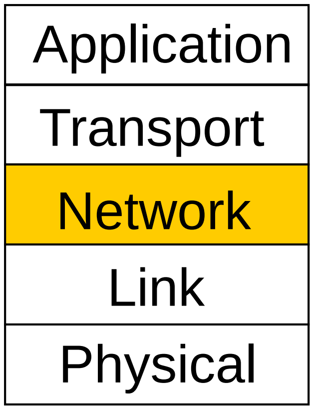
CSEP 561 – Routing

David Wetherall
djw@cs.washington.edu

Routing

- Focus:
  - How to find and set up paths through networks

- Distance-vector and link-state
- Shortest path routing
- Key properties of schemes
- Multicast

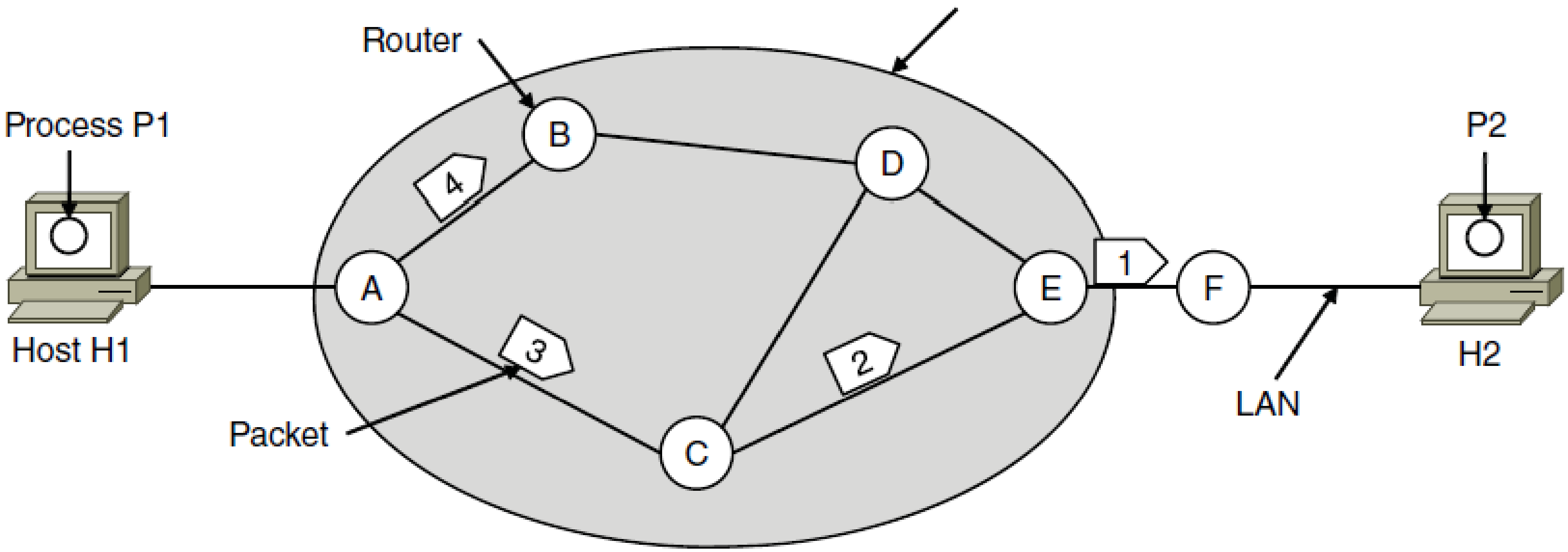


Routing versus Forwarding

- Routing is the process by which all nodes exchange control messages to calculate the *routes* packets will follow
  - Distributed process with *global* goals; emphasis is *correctness*
  - Nodes build a routing table that models the global network

- Forwarding is the process by which a node examines packets and sends them along their *paths* through the network
  - Involves *local* decisions; emphasis is *efficiency*
  - Nodes distill a forwarding table from their routing table (keyed by packet attributes, e.g., address) that gives the *next hop*

Datagram Forwarding



A's table (initially)

A	⊠
B	B
C	C
D	B
E	C
F	C

Dest. Line

A's table (later)

A	⊠
B	B
C	C
D	B
E	D
F	D

C's Table

A	A
B	A
C	⊠
D	E
E	E
F	E

E's Table

A	C
B	D
C	C
D	D
E	⊠
F	F

What is a “best” path anyhow?

- Ideally paths that:
  - Are as direct as possible (low latency)
  - Carry as much traffic as the network will fit (high bandwidth)
  - Carry traffic well for all of the nodes (fairness)

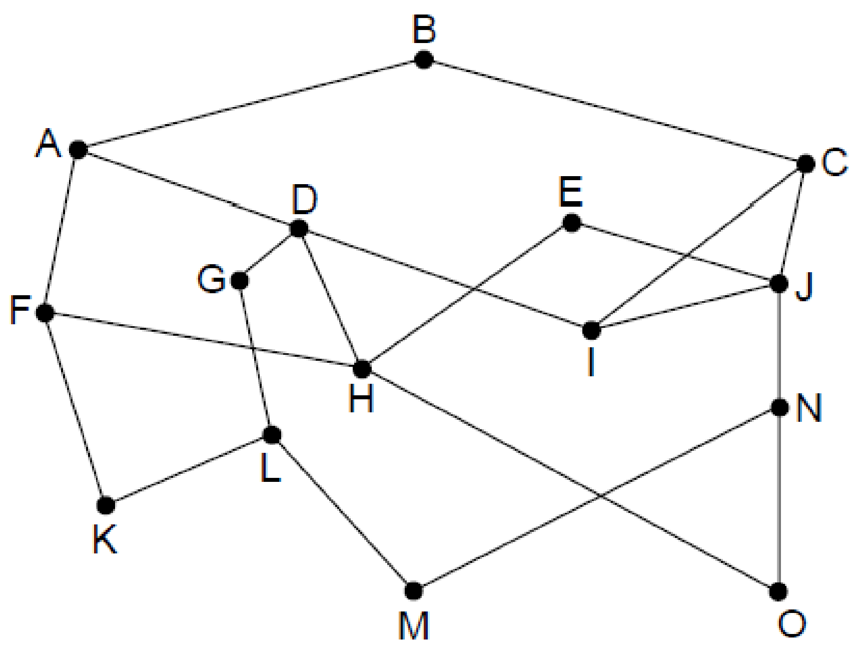
- This is a resource allocation problem with multiple constraints. Depends on topology and who sends how much traffic to who, which changes over time. Yikes!

- We want a simple, distributed solution

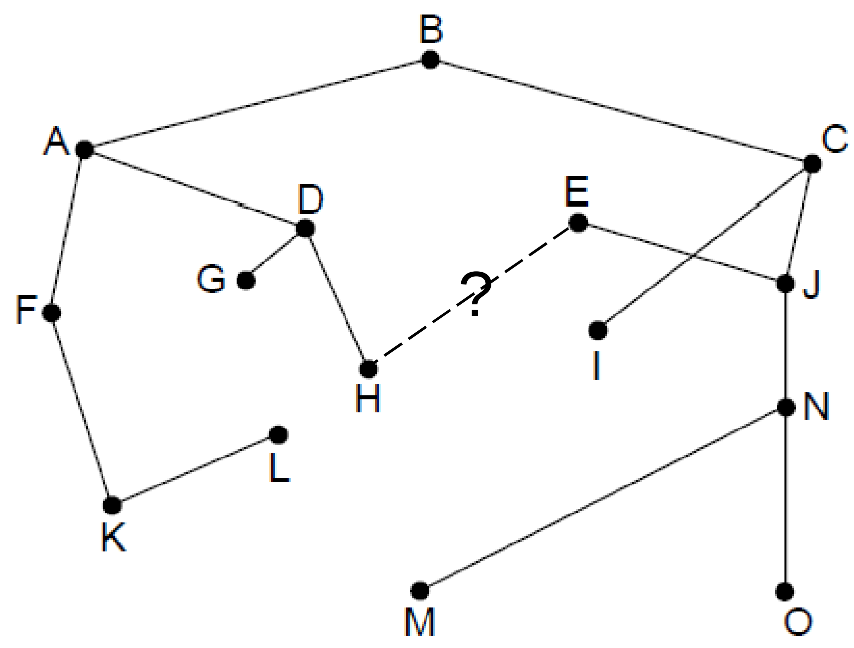
Lowest cost ("shortest path") routes

- Compute paths independently for different node pairs
  - Assign a cost or weight to each link
  - Find lowest total weight path between source/dest
- Typically costs are fixed
  - Does not take hotspots into account
  - Has simple subset optimality properties
- Costs usually set as a function of bandwidth and delay
  - Can tweak (traffic engineering) to match traffic to topology
  - More direct paths help with low latency and high bandwidth, so does a reasonable overall job

Sink trees



Network



Sink Tree for B

Equal-cost multi-path (ECMP)

- Generalization for load balancing
  - Allow multiple paths if they have the same lowest cost
- Single path lowest cost routing produces a spanning tree
- ECMP produces a directed acyclic graph
  - Still no possibility of loops
  - Simple for nodes: just keep a list of next hops
- Q: How to map traffic to the multiple paths?

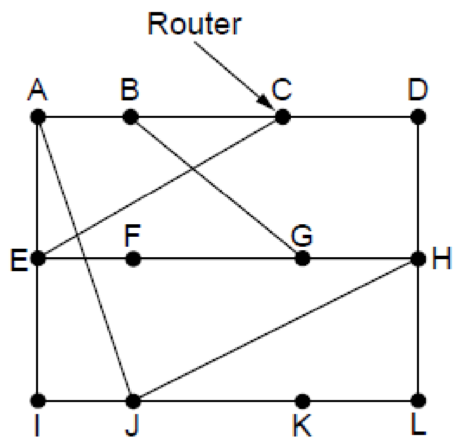
Two datagram routing methods

- Distance-vector and Link-state
- Scenario:
  - You’re driving from Seattle to Boston.
  - Gas station attendants in each city will tell you which way to go next to head towards your destination. But how do they know?
- Link-state method:
  - Every attendant shares their local cities with all others, makes their own map of the US, and consults it to direct you
- Distance-vector method:
  - Every attendant tells their neighbors the mileage to all cities and keeps the best directions to direct you

Distance Vector Algorithm

- Each router maintains a vector of costs to all destinations as well as routing table giving next hops
  - Initialize neighbors with known cost, others with infinity
- Periodically send copy of distance vector to neighbors
- On reception of a vector, if your neighbor's path to a destination plus cost to that neighbor cost is better
  - Update the cost and next-hop in your outgoing vectors
- Assuming no changes, will converge to shortest paths

DV Example



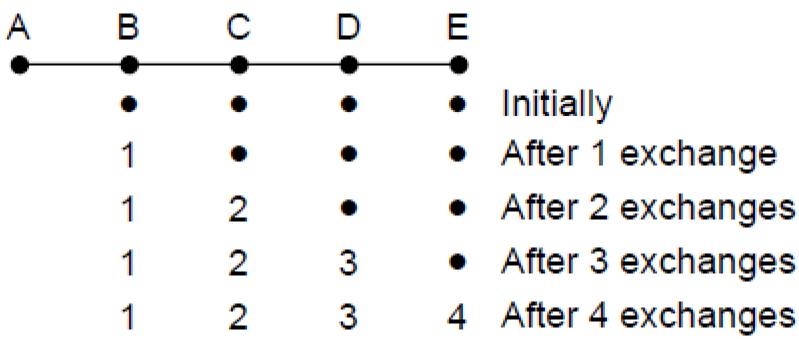
(a)

To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

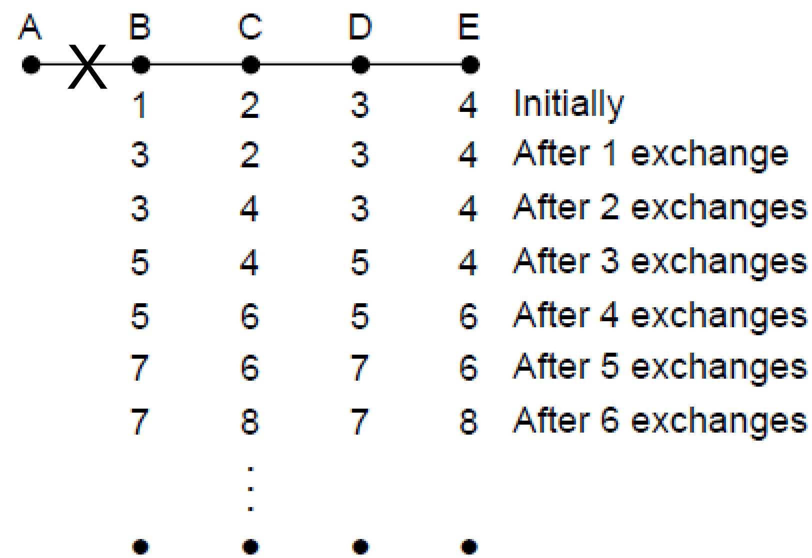
(b)

DV problem -- dynamics



(a)

Desired convergence



(b)

“Count to infinity scenario”

DV problem -- dynamics

- Good news (better routes) propagate quickly
- Bad news (failures) propagate slowly
  - inferred by exploration
- Leads to “count to infinity” loops
  - Many heuristics (split horizon, poison reverse)
  - Takes ordered updates to eliminate (e.g., EGIRP uses diffusing computations) that are complicated and slow convergence
  - No great solutions

- No longer widely used except for resource constrained or legacy networks.

Routing Information Protocol (RIP)

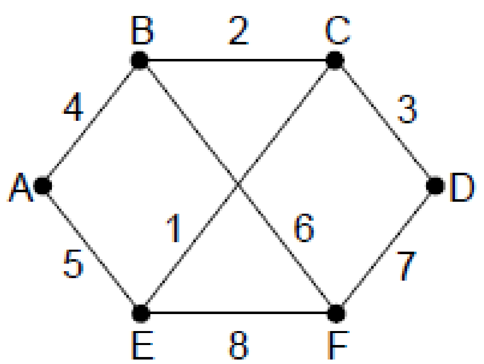
- DV protocol with hop count as metric
  - Infinity value is 16 hops; limits network size
  - Includes split horizon with poison reverse
- Routers send vectors every 30 seconds
  - With triggered updates for link failures
  - Time-out in 180 seconds to detect failures

- RIPv1 specified in RFC1058
  - www.ietf.org/rfc/rfc1058.txt
- RIPv2 (adds authentication etc.) in RFC1388
  - www.ietf.org/rfc/rfc1388.txt

Link State Routing

- Same assumptions/goals, but different idea than DV:
  - Tell all routers the topology and have each compute best paths
  - Two phases:
    1. Topology dissemination (flooding)
    2. Shortest-path calculation (Dijkstra's algorithm)
- Why?
  - In DV, routers hide their computation, making it difficult to decide what to use when there are changes
  - With LS, faster convergence and hopefully better stability
  - It is more complex though …

LS example database



(a)

Link State Packets

A		B		C		D		E		F	
Seq.		Seq.		Seq.		Seq.		Seq.		Seq.	
Age		Age		Age		Age		Age		Age	
B	4	A	4	B	2	C	3	A	5	B	6
E	5	C	2	D	3	F	7	C	1	D	7
		F	6	E	1			F	8	E	8

(b)

- Q: what is the flooding rule to build the database?
- Q: how are shortest paths computed from the database?

Open Shortest Path First (OSPF)

- Widely-used Link State protocol today; see also ISIS
- Basic link state algorithms plus many features:
  - Authentication of routing messages
  - Extra hierarchy: partition into routing areas
  - Load balancing: multiple equal cost routes

Routing – desirable properties

- Correctness
- Network efficiency
- Network fairness

- Rapid convergence
  - To correct routes that are stable after changes, with minimal transient loss
- Scalability
  - Of messages and router state
  - Particularly an issue for large, mobile, or multicast networks

Example

Property	Distance Vector	Link State
Correctness	Yes - Distributed Bellman Ford	Yes - Replicated shortest path
Efficiency	Approx- Least cost paths	Approx - Least cost paths
Fairness	Approx - Least cost paths	Approx - Least cost paths
Convergence	Slow – many exchanges	Fast – prop plus compute
Scalability	Good – O(1) per node/link	Moderate – at least O(edges)

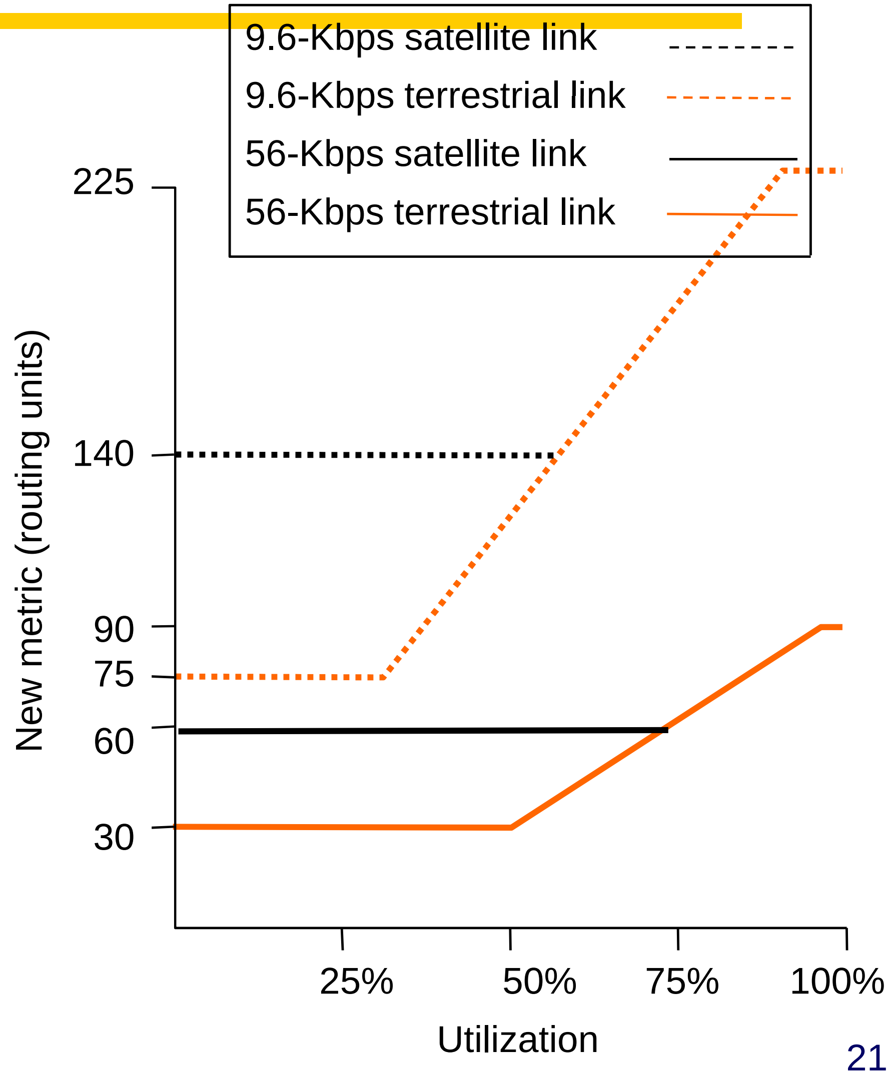
Resource allocation timescales today

- From fast (very reactive) to slow (carefully planned)
  - Use of different timescales largely decouples mechanisms

- Congestion control
  - Adapts to packet loss; slows source
- Routing
  - Adapts to failures; finds paths with connectivity
- Traffic engineering
  - Typically manual route adjustments for cost/performance
- Provisioning
  - Build out network to match traffic workload

What didn't work: Revised ARPANET Cost Metric

- Based on load and link
- Variation limited (3:1) and change damped
- Capacity dominates at low load; we only try to move traffic if high load
- Early attempt to use routing for congestion control – not stable



Delivery models

- Unicast
  - single sender to single receiver
- Broadcast
  - Single sender to all receivers
- Multicast
  - Single sender to multiple (but not all) receivers (in a group)
- Anycast
  - Single sender to nearest receiver in a set

Broadcast with RPF

- Reverse Path Forwarding (RPF)
  - Simplest broadcast using unicast tables
- Given broadcast from source S. At each router:
  - Look up outgoing interface O to reach S.
  - If packet arrives on O then forward to all other interfaces

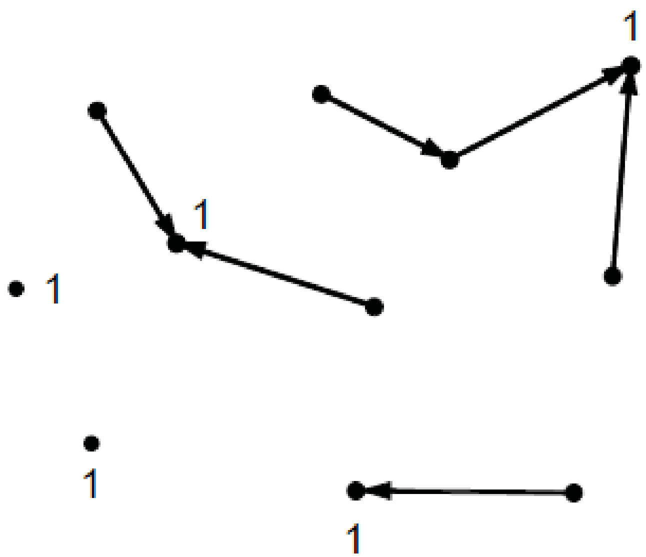
- Q: What assumptions does this make?
- Q: How does this compare to flooding?

- Alternative is construction of per-source broadcast trees
  - Often done in practice; not a big deal

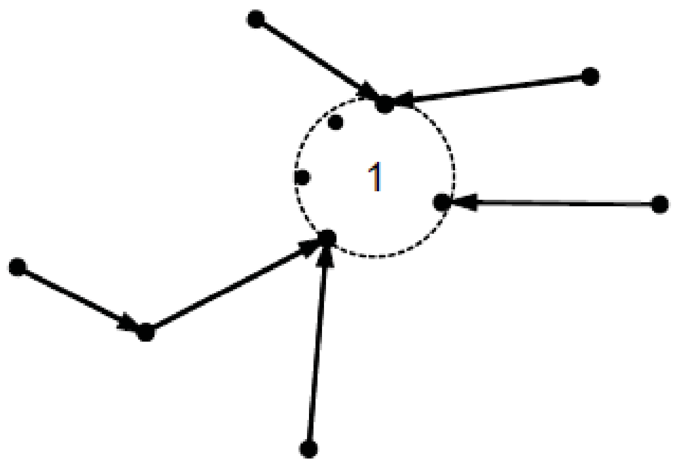
Anycast

- Simple extension for DV and LS algorithms
- Same destination “appears” at multiple places
  - Each router chooses the next hop with the lowest cost to the destination as before
- Used in the Internet for root nameservers
  - This is BGP routing across ISPs though, not within an ISP

Anycast example



Route to closest instance of "1"



Same thing viewed as a sink tree

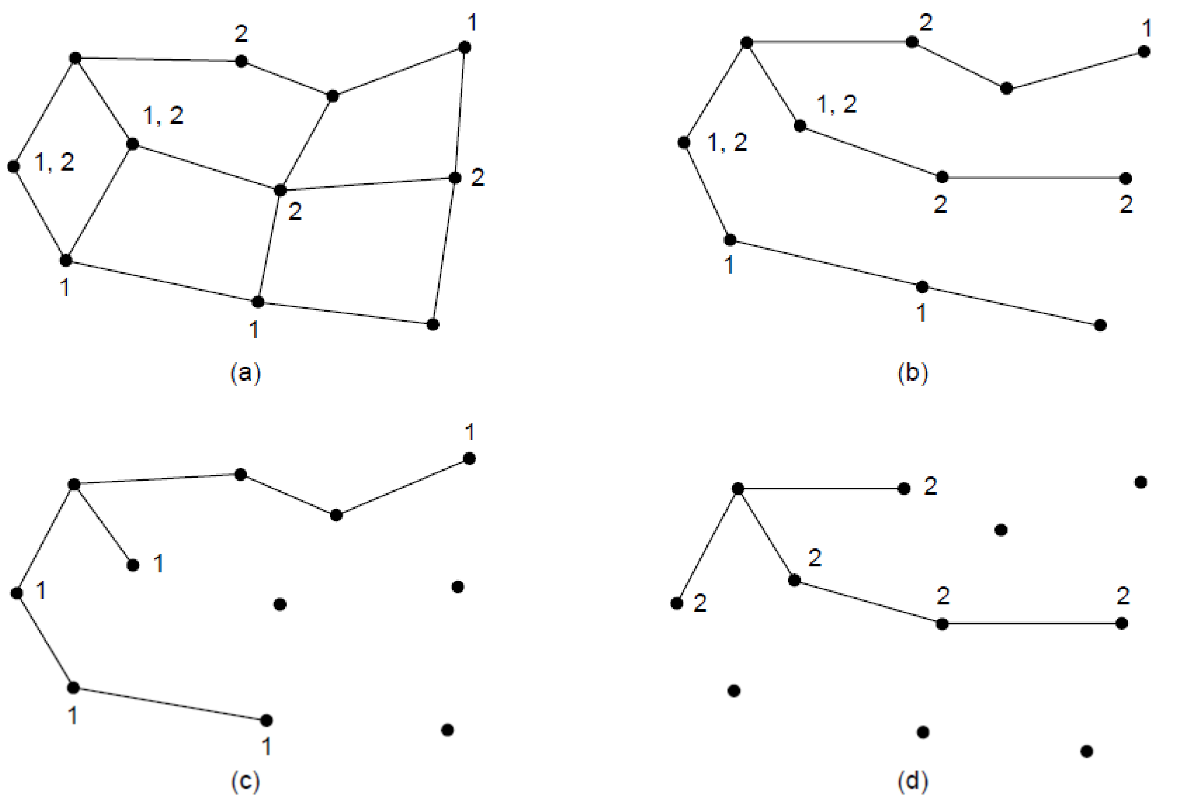
Multicast

- A long and checkered history:
  - Multicast is simple on LANs (just broadcast) and useful for service discovery (“Oi! Who is the printer here?”)
  - Brilliant idea – let’s add it to the Interent
  - But it turned out to be complex, motivated by bandwidth efficiency, and lacking a killer application
  - Finally happening, given simpler schemes and apps like IPTV for an ISP and datacenter distribution

Multicast components

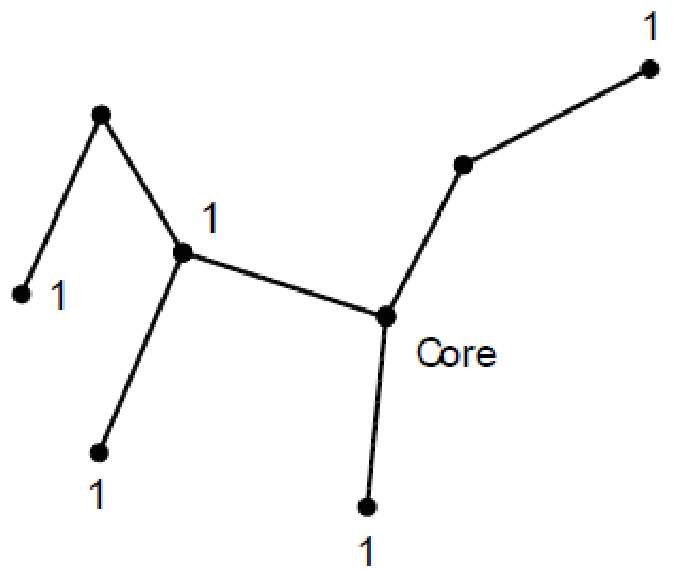
- Requires group membership management
  - To decide who is in the group of receivers
  - IGMP is used; hosts subscribe via routers
- Requires spanning trees to be computed
  - Key challenges are scalability and cross-ISP deployment
  - Handle dense and sparse cases separately
  - Dense: start with broadcast and prune a little
  - Sparse: make a tree just for nodes who need to know

Multicast – per sender, per group trees

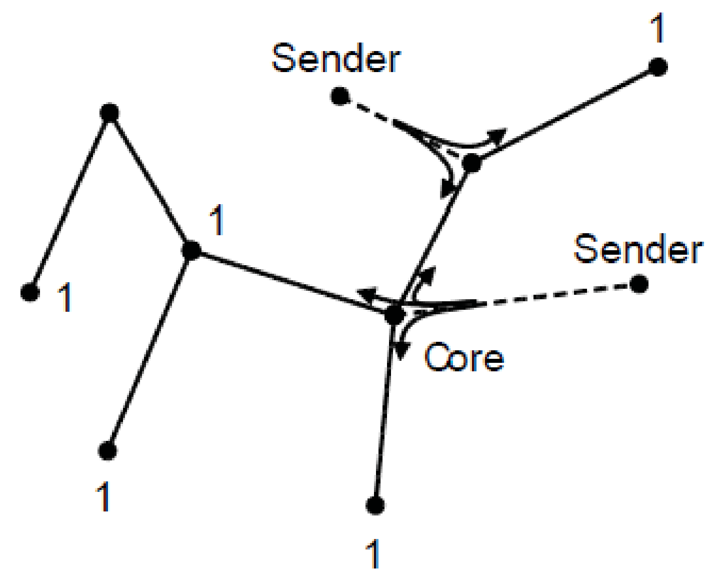


(a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

Multicast – core-based trees (CBT)



CBT for group 1



Sending to group 1

- Only a single tree per group, and only nodes on CBT need to know about the group

RPF – question on multicast

- RPF is 1) simple, and 2) not bandwidth optimal.
  - Suitable for multicast where benefit 1 matters more than cost 2
- Adequate for low-bandwidth multicast (e.g., service discovery) to a good portion of the network
- Inadequate for high-bandwidth multicast (e.g., video) to a small portion of the network
- In practice, separate multicast routing preferred for efficiency and security

RPF – question on reliability

- Sources of packet loss:
  - Routing changes
  - Congestion
  - Transmission errors (rare except for wireless)
- Unicast versus broadcast
  - Above factors apply to unicast as well as broadcast
  - Broadcast seen at a single receiver not necessarily less reliable
  - Reliability added at higher levels for both, e.g., TCP
- Reliable broadcast
  - Significantly harder than reliable unicast (TCP)
  - Specialized protocols and techniques (NACKs, FEC, …)

RPF – question on tradeoffs

- RPF is 1) simple, and 2) not bandwidth optimal.
  - Suitable for broadcast where benefit 1 matters more than cost 2
  - That is, low-bandwidth uses in simple networks
  - Not good for high-bandwidth uses
  - Not a big deal to use per-sender spanning tree in practice
- RPF provides unreliable broadcast
  - Specialized transport protocols needed for reliable broadcast