Administrivia/Last Time

- Project discussion
- Last Time
  - Wrapped up Multicast
  - IP Next Generation (IPNG)
- Today
  - Start End to end protocols
PIM Operation

RP = Rendezvous point
- - - - Source-specific tree for source R1

Join

Host

RP

R1
R3
R5
R2
R4

(a)

RP

R1
R3
R5
R2
R4

(b)

RP

R1
R3
R5
R2
R4

(c)

RP

R1
R3
R5
R2
R4

(d)

RP = Rendezvous point

- - - - Source-specific tree for source R1

Join

Host

RP

R1
R3
R5
R2
R4

SUNY-BINGHAMTON – CS428 SPRING ’05 LEC. #17
MBone

- Mbone (Multicast Backbone)
  - Large scale experiment in supporting multicasting in the internet
  - Collection of Islands supporting multicasting (overlay network)
  - Each Island has a multicast router (e.g., a host running mrouted)
  - Routers connected via tunnels (IP-in-IP)
  - DVMRP has been the routing protocol/being replaced by PIM

- Sample applications: video conferencing (vic); shared whiteboard (wb)

- Highlights the difficulty of adding functionality to the internet
  - Qbone, 6bone

- Book on Mbone: http://www.savetz.com/mbone/
IPv6 – why

- Address depletion is the main driver; but
  - **Difficult to add features post-facto to IPv4** (e.g., Multicast)
  - Since we are taking the painful decision to upgrade IP, might as well fix everything that is wrong with it
  - Address routing table growth (approx. 100,000 entries in backbone)
  - Easier to configure/use
  - Simplify packet processing
  - Multicast
  - Security
  - Quality of Service
  - Real time traffic support
  - Flow identification
  - IP billing
  - ...

- A lot of what is “optional” in IPv4 is required in IPv6
IPv6 Routing

- IPv4 backbone has big routing table size; headache for backbone operators
- IPv6 addressing specification restricts the number of routing table entries by using architecture-enforced “routing aggregation”
- Hierarchical routing
  - Geographically
  - Provider based (change of provider = change of address)
- How does NAT compare in terms of addressing? in terms of changing provider?
IPv6 Routing (cont’d)

- Aggregatable Global address
  - Top level – 13-bits (3 bits to indicate aggregatable unicast addresses)
  - Next level – 48-bits; Site level – 16-bits

- CIDR from the beginning, so the divisions are not too important

- Only 8192 entries in the top most level
Transition Plan

- One of the basic requirements is to make the transition from IPv4 to IPv6 easy – why?
  - RFC 1933

- In the early stages (today)
  - Internet is IPv4
  - Most nodes are IPv4
  - Some IPv6 nodes

- IPv6 nodes use their equivalent IPv4 compatible address

- IPv6 enabled nodes run both IPv4 and IPv6 stacks

- Use tunneling when crossing IPv4 network

- Similar to Mbone, 6bone spans 50 countries
Transition Plan

- Late stages – IPv6 most everywhere
- Some IPv4 nodes
- Not enough IPv4 addresses for all nodes
  - Must rely on translation
- IPv4 relegated to an option within the IPv6 stack
- Not there yet; it remains to be seen what will happen as we approach address depletion
But is IPv6 on the way?

• Not clear
  – Military has definite timeline; all military machines support IPv6 as of past October, and will completely switch over to IPv6 by 2008
  – But it's much less sure in the commercial world (which ends up determining what happens anyway)
    * They are motivated by $$: risks and opportunities
    * Risk not high enough: NAT and solutions based on it can overcome the address shortage problem – if we can solve the peer to peer addressability problem
    * Opportunity not immediately available: Not clear they can make money out of the additional features in IPv6
Alternative – Getting through NAT
Getting through NAT (2)
NUTSS

SIP INVITE message
To: bob@sip.bar.com
From: steve@sip.foo.com
SDP: 1.1.1.1:1234

SUNY-BINGHAMTON – CS428 SPRING ’05 LEC. #17
Discussion: IP/Network Layer

• So far
  – Discussion of digital communication (basics; no details)
  – Directly connected networks – point-to-point and shared medium
  – Switched networks (bridges vs. routers)
    * Global addressability
    * Heterogeneity (fragmentation, address translation)
    * Forwarding model (datagram, VC, source routing...)
    * Routing
    * Making it scale
    * Multicast
    * Limitations/the future – IPv6

• IP is the workhorse of the Internet
End-to-End (Transport) Protocols

• IP and the network layer provide host-to-host connectivity across a scalable heterogeneous network

• Unfortunately, IP is a best-effort network; it can
  – Drop Messages
  – Reorder messages
  – Duplicate messages
  – Delay messages a long time
  – Limit size of messages

• How do these features compare with the requirements of applications?
  – End-to-End Protocols provide better service models to applications

• How to get from host-to-host to process-to-process communication?
End-to-end Services

- Ideally: transport protocol worries about the end-to-end service provided to the application; it does not care about the communication path.

- What are common end-to-end services of interest?
  - Allow multiple processes on a host (is this possible with IP?)
  - Guarantee message delivery
  - Guarantee ordered delivery
  - No duplicates
  - Arbitrary size messages
  - Flow control (don't overflow receiver)
  - Congestion control (don't overflow network) – is this a service?
  - other? (QoS, Encryption, Synchronization...)

- Are these needed by all applications?
Discussion

- Why is end-to-end operation different from link-level communication
  - At the link level 2 ends on the link communicate with each other
  - End-to-end, 2 ends of the connection communicate with each other
Transport Protocols

- User Datagram Protocol
  - Basic transport (process to process IP)
  - Many other protocols built on top of it

- Transmission Control Protocol (TCP)
  - Reliable bytestream; many bells and whistles

- Others, including:
  - Realtime Transmission Protocol (RTP/RTCP)
  - Remote Procedure Call (RPC)
  - Stream Control Transmission Protocol (SCTP)

- Multicast Transport Protocols
  - MFTP, PGM, etc..

- Point – TCP is not the only transport protocol
The simplest end-to-end protocol is to extend IP to recognize multiple processes per host.

UDP provides a simple demultiplexing key to differentiate between processes – no other functionality is supported.

- e.g., when a message arrives, if queues are full it is dropped.
- Why is this interesting?

What should be used as a demultiplex key?

- How about process id?
• Port Numbers are used as a demultiplex key
  – A Port is a logical “mailbox” which is associated with a process

• How does a process know the “key” for a process it wants to communicate with?
  – Well known port numbers for most servers (e.g., http server at port 80; defined in RFC 1700)
  – Otherwise, by out-of-band agreement

• Try to tie this in with socket programming
IP Revisited

- How does the packet get to “UDP” in the first place??
- Protocol numbers also defined in RFC 1700
**UDP**

- UDP checksum is optional; when used, it checksums
  - the whole message body (including UDP header)
  - Psuedoheader from IP

```
0  7  8  15  16  23  24  31
+-------------------------------+
| source address               |
+-------------------------------+
| destination address          |
+-------------------------------+
| zero | protocol | UDP length |
+-------------------------------+
```

- Recall that IP checksum was on the IP header only
- Idea: protection against misrouted datagrams
End-to-End (Transport) Protocols

- Network layer (IP) provide host-to-host connectivity across a scalable heterogeneous network
- Unfortunately, IP is best effort. It can: reorder, duplicate, delay, or drop messages
- To accommodate applications
  1. Need to build service models that are more suitable for them
  2. Need to provide process to process connectivity, not just host to host
- These are the traditional roles of the transport (or end-to-end) protocols
UDP – User Datagram Protocol

- The simplest end-to-end protocol is to extend IP to recognize multiple processes per host

- UDP provides a simple demultiplexing key to differentiate between processes – no other functionality is supported
  - e.g., when a message arrives, if queues are full it is dropped
  - Why is this interesting?

- What should be used as a demultiplex key
  - How about process id?
Demultiplexing

- **Port Numbers** are used as a demultiplex key
  - A Port is a logical “mailbox” which is associated with a process
- How does a process know the “key” for a process it wants to communicate with?
  - Well known port numbers for most servers (e.g., http server at port 80; defined in RFC 1700)
  - Otherwise, by out-of-band agreement
- Try to tie this in with socket programming
Transmission Control Protocol (TCP)

- A reliable connection-oriented service model
  - Reliable: everything gets there exactly one time
  - connection-oriented: in-order delivery of a stream of bytes
  - Full duplex
- Most widely used and most carefully “tuned” transport protocol on the internet
- Like UDP supports multiple processes per host (also using port numbers)
- TCP implements both flow-control and congestion control (will discuss in detail later)
- Connection establishment is needed
- Sending process writes some bytes (any number)
- TCP breaks into segments and sends via IP
- Receiving process reads some bytes (any number)
- How big is the segment?
- When does TCP send the segments?
- How to implement Reliability and in-order delivery?
TCP – Overview (cont’d)

• Common choice for Maximum Segment Size (MSS): maximum size that will not cause IP to fragment locally
  – What is this equal to?

• When to send a “segment”?
  1. When there is enough data to send an MSS
  2. If the application demands an immediate send
  3. Set a timer when you send a segment; send again when it fires

• Why three different ways?

• Packet boundaries are not visible to a process

• Reliability? Need some form of ARQ (isn’t it supported at link layer?)

• In-order delivery? Don’t allow a receive until all preceding data has arrived
**TCP**

- Source port and Destination port identify processes
  - Along with source/destination IP addresses form a unique connection identifier – true/false?
- TCP runs a sliding window algorithm
  - Acknowledgements used to ack received segments
  - Sequence number of the first byte in the segment
  - Advertised window is the size of the window at the receiver (flow control)
- Checksum is identical to UDP
Connection Establishment

- The sequence number is the number of the byte received last + 1
- Initially randomly picked
- Note Duplex operation
- Normal operation occurs within the established state
- Why timewait state?
- Track connection establishment and teardown
TCP “Established” Operation

- Strategy – Sliding window ARQ
  - Use ACKs and Sequence numbers
    * ACK sets a flag bit to say that the ACK field is valid
  - Flow control using advertised window
  - Congestion control