Network Layer:
Overview;
Distance Vector Protocols

Qiao Xiang, Congming Gao, Qiang Su

https://sngroup.org.cn/courses/cnns-xmuf25/index.shtml

11/18/2025

This deck of slides are heavily based on CPSC 433/533 at Yale University, by courtesy of Dr. Y. Richard Yang.

Outline

- Admin and recap
- Network overview
- Network control-plane
  - Routing

Recap: BW Allocation Framework

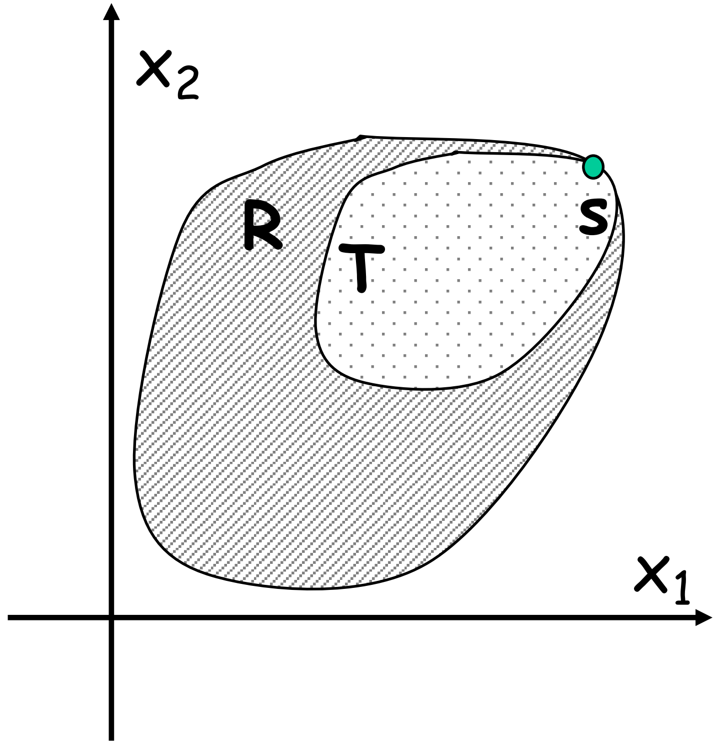
$$\begin{array}{ll} \max & \sum_{f \in F} U_f(x_f) \\ \text{subject to} & \sum_{f: f \text{ uses link } l} x_f \le c_l \text{ for any link } l \\ \text{over} & x \ge 0 \end{array}$$

- Forward engineering: systematically design
  - objective function
  - distributed alg to achieve objective
- Science/reverse engineering: what do TCP/Reno, TCP/Vegas achieve?

Objective	Allocation (x1, x2, x3)		
TCP/Reno	0.26	0.74	0.74
TCP/Vegas	1/3	2/3	2/3
Max throughput	0	1	1
Max-min	½	½	½
Max sum log(x)	1/3	2/3	2/3
Max sum of $-1/(RTT^2 x)$	0.26	0.74	0.74

Recap: Derive Objective Function

- NBS axioms
  - Pareto optimality
  - symmetry
  - invariance of linear transformation
  - independence of irrelevant alternatives



- NBS solution
  - the rate allocation point is the feasible point which maximizes

$$x_1 x_2 \cdots x_F$$

Recap: Primal-Dual Decomposition of Network-Wide Resource Allocation

- SYSTEM(U):

$$\begin{array}{ll} \max & \sum_{f \in F} U_f\left(x_f\right) \\ \text{subject to} & \sum_{f: f \text{ uses link } l} x_f \le c_l \text{ for any link } l \\ \text{over} & x \ge 0 \end{array}$$

- USER$_f$:

$$\begin{array}{ll} \max_{x_f} & U_f\left(x_f\right) - x_f p_f \\ \text{over} & x_f \ge 0 \end{array}$$

- NETWORK:

$$\min_{q \ge 0} \widetilde{D}(q) = \sum_l q_l \Big(c_l - \sum_{f: \text{f uses } l} x_f\Big)$$

TCP/Reno Dynamics

$$\boxed{\Delta x_f \propto U'_f\left(x_f\right) - p_f}$$

$$\Delta x = \frac{RTT}{2} x^2 \boxed{\left(\frac{2}{x^2 RTT^2} - p\right)}$$

$$U'_f\left(x_f\right) - p_f$$

$$\Rightarrow U'_f(x_f) = \left(\frac{\sqrt{2}}{x_f RTT}\right)^2 \quad \Rightarrow U_f(x_f) = -\frac{2}{RTT^2 x_f}$$

TCP/Vegas Dynamics

$$\boxed{\Delta x_f \propto U'_f(x_f) - p_f}$$

$$\Delta x = \frac{x}{RTT}\left(\boxed{\frac{\alpha}{x} - (RTT - \mathrm{RTTmin})}\right)$$

$$U'_f(x_f) - p_f$$

$$\Rightarrow U'_f(x_f) = \frac{\alpha}{x} \qquad \Rightarrow U_f(x_f) = \alpha \log(x_f)$$

Outline

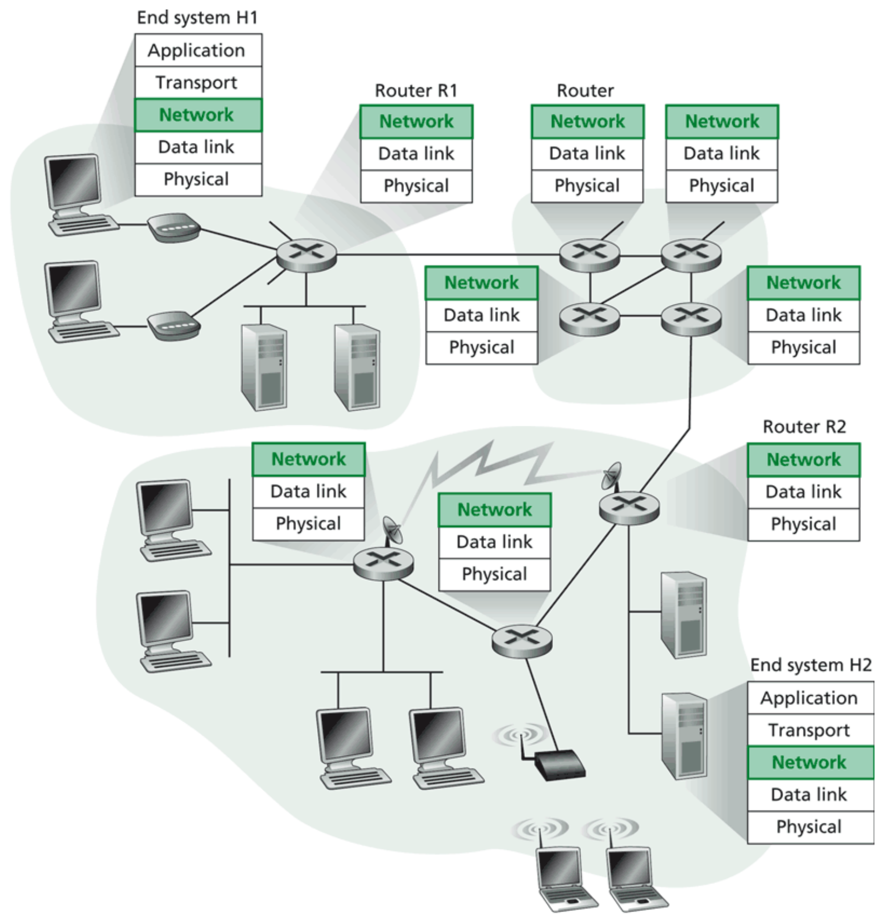
- Admin and recap
- *Network overview*

Network Layer

- Transport packet from source to dest.
- Network layer in *every* host, router

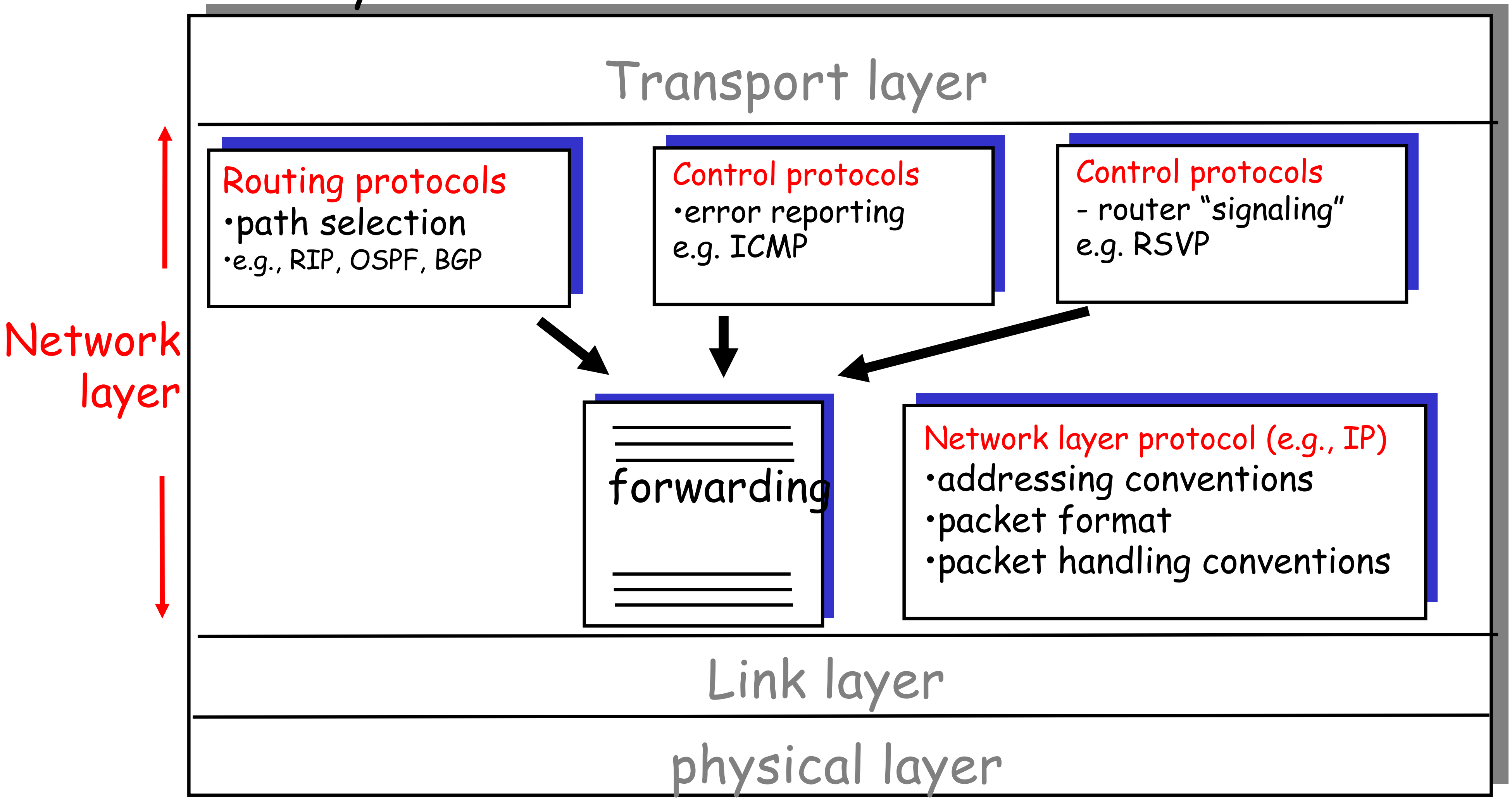
Basic functions:

- inter-networking (e.g., fragmentation/assembly)
- routing (determine route(s) taken by packets of a flow), and forwarding (move the packets along the route(s))

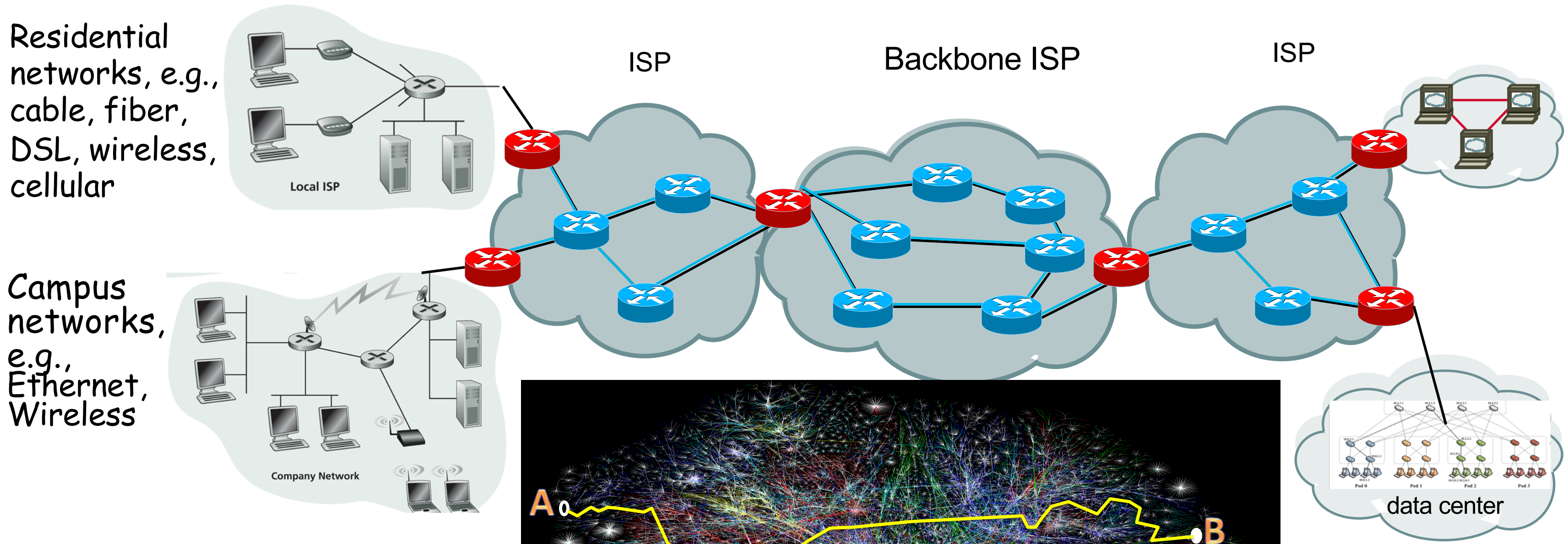


Current Internet Network Layer

Network layer functions:



Our Focus: Global Routing System



Discussion: What are the challenges/requirements of designing the global routing system?

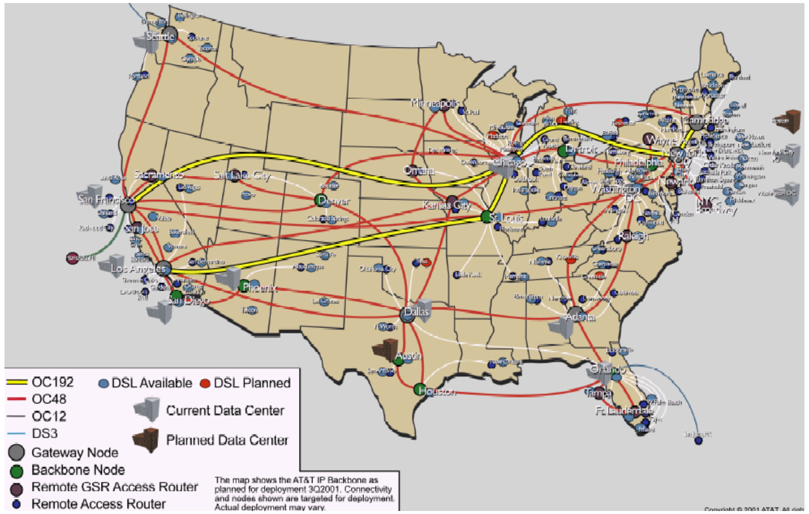
Global Routing Divide and Conquer: Routing with Autonomous Systems

- Global Internet routing is divided into intra-AS routing and inter-AS routing
  - Intra-AS routing (also called intradomain routing)
    - A protocol running insides an AS is called an Interior Gateway Protocol (IGP), each AS can choose its own protocol, such as RIP, E/IGRP, OSPF, IS-IS
  - Inter-AS routing (also called interdomain routing)
    - A protocol runs among autonomous systems is also called an Exterior Gateway Protocol (EGP)
    - The de facto EGP protocol is BGP

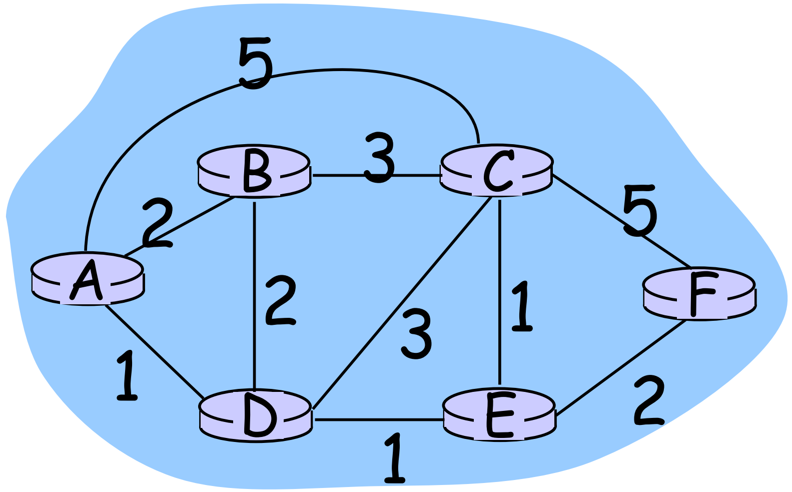
Routing: Overview

Routing

Goal: determine "good" paths (sequences of routers) thru networks from source to dest.



Graph abstraction for the routing problem:



- graph nodes are routers
- graph edges are physical links
  - links have properties: delay, capacity, $ cost, policy

Network Layer: Complexity Factors/Objectives

- For network providers
  - efficiency of routes
  - policy control on routes
  - scalability

- For users: quality of services
  - guaranteed bandwidth?
  - preservation of inter-packet timing (no jitter)?
  - loss-free delivery?
  - in-order delivery?

- Users and network may interact

Routing Design Space

- Robustness
- Optimality
- Simplicity

- Routing has a large design space
  - who decides routing?
    - source routing: end hosts make decision
    - network routing: networks make decision
  - how many paths from source s to destination d?
    - multi-path routing
    - single path routing
  - what does routing compute?
    - network cost minimization
    - QoS aware
  - will routing adapt to network traffic demand?
    - adaptive routing
    - static routing
  - ...

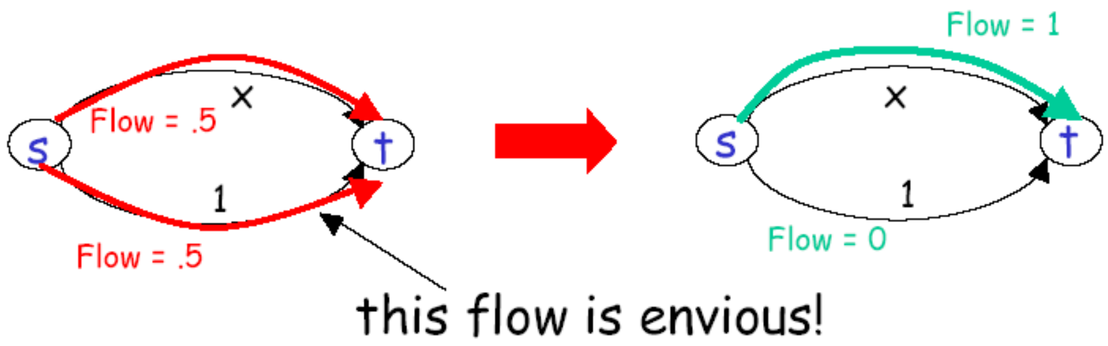
Routing Design Space: User-based, Multipath, Adaptive

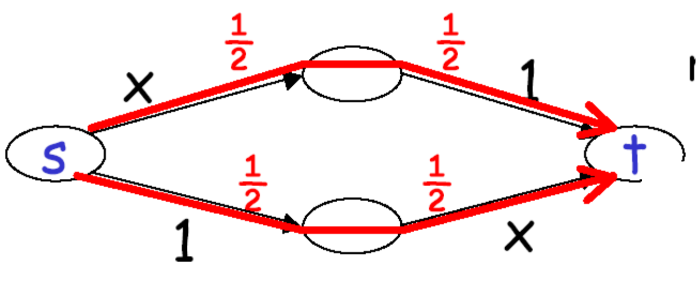
- Robustness
- Optimality
- Simplicity

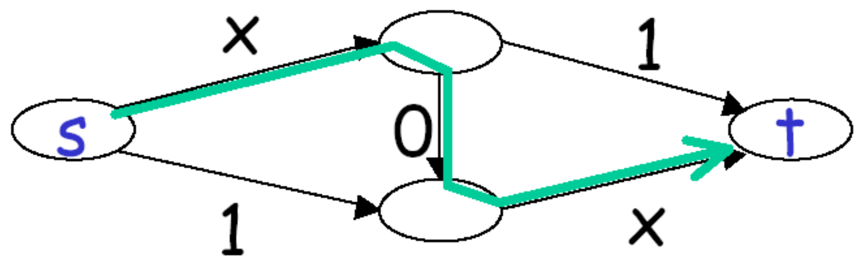
- Routing has a large design space
  - who decides routing?
    - source routing: end hosts make decision
    - network routing: networks make decision
  - how many paths from source s to destination d?
    - multi-path routing
    - single path routing
  - what does routing compute?
    - network cost minimization
    - QoS aware
  - will routing adapt to network traffic demand?
    - adaptive routing
    - static routing
  - ...

User Optimal, Multipath, Adaptive

- User optimal: users pick the shortest routes (selfish routing)







Braess's paradox

Price of Anarchy

For a network with linear latency functions

→

total latency of user (selfish) routing for given traffic demand

≤ 4/3

total latency of network optimal routing for the traffic demand

Price of Anarchy

- For any network with continuous, non-decreasing latency functions →

  total latency of user (selfish) routing for given traffic demand

  ≤

  total latency of network optimal routing for twice traffic demand

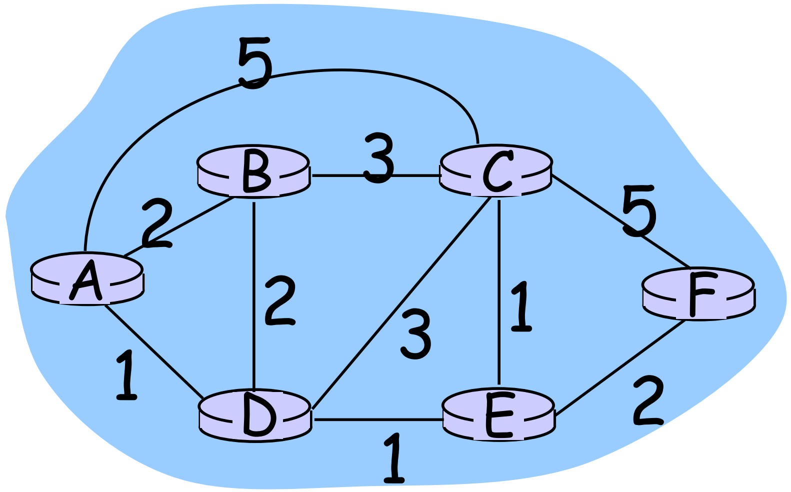
Routing Design Space: Internet

- Robustness
- Optimality
- Simplicity

- Routing has a large design space
  - who decides routing?
    - source routing: end hosts make decision
    - network routing: networks make decision
      - (applications such as overlay and p2p are trying to bypass it)
  - what does routing compute?
    - network cost minimization (shortest path)
    - QoS aware
  - how many paths from source s to destination d?
    - multi-path routing
    - single path routing (with small amount of multipath)
  - will routing adapt to network traffic demand?
    - adaptive routing
    - static routing (mostly static; adjust in larger timescale)
  - ...

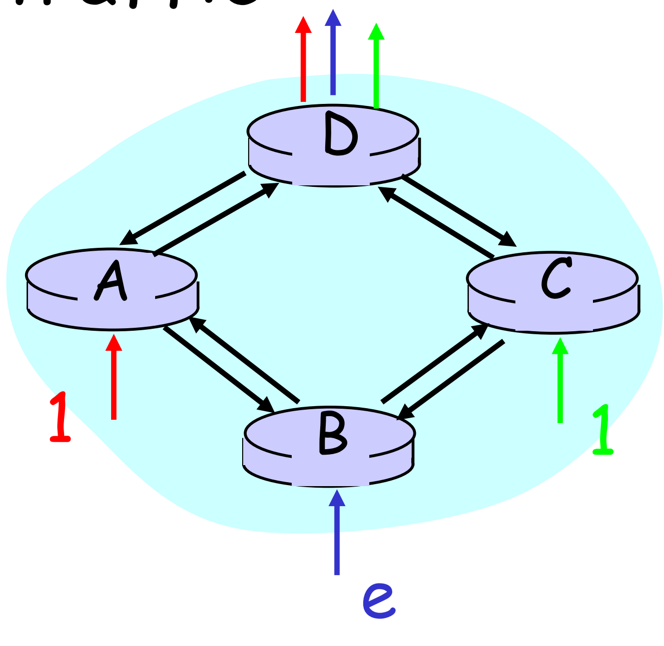
Basic Formulation

- Assign link weights
- Compute shortest path

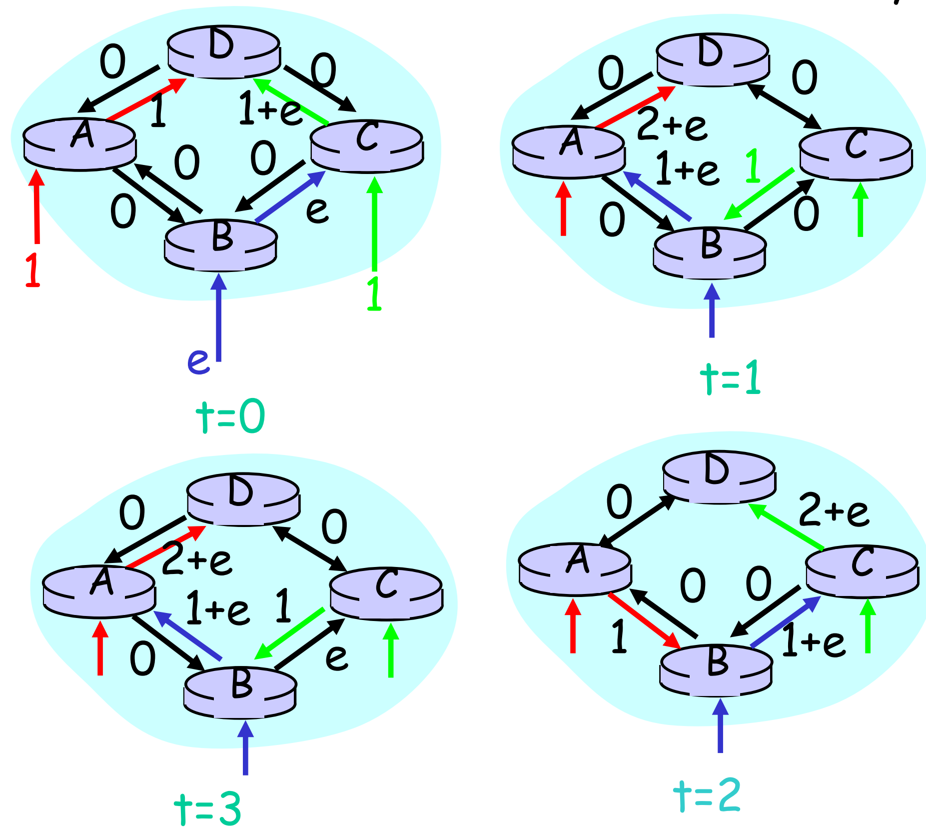


Assigning Link Weight: Dynamic Link Costs

- Assign link costs to reflect current traffic



Link costs reflect current traffic intensity



Solution: Link costs are a combination of current traffic intensity (dynamic) and topology (static). To improve stability, the static topology part should be large. Thus less sensitive to traffic; thus non-adaptive.

Example: Cisco Proprietary Recommendation on Link Cost

- Link metric:
  - metric = [K1 * bandwidth^{-1} + (K2 * bandwidth^{-1}) / (256 - load) + K3 * delay] * [K5 / (reliability + K4)] * 256

By default, k1=k3=1 and k2=k4=k5=0. The default composite metric for EIGRP, adjusted for scaling factors, is as follows:

$$EIGRP_{metric} = 256 \times \{ [10^7/BW_{min}] + [sum\_of\_delays] \}$$

BW_{min} is in kbps and the sum of delays are in 10s of microseconds.

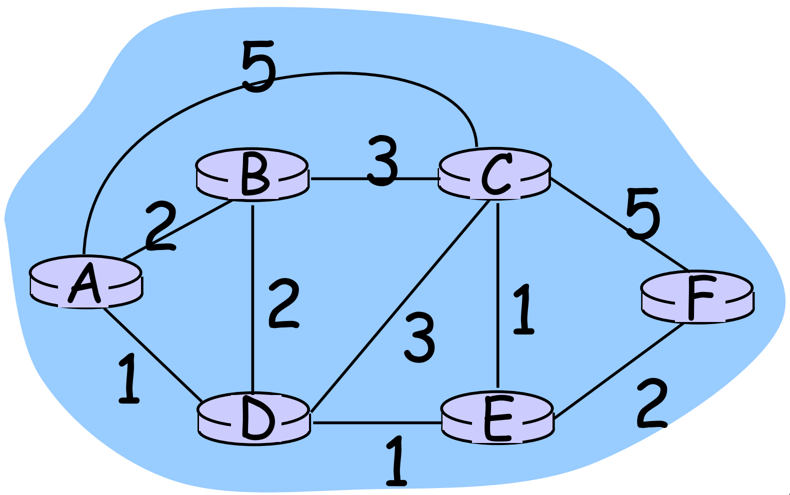
EIGRP : Enhanced Interior Gateway Routing Protocol

Example: EIGRP Link Cost

- The bandwidth and delay for an Ethernet interface are 10 Mbps and 1 ms, respectively.
- The calculated EIGRP BW metric is as follows:
  - $256 \times 10^7/BW = 256 \times 10^7/10{,}000$

    $= 256 \times 1000$

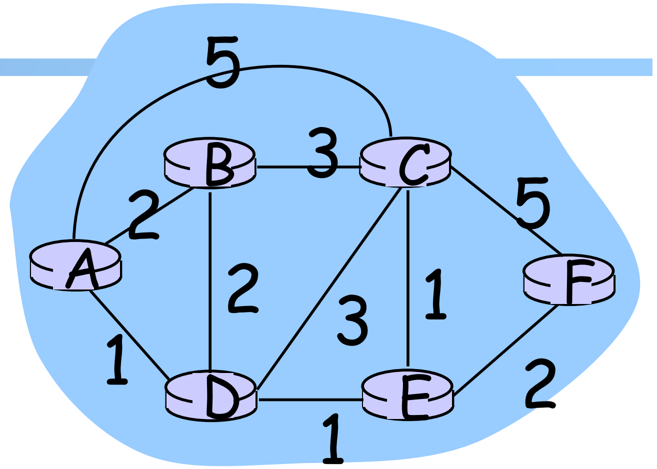
    $= 256{,}000$



Outline

- Admin and recap
- Network overview
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - *Distributed distance vector protocols*

Distance Vector Routing



- Setting: static (positive) costs assigned to network links
  - The static link costs may be adjusted in a longer time scale: this is called traffic engineering
- Goal: distributed computing to compute the shortest path from a source to a destination
  - Based on the Bellman-Ford algorithm (BFA)
  - Conceptually, runs for each destination separately
- Look ahead
  - Although few (e.g., RIP) use basic distance vector, it is a foundation for many other protocols
  - We also use the study to acquire another basic set of techniques to understand distributed protocols

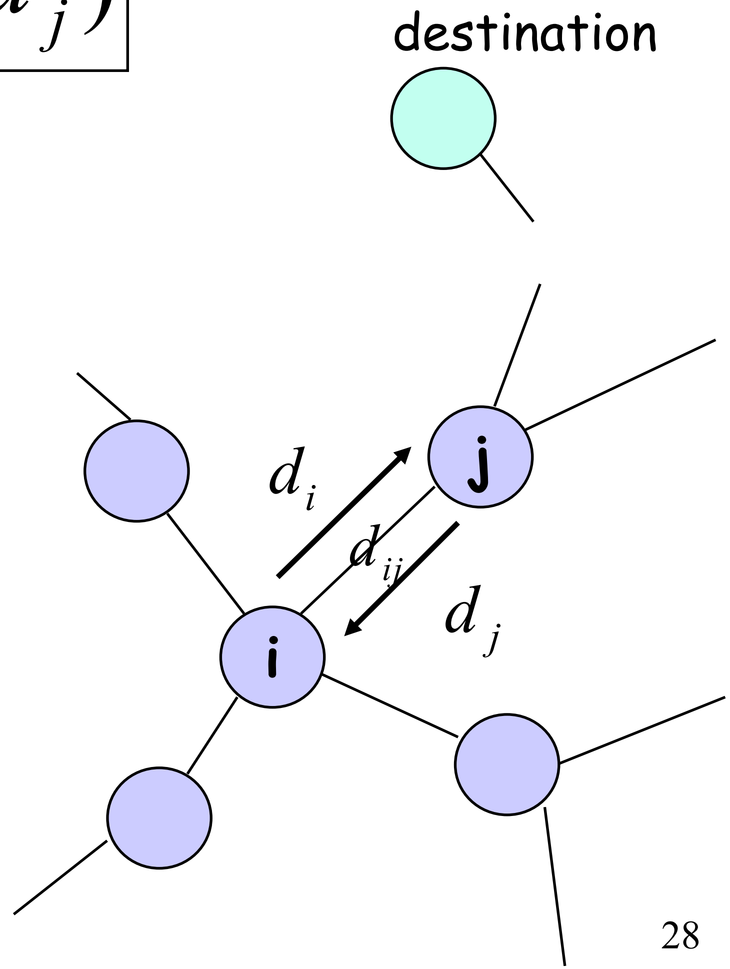
Distance Vector Routing: Basic Idea

- At node i, the basic update rule

$$\boxed{d_i = \min_{j \in N(i)} (d_{ij} + d_j)}$$

where

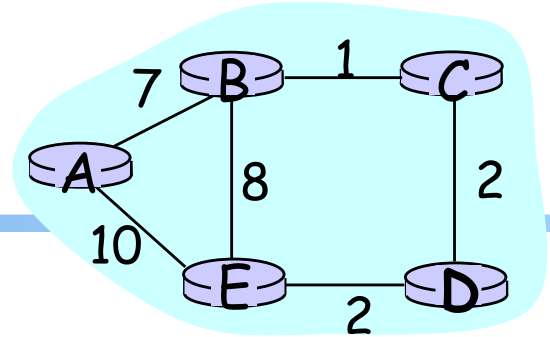
- d_i denotes the distance estimation from i to the destination,
- N(i) is set of neighbors of node i, and
- d_{ij} is the distance of the direct link from i to j



Outline

- Admin and recap
- Network overview
- Network control plane
    - Routing
        - Link weights assignment
        - Routing computation
            - *distributed distance vector protocols*
                - *synchronous Bellman-Ford (SBF)*

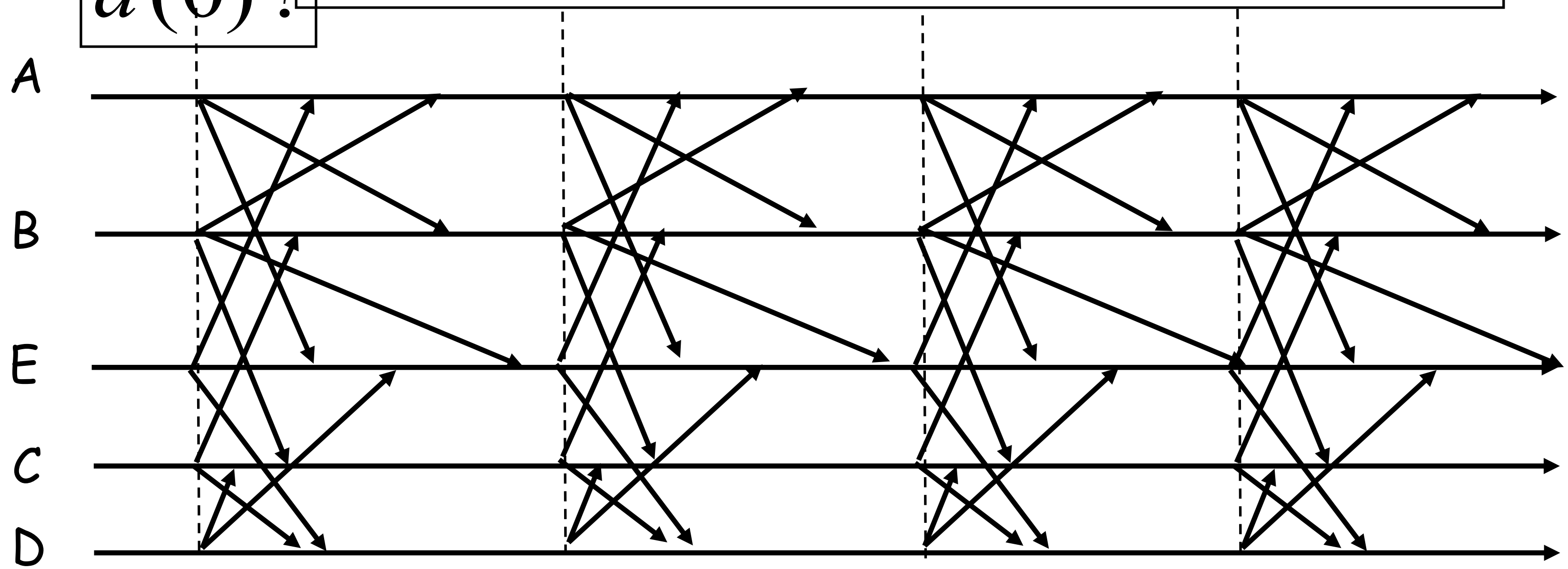
Synchronous Bellman-Ford (SBF)



- Nodes update in rounds:
  - there is a global clock;
  - at the beginning of each round, each node sends its estimate to all of its neighbors;
  - at the end of the round, updates its estimation

$$\boxed{d_i(h+1) = \min_{j \in N(i)} (d_{ij} + d_j(h))}$$

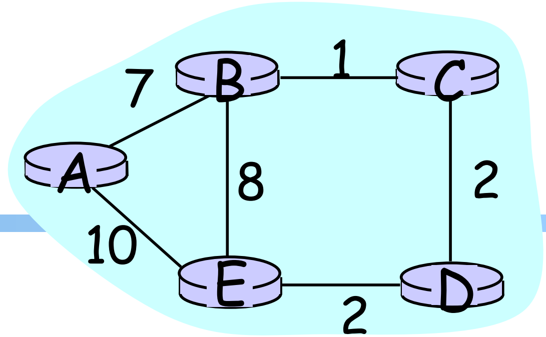
$$\boxed{d(0)\,?}$$



Outline

- Admin and recap
- Network overview
- Network control-plane path
  - Routing
    - Link weights assignment
    - Routing computation
      - *distributed distance vector protocols*
        - *synchronous Bellman-Ford (SBF)*
          - SBF/∞

SBF/∞

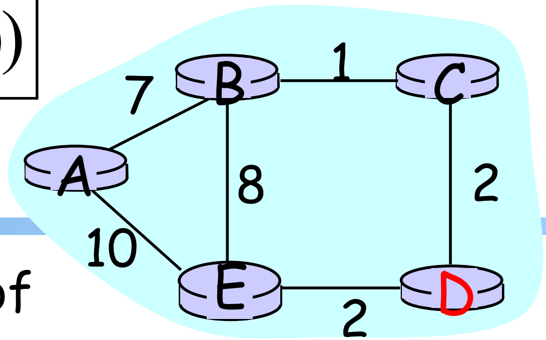


- Initialization (time 0):

$$d_i(0) = \begin{cases} 0 & \text{i} = \text{dest} \\ \infty & \text{otherwise} \end{cases}$$

$$d_i(h+1) = \min_{j \in N(i)} (d_{ij} + d_j(h))$$

Example



Consider D as destination; d(t) is a vector consisting of estimation of each node at round t

	A	B	C	E	D
d(0)	∞	∞	∞	∞	0
d(1)	∞	∞	2	2	0
d(2)	12	3	2	2	0
d(3)	10	3	2	2	0
d(4)	10	3	2	2	0

Observation: $d(0) \geq d(1) \geq d(2) \geq d(3) \geq d(4) = d^*$