Network Routing: Distance Vector Routing

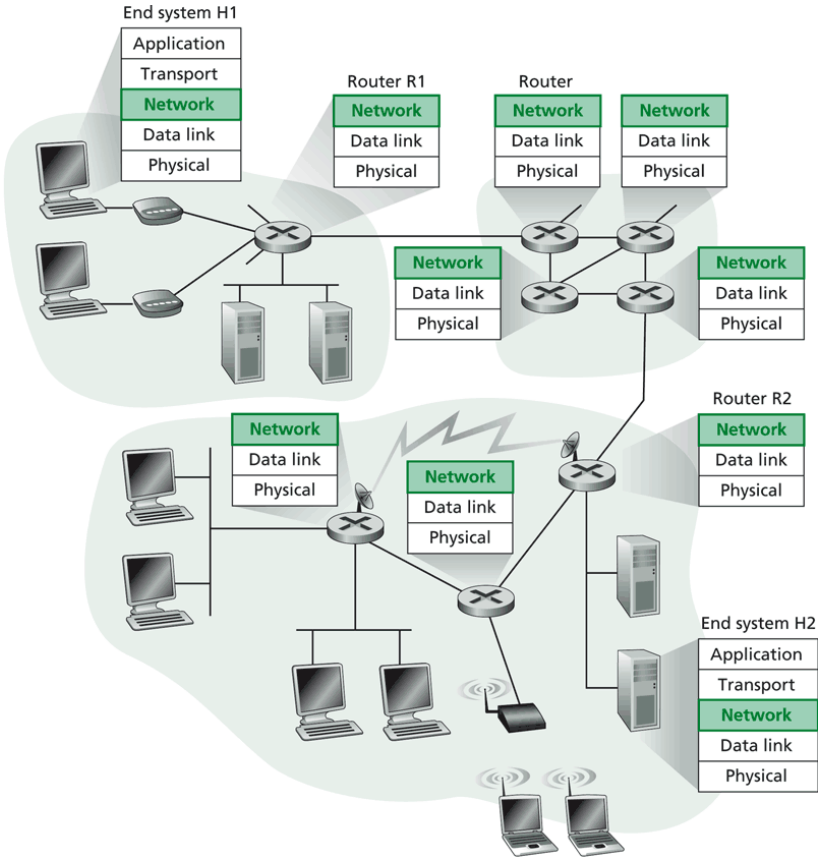
Reading KR 4.1, 4.2

Network Layer

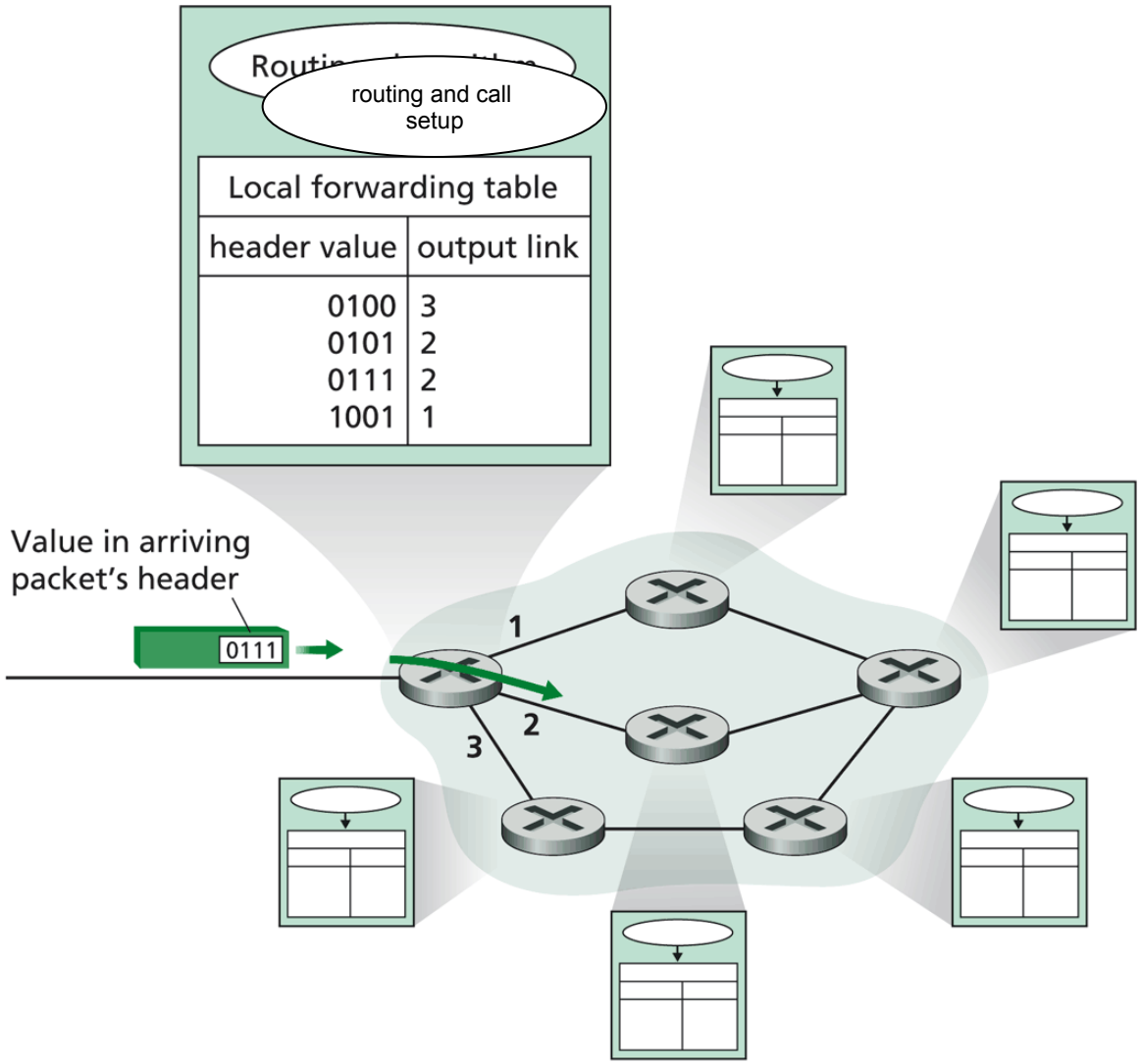
- Transport packet from source to dest.
- Network layer in *every* host, router

Basic functions:

- *Data plane: forwarding*
  - Move packets from router's input port to router output port
- *Control plane: path determination and call setup*
  - Determine route taken by packets from source to destination



Forwarding: Illustration



Network Layer: Complexity Factors

- For users: quality of service
  - Guaranteed bandwidth?
  - Preservation of inter-packet timing (no jitter)?
  - Loss-free delivery?
  - In-order delivery?
- For network providers
  - Efficiency
  - Policy of route control
  - Scalability
- Interaction between users and network providers
  - Signaling: congestion feedback/resource reservation

Network Layer Quality of Service

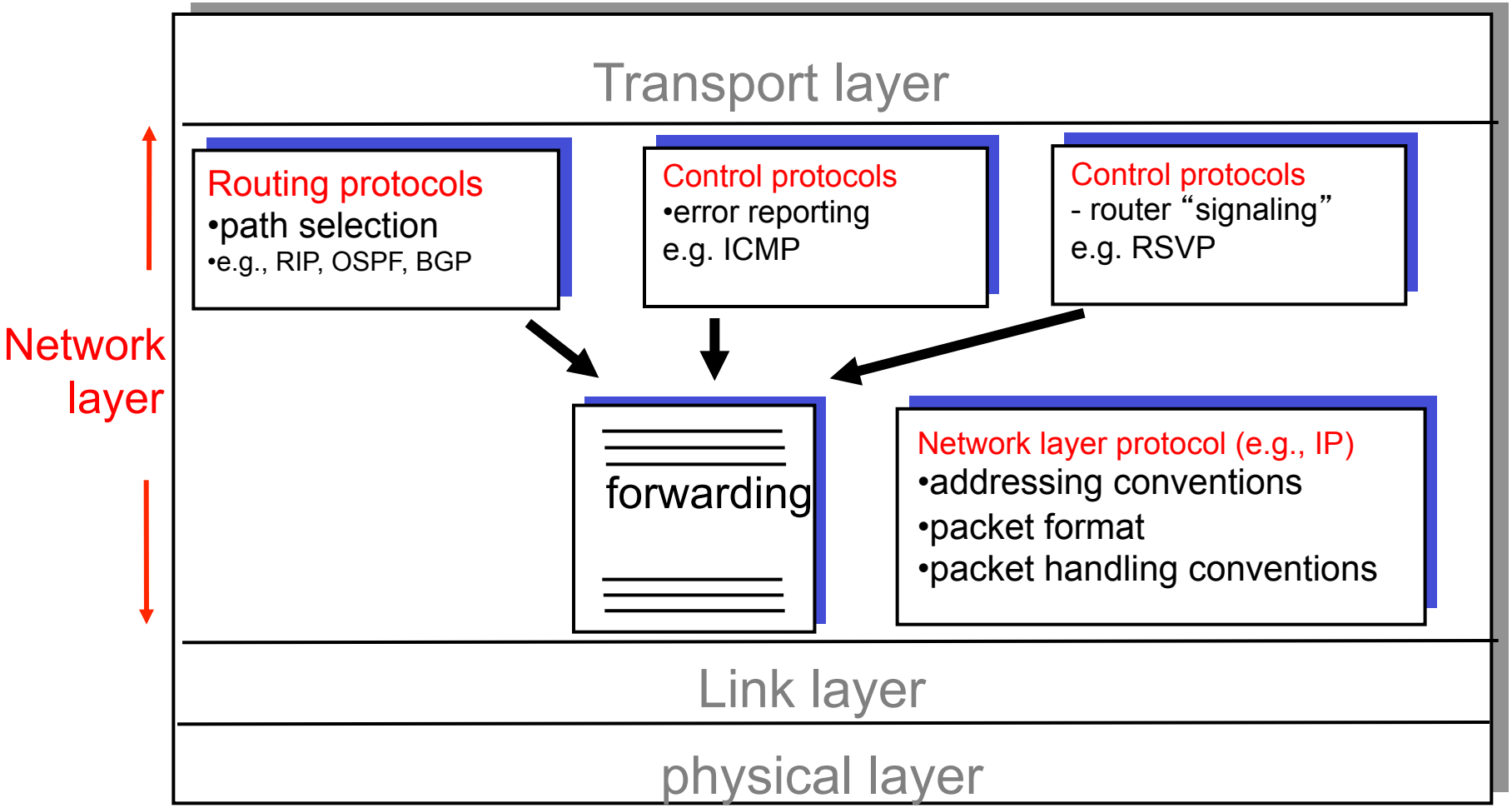
Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss/delay)
ATM	CBR	constant rate	yes	yes	yes	congestion won't occur
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

- Internet model being extended: Intserv, Diffserv
  - Multimedia networking

ATM: Asynchronous Transfer Mode; CBR: Constant Bit Rate; V: Variable; A: available; U: User

Network Layer: Protocols

Network layer functions:



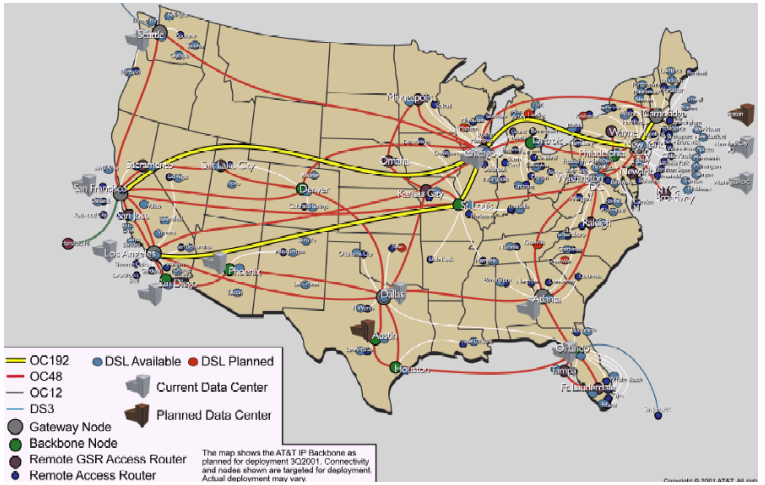
Outline

- Recap
- Network overview
- *Control plane: routing overview*

Control Plane: Routing

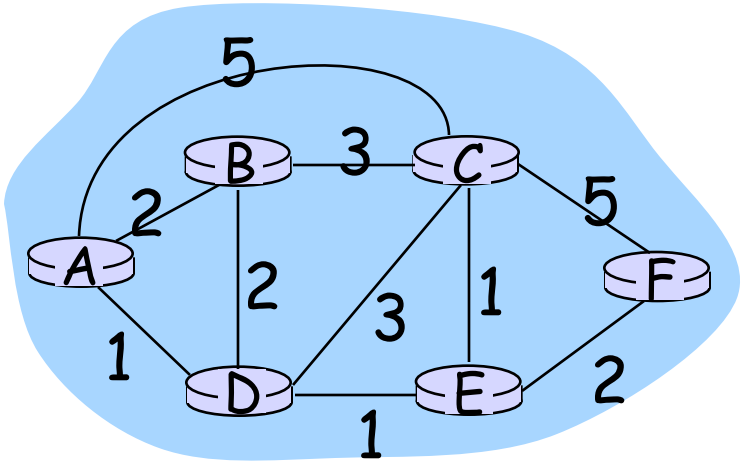
Routing

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



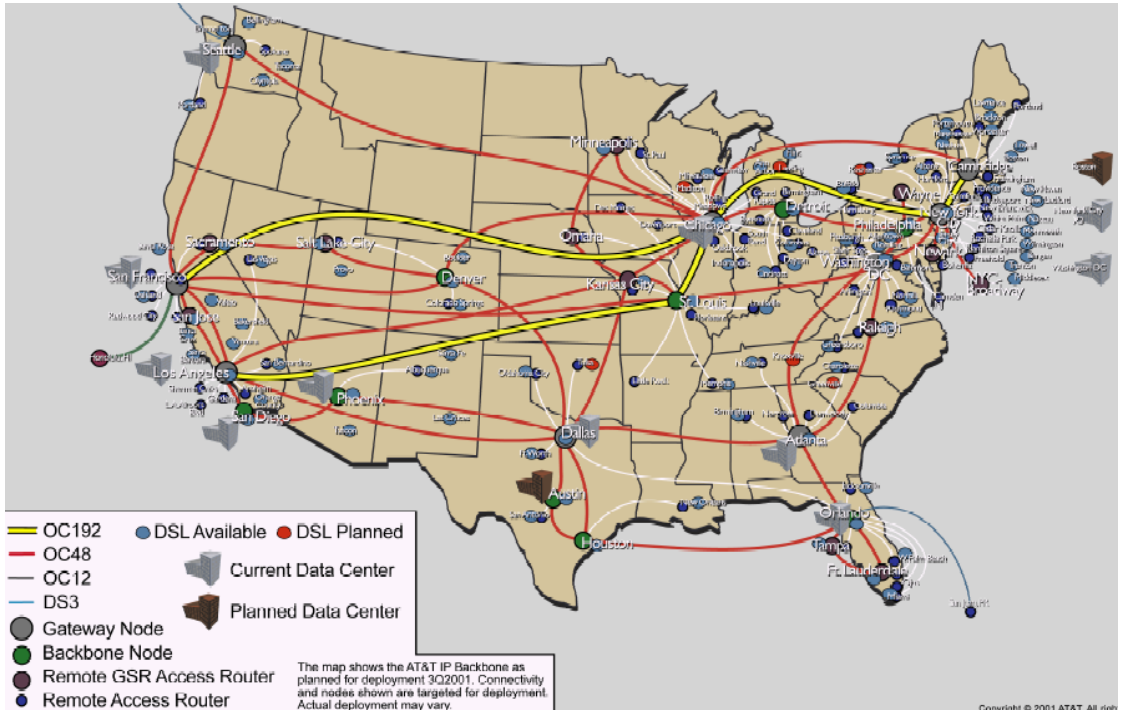
Graph abstraction for the routing problem:

- Nodes are routers
- Edges are physical links
  - Links have properties: delay, capacity, $ cost, policy



Key Desired Properties of a Routing Algorithm

- Robustness
- Optimality
  - Find good path (for user/ provider)
- Simplicity



Routing Design Space

- Robustness
- Optimality
- Simplicity

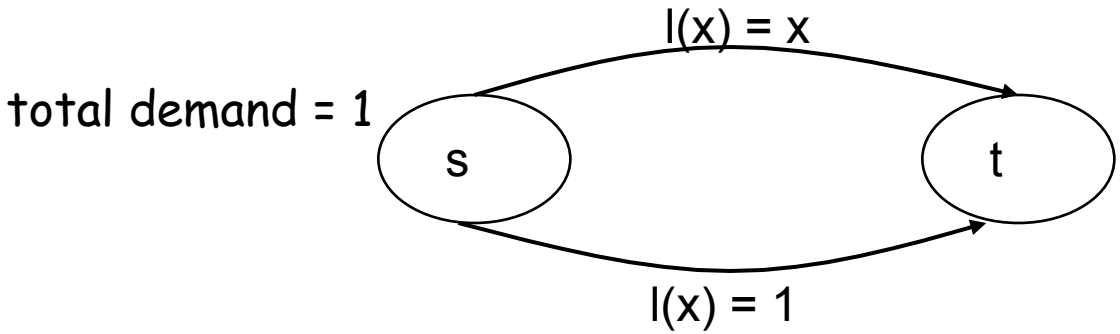
- Routing has a large design space
  - Who decides routing?
    - Source routing: end hosts make decision
    - Network routing: networks make decision
  - How many paths from source s to destination d?
    - Multi-path routing
    - Single path routing
  - Will routing adapt to network traffic demand?
    - Adaptive routing
    - Static routing
  - …

Example: Latency Optimal Routing

- Robustness
- Optimality
- Simplicity

- Assume each link has a latency function $l_e(x)$: latency of link e when x amount of traffic goes through e,
  - E.g., *prop.* + $1/(\mu - x)$, where prop. is link propagation delay and μ the link capacity

- Objective: min $\Sigma_e f_e \bullet l_e(f_e)$

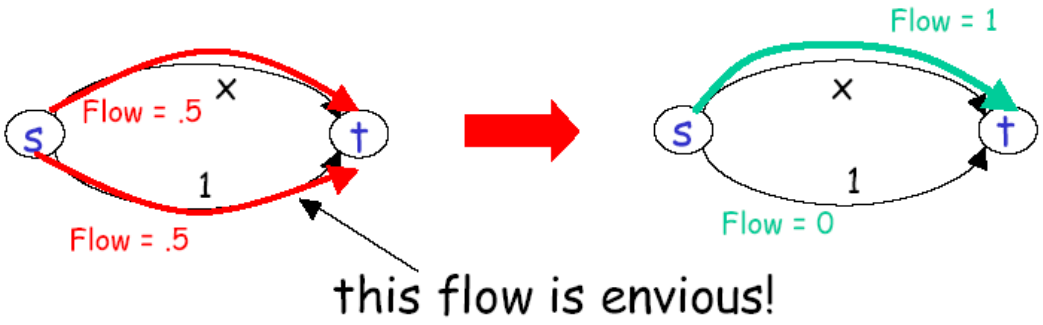


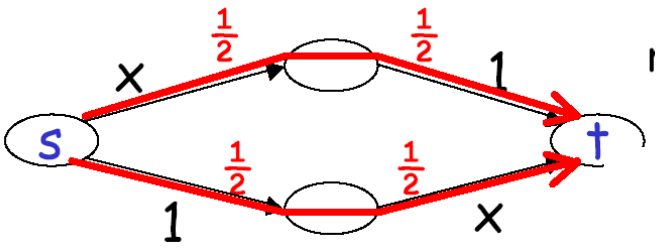
Routing Design Space: User-based, Multipath, Adaptive

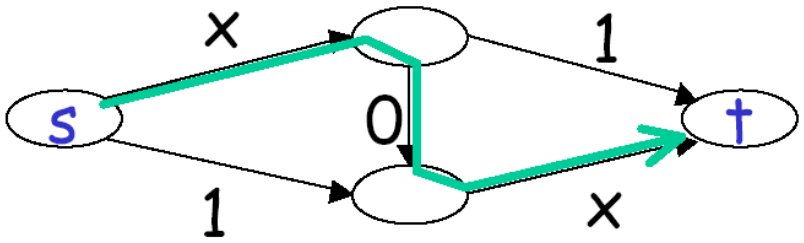
- Routing has a large design space
  - Who decides routing?
    - ➢ Source routing: end hosts make decision
    - Network routing: networks make decision
  - How many paths from sources to destination *d*?
    - ➢ Multi-path routing
    - Single path routing
  - Will routing adapt to network traffic 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

$\leq 4/3$

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

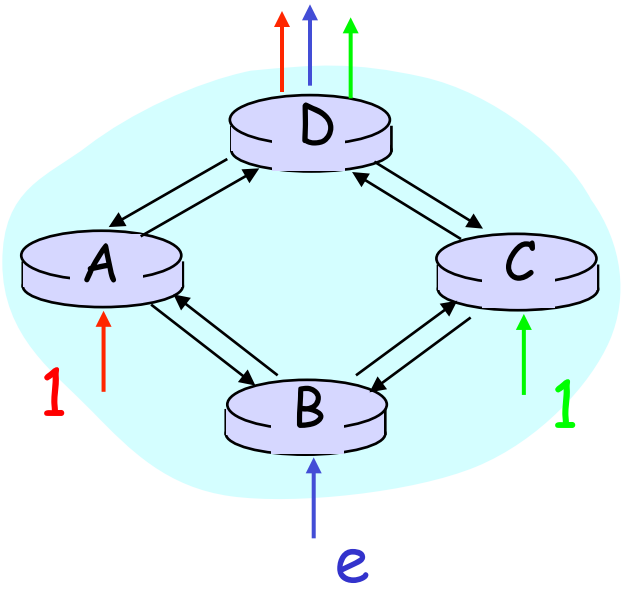
Routing Design Space: Internet

- Robustness
- Optimality
- Simplicity

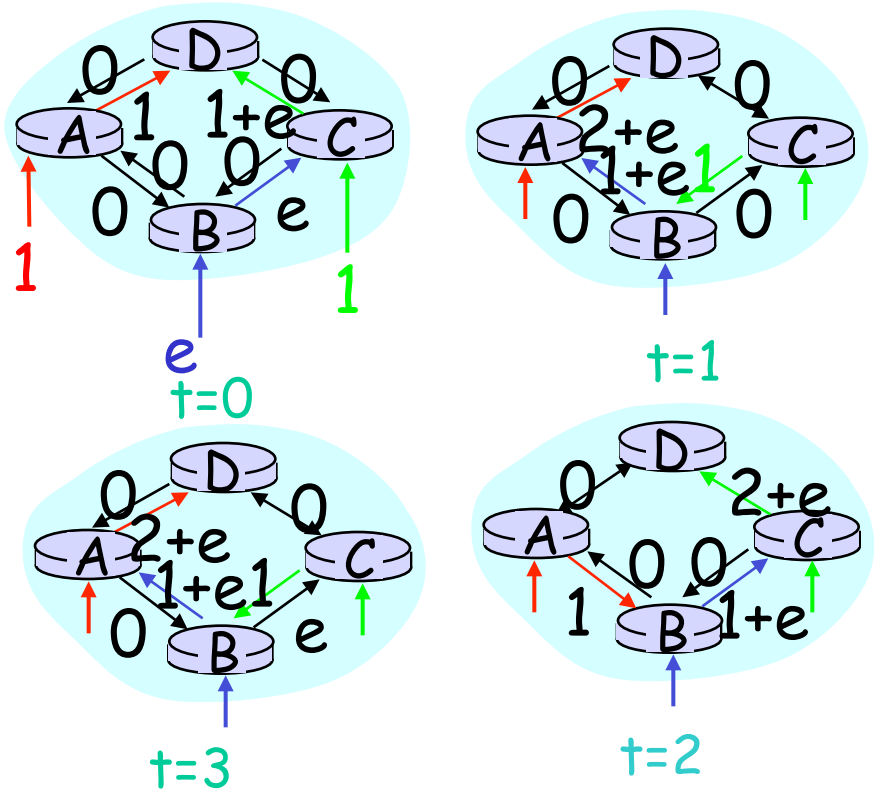
- Routing has a large design space
  - Who decides routing?
    - Source routing: end hosts make decision
    - Network routing: networks make decision
      - Applications such as overlay and p2p are trying to bypass it
  - How many paths from source s to destination d?
    - Multi-path routing
    - 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)
  - …

Routing: Single-Path, Adaptive Routing

- Assume link costs 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.

Shortest Path Routing Protocols

- Static (positive) costs assigned to network links, the path from a source to a destination chosen by the routing protocol is a shortest path among all possible paths
  - The static link costs may be adjusted in a longer time scale: this is called traffic engineering

Outline

- Recap
- Network overview
- Control plane: routing overview
  - *Distance vector protocols*

Distance Vector Routing

- Basis of RIP, IGRP, EIGRP routing protocols
- Based on the Bellman-Ford algorithm (BFA)
- Conceptually, runs for each destination separately

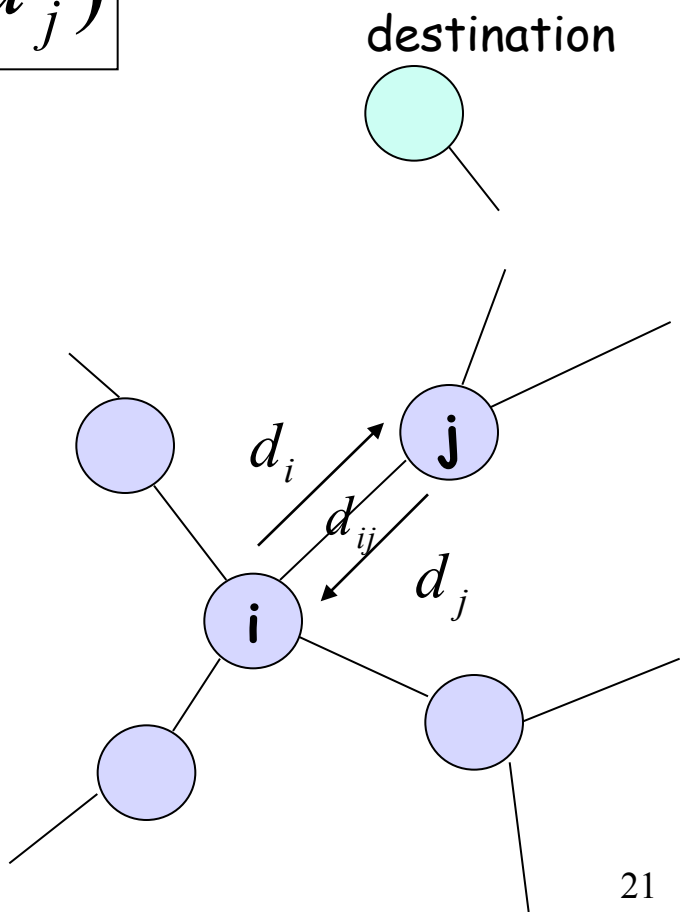
Distance Vector Routing: Basic Idea

- At node *i*, the basic update rule

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

where

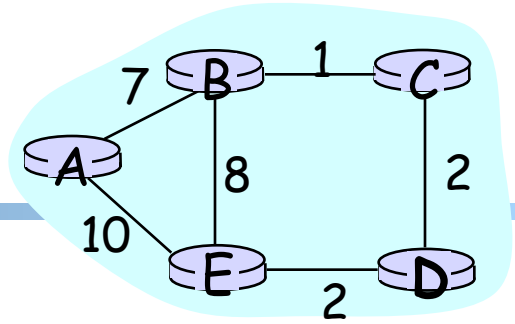
- d_i denotes the distance estimation from *i* to the destination,
- *N(i)* is set of neighbors of node *i*, and
- d_{ij} is the distance of the direct link from *i* to *j*; assume positive



Outline

- Recap
- Network overview
- Control plane: routing overview
- *Distance vector protocols*
  - *Synchronous Bellman-Ford (SBF)*

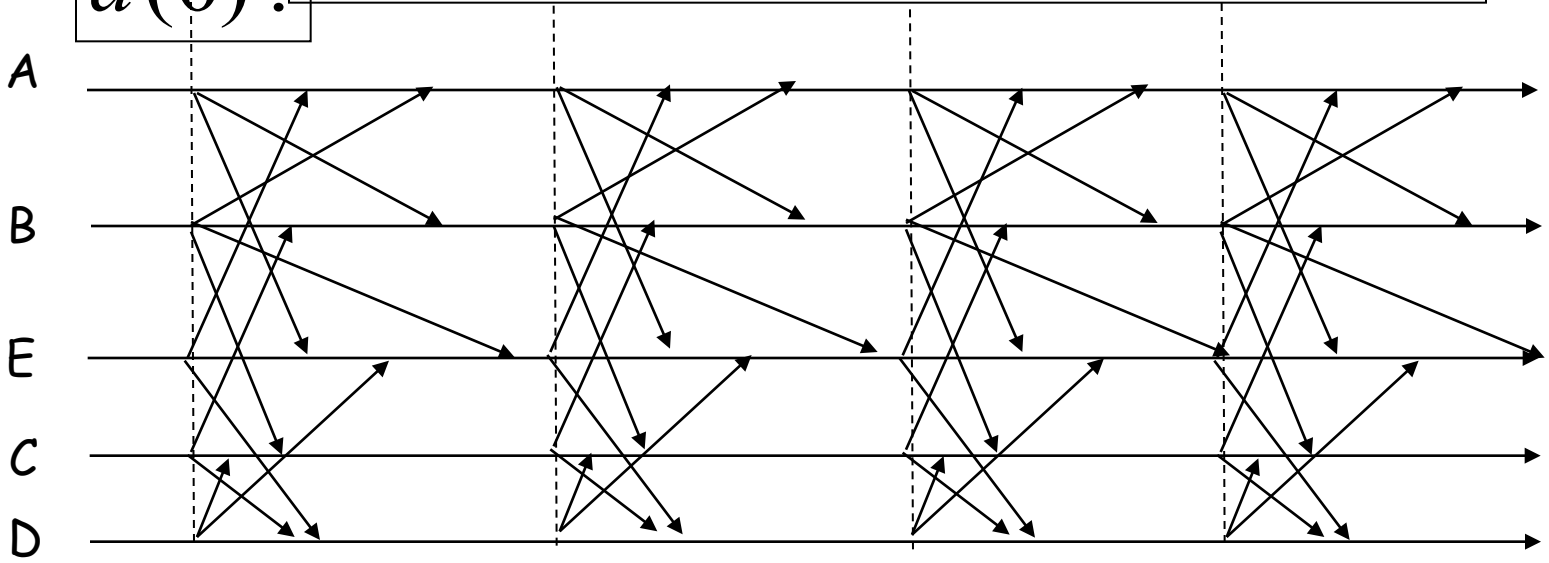
Synchronous Bellman-Ford (SBF)



- 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

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

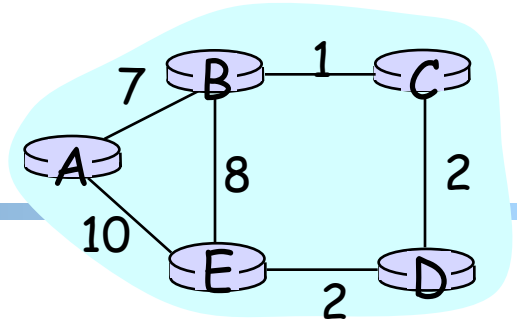
$d(0)$?



Outline

- Recap
- Network overview
- Control plane: routing overview
- *Distance vector protocols*
  - *synchronous Bellman-Ford (SBF)*
    - SBF/∞

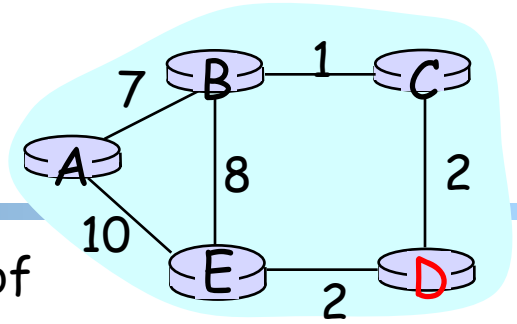
SBF/∞



- Initialization (time 0):

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

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

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

A Nice Property of SBF: Monotonicity

- Consider two configurations *d(t)* and *d'(t)*
- If *d(t)* ≥ *d'(t)*
  - I.e., each node has a higher estimate in one scenario (*d*) than in another scenario (*d'*),
- Then *d(t+1)* ≥ *d'(t+1)*
  - I.e., each node has a higher estimate in *d* than in *d'* after one round of synchronous update.

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

Correctness of SBF/∞

- Claim: $d_i(h)$ is the length $L_i(h)$ of a shortest path from i to the destination using $\leq h$ hops
  - Base case: h = 0 is trivially true
  - Assume true for ≤ h,
    i.e., $L_i(h) = d_i(h)$, $L_i(h-1) = d_i(h-1)$, …

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

Correctness of SBF/∞

- consider ≤ h+1 hops:

$$L_i(h+1) = \min(L_i(h), \min_{j \in N(i)}(d_{ij} + L_j(h)))$$

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

$$= \min(d_i(h), d_i(h+1))$$

since $d_i(h) \leq d_i(h-1)$

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

$$L_i(h+1) = d_i(h+1)$$

Bellman Equation

- We referred to the equations as Bellman equations (BE):

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

where $d_D = 0$.

- SBF/∞ solves the equations in a distributed way
- Does the equation have a unique solution (i.e., the shortest path one)?

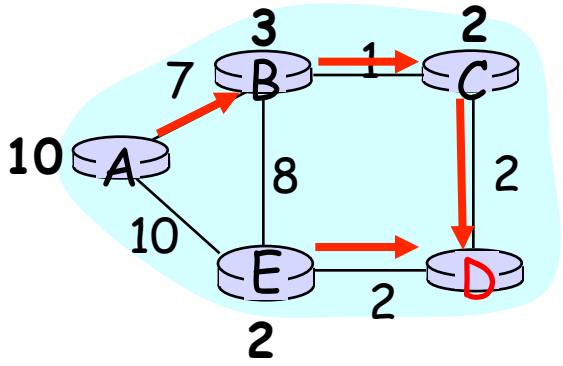
$$d_i = \min_{j \in N(i)} (d_{ij} + d_j)$$

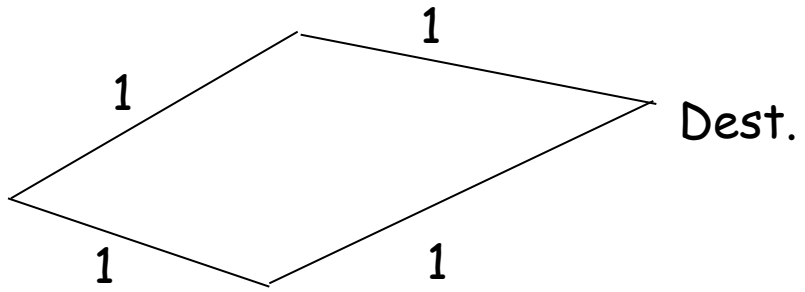
Uniqueness of Solution to BE

- Assume another solution d, we will show that $d = d^*$

case 1: we show $d \geq d^*$

Since d is a solution to BE, we can construct paths as follows: for each i, pick a j which satisfies the equation; since d^* is shortest, $d \geq d^*$





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

Uniqueness of Solution to BE

Case 2: we show d ≤ d*

Assume we run SBF with two initial configurations:

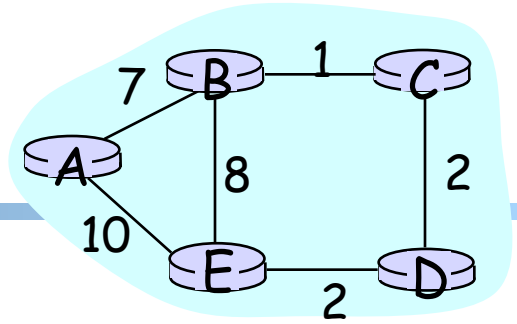
- One is d
- Another is SBF/∞ (d^∞),

-> Monotonicity and convergence of SBF/∞ imply that d ≤ d*

Outline

- Recap
- Network overview
- Control plane: routing overview
- *Distance vector protocols*
  - *synchronous Bellman-Ford (SBF)*
    - SBF/∞
    - SBF/-1

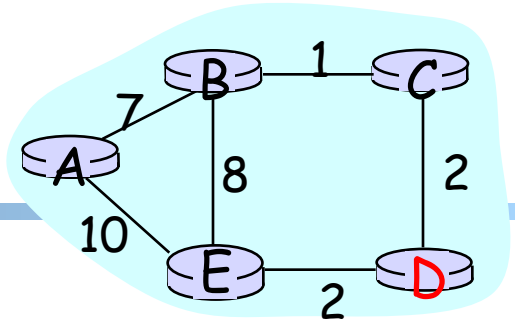
SBF at another Initial Configuration: SBF/-1



- Initialization (time 0):

$$d_i(0) = \begin{cases} 0 & \text{i} = \text{dest} \\ -1 & \text{otherwise} \end{cases}$$

Example



Consider D as destination

	A	B	C	E	D
d(0)	-1	-1	-1	-1	0
d(1)	6	0	0	2	0
d(2)	7	1	1	2	0
d(3)	8	2	2	2	0
d(4)	9	3	2	2	0
d(5)	10	3	2	2	0
d(6)	10	3	2	2	0

Observation: $d(0) \le d(1) \le d(2) \le d(3) \le d(4) \le d(5) = d(6) = d^*$

Bellman Equation and Correctness of SBF/-1

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

- SBF/-1 converges due to monotonicity
- At equilibrium, SBF/-1 satisfies the set of equations called Bellman equations (BE)

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

where $d_D = 0$.

- Another solution is shortest path solution d*
- Since there is a unique solution to the BE equations; thus SBF/-1 converges to shortest path

Question: will SBF converge under other non-negative initial conditions?

Problems of running *synchronous* BF?

Outline

- Recap
- Network overview
- Control plane: routing overview
- *Distance vector protocols*
  - *synchronous Bellman-Ford (SBF)*
  - *asynchronous Bellman-Ford (ABF)*

Asynchronous Bellman-Ford (ABF)

- No notion of global iterations
  - Each node updates at its own pace
- With finite delay, each node i computes

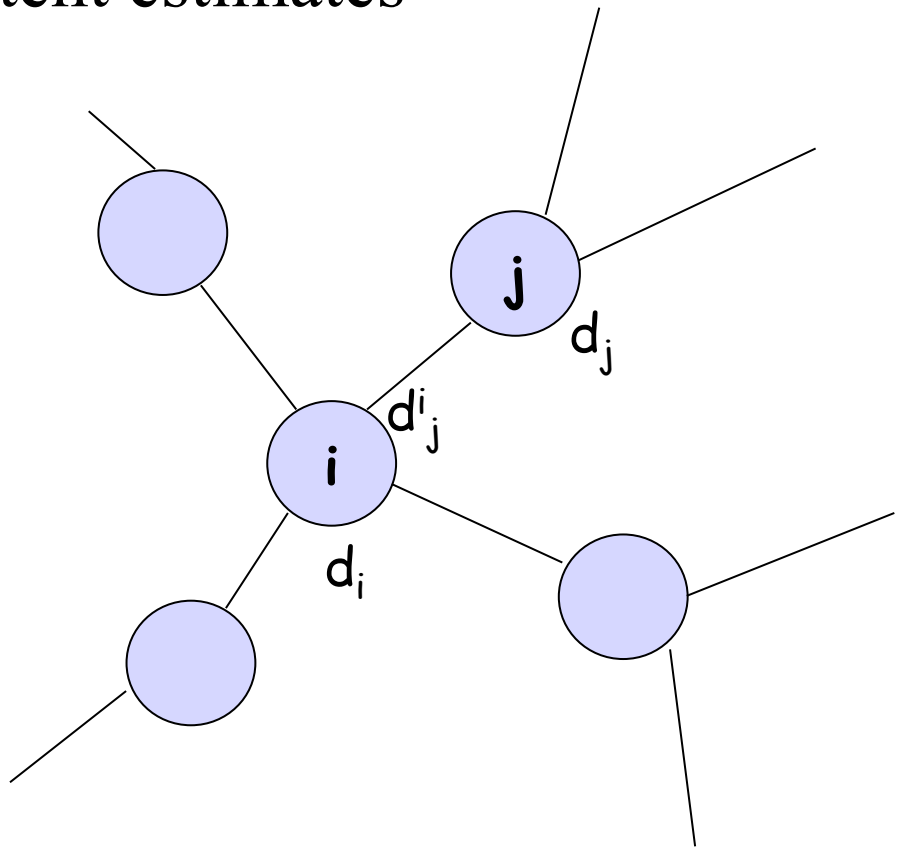
$$d_i = \min_{j \in N(i)} (d_{ij} + d_j^i)$$

using last received value d^i_j from neighbor j.

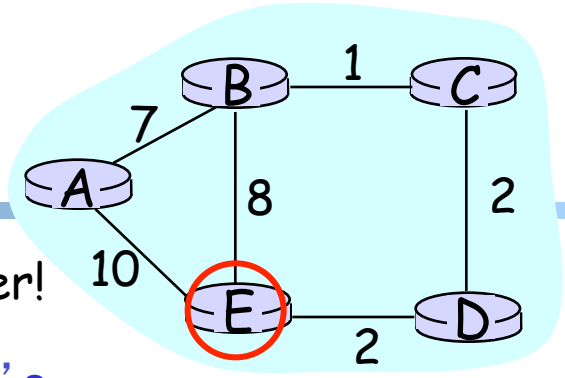
- With finite delay, node *j* sends its estimate to its neighbor *i*:
  - There is an upper bound on the delay of estimate packets

Asynchronous Bellman-Ford (ABF)

- In general, nodes are using different and possibly inconsistent estimates



Distance Table: Example



Below is just one step! The algorithm repeats forever!

	distance tables from neighbors			computation			E's distance table	distance table E sends to its neighbors
$d_E()$	A	B	D	A	B	D		
A	0	7	∞	10	15	∞	A: 10	A: 10
B	7	0	∞	17	8	∞	B: 8	B: 8
C	∞	1	2	∞	9	4	D: 4	C: 4
D	∞	∞	0	∞	∞	2	D: 2	D: 2
	10	8	2					E: 0

destinations

Asynchronous Bellman-Ford (ABF)

- Regardless of how asynchronous the nodes are, the algorithm will eventually converge to the shortest path
  - Links can go down and come up – but if topology is stabilized after some time *t*, the algorithm will eventually converge to the shortest path !

- If the network is connected, then ABF converges in finite amount of time