TCP/IP Protocol Suite

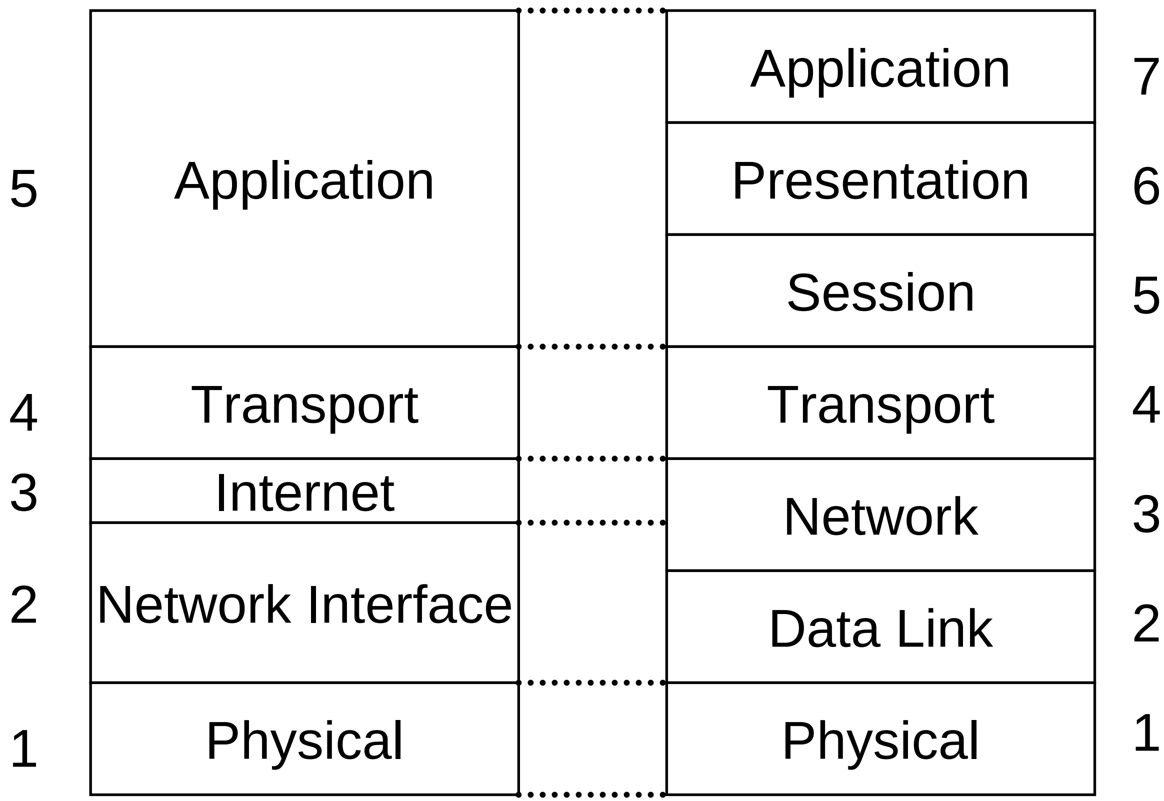
- A protocol suite for internetworking
- The *TCP/IP Internet Protocols* or, simply, *TCP/IP* is the mostly widely used internetworking protocol suite
- First internetworking protocol suite
- OSI 7-layer model does not explicitly include internetworking
- TCP/IP model replaces the old ISO model

TCP/IP Layering

Application	5
Transport	4
Internet	3
Network Interface	2
Physical	1

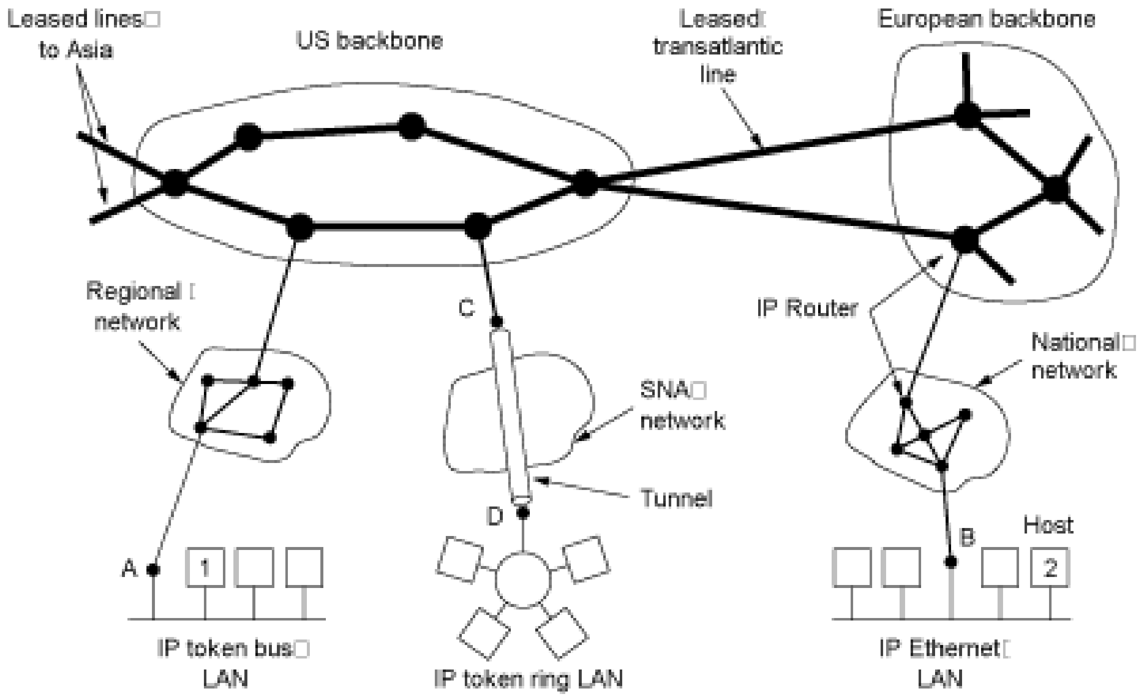
- <u>Layer #5: Application</u>
  - Corresponds to ISO model layers 6 and 7
  - Used for communication among applications
- <u>Layer #4: Transport</u>
  - Corresponds to layer 4 in the ISO model
  - Provides reliable delivery of data
- <u>Layer #3: Internet</u>
  - Defines uniform format of packets forwarded across networks of different technologies
  - Rules for forwarding packets in routers
- <u>Layer #2: Network Interface</u>
  - Corresponds to layer 2 in the ISO model
  - Defines formats for carrying packets in hardware frames
- <u>Layer #1: Hardware</u>
  - Corresponds to layer 1 in the ISO model
  - Defines basic networking hardware

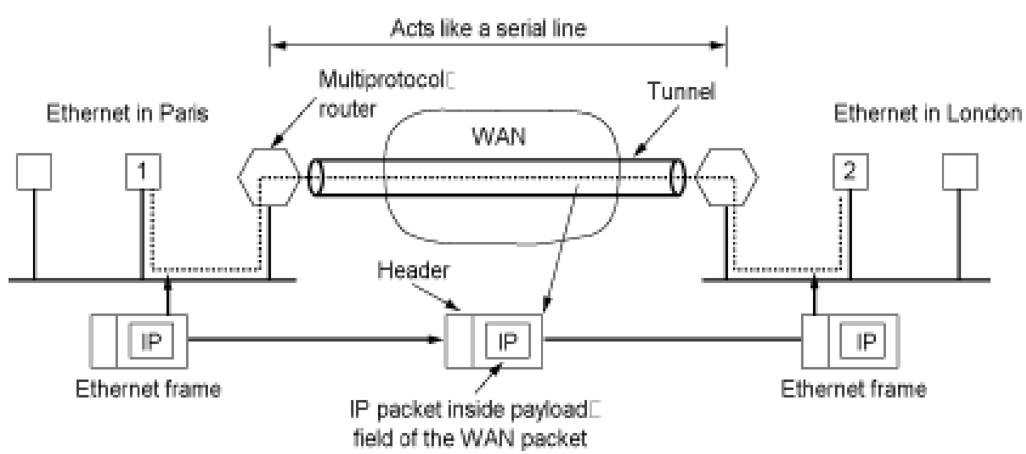
TCP/IP vs OSI



5 Application
4 Transport
3 Internet
2 Network Interface
1 Physical
Application 7
Presentation 6
Session 5
Transport 4
Network 3
Data Link 2
Physical 1

The Internet





Addresses for the Virtual Internet

- One key aspect for creating a virtual network is a single, uniform addressing format
- Can't use hardware addresses because different technologies have different address formats
- Address format must be independent of any particular hardware address format
- Sending host puts destination internet address in packet
- Destination address can be interpreted by any intermediate router
- Routers examine address and forward packet on to the destination
- TCP/IP addresses:

- Addressing in TCP/IP is specified by the *Internet Protocol* (IP)
- Each host (not really) is assigned a 32-bit number
- Called the *IP address* or *Internet address*
- Unique across entire Internet

IP Address Hierarchy

- Each IP address is divided into a prefix and a suffix
- Prefix identifies the network to which a computer is attached
- Suffix identifies a computer within that network
- Address format makes routing efficient
- Every network in a TCP/IP internet is assigned a unique *network number*
- Each host on a specific network is assigned a *host number* or *host address* that is unique *within that network*
- Host's IP address is the combination of the network number (prefix) and host address (suffix)

IP Address Assignment

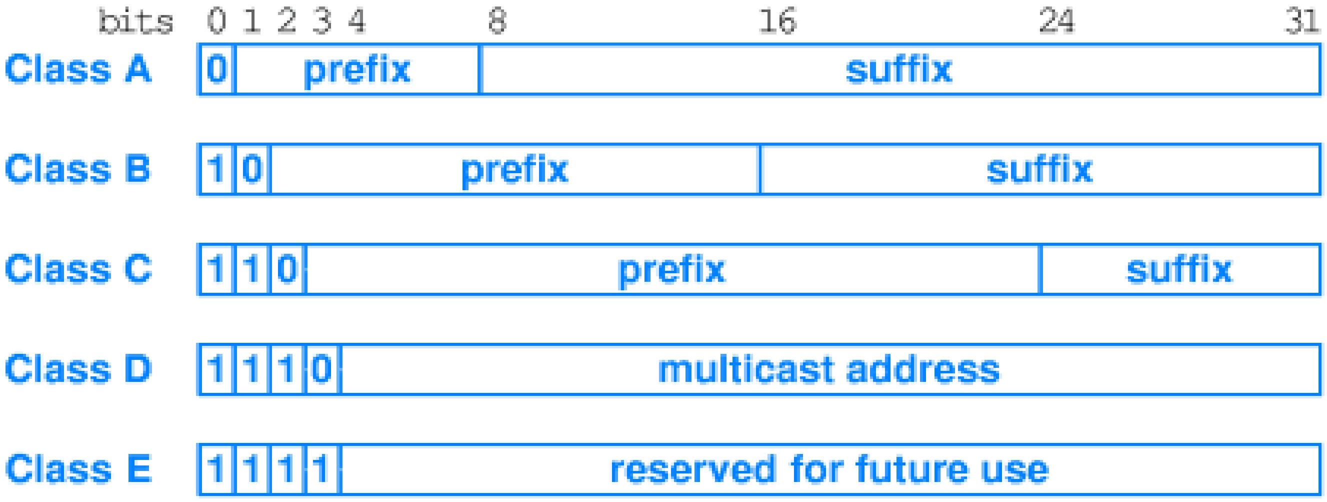
- An IP address does not identify a specific computer
- An IP address specifies an *interface*, or network attachment point, *not* a computer
- A computer with multiple network interconnections (e.g., a router) must be assigned one IP address for each connection
- Global authority assigns unique prefix to network
- Local administrator assigns unique suffix to host

IP Address Format Design

- IP designers chose 32-bit addresses
  - Allocate some bits for prefix, some for suffix
  - Large prefix, small suffix: many networks, few hosts per network
  - Small prefix, large suffix: few networks, many hosts per network
- Because of a variety of technologies, need to allow for both large and small networks
- Designers chose a compromise - multiple address formats that allow both large and small prefixes
- Each format is called an address *class*

Classful IP Addressing

- First four bits of an address determine the class to which the class belongs
- Specify how the remainder of the address is divided into prefix and suffix



Dotted Decimal Notation

- Shorthand for IP address
- Allows humans to avoid binary
- Represents each byte in decimal separated by dots
- Four decimal values per 32-bit address
- Each decimal number:

 - Represents eight bits

 - Is between 0 and 255

32-bit Binary Number	Equivalent Dotted Decimal
10000001 00110100 00000110 00000000	129 . 52 . 6 . 0
11000000 00000101 00110000 00000011	192 . 5 . 48 . 3
00001010 00000010 00000000 00100101	10 . 2 . 0 . 37
10000000 00001010 00000010 00000011	128 . 10 . 2 . 3
10000000 10000000 11111111 00000000	128 . 128 . 255 . 0

Classes and Dotted Decimal Notation

- Class identified by the decimal value of the first byte

Class	Range
A	0-127
B	128-191
C	192-223
D	224-239
E	240-255

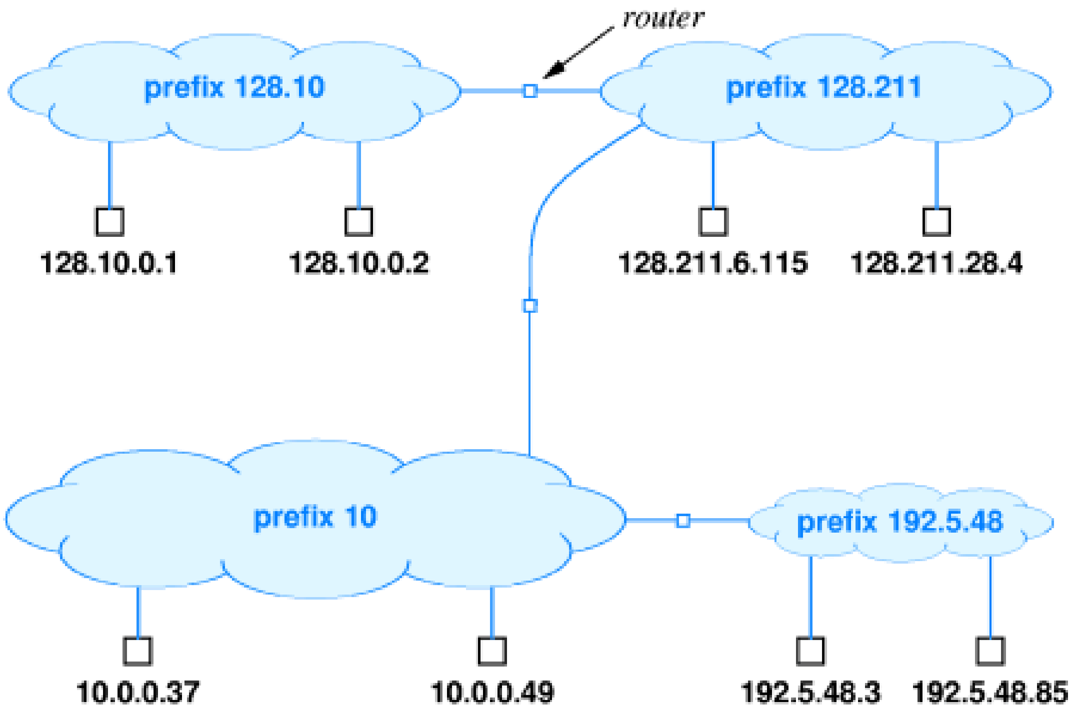
Classful Addressing and Network Sizes

- Maximum network size determined by class of address:

- Class A large
- Class B medium
- Class C small

Address Class	Bits In Prefix	Maximum Number of Networks	Bits In Suffix	Maximum Number Of Hosts Per Network
A	7	128	24	16777216
B	14	16384	16	65536
C	21	2097152	8	256

Classful Addressing Example



Address Masks

- IP address space was being exhausted
- Since one of three possible sizes had to be chosen many addresses were unused
- Idea: Instead of having three distinct classes, allow the division between prefix and suffix to occur on an arbitrary bit boundary
- Two pieces of information must be kept for each address:

  - The 32-bit address itself
  - A 32-bit value that specifies the boundary between the prefix and the suffix (*address mask* or *subnet mask*)
    * 1 bits mark the network prefix
    * 0 bits mark the host suffix

Address Masks Example (1/2)

- Suppose we have the 32-bit destination address 128.10.2.3

 - 10000000 00001010 00000010 00000011

- And a 32-bit mask 255.255.0.0

 - 11111111 11111111 00000000 00000000

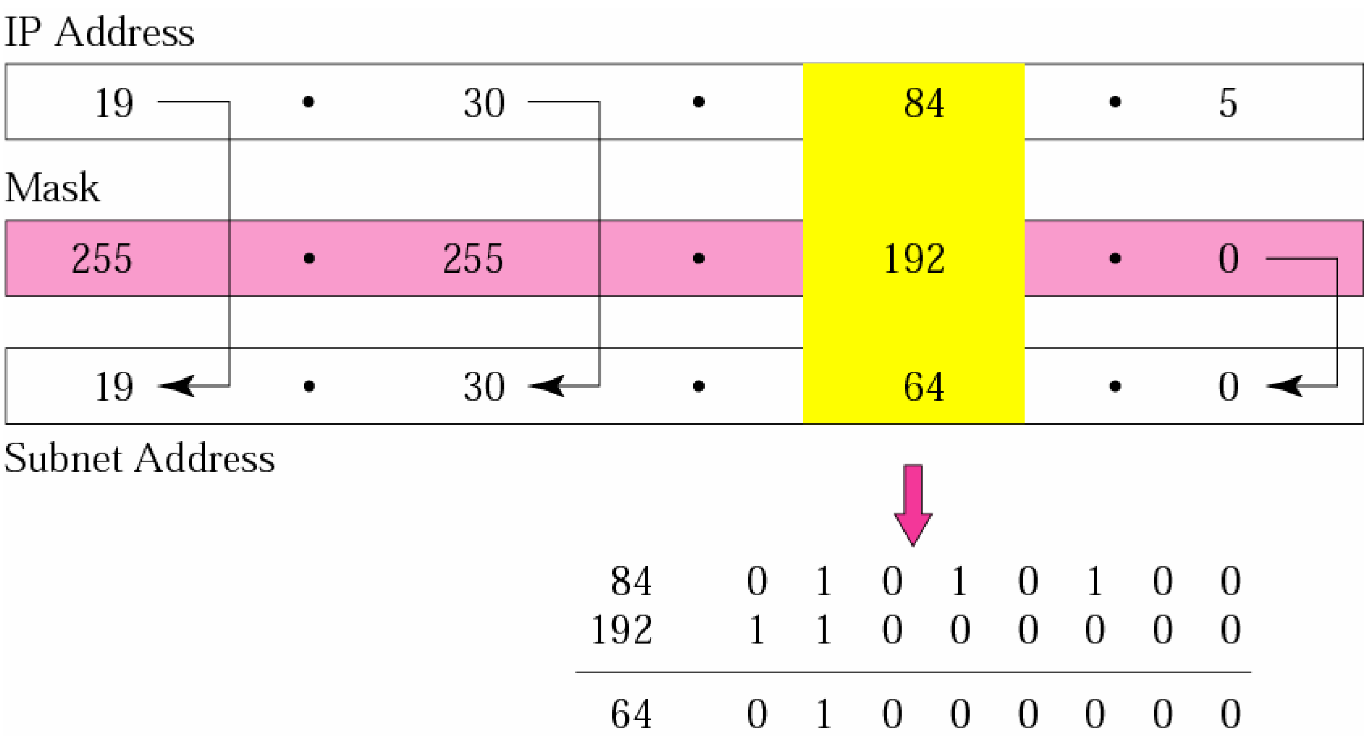
- In order a router to find the correct subnet to route an incoming packet to, performs a *logical and* between the destination address and the mask

 - 10000000 00001010 00000000 00000000

 - Which is 128.10.0.0 in the dotted decimal notation

- This is compared against a value in the router's routing table

Address Masks Example (2/2)



Classless Inter-Domain Routing

- A dotted decimal notation for address masks
  - The mask associated with an address is specified by appending a slash to an address and the size of the mask in decimal
- Example:
  - Classful: 128.10.0.0 (16-bit prefix, 16-bit suffix)
  - CIDR: 128.10.0.0/16

Address Translation

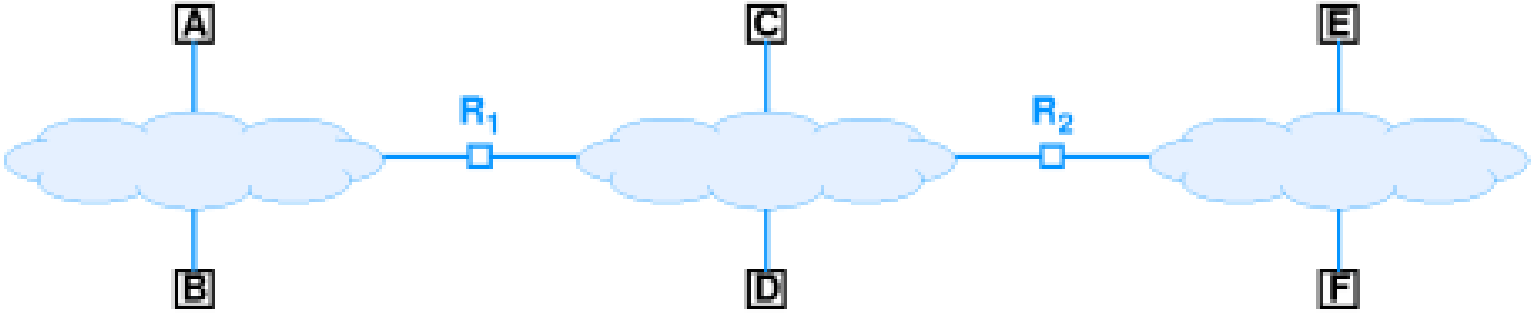
- Upper levels of protocol stack use *protocol addresses*
- Network hardware must use *hardware address* for eventual delivery
- Protocol address must be translated into hardware address for delivery; will discuss three methods
- Upper levels use only protocol addresses
  - ``Virtual network'' addressing scheme
  - Hides hardware details
- Translation occurs at *data link* layer
  - Upper layer hands down protocol address of destination
  - Data link layer translates into hardware address for use by hardware layer

Address Resolution

- Finding hardware address for protocol address:
  - *Address resolution*
  - Data link layer *resolves* protocol address to hardware address
- Resolution is local to a network
- Network component only resolves address for other components on same network

Address Resolution Example

- *A* resolves protocol address for *B* for protocol messages from an application on *A* sent to an application on *B*
- *A* does *not* resolve a protocol address for *F*
- Through the internet layer, *A* delivers to *F* by routing through *R1* and *R2*
- *A* resolves *R1* hardware address
- Network layer on *A* passes packet containing destination protocol address *F* for delivery to *R1*



Address Resolution Techniques

- Association between a protocol address and a hardware address is called a *binding*
- Three techniques:

- Table lookup:
  * Bindings stored in memory with protocol address as key
  * Data link layer looks up protocol address to find hardware address
- Closed-form computation:
  * Protocol address based on hardware address
  * Data link layer derives hardware address from protocol address
- Dynamic:
  * Network messages used for ``just-in-time'' resolution
  * Data link layer sends message requesting hardware address; destination responds with its hardware address

Address Resolution Protocol (ARP)

- IP uses dynamic (plus table lookup as an optimization)
- *Address Resolution Protocol* (ARP) - part of TCP/IP protocol suite
- Two-part protocol

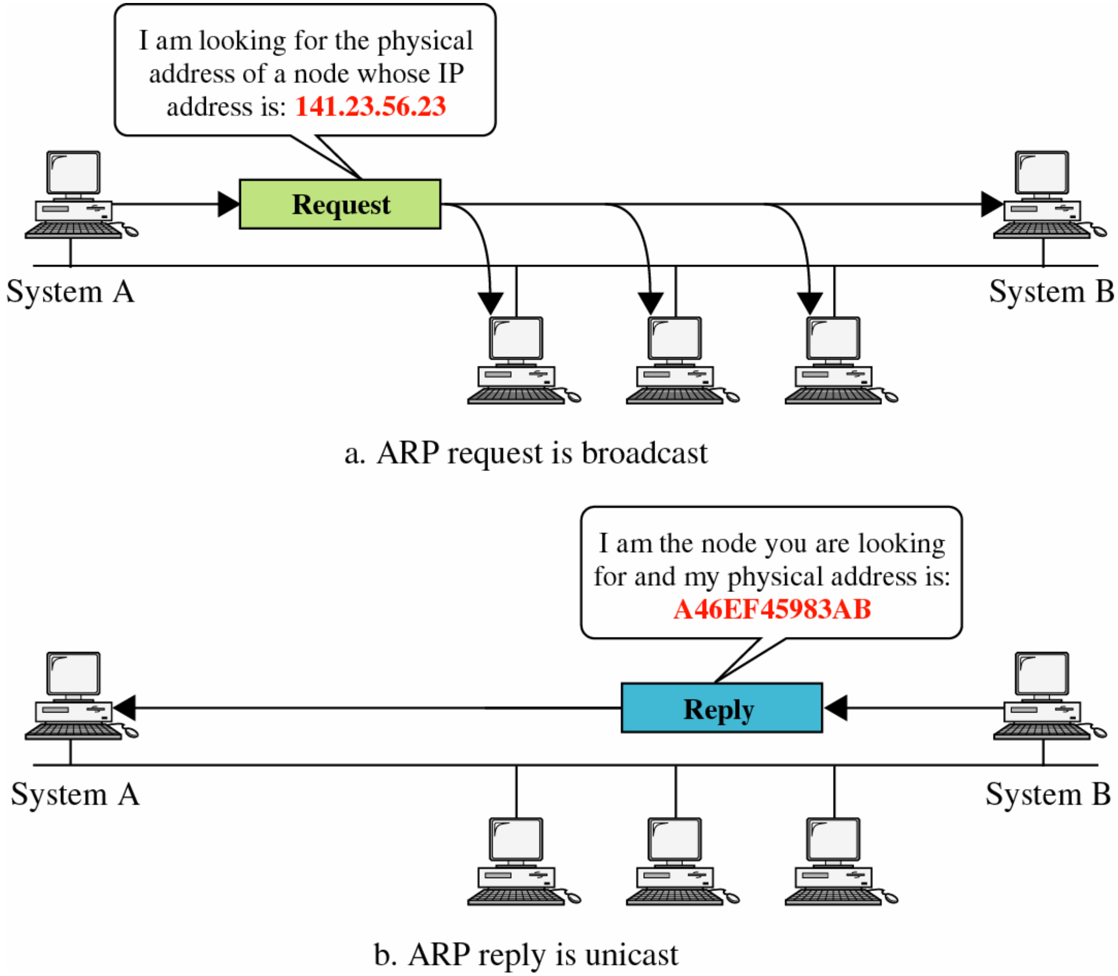
 - Request from source asking for hardware address
 - Reply from destination carrying hardware address

- ARP request message dropped into hardware frame and broadcast
- Uses separate protocol type in hardware frame (Ethernet: 0x0806)
- Sender inserts IP address into message and broadcast
- Every other computer examines request
- Computer whose IP address is in request responds

 - Puts hardware address in response
 - Unicasts to sender

- Original requester can then extract hardware address and send IP packet to destination

ARP Operation



ARP Packet

- The target machine responds by filling in the missing address

 - Swaps the target and sender pairs and changes the operation to a reply

- *A* will add an entry into its *ARP cache*

 - Other nodes may note the IP to MAC address mapping of the node that sent the ARP request

Hardware Type		Protocol Type
Hardware length	Protocol length	Operation Request 1, Reply 2
Sender hardware address (For example, 6 bytes for Ethernet)		
Sender protocol address (For example, 4 bytes for IP)		
Target hardware address (For example, 6 bytes for Ethernet) (It is not filled in a request)		
Target protocol address (For example, 4 bytes for IP)		

ARP Cache

- Using ARP for each IP packet adds two packets of overhead for each IP packet
- Computer caches ARP responses

 - Flushes cache at system startup

 - Entries discarded periodically

- Cache searched prior to sending ARP request
- ARP lookup algorithm:

 - Look for target IP address, B, in ARP table

 - If not found

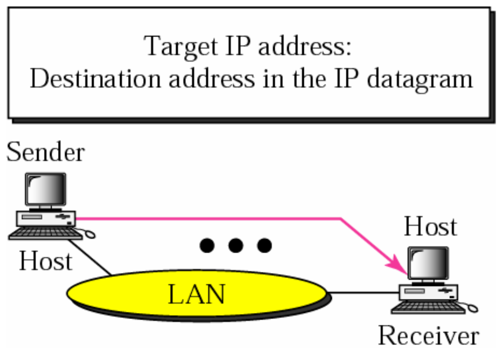
    * Send ARP request

    * Receive reply with B's hardware address

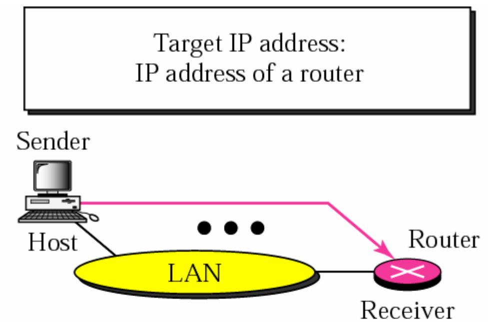
    * Add entry to table

 - Return hardware address from table

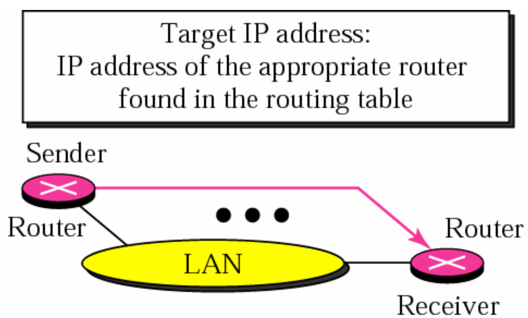
ARP Use Cases



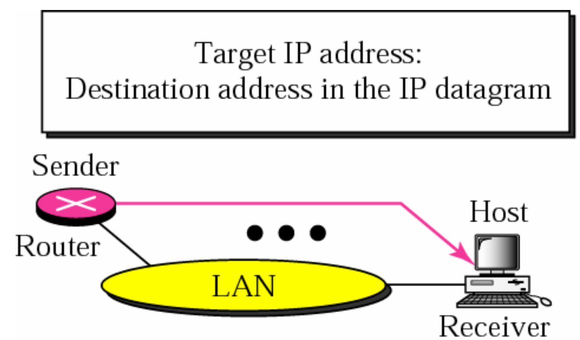
Case 1. A host has a packet to send to another host on the same network.



Case 2. A host wants to send a packet to another host on another network.
It must first be delivered to a router.



Case 3. A router receives a packet to be sent to a host on another network.
It must first be delivered to the appropriate router.



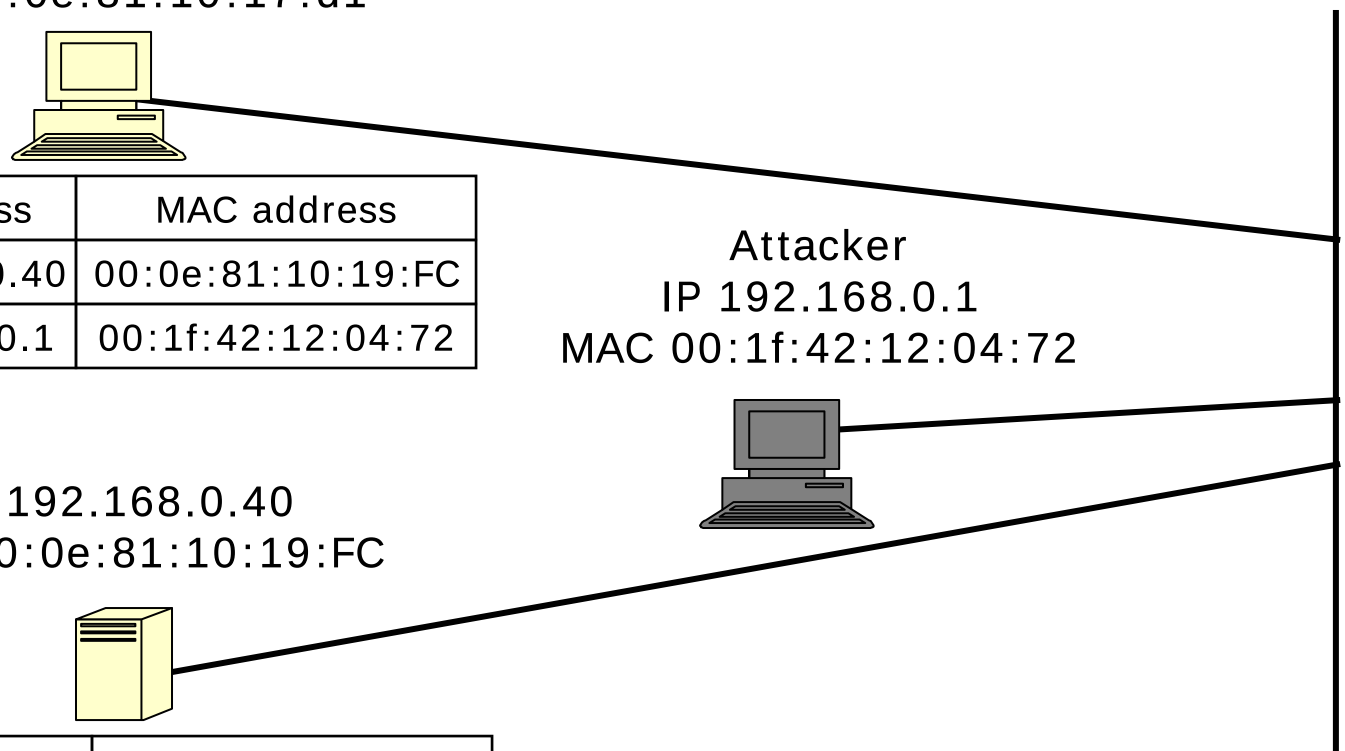
Case 4. A router receives a packet to be sent to a host on the same network.

ARP Vulnerability

- ARP replies overwrite existing information
- Also, legitimate hosts send ARP replies on joining network or changing IP address

  - Not in response to any ARP request
- ARP replies have no proof of origin, so an attacker can claim any hardware address
- Basis for many more attacks
- Knows as a *man-in-the-middle* (MITM) attack

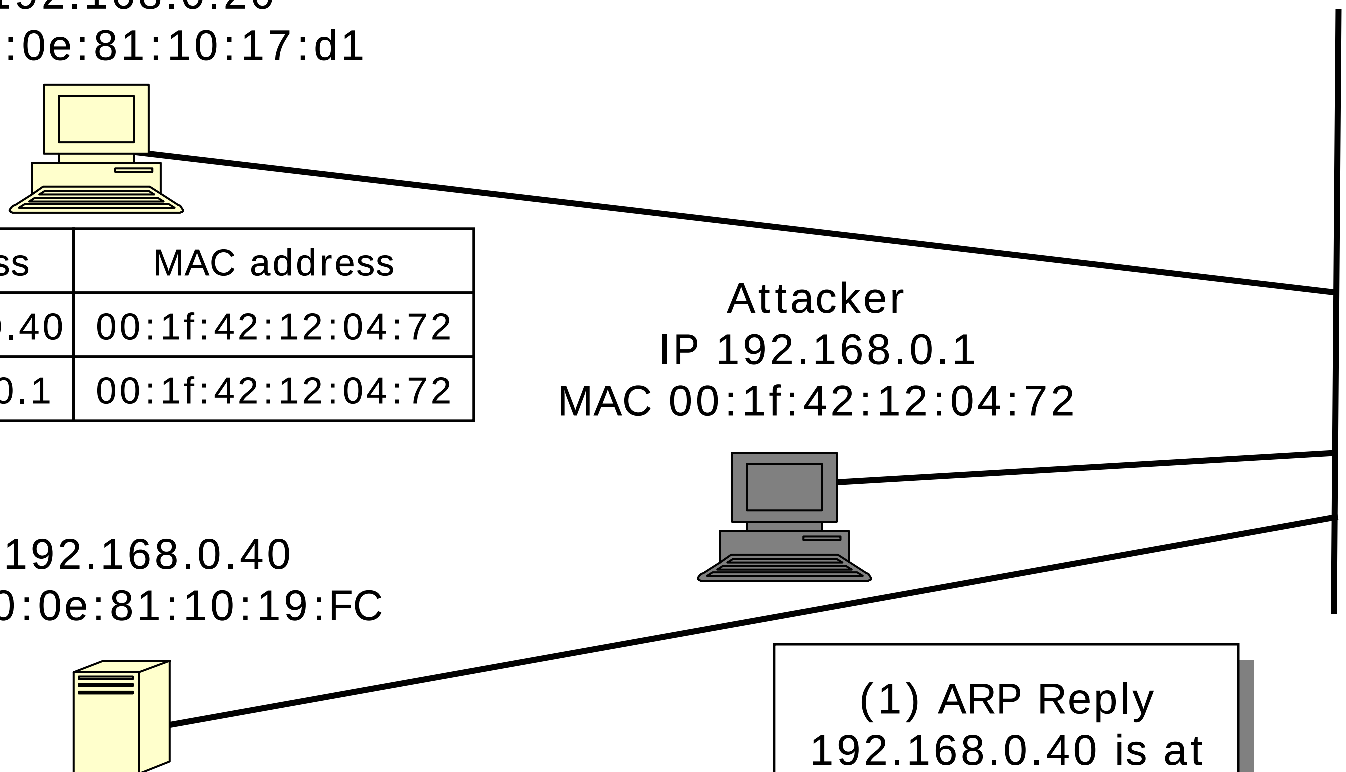
Example ARP Attack (1/3)



IP address	MAC address
192.168.0.40	00:0e:81:10:19:FC
192.168.0.1	00:1f:42:12:04:72

IP address	MAC address
192.168.0.20	00:0e:81:10:17:d1
192.168.0.1	00:1f:42:12:04:72

Example ARP Attack (2/3)



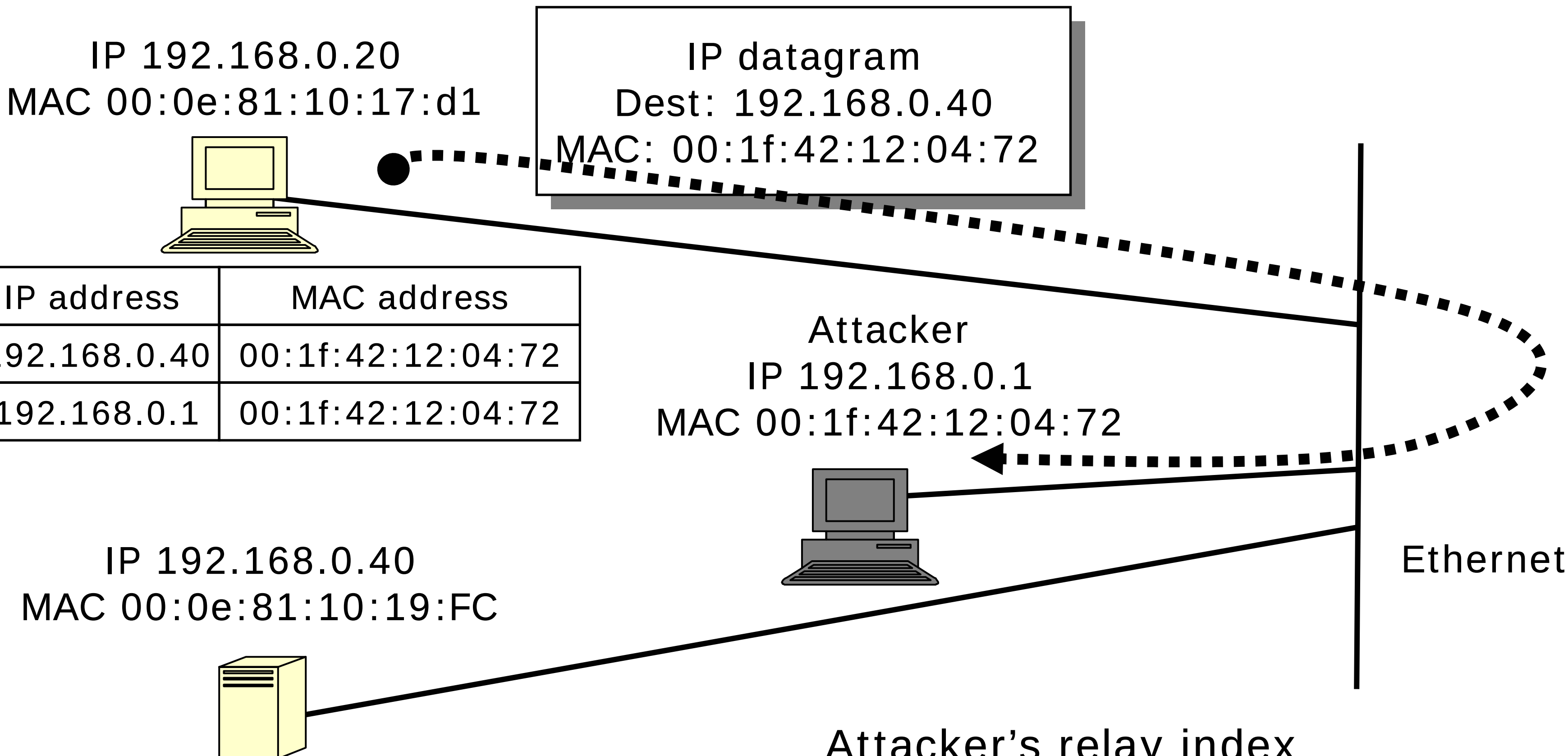
IP address	MAC address
192.168.0.40	00:1f:42:12:04:72
192.168.0.1	00:1f:42:12:04:72

IP address	MAC address
192.168.0.20	00:1f:42:12:04:72
192.168.0.1	00:1f:42:12:04:72

(1) ARP Reply
192.168.0.40 is at
00:1f:42:12:04:72

(2) ARP Reply
192.168.0.20 is at
00:1f:42:12:04:72

Example ARP Attack (3/3)



IP address	MAC address
192.168.0.40	00:1f:42:12:04:72
192.168.0.1	00:1f:42:12:04:72

IP address	MAC address
192.168.0.20	00:1f:42:12:04:72
192.168.0.1	00:1f:42:12:04:72

Attacker's relay index

IP address	MAC address
192.168.0.40	00:0e:81:10:19:FC
192.168.0.20	00:0e:81:10:17:d1

ARP Attack Effect

- Attacker keeps a *relay index*: a table containing the true association between MAC addresses and IP addresses
- But the two devices at 192.168.0.20 and 192.18.0.40 update their ARP caches with false information
- All traffic for 192.168.0.20 and 192.168.0.40 gets sent to attacker
- Attacker can re-route this traffic to the correct devices using his relay index
- So these devices are oblivious to the attack
- Many implementations out there for many purposes:

- Capturing traffic in switched Ethernet environments (i.e. active sniffing)
- Hijacking any application protocol that relies on IP
- Bypassing firewalls
    * http://www.althes.fr/ressources/avis/smartspoofing.htm
- MITM attacks on cryptographically secure protocols
- Denial of service