

## ΗΥ-335 : Δίκτυα Υπολογιστών

### Επίπεδο Δικτύου – Αλγόριθμοι Δρομολόγησης

Μαρία Παπαδοπούλη

Τμήμα Επιστήμης Υπολογιστών

Πανεπιστήμιο Κρήτης

Χειμερινό εξάμηνο 2014-2015

# Θέματα προς συζήτηση

... Ερωτήσεις από τα προηγούμενα lectures ...

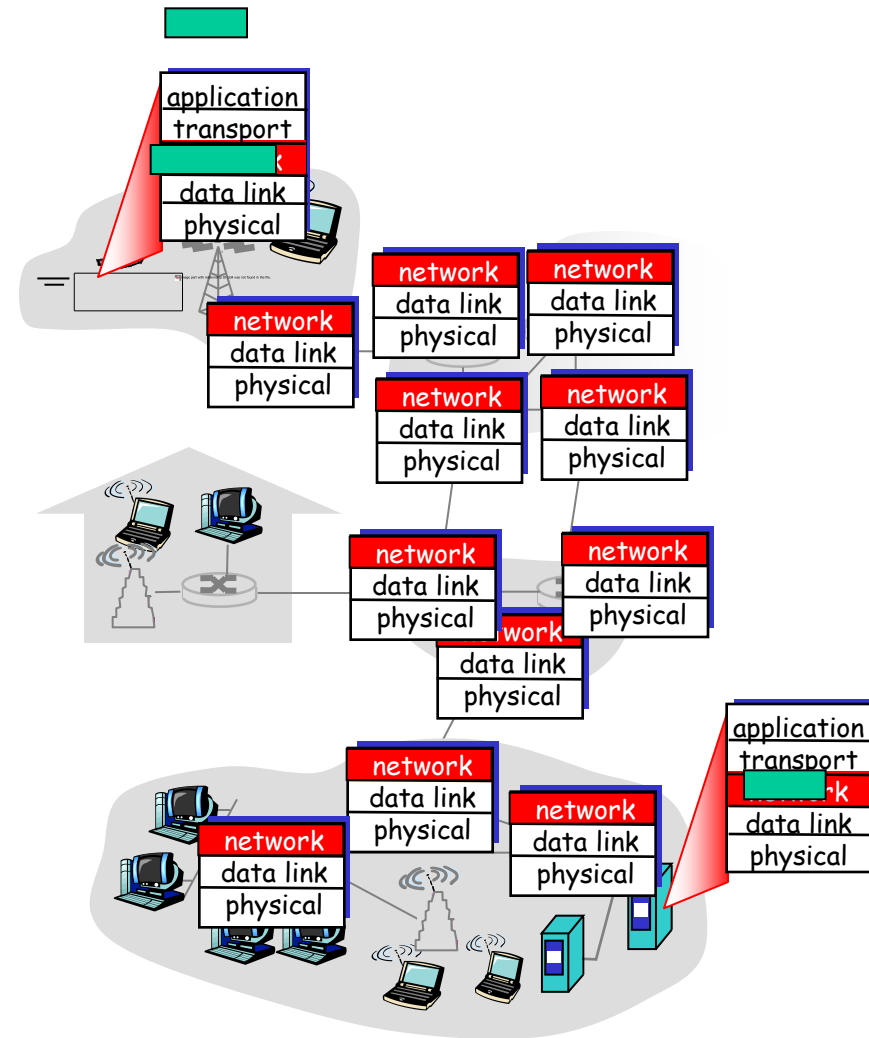
## Αλγόριθμοι δρομολόγησης

- Κατάστασης Ζεύξεων (Link state)
- Διανυσμάτων Απόστασης (Distance Vector)
- Ιεραρχικής Δρομολόγησης (Hierarchical routing)

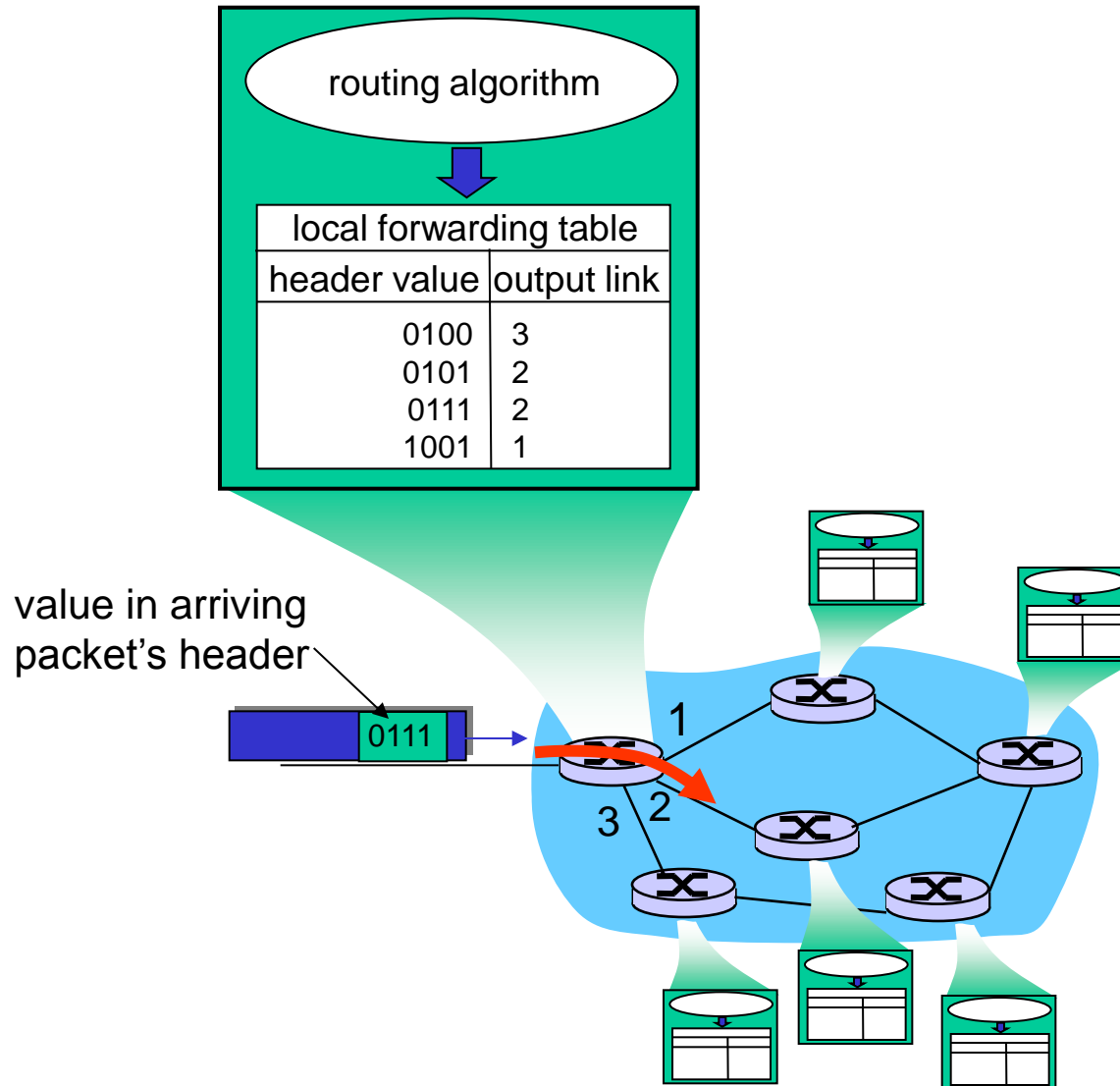
Βασισμένο κυρίως στο **Κεφ. 4** του βιβλίου των **Kurose/Ross**

# Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on rcving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it



# Interplay between routing and forwarding



# Forwarding table

4 billion  
possible entries

<u>Destination Address Range</u>	<u>Link Interface</u>
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

# Longest prefix matching

<u>Prefix Match</u>	<u>Link Interface</u>
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

## Examples

DA: 11001000 00010111 00010110 10100001

Which interface?

DA: 11001000 00010111 00011000 10101010

Which interface?

# IP datagram format

IP protocol version number

header length (bytes)

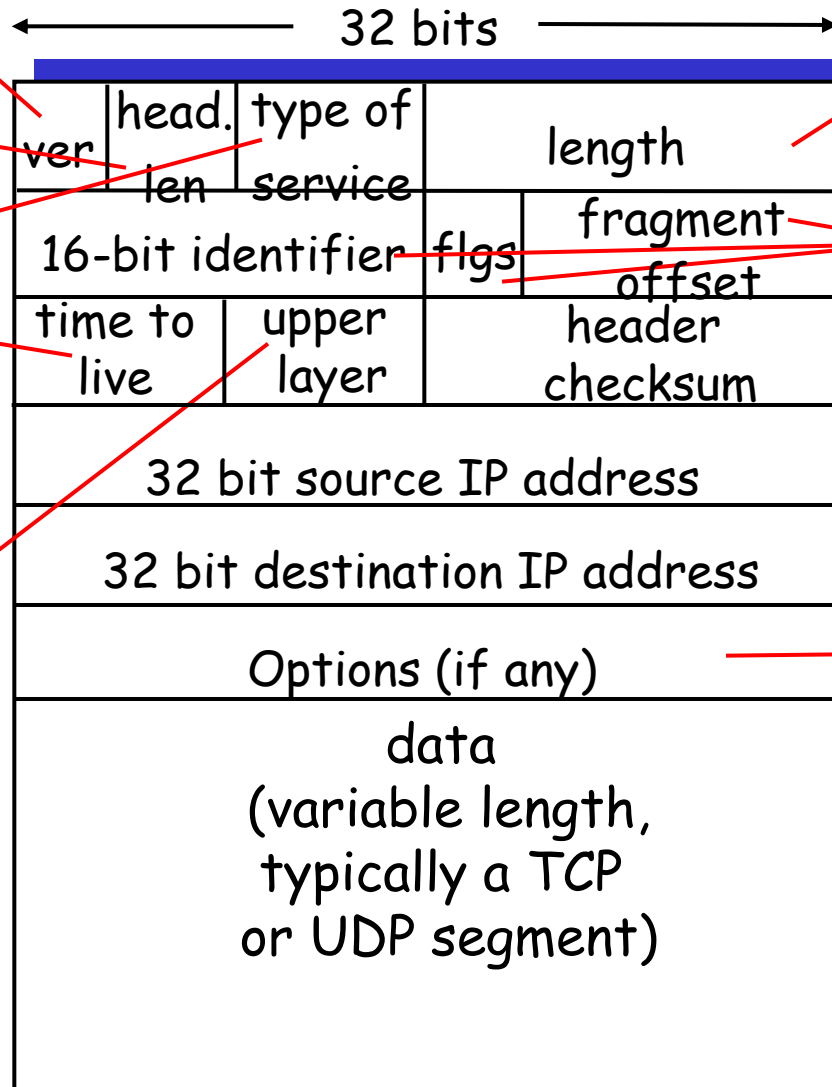
"type" of data

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to

## how much overhead with TCP?

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead



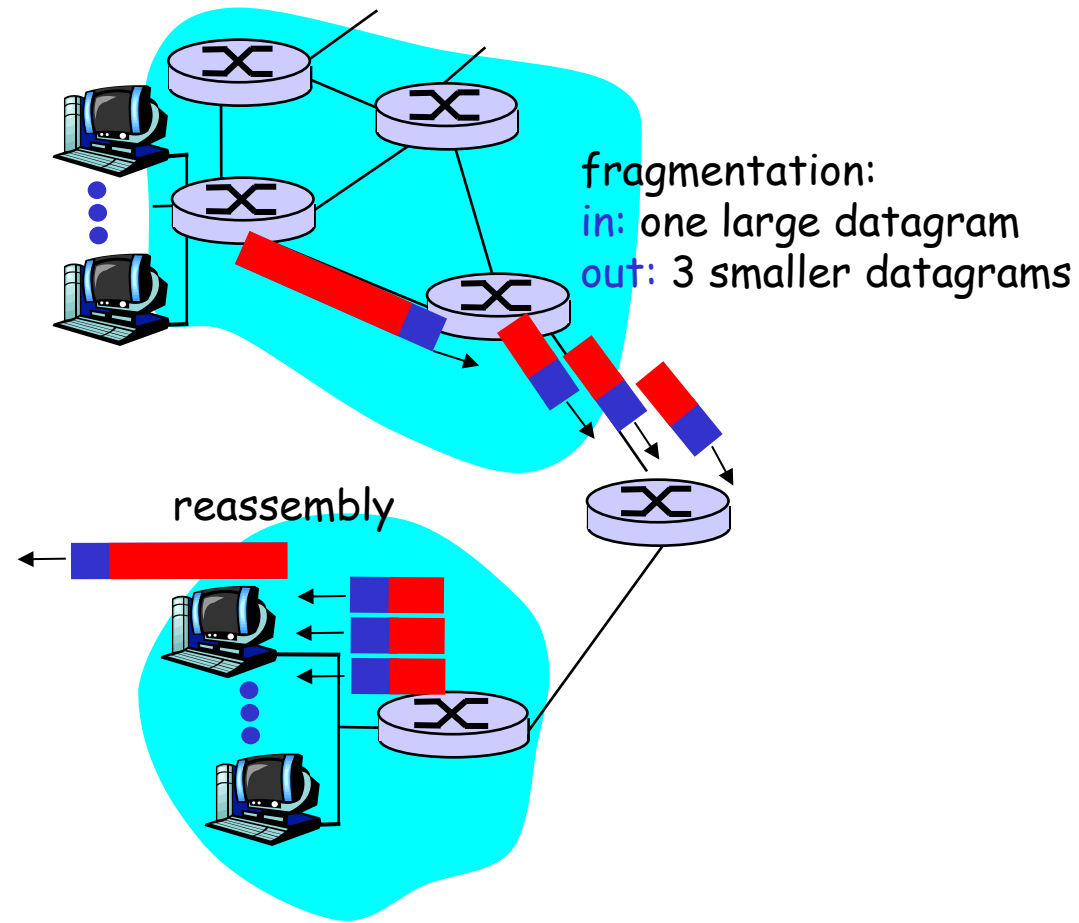
total datagram length (bytes)

for fragmentation/reassembly

E.g. timestamp, record route taken, specify list of routers to visit.

# IP Fragmentation & Reassembly

- network links have MTU (max.transfer size) - largest possible link-level frame
- different link types  $\Rightarrow$  different MTUs
- large IP datagram divided ("fragmented") within net



- one datagram becomes several datagrams
- "reassembled" only at **final destination**
- IP header bits used to identify, order related fragments



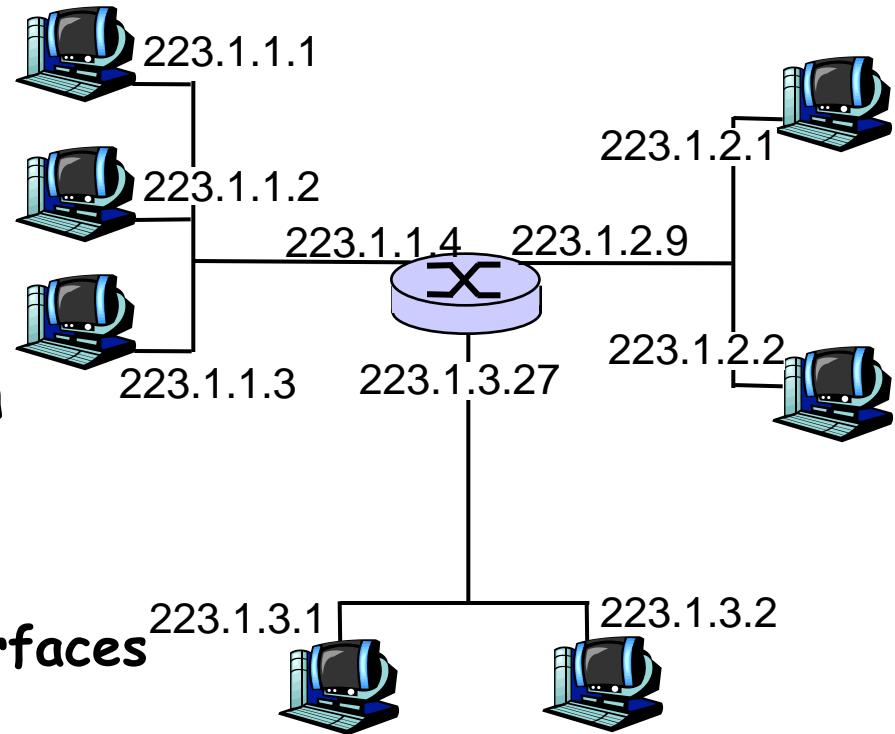
# IP Addressing: introduction

**IP address:** 32-bit identifier for the network interface of a host or router

**interface:** connection between host/router & physical link

router's typically have **multiple interfaces**

**IP addresses associated with each interface**



223.1.1.1 =  $\underbrace{11011111}_{223} \underbrace{00000001}_1 \underbrace{00000001}_1 \underbrace{00000001}_1$

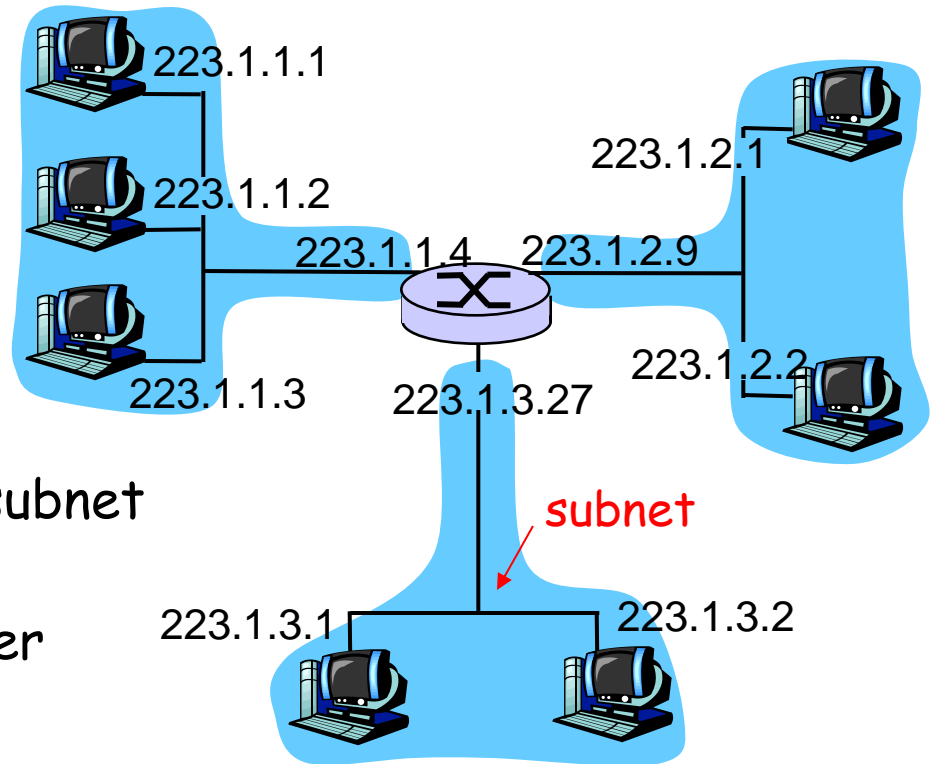
# Subnets

## IP address:

- subnet part (high order bits)
- host part (low order bits)

## *What's a subnet ?*

- device interfaces with same subnet part of IP address
- can physically reach each other without intervening router

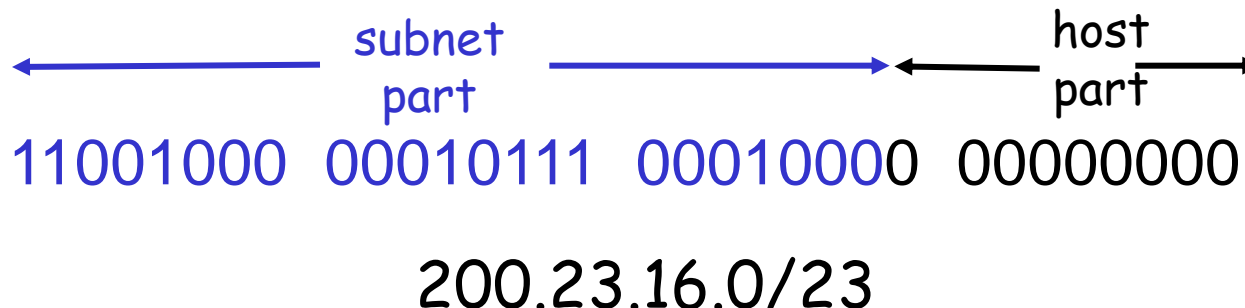


network consisting of 3 subnets

# IP addressing: CIDR

## CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format:  $a.b.c.d/x$ , where  $x$  is # bits in subnet portion of address



# IP addresses: how to get one?

Q: How does a *host* get IP address?

- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- **DHCP: Dynamic Host Configuration Protocol:** dynamically get address from as server
  - “plug-and-play”

# IP addresses: how to get one?

Q: How does *network* get subnet part of IP addr?

A: gets allocated portion of its provider ISP's address space

ISP's block	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000	200.23.16.0/20
Organization 0	<u>11001000</u>	<u>00010111</u>	<u>00010000</u>	00000000	200.23.16.0/23
Organization 1	<u>11001000</u>	<u>00010111</u>	<u>00010010</u>	00000000	200.23.18.0/23
Organization 2	<u>11001000</u>	<u>00010111</u>	<u>00010100</u>	00000000	200.23.20.0/23
...	.....	.....	.....	.....	.....
Organization 7	<u>11001000</u>	<u>00010111</u>	<u>00011110</u>	00000000	200.23.30.0/23

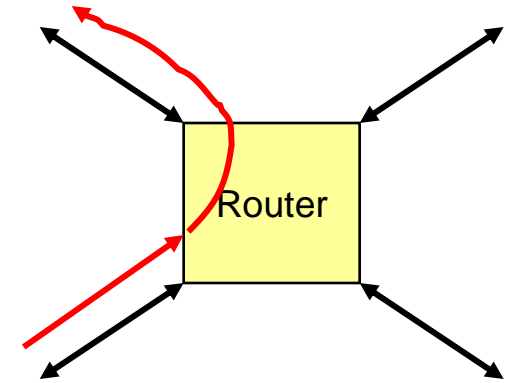
## IP addressing: the last word...

Q: How does an ISP get block of addresses?

A: **ICANN:** Internet **C**orporation for **A**ssigned  
**N**ames and **N**umbers

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

# Περίληψη



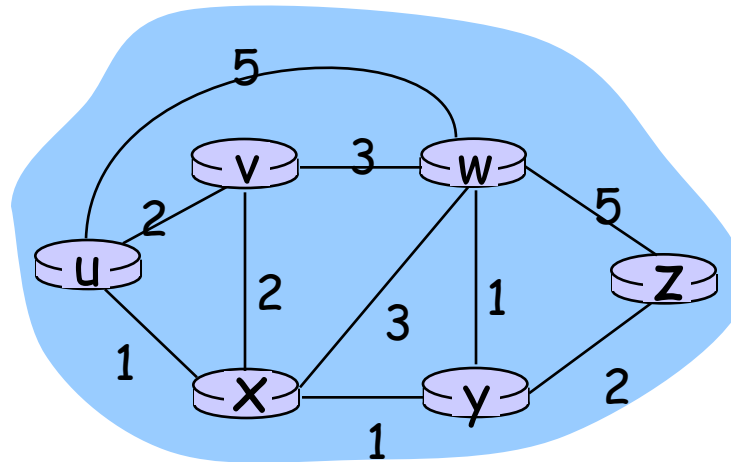
Η ιστορία μας μέχρι τώρα ...

- Οι IP διευθύνσεις αντανακλούν τη δομή του Internet  
όπως οι τηλεφωνικοί αριθμοί
- Οι IP επικεφαλίδες πακέτων "φέρουν" αυτή τη πληροφορία
- Όταν το πακέτο φτάνει στον δρομολογητή αυτός
  - **Εξετάζει την επικεφαλίδα** για να προσδιορίσει τον **προορισμό**
  - **Ψάχνει στον πίνακα** για να προσδιορίσει **τον επόμενο κόμβο στο μονοπάτι**
  - **Στέλνει το πακέτο** στην κατάλληλη θύρα

Σημερινή διάλεξη

Πως δημιουργείται ο πίνακας δρομολόγησης

# Αναπαράσταση γράφου (Graph abstraction)



Γράφος:  $G = (N, E)$

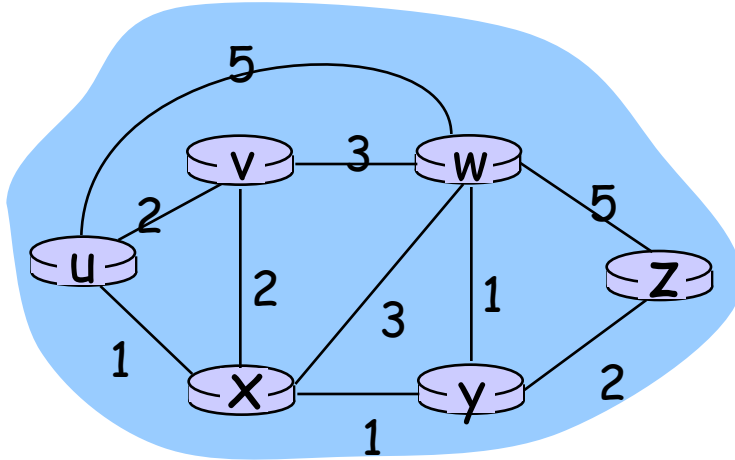
$N =$  σύνολο δρομολογητών =  $\{u, v, w, x, y, z\}$

$E =$  σύνολο ζεύξεων =  $\{(u, v), (u, x), (v, x), (v, w), (x, w), (x, y), (w, y), (w, z), (y, z)\}$



# Αναπαράσταση γράφου: κόστη

- $c(x,x')$  = κόστος ζεύξης  $(x,x')$   
- π.χ.,  $c(w,z) = 5$



- ☞ Το κόστος μπορεί να είναι:  
Se oles tis akmes 1, ή  
αντιστρόφως ανάλογο του **εύρους ζώνης**,  
ή ανάλογο της **συμφόρησης**

Κόστος μονοπατιού  $(x_1, x_2, x_3, \dots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

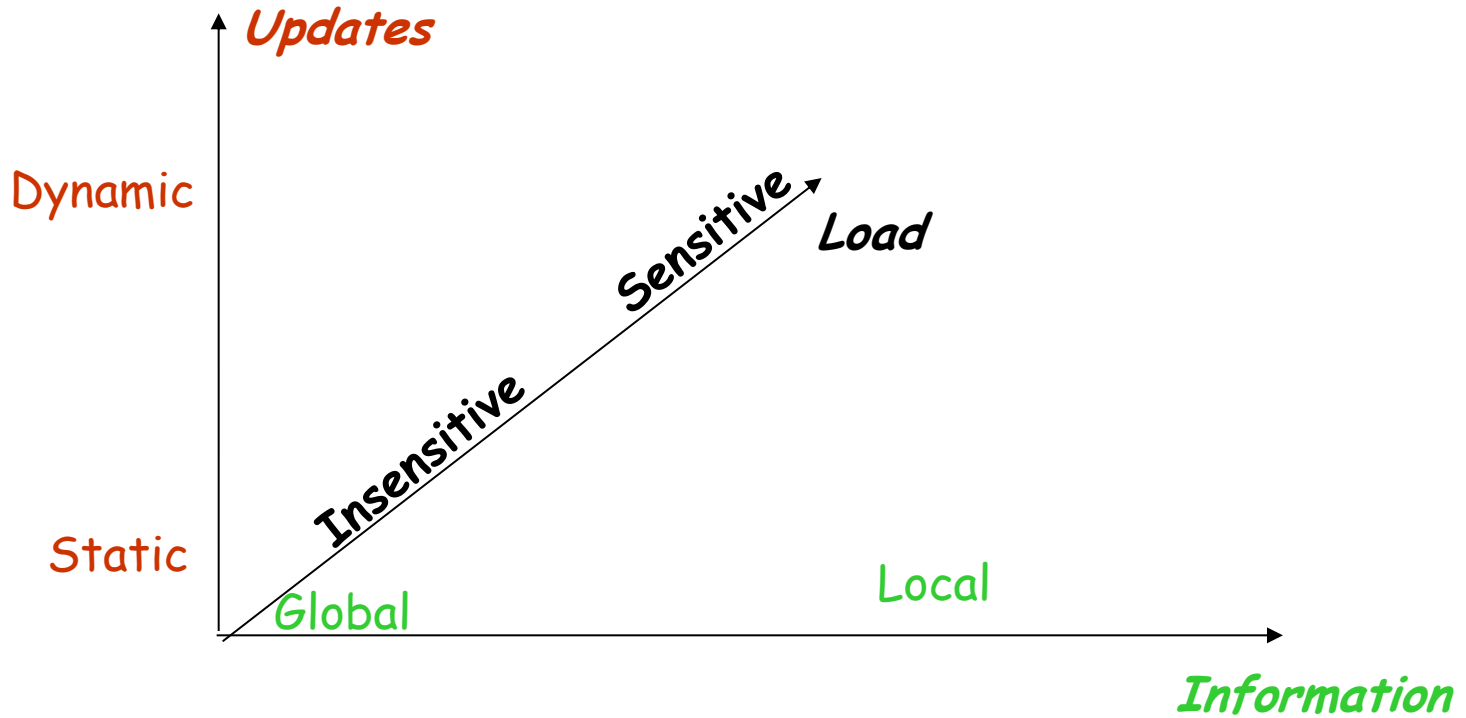
**Ερώτηση: Ποιό είναι το μονοπάτι με το μικρότερο κόστος μεταξύ των u και z;**

**Αλγόριθμος δρομολόγησης: αλγόριθμος που βρίσκει το μονοπάτι με το ελάχιστο κόστος**

# Αλγόριθμοι δρομολόγησης

- **Δυναμικοί αλγόριθμοι:** αλλάζουν τα μονοπάτια δρομολόγησης καθώς ο φόρτος κίνησης του δικτύου ή η τοπολογία του αλλάζουν
- Μπορεί να τρέχουν
  - ✓ περιοδικά ή
  - ✓ σε άμεση απάντηση αλλαγών στην τοπολογία και στα κόστη των ζεύξεων
- ☹ Επιρρεπή σε προβλήματα όπως επαναλήψεις διαδρομών ή διακυμάνσεις στις διαδρομές
- **Ευαίσθητα στον φόρτο:** τα κόστη των ζεύξεων αλλάζουν δραματικά για να αναπαριστούν το τωρινό επίπεδο συμφόρησης της ζεύξης
- ☞ Οι σημερινοί αλγόριθμοι δρομολόγησης (π.χ., RIP, OSPF, BGP) **δεν λαμβάνουν υπόψη το φόρτο**

# Ταξινόμηση Αλγορίθμων Δρομολόγησης



# Ταξινόμηση Αλγορίθμων Δρομολόγησης

☞ Αντιπαράθεση καθολικής & τοπικής πληροφορίας

## Καθολική:

- Όλοι οι δρομολογητές έχουν **πλήρη** εικόνα της τοπολογίας & κόστους ζεύξεων
- **Αλγόριθμοι κατάστασης ζεύξης (*link-state*)**

## Τοπική:

- Ο δρομολογητής ξέρει τους **φυσικά-συνδεδεμένους γείτονες του**, κόστη ζεύξεων προς τους γείτονες
- Επαναληπτική διαδικασία υπολογισμού και ανταλλαγής πληροφορίας με τους γείτονες
- **Αλγόριθμοι πίνακα αποστάσεων (*distance-vector*)**

# Ταξινόμηση Αλγορίθμων Δρομολόγησης

- ➔ **Αντιπαράθεση στατικών & δυναμικών αλγορίθμων**
  - Στατικοί:** οι διαδρομές αλλάζουν αργά με την πάροδο του χρόνου
  - Δυναμικοί:** οι διαδρομές αλλάζουν πιο γρήγορα
    - Περιοδική ενημέρωση
    - Ως απάντηση σε αλλαγές κόστους ζεύξεων

# Τρόποι υπολογισμού συντομότερων μονοπατιών

## Κεντριοποιημένοι

- Συλλογή δομής γράφου σε ένα μέρος
- Χρήση τυπικού αλγορίθμου γράφου
- Διάδοση πινάκων δρομολόγησης

## Κατάστασης ζεύξεων (Link-state)

-----> OSPF

- Κάθε κόμβος συλλέγει την πλήρη δομή του γράφου
- Καθένας υπολογίζει τα συντομότερα μονοπάτια
- Καθένας παράγει το δικό του πίνακα δρομολόγησης

RIP (ένα από τα παλιότερα πρωτόκολλα δρομολόγησης τον χρησιμοποιεί) ↑

## Διανυσμάτων απόστασης (Distance-vector)

----->

- Κανένας δεν έχει αντίγραφο του γράφου
- Οι κόμβοι δημιουργούν τους δικούς τους πίνακες επαναληπτικά
- Ο καθένας στέλνει πληροφορίες για τον πίνακά του στους γείτονες

Και οι 2 χρησιμοποιούνται στις μέρες μας

## Να θυμάστε για τους link-state & distance-vector:

- Δρομολόγηση κατάστασης ζεύξεων (link state):  
ένας κόμβος προσπαθεί να φτιάξει **μία πλήρη εικόνα του δικτύου** με το να "φωνάζει" ("πλημμυρίζει")
- Διανυσμάτων απόστασης (distance vector):  
Ένας κόμβος **ενδιαφέρεται μόνο για τους γείτονές του** και παίρνει **τοπική πληροφορία**  
  
☞ **Δεν υπάρχει καθολική θεώρηση του δικτύου**

# Πρωτόκολλο κατάστασης ζεύξεων (link-state)

Κάθε κόμβος παίρνει ένα *πλήρες αντίγραφο του γράφου*

Κάθε κόμβος "**πλημμυρίζει**" το δίκτυο με δεδομένα σχετικά με τις εξερχόμενες ζεύξεις του

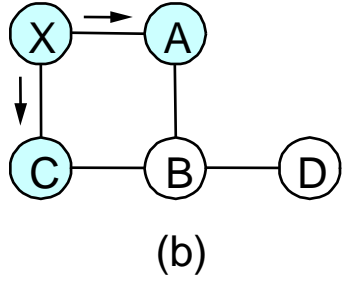
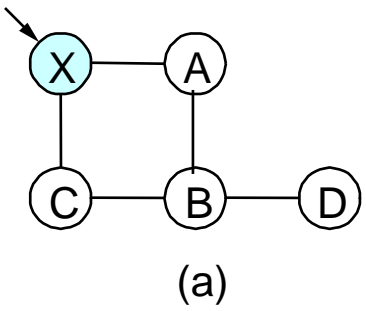
Κάθε κόμβος υπολογίζει τις διαδρομές *προς κάθε άλλον κόμβο*

- Χρησιμοποιώντας τον αλγόριθμο **μοναδικής πηγής, ελαχίστου μονοπατιού**
- Η διαδικασία γίνεται **όποτε χρειάζεται**
  - Όταν οι συνδέσεις κόβονται/επανεμφανίζονται

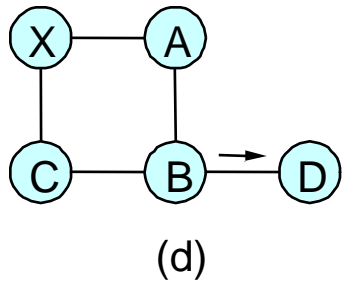
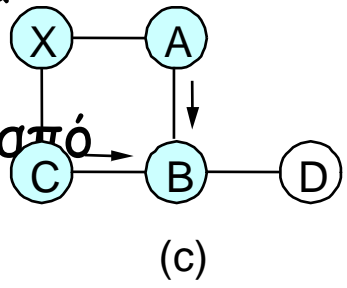


# Αποστολή καταστάσεων ζεύξεων “πλημμυρίζοντας” το δίκτυο

Ο Χ θέλει να στείλει πληροφορία  
Στέλνει σε **όλες τις εξερχόμενες ζεύξεις**



Όταν ο κόμβος Y λαμβάνει πληροφορία  
από τον Z  
Στέλνει σε **όλες τις ζεύξεις εκτός από την Z**



☺ Η μέθοδος της “πλημμύρας” είναι ένα συνηθισμένο πρωτόκολλο για την διάδοση πληροφορίας στο δίκτυο (περιοδικά ή μετά από γεγονότα)

# Ένας αλγόριθμος κατάστασης ζεύξεων (link state)

## 📢 Ο αλγόριθμος του Dijkstra

- Η τοπολογία του δικτύου και τα **κόστη των ζεύξεων είναι γνωστά σε όλους τους κόμβους**

- 📁 Πετυχαίνεται μέσω μετάδοσης της κατάστασης των ζεύξεων
- Όλοι οι κόμβοι έχουν τις ίδιες πληροφορίες

- Υπολογίζει **τα μονοπάτια ελαχίστου κόστους** από έναν κόμβο ("**πηγή**") προς όλους τους άλλους

- Δίνει τον **πίνακα προώθησης** για αυτόν τον κόμβο

- επαναληπτικός: ύστερα από  $k$  επαναλήψεις, γνωρίζει τα μονοπάτια ελαχίστου κόστους προς  $k$  προορισμούς

## Συμβολισμός:

• Κόμβος πηγής:  $s$

• " $\rightarrow$ ": μονοπάτι, " $\rightarrow$ ": ζεύξη

•  $c(x,y)$ : κόστος ζεύξης  $x \rightarrow y$ ;  
=  $\infty$  αν **δεν** είναι άμεσοι γείτονες

•  $D(v)$ : **τωρινό** κόστος μονοπατιού  
 $s \rightarrow v$

•  $p(v)$ : προηγούμενος κόμβος στο μονοπάτι  $s \rightarrow v$

•  $N'$ : {κόμβοι των οποίων το μονοπάτι ελαχίστου κόστους είναι γνωστό}

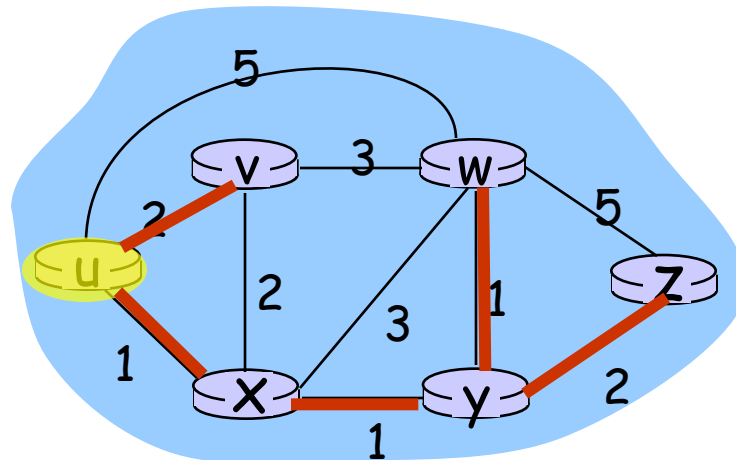
# Αλγόριθμος του Dijkstra

```
1 Initialization (Αρχικοποίηση):
2  $N' = \{u\}$ 
3 for all nodes  $v$ 
4 if  $v$  adjacent to  $u$  then
5      $D(v) = c(u, v)$ 
6 else  $D(v) = \infty$ 
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         $D(v) = \min( D(v), D(w) + c(w, v) )$ 

/* new cost to v is either old cost to v or known
shortest path cost to w plus cost from w to v */
13 until all nodes in  $N'$ 
```

# Αλγόριθμος του Dijkstra: παράδειγμα

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	$\infty$	$\infty$
1	ux	2,u	4,x		2,x	$\infty$
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxynw					4,y
5	uxynwz					



Η διαδικασία εκτελείται από τον κάθε δρομολογητή στο δίκτυο  
 Use of global information about the network topology & costs

# Συζήτηση για τον αλγόριθμο του Dijkstra

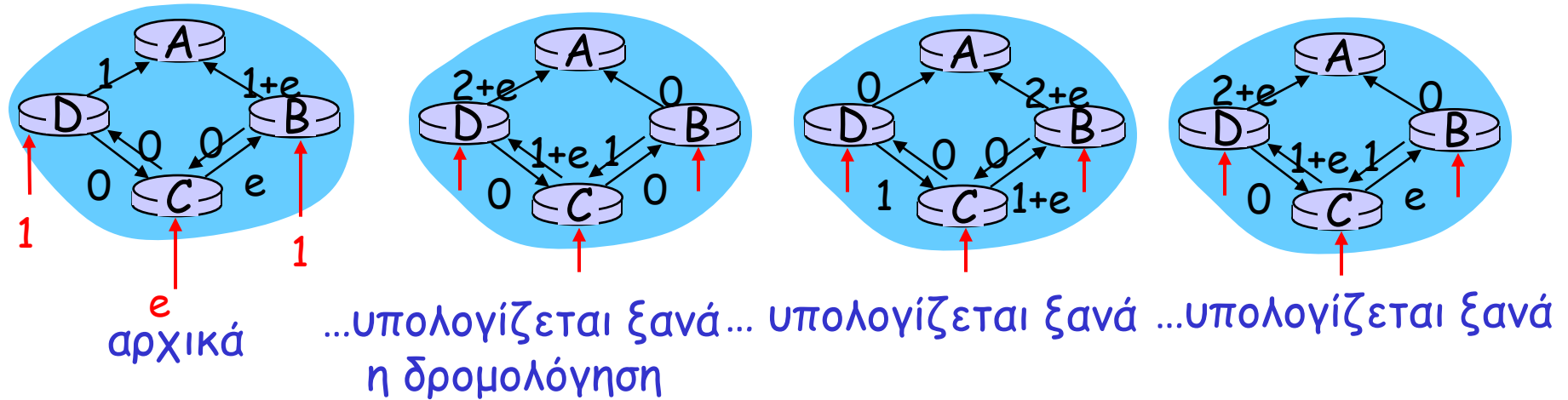
**Αλγοριθμική πολυπλοκότητα:**  $n$  κόμβοι

- Σε κάθε επανάληψη: χρειάζεται να ελέγξει όλους τους κόμβους  $w$ , που δεν ανήκουν στο σύνολο  $N$
- $n(n+1)/2$  συγκρίσεις  $\Rightarrow O(n^2)$
- ☞ Πιο αποδοτικές υλοποιήσεις είναι πιθανές:  $O(n \log n)$

**Πιθανές παραλλαγές:**

π.χ., κόστος ζεύξης = ποσότητα μεταφερόμενης κίνησης

# Oscillation



- Not all routers run simultaneously the algorithm
- Introduce randomization purposefully into the period between execution instants of the algorithm at each node

Δρομολόγηση κατάστασης ζεύξεων:

ένας κόμβος προσπαθεί να πάρει μία πλήρη εικόνα του δικτύου με το να "φωνάζει" ("πλημμυρίζει")

Διανύσματα-απόστασης (distance vector):

Ένας κόμβος ενδιαφέρεται μόνο για τους γείτονές του και παίρνει τοπική πληροφορία

Δεν υπάρχει καθολική θεώρηση του δικτύου

# Αλγόριθμος διανυσμάτων απόστασης (Distance-vector)

## Βασική ιδέα:

1. Κάθε κόμβος **περιοδικά** στέλνει τις δικές του εκτιμήσεις διανυσμάτων απόστασης **στους γείτονές του**
2. Όταν ένας κόμβος **x** λαμβάνει μία **νέα εκτίμηση** διανυσμάτων απόστασης από τον **γείτονα v**:  
ενημερώνει τον δικό του πίνακα διανυσμάτων απόστασης (DV) χρησιμοποιώντας την B-F εξίσωση:  
$$D_x(y) \leftarrow \min_v \{ c(x,v) + D_v(y) \} \quad \text{for each node } y \in N$$

$D_x(y)$  = εκτίμηση **ελαχίστου κόστους** από **x**  $\rightarrow$  **y**  
Ο κόμβος **x** διατηρεί το  $D_x = [ D_x(y) : y \in N ]$



# Αλγόριθμος διανυσμάτων απόστασης (Distance-Vector)

👉 Bellman-Ford εξίσωση (δυναμικός προγραμματισμός)

Ορίζομε  $D_x(y) :=$  κόστος τους μονοπατιού με το ελάχιστο κόστος από τον  $x$  στον  $y$

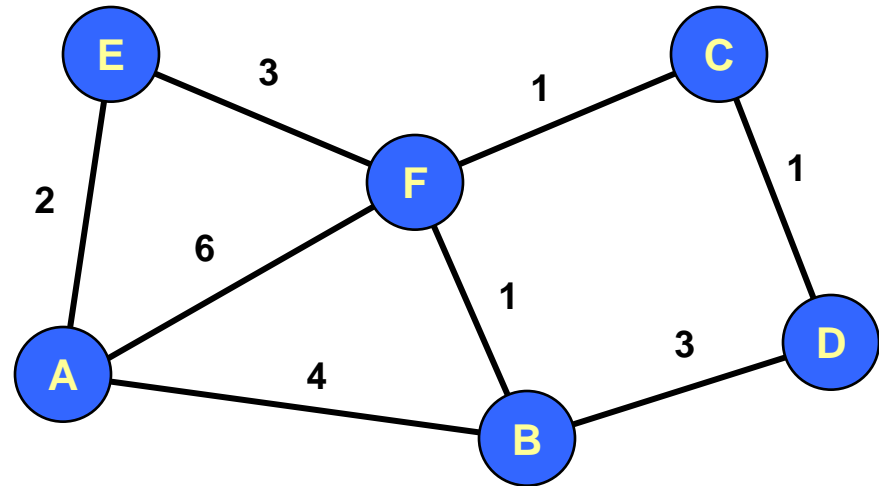
Τότε

$$D_x(y) = \min_v \{c(x,v) + D_v(y)\}$$

Όπου η ελάχιστη τιμή ελέγχεται για όλους τους γείτονες  $v$  του  $x$

# Μέθοδος διανυσμάτων απόστασης

Initial Table for A		
Dest	Cost	Next Hop
A	0	A
B	4	B
C	$\infty$	-
D	$\infty$	-
E	2	E
F	6	F

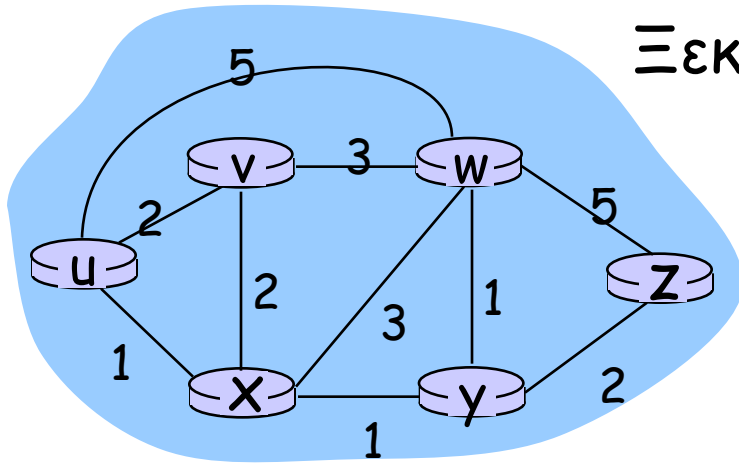


## Ιδέα

- Σε οποιαδήποτε στιγμή, έχουμε το κόστος/επόμενο κόμβο από το καλύτερο γνωστό μονοπάτι προς τον προορισμό
- Χρήση  $\infty$  κόστους όταν κανένα μονοπάτι δεν είναι γνωστό

Αρχικά: Υπάρχουν μόνο εγγραφές για τους άμεσα συνδεδεμένους γείτονες

# Bellman-Ford example



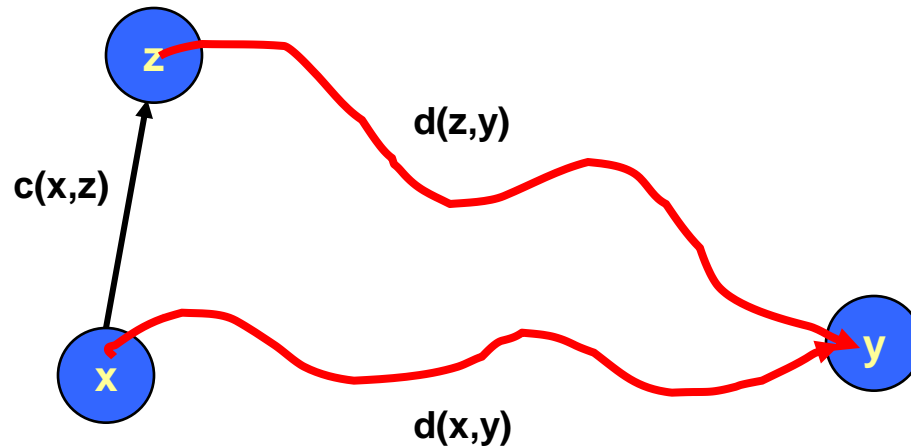
Ξεκάθαρα,  $d_v(z) = 5$ ,  $d_x(z) = 3$ ,  $d_w(z) = 3$

Η B-F εξίσωση λέει:

$$\begin{aligned} d_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\quad c(u,x) + d_x(z), \\ &\quad c(u,w) + d_w(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 1 + 3, \\ &\quad 5 + 3 \} = 4 \end{aligned}$$

Ο κόμβος που πετυχαίνει το ελάχιστο είναι ο επόμενος στο ελάχιστο μονοπάτι → πίνακας προώθησης

# Ενημέρωση του πίνακα διανυσμάτων απόστασης



Update(x,y,z)

$d \leftarrow c(x,z) + d(z,y)$  # Κόστος μονοπατιού από τον x στον y με 1<sup>ο</sup> κόμβο τον z

if  $d < d(x,y)$

# Βρέθηκε καλύτερο μονοπάτι

return d,z

# Ενημερωμένο κόστος/επόμενος κόμβος

else

return  $d(x,y)$ , nexthop(x,y)

# Υπάρχον κόστος/επόμενος κόμβος

# Αλγόριθμος διανυσμάτων απόστασης

- $D_x(y)$  = εκτίμηση του ελαχίστου κόστους από τον  $x \rightarrow y$
- Διάνυσμα απόστασης:  $D_x = [D_x(y): y \in N]$
- Ο κόμβος  $x$  γνωρίζει το κόστος  $x \rightarrow y$  :  $c(x,v)$
- ☞ Ο κόμβος  $x$  διατηρεί
  - $D_x = [D_x(y): y \in N]$   
και επίσης τους πίνακες διανυσμάτων απόστασης (DV) των γειτόνων του
  - Για κάθε γείτονα  $v$ :  $D_v = [D_v(y): y \in N]$

# Αλγόριθμοι διανυσμάτων απόστασης (cont'd)

## Επαναληπτικοί, ασύγχρονοι:

Κάθε τοπική επανάληψη προκαλείται από:

- **Τοπική αλλαγή κόστους ζεύξης**
- **Μήνυμα ενημέρωσης πίνακα διανυσμάτων απόστασης (DV) από κάποιον γείτονα**

## Κατανεμημένος:

☞ Κάθε κόμβος ειδοποιεί τους γείτονες **μόνο** όταν ο πίνακας διανυσμάτων απόστασης (DV) αλλάζει

- Οι γείτονες τότε ειδοποιούν τους γείτονές τους εάν είναι απαραίτητο

## Κάθε κόμβος:

**ΠΕΡΙΜΕΝΕΙ** για **αλλαγή** σε τοπικό κόστος ζεύξης ή μήνυμα από γείτονα

**ΥΠΟΛΟΓΙΖΕΙ** **ξανά** εκτιμήσεις

Εάν ο πίνακας διανυσμάτων απόστασης (DV) προς οποιονδήποτε προορισμό έχει αλλάξει, **ΕΙΔΟΠΟΙΕΙ** τους γείτονες

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

$$= \min\{2+0, 7+1\} = 2$$

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

$$= \min\{2+1, 7+0\} = 3$$

node x table

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

node y table

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

node z table

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

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

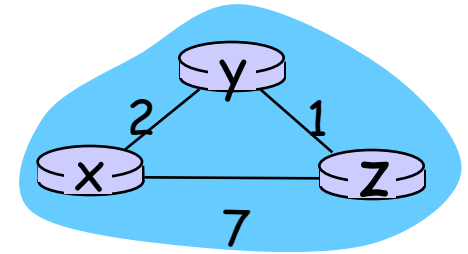
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

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

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

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

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

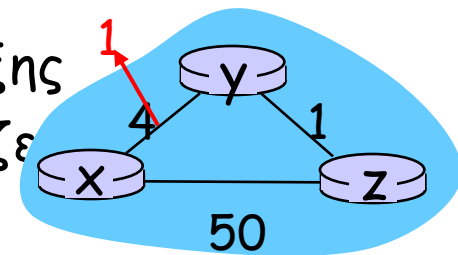


time →

# Πίνακας Διανυσμάτων Απόστασης: αλλαγές στα κόστη των ζεύξεων

## Αλλαγές στα κόστη των ζεύξεων:

- Ο κόμβος εντοπίζει τοπική αλλαγή στο κόστος μιας ζεύξης
- Ενημερώνει τις πληροφορίες δρομολόγησης και υπολογίζει ξανά τον πίνακα διανυσμάτων αποστάσης
- Εάν ο πίνακας διανυσμάτων αποστάσης (DV) αλλάξει, ειδοποιεί τους γείτονες



"good news travels fast"

Τη στιγμή  $t_0$ , ο **y** εντοπίζει αλλαγή σε κόστος ζεύξης, ενημερώνει τον πίνακά του και ειδοποιεί τους γείτονές του

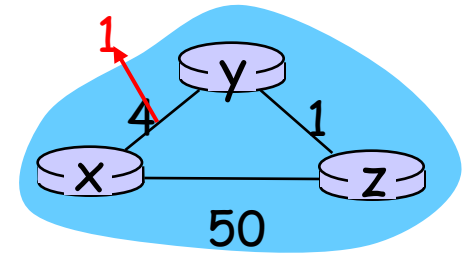
Τη στιγμή  $t_1$ , ο **z** λαμβάνει το μήνυμα του **y** και ενημερώνει τον πίνακά του. Υπολογίζει ένα νέο ελάχιστο κόστος προς τον **x** και στέλνει στους γείτονές του τον DV.

Τη στιγμή  $t_2$ , ο **y** λαμβάνει την ενημέρωση του **z** και ενημερώνει τον πίνακα αποστάσής του.

Τα ελάχιστα κόστη του **y** δεν αλλάζουν και για αυτό ο **y** δεν στέλνει κανένα μήνυμα στον **z**.



# Παράδειγμα (συνέχεια)

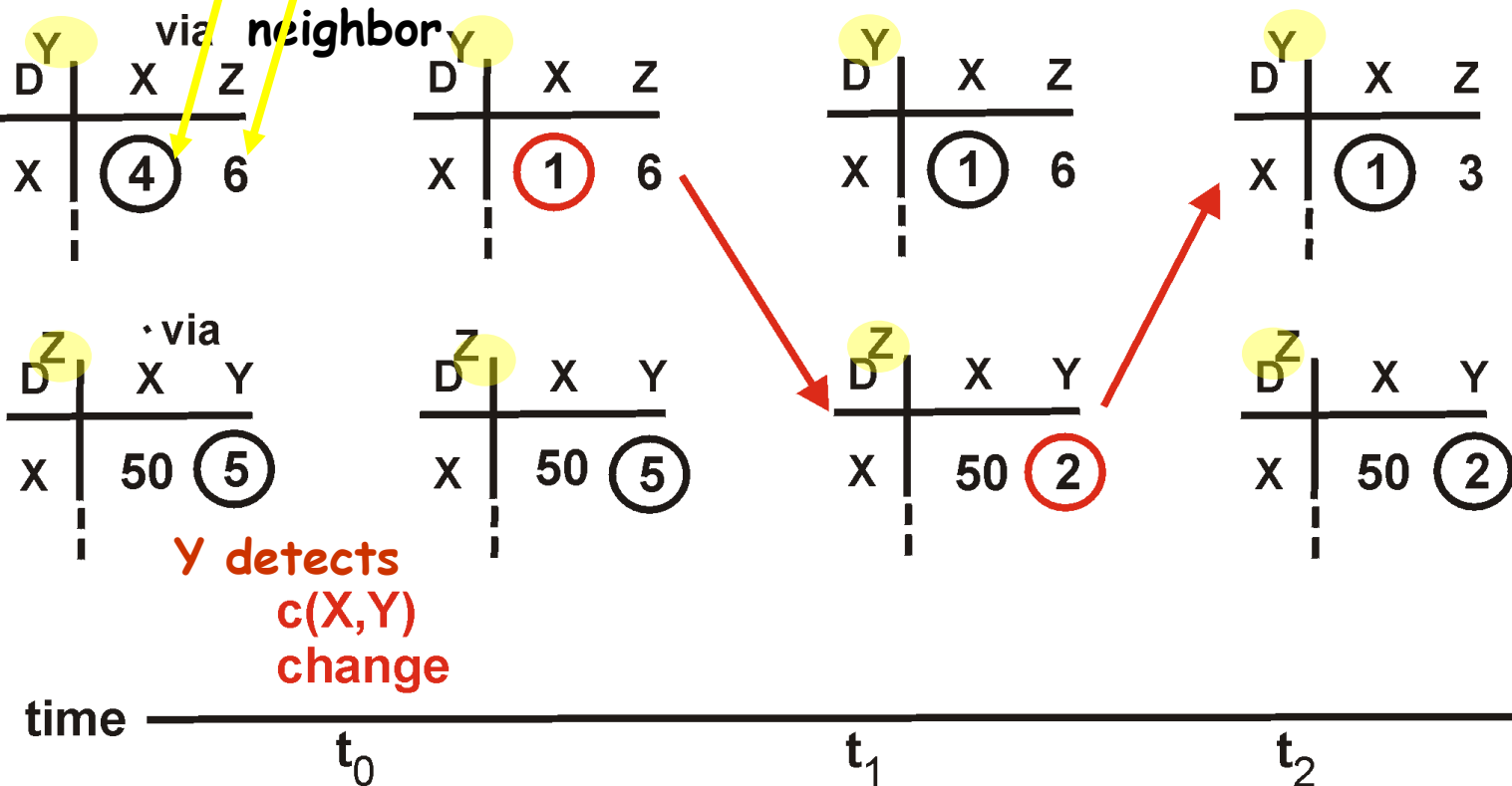


Συμβολισμός:  $D^Y$ : ο πίνακας που διατηρεί ο κόμβος Y

Προορισμός (κόμβος X)

σε κύκλο: η τελική επιλογή του κόμβου Y για να φτάσει στον προορισμό του

Κόστος του μονοπατιού (6) για τον Y για να φτάσει τον X μέσω του Z

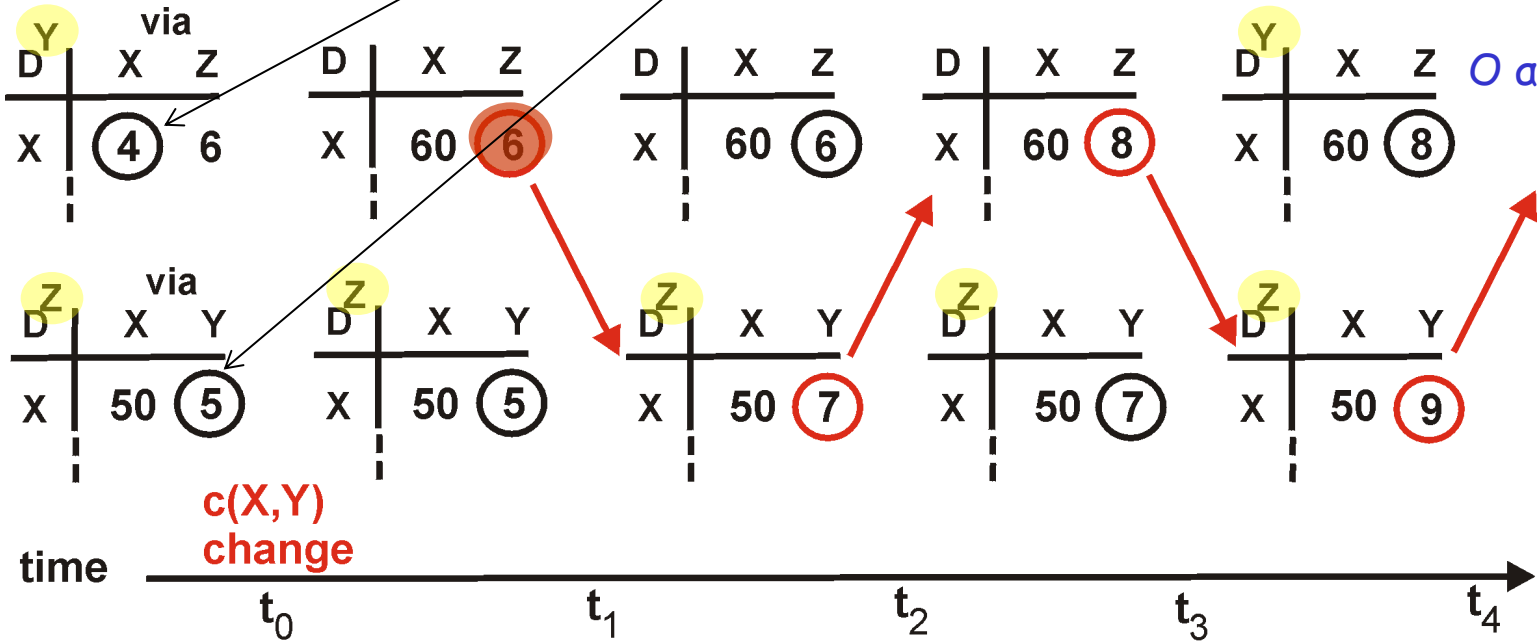
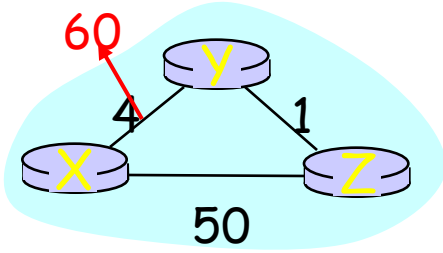


# Πίνακας Διανυσμάτων Απόστασης: αλλαγές στα κόστη των ζεύξεων

Για τον κάθε κόμβο, σημειώνουμε με «κύκλο» το κόστος για το πιο γρήγορο μονοπάτι (για τον κάθε ένα προορισμό)

## Αλλαγές στα κόστη των ζεύξεων:

- Τα καλά νέα ταξιδεύουν γρήγορα
- Τα κακά νέα ταξιδεύουν αργά - Πρόβλημα «μετρήματος ως το άπειρο!»

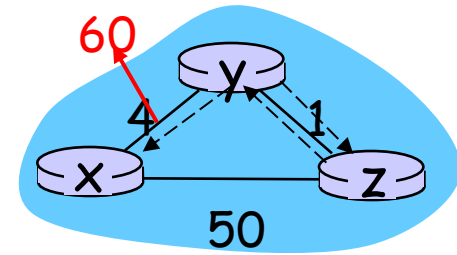


Ο αλγόριθμος συνεχίζει!

## Πίνακας Διανυσμάτων Απόστασης: αλλαγές στα κόστη των ζεύξεων

### Αλλαγές στα κόστη των ζεύξεων:

- Τα καλά νέα ταξιδεύουν γρήγορα
  - Τα κακά νέα ταξιδεύουν αργά -  
Πρόβλημα «μετρήματος ως το άπειρο!»
- ☹ 44 επαναλήψεις προτού ο αλγόριθμος σταθεροποιηθεί



# Σύγκριση των LS & DV αλγορίθμων

## Πολυπλοκότητα μηνυμάτων

- LS: με  $n$  κόμβους,  $E$  ζεύξεις,  $O(nE)$  μηνύματα στέλνονται
- DV: ανταλλαγές μεταξύ των γειτόνων μόνο  
ο χρόνος σύγκλισης ποικίλει

## Ταχύτητα σύγκλισης

- LS: ένας  $O(n^2)$  αλγόριθμος απαιτεί  $O(nE)$  μηνύματα
  - Μπορεί να έχει διακυμάνσεις
- DV: ο χρόνο σύγκλισης ποικίλει
  - Μπορεί να υπάρχουν κύκλοι στη δρομολόγηση
  - Πρόβλημα μετρήματος ως το άπειρο

# LS εναντίον DV αλγορίθμων

**Σταθερότητα:** τι συμβαίνει εάν ένας δρομολογητής δεν λειτουργεί

## LS:

- Ο κόμβος μπορεί να διαφημίσει ένα λανθασμένο κόστος *ζεύξης*
- Κάθε κόμβος υπολογίζει μόνο τον δικό του πίνακα

## DV:

- Ένας DV κόμβος μπορεί να διαφημίσει *λανθασμένο κόστος μονοπατιού*
- Ο πίνακας κάθε κόμβου χρησιμοποιείται και από άλλους
  - ☞ Το λάθος διαδίδεται στο δίκτυο

## Ο στοχασμός της ημέρας

"...ninety-nine parts of all things that proceed from the intellect are plagiarisms, pure and simple; and the lesson ought to make us modest. But nothing can do that." - Mark Twain