CSIS 4222

Ch 20: Internetworking

Ch 22: IP Datagrams and Forwarding

Physical Network Connections

The basic component used to connect heterogeneous networks is a *router*

– contains a processor, memory and a separate I/O interface for each network to which it connects

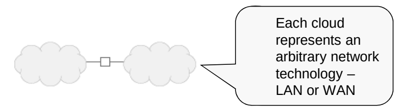


Figure 20.1 Two physical networks connected by a router, which has a separate interface for each network connection. Computers can attach to each network.

Achieving Universal Service

Routers must agree on how to forward information

– Frame formats and addressing schemes used by the underlying networks can differ

– Protocol software overcome the differences to make universal service possible among different network technologies

A Virtual Network

Internetworking provides the appearance of a single seamless communication system

Software hides the details of

– physical network connections

– physical addresses

– routing information

The Virtual Internet

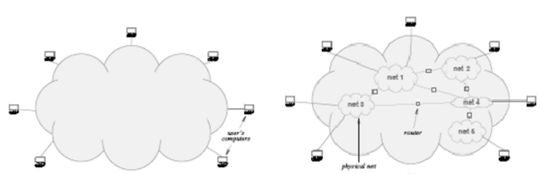


Figure 20.3 The Internet concept:
(a) The illusion of a single network provided to users and applications
(b) The underlying physical structure with routers interconnecting networks

Layered Software Processing

Layers on the sender:

– Accepts an outgoing message from the layer above

– Adds a header and does other processing

– Passes the resulting message to next lower layer

Layers on the receiver:

– Receives an incoming message from the layer below

– Removes the header for that layer and performs other processing

– Passes the resulting message to the next higher layer

TCP/IP Reference Model
The protocol used for internetworking

Based on a view of data communication as

- *Processes* - Fundamental entities that communicate
- *Hosts* - Execute processes (multiple-simultaneous processes possible)
- *Networks* - Connection between hosts

TCP/IP Reference Model

Transferring data to a process requires

1. Getting data to the host where the process resides
2. Getting data to the correct process on the host

TCP/IP Design Goals

- Interconnect multiple heterogeneous networks in a seamless way
- Be able to survive a partial loss of subnet hardware
- Flexibility to handle the requirements of diverse applications

Internet Protocol Stack

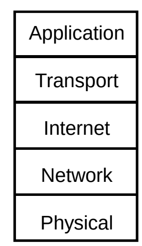
5. Application: supports network apps
- FTP, SMTP, HTTP

4. Transport: data transfer between applications
- TCP, UDP

3. Internet: routing of datagrams from source host to destination host
- IP, routing protocols

2. Network interface: data transfer between neighboring network elements
- Ethernet, MAC addressing

1. Physical: bits “on the wire”



Host Computers, Routers, and Protocol Layers

Hosts connect to an internet and run applications
- Cell phone, PC, server, mainframe

TCP/IP protocols make it possible for any pair of hosts to communicate

Both hosts and routers need TCP/IP protocol software
- But routers do not use protocols from all layers

Service Paradigms

Connection-oriented (like telephone system)
- Before data flows, two hosts and intervening routers establish a *virtual connection*
- Connection is maintained as long as they have data to exchange

Connectionless (like postal system)
- Endpoint puts data to send into a *packet* and hands it to the network for delivery
- Different packets between a source-destination pair may take different paths

No choice: The network provides one or the other
- VC used in ATM, frame-relay, X.25
- Connectionless used in today’s Internet

Connection-Oriented Service

- One endpoint requests a connection & the other agrees
- Computers exchange data through established connection
- Typically uses a *stream interface*
  - Source delivers a stream of data to network
  - Network breaks the stream into packets for delivery
- Like telephone, connection remains in place even while no data transmitted
- One endpoint requests to break the connection when transmission is complete

Connectionless Service

- No connection agreement necessary
- Source of data adds destination information and delivers to the network
- Network delivers each data item individually

TCP/IP Support

TCP/IP includes protocols for

- An unreliable connectionless delivery service
- A reliable connection-oriented service
  - that uses the underlying connectionless service

- This design forms the basis for all Internet communication

How does a packet cross the Internet?

Source host creates a packet

- Puts destination address in the packet header
- Sends the packet to a nearby router

Router receives a packet

- Uses the destination address to select the next router on the path
- Forwards the packet

Eventually, the packet reaches a router that can deliver the packet to its final destination

What format is used for Internet packets?

IP defines a *universal virtual* packet format that is independent of the hardware frame formats

- Virtual
  - Format is not tied directly to any hardware
  - The underlying hardware does not understand or recognize an Internet packet
- Universal
  - Each host and router in the Internet contains protocol software that recognizes Internet packets

Internet Packets

- Created and understood by software
- Contains sender and destination IP addresses
- A self contained packet that carries sufficient information for routing from source *host* to destination *host*

The IP Datagram

TCP/IP uses the name *IP datagram* to refer to a packet

- Datagram size is determined by the application that sends the data
  - Fixed size header fields
  - Payload can be up to 64K octets

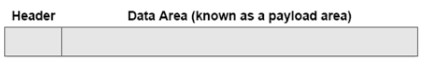
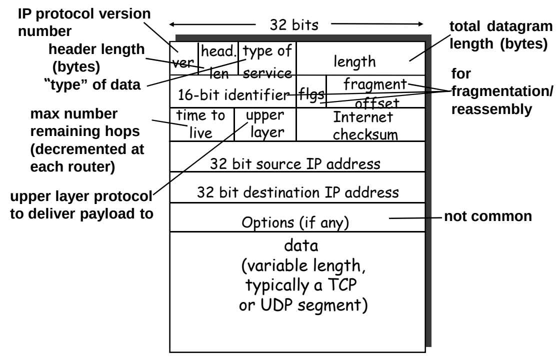


Figure 22.1 The general form of an IP datagram with a header followed by a payload.

IP Datagram Format



Best-Effort Delivery

IP does not guarantee that it will handle all problems

- Datagram duplication
- Delayed or out-of-order delivery
- Corruption of data
- Datagram loss

– IP makes a best-effort to deliver each datagram

IP Datagram Forwarding

Is performed by routers

- Table-driven
- Uses IP addresses
- Table entry specifies next hop
  - Next-hop is a router or the destination

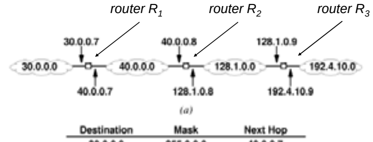
Next-hop Forwarding

Each router along the path

- receives the datagram
- extracts the destination address from the header
- uses the destination address to determine a next hop
- then forwards the datagram to the next hop
  - either the final destination or another router

Routing Table Example

Forwarding table for *router* R_2



(a)

Destination	Mask	Next Hop
30.0.0.0	255.0.0.0	40.0.0.7
40.0.0.0	255.0.0.0	deliver direct
128.1.0.0	255.255.0.0	deliver direct
192.4.10.0	255.255.255.0	128.1.0.9

(b)

Longest Prefix Match

Suppose a router's forwarding table contains entries for network prefixes:

128.10.0.0/16 and 128.10.2.0/24

- What happens if a datagram destination is 128.10.2.3?
- Which entry should be used?
- Internet forwarding uses a *longest prefix match*
- Choose the entry for 128.10.2.0/24

Routing Table Size

Since each destination in a routing table corresponds to a network, the number of entries is proportional to the number of networks in an internet.

Datagram Transmission

- Datagram sent across conventional network
  - From source host to router
  - Between intermediate routers
  - From final router to destination host
- Network hardware does not recognize
  - Datagram format
  - IP addresses

Datagram Transmission and Frames

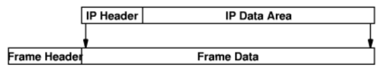
- Internet layer (IP)
  - Constructs datagram
  - Determines next hop
  - Hands to network interface layer
- Network interface layer
  - Binds next hop address to hardware address
  - Prepares datagram for transmission
- But hardware frame doesn't understand IP, so how is datagram transmitted?

Encapsulation

- Network interface layer *encapsulates* IP datagram as the data area in a hardware frame
  - Hardware ignores IP datagram format
- Standards define data type for IP datagram, ARP, etc.
- Receiving protocol stack interprets data area based on frame type

Encapsulation

- Entire datagram treated as data
- Frame type identifies contents as IP datagram (0x0800 in Ethernet)
- Frame destination MAC address gives next hop



Encapsulation Across Multiple Hops

- Each router in the path from the source to the destination:
  - *Unencapsulates* incoming datagram from frame
  - Processes datagram - determines next hop
  - *Encapsulates* datagram in outgoing frame
- Datagram may be encapsulated in different hardware format at each hop
  - Datagram survives entire trip across Internet
  - Frame only survives one hop