Network Layer:
Distance Vector Protocols Variations

Qiao Xiang, Congming Gao, Qiang Su

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

11/20/2025

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

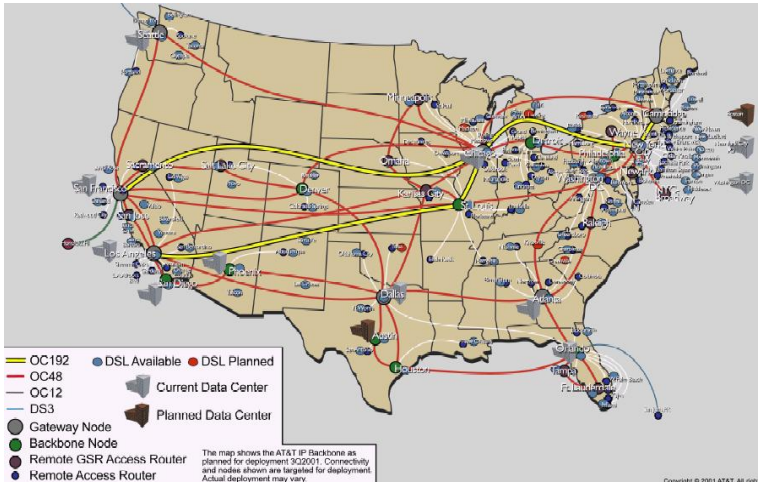
Outline

- Admin and recap
- Network overview
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Distance vector protocols (distributed computing)
      - Link state protocols (distributed state synchronization)

Recap: Routing Context

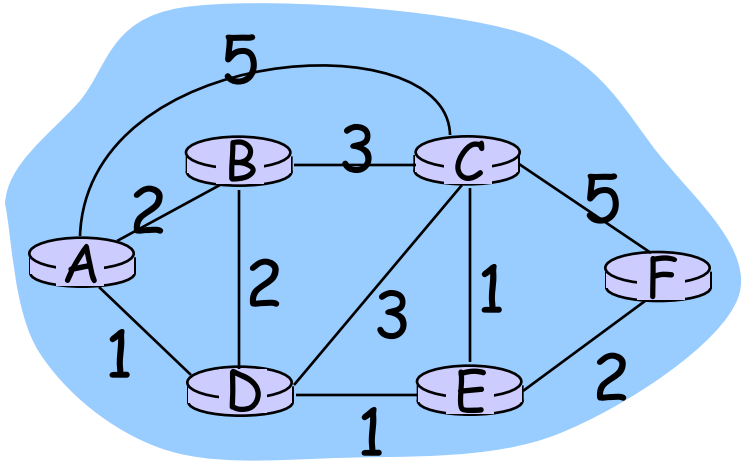
Routing

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



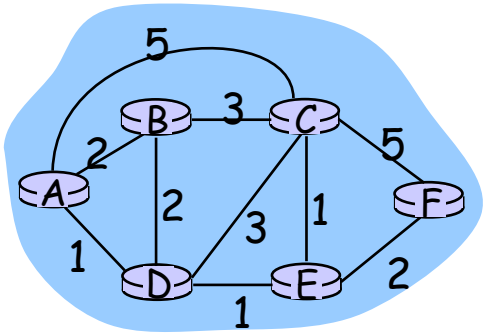
Often depends on a graph abstraction:

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



Recap: Routing Design Space

- 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 (shortest path routing)
    - QoS aware
  - will routing adapt to network traffic demand?
    - adaptive routing
    - static routing
  - ...



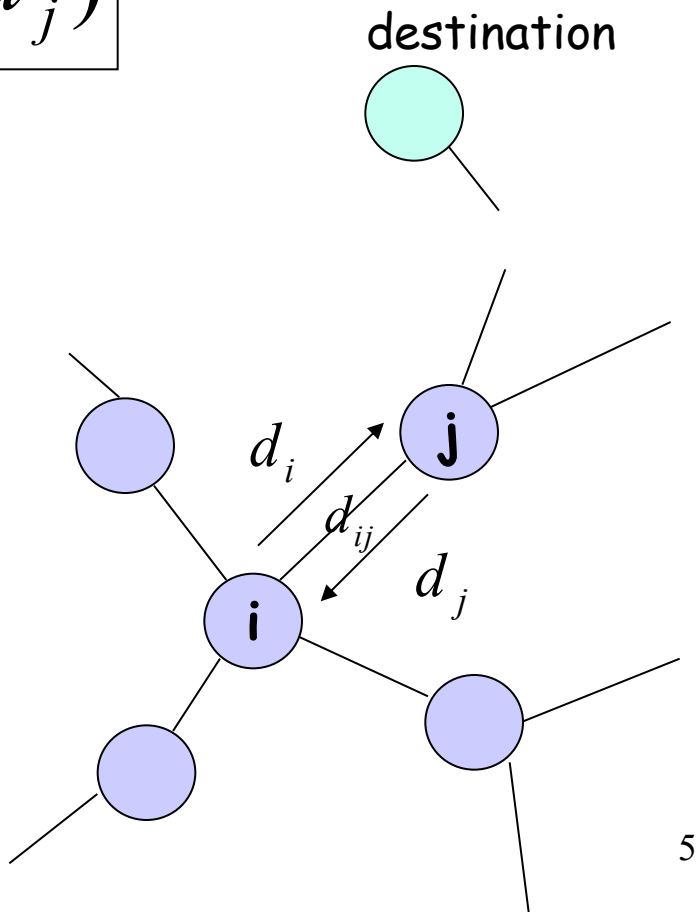
Recap: Distance Vector Routing: Basic Idea (Bellman-Ford Alg)

- At node i, the basic update rule

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

where

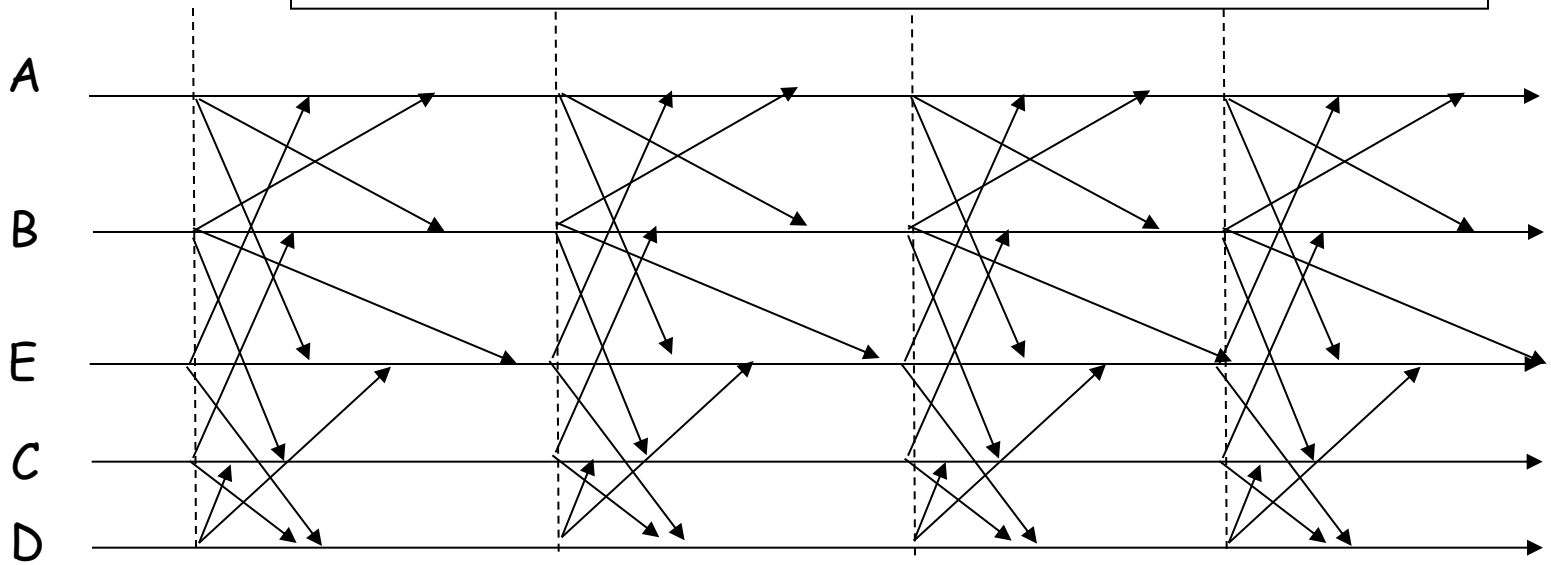
- d_i denotes the distance estimation from i to the destination,
- N(i) is set of neighbors of node i, and
- d_{ij} is the distance of the direct link from i to j



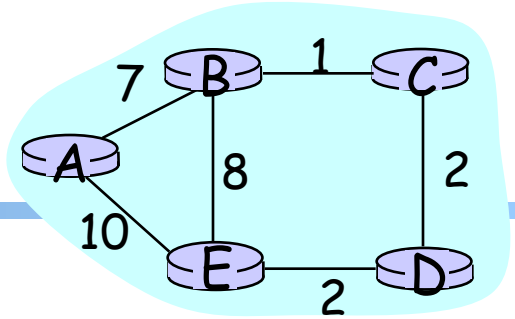
Recap: Synchronous Bellman-Ford (SBF)

- 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

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



Recap: SBF/∞

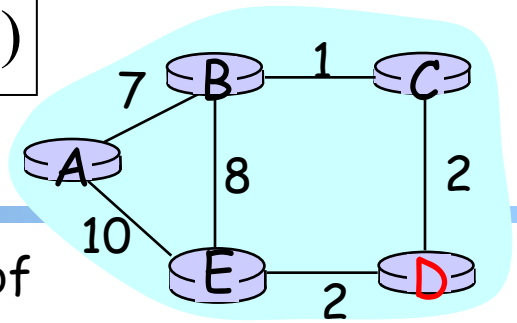


- Initialization (time 0):

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

Example

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



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

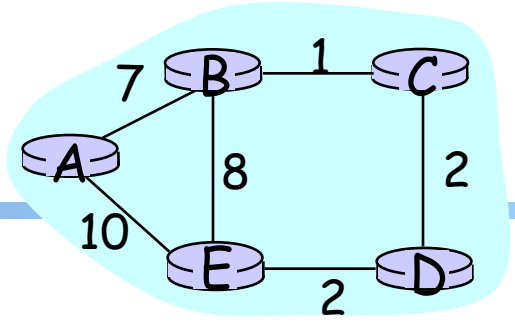
$$d_i(h+1) = \min_{j \in N(i)} (d_{ij} + d_j(h)) \leq \min_{j \in N(i)} (d_{ij} + d_j(h-1)) = d_i(h)$$

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

Outline

- Admin and recap
- Network overview
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - *Distributed distance vector protocols*
        - *synchronous Bellman-Ford (SBF)*
          - SBF/∞
          - SBF/-1 SBF/∞

SBF at another Initial Configuration: SBF/-1

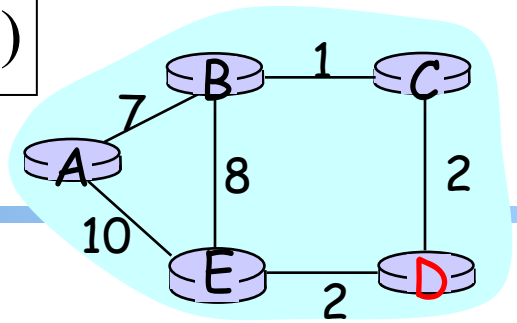


- Initialization (time 0):

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

Example

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



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	3	2	0
d(5)	10	3	3	2	0
d(6)	10	3	3	2	0

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

Correctness of SBF/-1

- SBF/-1 converges due to monotonicity

- Remaining question:
  - Can we guarantee that SBF/-1 converges to shortest path?

Correctness of SBF/-1

- Common between SBF/∞ and SBF/-1: they solve the Bellman equation

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

where $d_D = 0$.

- We have proven SBF/∞ is the shortest path solution.
- SBF/-1 computes shortest path if Bellman equation has a unique solution.

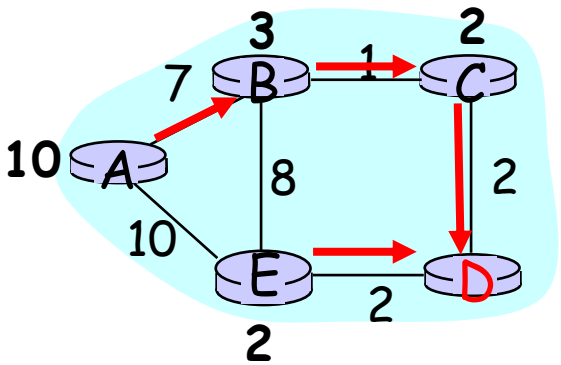
$$d_i = \min_{j \in N(i)} (d_{ij} + d_j)$$

Uniqueness of Solution to BE

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

case 1: we show d ≥ 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 ≥ 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*

Discussion

- Will SBF converge under other non-negative initial conditions?

- Problems of running *synchronous* BF?

Outline

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

Asynchronous Bellman-Ford (ABF)

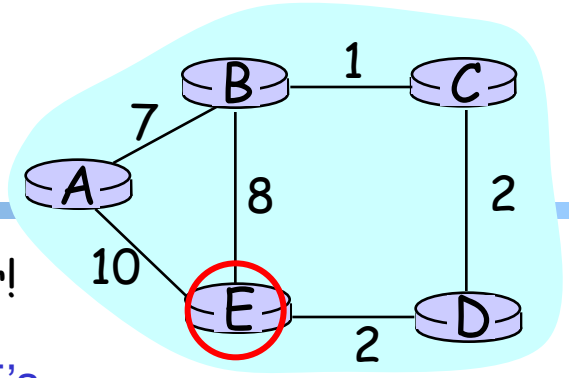
- No notion of global iterations
  - each node updates at its own pace
- Asynchronously each node i computes

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

using last received value d^i_j from neighbor j.

- Asynchronously node j sends its estimate to its neighbor i:
  - We assume that there is an upper bound on the delay of estimate packet

ABF: Example



Below is just one step! The protocol repeats forever!

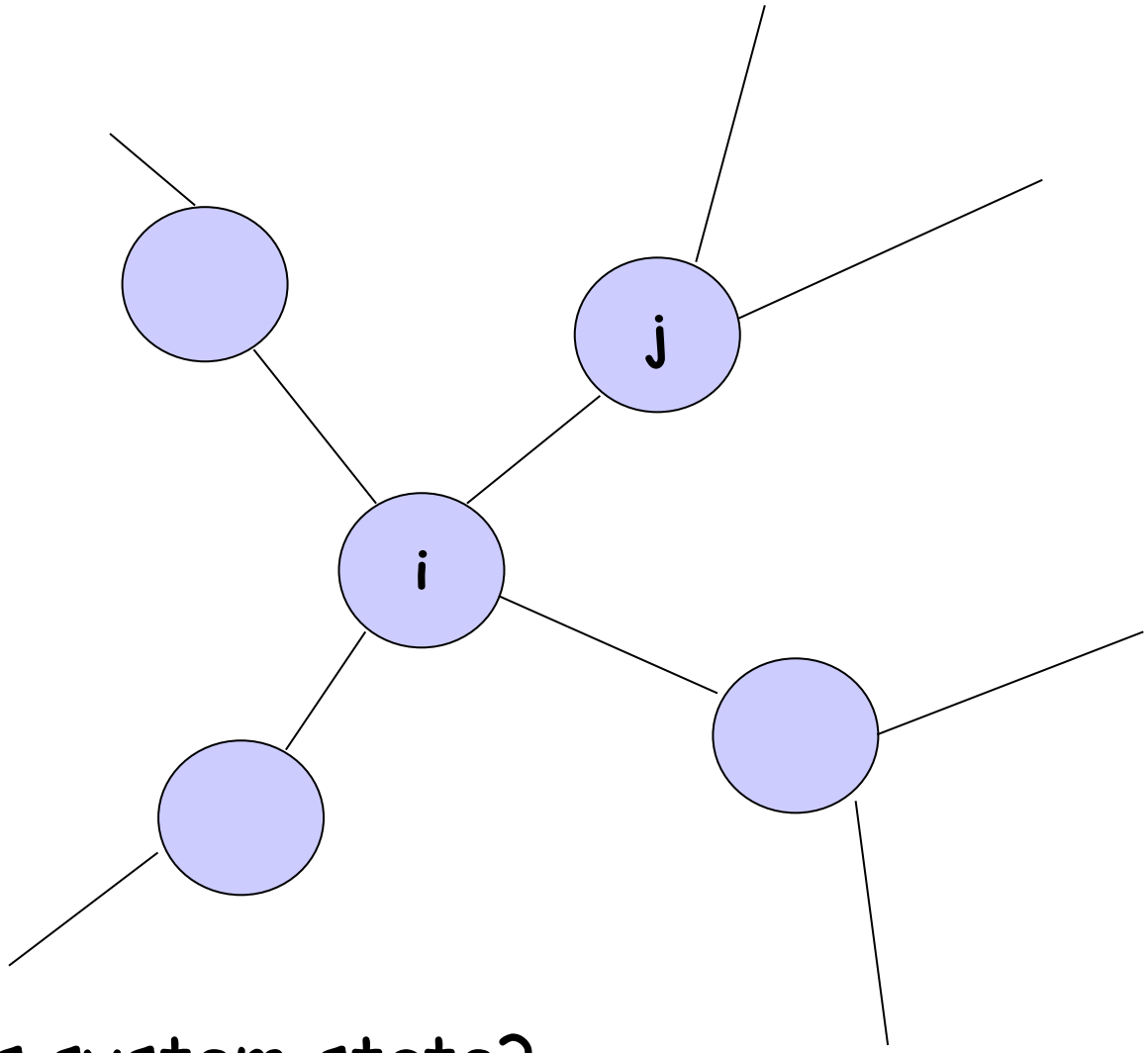
$d_E()$	distance tables from neighbors A	B	D	computation A	B	D	E's distance table	distance table E sends to its neighbors
A	0	7	∞	10	15	∞	A: 10	A: 10
B	7	0	∞	17	8	∞	B: 8	B: 8
C	∞	1	2	∞	9	4	D: 4	C: 4
D	∞	∞	0	∞	∞	2	D: 2	D: 2
	10	8	2					E: 0

destinations

Asynchronous Bellman-Ford (ABF)

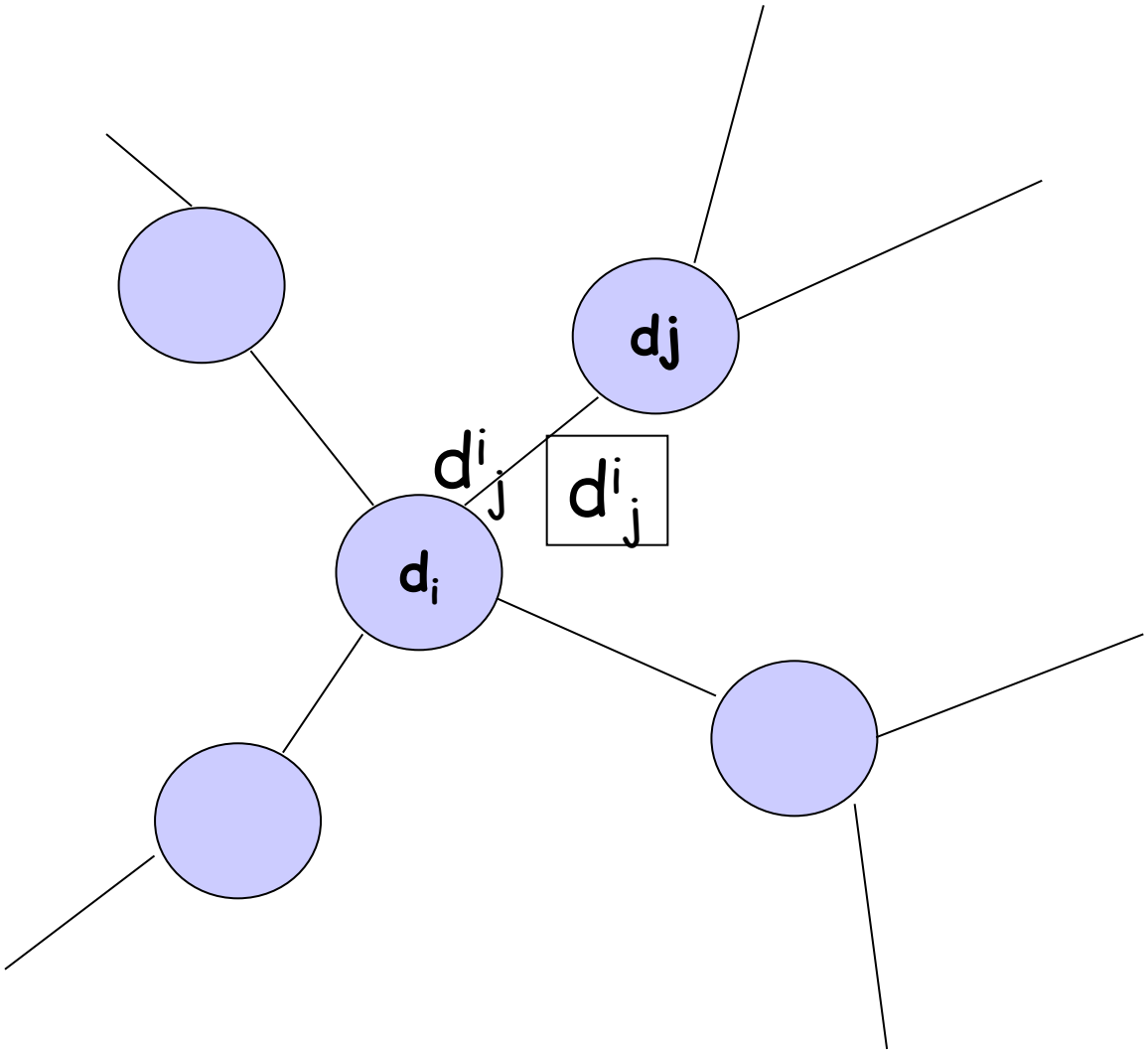
- ABF will eventually converge to the shortest path
  - links can go down and come up – but if topology is stabilized after some time t and connected, ABF will eventually converge to the shortest path !

ABF Convergence Proof Complexity: Complex System State



What is system state?

System State



three types of distance state from node j:

- d_j: current distance estimate state at node j

- d^i_j: last d_j that neighbor i received

- d^i_j: those d_j that are still in transit to neighbor i

ABF Convergence Proof: The Sandwich Technique

- Basic idea:
  - bound system state using extreme states
- Extreme states:
  - SBF/∞; call the sequence U()
  - SBF/-1; call the sequence L()

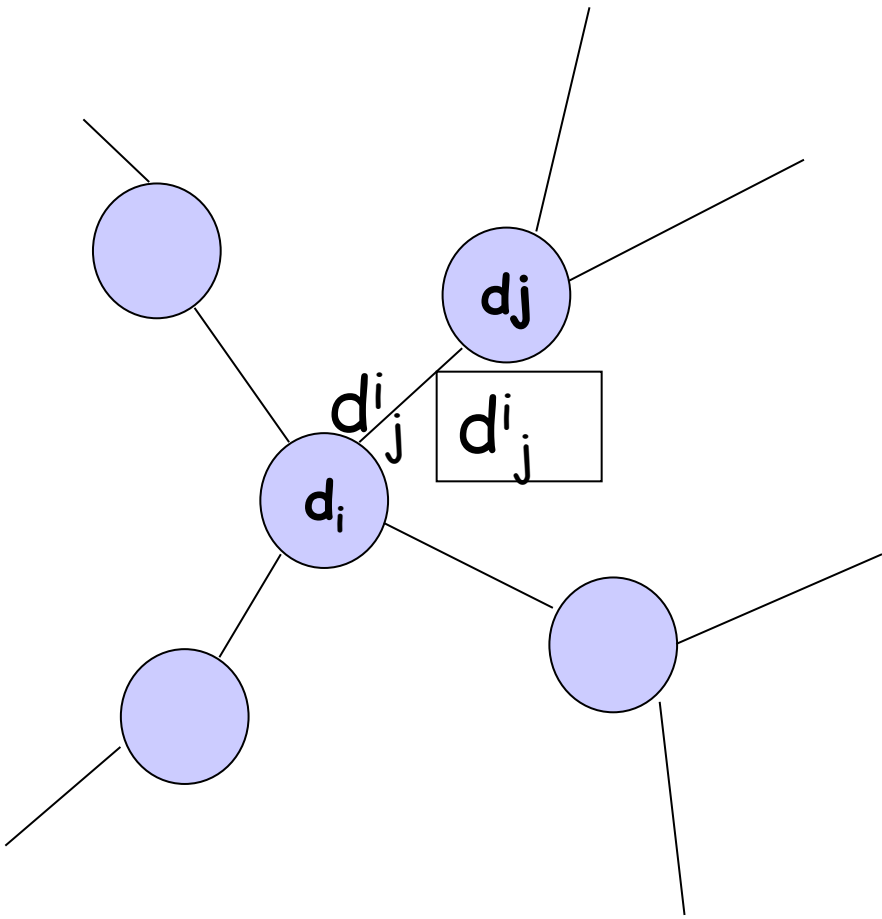
ABF Convergence

- Consider the time when the topology is stabilized as time 0

- U(0) and L(0) provide upper and lower bounds at time 0 on all corresponding elements of states
  - $L_j(0) \leq d_j \leq U_j(0)$ for all d_j state at node j
  - $L_j(0) \leq d^i_j \leq U_j(0)$
  - $L_j(0) \leq$ `update messages` $d^i_j \leq U_j(0)$

ABF Convergence

- d_j
  - after at least one update at node j: d_j falls between $L_j(1) \le d_j \le U_j(1)$
- d^i_j :
  - eventually all d^i_j that are only bounded by $L_j(0)$ and $U_j(0)$ are replaced with in $L_j(1)$ and $U_j(1)$



Asynchronous Bellman-Ford: Summary

- Distributed
  - each node communicates its routing table to its directly-attached neighbors
- Iterative
  - continues periodically or when link changes, e.g. detects a link failure
- Asynchronous
  - nodes need *not* exchange info/iterate in lock step!
- Convergence
  - in finite steps, independent of initial condition if network is connected

Summary: Distributed Distance-Vector

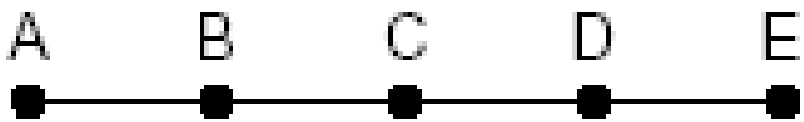
- Tool box: a key technique for proving convergence (liveness) of distributed protocols: monotonicity and bounding-box (sandwich) design
  - Consider two configurations d(t) and d'(t):
    - if d(t) <= d'(t), then d(t+1) <= d'(t+1)
  - Identify two extreme configurations to sandwich any real configurations

Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Distance vector protocols (distributed computing)
        - synchronous Bellman-Ford (SBF)
        - asynchronous Bellman-Ford (ABF)
        - *properties of DV*

Properties of Distance-Vector Algorithms

- Good news propagate fast



∞	∞	∞	∞	Initially
1	∞	∞	∞	After 1 exchange
1	2	∞	∞	After 2 exchanges
1	2	3	∞	After 3 exchanges
1	2	3	4	After 4 exchanges

Properties of Distance-Vector Algorithms

- Bad news propagate slowly

A-B link down

A	B	C	D	E	
	1	2	3	4	Initially
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges
	7	8	7	8	After 6 exchanges
		⋮			
	∞	∞	∞	∞	

- This is called the *counting-to-infinity* problem
- Q: what causes counting-to-infinity?

Counting-To-Infinity is Because of Routing Loop

- Counting-to-infinity is caused by a routing loop, which is a global state (consisting of the nodes' local states) at a global moment (observed by an oracle) such that there exist nodes A, B, C, … E such that A (locally) thinks B as next hop, B thinks C as next hop, … E thinks A as next hop

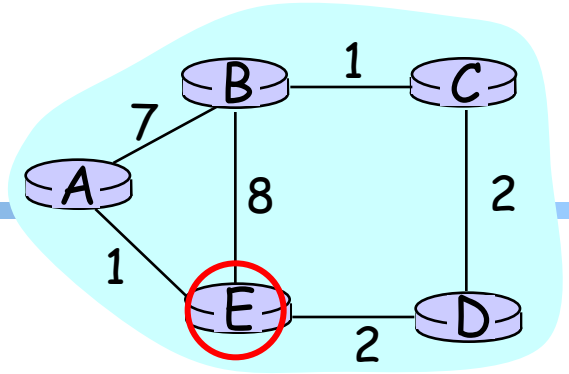
Discussion

- Why avoid routing loops is hard?
- Any proposals to avoid routing loops?

Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Distance vector protocols (distributed computing)
        - synchronous Bellman-Ford (SBF)
        - asynchronous Bellman-Ford (ABF)
        - properties of DV
          - DV w/ loop prevention
            - *reverse poison*

The Reverse-Poison (Split-horizon) Hack



If the path to dest is through neighbor h, report ∞ to neighbor h for dest.

distance tables from neighbors

$D^E()$	A	B	D
A	0	7	∞
B	7	0	∞
C	∞	1	2
D	∞	∞	0
	1	8	2
	c(E,A)	c(E,B)	c(E,D)

destinations

computation

A	B	D
1	15	∞
8	8	∞
∞	9	4
∞	∞	2

E's distance table

1, A
8, B
4, D
2, D

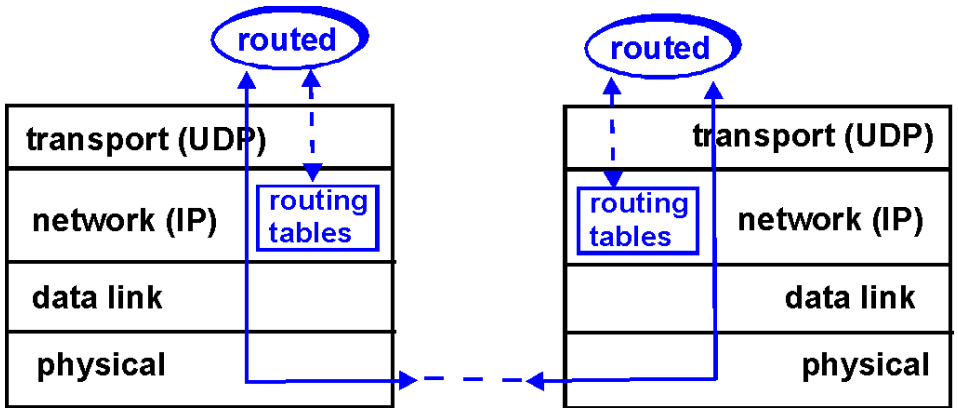
distance through neighbor

distance table E sends to its neighbors

To A	To B	To D
A: ∞	A: 1	A: 1
B: 8	B: ∞	B: 8
C: 4	C: 4	C: ∞
D: 2	D: 2	D: ∞
E: 0	E: 0	E: 0

DV+RP => RIP
(Routing Information Protocol)

- Included in BSD-UNIX Distribution in 1982
- Link cost: 1
- Distance metric: # of hops
- Distance vectors
  - exchanged every 30 sec via Response Message (also called **advertisement**) using UDP
  - each advertisement: route to up to 25 destination nets



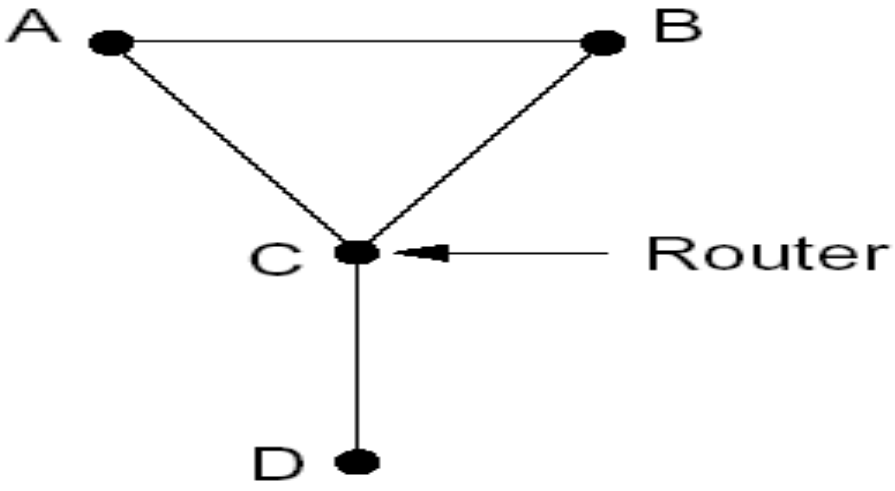
RIP: Link Failure and Recovery

If no advertisement heard after 180 sec --> neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly propagates to entire net
- reverse-poison used to prevent ping-pong loops
- set infinite distance = 16 hops (why?)

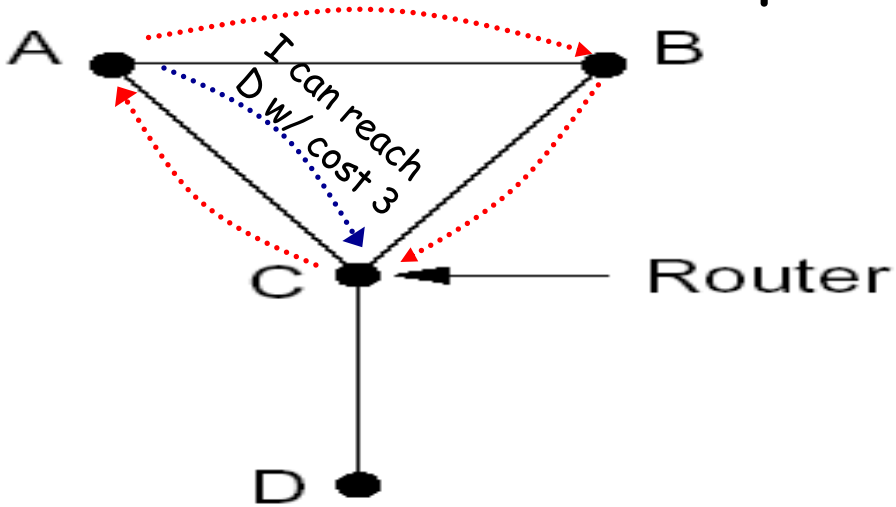
General Routing Loops and Reverse-poison

- Exercise: Can Reverse-poison guarantee no loop for this network?



General Routing Loops and Reverse-poison

- Reverse-poison removes two-node loops but may not remove more-node loops



- Unfortunate timing can lead to a loop
  - When the link between C and D fails, C will set its distance to D as ∞
  - A receives the bad news (∞) from C, A will use B to go to D
  - A sends the news to C
  - C sends the news to B

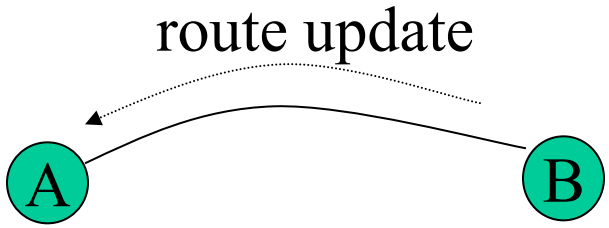
Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Distance vector protocols (distributed computing)
        - synchronous Bellman-Ford (SBF)
        - asynchronous Bellman-Ford (ABF)
        - properties of DV
          - DV w/ loop prevention
            - reverse poison
            - *destination-sequenced DV (DSDV)*

Destination-Sequenced Distance Vector protocol (DSDV)

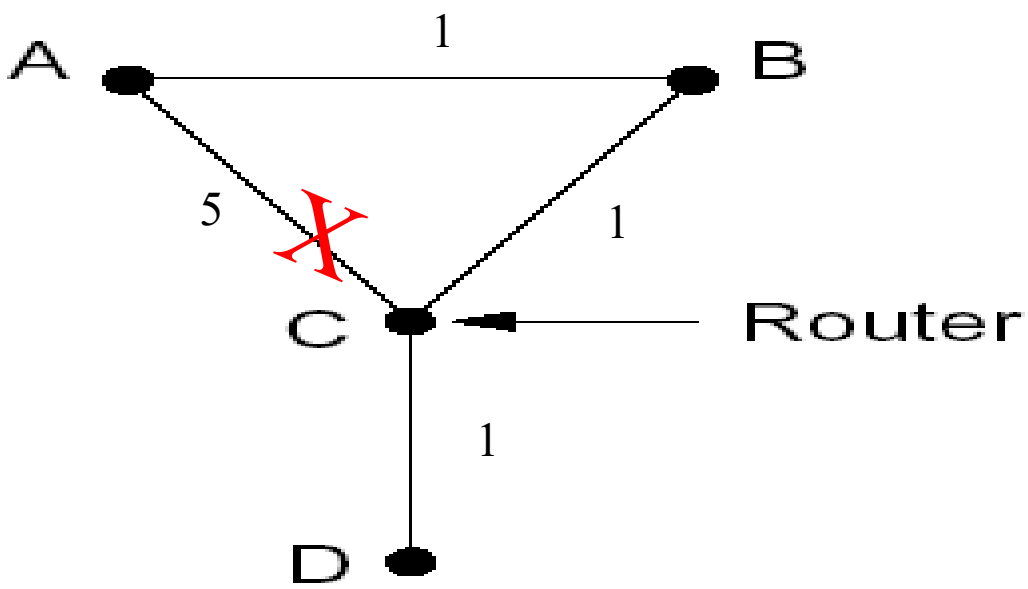
- Basic idea: use sequence numbers to partition computation
  - tags each route with a sequence number
  - each destination node D periodically advertises monotonically increasing even-numbered sequence numbers
  - when a node realizes that the link it uses to reach destination D is broken, it advertises an infinite metric and a sequence number which is one greater than the previous route (i.e., an odd seq. number)
    - the route is repaired by a later even-number advertisement from the destination

DSDV: More Detail



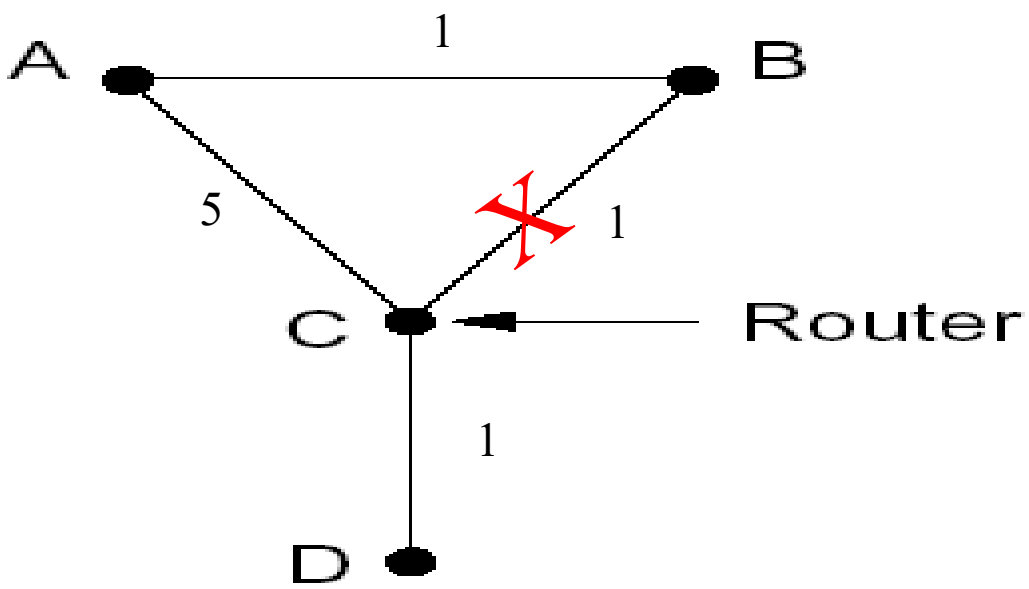
- Let's assume the destination node is D
- There are optimizations but we present a simple version:
  - each node B maintains (S^B, d^B), where S^B is the sequence number at B for destination D and d^B is the best distance using a neighbor from B to D
- Both periodical and triggered updates
  - periodically: D increases its seq. by 2 and broadcasts with (S^D, 0)
  - if B is using C as next hop to D and B discovers that C is no longer reachable
    - B increases its sequence number S^B by 1, sets d^B to ∞, and sends (S^B, d^B) to all neighbors

Example



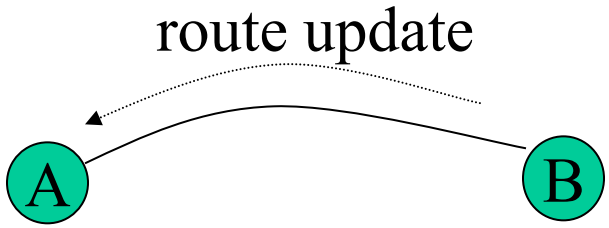
Will this trigger an update?

Example



Will this trigger an update?

DSDV: Update Alg.



- Consider simple version, no optimization
- Update after receiving a message
  - assume B sends to A its current state (S^B, d^B)
  - when A receives (S^B, d^B)
    - if $S^B > S^A$, then

      // always update if a higher seq#

      » $S^A = S^B$

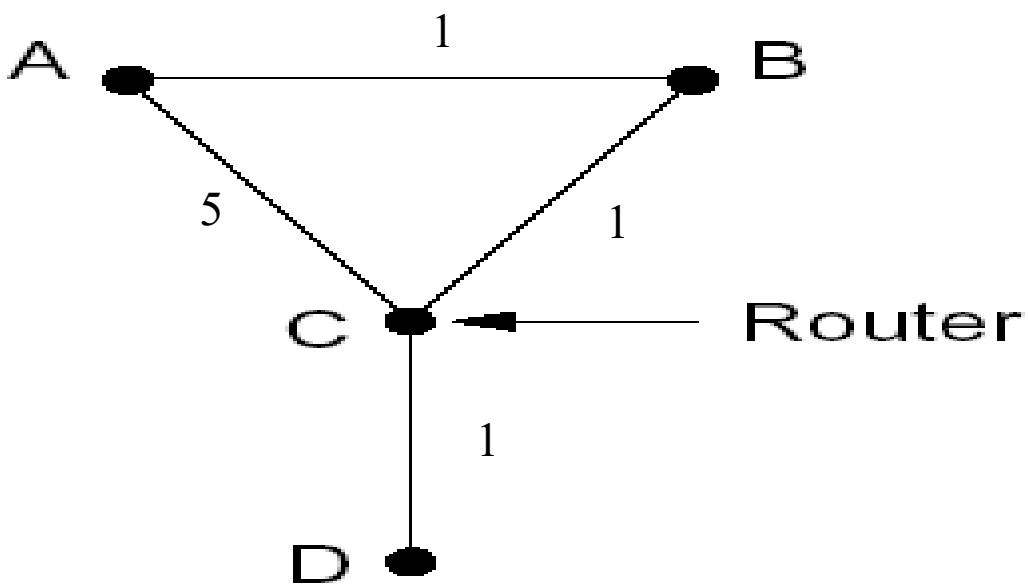
      » if ($d^B == \infty$) $d^A = \infty$; else $d^A = d^B + d(A,B)$
    - else if $S^A == S^B$, then

      » if $d^A > d^B + d(A,B)$

      // update for the same seq# only if better route

      $d^A = d^B + d(A,B)$ and uses B as next hop

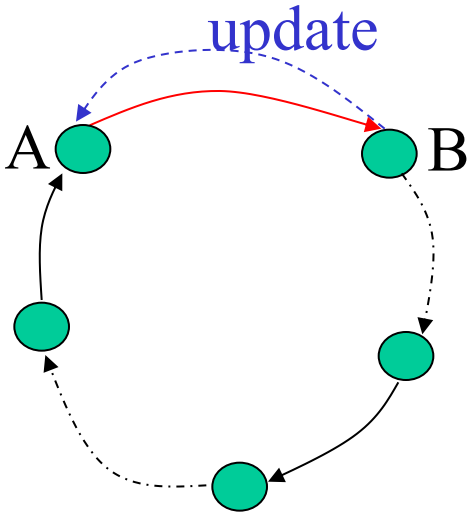
Example



Exercise: update process after D increases its seq# to next even number.

Claim: DSDV will NEVER Form a Loop

- Initially no loop (no one has next hop so no loop)
- Derive contradiction if a loop forms after a node processes an update,
  - e.g., when A receives the update from B, A decides to use B as next hop and forms a loop



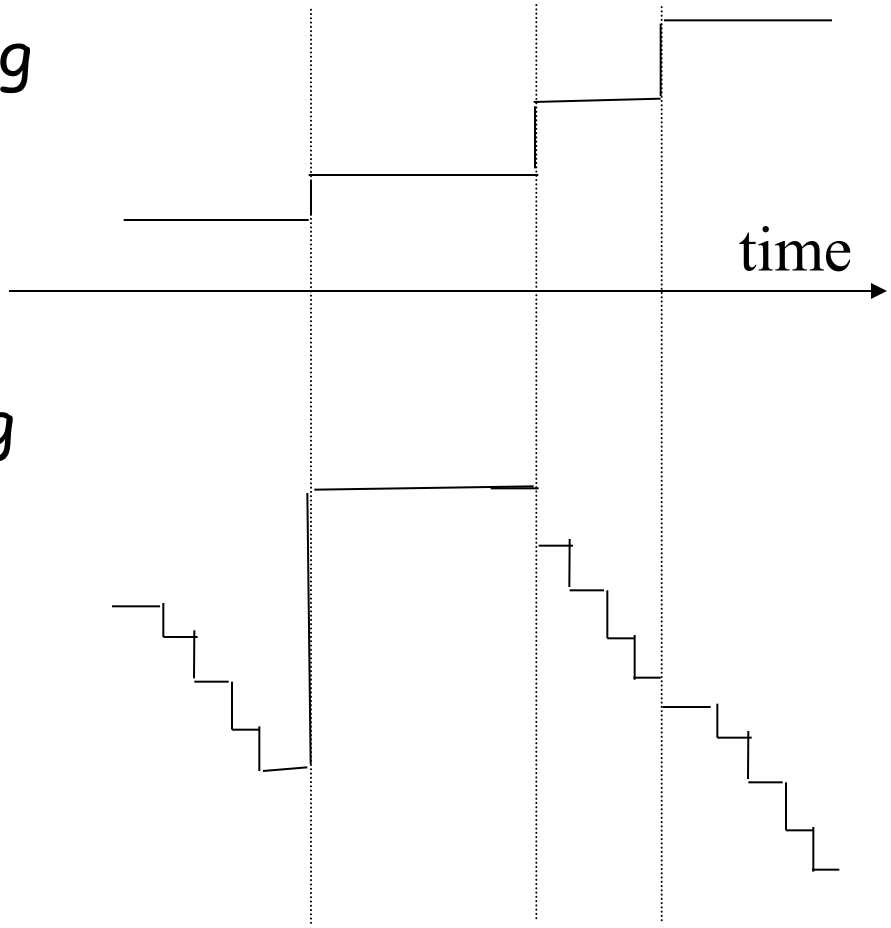
Technique: Global Invariants

- Global Invariant is a very effective method in understanding safety of distributed asynchronous protocols
- Invariants are defined over the states of the distributed nodes

- Consider any node B.
- Let's identify some invariants over the state of node B, i.e., (S^B, d^B).

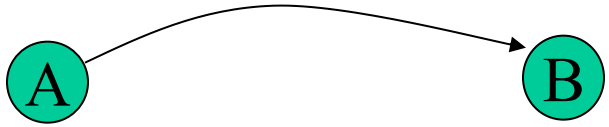
Invariants of a Single Node B

- Some invariants about the state of a node B
  - [I1] S^B is non-decreasing
  - [I2] d^B is non-increasing for the same sequence number



Invariants of if A Considers B as Next Hop

- Some invariants if A considers B as next hop



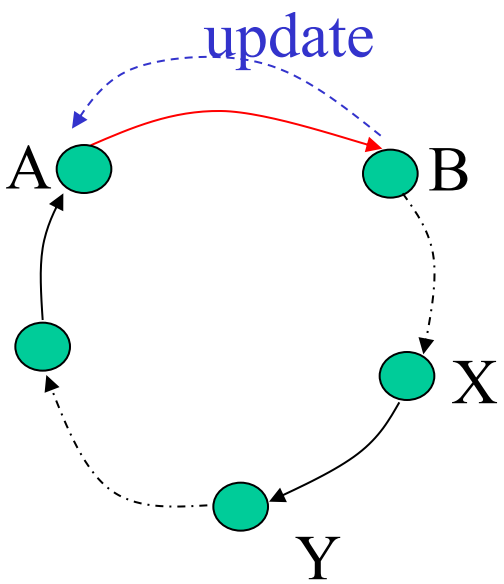
  - [I3] d^A is not ∞
  - [I4] $S^B >= S^A$

    because A is having the seq# which B last sent to A; B's seq# might be increased after B sent its state

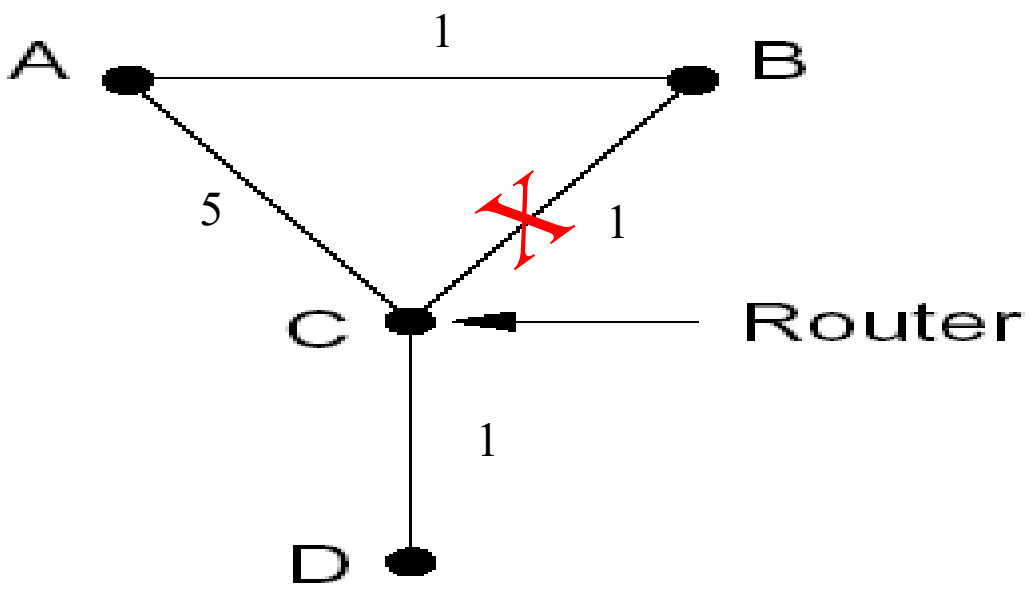
    - [I5] if $S^B == S^A$
    - then $d^B < d^A$ because d^A is based on d^B which B sent to A some time ago, $d^B < d^A$ since all link costs are positive; d^B might be decreased after B sent its state

Loop Freedom of DSDV

- Consider a critical moment
  - A starts to consider B as next hop, and we have a loop
- According to invariant I4 for each link in the loop (X considers Y as next hop) : $S^Y >= S^X$
- Two cases:
  - exists $S^Y > S^X$
    - by transition along the loop $S^B > S^B$
  - all nodes along the loop have the same sequence number
    - apply I5, by transition along the loop $d^B > d^B$



Issue of DSDV



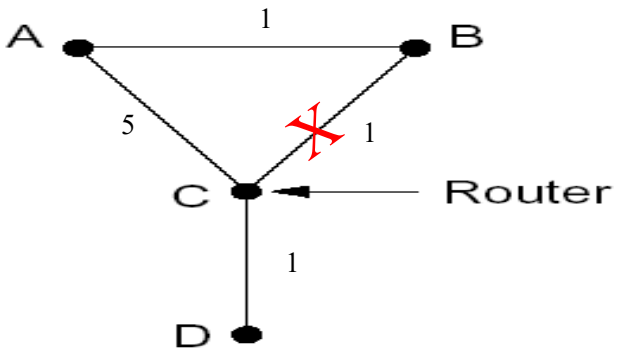
Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Distance vector protocols (distributed computing)
        - synchronous Bellman-Ford (SBF)
        - asynchronous Bellman-Ford (ABF)
        - properties of DV
          - DV w/ loop prevention
            - reverse poison
            - destination-sequenced DV (DSDV)
            - *diffusive update algorithm (DUAL) and EIGRP*

Basic Idea

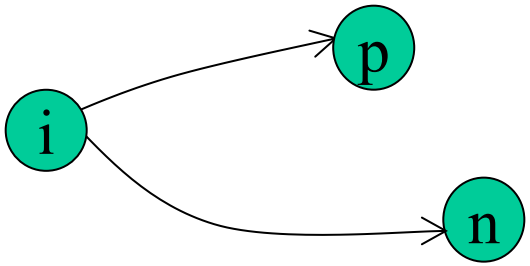
- DSDV guarantees no loop, but at the price of not using any backup path before destination re-announces reachability.

- Basic idea: Sufficient condition to guarantee no loop using backup paths (called switching)?



Key Idea: Feasible Successors

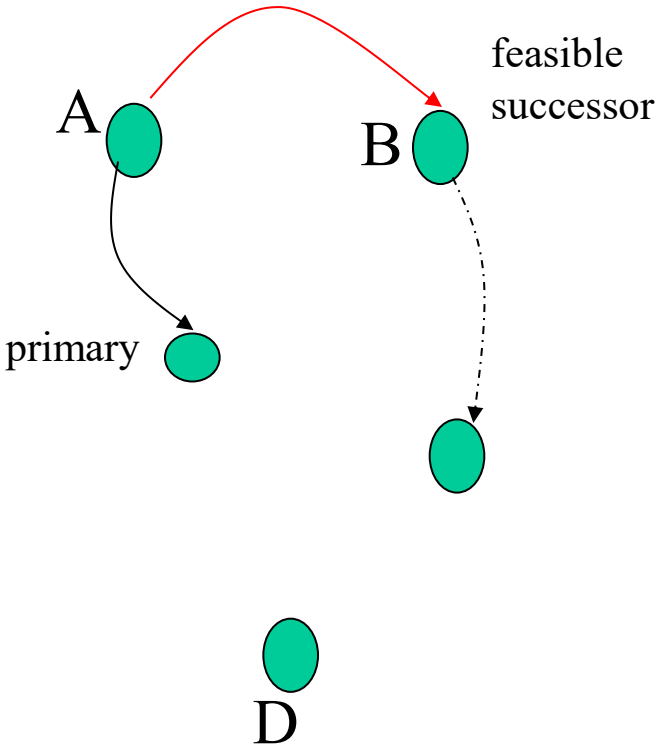
- If the reported distance of a neighbor n is lower than the total distance using primary (current shortest), the neighbor n is a feasible successor



$$d_n + d_{i \to \mathrm{n}} \geq d_{\mathrm{primary}} + d_{i \to \mathrm{primary}} > d_{\mathrm{n}}$$

Intuition

- Since the reported distance of B is lower than my total distance, B cannot be using me (along a path) to reach the destination



Example

```
A     (2)     B     (1)     C
+ - - - - - - + - - - - - - +
              |             |
          (2) |             | (3)
              |             |
              + - - - - - - +
              D     (1)     E
```

- Assume A is destination, consider E

	Reported Dist.	Total Dist.
Neighbor C	3	6
Neighbor D	4	5

Summary: Distance Vector Routing

- Basic DV protocol
  - take away: use monotonicity as a technique to understand liveness/convergence
    - highly recommended reading of Bersekas/Gallager chapter

- Fix counting-to-infinity problem
  - DSDV
    - Idea: uses sequence number to avoid routing loops
      - seq# partitions routing updates from different outside events
      - within same event, no loop so long each node only decreases its distance
    - Analysis: use global invariants to understand/design safety/no routing loops
  - EIRGP (DUAL)
    - Idea: introduces a sufficient condition for local recovery

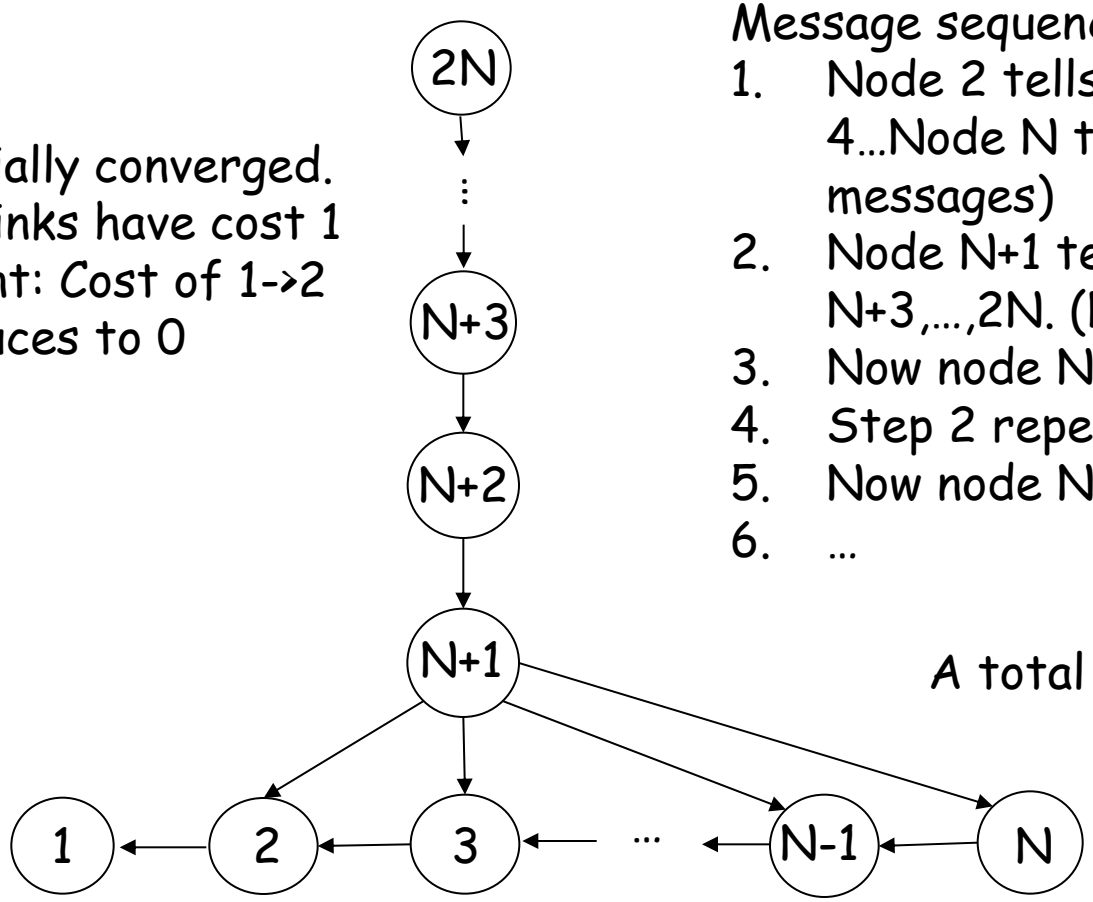
Discussion: Distance Vector Routing

- What do you like about distributed, distance vector routing?

- What do you not like about distributed, distance vector routing?

Churns of DV: One Example

Initially converged.
All links have cost 1
Event: Cost of 1->2 reduces to 0



Message sequences

1. Node 2 tells 3. Node 3 tells 4…Node N tells N+1. (N-1 messages)
2. Node N+1 tells N+2, N+2 tells N+3,…,2N. (N-1 messages)
3. Now node N-1 tells node N+1
4. Step 2 repeats
5. Now node N-2 tells node N+1
6. …

A total of $O(N^2)$ messages

Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Distance vector protocols (distributed computing)
      - *Link state protocols (distributed state synchronization)*

Link-State Routing

- Basic idea: Not distributed computing, only distributed state distribution
- Net topology, link costs are distributed to all nodes
  - all nodes have same info
  - Each node computes its shortest paths from itself to all other nodes
    - standard Dijkstra's algorithm as path compute alg
    - Allows multiple same-cost paths
    - Multiple cost metrics per link (for type of service routing)
- Most commonly used routing protocol (e.g., OSPF/ISIS) by most networks in Internet

Example: Link State and Directed Graph (OSPFv2)

```
                                        **FROM**

                                 *      |RT1|RT2|
     +---+Ia    +---+            *   ------------
     |RT1|------|RT2|            T   RT1|    | X |
     +---+    Ib+---+            O   RT2| X |    |
                                 *    Ia|    | X |
                                 *    Ib| X |    |

          Physical point-to-point networks
                                        **FROM**
    +---+      +---+
    |RT3|      |RT4|                |RT3|RT4|RT5|RT6|N2 |
    +---+      +---+          *  ------------------------
      |    N2    |             *  RT3|   |   |   |   | X |
 +----------------------+      T  RT4|   |   |   |   | X |
      |          |             O  RT5|   |   |   |   | X |
    +---+      +---+          *  RT6|   |   |   |   | X |
    |RT5|      |RT6|          *   N2| X | X | X | X |   |
    +---+      +---+

             Multi-access networks
                                            **FROM**
          +---+                  *
          |RT7|                  *       |RT7| N3|
          +---+                  T    ------------
            |                    O    RT7|   |   |
+-----------------------+        *     N3| X |   |
            N3                   *

           Stub multi-access networks
```

https://tools.ietf.org/html/rfc1583#page-12

Example: Link State and Directed Graph (OSPFv2)

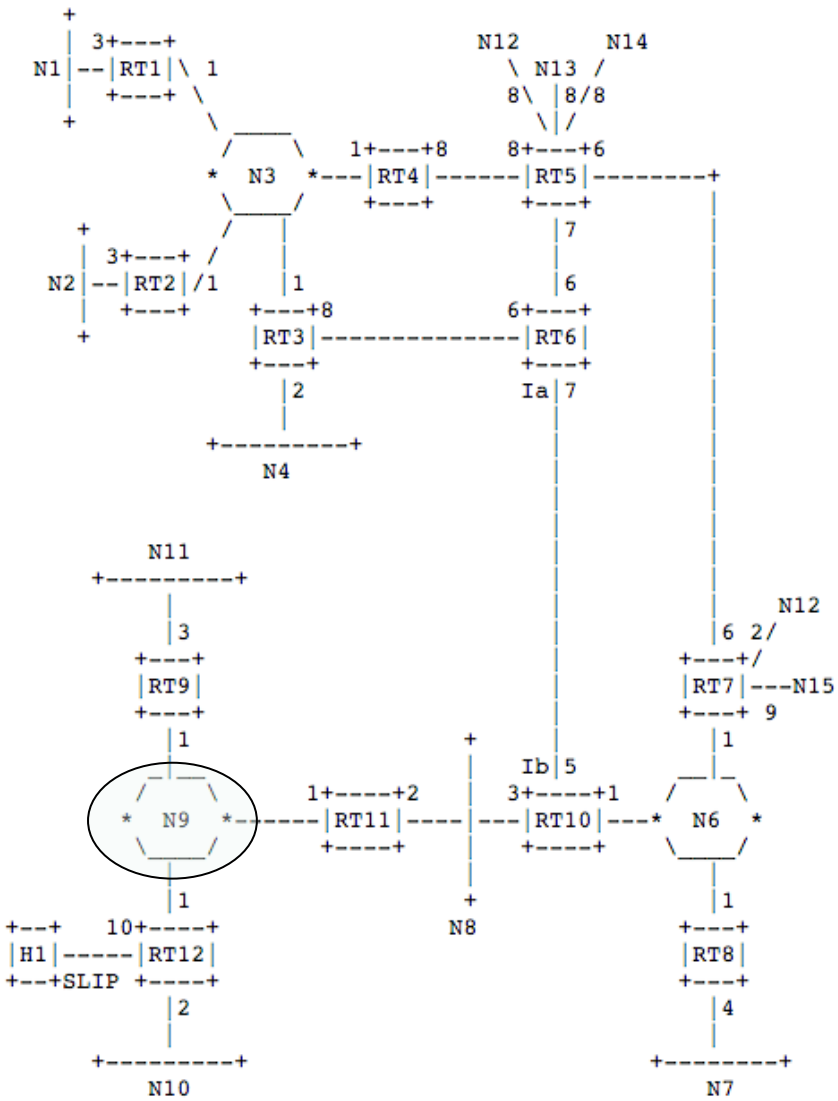


Figure 2: A sample Autonomous System

FROM

TO	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	N3	N6	N8	N9
RT1													0			
RT2													0			
RT3						6							0			
RT4					8								0			
RT5				8		6	6									
RT6			8		7					5						
RT7					6									0		
* RT8														0		
* RT9																0
T RT10						7								0	0	
O RT11															0	0
* RT12																0
* N1	3															
N2		3														
N3	1	1	1	1												
N4			2													
N6							1	1		1						
N7								4								
N8										3	2					
N9									1		1	1				
N10												2				
N11									3							
N12					8		2									
N13					8											
N14					8											
N15							9									
H1												10				

Figure 3: The resulting directed graph

Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Distance vector protocols (distributed computing)
      - *Link state protocols (distributed state synchronization)*
        - data structure to be distributed
        - *state distribution protocol*

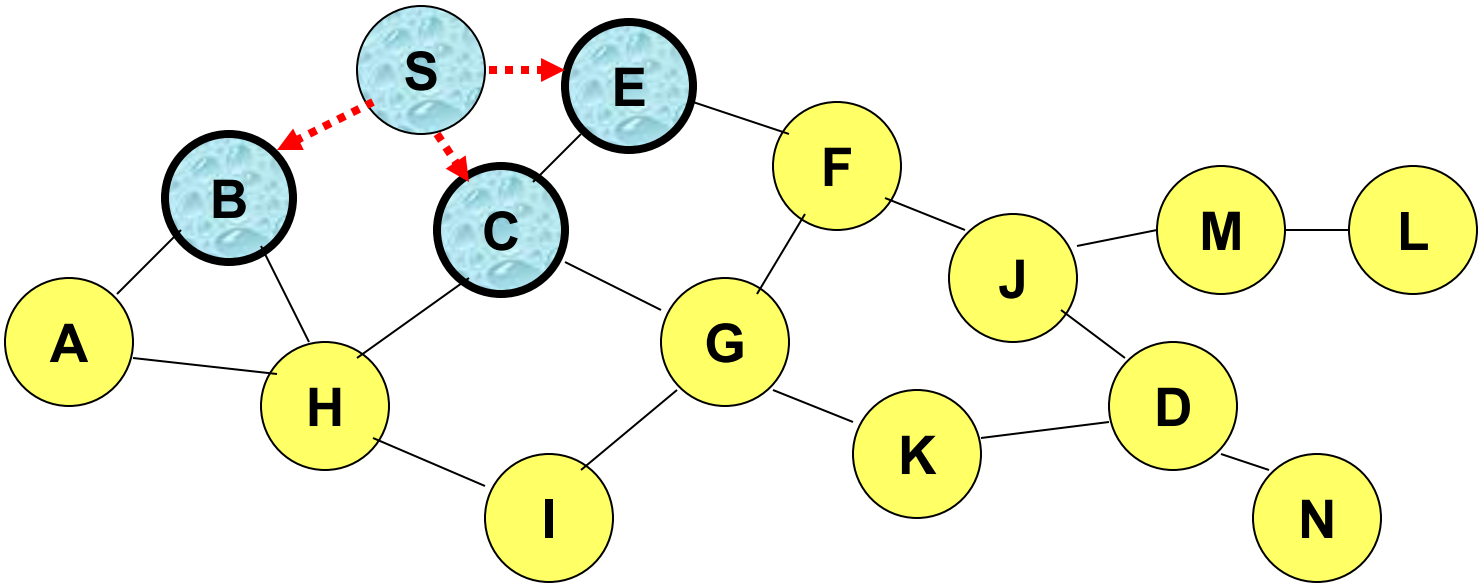
Basic Link State Broadcast Protocol

Basic event structure at node n

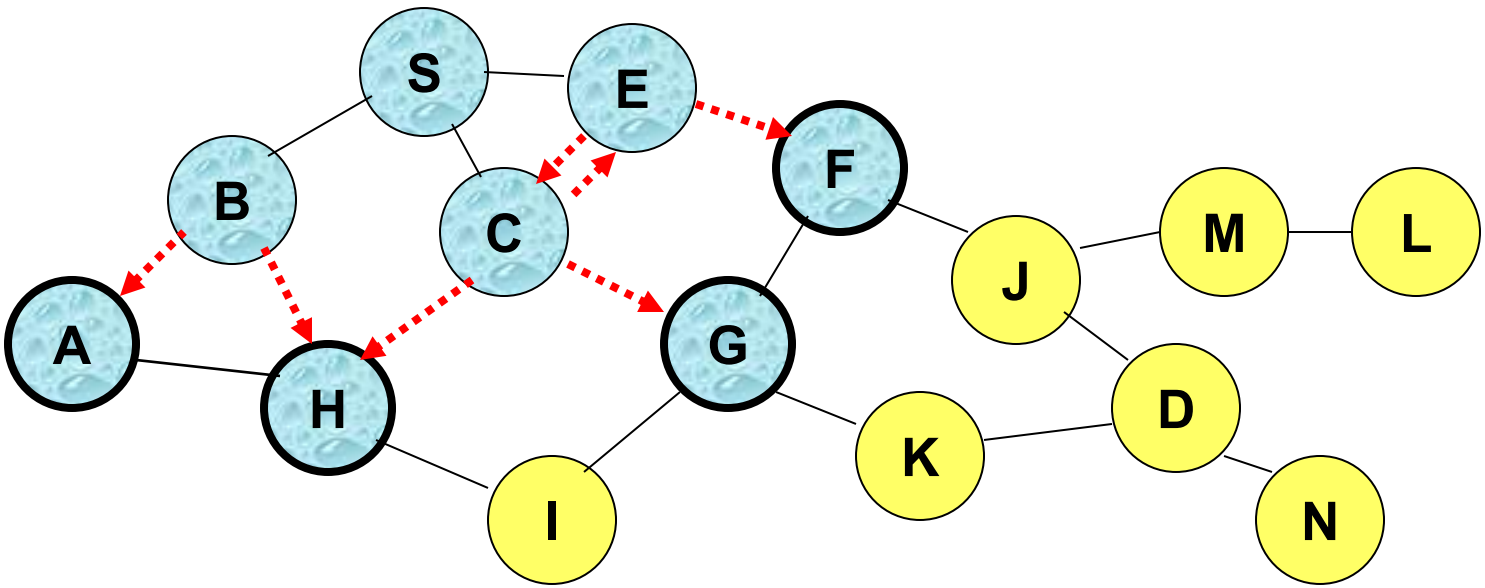
- on initialization:
  - broadcast LSA[e] for each link e connected to n
- on state change to a link e connected to n:
  - broadcast LSA[e] = new status
- on receiving an LSA[e]:
  - if (does not have LSA[e])
    forwards LSA[e] to all links except the incoming link

Link State Broadcast

Node S updates link states connected to it.



Link State Broadcast



To avoid forwarding the same link state announcement (LSA) multiple times (forming a loop), each node remembers the received LSAs.

- Second LSA[S] received by E from C is discarded
- Second LSA[S] received by C from E is discarded as well
- Node H receives LSA[S] from two neighbors, and will discard one of them

Discussion

- Issues of the basic link state protocol?
  - Recall: goal is to efficiently distribute to each node to a correct, complete link state map

Link State Broadcast: Issues

- Problem: Out of order delivery
  - link down and then up
  - A node may receive up first and then down

- Solution
  - Each link update is given a sequence number: (initiator, seq#, link, status)
    - the initiator should increase the seq# for each new update
  - If the seq# of an update of a link is not higher than the highest seq# a router has seen, drop the update
  - Otherwise, forward it to all links except the incoming link (real implementation using packet buffer)

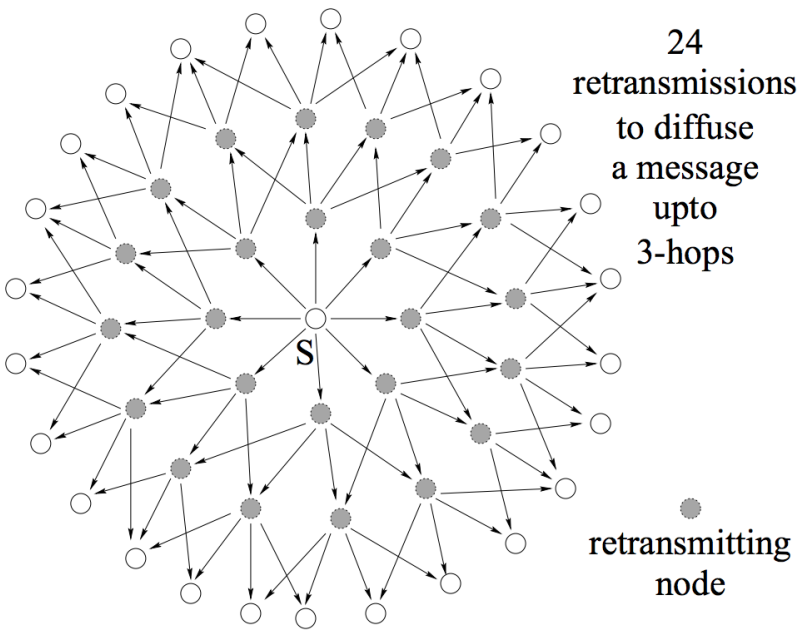
  - Problem of solution: seq# corruption
  - Solution: age field (e.g., https://tools.ietf.org/html/rfc1583#page-102)

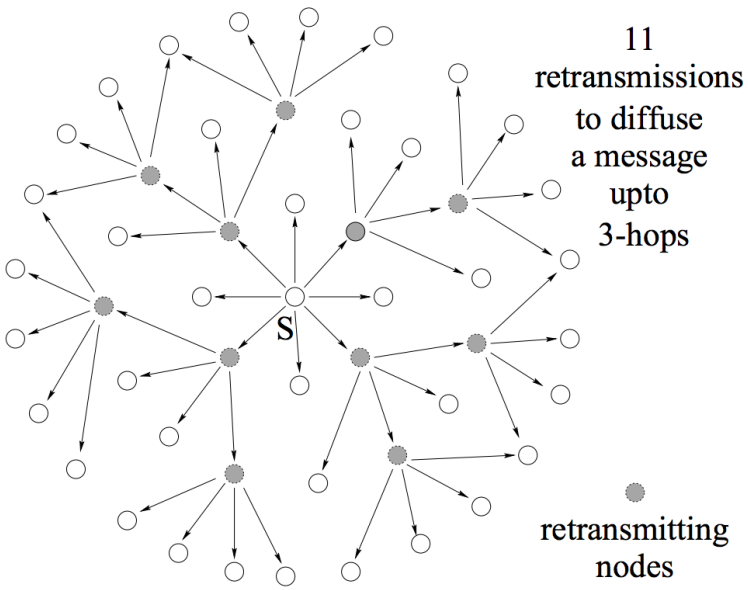
Link State Broadcast: Issues

- Problem: network partition and then reconnect, how to sync across the reconnected components
- Solution: updates are sent periodically

Link State Broadcast: Issues

- Problem: Broadcast redundancy

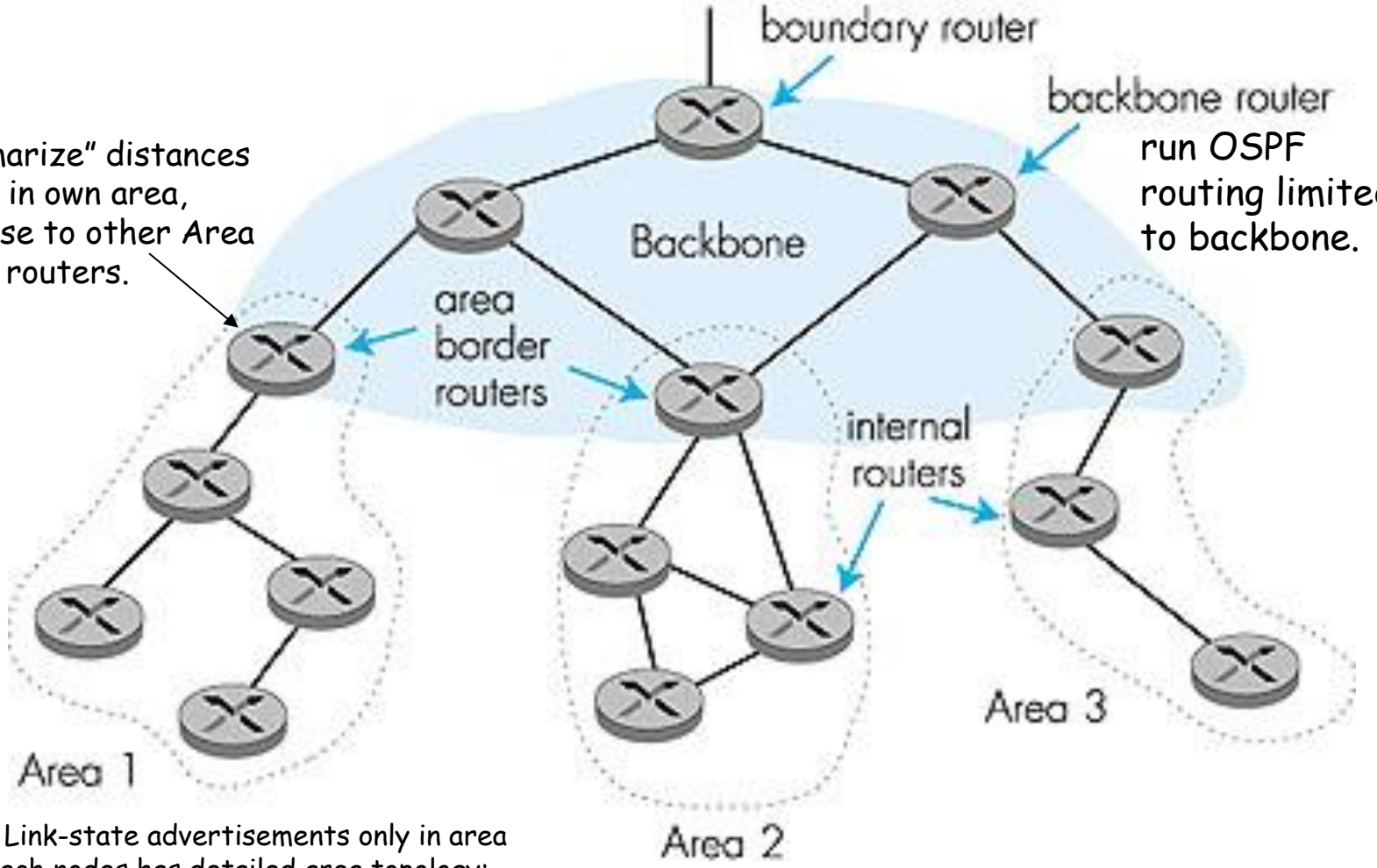




https://hal.inria.fr/inria-00072756/document

Hierarchical OSPF

"summarize" distances to nets in own area, advertise to other Area Border routers.



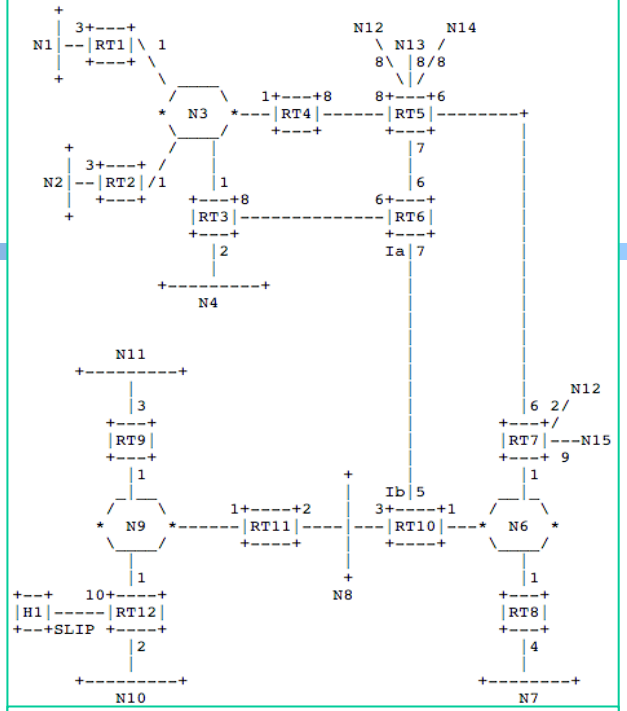
run OSPF routing limited to backbone.

- Link-state advertisements only in area each nodes has detailed area topology;
- only know direction (shortest path) to nets in other areas.

Two-level hierarchy: local area, backbone.

Summary: Link State

- Basic LS protocol
  - take away: instead of computing routing results using distributed computing, distributed computing is for only link state distribution (synchronization)
- Link state distribution can still have much complexity, e.g., out of order delivery, partition and reconnect, scalability

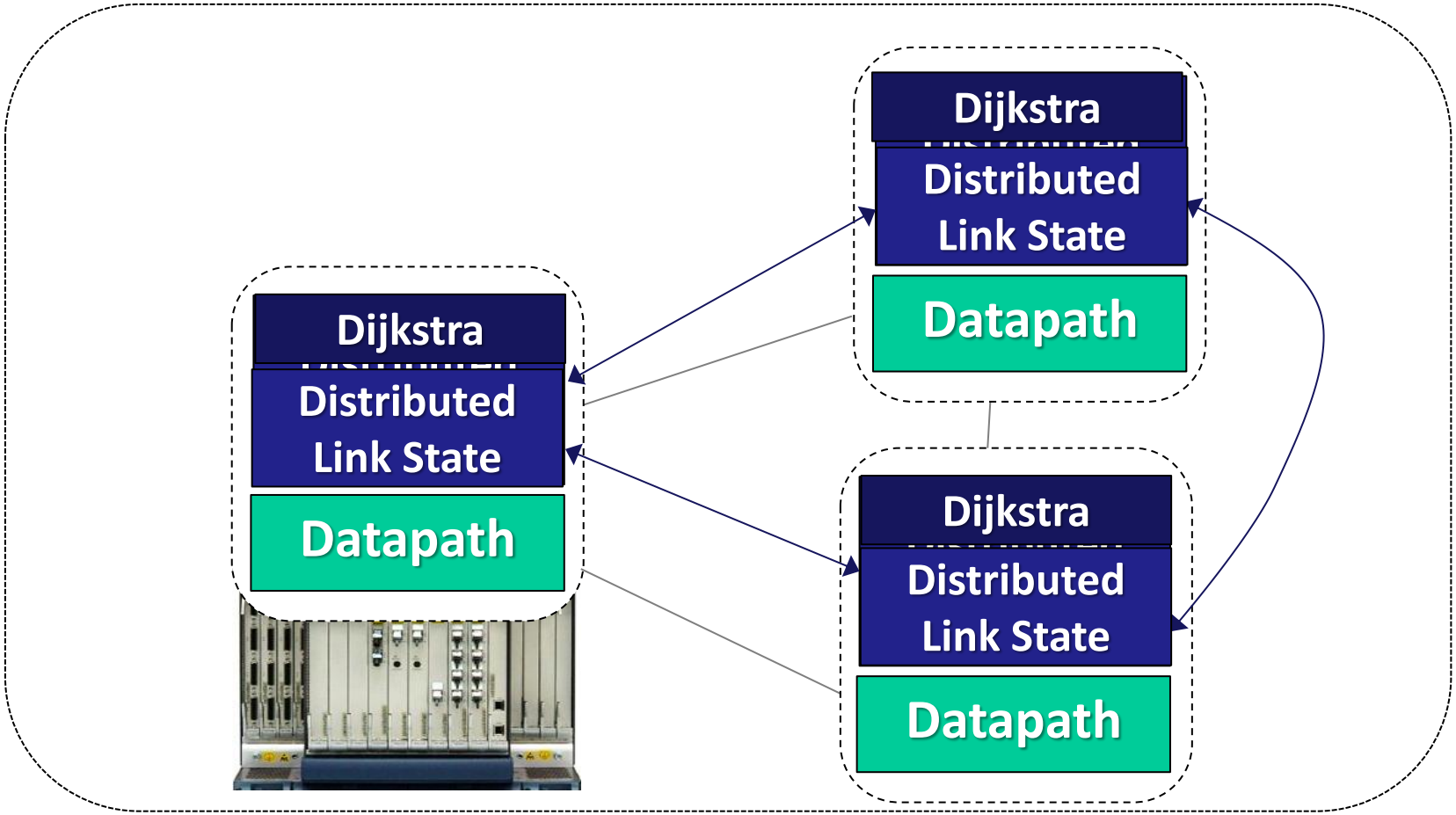


FROM

		RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	N3	N6	N8	N9
	RT1													0			
	RT2													0			
	RT3						6							0			
	RT4					8								0			
	RT5				8		6	6									
	RT6			8		7					5						
	RT7					6									0		
*	RT8														0		
*	RT9																0
T	RT10						7								0	0	
O	RT11															0	0
*	RT12																0
*	N1	3															
	N2		3														
	N3	1	1	1	1												
	N4			2													
	N6							1	1		1						
	N7								4								
	N8										3	2					
	N9									1		1	1				
	N10												2				
	N11									3							
	N12					8		2									
	N13					8											
	N14					8											
	N15							9									
	H1												10				

Figure 3: The resulting directed graph

Roadmap: Routing Computation Architecture Spectrum

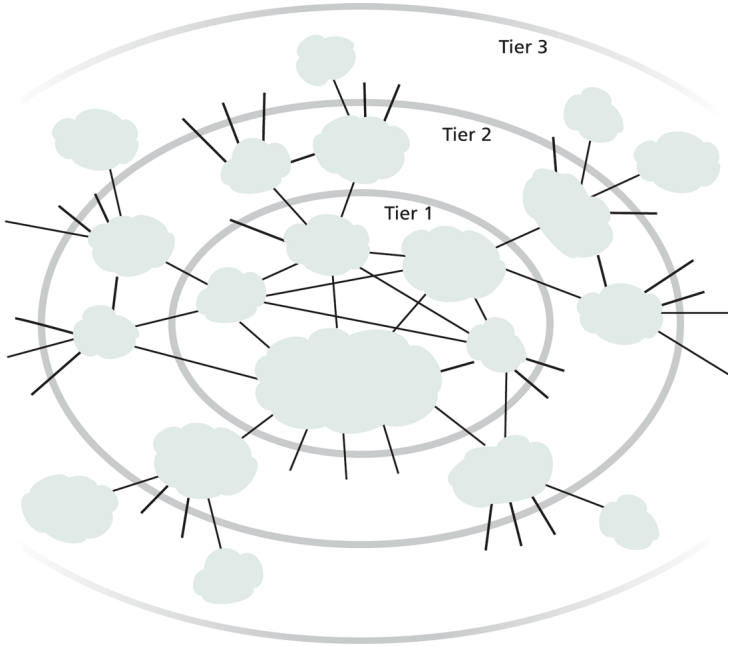


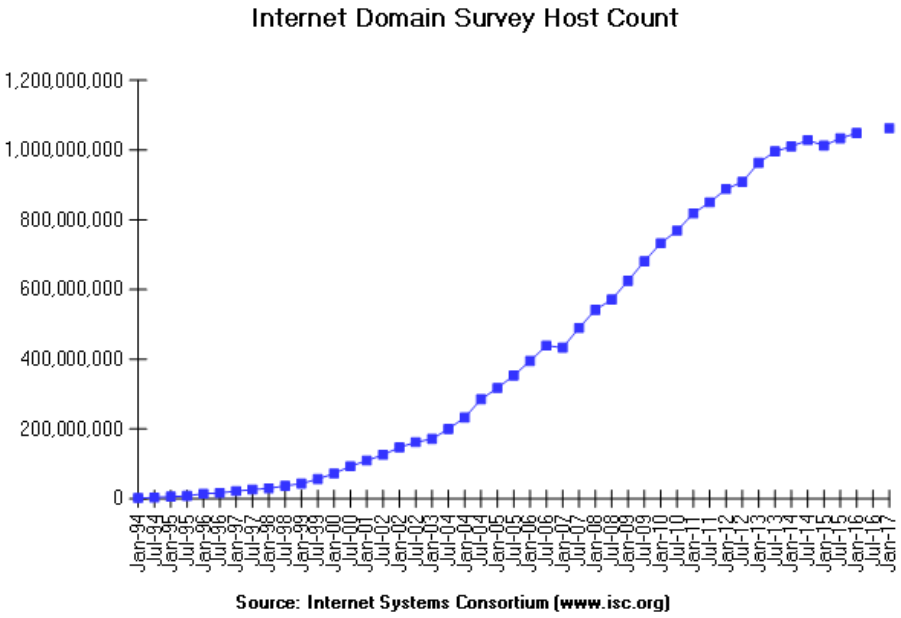
Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Basic routing computation protocols
        - Distance vector protocols (distributed computing)
        - Link state protocols (distributed state synchronization)
      - Global Internet routing

Exercise

- Does it work to use DV or LS as we discussed for global Internet routing?





Requirements and Solution of Current Global Internet Routing

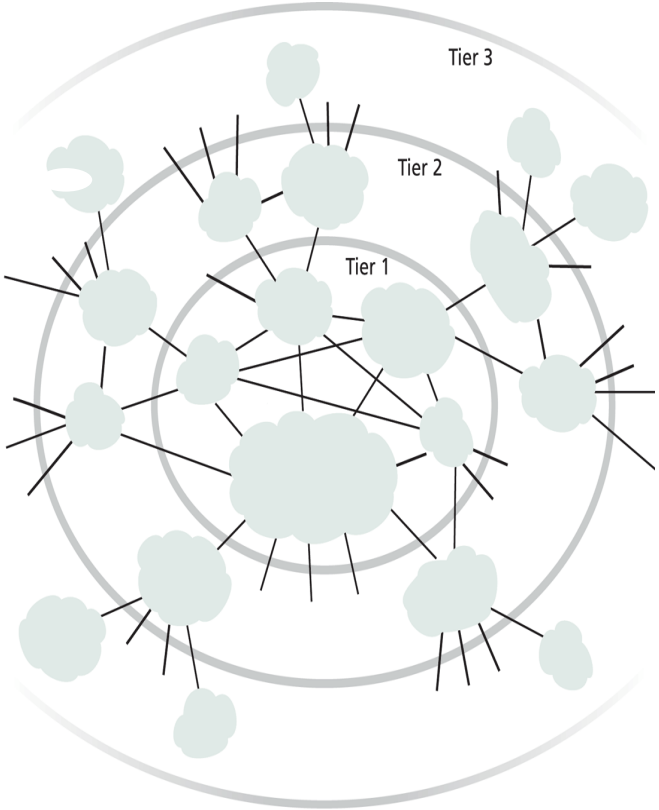
- Scalability: handle network size (#devices) much higher than typical DV or LS can handle
  - Solution: Introduce new abstraction to reduce network (graph) size

- Autonomy: allow each network to have individual preference of routing (full control of its internal routing; control/preference of routing spanning multiple networks)
  - Solution: hierarchical routing and policy routing

New Abstraction: Autonomous Systems (AS)

- Abstract each network as an autonomous system (AS), identified by an AS number (ASN)

- Conceptually the global routing graph consists of only autonomous systems as nodes



http://www.bgplookingglass.com/list-of-autonomous-system-numbers

Routing with Autonomous Systems

- 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