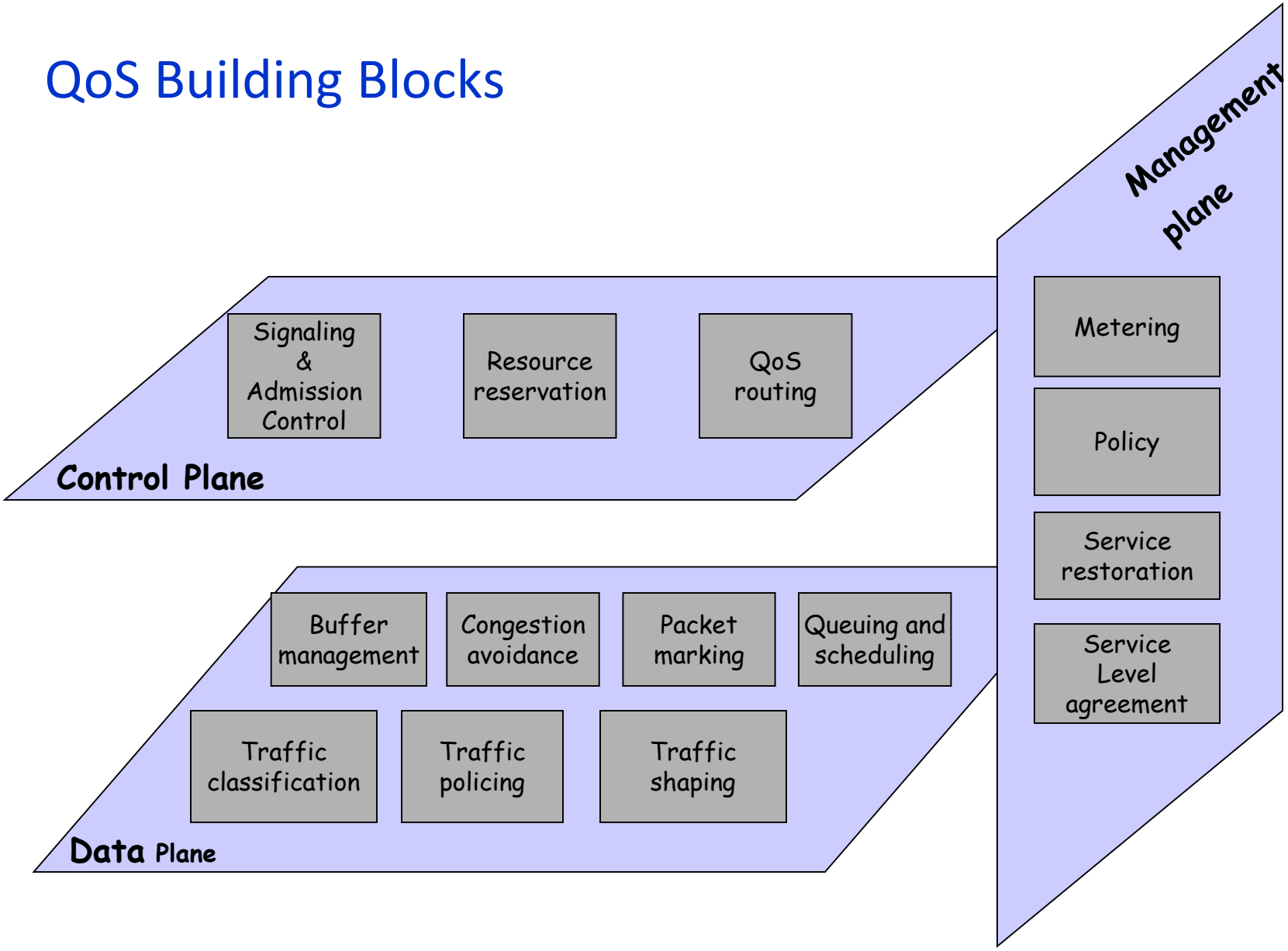
Traffic Management
&
Traffic Engineering

QoS Routing

Outline

- Economic principles
- Traffic classes
- Mechanisms at each time scale
  - Faster than one RTT
  - One RTT
  - Session
  - Day
  - Weeks to months
- Some open problems

QoS Building Blocks

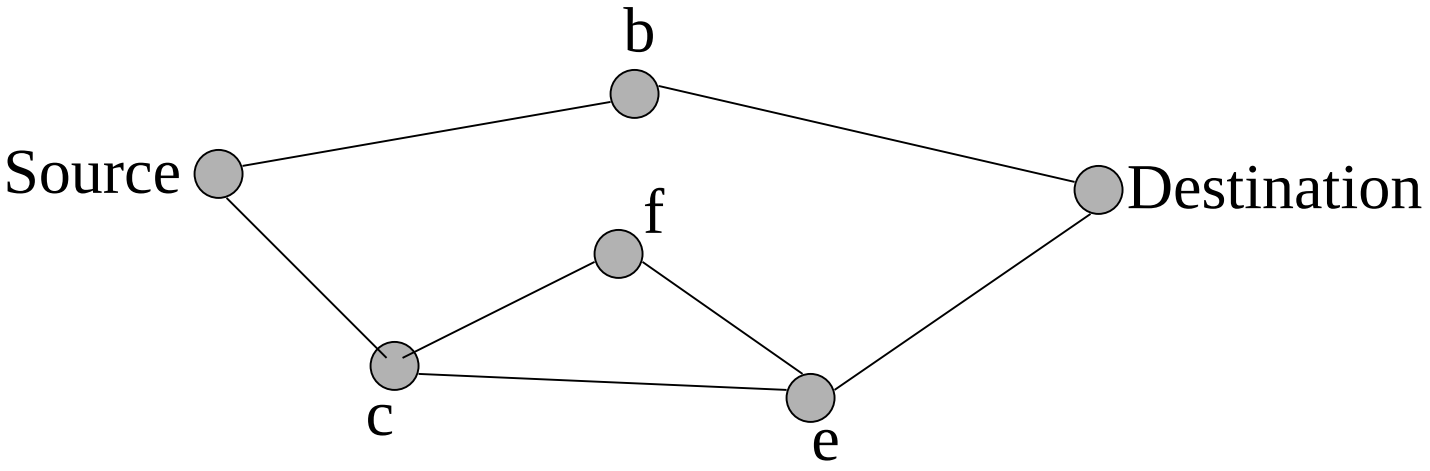


QoS Routing

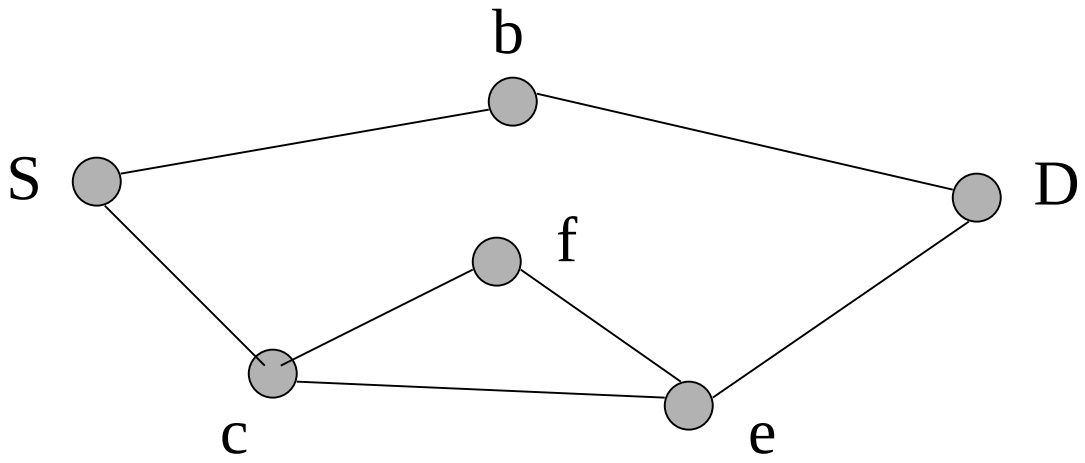
- Selection of a path satisfying the QoS requirements of a flow
  - *Not necessarily the shortest path*
- Parameter (Constraint) Consideration
  - Single QoS metric (Single Constraint)
    - Bandwidth , delay
  - Multiple QoS metrics (Multiple Constraints)
    - Cost-delay, cost-bandwidth, and bandwidth-delay
- Path selection process
  - Find a path considering Flow's QoS requirements, characteristic, and availability of network resources
- QoS routing tends to entail more frequent and complex path computation

What is Routing?

- Need to find a route from source to destination

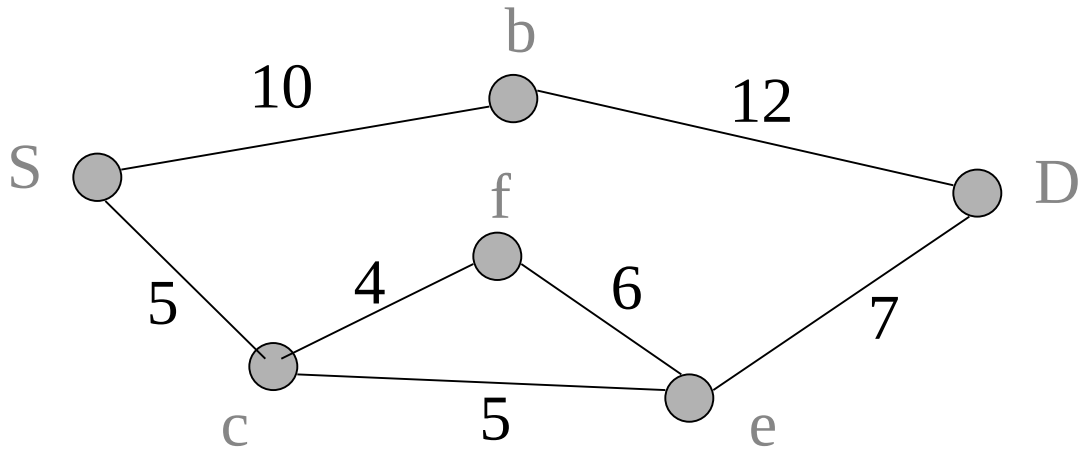


Single Objective Routing



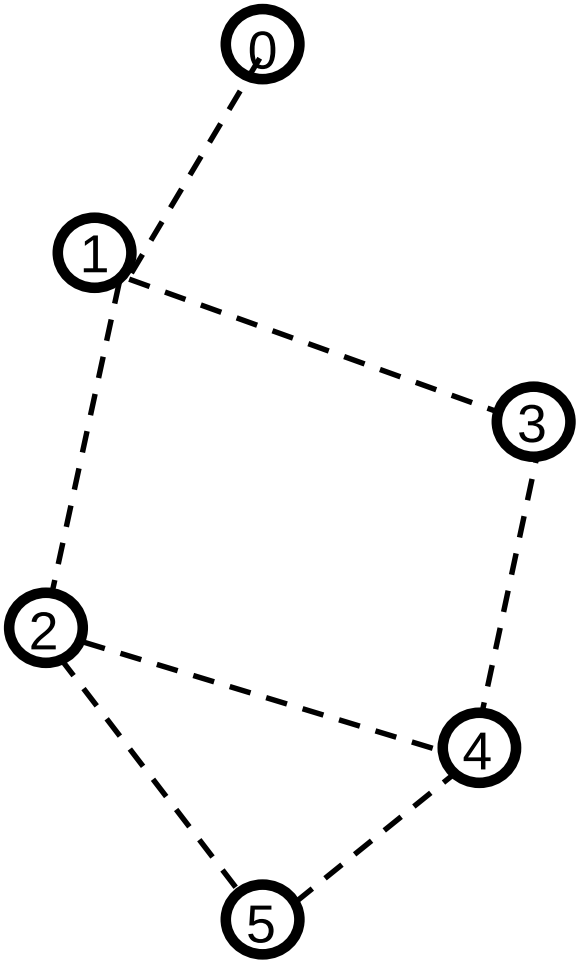
- Assign certain weights (additive, typically, hopefully) to edges of graph.
- Find “shortest” route from source to destination (the route for which the SUM of weights is minimal).

Shortest Hop Count and Edge Length



- Shortest hop-count: “S-b-D” = 2
- Shortest edge length: “S-c-e-D” = 17

Routing – Distance Vector

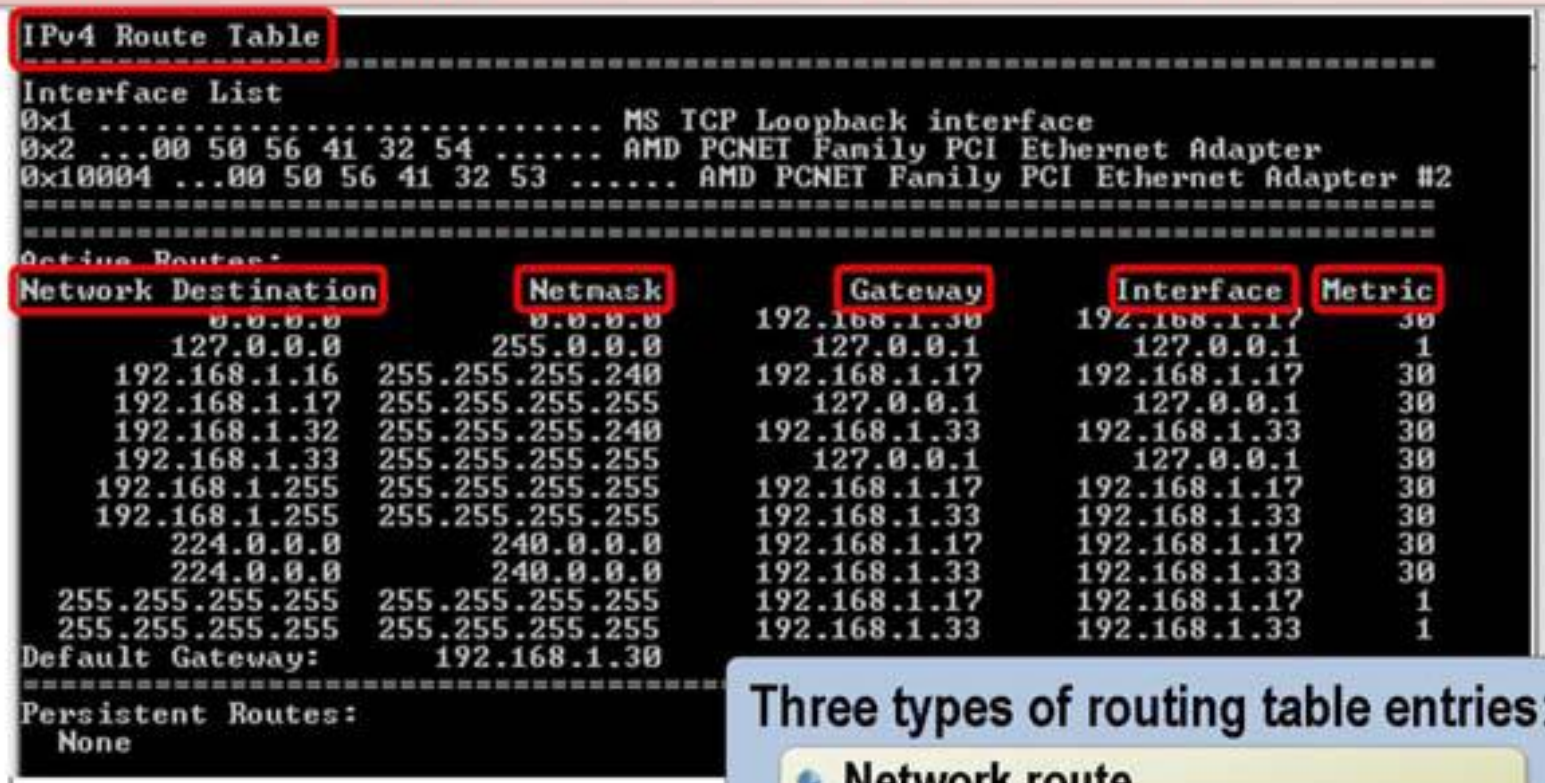


Routing table at node 5:

Destination	NextHop	Distance
0	2	3
1	2	2
..	..	..

What Are Routing Tables?

A *routing table* is a series of entries called routes that contain information about the location of the network IDs in the internetwork



```
IPv4 Route Table
===========================================================================
Interface List
0x1 ........................... MS TCP Loopback interface
0x2 ...00 50 56 41 32 54 ...... AMD PCNET Family PCI Ethernet Adapter
0x10004 ...00 50 56 41 32 53 ...... AMD PCNET Family PCI Ethernet Adapter #2
===========================================================================
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0     192.168.1.30    192.168.1.17      30
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1       1
    192.168.1.16  255.255.255.240    192.168.1.17    192.168.1.17      30
    192.168.1.17  255.255.255.255       127.0.0.1       127.0.0.1      30
    192.168.1.32  255.255.255.240    192.168.1.33    192.168.1.33      30
    192.168.1.33  255.255.255.255       127.0.0.1       127.0.0.1      30
   192.168.1.255  255.255.255.255    192.168.1.17    192.168.1.17      30
   192.168.1.255  255.255.255.255    192.168.1.33    192.168.1.33      30
        224.0.0.0        240.0.0.0    192.168.1.17    192.168.1.17      30
        224.0.0.0        240.0.0.0    192.168.1.33    192.168.1.33      30
  255.255.255.255  255.255.255.255    192.168.1.17    192.168.1.17       1
  255.255.255.255  255.255.255.255    192.168.1.33    192.168.1.33       1
Default Gateway:       192.168.1.30
===========================================================================
Persistent Routes:
  None
```

Three types of routing table entries:

- Network route
- Host route
- Default route

Routing Tables

Purpose of a routing table

- The information in a routing table helps to determine the optimal route within an internetwork.
- The routing table is not exclusive to a router.
- Hosts (nonrouters) may also have a routing table that they use to determine the optimal route.

Types of routing table entries

- ***Network route.*** A network route is a path to a specific network ID in the internetwork.
- ***Host route.*** A host route is a path to an internetwork address (network ID and node ID). Host routes are typically used to create custom routes to specific hosts to control or optimize network traffic.
- ***Default route.*** A default route is used when no other routes in the routing table are found.

Routing Protocols

- A ***routing protocol*** is a set of messages that routers use to determine the network topology and appropriate path to forward data. Routing protocols automatically manage changes in the routing table that occur because of network changes.
- ***Routing Information Protocol (RIP):*** Designed for exchanging routing information within a small to medium-size network.
- ***Open Shortest Path First (OSPF)***: Designed for exchanging routing information within a large or very large network.
- ATM uses *Private Network to Network Interface* (***PNNI)***

RIP

- RIP dynamically builds routing tables by announcing the contents of its routing table to its configured interfaces.
  - Uses Distance-Vector and hop count as metric
  - When a router receives a routing update that includes changes to an entry, it updates its routing table to reflect the new route
  - RIP routers maintain only the best route (the route with the lowest metric value) to a destination
- Routers connected to those interfaces receive these announcements and use them to build the appropriate routing tables.
- The routers that receive the announcements then compile their own routing table, which is then transmitted to other routers. This process continues in a manner that should provide each configured router with the routes from each of the other routers.

OSPF

- Instead of exchanging routing table entries as RIP routers do, OSPF (link state protocol) routers maintain *a map of the network* that is updated after any change in the network topology. This map is called the *link-state database*.
- OSPF allows a router to calculate the shortest path for sending packets to each node.
- The router sends information, called *link-state advertisements (LSAs),* about the nodes to which it is linked to all other routers on the network.
  - Information is flooded to all routers in the network
  - In large networks, flooding delays and overheads can cause instabilities in the routing database
- The router collects information from the other routers, which it uses for link-state information and to make calculations.

Routing Entities

- The routing protocol manages the dynamics of the routing process: capturing the state of the network and its available network resources and distributing this information throughout the network.
- The routing algorithm uses this information to compute paths that optimize a criterion and/or obey constraints. Current best-effort routing consists of shortest path routing that optimizes the sum over the constituent links of a single measure like hop count or delay.
- QoS routing takes into account multiple QoS requirements, link dynamics, as well as the implication of the selected routes on network utilization, turning QoS routing into a notoriously challenging problem

Routing Problem

- The routing protocols (RIP, OSPF, etc.) mainly use hop counts (link costs generally set to 1) to select paths.

- This does not meet the requirements of many emerging communication applications.

- For example, live multimedia applications must make sure that
  - Packet delays are bounded.
  - Jitters (changes in packet delays) are well controlled.
  - Bandwidth guarantees must be met

Today's Routing

- Best Effort routing
- The network resources are fairly shared by packets from different sources
- Disadvantages
  - Does not support resource reservation for guaranteed end-to-end performance.
  - Delays experienced by packets are unpredictable.
- The routing (for Traffic Engineering) for the next generation of high-speed wide area networks will be virtual connection-oriented QoS routing (e.g., MPLS)
- ATM PNNI uses QoS Routing!!

QoS Routing

- Dynamic determination of feasible paths
- Feasible path selection may be subject to *policy constraints*, such as path cost, provider selection, protection requirements etc or subject to *QoS constraints* such as bandwidth, delay, jitter.
- Optimization of resource usage.
- Based on efficient state-dependent network engineering.
  - Routing protocol has to periodically distribute the current state of the link QoS metrics (e.g., delay, available bandwidth) to all nodes in the network.

Two States maintained by nodes

- Local State:
  - Each node is assumed to maintain its up-to-date local state (queuing and propagation delay, the residual bandwidth of the outgoing link and availability of any other resource information)
  - The local states are flooded in the network periodically to update other nodes
- Global State:
  - The combination of the local state of all nodes.
  - The global state kept by a node is always an approximation of the current network due to the delay of propagating local states as the network size grows.

What is QoS Routing?

- One of the key issues in providing QoS guarantees is how to determine paths that satisfy QoS constraints.
- Solving this problem is known as “QoS routing” or “Constraint-Based Routing (CBR)” or “Multi-Constrained Path (MCP)”
- Need:
  - Link state database with *up to date QoS information of all links*
  - Routing protocols are modified to provide this extra information to nodes in the network
- Hard problem:
  - Accurate network state information is very expensive to maintain (flooding costs, how frequently and how often)
  - Computing QoS paths can be expensive and may need to be done for each incoming request

QoS Routing

- Find the path for a given source and destination that best satisfies a given set of criteria (Multiple Constraints).

Performance metrics include:

– Hop count

– Delay

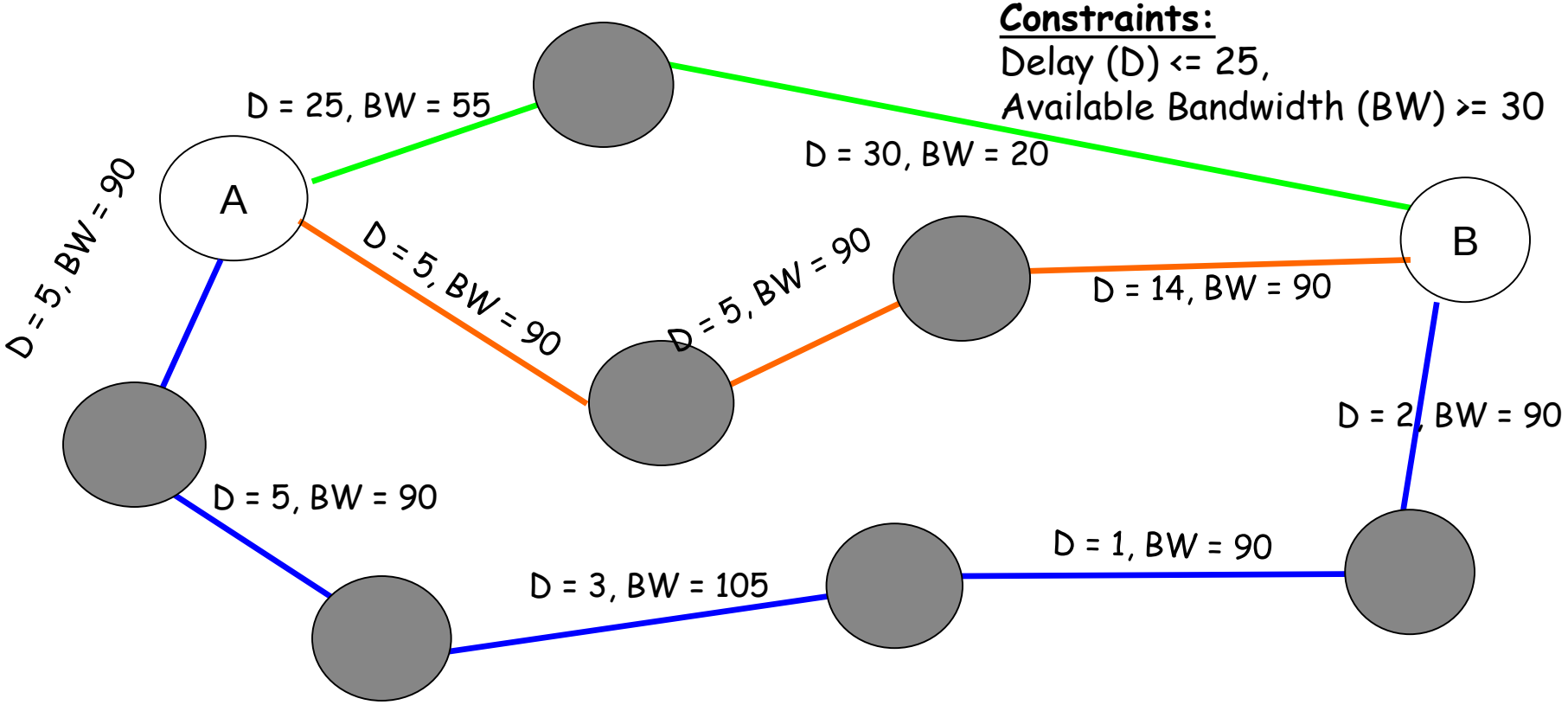
– Jitter

– Data loss rate

– Available bandwidth

– Queue length (available buffer space)

Look for feasible path with least number of hops



2 Hop Path ----> Fails (Total delay = 55 > 25 and Min. BW = 20 < 30)

3 Hop Path ----> Succeeds!! (Total delay = 24 < 25, and Min. BW = 90 > 30)

5 Hop Path ----> Don't consider, although (Total Delay = 16 < 25, Min. BW = 90 > 30)

QoS Routing benefits

- Path setup Without QoS Routing
  - must probe path & backtrack
  - non optimal path
  - Control traffic and processing overhead and latency
- Path setup with QoS Routing
  - optimal route; “focused congestion” avoidance (TE)
  - more efficient Call Admission Control (at the source)
  - more efficient bandwidth allocation (per traffic class)
  - resource renegotiation possible

Routing Strategies

- Tasks of QoS routing
  - Collect the state information and keep it up to date
  - Find a feasible path for a new connection
- Routing can be divided into three categories according to how the state information is maintained and the search of feasible paths is carried out:
  - Source routing
  - Distributed routing
  - Hierarchical routing

Source Routing

- Each node maintains a database (image) of the global network state, based on which a feasible routing path is centrally computed at the source.
- The global network state is typically updated periodically by a link-state algorithm.

Strengths

- Achieves simplicity by transforming a distributed problem into a centralized one.
- Guarantees loop-free.
- Easy to implement, evaluate, debug and upgrade

Weakness

- Communication overhead excessively high for large scale networks
- The inaccuracy in the global state may cause the QoS routing fail.
- Computation overhead at the source is excessively high, especially when multiple constraints are involved.

Distributed Routing

- The path computation is distributed among the intermediate nodes between the source and the destination.
- Some algorithms may require each node to maintain global network state, based on which the routing decision is made on a hop-by-hop basis.
- In some flooding-based algorithms, the routing decision depends entirely on the local state.

Strengths

- The routing response time can be made shorter and more scalable.
- Searching for multiple paths in parallel for a feasible one increase the chance of success

Weaknesses

- Same problem as source routing because of the need of global state share.
- When global states at different nodes are inconsistent, loops may occur.

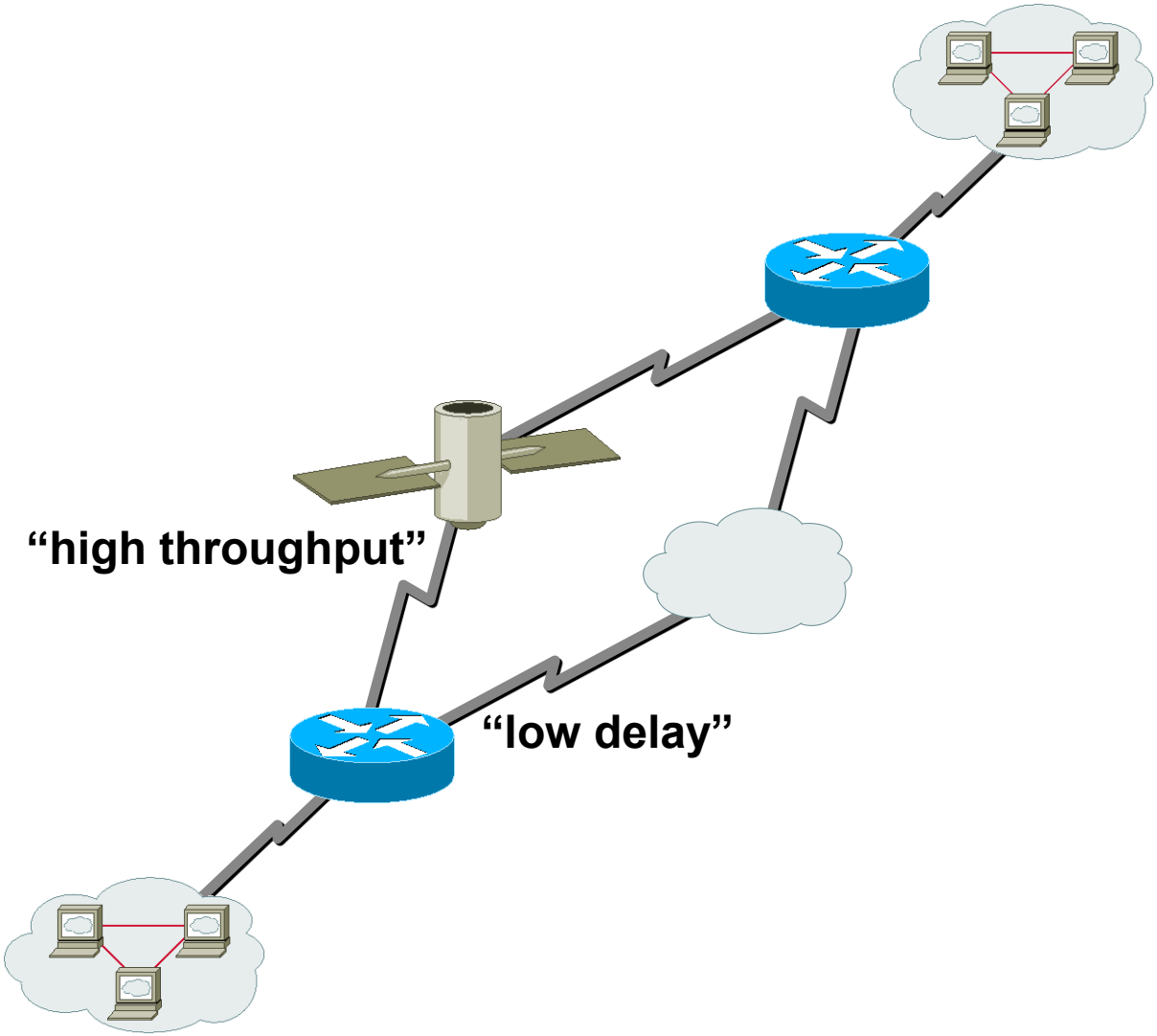
Hierarchical Routing

- Intra-domain routing: within a single autonomous system (or routing domain). Routing protocols are known as Interior Gateway Protocols (IGPs). (e.g., OSPF, RIP)
- Inter-domain routing: between multiple autonomous systems (or routing domains). Routing protocols are known as Exterior Gateway Protocols (EGPs) (e.g. BGP)
- How to extend QoS Routing across multiple areas and multiple domains (AS) is ongoing research at IETF

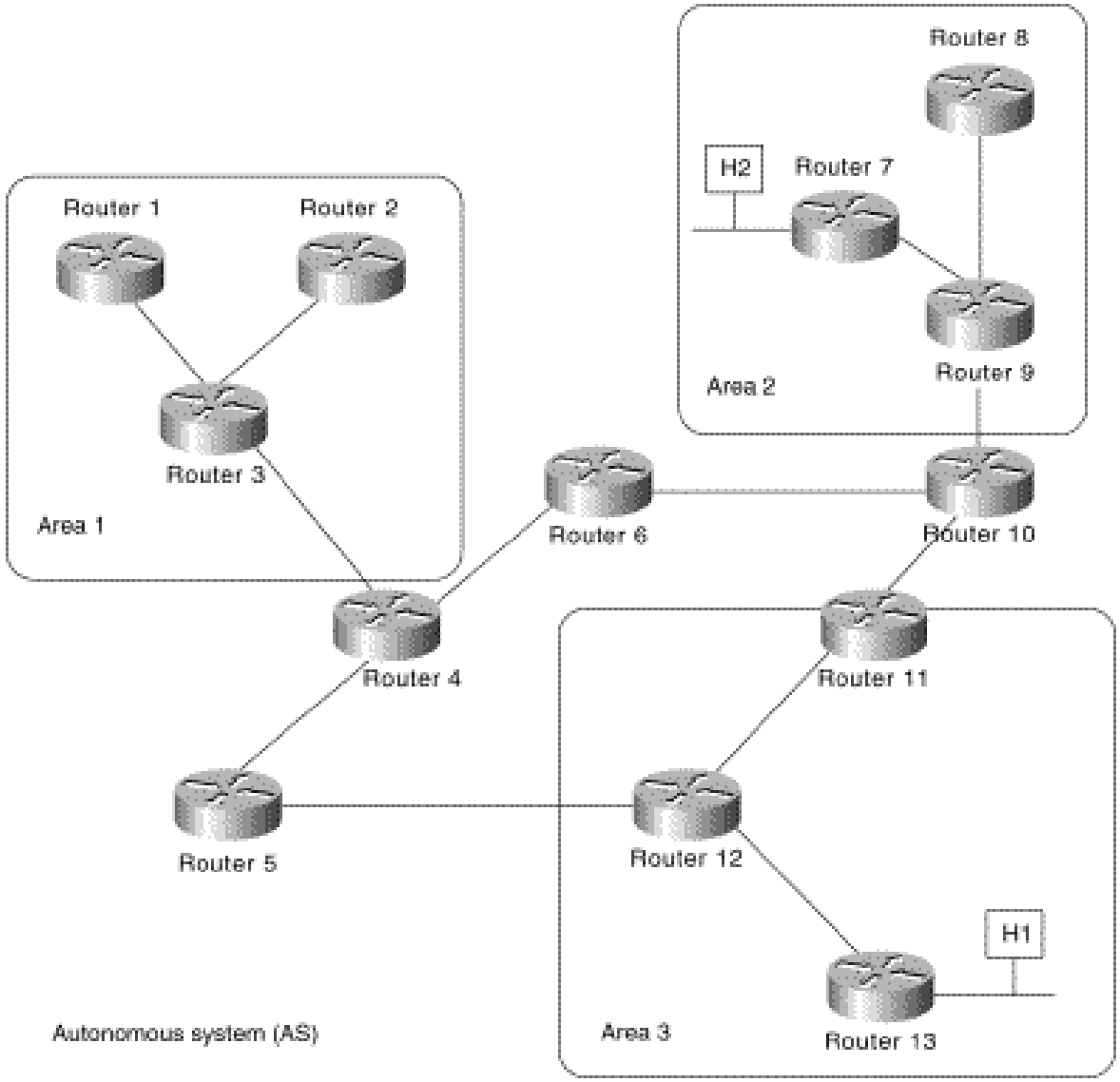
Intra-domain Routing

- OSPF: open shortest path first
- The domain is divided into various areas
- Using link state algorithm to determine routes
- Different costs can be used for different TOS
  - Networks without virtual connections can use this
- Load will be distributed across several equal-cost-paths to destination (Balancing) (ECMP)
  - Networks without virtual connections can use this
- Support for hierarchy through multiple areas

Type of Service (TOS) Routing



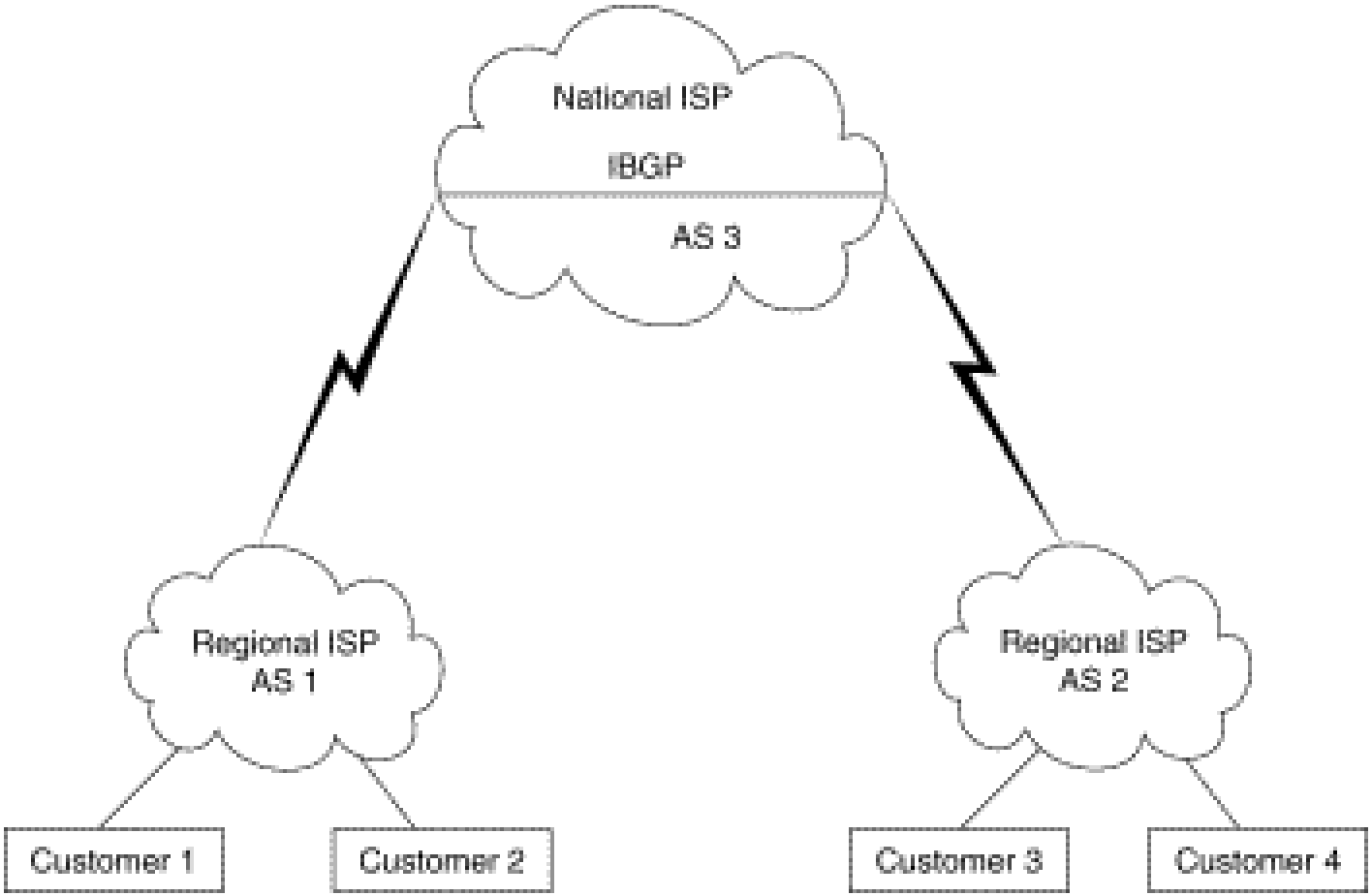
OSPF Areas


Router 8
H2
Router 7
Router 1
Router 2
Router 9
Area 2
Router 3
Router 6
Router 10
Area 1
Router 4
Router 11
Router 5
Router 12
H1
Autonomous system (AS)
Area 3
Router 13

Intra-domain routing

- BGP: Border Gateway Protocol
- Routing between nodes in different Autonomous Systems (AS).
- When the protocol is used within an AS for route exchange, it is called Interior BGP (IBGP)
- When it is used between AS, it is called Exterior BGP (EBGP)
- Uses a distance vector approach
- Policy-Based Routing

BGP Example


National ISP
IBGP
AS 3
Regional ISP
AS 1
Regional ISP
AS 2
Customer 1
Customer 2
Customer 3
Customer 4

TE extensions to OSPF

- RFC3630
- Intra-area only (not for Inter-area and Inter-AS)
- This extension makes use of the *Opaque LSA* of OSPF
  - Opaque LSA (RFC 2370) is a mechanism to distribute any application specific information to routers.
  - Based on this, a new LSA is defined, called the Traffic Engineering LSA
  - Some parameters that are distributed are:
    - Traffic engineering metric (4 octets)
    - Maximum bandwidth (4 octets)
    - Maximum reservable bandwidth (4 octets)
    - Unreserved bandwidth (32 octets) (8 QoS)
    - Administrative group (4 octets): a bit mask designating the group's Resource Color

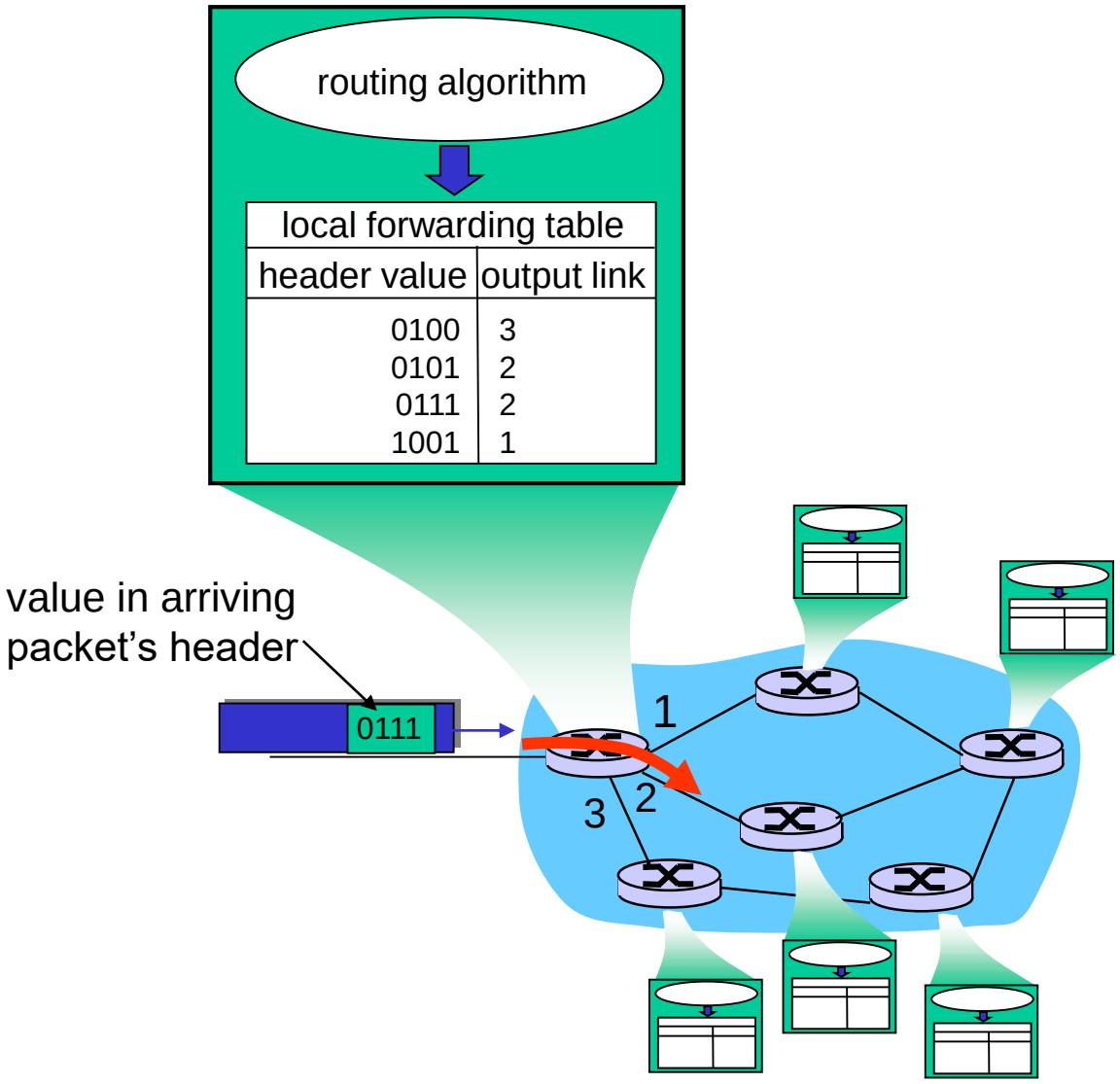
Evaluating QoS Routing Algorithms

- Measuring routing performance
  - Blocking ratio, routed bandwidth ratio, average path length
- Topology
  - Linear, mesh, ring
- Type of traffic
  - Uniform, Hotspots

QoS Update Policies

- When should routers update the QoS changes?
- *Threshold based*
  - update triggered if *relative change* in bandwidth exceeds a threshold value
  - more accurate for smaller values of available bandwidth
- Using clamp-down timers enforces a minimum spacing between two successive updates
  - Large values will have adverse effect on routing performance
  - small values increase network traffic with many updates and brings down efficiency

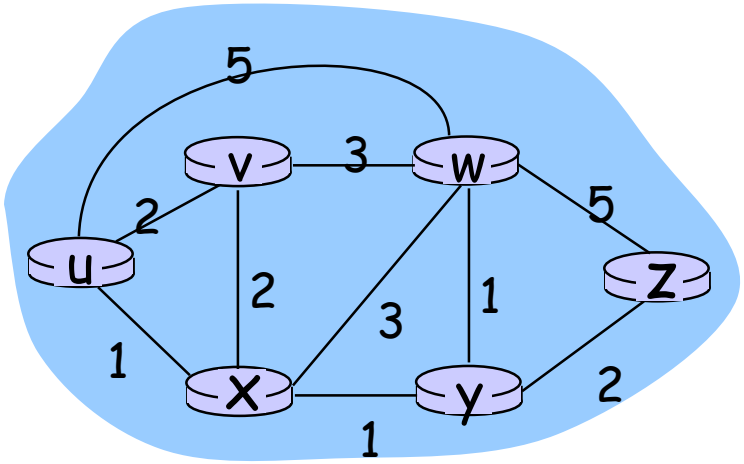
Interplay between routing, forwarding



Routing Algorithms

- Given a graph G=(N,E), a shortest path algorithm finds a path with minimal distance, according to the given link costs, between a pair of source and destination.
- Shortest path algorithms are the foundation of network routing.
- Every real-world network routing protocol is either a centralized, distributed, or hybrid implementation of such algorithm
  - Dijkstra
  - Bellman-Ford

Graph abstraction



Graph: G = (N,E)

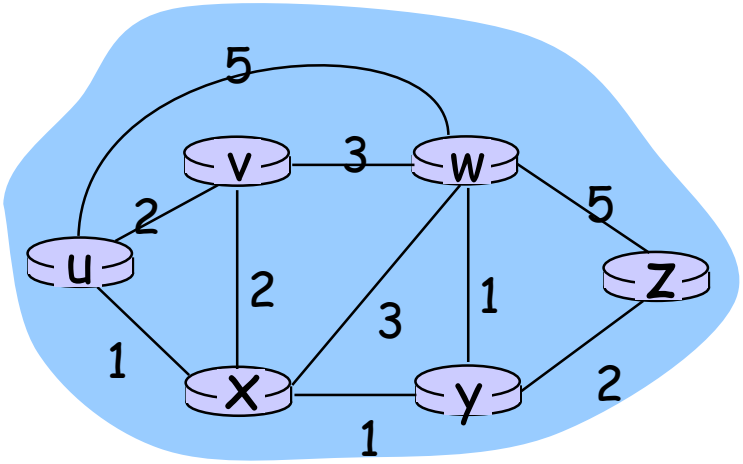
N = set of routers = { u, v, w, x, y, z }

E = 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 $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

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, link cost info
- "link state" algorithms

Decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance 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 Link-State Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives forwarding table for that node
- iterative: after k iterations, knows least cost path to k dest.'s

Notation:

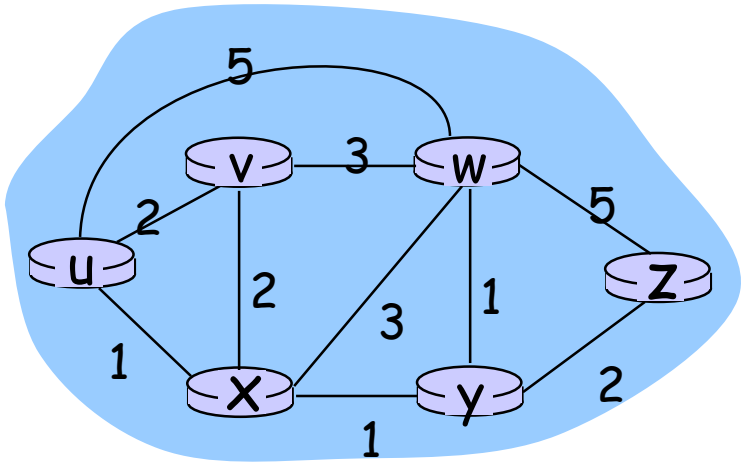
- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra's Algorithm

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5           then D(v) = c(u,v)
6      else D(v) = ∞
7
8   Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12        D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15  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					



Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Define

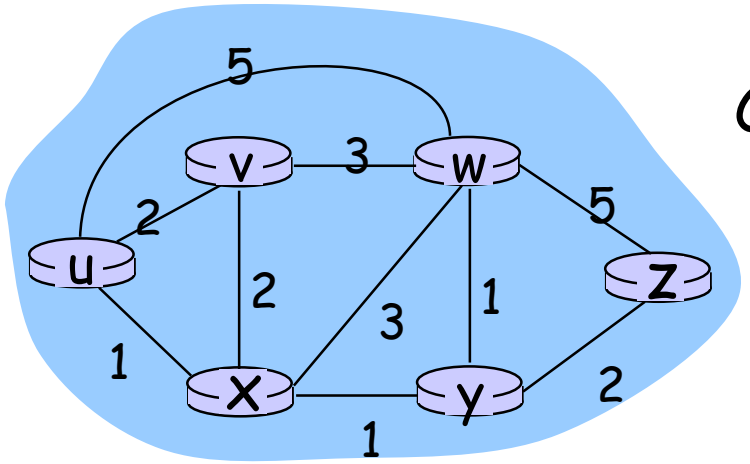
$d_x(y)$:= cost of least-cost path from x to y

Then

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

where min is taken over all neighbors v of x

Bellman-Ford example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

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 achieves minimum is next hop in shortest path ➜ forwarding table

Distance Vector Algorithm

- $D_x(y)$ = estimate of least cost from x to y
- Node x knows cost to each neighbor v: $c(x,v)$
- Node x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

Distance vector algorithm

Basic idea:

- From time-to-time, each node sends its own distance vector estimate to neighbors
- Asynchronous
- When a node x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

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

Distance Vector Algorithm

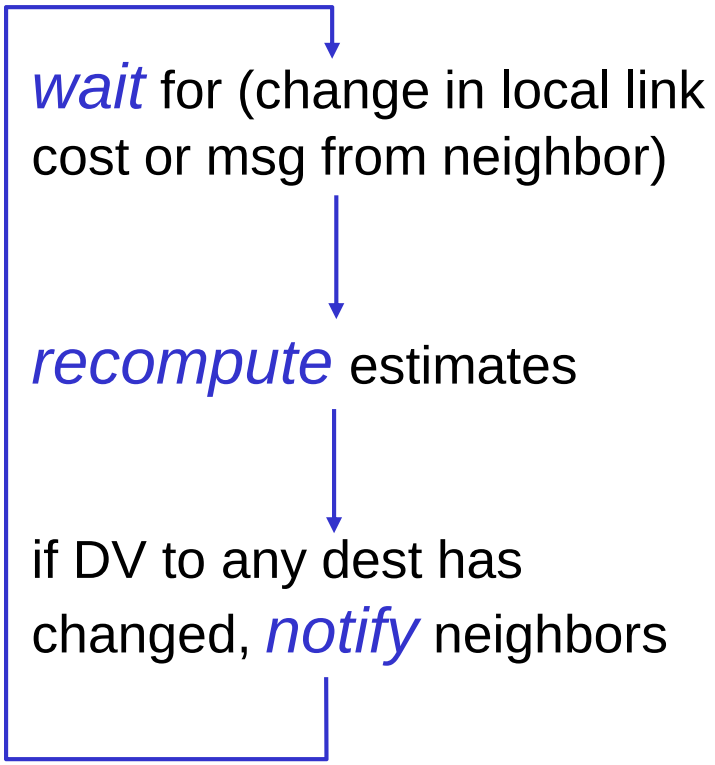
Iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

Each node:



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

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

node x table

from	cost to x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

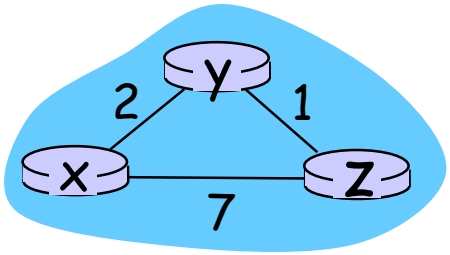
from	cost to x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

from	cost to x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

from	cost to x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



time

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

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

node x table

from	cost to x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from	cost to x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from	cost to x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

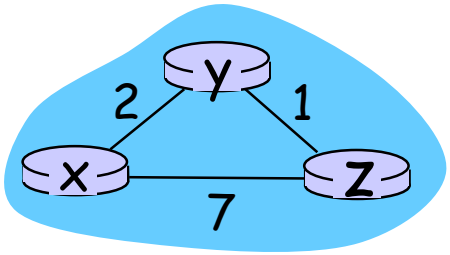
from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

from	cost to x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from	cost to x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0



time

QoS (Multi-Constraint) Routing Problem

- Consider a graph *G = (N,E)* in which each link $u \rightarrow v$ from node *u* to node *v* is characterized by a *m* dimensional link weight vector

$$\overline{w}(u \rightarrow v) = [w_1(u \rightarrow v), w_2(u \rightarrow v), \ldots, w_m(u \rightarrow v)]$$

  where the component $w_i > 0$ is a QoS measure such as delay, jitter, loss, minimum bandwidth, cost, etc.

- The QoS routing algorithm computes the path *P* that obeys multiple constraints, $w_i(P) \leq L_i$ for all $1 \leq i \leq m$.

- For example, we seek a path for which the source-destination delay < 10 ms, total cost < 10, and minimum bandwidth per link is at least 1 Mb/s.

- The set L_i is user requested quality of service desires and constitutes a constraint vector

Multi-Constraint Routing Example

- Consider one objective to be minimized (w1, cost) and one constraint (w2, delay) to be met.
  1. Each edge has two weights w1(e) and w2(e).
  2. Want to minimize the two objectives (or minimize one while constraining the other).
  3. One approach is to consider some objective function (e.g., linear sum of the two weights, i.e, w1 + w2, a variable) as link cost
  4. Run Dijkstra and find shortest route
  5. If w2 constraint is met: OK. Done.
  6. Otherwise: modify the object function and go back to 4.

P and NP Problems

- A Class P problem can be solved in polynomial time on real machines and is considered tractable.
  - Sorting, accounting, shortest path problems, spanning tree problems and many other problems you use computers to solve daily
- A Class NP problem can be solved in exponential time on real machines.
  - You *may* be able to solve it in polynomial time.
  - All Class P problems are also NP.
- A problem in NP-P, if exists, cannot be solved in polynomial time on real machines and is considered intractable in practice.
- A good way to find a NP-P problem is to consider problems that do not have known polynomial solutions (algorithms).
  - map coloring problem, traveling salesman problem, automatic theorem proving, and some QoS routing problems

NP-complete

- A metric *d* is said to be additive if, given a path

  P=L1,L2,…Ln, d(P) = d(L1)+d(L2)+ … +d(Ln).

  – The delay metric is additive.
- A metric *d* is said to be multiplicative if, given a

  path *P=L1,L2,…Ln, d(P) = d(L1)*d(L2)* … *d(Ln).*

- Theorem:

  Given any *N* additive/multiplicative metrics and their respective constraints, the problem of finding a path satisfying the *N* constraints is NP-complete.

Routing Types as per some metrics

For some metrics (e.g. bandwidth, buffer space), the state of a path is determined by the state of its bottleneck link

"Link-optimization routing" finds the path that "*optimizes*" the performance of its bottleneck link according to a given criteria.

- Ex: bandwidth-optimization routing finds the path with the largest bandwidth in the bottleneck link

"Link-constrained routing" finds a path whose bottleneck "*satisfies*" a given criteria.

- Ex: bandwidth-constrained routing finds a path whose bottleneck supports the given bandwidth

Routing Types as per some metrics (contd ..)

For other QoS metrics, such as delay and jitters, the state of a path is determined by the combined state over all links of the path.

"Path-optimization routing" finds the path that *optimizes* given metric.

- Example: delay-optimization routing finds a path with the minimum (accumulated) delay.

"Path-constrained routing" finds a path that *satisfies* the requirement of the given metric.

- Example: delay-constrained routing finds a path whose delay is bounded by the given value.

Some routing problems

- Link-constrained, path-optimization routing
- Link-constrained, link-optimization routing
- Link-constrained, path-constrained routing
- Path-constrained, link-optimization routing

Bandwidth-Delay Constrained Routing

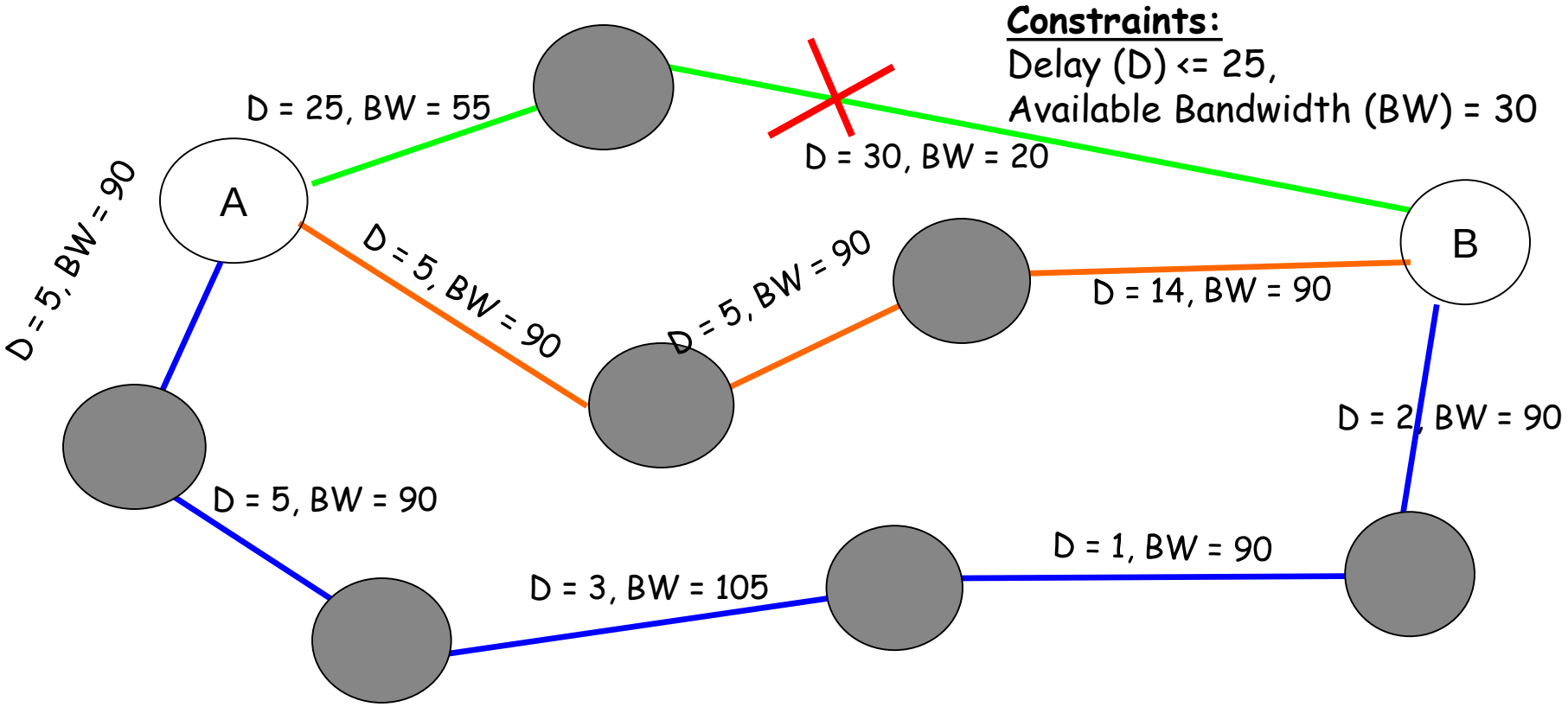
This is a case of link-constrained, path-constrained routing. It lends itself to multimedia applications that demand bandwidth availability and delay bound.

Algorithm

1. Eliminate (Prune) all links that do not meet the bandwidth requirements.
2. Run a traditional shortest path algorithm to find the minimum delay path.
3. The path is accepted, if it meets the delay constraint; otherwise report failure.

We can always get rid of the “link constrained” part by eliminating (pruning) unsatisfactory links. The trick gives rise to the solutions for all the polynomial cases, except the last one, path-constrained, link-optimization routing

Look for feasible path with least number of hops



2 Hop Path ----> Fails (Total delay = 55 > 25 and Min. BW = 20 < 30)

3 Hop Path ----> Succeeds!! (Total delay = 24 < 25, and Min. BW = 90 > 30)

5 Hop Path ----> Don't consider, although (Total Delay = 16 < 25, Min. BW = 90 > 30)

Inter-Area and Inter-AS

- Generally we do not want to distribute QoS information across areas
  - Unnecessary (other areas need not know)
  - Increased complexity in large networks
  - Flooding complexity, policy problems
- One solution is to use TE exchangers
  - Border nodes at the intersection of areas or AS can be used as TE exchanges
  - TE exchanges have QoS information in the area or AS
  - Query the TE exchanges to a compute a feasible path in their respective areas when crossing multiple areas
  - Compile the whole path