0	.10.2	1	4
.2.0/.255.0	.6.2	0	2
.2.0/.255.0	.10.2	1	3
.3.0/.255.0	.6.2	0	3
.3.0/.255.0	.10.2	1	3
.4.0/.255.0	.6.2	0	1
.4.0/.255.0	.10.2	1	4
.5.0/.255.0	.6.2	0	3
.5.0/.255.0	.10.2	1	3
.7.0/.255.0	.6.2	0	5
.7.0/.255.0	.10.2	1	1
.8.0/.255.0	.6.2	0	1
.8.0/.255.0	.10.2	1	6
.9.0/.255.0	.6.2	0	4
.9.0/.255.0	.10.2	1	2

1. AN EXAMPLE

85

If we eliminate the costlier routes we obtain:

Network/Netmask	Gateway (Next hop)	Interface	Metric
.6.0/.255.0	—	0	0
.10.0/.255.0	—	1	0
.1.0/.255.0	.6.2	0	2
.2.0/.255.0	.6.2	0	2
.3.0/.255.0	.6.2	0	3
.3.0/.255.0	.10.2	1	3
.4.0/.255.0	.6.2	0	1
.5.0/.255.0	.6.2	0	3
.5.0/.255.0	.10.2	1	3
.7.0/.255.0	.10.2	1	1
.8.0/.255.0	.6.2	0	1
.9.0/.255.0	.10.2	1	2

This is the routing table we would expect **E** to use under normal circumstances.

2 Routing protocols

Rather than enter tables such as required for the previous example by hand (so called *static routing*), we would like to use a *routing protocol* to find and to update the table entries automatically as networks are added and removed. The next section explains how such routing protocols work. A *routing protocol* is simply a language that nearby routers use to exchange routing information.

2. ROUTING PROTOCOLS

87

Common routing protocols used on the internet are

- RIP
 - Mainly an “interior” protocol
 - Old and falling out of favor.
 - All packets use the best route—in other words, all packets use the *same* route.
 - Distance Vector Routing—Slow at recovering from link failures
 - Easily fooled.
- OSPF
 - Interior protocol that scales well
 - Secure
 - Routes packets different depending on type of service wanted
- BGP

88

- Exterior protocol
- Policy based routing

3 Distance Vector Routing

Distance vector routing is one of the earliest methods for generating and updating routing table entries. This method is used by RIP and in a modified form by BGP. BGP and RIP are two common internet routing protocols.

Assuming the “distance metric” is simply the hop count, the method works like this:

- Each router’s table is initialized with routes to each of its directly attached networks using a metric of zero.
- Each router tells its “neighbor” routers all the networks it knows about along with the distance of that router (or metric) from that network.
- Each router that receives a route from a neighbor adds one to the metric for every route that it receives.
- If the metric for a received route for a network is smaller than the metric for that network currently in the router’s routing table, the newly received route replaces the route in the routing table. If the newly received route metric for a network is larger than one the router already knows about, the newly received route is ignored.

3. DISTANCE VECTOR ROUTING

91

Let's observe this in action. The routings tables are initialized as follows: B's table is

Network/Netmask	Gateway (Next hop)	Interface	Metric
.2.0/.255.0	—	0	0
.1.0/.255.0	—	1	0
.4.0/.255.0	—	2	0

C's table is

Network/Netmask	Gateway (Next hop)	Interface	Metric
.2.0/.255.0	—	0	0
.1.0/.255.0	—	1	0
.5.0/.255.0	—	2	0

E's table is

Network/Netmask	Gateway (Next hop)	Interface	Metric
.6.0/.255.0	—	0	0
.10.0/.255.0	—	1	0

F's table is

Network/Netmask	Gateway (Next hop)	Interface	Metric
.6.0/.255.0	—	0	0
.4.0/.255.0	—	1	0
.8.0/.255.0	—	2	0

G's table is

Network/Netmask	Gateway (Next hop)	Interface	Metric
.5.0/.255.0	—	0	0
.7.0/.255.0	—	1	0
.9.0/.255.0	—	1	0

H's table is:

Network/Netmask	Gateway (Next hop)	Interface	Metric
.7.0/.255.0	—	0	0
.10.0/.255.0	—	1	0

3. *DISTANCE VECTOR ROUTING*

93

During the first cycle, **B** receives routes from **C** and **F**.

B's table is:

Network/Netmask	Gateway (Next hop)	Interface	Metric
.2.0/.255.0	—	0	0
.1.0/.255.0	—	1	0
.4.0/.255.0	—	2	0
.3.0/.255.0	.2.2	1	1
.5.0/.255.0	.2.2	1	1
.6.0/.255.0	.4.2	2	1
.8.0/.255.0	.4.2	2	1

C receives updates from **B** and **G** and generates its table:

Network/Netmask	Gateway (Next hop)	Interface	Metric
.2.0/.255.0	—	0	0
.3.0/.255.0	—	1	0
.5.0/.255.0	—	2	0
.1.0/.255.0	.2.1	0	1
.4.0/.255.0	.2.1	0	1
.7.0/.255.0	.4.2	2	1
.9.0/.255.0	.4.2	2	1

3. *DISTANCE VECTOR ROUTING*

95

Eventually, this process stops when B's table is

Network/Netmask	Gateway (Next hop)	Interface	Metric
.2.0/.255.0	—	0	0
.1.0/.255.0	—	1	0
.4.0/.255.0	—	2	0
.3.0/.255.0	.2.2	1	1
.5.0/.255.0	.2.2	1	1
.6.0/.255.0	.4.2	2	1
.7.0/.255.0	.2.2	1	2
.8.0/.255.0	.4.2	2	1
.9.0/.255.0	.2.2	1	2
.10.0/.255.0	.4.2	2	2

and C's table is

Network/Netmask	Gateway (Next hop)	Interface	Metric
.2.0/.255.0	—	0	0
.3.0/.255.0	—	1	0
.5.0/.255.0	—	2	0
.1.0/.255.0	.2.1	0	1
.4.0/.255.0	.2.1	0	1
.6.0/.255.0	.2.1	0	2
.7.0/.255.0	.5.2	2	1
.8.0/.255.0	.2.1	0	2
.9.0/.255.0	.5.2	2	1
.10.0/.255.0	.5.2	2	2

At this point, **B** will refuse to consider **C**'s route to .6.0, .8.0 or .10, because the metrics are farther (after adding one) than those already in **B**'s table.

3.1 Disadvantages of Distance Vector Routing

Count-to-infinity problem. In the previous example, if the link between **B** and **C** breaks, **E** will believe it has a route to .3.0 through **F**. It will advertise this to **F**, who will believe **E**, even though the advertised route is via **F**! **F** has no way of knowing this. **F** will increase its count for .3.0 and set the gateway to .3.0 to **E**! It will advertise this new route. **E** will hear the advertisement and will believe it, even though the route is via **E**, and so on. For this reason, distance vector routing is very slow in removing bad routes.

There are various partial work-arounds for this problem called “split-horizon” and “poisoned reverse.” Link-state routing protocols (such as OSPF) avoid this problem.

3.2 Link State Protocols

Instead of advertising all the routes it knows, a link state protocols advertises only the list of its neighbors to its neighbors. Every neighbor advertises the list of its neighbors' neighbors to it neighbors, and so on. Once all the routers on the network know who is adjacent to whom, they can draw a map of the network and determine the optimum route locally. This approach requires more memory and CPU time then distance vector routing.

3. *DISTANCE VECTOR ROUTING*

Chapter 5
More about the network
layer

1 What's in an IP packet?

The IP Header

As discussed in Chapter 3 the network layer adds its own header with the information it needs to route packets. In Chapter 3, only sketchy details were given as to the contents of the IP header.

1. WHAT'S IN AN IP PACKET?

103

The *minimum* contents of the IP header are shown in the following table:

Version	IHL	TOS	Total length
Identification		Fragmentation info	
TTL	Protocol	Header checksum	
Source IP address			
Destination IP address			

- Each “row” of the table corresponds to a 4 byte word of information.
- All IP packets have at least 5 words (rows) (20 bytes of information).
- Additional “rows” are optional.

The fields are defined as follows:

- *Version*—For IP version 4, the version field is simply **4**.
- *IHL*—The Internet Header Length (IHL) field represents the number of words (rows) in the IP header. Often this number is 5, so the first byte of an IPV4 packet is often **45**.
- *TOS*—The Type of Service (TOS) field can be used to route packets differently based on priority, reliability, bandwidth, and other considerations. Most routers ignore this field, so it is likely to be set **0**.
- *Total length*—The Total length represents the complete length of the packet with header and data combined. Because this field is only two bytes long, the maximum IP packet size is limited to 65,535.

1. WHAT'S IN AN IP PACKET?

105

- *Identification*—The Identification field is used for defragmenting packets. When routers cannot handle a full sized packet, they are allowed to break into smaller pieces called fragments. All fragments of a packet contain the same value in this field.
- *Fragmentation Information*—This field contains information required to reconstitute a fragmented packet.
- *TTL*—The Time to Live (TTL) field represents the lifetime of the packet. Each router that forward the packet is supposed to adjust the TTL field. It is supposed to represent the lifetime in seconds, but in practice, it counts hops. Since the field is only one byte wide, the maximum packet lifetime is 255 hops.

- *Protocol*—The protocol field identifies the higher layer protocol (e.g., TCP or UDP) contained in the packet. The following table shows protocol field values and the associated protocols:

Protocol	Value
ICMP	1
IP (tunneled)	4
TCP	6
EGP	8
UDP	17
IGRP	88
IPIP (tunneled)	94
ETHERIP (tunneled)	97

1. WHAT'S IN AN IP PACKET?

107

- *Header checksum*—The header checksum field is used to detect transmission errors in the header. Routers don't need to ensure reliability of the payload, but they do need to ensure reliability of the header. Otherwise packets could be routed incorrectly.
- *Source IP address*—The source IP address is the four-byte IP address of the sender.
- *Destination IP address*—The destination IP address is the four-byte IP address of the intended recipient.

Options

The first byte of the options field, if present, identifies the option. Possible options include “source routing” and “record route.” Source routing allows the packet originator to explicitly specify the path the packet is to take (generally considered a bad). Many routers reject source routed packets today.

2 ICMP

The Internet Control Message Protocol (ICMP) is used to return control information to the recipient of a packet. For example, if a packet cannot be delivered, the first router that is unable to route a packet returns a “Destination unreachable” ICMP message to the sender. If the packet were silently dumped, the sender might wait a very long before it times out. The first byte of the payload identifies the type of ICMP message.

110

Common ICMP messages and their identifiers are shown in the following table:

Message	Type Value
Destination unreachable	3
Time exceeded	11
Redirect	5
Echo request	8
Echo reply	0

2. *ICMP*

111

The “Time exceeded” ICMP message indicates that a packet was killed by a router because its lifetime expired.

The “Redirect” message is used when a router gets a packet that it believe should have gone to a different router. The router that believes it received the packet in error tells the sender the correct router to use in its place. The originator then sends the packet to the new router.

The “Echo request” message is used by the **ping** command to request a host to generate test packet.

The “Echo reply” message is generated in response to an “Echo request” message.

How does traceroute use ICMP? The **traceroute** command sends a packet toward the specified destination using a lifetime of 1. The first router to see the packet will kill it and send a return message to the originator, identifying the killer router. The killer router is the first router the packet encountered. The **traceroute** command sends a second packet toward the specified destination using a lifetime of 2, knowing that the second router to encounter the packet will kill it. By continuing this process, the **traceroute** command generates a table of the path taken by the packet.

Sample captured packets

Digging through packets can be a tedious job, but it's definitely a valuable learning experience. Let's examine a few packets (in hex). Start by breaking the packet into 4-byte blocks. Here's a captured packet:

```
45 00 00 54
1f 32 00 00
40 01 87 a7
c6 6e 04 61
09 00 00 01
08 00 73 e4
⋮ ⋮ ⋮ ⋮
```

What's in this packet? The **45** identifies the packet as an IPv4 packet with a five word header. The total length of the packet is **54** (84 bytes). The time-to-live is **40** (64 hops) and the protocol is **01** (ICMP). The source address is **C6.6E.04.61** (that's 198.110.4.97) and the destination address is **09.00.00.01** (that's 9.0.0.1). The ICMP type is **08** (Echo request).

2. *ICMP*

115

Here's the response to that packet:

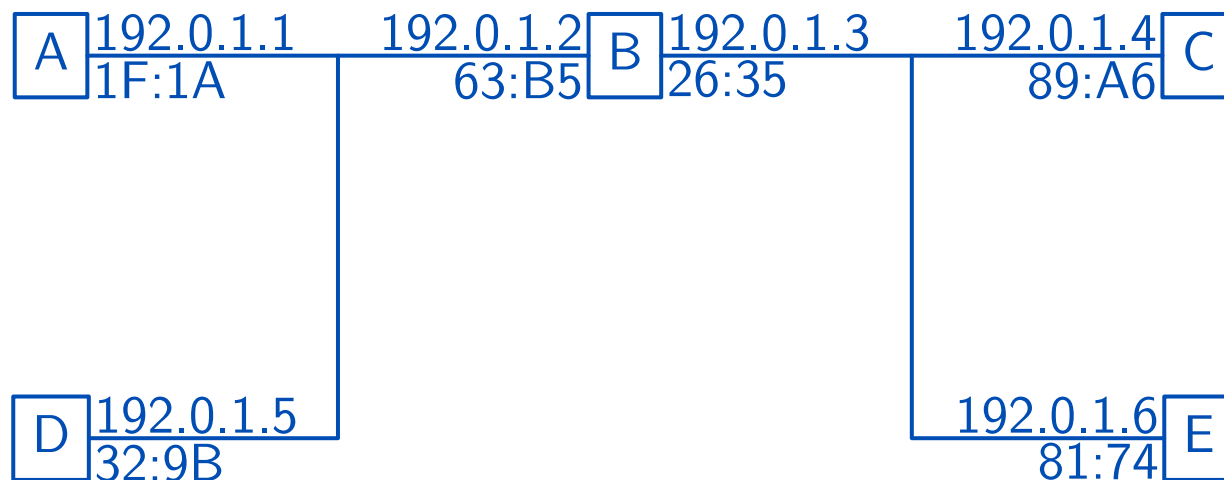
```
45 00 00 38
7a 9f 00 00
f9 01 9f 6c
c6 6c 16 7d
c6 6e 04 61
03 01 97 f5
00 00 00 00
45 00 00 54
1f 32 00 00
39 01 8e a7
c6 6e 04 61
09 00 00 01
08 00 73 e4
c2 24 27 00
```

What's in this packet? The **45** identifies the packet as an IPv4 packet with a five word header. The total length of the packet is **38** (56 bytes). The time-to-live is **f9** (249 hops) and the protocol is **01** (ICMP). The source address is **C6.6C.16.7D** (that's 198.108.22.125) and the destination address is **c6.6e.04.61** (that's 198.110.4.97). The ICMP type is **03** (Destination unreachable).

3 Proxy ARP

Proxy ARP is a method used to make a router look somewhat like a bridge. A router separates two media segments, so two hosts on either side of the router cannot communicate at the datalink layer.

Consider the following network:



Suppose host **A** wants to talk to host **E** using IP. Assuming the netmask is 255.255.255.0, host **A** assumes that host **E** is directly connected to the same media. In reality, it is not. Host **A** sends out an ARP request, assuming that **E** will see it. Of course, without some special mechanism, host **E** will not see the ARP request sent by host **A**.

If host **B** is running proxy ARP, it can respond *on behalf* of host **E**,

supplying *its own MAC address* when it sees the ARP request from **A**.

From **A**'s point of view, the conversation goes something like this:

- **A**: Broadcast from **1F:1A**—Who has 192.0.1.6?
- **B**: **63:B5** to **1F:1A**—192.0.1.6 is at **63:B5**.
A now adds **63:B5** to its ARP table for 192.0.1.6.
- **A**: **1F:1A** to **63:B5**—Here's a packet for 192.0.1.6 from 192.0.1.1.

At this point, **B** will take the packet and send it to **E** in the normal manner. When **E** returns a packet to **A**, the process is reversed.

120

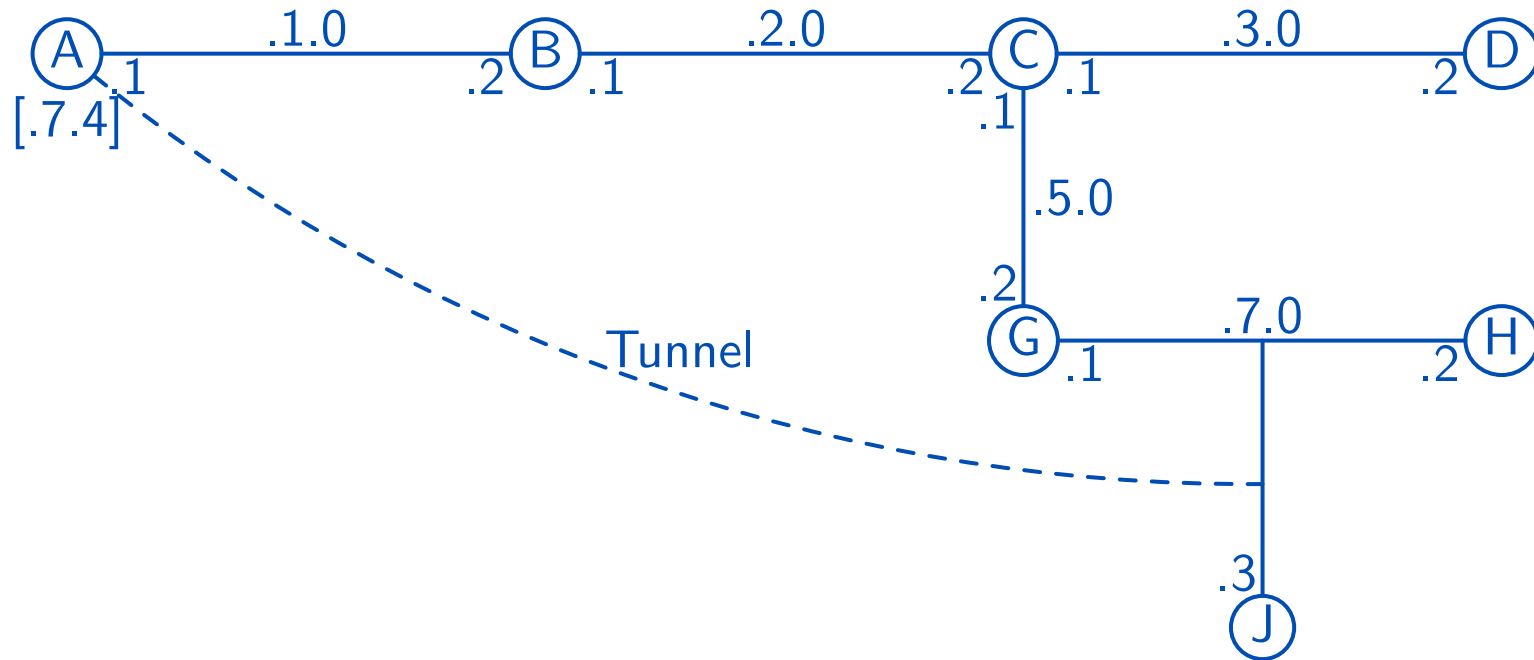
Why use proxy ARP?

Proxy ARP is sometimes useful when a few hosts on a media segment that would prevent normal routing table entries from reaching them. It can be used to make these hosts appear to be on a on media segment different than their true physical location. For example, PPP dialup lines sometimes use proxy ARP to assign a dialup user appear an IP address that is normally routed to an ethernet segment.

4 Tunneling

Tunneling protocols are used when one computer wants to appear as if it's directly connected to a remote network.

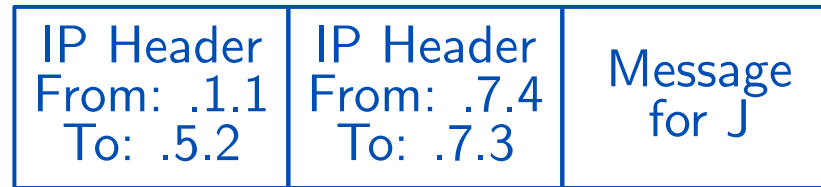
Suppose **A** wants to send data to **J** but appear to have an address on **J**'s LAN. Let's say that address is `.7.4` (if you are wondering which interface gets the `.7.4` address, the answer is that there's a virtual tunnel interface that gets that address):



4. TUNNELING

123

- First, **A** encapsulates its packet to **J** in a packet addressed to **G**:



- **B** and **C** both ignore the inner IP header and route based on the outer IP header. **B** and **C** don't even need to know that the .7 network exists.
- Eventually, **G** gets the packet and removes the outer header. **G** forwards the packet directly to **J**.

Let's consider the reverse process:

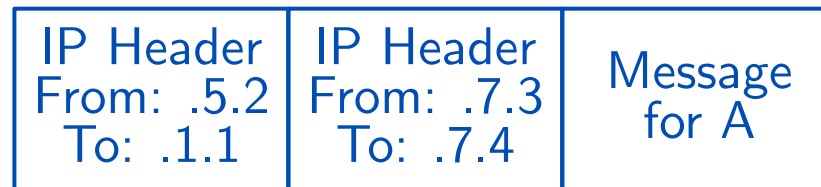
- J wants to reply to A, so J broadcasts an ARP request for A on the .7 network, since that's where it thinks that A is.
- Since it's the tunnel endpoint, G responds to the ARP request on behalf of A using G's own IP address.
- J addresses a packet to A at the IP layer and sends it to G using the datalink layer protocol.

Datalink Header From: J To: G	IP Header From: .7.3 To: .7.4	Message for A
-------------------------------------	-------------------------------------	------------------

4. TUNNELING

125

- **G** encapsulates the packet in a header addressed to **A** and sends it using the IP layer.



- **A** receives the packet and, being a tunnel endpoint, strips off the outer header. It finds the inner packet addressed to its own virtual tunnel interface, so it examines the contents.

5 Network Address Translation

Sometimes it's convenient to hide a network behind a single IP address or a group of IP addresses. There are several reasons you might want to do this:

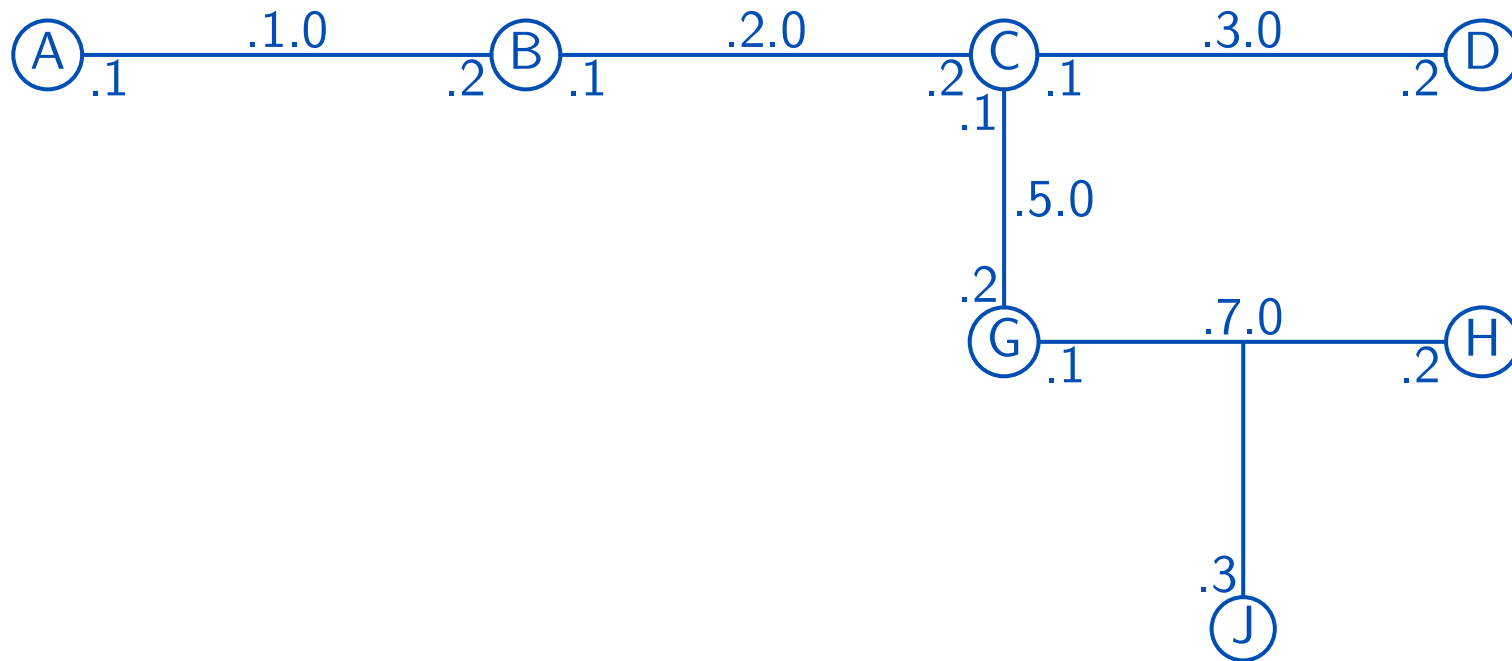
1. The network could consist of “private” IP addresses that are not registered on the internet.
2. You may not want the network addresses to be known on the internet, so choose not to advertise routes to them.
3. You may want to use a network address for these machines because you know that will change the way packets are routed to them.

Network Address Translation is a method of allow the “hidden” machines to connect to the internet even though their true network address may not exist and may not be reachable from the internet.

5. NETWORK ADDRESS TRANSLATION

127

Let's reconsider the earlier network:



128

Suppose you want to hide the .7 network by using **G** to translate the .7 addresses to its own address. Suppose that **J** and **H** connect to **A**, and that **A** responds.

- **J** and **H** start normal packets toward **A**:

IP Header From: .7.3 To: .1.1	Message for A
-------------------------------------	------------------

IP Header From: .7.2 To: .1.1	Message for A
-------------------------------------	------------------

5. NETWORK ADDRESS TRANSLATION

129

- **G** gets the packets, notices the source addresses are from a “hidden” network, and *removes* the source addresses, substituting its own:

IP Header From: .5.2 To: .1.1	Message for A
-------------------------------------	------------------

IP Header From: .5.2 To: .1.1	Message for A
-------------------------------------	------------------

- **A** gets the packets, believing that it is talking to **G**, when the packets actually originated from **J** and **H**.

130

- A responds to both packets, which it recognizes as two different connection requests, but addresses the return packets to G.

IP Header From: .1.1 To: .5.2	Message for G?
-------------------------------------	-------------------

IP Header From: .1.1 To: .5.2	Message for G?
-------------------------------------	-------------------

5. NETWORK ADDRESS TRANSLATION

131

- **G** gets the return packets. **G** has the rather difficult job of figuring out which packet should go to **J** and which should go to **H**. In order to do this, it must keep fairly large tables that allow it to remember who connected to whom. *Network address translation requires more memory than normal routing.* It alters the addresses in the packets and forwards them to **J** and **H** correctly.

IP Header From: .1.1 To: .7.3	Message for G?
-------------------------------------	-------------------

IP Header From: .1.1 To: .7.2	Message for G?
-------------------------------------	-------------------

5. NETWORK ADDRESS TRANSLATION

133

Chapter 6
Basics of Packet Filtering

1 Introduction

This chapter introduces you to some ideas used in packet filtering. A packet filter is the front-line of defense in virtually any firewall strategy.

2 Some Layer 4 preliminaries

To this point, we have avoided discussions of the contents of Layer 4 headers. However, some basic TCP/UDP concepts are required to design effective packet filters. In this section we discuss TCP/UDP port numbers and connection information.

Port numbers

All network layers have some sort of addressing scheme. The datalink layer has MAC addresses; the network layer has IP addresses; the transport layer has “port” addresses. The port address is a two-byte field in the TCP/UDP header, so there are 65,536 possible port addresses. An IP address refers to a network interface on a computer. A port refers to the address of a specific software program running on the host computer. All packets arriving at a computer from *any* network interface having the same port address connect to the same software server program. A set of “standard” software server programs that run on various port addresses has emerged on the Internet. Some of the more common port assignments are shown in the following table:

2. *SOME LAYER 4 PRELIMINARIES*

139

Port	Protocol	Software Server
23	TCP	telnet
25	TCP	Mail servers
21	TCP	FTP servers
20	TCP	FTP (return data)
22	TCP	ssh servers
53	UDP	DNS servers
80	TCP	Web servers
111	TCP	Portmapper
137–139	TCP	Windows Browsing/File Sharing
520	UDP	RIP
6000	TCP	X servers

Usually, the port number identifies the particular software server whose service is being requested.

2.1 Connection information

A second issue is the connection information. Recall that TCP is a connection-oriented protocol and that UDP is a connectionless protocol. A connection appears to be a seamless connection between two endpoints. Once established, it remains connected until one side “hangs up.” To establish the connection, in effect, one computer makes a “connection request (SYN)” to the other computer. The second computer responds with an “acknowledgement (ACK)” and a return SYN. At this point the connection is said to be “established.” The SYN and ACK information is contained in the TCP header.

2. *SOME LAYER 4 PRELIMINARIES*

141

A normal connection from **A** to **B** appears as follows:

- A sends SYN to B.
- B sends SYN+ACK to A.
- A sends ACK to B.
- B sends ACK to A.
- A sends ACK to B.
- ...
- A sends FIN to B.
- B responds with ACK.
- B sends FIN to A.
- A sends ACK to B.

The key information here is that a connection has a definite initiator. Compare this to a telephone call. One and only one person dialed the number. If you enter a room in the middle of a phone call, it's hard to say who initiated the call. However, if you see one person dial the phone or pickup the phone, there's little doubt. If you block the "ring" signal, you block the entire call.

In the TCP protocol, the SYN is like the ring signal. Only the caller sends a SYN without an ACK. A packet with an ACK is *not* a connection request. We can always tell if a packet belongs to an established, or to a connection request. A standard packet filtering technique is to block connection requests in the direction that you don't want them. If you want to block incoming connection requests, you block inbound SYNs without ACKs, or alternatively, you *allow* inbound ACKs and FINs and block anything else.

2. *SOME LAYER 4 PRELIMINARIES*

143

UDP, on the other hand, is connectionless. Given a random UDP packet, there's no way to tell if that packet is attempting to initiate some action or is a response.

3 Typical Policies

At a minimum, all sites should implement good citizenship policies:

- *For “stub networks”, block packets leaving your site that have addresses not assigned by your site.*
- *Block packets destined for your network or broadcast addresses (so-called directed broadcast) packets.* This prevents your site from being used as a broadcast amplifier (SMURF attack) and protects you too.

Additionally, sites should consider blocking services that would not normally be used from off-campus by legitimate users (e.g., Portmapper).