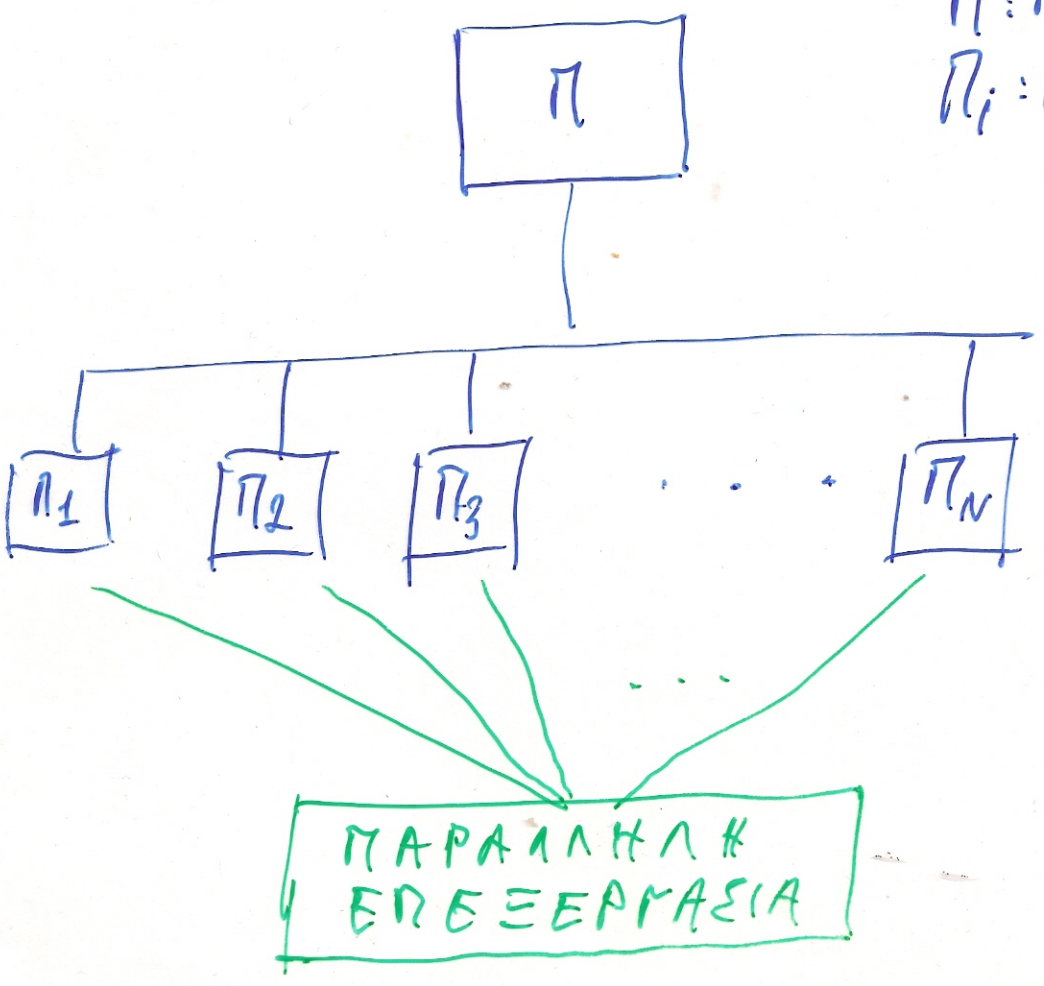
1. ΕΙΣΑΓΩΓΗ

Η ανάγκη για παράλληλους υπολογιστές

- Οι εφαρμογές απαιτούν χρονοβόρους υπολογισμούς
  - σχεδιασμός αεροσκαφών
  - αριθμητική προσομοίωση σε προβλήματα Υπολογιστικής Δυναμικής των Ρευστών
- Η αύξηση των υπολογισμών θα χορεύτει.

Λύση: χρήση της παραλληλίας

Π: πρόβλημα
Π_i: υποπρόβλημα



Παράλληλος Αλγόριθμος

Μία μέθοδος λύσης για ένα δεδομένο πρόβλημα που πρόκειται να υλοποιηθεί σε ένα παράλληλο υπολογιστή

Παράλληλος Υπολογιστής

Ένας υπολογιστής με πολλές μονάδες επεξεργασίας ή επεξεργαστές

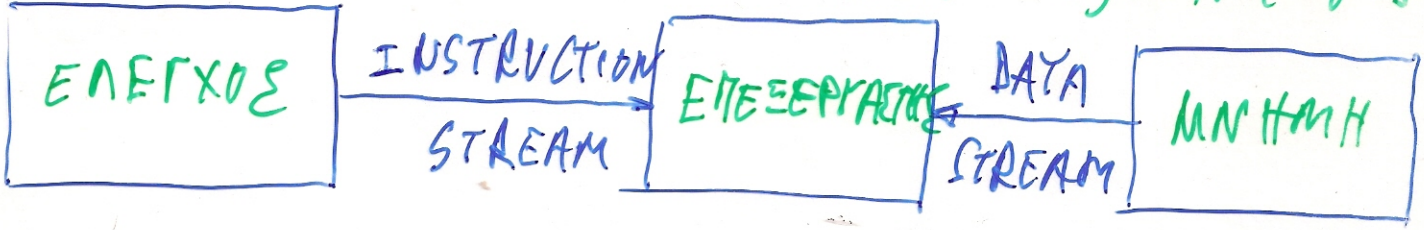
Μοντέλα Υπολογισμού

- SISD
- MISD
- SIMD
- MIMD

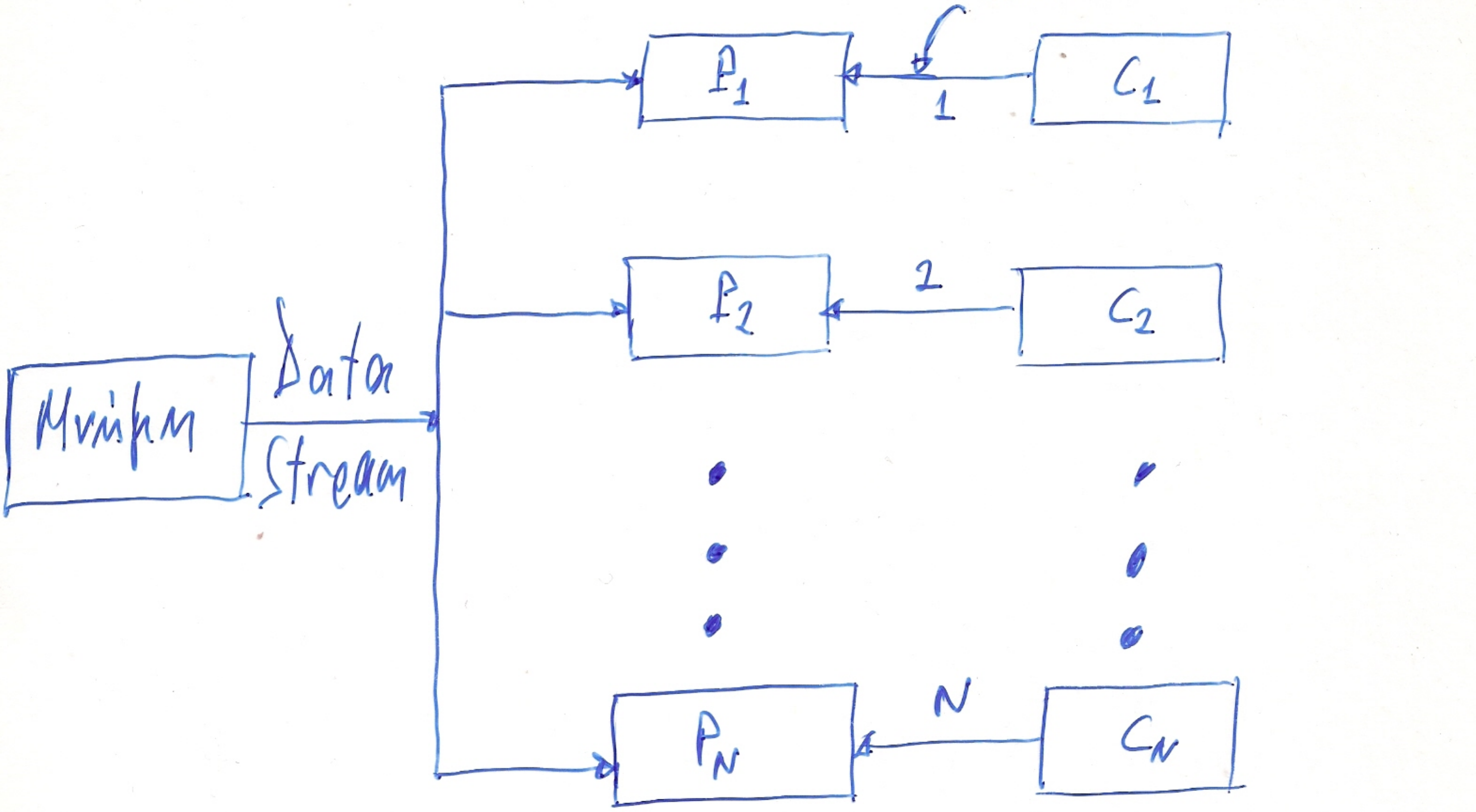
S: single
I: instruction
D: data
M: multiple

SISD

- John von Neumann 1940
- Ακολουθιακός (σειριακός) Αλγόριθμος



MISD



Παράδειγμα

Έλεγχος αν ένας αριθμός z είναι πρώτος

Παράλληλος αλγόριθμος

- Έστω ότι υπάρχουν τόσοι επεξεργαστές όσοι είναι και οι υποψήφιοι διαιρέτες του z.
- Έλεγχος από κάθε επεξεργαστή αν ο υποψήφιος διαιρέτης διαιρεί τον z.

Παράδειγμα

Εύρεση του συνόλου (κατηγορίας) στην οποία ανήκει ένα αντικείμενο

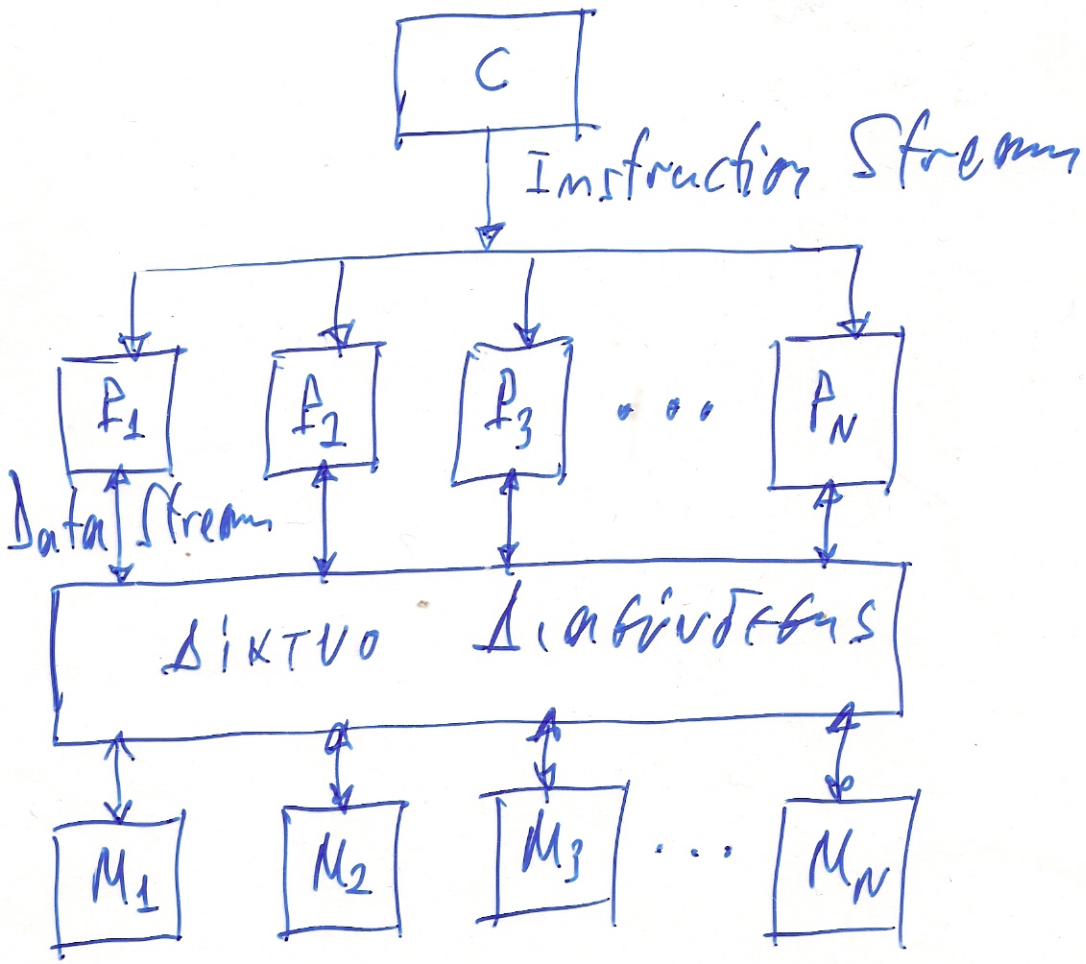
Παράδειγμα Αλγόριθμος

- Υποθέτουμε ότι υπάρχουν τόσοι επεξεργαστές όσες είναι οι κατηγορίες.
- Στέλνεται το αντικείμενο σε κάθε επεξεργαστή, ο οποίος εξέχει αν ανήκει στην κατηγορία του.

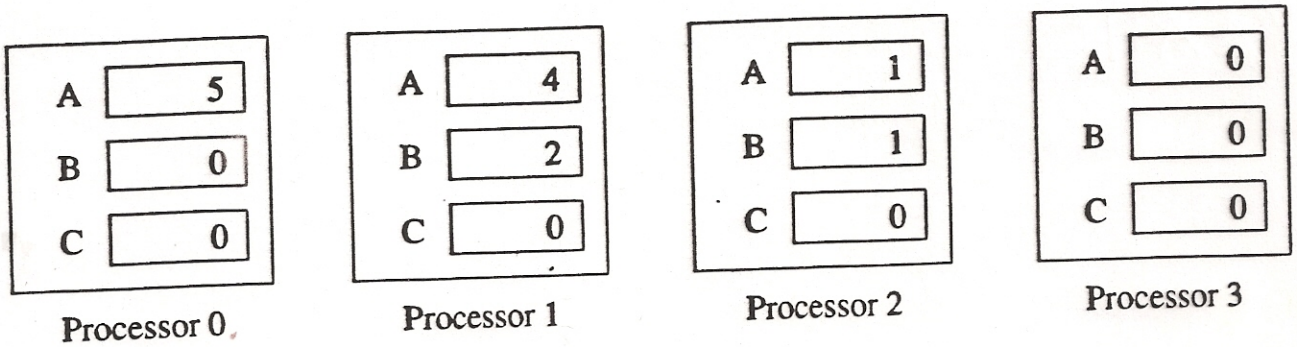
Μειονεκτήματα

- Ειδικής φοράς υπολογισμοί

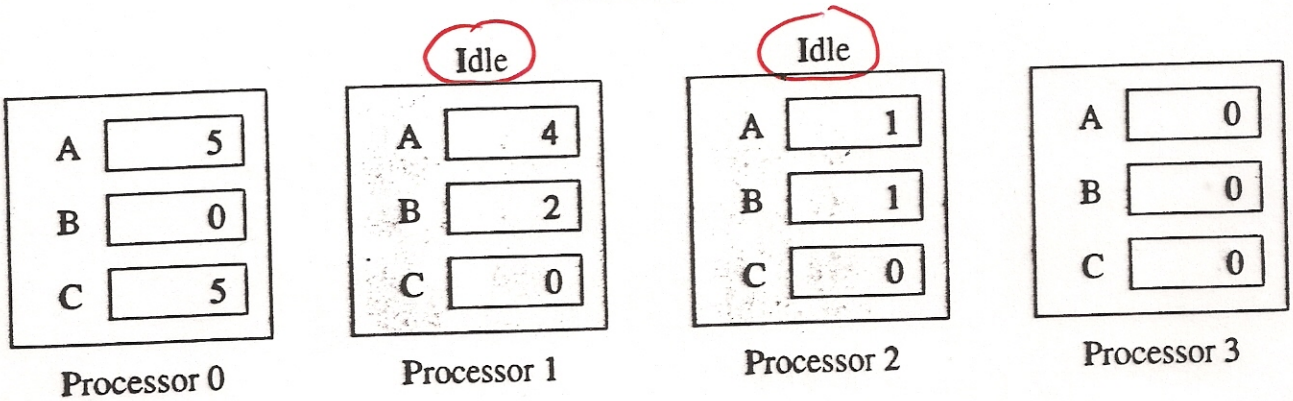
SIMD



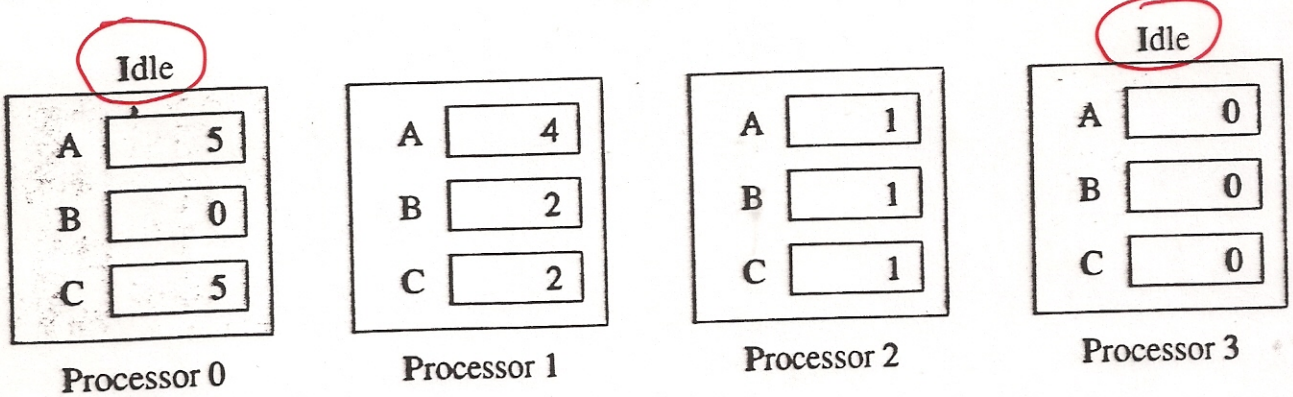
```
if (B == 0)
        C = A;
else
        C = A/B;
```



Initial values



Step 1



Step 2

Κοινή Μνήμη

(a) : UMA

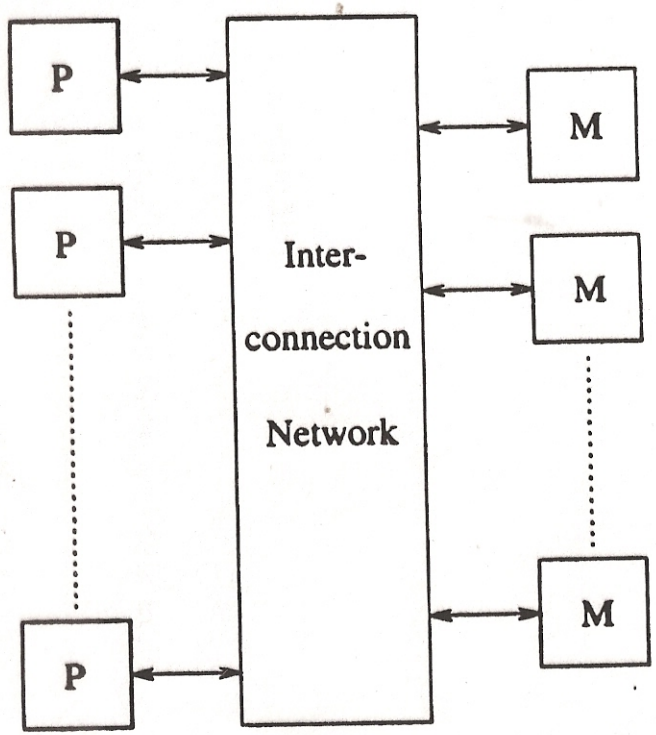
(b) : NUMA με τοπική και ολική μνήμη

(c) : NUMA με τοπική μνήμη μόνο

ομοιόμορφος χρόνος επεξεργασίας επεξεργαστή - μνήμης

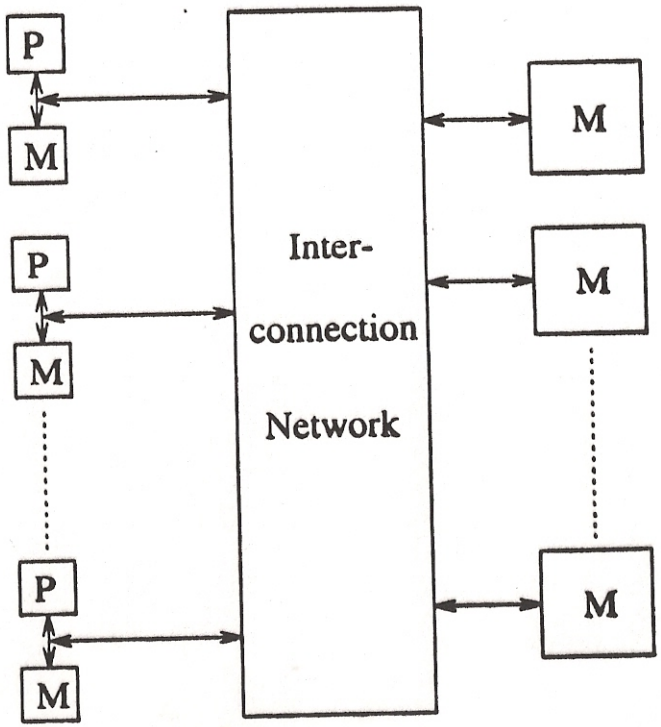
κάθε επεξεργαστής έχει τη δική του τοπική μνήμη και διατηρεί και την ολική διαμοιραζόμενη μνήμη

Δεν υπάρχει καθολική μνήμη γιατί έγινε τοπική σε κάθε επεξεργαστή. Ο κάθε επεξεργαστής επικοινωνεί με την τοπική του μνήμη και αν δεν βρει αυτό που θέλει μπορεί να έχει μέσω δικτύου προσπέλαση σε οποιαδήποτε άλλη μνήμη.



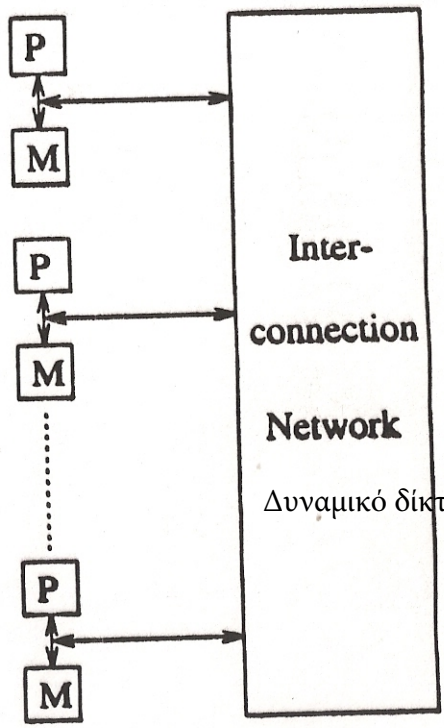
(a)

1. Όλοι οι επεξεργαστές βλέπουν κοινά την κοινή μνήμη
2. Ίδιος χρόνος προσπέλασης επεξ/μνήμη



(b)

Ο κάθε επεξεργαστής έχει τοπική μνήμη
Στόχος: Αύξηση/ελάττωση του χρόνου προσπέλασης στην τοπική μνήμη.



Δυναμικό δίκτυο

(c)

Η μνήμη είναι κατανεμημένη στους επεξεργαστές. Πρώτα ο κάθε επεξ. ψάχνει τη δική του μνήμη και αν δεν βρει ψάχνει μέσω δικτύου τις άλλες

Διαμοιραζόμενης Μνήμης Υπολογιστές

- C.mmp
- NYU Ultracomputer

Μειονεκτήματα

- μεγάλος τύπος δικτύου διασύνδεσης

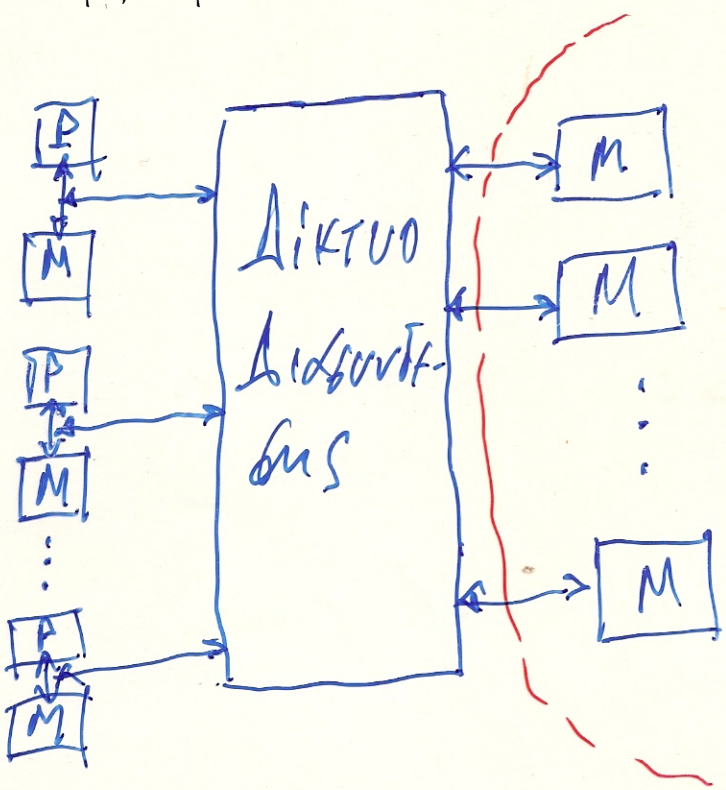
γιατί σε κάθε κύκλο εντολής ο κάθε επεξεργαστής ίσως χρειάζεται να προσπελάσει μια λέξη από τη διαμοιραζόμενη μνήμη μέσω του δικτύου διασύνδεσης, προκειμένου να έχουν καλή αποδοτικότητα

- κρυφή προσπέλαση μνήμης

καθώς μια αίτηση για ανάγνωση ή εγγραφή πρέπει να περάσει μέσα από πολλά stages στο δίκτυο

Θεραπεία ⟶ τοπική μνήμη

Η τοπική μνήμη αποθηκεύει το πρόγραμμα που εκτελείτε στον επεξεργαστή και όχι στις μη διαμοιραζόμενες δομές δεδομένων



Διαμοιραζόμενης Μνήμης

UMA

NUMA
TC-2000
KSR-1

UMA: Αν ο χρόνος που χρειάζεται ένας επεξεργαστής για να προσπελάσει μια λέξη μνήμης στο σύστημα είναι ο ίδιος (μοναδιαίος)

NUMA: Αν ο χρόνος προσπέλασης ενός μακρινού memory bank είναι μεγαλύτερος από το χρόνο ενός τοπικού.

Γενικά στην παραλληλία θέλουμε να μην υπάρχει αδράνεια.

- Κάθε επεξεργαστής έχει τη δική του τοπική μνήμη (πρόγραμμα και δεδομένα)
- Οι επεξεργαστές λειτουργούν σύγχρονα: σε κάθε βήμα, όλοι οι επεξεργαστές εκτελούν την ίδια εντολή, ο κάθε ένας σε διαφορετικά δεδομένα
- κατάλληλοι για παράλληλα-δεδομένα (data-parallel)

Επικοινωνία

distributed memory ή/και message-passing

διαμοιραζόμενη (κοινή) μνήμη (shared memory)

δίκτυο διασύνδεσης (interconnection network)

επόμενη διάλεξη

Διαμοιραζόμενης Μνήμης SIMD (PRAM)

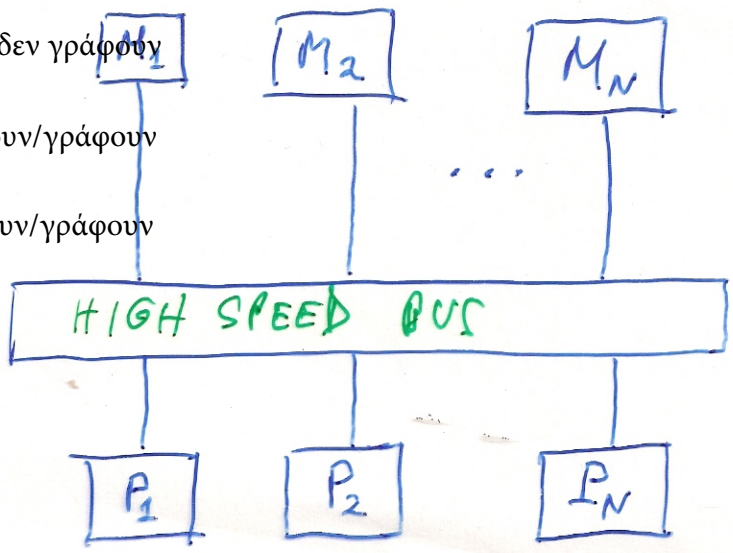
- EREW
- CREW
- ERCW
- CRCW

EREW: Δεν μπορούν ταυτόχρονα 2 επεξεργαστές να διαβάσουν/γράψουν στην ίδια τοποθεσία μνήμης

διαβάζουν/δεν γράφουν

δεν διαβάζουν/γράφουν

διαβάζουν/γράφουν



Παράδειγμα

Να προσδιοριστεί αν ένα δεδομένο στοιχείο x βρίσκεται σε ένα πεδίο με n θέσεις.

Ακολουθιακός αλγόριθμος απαιτεί n βήματα.

Παράλληλος αλγόριθμος

EREW SM SIMD με $N \leq n$ επεξεργαστές $P_1, P_2, \ldots, P_N$.

N=πλήθος επεξεργαστών κοινής μνήμης, n=πλήθος εγγραφών

- Μετάδοση του x στους επεξεργαστές

1. P_1 διαβάζει το x και το μεταδίδει στον P_2

αφού υπάρχει κοινή μνήμη η επικοινωνία γίνεται μέσω αυτής.

2. Οι P_1 και P_2 μεταδίδουν ταυτόχρονα το x στους P_3 και P_4, αντίστοιχα σε κ.ο.κ. ταυτόχρονα

3. Οι P_1, P_2, P_3 και P_4 μεταδίδουν το x στους P_5, P_6, P_7, P_8, αντίστοιχα, κ.ο.κ. Broadcasting

Η μετάδοση του x σε όλους τους επεξεργαστές απαιτεί $\log_2 N$ βήματα

- Χωρισμός του πεδίου σε υποσύνολα με n/N στοιχεία

n/N πλήθος των επεξεργαστών

P_i αναζητεί σε n/N στοιχεία άρα

θα γίνει η αναζήτηση αυτή ταυτόχρονα από όλους τους επεξεργαστές

n/N βήματα

Σύνολο: $\log N + n/N$ βήματα

Μόλις βρεθεί η εγγραφή οι υπόλοιποι δεν χρειάζεται να συνεχίσουν

Αν χρησιμοποιούμε μια τοπική μεταβλητή F για την δήλωση ότι βρέθηκε το x, τότε

Η F παίρνει τιμές 0,1 και χρειάζεται για να σταματήσουν οι υπόλοιποι επεξ. την αναζήτηση

$\log N + (n/N)\log N$

EREW χειρότερη πολυπλοκότητα

για σε κάθε βήμα της αναζήτησης αναζητούν η τιμή της F σε όλους τους επεξεργαστές (broadcasting $\log N$).

Για CREW SM SIMD

1 βήμα για μετάδοση του x

1 βήμα για μετάδοση της F (τέλος κάθε αναζήτησης).

Άρα μόνο n/N βήματα για αναζήτηση

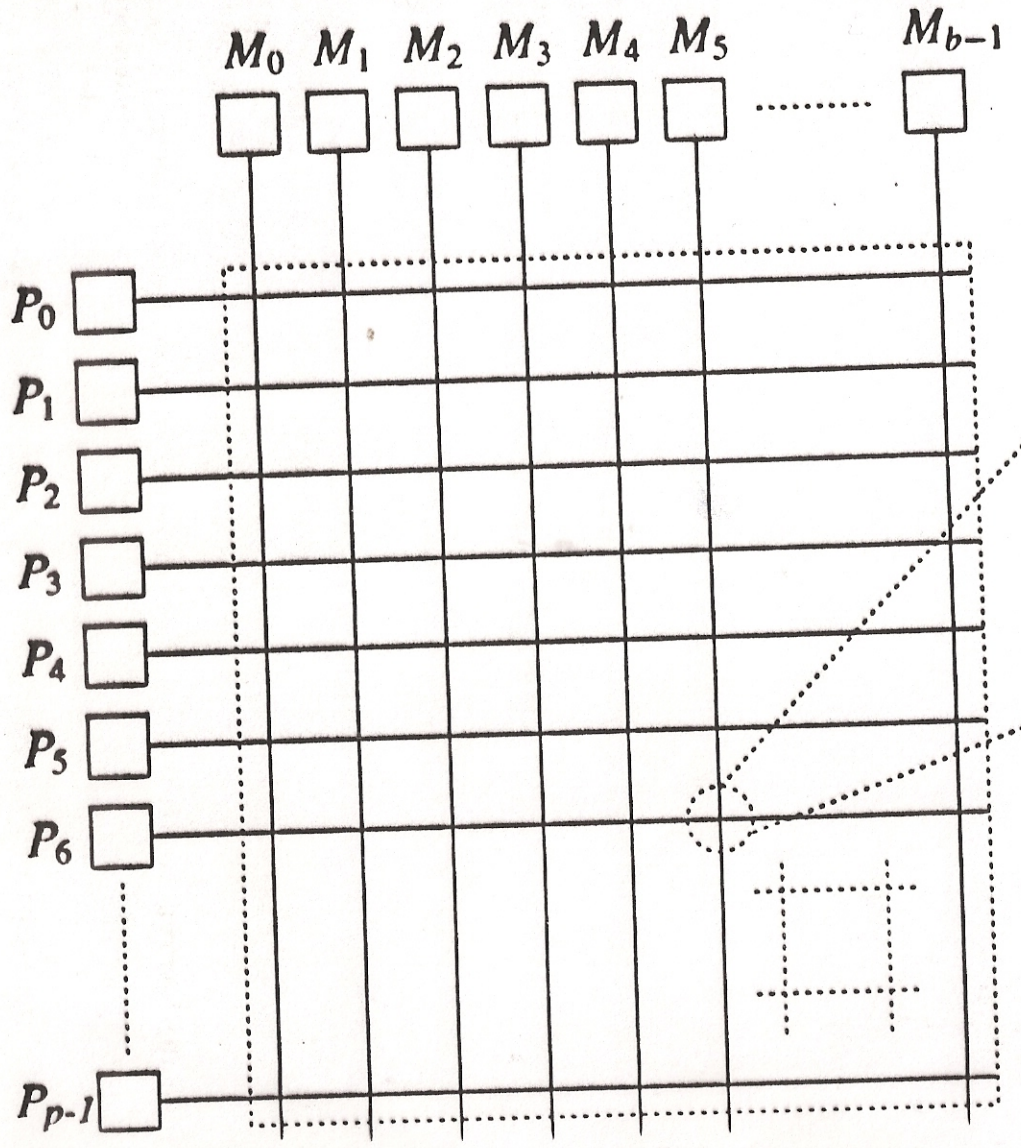
πλήθος εγγραφών

Crossbar Switch : Cray Y-MP, Fujitsu VPP 500

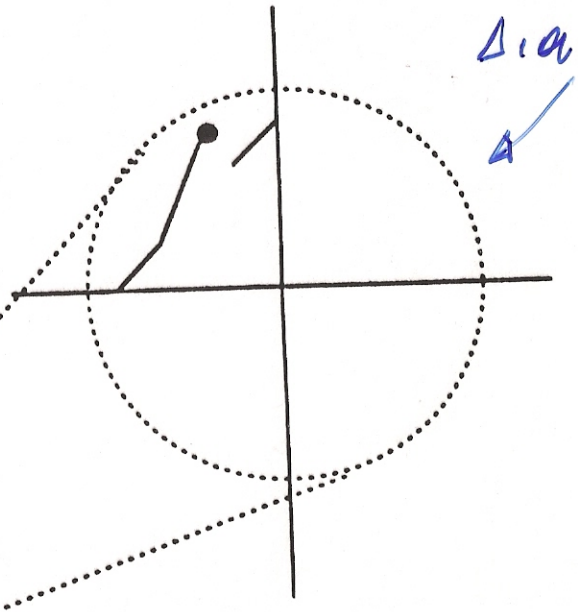
non-blocking δίκτυο γιατί η σύνδεση ενός επεξ. με ένα memory bank δεν εμποδίζει τη σύνδεση ενός άλλου επεξ. με ένα άλλο memory bank

Πρέπει να υπάρχει η δυνατότητα όλοι οι επεξ. να μπορούν να προσπελάσουν τουλάχιστον ένα memory bank. Γι' αυτό p<=b. Διαφορετικά, μπορεί κάποια χρονική στιγμή να υπάρξουν επεξ. που δεν θα μπορούν να προσπελάσουν κάποιο memory bank.

b memory banks



p επεξεργαστές



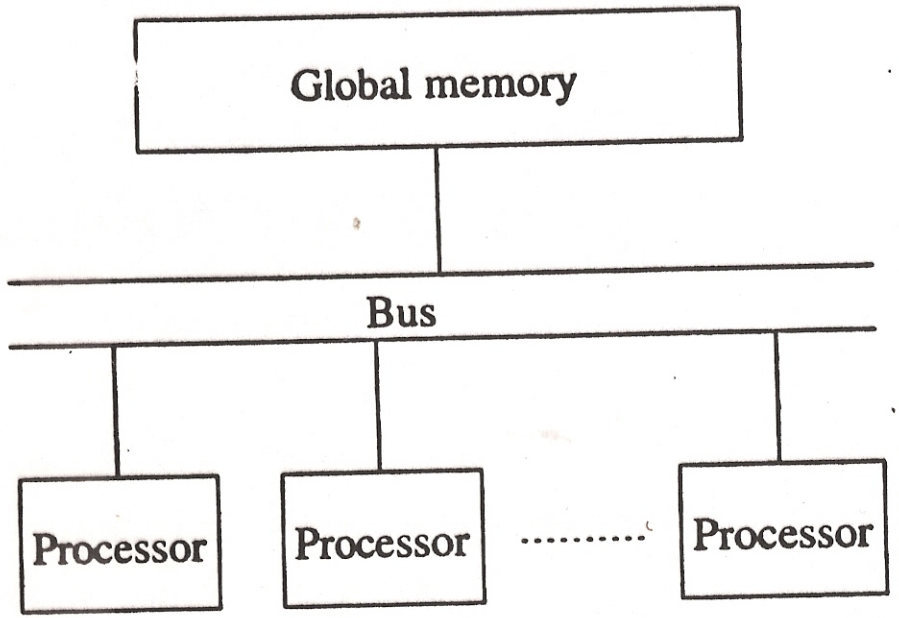
$O(pb)$, $p \leq b$

$O(p^2)$ ~ αν p μεγάλο τότε το δίκτυο είναι μη οικονομικό

αν $p=b \rightarrow p^2$

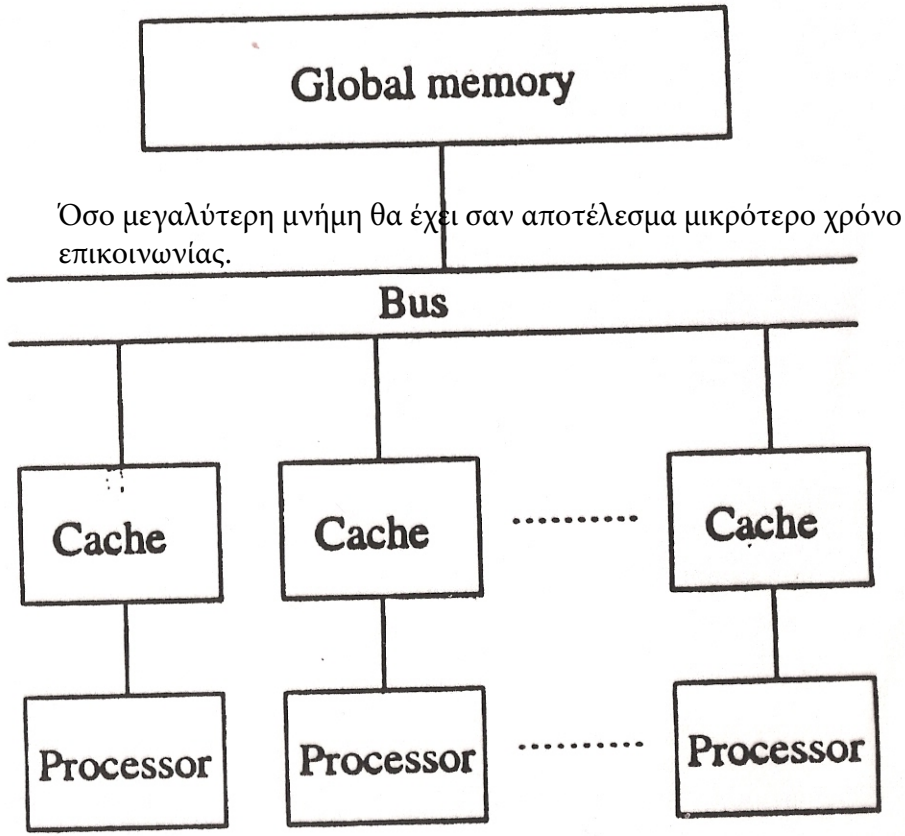
Αρχιτεκτονική με bus : Symmetry, Multimax

• κορεσμός με μικρό πλήθος επεξεργαστών



Τυπική αρχιτεκτονική bus: Η αποδοτικότητα του μπορεί να κορεστεί σε μικρό αριθμό επεξεργαστών.

Ο λόγος που συμβαίνει αυτό οφείλεται στο εξής:
Όποτε ένας επεξεργαστής θέλει να προσπελάσει την κύρια μνήμη κάνει αίτηση στο δίαυλο. Τα δεδομένα τότε αποσπώνται από τη μνήμη του διαύλου. Λόγω απλότητας είναι πολύ ελκηστικός αλλά ο δίαυλος μπορεί να μεταφέρει μόνο ένα περιορισμένο ποσό δεδομένων μεταξύ μνήμης και επεξεργαστών. Αν αυξήσουμε τον αριθμό των επεξεργαστών τότε ο κάθε επεξεργαστής θα ξοδεύει αυξητικά κάποιο χρόνο αναμονής για να προσπελάσει τη μνήμη τη στιγμή που ο δίαυλος χρησιμοποιείται από άλλους επεξεργαστές.



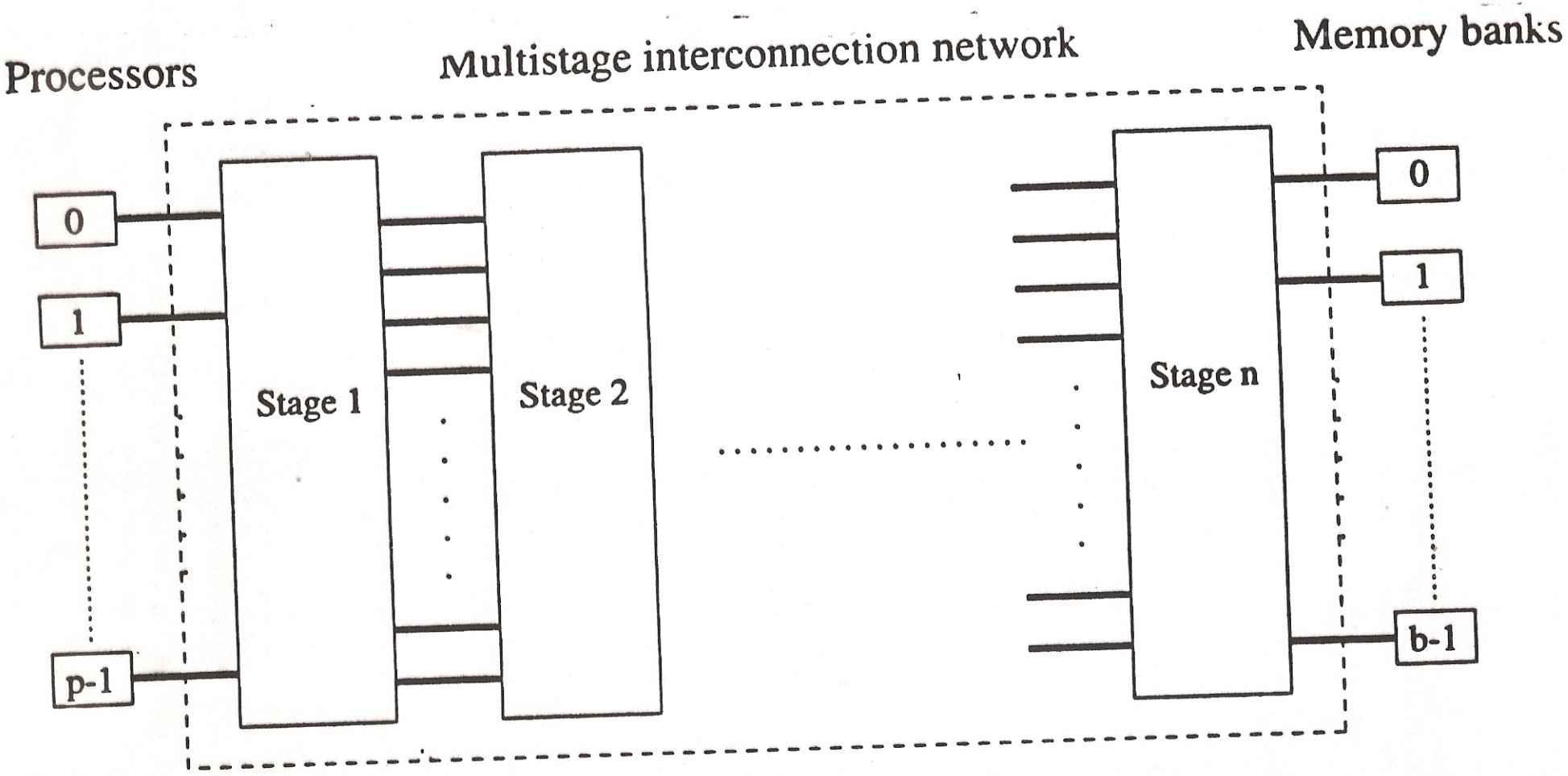
Όσο μεγαλύτερη μνήμη θα έχει σαν αποτέλεσμα μικρότερο χρόνο επικοινωνίας.

Λύση: Σε κάθε επεξ. παρέχω τοπική μνήμη η οποία μειώνει τον συνολικό αριθμό προσπελάσεων στην κύρια μνήμη.

Crossbar interconnection network: Συμφέρει από άποψη αποδοτικότητας, δεν συμφέρει από άποψη κόστους.

Shared bus network: Συμφέρει από άποψη κόστους, δεν συμφέρει από άποψη αποδοτικότητας

Multistage (Πολυφασικό) interconnection network: Ενδιάμεσο των παραπάνω. Η κάθε φάση (stage) είναι δίκτυο μέσω του οποίου ο κάθε επεξ. μπορεί να προσπελάσει κάποιο memory bank.



Το πιο γνωστό είναι το Omega δίκτυο, n=logp

Πολυφασικά Δίκτυα Διασύνδεσης

Αποτελείται από logp φάσεις. Η κάθε φάση αποτελείται από ένα πρότυπο ενδοεπικοινωνίας που συνδέει p εισόδους i και p εξόδους j.

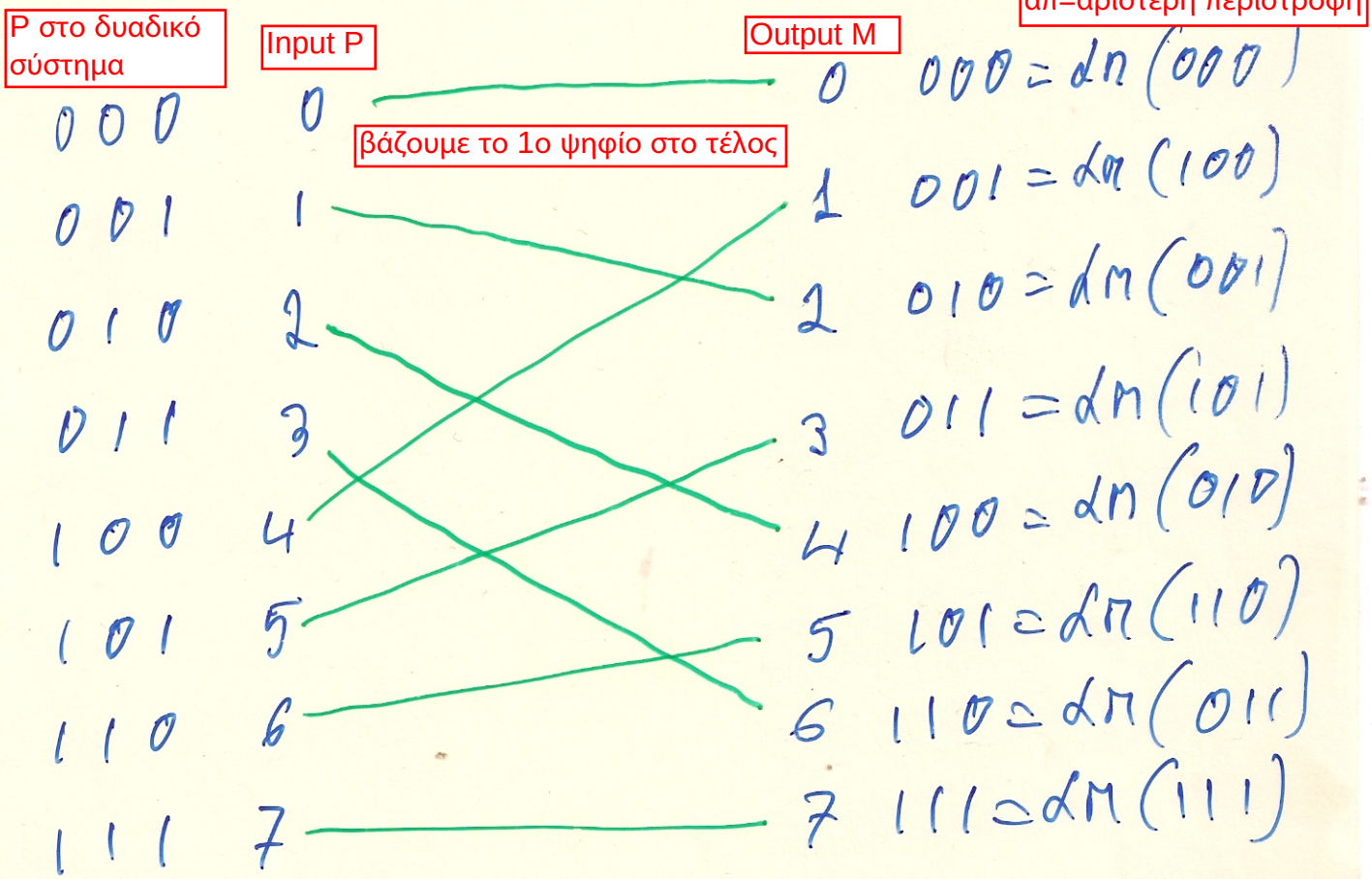
- Δίκτυο ανταλλαγής

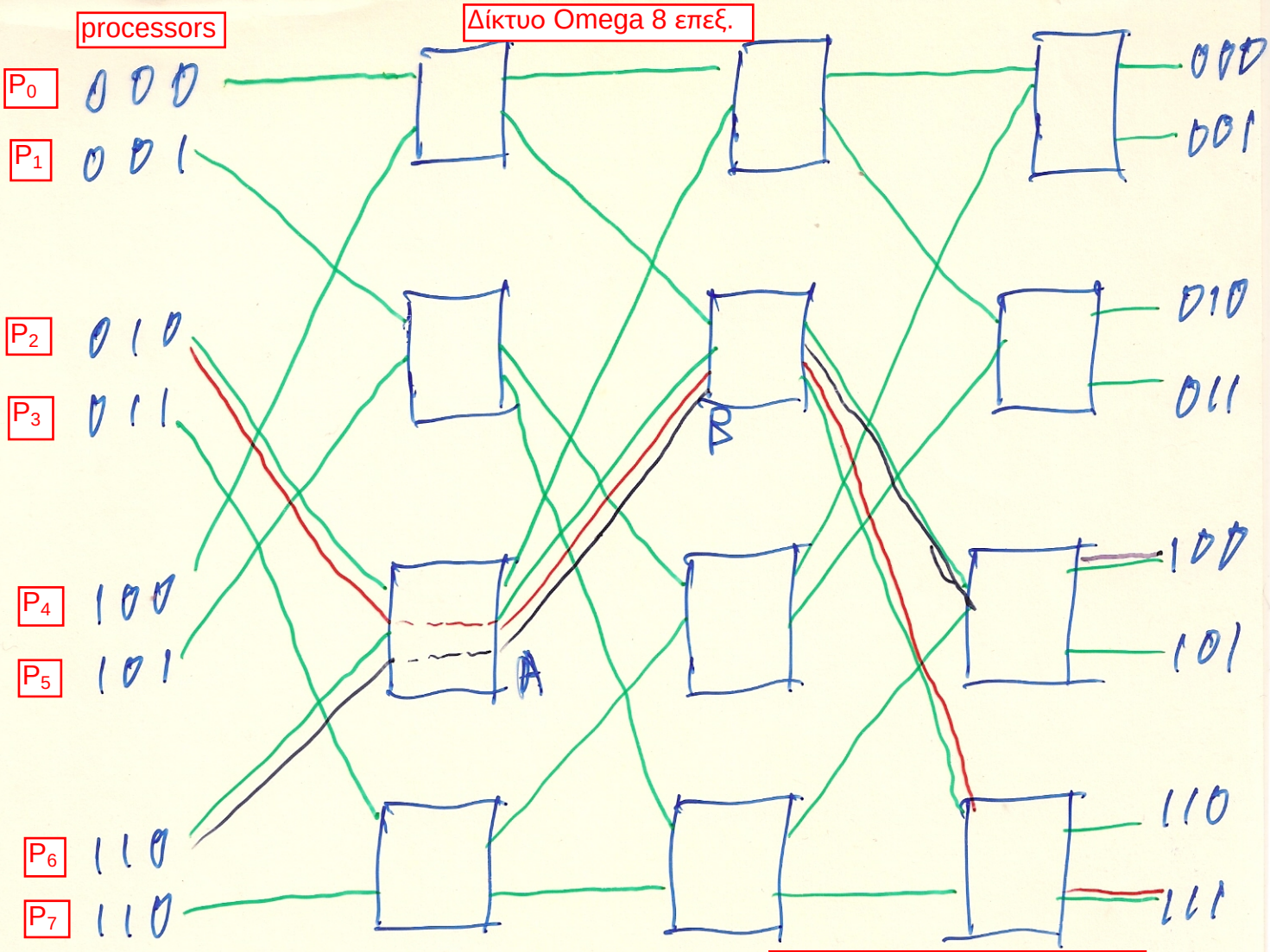
$$j = \begin{cases} 2i, & 0 \le i \le p/2 - 1 \\ 2i + 1 - p, & p/2 \le i \le p - 1 \end{cases}$$

p=αριθμός επεξ. =αριθμός memory banks

αριστερή περιστροφή στη δυαδική παράσταση i για να βρεθεί το j

απ=αριστερή περιστροφή





$p = 8$.

$p/2 \times \log p$ στοιχεία διακόπτες

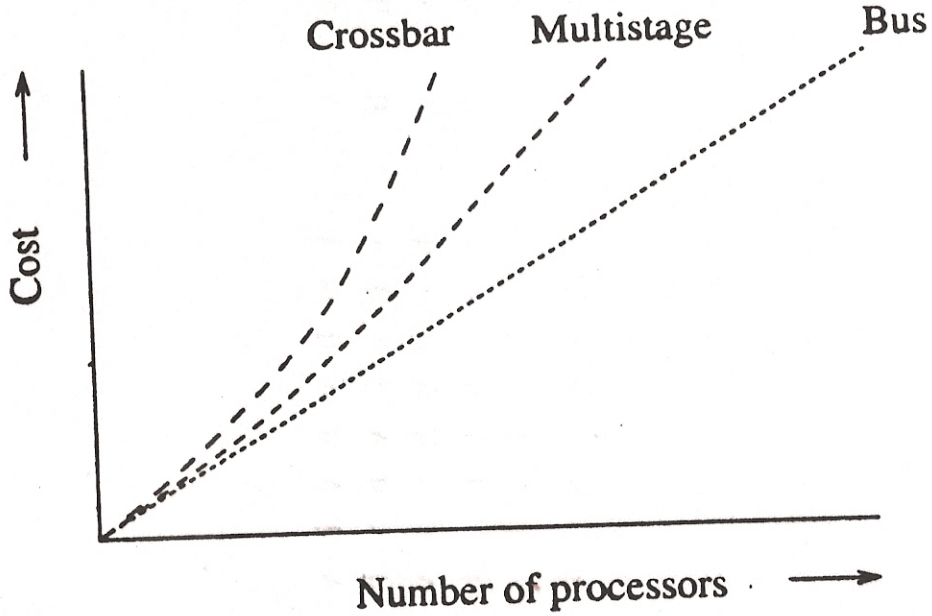
Δίκτυο Θήβα

AB : blocking network

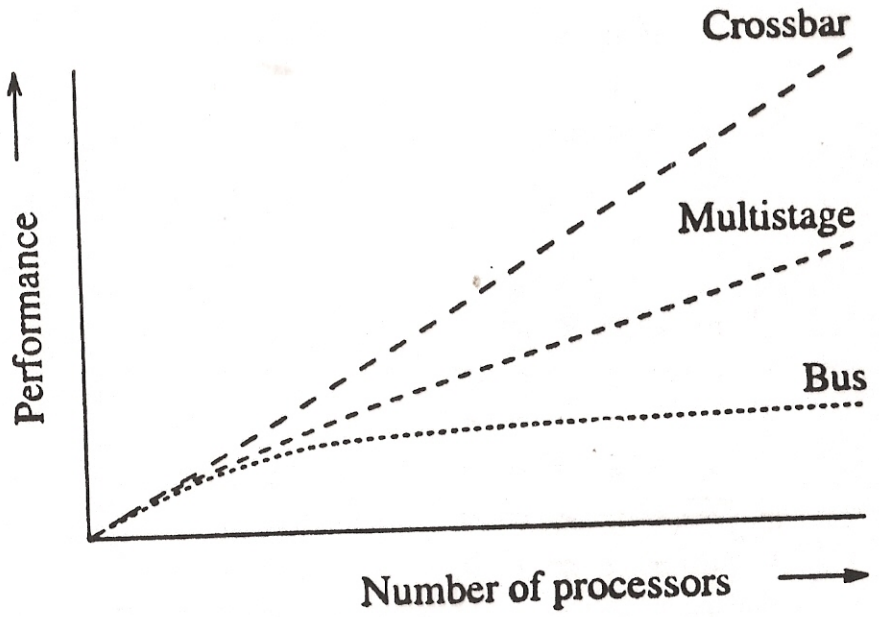
ο ένας επεξ. εμποδίζει τον άλλο

Παράδειγμα υπολογιστές: BBN Butterfly, IBM RP-3, NYU Ultracomputer

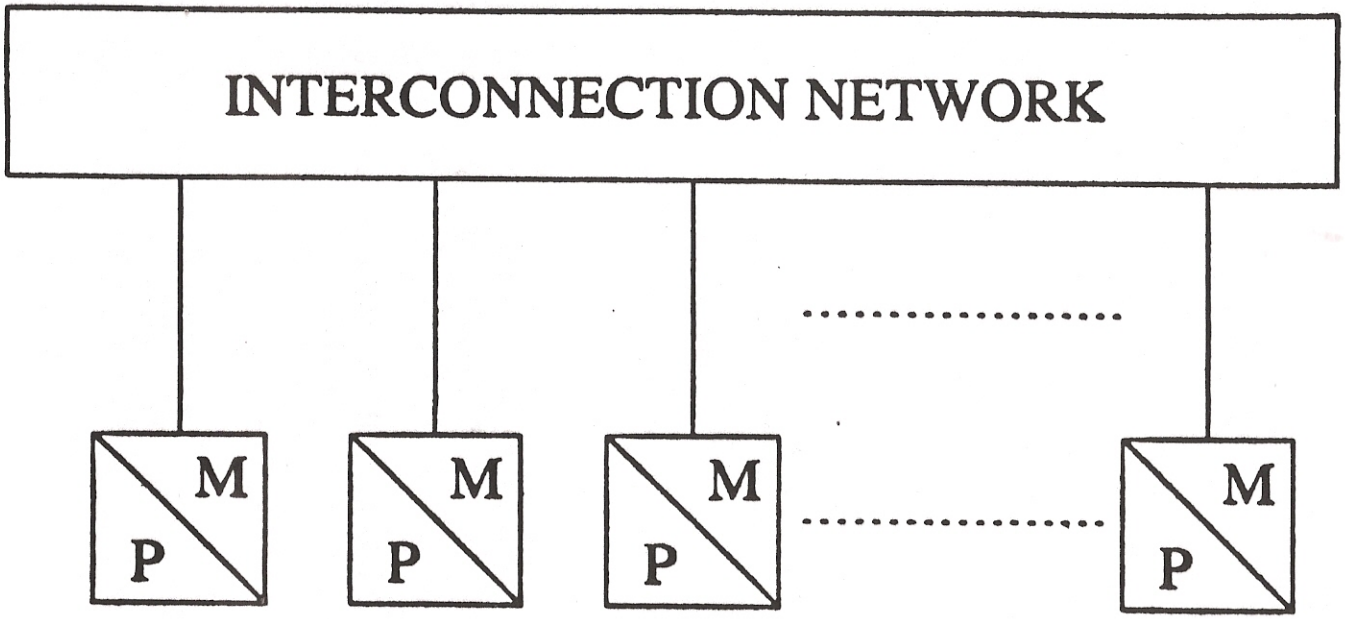
Crossbar: Δεν συμφέρει από άποψη κόστους
Bus: Συμφέρει από άποψη κόστους



Crossbar: Συμφέρει από άποψη αποδοτικότητας
Bus: Δεν συμφέρει από άποψη αποδοτικότητας



Στατικό Δίκτυο: Αποτελείται από point to point συνδέσμους επικοινωνίας μεταξύ των επεξεργαστών.
Δυναμικό Δίκτυο: Φτιάχνεται χρησιμοποιώντας switches και συνδέσμους επικοινωνίας. Οι σύνδεσμοι επικοινωνίας συνδέονται δυναμικά μέσω των διακοπτών προκειμένου να φτιάξουν μονοπάτια μεταξύ επεξεργαστών και memory banks.



P: Processor

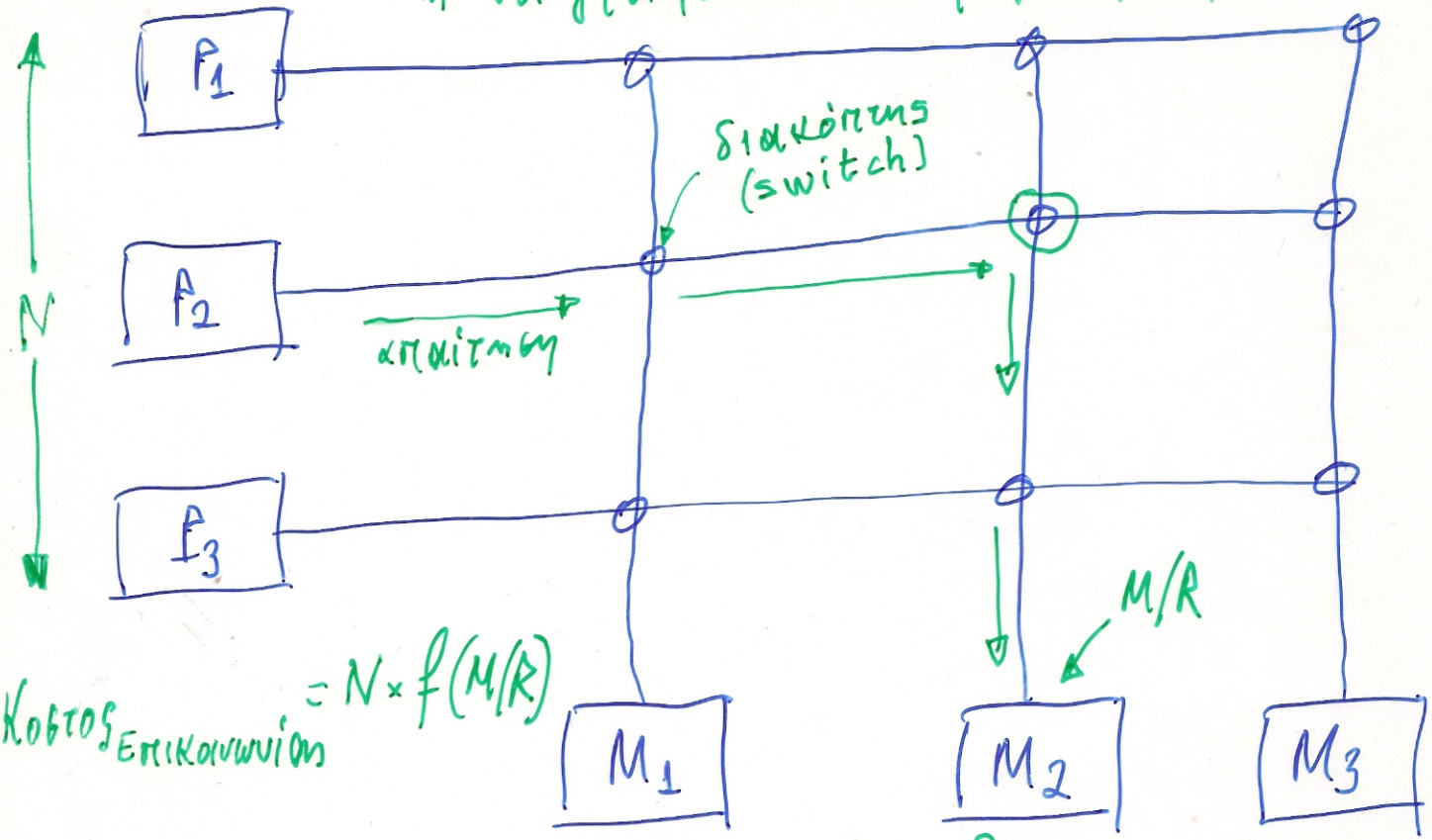
M: Memory

A Typical message passing architecture

επειδή η επικοινωνία μεταξύ των επεξεργαστών γίνεται με μηνύματα.

Cross Bar Switch

• Μόνο ένας επεξεργαστής μπορεί να διαβάσει ή να γράψει σε ένα τμήμα μνήμης.



Κόστος επικοινωνίας $= N \times f(M/R)$

Cray-YMP, Fujitsu VPP500

← επίπεδη διάφ.

Δίκτυο Διασύνδεσης

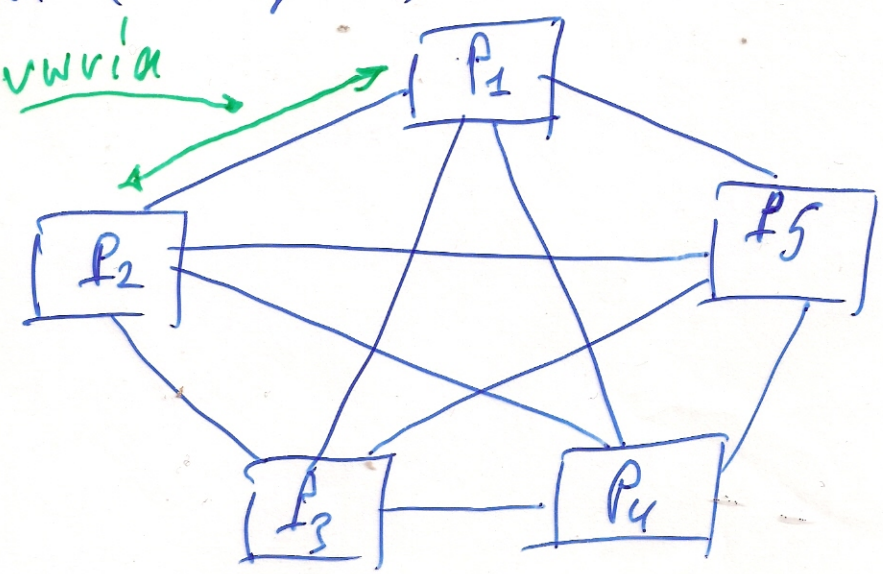
• Οι μνήμες M_i είναι καταναλωμένες μεταξύ των N επεξεργαστών ($M_i = M/N$ δίθυρες μνήμης)

• Δίπλης κατεύθυνσης γραμμή επικοινωνίας μεταξύ των επεξεργαστών. (Μυνήματα)

Κλίκα (clique)

Σύνολο $\frac{N(N-1)}{2}$ γραμμές

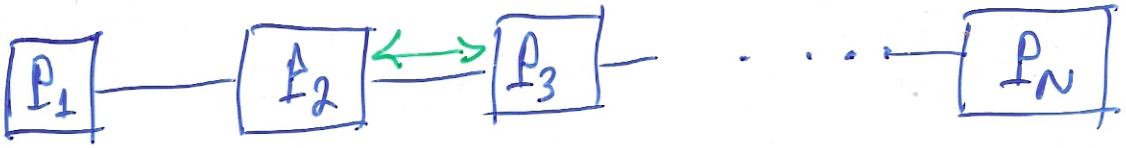
Επικοινωνία



Εξωπραγματικό για μεγάλο N

Δίκτυα σε περίπτωση επεξ. με κατανεμημένη μνήμη

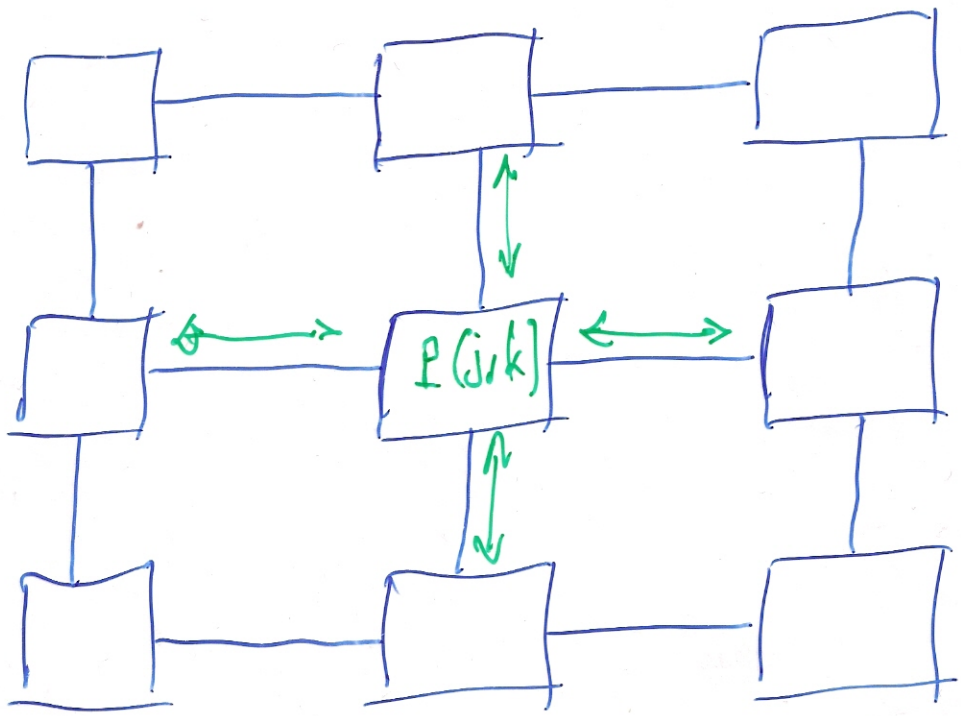
Γραμμικός πίνακας



P_i<->P_j . Επίσης μπορεί και δακτύλιος

Διδιάστατος πίνακας

m x m , $m = \sqrt{N}$ (mesh)

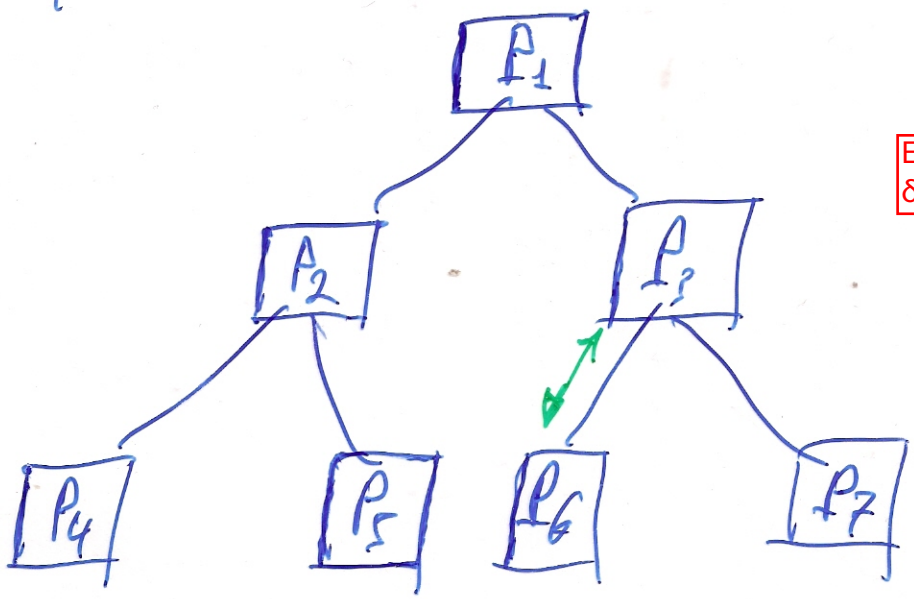


$0 \le j \le m-1$

$0 \le k \le m-1$

P(j,k)<-> 4 γείτονες

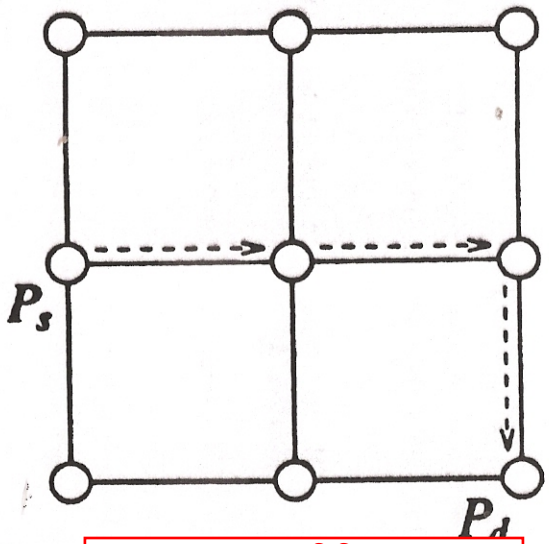
Δέντρο



Επικοινωνία μέσω της ρίζας δέντρου ή υποδέντρου.

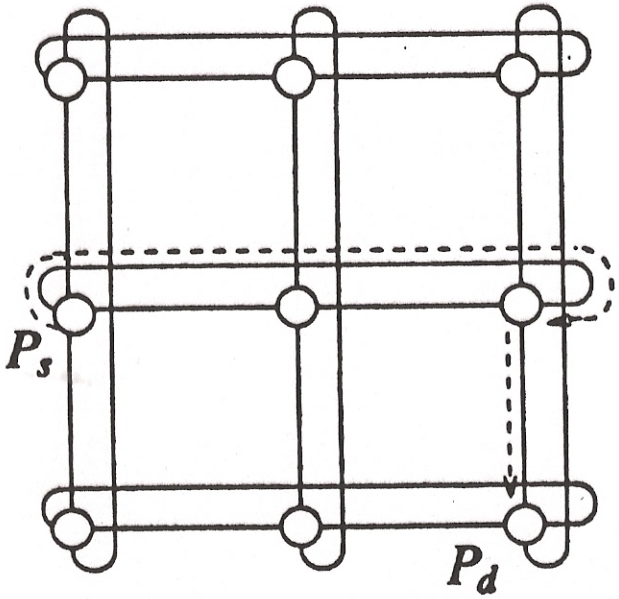
Στο γραμμικό πίνακα η επικοινωνία μεταξύ δύο επεξεργαστών μπορεί να γίνει προς τη μία κατεύθυνση είτε προς την άλλη. Εξαρτάται από το που βρίσκονται. Στο δακτύλιο η επικοινωνία μπορεί να γίνει είτε προς τη μία είτε προς την άλλη κατεύθυνση. Ο δρόμος που θα ακολουθηθεί εξαρτάται από την πολυπλοκότητα μετάδοσης του μηνύματος.

Square Mesh



Το μύνημα μεταδίδεται προς τη μια διάσταση μέχρι να συναντήσουμε συντεταγμένη επεξ. προορισμού οπότε αλλάζει κατεύθυνση προς P_d.

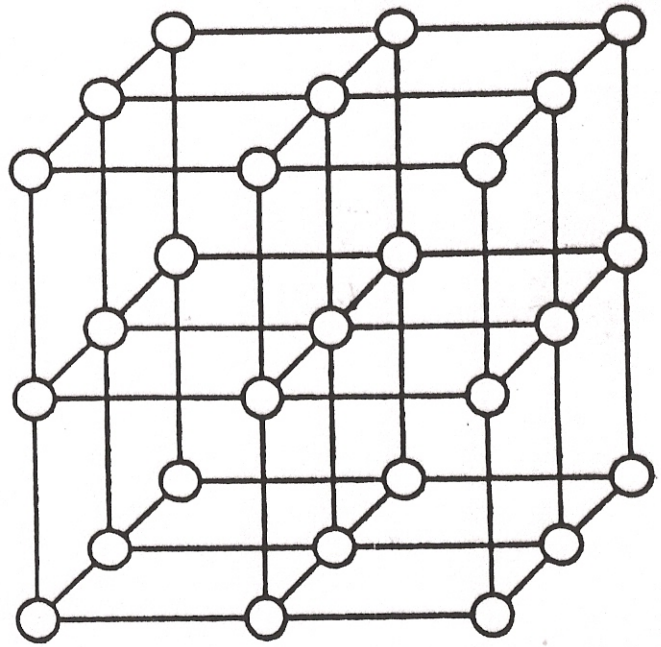
Wraparound Mesh
(Αναδιπλούμενο πλέγμα)



DAP, Paragon XP/S

Επιλογή συντομότερου δρόμου για τη μετάδοση του μυνήματος.

3D Mesh



Cray T3D

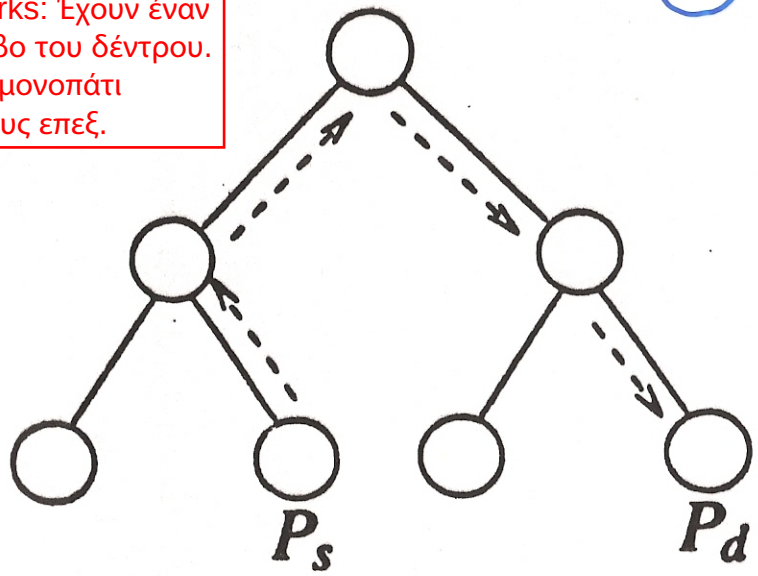
Απλό ή αναδιπλούμενο

Δίκτυο Δέντρου

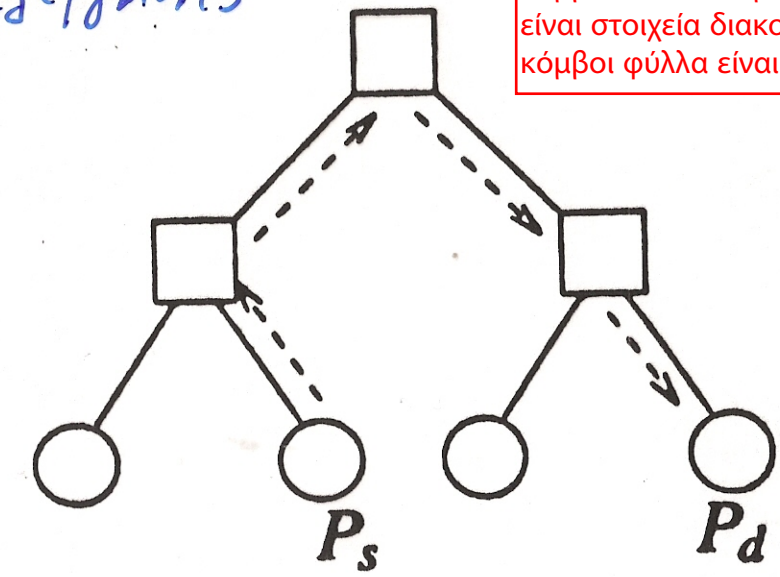
□ Στοιχεία Διακοπτών

○ Επεξεργαστές

Static Tree networks: Έχουν έναν επεξ. σε κάθε κόμβο του δέντρου. Υπάρχει μόνο ένα μονοπάτι μεταξύ κάθε ζεύγους επεξ.



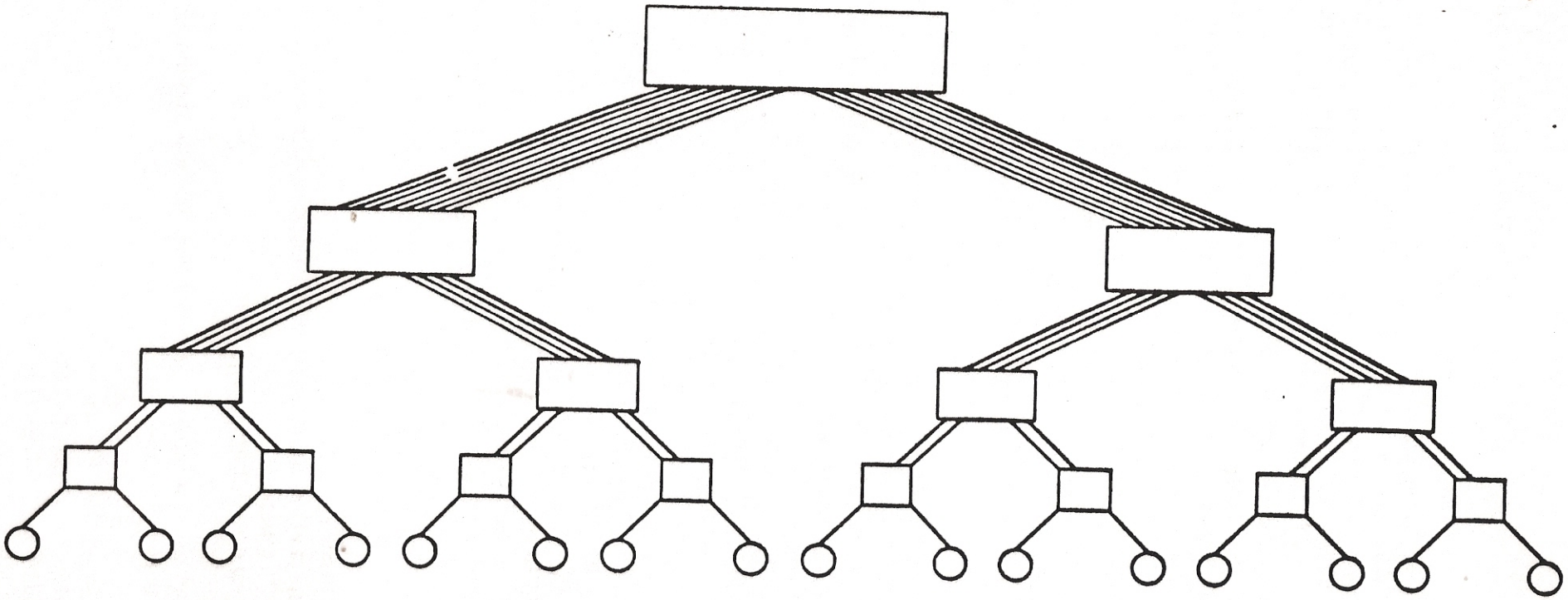
Dynamic tree networks: Οι κόμβοι στα ενδιάμεσα επίπεδα είναι στοιχεία διακοπτών και οι κόμβοι φύλλα είναι επεξ.



- Δίκτυα πλήρους Δυαδικού Δέντρου
- Διαδρομή Μινίμαξ

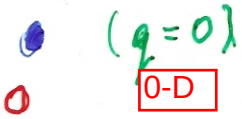
Πρόβλημα: Όταν πολλοί επεξ. του αριστερού υποδέντρου ενός κόμβου επικοινωνούν με επεξ. του δεξιού υποδέντρου ο κόμβος ρίζα έχει να κρατήσει όλα τα μυνήματα.
Λύση: Αύξηση του αριθμού των συνδέσμων επικοινωνίας μεταξύ των επεξ. που βρίσκονται πιο κοντά στη ρίζα.

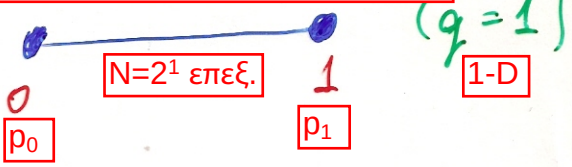
fat tree network

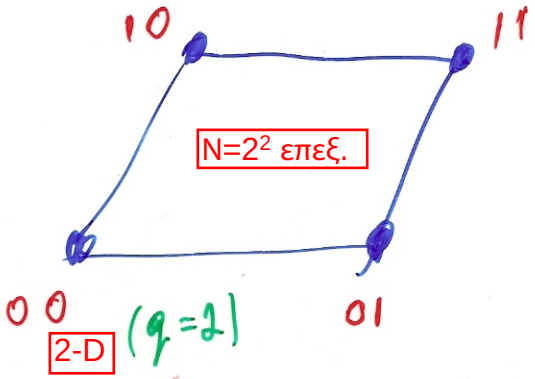


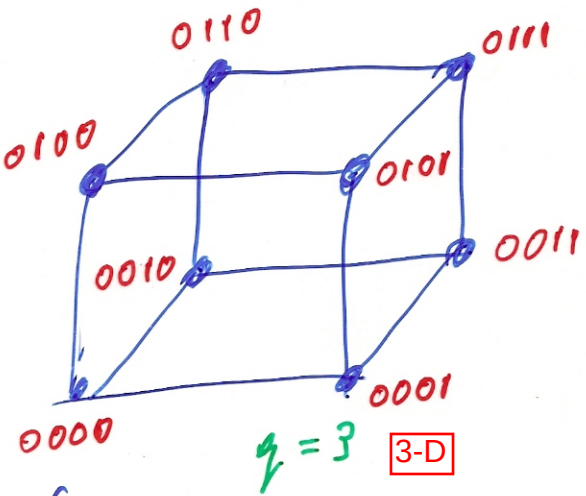
Κύβοι $N = 2^q$, $q \geq 1$

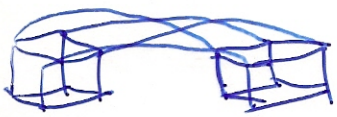
Υπερκύβος: πολυδιάστατο πλέγμα επεξ. με 2 επεξ. σε κάθε διάσταση. Ένας d-διάστασης υπερκύβος αποτελείται από $N=2^q$ επεξ.



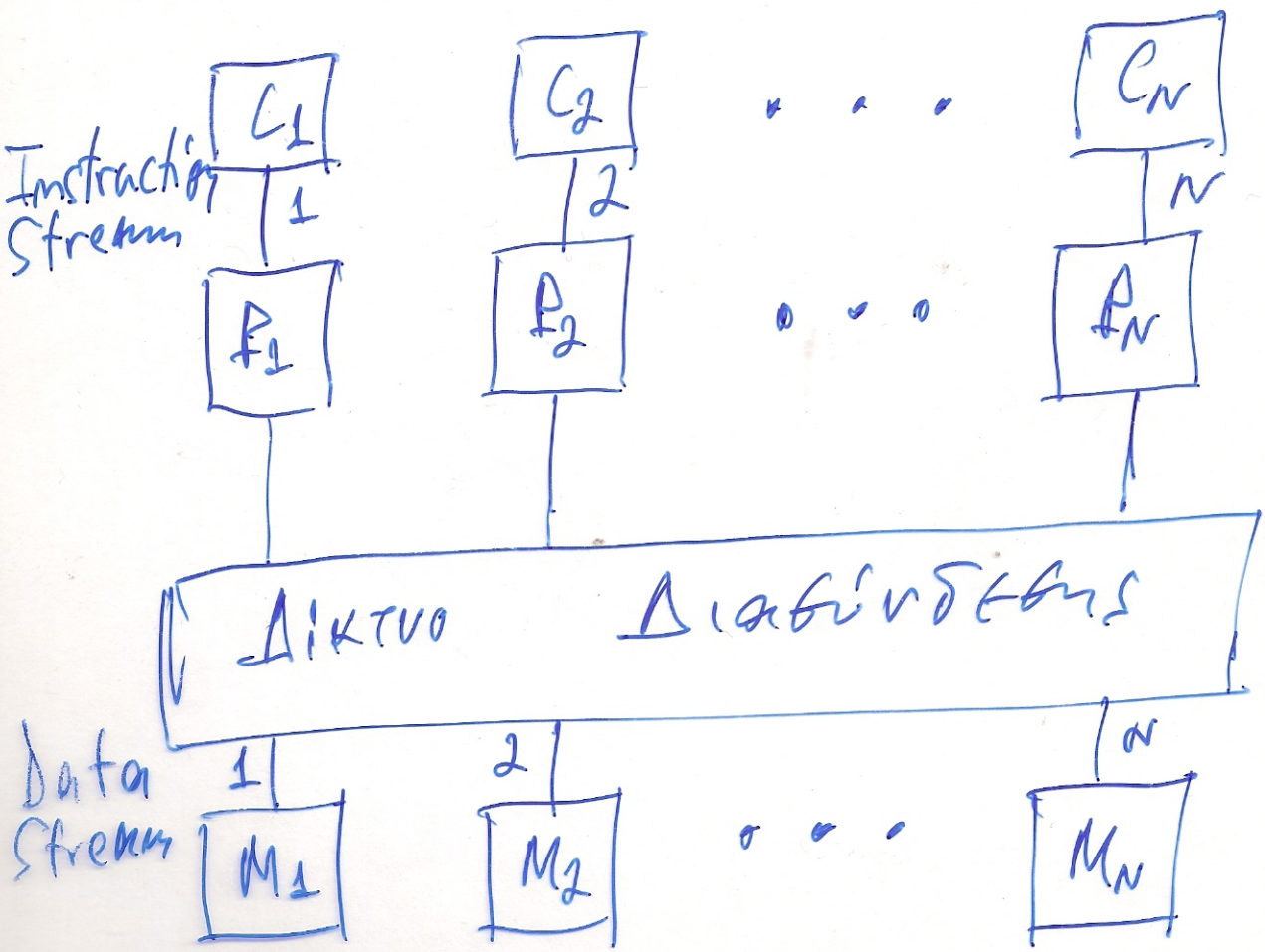








MIMD Υπολογιστές

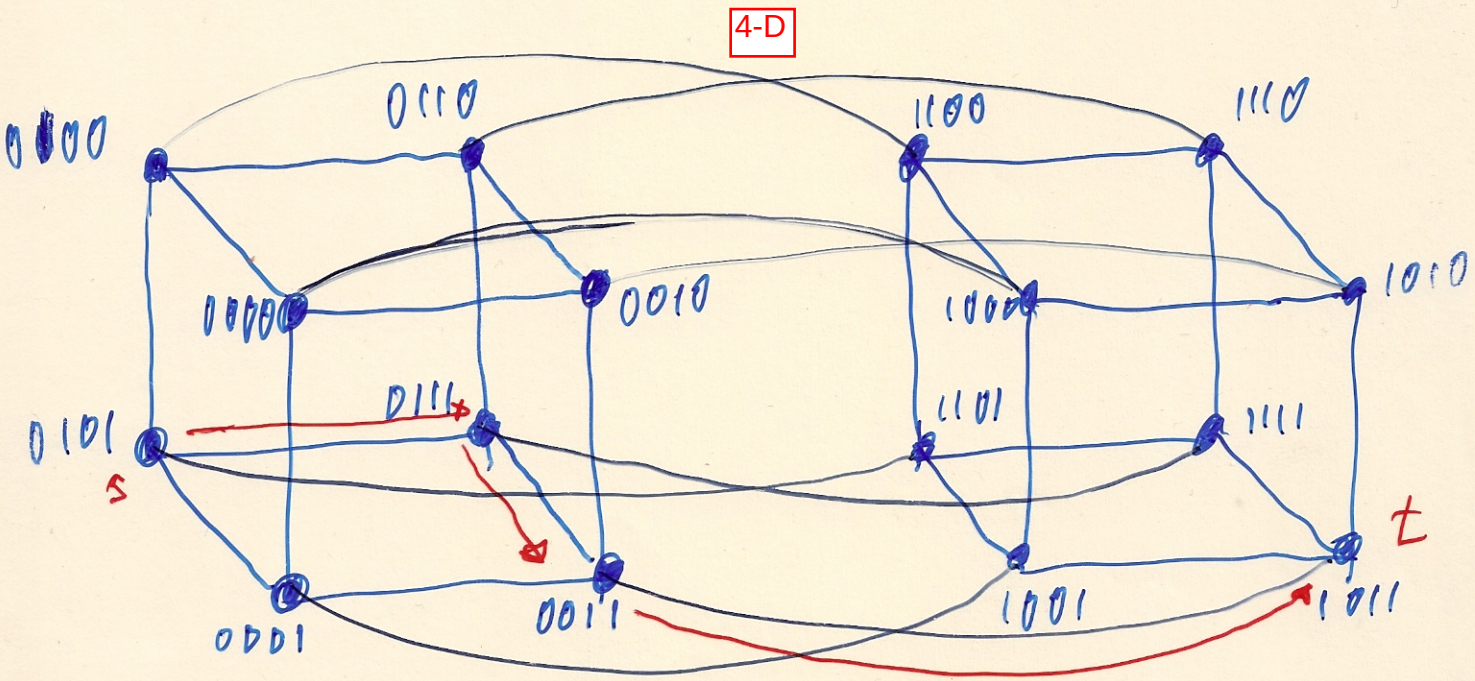


- σύγχρονη λειτουργία
- ασύγχρονη "

Δίκτυο Υπερκύβου

d - διάσταση υπερκύβου

$p = 2^d$ επεξεργαστές



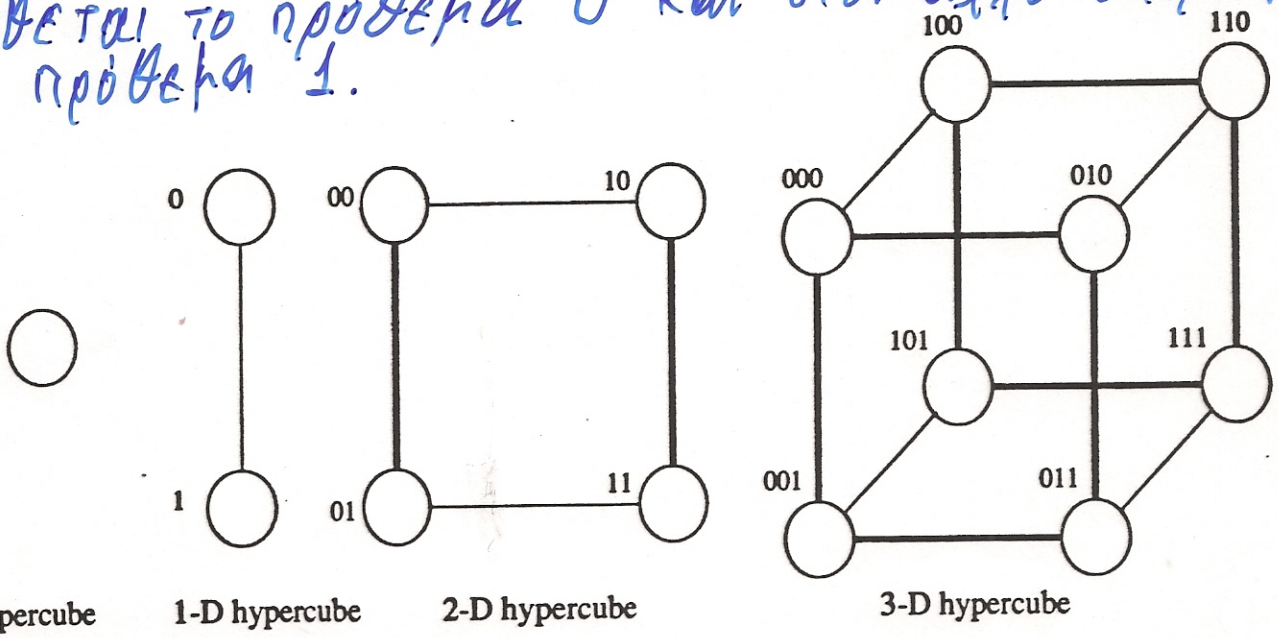
πάμε από το 0101=s στο 0111=t γιατί s XOR t=1110. Αφού έχω 3 φορές το 1 συμπεραίνω ότι ξεκινάω από την 3η θέση και ότι η απόσταση Hamming=3

Ιδιότητες

- Δύο επεξεργαστές συνδέονται τότε και μόνον τότε αν διαφέρουν κατά ένα ακριβώς bit.
- Σε ένα d-διάστατο υπερκύβο ο κάθε επεξεργαστής συνδέεται άμεσα με d άλλους επεξεργαστές.
- Ο συνολικός αριθμός των bit θέσεων στα οποία διαφέρουν δύο επεξεργαστές λέγεται απόσταση του Hamming.

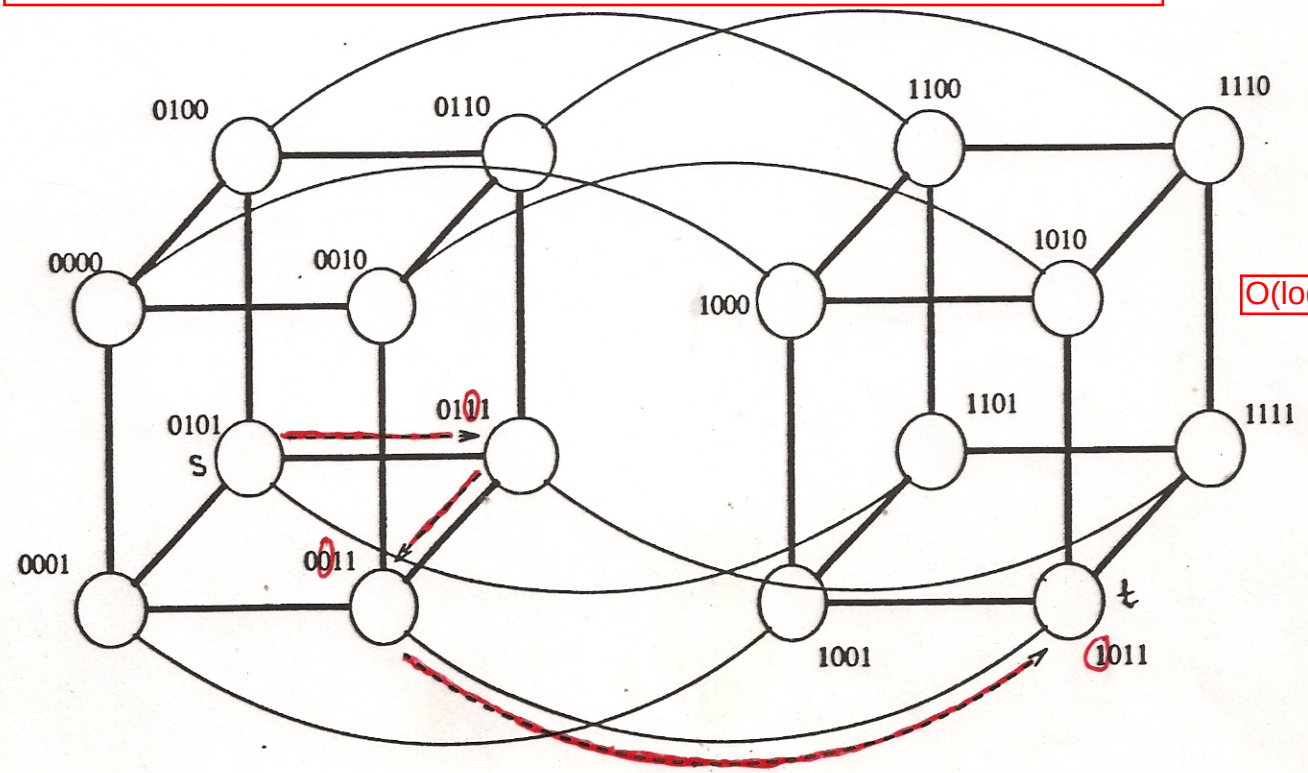
Αριθμός των επεξεργαστών

Όταν ένας υπερκύβος $(d+1)$-διάστασης κατασκευάζεται με την σύνδεση δύο υπερκύβων d-διάστασης, στις ετικέτες του ενός υπερκύβου τίθεται το πρόθεμα 0 και στον άλλο υπερκύβο το πρόθεμα 1.



0-D hypercube 1-D hypercube 2-D hypercube 3-D hypercube

αν είμαστε στον 0111 τότε ξεκινάμε με το 1 και ψάχνουμε το 0 στην αριστερή θέση, άρα πάμε στον 0011. Μετά ψάχνουμε το 1 στην αριστερή θέση και πάμε στον 1011



4-D hypercube

O(log2)

Απόσταση Hamming 0101 και 1011 = 3

s XOR t=1110 $s \oplus t = 1110$

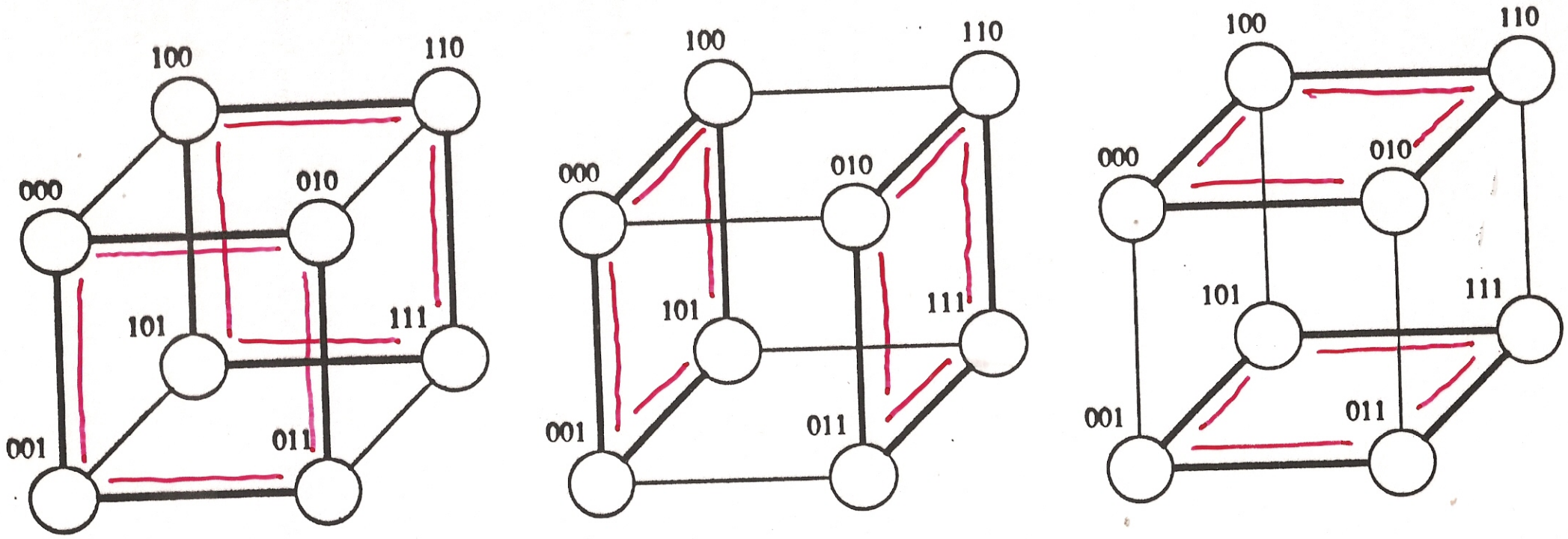
Αφού έχω 3 άσσους -> απόστ. Hamming 3

Ο μόνος επεξεργαστής (γειτονικός) του s που έχει bit 1 στη θέση ένα είναι αυτός που δείχνεται στο σχήμα, κ.ο.κ.

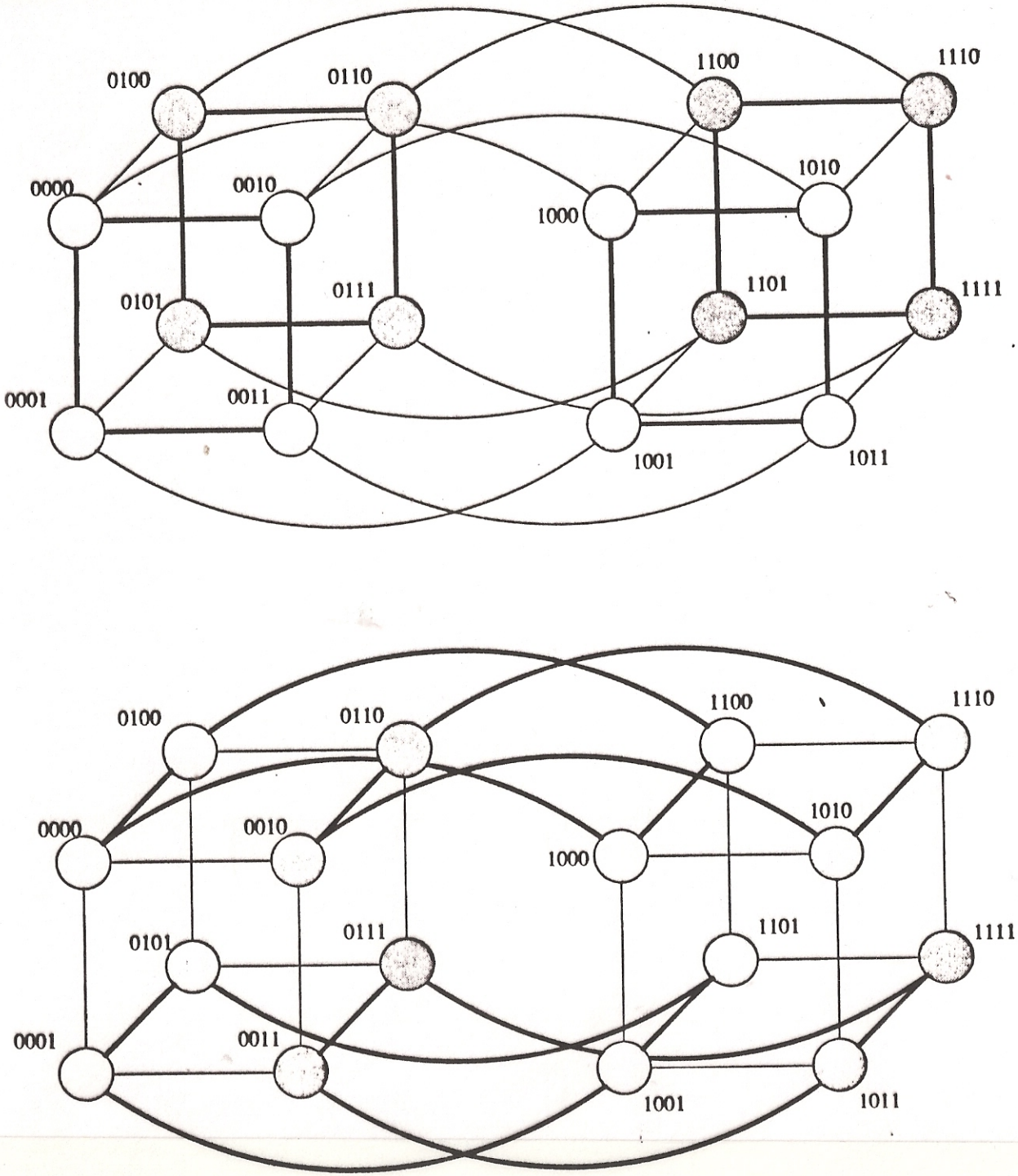
- Η απόσταση του Hamming μεταξύ δύο επεξεργαστών s και t είναι ο συνολικός αριθμός των bits τα οποία είναι 1 στη s⊕t όπου ⊕ συμβολίζει xor.

- Ο αριθμός των συνδέσμων επικοινωνίας στο συντομότερο μονοπάτι μεταξύ δύο επεξεργαστών είναι ίσος με την απόσταση του Hamming. Ένα μήνυμα δρομολογείται από τον επεξεργαστή s στον επεξεργαστή t διαμέσου διαστάσεων οι οποίες αντιστοιχούν σε θέσεις bit που έχουν 1 στη δυαδική παράσταση της s⊕t. Επειδή η δυαδική παράσταση του s⊕t μπορεί να περιέχει το πολύ d μονάδες, το συντομότερο μονοπάτι μεταξύ δύο οποιωνδήποτε επεξεργαστών σε ένα υπερκύβο δεν μπορεί να έχει περισσότερους από d συνδέσμους.

Παράλληλοι Υπολογιστές βασισμένοι σε δίκτυο υπερκύβου είναι οι: nCUBE, Cosmic Cube και iPSC.



Τρεις διαφορετικές διαμερίσεις ενός 3-D υπερκύβου σε κύβους 2 διαστάσεων.
Οι σύνδεσμοι μεταξύ των επεξεργαστών στην κάθε διαμέριση εμφανίζονται με τις έντονες γραμμές.



Ένωση υποκύβων 2-D σ' ένα υπερκύβο 4-D ταιριάζοντας (α) τα δύο περισότερο σημαντικά label bits και (β) τα δύο λιγότερο σημαντικά label bits. Οι επεξεργαστές σε κάθε υποκύβο είναι συνδεδεμένοι με τις έντονες γραμμές.

Diameter: Η μέγιστη απόσταση μεταξύ δύο οποιονδήποτε επεξεργαστών του δικτύου. Η απόσταση αυτή ορίζεται σαν το συντομότερο μονοπάτι (ως προς τον αριθμό των συνδέσμων)μεταξύ τους. Αφού η απόσταση προσδιορίζει τον χρόνο επικοινωνίας, είναι καλύτερα τα δίκτυα με τις μικρότερες διαμέτρους.

Connectivity: Μέτρο πλήθους μονοπατιών μεταξύ δύο επεξεργαστών.
Arc connectivity: Ο ελάχιστος αριθμός τόξων που πρέπει να διαγραφούν από ένα δίκτυο ώστε αυτό να χωρίσει σε δύο μη συνδεδεμένα δίκτυα.

Bisection Width: Ο ελάχιστος αριθμός των συνδέσμων επικοινωνίας που πρέπει να διαγραφούν ώστε να χωρίσει το δίκτυο σε δύο ίσα μέρη.

Cost: Ο αριθμός των συνδέσμων επικοινωνίας.

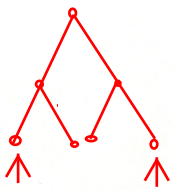
Network	Diameter	Bisection Width	Arc Connectivity	Cost (No. of links)
Completely-connected	1	$p^2/4$	$p-1$	$p(p-1)/2$
Star	2	1	1	$p-1$
Complete binary tree	$2\log((p+1)/2)$	1	1	$p-1$
Linear array	$p-1$	1	1	$p-1$
Ring	$\lfloor p/2 \rfloor$	2	2	p
2-D mesh without wraparound	$2(\sqrt{p}-1)$	$\sqrt{p}$	2	$2(p-\sqrt{p})$
2-D wraparound mesh	$2\lfloor \sqrt{p}/2 \rfloor$	$2\sqrt{p}$	4	$2p$
Hypercube	$\log p$	$p/2$	$\log p$	$(p \log p)/2$
Wraparound k-ary d-cube	$d\lfloor k/2 \rfloor$	$2k^{d-1}$	$2d$	dp

Επικοινωνία πρώτου με τελαυταίο

Στην επικ. θα πάμε είτε από αριστερά είτε από δεξιά

σαν ring

$p=2^d \rightarrow d=\log p$

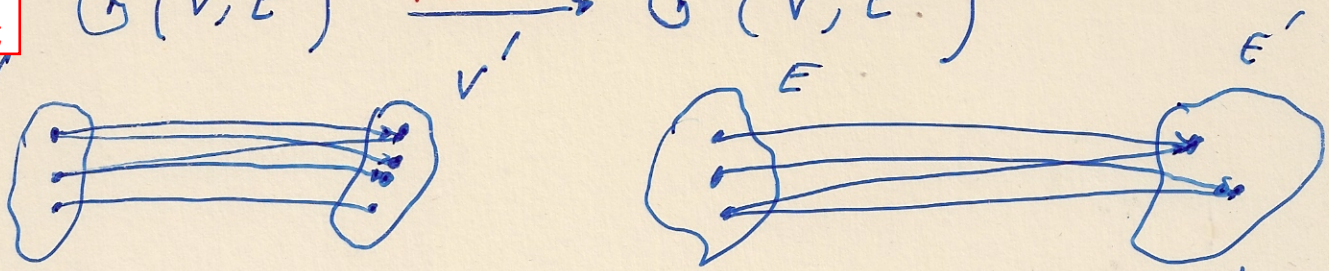


Complete binary tree:
για την επικοινωνία των δύο ακραίων επεξ. έχουμε
$p=2^{n+1}-1 \rightarrow (p+1)/2=2^h \rightarrow h=\log((p+1)/2)$
οπότε για να επικοινωνήσω θέλω
$2h=2\log((p+1)/2)$

V=κορυφές=
επεξεργαστές
E=ακμές=
σύνδεσμοι
επικοινωνίας

ΕΜΦΥΤΕΥΣΗ ΔΙΚΤΥΟΥ ΣΤΟΝ ΥΠΕΡΚΥΒΟ

$G(V,E)$ —εμφύτευση→ $G'(V',E')$



Η εμφύτευση ενός γραφήματος εντός ενός άλλου είναι επιθυμητή διότι ένας αλγόριθμος που έχει σχεδιαστεί για ένα συγκεκριμένο δίκτυο διασύνδεσης μπορεί να είναι αναγκαίο να προσαρμοστεί σε ένα άλλο δίκτυο.

- **συμφόρηση** (congestion)
  Το μέγιστο πλήθος ακμών που απεικονίζονται σε μια ακμή στο E'
  Πολλές ακμές του E μπορούν να απεικονιστούν σε μια γραμμή του E'
- **διαστολή** (dilation)
  Το πλήθος συνδέσμων στο E' στους οποίους απεικονίζεται μία ακμή του E.
  Μία ακμή του E μπορεί να απεικονιστεί σε περισσότερες από μία του E'
- **επέκταση** (expansion)
  Επεξεργαστές στο V' / Επεξεργαστές στο V
  Κάθε επεξ. του V αντιστοιχεί σε περισσότερους από έναν επεξ. του V'

Στη συνέχεια θα μελετηθούν οι εμφυτεύσεις διαφόρων δικτύων διασύνδεσης εντός του υπερκύβου. Η μελέτη θα περιορισθεί στην περίπτωση όπου το πλήθος των επεξεργαστών είναι το ίδιο στα δύο δίκτυα (επέκταση = 1). Επιπλέον το πολύ μία ακμή του E απεικονίζεται σε μία ακμή του E' (συμφόρηση = 1).

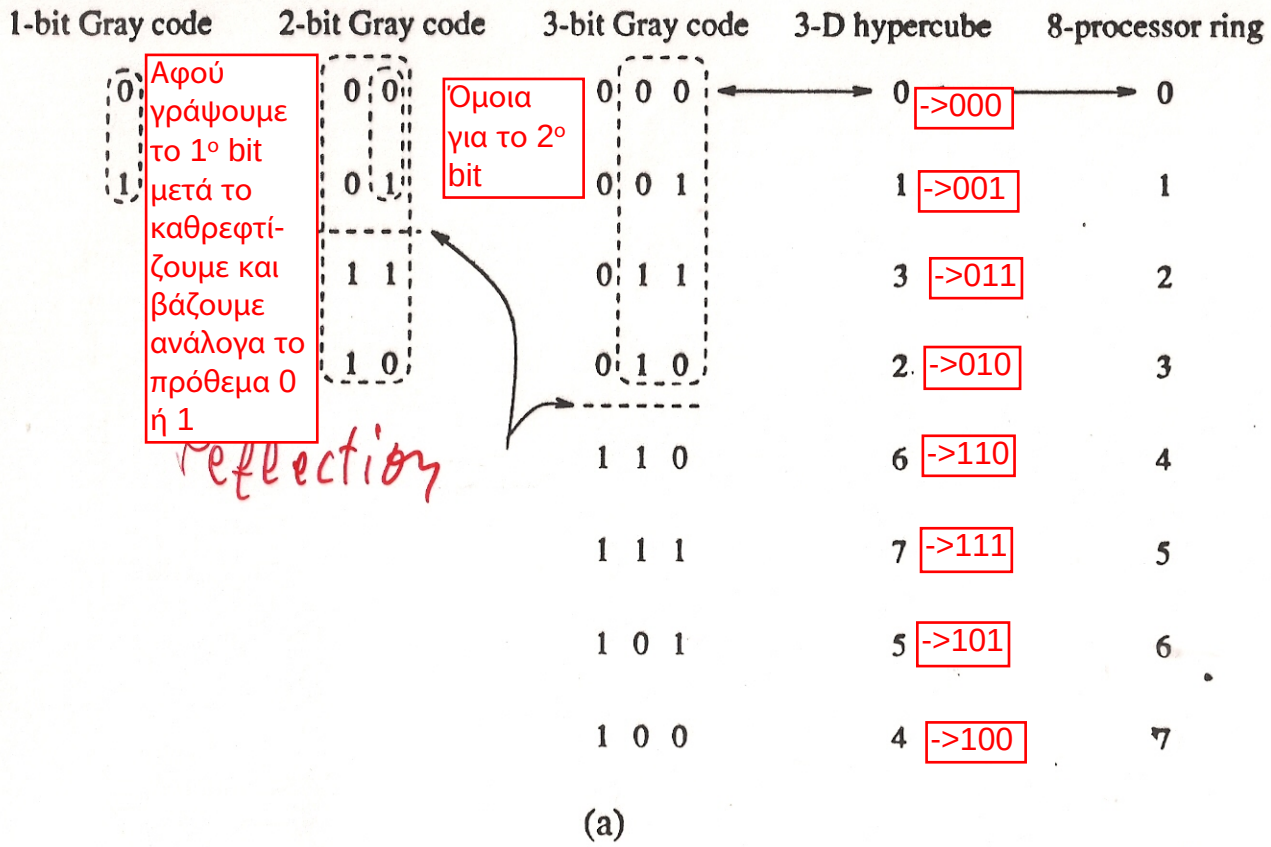
Εφύτευση γραφημικού πίνακα σε υπερκύβο

Ένας γραφημικός πίνακας (ή ένας δακτύλιος) που αποτελείται από 2^d επεξεργασίες μπορεί να εφυτευθεί σε ένα υπερκύβο d διαστάσεως αντιστοιχίζοντας την i-οστή επεξεργασία του γραφημικού πίνακα στην επεξεργασία $G(i,d)$ του υπερκύβου, όπου

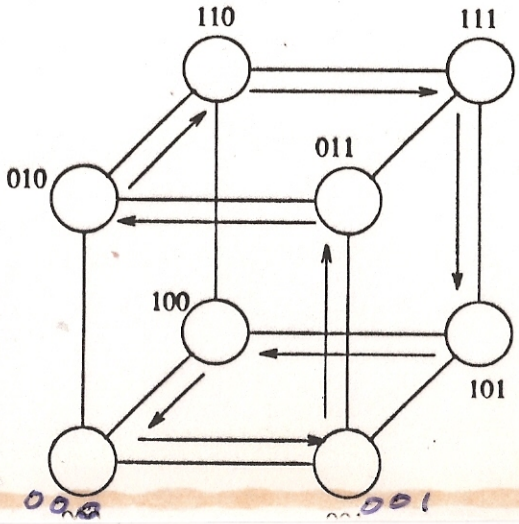
$$G(0,1) = 0$$
$$G(1,1) = 1$$

$$G(i,x+1) = \begin{cases} G(i,x), & i < 2^x \\ 2^x + G(2^{x+1}-1-i,x), & i \geq 2^x \end{cases}$$

Η συνάρτηση G λέγεται binary reflected Gray code (RGC). Η $G(i,d)$ αντιστοιχίζει τον i-οστό στοιχείο στην ακολουθία των κωδίκων Gray με d bits. Οι κώδικες Gray $d+1$ bits παράγονται από ένα πίνακα των κωδίκων Gray με d bits ως εξής: Παίρνουμε το αντιστοιχικό πίνακα και τοποθετώντας στα στοιχεία του αντιστοιχικού πίνακα το πρόθεμα 1 και στα λοιπά στοιχεία το πρόθεμα 0.



(a)



Εκφύτευση ενός δακτυλίου 8 επεξεργαστών σε ένα υπερκύβο διάστασης 3.

Η G(i,d) συμβολίζει το i-οστό στοιχείο στην ακολουθία των κωδίκων Gray με d bits
i-> G(i,d), d=3 2^d
i=5-> G(5,3)->7

d=πλήθος bits ή d διάσταση για τον i,
i=i-οστό στοιχείο ακολουθίας

Τα στοιχεία $G(i,d)$ και $G(i+1,d)$ διαφέρουν μόνο σε ένα bit. Επειδή ο επεξεργαστής i στο γραμμικό πίνακα αντιστοιχίζεται στον επεξεργαστή $G(i,d)$ του υπερκύβου και ο επεξεργαστής i+1 του γραμμικού πίνακα στον επεξεργαστή $G(i+1,d)$ του υπερκύβου, υπάρχει ένας άμεσος σύνδεσμος στον υπερκύβο, ο οποίος αντιστοιχεί σε κάθε άμεσο σύνδεσμο στο γραμμικό πίνακα (οι ετικέτες δύο γειτονικών επεξεργαστών διαφέρουν μόνο κατά ένα bit). Συνεπώς, η απεικόνιση που ορίζεται από την G έχει συμφόρηση ένα και διασπορά ένα.

Εμφύτευση δυδιάστατου δικτύου (mesh) σε υπερκύβο

Μπορούμε να εμφυτεύσουμε ένα $2^r \times 2^s$ wraparound mesh σε ένα 2^{r+s} επεξεργαστές υπερκύβο απεικονίζοντας τον επεξεργαστή (i,j) του mesh στον επεξεργαστή $G(i,r) \| G(j,s)$ του υπερκύβου όπου $\|$ συμβολίζει τη συνένωση των δύο κωδικών του Gray. Ας υποθέσουμε ότι οι άμεσοι γείτονες

Άρα, κάθε επεξ. του πλέγματος τον αντιστοιχώ σε 1 επεξ. του υπερκύβου.

Στο mesh αντιστοιχούνται σε επεξεργαστές του υπερκύβου των οποίων οι ετικέτες διαφέρουν μόνο σε ένα bit. Συνεπώς η αντιστοίχιση έχει μία διάσταση = ένα και μία συσσώρευση = 1.

2x4 mesh

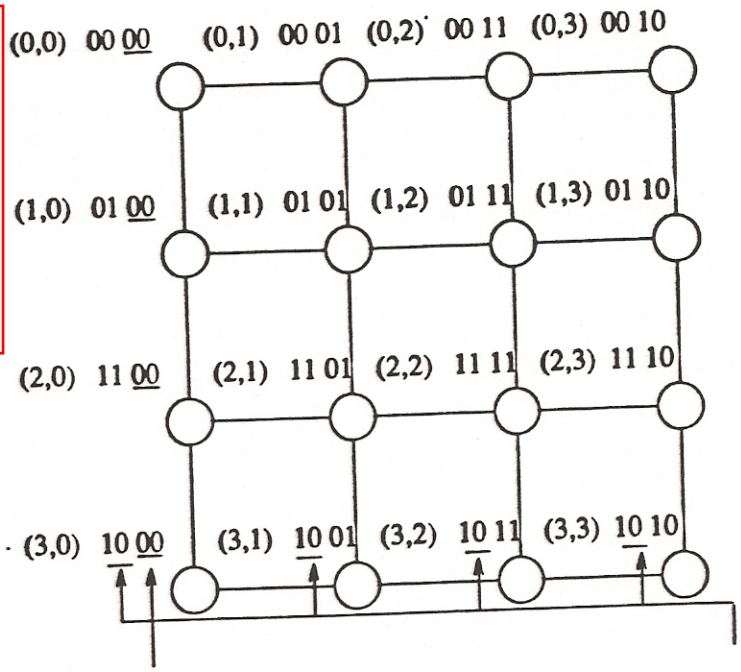
r=1, s=2 (i,j) ⟶ G(i,1)||G(j,2)

(0,0) → G(0,1)||G(0,2) = 0||00 = 000

(0,1) → G(0,1)||G(1,2) = 0||01 = 001

G(0,1)->θέλουμε να εκφράσουμε με 1 bit το 0, δηλ. 0
G(0,2)->θέλουμε να εκφράσουμε με 2 bit το 0, δηλ. 00
G(1,2)->θέλουμε να εκφράσουμε με 2 bit το 1, δηλ. 01

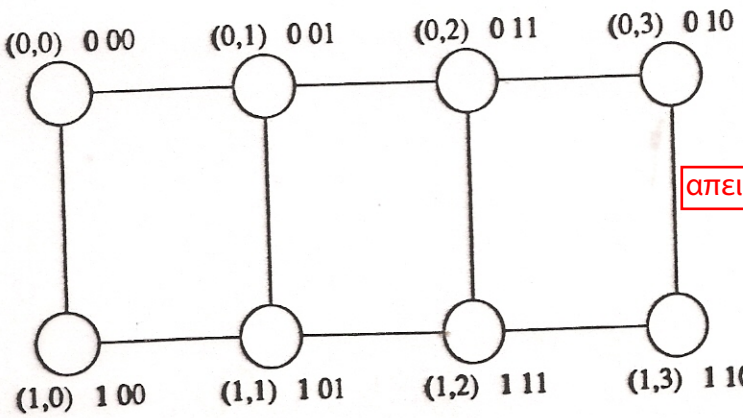
(a) 4x4 πλέγμα που εμφανίζει την απεικόνιση ενός πλέγματος επεξεργαστών σε έναν 4-D υπερκύβο



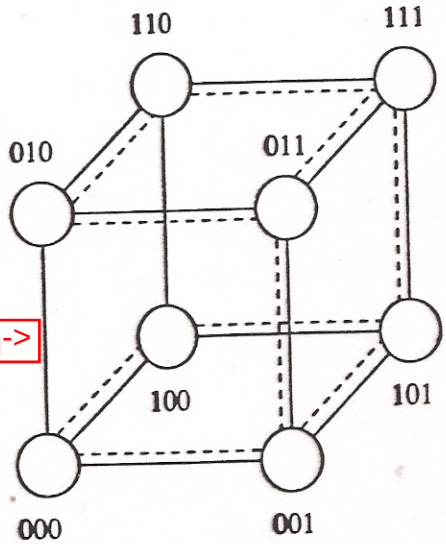
Processors in a column have identical two least-significant bits

Processors in a row have identical two most-significant bits

(b) 2x4 πλέγμα επεξεργαστών εμφωλευμένο σ' ένα 3-D υπερκύβο



απεικόνιση->

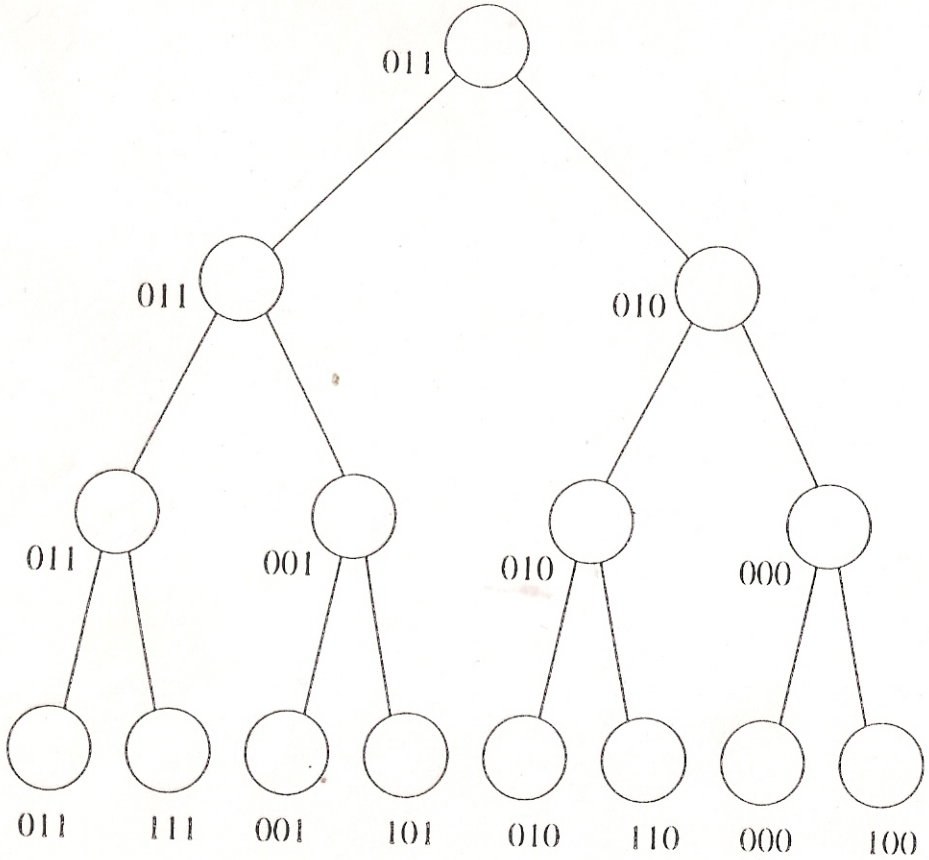


2x4 mesh-> $2^1 x 2^2$ mesh άρα r=1, s=2

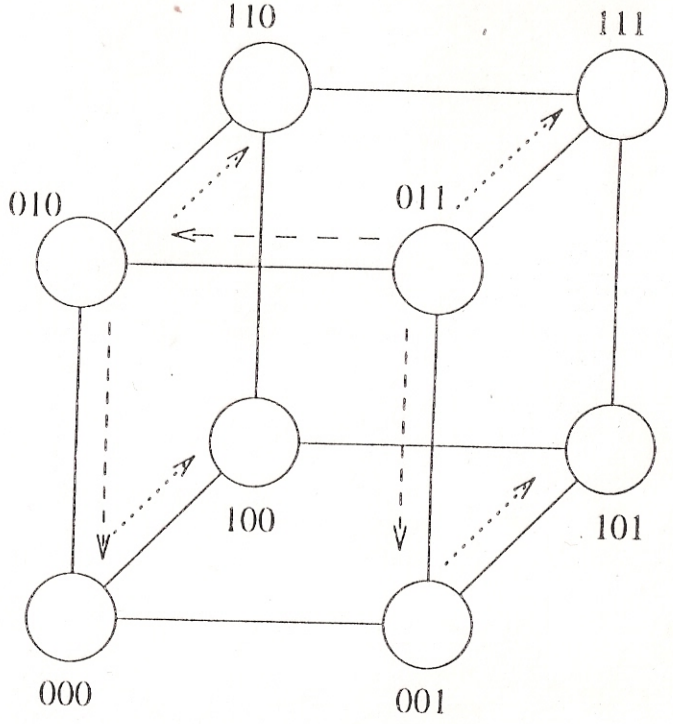
Για τον τυχόν (i,j) επεξεργαστή έχουμε
(i,j)->G(i,r)||G(j,s)=G(i,1)||G(j,2)

(0,0)->G(0,1)||G(0,2)=0||00=000
(0,1)->G(0,1)||G(1,2)=0||01=001
(0,2)->G(0,1)||G(2,2)=0||11=011
(0,3)->G(0,1)||G(3,2)=0||10=010
(1,0)->G(1,1)||G(0,2)=1||00=100
(1,1)->G(1,1)||G(1,2)=1||01=101
(1,2)->G(1,1)||G(2,2)=1||11=111
(1,3)->G(1,1)||G(3,2)=1||10=110

Ένα δέντρο δρομολογημένο στον επεξεργαστή 011 (=3) και εμφωλευμένο σε ένα 3-D υπερκύβο.
(α) Η οργάνωση του δέντρου δρομολογείται από τον επεξεργαστή 011 και
(β) το εμφωλευμένο δέντρο σ' ένα 3-D υπερκύβο.



(a)



(b)

Δρομολόγηση Μηνυμάτων

Ένας μηχανισμός δρομολόγησης καθορίζει το μονοπάτι που θα ακολουθήσει το μήνυμα για να πάει από τον πηγαίο επεξεργαστή (P_s) στον επεξεργαστή προορισμού (P_d).

Μηχανισμοί Δρομολόγησης

εξάρχιστοι

Ακολουθούν την συντομότερη διαδρομή μεταξύ P_s και P_d.

μη-εξάρχιστοι

Ακολουθούν την μακρύτερη διαδρομή προκειμένου να αποφευχθεί συμφόρηση του δικτύου

Μηχανισμοί Δρομολόγησης

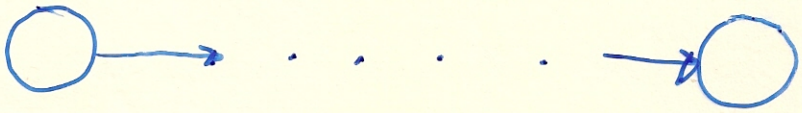
προσδιοριστικοί

Καθορίζει ένα μοναδικό μονοπάτι για το μήνυμα, βασισμένο στον πηγαίο κόμβο και στον κόμβο προορισμού.

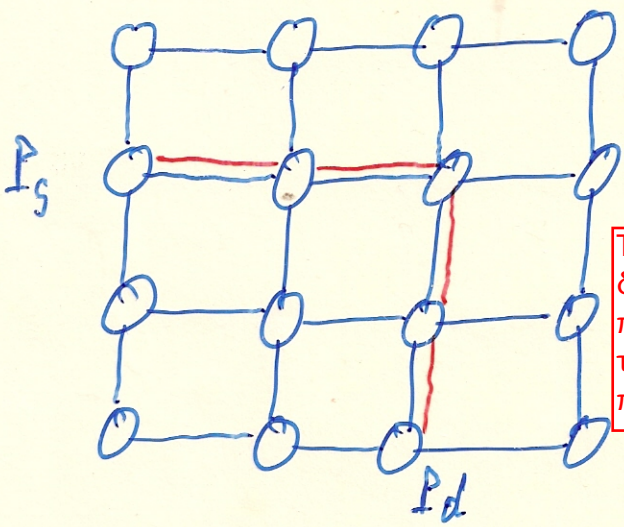
προσαρμοστικοί

Παίρνουν πληροφορία κοιτάζοντας την τρέχουσα κατάσταση του δικτύου για να καθορίσουν το μονοπάτι του μηνύματος.

πηγή (source) → προορισμός (destination)



XY-δρομολόγηση σε mesh



$\sqrt{p} \times \sqrt{p}$

$2(\sqrt{p}-1)$

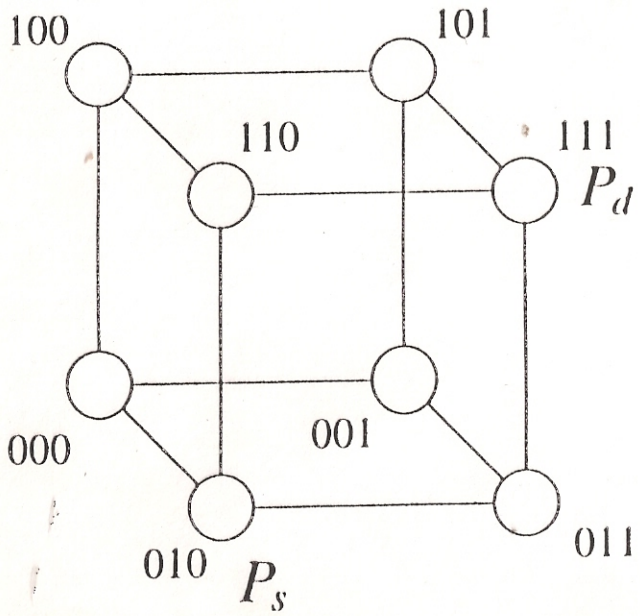
Το μήνυμα αποστέλλεται πρώτα στην X διάσταση μέχρι να φτάσει τη στήλη του επεξ. προορισμού και στη συνέχεια πάει κατά μήκος της διάστασης Y μέχρι να φτάσει στον προορισμό του.

E-cube δρομολόγηση

δηλ. XY- δρομολόγηση σε υπερκύβο.

Δρομολόγηση μηνύματος από τον επεξεργαστή P_s (010) στον επεξεργαστή P_d (111) σ' ένα 3-D υπερκύβο χρησιμοποιώντας E-cube δρομολόγηση.

Σ' έναν E-cube αλγόριθμο, ο επεξεργαστής P_s υπολογίζει το P_s XOR P_d και στέλνει το μήνυμα κατά μήκος της διάστασης k, όπου k είναι η θέση του λιγότερου σημαντικού μη μηδενικού bit στο P_s XOR P_d . Σε κάθε ενδιάμεσο βήμα, ο επεξεργαστής P_i, ο οποίος λαμβάνει ένα μήνυμα, υπολογίζει το P_s XOR P_d και προωθεί το μήνυμα προς τη διάσταση που αντιστοιχεί στο λιγότερο σημαντικό μη μηδενικό bit. Αυτή η διαδικασία συνεχίζεται μέχρι το μήνυμα να φτάσει στον προορισμό του.



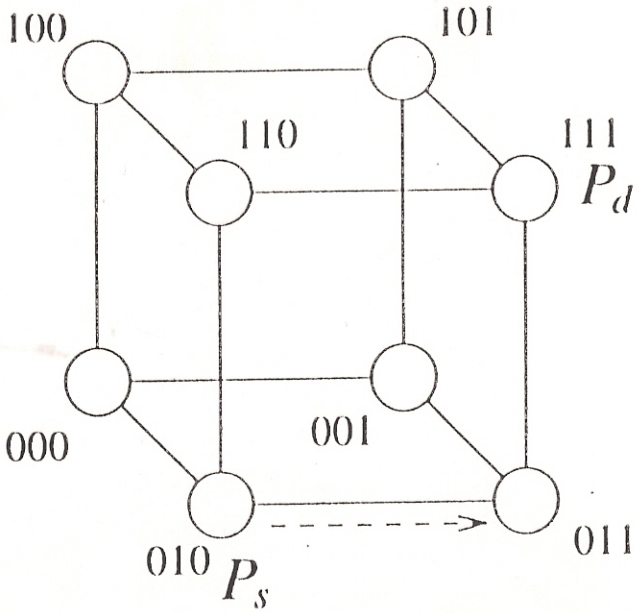
010 ⊕ 110 = 100

010 ⊕ 111 = 101

010 ⊕ 011 = 001

010 ⊕ 000 = 010

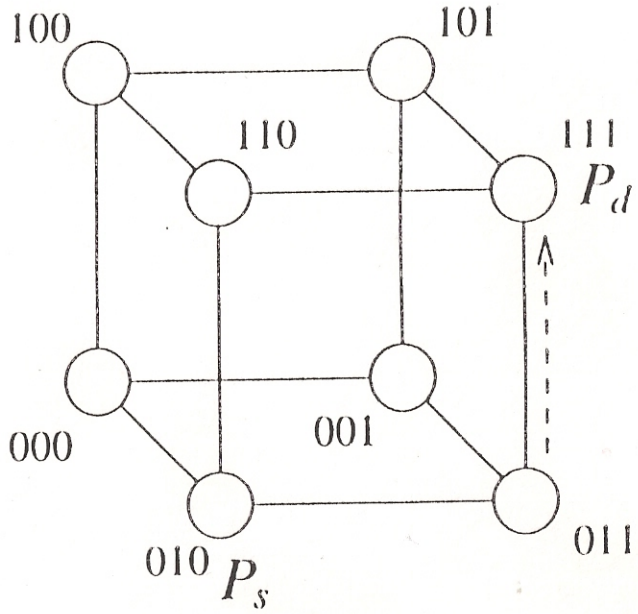
010 ⊕ 110 = 100



Step 1 (010 → 011)

010 ⊕ 011 = 001

011 ⊕ 111 = 100



Step 2 (011 → 111)

011 ⊕ 111 =

XOR
0 XOR 0=0
0 XOR 1=1
1 XOR 0=1
1 XOR 1=0

Κόστος Επικοινωνίας

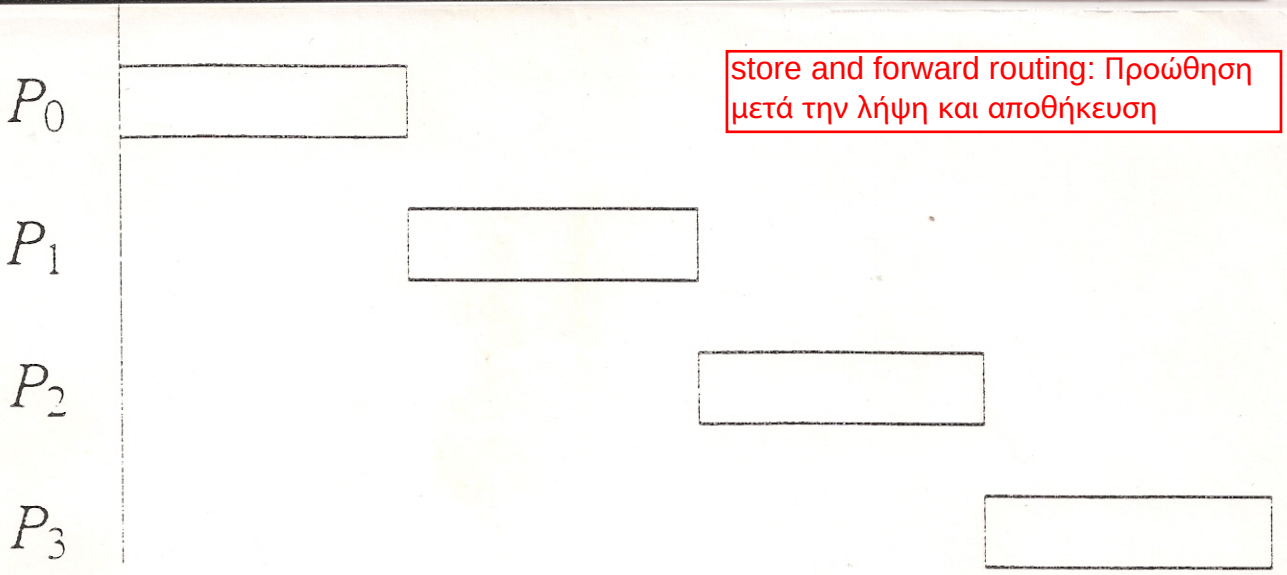
- καθυστέρηση επικοινωνίας (latency)
  Ο απαιτούμενος χρόνος για την επικοινωνία ενός μηνύματος μεταξύ δύο επεξεργαστών
- χρόνος ξεκινήματος (startup time) (t_s)
  Ο απαιτούμενος χρόνος για τη μεταχείριση του μηνύματος (προετοιμασία).
- per-hop χρόνος (t_h)
  Ο απαιτούμενος χρόνος για να φθάσει η επικεφαλίδα του μηνύματος στον προορισμό της
- χρόνος ανά word (t_w)

$$t_w = 1/r$$

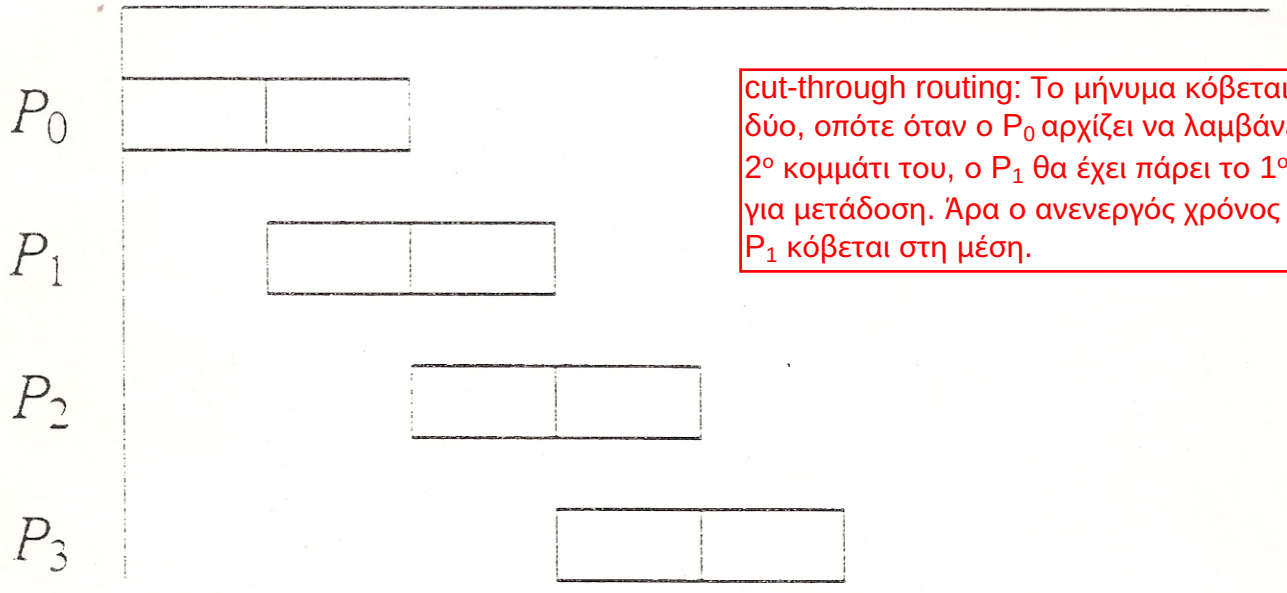
r = αριθμός words ανά δευτερόλεπτο

καθυστέρηση επικοινωνίας

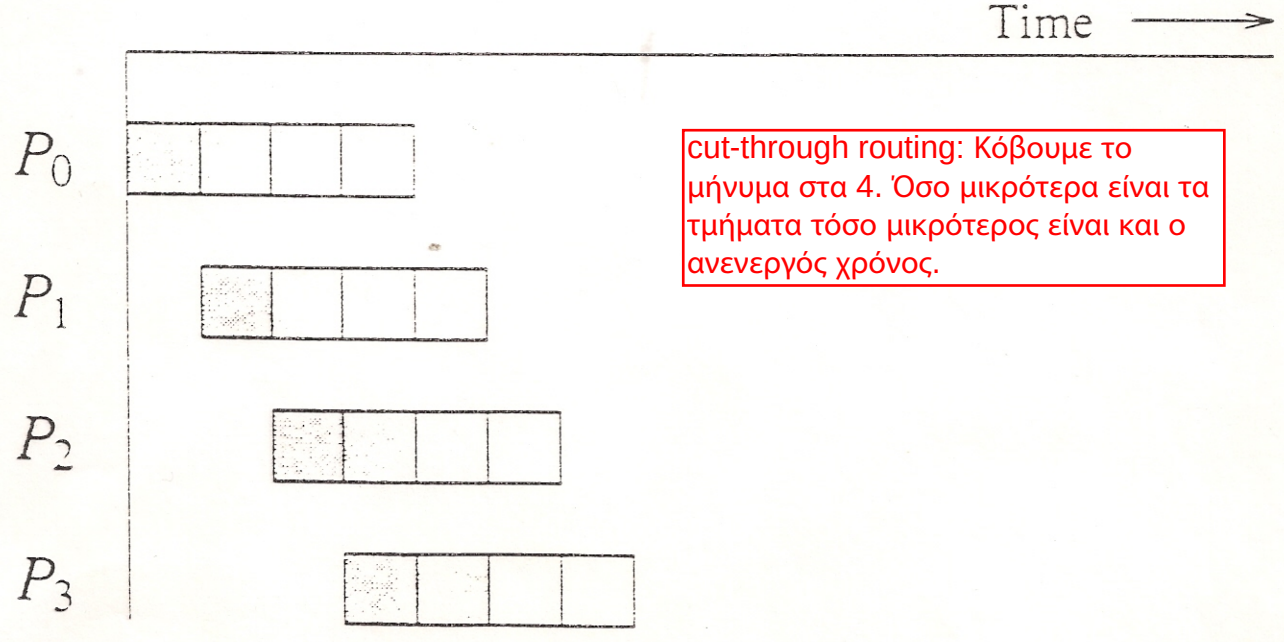
- τοπολογία δικτύου
- τεχνικές μεταφοράς μηνυμάτων (routing techniques)
  - store and forward routing
  - cut-through routing



store and forward routing: Προώθηση μετά την λήψη και αποθήκευση



cut-through routing: Το μήνυμα κόβεται στα δύο, οπότε όταν ο P_0 αρχίζει να λαμβάνει το 2ο κομμάτι του, ο P_1 θα έχει πάρει το 1ο του για μετάδοση. Άρα ο ανενεργός χρόνος του P_1 κόβεται στη μέση.



cut-through routing: Κόβουμε το μήνυμα στα 4. Όσο μικρότερα είναι τα τμήματα τόσο μικρότερος είναι και ο ανενεργός χρόνος.

Υπολογισμός κόστους επικοινωνίας

$$t_{comm} = t_s + (m t_w + t_h)\, l$$

m = πλήθος words, l = πλήθος συνδέσμων

mt_w=χρόνος μυνήματος
t_h=χρόνος επικεφαλίδας->πολύ μικρό

$$t_{comm} = t_s + m t_w l \quad \text{(store and forward)}$$
$$O(ml)$$

l=1->O(m)

$$t_{comm} = t_s + l t_h + m t_w \quad \text{cut through}$$
$$O(l+m)$$

$l = 1$ τοπικότητα

ΒΑΣΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΕΠΙΚΟΙΝΩΝΙΑΣ

- Οι σύνδεσμοι επικοινωνίας είναι διπλής κατεύθυνσης
- Ο κάθε επεξεργαστής μπορεί να στείλει ένα μήνυμα σε ένα μόνο από τους συνδέσμους του κάθε φορά
- Θμοια μπορεί να λάβει ένα μήνυμα από ένα σύνδεσμο κάθε φορά
- Μπορεί να λάβει ένα μήνυμα ενώ στέλνει ένα άλλο την ίδια στιγμή στον ίδιο ή διαφορετικό σύνδεσμο.

SF δρομολόγηση

store and forward

Χρόνοι για την μεταφορά ενός απλού μυνήματος

- $t_s + t_w \, m \, [P/2]$ δακτύλιο
- $t_s + 2 t_w \, m \, [\sqrt{P}/2]$ δίκτυο
- $t_s + t_w \, m \log P$ υπερκύβο

CT δρομολόγηση

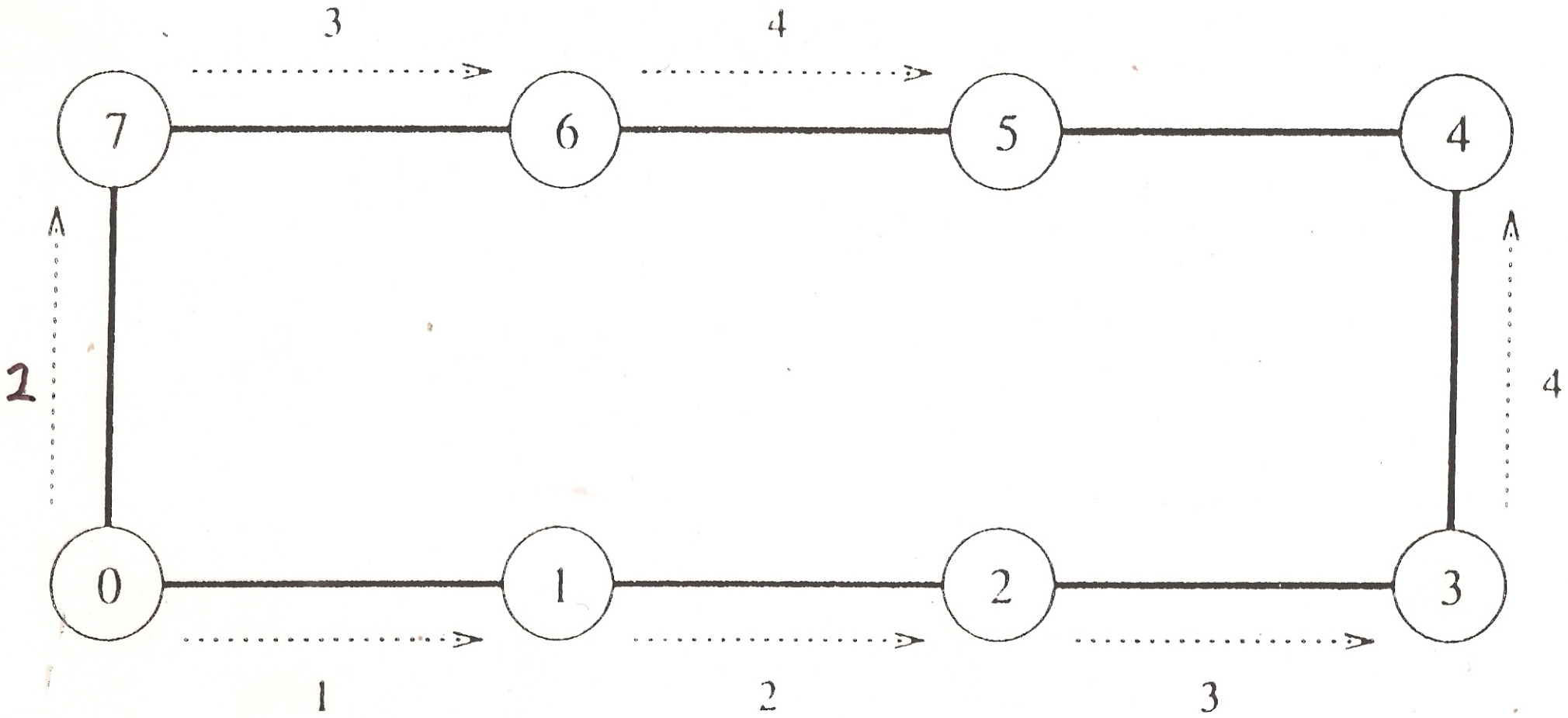
cut-through: Το μύνημα μπορεί να σταλεί απ' ευθείας από την πηγή στον προορισμό, \ell συνδέσμους μακριά σε χρόνο

$t_s + m t_w + t_h \ell$

Ενός προς όλους Μετάδοση

Δακτύλιος SF

Δακτύλιος

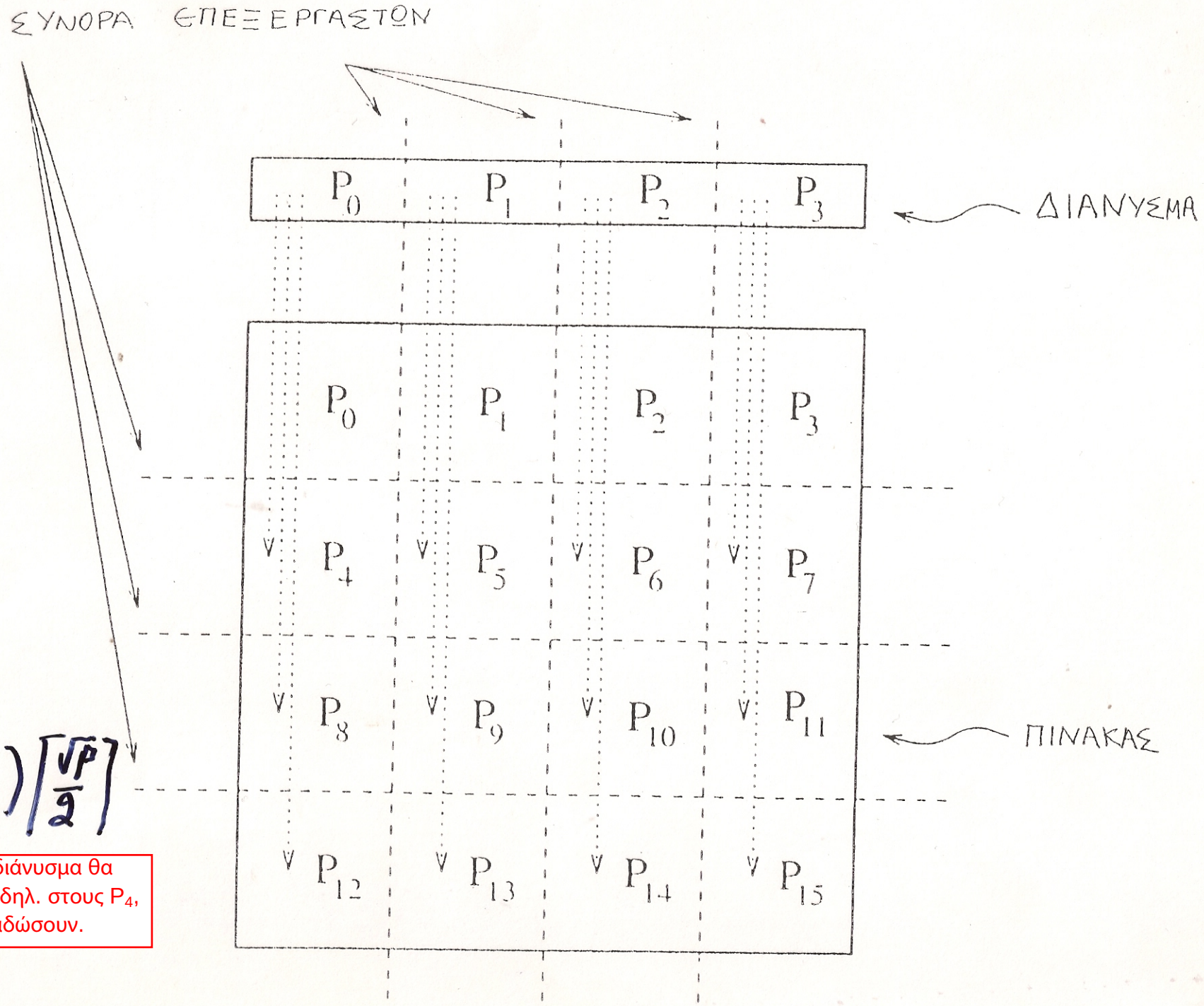


Ο επεξ. 0 στέλνει στον 1
Ο επεξ. 1 στέλνει στον 2 και ο 0 στον 7
Ο επεξ. 2 στέλνει στον 3 και ο 7 στον 6
Ο επεξ. 3 στέλνει στον 4 και ο 6 στον 5
Η διεργασία αυτή συνεχίζεται μέχρι όλοι οι επεξ. να έχουν ένα αντίγραφο του μηνύματος.

$$T = t_s + t_w m \lceil p/2 \rceil$$

Κάθε μια από τις [p/2] επικοινωνίες με κοντινό γείτονα παίρνει $t_s+t_w m$ χρόνο

Ένας προς όλους μετάδοση για τον πολλαπλασιασμό ενός 4x4πίνακα με ένα 4x1 διάνυσμα.

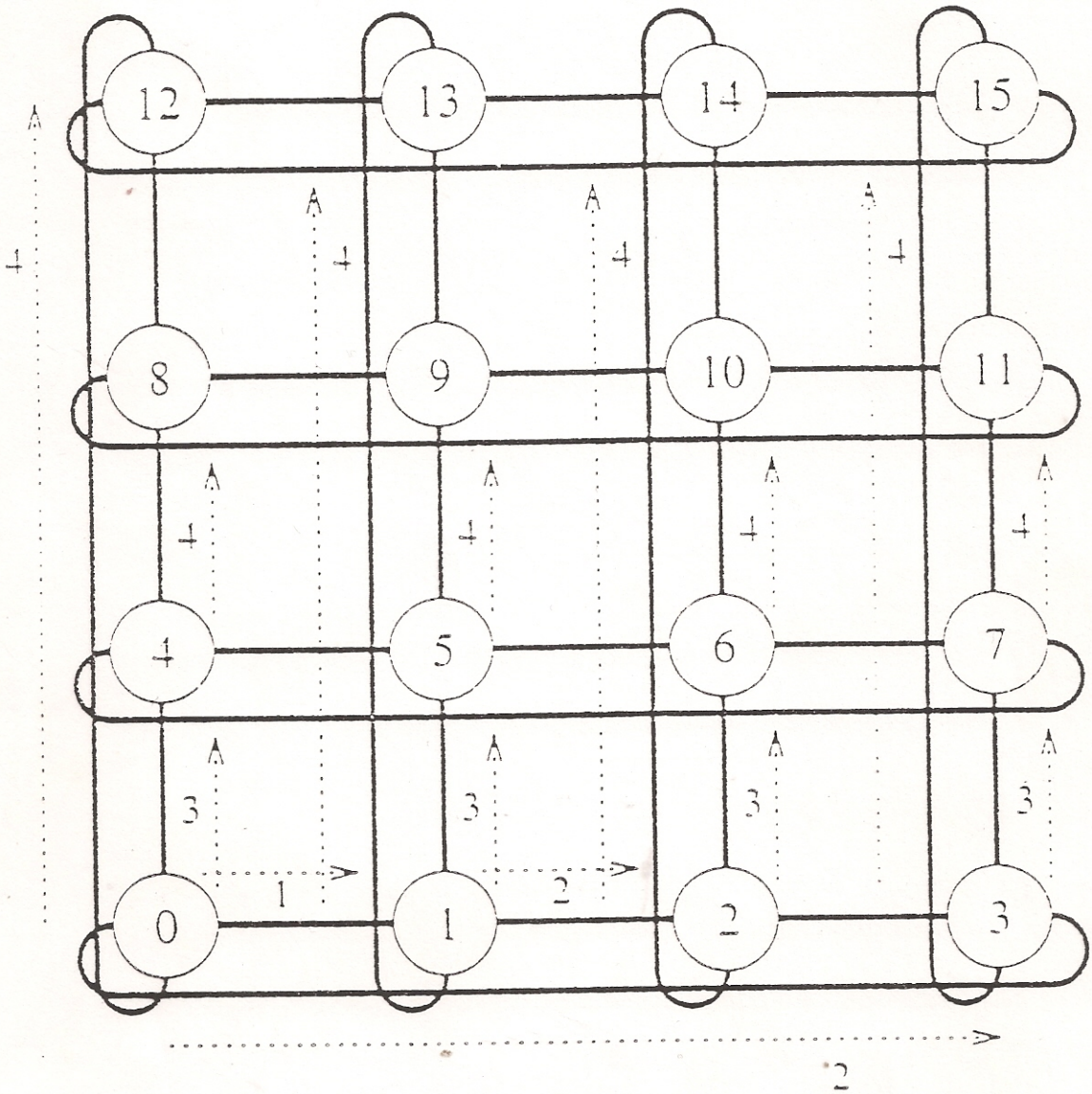


κόστος επικοινωνίας

$$T = a\left(t_s + m t_w\right)\left\lceil \frac{\sqrt{p}}{2} \right\rceil$$

δηλ. αν ο P_0 έχει αρχικά το διάνυσμα θα στείλει το μύνημα σε όλους, δηλ. στους P_4, P_8, P_{12} και όλοι μαζί θα μεταδώσουν.

Ένας προς όλους μετάδοση σε πλέγμα 16 επεξ. με store and forward μετάδοση



$$T = 2(t_s + t_w m)\left\lceil \sqrt{p}/2 \right\rceil$$

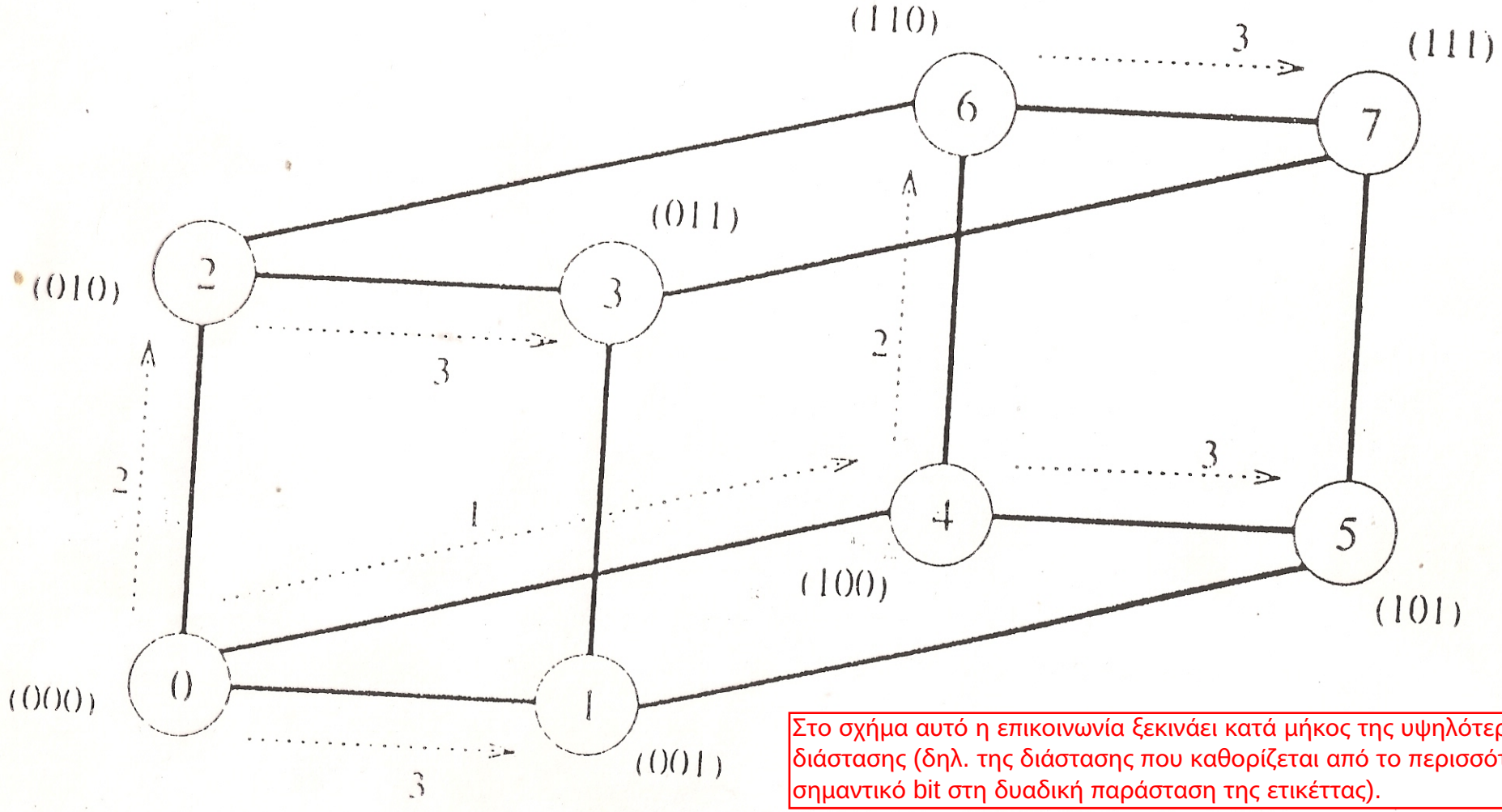
σε αναδιπλούμενο πλέγμα

$$T = 3(t_s + t_w m)\left\lceil \frac{p^{1/3}}{2} \right\rceil$$ 3D mesh

Ένας προς όλους μετάδοση σε 3-D υπερκύβο

$$T = (t_s + t_w m) \log p$$

Κάθε ένα από τα logp βήματα θέλει $t_s+t_w m$ χρόνο για τη μεταφορά ενός απλού μηνύματος σε κάθε διάσταση.



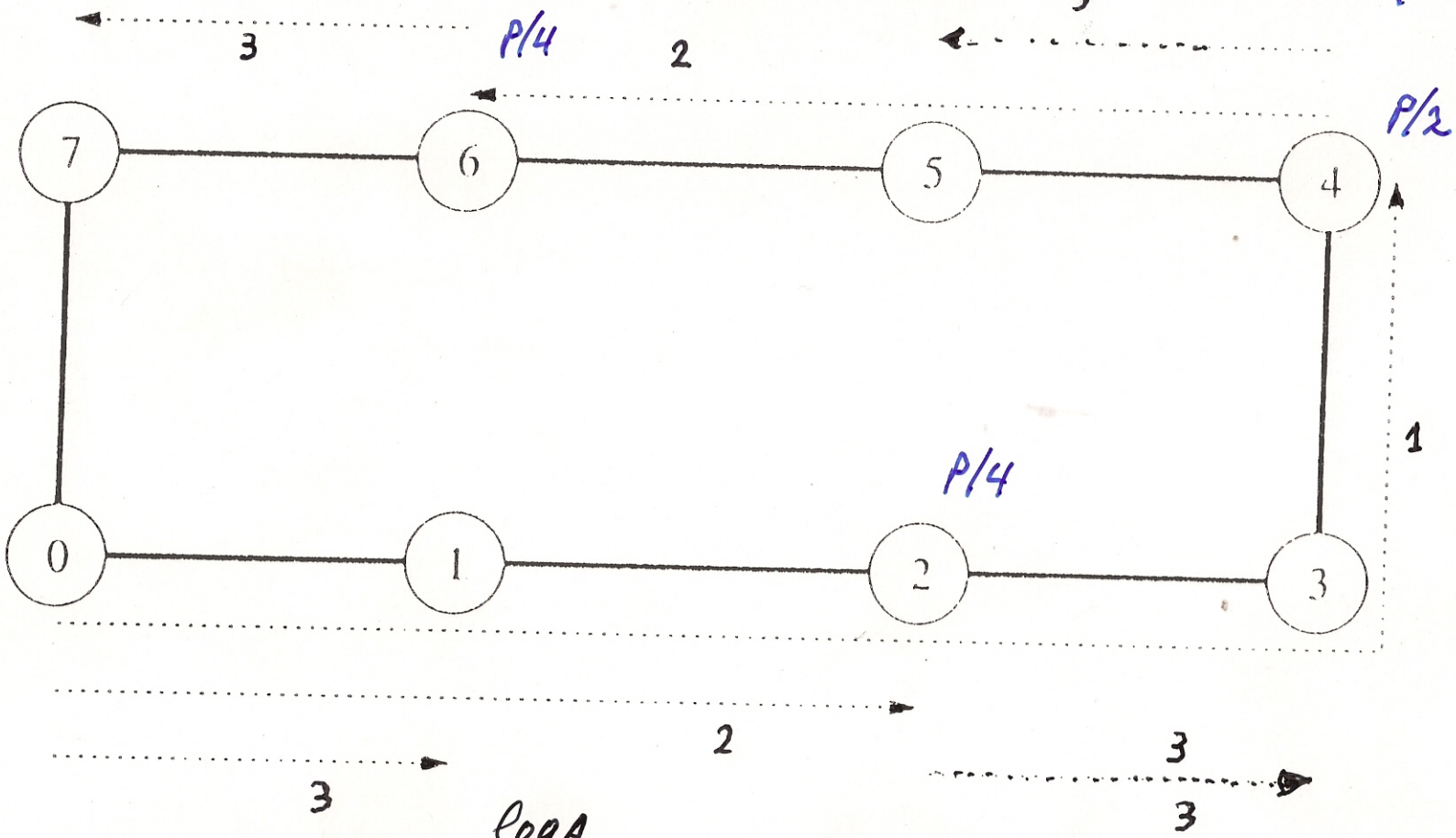
Στο σχήμα αυτό η επικοινωνία ξεκινάει κατά μήκος της υψηλότερης διάστασης (δηλ. της διάστασης που καθορίζεται από το περισσότερο σημαντικό bit στη δυαδική παράσταση της ετικέττας).

```
1.  procedure ONE_TO_ALL_BC(d, my_id, X)
2.  begin
3.      mask := 2^d − 1;          /* Set all d bits of mask to 1 */
4.      for i := d − 1 downto 0 do   /* Outer loop */
5.      begin
6.          mask := mask XOR 2^i;  /* Set bit i of mask to 0 */
7.          if (my_id AND mask) = 0 then
            /* If the lower i bits of my_id are 0 */
8.              if (my_id AND 2^i) = 0 then
9.              begin
10.                 msg_destination := my_id XOR 2^i;
11.                 send X to msg_destination;
12.             endif
13.             else
14.             begin
15.                 msg_source := my_id XOR 2^i;
16.                 receive X from msg_source;
17.             endelse;
18.     endfor;
19. end ONE_TO_ALL_BC
```

```
1.  procedure GENERAL_ONE_TO_ALL_BC(d, my_id, source, X)
2.  begin
3.      my_virtual_id := my_id XOR source;
4.      mask := 2^d − 1;
5.      for i := d − 1 downto 0 do      /* Outer loop */
6.      begin
7.          mask := mask XOR 2^i;   /* Set bit i of mask to 0 */
8.          if (my_virtual_id AND mask) = 0 then
9.              if (my_virtual_id AND 2^i) = 0 then
10.             begin
11.                 virtual_dest := my_virtual_id XOR 2^i;
12.                 send X to (virtual_dest XOR source); /* Convert virtual_dest
                                    to the label of the physical destination */
13.             endif
14.             else
15.             begin
16.                 virtual_source := my_virtual_id XOR 2^i;
17.                 receive X from (virtual_source XOR source);
    /* Convert virtual_source to the label of the physical source */
18.             endelse;
19.     endfor;
20. end GENERAL_ONE_TO_ALL_BC
```

```
1.  procedure SINGLE_NODE_ACC(d, my_id, m, X, sum)
2.  begin
3.      for j := 0 to m − 1 do sum[j] := X[j];
4.      mask := 0;
5.      for i := 0 to d − 1 do
6.      begin /* Select processors whose lower i bits are 0 */
7.          if (my_id AND mask) = 0 then
8.              if (my_id AND 2^i) ≠ 0 then
9.              begin
10.                 msg_destination := my_id XOR 2^i;
11.                 send sum to msg_destination;
12.             endif
13.             else
14.             begin
15.                 msg_source := my_id XOR 2^i;
16.                 receive X from msg_source;
17.                 for j := 0 to m − 1 do
18.                     sum[j] :=sum[j] + X[j];
19.             endelse;
20.         mask := mask XOR 2^i;   /* Set bit i of mask to 1 */
21.     endfor;
22. end SINGLE_NODE_ACC
```

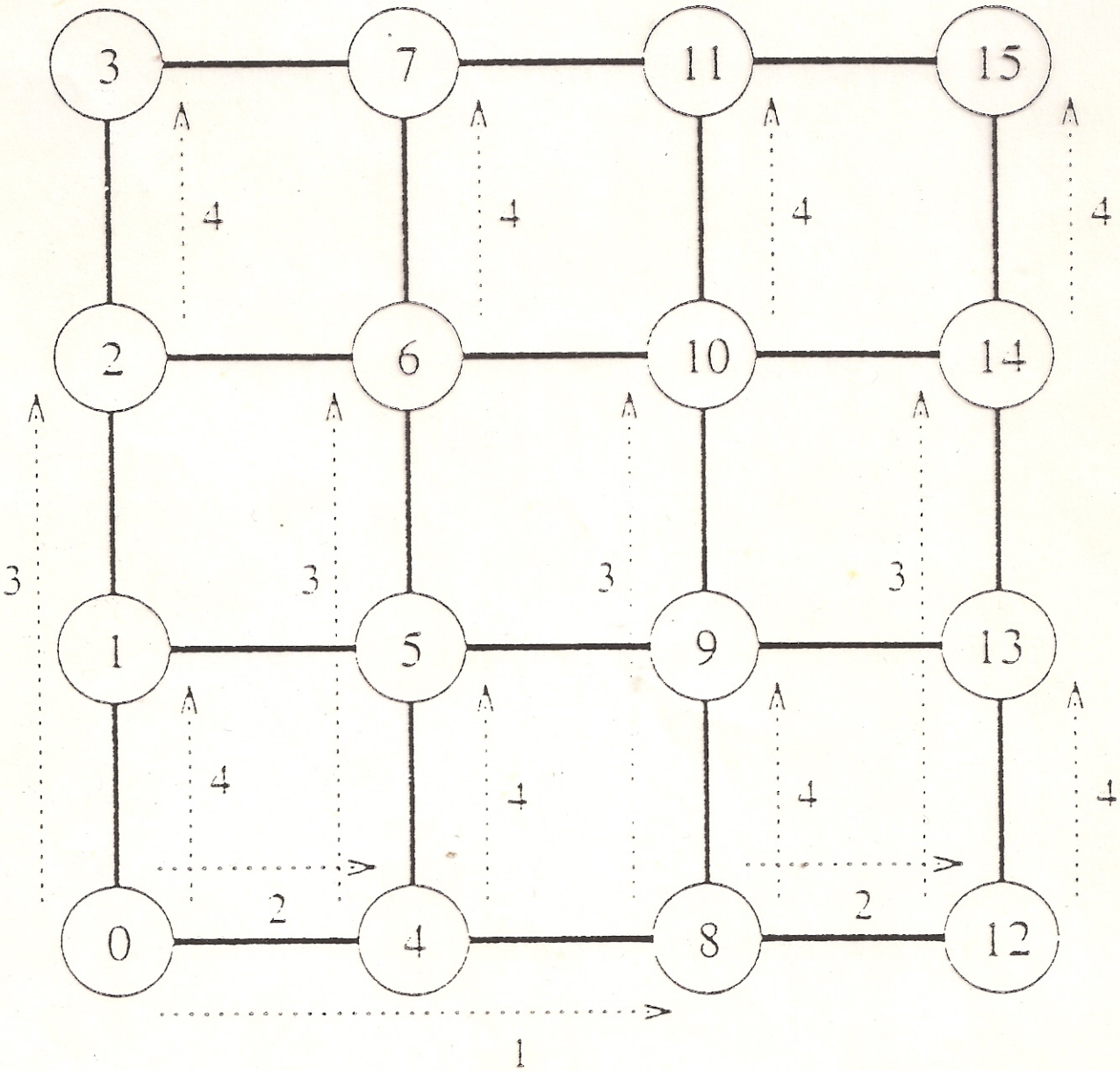
- Ενας σε όλους μετάδοση με CT δρομολόγηση
- Στο i-οστο βήμα κάθε επεξεργαστής που έχει τα δεδομένα τα στέλνει σε ένα επεξεργαστή απόσταση $P/2^i$



$$T_{ενας\text{-}προς\text{-}όλους} = \sum_{i=1}^{\log P} (t_s + t_w m + t_h P/2^i) = (t_s + t_w m)\log P + t_h(P-1)$$

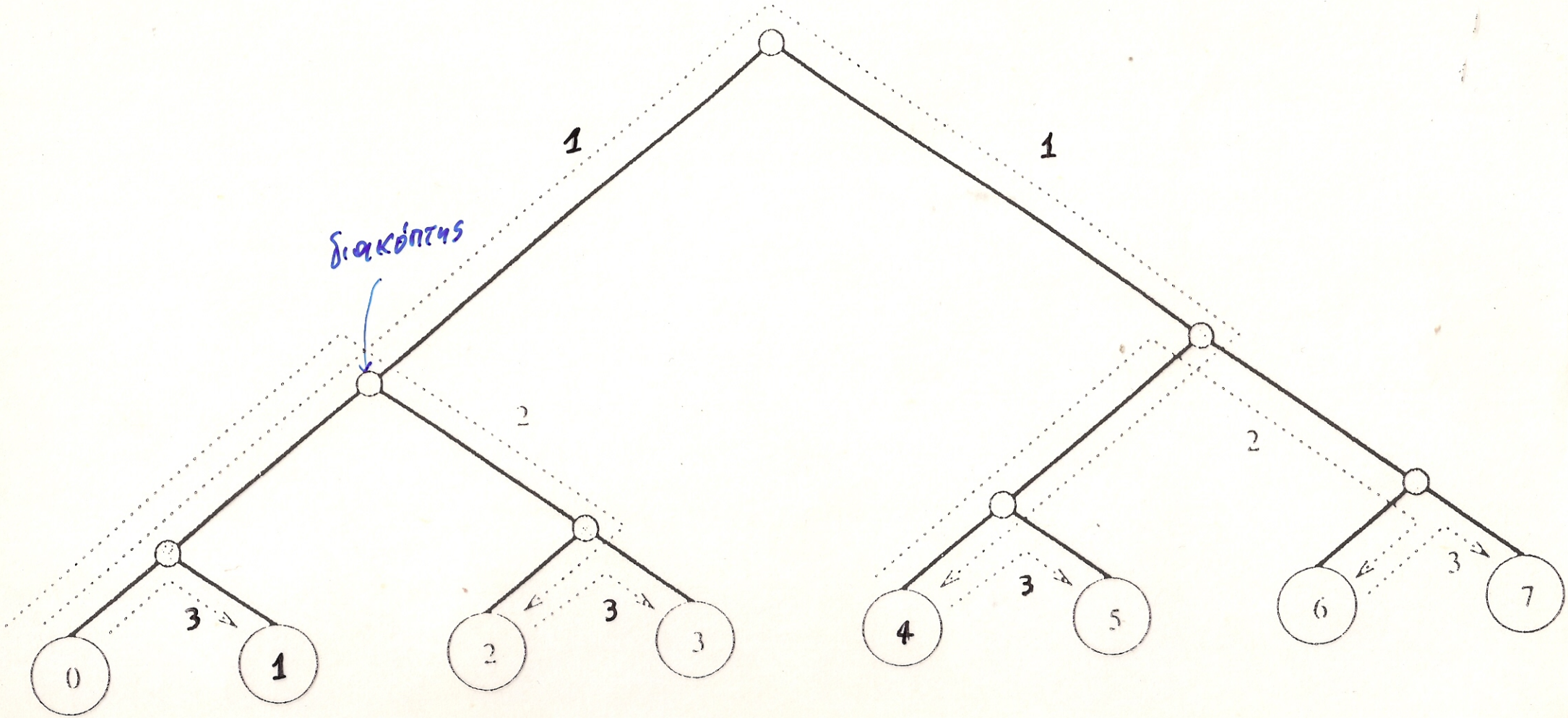
- Μετάδοση ένας προς όλους με CT δρομολόγηση
- $t_s + m t_w \log\sqrt{p} + (\sqrt{p} - 1) t_h$ κάθε φάση

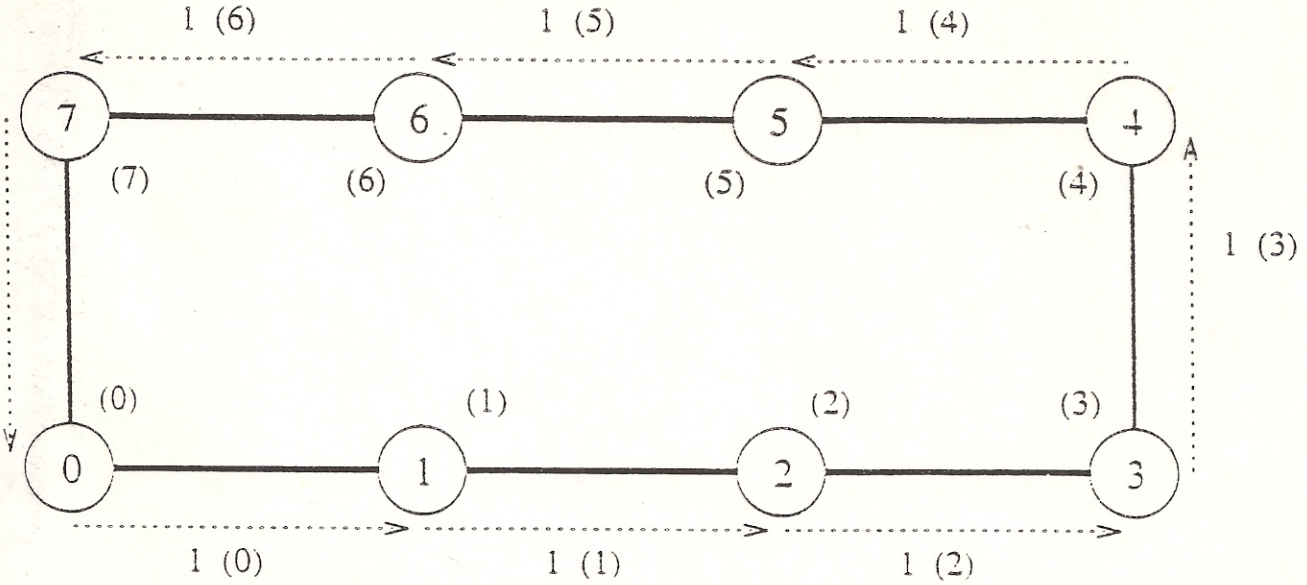
$$T_{ένας\text{-}προς\text{-}όλους} = (t_s + m t_w) \log p + 2(\sqrt{p} - 1) t_h$$

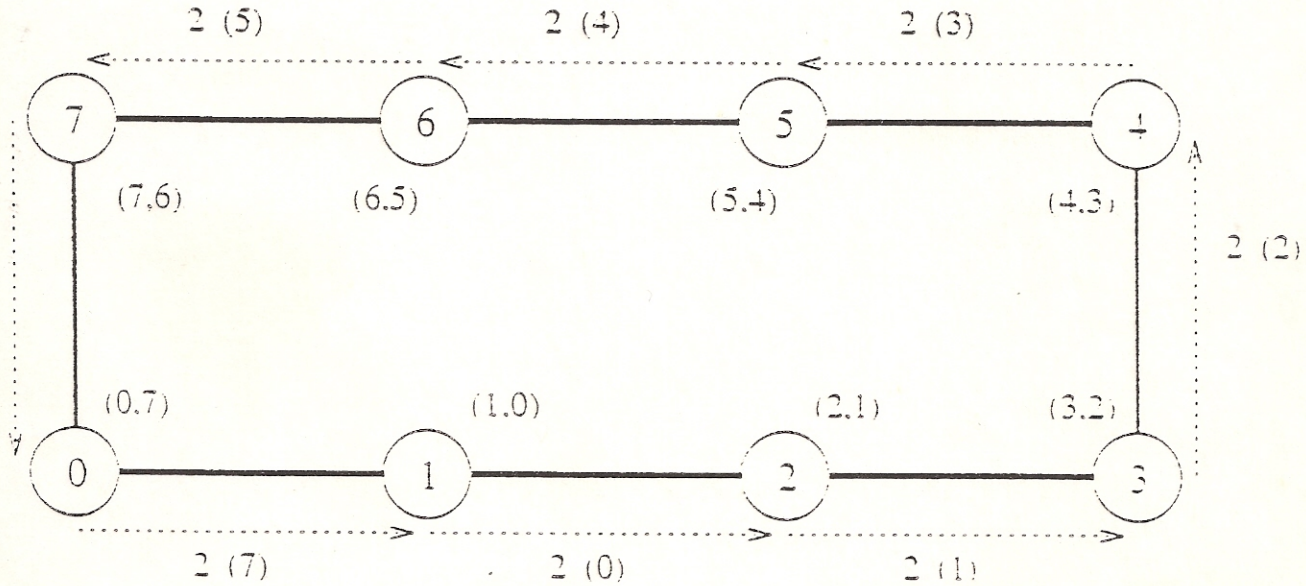


$$T_{\text{ένας-προς-όλους}} = (t_s + m t_w + (\log p + 1)) \log p$$

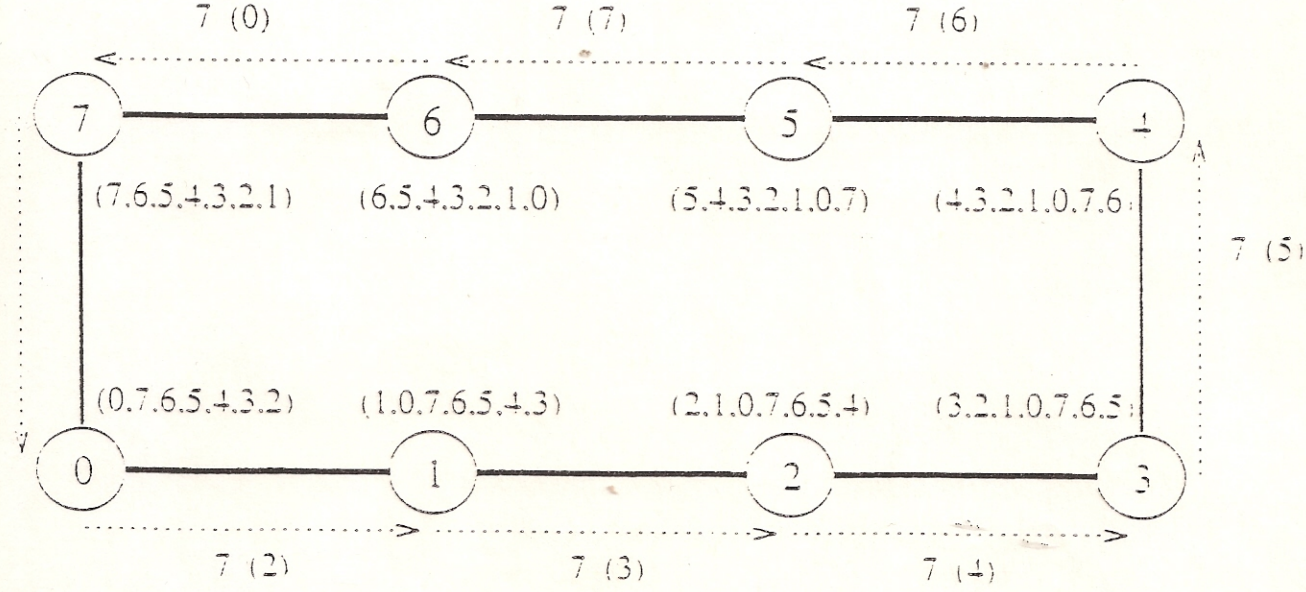
Ισορροπημένο Δυαδικό Δέντρο

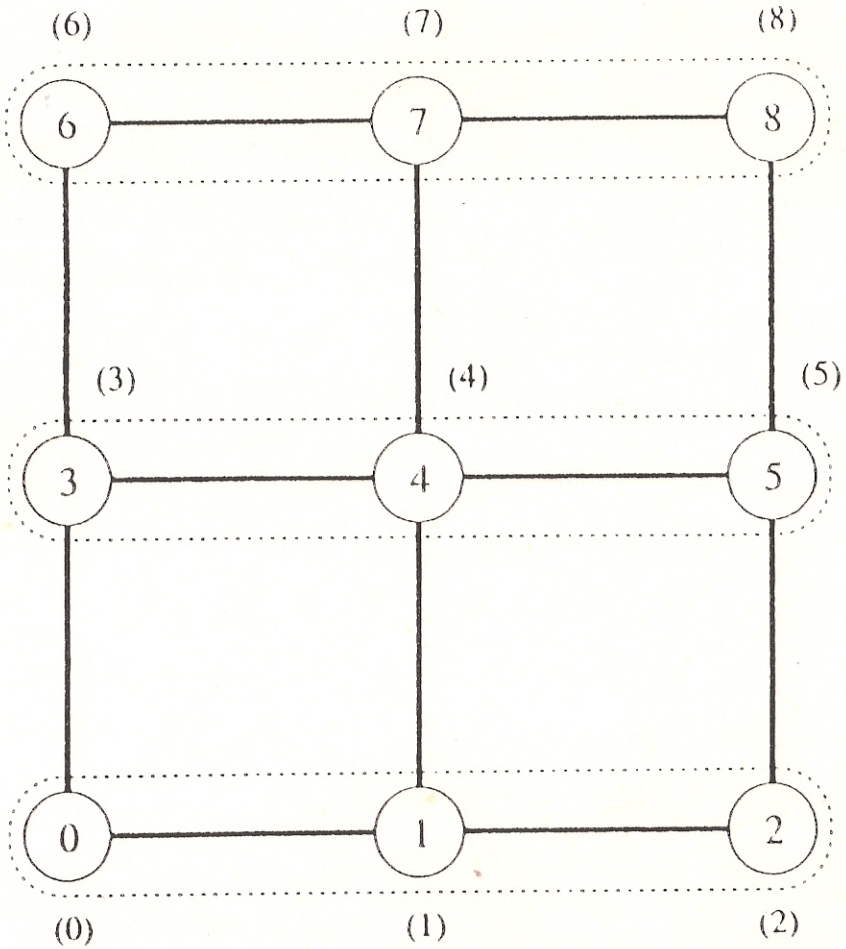


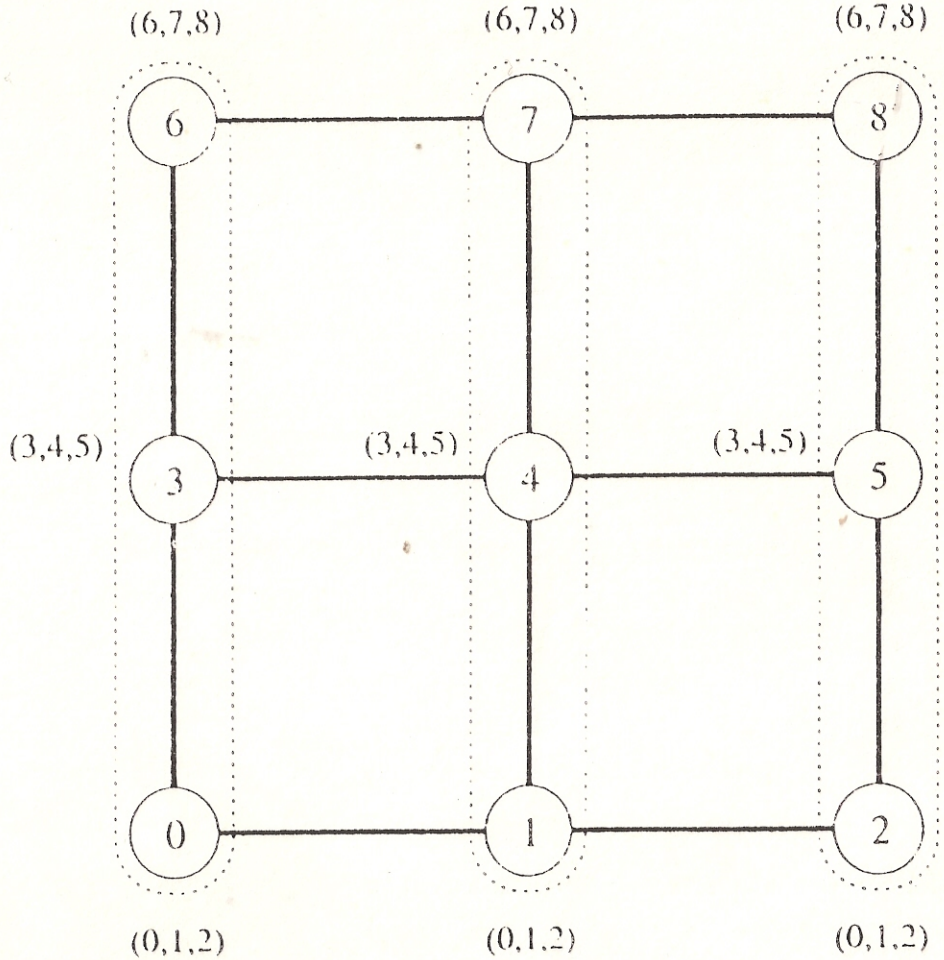

1 (6)
1 (5)
1 (4)
7
6
5
4
(7)
(6)
(5)
(4)
1 (3)
(0)
(1)
(2)
(3)
0
1
2
3
1 (0)
1 (1)
1 (2)


2 (5)
2 (4)
2 (3)
7
6
5
4
(7,6)
(6,5)
(5,4)
(4,3)
2 (2)
(0,7)
(1,0)
(2,1)
(3,2)
0
1
2
3
2 (7)
2 (0)
2 (1)

•
•
•


7 (0)
7 (7)
7 (6)
7
6
5
4
(7.6.5.4.3.2.1)
(6.5.4.3.2.1.0)
(5.4.3.2.1.0.7)
(4.3.2.1.0.7.6)
7 (5)
(0.7.6.5.4.3.2)
(1.0.7.6.5.4.3)
(2.1.0.7.6.5.4)
(3.2.1.0.7.6.5)
0
1
2
3
7 (2)
7 (3)
7 (4)


(6)
(7)
(8)
6
7
8
(3)
(4)
(5)
3
4
5
0
1
2
(0)
(1)
(2)


(6,7,8)
(6,7,8)
(6,7,8)
6
7
8
(3,4,5)
(3,4,5)
(3,4,5)
3
4
5
0
1
2
(0,1,2)
(0,1,2)
(0,1,2)

```
1.  procedure ALL_TO_ALL_BC_RING(my_id, my_msg, p, result)
2.  begin
3.      left := (my_id − 1) mod p;
4.      right := (my_id + 1) mod p;
5.      result := my_msg;
6.      msg := result;
7.      for i := 1 to p − 1 do
8.      begin
9.          send msg to right;
10.         receive msg from left;
11.         result := result ∪ msg;
12.     endfor;
13. end ALL_TO_ALL_BC_RING
```

```
1.    procedure ALL_TO_ALL_BC_MESH(my_id, my_msg, p, result)
2.    begin

/* Communication along rows */
3.        left := (my_id − 1) mod p;
4.        right := (my_id + 1) mod p;
5.        result := my_msg;
6.        msg := result;
7.        for i := 1 to √p − 1 do
8.        begin
9.            send msg to right;
10.           receive msg from left;
11.           result := result ∪ msg;
12.       endfor;

/* Communication along columns */
13.       up := (my_id − √p) mod p;
14.       down := (my_id + √p) mod p;
15.       msg := result;
16.       for i := 1 to √p − 1 do
17.       begin
18.           send msg to down;
19.           receive msg from up;
20.           result := result ∪ msg;
21.       endfor;
22.   end ALL_TO_ALL_BC_MESH
```

MIMD

Διαμοιρασμένης Μνήμης (tightly coupled)

Κατανεμημένης Μνήμης (loosely coupled)

- Scheduling

Ανάλυση Παράλληλων Αλγορίθμων

$$T_p = T_{comp} + T_{comm}$$

Ταχύτητα και Αποδοτικότητα

$$S_p = \frac{T_1}{T_p}$$ ταχύτητα

T_1 : σειριακός χρόνος

T_p : παράλληλος χρόνος με p επεξεργαστές

$$E_p = \frac{S_p}{p}$$ αποδοτικότητα

Ταχύτητα S_p: Δείχνει το όφελος της επίλυσης ενός προβλήματος παράλληλα.
Σειριακός χρόνος T_1: Ο χρόνος που περνά μεταξύ της αρχής και του τέλους της εκτέλεσης σ' έναν σειριακό υπολογιστή.
Παράλληλος χρόνος T_p: Ο χρόνος που περνά από την στιγμή που αρχίζει ο παράλληλος υπολογισμός ως τη στιγμή που ο τελευταίος επεξεργαστής τελειώνει την εκτέλεση.
Αποδοτικότητα E_p: Είναι το μέτρο του ελάχιστου χρόνου που ο επεξεργαστής απασχολείται πραγματικά.

Είναι φανερό ότι $S_p \geq 1$. Επίσης ένα βήμα ενός παράλληλου αλγορίθμου χρησιμοποιώντας p επεξεργαστές χρειάζεται p βήματα το πολύ όταν υλοποιηθεί σειριακά. Άρα

$$T_1 \leq p T_p$$

Συνεπώς

$$1 \leq S_p \leq p \quad \text{και} \quad 0 \leq E_p \leq 1$$

Ο νόμος του Amdahl

Όσο και να αυξάνουμε τους επεξεργστές από ένα σημείο και μετά η επιτάχυνση παραμένει η ίδια.

T1: Ο χρόνος που δεν μπόρεσα να παραλληλοποιήσω

$T_1 = T_{seq} + T_{par}$, $T_{seq} = f T_1$

f: ποσοστό σειριακού χρόνου $T_{par} = (1-f) T_1$

Με p επεξεργαστές

$$T_p \geq T_{seq} + T_{par}/p,$$

$$S_p = \frac{T_1}{T_p} \leq \frac{1}{f + \frac{1-f}{p}} \leq \frac{1}{f}$$

δηλ. όσο και να αυξήσω τους επεξεργαστές δεν μπορώ να έχω $S_p >= 1/f$.

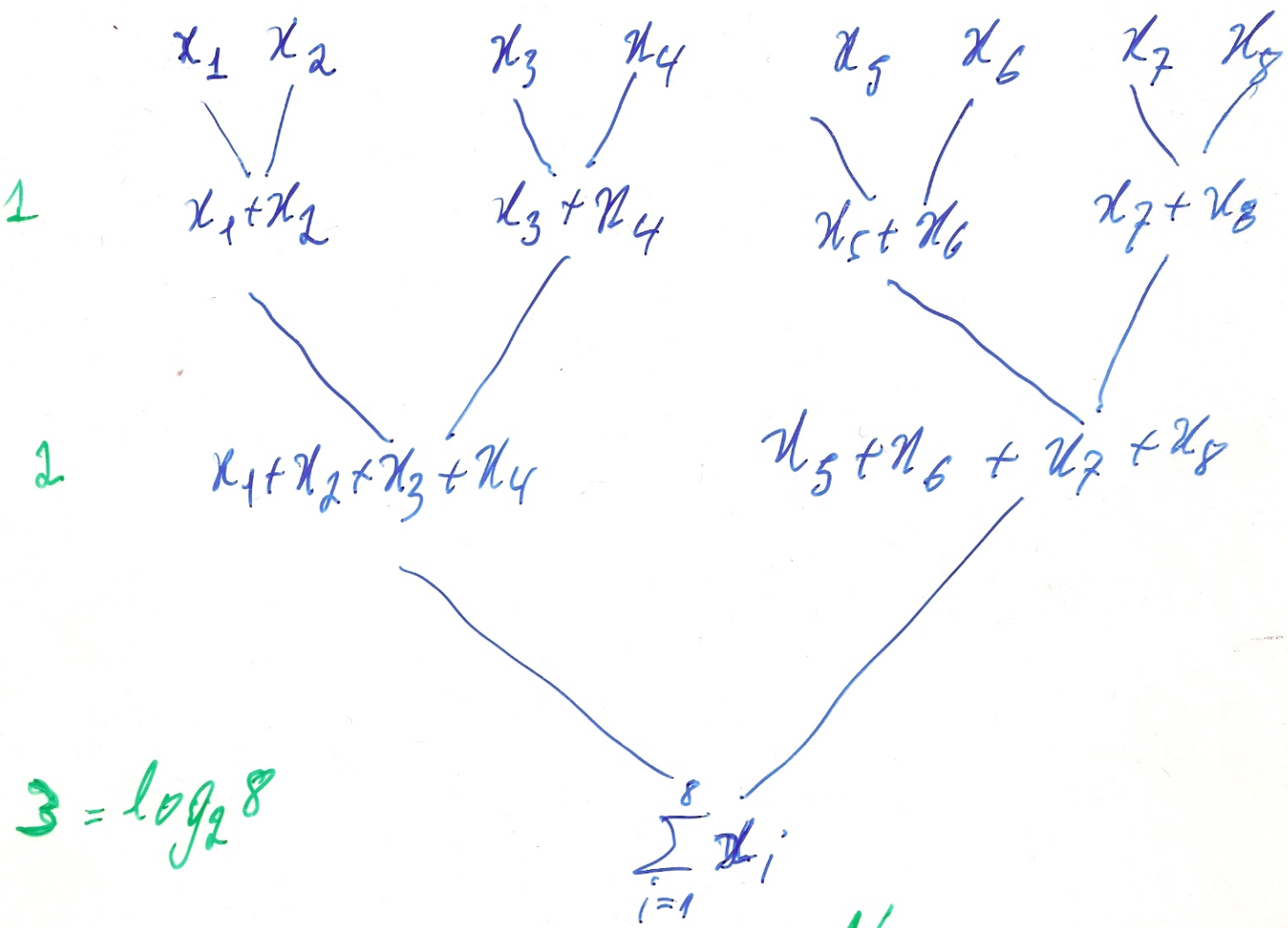
Αν $f = 10\%$ $\rightsquigarrow$ $S_p \leq 10$ για οποιοδήποτε p!

δηλαδή όσο και να αυξήσω τους επεξεργαστές δεν μπορώ να έχω Sp>=1/f

Παράδειγμα (SIMD) $\sum_{i=1}^{N} x_i$

Υπολογισμός του

Ο κάθε επεξεργαστής εκτελεί την ίδια εντολή ταυτόχρονα.



Αριθμός Επεξεργαστών $p = \frac{N}{2}$

CREW

$\log N$ βήματα

$$S_p = \frac{T_1}{T_p} = \frac{N-1}{\log N} \simeq \frac{N}{\log N}$$

N-1: οι προσθέσεις για N αριθμούς
logN: ύψος δέντρου

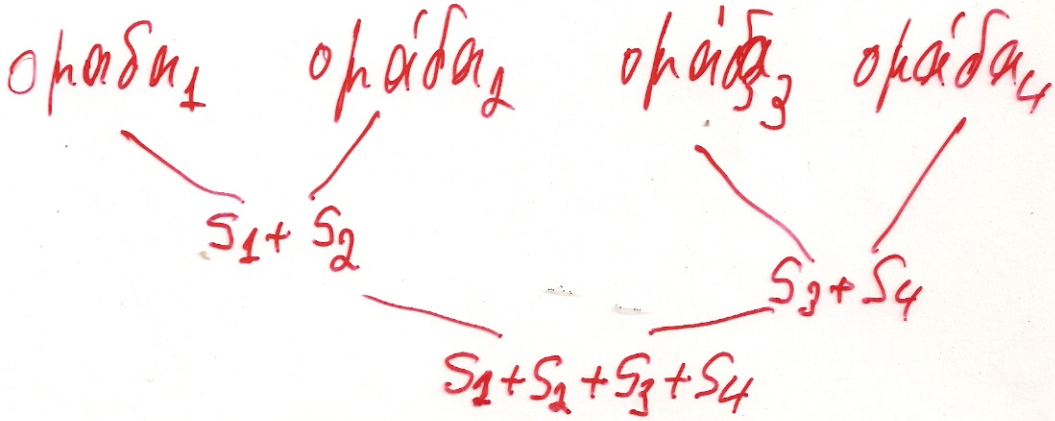
$$E_p \simeq \frac{1}{\log N} \to 0 \text{ για } N \to \infty$$

Απόδειξη: $E_p = S_p/p = S_p/(N/2) = (N/\log N)/(p/2) = 2/\log N$

Ανάλογος αλγόριθμος για προβλήματα
- εσωτερικό γινόμενο διανυσμάτων
- εύρεση μέγιστου - ελάχιστου αριθμών
κ.α

Παράδειγμα
Υπολογισμός του $\sum_{i=1}^{N} x_i$ με τη χρήση p επεξεργαστών $1 \leq p < N$.

1. Χωρισμός των N αριθμών σε p ομάδες. Κάθε ομάδα περιέχει το πολύ $\lceil N/p \rceil$ αριθμούς
2. Καταχώρηση μιας ομάδας σε κάθε ένα από τους p επεξεργαστές
3. Υπολογισμός του αθροίσματος των αριθμών σε κάθε ομάδα σειριακά σε $\lceil N/p \rceil - 1$ βήματα
4. Υπολογισμός του αθροίσματος των επιμέρους αθροισμάτων (σε μορφή δυαδικού δέντρου)



$$T_p = \left\lceil \frac{N}{p} \right\rceil - 1 + \lceil \log p \rceil$$

T_1

$$S_p = \frac{N-1}{\lceil N/p \rceil + \lceil \log p \rceil - 1}$$

Αν $N = L p \log p$

$$S_p = \frac{Lp}{L+1} \rightsquigarrow \lim_{L \to \infty} S_p = p$$ γραμμική επιτάχυνση!

Επίσης

$$E_p = \frac{L}{L+1} \rightsquigarrow \lim_{L \to \infty} E_p = 1$$ -> βέλτιστη περίπτωση

Παράδειγμα

Δίνονται τα $X = (x_1, x_2, \dots x_N)$, $Y = (y_1, y_2, \dots y_N)$.

Να υπολογιστεί το $\sum_{i=1}^{N} x_i y_i$

Έστω ότι το p διαιρεί το N ώστε

$$\sum_{i=1}^{N} x_i y_i = \sum_{i=1}^{N/p} x_i y_i + \sum_{i=N/p+1}^{2(N/p)} x_i y_i + \dots \sum_{i=(N/p)(k-1)+1}^{(N/p)k} x_i y_i + \dots \sum^{N} x_i y_i$$

Άρα ο k επεξεργαστής υπολογίζει

$$\sum_{i=(N/p)(k-1)+1}^{(N/p)k} x_i y_i, \quad k = 1, 2, \dots, p$$

Για τον ανωτέρω υπολογισμό απαιτούνται

$\frac{N}{p}$ πολ/σμοί

$\frac{N}{p}-1$ προσθέσεις

π.χ. Αν είχα
Σ_{i=1}^3 $x_iy_i=x_1y_1+x_2y_2+x_3y_3$ θα είχα 3 πολ/μους και 2 προσθέσεις, άρα για Σ_{i=1}^{N/p}x_iy_i έχουμε N/p πολ/μούς και N/p-1 προσθέσεις.

Τα μερικά αθροίσματα προστίθενται με τη μορφή του δυαδικού δέντρου σε $\lceil \log p \rceil$ βήματα. Άρα

$$T_p = 2\left(\frac{N}{p}-1\right)+\lceil \log p \rceil$$

και

$$S_p = \frac{2N-1}{2\left(\frac{N}{p}-1\right)+\lceil \log p \rceil}$$

Λάθος στον παρονομαστή και στο T_p. Είναι:
(2N/p-1)+[logp], όπου 2N/p-1=N/p+N/p-1

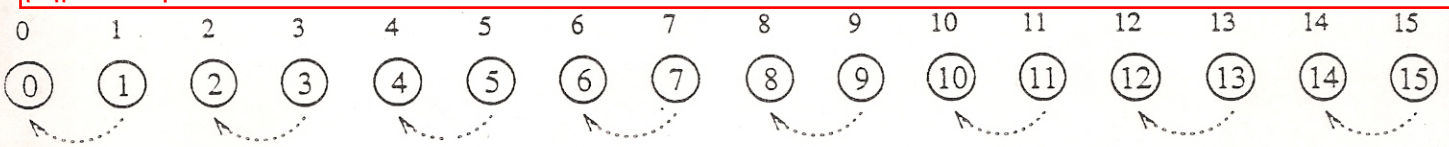
Αν $p=N$ $\quad S_p = \frac{2N-1}{\lceil \log N \rceil + 1}$

Αν $p=2^k$ και $N=Lp\log p$ τότε

$$S_p = \frac{2L}{2L+1}\,p \leadsto \lim_{L\to\infty} S_p = p \;!$$

Πρόσθεση n αριθμών σε ένα Υπερκύβο με n επεξεργαστές

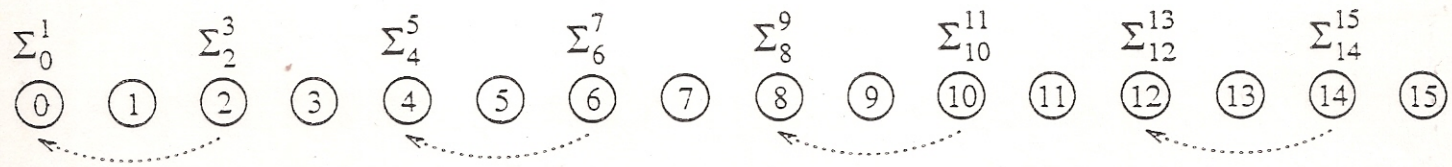
Αρχικά, σε κάθε επεξ. εκχωρείται ένας αριθμός και στο τέλος ένας από τους επεξ. αποθηκεύει το άθροισμα όλων των αριθμών. Έστω ότι το n είναι δύναμη του 2, άρα για εκτέλεση σε υπερκύβο ή σε SM επεξ. χρειαζόμαστε logn βήματα. p=n



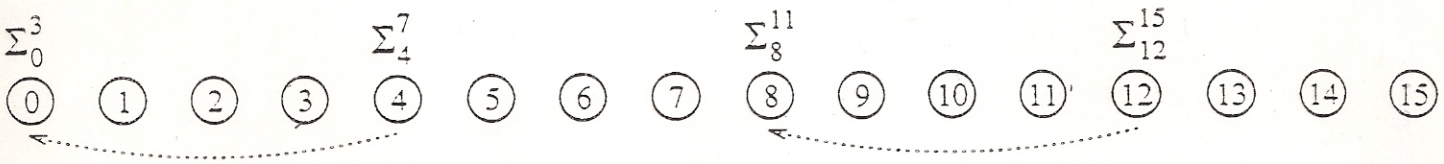
n=16

(a)

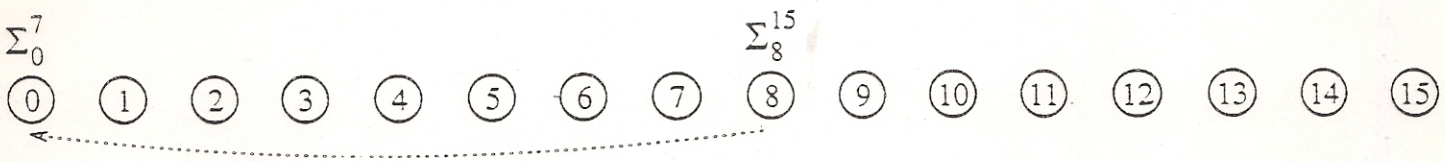
Κάθε βήμα αποτελείται από μια πρόσθεση και την επικοινωνία μιας λέξης. Οι επεξ. που επικοινωνούν μεταξύ τους είναι άμεσα συνδεδεμένοι σε έναν υπερκύβο, αφού οι ετικέττες τους διαφέρουν μόνο κατά 1 bit.



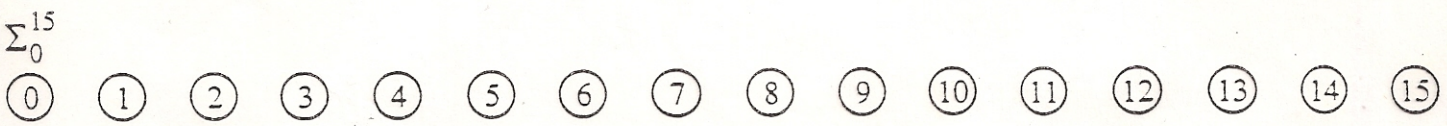
(b)



(c)



(d)



Σε απλό επεξ. -> Θ(n), δηλ. Sp ->

(e)

Η πρόσθεση και η επικοινωνία απαιτούν σταθερό ποσοστό χρόνου T_p

$T_p = \Theta(\log n)$, $S_p = \Theta\left(\frac{n}{\log n}\right)$

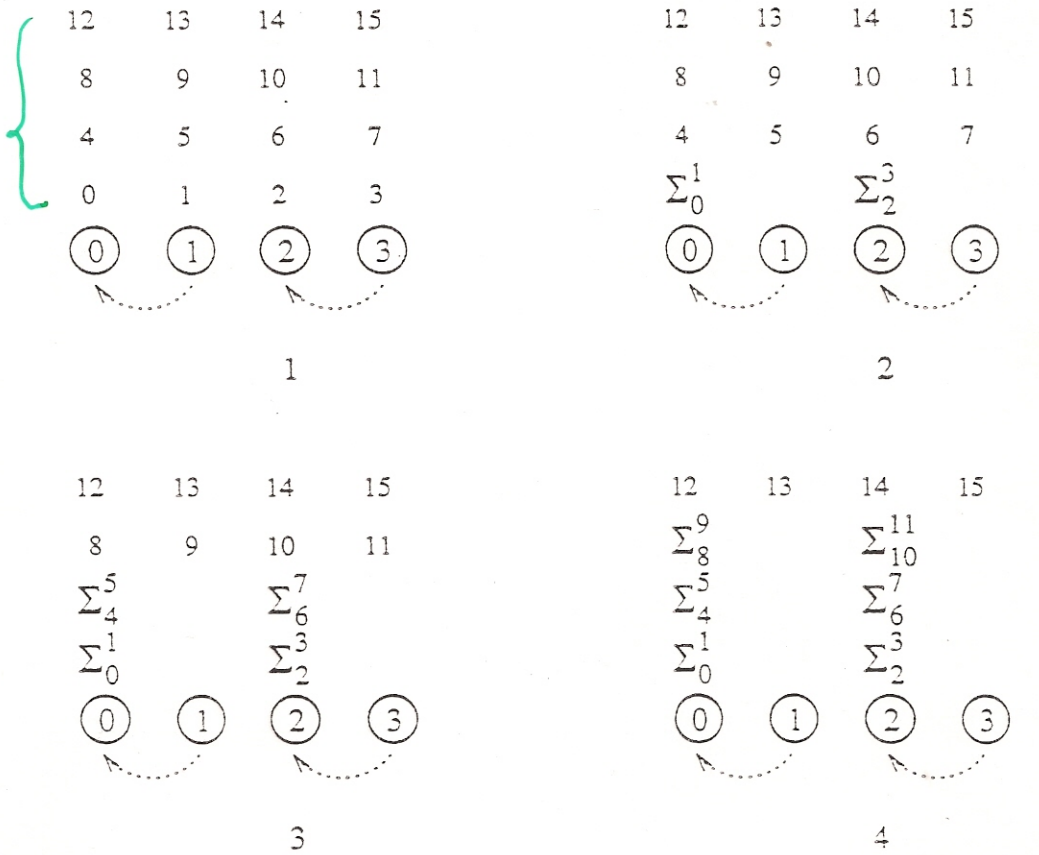
$E_p = \Theta\left(\frac{1}{\log n}\right)$ $c_p = \Theta(n \log n)$ not cost optimal

cost not opt. γιατί ο αντίστ. ακολουθ. χρόνος=Θ(n) και E<Θ(1)

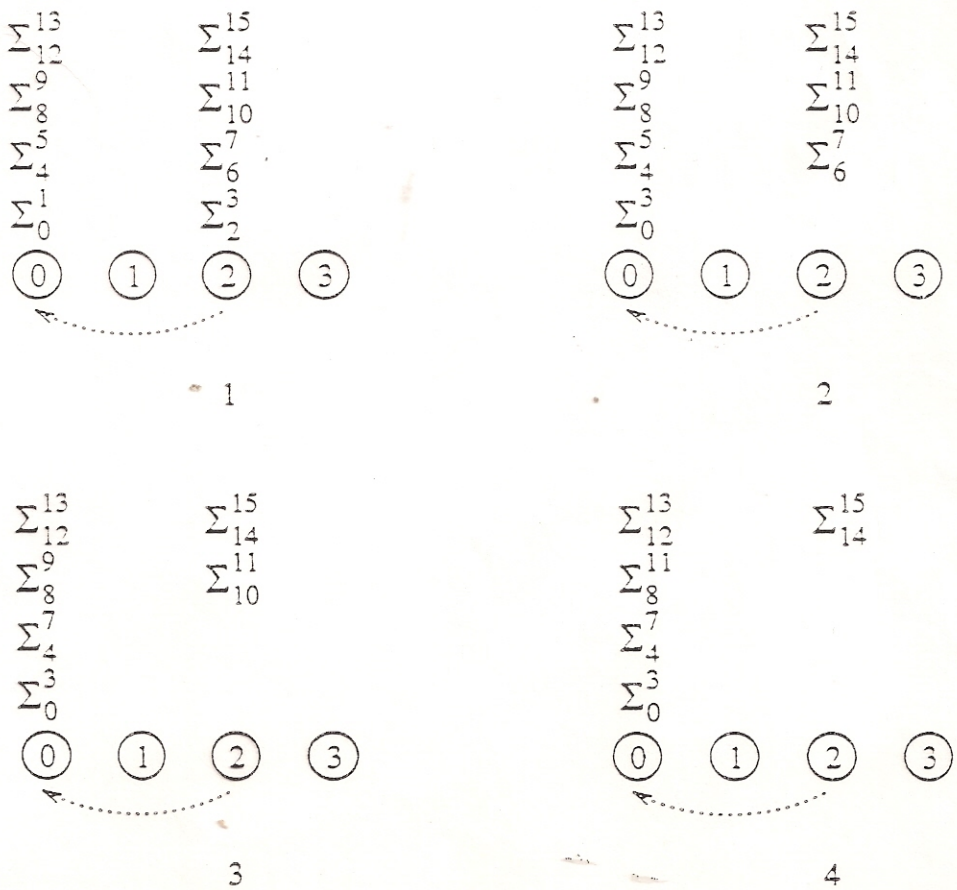
p<n, n,p δυνάμεις του 2

δεδομένα που παίρνει ο κάθε επεξ.->

n/p



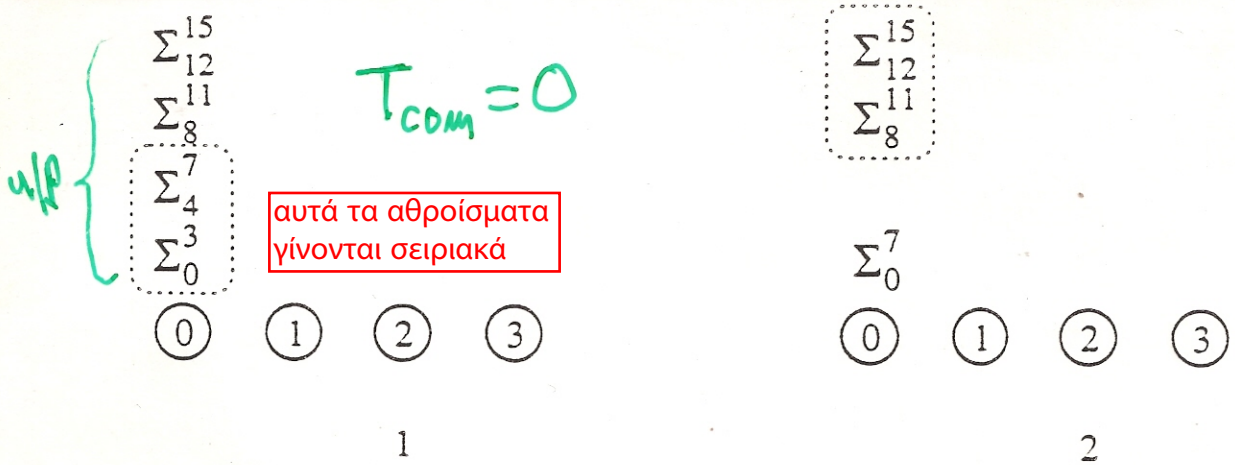
(a)



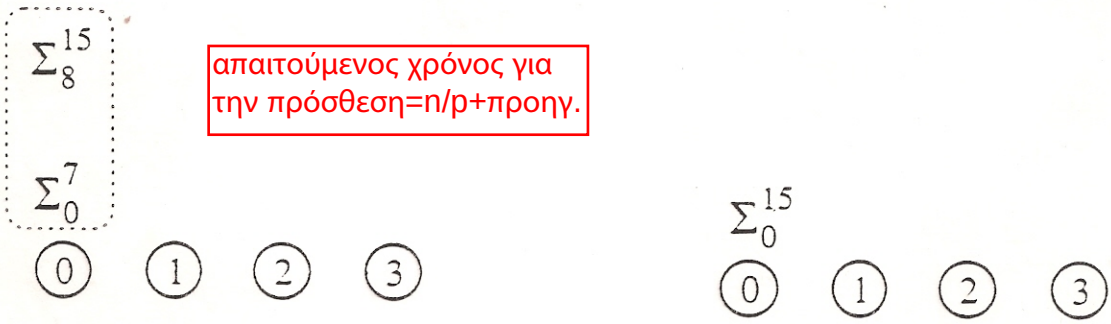
(b)

(n/p) log p

-> σε κάθε πράξη logp βήματα και άθροισμα + επικ ανά βήμα=σταθ.=n/p



(c)

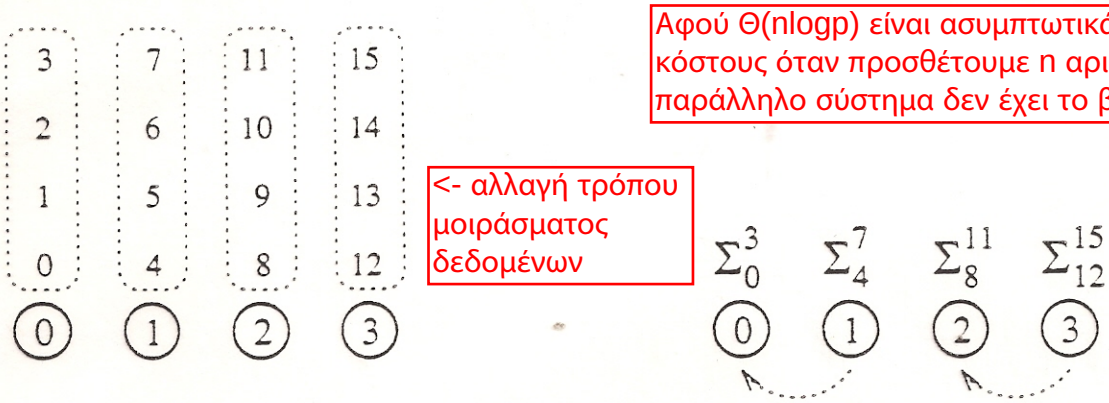


(d) $T_p = \Theta((n/p)\log p)$

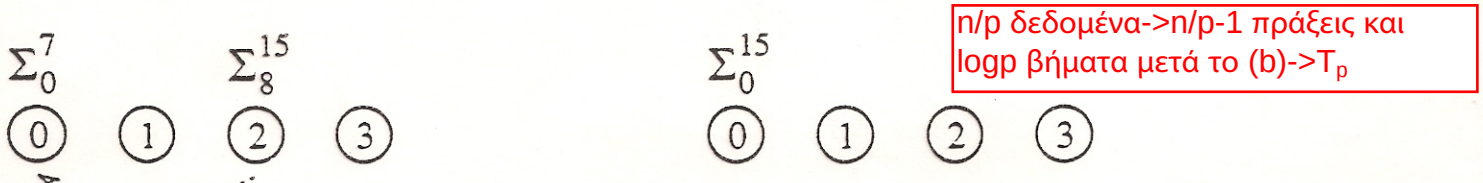
$C_p = \Theta(n\log p) > \Theta(n) = C_1$ όχι βέλτιστου κόστους

(e) Θ(n/p)-> ο χρόνος που χρειάζεται ο κάθε επεξ. για να προσθέσει n/p αριθμούς

Αφού Θ(nlogp) είναι ασυμπτωτικά υψηλότερο του Θ(n) κόστους όταν προσθέτουμε n αριθμούς σειριακά-> το παράλληλο σύστημα δεν έχει το βέλτιστο κόστος.



(a) (b)



(c) $T_p = \Theta(n/p + \log p)$

Αν $n = \Omega(p\log p)$ τότε $C_A = \Theta(n)$ ☆

(d) $C_A = \Theta(n + p\log p)$

n/p δεδομένα->n/p-1 πράξεις και logp βήματα μετά το (b)->T_p

$T_p = \frac{n}{p} + 2\log p$, $S = \frac{n}{n+2p\log p}$

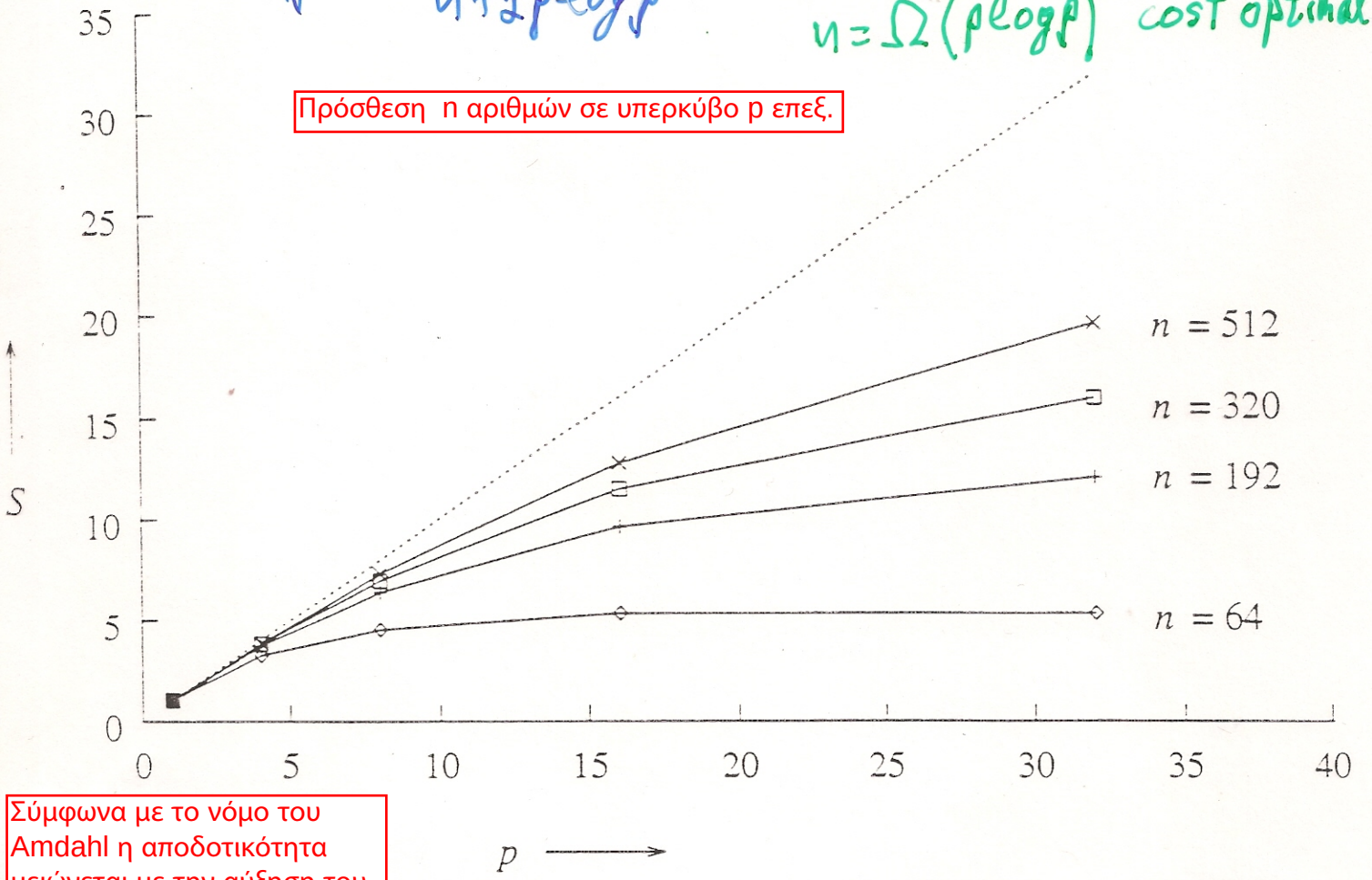
+ com

$E_p = \frac{n}{n+2p\log p}$

$c_p = O(n+2p\log p)$

$n = \Omega(p\log p)$ cost optimal

Πρόσθεση n αριθμών σε υπερκύβο p επεξ.



Σύμφωνα με το νόμο του Amdahl η αποδοτικότητα μειώνεται με την αύξηση του αριθμού των επεξ.

$n = 8p\log p$

E_p

n	$p = 1$	$p = 4$	$p = 8$	$p = 16$	$p = 32$
64	1.0	.80	.57	.33	.17
192	1.0	.92	.80	.60	.38
320	1.0	.95	.87	.71	.50
512	1.0	.97	.91	.80	.62

- Η ταχύτητα δεν αυξάνει γραμμικά όταν αυξάνει το μέγεθος των επεξεργαστών
- Για μεγαλύτερα προβλήματα η ταχύτητα και η αποδοτικότητα αυξάνουν

$N = 2^n$, $n \geq 1$, $x_1, x_2, \ldots, x_N \longrightarrow M_1, M_2, \ldots, M_N$

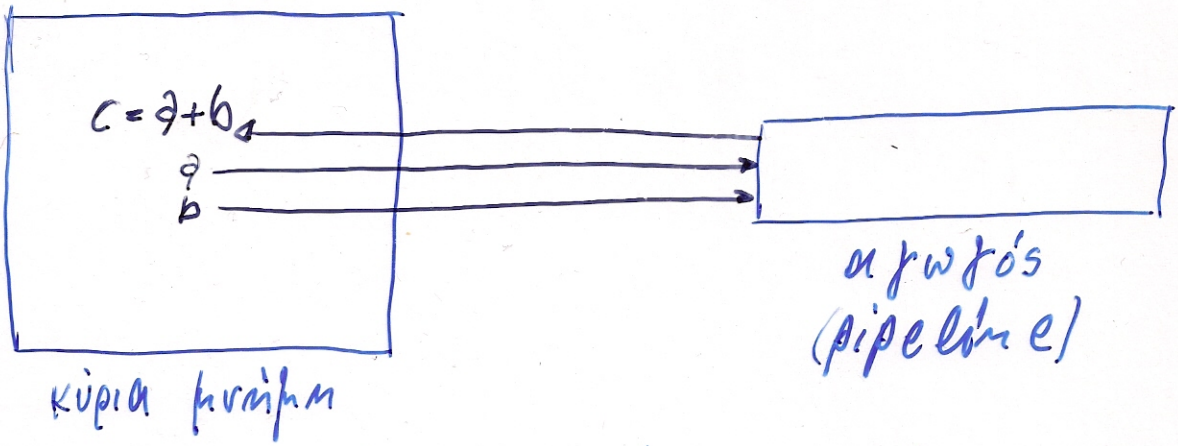
Το τελικό αποτέλεσμα στην M_1.

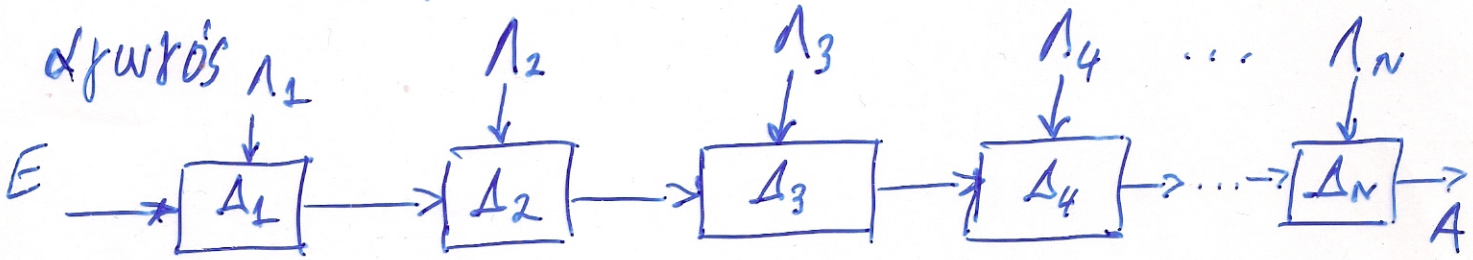
```
inc = 1
for j = 1 to log N do
for i ∈ {1 + 2·k·inc | k = 0, 1, 2, ..., N/2^j − 1} do parallel
/* επεξεργασία Pi */
   read Mi και Mi+inc
   add περιεχόμενα Mi και Mi+inc
   write το άθροισμα στο Mi
   inc = 2 inc
```

Associative fan-in αλγόριθμος

j	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
0	1	2	3	4	5	6	7	8
1	3		7		11		15	
2	10				26			
3	36							

ΔΙΑΝΥΣΜΑΤΙΚΟΙ ΥΠΟΛΟΓΙΣΤΕΣ





E = εντολή, Δ_i = δεδομένο$_i$

Λ_i = i-οστή λειτουργία, A = αποτέλεσμα

Σωλήνωση τριών τμημάτων

Περίοδος χρόνου	1	2	3	4	...	n	n+1
Πάρε	a_1	a_2	a_3	a_4	...	a_n	–
Πάρε και Πρόσθεσε	–	a_1+b_1	a_2+b_2	a_3+b_3		$a_{n-1}+b_{n-1}$	
Εκτύπωσε	–	–	a_1+b_1	a_2+b_2		$a_{n-2}+b_{n-2}$	