IT441: Network Servers & Infrastructure

CLASS 8 : 21 Apr 2005
13:30 – 16:15

Last Time:

- midterm; and before that:
- the Internet
This Time

- routing, routing protocols
- transport protocols

Routing Tables

- routing info kept in tables
  - on hosts
  - in routers
- routes in tables may be
  - static
    - used by hosts
    - may also be used in routers

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Genmask</th>
<th>Flags</th>
<th>Iface</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.0</td>
<td>*</td>
<td>255.255.255.0</td>
<td>U</td>
<td>eth0</td>
</tr>
<tr>
<td>default</td>
<td>192.168.0.1</td>
<td>0.0.0.0</td>
<td>UG</td>
<td>eth0</td>
</tr>
</tbody>
</table>
Routing Tables

- routing info kept in tables
- routes in tables may be
  - static
  - dynamic
    - initialize from 'static' table
    - adjust with new information while running
    - used by routers
    - generally not used by hosts

Simple Router Setting

- consider small company network:

```
R_1
h_1  h_2  h_3  ...  h_n
```

to ISP and Internet
Scaling Up

- recall goal of universal service
  - any host can reach any other ⇒ must know route to everywhere
- introduce a structure to sets of routers

AS: The Bigger Picture

- consider a collection of networks and related routers to be a ‘group’
  - e.g., a single corporation
  - e.g., a university
  - e.g., an ISP
- group is under a single ‘administrative authority’
- group is called an autonomous system (AS)
- traffic in AS is
  - local (originates or terminates in AS)
  - transit (originates and terminates outside AS)
In or Out

- a network consisting of two or more AS will have
  - routers internal to each AS using an IGP
  - routers external to (between) AS using an EGP
- each AS can choose a particular IGP for its internal routing traffic
  - so can be different in different AS

Optimal Routes

- routers should have the optimal route for any destination
- ‘optimal’ with respect to what criterion? (routing metric)
  - lowest co$t?
  - lowest delay?
  - lowest hop count/distance?
  - lowest jitter?
- costs are assessed by admins of an AS
  - so who doesn’t use these routing metrics?
- can you really have true optimum path?
RIP

- Routing Information Protocol (RIP) [RFC 2453]
  - distance vector, uses hop count, via UDP
  - designed for use over LANs with hw broad/multi-
    cast capability
  - has messages to exchange routing tables
    - can include advisement of a default route
  - comes with UNIX
  - slow convergence
  - passive version for hosts

RIP-2 Packet Format
RIP

- basic message contents:
  
  `<dest_network : distance>`

- distance measured in hops
  - number of networks pkt must travel over

- router receives RIP msg, for each dest_network:
  - if not in my table, add it
  - if lower cost than what I have, replace what I have

- simple to use, little config overhead

OSPFB

- Open Shortest Path First (OSPF) [RFC 2328]
  - link state (not distance vector)
  - multiple metrics, via IP directly
  - type of service routing via TOS field in IP hdr
  - load balancing
  - information exchanged among OSPF routers by Link State Advertisement messages
OSPF and Areas

- OSPF allows partitioning of an AS into subregions called areas
  - AS admin determines layout
  - each AS has an “area 0” backbone allowing access from any area to any area in the AS
- each area’s routers use OSPF to communicate info about that area
- designated ‘edge’ routers join separate areas
  - exchange summaries of what they can reach
  - reduces size of data set exchanged

OSPF and Network Topologies

- three kinds of connections and networks:
  1. point to point between two routers
  2. multi-access network with broadcasting
  3. multi-access networks without broadcasting

- above is achieved by abstracting collection of routers, networks and links into directed graph
  - vertices are routers or networks
  - edges are connections
  - see Comer §27.13 for example
OSPF Protocols

- hello protocol
  - checks that links are operational and elects designated routers and backups
  - hello packets sent every “hello interval”
  - run distributed elections
- exchange protocol
  - used to synchronize routing databases
- flooding protocol
  - distributes updates
  - includes acknowledgment mechanism

EGP

- what do routers ‘between’ AS say to each other?
  - info about routes offered in their respective AS
- how do they say it?
  - using an EGP
  - most widely used is Border Gateway Protocol, BGP (currently version 4) [RFC 1771]
Border Gateway Protocol (BGP)

- gateways exchange routing information among autonomous systems
- gateway "advertises" that it can reach certain IP networks and its distance to them
- distance-vector based
- distance metrics not standardized
- exterior routing through autonomous systems
  - BGP4 allows use of > 1 path for backup
  - to do this, advertise higher metric to undesirable paths
- uses TCP to exchange routing information

BGP, cont’d

- supports enforcement of AS policy
- important to distinguish between:
  - advertising: telling another AS what nets are reachable through this one
  - routing: setting up the tables to support forwarding
Multicast Routing Protocols

- distance vector:
  - each node maintains distance (multicast hops) from itself to each dest. (e.g., RIP)

- multicast version: distance vector multicast routing protocol (DVMRP)
  - a tunnel counts as one hop
Multicast Routing Protocols, cont’d

- link state:
  - each router creates a ‘link state packet’ for each link to a neighbour containing cost
  - these packets xmit’ed throughout network whenever cost changes (e.g., OSPF)

- multicast version: multicast open shortest path first (MOSPF)
Multicast Routing Protocols, cont’d

- border gateway:
  - exterior routers advertise distance to other networks they can reach (e.g., BGP)

- multicast border gateway protocol (MBGP) for inter-domain routing
Multicast Routing Protocols, cont’d

- newest: Protocol Independent Multicast (PIM) routing
  - adds “relay points” for multicast–sparse portions of the Internet
  - PIM–Sparse Mode (PIM–SM) intended to avoid redundant, hand– provisioned tunnels
  - PIM–dense (PIM–DM) for multicast–dense portions is like DVMRP
  - for SSM, experimental PIM–SSM
  - primarily available in Cisco routers

Types of Service

- connectionless:
  - fast, easy, low overhead
  - each pkt sent as independent event
  - may or may not arrive; may arrive more than once
  - each pkt may follow different route to dest
  - no assurances of reliability, correct ordering, …

- connection–oriented:
  - higher overhead: e.g., setup and tear–down
  - assures reliable, correctly ordered delivery of pkts
  - uses virtual–circuit connection
Delivery Services

1. best effort
2. semi-reliable/unordered
   - sender notified re. receipt of pkts
3. reliable
   - delivers correct pkts in correct order to next layer

IP: Internet Protocol

- delivery protocol:
  - packet src → dest
  - over internet or Internet (i.e., possibly crossing multiple networks)
- simple packet format: header + data
- no checksum coverage of payload
- used by protocols we've already seen, including:
  - OSPF
  - ICMP
  - IGMP
IP: Internet Protocol

- best–effort delivery: makes best try to deliver, but doesn’t handle problems of datagrams being:
  - duplicated
  - delayed or arriving out–of–order (sequence)
  - corrupted
  - lost entirely
- solving these problems is left to higher–level protocols

Need More Than IP?

- IP is a delivery mechanism
  - identifies source and destination by IP address
- to be useful, must have software on dest
  - listening for pkts to arrive
    - i.e., someone to deliver the pkt to
- how identify which of possibly many listeners pkt should go to?
  - no provision in IP dgram for recipient identification beyond host IP address
Transport Protocol

- need a transport protocol:
  - can identify individual end–point (process)
  - both source process and dest process
  - a.k.a. “end-to-end” protocol
- implemented as a separate layer in IP:

<table>
<thead>
<tr>
<th>Layer</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION</td>
<td>5</td>
</tr>
<tr>
<td>TRANSPORT</td>
<td>4</td>
</tr>
<tr>
<td>INTERNET</td>
<td>3</td>
</tr>
<tr>
<td>NETWORK INTERFACE</td>
<td>2</td>
</tr>
<tr>
<td>PHYSICAL</td>
<td>1</td>
</tr>
</tbody>
</table>

UDP: User Datagram Protocol

- simple, cheap, fast
- connectionless
- best–effort
- message–oriented
- no control msgs
- n:m interaction; n,m ≥ 1
**UDP: User Datagram Protocol**

- message oriented:
  - targeted for sending individual msgs, one per pkt
- UDP does no fragmenting nor reassembly:
  - each msg must fit within one UDP dgram
  - penalize processes sending small amounts of data
  - can penalize processes sending large messages
- uses IP for delivery
  - hence max size as determined by IP

---

**UDP: User Datagram Protocol**

- interaction model allows for IP
  - unicast: one sender → one receiver
  - multicast: one sender → many receivers
  - broadcast: one sender → everyone
Who's Listening?

- pkts are intended for a process at dest
  - having originated from a process at src
- different platforms have different ways to identify their processes
  - no one standard OS process identification method
- define 'our own' method of identifying listening processes: by port number
- port numbers identify a service
  - whose implementation is platform specific
  - but that meets protocol definition of the service

UDP Ports

- port numbers are 16-bit values (0 to 65535)
- some port numbers are assigned:
  - 0 – 1023 are “well-known” port numbers
  - originally listed in RFC 1700
  - now found online at:
    http://www.iana.org/assignments/port-numbers
- see /etc/services on UNIX platforms for list, e.g.,
  - daytime 13/udp
  - http 80/udp
UDP Datagram

- has port numbers for source and dest
  - identifies endpoints of the dgram
- checksum covers UDP header and data
  - and is optional
  - if checksum is wrong, pkt is silently discarded
- what don’t you see in the datagram?
  - why aren’t they there?

UDP Datagram Pseudo–Header

- pseudo header used only in checksum calculation
UDP Checksums

- if not being provided—bad idea — then report as all 0–bits
  - receiver understands this to mean no checksum was calculated
  - what if actual checksum value was 0?
- bad because:
  - no one else checks integrity of header info
  - some link layer protocols have no checksums of their own

A Better Model...

- applications want
  - to focus on the nature of the app
  - not to deal with handling communications reliability problems
- need a transport protocol able to deliver end-to-end reliable transport:
  - no duplicate packets
  - no out of order packets
  - no missing packets
  - no errors in packets
...Hard to Build

- easy to ensure such reliability across single LAN
- hard to ensure across internet with
  - multiple networks of different capabilities
  - routers with different capabilities
  - nodes and routers that can fail at random moments

Transport Control Protocol: TCP

- TCP is IP's other transport protocol
- uses IP for delivery, yet provides:
  - connection-oriented service
  - point-to-point interaction (1:1)
  - complete reliability
    - startup
    - data exchange
    - shutdown
  - fdx communication
  - stream data interface
TCP: Connection–Oriented

- uses virtual connection between src and dest
- connection:
  - is set–up by src and dest before data exchanged
  - used for data exchange
  - is shutdown when done
- need for control messages of some kind

Reliability

- TCP provides reliability by:
  - handling re-transmits of data believed not received with adaptive timers
  - using sequence numbers in packets (segments)
  - performing flow control with windows
  - performing congestion control
ACKs, Timers, Re-Transmits

- each received TCP pkt is ACKnowledged
  - so sender knows it has been received
- each TCP pkt sent starts a timer
  - if timer counts down to 0 before ACK received, sender sends same pkt again
- this handles data corruption errors as well
- see Comer fig. 25.2

One Time Fits All?

- what should time-out timer be set to?
One Time Fits All?

- what should time-out timer be set to?
  - on Ethernet LAN can be small ms. range
  - on satellite link needs to be hundreds of ms

TCP’s timer

- what should time-out timer be set to?
  - on Ethernet LAN can be small ms. range
  - on satellite link needs to be hundreds of ms
- TCP uses adaptive timers based on measured round trip time (RTT)
  - measures time from send to ACK for a segment
  - keeps weighted average
  - uses statistical formula to adjust timeout value to follow conditions and avoid excess re-xmits
Flow Control

- TCP manages flow of data using a window
  - size of memory buffer to hold received data
  - initialized (both sides) during connection set-up
  - window = amount of buffer space currently remaining
  - src/dest use window advertisement to inform each other of buffer availability
  - if receive 0-window, stop sending until receive non-zero window
Congestion Control

- network is congested when traffic load is too high
- congested networks increase delay in segment delivery
- can lead to increase in re-xmits

Congestion Control

- TCP uses packet loss as metric of congestion
  - if congestion, reduce rate of (re)transmission
- to reduce rate:
  - make window artificially smaller
- recovering from congestion:
  - send small segment (< window size)
  - if ACKed without further loss, send two small segments
  - continue exponential ramp up until:
    - loss encountered
    - reached 50% window size, switch to linear
TCP Segment

- TCP segment format [RFC 793]
  
  [Diagram of TCP segment format]

source port number [16] ... dest port number [16]
# TCP Segment

<table>
<thead>
<tr>
<th>SRC PORT</th>
<th>DEST PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DATA

**sequence number [32]**

(outbound): number of first byte of data in this segment – used by receiver to re-sequence data received out-of-order

---

<table>
<thead>
<tr>
<th>SRC PORT</th>
<th>DEST PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DATA...

**acknowledgement number [32]**

(inbound): next expected sequence number, i.e., number of first byte of data in next expected segment;
TCP Segment

<table>
<thead>
<tr>
<th>SRC PORT</th>
<th>DEST PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQUENCE NUMBER</td>
<td>ACK NUMBER</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>window [16]</td>
</tr>
</tbody>
</table>

(inbound): how much receiver buffer space is left for data starting at indicated ack number

TCP Segment

<table>
<thead>
<tr>
<th>SRC PORT</th>
<th>DEST PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQUENCE NUMBER</td>
<td>ACK NUMBER</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>window</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

header length [4]
TCP Segment

<table>
<thead>
<tr>
<th>SRC PORT</th>
<th>DEST PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQUENCE NUMBER</td>
<td></td>
</tr>
<tr>
<td>ACK NUMBER</td>
<td></td>
</tr>
<tr>
<td>HLEN</td>
<td>WINDOW</td>
</tr>
<tr>
<td>DATA...</td>
<td></td>
</tr>
</tbody>
</table>

RFU [6]

TCP Segment

ACK NUM is valid
PSH: rcvr push to app as soon as possible
SYN: sync seq nums to start connection
FIN: sender has finished
URG pointer is valid
RST: reset connection
### TCP Segment

<table>
<thead>
<tr>
<th>SRC PORT</th>
<th>DEST PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEQUENCE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK NUMBER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HLEN</th>
<th>RFU</th>
<th>CODE BITS</th>
<th>WINDOW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Checksum [16]**

Computed on all of TCP header and data payload.

---

### TCP Segment

<table>
<thead>
<tr>
<th>SRC PORT</th>
<th>DEST PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEQUENCE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK NUMBER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HLEN</th>
<th>RFU</th>
<th>CODE BITS</th>
<th>WINDOW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHECKSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Urgent Pointer [16]**

Offset beyond current seq num where urgent data ends.
TCP Segment

<table>
<thead>
<tr>
<th>SRC PORT</th>
<th>DEST PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQUENCE NUMBER</td>
<td></td>
</tr>
<tr>
<td>ACK NUMBER</td>
<td></td>
</tr>
<tr>
<td>HLEN</td>
<td>RFU</td>
</tr>
<tr>
<td>CHECKSUM</td>
<td></td>
</tr>
</tbody>
</table>

DATA...

data (if any)  options (if any)

TCP Handshaking

- TCP uses “3-way handshake” both to set-up and shutdown a virtual connection

**active open**

**passive open**

| SYN 522173/0/1024 |
| SYN 82441/0/1024 |
| ACK 522174/1/1024 |
| ACK 82442 |
TCP Handshaking

- shutdown is similar:

  - FIN_WAIT_1 → FIN
  - ACK of FIN → CLOSE_WAIT
  - FIN_WAIT_2 → FIN
  - ACK of FIN → LAST_ACK
  - TIME_WAIT → FIN
  - ACK of FIN → CLOSED

- but what if only one side is done?

Getting Started

- how does a host get 'on the air' when starting up?
- needs to know certain things (in order):
  - its own IP address
  - its own address mask
  - a default gateway
- usually can be stored in local disk files
  - but what if there's no local disk?
Step By Step

- problem:
  - have hardware address (read from ROM on interface card)
  - need IP address
- similar problem seen before?
  - have IP address and need hardware address
  - used ARP to resolve
- this is reverse of earlier problem, so...

RARP

- Reverse Address Resolution Protocol: RARP [RFC903]
- given hardware address, return corresponding IP address
- packet format very similar to ARP
  - “op type” different values
    - 3 for RARP request
    - 4 for RARP response
- request normally broadcast, response normally unicast
RARP

- server receives request
  - e.g., Ethernet frame type 0x8035
- resolving binding usually requires server to consult a file
  - containing IP:hardware_address entries
- server sends unicast reply
  - to whom?

ICMP Address Mask request/reply

- recall ICMP has two main jobs:
  - carry error messages back to a sender
  - perform request/reply information acquisition
    - distinguished by type field in pkt
      - see slide 04.20
- host sends broadcast ICMP msg
  - op type 17: address mask request
- server replies with unicast ICMP msg
  - op type 18: address mask reply
ICMP Router Solicitation

- ICMP to the rescue again
  - msg type 10: router solicitation
  - msg type 9: router advertisement
- on startup, host sends 3 requests, 3 seconds apart
  - stops as soon as first router advertisement arrives
- routers use type 9 routinely to advertise routes
  - can advertise multiple routers per advertisement
  - gives each a ‘preference’ level from its p.o.v.

Or, All At Once

- a host can get all of this information (and more)
  in a more convenient, single-step operation
- uses a different protocol: Bootstrap Protocol
  BOOTP [RFC951]
- BOOTP can, in a single interaction, provide:
  - IP address
  - router IP address
  - name of bootstrap file to load (to get OS)
BOOTP Messages

OP [8]: 1 = request
2 = reply

HLEN [8]: network hardware type
1 = Ethernet
6 = Ethernet

HOPS [8]: number of servers that forwarded this request
client sets to 0

BOOTP Messages

OP    HTYPE    HLEN    HOP_CNT

Transaction Identifier [32]:
client sets to x; matches response to request
BOOTP Messages

Seconds elapsed [16]: set by client when sends request as time since began attempt to boot

client hardware address [128]: set by client to its interface hardware address

client IP address [32]: set to 0 if client doesn't know its IP addr
IP address of client if it does know
### BOOTP Messages

<table>
<thead>
<tr>
<th>OP</th>
<th>HTYPE</th>
<th>HLEN</th>
<th>HOP CNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSACTION NUMBER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SECONDS ELAPSED</td>
<td></td>
<td></td>
<td>unused</td>
</tr>
<tr>
<td>CLIENT IP ADDR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLIENT HARDWARE ADDRESS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Router IP address [32]: set by server to default router’s IP address

Your IP address [32]: set by server to IP address of requesting client

### BOOTP Messages

<table>
<thead>
<tr>
<th>OP</th>
<th>HTYPE</th>
<th>HLEN</th>
<th>HOP CNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSACTION NUMBER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SECONDS ELAPSED</td>
<td></td>
<td></td>
<td>unused</td>
</tr>
<tr>
<td>CLIENT IP ADDR</td>
<td></td>
<td></td>
<td>your IP addr [32]</td>
</tr>
<tr>
<td>YOUR IP ADDR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROUTER IP ADDR</td>
<td></td>
<td></td>
<td>server IP addr [32]</td>
</tr>
<tr>
<td>CLIENT HARDWARE ADDRESS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Server host name [512]

Boot file name [1024]: name of bootstrap file
**BOOTP Messages**

<table>
<thead>
<tr>
<th>OP</th>
<th>HTYPE</th>
<th>HLEN</th>
<th>HOP_CNT</th>
<th>TRANSACTION NUMBER</th>
<th>SECONDS ELAPSED</th>
<th>UNUSED</th>
</tr>
</thead>
</table>

**vendor-specific area [512]:**

**BOOTP: how delivered?**

- RARP, ICMP use IP datagrams directly
- BOOTP uses UDP (port 67, 68)

- client usually sends as link-layer broadcast
  - with IP address 255.255.255.255 (limited b’cast)
  - what’s source IP address?
Chicken and Egg Problem

- booting client asks some server for info
  - uses broadcast
- server responds sending UDP unicast reply
  - since server knows IP address
- but: think about what the server does...

Chicken and Egg Problem

- but: think about what the server does...
  - server is a process running on some host
  - has IP address of target
  - will send a UDP unicast to the designated IP addr
  - lower-level wants to map target IP address to local hardware address
  - looks in ARP table but doesn’t find target IP addr
Chicken and Egg Problem

- **but: think about what the server does...**
  - server is a process running on some host
  - has IP address of target
  - will send a UDP unicast to the designated IP addr
  - lower-level wants to map target IP address to local hardware address
  - looks in ARP table but doesn’t find target IP addr
  - so sends ARP request for anyone to tell it
  - but nobody knows, least of all client who asked

Chicken and Egg Problem

- **solutions:**
  1. BOOTP server software should make entry in server host’s ARP table before sending reply
  2. use broadcast for reply (not highly recommended)