Problem: There is more than one network (heterogeneity & scale)

Internetworking:
- Internet Protocol (IP)
- Routing and scalability
- Group Communication

Internetworking

Hongwei Zhang

http://www.cs.wayne.edu/~hzhang

Every seeming equality conceals a hierarchy.

--- Mason Cooley

Acknowledgement: this lecture is partially based on the slides of Dr. Larry Peterson
Outline

- Algorithms
- Scalability
Outline

- Algorithms
- Scalability
Overview

- **Forwarding vs. Routing**
  - forwarding: to select an output port based on destination address and routing table
  - routing: process by which routing table is built

- **Network as a Graph**

![Graph Image]

- Problem: Find lowest cost path between two nodes
- Prominent factors affecting routes used
  - Topology: relatively static, especially in wired networks
  - Traffic load: more dynamic
  - Others: security, reliability, etc

Q: how would you build routing tables in a distributed manner?
Distance Vector routing

- Based on distributed Bellman-Ford algorithm

- Objective: enable each node to maintain a set of triples
  - \((\text{Destination}, \text{Cost}, \text{NextHop})\)

- Approaches:
  - Directly connected neighbors exchange updates
    - periodically (on the order of several seconds; e.g., 30 seconds in RIP)
    - whenever table changes (called \textit{triggered} update)
  - Each update is a list of pairs: \((\text{Destination}, \text{Cost})\)

- Update local table if receive a “better” route
  - smaller cost, or
  - came from next-hop

- Refresh existing routes; delete if they time out --- \textit{soft state}
Maintain soft-state.

$$\forall$$

else

$$f_i$$

$$k = n_i \land c_i^j = c_i^j - 1 \rightarrow c_i^j := c_i^j + 1$$

$$N_j := k$$

$$k = n_j \land c_j^i = c_j^i - 1 \rightarrow c_j^i := c_j^i + 1$$

for each $$i$$

Then fires V local trigger $\rightarrow$ broadcast ($$\langle i, c_i^j \rangle$$)

Actions

Node setup $\rightarrow$ initialize variables, set up tiers etc., including set $$c_i^j$$, $$N_j$$, $$k$$

Example: routing table at node B

<table>
<thead>
<tr>
<th>Destination</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>A</td>
</tr>
</tbody>
</table>
In practice, rather than advertising the cost of reaching other routers, the routers advertise the cost of reaching “networks”

- E.g., router C advertises distances to networks 2 and 3 as 0, to networks 5 and 6 as 1, etc
Routing Loops: focus on node A

- **Example 1**: (F, G) breaks; no loop
  - F detects that link to G has failed
  
  - F sets distance to G to infinity and sends update to A
  - A sets distance to G to infinity since it uses F to reach G
  - A receives periodic update from C with 2-hop path to G
  - A sets distance to G to 3 and sends update to F
  - F decides it can reach G in 4 hops via A

- **Example 2**: looping & count-to-infinity
  - link from A to E fails

  - A advertises distance of infinity to E
  - (B and C have advertised a distance of 2 to E)
  - B decides it can reach E in 3 hops; advertises this to A
  - A decides it can read E in 4 hops; advertises this to C
  - C decides that it can reach E in 5 hops …
More on routing loops

- Two types of effect
  - Bouncing effect: loop will break in the end, i.e., transient loops
  - Count-to-infinity: loop will not break
Routing loops: Bouncing effect
Routing loops: Count to infinity
Loop-Breaking techniques

- **Heuristics**
  - Set infinity to a fixed number (e.g., 16 in RIP)
  - Deal with loops involving 2 nodes
    - *Split horizon*: when a node B sends routing updates to a neighbor A, B does not send routes learned from A
    - *Split horizon with poison reverse*: B still sends the routes learned from a neighbor A, but with distance value being “infinity” so that A will not use B as next-hop at all

- **Guaranteed loop freedom**
  - Subtree removal upon link failure; two-way diffusing computation
Fault propagation in D-V routing

Ideally, only h needs to correct (i.e., correct) its state. But the state corruption at h may well propagate unboundedly until the boundary of the network.
Guaranteed fault containment & loop freedom

- **The cause for fault propagation:**
  
  "correction" action always lags behind "fault propagation" action

- **Solution:**
  
  - the "source of fault propagation (such as node 8)" detects the fault propagation, and initiates a "containment" action that catches up with and stops the "fault propagation" action
  
  - avoid forming cycles during stabilization, and remove existing cycles fast
Approach: layering of diffusing waves

- Use three diffusing waves such that
  - Each diffusing wave has different propagation speed
    - Speed is controlled by introducing delay in action execution
  - A mistakenly initiated layer-i wave $W_i$ is contained and prevented from propagating unbounded by a layer-(i+1) wave that is initiated at the same node which has initiated $W_i$
  - The top-layer wave self-stabilizes itself locally upon perturbations

- Specifically,

```
Super-containment Wave V_2
<table>
<thead>
<tr>
<th>Containment Wave V_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stabilization Wave V_0</td>
</tr>
</tbody>
</table>
```

$V_2 > V_1 > V_0$

$V_1 > V_0$

Link state routing

Motivation

- Fast, “loopless” convergence

- Easier to support precise metrics (e.g. throughput, delay, cost, reliability) and, if needed, multiple metrics;

  Easier to incorporate external routes in terms of "precise metric to exit"
  
  - As a result of loop freedom, and thus not worrying about upper limit on route cost

- Support for multiple paths to a destination (for load balancing)
Link State routing (contd.)

- **Strategy**
  - send to all nodes (not just neighbors) information about directly connected links (not entire routing table)

- **Overhead control**
  - Low frequency of periodic flooding of local link state; e.g., *once every a few hours*
  - Triggered update when topology/local-network-condition changes
L-S routing: reliable flooding

- Link State Packet (LSP)
  - *id* of the node that created the LSP
  - *cost* of link to each directly connected neighbor

- Two basic issues in flooding link states
  - Termination control: the flooding has to stop
    - Via time-to-live (TTL) for this packet
  - Version control: order of states
    - Via sequence number (SEQNO)
Reliable flooding (contd.)

- Each node generates new LSP periodically
  - increment SEQNO

- When receiving a LSP,
  - store it locally, if it is the most recent LSP for the corresponding originator
    - decrement TTL of the stored LSP
    - discard when TTL=0
  - forward a “recent/fresh”, newly received LSP to all nodes but one that sent it

- Reliable message exchange between neighbors (using acks & retransmission)
Reliable flooding (contd.)

- A node “ages” stored LSPs by decrementing their TTLs

- When TTL reaches 0, refloods LSP with TTL=0 so that all the nodes in the network removes the corresponding LSP
  - Q: is this necessary for the correctness of L-S routing?

- When a node reboots, it starts SEQNO at 0
  - Either other nodes have removed the old LSPs corresponding to this node (if the node has failed for a long time)
  - Or the node receives LSP from other node with larger sequence number (with TTL=0), and set its sequence number to the number plus 1
L-S routing: Route Calculation

- Dijkstra’s shortest path algorithm

- Let
  - \( N \) denotes set of nodes in the graph
  - \( l(i, j) \) denotes non-negative cost (weight) for edge \((i, j)\)
  - \( s \) denotes this node
  - \( M \) denotes the set of nodes incorporated so far
  - \( C(n) \) denotes cost of the path from \( s \) to node \( n \)

\[
M = \{s\}
\]

for each \( n \) in \( N - \{s\} \)
\[
C(n) = l(s, n)
\]

while \( (N \neq M) \)
   \[
   M = M \cup \{w\} \text{ such that } C(w) \text{ is the minimum for all } w \text{ in } (N - M)
   \]
   for each \( n \) in \( (N - M) \)
   \[
   C(n) = \text{MIN}(C(n), C(w) + l(w, n))
   \]
Routing metrics (in the context of ARPANET)

- Original ARPANET metric
  - measures number of packets queued on each link, i.e., queue length
  - (-) took neither latency or bandwidth into consideration

- New ARPANET metric: delay based (including queuing delay)
  - stamp each incoming packet with its arrival time ($AT$)
  - record departure time ($DT$)
  - when link-level ACK arrives, compute
    \[
    \text{Delay} = (DT - AT) + \text{TransmissionTime} + \text{Latency},
    \]
    - with "TransmissionTime" and "Latency" capturing the BW and latency of a link
  - if timeout, reset $DT$ to departure time for retransmission

- link cost = average delay over some time period
Routing metrics (contd.)

( contd. )

- (-) instability in the case of high traffic load: queuing delay is traffic sensitive
  - This cause many links to be IDLE when traffic load is high 😞
- (-) range of link values was too large
  - e.g., the cost of a link (e.g., satellite link) could be more than 127 times greater than the cost of another link (e.g., high speed LAN)
    - a route of 127 hops could be preferred over a direct-link route 😞
Routing metrics (contd.)

- Revised ARPANET metric
  - replaced *Delay* with *link utilization*
  - *smoothing* of estimated link utilization to avoid abrupt changes in link/route cost, so that the prob. of all nodes abandoning a link is small
  - *compressed dynamic range* of route cost
Routing metrics (contd.)

- Cost is a function of link utilization only at moderate to high loads.
- Cost of a highly loaded link is no more than 3 times greater than its cost when idle.
- The most expensive link is only 7 times the cost of the least expensive.
- High speed satellite link is preferred over low speed terrestrial link.
- Acts similar to delay-based metric under light load and to a capacity-based metric under heavy load.

Revised ARPANET routing metric vs. link utilization

(determined through a great deal of trial & error!!!)

Atul Khanna, John Zinky, “The revised ARPANET Routing Metric”, ACM SIGCOMM’89
Theoretical foundation & tools for analyzing routing behavior?

- Unsolved challenge problem !!!

- State of the art: experimental measurement and model building
Mobile IP

- What if node moves?
  - Need to change IP address, and other configurations (such default router/gateway)

- Would DHCP work?
  - Does not support mobility of nodes in the presence of an active session

- Mobile IP
  - Relay at home network
Mobile IP (contd.)

- Home agent relays packets (destined for mobile host) to foreign agent who will then forward the packets to mobile host.

- Route optimization in mobile IP (to deal with the triangle-routing-problem)
  - Direct connection between sending host and foreign agent, via IP tunneling.
Outline

- Algorithms
- Scalability
How to Make Internet Scale

- Address utilization

- Routing
  - Flat vs. Hierarchical Addresses
  - Still Too Many Networks (e.g., thousands, millions …)
    - routing tables do not scale
    - route propagation protocols do not scale
Subnetting

- Limitation of “basic classful addressing”: inefficient use of Hierarchical Address Space
  - class C with 2 hosts: $2/255 = 0.78\%$ efficient
  - class B with 256 hosts (just a little over the limit of class C):
    $256/65535 = 0.39\%$ efficient

- **Subnet**: add another level to address/routing hierarchy
  - More efficient use of IP address space
  - Scalable routing: subnets visible only within site, and being transparent to outside networks
## Subnet mask

- *Subnet masks* define variable partition of host part

<table>
<thead>
<tr>
<th>Network number</th>
<th>Host number</th>
</tr>
</thead>
<tbody>
<tr>
<td>111111111111111111111111</td>
<td>00000000</td>
</tr>
</tbody>
</table>

**Class B address**

**Subnet mask (255.255.255.0)**

<table>
<thead>
<tr>
<th>Network number</th>
<th>Subnet ID</th>
<th>Host ID</th>
</tr>
</thead>
</table>

**Subnetted address**
Subnet Example

**Subnet mask:** 255.255.255.128
**Subnet number:** 128.96.34.0

**Subnet mask:** 255.255.255.0
**Subnet number:** 128.96.33.0

**Forwarding table at router R1**

<table>
<thead>
<tr>
<th>Subnet Number</th>
<th>Subnet Mask</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.96.34.0</td>
<td>255.255.255.128</td>
<td>interface 0</td>
</tr>
<tr>
<td>128.96.34.128</td>
<td>255.255.255.128</td>
<td>interface 1</td>
</tr>
<tr>
<td>128.96.33.0</td>
<td>255.255.255.0</td>
<td>R2</td>
</tr>
</tbody>
</table>

*subnet number:*

IP address AND subnet mask
Forwarding Algorithm (within a network of subnets)

- Route search is based on “destination addr. AND SubnetNum”

\[
D = \text{destination IP address}
\]

for each entry (SubnetNum, SubnetMask, NextHop)

\[
D1 = \text{SubnetMask} \& D
\]

if \(D1 = \text{SubnetNum}\)

- if NextHop is an interface
  
deliver datagram directly to \(D\)

- else
  
deliver datagram to NextHop

- Use a default router if nothing matches
Subtle points of subnetting

- Not necessary for all 1s in subnet mask to be contiguous
  - But “contiguity” is usually assumed in practice (for simplicity and efficiency)

- Can put multiple subnets on one physical network
  - E.g., for creating “Virtual Private Network”

- Subnets not visible from the rest of the Internet (e.g., in routing)
CIDR: Classless inter-domain routing (original name: supernetting)

- Observation: exhaustion of IP address space centers on exhaustion of class B network numbers

- To address the problem,
  - Not assign a class B address unless an organization shows a need for something close to 64K addresses
  - Instead, assign a set of class C addresses if need more than 255 addresses => address utilization is at least 50% (why?)

- Drawbacks of the above approach:
  - Backbone routers need to maintain a lot of routing table entries, for instance, for all small class C network
CIDR (contd.)

- CIDR: to balance routing overhead and address assignment efficiency
  - Assign blocks of contiguous network numbers to nearby networks
  - Use the longest common *network prefix* to represent the whole set of networks
    - Thus, restrict block sizes to powers of 2
  - ID of the aggregate network number: \(<\text{Length}, \text{Value}\rangle\), where \(\text{Length}\) gives the number of bits in the network prefix

Route aggregation with CIDR
CIDR (contd.)

- All routers must understand CIDR addressing

- IP forwarding revisited: e.g.,
  - Two routing entries: 171.69 (a 16-bit prefix), 171.69.10 (a 24-bit prefix)
  - A packet with IP address: 171.69.10.5
    - *Longest match*: used entry for 171.69.10

- Q: how bout a packet with IP address 171.69.20.5?
Route Propagation

- Hierarchical structure: know a smarter router
  - hosts know local router
  - local routers know site routers
  - site routers know core router
  - core routers know everything

- Autonomous System (AS)
  - corresponds to an administrative domain
  - examples: University, company, backbone network
  - assign each AS a 16-bit number

- Two-level route propagation hierarchy
  - interior gateway protocol: intra-domain
    - each AS selects its own
  - exterior gateway protocol: inter-domain
    - Internet-wide standard
Popular Interior Gateway Protocols

- **RIP: Route Information Protocol**
  - developed for XNS (Xerox Network System)
  - distributed with Unix
  - distance-vector algorithm
  - based on hop-count

- **RIP V1**
  - RFC-1058 by Charles Hedrick, June 1988

- **RIP V2**
  - Added support: subnetting, CIDR (proposed in 1996), authentication, and multicast transmission
  - To complement/compete with other IGP protocols such as OSPF? RIP has been deployed in many systems, perhaps more than OSPF when RIP V2 was designed
Popular Interior Gateway Protocols (contd.)

- **OSPF: Open Shortest Path First**
  - recent Internet standard
  - uses link-state algorithm
  - supports load balancing via multiple-path traffic splitting
  - supports authentication (of link-state update)

- **OSPF V1**

- **OSPF V2**
  - Added support: Stub area (where all external routes are summarized by a “default” route), optional TOS support, simplified packet format, corrected engineering issues of V1
Other intra-domain routing protocols

- **GGP (Gateway to Gateway Protocol)**
  - Distance vector protocol used in early Arpanet
  - Somewhat more complex than RIP
    - Routing updates are explicitly numbered and acked (note: links in 1970s tend to be unreliable)
    - Neighboring gateways need to synchronize their clocks for exchanging certain control information
  - April 1979

  - for OSI network layer: on top of CLNP (ConnectionLess Network Protocol)
  - link-state protocol, similar to OSPF
  - Feb. 1990
Other intra-domain protocols (contd.)

- **IGRP (Interior Gateway Routing Protocol)**
  - developed in the mid-1980s by Cisco
  - improvements over RIP:
    - support for composite/multiple metrics
    - conservative protection against loops
      - *Path holddown*: quarantine period after link failure, during which no update is accepted
      - *Route poisoning*: regarding paths with increasing hop-count as “invalid”, and won’t use the path until its hop count is confirmed by another update
    - support for multi-path routing
    - automatic selection of default route

- **EIGRP (Enhanced IGRP)**
  - incorporated DUAL (by J.J Garcia, Sept. 1998) algorithm to guarantee loop freedom
  - support supernets and variable-length subnets
Inter-domain routing

- Split Internet into Autonomous Systems (ASs)
- EGP (Exterior Gateways Protocol)
- BGP (Border Gateway Protocol)
Why split Internet into ASs?

- As Internet grows,
  - *routing overhead* increases (as # of routers increases)
  - *size of routing table* increases (as # of destinations increases)
  - *frequency of routing exchanges* increases (because the failure probability increases as the network size increases)

- the types of routers with different implementations of IGPs increases, thus *maintenance and fault isolation* is difficult

- the large number of routers and the fact that the routers are owned by diff. organizations make it *difficult to deploy new versions of routing algorithms and software*
EGP: Exterior Gateway Protocol

- **Overview**
  - Distance-vector routing
  - designed for tree-structured Internet
  - concerned with *reachability*, not optimal routes

  - RFC 827 by Eric C. Rosen, Oct. 1982; used until the end of 1980s when it is replaced by BGP

- **Limits of EGP**
  - designed for a simple tree topology in early ARPANET, and slow convergence upon loops
  - difficulty in supporting policy routing
  - build upon IP, thus control messages can get lost and instability can be introduced
Why not link-state protocol in inter-domain routing?

Has been experimented in Inter-Domain Policy Routing protocol (IDPR, July 1993). However,

- unscalable to maintain the whole Internet map even at the AS level
  - At the beginning of 1994, the # of ASs was more than 700, whereas the recommended maximum size of an OSPF area is only 200
- needs to solve the “inconsistent routing database” problem (large scale networks) which makes it possible for loops to be formed
  - Therefore, IDPR has to use "explicit source routing" which introduces high overhead
  - To address the high overhead problem, IDPR uses “virtual circuit” technique; yet this is a departure from the standard IP architecture and from the “end-to-end principle (i.e., stateless in the network)”
BGP (Border Gateway Protocol)

- RFC 1105 (June 1989): BGP-1
- RFC 1163 (June 1990): BGP-2
- RFC 1654 (July 1994), RFC 1771 (March 1995): BGP-4

- A path-vector protocol
- Built on top of TCP
  - makes BGP protocol much simpler than EGP
  - enables "incremental updates"

- Strengths
  - loops are easily prevented: use path-vector routing
  - does not require that all relays use the same metric
  - easy to incorporate policy routing (route ranking policy, export and import policies)
Traffic type & AS structure

- **Network traffic**
  - Local: originates at or terminates on nodes within an AS
  - Transit: passes through an AS

- **AS Types**
  - stub AS: has a single connection to one other AS
    - carries local traffic only
  - multihomed AS: has connections to more than one AS
    - refuses to carry transit traffic
  - transit AS: has connections to more than one AS
    - carries both transit and local traffic

- Each AS has one or more BGP *speakers* that advertise:
  - local networks
  - other reachable networks (transit AS only)
  - gives *path* information for advertised networks
BGP Example

- Speaker for AS2 advertises reachability to P and Q
  - network 128.96, 192.4.153, 192.4.32, and 192.4.3, can be reached directly from AS2

- Speaker for backbone advertises
  - networks 128.96, 192.4.153, 192.4.32, and 192.4.3 can be reached along the path (AS1, AS2)
    - Path information is used for loop detection: how?

- Speaker can cancel previously advertised paths
More on BGP

- Route announcement
  - *nlri*: network layer reachability info (addr. prefix)
  - *next_hop*: addr. of next hop router
  - *as_path*: ordered list of AS traversed
  - *med*: multi-exit discriminator
  - *local_pref*: local preference of a route

- Rank a route \( r \)

\[
\left\langle r.\text{local\_pref}, \frac{1}{|r.\text{as\_path}|}, \frac{1}{r.\text{med}}, \frac{1}{r.\text{next\_hop}} \right\rangle
\]
Policy routing: does it always converge?

- No

- It is NP-hard to check whether a set of BGP policy converge or not
Path-vector routing: does it converge quickly?

- Observation from the Internet
  - Average 3 minutes, with oscillations lasting up to 15 minutes
  - Upper bound $O(n!)$; lower bound $O(n)$

- Cause: exploration of invalid routes
An example of BGP slow convergence

- b withdraws its route to destination
- m withdraws its route; but the withdrawal by f is delayed
- g mistakenly regards route [f, b, a] as valid, and adopts it
- Route ranking at g:
  - [m, b, a] most preferred
  - [f, b, a] secondly preferred
  - [j, h, a] least preferred
References for BGP

1) Vern Paxson, “End-to-end routing behavior in the Internet”, SIGCOMM ’96

2) Kannan Varadhan, Deborah Estrin etc., “Persistent Route Oscillations in Inter-Domain Routing”, TR of USC ’96


7) On the correctness of IBGP configuration, SIGCOMM 2002

8) Route oscillations in I-BGP with route reflection, SIGCOMM 2002

9) Craig Labovitz etc., “Internet Routing Instability”, SIGCOMM ’97


11) Craig Labovitz etc., “Delayed Internet Routing Convergence”, SIGCOMM ’00


13) Hongwei Zhang, Anish Arora, Zhijun Liu, A Stability-oriented Approach to Improving BGP Convergence, SRDS 2004
IP Version 6 (IPv6)

- Work started in 1991

- Originally called “IP Next Generation (IPng)”, later assigned specific number as “version 6”; (original IP is called IPv4)
  - Number 5 has been used for other purposes 😊

- Features
  - 128-bit/16-byte addresses (classless): larger address space
    - Original motivation for designing IPv6
  - autoconfiguration
    - routers be in charge, thus do not need special DHCP server
  - multicast
  - real-time service
  - authentication and security
IPv6 (contd.)

- Header
  - 40-byte “base” header
  - *extension headers* (fixed order, mostly fixed length): flexible and accommodate unexpected future need
    - source routing
    - fragmentation
    - authentication and security
    - ...

- Incremental transition from IPv4 to IPv6 via
  - Dual stack for IPv6 nodes
  - Tunneling across IPv4 network to enable separated IPv6 nodes to talk to one another
Summary on “routing and scalability”

- **Routing**
  - Distance-vector routing
  - Link-state routing
  - Routing metrics
  - Mobile IP

- **Scalability (address space & routing)**
  - Subnetting, supernetting (CIDR)
  - Interdomain routing (BGP)
  - IPv6