Request > proxy > get http > validate > forward

\*listener in proxy ( bind > listen socket [pass in port #] )

Setup connections

\*\*proxy: handles request from client and responses from server

Initiate request to server, and handle request from server

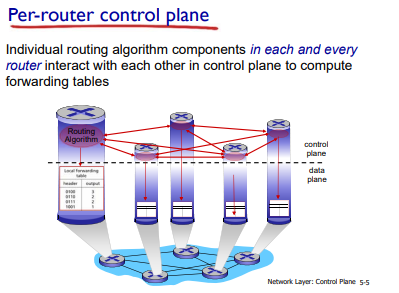
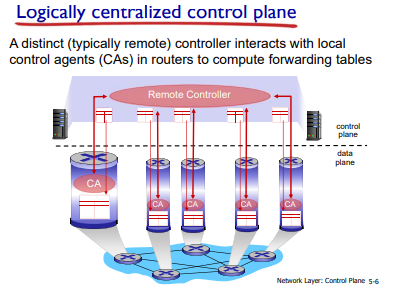
Chunk encoding( isChunk() >>> then handle chunk)

\*\*get body working for non-chunk, then work on non-chunk

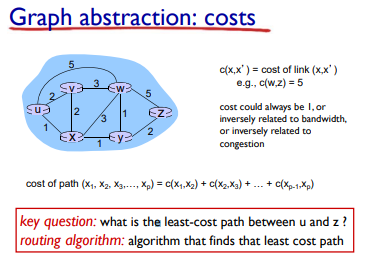
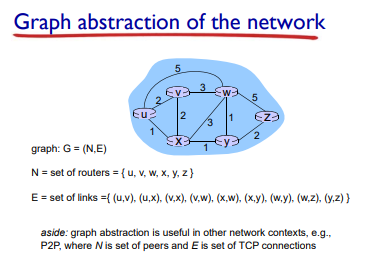
TA provided code snippets for chunk

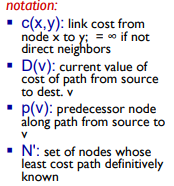
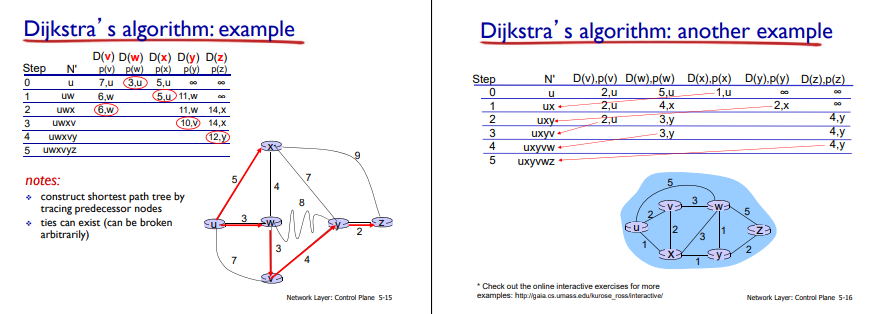
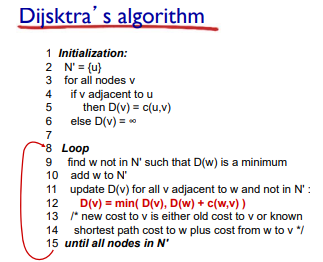
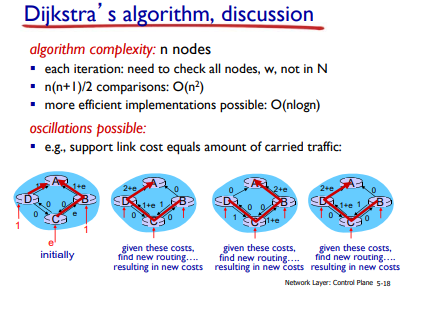
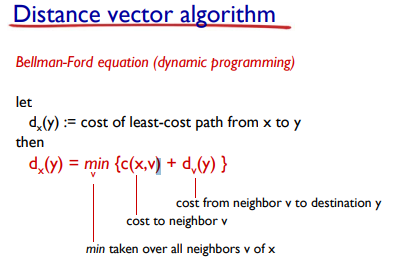
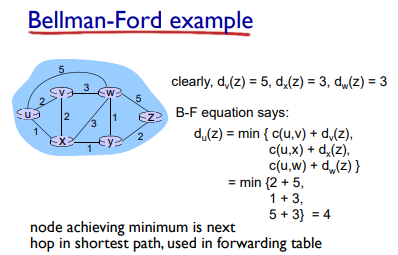
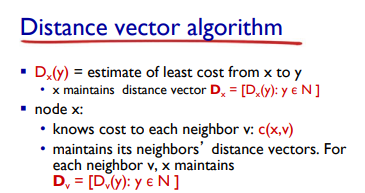
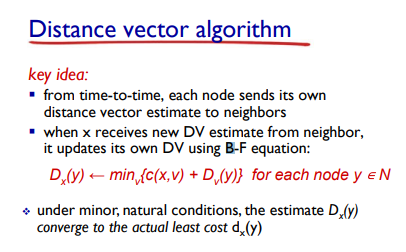
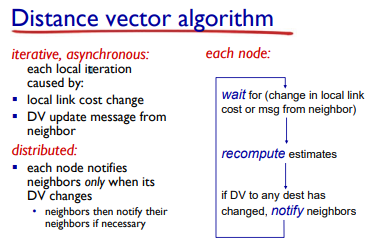
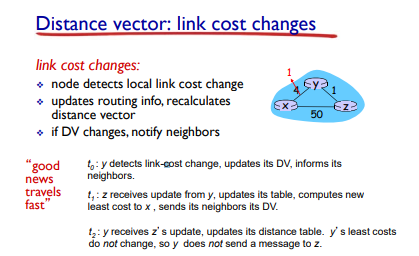
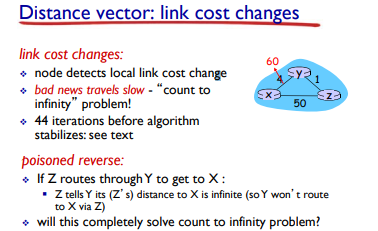
Network Layer : the Control Plane

* Network-layer functions
  + forwarding: move packets from router’s input to appropriate router output
  + routing: determine route taken by packets from source to destination
  + Two approaches to structuring network control plane:
    - per-router control (traditional)
    - logically centralized control (software defined networking)



* Routing Protocols
  + Routing protocol goal: determine “good” paths (equivalently, routes), from sending hosts to receiving host, through network of routers
    - path: sequence of routers packets will traverse in going from given initial source host to given final destination host
    - “good”: least “cost”, “fastest”, “least congested”
    - routing: a “top-10” networking challenge!



* + Routing Algorithm Classification
    - 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:
      * 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 
  + 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, know least cost path to k dest.’s
* Dijsktra’s Algorithm
  + 
* 
* Distance Vector Algorithm
  + 
* Bellman-Ford Example
  + 
* 
* 
* 
* 
* 
* 