Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

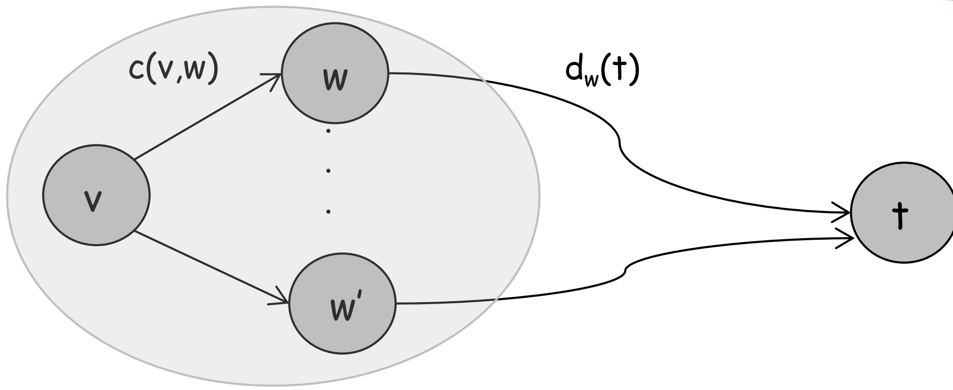
5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Bellman-Ford Algorithm (Dynamic Programming)



Let $d_v(t)$:= cost of shortest path from v to t

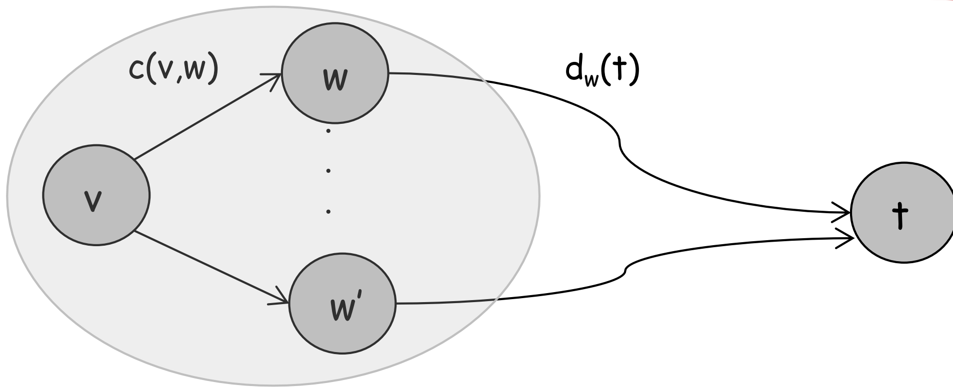
Then $d_v(t) = \min_w \{c(v,w) + d_w(t)\}$

cost from neighbor w to destination t

cost from v to neighbor w

min taken over all neighbors w of v

Single destination "t" can be implicit



Let $d_v\cancel{(t)}$:= cost of shortest path from v to t

Then $d_v\cancel{(t)} = \min_w \{c(v,w) + d_w\cancel{(t)}\}$

cost from neighbor w to destination t

cost from v to neighbor w

min taken over all neighbors w of v

Shortest path from v ~~to t~~

- Each node v maintains local information
- 1- d(v): length of shortest path from v to t.
- 2- w: the next hop after v, on the shortest path to t.

- Each node makes local decisions

$$d(v) = \min_{w:(v,w)\in E} \{ d(w) + c_{vw} \}$$

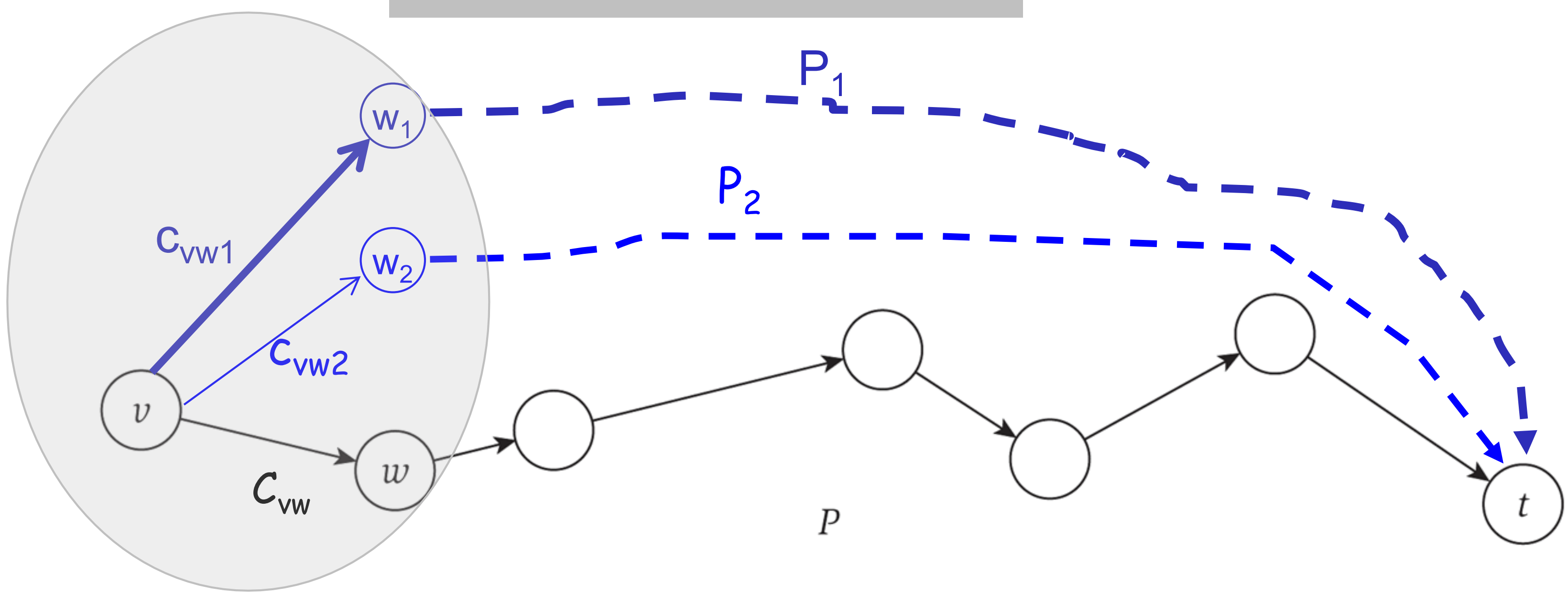
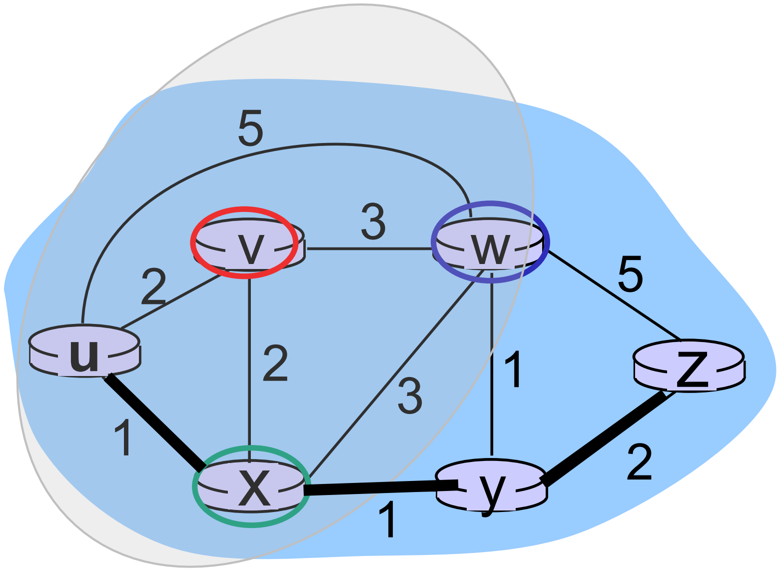


Figure 6.22 The minimum-cost path P from v to t using at most i edges.

Bellman-Ford equation example: $d_u(z)$



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

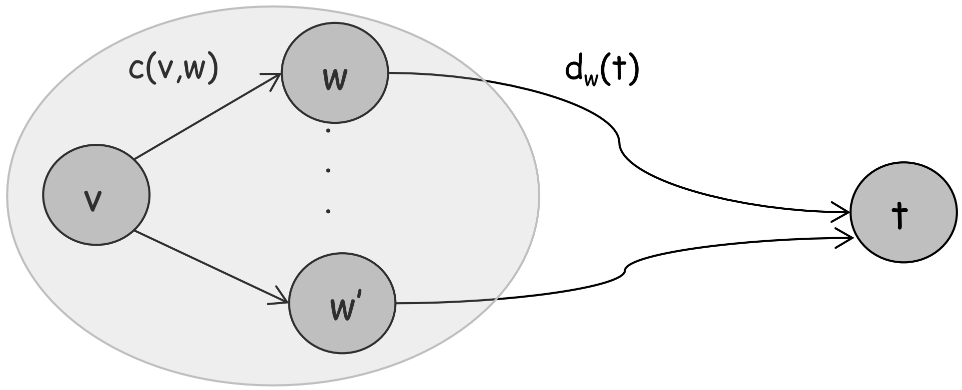
Compute $\mathbf{d_u(z)}$ according to B-F:

$$\mathbf{d_u(z)} = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$

$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

- $d_u(z)$=4 via next hop x
- Entry at Forwarding Table at u: next hop to **z** is **x**

Single node's operation



- v's local view
  - v knows the costs c(v,w) to all his neighbors w
  - v exchanges updates [when? how?] with this neighbors about their $d_w(t)$ (and his $d_v(t)$)
- v locally computes: $d_v(t) = \min_w \{c(v,w) + d_w(t)\}$
- v's forwarding table entry
  - $(t, w^*, d_v(t))$
  - w^*: the next hop after v, on the shortest path to t
  - $d_v(t)$: length of shortest path from v to t

- Does the same for all possible destinations t ϵ N
  - v maintains distance **vector** $\mathbf{d}_v = [d_v(t): t \in N]$ to all t ϵ N.

Distance vector algorithm

key idea:

- from time-to-time [when??], each node v sends its own distance vector estimate to neighbors
- When v receives new DV estimate from neighbor, it updates its own DV using Bellman-Ford equation:

$$d_v(t) \leftarrow \min_w\{c(v,t) + d_w(t)\} \quad \textit{for each node } t \in N$$

- under minor, natural conditions, the estimate $d_v(t)$ *converge to the actual least cost* $d_v(t)$

Distance vector algorithm

iterative, synchronous:

each node:

In each iteration

↓

Exchange DV with all neighbors

↓

Recompute estimates to all destinations

↓

Until the estimate does not change

Example of Synchronous Execution: **distance to x**

node x table

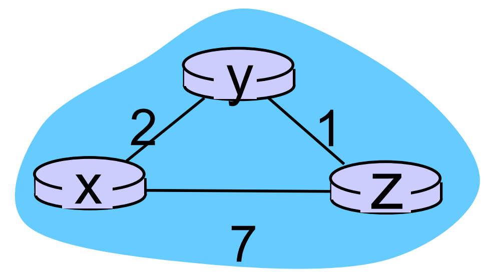
from \ *cost to*	x
x	0

node y table

from \ *cost to*	x
y	2

node z table

from \ *cost to*	x
z	7



time

Example of Synchronous Execution: **distance to x**

node x table

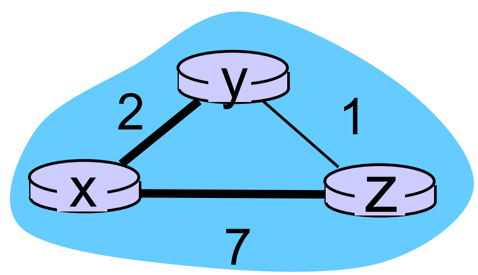
from \ cost to	x
x	0

node y table

from \ cost to	x
y	2

node z table

from \ cost to	x
z	7



time

Example of Synchronous Execution: **distance to x**

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

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

node x table

from \ *cost to*	x
x	0

from \ *cost to*	x
x	0
y	2
z	7

node y table

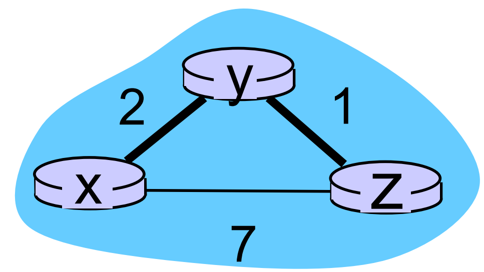
from \ *cost to*	x
y	2

from \ *cost to*	x
x	0
y	2
z	7

node z table

from \ *cost to*	x
z	7

from \ *cost to*	x
x	0
y	2
z	3



time

Example of Synchronous Execution: **distance to x**

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

node x table

from \ cost to	x
x	0

from \ cost to	x
x	0
y	2
z	7

from \ cost to	x
x	0
y	2
z	3

node y table

from \ cost to	x
y	2

from \ cost to	x
x	0
y	2
z	7

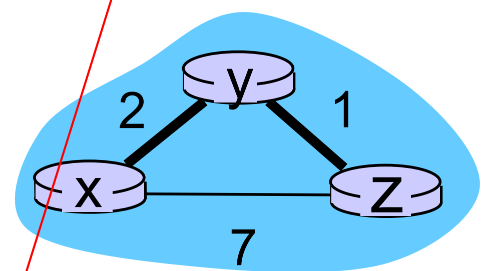
from \ cost to	x
x	0
y	2
z	3

node z table

from \ cost to	x
z	7

from \ cost to	x
x	0
y	2
z	3

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



time

Example of Synchronous Execution: **all dest**

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

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

node x table

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

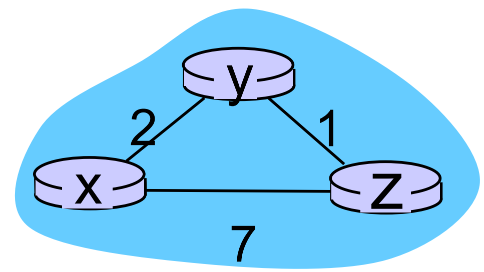
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

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

Example of Synchronous Execution: **all dest**

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

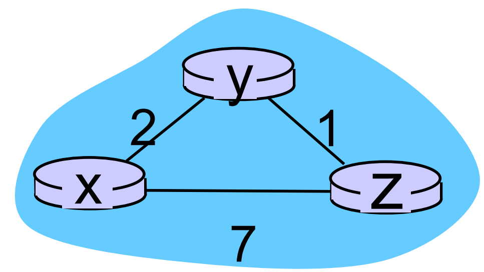
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

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

Distance vector algorithm

iterative, asynchronous:
each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed, push-based:

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

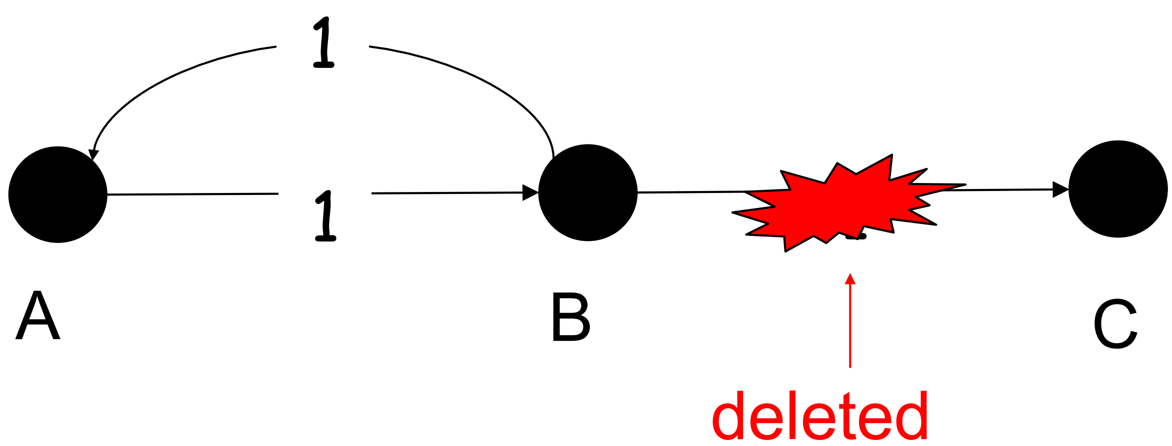
each node:

wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

"Counting to Infinity" Example

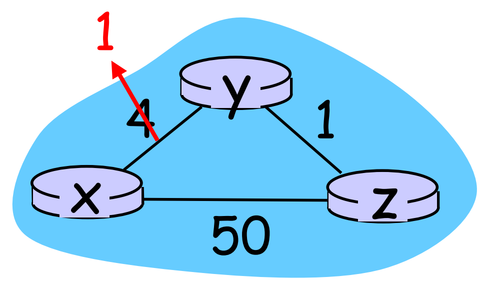


- Originally:
  - B to C: via C, cost 1,
  - A to C: via B, cost 2
- After link failure:
  - B to C: via A, cost 3
  - A to C: via B, cost 4
  - B to C: via A, cost 5
  - A to C: via B, cost 6,
  - ++
- Fixes: (1) max hop count (2) poisonous reverse (3) keep path state.

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



"good news travels fast"

t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

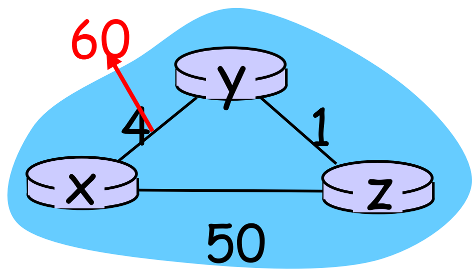
t_2 : *y* receives *z*'s update, updates its distance table. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*.

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- 44 iterations before convergence
  - y to x: 4 via x
  - z to x: 5 via y
  - change of c(x,y): 4→60
  - y to x: 6 via z
  - z to x: 7 via y
  - …
  - Counting to 50



Solutions:

- *Poisoned Reverse:* if Z routes through Y to get to X, then Z advertises to Y that its (Z's) distance to X is infinite (so Y won't route to X via Z)
- Will this completely solve count to infinity problem?
  - No, if loops involve more than 2 nodes.
- Other fixes: (1) max hop count (2) keep path state.

Comparison of Link State and Distance Vector

- ***In LS:*** each node broadcasts to all other nodes information about its neighborhood (neighbors+link costs)
- ***In DV:*** each node talks only to its neighbors and tells them its estimates of path costs to all other destinations

message complexity

- with n nodes, E links
- ***LS:*** O(nE) msgs sent
- ***DV:*** exchange between neighbors
  - convergence time varies

speed of convergence

- ***LS:***
  - Naïve: $O(n^2)$
  - Efficient implementation: O(mlogE)
  - oscillations for load-dependent weights
- ***DV:*** convergence time varies
  - Efficient implementation: O(mn) time, O(m+n) memory
  - changes may cause routing loops
  - count-to-infinity problem

robustness: what if router malfunctions?

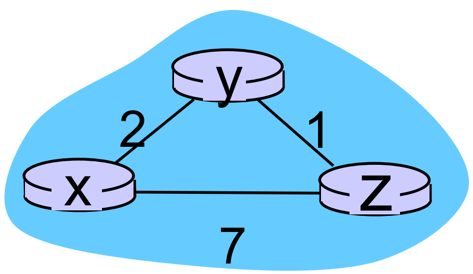
LS:

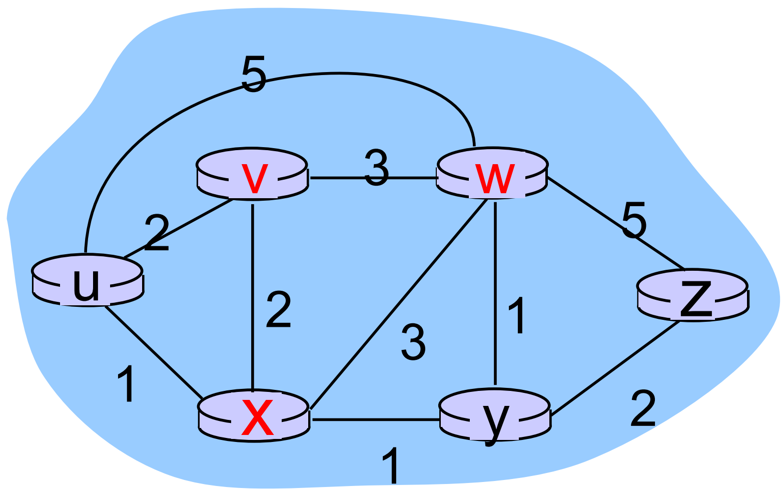
- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

Practice examples





More practice on routing algorithms

- Find shortest paths from s to all other nodes, using Dijkstra
- Find shortest paths from all nodes to T, using Bellman-Ford
  - Synchronous execution, asynchronous execution
  - Consider that a link fails or changes cost. What happens?
- Discussion session. Interactive exercises. HW4