Master Course Computer Networks IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Dr. Nils Kammenhuber

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München
http://www.net.in.tum.de

Routing

Short note on pronunciation of the word "routing"

- [‘ru:tɪŋ] /r-oo-ting/ = British English
- [‘raʊdɪŋ] /r-ow-ding/ = American English
- Both are correct!

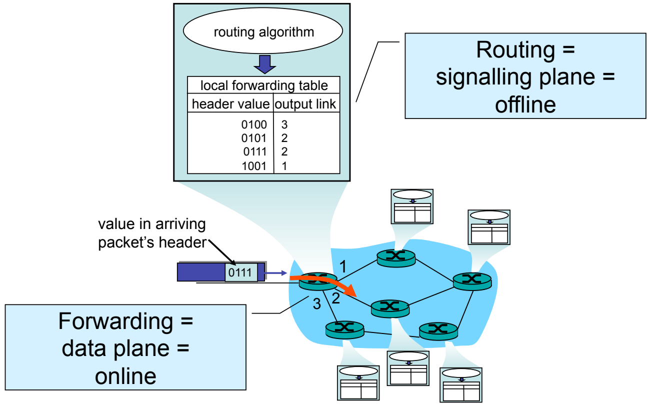
Chapter outline: Routing

- Routing and forwarding
- Routing algorithms recapitulated
  - Link state
  - Distance Vector
  - Path Vector
- Intradomain routing protocols
  - RIP
  - OSPF
- Interdomain routing
  - Hierarchical routing
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering
- Multicast routing

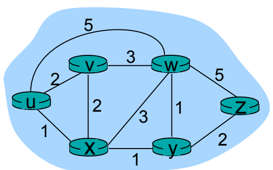
Routing ≠ Forwarding

- Routing:
  - The process of determining the best path for a specific type of packets (usually: all packets with the same destination) through the network
  - Performed jointly by the routers of a network by exchanging many messages
  - Analogy: Read street map, plan journey
- Forwarding:
  - The process where a router relays a packet to a neighbouring router. Selection of the neighbouring router depends on the previous routing protocol calculations
  - Performed by one router on one packet
  - Analogy: Read a street sign and determine if we should take the next exit
- In practice, this distinction is often ignored
  - “If router A routes packet X, then …”
  - Actually, it doesn‘t – it *forwards* X.

Signalling plane and data plane



Graph abstraction



Graph: G = (N,E)

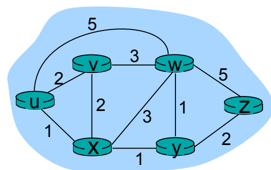
N = nodes = set of routers = { u, v, w, x, y, z }

E = edges = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where N is set of peers and E is set of TCP connections

Graph abstraction: costs



- c(x,x’) =: cost of link (x,x’)
  e.g.: c(w,z) = 5
- cost could always be 1,
- or inversely related to bandwidth,
- or inversely related to congestion

Cost of path $(x_1, x_2, x_3,\dots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \dots + c(x_{p-1},x_p)$

Question: What’s the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

Global or decentralized information?

Global:
- All routers have complete topology and link cost info
- *link state* algorithms (L-S)

Decentralized:
- Router only knows physically-connected neighbors and link costs to neighbors
- Iterative process of computation = exchange of info with neighbours
- *distance vector* algorithms (D-V)
- Variant: *path vector* algorithms

Static or dynamic?

Static:
- Routes change slowly over time

Dynamic:
- Routes change more quickly
  - periodic update
  - in response to link cost changes

A broader routing classification

- Type of algorithm: Link State, Distance Vector, Path Vector, …
- Scope:
  - Intradomain
  - Interdomain
  - Special purpose (e.g., sensor network)
- Type of traffic: Unicast vs. multicast
- Type of reaction: "Static" vs. Dynamic/adaptive
  - Warning: "Dynamic routing" is a fuzzy term:
    a) Dynamic ≔ reacts to topology changes (state of the art)
    b) Dynamic ≔ reacts to traffic changes (even better, but most protocols don't do that!)
- Trigger type:
  - Permanent routing (standard)
  - On-demand routing: only start routing algorithm if there is traffic to be forwarded (e.g., some wireless ad-hoc networks)

A Link-State Routing Algorithm

- Net topology and link costs made known to each node
  - Accomplished via *link state broadcasts*
  - All nodes have same information (…after all information has been exchanged)
- Each node independently computes least-cost paths from one node ("source") to all other nodes
  - Usually done using Dijkstra's shortest-path algorithm
    - refer to any algorithms & data structures lecture/textbook
    - n nodes in network ⇒ $O(n^2)$ or $O(n \log n)$
  - Gives forwarding table for that node
- Result:
  - All nodes have the same information,
  - … thus calculate the same shortest paths,
  - … hence obtain consistent forwarding tables

A Link-State Routing Algorithm

- Net topology and link costs made known to each node
  - Accomplished via *link state broadcasts*
  - All nodes have same info
- Each node independently computes least-cost paths from one node ("source") to all other nodes
  - Usually done using Dijkstra's algorithm
  - Yields forwarding table for that node

Notation:
- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra's Algorithm

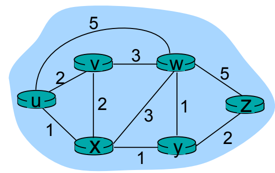
Initialization:

N' = {u}
for all nodes v
  if v adjacent to u
    then D(v) = c(u,v)
  else D(v) = ∞

Loop
  find w not in N' such that D(w) is a minimum
  add w to N'
  update D(v) for all v adjacent to w and not in N' :
    D(v) = min(D(v), D(w) + c(w,v))
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N'

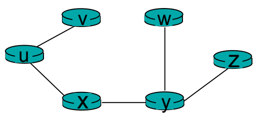
Dijkstra's algorithm: example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:



Resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

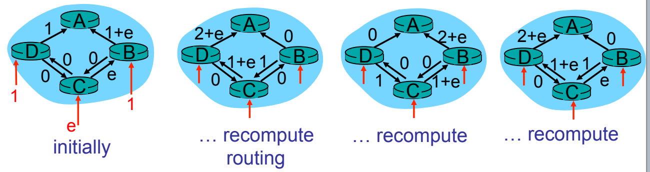
Dijkstra's algorithm, discussion

Algorithm complexity: *n* nodes

- each iteration: need to check all nodes, w, not in N
- $n(n+1)/2$ comparisons: $O(n^2)$
- more efficient implementations possible: $O(n \log n)$

Oscillations possible:

- e.g., link cost = amount of carried traffic



Distance Vector Algorithm

- No node knows entire topology
- Nodes only communicate with neighbours (i.e., no broadcasts)
- Nodes *jointly* calculate shortest paths
  - Iterative process
  - Algorithm == protocol
- Distributed application of Bellman-Ford algorithm
  - refer to any algorithms&data structures lecture/textbook

Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Let

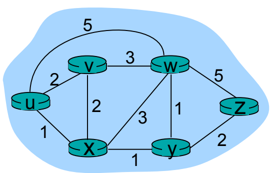
- $c(x,y)$ ≔ cost of edge from x to y
- $d_x(y)$ ≔ cost of least-cost path from x to y
- Set to ∞ if no path / no edge available

Then

$d_x(y) = \min \{c(x,v) + d_v(y)\}$

where min is taken over all neighbours v of x

Bellman-Ford example



We can see that
$d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z),\ c(u,x) + d_x(z),\ c(u,w) + d_w(z) \} = \min \{2 + 5,\ 1 + 3,\ 5 + 3\} = 4$$

Node that calculated minimum is next hop in shortest path
→ forwarding table

Distance Vector Algorithm

- Define $D_x(y)$ ≔ estimate of least cost from x to y
- Node x knows cost to each neighbour v: $c(x,v)$
- Node x maintains distance vector $\mathbf{D}_x$ ≔ [$D_x(y)$: $y \in N$]
  (N ≔ set of nodes)
- Node x also maintains copies of its neighbours' distance vectors
  - Received via update messages from neighbours
  - For each neighbour v,
    x knows $\mathbf{D}_v$ = [$D_v(y)$: $y \in N$]

Distance vector algorithm (4)

Basic idea:

- From time to time, each node sends its own distance vector estimate D to its neighbours
  - Asynchronously
- When a node x receives new DV estimate from neighbour, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- Under minor, natural conditions, these estimates $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance Vector Algorithm (5)

Iterative, asynchronous:
Each local iteration caused by:

- Local link cost change
- DV update message from neighbour

Distributed:

- Each node notifies neighbours *only* when its DV changes
  - neighbours then notify their neighbours if this caused *their* DV to change
  - etc.

Usually some waiting delay between consecutive updates

Each node:

Forever:

wait for (change in local link cost *or* message arriving from neighbour)

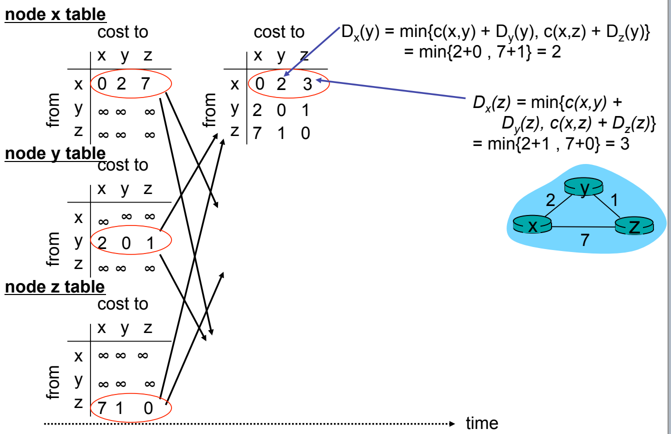
↓

recompute estimates

↓

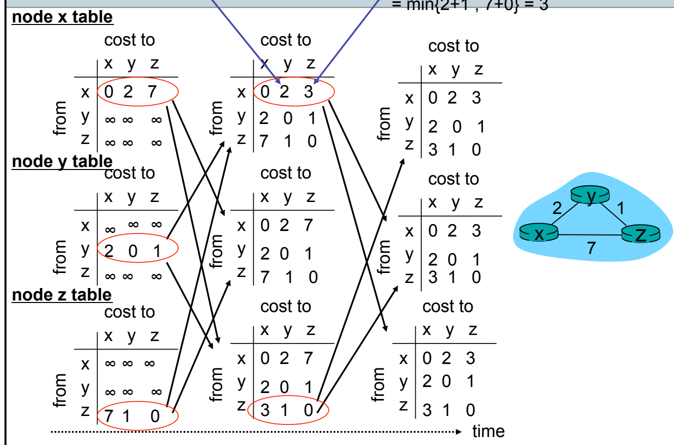
if (DV to any destination has changed) { *notify* neighbours }

Distance Vector Algorithm (6)



$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = \min\{2+0 , 7+1\} = 2$

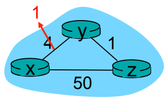
$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = \min\{2+1 , 7+0\} = 3$



Distance Vector: link cost changes (1)

Link cost changes:

- Node detects local link cost change
- Updates routing info, recalculates distance vector
- If DV changes, notify neighbours



"good news travels fast"

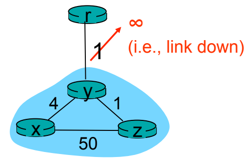
At time t_0, *y* detects the link-cost change, updates its DV, and informs its neighbours.

At time t_1, *z* receives the update from *y* and updates its table. It computes a new least cost to *x* and sends its neighbours its new DV.

At time t_2, *y* receives *z*'s update and updates its distance table. *y*'s least costs do not change and hence *y* does *not* send any message to *z*.

Distance Vector: link cost changes (2)

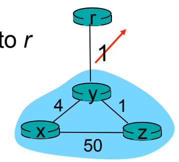
- But: bad news travels slow
- In example: Many iterations before algorithm stabilizes!
  1. Cost increase for *y*→*r*:
     - *y* consults DV,
     - *y* selects "cheaper" route via *z* (cost 2+1 = 3),
     - Sends update to *z* and *x* (cost to *r* now 3 instead of 1)
  2. *z* detects cost increase for path to *r*:
     - was 1+1, is now 3+1
     - Sends update to *y* and *x* (cost to *r* now 4 instead of 2)
  3. *y* detects cost increase, sends update to *z*
  4. *z* detects cost increase, sends update to *y*
  5. ….



- Symptom: "count to infinity" problem

Distance Vector: Problem Solutions…

- Finite infinity: Define some number to be ∞ (in RIP: ∞ ≔ 16)
- Split Horizon:
  - Tell to any neighbour that is part of a best path to a destination that the destination cannot be reached
  - If *z* routes through *y* to get to *r* *z* tells *y* that its own (i.e., *y*'s) distance to *r* is infinite (so *y* won't route to *r* via *z*)
- Poisoned Reverse:
  - In addition, *actively* advertise a route as unreachable to the neighbour from which the route was learned
- (Warning: Terms often used interchangeably!)



…that only half work

- Mechanisms can be combined
- Both mechanisms can significantly increase number of routing messages
- Often help, but cannot solve all problem instances
  - Think yourselves: Come up with a topology where this does not help
  - Try it – it's not hard and a good exercise

Comparison of LS and DV algorithms

Message complexity
- LS: with n nodes, E links, $O(nE)$ msgs sent
- DV: exchange between neighbours only
  - convergence time varies

Speed of Convergence
- LS: $O(n^2)$ algorithm requires $O(nE)$ msgs
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:
- node can advertise incorrect *link* cost
- each node computes only its *own* table

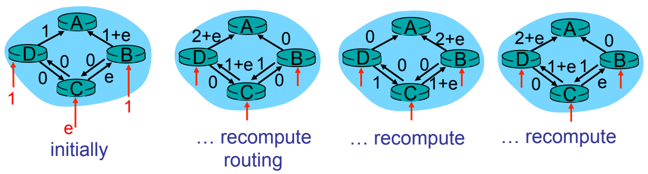
DV:
- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagates through network

Path Vector protocols

- Problem with D-V protocol:
  Path cost is "anonymous" single number; does not contain any topology information
- Path Vector protocol:
  - For each destination, advertise entire path (=sequence of node identifiers) to neighbours
  - Cost calculation can be done by looking at path
    - E.g., count number of hops on the path
  - Easy loop detection: Does my node ID already appear in the path?
- Not used very often
  - only in BGP …
  - … and BGP is much more complex than just paths

Dynamic (i.e., traffic-adaptive) routing?

- Dangerous: Oscillations possible!
- e.g., link cost = amount of carried traffic



- Why is this a bad thing?
  - Possibly sub-optimal choice of paths (as in example above)
  - Additional routing protocol traffic in network
  - Increased CPU load on routers
  - Inconsistent topology information during convergence: worst! (why?)

Inconsistent topology information

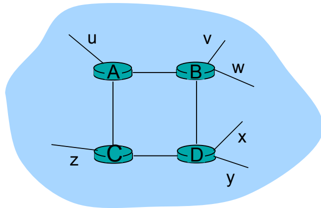
- Typical causes (not exhaustive)
  - One router finished with calculations, another one not yet
  - Relevant information has not yet reached entire network
    - LS: Broadcasts = fast
    - DV: Receive message, calculate table, inform neighbours: slow
  - DV: Count-to-infinity problem
  - LS: Different algorithm implementations!
  - LS: Problem if there is no clear rule for handling equal-cost routes
- Possible consequences?
  - Erroneously assuming some dst is not reachable
  - Routing loops
    - Think yourselves: What happens when there is a routing loop?

Intra-AS Routing

- Also known as Interior Gateway Protocols (IGP)
- Most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol — DV (typically small systems)
  - OSPF: Open Shortest Path First — hierarchical LS (typically medium to large systems)
  - IS-IS: Intermediate System to Intermediate System — hierarchical LS (typically medium-sized ASes)
  - (E)IGRP: (Enhanced) Interior Gateway Routing Protocol (Cisco proprietary) — hybrid of LS and DV

RIP (Routing Information Protocol)

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- Distance metric: # of hops (max = 15 hops, ∞ := 16)
- Sometimes still in use by very small ISPs



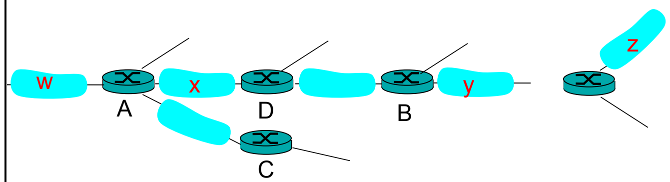
From router A to subnets:

destination	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP advertisements

- *distance vectors:* exchanged among neighbors every 30 sec via Response Message (also called advertisement)
- each advertisement: list of up to 25 destination subnets within AS

RIP: Example



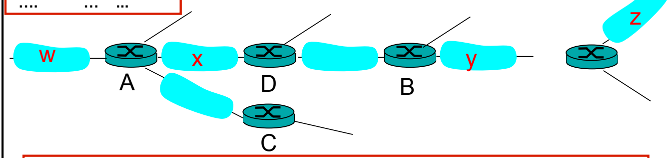
Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

Routing/Forwarding table in D

RIP: Example

Dest	Next	hops
w	-	1
x	-	1
z	C	4
....	...	...

Advertisement from A to D



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1
....	....	....

Routing/Forwarding table in D

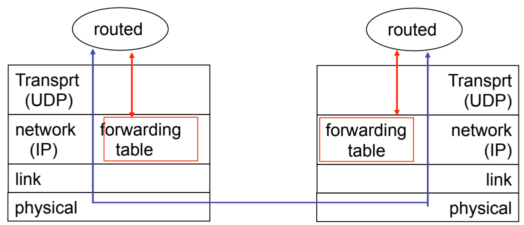
RIP: Link Failure and Recovery

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
- *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)

RIP Table processing

- RIP routing tables managed by **application-level** process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated



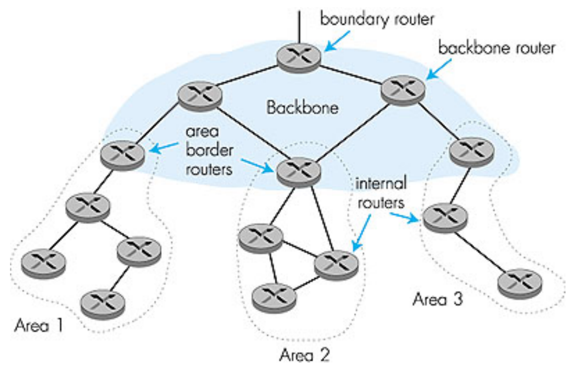
OSPF (Open Shortest Path First)

- "Open": publicly available (vs. vendor-specific, e.g., EIGRP = Cisco-proprietary)
- Uses Link State algorithm
  - LS packet dissemination (broadcasts)
  - Unidirectional edges (⇒costs may differ by direction)
  - Topology map at each node
  - Route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbour router
- Advertisements disseminated to entire AS (via flooding)
  - (exception: hierarchical OSPF, see next slides)
  - carried in OSPF messages directly over IP (rather than TCP or UDP)

OSPF "advanced" features (not in, e.g., RIP)

- Security: all OSPF messages authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed (only one path in RIP): *ECMP* (equal-cost multipath)
- For each link, multiple cost metrics for different Type of Service (TOS):
  e.g., satellite link cost set to "low" for best effort, but to "high" for real-time traffic like (telephony)
- Integrated unicast *and* multicast support:
  - Multicast OSPF (MOSPF)
  - Uses same topology data base as OSPF ➔ less routing protocol traffic
- Hierarchical OSPF in large domains
  - Drastically reduces number of broadcast messages

Hierarchical OSPF



Hierarchical OSPF

- OSPF *can* create a two-level hierarchy
  - (similar, but not identical to to inter-AS and intra-AS routing within an AS)
- Two levels: local *areas* and the *backbone*
  - Link-state advertisements only within local area
  - Each node has detailed area topology;
    but only knows coarse direction to networks in other areas (shortest path to border router)
- *Area border routers:* "summarize" distances to networks in own area; advertise distances to other Area Border and Boundary routers
- *Backbone routers:* run OSPF routing limited to backbone
- *Boundary routers:* connect to other Ases
  - "The outside world" ≈ another area

Hierarchical Routing in the Internet

Our routing study thus far = idealisation
- all routers identical
- network "flat"

… *not* true in practice!

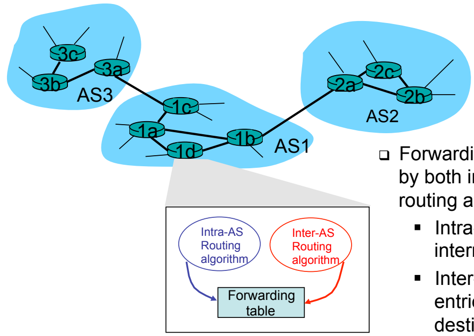
Scale = billions of destinations:
- Cannot store all destinations in routing tables
- Routing table exchange would swamp links
- Thousands of OSPF Areas? Would not scale!

Administrative autonomy
- Internet = network of networks
- Each network admin may want to control routing in its own network — no central administration!

Hierarchical Routing

- Aggregate routers into regions called "autonomous systems" (short: AS; plural: ASes)
  - One AS ≈ one ISP / university
- Routers in same AS run same routing protocol
  - = "intra-AS" routing protocol (also called "intradomain")
  - Routers in different ASes can run different intra-AS routing protocols
- ASes are connected: via gateway routers
  - Direct link to [gateway] router in another AS = "inter-AS" routing protocol (also called "interdomain")
  - Warning: Non-gateway routers need to know about inter-AS routing as well!

Interconnected ASes



- Forwarding table configured by both intra- *and* inter-AS routing algorithm:
  - Intra-AS sets entries for internal destinations
  - Inter-AS *and* intra-AS set entries for external destinations

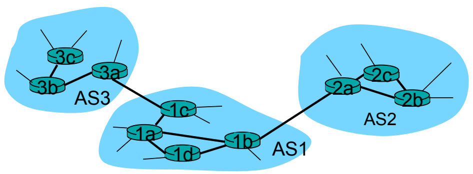
Inter-AS tasks

- Suppose router in AS1 receives datagram destined outside of AS1:
  - Router should forward packet to gateway router
  - …but to which one?

AS1 must:

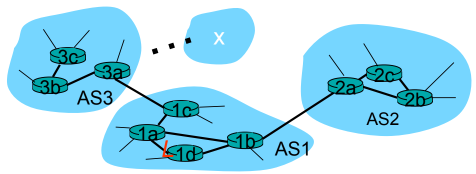
1. learn which destinations are reachable through AS2, which through AS3
2. propagate this reachability info *to all* routers in AS1 (i.e., not just the gateway routers)

Job of inter-AS routing!



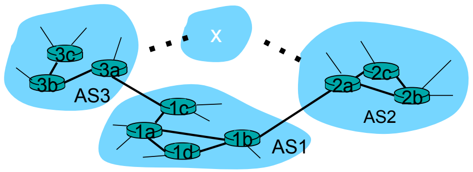
Example: Setting forwarding table in router 1d

- Suppose AS1 learns (via inter-AS protocol) that subnet *x* is reachable via AS3 (gateway 1c) but not via AS2.
- Inter-AS protocol propagates reachability info to all internal routers.
- Router 1d determines from intra-AS routing info that its interface *I* (i.e., interface to 1a) is on the least cost path to 1c.
  - installs forwarding table entry *(x,I)*



Example: Choosing among multiple ASes

- Now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- To configure forwarding table, router 1d must determine towards which gateway it should forward packets for destination x.
  - "Do we like AS2 or AS3 better?"
  - Also the job of inter-AS routing protocol!

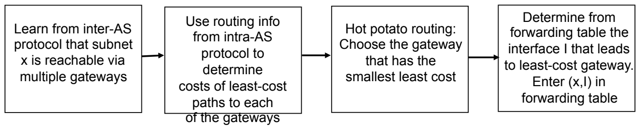


Interplay of inter-AS and intra-AS routing

- Inter-AS routing
  - Only for destinations outside of own AS
  - Used to determine gateway router
  - Also: Steers transit traffic (from AS *x* to AS *y* via our own AS)
- Intra-AS routing
  - Used for destinations within own AS
  - Used to reach gateway router for destinations outside own AS
- ⇒ Often, routers need to run *both* types of routing protocols… even if they are not directly connected to other ASes!

Example: Choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x.
  - this is also job of inter-AS routing protocol!
- hot potato routing: send packet towards closest of two routers.



Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *The* de facto standard for inter-AS routing
- BGP provides each AS a means to:
  1. Obtain subnet reachability information from neighbouring ASes.
  2. Propagate reachability information to all AS-internal routers.
  3. Determine "good" routes to subnets based on reachability information and policy.
- Allows an AS to advertise the existence of an IP prefix to rest of Internet: *"This subnet is here"*

BGP basics

- Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: BGP sessions
  - BGP sessions need not correspond to physical links!
- When AS2 advertises an IP prefix to AS1:
  - AS2 *promises* it will forward IP packets towards that prefix
  - AS2 can aggregate prefixes in its advertisement (e.g.: 10.11.12.0/26, 10.11.12.64/26, 10.11.12.128/25 into 10.11.12.0/24)

How does BGP work?

- BGP = "path++" vector protocol
- BGP messages exchanged using TCP
  - Possible to run eBGP sessions not on border routers
- BGP Message types:
  - OPEN: set up new BGP session, after TCP handshake
  - NOTIFICATION: an error occurred in previous message → tear down BGP session, close TCP connection
  - KEEPALIVE: "null" data to prevent TCP timeout/auto-close; also used to acknowledge OPEN message
  - UPDATE:
    - Announcement: inform peer about new / changed route to some target
    - Withdrawal: (inform peer about non-reachability of a target)

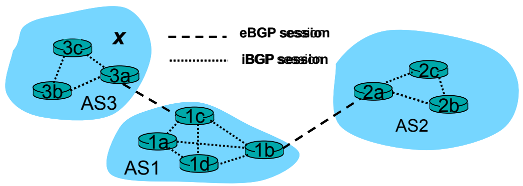
BGP updates

- Update (Announcement) message consists of
  - Destination (IP prefix)
  - AS Path (=Path vector)
  - Next hop (=IP address of our router connecting to other AS)
- …but update messages also contain a lot of further attributes:
  - Local Preference: used to prefer one gateway over another
  - Origin: route learned via { intra-AS | inter-AS | unknown }
  - MED, Community, …

⇒ Not a pure path vector protocol: More than just the path vector

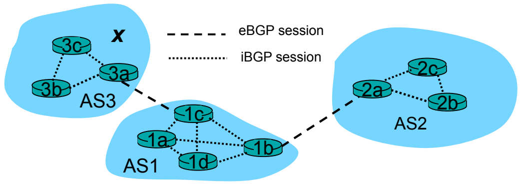
eBGP and iBGP

- External BGP: between routers in *different* ASes
- Internal BGP: between routers in *same* AS
  - Remember: In spite of intra-AS routing protocol, *all* routers need to know about external destinations (not only border routers)
- No different protocols—just slightly different configurations!



Distributing reachability info

- Using eBGP session between 3a and 1c, AS3 sends reachability info about prefix *x* to AS1.
  - 1c can then use iBGP to distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- When router learns of new prefix *x*, it creates entry for prefix in its forwarding table.



Path attributes & BGP routes

- Advertised prefix includes [many] BGP attributes
  - prefix + attributes = “route”
- Most important attributes:
  - AS-PATH: contains ASs through which prefix advertisement has passed: e.g, AS 67, AS 17
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS (may be multiple links from current AS to next-hop-AS)
- When gateway router receives route advertisement, it uses an import policy to accept/decline the route
  - More on this later

AS Numbers

- How do we express a BGP path?
- ASes identified by *AS Numbers* (short: ASN)
  Examples:
  - Leibnitz-Rechenzentrum = AS12816
  - Deutsche Telekom = AS3320
  - AT&T = AS7018, AS7132, AS2685, AS2686, AS2687
- ASNs used to be 16bit, but can be 32bit nowadays
  - Issues with 16bit ASNs on old routers
- ASN assignment: similar to IP address space
  - ASN space administered IANA
  - Local registrars, e.g., RIPE NCC in Europe

BGP route selection

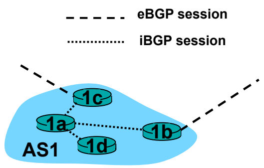
- Router may learn about more than 1 route to some prefix
  ⇒ Router must select the best route.
- Elimination rules (simplified):
  1. Local preference value attribute: policy decision
  2. Shortest AS-PATH
  3. Closest NEXT-HOP router: hot potato routing
  4. Additional criteria

iBGP scalabtility problem

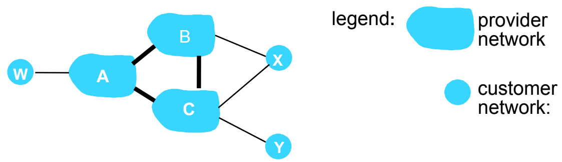
- Every router in AS should know external routes
  - Not only local neighbours, but also neighbours connected at other routers
  - ⇒ Many/all routers in AS have to run BGP sessions
- Need to select best inter-AS routes
  - ⇒ Routers need to exchange routing information via iBGP
- O(n) BGP routers ⇒ O(n^2) iBGP sessions ⚡ ⚡ ⚡
  - This does not scale!

Solution: BGP Route Reflectors (RR)

- **Idea:**
  - One special router = Route Reflector (RR)
  - Every **e**BGP router sends routes learned from **e**BGP via iBGP to RR
  - RR collects routes, may do policing
  - RR distributes routes to all other BGP routers in AS via iBGP
- Result: O(n) BGP routers, O(n) BGP sessions ☺

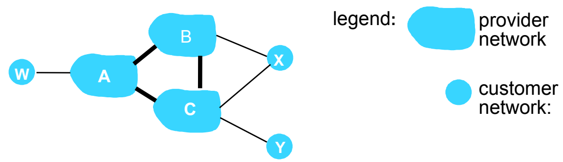


BGP routing policy



- A,B,C are provider networks
- X,W,Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

BGP routing policy (2)



- A advertises path AW to B
- B advertises path BAW to X
- Should B advertise path BAW to C?
  - No way! B gets no “revenue” for routing CBAW since neither W nor C are B’s customers
  - B wants to force C to route to w via A
  - B wants to route *only* to/from its customers!

Why different Intra- and Inter-AS routing?

Policy:
- Inter-AS: admin wants control over how its traffic routed, who routes through its net.
- Intra-AS: single admin, so no policy decisions needed

Scale:
- hierarchical routing saves table size, reduced update traffic

Performance:
- Intra-AS: can focus on performance
- Inter-AS: policy may dominate over performance

Business relationships

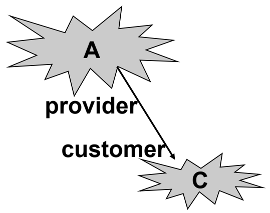
- Internet = network of networks (ASes)
  - Many thousands of ASes
  - Not every network connected to every other network
  - BGP used for routing between ASes
- Differences in economical power/importance
  - Some ASes huge, intercontinental (AT&T, Cable&Wireless)
  - Some ASes small, local (e.g., München: M"Net, SpaceNet)
- Small ASes customers of larger ASes: Transit traffic
  - Smaller AS pays for connecting link + for data = buys transit
  - Business relationship = customer—provider
- Equal-size/-importance ASes
  - Usually share cost for connecting link[s]
  - Business relationship = peering (no transit traffic)
- Warning: peering ("equal-size" AS) ≠ peers of a BGP connection (also may be customer or provider) ≠ peer-to-peer network

Business and policy routing (1)

- Basic principle #1
  - Prefer routes that incur financial gain
- Basic principle #2
  - Announce routes that incur financial gain if others use them
    - Others = customers
  - Announce routes that reduce costs if others use them
    - Others = peers
  - Do not announce routes that incur financial loss (…as long as alternative paths exist)

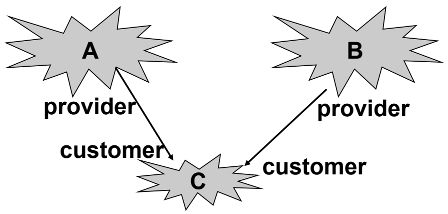
Business and policy routing (2)

- A tells C all routes it uses to reach other ASes
  - The more traffic comes from C, the more money A makes



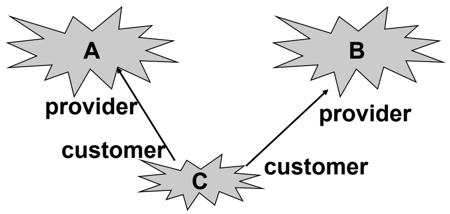
Business and policy routing (3)

- A and B tell C all routes they use to reach other ASes
  - The more traffic flows from C to A, the more money A makes
  - The more traffic flows from C to B, the more money B makes



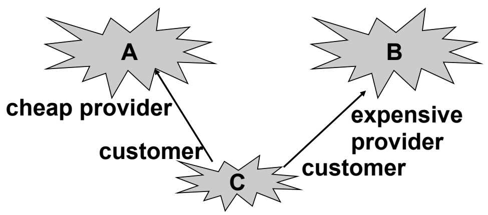
Business and policy routing (4)

- C tells A its own prefixes; C tells B its own prefixes
  - C wants to be reachable from outside
- C does not tell A routes learned from/via B
  C does not tell B routes learned from/via A
  - C does not want to pay money for traffic …↔A ↔C ↔B ↔…



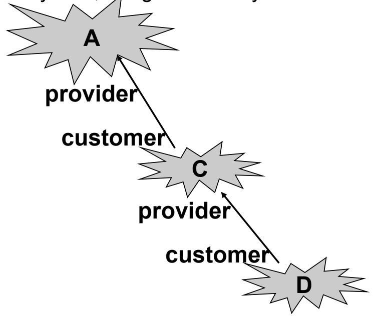
Business and policy routing (5): AS path prepending

- C tells A its prefixes
- C may tell B its own prefixes
  - …but inserts “C” multiple times into AS path
  - Result: Route available, but longer path = less attractive
  - Technique is called *AS path prepending*



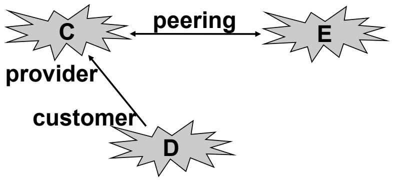
Business and policy routing (6)

- What should C announce here?
  - C tells A about its own prefixes
  - C tells A about its route to D’s prefixes:
    loses money to A, but gains money from D



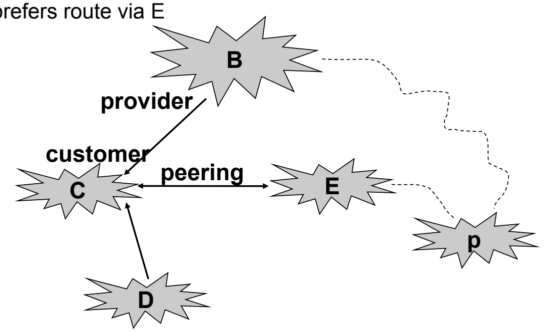
Business and policy routing (7)

- What should C announce here?
  - C tells peering partner E about its own prefixes and route to D: no cost on link to E, but gains money from D



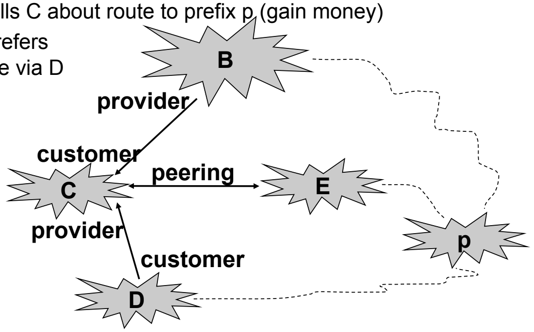
Business and policy routing (8a)

- Which route should C select?
  - B tells C about route to prefix p (lose money)
  - E tells C about route to prefix p (± 0)
  - C prefers route via E



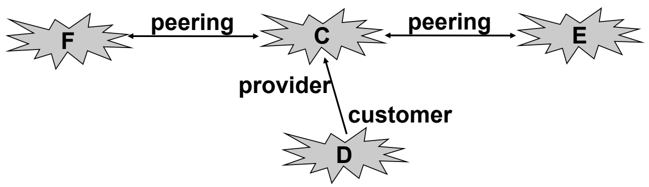
Business and policy routing (8b)

- What should C announce here?
  - B tells C about route to prefix p (lose money)
  - E tells C about route to prefix p (± 0)
  - D tells C about route to prefix p (gain money)
  - C prefers route via D



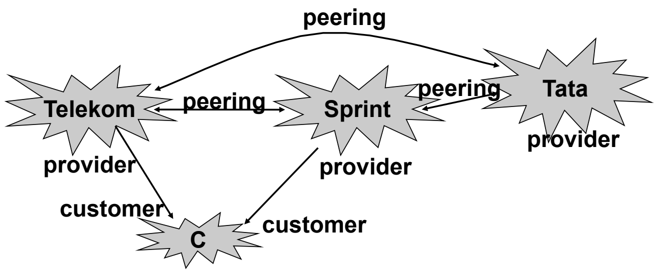
Business and policy routing (9)

- What should C announce here?
  - C announces to F and E: its own prefixes and D's routes
  - C does *not* announce to E: routes going via F
    - Otherwise: E could send traffic towards F but wouldn't pay anything, F wouldn't pay either, and C's network gets loaded with additional traffic
  - C does *not* announce to F: routes going via E
    - Same reason



Business and policy routing (10): “Tiers” / “DFZ”

- Big players have no providers, only customers and peers
  - “Tier-1” providers
  - or “Default-Free Zone” (have no default route to “provider”)
- Each Tier-1 peers with each other



Tier-1, Tier-2, Tier-3 etc.

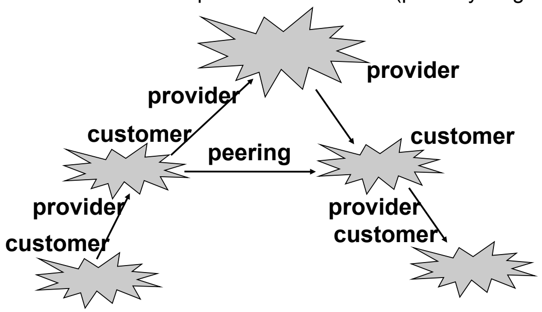
- Tier-1/DFZ = only peerings, no providers
- Tier-2 = only peerings and one or more Tier-1 providers
- Tier-3 = at least one Tier-2 as a provider
- Tier-*n* = at least one Tier-(n-1) provider
  - defined recursively
  - $n \geq 4$: Rare in Western Europe, North America, East Asia

- “Tier-1.5” = almost a Tier-1 but pays money for *some* links
  - Example: Deutsche Telekom used to pay money to Sprint, but is now Tier-1
  - Marketing purposes: Tier-1 sounds better

Valley-free routing

Results: Packets always travel…

1. upstream: sequence of C→P links (possibly length = 0)
2. then possibly across *one* peering link
3. then downstream: sequence of P→C links (possibly length = 0)



Siblings

- Not everything is provider/customer or peering
- Sibling = mutual transit agreement
  - Provide connectivity to the rest of the Internet for each other
  - ≈ very extensive peering
- Examples
  - Two small ASes close to each other that cannot afford additional Internet services
  - Merging two companies
    - Merging two ASes into one = difficult,
    - Keeping two ASes and exchaning everything for free = easier
  - Example: AT&T has five different AS numbers (7018, 7132, 2685, 2686, 2687)

To peer or not to peer, this is the question

Peer:

- Reduce upstream costs
- Possibly increases performance
- Perhaps only way to connect your customers (Tier-1)

Don't peer

- You don't gain any money
- Peers are usually your competitors
- What if it turns out the peering is more beneficial to you peer than to you? ⇒ Require periodic regenotiation

Where to peer

- Private peering
  - "Let's use a cable from your server room to our server room"
- At public peering locations (Internet Exchange Point, IX, IXP)
  - "A room full of switches that many providers connect to"
  - Examples:
    - DE-CIX Frankfurt (purportedly largest in world)
    - AMS-IX Amsterdam
    - LINX London
    - MSK-IX Moscow

BGP/Policy routing Summary

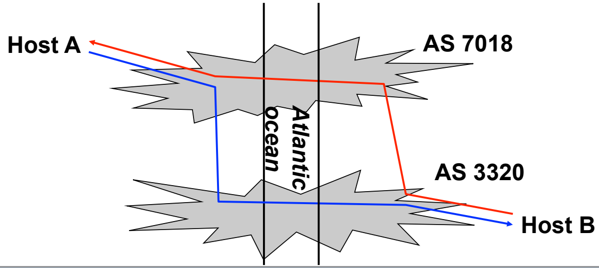
- Import Policy = Which routes to use
  - Select path that incurs most money
  - Special/political considerations (e.g., Iranian AS does not want traffic to pass Israeli AS; other kinds of censorship)
- Export Policy = Which routes to propagate to other ASes
  - Not all possible routes propagate:
    Export only…
    - If it incurs revenue
    - If it reduces cost
    - If it is inevitable
  - Propagation driven by business considerations
  - Propagation not driven by technical considerations!
    Example: Slower route via peer may be preferred over faster route via provider

BGP policy routing: Technical summary

1. Receive BGP update
2. Apply import policies
   - Filter routes
   - Tweak attributes (advanced topic…)
3. Best route selection based on attribute values
   - Install forwarding tables entries for best routes
   - Possibly transfer to Route Reflector
4. Apply export policies
   - Filter routes
   - Tweak attributes
5. Transmit BGP updates

Hot-potato routing

- Interaction between Inter-AS and Intra-AS routing
  - Business: If traffic is destined for other AS, get rid of it ASAP
  - Technical: Intra-AS routing finds shortest path to gateway
- Multiple transit points ⇒ asymmetrical routing
  - Asymmetrical paths are very common on the Internet



Routing: Optimization purposes

- Inter-AS routing
  - Optimality = select route with highest revenue/least loss
- Intra-AS routing
  - Optimality = configure routing such that network can host as much traffic as possible

Traffic Engineering

1. Collect traffic statistics: Traffic Matrix
   - How much traffic flowing from A to B?
   - Difficult to measure! (drains router performance); thus often estimated: research area
2. Optimize routing
   - E.g., calculate good choice of OSPF weights
   - Goal: minimize maximum link load in entire network; keep average link load below 50%
     - why? Fractal TCP traffic leads to spikes!
3. Deploy new routing
   - Performance may deteriorate during update
   - E.g., routing loops during OSPF convergence

Dynamic traffic engineering

Why not dynamic?

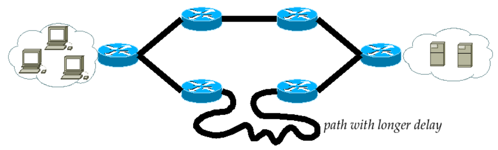
- Routing loops during convergence
- Packet reordering:
  - Packet P1 arrives later than Packet P2
  - TCP will think that P1 got lost! ⇒ congestion control!
- Prone to oscillations and chaotic behaviour
  - Bad experiences in the ARPANET
  - Ex.: Route A congested, route B free
    → Everyone switches from A to B
    → Route A free, route B congested → …
- Actually, a difficult problem
  - Stale information
  - Interaction with TCP congestion control
  - Interaction with dynamic TE mechanisms in other ASes
- Thus: Congestion control in end hosts (TCP), not in network

Multipath routing

- Routing = finding best-cost route
- What if more than one exists?
- Some routing protocols allow Equal-Cost Multipath (ECMP) routing, e.g., OSPF
  - ≥ 2 routes of same cost exist to destination prefix? → Evenly distribute traffic across these routes

Multipath routing: TCP problem

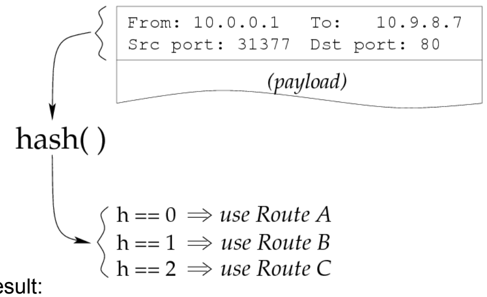
- How to distribute traffic? Naïve approaches:
  - Round-robin
  - Distribute randomly
- Equal cost does not mean equal latency:



- Again: Problem with TCP = Packet reordering!
  - Packets sent: P1, P2
  - Packets received: P2, P1
  - Receiver receives P2 → believes P1 to be lost → triggers congestion control mechanisms → performance degrades

Multipath routing: Solution

- Hash “randomly”…
- …but use packet headers as “random” values:



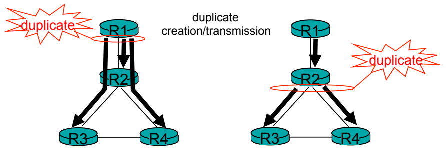
- Result:
  - Packets from same TCP connection yield same hash value
  - No reordering possible

Network Layer: Weaknesses and shortcomings (3)

- Manageability
  - Routing = complex to set up
  - Even more complex to manage/debug
    - What/who caused the error? – Difficult to answer!
- End hosts: increasingly mobile
  - WLAN → UMTS? = IP address changes!
- Multicast: works in theory and lab -- but is not deployed
- Quality of service
  - Different applications have different service demands
    - File transfer: max bandwidth
    - Chat, VoIP, games: min delay
    - E-Mail: min cost
  - QoS = different *classes of service*
  - Works in theory and lab – but is not deployed (same reasons as with multicast)

Broadcast Routing

- Deliver packets from source to all other nodes
- Source duplication is inefficient:



source duplication

in-network duplication

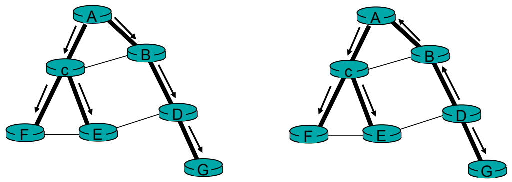
- source duplication: how does source determine recipient addresses?

In-network duplication

- flooding: when node receives brdcst pckt, sends copy to all neighbors
  - Problems: cycles & broadcast storm
- controlled flooding: node only brdcsts pkt if it hasn't brdcst same packet before
  - Node keeps track of pckt ids already brdcsted
  - Or reverse path forwarding (RPF): only forward pckt if it arrived on shortest path between node and source
- spanning tree
  - No redundant packets received by any node

Spanning Tree

- First construct a spanning tree
- Nodes forward copies only along spanning tree

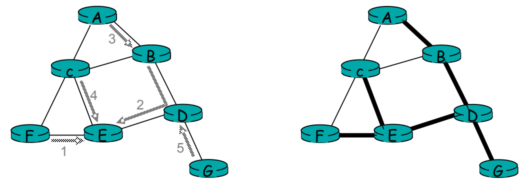


(a) Broadcast initiated at A

(b) Broadcast initiated at D

Spanning Tree: Creation

- Center node
- Each node sends unicast join message to center node
  - Message forwarded until it arrives at a node already belonging to spanning tree

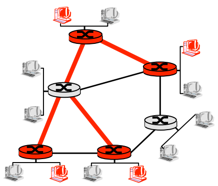


(a) Stepwise construction of spanning tree

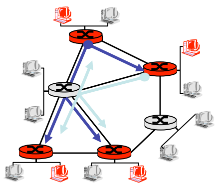
(b) Constructed spanning tree

Multicast Routing: Problem Statement

- ***Goal:*** find a tree (or trees) connecting routers having local mcast group members
  - *tree:* not all paths between routers used
  - *source-based:* different tree from each sender to rcvrs
  - *shared-tree:* same tree used by all group members



Shared tree



Source-based trees

Approaches for building mcast trees

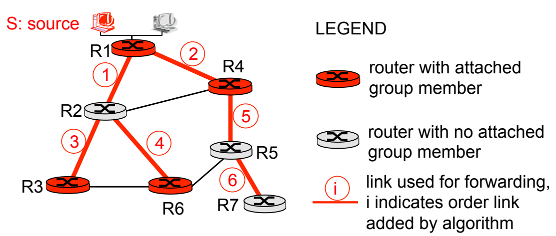
Approaches:

- source-based tree: one tree per source
  - shortest path trees
  - reverse path forwarding
- group-shared tree: group uses one tree
  - minimal spanning (Steiner)
  - center-based trees

…we first look at basic approaches, then specific protocols adopting these approaches

Shortest Path Tree

- mcast forwarding tree: tree of shortest path routes from source to all receivers
  - Dijkstra's algorithm

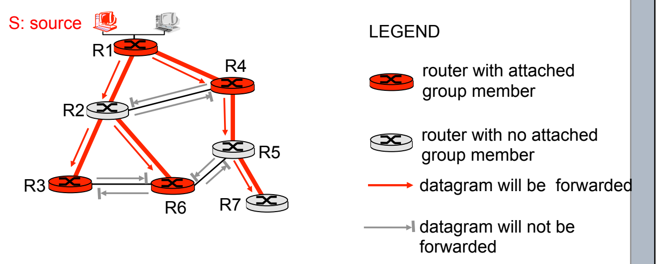


Reverse Path Forwarding

- rely on router's knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

if (mcast datagram received on incoming link on shortest path back to center)
then flood datagram onto all outgoing links
else ignore datagram

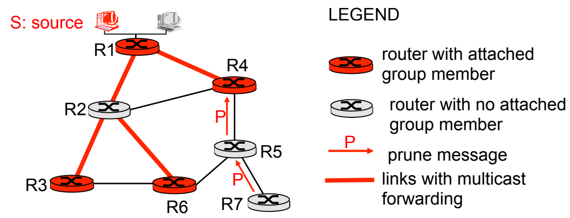
Reverse Path Forwarding: example



- result is a source-specific *reverse* SPT
  - may be a bad choice with asymmetric links

Reverse Path Forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - "prune" msgs sent upstream by router with no downstream group members



Shared-Tree: Steiner Tree

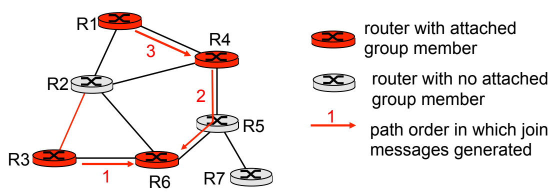
- Steiner Tree: minimum cost tree connecting all routers with attached group members
- problem is NP-complete
- excellent heuristics exists
- not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave

Center-based trees

- single delivery tree shared by all
- one router identified as *"center"* of tree
- to join:
  - edge router sends unicast *join-msg* addressed to center router
  - *join-msg* "processed" by intermediate routers and forwarded towards center
  - *join-msg* either hits existing tree branch for this center, or arrives at center
  - path taken by *join-msg* becomes new branch of tree for this router

Center-based trees: an example

Suppose R6 chosen as center:



Internet Multicasting Routing: DVMRP

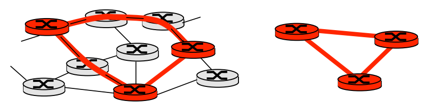
- DVMRP: distance vector multicast routing protocol, RFC1075
- *flood and prune:* reverse path forwarding, source-based tree
  - RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs

DVMRP: continued…

- *soft state:* DVMRP router periodically (1 min.) "forgets" branches are pruned:
  - mcast data again flows down unpruned branch
  - downstream router: reprune or else continue to receive data
- routers can quickly regraft to tree
  - following IGMP join at leaf
- odds and ends
  - commonly implemented in commercial routers
  - Mbone routing done using DVMRP

Tunneling

Q: How to connect "islands" of multicast routers in a "sea" of unicast routers?



physical topology

logical topology

- multicast datagram encapsulated inside "normal" (non-multicast-addressed) datagram
- normal IP datagram sent through "tunnel" via regular IP unicast to receiving multicast router
- receiving multicast router unencapsulates to get multicast datagram

PIM: Protocol Independent Multicast

- not dependent on any specific underlying unicast routing algorithm (works with all)
- two different multicast distribution scenarios :

Dense:
- group members densely packed, in "close" proximity.
- bandwidth more plentiful

Sparse:
- # networks with group members small wrt # interconnected networks
- group members "widely dispersed"
- bandwidth not plentiful

Consequences of Sparse—Dense Dichotomy:

Dense
- group membership by routers *assumed* until routers explicitly prune
- *data-driven* construction on mcast tree (e.g., RPF)
- bandwidth and non-group-router processing *profligate*

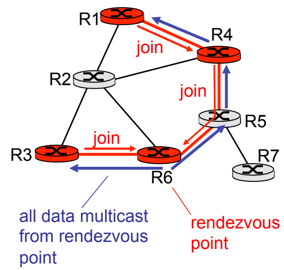
Sparse:
- no membership until routers explicitly join
- *receiver- driven* construction of mcast tree (e.g., center-based)
- bandwidth and non-group-router processing *conservative*

PIM – Dense Mode

flood-and-prune RPF, similar to DVMRP but
- underlying unicast protocol provides RPF info for incoming datagram
- less complicated (less efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm
- has protocol mechanism for router to detect it is a leaf-node router

PIM – Sparse Mode

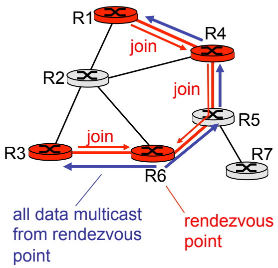
- center-based approach
- router sends *join* msg to rendezvous point (RP)
  - intermediate routers update state and forward *join*
- after joining via RP, router can switch to source-specific tree
  - increased performance: less concentration, shorter paths



PIM – Sparse Mode

sender(s):

- unicast data to RP, which distributes down RP-rooted tree
- RP can extend mcast tree upstream to source
- RP can send *stop* msg if no attached receivers
  - "no one is listening!"



Routing: Weaknesses and shortcomings (1)

- No network congestion control:
  Dynamic routing / dynamic traffic engineering = difficult!
  - Tried out in ARPANET: Oscillations everywhere
  - Today: Interaction with TCP congestion control feedback loop → even worse!
- Convergence speed (link/router failures)
  - OSPF: 200ms … several seconds
    - Routing loops may occur during convergence = black holes
  - BGP: seconds to several minutes!
    - Never really converges: there's always something going on
- More and more prefixes in routing tables
  - 300,000 and growing

Routing: Weaknesses and Shortcomings (2)

- Routing = destination-based
  - No complete choice of paths
  - Restricts solutions for traffic engineering
- Security
  - Denial of service attacks:
    Undesired traffic dropped at receiver, not in network
  - Other attacks: hard to trace, no sender signature
  - BGP misconfiguration can create havoc
    - Example: Pakistan created YouTube black hole
  - BGP implementation errors can wreak havoc
    - Example: Czech provider creates huge AS path
      => Many routers crash world-wide
      => Wildly oscillates
  - Question: What about concerted attack on BGP…? ☹ ☹ ☹