ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Δίκτυα Υπολογιστών

Μαρία Παπαδοπούλη
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση, Όχι Παράγωγο Έργο 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/

- Ως **Μη Εμπορική** ορίζεται η χρήση:
  - που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
  - που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
  - που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

- Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

.

Σκοποί ενότητας

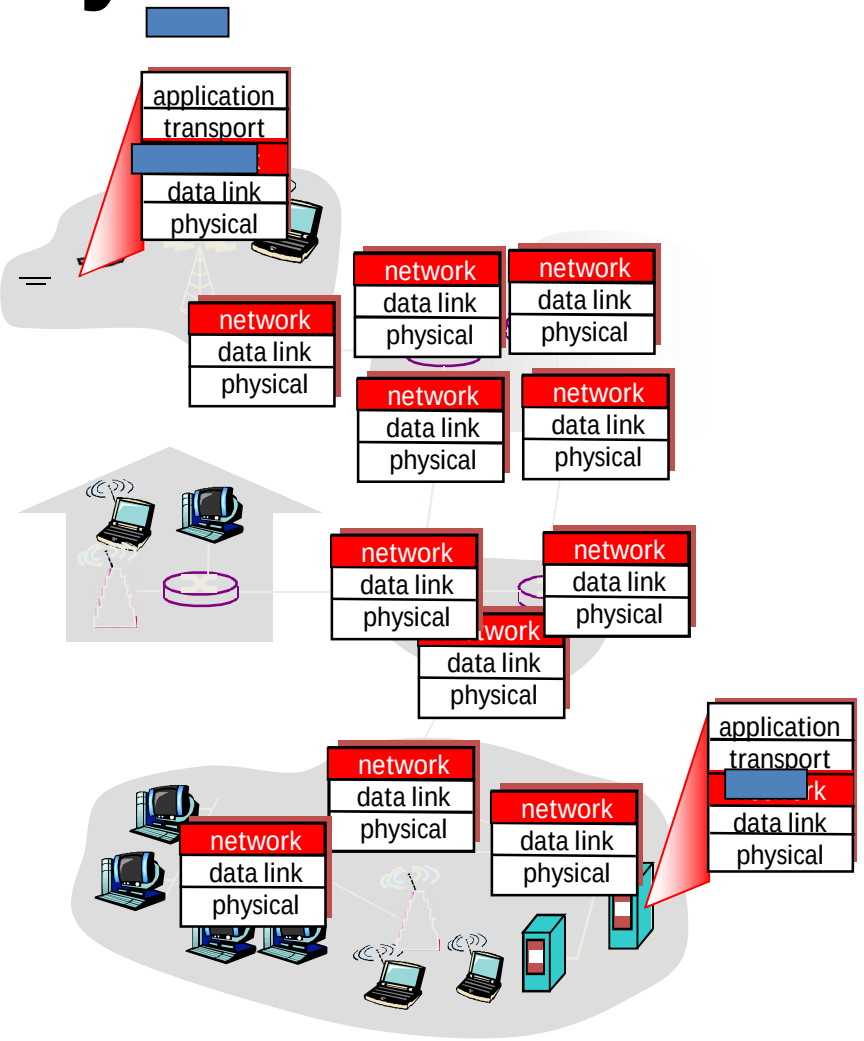
... Ερωτήσεις από τα προηγούμενα lectures ...

Αλγόριθμοι δρομολόγησης

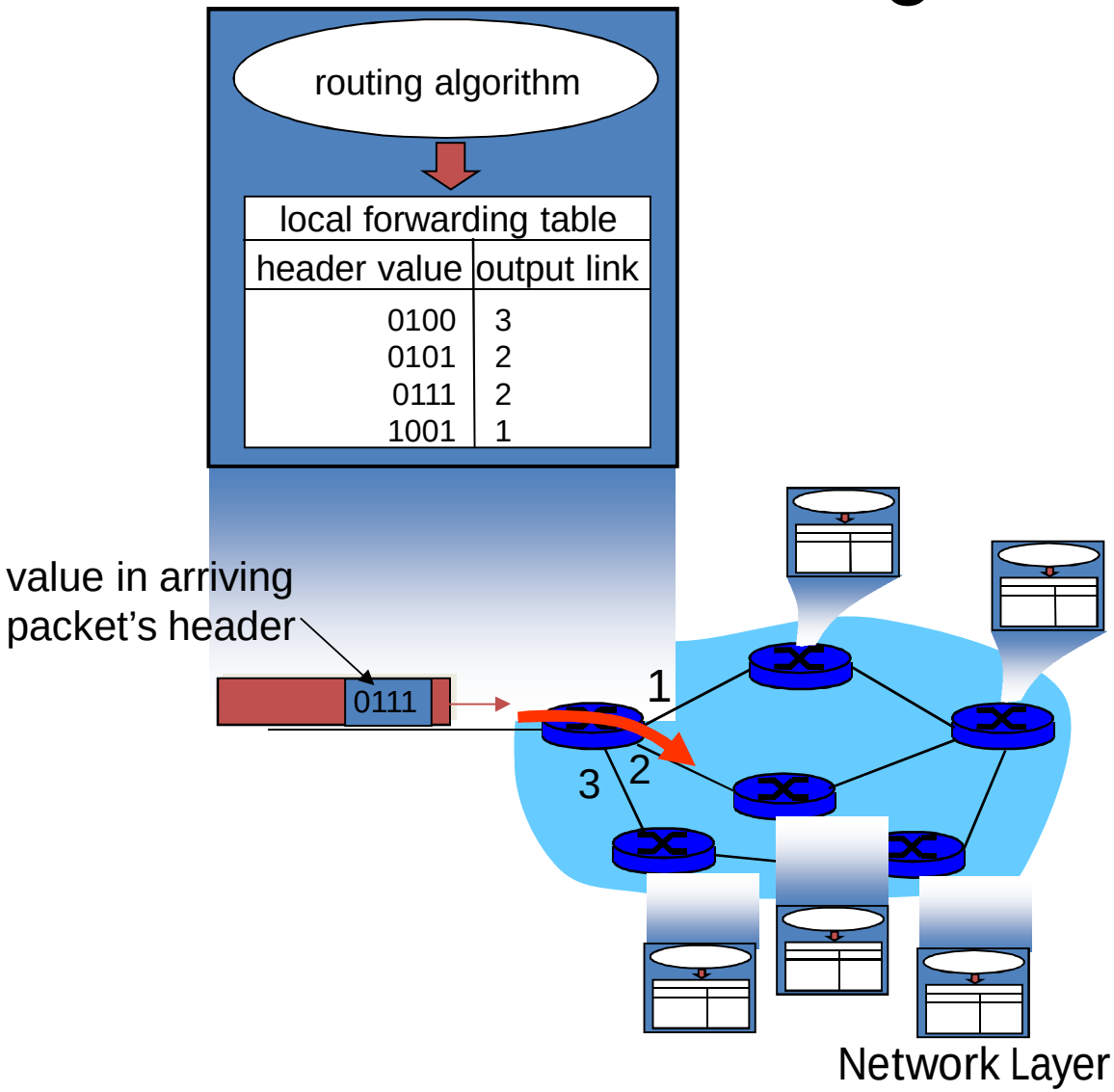
- Κατάστασης Ζεύξεων (Link state)
- Διανυσμάτων Απόστασης (Distance Vector)
- Ιεραρχικής Δρομολόγησης (Hierarchical routing)

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

Destination Address Range	Link Interface
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

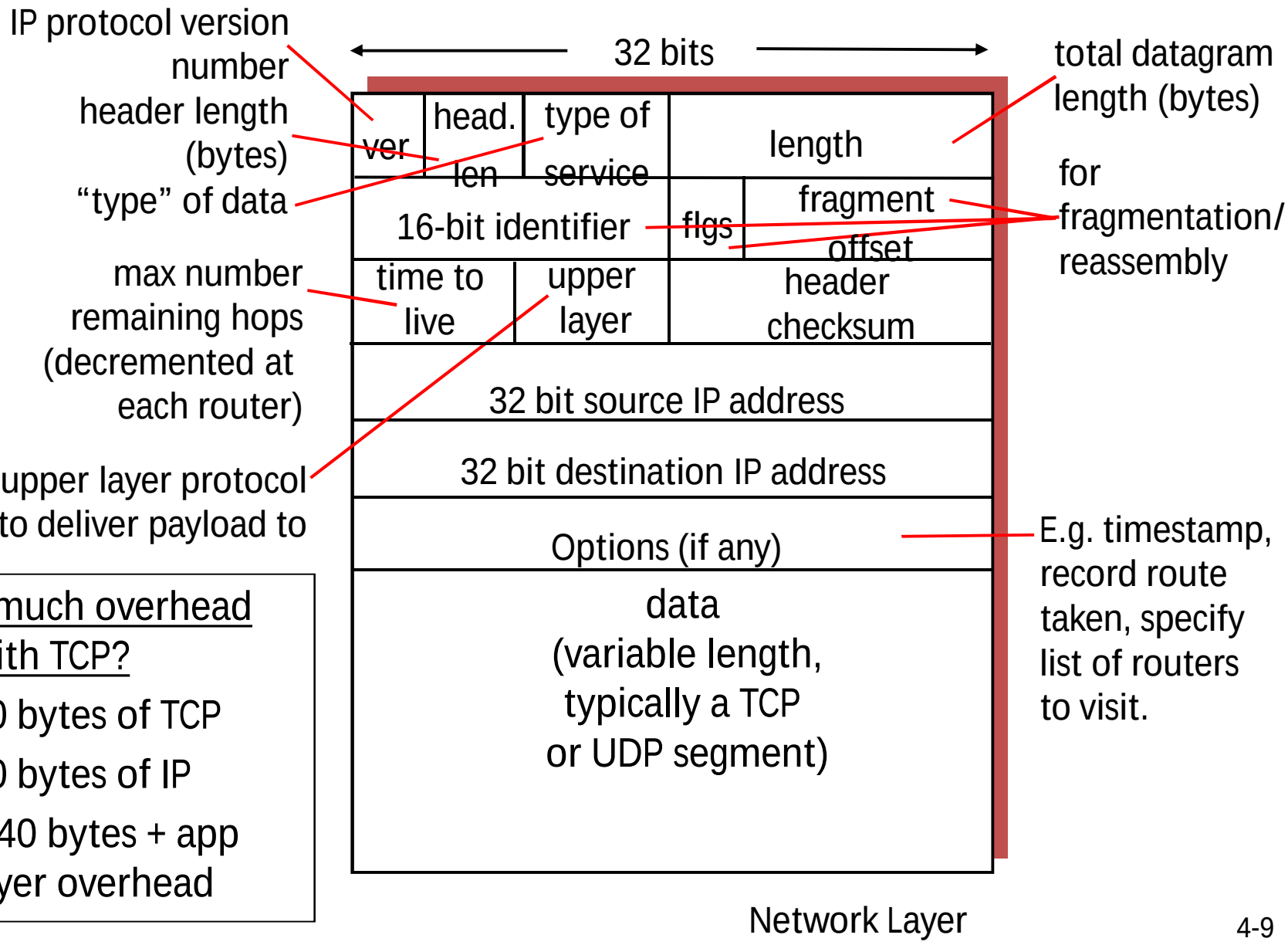
Prefix Match	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

Examples

DA: 11001000 00010111 00010110 10100001 Which interface?

DA: 11001000 00010111 00011000 10101010 Which interface?

IP datagram format



how much overhead with TCP?

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

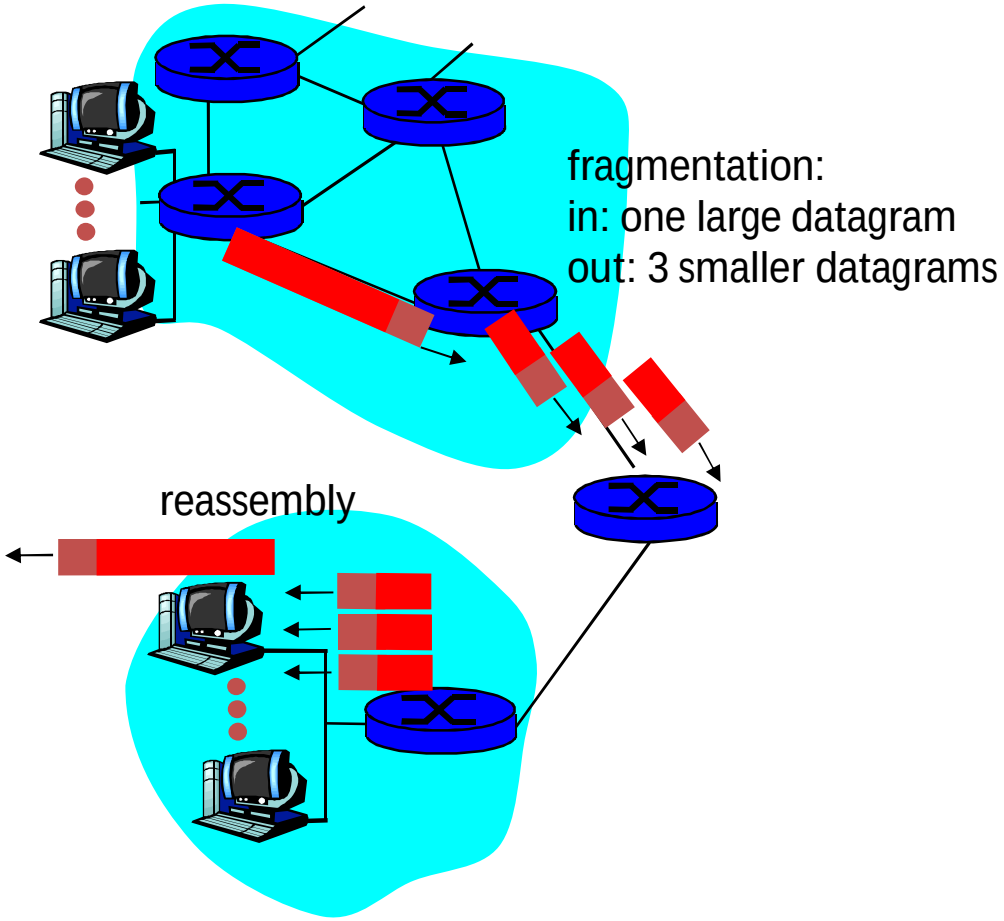
IP Fragmentation & Reassembly

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

one datagram becomes several datagrams

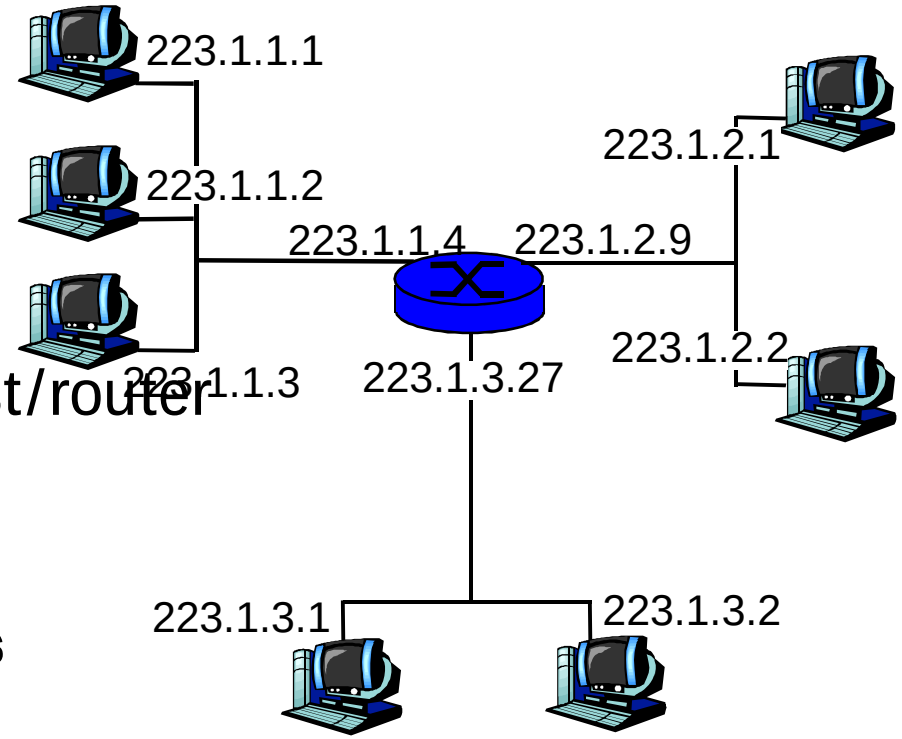
"reassembled" only at **final destination**

- 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 = 11011111 00000001 00000001 00000001

223 1 1 1

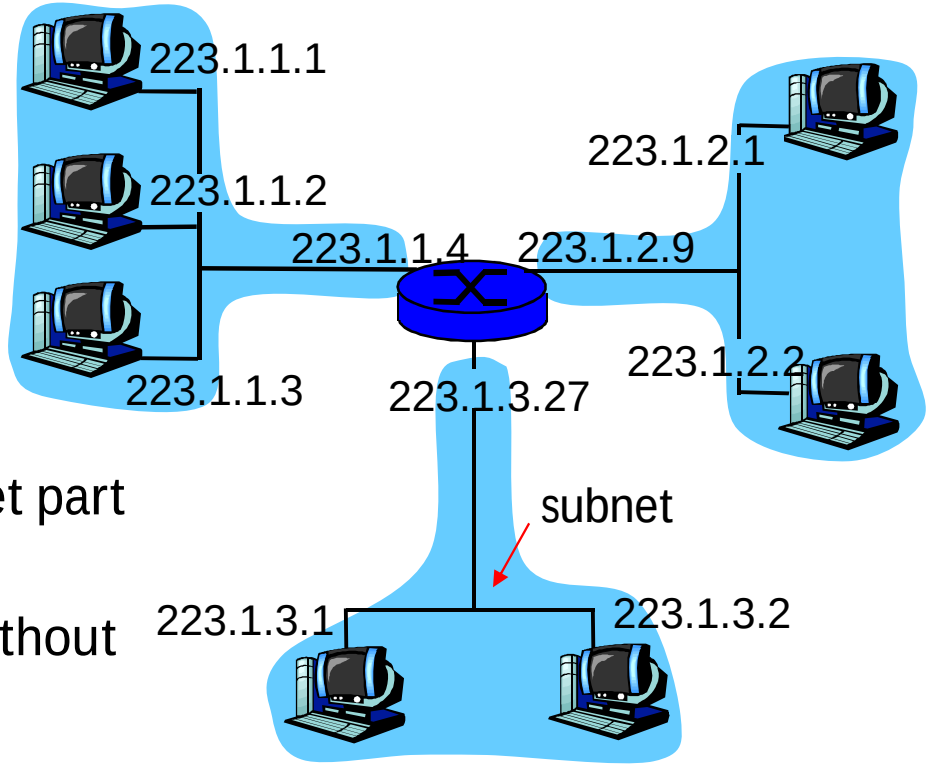
Subnets

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

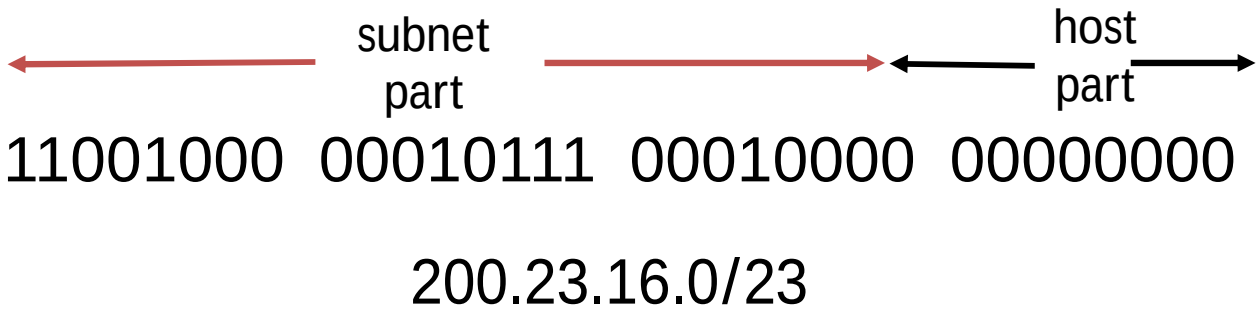


network consisting of 3 subnets

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	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	.....	....	....
Organization 7	11001000 00010111 00011110	00000000	200.23.30.0/23

IP addressing: the last word...

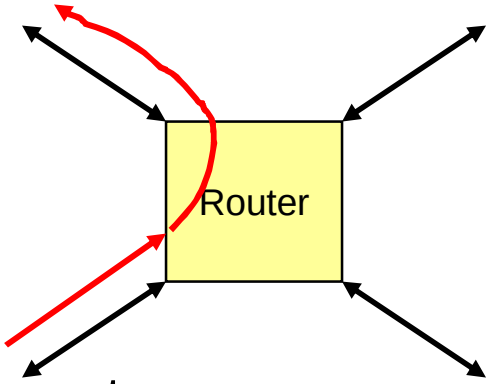
Q: How does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned

Names and Numbers

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

Περίληψη



Η ιστορία μας μέχρι τώρα …

- Οι IP διευθύνσεις αντανακλούν τη δομή του Internet

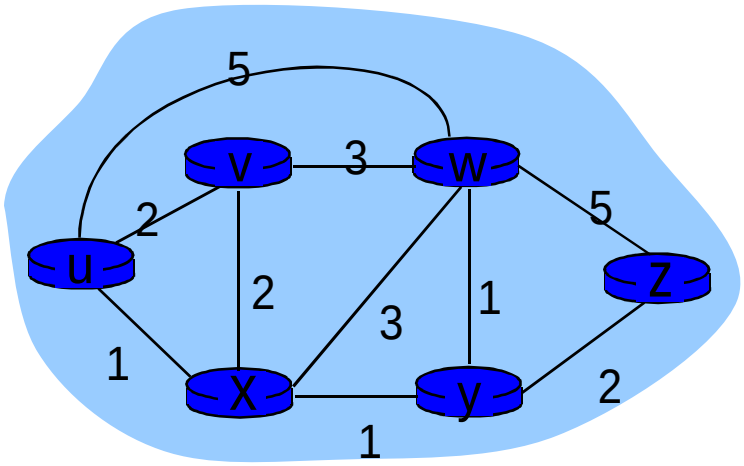
  όπως οι τηλεφωνικοί αριθμοί

- 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)

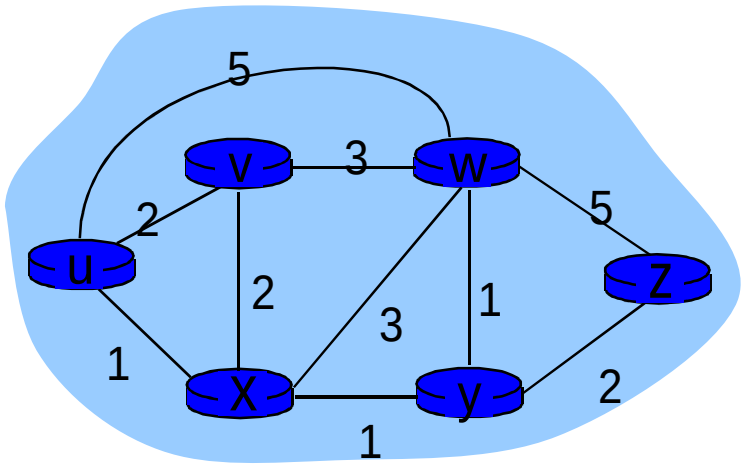


Γράφος: *G = (N,E)*

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 mporei na είναι:
Se oles tis akmes 1, ή
αντιστρόφως ανάλογο του εύρους ζώνης, ή
ανάλογο της συμφόρησης

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

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

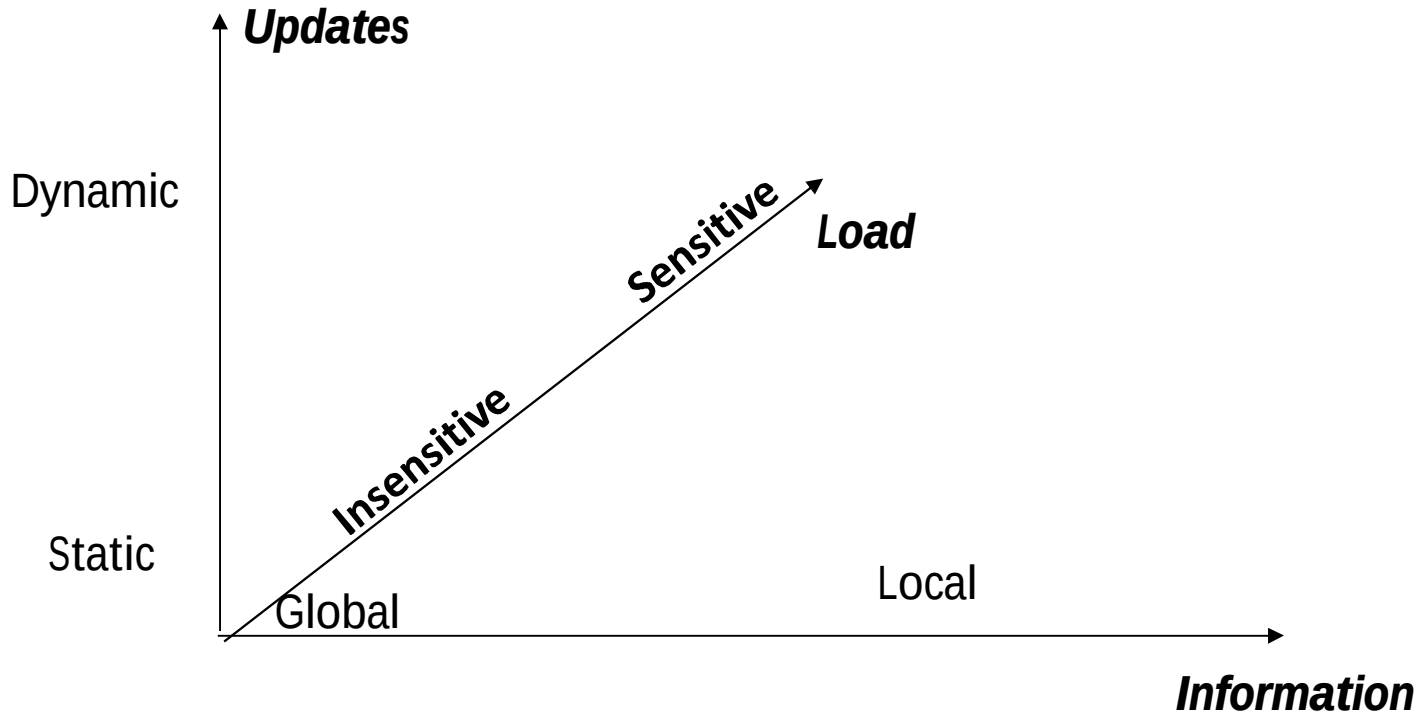
- Δυναμικοί αλγόριθμοι: αλλάζουν τα μονοπάτια δρομολόγησης καθώς ο φόρτος κίνησης του δικτύου ή η τοπολογία του αλλάζουν
- Μπορεί να τρέχουν
  - ✓ περιοδικά ή
  - ✓ σε άμεση απάντηση αλλαγών στην τοπολογία και στα κόστη των ζεύξεων

☹ 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***

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

Κεντρικοποιημένοι

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

Και οι 2 χρησιμοποιούνται στις μέρες μας

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

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

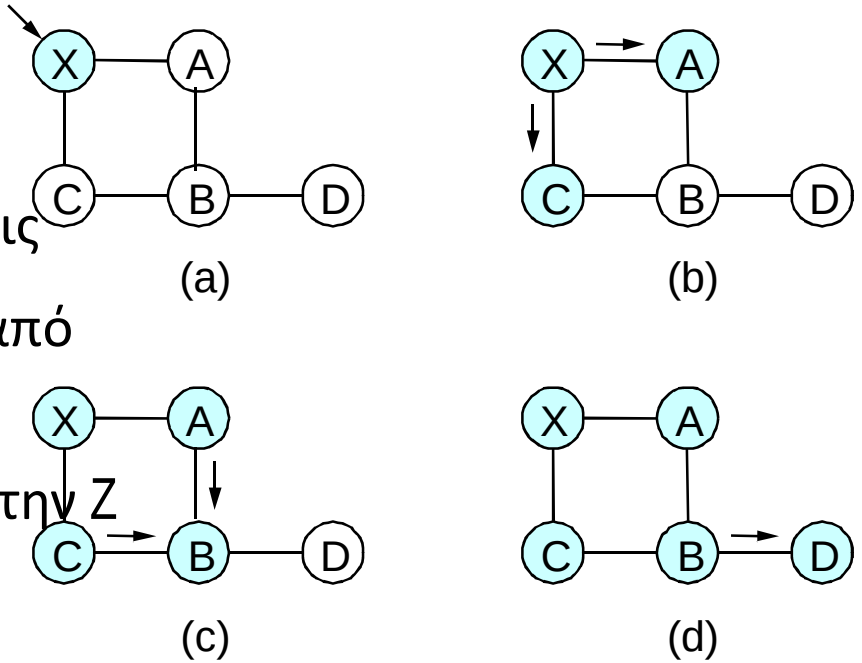
Αποστολή καταστάσεων ζεύξεων “πλημμυρίζοντας” το δίκτυο

Ο Χ θέλει να στείλει πληροφορία

Στέλνει σε όλες τις εξερχόμενες ζεύξεις

Όταν ο κόμβος Υ 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
- "↦" : μονοπάτι, "→": ζεύξη
- c(x,y): κόστος ζεύξης x →y;
  = ∞ αν **δεν** είναι άμεσοι γείτονες
- **D(v):** ***τωρινό*** κόστος μονοπατιού **s** ↦ v
- p(v): προηγούμενος κόμβος στο
- μονοπάτι s↦v
- **N'**: {κόμβοι των οποίων το μονοπάτι ελαχίστου κόστους είναι γνωστό}

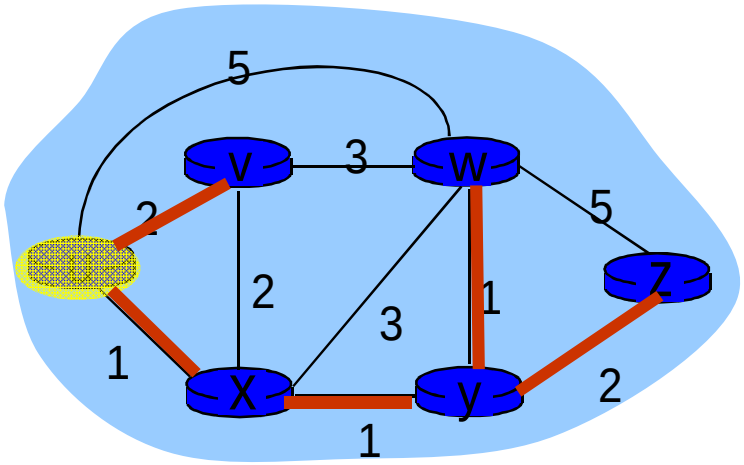
Αλγόριθμος του Dijsktra

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) = ∞**
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 Dijsktra: 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	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Η διαδικασία εκτελείται από τον **κάθε δρομολογητή στο δίκτυο**

Use of global information about the network topology & costs

Συζήτηση για τον αλγόριθμου του Dijkstra

Αλγοριθμική πολυπλοκότητα: n κόμβοι

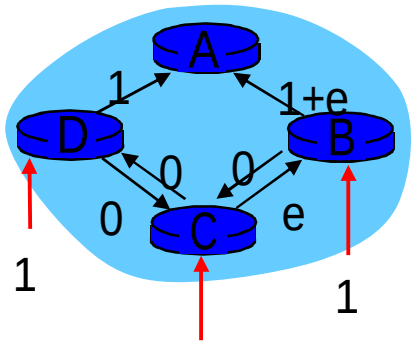
- Σε κάθε επανάληψη: χρειάζεται να ελέγξει όλους τους κόμβους w, που δεν ανήκουν στο σύνολο N
- n(n+1)/2 συγκρίσεις ⇨ $O(n^2)$

☞ Πιο αποδοτικές υλοποιήσεις είναι πιθανές: O(nlogn)

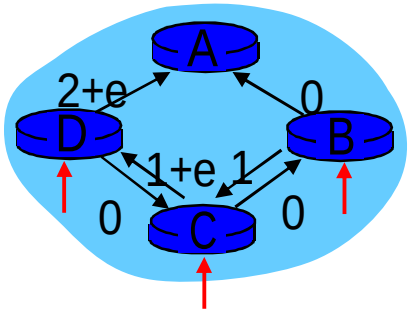
Πιθανές παραλλαγές:

π.χ., κόστος ζεύξης = ποσότητα μεταφερόμενης κίνησης

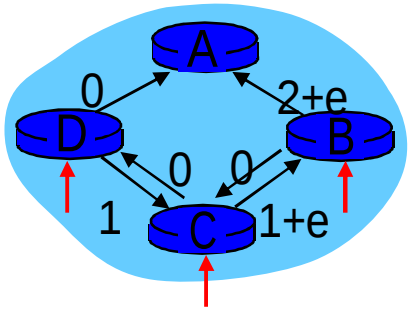
Oscillation



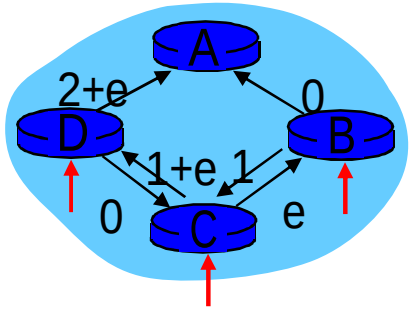
e
αρχικά



...υπολογίζεται ξανά
η δρομολόγηση



... υπολογίζεται ξανά



...υπολογίζεται ξανά

- 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:

$$\mathbf{D_x(y) \leftarrow \min_v \{ c(x,v) + D_v(y) \} \quad for\ each\ node\ y \in N}$$

$D_x(y)$ = εκτίμηση **ελαχίστου κόστους** από x ⇥ι y

Ο κόμβος x διατηρεί το $\mathbf{D_x} = [D_x(y) : y \in N]$

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

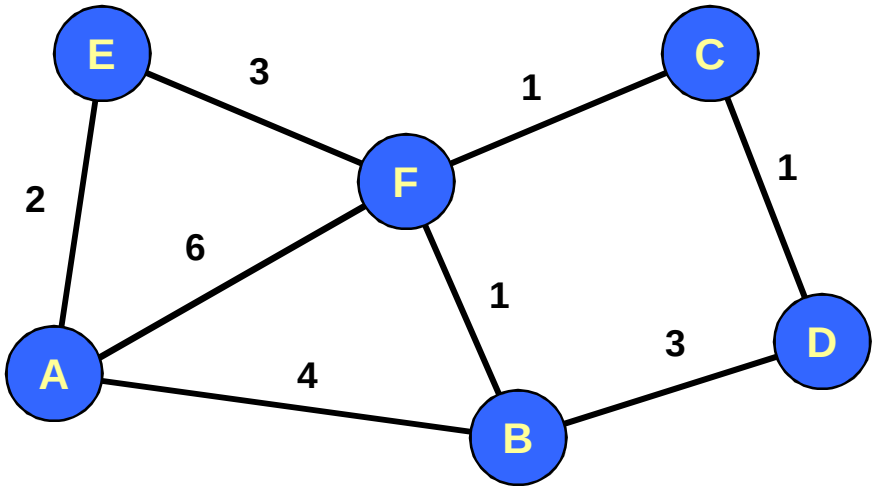
Τότε

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

Οπου ***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	∞	–
D	∞	–
E	2	E
F	6	F

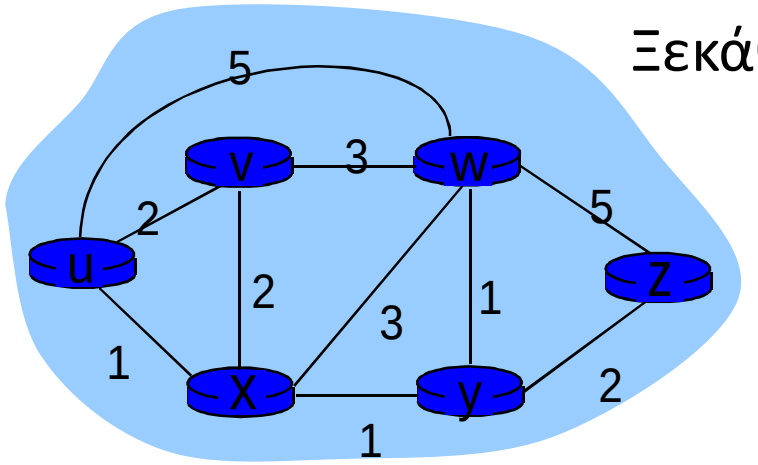


Ιδέα

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

Αρχικά: Υπάρχουν μόνο εγγραφές για τους άμεσα συνδεδεμένους γείτονες

Bellman-Ford example



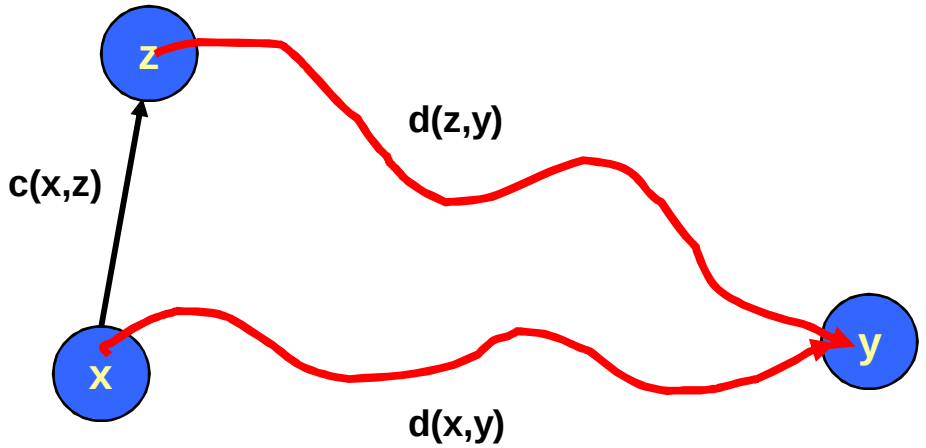
Ξεκάθαρα, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

Η B-F εξίσωση λέει:

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

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

Ενημέρωση του πίνακα διανυσμάτων απόστασης



Update(x,y,z)

d ← c(x,z) + d(z,y) # Κόστος μονοπατιού από τον x στον y με 1ο 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 ⇥ y
- Διάνυσμα απόστασης: $\mathbf{D}_x = [D_x(y): y \in N]$
- Ο κόμβος x γνωρίζει το κόστος x→y : c(x,v)

☞ Ο κόμβος x διατηρεί

- $\mathbf{D}_x = [D_x(y) : y \in N]$

και επίσης τους πίνακες διανυσμάτων απόστασης (DV) των γειτόνων του

- Για κάθε γείτονα v: $\mathbf{D}_v = [D_v(y): y \in N]$

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:

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

υπολογίζει ***ξανά*** εκτιμήσεις

Εάν ο πίνακας διανυσμάτων απόστασης (DV) προς οποινδήποτε προορισμό έχει αλλάξει, *ειδοποιεί* 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

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

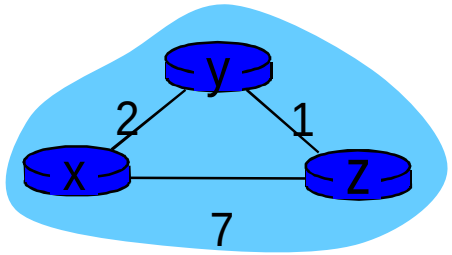
from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

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

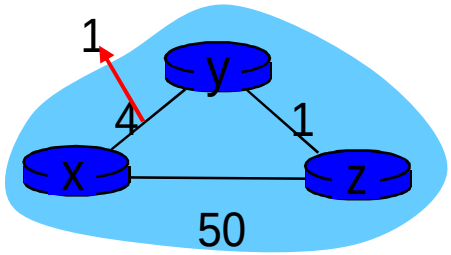


time

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



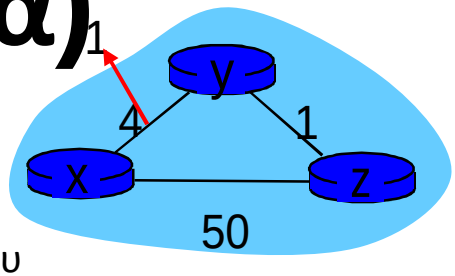
"good news travels fast"

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

Τη στγμη 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.

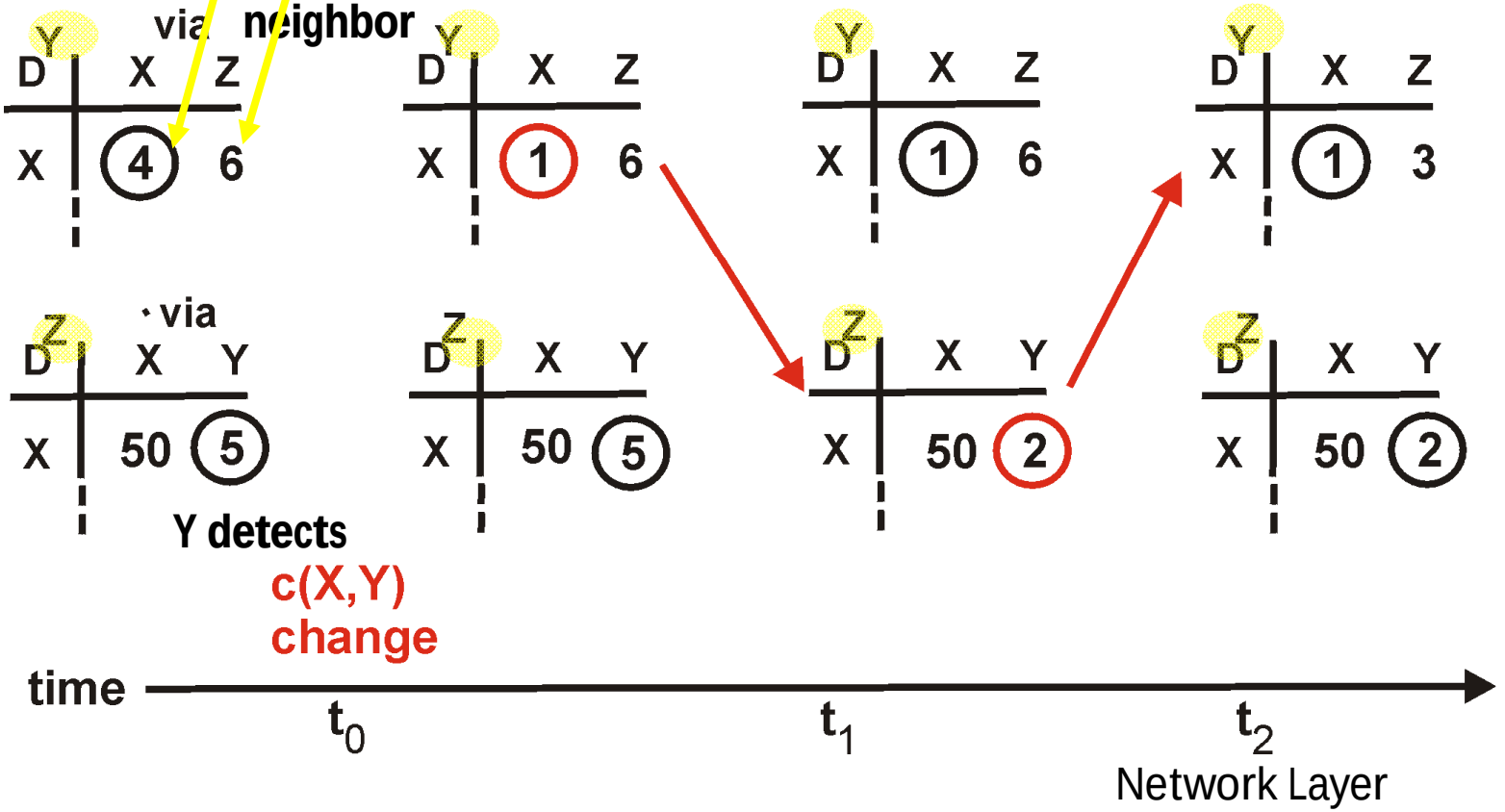
Τη στιγμή t_2, ο *y* λαμβάνει την ενημέρωση του*z και ενημερώνει τον πίνακα απόστασης του*.
Τα ελάχιστα κόστη του y δεν αλλάζουν και για αυτό ο *y* δεν στέλνει κανένα μήνυμα στον *z*.

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

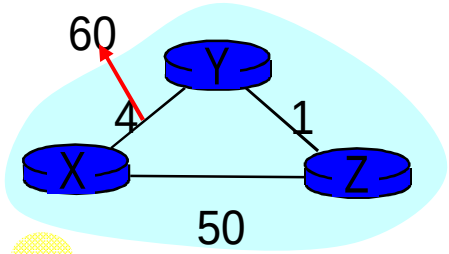


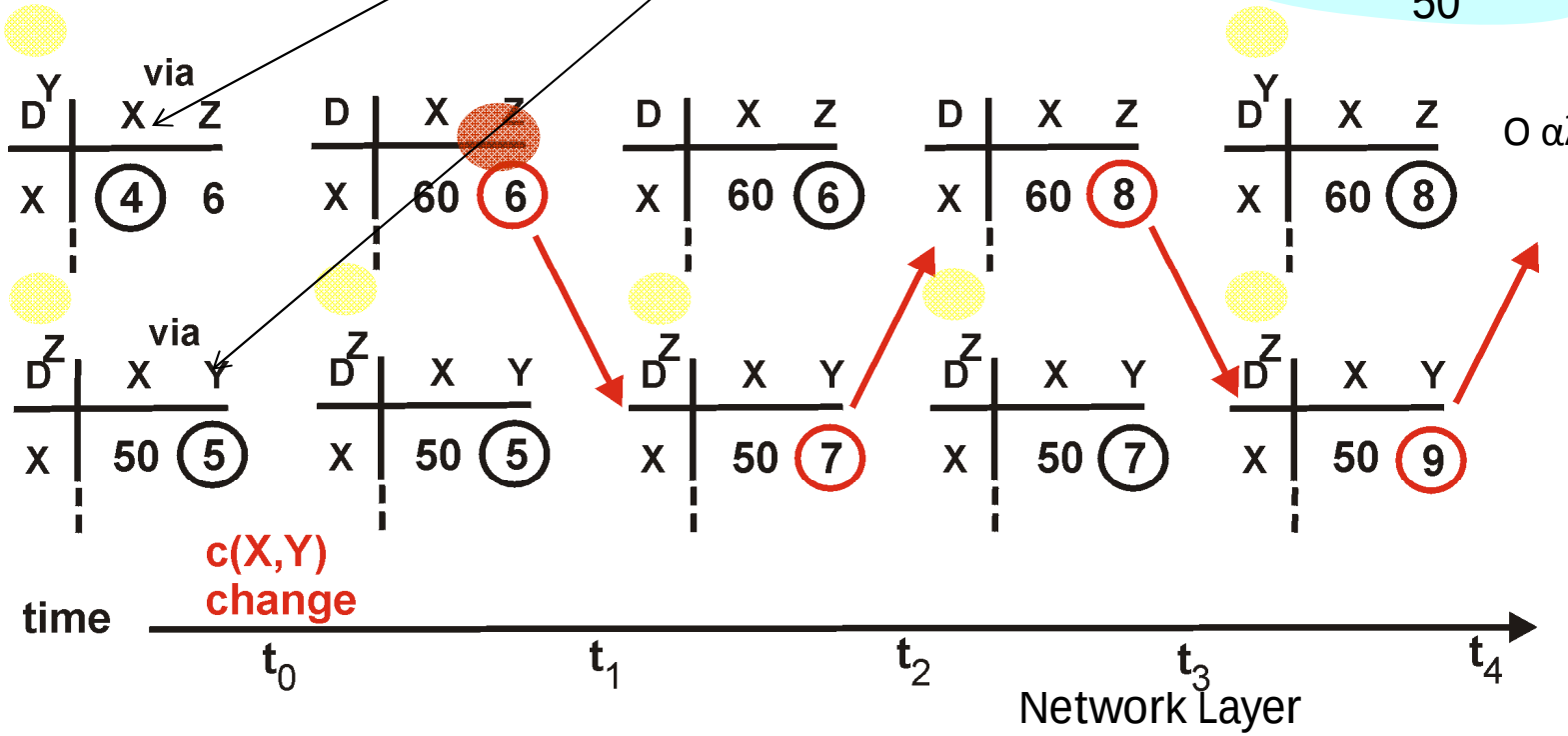
Πίνακας Διανυσμάτων Απόστασης: αλλαγές στα κόστη των ζεύξεων

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!

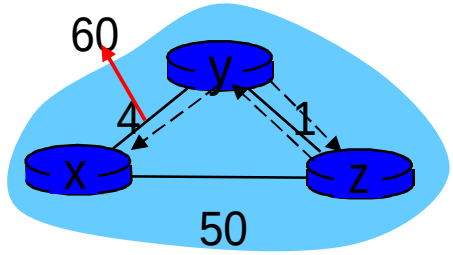
Πίνακας Διανυσμάτων Απόστασης: αλλαγές στα κόστη των ζεύξεων

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 ν κόμβους, Ε 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**

Τέλος Ενότητας