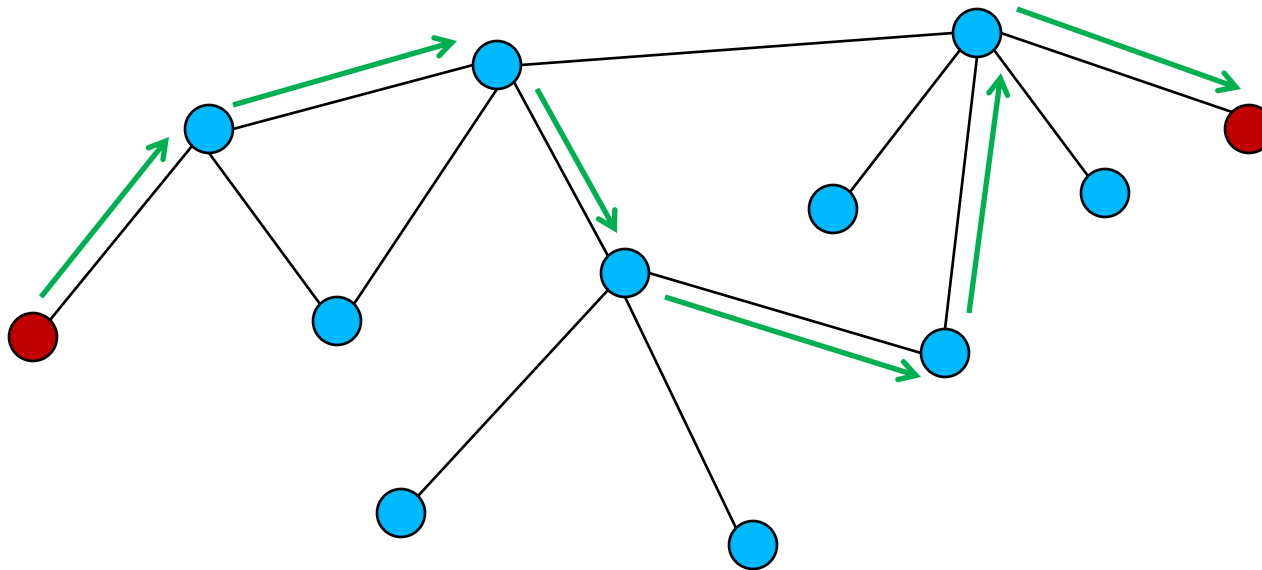


IP Routing



Networks as Graphs

- Networks are graphs
 - Edges = communication links
 - Vertices = computers, switches, routers, etc

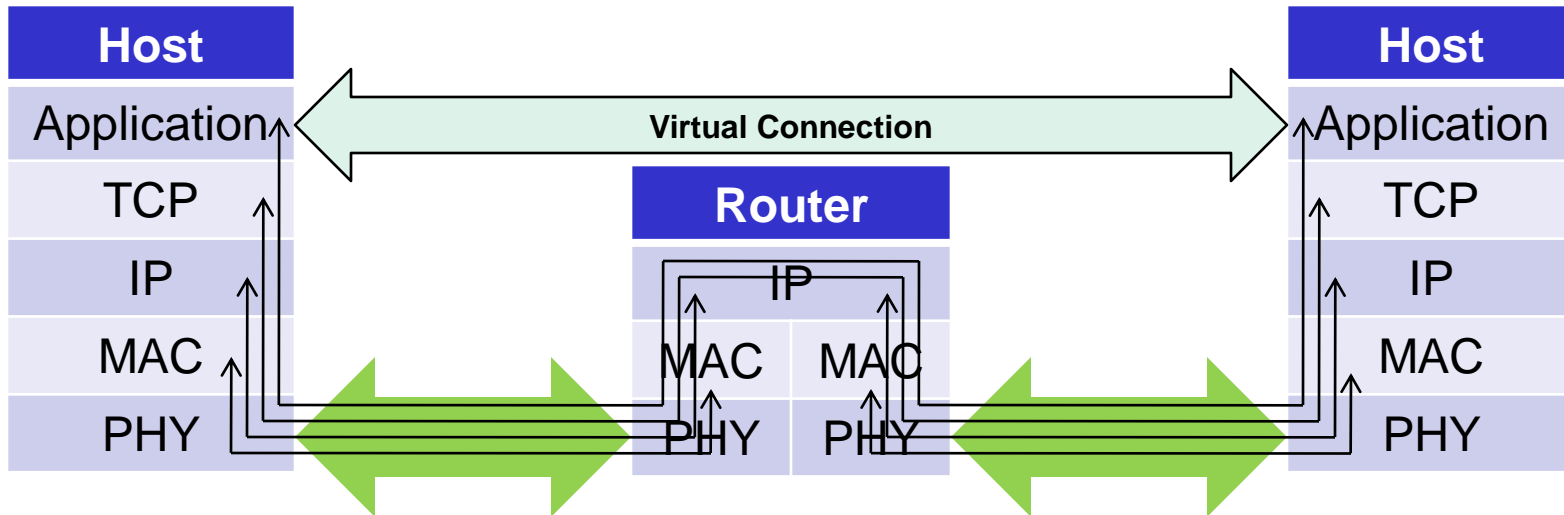


- For packet inbound at a particular vertex, determine what output edge to use



Routers

- Processes IP header on inbound packets
- Uses destination IP address to determine through which interface the packet should be routed



Approaches

- Discussed some approaches for packet switching
 - Virtual Circuits
 - Switching Tables
 - Source Routing
- Not scalable on the Internet
 - Imagine running STP across entire Internet
- Each network independent *administrative domain*, local policy controls local configuration
- Need way to route between *networks*, not between *computers*



Routing Tables

- Each host (both computers and routers) maintain a list of “next hops” in addition to a default route

- Residential Host:

Network	Interface	Gateway
192.168.0.123/32	lo0	-
192.168.0.0/24	eth0	-
0.0.0.0/0	eth0	192.168.0.1

- Residential Router:

Network	Interface	Gateway
192.168.0.0/24	eth0	-
12.34.56.0/23	wan0	-
0.0.0.0/0	wan0	12.34.56.1

- Core Internet routers have very large routing tables for every top-level network



Building Routing Tables

- Statically configured
 - Works well for end-hosts, configured via DHCP
 - Bad for core routers with complex routing tables
- Need algorithms for automatically populating routing tables



Distance Vector Routing

- Based on Bellman-Ford Algorithm
 - Computes single-source shortest path on a weighted digraph
 - <http://links.math.rpi.edu/applets/appindex/graphtheory.html>
- Run Bellman-Ford for all nodes
- Each node initializes a vector V of all routers
 - Entries equal weight to neighbors and infinity to others, initially
- During each iteration
 - Send $V+W$ to neighbor, where W =weight to neighbor
 - Neighbor sets own vector to $\min(\text{received } V, \text{own } V)$
 - Track neighbor that provides minimum distance in second vector R
- Reinitiate whenever there is a topology or vector change



Distance Vector Routing

- Link Failure
 - Send out distance of infinity, nodes update
- Count to infinity problem
 - When link goes down, update loops can happen
 - Distances slowly count to infinity
- Solutions
 - Make “infinity” small
 - Split horizon: don't send information learned from neighbor back to neighbor



Distance Vector Routing

- Distance Vector basis of Routing Information Protocol (RIP)
- Scales as $O(V \cdot E)$ for a network of V vertices and E edges



Link State Routing

- Requires reliable flooding
 - Routers must be able to send broadcast messages out to all other routers
- Routers broadcast “link state packets” (LSPs) that contain
 - Name of the source node
 - Sequence number
 - Names of neighbor nodes and link cost to each
 - TTL for packet
- Each node uses all LSPs to compute shortest path information



Link State Routing

- Each node has global knowledge of the network
- Each node can run Dijkstra's algorithm to determine routing information



Dijkstra's Algorithm

- N = set of nodes, $L(i, j)$ = weights, s = source, $C(n)$ = graph distance from s to n
- $M = \{s\}$
- For each n in $N-M$
 - $C(n) = L(s, n)$
- While ($N \neq M$)
 - $M = M \cup \{w\}$ such that $C(w)$ is the minimum for all w in $(N-M)$
 - For each n in $(N-M)$
 - $C(n) = \text{MIN}(C(n), C(w) + L(w, n))$



Link State Routing

- Better link-failure propagation properties
- Stabilizes quickly for changes in the network
- Requires each node to have more memory, processing
- Basis of Open Shortest Path First Protocol (OSPF)



Edge Weights

- How do you determine link weights?
- All links weight 1
 - Does not consider capacity of links
 - Does not consider load on links
- ARPANET
 - Original metric = number of queued packets
 - Problematic, did not consider latency or capacity
 - Revised metric based on delay of packets through router
 - Delay = (depart-arrive) + transmission + latency
 - Worked well for light load, but oscillation in high load
 - Improved metric smoothed revised metric
 - Time-average, bounded velocity of change

