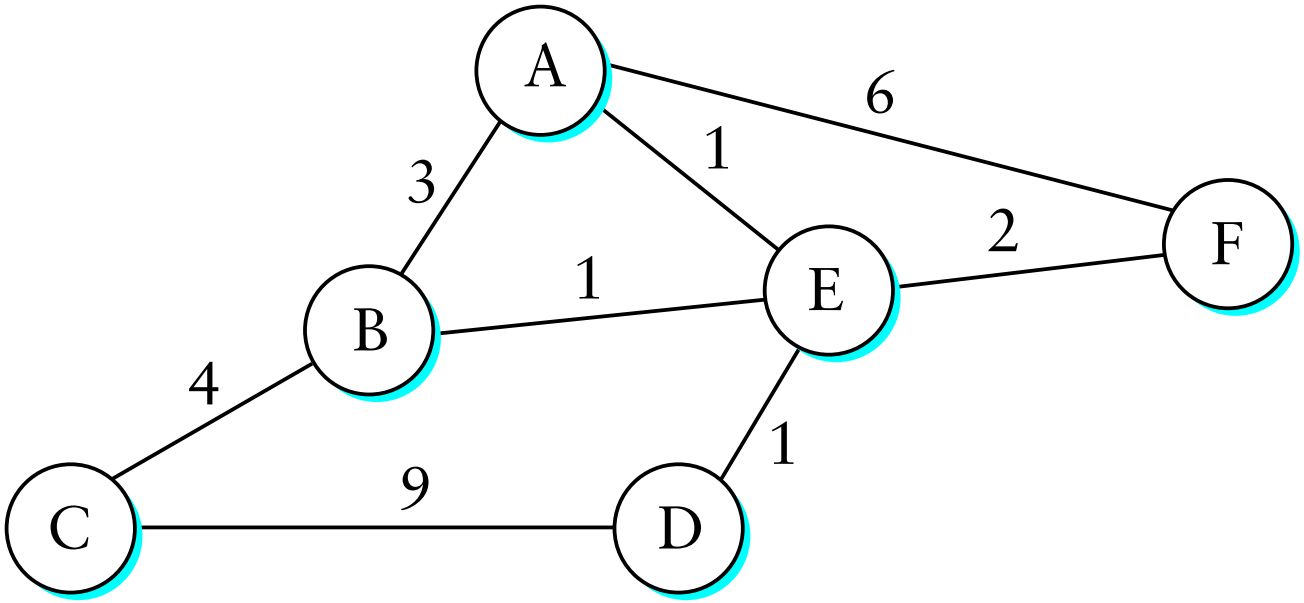
What is routing?

- **Packet *forwarding* – moving packets between ports**
  - Look up destination address in forwarding table
  - Find *out-port* or ⟨*out-port*, *MAC addr*⟩ pair
- ***Routing* is process of populating forwarding table**
  - Routers exchange messages about nets they can reach
  - Goal: Find optimal route for every destination, or maybe good route, or maybe just any route (depending on scale)
- ***Intra*-domain vs. *Inter*-domain routing**
  - Intra-: All routers under same administrative control
  - Intra-: Scale to ∼100 networks (e.g., campus like NYU)
  - Inter-: Decentralized, scale to Internet

Optimality

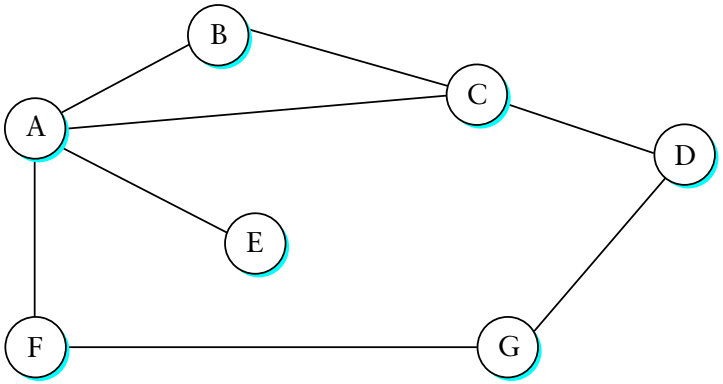


- **View network as a graph**
- **Assign *cost* to each edge**
  - Can be based on latency, b/w, utilization, queue length, . . .
- **Problem: Find lowest cost path between two nodes**
  - Must be computed in *distributed* way

Distance Vector

- **Each node maintains a set of triples**
  - $(Destination, Cost, NextHop)$
- **Exchange updates directly connected neighbors**
  - periodically (on the order of several seconds to minutes)
  - whenever table changes (called triggered update)
- **Each update is a list of pairs:**
  - $(Destination, Cost)$
- **Update local table if receive a "better" route**
  - smaller cost
  - came from next-hop
- **Refresh existing routes; delete if they time out**

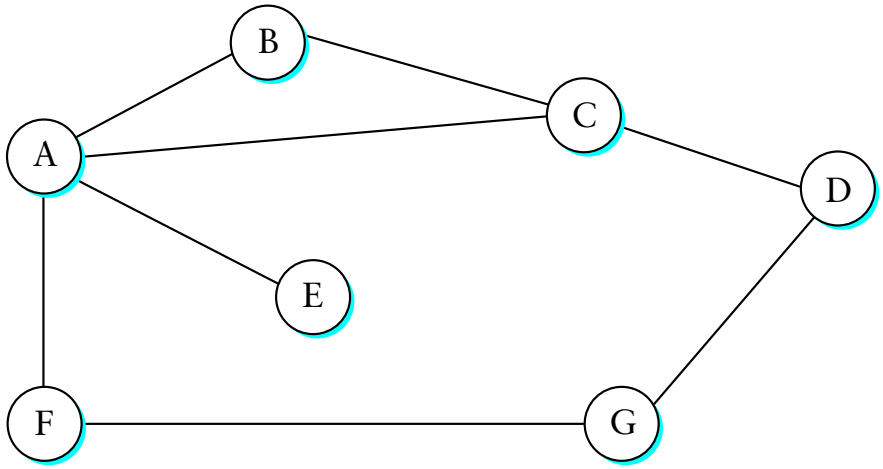
Example



Destination	Cost	NextHop
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	3	A

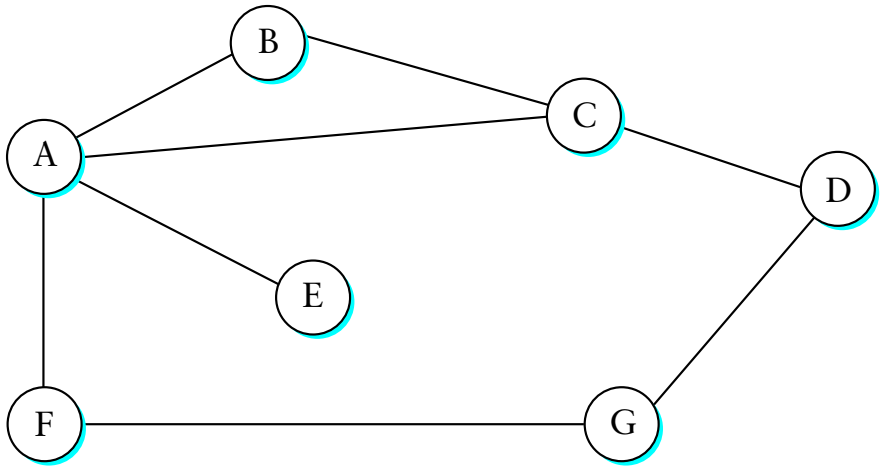
- **B's routing table**

Adapting to failures



- 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

Danger: Loops



- link from A to E fails
- A advertises distance of infinity to E
- B and C advertise a distance of 2 to E
- B decides it can reach E in 3 hops; advertises this to A
- A decides it can reach E in 4 hops; advertises this to C
- C decides that it can reach E in 5 hops…

How to avoid loops

- **Consider small value (e.g., 16) to be infinity**
  - Will quickly decite node is unavailable
- **Split horizon**
  - When sending updates to node A, don't include destinations you route to through A
- **Split horizon with poison reverse**
  - When sending updates to node A, explictly include very high cost ("poison") for destinations you route to through A
- **Note: Latter two only help between two nodes**
  - Can still get loop with three nodes involved
  - Might need to delay advertising routes after changes, but will affect convergence time

Link State

- **Strategy**
  - Send to all nodes (not just neighbors)
  - Send only information about directly connected links (not entire routing table)
- **Link State Packet (LSP)**
  - ID of the node that created the LSP
  - Cost of link to each directly connected neighbor
  - Sequence number (SEQNO)
  - Time-to-live (TTL) for this packet

Reliable flooding

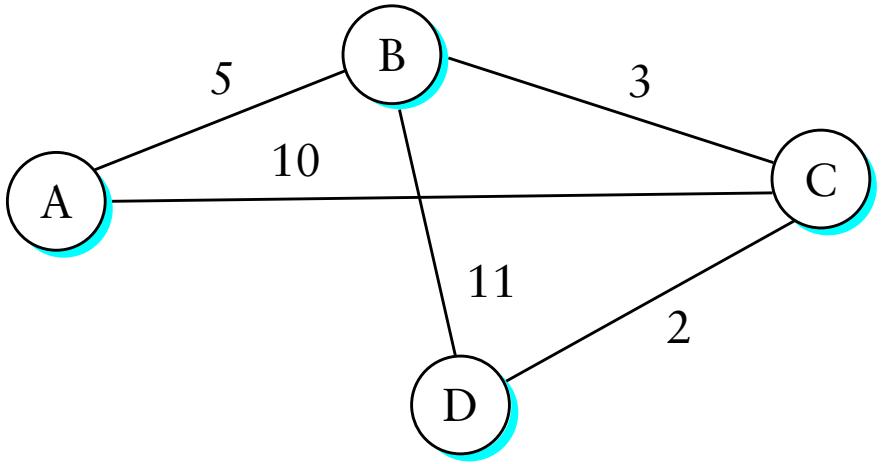
- **Store most recent LSP from each node**
- **Forward LSP to all nodes but one that sent it**
- **Generate new LSP periodically**
  - Increment SEQNO
- **Start SEQNO at 0 when reboot**
  - If you hear your own packet w. SEQNO$= n$, set your next SEQNO to $n + 1$
- **Decrement TTL of each stored LSP**
  - discard when TTL$= 0$

Calculating best path

- **Dijkstra's shortest path algorithm**
- **Let:**
  - N denote set of nodes in the graph
  - $l(i, j)$ denotes non-negative cost (weight) for edge (i, j)
  - s denotes yourself (node computing paths)
- **Initialize variables**
  - $M \leftarrow \{s\}$ (set of nodes "incorporated" so far)
  - $C_n \leftarrow l(s, n)$ (cost of the path from s to n)
  - $R_n \leftarrow \perp$ (next hop on path to n)

Dijkstra's algorithm

- **While** $N \neq M$
  - Let $w \in (N - M)$ be node with lowest C_w
  - $M \leftarrow M \cup \{w\}$
  - Foreach $n \in (N - M)$, if $C_w + l(w, n) < C_n$ then $C_n \leftarrow c_w + l(w, n)$, $R_n \leftarrow w$
- **Example: D** $(D, 0, \perp)(C, 2, C)(B, 5, C)(A, 10, C)$



Distance Vector vs. Link State

- **# of messages**
  - DV: $O(d)$ where d is # of neighbors of node
  - LS: $O(n \cdot d)$ for n nodes in system
- **Computation**
  - DV: Could count all the way to ∞ if loop
  - LS: $O(n^2)$
- **Robustness – what happens with malfunctioning router?**
  - DV: Node can advertise incorrect *path* cost
  - DV: Costs used by others, errors propagate through net
  - LS: Node can advertise incorrect *link* cost

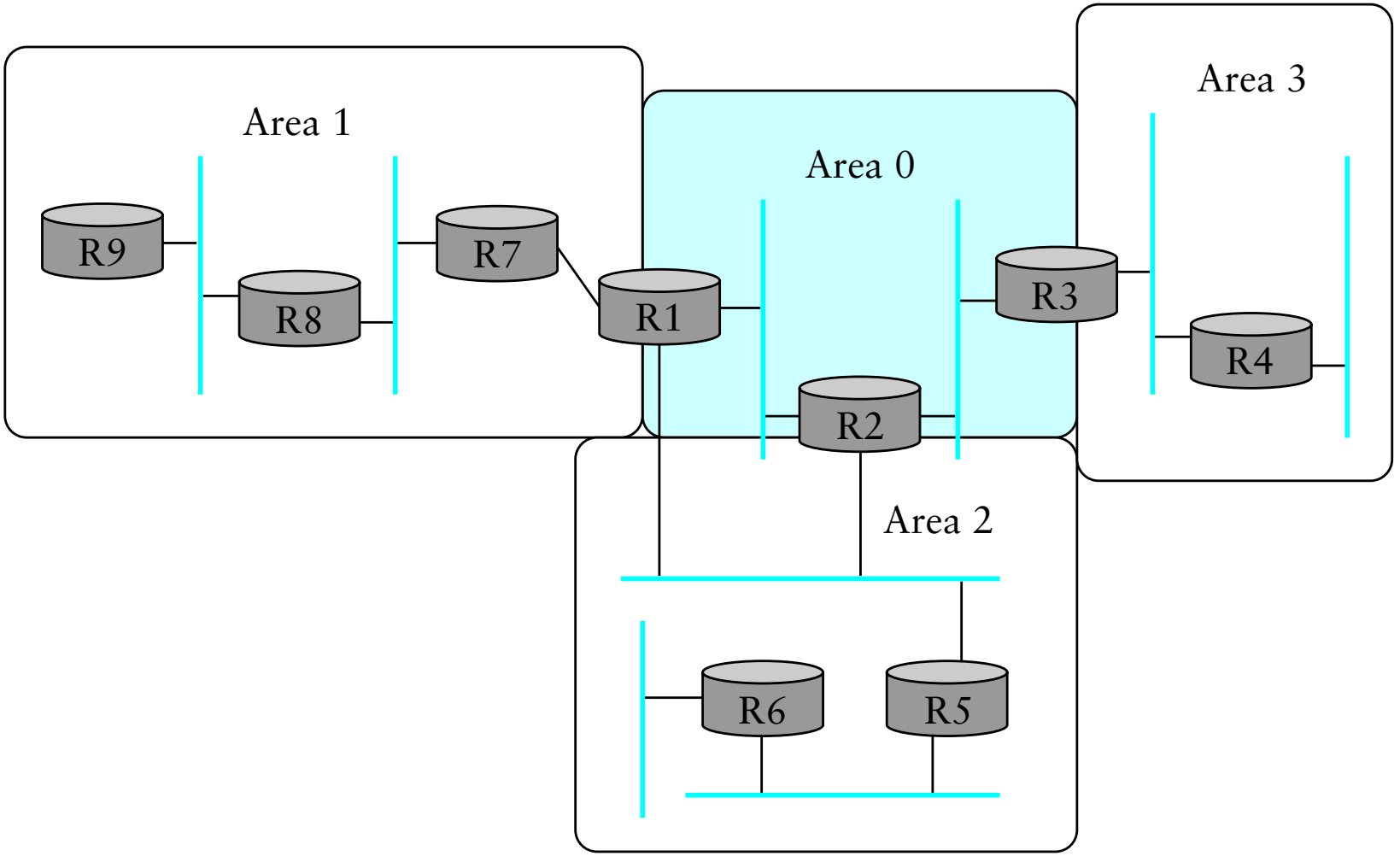
Metrics

- **Original ARPANET metric**
  - measures number of packets enqueued on each link
  - took neither latency nor bandwidth into consideration
- **New ARPANET metric**
  - stamp each incoming packet with its arrival time (AT)
  - record departure time (DT)
  - when link-level ACK arrives, compute
    $Delay = (DT - AT) + Transmit + Latency$
  - if timeout, reset DT to departure time for retransmission
  - link cost = average delay over some time period
- **Fine Tuning**
  - compressed dynamic range
  - replaced *Delay* with link utilization

Intradomain routing protocols

- **RIP (routing information protocol)**
  - Fairly simple implementation of DV
- **OSPF (open shortest path first)**
  - LS-based protocol
  - Adds notion of *areas* for scalability
  - Area 0 is special "backbone" area
  - Traffic between two areas must always go through area 0
  - Only need to know how to route exactly within area
  - Else, just route to appropriate area
  - (Virtual links can allow distant routers to be in area 0)

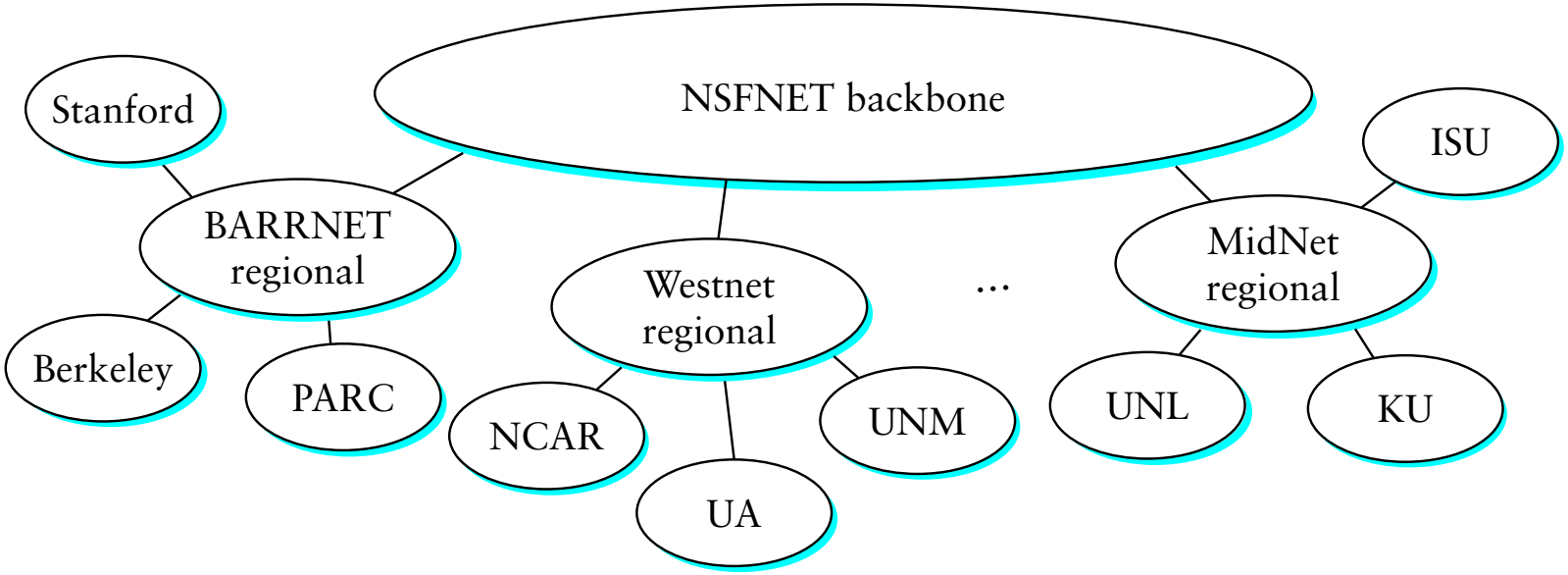
OSPF areas



Scaling issues

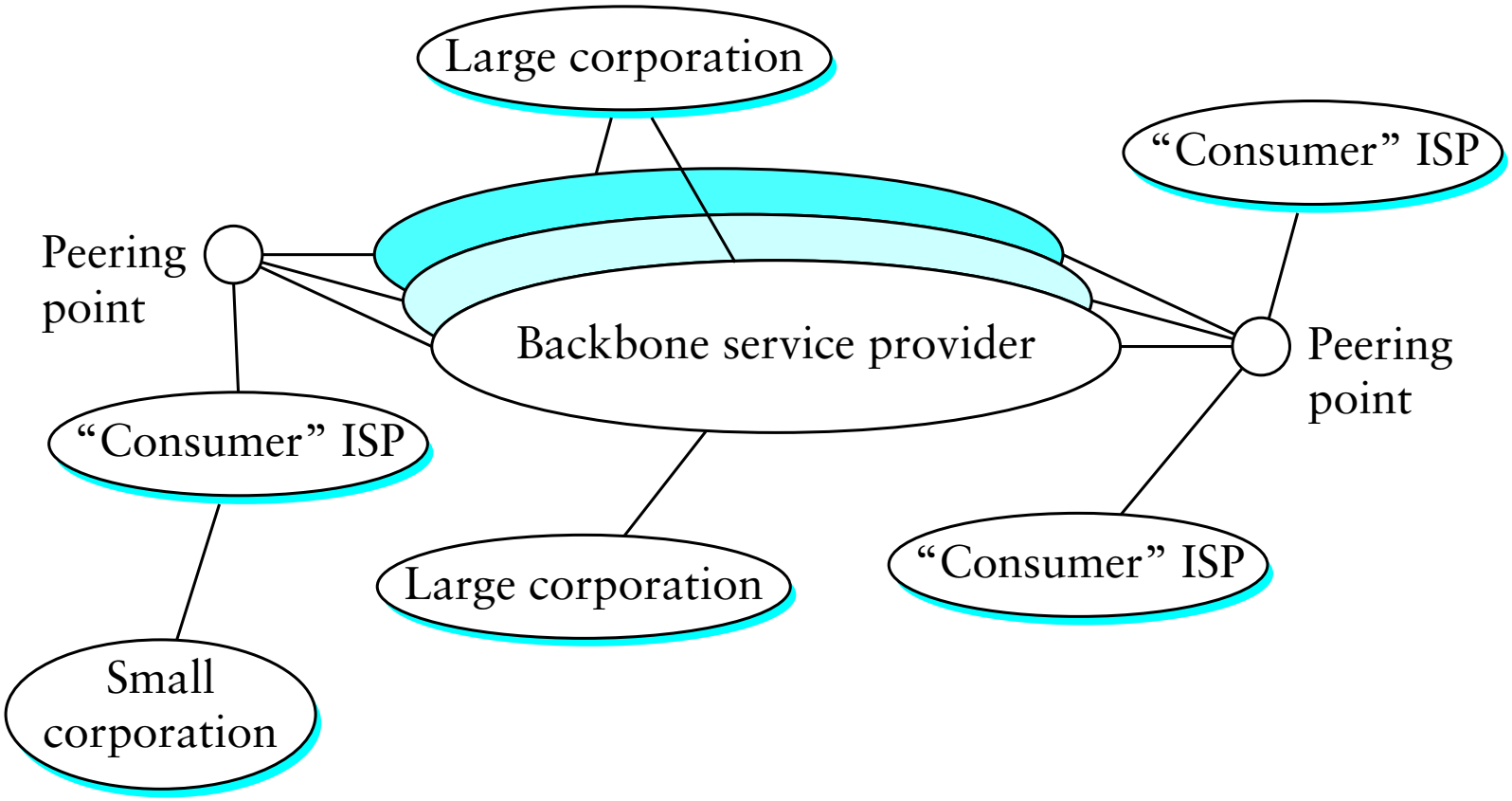
- **Every router must be able to forward based on *any* destination IP address**
  - Given address, it needs to know ”next hop” (table)
  - Naïve: Have an entry for each address
  - There would be 10^8 entries!
- **Solution: Entry covers range of addresses**
  - But can’t do this if addresses are assigned randomly! (e.g., Ethernet addresses)
  - Addresses allocation is a big deal
  - But can take advantage of network structure

The Internet, 1990



- **Hierarchical structure**

The Internet, today

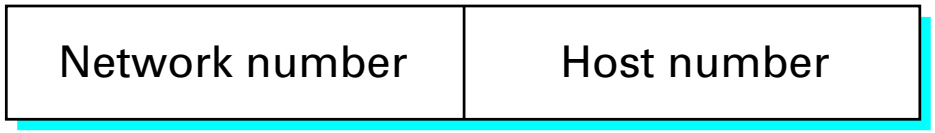


- **Multiple "backbones"**

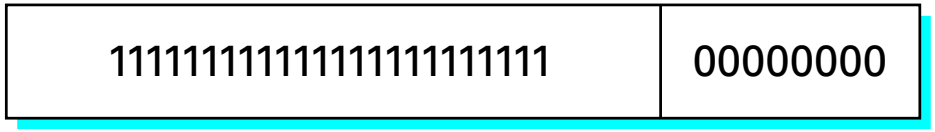
Exploit structure of network

- **Recall hierarchical nature of IP addresses**
  - Class A (8-bit prefix), B (16-bit), C (24-bit)
  - Routers need only know route for each network
- **But inefficient use of Hierarchical Address Space**
  - class C with 2 hosts (2/255 = 0.78% efficient)
  - class B with 256 hosts (256/65535 = 0.39% efficient)
  - Causes shortage of IP addresses (esp. class B)
  - Makes address authorities reluctant to give out class Bs
- **Still Too Many Networks**
  - routing tables do not scale
  - route propagation protocols do not scale

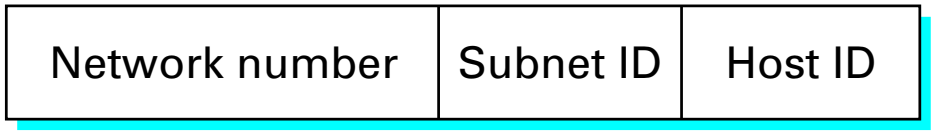
Subnetting



Class B address



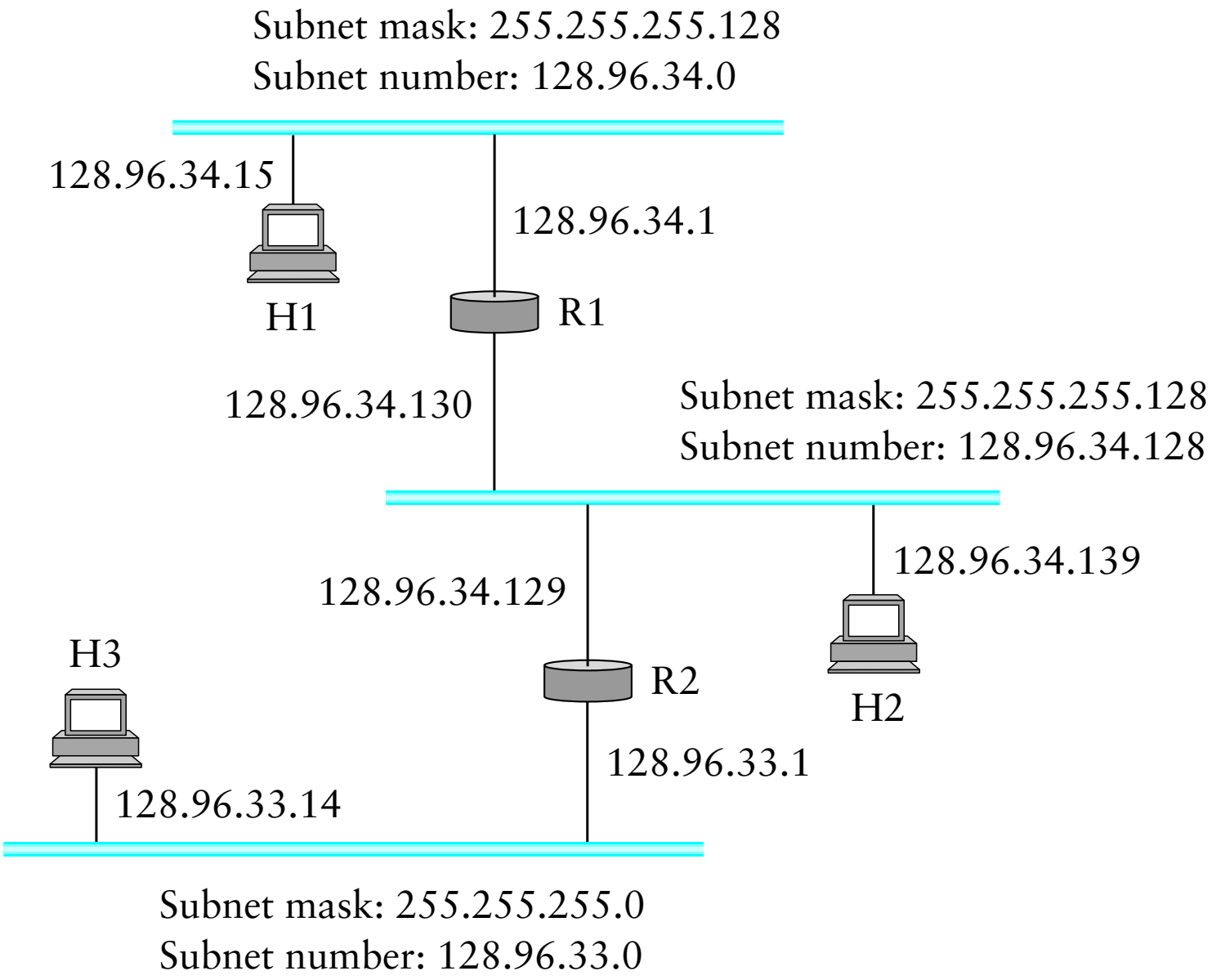
Subnet mask (255.255.255.0)



Subnetted address

- **Add another level to address/routing hierarchy**
- **Subnet masks define variable partition of host part**
- **Subnets visible only within site**

Example


Subnet mask: 255.255.255.128
Subnet number: 128.96.34.0
128.96.34.15
128.96.34.1
H1
R1
128.96.34.130
Subnet mask: 255.255.255.128
Subnet number: 128.96.34.128
128.96.34.129
128.96.34.139
H3
R2
H2
128.96.33.1
128.96.33.14
Subnet mask: 255.255.255.0
Subnet number: 128.96.33.0

Supernetting

- **Assign block of contiguous network numbers to nearby networks**
- **Called CIDR: Classless Inter-Domain Routing**
- **Represent blocks with a single pair**
  $(first\ network\ address, count)$
- **Restrict block sizes to powers of 2**
- **Use a bit mask (CIDR mask) to identify block size**
- **All routers must understand CIDR addressing**

Route Propagation

- **Know a smarter router**
  - hosts know local router
  - local routers know site routers
  - site routers know core router
  - core routers know everything
- **Introduce notion of *Autonomous System* (AS)**
- **Two-level route propagation hierarchy**
  - interior gateway protocol (each AS selects its own)
  - exterior gateway protocol (Internet-wide standard)

Autonomous systems

- **Corresponds to an administrative domain**
  - Internet is not a single network
  - ASes reflect organization of the Internet
  - E.g., NYU, large company, etc.
- **Goals:**
  - ASes want to choose their own local routing algorithm
  - ASes want to set policies about non-local routing
- **Each AS assigned unique 16-bit number**

Types of AS

- ***Local traffic* – packets with src or dst in local AS**
- ***Transit traffic* – passes through an AS**
- ***Stub* AS**
  - Connects to only a single other AS
- ***Multihomed* AS**
  - Connects to multiple ASes
  - Carries no transit traffic
- ***Transit* AS**
  - Connects to multiple ASes and carries transit traffic

EGP: Exterior Gateway Protocol

- **Overview**
  - designed for tree-structured Internet
  - concerned with reachability, not optimal routes
- **Protocol messages**
  - neighbor acquisition: one router requests that another be its peer; peers exchange reachability information
  - neighbor reachability: one router periodically tests if another is still reachable; exchange HELLO/ACK messages; uses a k-out-of-n rule
  - routing updates: peers periodically exchange their routing tables (distance-vector)

Today: BGP-4

- **Goal: Share connectivity information across ASes**
  - Don't strive for "optimal" routes—too hard
  - Different ASes may have different notions of cost
  - May have policies that dictate suboptimal routes
- **BGP used by two types of routers:**
  - *edge* routers, connecting organization to world
  - *core* routers, making up backbone
- **Within ASes, can use any routing protocol**
  - But backbones too big for RIP or OSPF
  - So *internal* BGP (IBGP) variant for use within ASes

Choice of Routing Algorithm

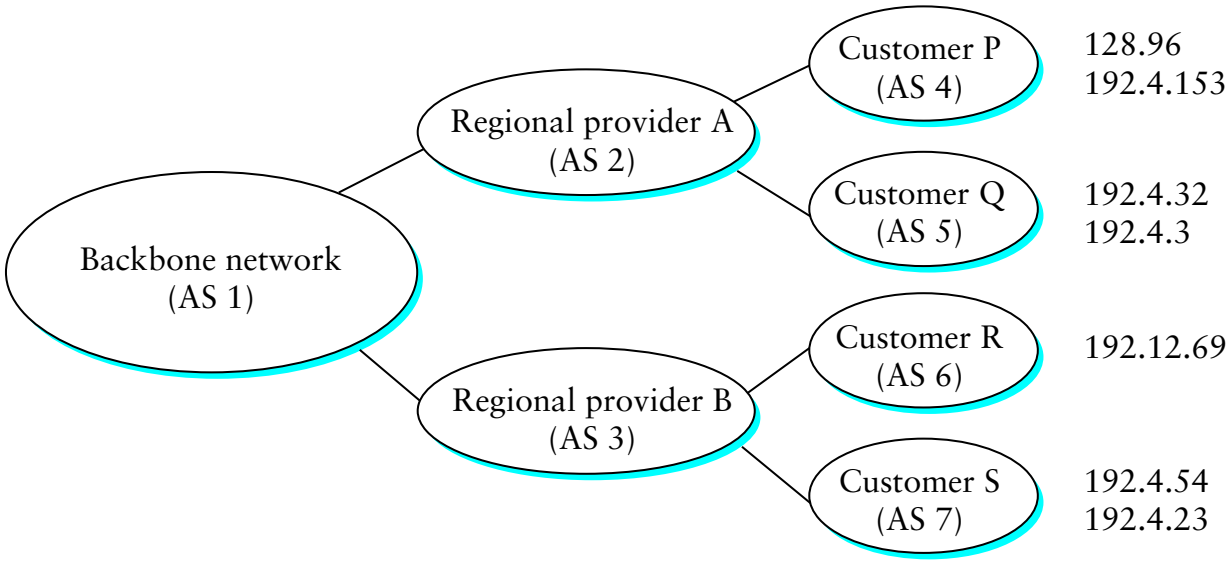
- **Constraints:**
  - Scaling
  - Autonomy (policy and privacy)
- **Link-state?**
  - Requires sharing of complete network informatin
  - Information exchanges don't scale
  - Can't express policy
- **Distance Vector?**
  - Scales and retains privacy
  - Can't implement policy
  - Can't avoid loops if shortest paths not taken

Path Vector Protocol

- **Distance vector algorithm with extra information**
  - For each route, store the complete path (ASs)
  - No extra computation, just extra storage
- **Advantages:**
  - Can make policy choices based on set of ASs in path
  - Can easily avoid loops
- **In addition, separate *speaker* & *gateway* roles**
  - *speaker* talks BGP protocol to other ASes
  - *gateways* are routers that border other ASes
  - Can have more gateways than speakers
  - Speaker can reach gateways over local network

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).
- **Speaker can cancel previously advertised paths**

Basic BGP Messages

- **Open:**
  - Establishes BGP session (uses TCP port #179)
- **Notification:**
  - Report unusual conditions (message header error, . . .)
- **Update:**
  - Inform neighbor of new routes that become active
  - Inform neighbor of old routes that become inactive
- **Keepalive:**
  - Inform neighbor that connection is still viable

Attributes of BGP routes

- **AS path**
- **Origin**
  - Who originated the announcement?
  - IGP, EGP, or "incomplete" (for static routes)
- **Multi-Exit Discriminator (MED)**
  - How close prefix is to link it is announced on
  - Used if ASes A & B connect at multiple points
- **Local preference**
  - Used in IBGP to select (or give preference to) a particular exit for a particular prefix

IPv6

- **Features**
  - 128-bit addresses (classless), includes multicast addresses
  - real-time service
  - authentication and security
  - autoconfiguration
  - end-to-end fragmentation
  - protocol extensions
- **40-byte "base" header, points to next extension hdr**
  - fragmentation
  - source routing
  - authentication and security
  - other options