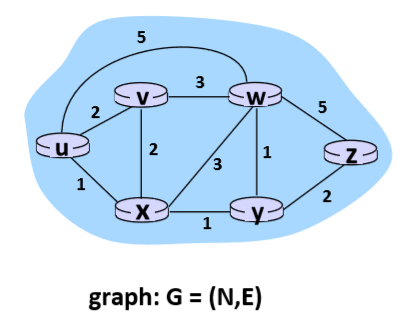
**Graph Abstraction**



N = set of routes = {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)}

Graph Abstraction costs

* C(x,x`) = cost of link (x,x`) e.g. c(w,z)=5
* Routing algorithms find the least cost path

**Routing Algorithm classification**

Q: global or decentralized

* Global:
  + All routers have complete topology (arrangement), link cost info
  + = “link-state” Algorithm
* Decentralized:
  + Router knows physically connected neighbor, link costs to neighbors
  + Iterative process of computation, exchange of info with neighbors
  + “distance vector” Algorithms

Q: static or dynamic

* Static
  + Routes change slowly over time
* Dynamic
  + Routes change more quickly
  + Periodic update
  + In response to link cost changes

**Link-state routing algorithm**

Dijkstra’s algorithm

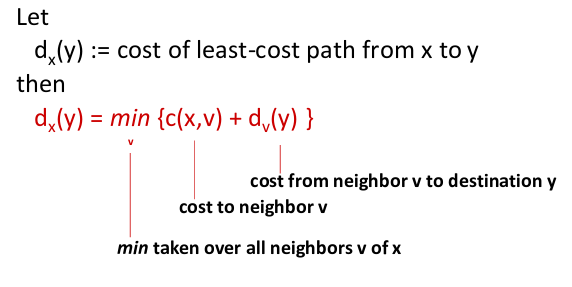
* Network topology, link costs known to all nodes
  + Accomplished via “link-state broadcast”
  + All nodes have the same info
* Computes least cost paths form one node (‘source’) to all other nodes
  + Give forwarding table for that node
* Iterative: after k iterations, know least cost path to k destinations

Notations:

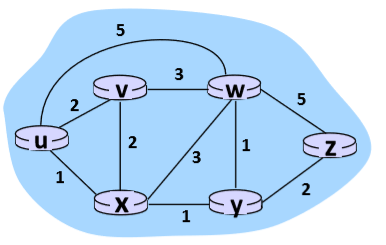
* C(x,y): link cost from node x to y; infinity if not direct neighbor
* D(v): current value of cost of path from source to dest
* P(v): predecessor node along path from source to v
* N’: set of nodes who least cost path definitively known

**Distance Vector Algorithm**

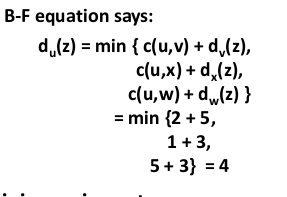
* Bellman-Ford equation (Dynamic Programming)



**Bellman-Ford example**







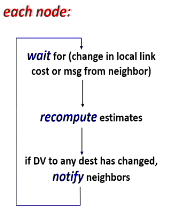
Key Idea with Distance Vector:

* From time-to-time, each node send its own distance vector estimate to neighbors
* When node receives new DV, estimate from neighbor, update its own DV using B-F equation



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
    - Notifies their neighbors if necessary



**Link Cost Changes**

* Node detects local link changes
* Updates routing info, recalculates distance vector
* If DV changes, notify neighbors

**Poisoned reverse**

* If Z routes through Y to get to X:
  + Z tells Y it’s Distance to X is infinite so Y doesn’t route x via Z

**Comparison of LS and DV Algorithms**

Information exchange overhead

* LS: with n nodes, E links, O(nE) messages sent
* DV: exchange between neighbors only, number of exchange messages varies

Convergence

* LS: O(n^3) algorithm requires O(nE) messages
  + Micro-loops
* DV: convergence time varies
  + Potential transmit routing loops
  + Count-to-infinity problem

Misconfigurations: What happens if route malfunctions

* + LS:
    - Node can advertise incorrect link cost
    - Each node computes only its own table => local
  + DV
    - DV node can advertise incorrect path cost
    - Each node’s table used by others
      * Errors propagate through the whole network

Inter Autonomous Systems protocol:

* Gateway router: at edge
* Links to other AS’s
* AS keeps track of other subnetworks it can connect to via gateways
* Which gateways connect to which subnets
* if subnet w leads to subnet x and subnet y:
  + subnet x and y lead to subnet z
  + sends packet to closest router
  + hot potato