Network Layer:

Distance Vector Routing, Link State Routing
Global Internet Routing (Interdomain, BGP)

Qiao Xiang, Congming Gao

https://sngroup.org.cn/courses/cnns-xmuf23/index.shtml

12/07/2023

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 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

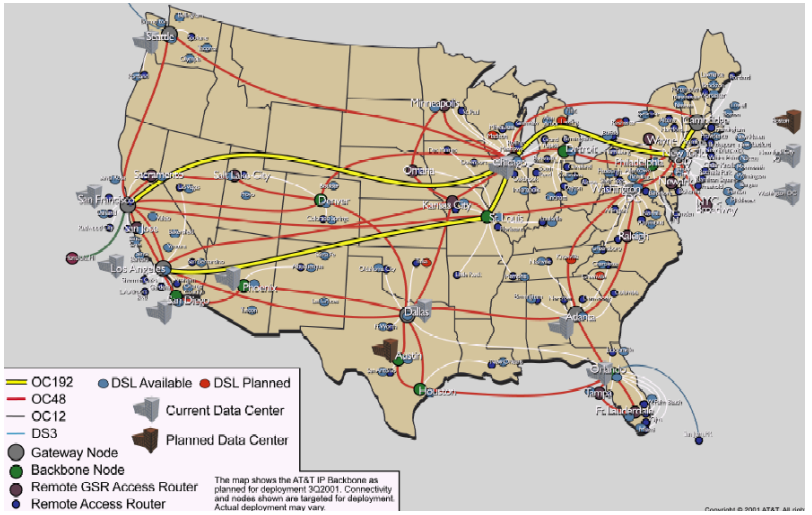
Admin.

- Lab 4 and 5 to be posted by the end of the day
  - Lab 4: Transport layer, implementing reliability data transfer, due on Jan. 7
  - Lab 5: Network layer, questions on routing and forwarding, due on Dec. 24
- Class project: extend lab 4 to implement flow control and congestion control
  - Due on Jan. 21.

Recap: Routing Context

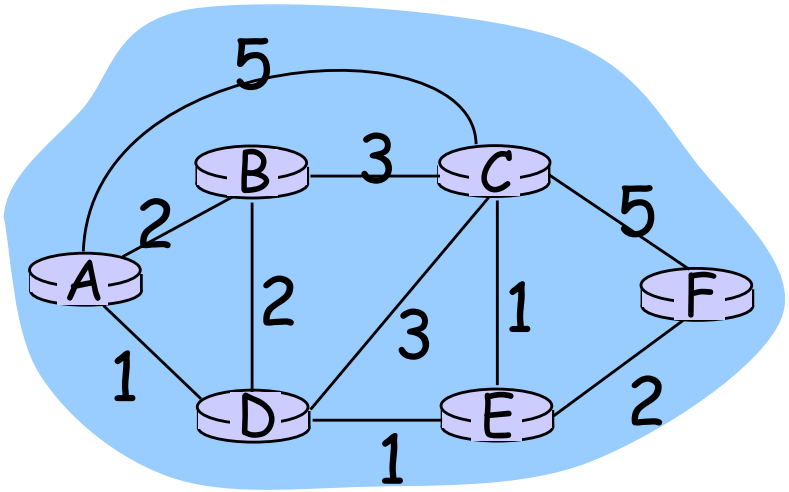
Routing

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



Often depends on a graph abstraction:

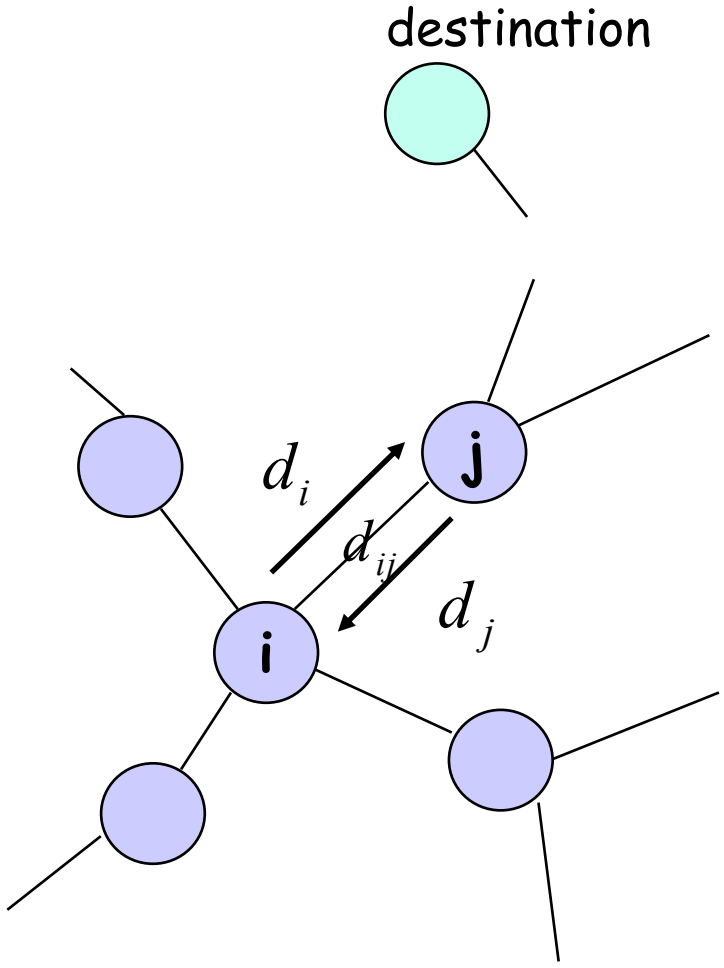
- graph nodes are routers
- graph edges are physical links
  - links have properties: delay, capacity, $ cost, policy



Recap: Routing Computation using Distance Vector/Bellman-Ford Routing

- Distributed computation:
  At node i, computes

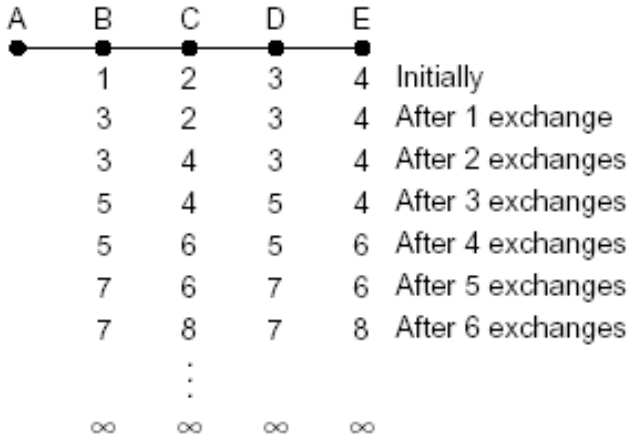
$$d_i = \min_{j \in N(i)} (d_{ij} + d_j)$$



- One way to understand BFA is to consider it as a dynamic programming alg, propagating from dest to other nodes

Recap: Fixing DV/BFA

- Property of BFA
  - Bad news may propagate slowly due to loops



- Techniques
  - Reverse poison
    - Avoid two-node loops
  - DSDV
    - Using destination seq to partition into epochs
    - A good example of analysis using global invariants
  - Diffusive Update Alg (DUAL)
    - Utilize backup routes

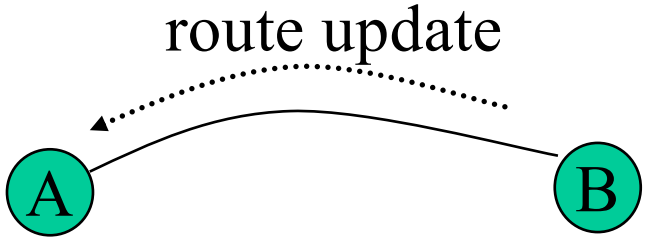
Outline

- 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)

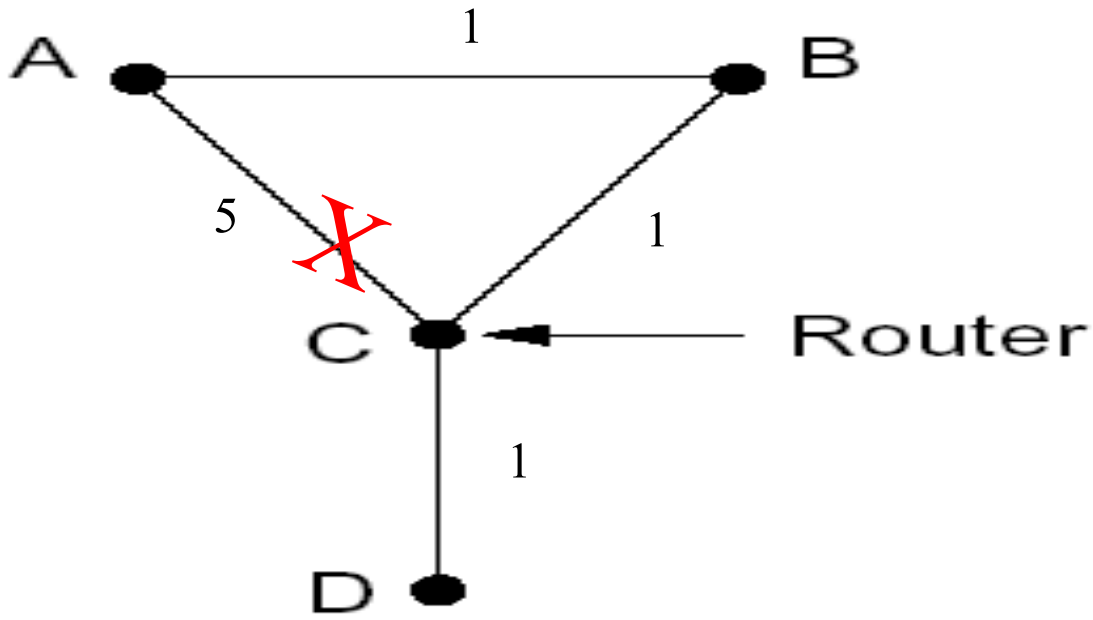
- Basic idea: use sequence numbers to partition computation
  - tags each route with a sequence number
  - each destination node D periodically advertises monotonically increasing even-numbered sequence numbers
  - when a node realizes that the link it uses to reach destination D is broken, it advertises 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



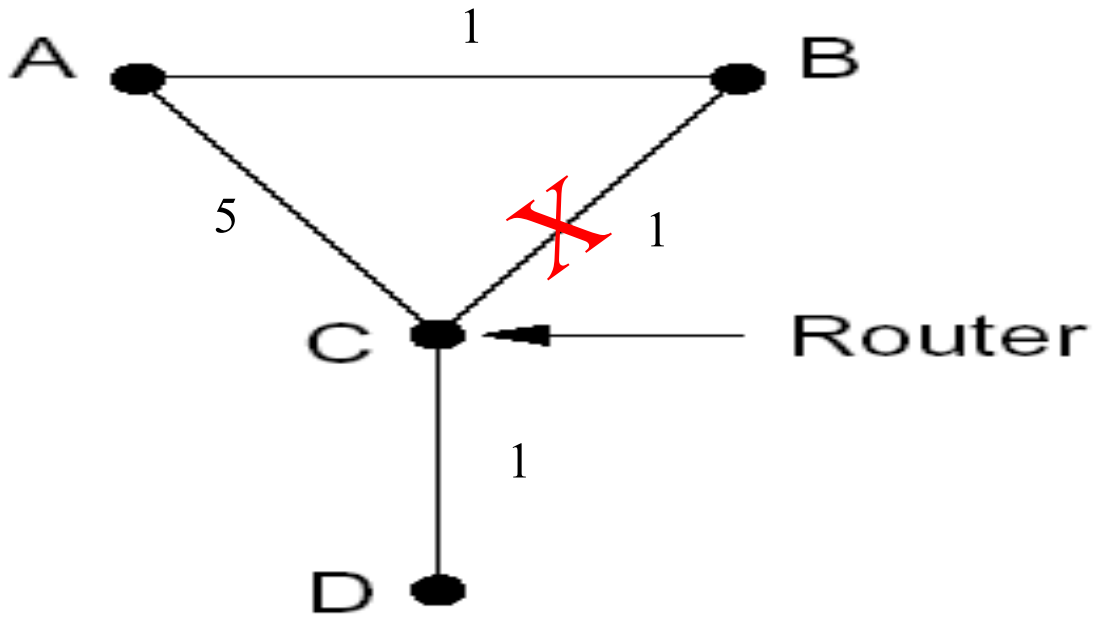
- Let's assume the destination node is D
- There are optimizations but we present a simple version:
  - each node B maintains (S^B, d^B), where S^B is the sequence number at B for destination D and d^B is the best distance using a neighbor from B to D
- Both periodical and triggered updates
  - periodically: 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



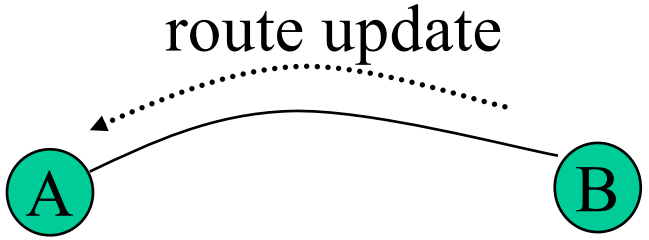
Will this trigger an update?

Example



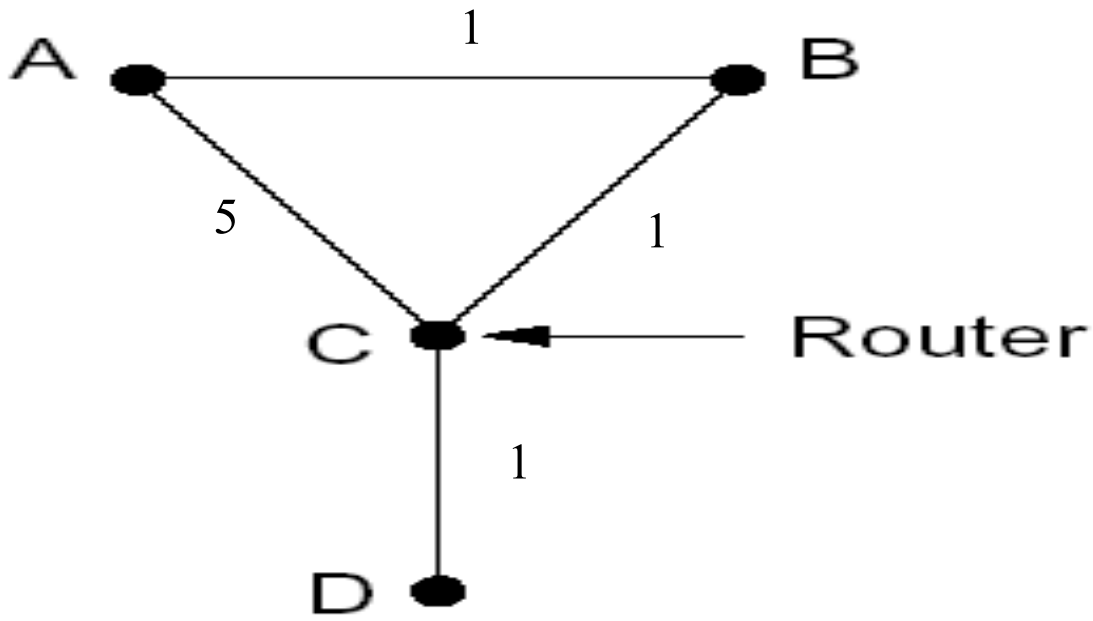
Will this trigger an update?

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

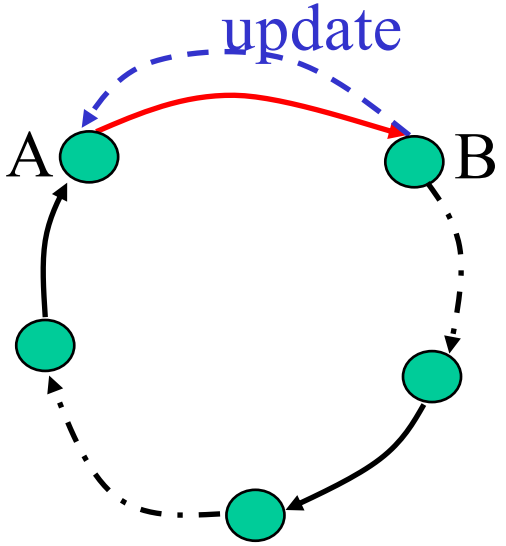
Example



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



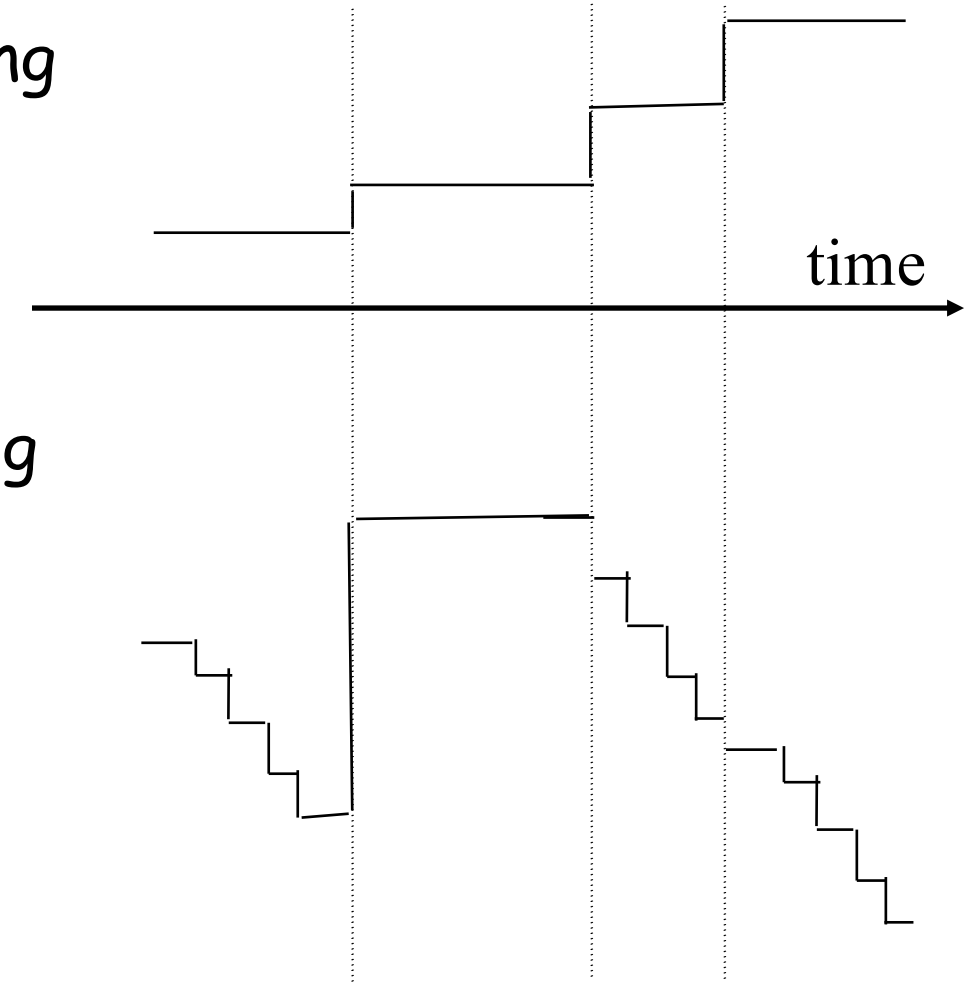
Technique: Global Invariants

- 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).

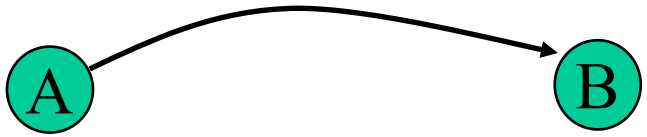
Invariants of a Single Node B

- 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



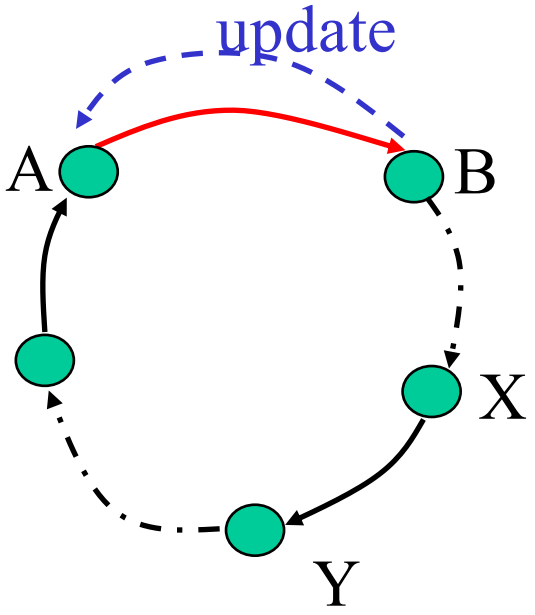
  - [I3] d^A is not ∞
  - [I4] $S^B >= S^A$

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

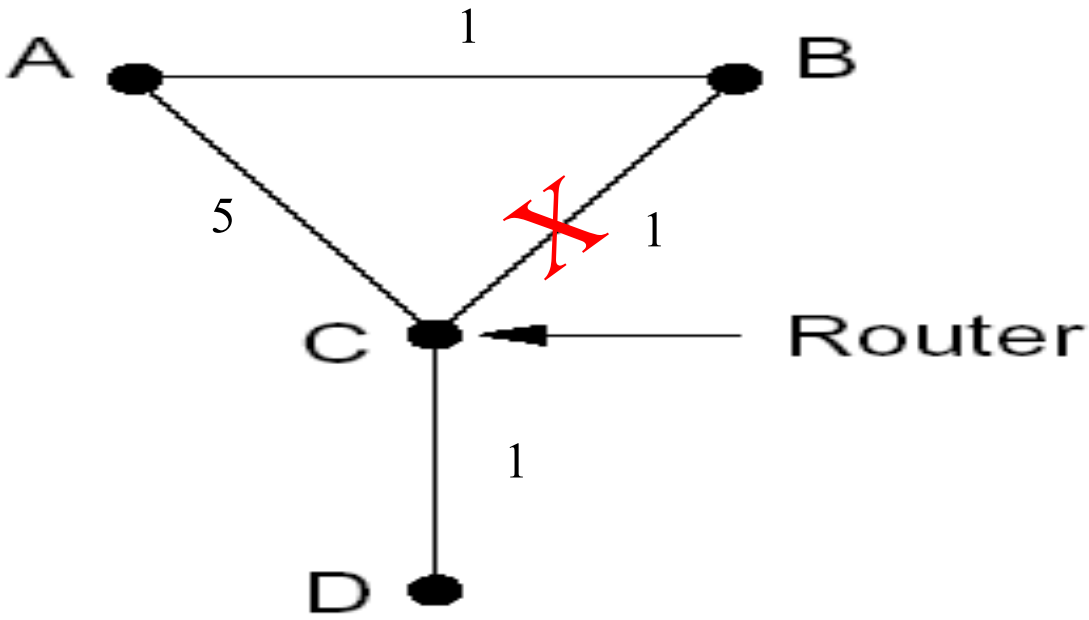
    - [I5] if $S^B == S^A$
    - then $d^B < d^A$ because d^A is based on d^B which B sent to A some time ago, $d^B < d^A$ since all link costs are positive; d^B might be decreased after B sent its state

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



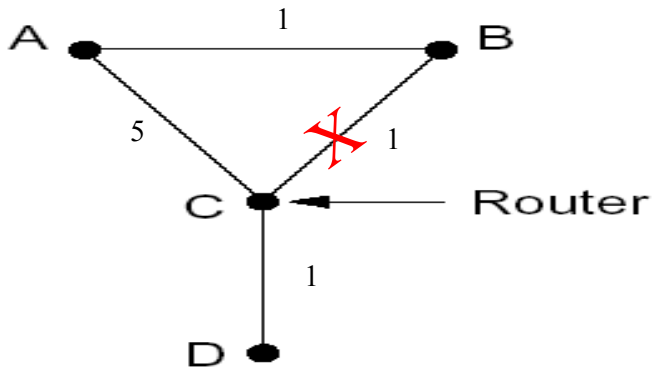
Outline

- 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

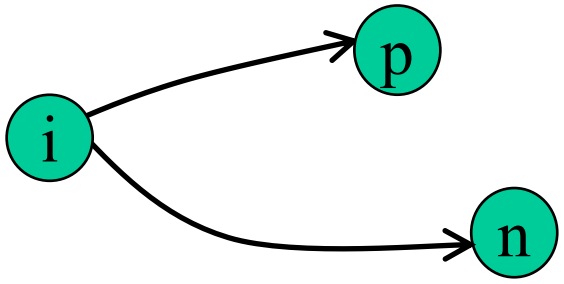
- DSDV guarantees no loop, but at the price of not using any backup path before destination re-announces reachability.

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



Key Idea: Feasible Successors

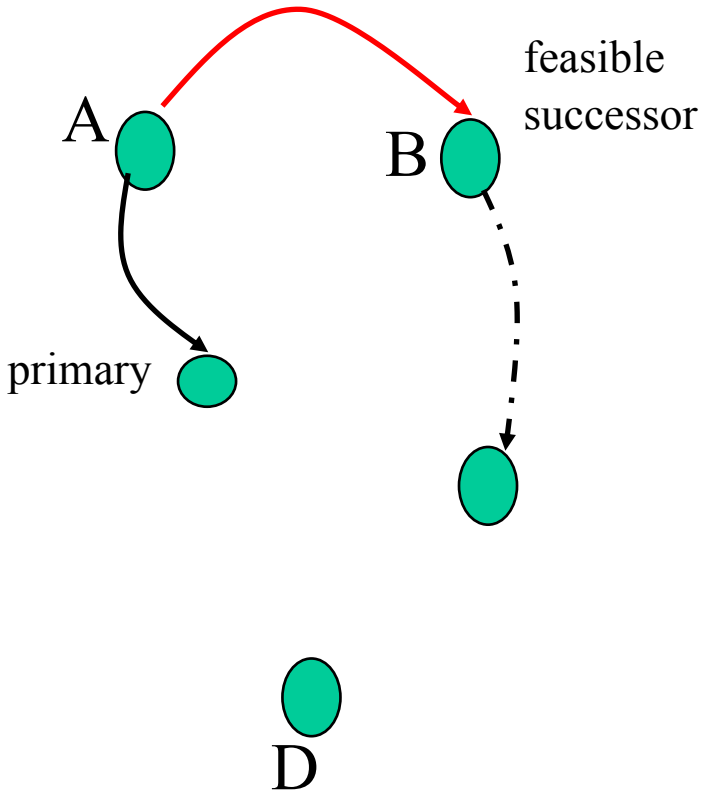
- If the reported distance of a neighbor n is lower than the total distance using primary (current shortest), the neighbor n is a feasible successor



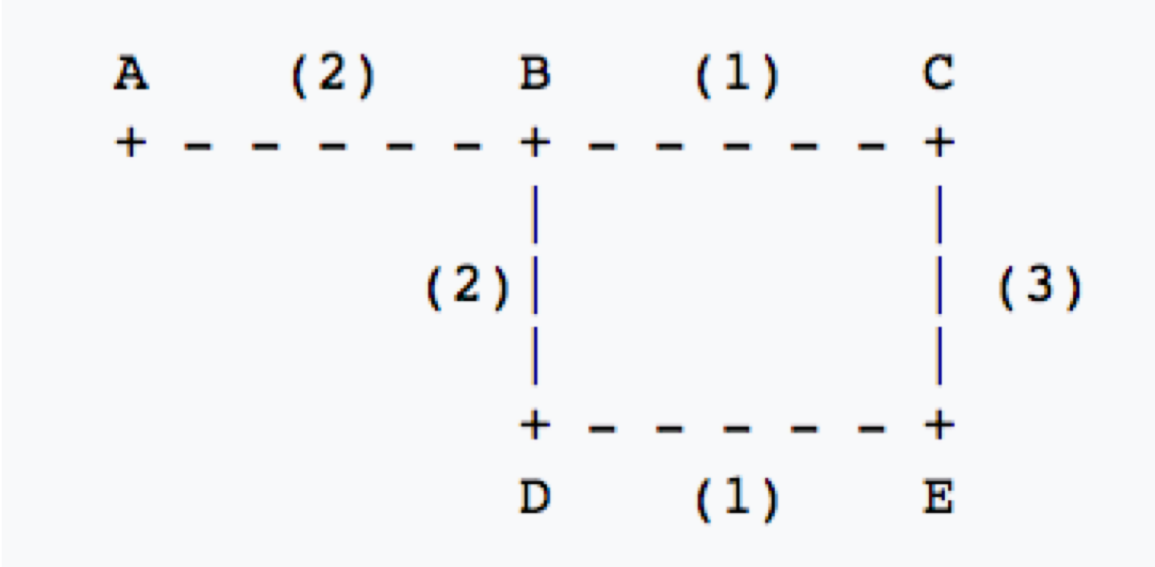
$$d_n + d_{i \to \text{n}} \geq d_{\text{primary}} + d_{i \to \text{primary}} > d_{\text{n}}$$

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



- 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

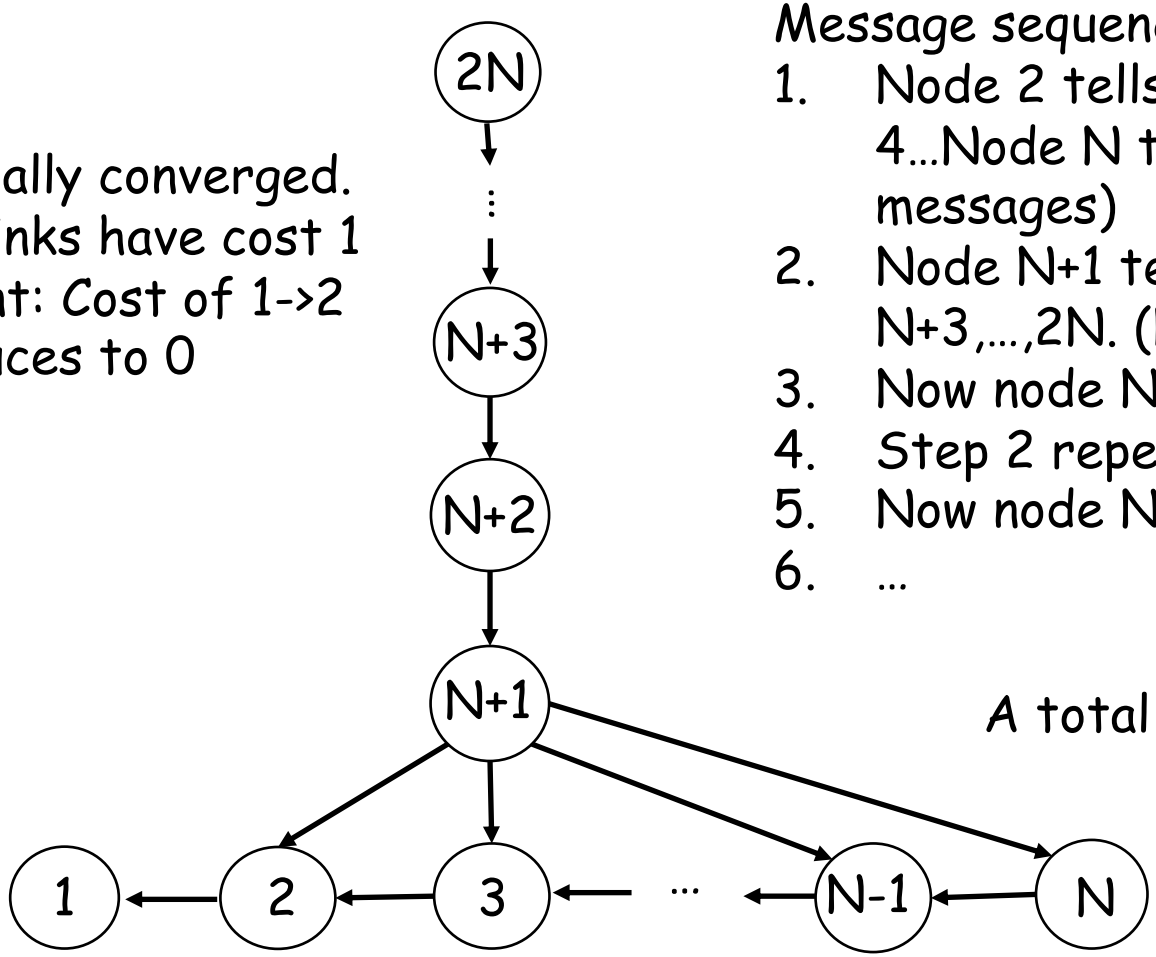
Discussion: Distance Vector Routing

- 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

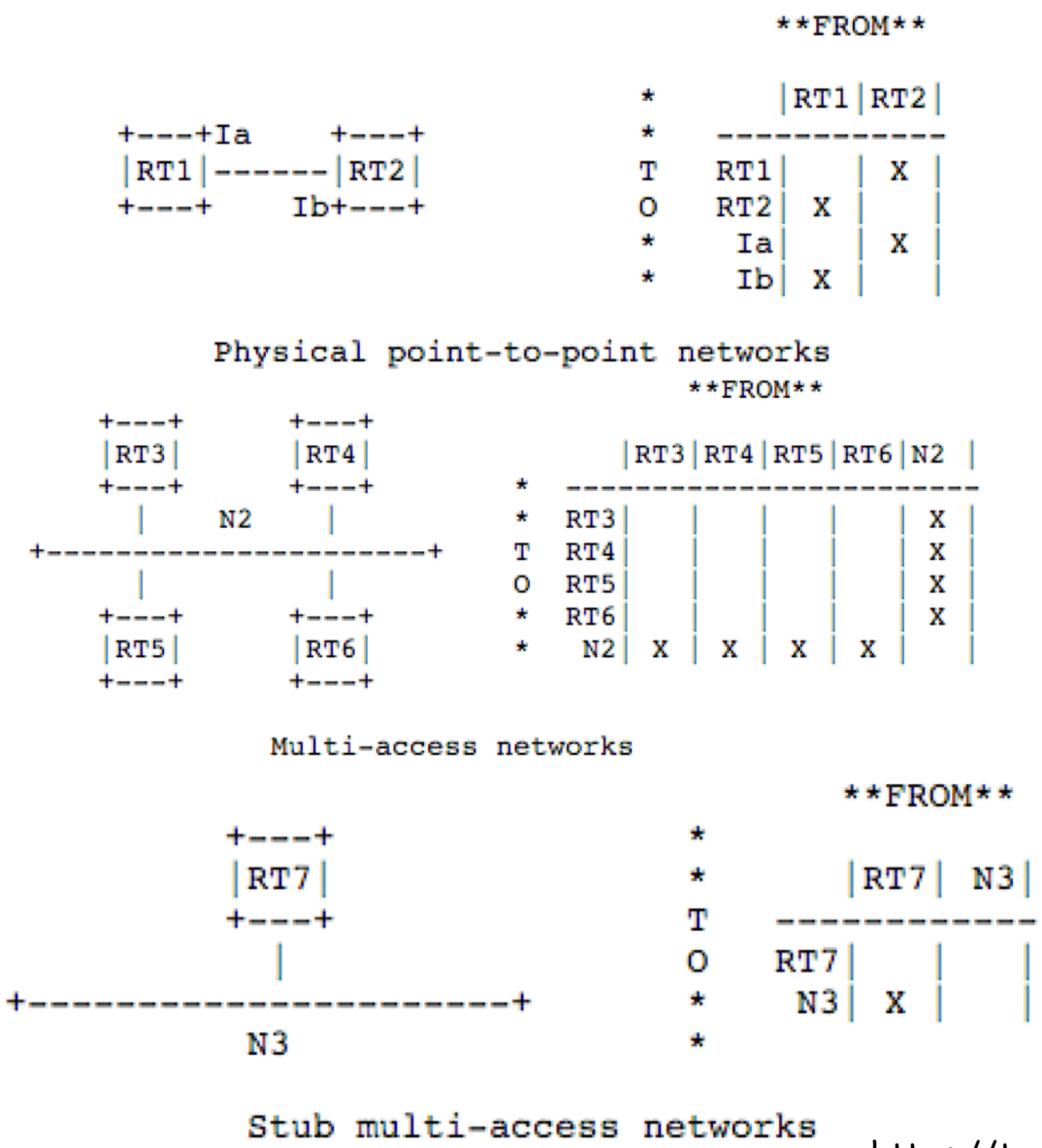
Outline

- 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)



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

Example: Link State and Directed Graph (OSPFv2)

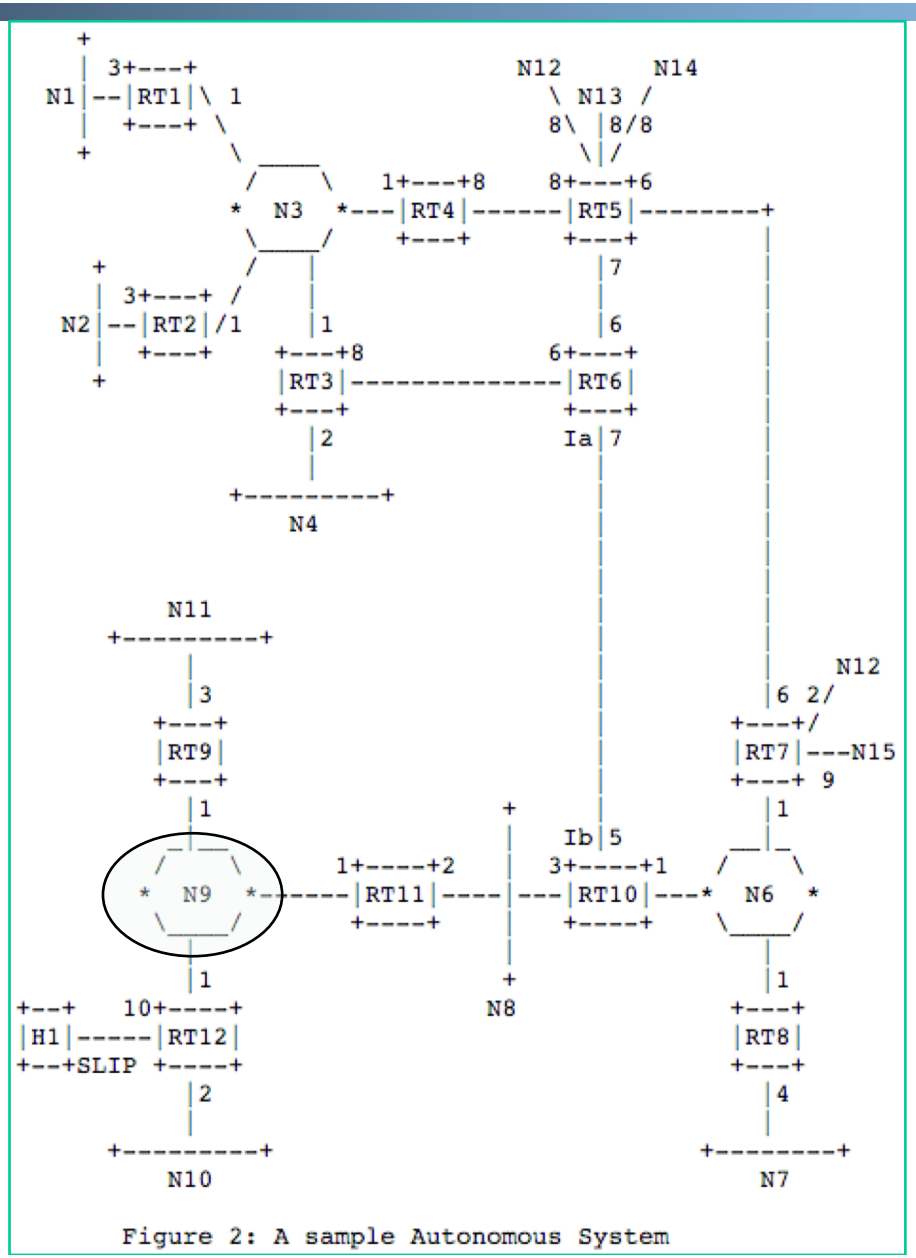


Figure 2: A sample Autonomous System

FROM

		RT 1	RT 2	RT 3	RT 4	RT 5	RT 6	RT 7	RT 8	RT 9	RT 10	RT 11	RT 12	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*

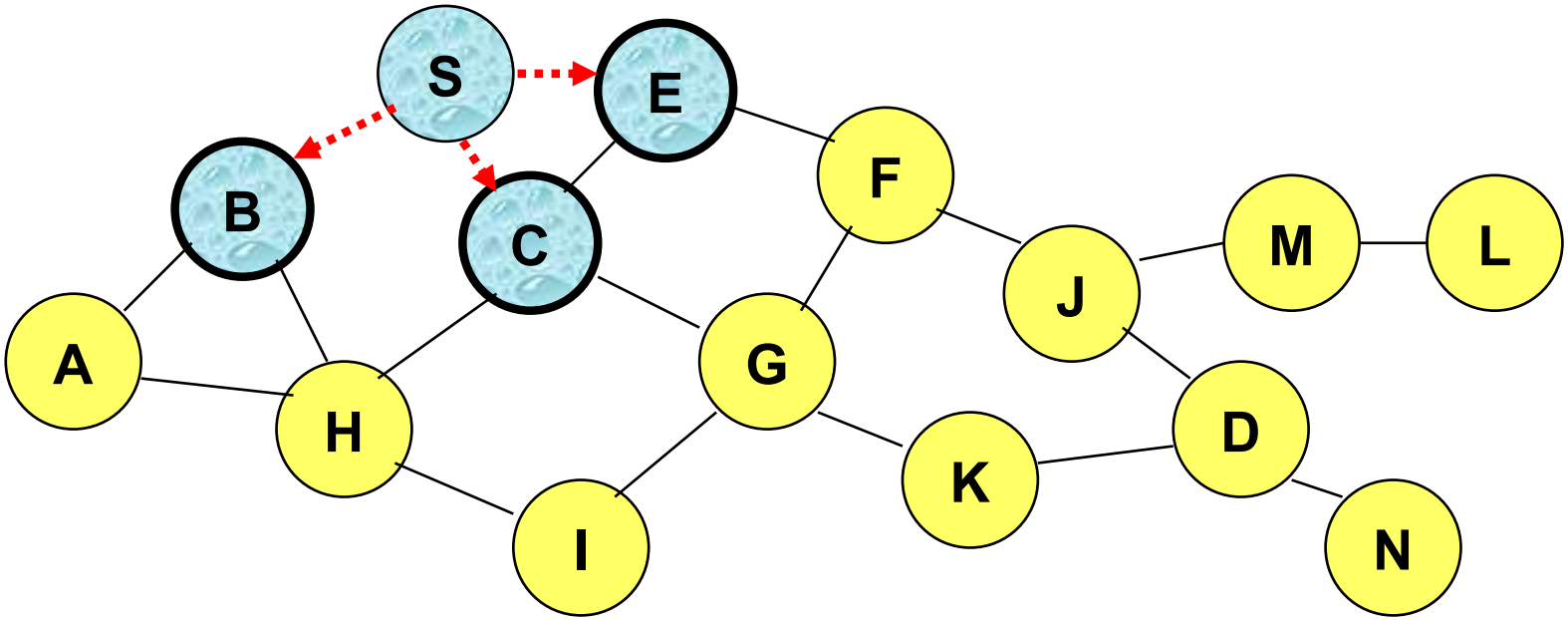
Basic Link State Broadcast Protocol

Basic event structure at node n

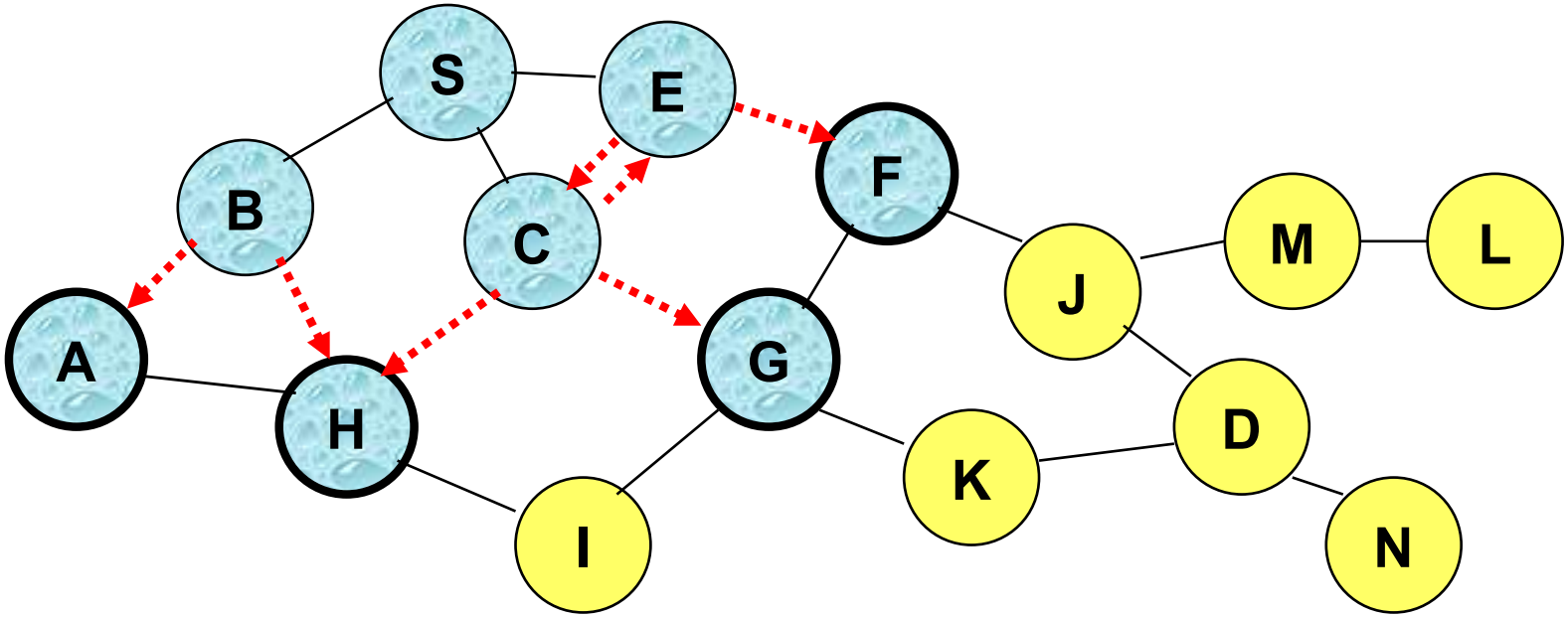
- on initialization:
  - broadcast LSA[e] for each link e connected to n
- on state change to a link e connected to n:
  - broadcast LSA[e] = new status
- on receiving an LSA[e]:
  - if (does not have LSA[e])
    forwards LSA[e] to all links except the incoming link

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)

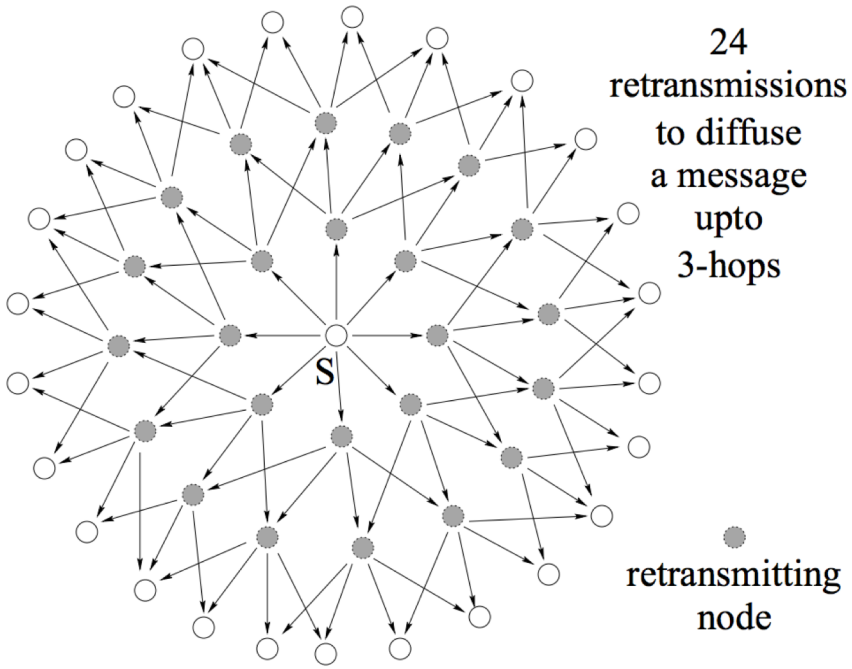
  - Problem of solution: seq# corruption
  - Solution: age field (e.g., https://tools.ietf.org/html/rfc1583#page-102)

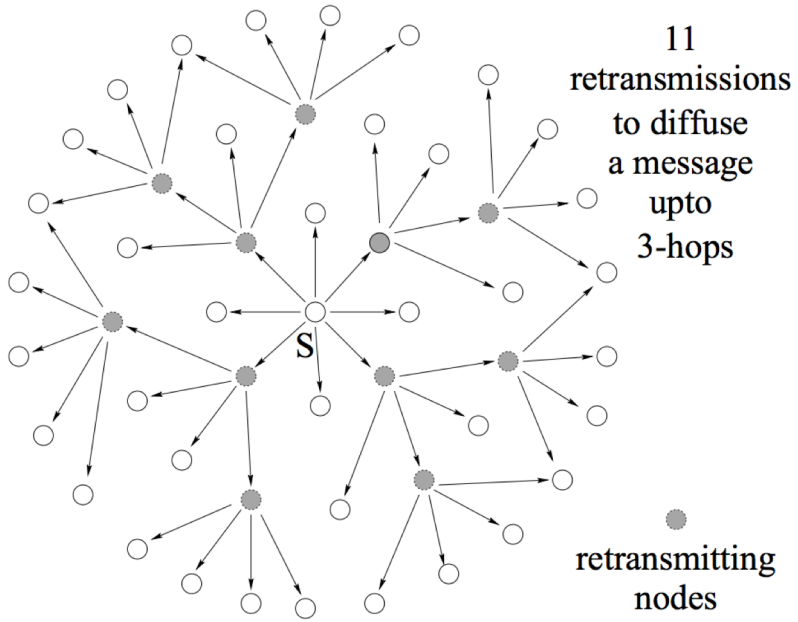
Link State Broadcast: Issues

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

Link State Broadcast: Issues

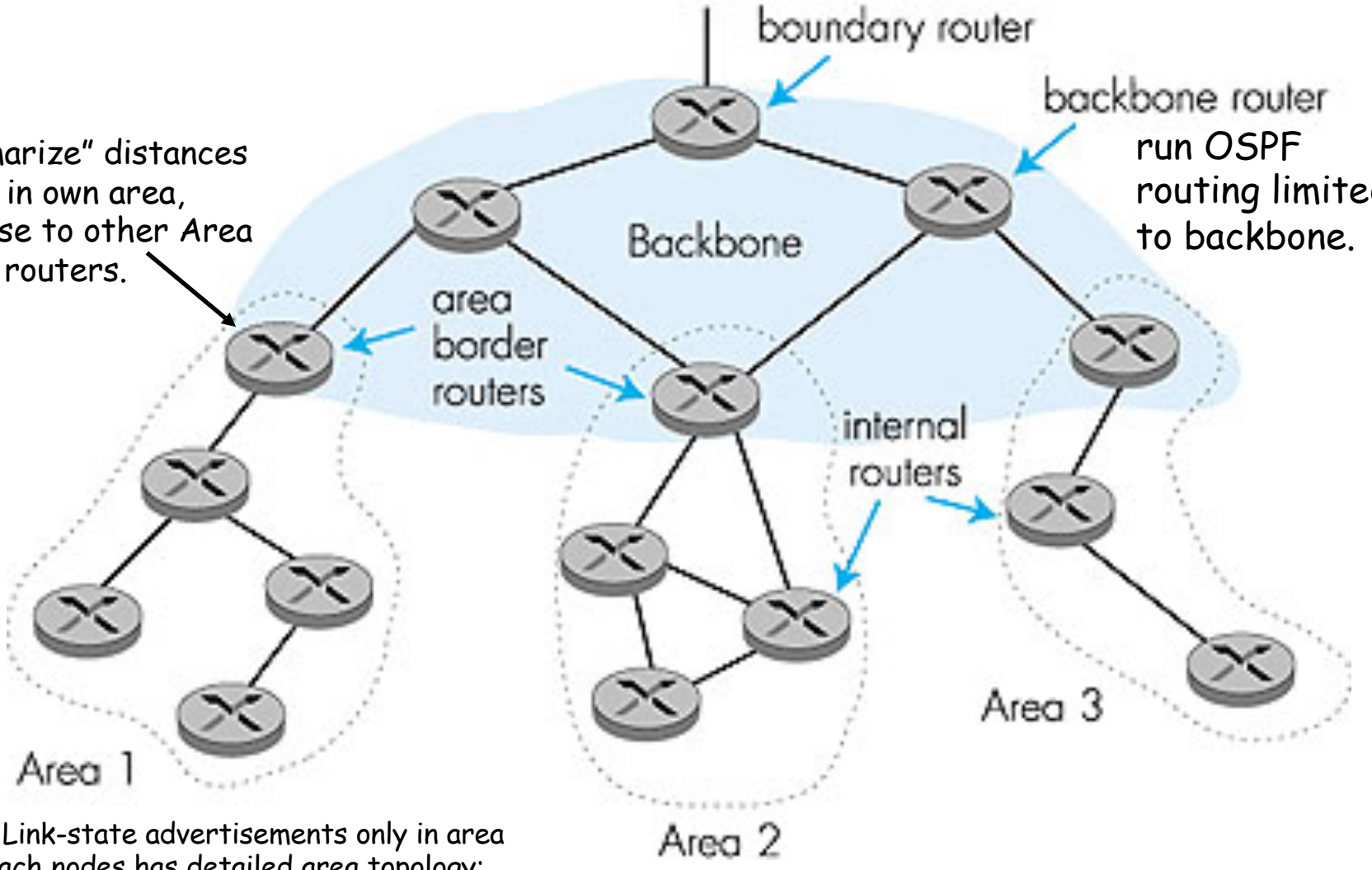
- Problem: Broadcast redundancy





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

Hierarchical OSPF

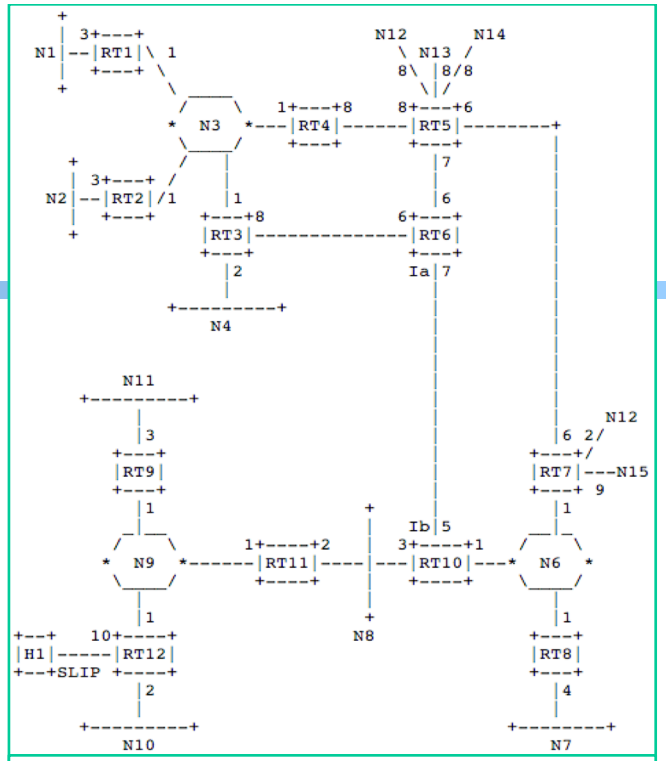


- Link-state advertisements only in area each nodes has detailed area topology;
- only know direction (shortest path) to nets in other areas.

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

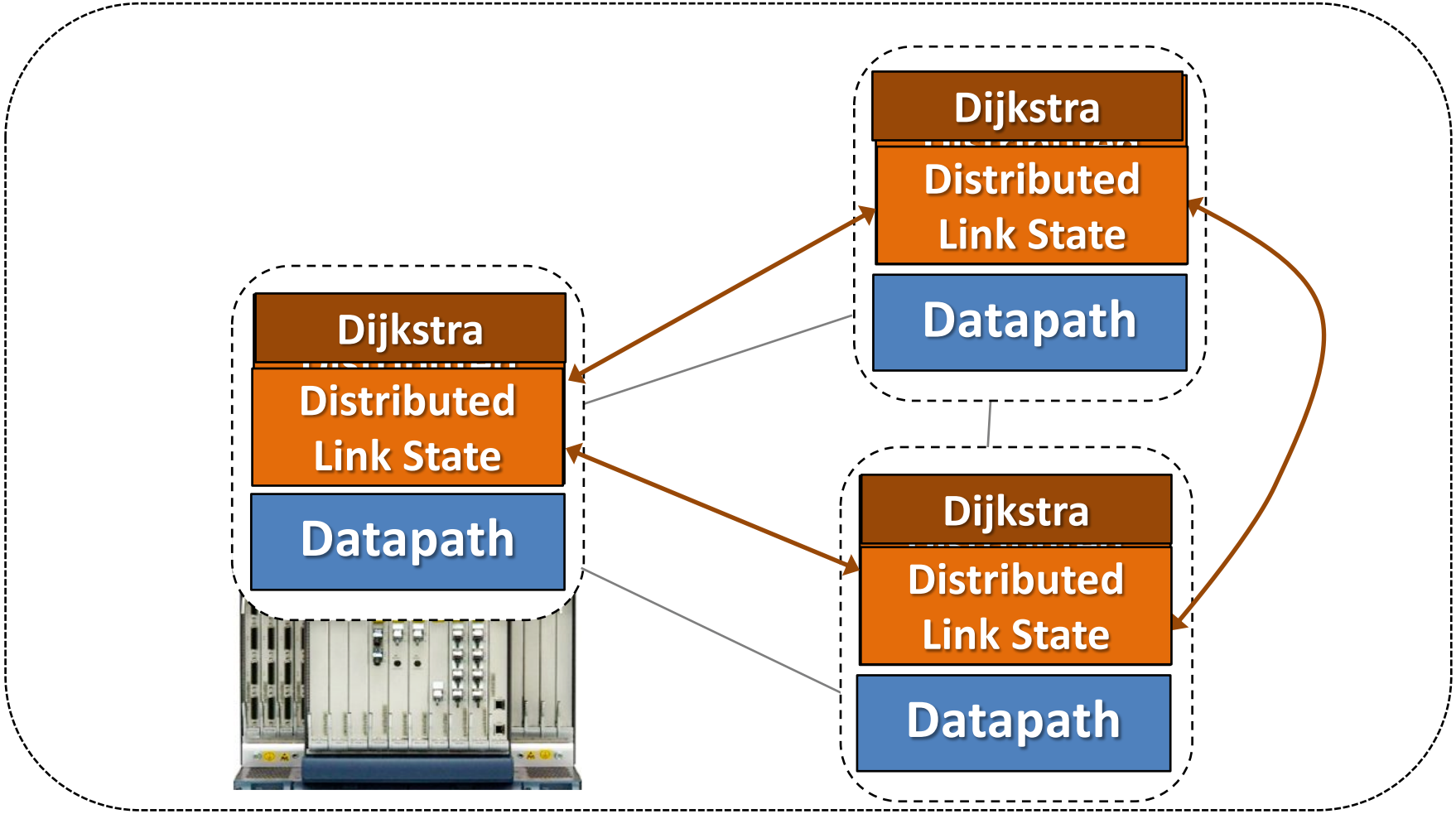


FROM

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

Roadmap: Routing Computation Architecture Spectrum

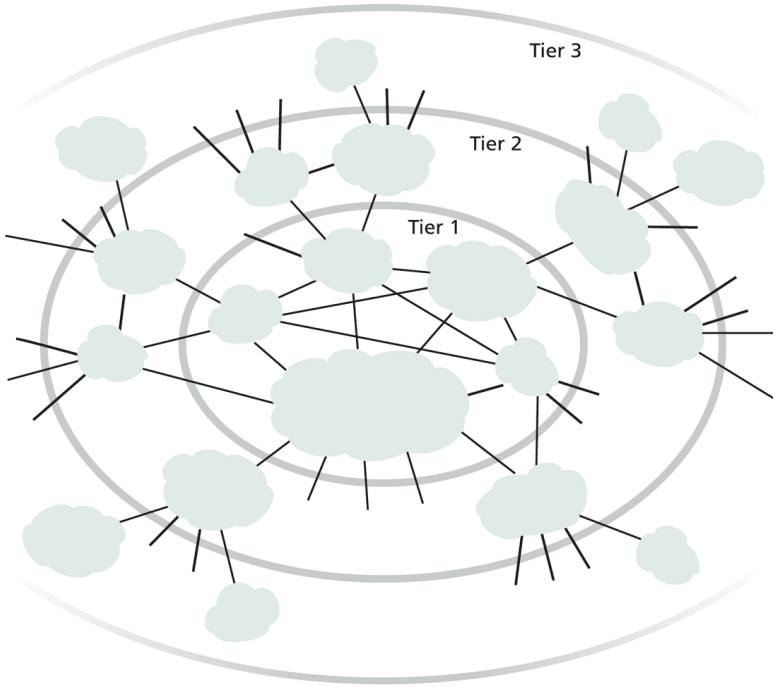


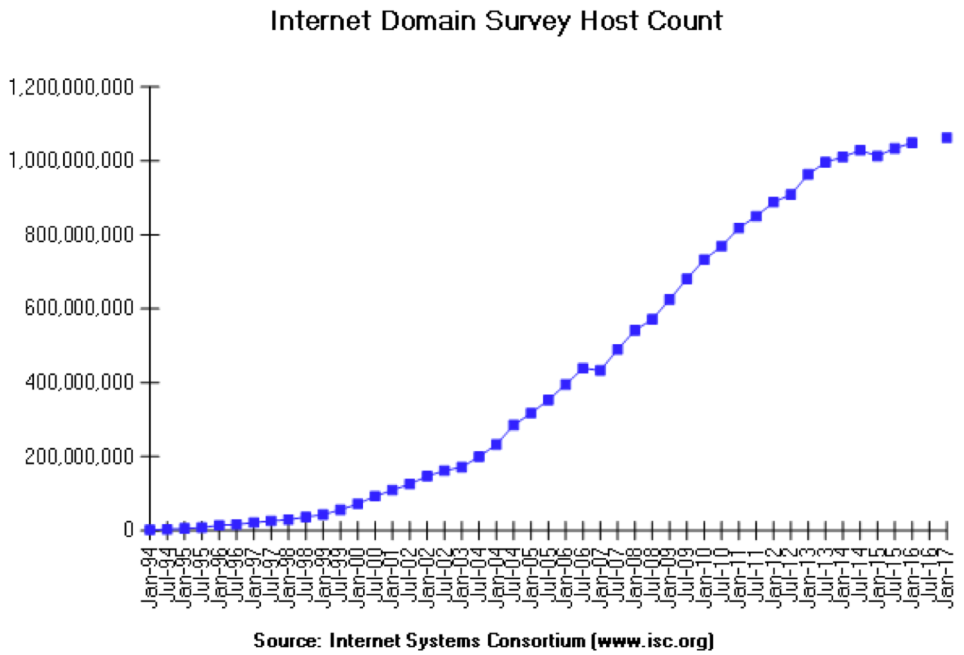
Outline

- 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

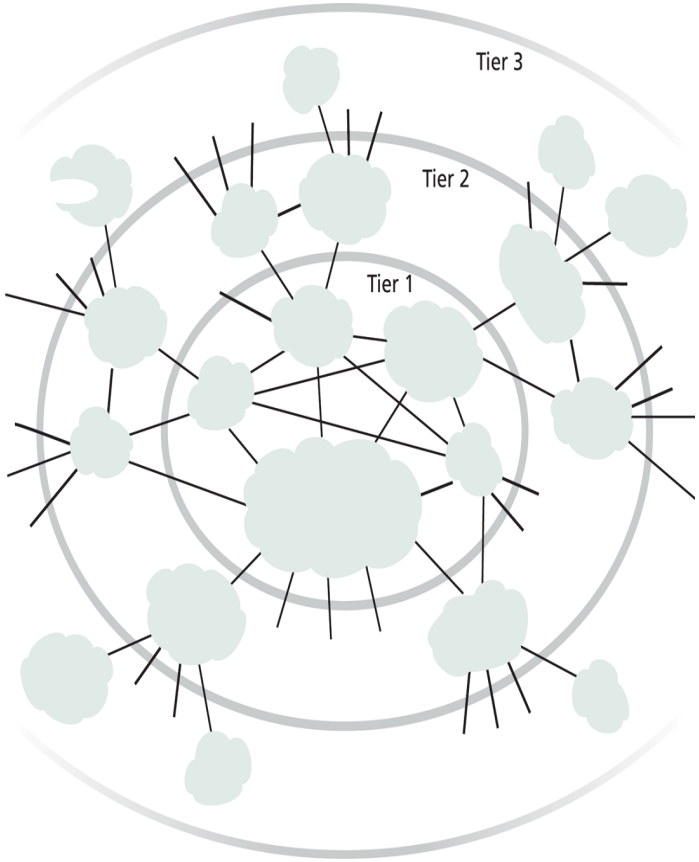
- Scalability: handle network size (#devices) much higher than typical DV or LS can handle
  - Solution: Introduce new abstraction to reduce network (graph) size

- 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

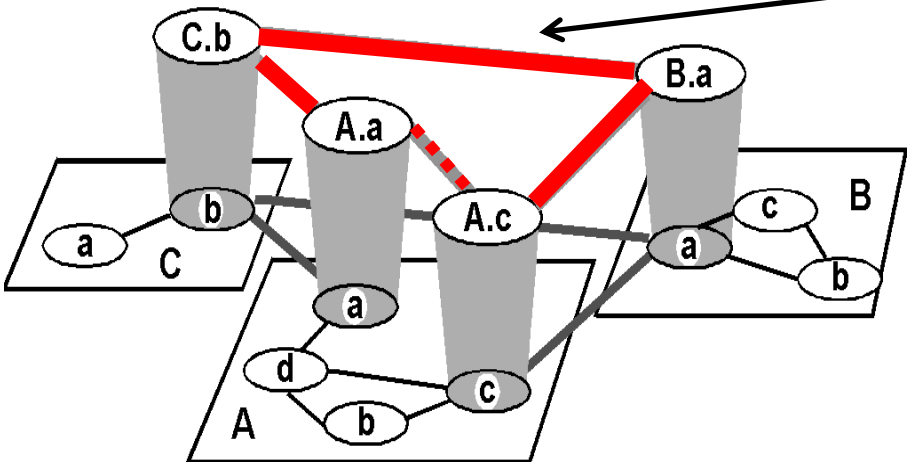


http://www.bgplookingglass.com/list-of-autonomous-system-numbers

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

Routing with Autonomous Systems



Inter-AS routers form an overlay

Gateway routers of same AS share learned external routes using iBGP.

Gateway routers participate in intradomain to learn internal routes.

AS C (RIP intra routing)

AS A (OSPF intra routing)

AS B (OSPF intra routing)

a

b

e

eBGP

iBGP

d

c

f

g

i

h

Gateway routers of diff. auto. systems exchange routes using eBGP

Summary: Internet Routing Architecture

- Autonomous systems have flexibility to choose their own intradomain routing protocols
  - allows autonomy
- Only a small # of routers (gateways) from each AS in the interdomain level
  - improves scalability
- Interdomain routing using AS topology instead of detailed topology
  - improves scalability/privacy

Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Basic routing computation protocols
      - Global Internet routing
        - Basic architecture
        - *BGP (Border Gateway Protocol): The de facto Inter-domain routing standard*

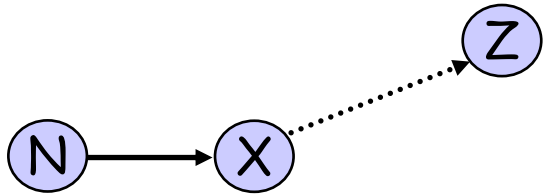
BGP Basic Operations

- BGP is **a Path Vector** protocol
  - similar to Distance Vector protocol
  - a border gateway sends to a neighbor *entire path* (i.e., a sequence of ASNs) to a destination, e.g.,
    - gateway X sends to neighbor N its path to dest. Z:

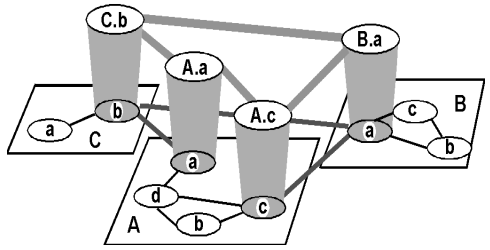
$$\text{path } (X,Z) = X,Y1,Y2,Y3,\ldots,Z$$

  - if N selects path(X, Z) advertised by X, then:

$$\text{path } (N,Z) = N, \text{ path } (X,Z)$$



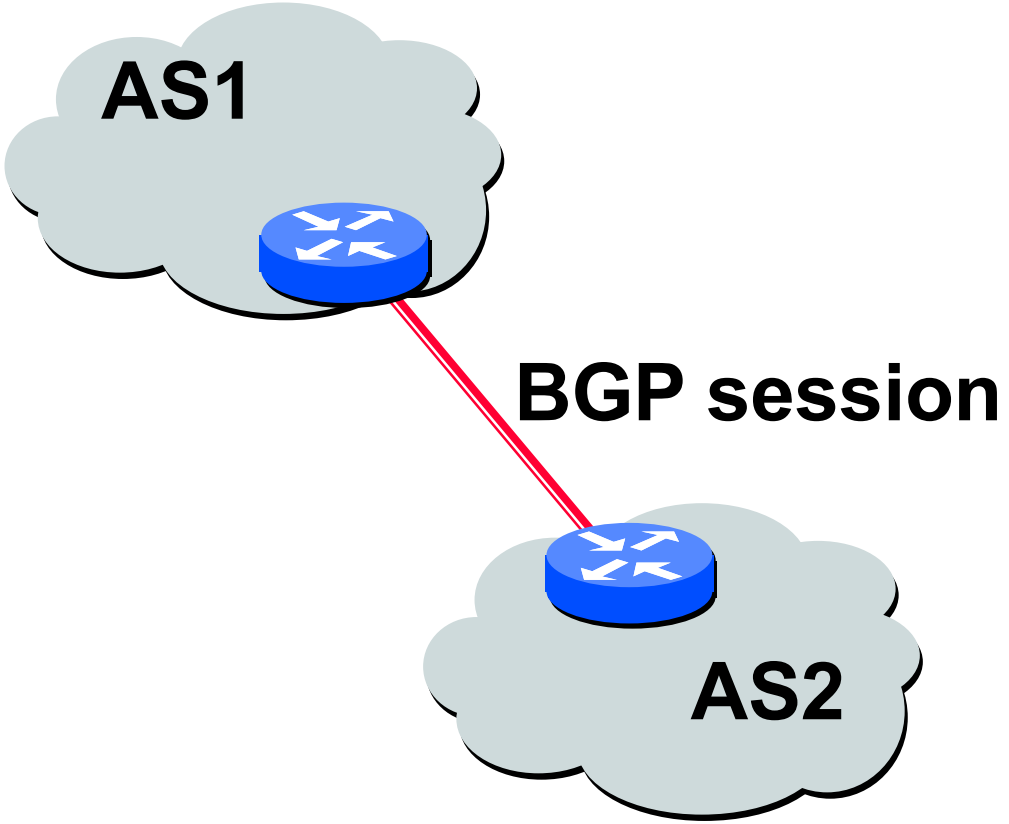
BGP Basic Operations



Establish session on TCP port 179

Exchange all active routes

Exchange incremental updates



while (connection is ALIVE)
exchange UPDATE message
select best available route
if route changes, export to neigh.

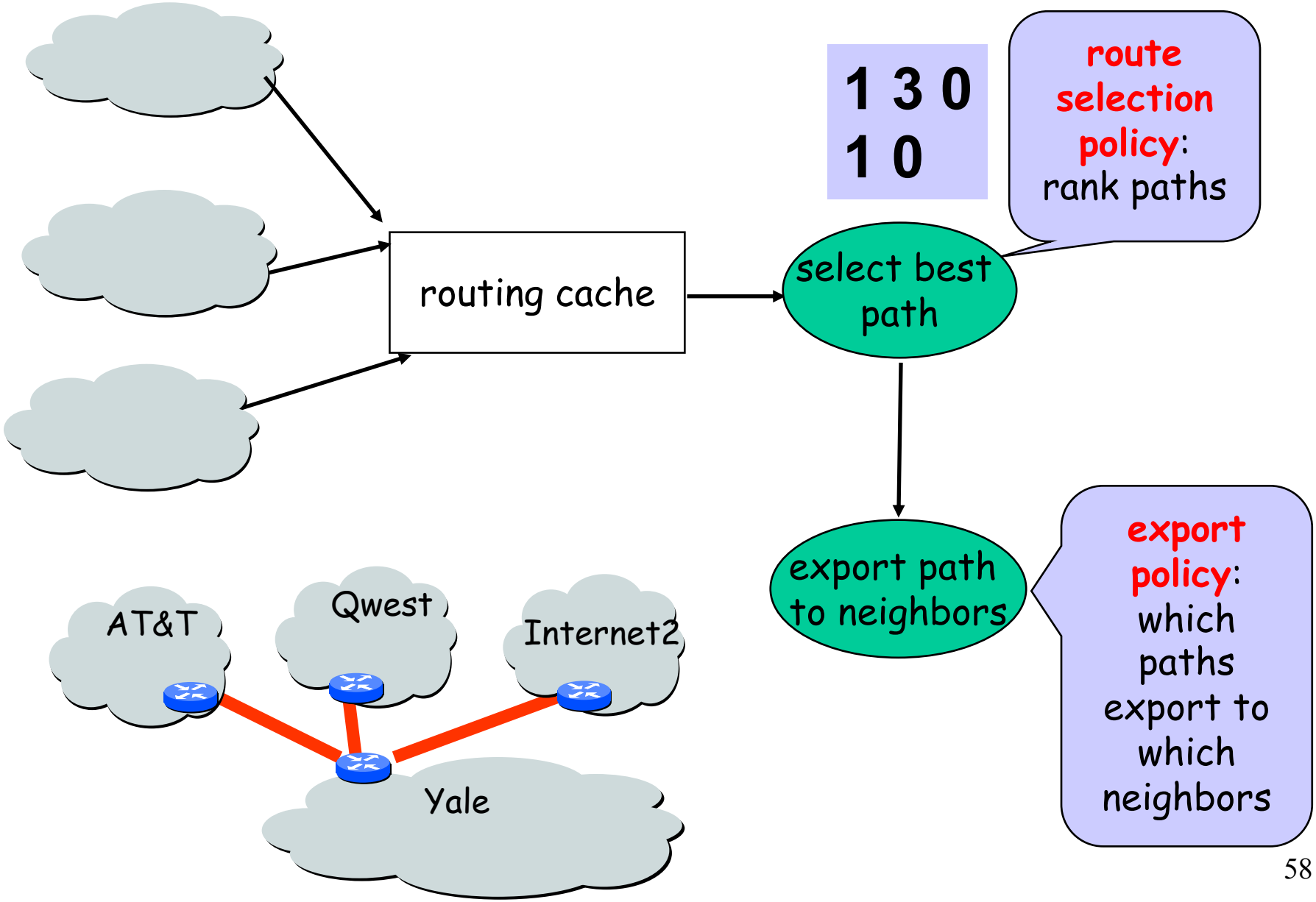
BGP Messages

- Four types of messages
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

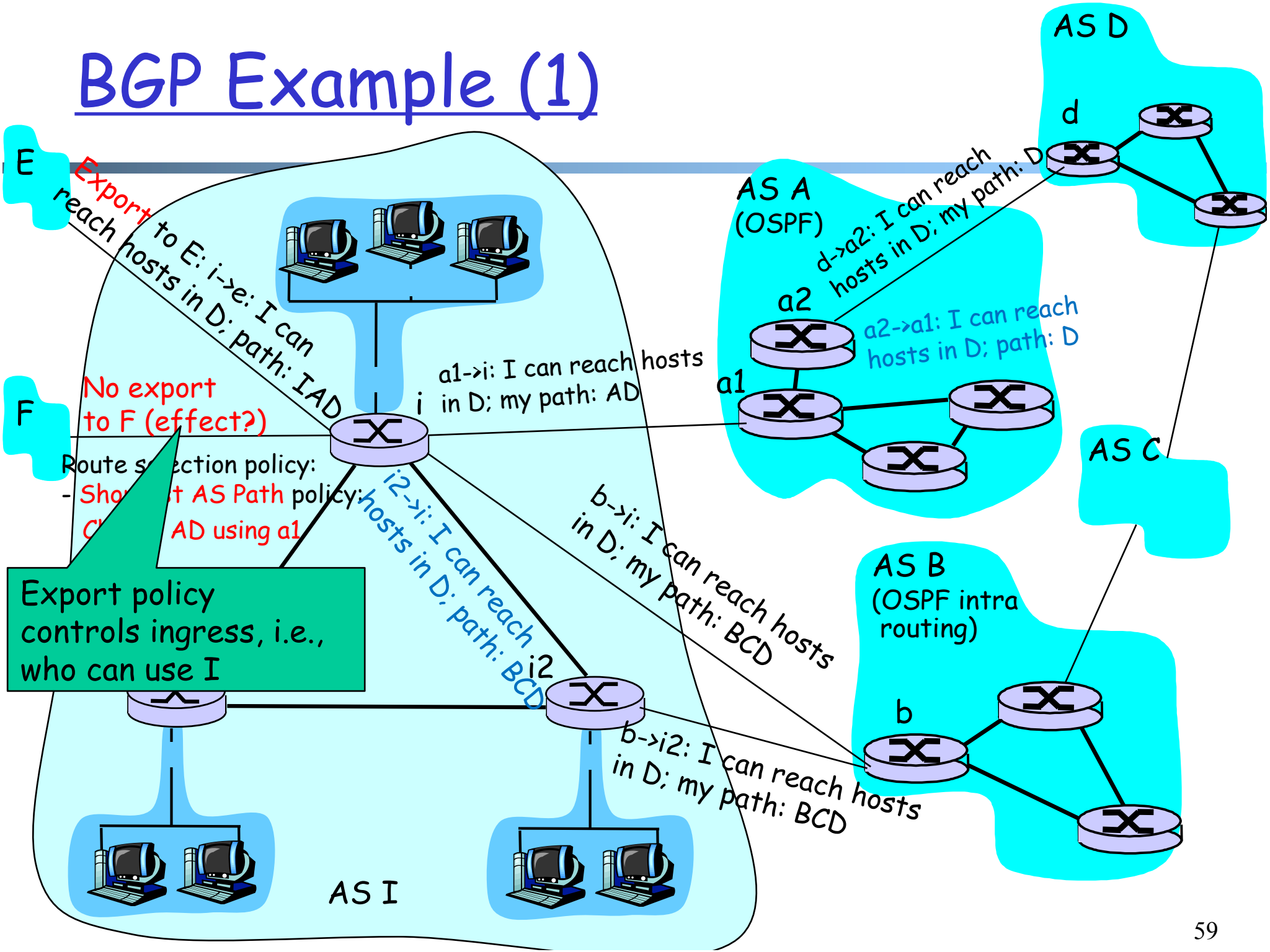
Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Basic routing computation protocols
      - Global Internet routing
        - Basic architecture
        - *BGP (Border Gateway Protocol): The de facto Inter-domain routing standard*
          - Basic operations
          - *BGP as a policy routing framework (control interdomain routes)*

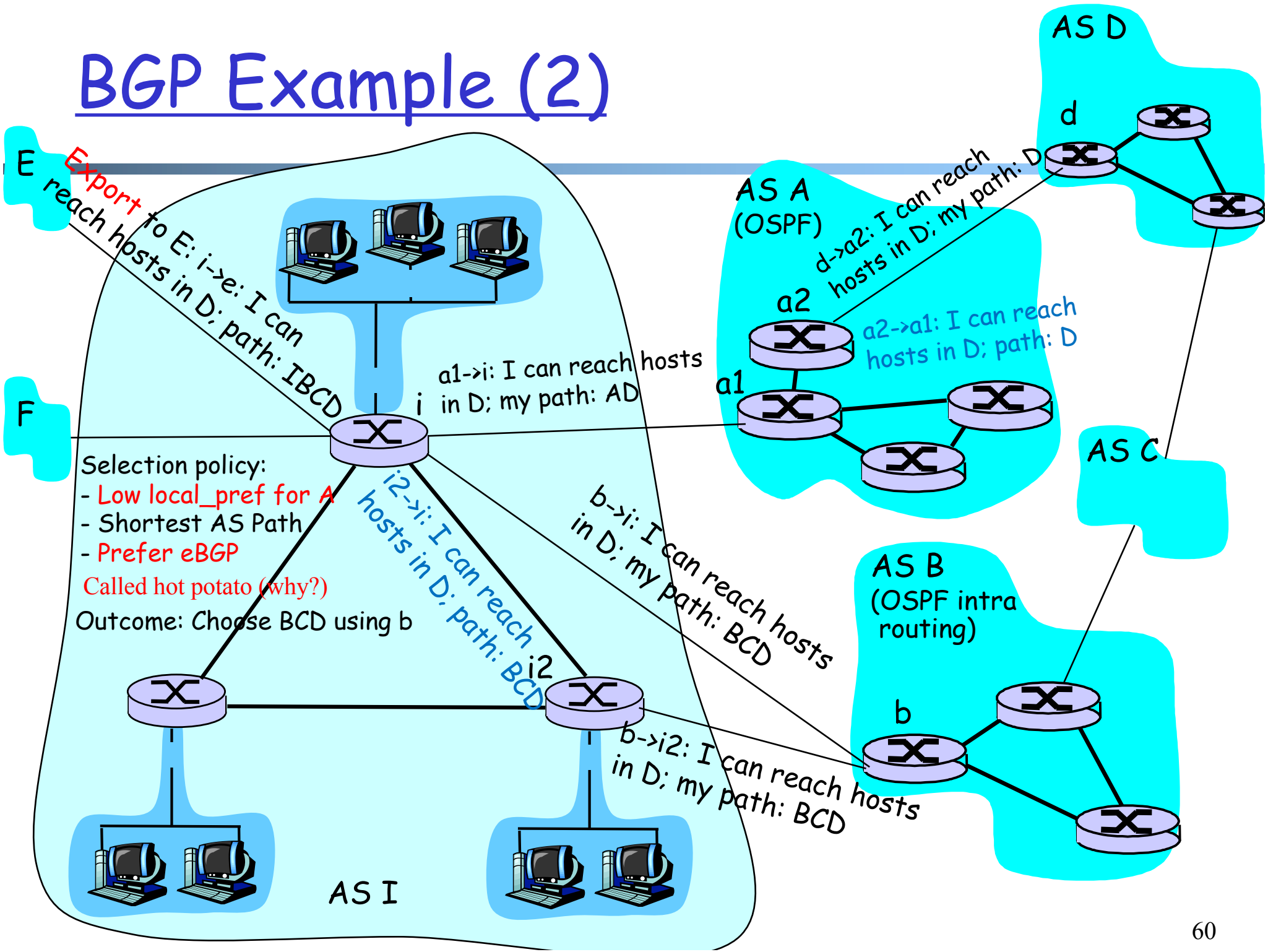
BGP Policy Routing Framework: Decision Components



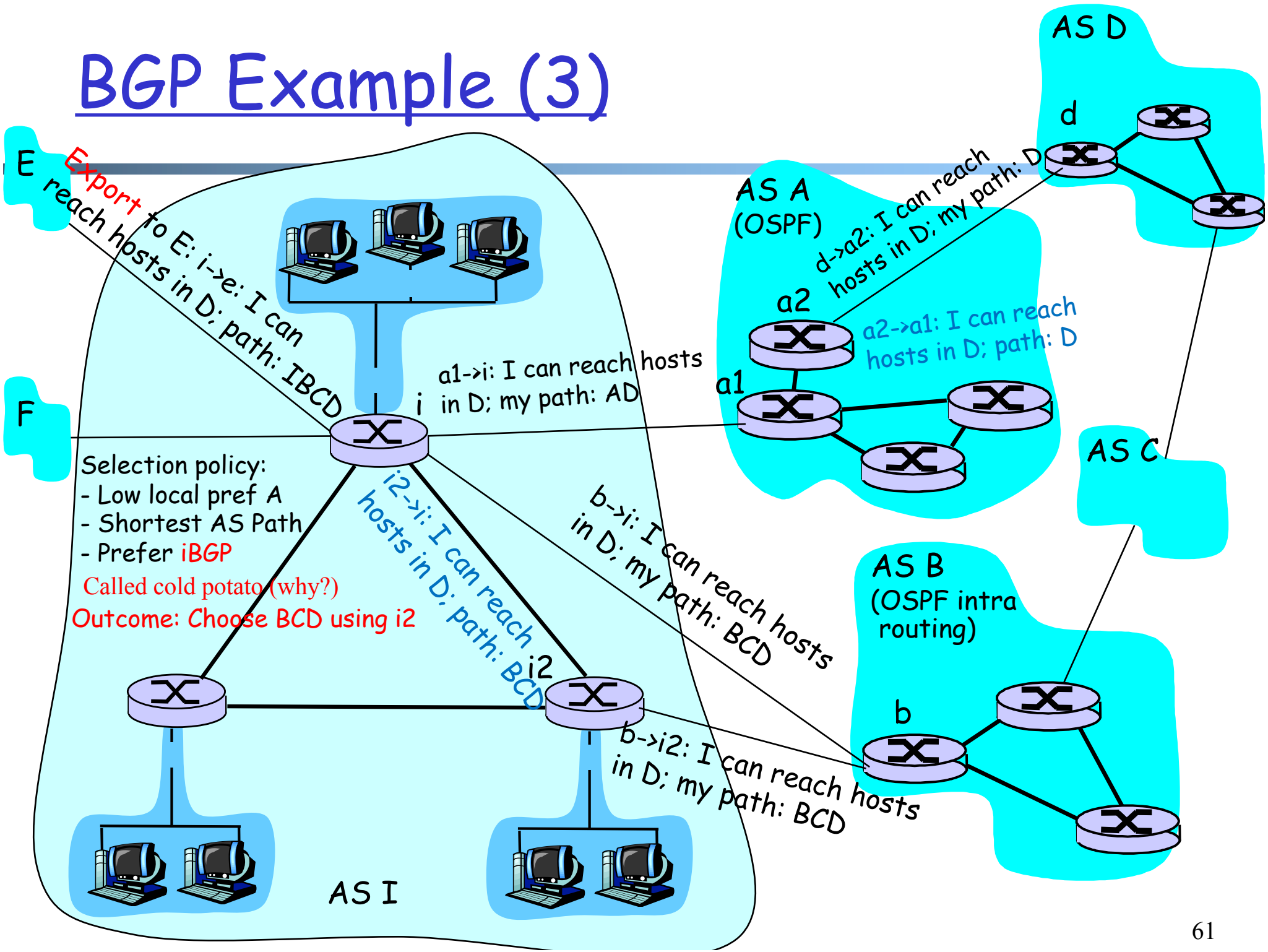
BGP Example (1)



BGP Example (2)



BGP Example (3)



Observing BGP Paths

- Using one of the looking glass servers:
  http://www.bgp4.as/looking-glasses

  https://www.gin.ntt.net/looking-glass/

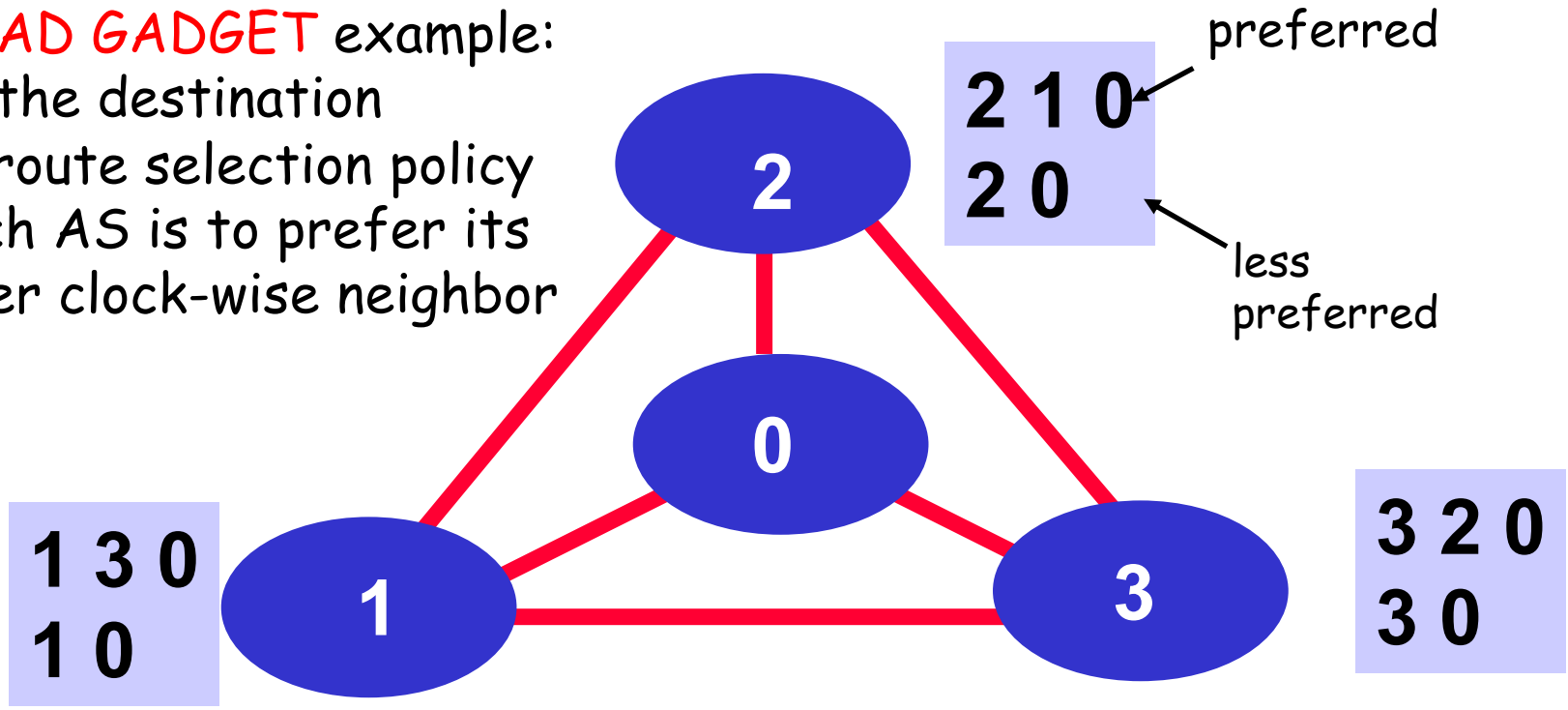
Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Basic routing computation protocols
      - Global Internet routing
        - Basic architecture
        - *BGP (Border Gateway Protocol): The de facto Inter-domain routing standard*
          - Basic operations
          - BGP as a policy routing framework (control interdomain routes)
        - *Policy/interdomain routing analysis*

Motivation: Policy Routing Stability

- A policy routing system can be considered as a system to aggregate local preferences, but aggregation may not be always successful.

The BAD GADGET example:
- 0 is the destination
- the route selection policy of each AS is to prefer its counter clock-wise neighbor



Policy (preferences) aggregation fails: routing instability !