Θέματα Κατανεμημένων και Παράλληλων Συστημάτων

Μάθημα #2

Βασικές Έννοιες Παράλληλου Υπολογισμού. Στοιχεία Αρχιτεκτονικής Παράλληλων Μηχανών. Τοπολογίες Δικτύων Διασύνδεσης.

Παράλληλος Υπολογισμός

Σκοπός: **Εύρεση αποδοτικών λύσεων σε μεγαλύτερα και περισσότερο σύνθετα προβλήματα**

Διαθεματική Περιοχή (Interdisciplinary area) ***διότι περιλαμβάνει θέματα:***

- **Σχεδιασμού Αλγορίθμων**
- **Γλωσσών Προγραμματισμού και Μεταγλωττιστών.**
- **Λειτουργικών Συστημάτων**
- **Ανάπτυξης Λογισμικού**
- **Αρχιτεκτονικής Συστημάτων**

Gill, S. (1958), "Parallel Programming," *The Computer Journal*, vol. 1, April, pp. 2-10.

"... There is therefore nothing new in the idea of parallelism, but its application to computers.

The author cannot believe that there will be any insuperable difficulty in extending it to computers.

It is not to be expected that the necessary programming techniques will be worked out overnight. Much experimenting remains to be done.

After all, the techniques that are commonly used in programming today were only won at the cost of considerable toil several years ago.

In fact the advent of parallel programming may do something to revive the pioneering spirit in programming which seems at the present to be degenerating into a rather dull and routine occupation ..."

Δυσκολίες στον Παράλληλο Υπολογισμό

Δύσκολος ο σχεδιασμός αλγορίθμων

- **Πολύπλοκος ο προσδιορισμός και ο συντονισμός των παράλληλων διεργασιών**

Δύσκολη η ανάπτυξη λογισμικού

- **Έλλειψη αποδοτικών εργαλείων ανάπτυξης, προγραμματιστικών προτύπων και περιβαλλόντων**

Ποικιλία παράλληλων μηχανών

- **Ένας παράλληλος αλγόριθμος δεν είναι κατάλληλος για κάθε παράλληλ η μηχανη!**

Τύποι Παραλληλισμού

Εμφανής Παραλληλισμός (Overt Parallelism)

- **Χρήση Παράλληλου Αλγόριθμου**

***Συγκεκαλυμμένος ή Κρυφός (Covert Parallellism)*:**

- **Ο compiler «εξάγει» παραλληλισμό από τον ακολουθιακό αλγόριθμο**

Τύποι Παραλληλισμού

Εμφανής Παραλληλισμός (Overt Parallelism)

- **Χρήση Παράλληλου Αλγόριθμου**
- **Ο παραλληλισμός είναι φανερός στον προγραμματιστή**
- **Συνήθως έχουμε μεγάλες επιταχύνσεις (speedups) διότι ο προγραμματιστής έχει καλύτερη γνώση του προβλήματος από το compiler.**

Παράλληλος Αλγόριθμος

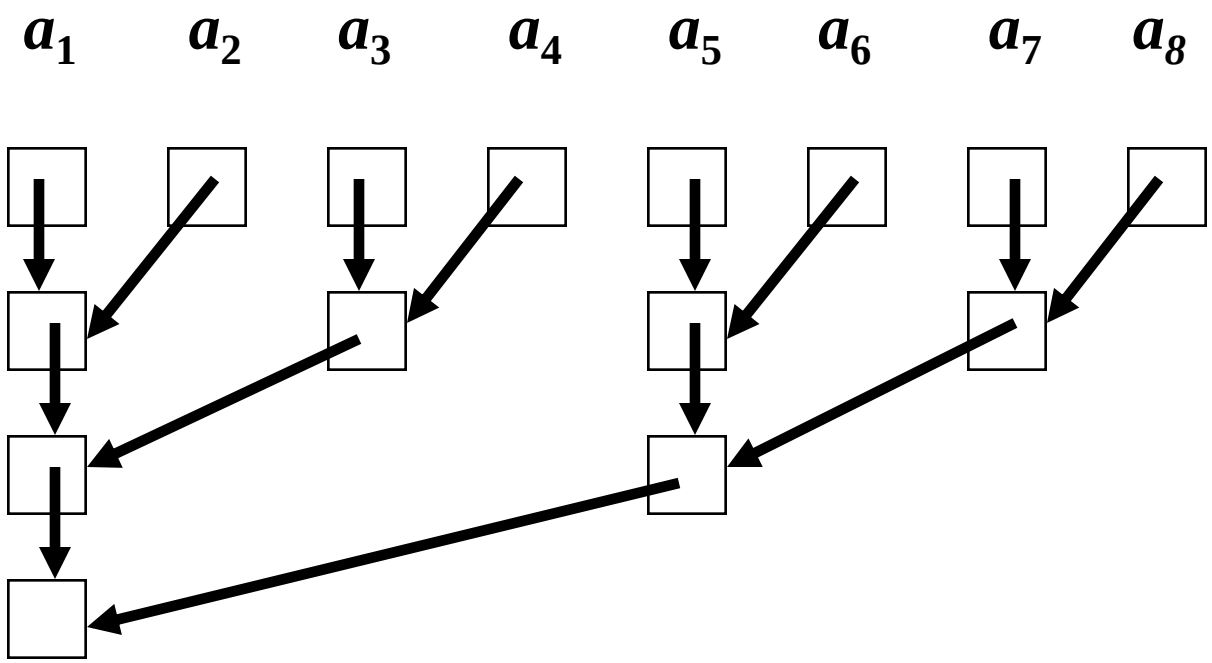
Αλγόριθμος : μια πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, που στοχεύουν στην επίλυση ενός προβλήματος.

Παράλληλος αλγόριθμος : αλγόριθμος του οποίου τμήματα μπορούν να εκτελεστούν ταυτόχρονα (σε διαφορετικές μονάδες επεξεργασίας).

Τα αποτελέσματα της εκτέλεσης των διαφόρων τμημάτων που εκτελούνται ταυτόχρονα, συνδυάζονται προκειμένου να προκύψει η λύση του προβλήματος.

Παράλληλος αλγόριθμος πρόσθεσης αριθμών

- **Δίνεται διάνυσμα A, $n = 2^k$ στοιχείων.**
- **Υπολογίζουμε το άθροισμα Sum = $a_1 + ... + a_n$**



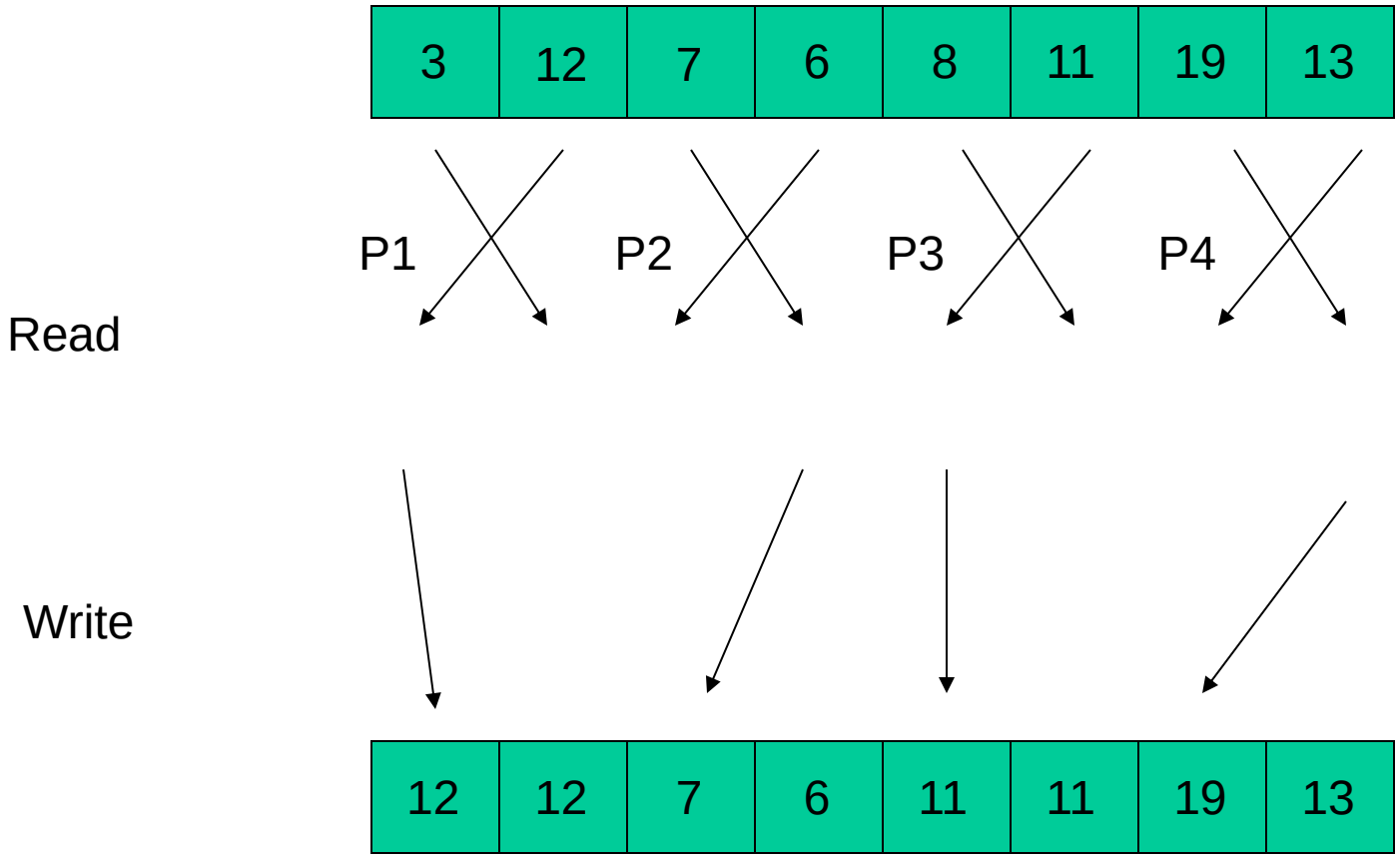
Παράλληλος αλγόριθμος πρόσθεσης αριθμών

```
integer  A[1..n]
Forall  i in 1 .. n  do
    A[i] := α_i
enddo
for h = 1 to k do
    forall  i in 1 .. n/2^h do
        A[i] := A[2i-1] + A[2i]
    enddo
enddo
Sum := A[1]
```

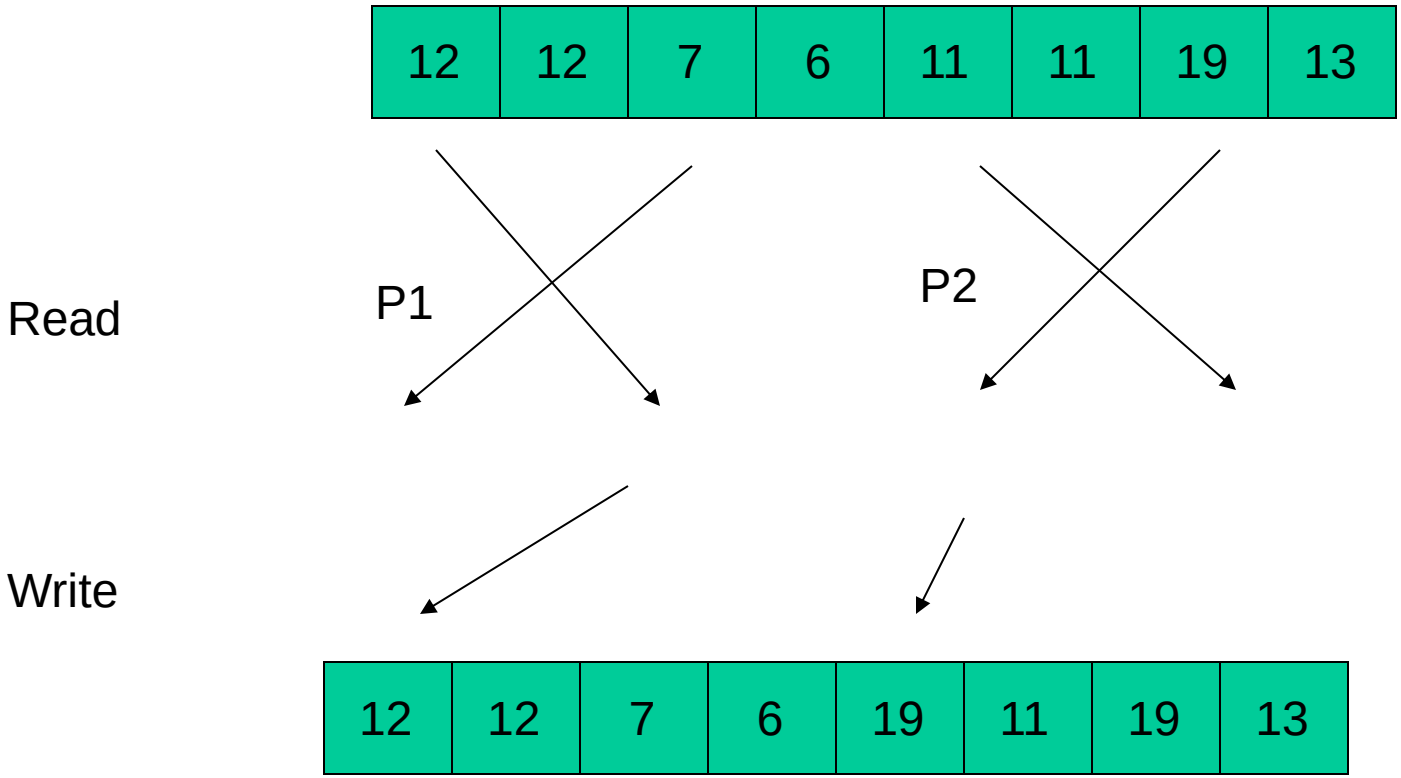
Παράλληλος αλγόριθμος εύρεσης της μεγαλύτερης τιμής σε διάνυσμα

- **Υποθέστε ότι το n είναι δύναμη του 2 και ότι διαθέτουμε n / 2 επεξεργαστές**
- **Κάθε επεξεργαστής διαβάζει τα στοιχεία δύο θέσεων του διανύσματος**
- **Κάθε επεξεργαστής γράφει τη μεγαλύτερη τιμή που διάβασε στην πρώτη από τις θέσεις που διάβασε**
- **Η διαδικασία επαναλαμβάνεται και μετά από lg n βήματα η μεγαλύτερη τιμή τοποθετείται στο A[1]**

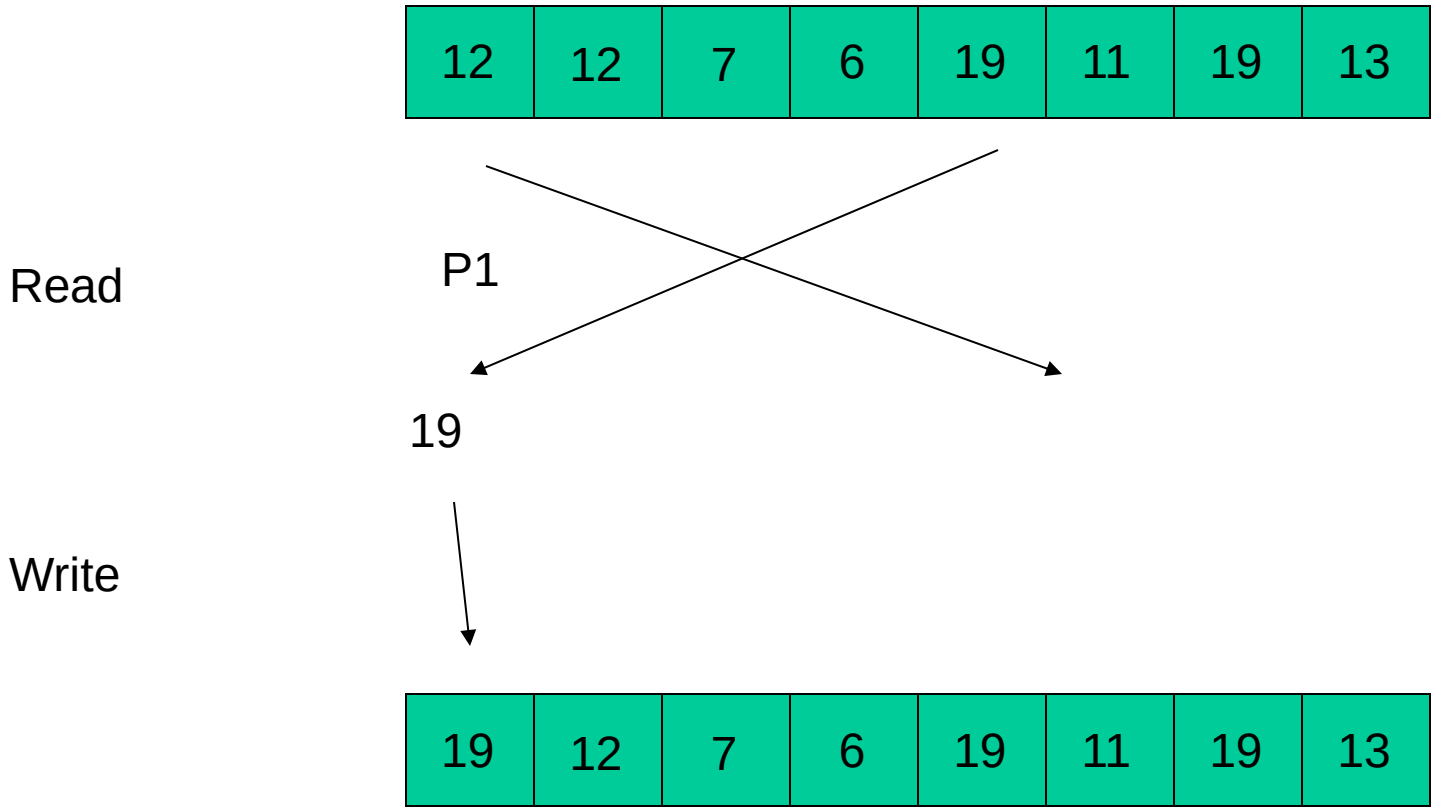
Παράδειγμα εύρεσης της μεγαλύτερης τιμής σε διάνυσμα



Παράδειγμα εύρεσης της μεγαλύτερης τιμής σε διάνυσμα



Παράδειγμα εύρεσης της μεγαλύτερης τιμής σε διάνυσμα



Τύποι Παραλληλισμού

***Συγκεκαλυμμένος ή Κρυφός (Covert Parallellism)*:**

- **Ο compiler «βγάζει» παραλληλισμό από τον ακολουθιακό αλγόριθμο**
- **Ο παραλληλισμός δεν είναι φανερός στον προγραμματιστή**
- **Συνήθως έχουμε μικρές επιταχύνσεις (speedups)**

Επιτάχυνση (Speedup) and Αποδοτικότητα (Efficiency)

A : παράλληλος αλγόριθμος που λύνει το πρόβλημα π.

t_s : ο χρόνος εκτέλεσης του καλύτερου ακολουθιακού αλγορίθμου που λύνει το π

t_p : ο χρόνος που ο A χρειάζεται για να λύσει το π σε παράλληλη μηχανή με n επεξεργαστές. Τότε η *επιτάχυνση (speedup) που επιτυγχάνει ο A* είναι

$$S(n) = t_s / t_p$$

Η *μέγιστη επιτάχυνση ισούται με n (Linear speedup):* ο υπολογισμός μοιράζεται σε διεργασίες ίσης διάρκειας (υποθέτουμε ότι δεν υφίσταται κόστος επικοινωνίας)

$$S(n) \leq t_s / (t_s / n) = n$$

Μη γραμμικές (Superlinear) επιταχύνσεις* $(S(n) > n)$ *είναι δυνατές!

Η αποδοτικότητα (*efficiency) του A* είναι

$$E_n = t_1 / (t_n * n)$$

t_1 : ο χρόνος που χρειάζεται ο αλγόριθμος με τη χρήση ενός επεξεργαστή

t_n : ο χρόνος που χρειάζεται ο αλγόριθμος με τη χρήση n επεξεργαστών

Η αποδοτικότητα αποτελεί ένδειξη της καλής (αποδοτικής) χρήσης των *n* επεξεργαστών.

Όταν η E_p δίνεται %, ήτοι

$$E_n = E_n * 100\%$$

E_n = 100% όταν όλοι οι επεξεργαστές χρησιμοποιούνται στο σύνολο της διάρκειας του υπολογισμού.

E_n = 50% όταν οι επεξεργαστές χρησιμοποιούνται κατά μέσο όρο κατά το μισό χρόνο του υπολογισμού.

Παράγοντες που περιορίζουν την επιτάχυνση

- Περίοδοι που κάποιοι επεξεργαστές παραμένουν ανενεργοί
- Επιπλέον υπολογισμοί στον παράλληλο αλγόριθμο που δεν εμφανίζονται στον ακολουθιακό (π.χ., υπολογισμός σταθερών τοπικά από κάθε επεξεργαστή)
- Επικοινωνία μεταξύ των διεργασιών
- Μέρη του υπολογισμού που δεν μπορούν να «παραλληλοποιηθούν»

Ο Νόμος του Amdahl

Έστω *f* το ποσοστό του υπολογισμού που δεν μπορεί να διαιρεθεί σε παράλληλες διεργασίες

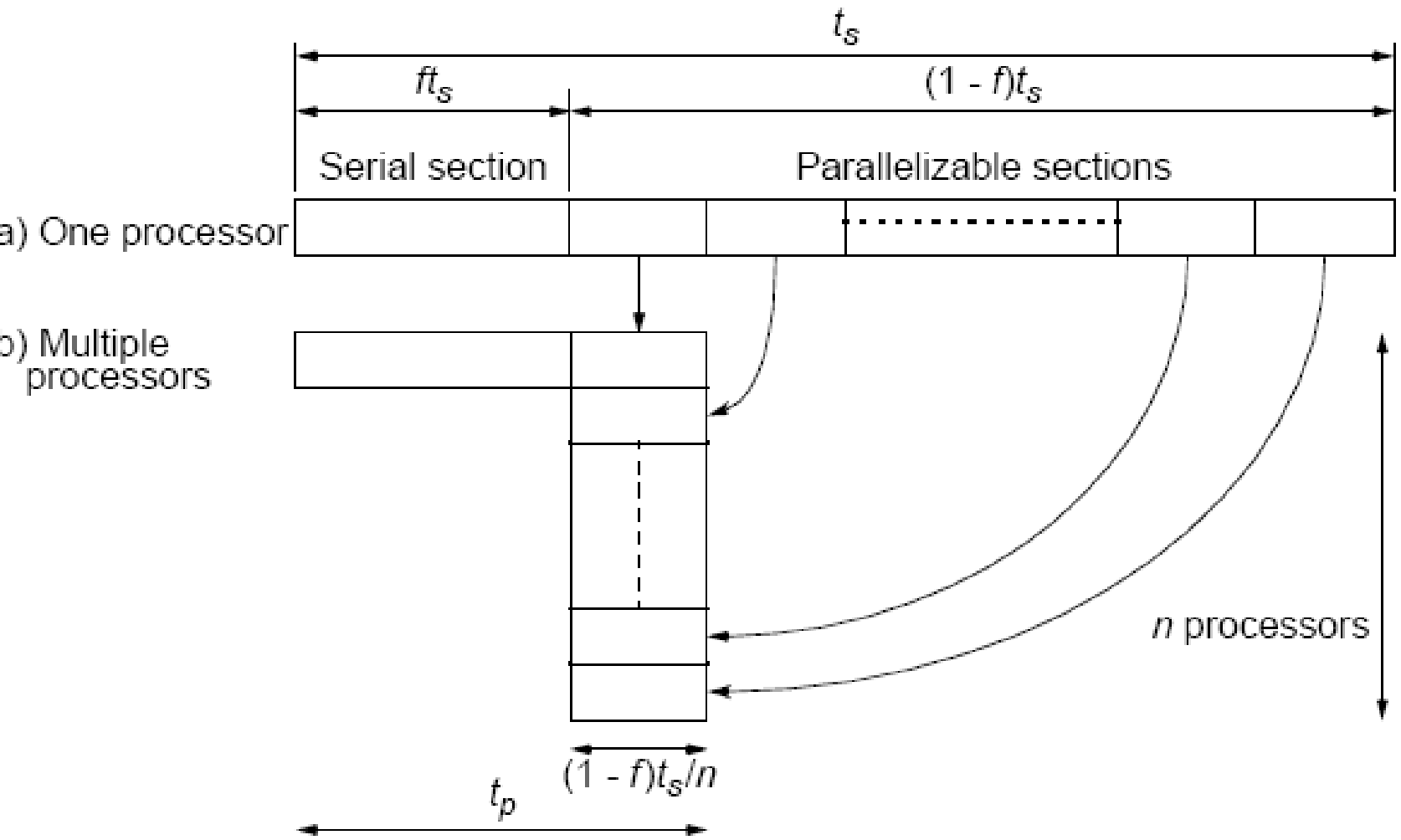
Υποθέτουμε ότι δεν υφίσταται κόστος επικοινωνίας.

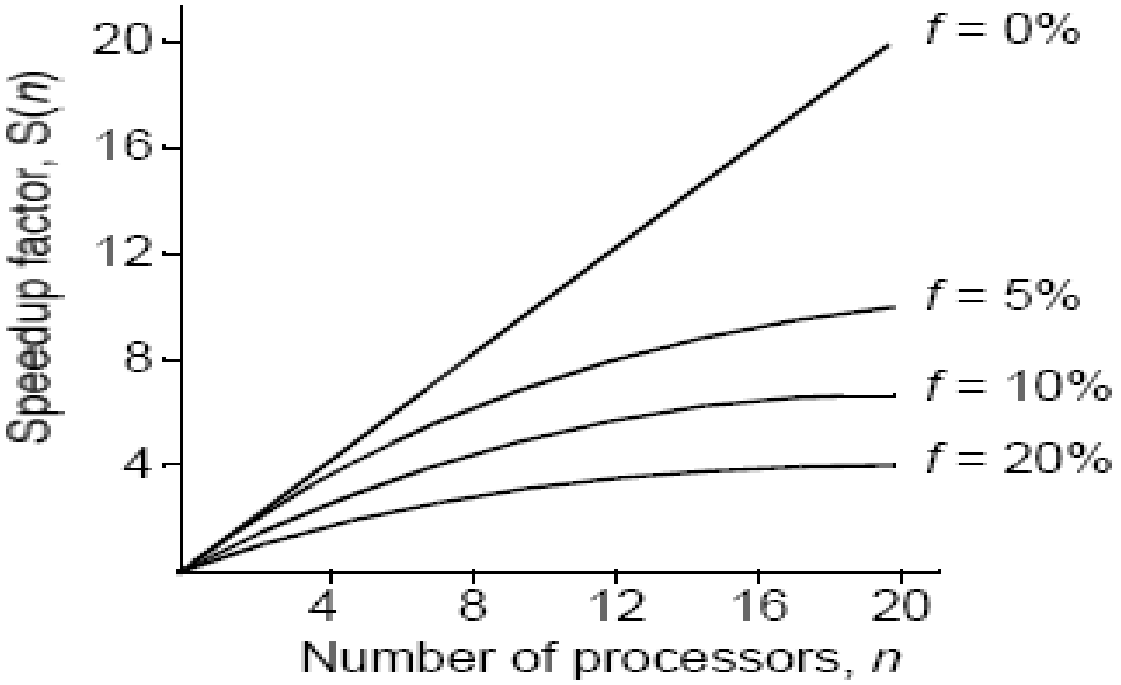
Τότε $t_p = ft_s + (1-f)t_s/n$, *n: # επεξεργαστών*

$$S(n) = \frac{t_s}{ft_s + (1-f)t_s/n} = \frac{n}{1 + (n-1)f}$$

$$S(n) = t_s/t_p = 1/[f + (1-f)/n]$$

Maximum Speedup - Amdahl's law





Αν *n* →∞ τότε *S(n)= 1/f*

Αν μόνον 5% του υπολογισμού είναι ακολουθιακό τότε S(n)= 20 με άπειρο αριθμό επεξεργαστών!

Κατηγοριοποίηση Υπολογιστικών Συστημάτων κατά τον Flynn

***Single Instruction Single Data stream* (SISD)**

***Multiple Instruction streams Single Data stream* (MISD)**

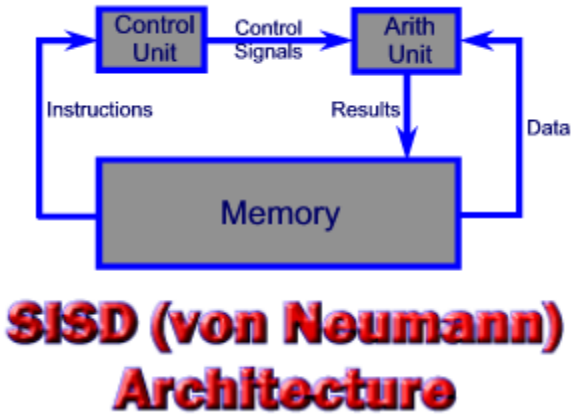
***Single Instruction stream Multiple Data streams* (SIMD)**

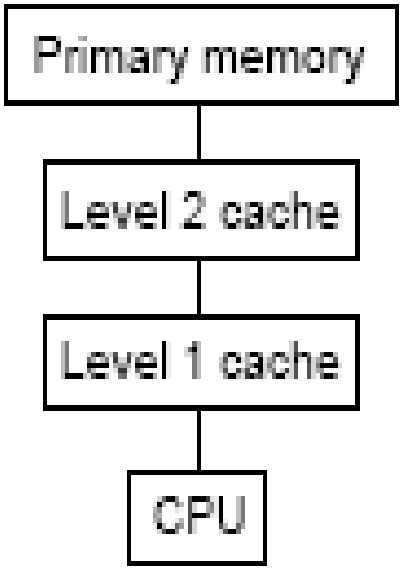
- **Μία μονάδα ελέγχου – όλοι οι επεξεργαστές εκτελούν την ίδια εντολή**
- **Οι επεξεργαστές εργάζονται πάνω σε διαφορετικά δεδομένα**

***Multiple Instruction stream Multiple Data stream* (MIMD)**

- **Κάθε επεξεργαστής έχει τη δική του μονάδα ελέγχου**
- **Κάθε επεξεργαστής μπορεί να εκτελεί διαφορετική εντολή**
- **Οι επεξεργαστές εργάζονται πάνω σε διαφορετικά δεδομένα**

SISD - **O von Neumann Υπολογιστής**

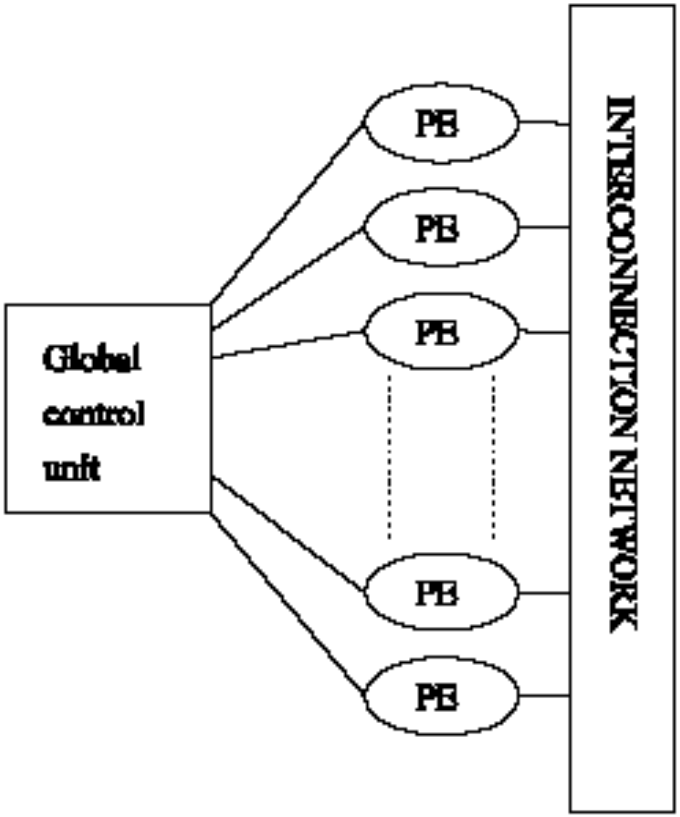




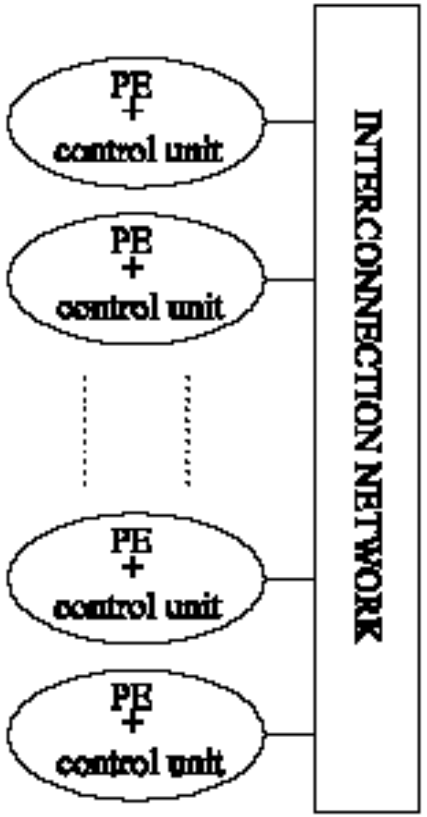
SISD Αρχιτεκτονική

Processors, cache, and memory in a modern machine.

Παράλληλες Μηχανές



SIMD architecture



MIMD architecture

SIMD / MIMD Αρχιτεκτονικές

SIMD

- **special purpose (όχι κατάλληλες για όλες τις εφαρμογές)**
- **συνήθως ακριβές (specially designed)**
- **λιγότερο hardware (single control unit)**
- **χρειάζονται λιγότερη μνήμη (μόνο ένα αντίγραφο του προγράμματος)**

MIMD

- **κατάλληλες για όλες τις εφαρμογές**
- **συνήθως φθηνές (off-the-shelf components & short design cycle)**
- **χρειάζονται λιγότερη μνήμη (πρόγραμμα και λειτουργικό σε κάθε επεξεργαστή)**

Επικοινωνία στις Παράλληλες Μηχανές

Δύο τρόποι επικοινωνίας των επεξεργαστών ενός παράλληλου συστήματος:

(1) Η χρήση *κοινής* ή *διαμοιραζόμενης μνήμης (shared memory)*.

(2) Η επικοινωνία των επεξεργαστών μέσω *ανταλλαγής* ή *περάσματος μηνυμάτων (message passing)*.

Συστήματα τα οποία διαθέτουν κοινό χώρο διευθύνσεων (shared dataspace):

- **Πολυεπεξεργαστές – Συστήματα Κοινής Μνήμης (Shared-memory systems - multiprocessors)**

Συστήματα τα οποία υποστηρίζουν πέρασμα μηνυμάτων:

- **Πολυυπολογιστές - Συστήματα Κατανεμημένης Μνήμης (Distributed-memory systems - multicomputers)**
- **Clusters (Networks of Workstations (NoWs) and Beowulf Clusters)**

Πολυεπεξεργαστές (Multiprocessors)

Αποτελούνται από έναν αριθμό επεξεργαστών οι οποίοι έχουν πρόσβαση στην ίδια κοινή μνήμη.

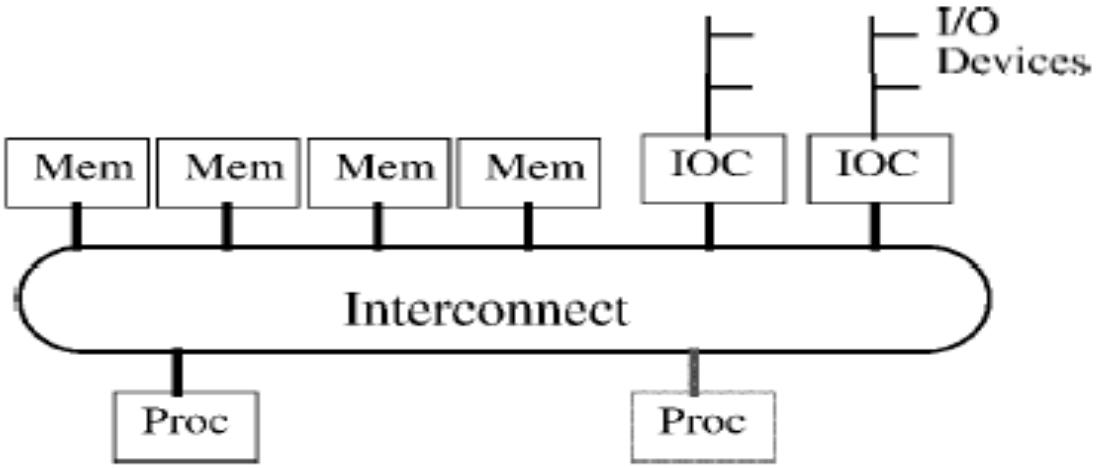
Δύο κατηγορίες πολυεπεξεργαστών:

- **Πολυεπεξεργαστές *Ομοιόμορφης Προσπέλασης Μνήμης (Uniform Memory Access – UMA)*** **στους οποίους η κοινή μνήμη είναι κεντρική.**
- **Πολυεπεξεργαστές *Μη Ομοιόμορφης Προσπέλασης Μνήμης (Non-Uniform Memory Access – NUMA)*** **στους οποίους η κοινή μνήμη είναι κατανεμημένη στους επεξεργαστές.**

Πολυεπεξεργαστές Ομοιόμορφης Προσπέλασης Μνήμης (UMA)

Οι επεξεργαστές έχουν πρόσβαση σε μία κεντρική κοινή μνήμη μέσω ενός κεντρικού μηχανισμού προσπέλασης μνήμης.

Κοινή μνήμη (shared memory)



- **Η επικοινωνία των επεξεργαστών γίνεται μέσω των συμβατικών εντολών πρόσβασης μνήμης (load & store)**
- **Μέρος του virtual address space των διεργασιών (processes) είναι κοινό i.e. αντιστοιχεί σε κοινό χώρο της μνήμης**

David Culler, J.P. Singh, Anoop Gupta, "*Parallel Computer Architecture: A Hardware/Software Approach*", Morgan Kaufmann 1998.

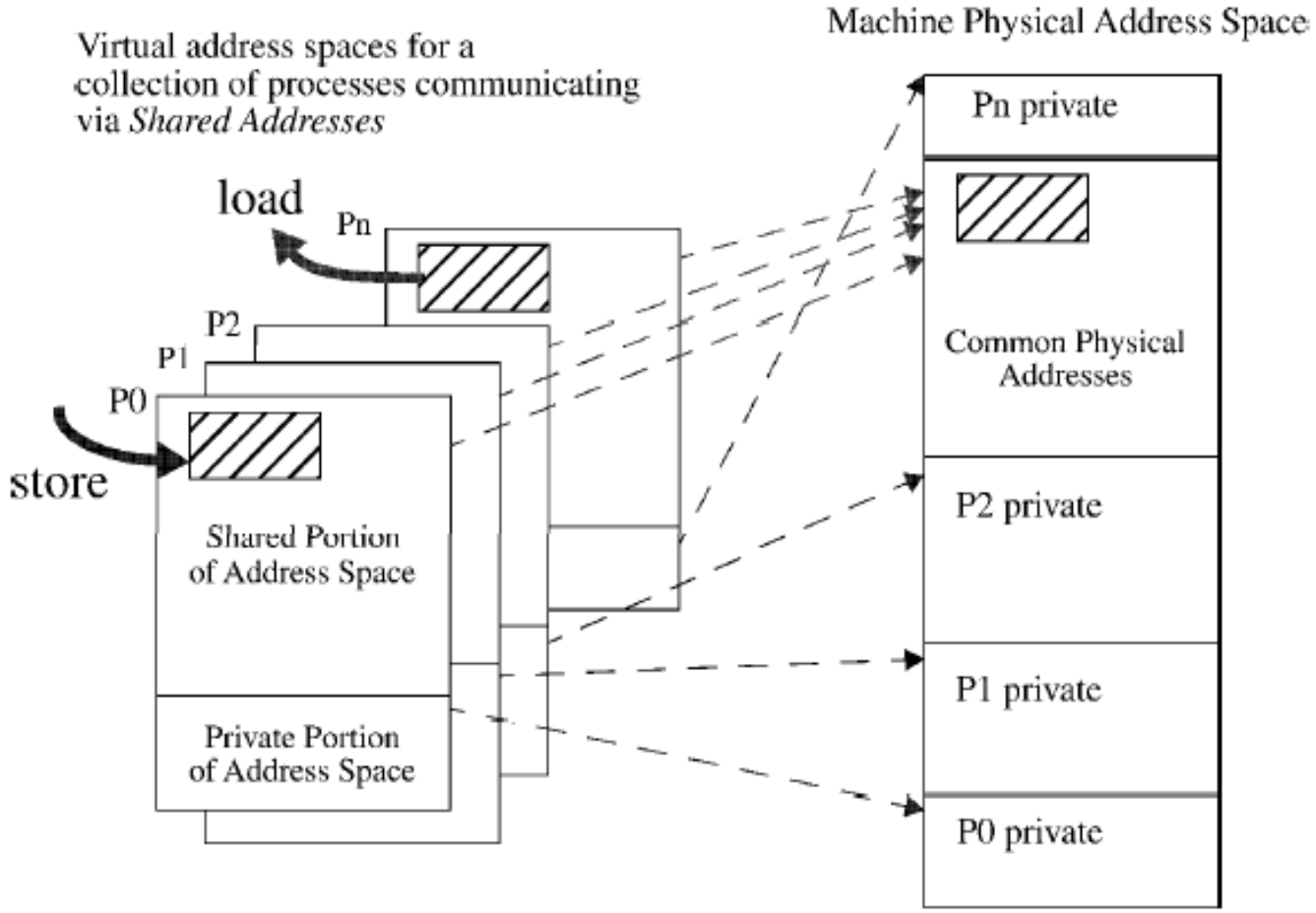
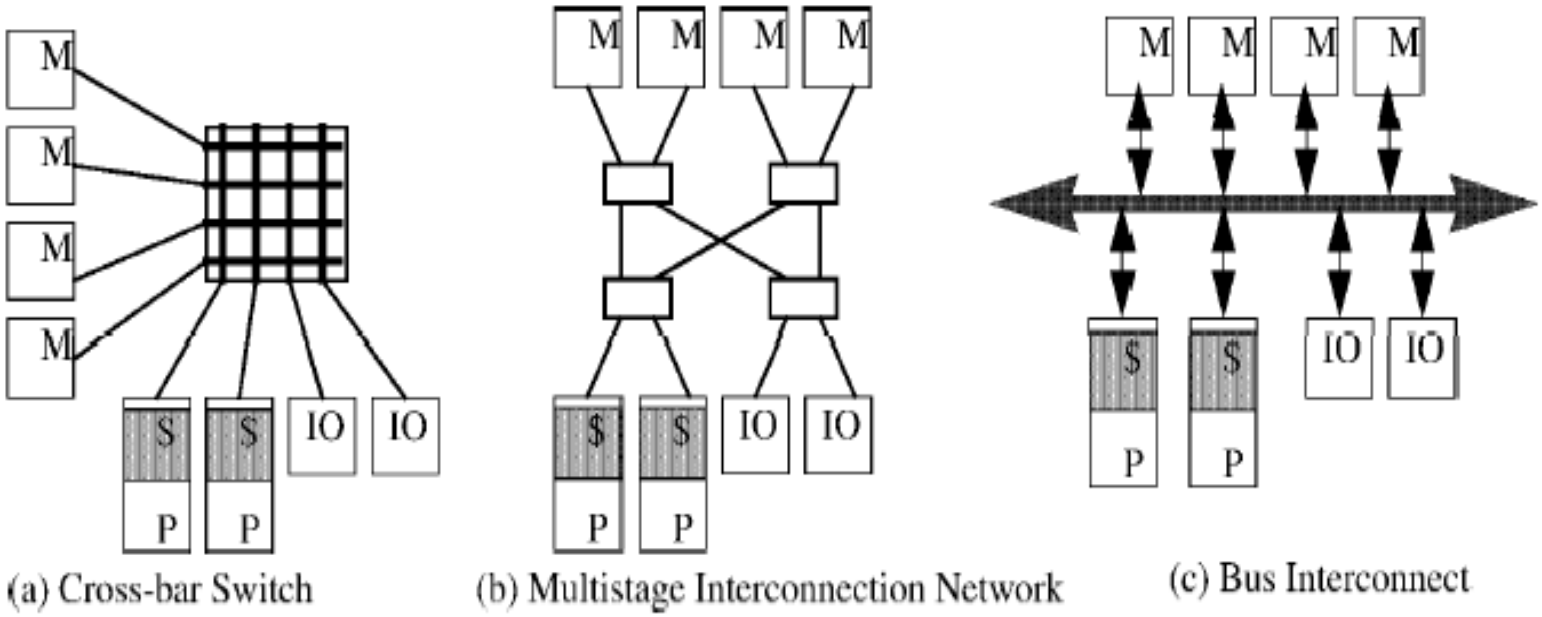


Figure 1-14 Typical memory model for shared-memory parallel programs

Collection of processes have a common region of physical addresses mapped into their virtual address space, in addition to the private region, which typically contains the stack and private data.

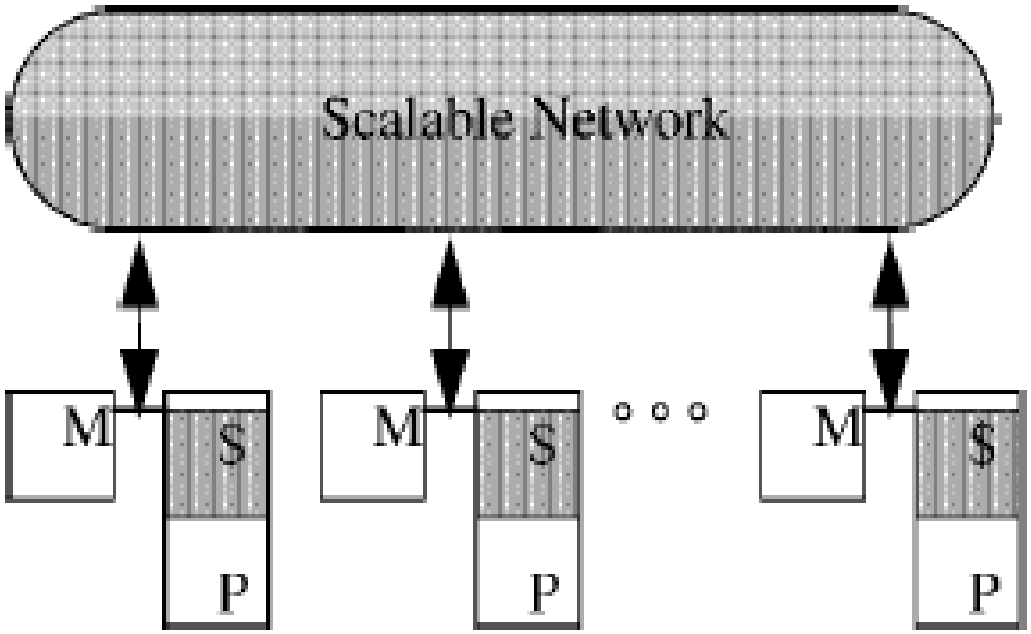
Τυπικά σχήματα διασύνδεσης μνημών – επεξεργαστών

Bus-based vs. switch-based organization



(a) Cross-bar Switch (b) Multistage Interconnection Network (c) Bus Interconnect

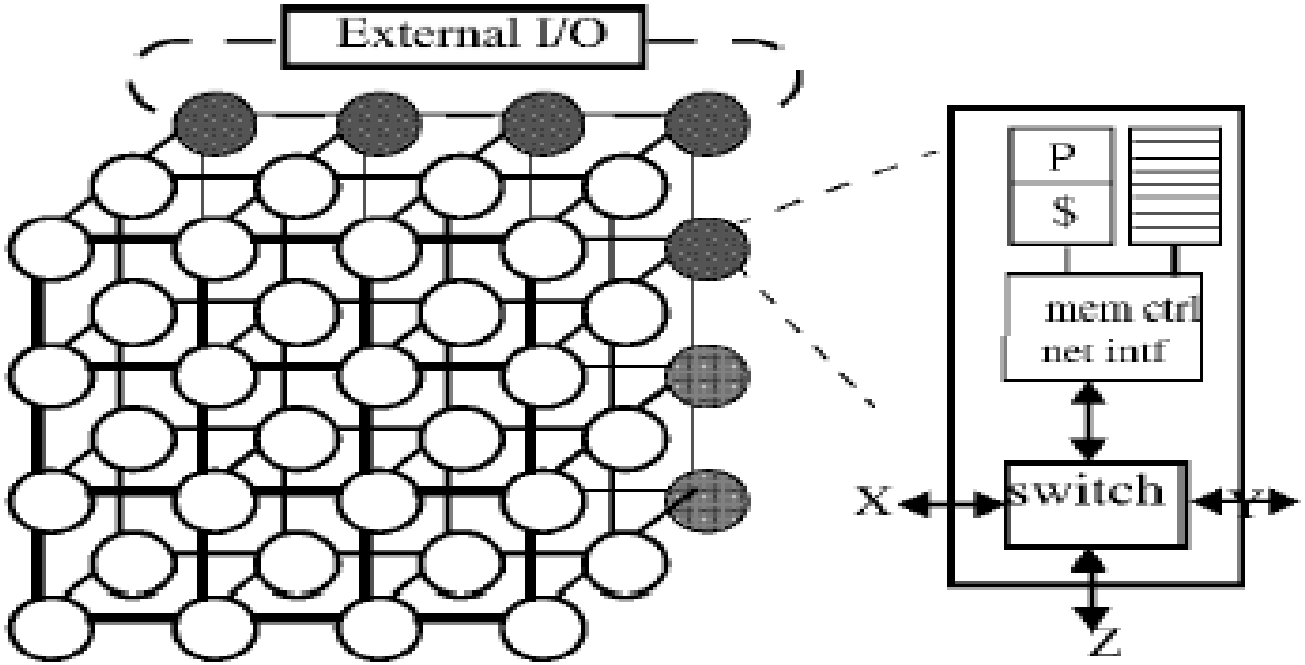
Πολυεπεξεργαστές Μη Ομοιόμορφης Προσπέλασης Μνήμης (NUMA)



Πολυεπεξεργαστές Μη Ομοιόμορφης Προσπέλασης Μνήμης (NUMA)

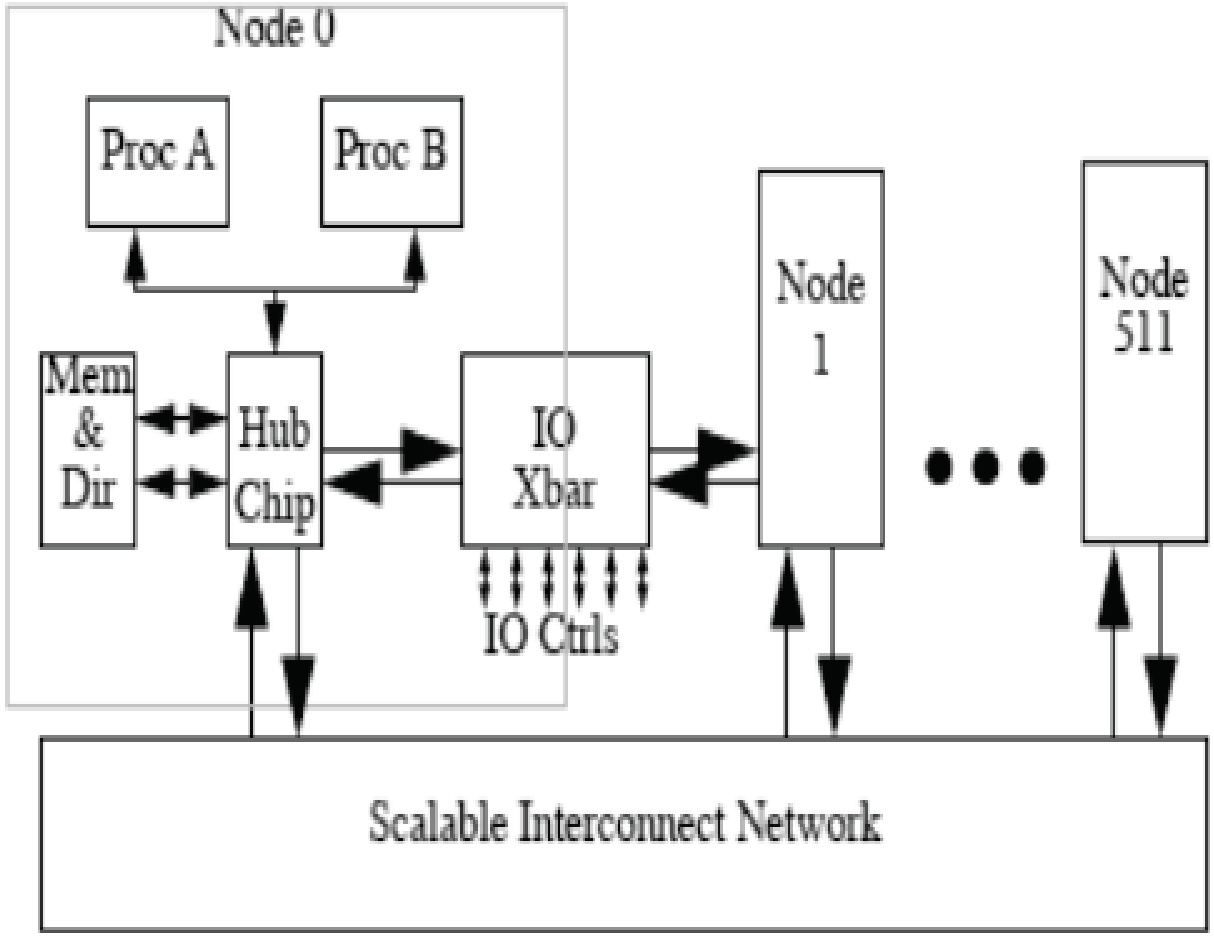
- **Χαρακτηρίζονται από *κοινό χώρο διευθύνσεων (shared address space)*. Ωστόσο, κάθε επεξεργαστής έχει τη δική του τοπική μνήμη στην οποία όμως, μπορούν να έχουν «άμεσα» (μέσω ενός δικτύου διασύνδεσης) πρόσβαση οι άλλοι επεξεργαστές.**
- **Ο τοπικός ελεγκτής μνήμης κάθε επεξεργαστή καθορίζει εάν θα εκτελέσει πρόσβαση στην τοπική μνήμη του επεξεργαστή (load/store) ή εάν θα εκτελέσει ένα απομακρυσμένο transaction με έναν ελεγκτή μνήμης άλλου επεξεργαστή.**
- **Η πρόσβαση ενός επεξεργαστή στην κοινή μνήμη μπορεί να αποτελεί πρόσβαση στην τοπική μνήμη του επεξεργαστή αν τα δεδομένα *τυχαία ή σκόπιμα* έχουν αποθηκευτεί στην τοπική μνήμη του εν λόγω επεξεργαστή.**
- **Η απόδοση ενός πολυεπεξεργαστή NUMA εξαρτάται από την *τοπικότητα των δεδομένων (data locality)* και οι αλγόριθμοι σε NUMA συστήματα μπορούν να αξιοποιήσουν την τοπικότητα για να αυξήσουν την απόδοσή τους.**

Cray T3E NUMA Scalable Shared Address Space Machine

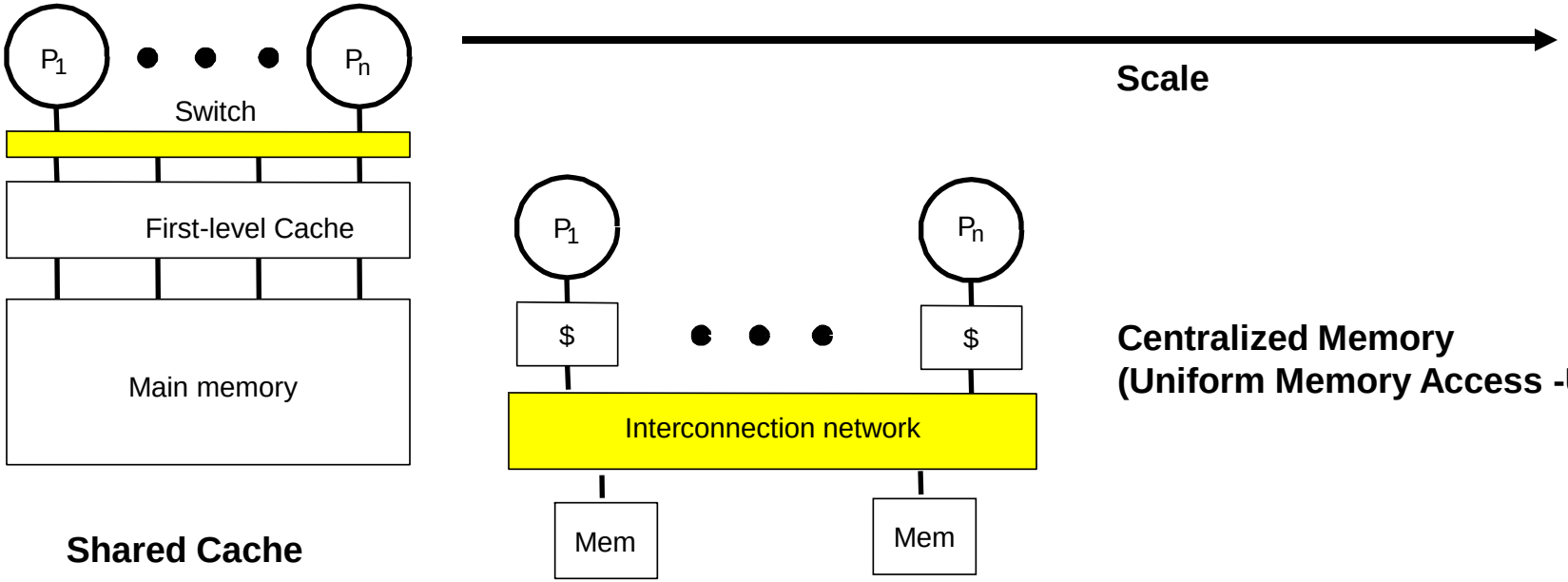


The SGI Origin: a ccNUMA highly scalable server.

J. Laudon and D. Lenoski. *Proc. of the 24th Intl. Symp. on Computer Architecture*, 1997.



Πολυεπεξεργαστές



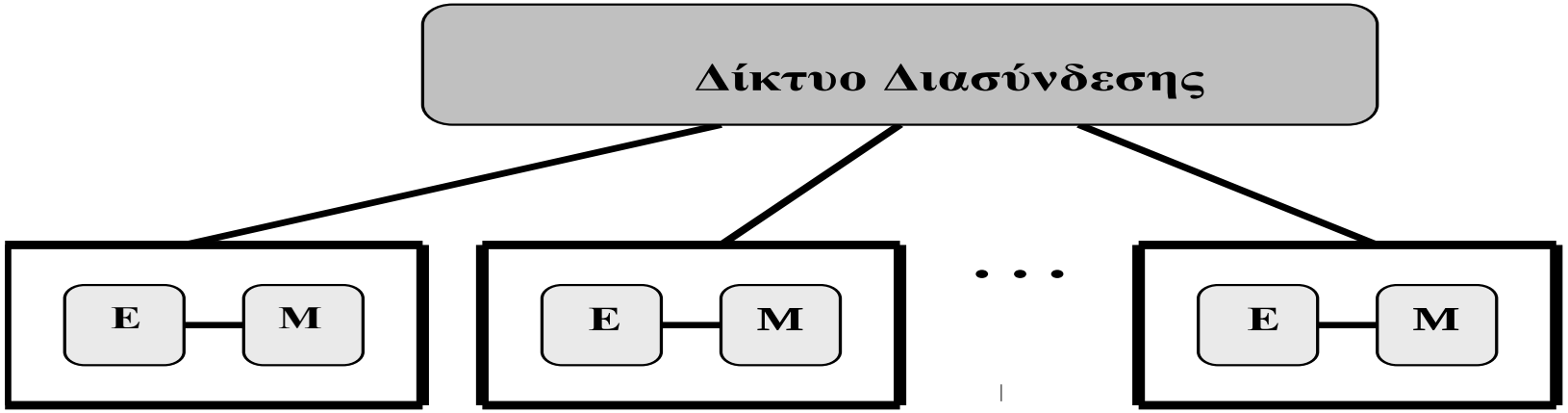
Shared Cache

Centralized Memory
(Uniform Memory Access -UMA)

Distributed Memory
(Non Uniform Memory Access - UMA)

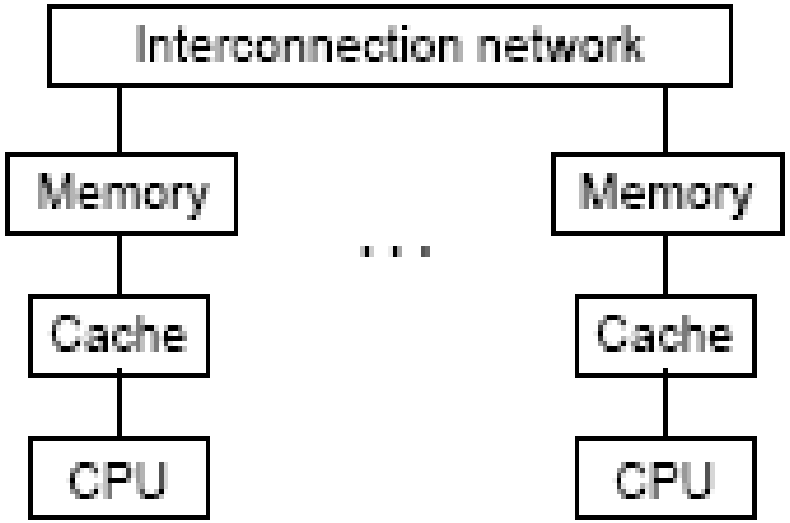
Πολυυπολογιστές (Multicomputers)
Συστήματα Κατανεμημένης Μνήμης

- **Κάθε επεξεργαστής έχει τη δική του τοπική μνήμη. Δεν υπάρχει κοινή μνήμη στην οποία έχουν πρόσβαση όλοι οι επεξεργαστές.**
- **Η επικοινωνία των επεξεργαστών γίνεται με πέρασμα μηνυμάτων (message-passing) μέσω δικτύου διασύνδεσης.**



E : Επεξεργαστής
M : Μνήμη

Πολυυπολογιστές (Multicomputers)
Συστήματα Κατανεμημένης Μνήμης



Structure of distributed-memory machines.

Πολυυπολογιστές (Multicomputers)
Συστήματα Κατανεμημένης Μνήμης

Consist of

- **Processors (each processor with its own private memory and I/O system)**
- **Interconnection network**

Provide

Communication between processors as explicit I/O operations

The primary difference between Multicomputers and NUMA systems is **that in the Multicomputers the communication is integrated in the I/O level rather than into the memory system**

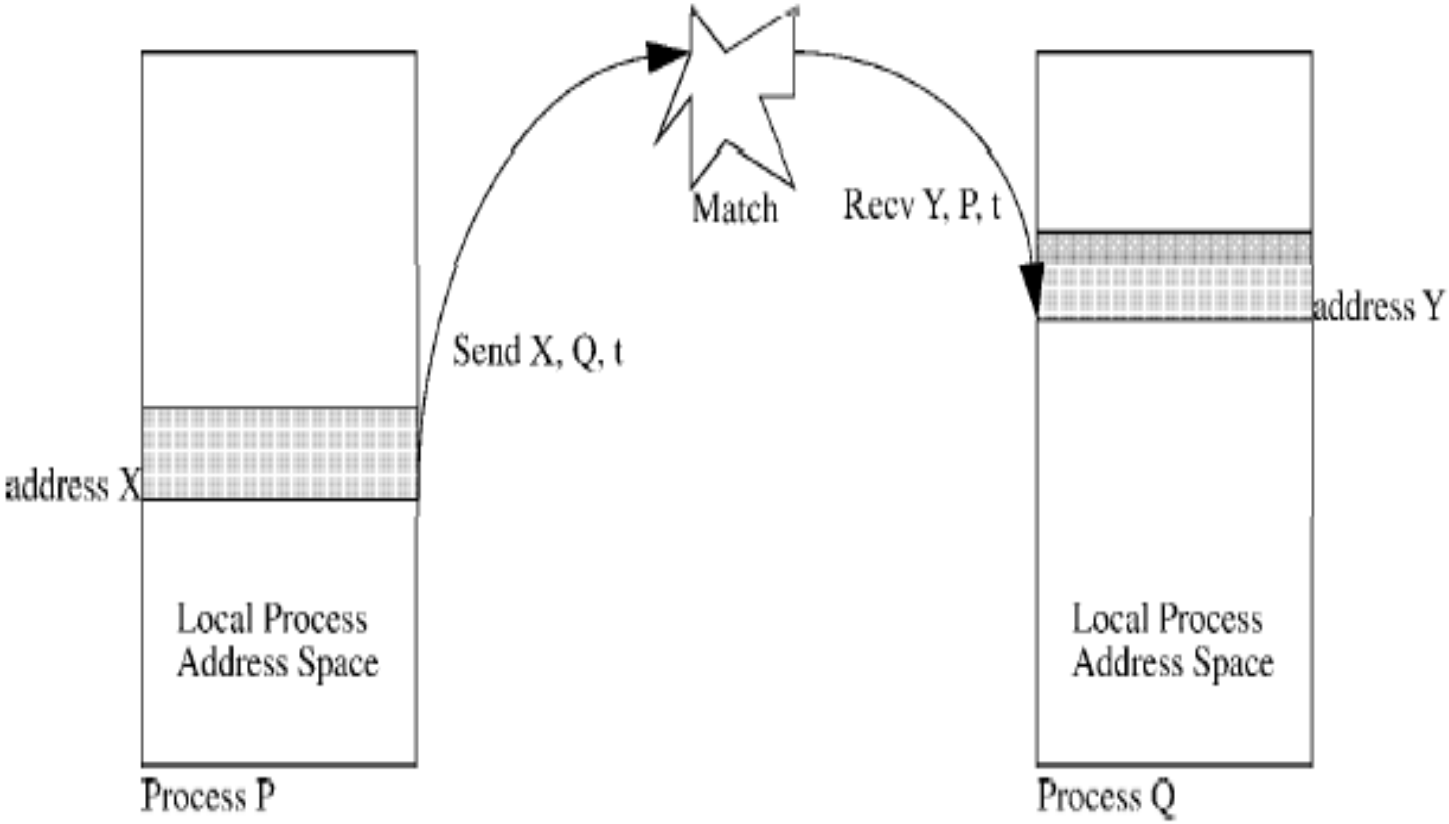
Same design style share the **Networks of Workstations (NoWs) and Beowulf Clusters) but in multicomputers:**

- **the packaging of the nodes is tighter,**
- **there is no direct user access to the nodes and**
- **the network is of much higher capability than standard local area network**

Programming strategy

- **Exchange data using send and receive primitives. The combination of a send and a matching receive accomplishes a memory to memory copy, where each end specifies its local data address and a synchronization event.**

User level send/receive message passing abstraction



Πολυυπολογιστές - Συστήματα Κατανεμημένης Μνήμης

- **Ο χρόνος εκτέλεσης ενός παράλληλου προγράμματος εξαρτάται από το πως έχουν κατανεμηθεί τα δεδομένα στις τοπικές μνήμες των διαφόρων επεξεργαστών.**

- **Ο προγραμματισμός των πολυεπεξεργαστών διαφέρει κατά πολύ, από τον προγραμματισμό των πολυυπολογιστών, αφού τα πρότυπα παράλληλου υπολογιστή τα οποία αντιστοιχούν στις δύο κατηγορίες μηχανών είναι διαφορετικά.**

- **Πιο εύκολος ο προγραμματισμός των πολυεπεξεργαστών από τον προγραμματισμό των πολυυπολογιστών.**

Δίκτυα Διασύνδεσης

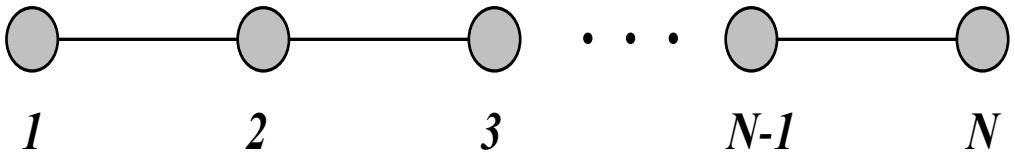
Αναπαριστάνονται ως γραφήματα των οποίων:

- **οι *κόμβοι* αντιπροσωπεύουν επεξεργαστές, ή μεταγωγείς δικτύου (network switches), ή μονάδες μνήμης, και**
- **οι *ακμές* αντιπροσωπεύουν φυσικούς συνδέσμους μεταξύ ζευγών κόμβων**

Προκειμένου να σχεδιασθεί το δίκτυο διασύνδεσης ενός παράλληλου υπολογιστή, θα πρέπει να εξετασθούν:

- **το κόστος υλοποίησης του δικτύου και**
- **η απόδοση του δικτύου**

Γραμμικό δίκτυο (linear network)

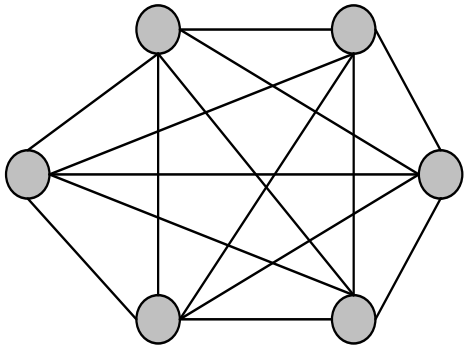


- **Για την διασύνδεση και επικοινωνία *N* επεξεργαστών χρησιμοποιούνται *N-1* σύνδεσμοι .**

  Κόστος: O(N)

- **Για να ανταλλάξουν δεδομένα οι επεξεργαστές *1* και *N* απαιτούνται *N-1* βήματα.**

Πλήρες δίκτυο (fully connected network)



- **Για τη διασύνδεση και επικοινωνία μεταξύ *N* επεξεργαστών χρησιμοποιούνται *N(N-1)* ακμές.**

  Κόστος: $O(N^2)$
- **Οποιοιδήποτε δύο επεξεργαστές επικοινωνούν μεταξύ τους σε *ένα* βήμα.**

Βασικές παράμετροι της τοπολογίας ενός δικτύου διασύνδεσης

- **η *διάμετρος (diameter)* ή ο ελάχιστος αριθμός ακμών μεταξύ των δύο πιο απομακρυσμένων κόμβων του**
- **η *συνεκτικότητα (connectivity)* ή ο αριθμός των ακμών που πρόσκεινται σε κάθε κόμβο, και**
- **η υποστήριξη *διαμέρισης (partitioning)*.**

Για να είναι ένα παράλληλο σύστημα *διαμερίσιμο* θα πρέπει

- **το δίκτυο διασύνδεσης του συστήματος να μπορεί να διαμερίζεται σε υποδίκτυα, τα οποία διατηρούν την πλήρη λειτουργικότητα του αρχικού δικτύου, και**
- **το παράλληλο σύστημα να μπορεί να διαμερισθεί σε ανεξάρτητα υποσυστήματα, με την έννοια ότι, κανένα υποσύστημα δεν μπορεί να επεμβαίνει στην απρόσκοπτη λειτουργία κάποιου άλλου υποσυστήματος**

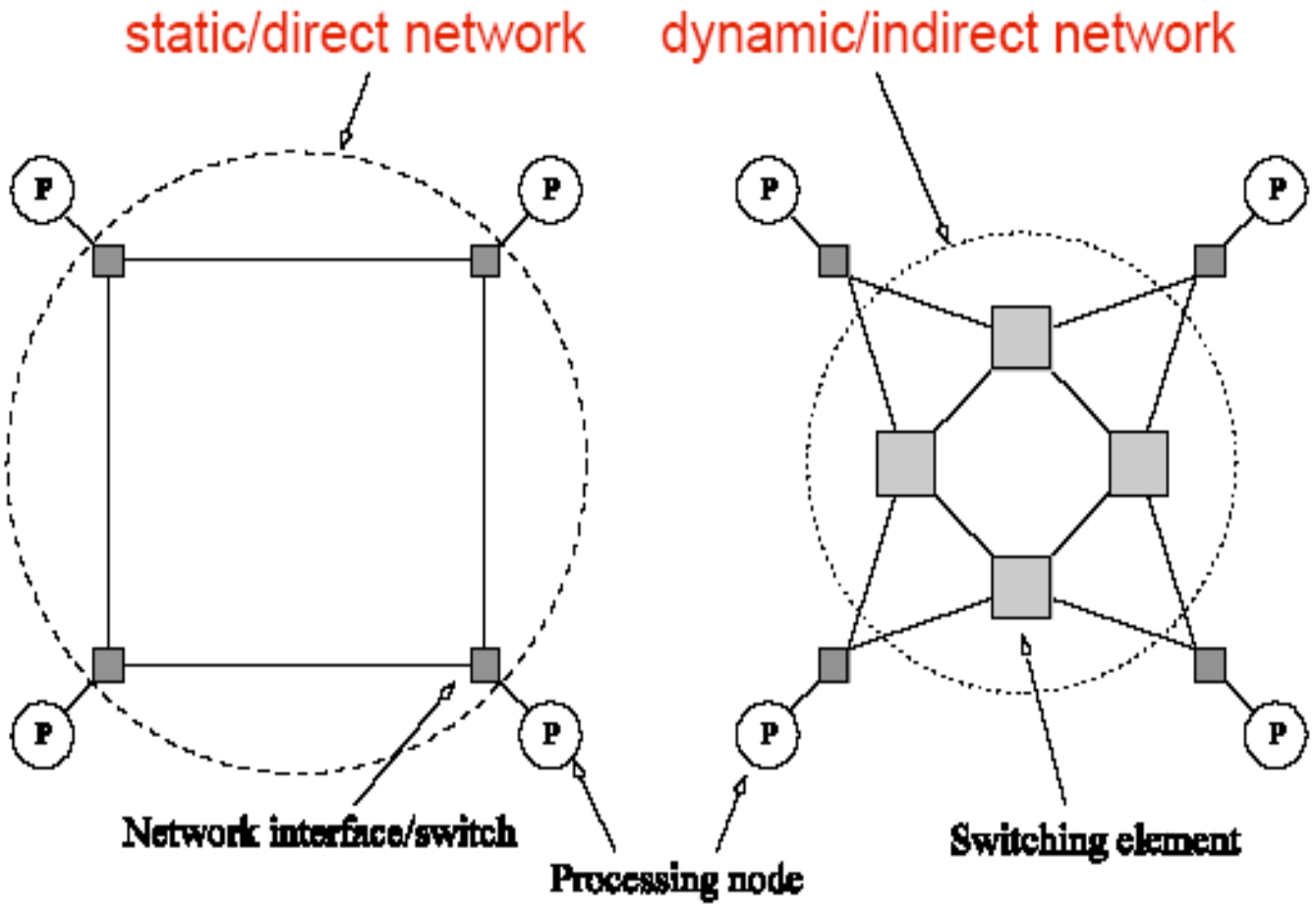
Χαρακτηριστικά διαμερίσιμων συστημάτων

- ***Ανοχή στα λάθη (fault tolerance):*** **η δυσλειτουργία ενός επεξεργαστή επηρεάζει μόνον τις διαμερίσεις που περιλαμβάνουν τον επεξεργαστή.**
- ***Εντοπισμός λαθών (fault detection):*** **στις εφαρμογές που απαιτείται μεγάλη αξιοπιστία, εκτελείται το ίδιο πρόγραμμα πάνω σε διαφορετικές διαμερίσεις και συγκρίνονται τα αποτελέσματα.**
- ***Ευκολότερη ανάπτυξη προγραμμάτων:*** **η διόρθωση λαθών είναι πιο εύκολη εάν γίνεται σε διαμερίσεις μικρού μεγέθους.**
- ***Πολλαπλοί ταυτόχρονοι χρήστες:*** **καθένας από τους οποίους, μπορεί να τρέχει διαφορετικό παράλληλο πρόγραμμα σε διαφορετική διαμέριση.**
- ***Αυξημένη χρήση και αξιοποίηση του συστήματος:*** **εάν μία διεργασία χρειάζεται μόνον μέρος των διαθέσιμων επεξεργαστών, οι υπόλοιποι μπορούν να αξιοποιηθούν από άλλες διεργασίες.**
- ***Χρήση του βέλτιστου αριθμού επεξεργαστών:*** **Επιλέγοντας το κατάλληλο μέγεθος του υποσυστήματος που θα χρησιμοποιήσουμε, είναι δυνατό να ελαχιστοποιήσουμε το χρόνο εκτέλεσης του προγράμματός μας.**

Κατηγορίες δικτύων διασύνδεσης:

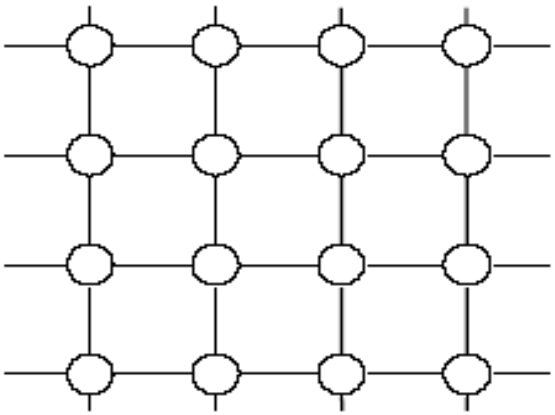
- ***Στατικά ή Δίκτυα ενός σταδίου (static / single stage networks)*: κάθε μήνυμα κατευθύνεται από έναν κόμβο-επεξεργαστή/μεταγωγέα σε έναν άλλον κόμβο-επεξεργαστή/μεταγωγέα μέσω ενός και μοναδικού σταδίου μεταγωγής (single stage of switching).**

- ***Δυναμικά ή Δίκτυα πολλαπλών σταδίων (dynamic / multiple stage networks):* κάθε μήνυμα κατευθύνεται από έναν κόμβο-επεξεργαστή μέσω πολλαπλών σταδίων (συνήθως $log_2 N$) κόμβων-μεταγωγέων σε έναν άλλον κόμβο-επεξεργαστή ή κόμβο-μονάδας μνήμης.**

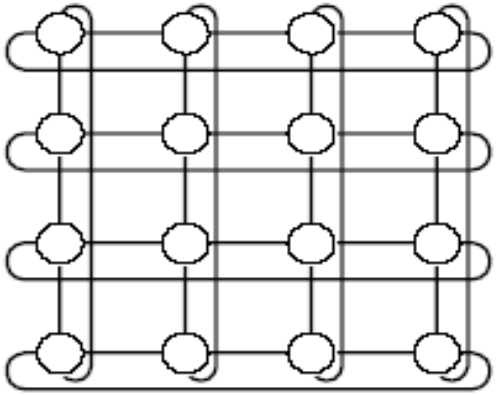

static/direct network
dynamic/indirect network
P
P
P
P
P
P
P
P
Network interface/switch
Processing node
Switching element

Δίκτυα Ενός Σταδίου: Πλέγμα (Mesh)

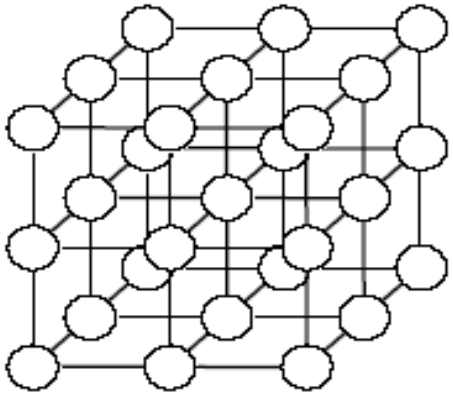
- **Mesh: generalization of linear array to 2D**
  - —nodes have 4 neighbors: north, south, east, and west.
- **k-d mesh:**
  - —d-dimensional mesh
  - —node have *2d* neighbors



2D mesh



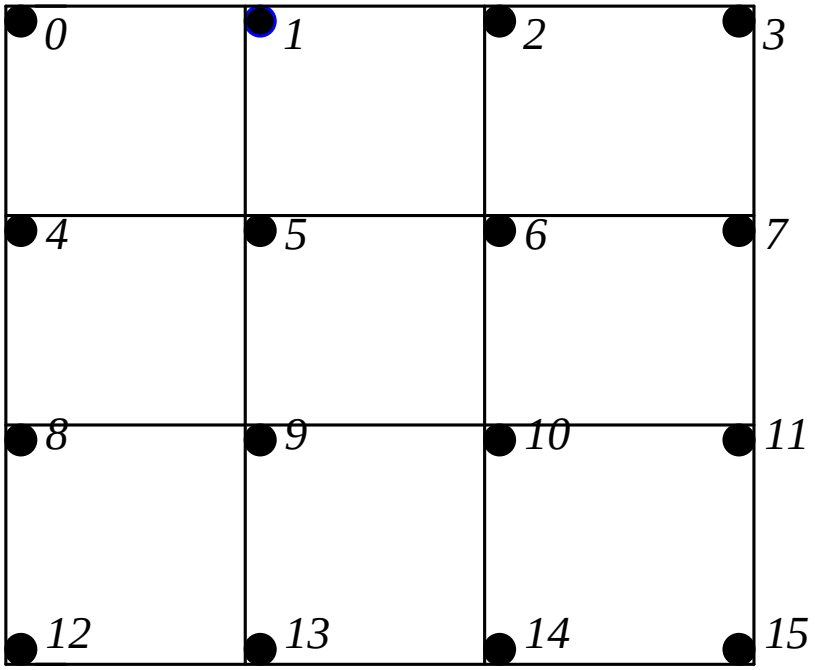
2D torus



3D mesh

Δυσδιάστατο πλέγμα (2D mesh)

N κόμβοι-επεξεργαστές τοποθετημένοι στο επίπεδο σε ένα $\sqrt{N}\times\sqrt{N}$ πίνακα, και κάθε επεξεργαστής επικοινωνεί με τέσσερις γειτονικούς επεξεργαστές.



Δυσδιάστατο πλέγμα (2D mesh)

Διάμετρος **Δυσδιάστατου τετραγωνικού πλέγματος *N* κόμβων:** $2(\sqrt{N}-1)$

Διαμέριση:

- **Ένα τέτοιο πλέγμα διαμερίζεται σε 4 υποπλέγματα *N/4* κόμβων.**
- **Αν *N/4* τέλειο τετράγωνο, τότε κάθε ένα από τα υποπλέγματα έχει όλες τις ιδιότητες του αρχικού πλέγματος.**
- **Η διαμέριση γίνεται χωρίζοντας το αρχικό πλέγμα σε 4 υποπλέγματα και αγνοώντας τις ακμές που συνδέουν τους εξωτερικούς κόμβους των υποπλεγμάτων.**

Δρομολόγηση στο Δυσδιάστατο Πλέγμα

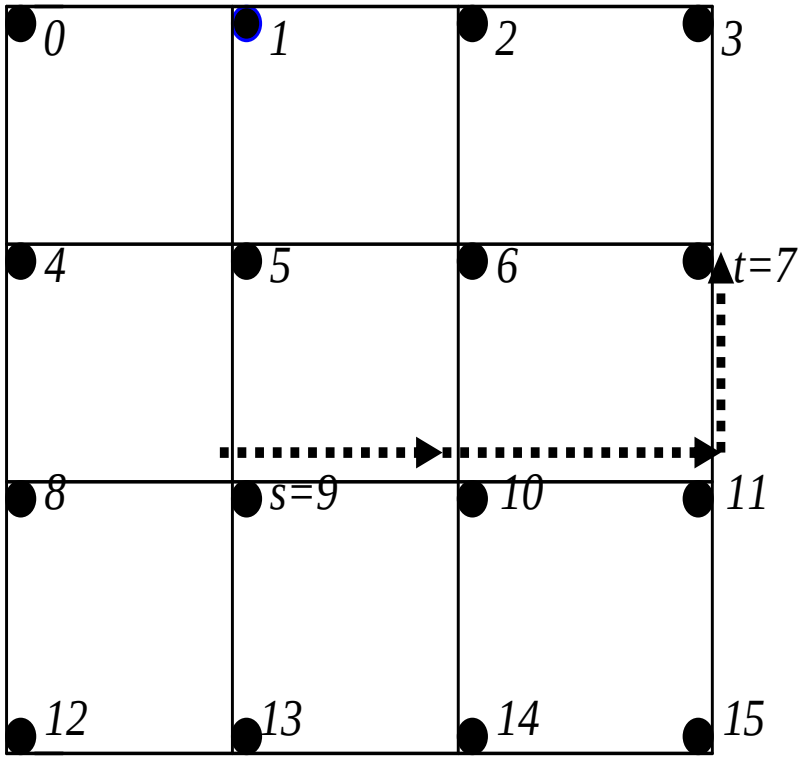
Αλγόριθμος δρομολόγησης μηνύματος από έναν κόμβο *s* σε έναν κόμβο *t* :

1. **Υπολογίζεται ο αριθμός των σειρών $R = \lfloor t/\sqrt{N} \rfloor - \lfloor s/\sqrt{N} \rfloor$ και των στηλών $L = (t \bmod \sqrt{N}) - (s \bmod \sqrt{N})$ που απαιτείται να μετακινηθεί το μήνυμα.**
2. **Οι *R* και *L* αποτελούν την *επικεφαλίδα* του κυρίως μηνύματος και αποστέλλονται από τον *s*.**
3. **Όταν το μήνυμα φθάσει σε έναν κόμβο *v*, ενημερώνεται ο αριθμός *R* ή ο *L*, ανάλογα με το αν το μήνυμα έφθασε στον κόμβο *v* μέσω μιας κατακόρυφης ή οριζόντιας ακμής.**

   Αν το μήνυμα έφθασε στον *v* από τον κόμβο δεξιά (αντίστοιχα, αριστερά) του *v* στο επίπεδο, τότε ο αριθμός *L* αυξάνεται (αντίστοιχα, μειώνεται) κατά *1*.

   Αν το μήνυμα έφθασε στον *v* από τον κόμβο κάτω (αντίστοιχα, επάνω) από τον v στο επίπεδο, τότε ο αριθμός *R* αυξάνεται (αντίστοιχα, μειώνεται) κατά *1*.

<u>Παράδειγμα</u>: Έστω *s* = 9 και *t* = 7. Τότε *R* = ⌊7/4⌋- ⌊9/4⌋= -1 και *L* = 7mod4 - 9mod4 = 2. Επομένως, το μήνυμα θα μετακινηθεί μία σειρά προς τα επάνω και δύο στήλες δεξιά.

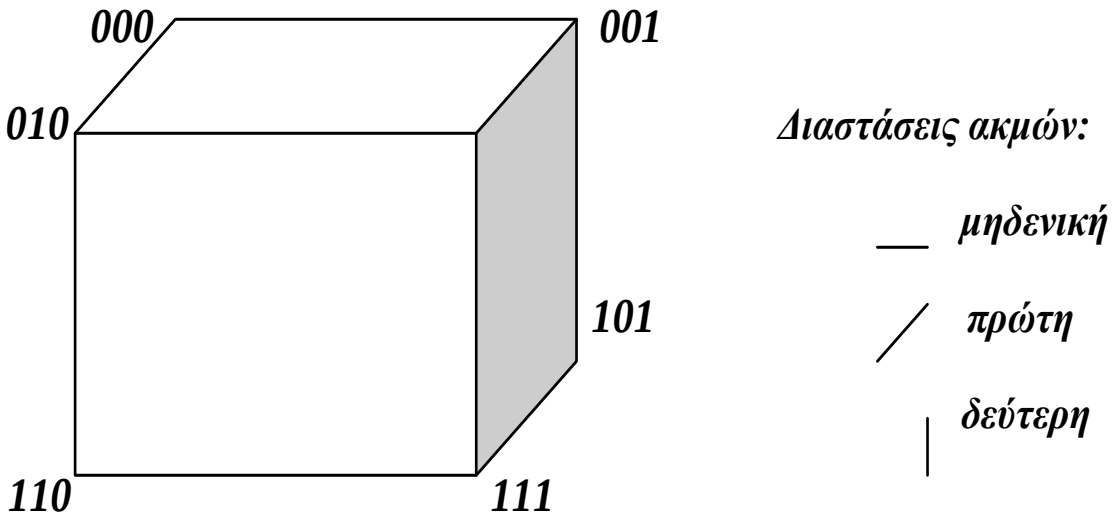


Δίκτυα Ενός Σταδίου: Υπερκύβος (Hypercube)

Εάν ο αριθμός των κόμβων N = 2^n, τότε ο αριθμός *n* είναι η διάσταση του υπερκύβου.

Η δυαδική αναπαράσταση του κωδικού αριθμού κάθε κόμβου αποτελείται από *n* bits.

Στον υπερκύβο υπάρχουν ακμές μεταξύ ζευγών κόμβων των οποίων οι δυαδικές αναπαραστάσεις διαφέρουν σε ένα και μόνον bit. Μία ακμή, η οποία συνδέει κόμβους που διαφέρουν στο *i*-οστό bit, καλείται *ακμή διάστασης i*.

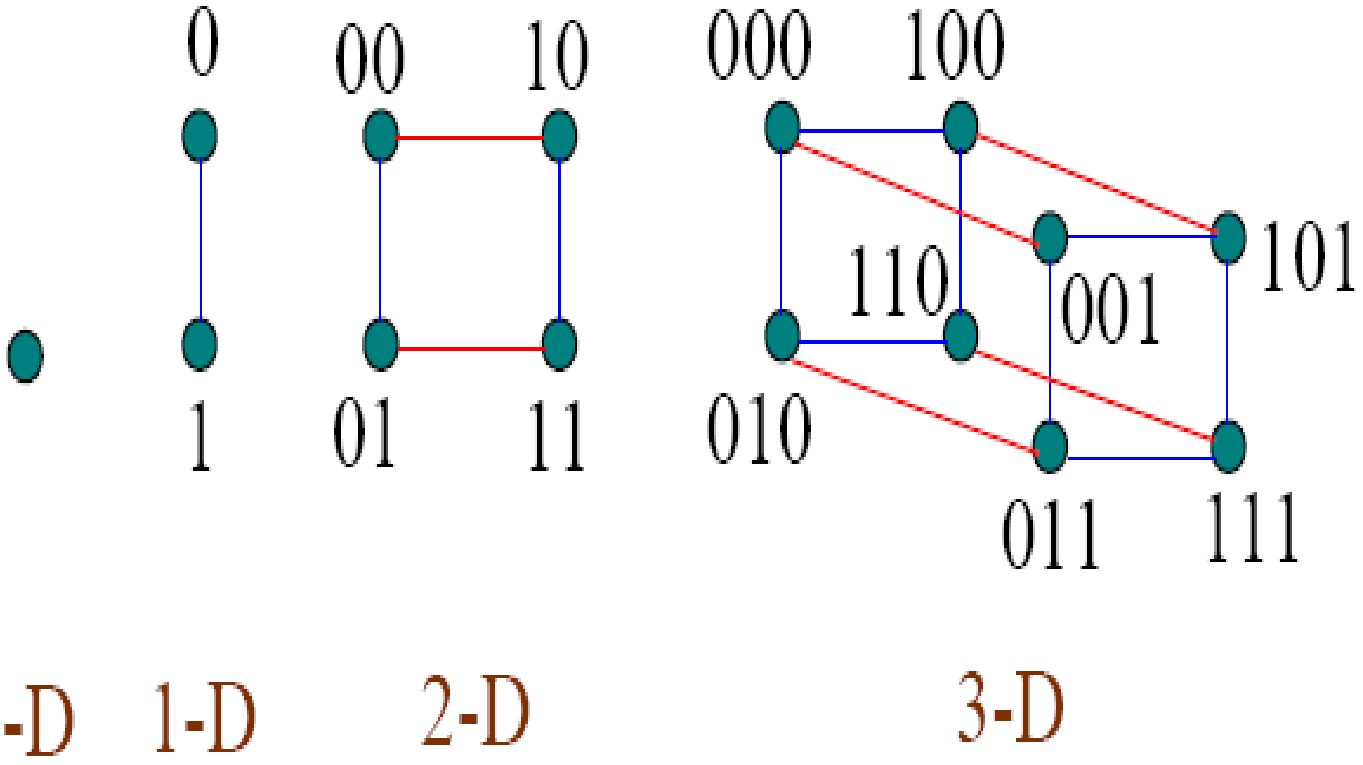


Αναδρομική Κατασκευή Υπερκύβου

Ένας υπερκύβος διάστασης *i* κατασκευάζεται αναδρομικά ως εξής:

1. **Θεωρούμε δύο υπερκύβους διάστασης *i-1* με αριθμημένους τους κόμβους τους.**
2. **Προσθέτουμε ακμές μεταξύ των κόμβων στους δύο υπερκύβους διάστασης *i-1* που έχουν τους ίδιους κωδικούς αριθμούς.**
3. **Προσθέτουμε ως το πιο σημαντικό bit, στους αριθμούς του ενός υπερκύβου διάστασης *i-1* το *0* και στους αριθμούς του άλλου υπερκύβου διάστασης *i-1* το *1*.**

Αναδρομική Κατασκευή Υπερκύβου



Διάμετρος:

Η διάμετρος υπερκύβου διάστασης *n* ισούται με *n*.

***Απόσταση Hamming*:**

- **Η ελάχιστη απόσταση μεταξύ δύο κόμβων ενός υπερκύβου, ήτοι το μήκος του μικρότερου μονοπατιού μεταξύ των δύο αυτών κόμβων.**
- **Η απόσταση αυτή ισούται με τον αριθμό των bits στα οποία διαφέρουν οι κωδικοί αριθμοί των κόμβων αυτών**

Δρομολόγηση στον Υπερκύβο

Αλγόριθμος δρομολόγησης μηνυμάτων από έναν κόμβο *s* σε έναν κόμβο *t* :

1. **Υπολογίζεται το *αποκλειστικό ή* (⊕) των κωδικών αριθμών των κόμβων *s* και *t*. Το *i*-οστό bit του δυαδικού αριθμού $d = s \oplus t$ θα είναι ίσο με *1* αν τα *i*-οστά bits των *s* και *t* διαφέρουν, και *0* αν είναι ίσα. Ο αριθμός *d* αποστέλλεται μαζί με το μήνυμα.**
2. **Κάθε επεξεργαστής που λαμβάνει ένα μήνυμα του οποίου το *d* έχει κάποιο bit ίσο με *1*, έστω στην *i*-οστή θέση, αλλάζει αυτό το bit σε *0* και στέλνει το μήνυμα μέσω της ακμής διάστασης *i*. Ο κόμβος που λαμβάνει μήνυμα του οποίου το *d* δεν έχει κανένα bit ίσο με *1* *(d=0)*, είναι ο κόμβος προορισμού *t*.**

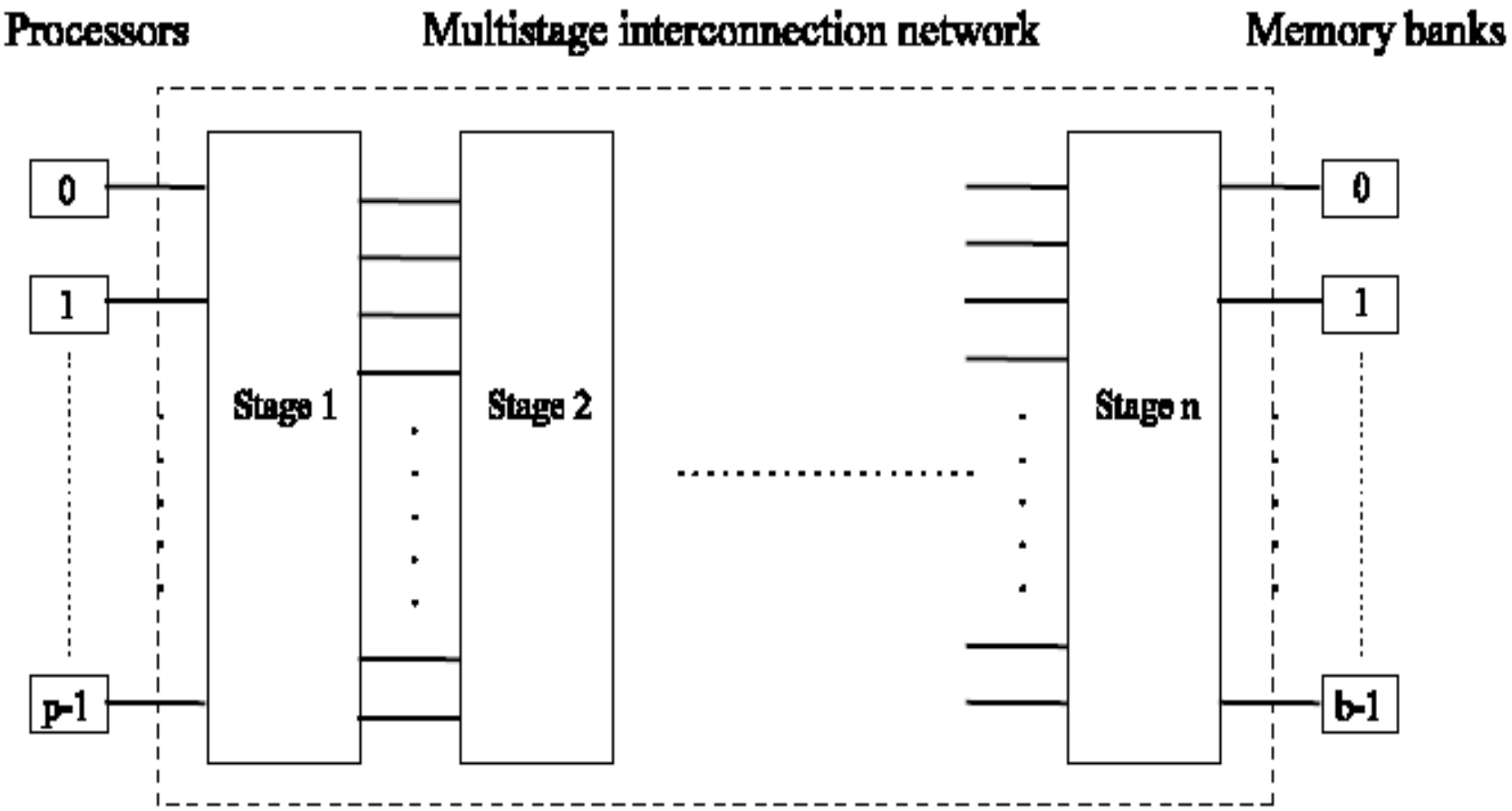
Δρομολόγηση στον Υπερκύβο

Παράδειγμα δρομολόγησης μηνύματος:

- ***s = 1111* και *t=0110***
- **Τότε, *d = s⊕t = 1001***
- **Ο *s* θέτει το μηδενικό bit του *d* ίσο με το *0* και στέλνει το μήνυμα μέσω της ακμής διάστασης *0* στον *1110*.**
- **Ο *1110* θέτει το τρίτο bit του *d* ίσο με το *0* και στέλνει το μήνυμα μέσω της ακμής διάστασης τρία, στον κόμβο *0110*. Ο κόμβος *0110*, ο οποίος είναι ο προορισμός, λαμβάνει το μήνυμα με *d = 0*.**

Εναλλακτικά, θα μπορούσε ο *s* να θέσει το τρίτο bit ίσο με το *0* και να στείλει το μήνυμα στον *0111* μέσω της ακμής διάστασης τρία, και ο *0111* να θέσει το μηδενικό bit ίσο με το *0* και να στείλει το μήνυμα στον *0110*, ο οποίος είναι ο κόμβος-προορισμός.

Δίκτυα Πολλαπλών Σταδίων (multiple stage networks)



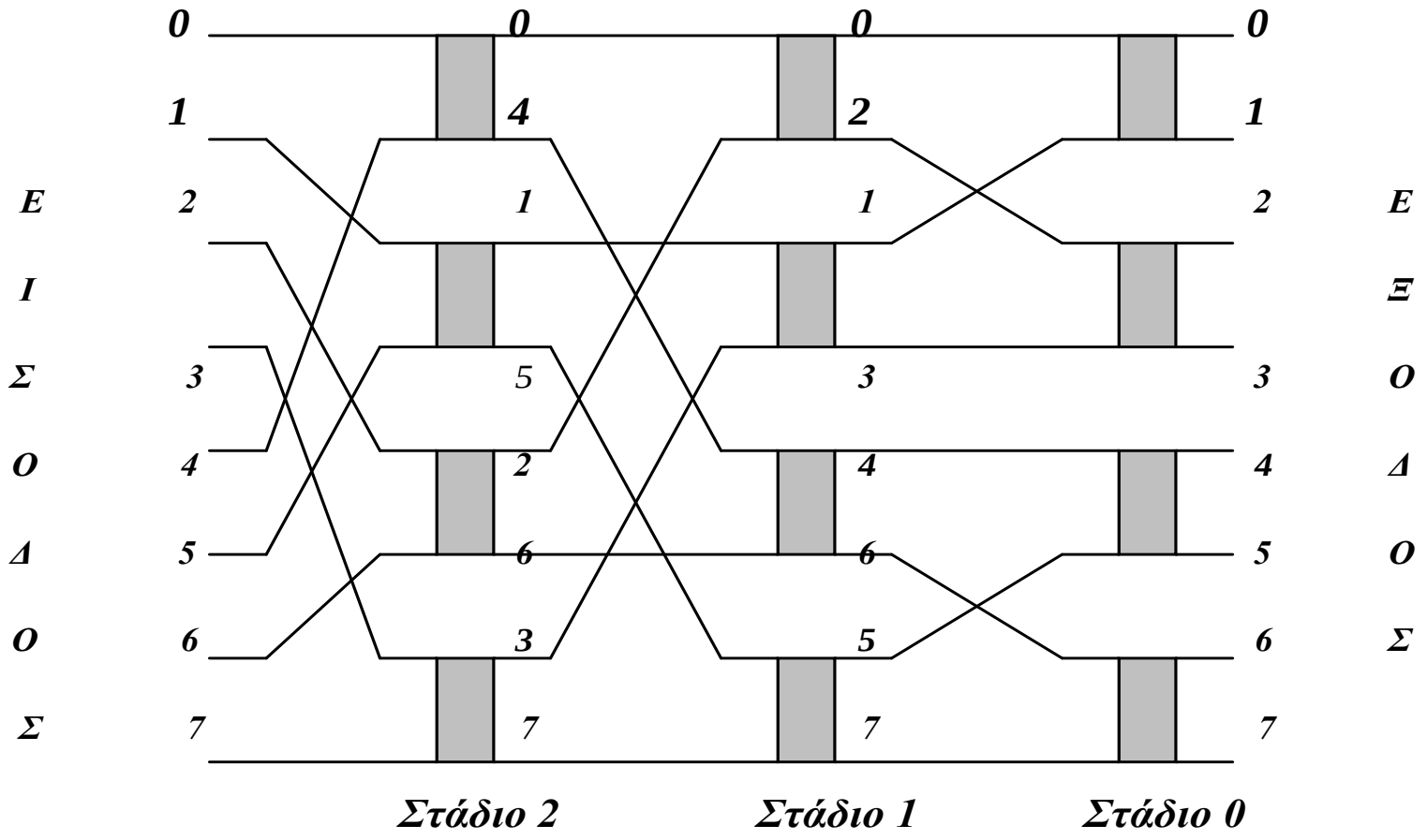
Δίκτυα Πολλαπλών Σταδίων (multiple stage networks)

- ***Κύβος πολλαπλών σταδίων (multistage cube),***
- ***Δενδρικά δίκτυα (tree networks).***

Κύβος πολλαπλών σταδίων: **Δίκτυα με ισοδύναμη τοπολογία:**

- ***δίκτυο πεταλούδας (butterfly network)***
- ***shuffle-exchange πολλαπλών σταδίων***
- ***γενικευμένος κύβος (generalized cube)***
- ***baseline***
- ***omega***

Κύβος Πολλαπλών Σταδίων *N* εισόδων και *N* εξόδων για *N=8*

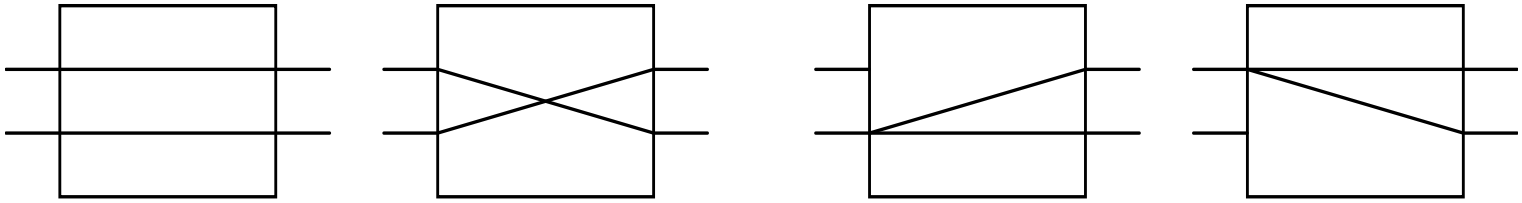


Κύβος Πολλαπλών Σταδίων

- **Οι *N* είσοδοι και οι *N* έξοδοι του δικτύου είναι οι *N* επεξεργαστές.**
- **Ένα τέτοιο δίκτυο έχει $n = log_2 N$ στάδια αριθμημένα από το *0* έως το *N-1*.**
- **Κάθε στάδιο αποτελείται από *N* ακμές, αριθμημένες από το *0* έως το *N-1*, οι οποίες συνδέονται σε *N/2 κουτιά εναλλαγής (interchange boxes)*.**
- **Σε κάθε στάδιο *i*, οι αριθμοί των δύο ακμών που αποτελούν τις δύο εισόδους (αλλά και τις δύο εξόδους) σε κάθε κουτί εναλλαγής, διαφέρουν μόνο στο *i*-οστό bit τους.**

Κύβος Πολλαπλών Σταδίων

Κάθε κουτί εναλλαγής είναι ένας μεταγωγέας δύο εισόδων και δύο εξόδων, ο οποίος μπορεί να τεθεί σε μια από τις εξής 4 καταστάσεις:



ευθεία (straight)

αντιστροφή (exchange)

μετάδοση κάτω εισόδου (lower broadcast)

μετάδοση πάνω εισόδου (upper broadcast)

Δρομολόγηση στον Κύβο Πολλαπλών Σταδίων

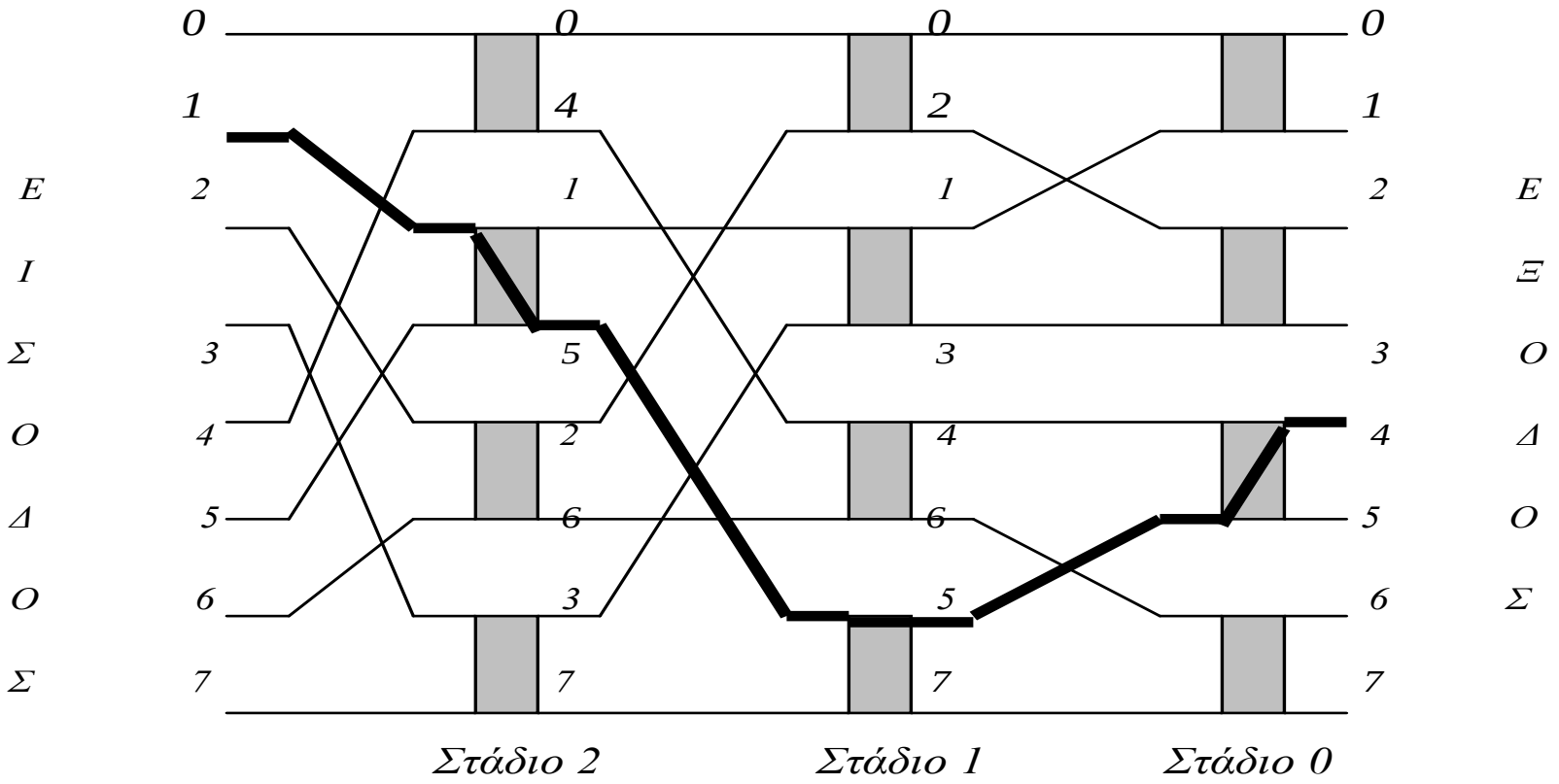
Αλγόριθμος δρομολόγησης μηνυμάτων από έναν επεξεργαστή-αφετηρία $s = s_{n-1}s_{n-2}\ldots s_1s_0$ σε έναν επεξεργαστή-προορισμό

$t=t_{n-1}t_{n-2}\ldots t_1t_0$:

Αν $s_i=t_i$ τότε ο μεταγωγέας του σταδίου i τίθεται στην κατάσταση «ευθεία».

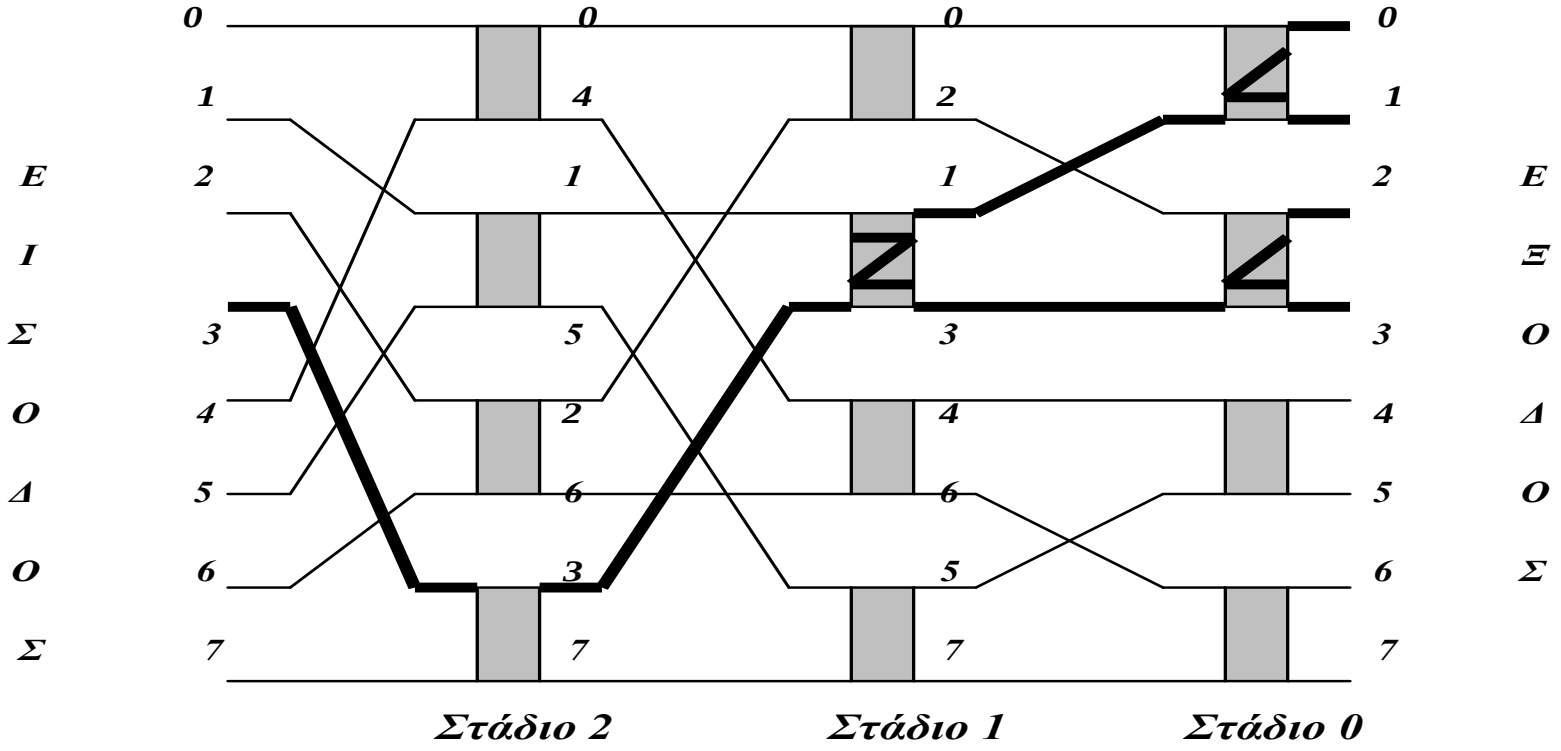
Αν $s_i\neq t_i$ τότε ο μεταγωγέας του σταδίου i τίθεται στην κατάσταση «αντιστροφή».

Παράδειγμα: s=001 και t=100



Το μονοπάτι από τον κόμβο *s=001* στον κόμβο *t=100*

Σύνδεση «εκπομπής»: **δημιουργείται όταν κάποιοι μεταγωγείς τίθενται στις καταστάσεις «μετάδοση κάτω εισόδου» ή/και «μετάδοση πάνω εισόδου».**



Μονοπάτι εκπομπής από *s=011* στις εξόδους *000, 001, 010, 011*

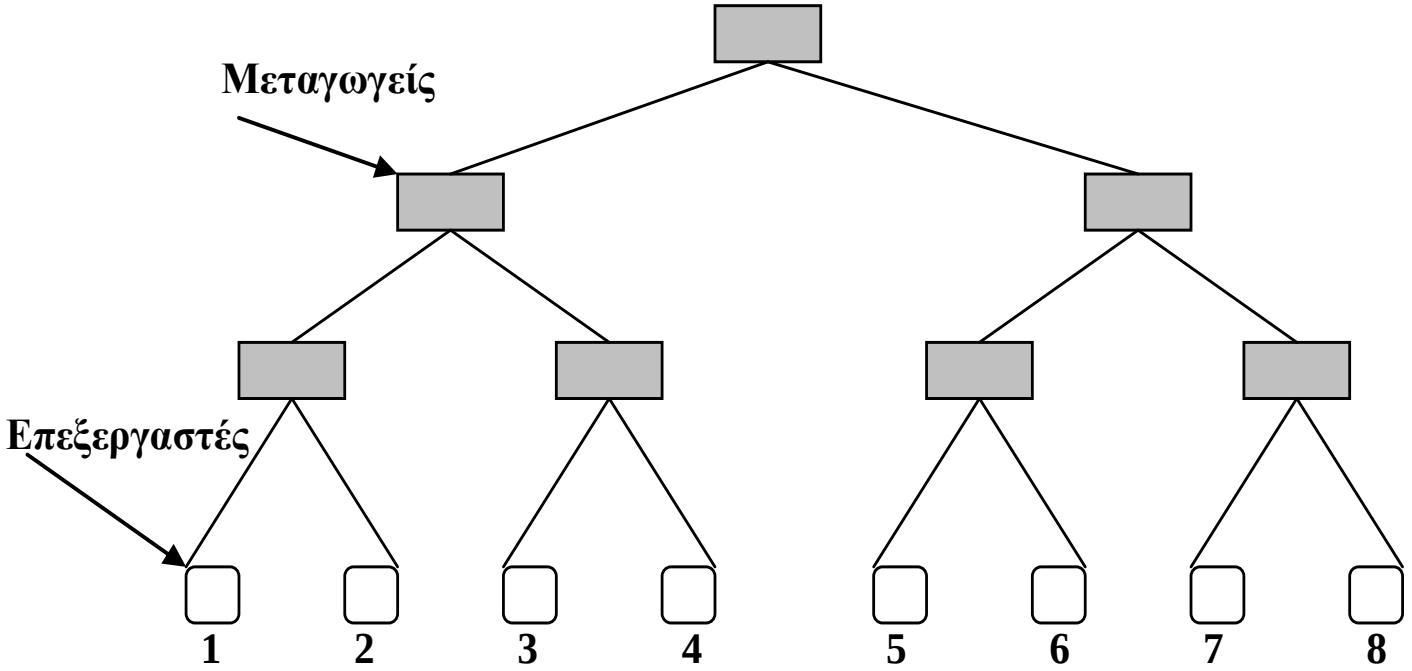
Δενδρικά Δίκτυα

Η τοπολογία των *δενδρικών δικτύων* έχει δομή δένδρου του οποίου τα φύλλα αντιπροσωπεύουν επεξεργαστές και οι εσωτερικοί κόμβοι αντιπροσωπεύουν μεταγωγείς.

- ***δίκτυα δυαδικού δένδρου (binary tree networks)***
- ***δίκτυα υπερδένδρου (hypertree networks).***

***Δίκτυο δυαδικού δένδρου* με $N=2^n$ κόμβους-επεξεργαστές: έχει τη μορφή *πλήρους δυαδικού δένδρου* βάθους n.**

- **Κάθε εσωτερικός κόμβος-μεταγωγέας *v* έχει τρεις πύλες εισόδου/εξόδου. Η μία είναι συνδεδεμένη με τον πατέρα του κόμβου *v* ενώ οι δύο άλλες με τα δύο παιδιά του *v*.**



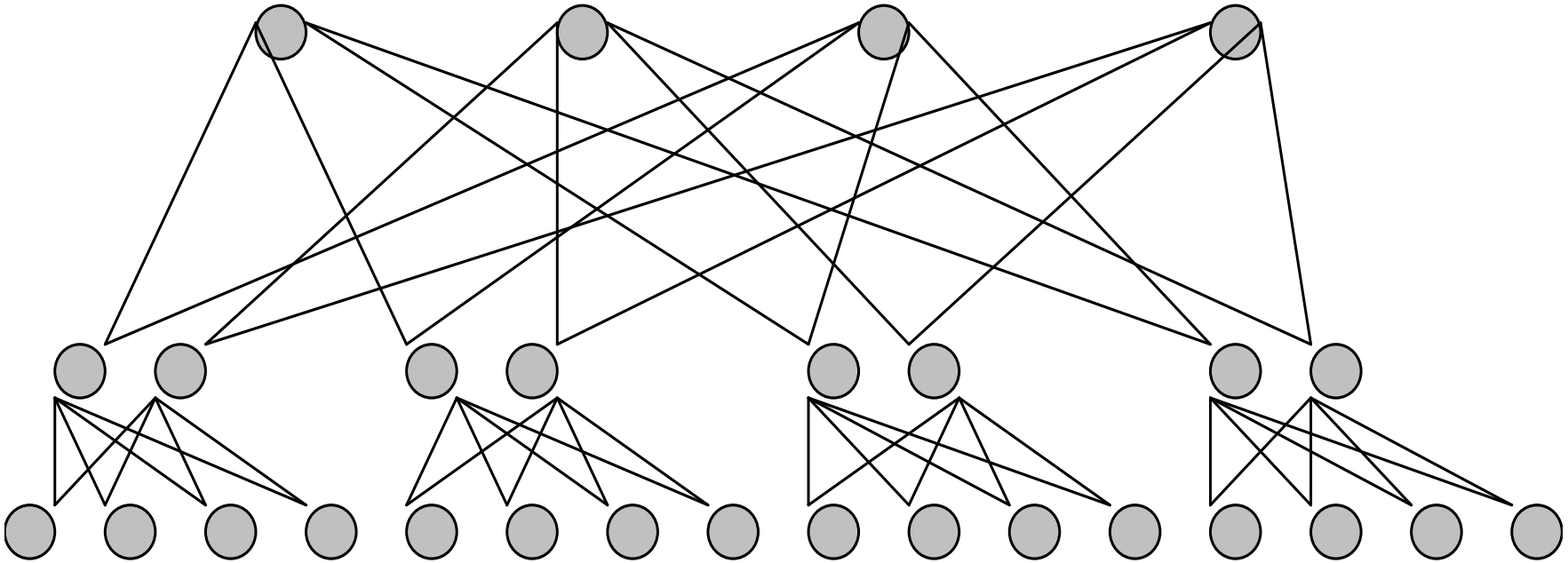
Η Διάμετρος **δικτύου δυαδικού δένδρου με $N=2^n$ κόμβους είναι 2n.**

Διαμερίζεται **σε δύο υποδίκτυα με 2^{n-1} κόμβους.**

Δρομολόγηση ενός μηνύματος από έναν κόμβο s σε έναν κόμβο t:

- **Ετικέτα δρομολόγησης: ο κωδικός αριθμός του κόμβου t.**
- **Το μήνυμα δρομολογείται πρώτα στην ρίζα r του υποδένδρου που περιέχει τους δύο κόμβους s και t, και κατόπιν δρομολογείται από τον r στον t.**
- **Ο έλεγχος της δρομολόγησης κατανέμεται στο δίκτυο.**

***Δίκτυα υπερδένδρου βαθμού (degree) k* και *βάθους (depth) d*: κάθε κόμβος έχει *k* παιδιά ενώ το δίκτυο έχει k^d φύλλα-επεξεργαστές και διάμετρο *2d*.**



Σχήμα:* Δίκτυο υπερδένδρου βαθμού *4* και βάθους *2

Fat Tree