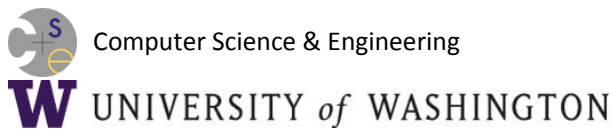


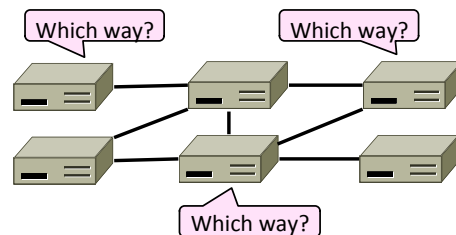
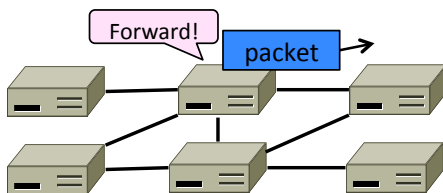
Introduction to Computer Networks

Routing Overview



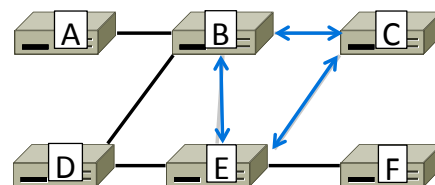
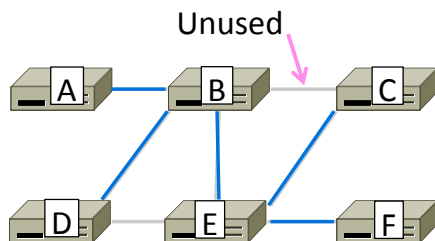
Routing versus Forwarding

- Forwarding is the process of sending a packet on its way
- Routing is the process of deciding in which direction to send traffic



Improving on the Spanning Tree

- Spanning tree provides basic connectivity
 - e.g., some path $B \rightarrow C$
- Routing uses all links to find “best” paths
 - e.g., use BC, BE, and CE



CSE 461 University of Washington

3

Perspective on Bandwidth Allocation

- Routing allocates network bandwidth adapting to failures; other mechanisms used at other timescales

Mechanism	Timescale / Adaptation
Load-sensitive routing	Seconds / Traffic hotspots
Routing	Minutes / Equipment failures
Traffic Engineering	Hours / Network load
Provisioning	Months / Network customers

CSE 461 University of Washington

4

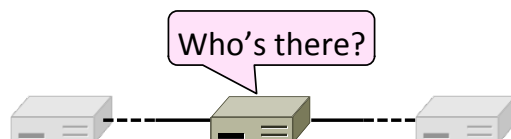
Goals of Routing Algorithms

- We want several properties of any routing scheme:

Property	Meaning
Correctness	Finds paths that work
Efficient paths	Uses network bandwidth well
Fair paths	Doesn't starve any nodes
Fast convergence	Recovers quickly after changes
Scalability	Works well as network grows large

Rules of Routing Algorithms

- Decentralized, distributed setting
 - All nodes are alike; no controller
 - Nodes only know what they learn by exchanging messages with neighbors
 - Nodes operate concurrently
 - May be node/link/message failures



Propose a Routing Algorithms

- Suppose you were to design a routing algorithm, how would you go about it?

Introduction to Computer Networks

Shortest Path Routing

(§5.2.1-5.2.2)



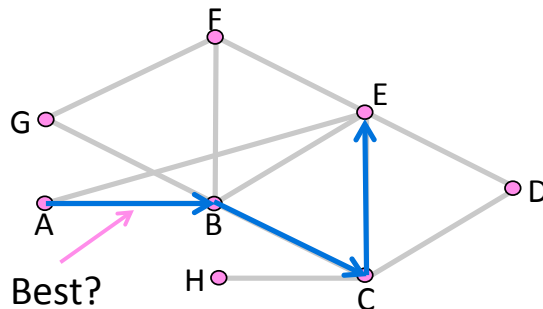
Computer Science & Engineering



UNIVERSITY *of* WASHINGTON

Topic

- Defining “best” paths with link costs
 - These are shortest path routes

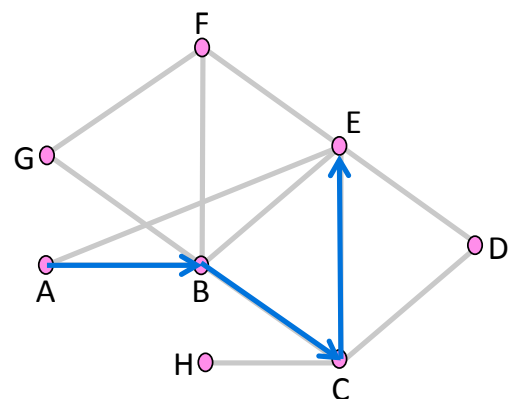


CSE 461 University of Washington

10

What are “Best” paths anyhow?

- Many possibilities:
 - Latency, avoid circuitous paths
 - Bandwidth, avoid slow links
 - Money, avoid expensive links
 - Hops, to reduce switching
- But only consider topology
 - Ignore workload, e.g., hotspots



CSE 461 University of Washington

11

Shortest Paths

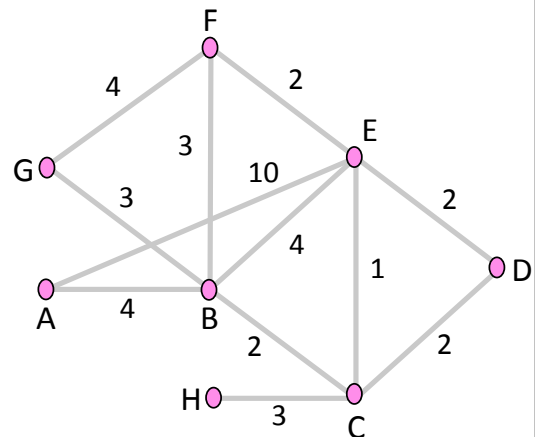
We'll approximate "best" by a cost function that captures the factors

- Often call lowest "shortest"

1. Assign each link a cost (distance)
2. Define best path between each pair of nodes as the path that has the lowest total cost (or is shortest)
3. Pick randomly to any break ties

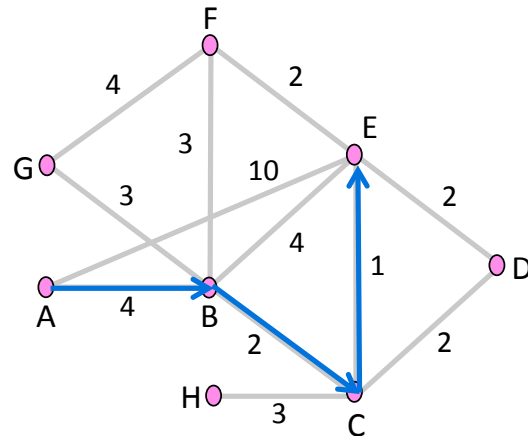
Shortest Paths (2)

- Find the shortest path $A \rightarrow E$
- All links are bidirectional, with equal costs in each direction
 - Can extend model to unequal costs if needed



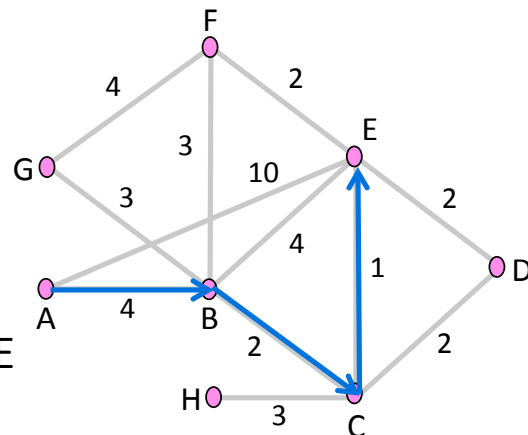
Shortest Paths (3)

- ABCE is a shortest path
- $\text{dist}(ABCE) = 4 + 2 + 1 = 7$
- This is less than:
 - $\text{dist}(ABE) = 8$
 - $\text{dist}(ABFE) = 9$
 - $\text{dist}(AE) = 10$
 - $\text{dist}(ABCDE) = 10$



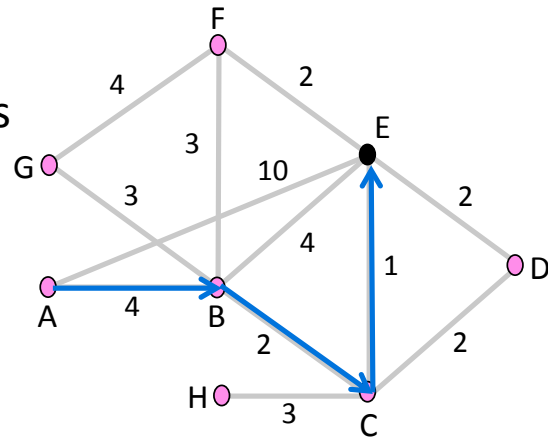
Shortest Paths (4)

- Optimality property:
 - Subpaths of shortest paths are also shortest paths
- ABCE is a shortest path
 - So are ABC, AB, BCE, BC, CE



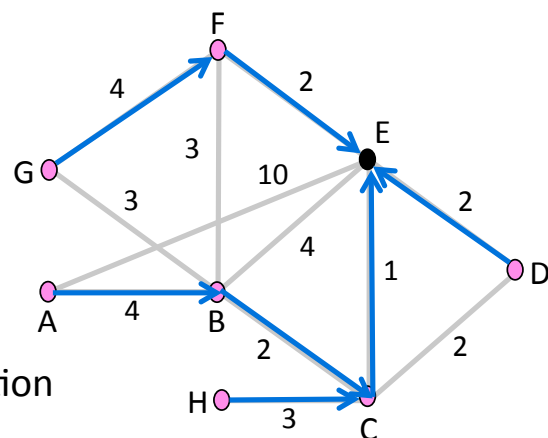
Sink Trees

- Sink tree for a destination is the union of all shortest paths towards the destination
 - Similarly source tree



Sink Trees (2)

- Implications:
 - Only need to use destination to follow shortest paths
 - Each node only need to send to the next hop
- Forwarding table at a node
 - Lists next hop for each destination
 - Routing table may know more



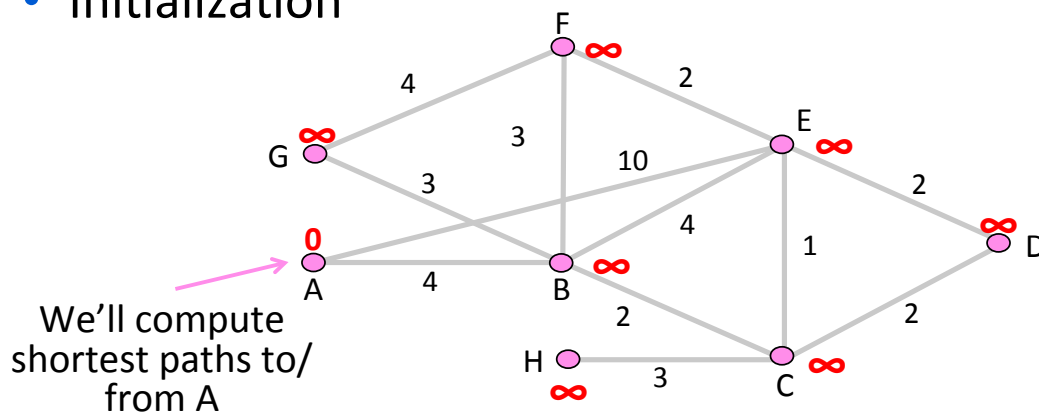
Dijkstra's Algorithm

Algorithm:

- Mark all nodes tentative, set distances from source to 0 (zero) for source, and ∞ (infinity) for all other nodes
- While tentative nodes remain:
 - Extract N, the one with lowest distance
 - Add link to N to the shortest path tree
 - Relax the distances of neighbors of N by lowering any better distance estimates

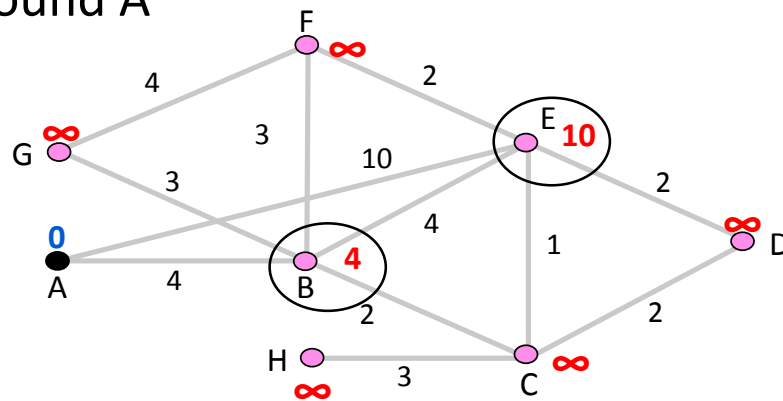
Dijkstra's Algorithm (2)

- Initialization



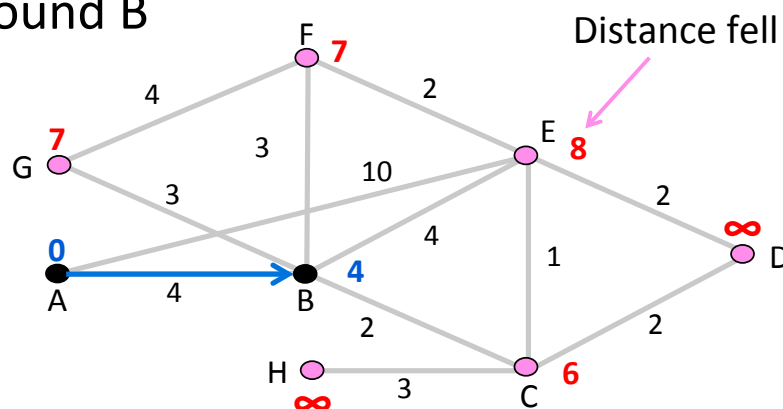
Dijkstra's Algorithm (3)

- Relax around A



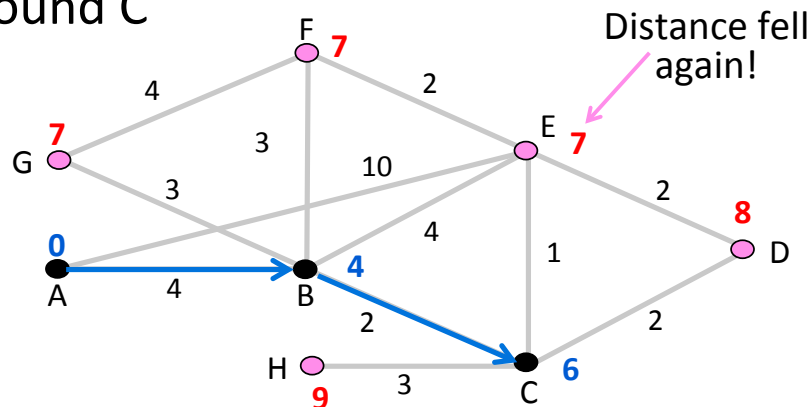
Dijkstra's Algorithm (4)

- Relax around B



Dijkstra's Algorithm (5)

- Relax around C

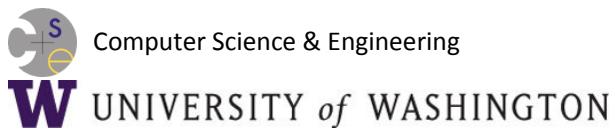


Dijkstra Comments

- Dynamic programming algorithm; leverages optimality property
- Runtime depends on efficiency of extracting min-cost node
- Gives us complete information on the shortest paths to/from one node
 - But requires complete topology

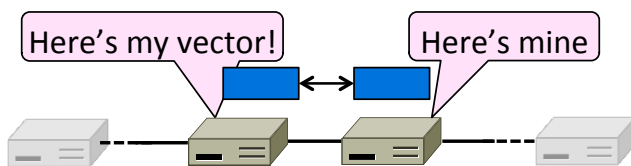
Introduction to Computer Networks

Distance Vector Routing (§5.2.4)



Topic

- How to compute shortest paths in a distributed network
 - The Distance Vector (DV) approach



Distance Vector Setting

Each node computes its forwarding table in a distributed setting:

1. Nodes know only the cost to their neighbors; not the topology
2. Nodes can talk only to their neighbors using messages
3. All nodes run the same algorithm concurrently
4. Nodes and links may fail, messages may be lost

Distance Vector Algorithm

Each node maintains a vector of distances to all destinations

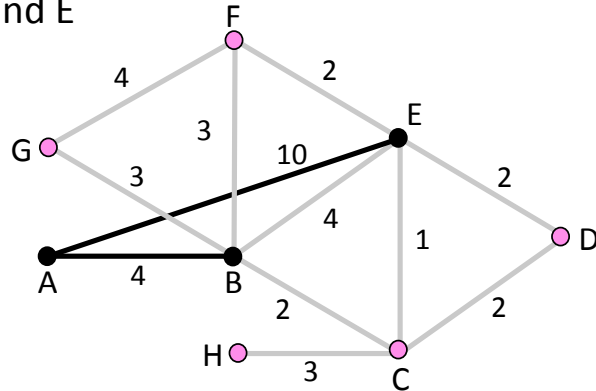
1. Initialize vector with 0 (zero) cost to self, ∞ (infinity) to other destinations
2. Periodically send vector to neighbors
3. Update vector for each destination by selecting the shortest distance heard, after adding cost of neighbor link
 - Use the best neighbor for forwarding

Distance Vector (2)

- Consider from the point of view of node A
 - Can only talk to nodes B and E

Initial vector →

To	Cost
A	0
B	∞
C	∞
D	∞
E	∞
F	∞
G	∞
H	∞



Distance Vector (3)

- First exchange with B, E; learn best 1-hop routes

To	B says	E says
A	∞	∞
B	0	∞
C	∞	∞
D	∞	∞
E	∞	0
F	∞	∞
G	∞	∞
H	∞	∞

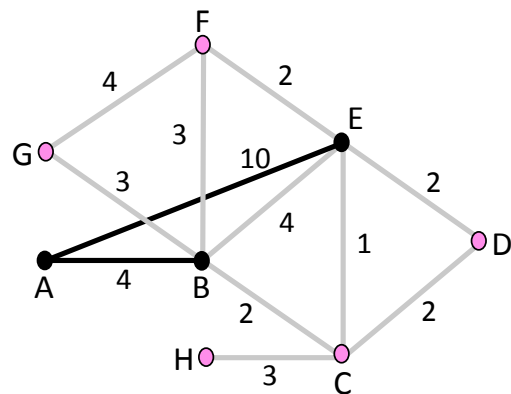
→

	B +4	E +10
A	∞	∞
B	4	∞
C	∞	∞
D	∞	∞
E	∞	10
F	∞	∞
G	∞	∞
H	∞	∞

→

To	A's Cost	A's Next
A	0	--
B	4	B
C	∞	--
D	∞	--
E	10	E
F	∞	--
G	∞	--
H	∞	--

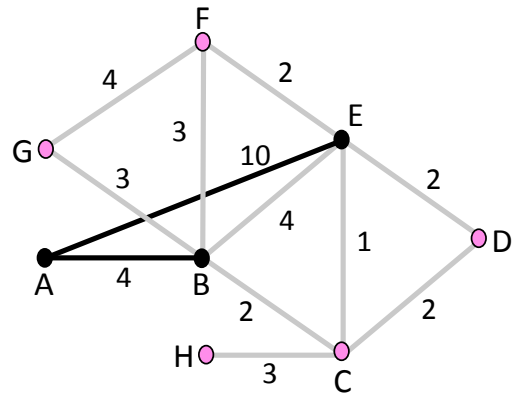
Learned better route



Distance Vector (4)

- Second exchange; learn best 2-hop routes

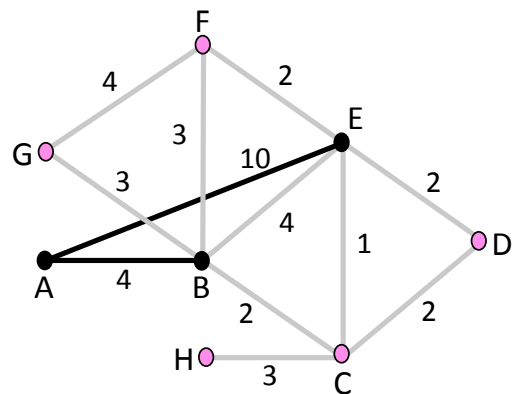
To	B says	E says		B +4	E +10		A's Cost	A's Next
A	4	10		8	20		0	--
B	0	4		4	14		4	B
C	2	1	→	6	11	→	6	B
D	∞	2		∞	12		12	E
E	4	0		8	10		8	B
F	3	2		7	12		7	B
G	3	∞		7	∞		7	B
H	∞	∞		∞	∞		∞	--



Distance Vector (4)

- Third exchange; learn best 3-hop routes

To	B says	E says		B +4	E +10		A's Cost	A's Next
A	4	8		8	18		0	--
B	0	3		4	13		4	B
C	2	1	→	6	11	→	6	B
D	4	2		8	12		8	B
E	3	0		7	10		7	B
F	3	2		7	12		7	B
G	3	6		7	16		7	B
H	5	4		9	14		9	B



Distance Vector (5)

- Subsequent exchanges; converged

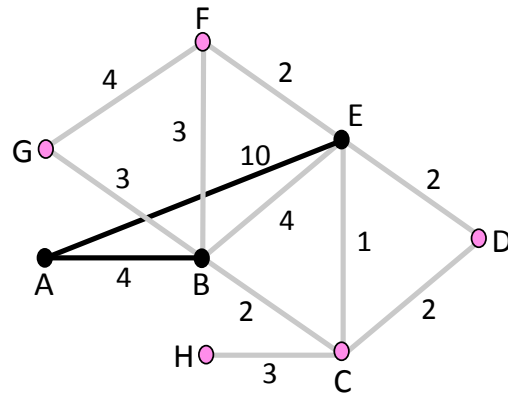
To	B says	E says
A	4	7
B	0	3
C	2	1
D	4	2
E	3	0
F	3	2
G	3	6
H	5	4

→

B +4	E +10
8	17
4	13
6	11
8	12
7	10
7	12
7	16
9	14

→

A's Cost	A's Next
0	--
4	B
6	B
8	B
8	B
7	B
7	B
7	B
9	B

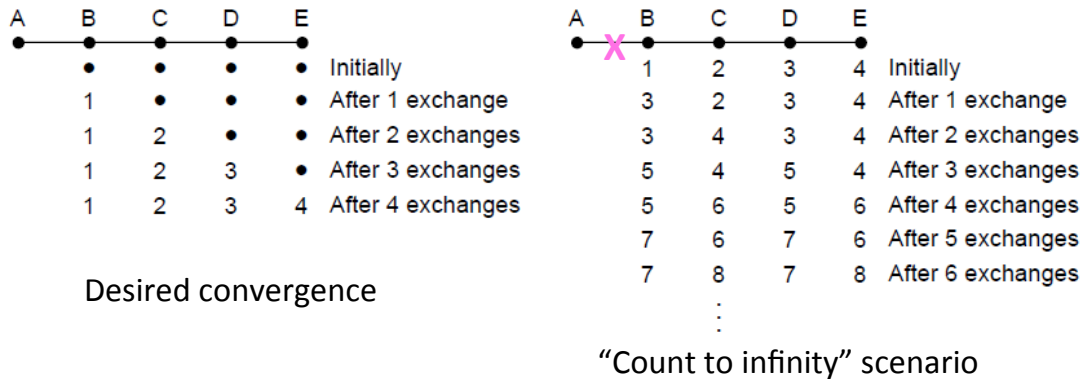


Distance Vector Dynamics

- Adding routes:
 - News travels one hop per exchange
- Removing routes
 - When a node fails, no more exchanges, other nodes forget
- But partitions (unreachable nodes in divided network) are a problem
 - “Count to infinity” scenario

Dynamics (2)

- Good news travels quickly, bad news slowly (inferred)

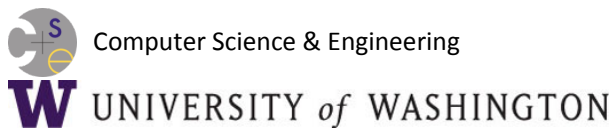


Dynamics (3)

- Various heuristics to address
 - e.g., “Split horizon, poison reverse” (Don’t send route back to where you learned it from.)
- But none are very effective
 - Link state now favored in practice
 - Except when very resource-limited

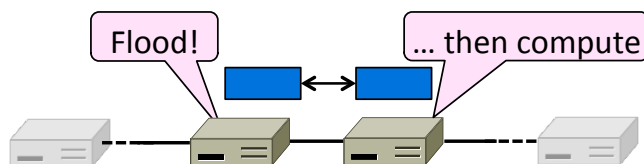
Introduction to Computer Networks

Link State Routing (§5.2.5)



Topic

- How to compute shortest paths in a distributed network
 - The Link-State (LS) approach



Link-State Routing

- One of two approaches to routing
 - Trades more computation than distance vector for better dynamics
- Widely used in practice
 - Used in Internet/ARPANET from 1979
 - Modern networks use OSPF and IS-IS

Link-State Algorithm

Proceeds in two phases:

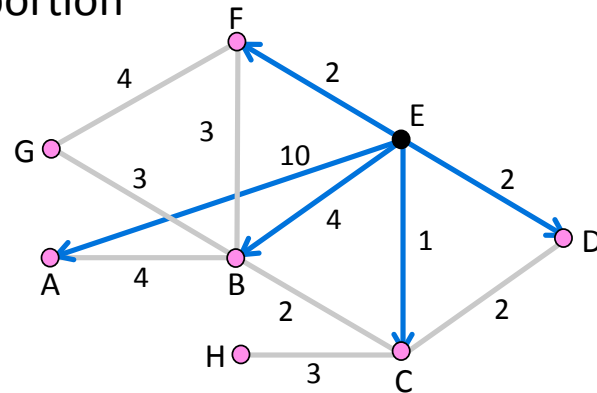
1. Nodes flood topology in the form of link state packets
 - Each node learns full topology
2. Each node computes its own forwarding table
 - By running Dijkstra (or equivalent)

Topology Dissemination

- Each node floods link state packet (LSP) that describes their portion of the topology

Node E's LSP flooded to A, B, C, D, and F

	Seq. #
A	10
B	4
C	1
D	2
F	2



Route Computation

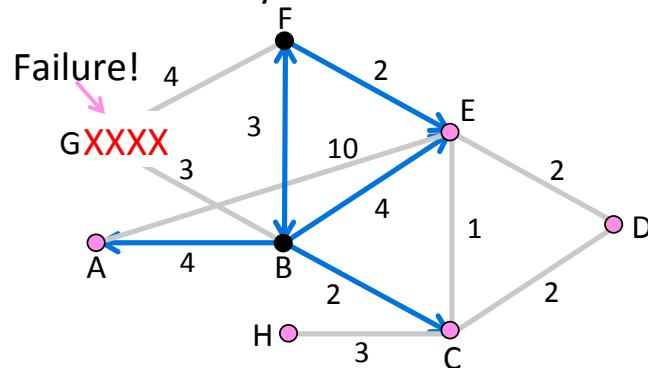
- Each node has full topology
 - By combining all LSPs
- Each node simply runs Dijkstra
 - Some replicated computation, but finds required routes directly
 - Compile forwarding table from sink/source tree
 - That's it folks!

Handling Changes

- Nodes adjacent to failed link or node will notice
 - Flood updated LSP with less connectivity

B's LSP	
Seq. #	
A	4
C	2
E	4
F	3
G	3

F's LSP	
Seq. #	
B	3
E	2
G	4



CSE 461 University of Washington

59

Handling Changes (2)

- Link failure
 - Both nodes notice, send updated LSPs
 - Link is removed from topology
- Node failure
 - All neighbors notice a link has failed
 - Failed node can't update its own LSP
 - But it is OK: all links to node removed

CSE 461 University of Washington

60

Handling Changes (3)

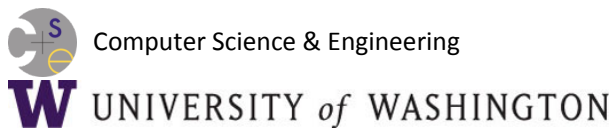
- Addition of a link or node
 - Add LSP of new node to topology
 - Old LSPs are updated with new link
- Additions are the easy case ...

DV/LS Comparison

Property	Distance Vector	Link-State
Correctness	Distributed Bellman-Ford	Replicated Dijkstra
Efficient paths	Approx. with shortest paths	Approx. with shortest paths
Fair paths	Approx. with shortest paths	Approx. with shortest paths
Fast convergence	Slow – many exchanges	Fast – flood and compute
Scalability	Excellent – storage/compute	Moderate – storage/compute

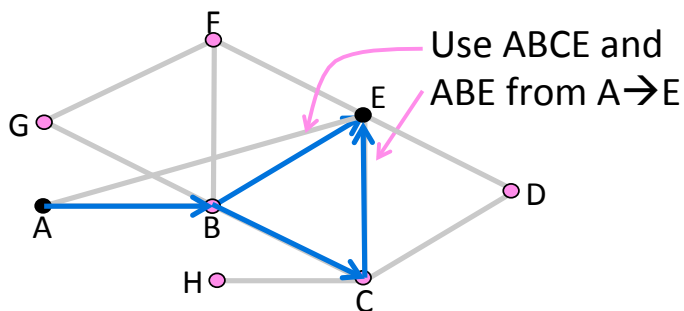
Introduction to Computer Networks

Equal-Cost Multi-Path Routing (§5.2.1)



Topic

- More on shortest path routes
 - Allow multiple shortest paths

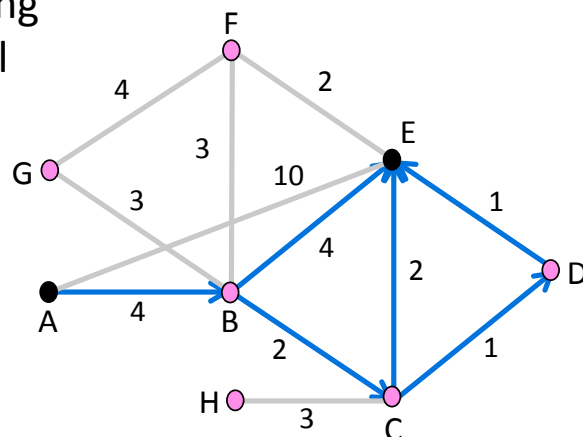


Multipath Routing

- Allow multiple routing paths from node to destination be used at once
 - Topology has them for redundancy
 - Using them can improve performance
- Questions:
 - How do we find multiple paths?
 - How do we send traffic along them?

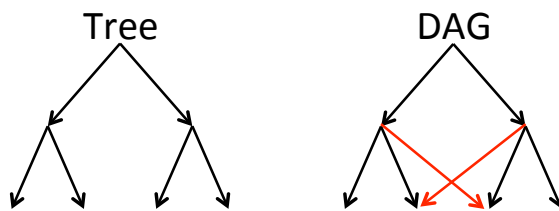
Equal-Cost Multipath Routes

- One form of multipath routing
- Extends shortest path model
 - Keep set if there are ties
- Consider $A \rightarrow E$
 - $ABE = 4 + 4 = 8$
 - $ABCE = 4 + 2 + 2 = 8$
 - $ABCDE = 4 + 2 + 1 + 1 = 8$
 - Use them all!



Source “Trees”

- With ECMP, source/sink “tree” is a directed acyclic graph (DAG)
 - Each node has set of next hops
 - Still a compact representation

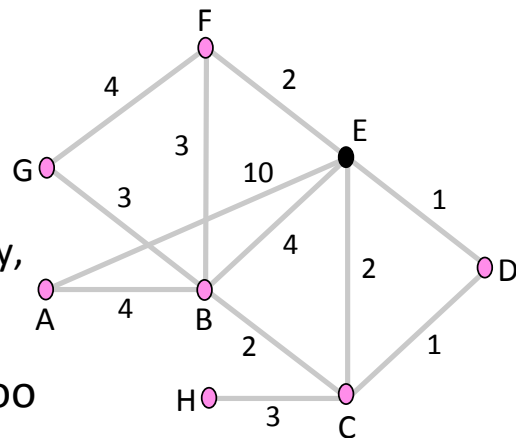


CSE 461 University of Washington

69

Source “Trees” (2)

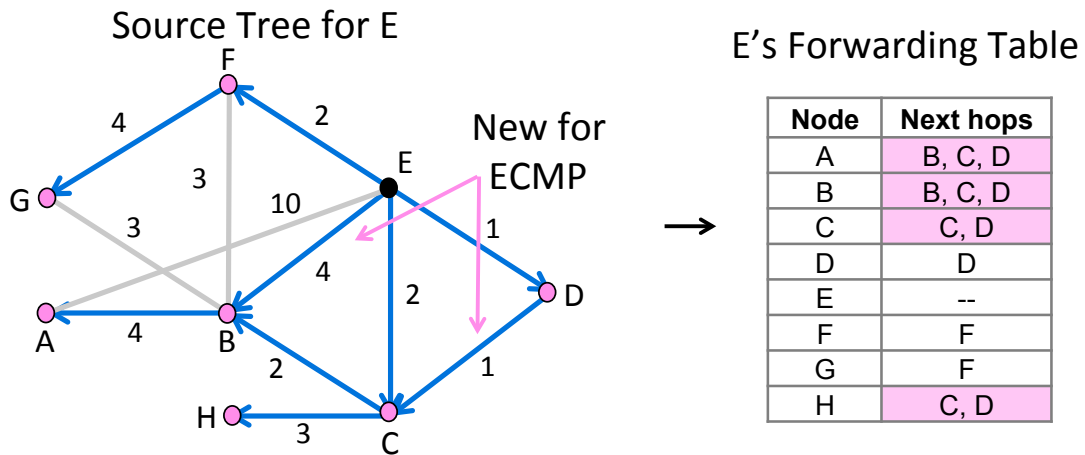
- Find the source “tree” for E
 - Procedure is Dijkstra, simply remember set of next hops
 - Compile forwarding table similarly, may have set of next hops
- Straightforward to extend DV too
 - Just remember set of neighbors



CSE 461 University of Washington

70

Source "Trees" (3)



CSE 461 University of Washington

71

ECMP Forwarding

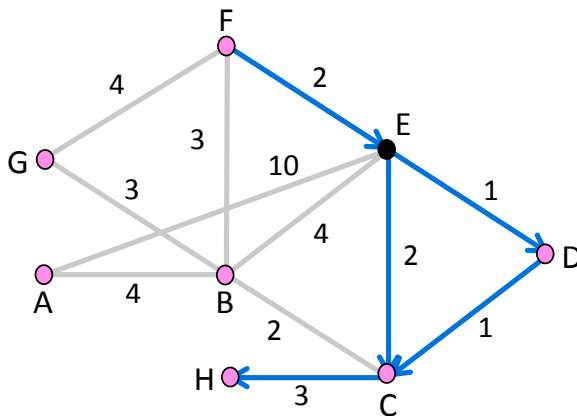
- Could randomly pick a next hop for each packet based on destination
 - Balances load, but adds jitter
- Instead, try to send packets from a given source/destination pair on the same path
 - Source/destination pair is called a flow
 - Hash flow identifier to next hop
 - No jitter within flow, but less balanced

CSE 461 University of Washington

72

ECMP Forwarding (2)

Multipath routes from F to H



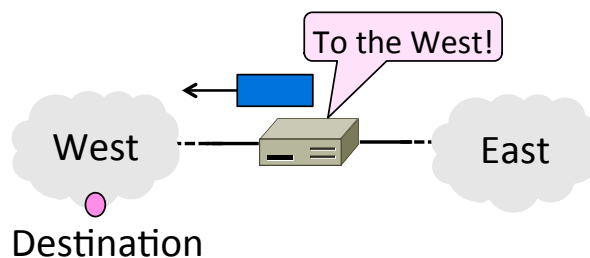
E's Forwarding Choices

Flow	Possible next hops	Example choice
F → H	C, D	D
F → C	C, D	D
E → H	C, D	C
E → C	C, D	C

Use both paths to get to one destination

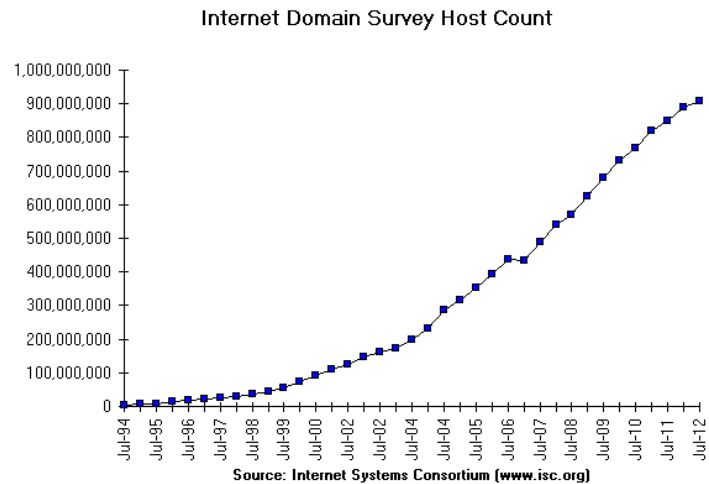
Topic: Hierarchical Routing

- How to scale routing with hierarchy in the form of regions
 - Route to regions, not individual nodes



Internet Growth

- At least a billion Internet hosts and growing ...

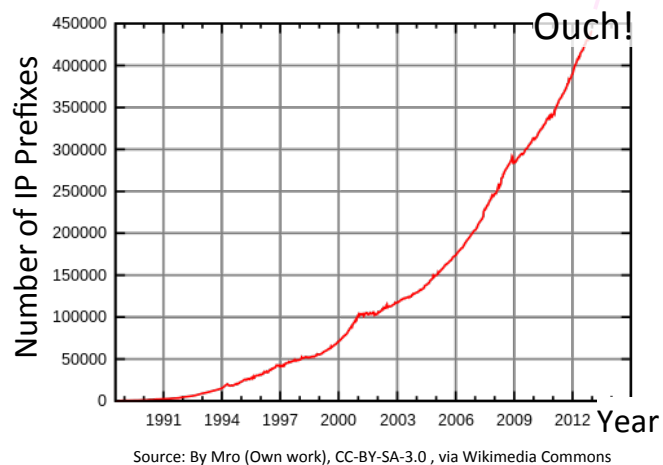


CSE 461 University of Washington

76

Internet Routing Growth

- Internet growth translates into routing table growth (even using prefixes) ...



CSE 461 University of Washington

77

Impact of Routing Growth

1. Forwarding tables grow
 - Larger router memories, may increase lookup time
2. Routing messages grow
 - Need to keep all nodes informed of larger topology
3. Routing computation grows
 - Shortest path calculations grow faster than the size of the network

Techniques to Scale Routing

1. IP prefixes
 - Route to blocks of hosts
2. Network hierarchy
 - Route to network regions
3. IP prefix aggregation
 - Combine, and split, prefixes

} Last week

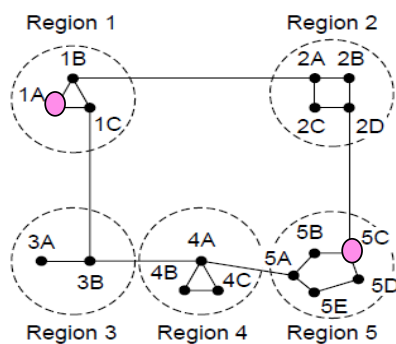
} This topic

} Next topic

Hierarchical Routing

- Introduce a larger routing unit
 - IP prefix (hosts) ← from one host
 - Region, e.g., ISP network
- Route first to the region, then to the IP prefix within the region
 - Hide details within a region from outside of the region

Hierarchical Routing (2)



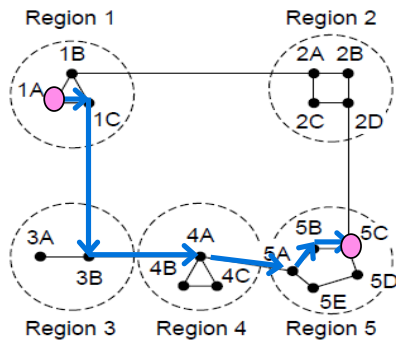
Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Hierarchical Routing (3)



Full table for 1A

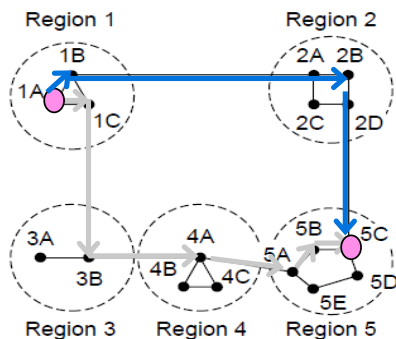
Dest.	Line	Hops
1A	—	—
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	—	—
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Hierarchical Routing (4)

- Penalty is longer paths



Full table for 1A

Dest.	Line	Hops
1A	—	—
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	—	—
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

1C is best route to region 5, except for destination 5C

Observations

- Outside a region, nodes have one route to all hosts within the region
 - This gives savings in table size, messages and computation
- However, each node may have a different route to an outside region
 - Routing decisions are still made by individual nodes; there is no single decision made by a region

Introduction to Computer Networks

IP Prefix Aggregation and Subnets (§5.6.2)



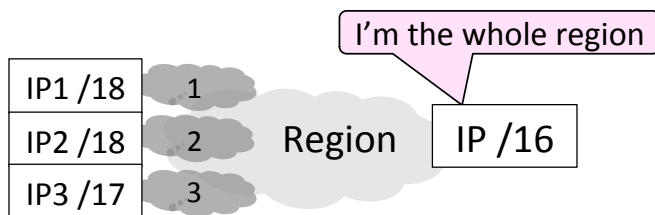
Computer Science & Engineering



UNIVERSITY of WASHINGTON

Prefixes and Hierarchy

- IP prefixes already help to scale routing, but we can go further
 - We can use a less specific (larger) IP prefix as a name for a region

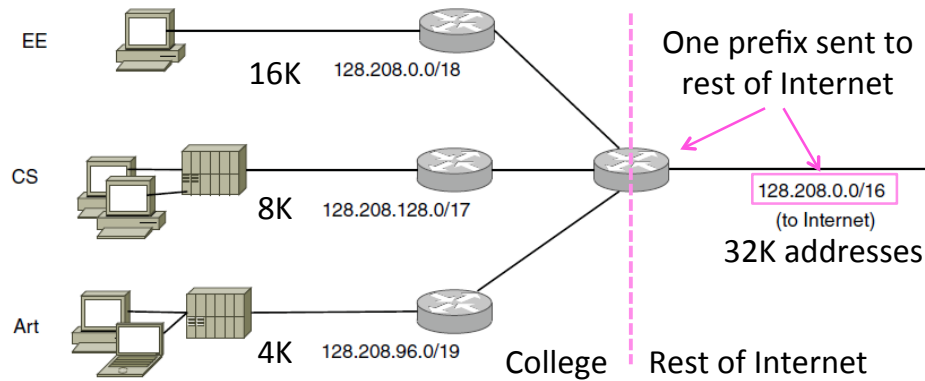


Subnets and Aggregation

1. Subnets
 - Internally split one large prefix into multiple smaller ones
2. Aggregation
 - Externally join multiple smaller prefixes into one large prefix

Subnets

- Internally split up one IP prefix

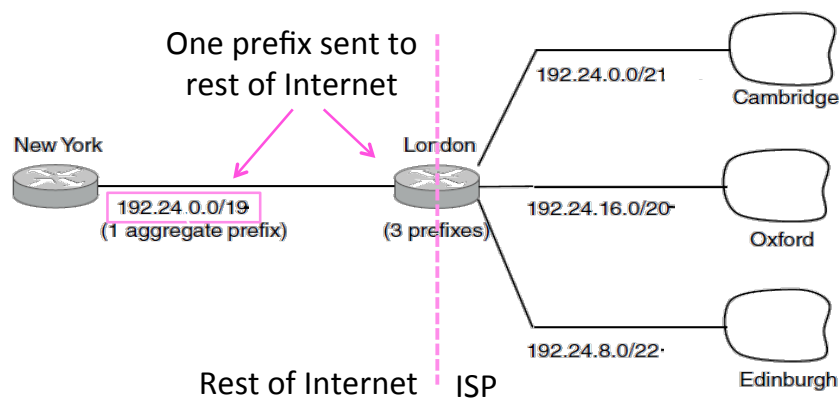


CSE 461 University of Washington

91

Aggregation

- Externally join multiple separate IP prefixes



CSE 461 University of Washington

92

Announcements

- Proxy project due today
- TOR project will be released on Wednesday
 - Part 1: design due in a week
 - Part 2: two more weeks for coding
- Homework 3 will be released next week; due in a week

Introduction to Computer Networks

Routing with Policy (BGP)
(§5.6.7)



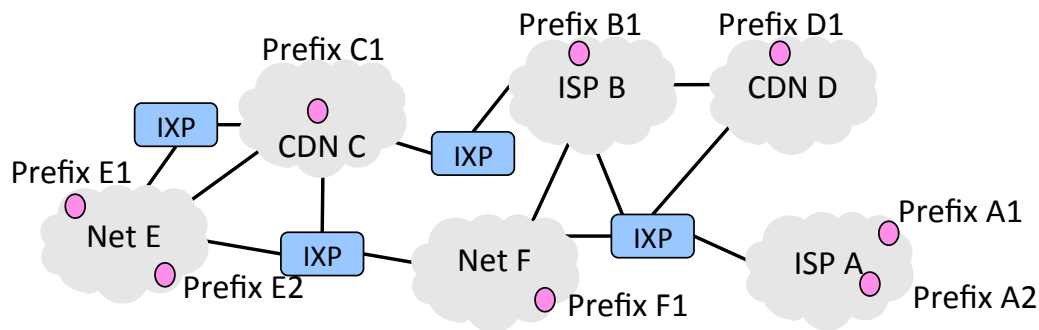
Computer Science & Engineering



UNIVERSITY *of* WASHINGTON

Structure of the Internet

- Networks (ISPs, CDNs, etc.) group hosts as IP prefixes
- Networks are richly interconnected, often using IXPs



CSE 461 University of Washington

96

Internet-wide Routing Issues

- Two problems beyond routing within an individual network
 1. Scaling to very large networks
 - Techniques of IP prefixes, hierarchy, prefix aggregation
 2. Incorporating policy decisions
 - Letting different parties choose their routes to suit their own needs

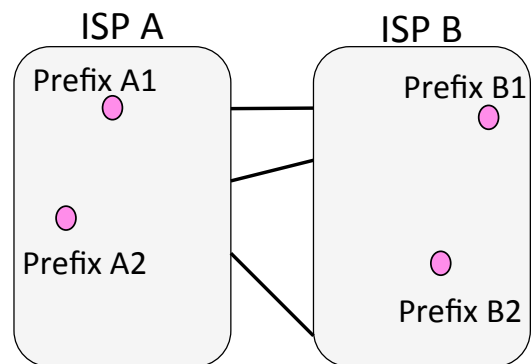
← Yikes!

CSE 461 University of Washington

97

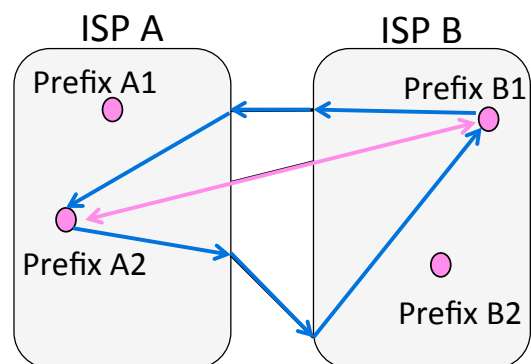
Effects of Independent Parties

- Each party selects routes to suit its own interests
 - e.g, shortest path in ISP
- What path will be chosen for $A2 \rightarrow B1$ and $B1 \rightarrow A2$?
 - What is the best path?



Effects of Independent Parties (2)

- Selected paths are longer than overall shortest path
 - And asymmetric too!
- This is a consequence of independent goals and decisions, not hierarchy

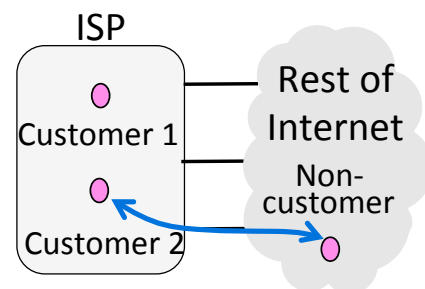


Routing Policies

- Capture the goals of different parties – could be anything
 - E.g., Internet2 only carries non-commercial traffic
- Common policies we'll look at:
 - ISPs give TRANSIT service to customers
 - ISPs give PEER service to each other

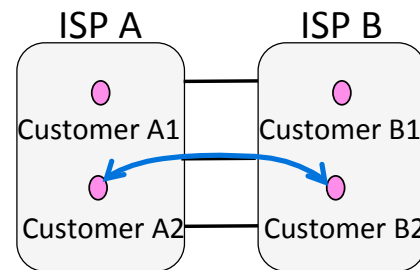
Routing Policies – Transit

- One party (customer) gets TRANSIT service from another party (ISP)
 - ISP accepts traffic for customer from the rest of Internet
 - ISP sends traffic from customer to the rest of Internet
 - Customer pays ISP for the privilege



Routing Policies – Peer

- Both party (ISPs in example) get PEER service from each other
 - Each ISP accepts traffic from the other ISP only for their customers
 - ISPs do not carry traffic to the rest of the Internet for each other
 - ISPs don't pay each other

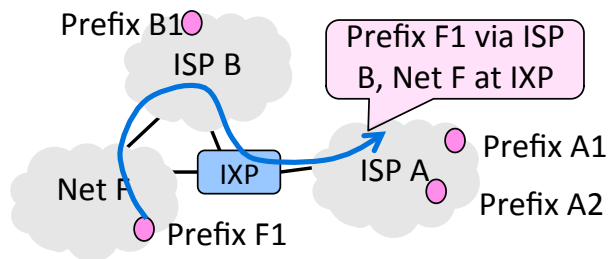


Routes on the Internet

- Assume that you have transit & peering links
- Questions:
 - What strategy should ISPs use to select paths?
 - What do the resulting end-to-end paths look like?

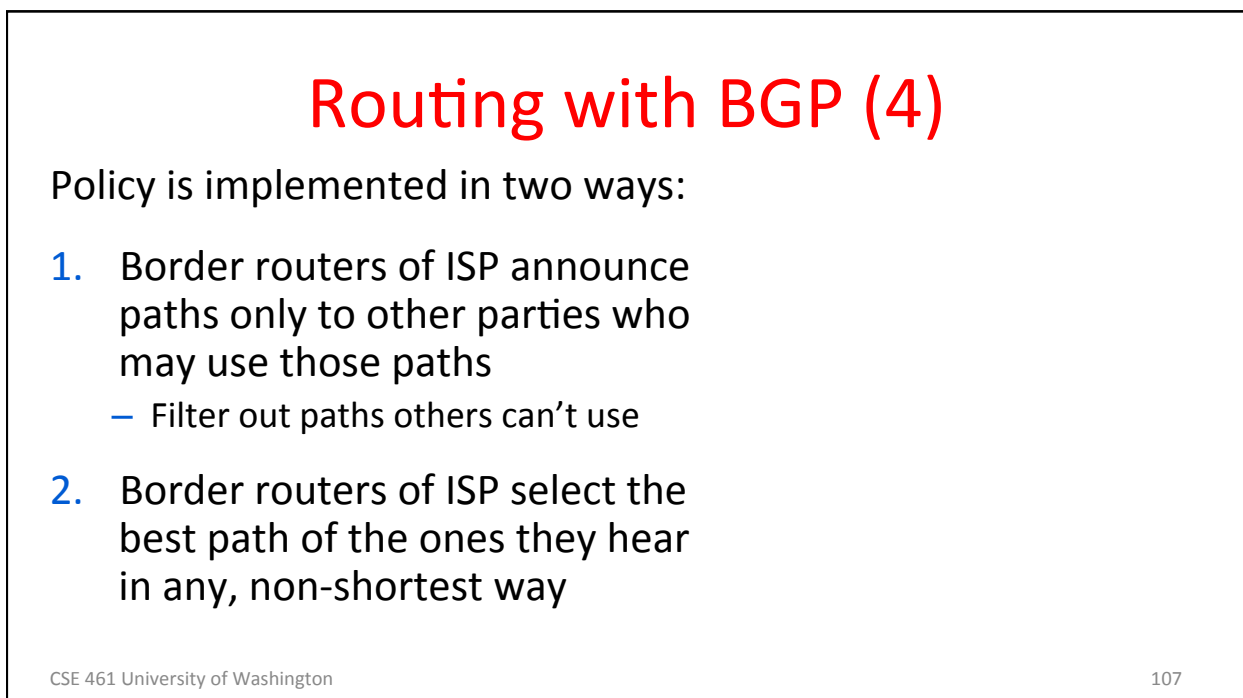
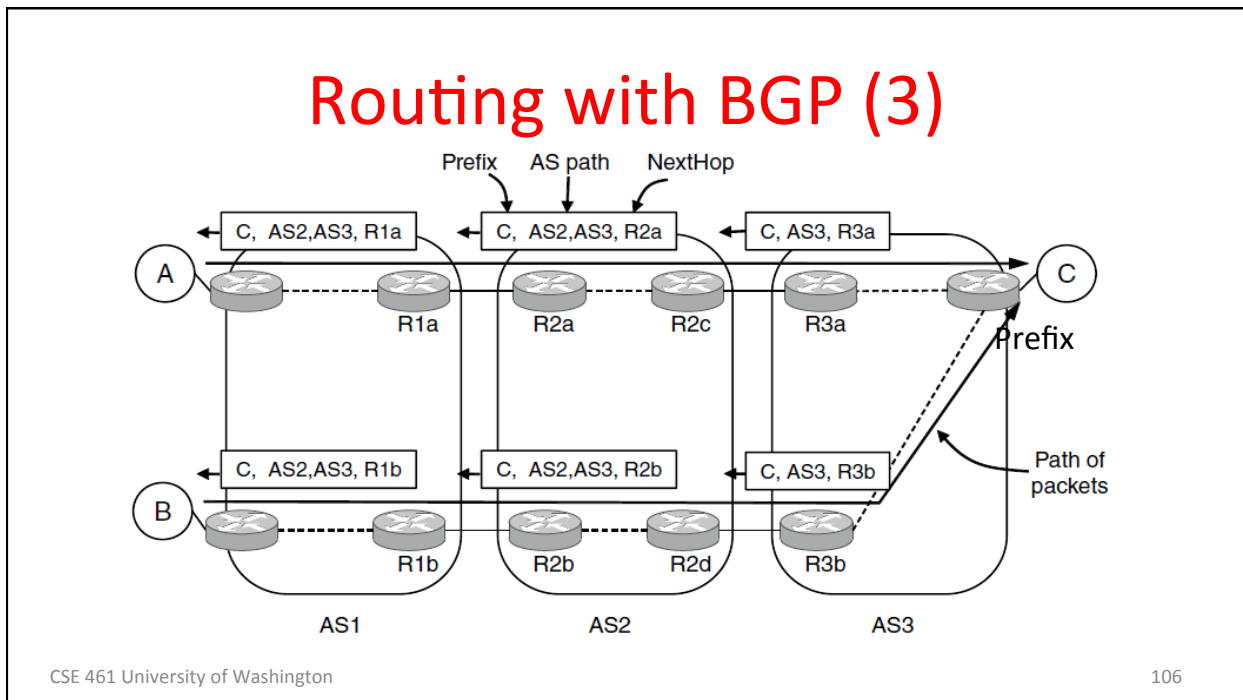
Routing with BGP (Border Gateway Protocol)

- BGP is the interdomain routing protocol used in the Internet
 - Path vector, a kind of distance vector



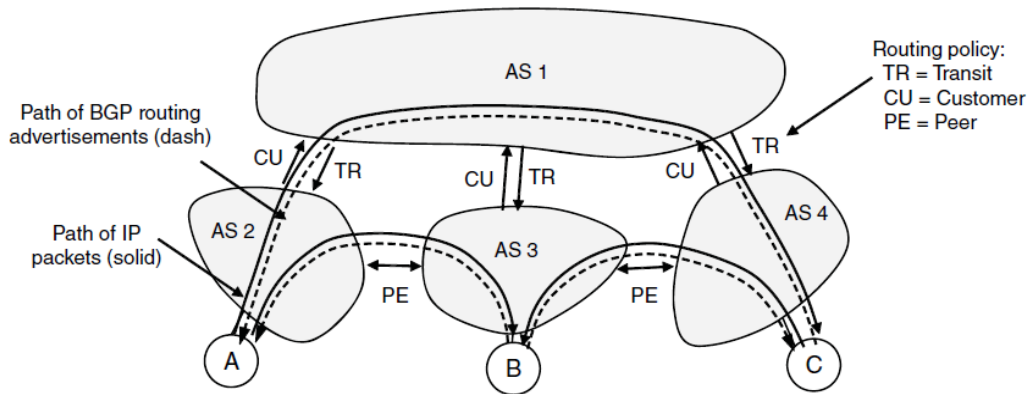
Routing with BGP (2)

- Different parties like ISPs are called AS (Autonomous Systems)
- Border routers of ASes announce BGP routes to each other
- Route announcements contain an IP prefix, path vector, next hop
 - Path vector is list of ASes on the way to the prefix; list is to find loops
- Route announcements move in the opposite direction to traffic



Routing with BGP (5)

- TRANSIT: AS1 says [B, (AS1, AS3)], [C, (AS1, AS4)] to AS2

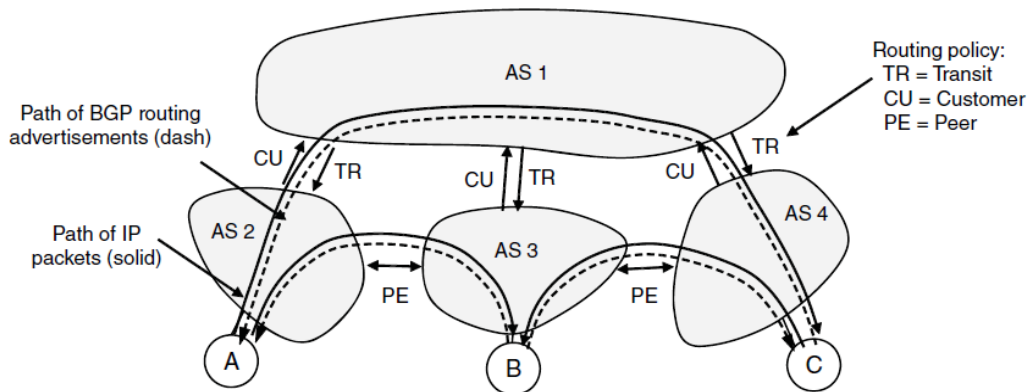


CSE 461 University of Washington

108

Routing with BGP (6)

- CUSTOMER (other side of TRANSIT): AS2 says [A, (AS2)] to AS1

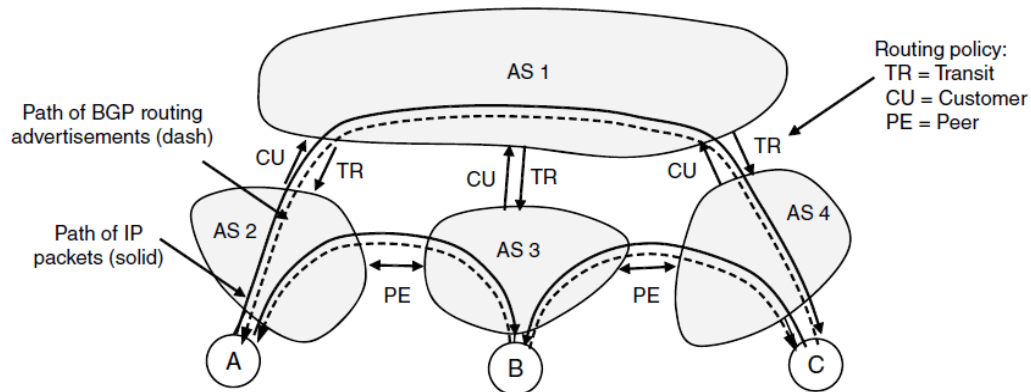


CSE 461 University of Washington

109

Routing with BGP (7)

- PEER: AS2 says [A, (AS2)] to AS3, AS3 says [B, (AS3)] to AS2

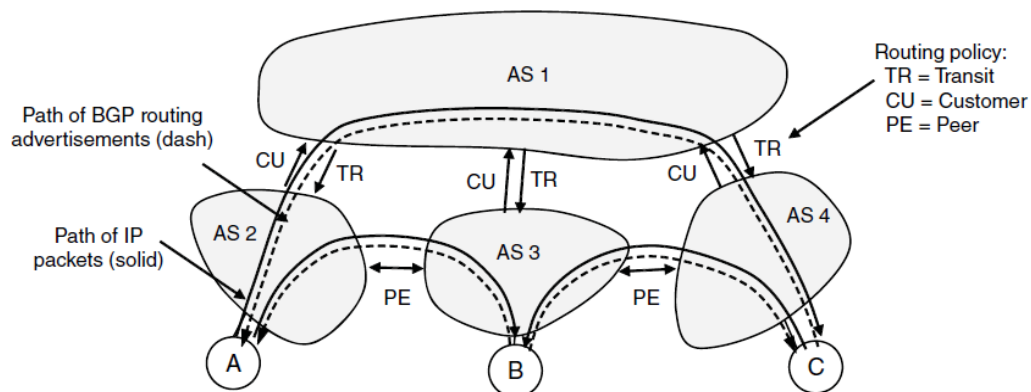


CSE 461 University of Washington

110

Routing with BGP (8)

- AS2 hears two routes to B (via AS1, AS3) and chooses AS3 (Free!)



CSE 461 University of Washington

111

BGP Thoughts

- Much more beyond basics to explore!
- Policy is a substantial factor
 - Can we even be independent decisions will be sensible overall?
- Other important factors:
 - Convergence effects
 - How well it scales
 - Integration with intradomain routing
 - And more ...