The University Of Alabama in Huntsville   Computer Science

cs570

Routing and Routers

The University Of Alabama in Huntsville   Computer Science

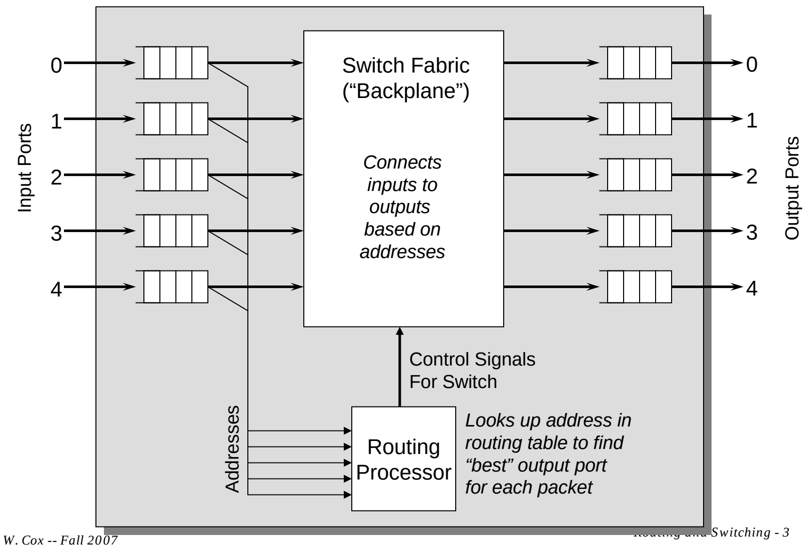
cs570

Router architecture

The University Of Alabama in Huntsville Computer Science

Logical architecture of a router

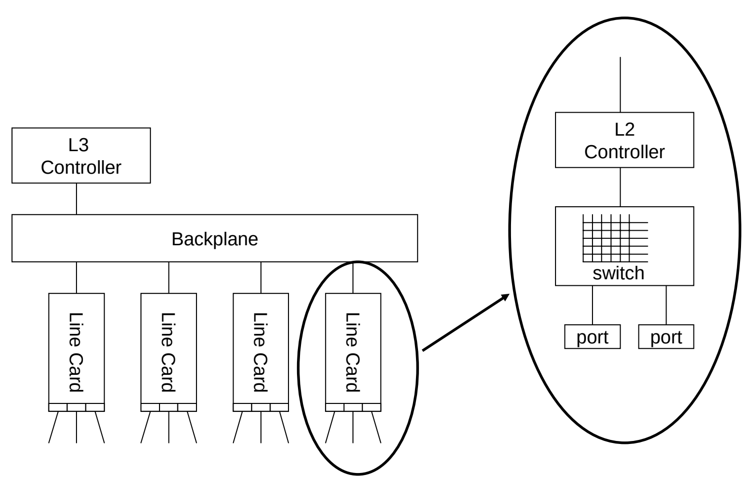
cs570



The University Of Alabama in Huntsville Computer Science

A typical router design

cs570

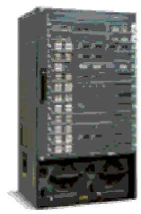


Cisco 7400

Cisco 12416 Internet Router

- •320 Gbps, 270Mpps (OC-192)
- •Crossbar switch
- •16 slots
  - •15x10Gbe or
  - •15 x OC192 or
  - •120xFastEthernet or
  - •15120 x T1s
- •6’ x 1.5’ x 2’
- •390# fully populated, 4.7KW
- •$100K+ (2005)

Cisco 7613 Edge Router



- 30 Mpps
- 256 Gbps
- 33” x 17” x 18”
- 13 slots
    - Up to OC-48, 10GbE
- 240# fully populated, 4KW
- $50K+ (2005)

Procket PRO/8812 Internet Router

- 960Gbps, 1.2Gpps
- 12 linecard slots

Shared-memory-based switch fabric

- Up to OC-768
- 37.5” x 17.4” x 25.5”
- 571# fully loaded, 5.4 KW
- ~$250K (2005)

Switch Fabrics

cs570

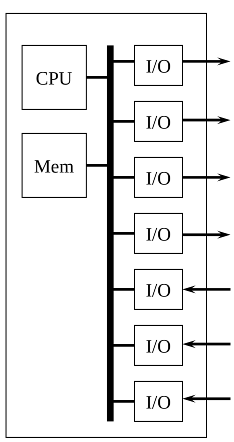
The University Of Alabama in Huntsville Computer Science

- Many types of designs
- Considerations:
  - Speed
  - Flexibility (Blocking behavior)
  - Scalability (cost/size/power/…)

A simple way to build a switch fabric

cs570

The University Of Alabama in Huntsville Computer Science

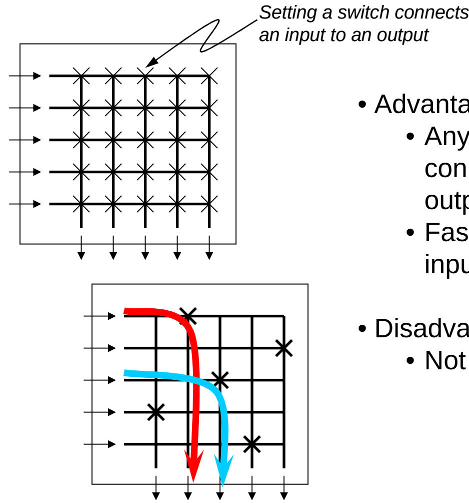


- Advantages
  - Not custom technology
  - Highly Flexible
  - Smart/Programmable
- Disadvantages
  - Single bus limits scalability
  - Hard to build general-purpose design that's as fast as a custom design

The University Of Alabama in Huntsville Computer Science

Non-blocking switch (Crossbar)

cs570



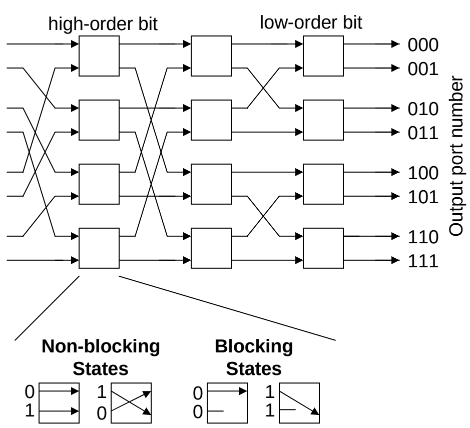
- Advantages
  - Any combination of inputs can be connected to any combination of outputs (“non-blocking”)
  - Fast – just one switch between input and output
- Disadvantages
  - Not scalable: $O(n^2)$

The University Of Alabama in Huntsville Computer Science

Multi-stage switch (Banyan)

cs570

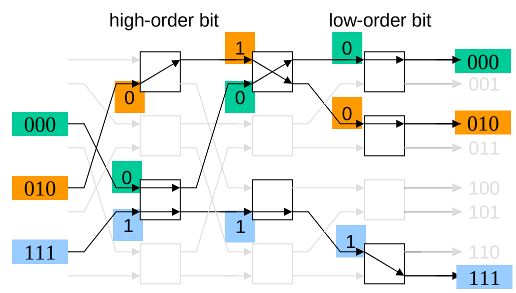
switches in a column are set by the same bit of the desired output port number



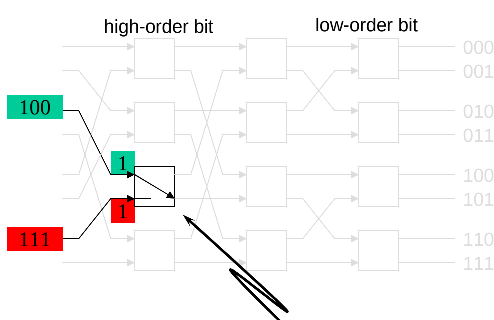
- Advantages
  - Scalable: $O(n \log_2 n)$
  - Modest number of switches input to output ($\log_2 n$)
- Disadvantages
  - Blocking

A Banyan example

We want to connect these three inputs to the indicated output ports



A combination of inputs that blocks



One of the inputs will have to remain queued until the next clock cycle

The University Of Alabama in Huntsville Computer Science

cs570

Routing algorithms

The University Of Alabama in Huntsville Computer Science

Routing algorithms

cs570

- How do we build the routing tables?
- How do we keep the routing tables current?
- Some dead ends:
  - Static tables -- networks are too dynamic
  - Human-built tables -- networks are too large
- Ideally, we'd like the routers to build and maintain their own routing tables

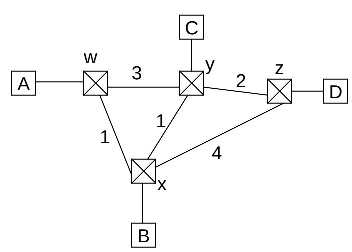
"Cost" of a route

cs570

- There are often multiple paths from any router to a destination. We need to be able to pick the "best" one of them.
- We don't always simply want to determine the shortest route to the destination:
  - A path with more hops may be faster
  - Some paths have a higher monetary cost than others
  - We may have to take a longer path to meet a user performance requirement (e.g, bandwidth)
- In general, we want to select the path that minimizes whatever "cost" we are interested in. The cost may be a single factor (delay, $ cost, etc) or a weighted combination of factors.
- Cost factors are usually associated with links

Path cost examples

cs570



Costs from router y to destination D:

Path	Cost
y z D	2
y x z D	5
y w x z D	8

The University Of Alabama in Huntsville Computer Science

Distance Vector Routing

cs570

- The idea:
  - Each router knows the cost to each of its immediate neighbors
  - Each router builds a “distance vector” that contains the total cost of the best-known route to every destination (initial costs = ∞)
  - At intervals, each router sends its DV to all neighbors
  - When a router *R* receives a DV from a neighbor *N*, *R* scans the table to see if there are any cases where, for Destination *D*:

    N*'s cost to get to *D* + *R*'s cost to get to *N* < *R*'s current cost to get to *D

    If there are any such cases, R updates its table so that future traffic for D is sent to N.

Example:
- Assume router R knows a path to D with a cost of 25.
- R's neighbor N knows a path to D with a cost of 20.
- If the cost from R to N is 3, then R can get to D through N with a total cost of 23.
- Since this is less than the current cost, R will update its routing table so that all traffic for D goes to N.

The University Of Alabama in Huntsville Computer Science

G. W. Cox -- Fall 2007

Routing and Switching - 19

Note about the following DV example

cs570

Although the final state is not meaningfully affected by the ordering of actions between the different routers, interim results can vary.

For clarity in the following example, I am assuming that the algorithm runs in network-wide cycles and that in each cycle, each router broadcasts its DV *BEFORE* it processes incoming DV's.

The University Of Alabama in Huntsville Computer Science

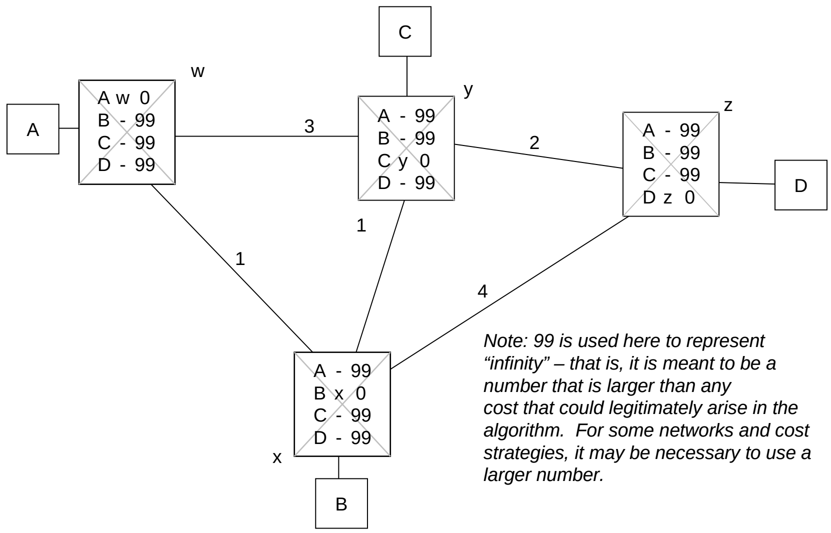
G. W. Cox -- Fall 2007

Routing and Switching - 20

The University Of Alabama in Huntsville Computer Science

Initial DV's

cs570

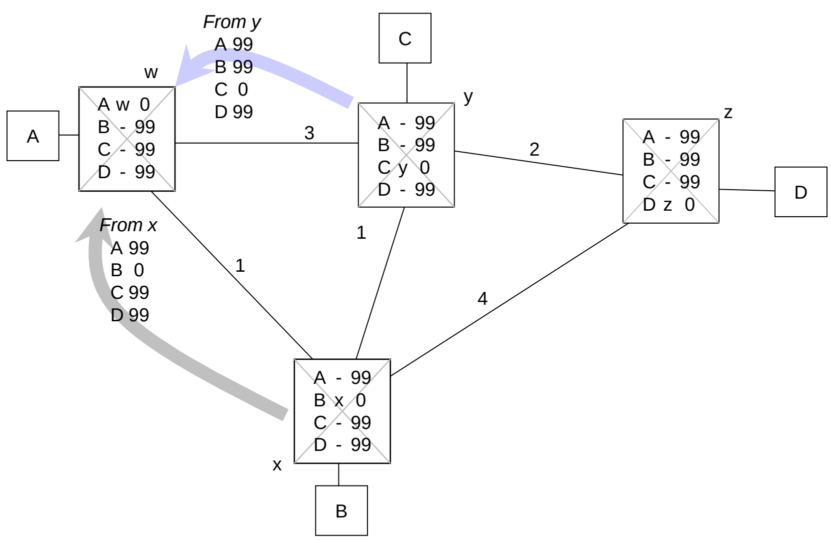


Note: 99 is used here to represent "infinity" – that is, it is meant to be a number that is larger than any cost that could legitimately arise in the algorithm. For some networks and cost strategies, it may be necessary to use a larger number.

The University Of Alabama in Huntsville Computer Science

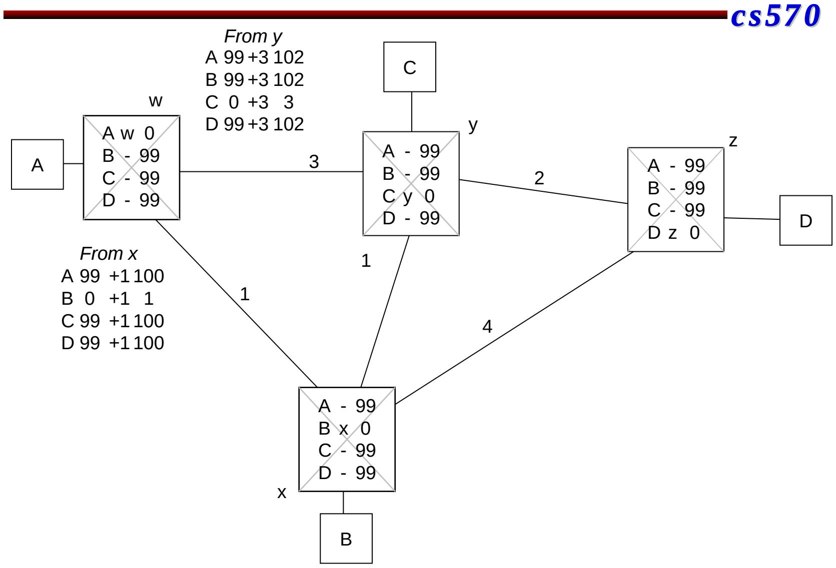
First Cycle – focusing on w: 1. neighbors send DV to w

cs570



First Cycle – focusing on w: 2. w adds cost to get to the neighbor

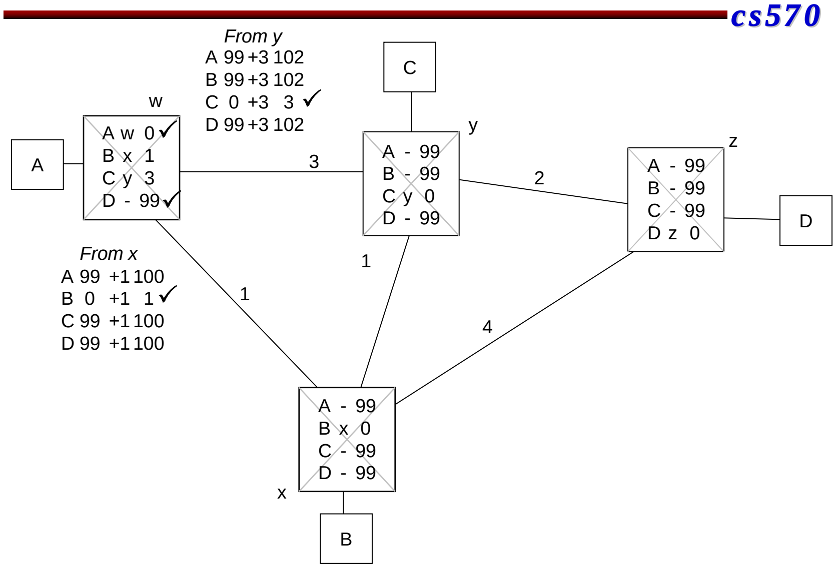
cs570



The University Of Alabama in Huntsville Computer Science

First Cycle – focusing on w: 3. w updates its RT with lowest costs

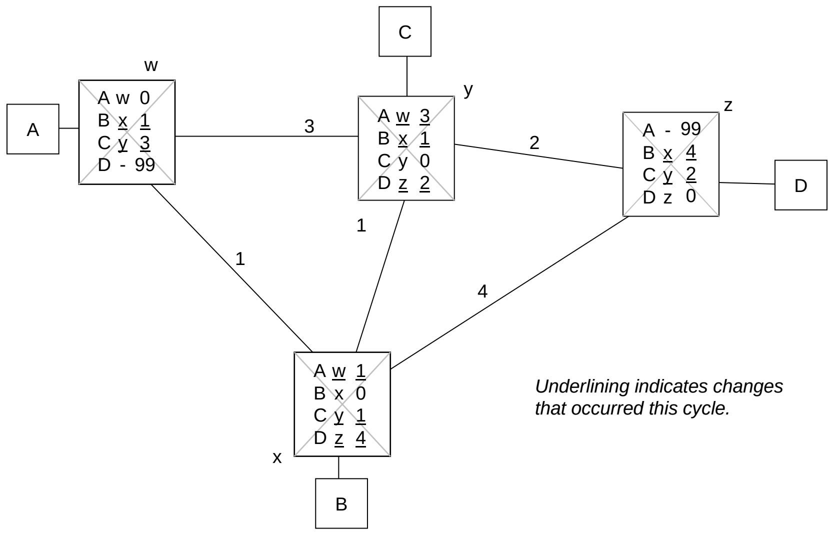
cs570



The University Of Alabama in Huntsville Computer Science

State after First Cycle

cs570

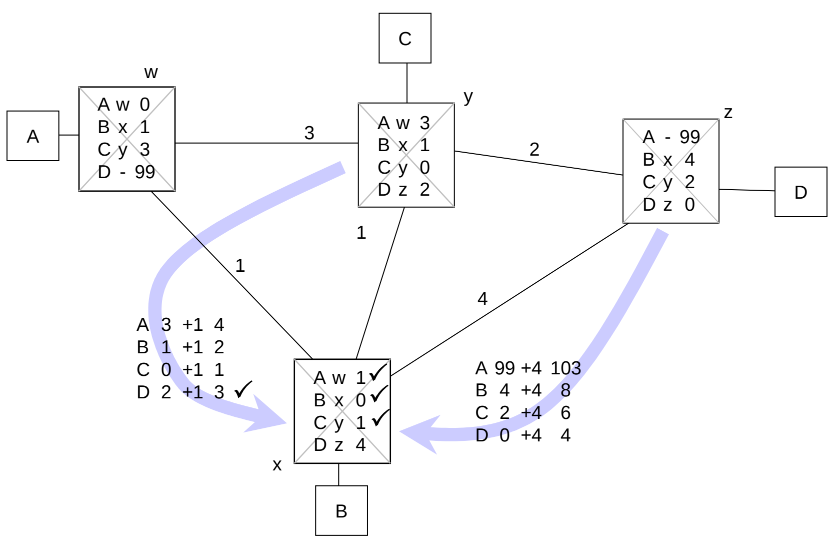


Underlining indicates changes that occurred this cycle.

The University Of Alabama in Huntsville Computer Science

Second cycle - focusing on x

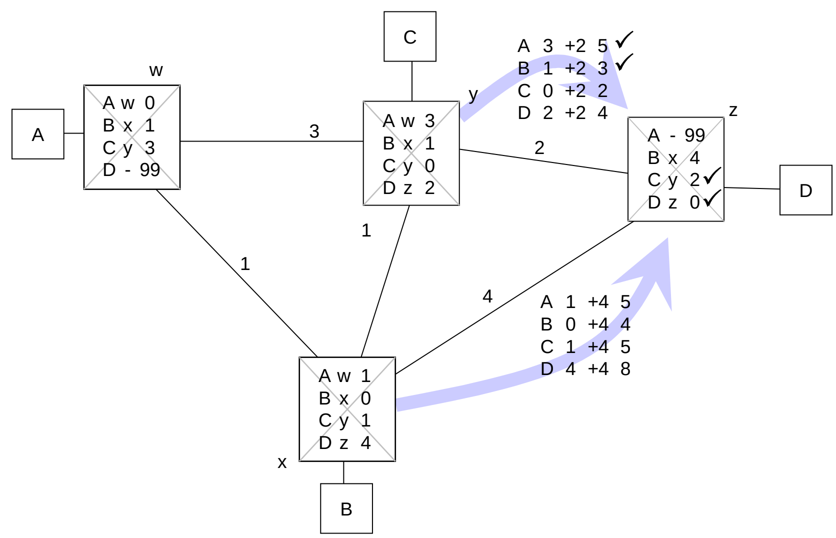
cs570



The University Of Alabama in Huntsville Computer Science

Second cycle - focusing on z

cs570



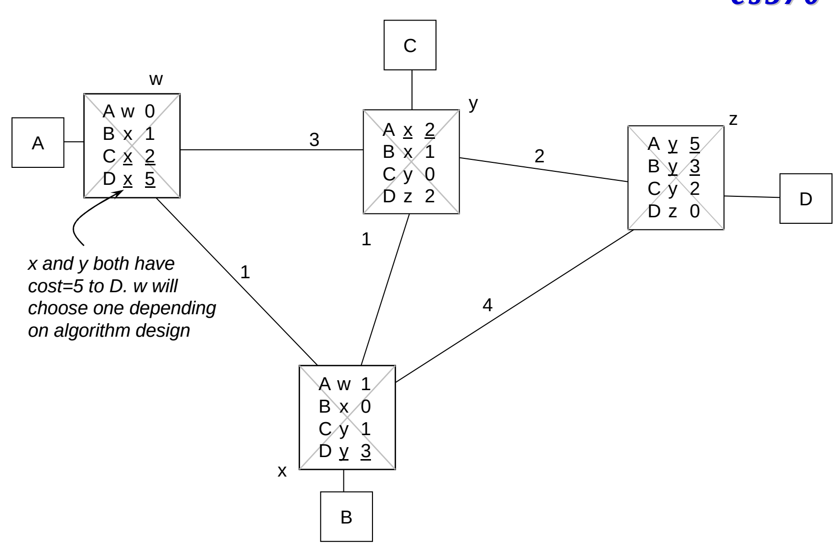
G. W. Cox -- Fall 2007

Routing and Switching - 27

The University Of Alabama in Huntsville Computer Science

State after second cycle

cs570

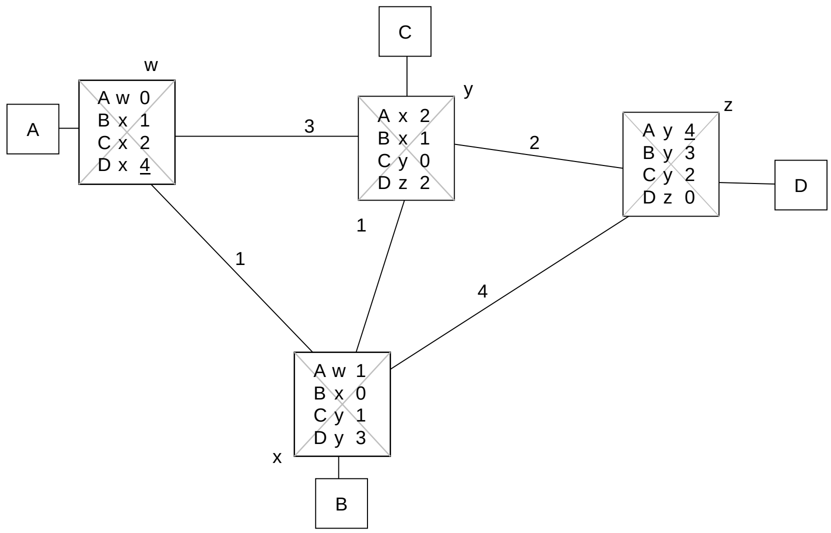


G. W. Cox -- Fall 2007

Routing and Switching - 28

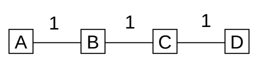
State after third cycle

cs570



A problem with the basic DV algorithm

cs570



Cost to get to A

stable state	1A	2B	3C	
A-B link fails	99A	2B	3C	When B discovers the link to A is down, B sets its cost to infinity
After Exchange	3C	2B	3C	B thinks that C knows a path of cost 2 to A, so B updates its path
After next exchange	3C	4B	3C	
After next exchange	5C	4B	5C	

The costs will continue to slowly rise without stopping. This is called the “Count to Infinity” problem.

Fixes for the Count to Infinity problem

The University Of Alabama in Huntsville Computer Science

cs570

- Define a small number as "infinity" so that problem becomes apparent sooner
- Don't send cost to the neighbor you received your current cost from ("split horizon")
- Send infinity to the neighbor you received your current cost from ("split horizon with poison reverse")

The Link-State routing algorithm

The University Of Alabama in Huntsville Computer Science

cs570

- Used for Open Shortest-Path First (OSPF) routing in Internet
- The idea:
  - Each router discovers cost to immediate neighbors (by pinging, etc)
  - At intervals, this info is flooded to all other routers in a "Link State Packet". This gives all routers a map of the network and link costs.
  - Each router runs a part-finding algorithm (e.g, Dijkstra's Shortest Path Algorithm) to calculate least-cost paths.

Dijkstra's Shortest Path Algorithm: Terminology and Notation

- Goal is to determine the minimum-cost path from Source node "S" to Destination node "D".
- "W" = the ID of the "working node"
- "Cn" is the sum of the link costs for every link in the least-cost presently known path from S to node n.
- "L(a,b)" = cost associated with the link from node a to node b. L(a,b) = infinity if a and b are not directly connected.
- "Label" of a node = n(Cn,y)
  - "n" = the node's ID
  - "y" = ID of the preceding node on the least-cost presently known path from S to node n.
- At various stages of the algorithm, a node label can be "tentative" or "permanent."
  - A tentative label is one that might be changed at a later stage of the algorithm.
  - A permanent label will not be changed. We indicate permanent labels with bold text.

Dijkstra's Shortest Path Algorithm

0. W=S. Assign S the permanent label **S(-,0)** For every node r, r<>S, assign the label r(-, 99) *

While D is not permanently labeled:

1. For each neighbor N of W that is not permanently labeled, calculate x=CW+L(W,N). If x < CN, assign the tentative label N(W, x).
2. Examine the entire graph and find the tentatively-labeled node, w, with the minimum cost in its label. Set W = w.
3. Change W's tentative label to a permanent label.

End While

5. Record the name of D. The "copying node", C = D.
6. Record the name P, where **C(P,x)** is the label of the copying node.
7. P is the new copying node. If P <> S, repeat 5-6.
8. The least-cost path is the reverse order of the recorded node names.

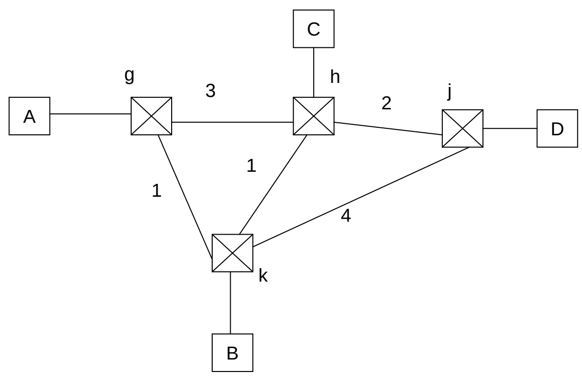
* 99 is used here to represent "infinity" – that is, it is intended to be a number that is larger than any cost that could legitimately arise in the Djikstra's calculation. For some networks and cost strategies, it may be necessary to use a larger number.

Dijkstra's Shortest Path Example

The University Of Alabama in Huntsville Computer Science

cs570

Find the least-cost path from A (g) to D (j)

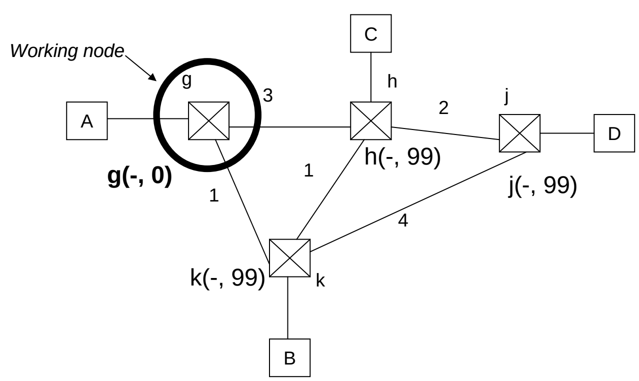


Dijkstra's Shortest Path Example

The University Of Alabama in Huntsville Computer Science

cs570

0. W=S. Assign S the permanent label S(-,0).
For every node r, r<>S, assign the label r(- , 99)

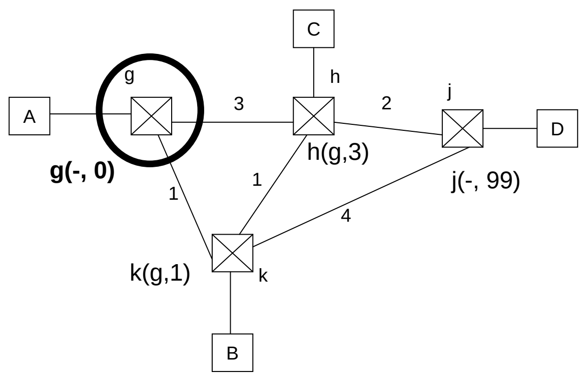


Dijkstra's Shortest Path Example

The University Of Alabama in Huntsville Computer Science

cs570

1. For each neighbor N of W that is not permanently labeled, assign the tentative label N (W, CW+L(W,N))

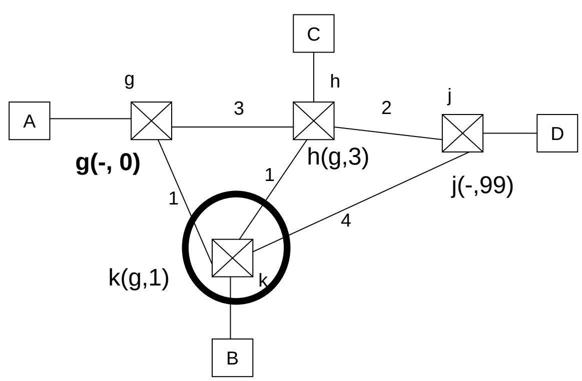


Dijkstra's Shortest Path Example

The University Of Alabama in Huntsville Computer Science

cs570

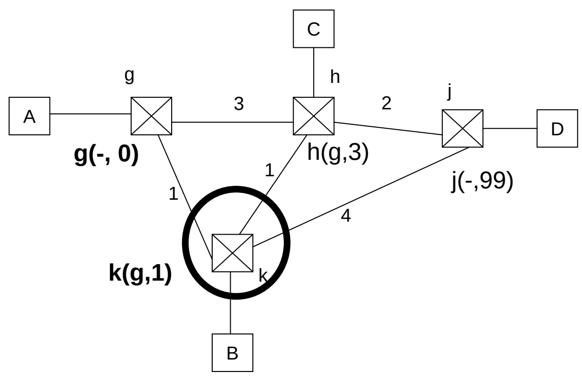
2. Examine the entire graph and find the tentatively-labeled node, w, with the minimum cost in its label. Set W = w.



Dijkstra's Shortest Path Example

cs570

3. Change W's tentative label to a permanent label.



The University Of Alabama in Huntsville Computer Science

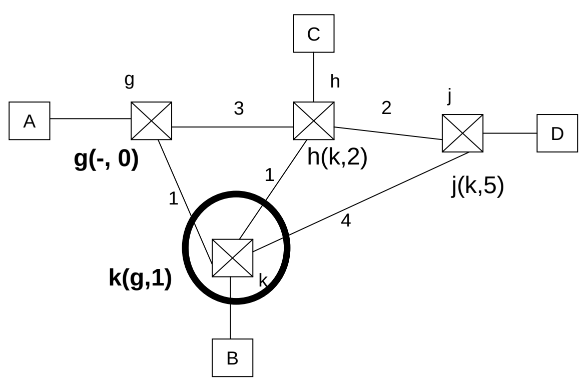
G. W. Cox -- Fall 2007

Routing and Switching - 39

Dijkstra's Shortest Path Example

cs570

1. For each neighbor N of W that is not permanently labeled, assign the tentative label N (W, CW+L(W,N))



The University Of Alabama in Huntsville Computer Science

G. W. Cox -- Fall 2007

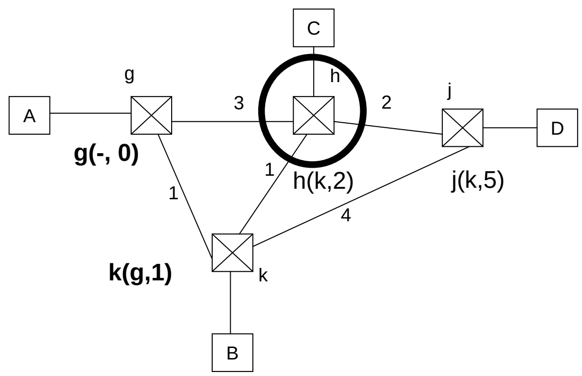
Routing and Switching - 40

The University Of Alabama in Huntsville Computer Science

Dijkstra's Shortest Path Example

cs570

2. Examine the entire graph and find the tentatively-labeled node, w, with the minimum cost in its label. Set W = w.

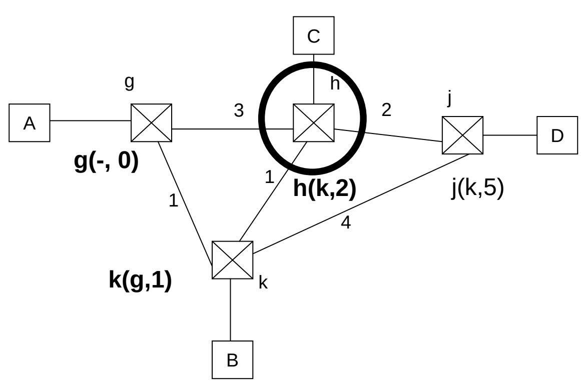


The University Of Alabama in Huntsville Computer Science

Dijkstra's Shortest Path Example

cs570

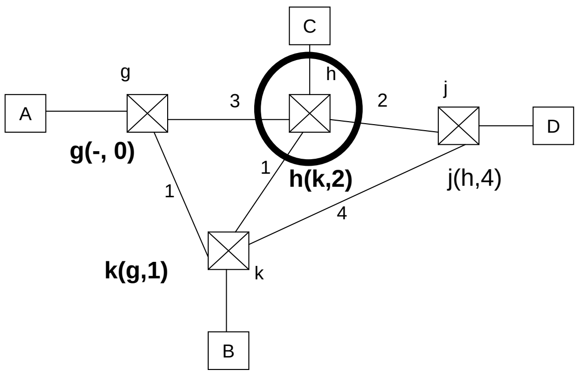
3. Change W's tentative label to a permanent label.



Dijkstra's Shortest Path Example

cs570

1. For each neighbor N of W that is not permanently labeled, assign the tentative label N (W, CW+L(W,N))

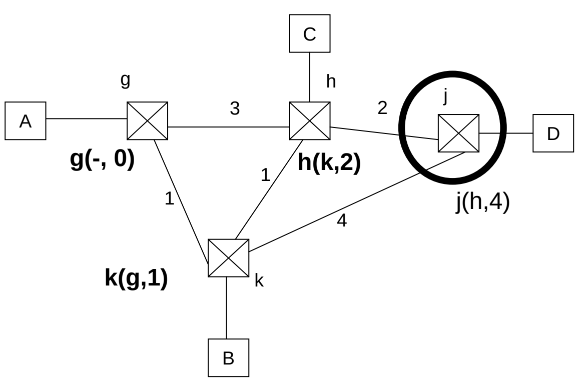


The University Of Alabama in Huntsville Computer Science

Dijkstra's Shortest Path Example

cs570

2. Examine the entire graph and find the tentatively-labeled node, w, with the minimum cost in its label. Set W = w.



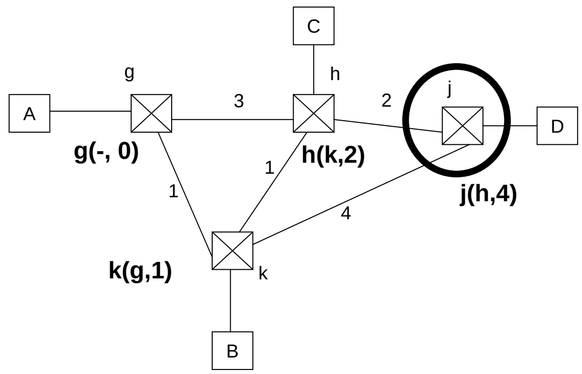
The University Of Alabama in Huntsville Computer Science

Dijkstra’s Shortest Path Example

cs570

The University Of Alabama in Huntsville Computer Science

3. Change W’s tentative label to a permanent label.

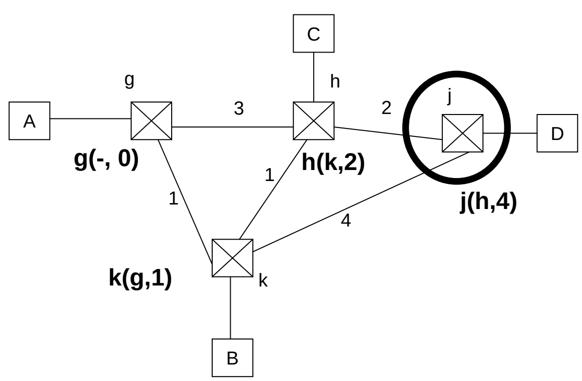


Dijkstra’s Shortest Path Example

cs570

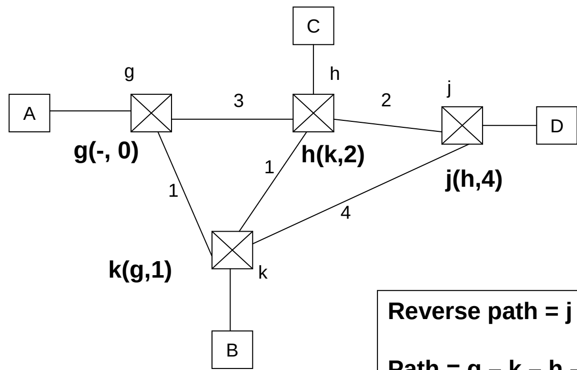
The University Of Alabama in Huntsville Computer Science

3. Change W’s tentative label to a permanent label.



Dijkstra's Shortest Path Example

Destination is permanently labeled – copy path back to S and read in reverse



Reverse path = j – h – k – g

Path = g – k – h – j