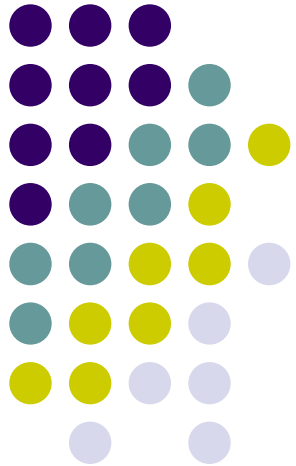
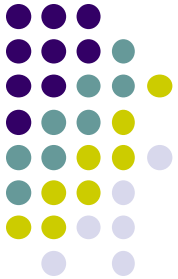
Κατανεμημένα Συστήματα



Βιβλιογραφία

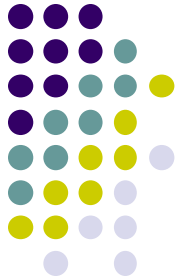
- Κατανεμημένα Συστήματα, A. Tanenbaum and M. Steen, εκδ. Κλειδάριθμος, 2006.
- Κατανεμημένα Συστήματα με Java, Ι. Κάβουρας, Γ. Ξυλωμένος, Γ. Μήλης, Κ. Ρουκουνάκης, εκδ. Κλειδάριθμος, 2008.

ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ



- **Ορισμός /Χαρακτηριστικά Κατανεμημένων Συστημάτων**
- **Οργάνωση Κατανεμημένων Συστημάτων**
  - **Το Μοντέλο Πελάτη-Διακομιστή**
  - **Ομότιμα Συστήματα (P2P Systems)**
- **Ενδιάμεσο Λογισμικό (Middleware)**
  - **Κλήση Απομακρυσμένων Διαδικασιών (RPC)**
  - **Κλήση απομακρυσμένων μεθόδων (RMI)**
- **Θέματα Κατανεμημένων Συστημάτων**

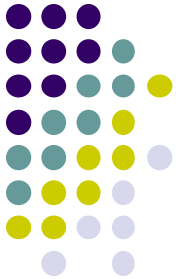
Ορισμός Κατανεμημένου Συστήματος



- Κατανεμημένο σύστημα είναι ένα σύνολο από ανεξάρτητους υπολογιστές το οποίο παρουσιάζεται στους χρήστες σαν ένα λογικό ενιαίο σύστημα.
- Ο ορισμός έχει 2 πλευρές:
  - Το **υλικό** - τα μηχανήματα είναι αυτόνομα
  - Το **λογισμικό** - οι χρήστες θεωρούν ότι έχουν να κάνουν με ένα ενιαίο σύστημα. Οι διαφορές των υπολογιστών μεταξύ τους ή ο τρόπος με τον οποίο επικοινωνούν δεν απασχολούν τους χρήστες

- "Γνωρίζεις ότι έχεις ένα κατανεμημένο σύστημα όταν η δυσλειτουργία ενός αγνώστου συστήματος δεν σου επιτρέπει να κάνεις τη δουλειά σου" – Leslie Lamport

Χαρακτηριστικά Κατανεμημένων Συστημάτων



- Οι **διαφορές μεταξύ των διαφόρων υπολογιστών** και οι τρόποι με τους οποίους αυτοί επικοινωνούν παραμένουν κρυφοί για τους χρήστες
- Το ίδιο ισχύει και για την **εσωτερική οργάνωση** των κατανεμημένων συστημάτων.
- Οι χρήστες και οι εφαρμογές μπορούν να αλληλεπιδρούν με ένα κατανεμημένο σύστημα με **συνεπή και ομοιόμορφο τρόπο**, ανεξάρτητα από το που και πότε πραγματοποιείται η αλληλεπίδραση
- Επιτρέπουν με σχετική **ευκολία την επέκτασή τους** ή την προσαρμογή της κλίμακας του μεγέθους τους

Τομείς Εφαρμογής

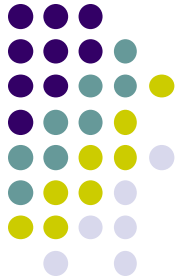
Ιατρική, βιολογία και γενετική
(διάγνωση νοσημάτων μέσω επεξεργασίας ιατρικών εικόνων)

Αγορές χρήματος
(χρηματοοικονομικές αναλύσεις)

Επιστημονικό και Εκπαιδευτικό κλάδο
(ερευνητικά ινστιτούτα, δημόσια έρευνα)

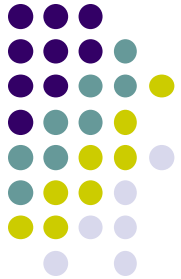
Βιομηχανικό και Κατασκευαστικό τομέα
(αυτοκινητοβιομηχανία)

Παραδείγματα



- Διαδίκτυο, παγκόσμιος ιστός, κινητή τηλεφωνία.
- Τηλε* (εκπαίδευση, ιατρική, συνεργασία, ...).
- Ενδο-και δια-επιχειρησιακά συστήματα (όμιλος εταιρειών, τράπεζες, ηλεκτρονικό επιχειρείν, ...).
- Αυτόματος έλεγχος(παραγωγή, μεταφορές, ...).
- Ενοποιημένοι υπολογιστικοί και αποθηκευτικοί πόροι σε παγκόσμια κλίμακα (grid computing).
- Ομότιμα συστήματα υπηρεσιών (peer-to-peer).
- Δίκτυα αισθητήρων (sensor networks).
- Έξυπνα κτήρια, συστήματα και περιβάλλοντα αλληλεπίδρασης (pervasive/ubiquitous computing).

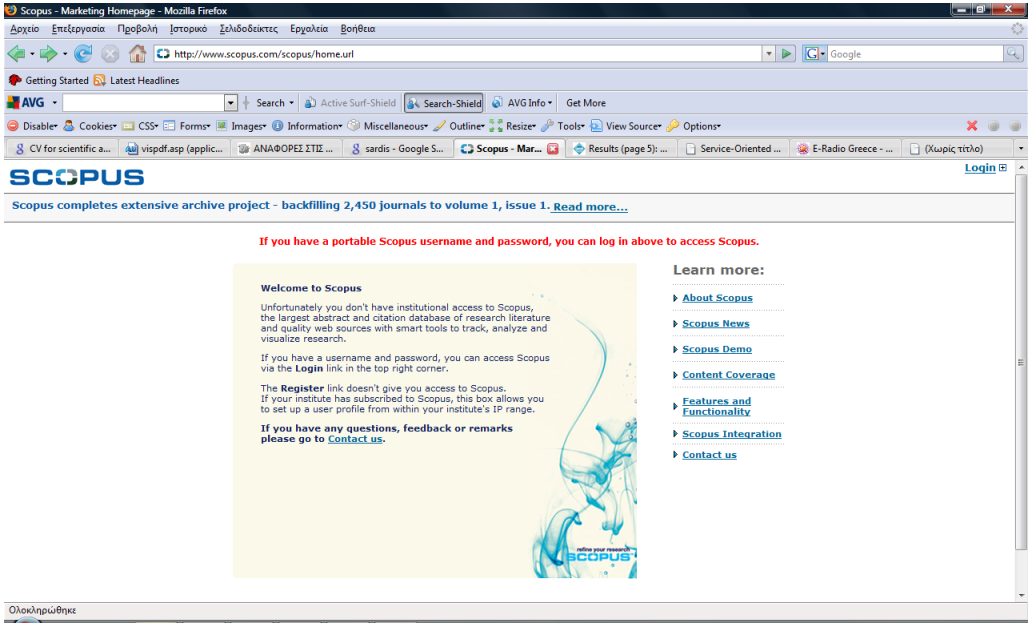
Εφαρμογή: Ο Παγκόσμιος Ιστός



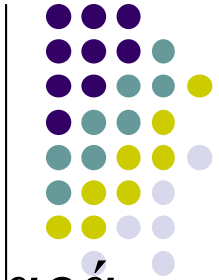
- Αποτελεί ένα τεράστιο σύστημα πληροφοριών βασιζόμενο σε έγγραφα, όπου κάθε έγγραφο έχει ένα δικό του όνομα με τη μορφή μιας διεύθυνσης URL
- Από **λογική άποψη** φαίνεται σαν να υπάρχει ένας και μόνο διακομιστής
- Από **φυσική άποψη** ο Ιστός είναι κατανεμημένος σε έναν τεράστιο αριθμό διακομιστών, κάθε ένας από τους οποίους χειρίζεται έναν αριθμό εγγράφων του Ιστού.
- Το όνομα του διακομιστή που χειρίζεται ένα έγγραφο είναι **κωδικοποιημένο** μέσα στη διεύθυνση URL αυτού του εγγράφου

Εφαρμογή: Ο Παγκόσμιος Ιστός

- **Ο Παγκόσμιος Ιστός (WWW) παρέχει ένα απλό και ομοιόμορφο μοντέλο κατανεμημένων εγγράφων.**
- **Για να δει ο χρήστης ένα έγγραφο χρειάζεται μόνο να ενεργοποιήσει μια αναφορά, και το έγγραφο εμφανίζεται στην οθόνη.**
- **Η δημοσίευση ενός εγγράφου είναι απλή. Χρειάζεται μόνο να δώσουμε ένα μοναδικό όνομα με την μορφή μιας διεύθυνσης URL η οποία παραπέμπει σε ένα τοπικό αρχείο που περιέχει το περιεχόμενο του εγγράφου.**



Κατανεμημένη επεξεργασία



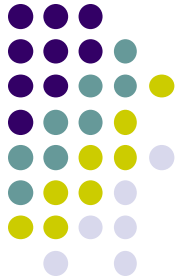
- Πολλοί υπολογιστικοί κόμβοι συνεργάζονται χαλαρά (*loosely coupled* nodes)
- Στόχος η κατανομή υπηρεσιών, εφαρμογών, επεξεργαστικής ισχύος, ή δεδομένων
- Δεύτερος στόχος είναι η ταχύτητα
- Η επικοινωνία δεν είναι συνήθως συχνή
- Παρότι οι κόμβοι είναι ετερογενείς το δίκτυο είναι τυποποιημένο

Παράδειγμα: Διαχείριση Αυτόματων Ταμειακών Μηχανών (ATM) μιας τράπεζας

Παράλληλα - Κατανεμημένα Συστήματα

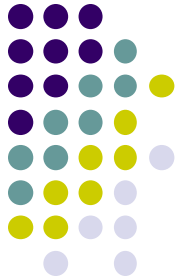
	Παράλληλα Συστήματα	Κατανεμημένα Συστήματα
Στόχος	Επιτάχυνση	1. Τοπολογική κατανομή εφαρμογών, επεξεργαστών ή δεδομένων 2. Ταχύτητα
Δίκτυο	Έμφαση στην ταχύτητα	Ευρείας χρήσης (π.χ. ISDN, Ethernet)
Εφαρμογές	Επιστημονικές εξομοιώσεις	Υπηρεσίες, Βάσεις δεδομένων
Αποστάσεις	<1m	>1m συχνά >1Km
Τοπολογία	Γεωμετρική	Αυθαίρετη
Κόμβοι	Ομοιογενείς	Πιθανώς ετερογενείς

Στόχοι Κατανεμημένων Συστημάτων



- Να συνδέει εύκολα τους χρήστες με τους πόρους
- Να κρύβει το γεγονός ότι οι πόροι είναι κατανεμημένοι σε δίκτυο (transparency)
- Να υποστηρίζει ανοικτές αρχιτεκτονικές (openness)
- Να είναι επεκτάσιμο (scalability)
  - στο μέγεθος
  - γεωγραφικά
  - διαχειριστικά
  - …

Συνδεσιμότητα



Σκοπός

- **εύκολη προσπέλαση απομακρυσμένων πόρων**
- **να επιτρέπει τον διαμοιρασμό αυτών των πόρων με πολλούς χρήστες με ένα ελεγχόμενο τρόπο**
- **Διευκόλυνση της συνεργασίας (computer-supported cooperative work/games), π.χ. groupware, e-commerce**

<u>Πρόβλημα:</u> Η αυξανόμενη συνδεσιμότητα μπορεί να οδηγήσει

- **σε ανεπιθύμητη επικοινωνία (π.χ. Spam)**
- **προβλήματα ασφάλειας και πρόσβασης σε προσωπικά δεδομένα**

Διαφάνεια

Ένα κατανεμημένο σύστημα που μπορεί να εμφανίζεται στους χρήστες και τις εφαρμογές σαν ένα ενιαίο υπολογιστικό σύστημα χαρακτηρίζεται ως διαφανές (transparent)

Σκοπός:

- **Απόκρυψη του γεγονότος ότι οι διεργασίες και οι πόροι του είναι από φυσική άποψη κατανεμημένοι σε πολλούς υπολογιστές.**

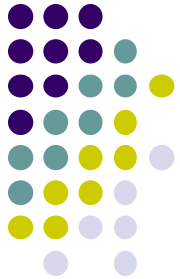
Η διαφάνεια δεν είναι πάντα εφικτή ή εύκολα υλοποιήσιμη. Εξαρτάται από την αρχιτεκτονική του ΚΣ και την απόδοση του συστήματος, π.χ.

- **αντίγραφα βάσεων δεδομένων,**
- **ταχύτητα μετάδοσης σήματος,**
- **…**

Μορφές Διαφάνειας

Μορφή Διαφάνειας	Περιγραφή
Προσπέλασης (Access)	Απόκρυψη των διαφορών στην αναπαράσταση των δεδομένων και στον τρόπο πρόσβασης ενός κόμβου (π.χ. διαφορετικές συμβάσεις ονομασίας αρχείων σε Λ.Σ.)
Θέσης (Location)	Απόκρυψη της τοποθεσίας του κόμβου (απόδοση μόνο λογικών ονομάτων στους κόμβους)
Μετανάστευσης (Migration)	Απόκρυψη του γεγονότος ότι ένας κόμβος μπορεί να αλλάξει τοποθεσία
Μετακόμισης (Relocation)	Απόκρυψη του γεγονότος ότι ένας κόμβος μπορεί να αλλάξει τοποθεσία ενώ χρησιμοποιείται
Αναπαραγωγής (Replication)	Απόκρυψη του γεγονότος ότι υπάρχουν πολλά αντίγραφα ενός πόρου
Αστοχιών (Failure)	Απόκρυψη της αποτυχίας και ανάκτησης ενός κόμβου
Διατήρησης (Persistence)	Απόκρυψη της θέσης του λογισμικού (στη μνήμη ή στο δίσκο)

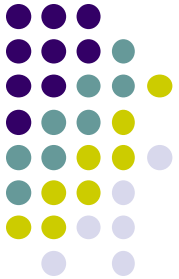
Ανοικτή Αρχιτεκτονική



Αποτελεί σημαντικό παράγοντα:

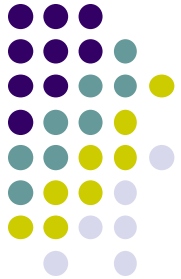
- **Διαλειτουργικότητας (interoperability):** **Χαρακτηρίζει το κατά πόσον δύο υλοποιήσεις συστημάτων ή στοιχείων που προέρχονται από διαφορετικούς κατασκευαστές μπορούν να συνυπάρχουν και να συνεργάζονται βασιζόμενες απλώς η μία στις υπηρεσίες της άλλης, όπως αυτές καθορίζονται από ένα κοινό πρότυπο.**
- **Φορητότητας (portability):** **Χαρακτηρίζει το κατά πόσον μια εφαρμογή που έχει αναπτυχθεί για ένα κατανεμημένο σύστημα Α μπορεί να εκτελεστεί χωρίς τροποποιήσεις σε ένα διαφορετικό κατανεμημένο σύστημα Β, το οποίο υλοποιεί τις ίδιες διασυνδέσεις με το Α.**
- **Ευελιξίας (flexibility):**
  - **δυνατότητα ένα σύστημα να διαμορφωθεί από διαφορετικά στοιχεία υλοποιημένα από διαφορετικούς κατασκευαστές**
  - **Το σύστημα είναι μία συλλογή επιμέρους στοιχείων τα οποία μπορούν εύκολα να προσαρμοστούν ή να αντικατασταθούν**

Επεκτασιμότητα (Scalability)



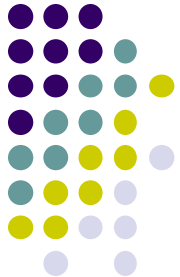
- **Η επεκτασιμότητα αποτελεί έναν από τους σημαντικότερους σχεδιαστικούς στόχους για τους κατασκευαστές κατανεμημένων συστημάτων**
- **Η επεκτασιμότητα μετριέται σε τρεις διαστάσεις:**
  - **το μέγεθος: μπορούμε να προσθέσουμε στο σύστημα περισσότερους χρήστες και πόρους**
  - **γεωγραφική επεκτασιμότητα: οι χρήστες και οι πόροι είναι δυνατόν να βρίσκονται σε μεγάλες αποστάσεις μεταξύ τους**
  - **διαχειριστική επεκτασιμότητα: η διαχείριση του συστήματος μπορεί να παραμένει εύκολη ακόμα και αν αυτό εκτείνεται σε πολλούς διαχειριστικά ανεξάρτητους οργανισμούς.**

Οργάνωση Κατανεμημένων Συστημάτων



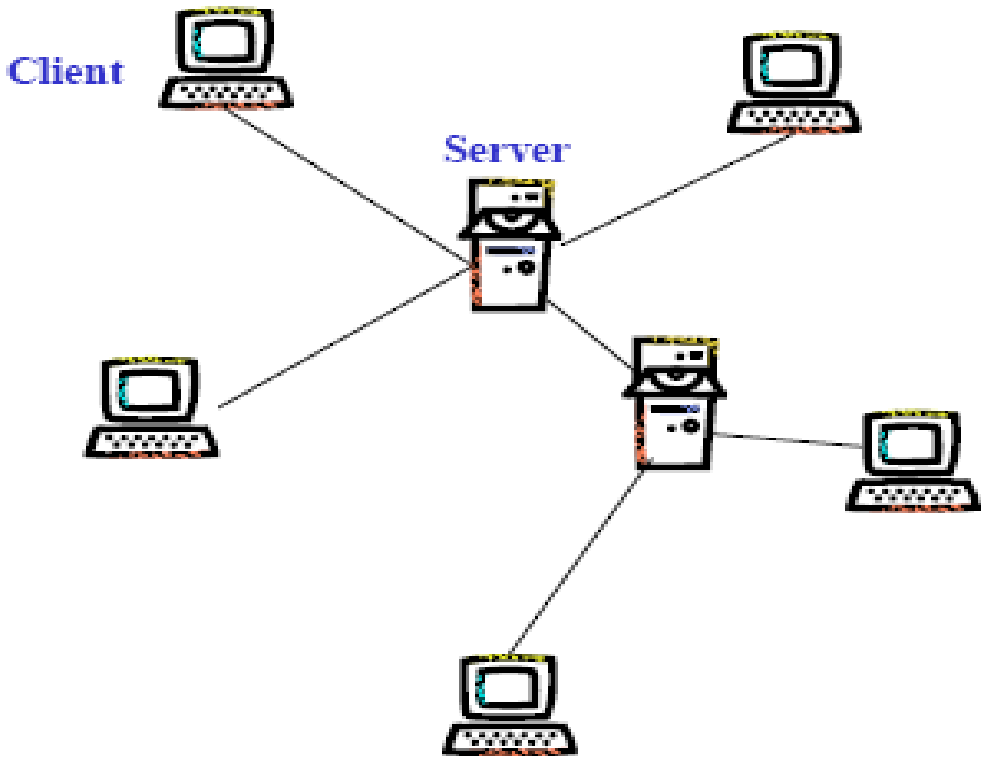
- Το Μοντέλο Πελάτη-Διακομιστή
- Ομότιμα Συστήματα

Το Μοντέλο Πελάτη-Διακομιστή

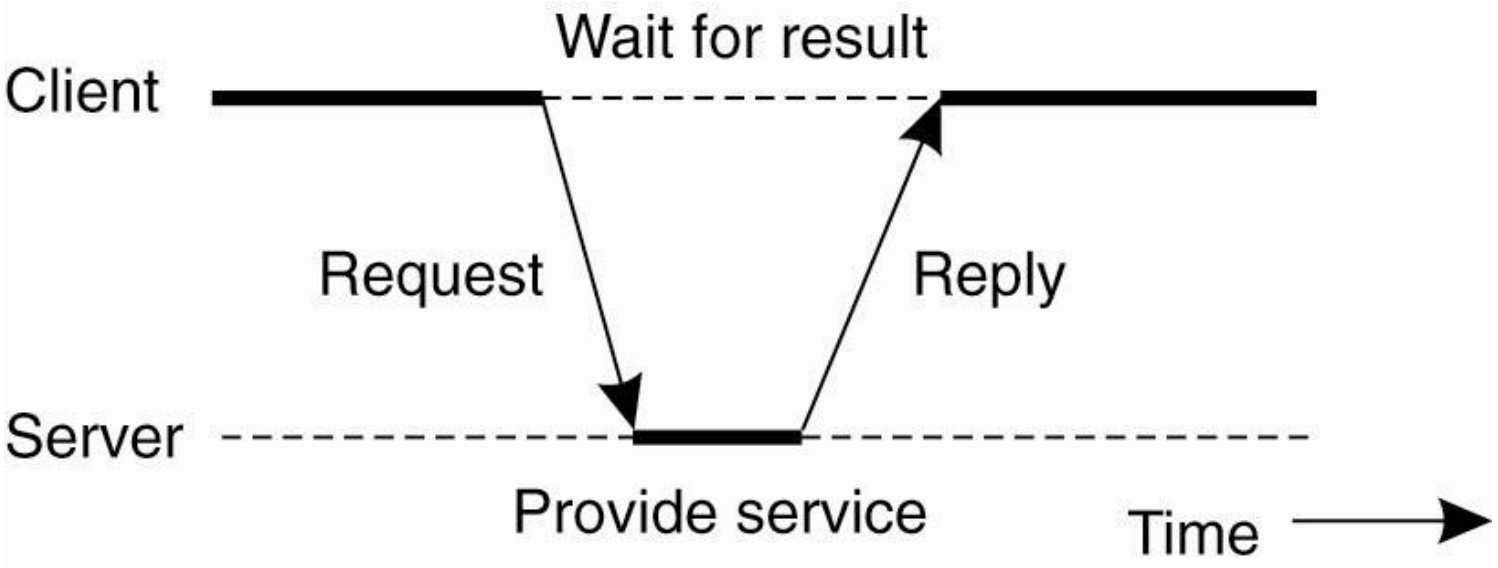


- Σε ένα σύστημα client-server το δίκτυο ενώνει διάφορους υπολογιστικούς πόρους, ώστε οι clients (front end) να μπορούν να ζητούν υπηρεσίες από έναν server (back end) που προσφέρει πληροφορίες ή πρόσθετη υπολογιστική ισχύ
- Το client-server μοντέλο βασίζεται, συνήθως, σε ένα απλό πρωτόκολλο αίτησης/απάντησης (request/reply).
  - Ο πελάτης στέλνει ένα μήνυμα (αίτηση) ζητώντας από τον εξυπηρετητή κάποια υπηρεσία.
  - Ο εξυπηρετητής ενεργοποιείται άμεσα ή προσθέτει την αίτηση σε μια ουρά. Τελικά αποκρίνεται εκτελώντας μια σειρά από ενέργειες και επιστρέφει μια απάντηση, συνήθως με τα δεδομένα που ζητήθηκαν ή ένα μήνυμα λάθους.

Το Μοντέλο Πελάτη-Διακομιστή

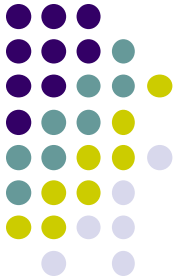


Οργάνωση Κατανεμημένων Συστημάτων – Το Μοντέλο Πελάτη-Διακομιστή



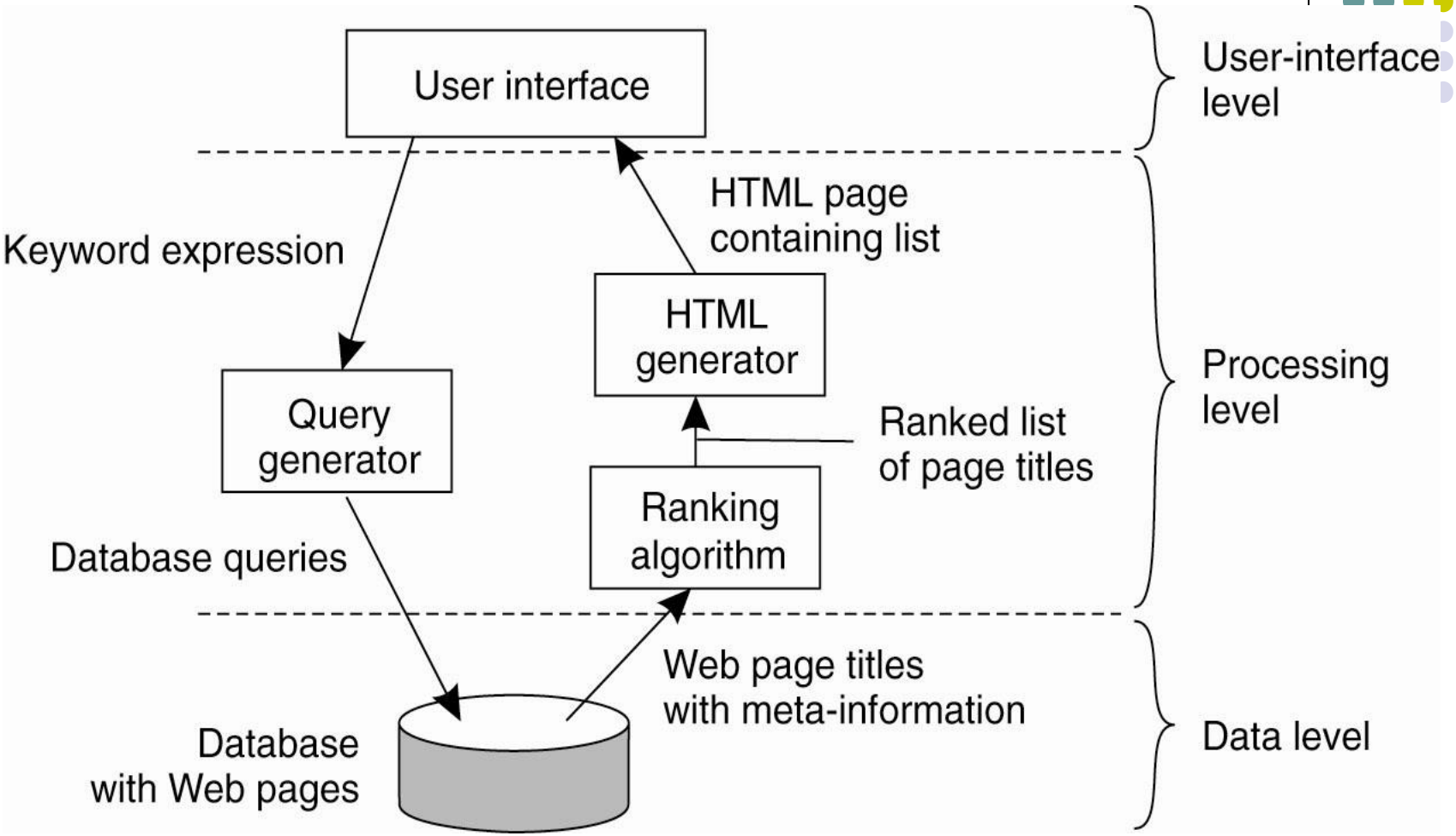
Γενική αλληλεπίδραση μεταξύ διεργασίας-πελάτη και διεργασίας-διακομιστή

Επίπεδα Εφαρμογής



- ***Επίπεδο διεπαφής χρήστη*: χειρίζεται την αλληλεπίδραση με το χρήστη**
- ***Επίπεδο επεξεργασίας*: παρέχει τις λειτουργικές δυνατότητες της εφαρμογής**
- ***Επίπεδο δεδομένων*: βάση δεδομένων ή σύστημα αρχείων**

Επίπεδα Εφαρμογής

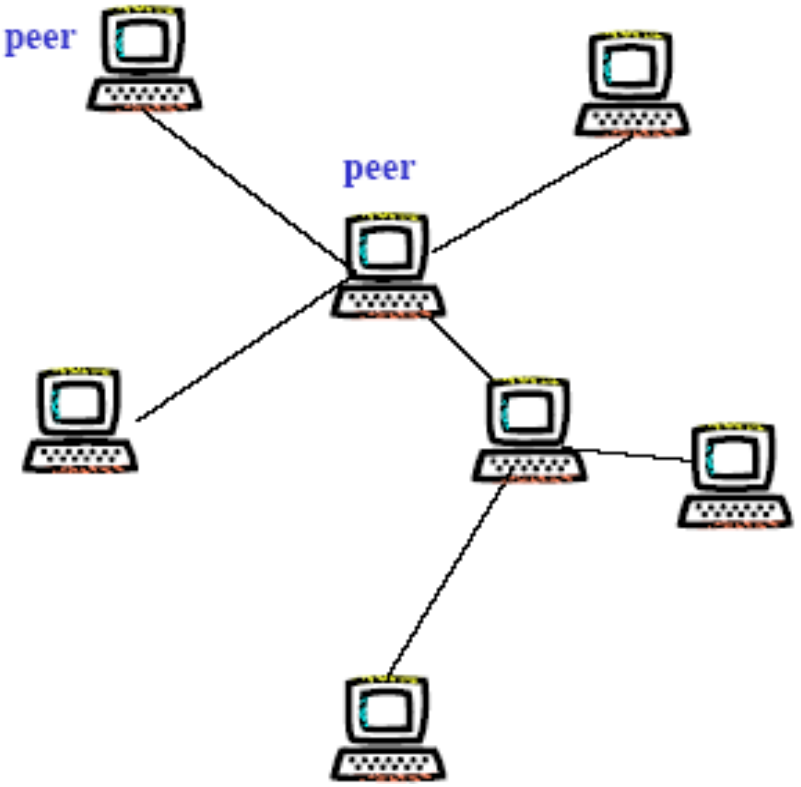


Οργάνωση μιας μηχανής αναζήτησης του Internet σε τρία διαφορετικά επίπεδα

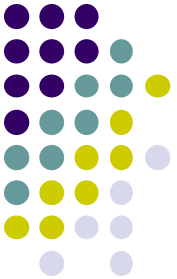
Ομότιμα Συστήματα (Peer-to-Peer Systems)

Κάθε κόμβος διαθέτει το ίδιο λογισμικό και επομένως συμπεριφέρεται ως πελάτης και ως διακομιστής.

- **Ομότιμη δράση των κόμβων.**
- **Οι κόμβοι επικοινωνούν απ'ευθείας και όχι μέσω ενός server**

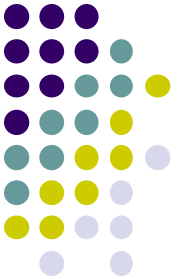


Ομότιμα Συστήματα (P2P Systems)



- **Αξιοποίηση των ελεύθερων πόρων συστημάτων**
- **Δημιουργία περισσότερο κλιμακούμενων συστημάτων**
- **Δημιουργία συστημάτων με μεγαλύτερη διαθεσιμότητα**
- **Κατάργηση μονοπωλίων στην διάθεση της πληροφορίας**
- **Αυτο-οργάνωση αντί κεντρικής διαχείρισης**

Εφαρμογές Ομότιμων Συστηματων



- *Κατανεμημένος Υπολογισμός*
- *Διαμοιρασμός Αρχείων*
- *Συνεργατικές Εφαρμογές*
- Εφαρμογές Κατανεμημένου Υπολογισμού απαιτούν την αποσύνθεση μεγάλων προβλημάτων σε μικρότερα, παράλληλα προβλήματα
- Διαμοιρασμός αρχείων απαιτεί αποδοτική αναζήτηση και μεταφορά αρχείων μέσα σε wide area networks
- Συνεργατικές εφαρμογές απαιτούν μηχανισμούς ενημέρωσης (update mechanisms) για περιβάλλοντα με πολλούς χρήστες (multi-user environments)

- Κατανεμημένου Υπολογισμού (π.χ., SETI@home, Search for Extraterrestrial Intelligence)
- Διαμοιρασμός αρχείων(π.χ., Gnutella, BitTorrent, eMule)
- Συνεργατικές (π.χ., Magi, Groove, Jabber, Chat – icq, yahoo!, etc)

SETI@home

- **Anybody with an at least intermittently Internet-connected computer can participate in SETI@home by running a free program that downloads and analyzes radio telescope data.**

- **The SETI@home distributed computing software runs either as a screensaver or continuously while a user works, making use of processor time that would otherwise be unused.**

- **With over 5.2 million participants worldwide, the project is the distributed computing project with the most participants to date. SETI@home has the ability to compute over 769 teraFLOPS.**

Τύποι Ομότιμων Συστημάτων

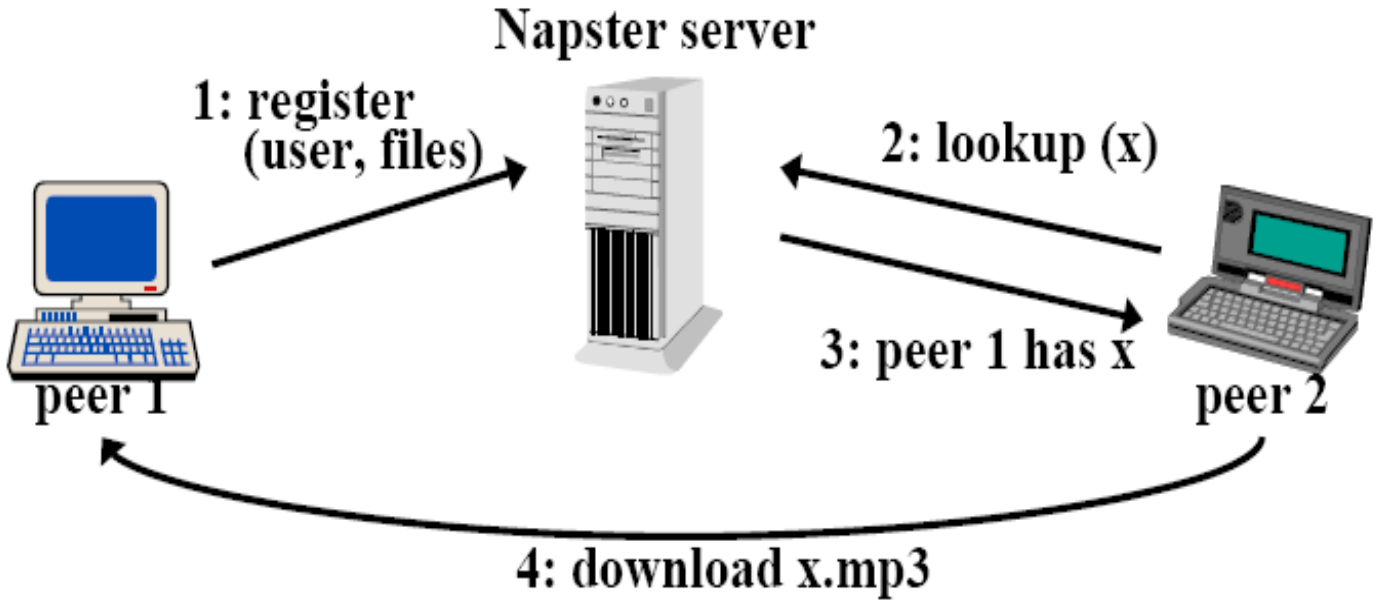
- Υβριδικά Ομότιμα Συστήματα
- Αδόμητα Ομότιμα Συστήματα
- Ιεραρχικά Ομότιμα Συστήματα
- Δομημένα Ομότιμα Συστήματα

Υβριδικά Ομότιμα Συστήματα (Hybrid P2P systems)

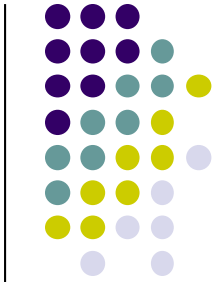
Υπάρχει ένας κεντρικός διακομιστής

- **Διαμοιρασμός αρχείων:** ***Αν ένας κόμβος διαθέτει ένα αρχείο, οι άλλοι μπορούν να το αναζητήσουν στον κεντρικό διακομιστή και να το κατεβάσουν από τη μηχανή όπου βρίσκεται***

Napster (1998-2001): διαμοιρασμός MP3



Google (Client-Server) vs. Napster (P2P)



Ίδιας κλίμακας συστήματα

– εκατομμύρια αναζητήσεις ημερησίως

– Terabytes δεδομένων

Google

– Στηρίζεται σε περίπου 100.000 μηχανές

– το στήσιμο μιας τέτοιας εφαρμογής έχει μεγάλο κόστος

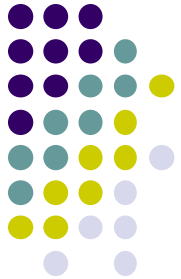
Napster

– ο server χρησιμοποιεί μόνο 100 μηχανές

– μικρό κόστος αφού το κόστος αποθήκευσης και μεταφοράς των μουσικών αρχείων χρεώνεται στις μηχανές των χρηστών του συστήματος

Υβριδικά Ομότιμα Συστήματα

Πλεονεκτήματα



Διαμερισμός πόρων

– αποθηκευτικών, αφού οι χρήστες του συστήματος αποθηκεύουν τα αρχεία, όχι ο διακομιστής

-υπολογιστικών, αφού οι χρήστες του συστήματος εκτελούν τον υπολογισμό όχι ο διακομιστής

– επικοινωνίας, αφού το κατέβασμα αρχείων γίνεται μεταξύ των χρηστών χωρίς να παρεμβάλλεται ο διακομιστής

– εισαγωγής στοιχείων αφού

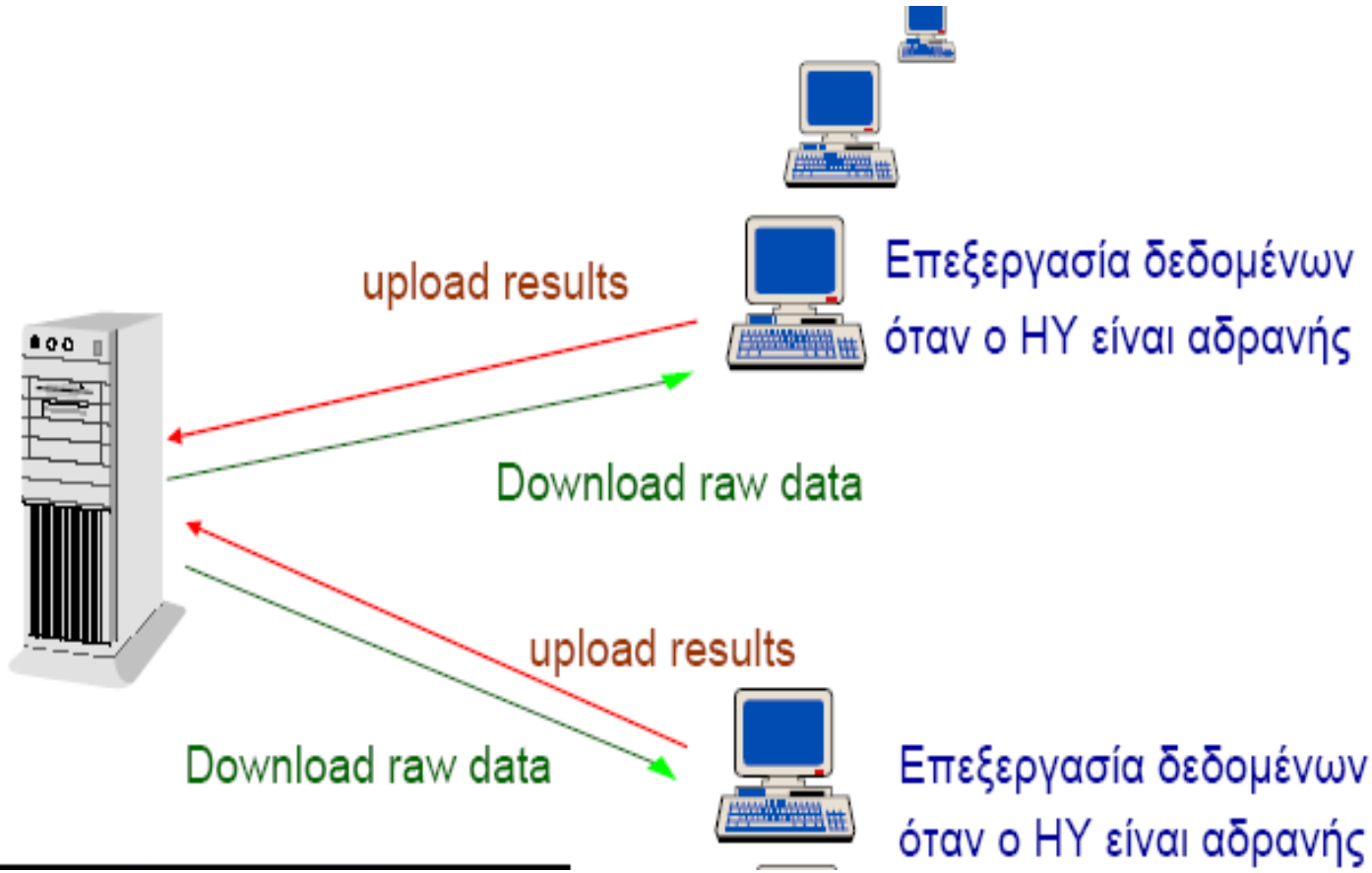
οι χρήστες του συστήματος εισάγουν τα δεδομένα στο σύστημα

οι χρήστες του συστήματος τα κατηγοριοποιούν

Η αποκέντρωση επιτρέπει τη δημιουργία εφαρμογών παγκόσμιας κλίμακας με μικρό κόστος!

Υβριδικά Ομότιμα Συστήματα

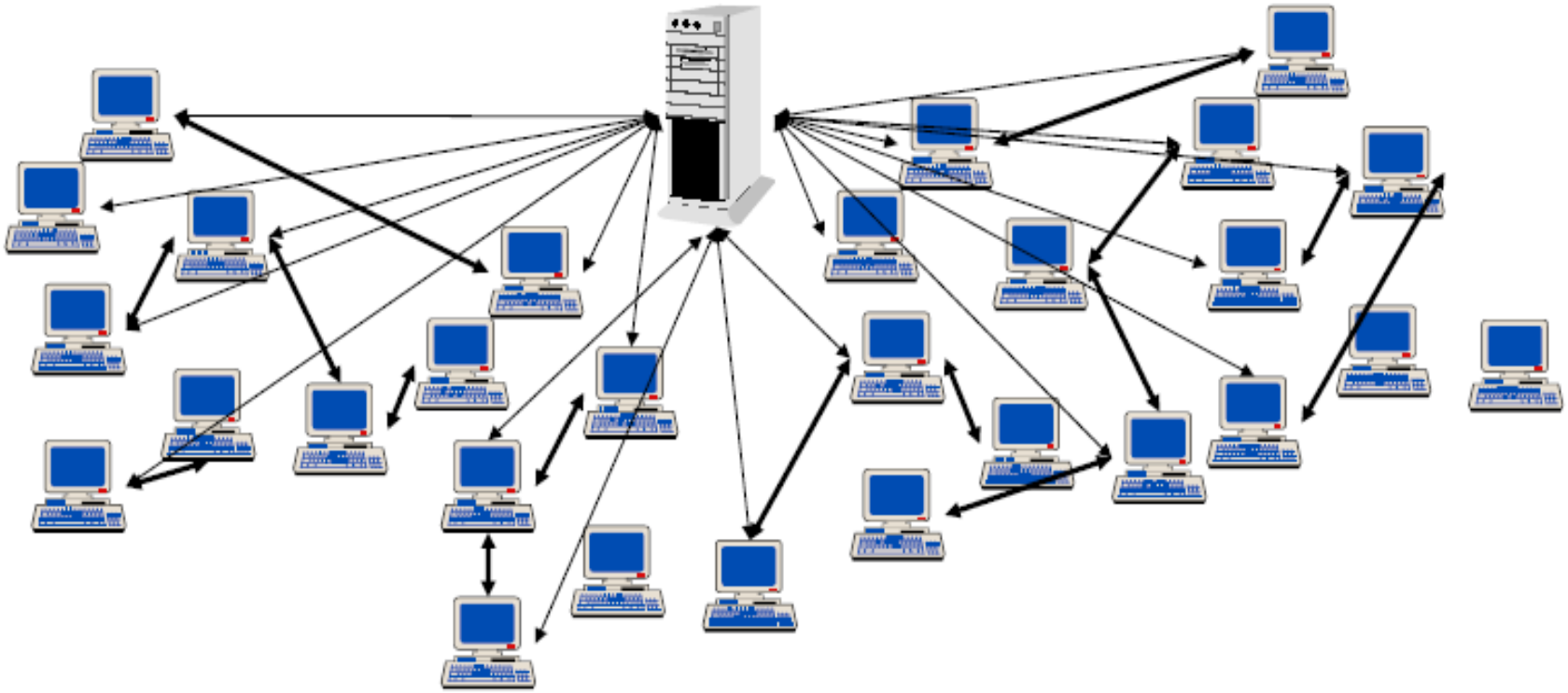
Διαμοιρασμός υπολογιστικών πόρων



Υβριδικά Ομότιμα Συστήματα

Μειονεκτήματα:

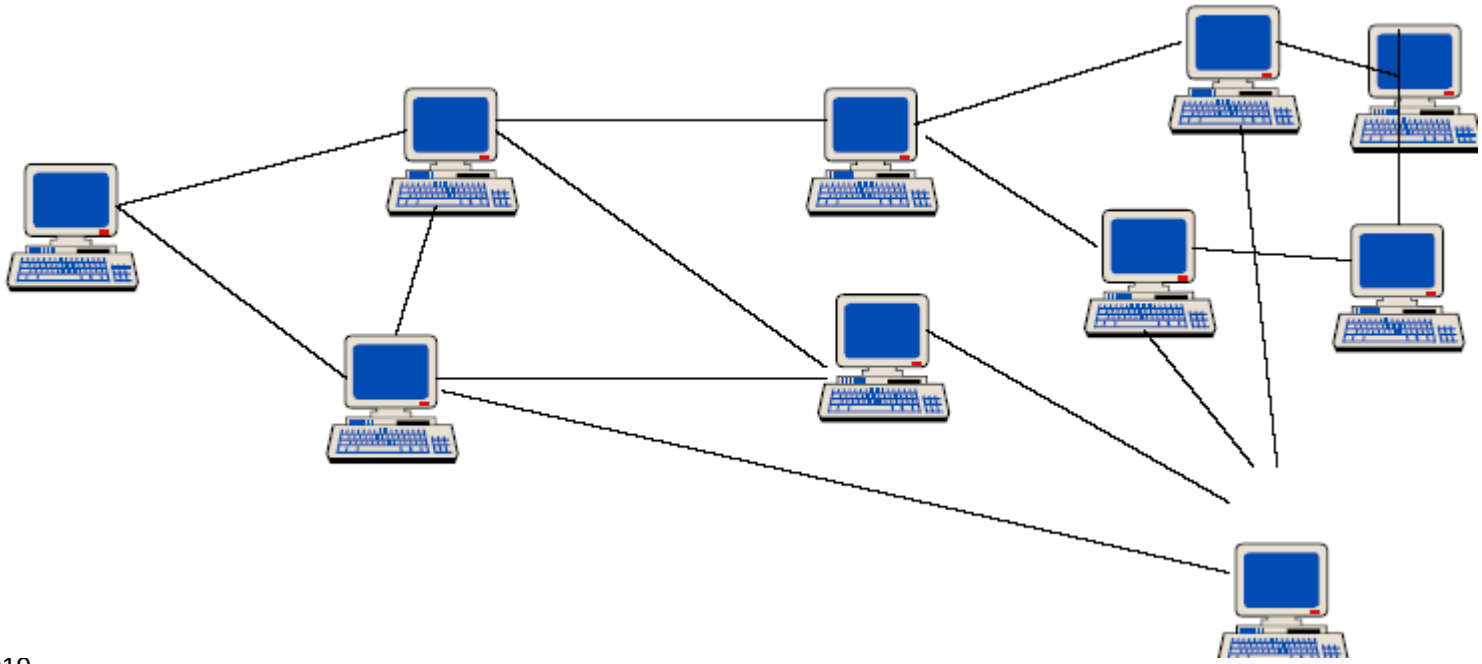
Central Point of Failure



Αδόμητα Ομότιμα Συστήματα **(Unstructured ή Pure** P2P systems)

- **Δεν υπάρχει κεντρικός εξυπηρετητής**

Gnutella (1999)

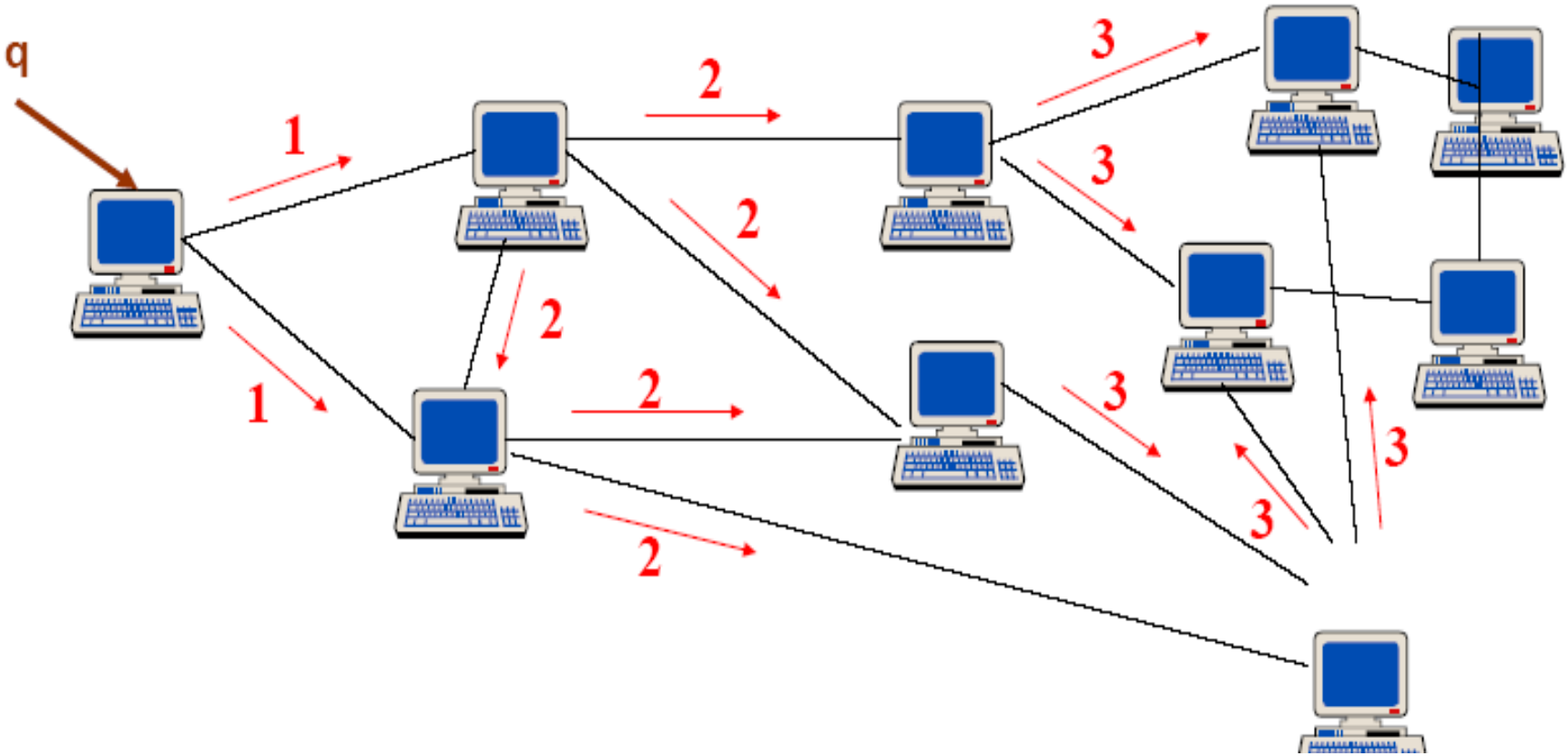


Gnutella

- **Το δίκτυο αποτελείται από κόμβους που ονομάζονται servents από το SERV(er) + cliENT.**
- **Η εισαγωγή ενός κόμβου στο δίκτυο γίνεται με την σύνδεση σε κάποιον κόμβο που είναι συνδεδεμένος στο δίκτυο**
- **Κάθε κόμβος στέλνει περιοδικά μηνύματα ping για να βρίσκει νέο-εισαχθέντες κόμβους. Έτσι διατηρείται ο δυναμικός χαρακτήρας του δικτύου**
- **Τα μηνύματα είτε γίνονται broadcast είτε back-propagated**
- **Κάθε μήνυμα έχει ένα τυχαίο μοναδικό id**
- **Κάθε κόμβος διαθέτει λίγη μνήμη για τα μηνύματα που πρόσφατα έχει δρομολογήσει την οποία χρησιμοποιεί για να μην ξανακάνει broadcast αλλά και για να υλοποιήσει το back-propagation**
- **Ορίζεται μια τιμή για το χρόνο ζωής TTL**
- **Οι κόμβοι αποφασίζουν που να συνδεθούν στο δίκτυο βασιζόμενοι αποκλειστικά σε τοπική πληροφορία. Το Gnutella είναι ένα δίκτυο το οποίο οργανώνεται μόνο του.**

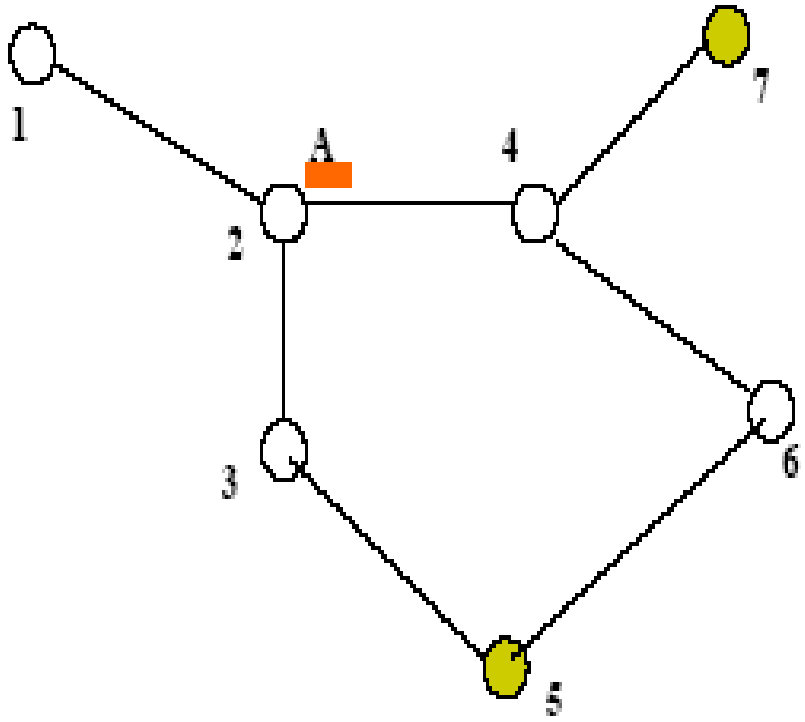
Αδόμητα Ομότιμα Συστήματα

Χρήση κατακλυσμού μηνυμάτων (message flooding) για τον εντοπισμό πόρων (resource discovery)



Gnutella

Εντοπισμός πόρων

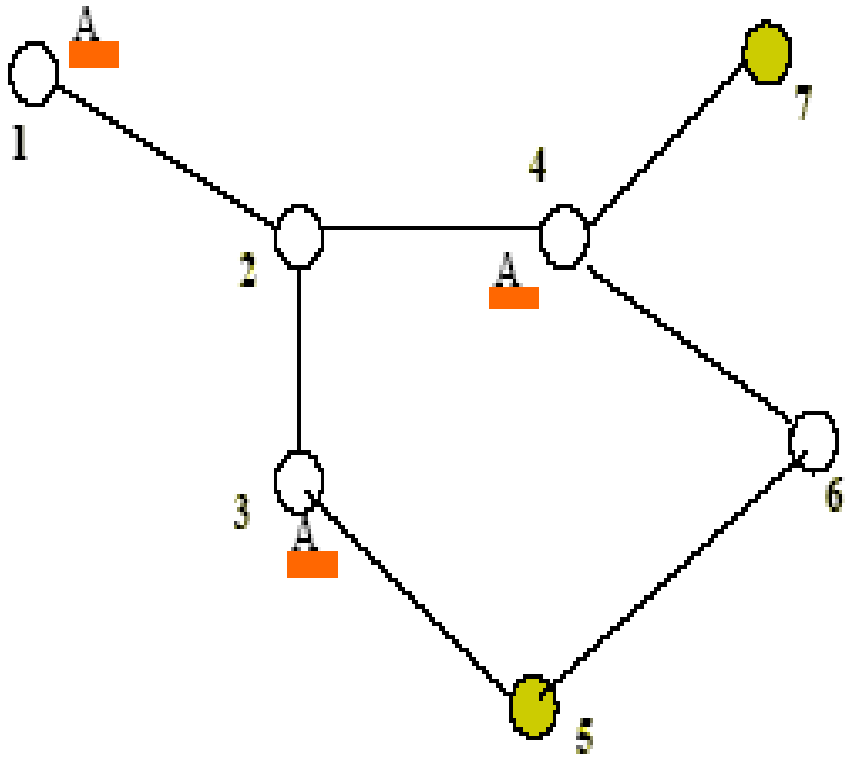


Βήματα:

- Ο κόμβος 2 αρχίζει μια αναζήτηση για το αρχείο A.

Gnutella

Εντοπισμός πόρων

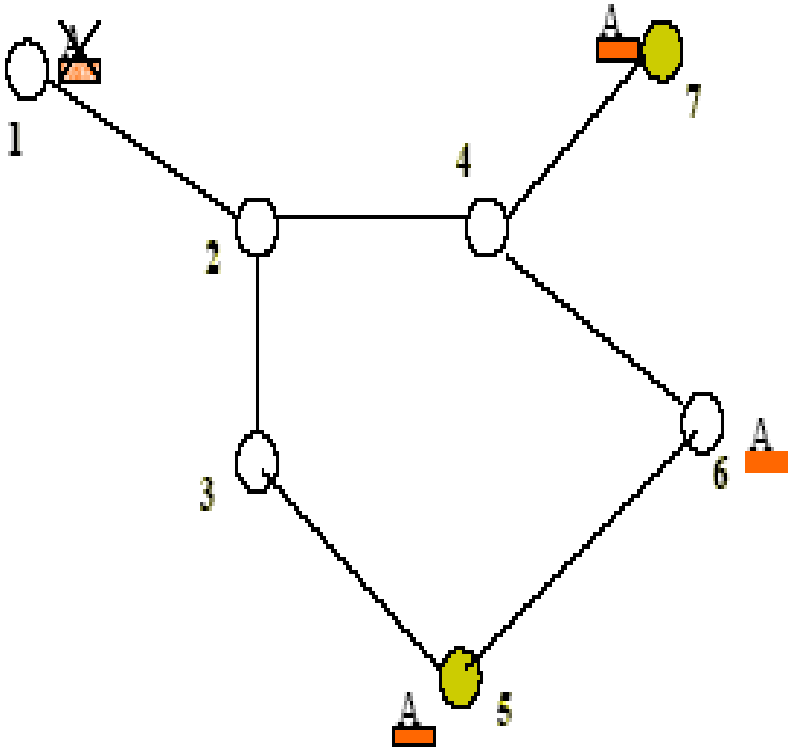


Βήματα:

- Ο κόμβος 2 αρχίζει μια αναζήτηση για το αρχείο Α.
- Στέλνει μήνυμα σε όλους τους γείτονες.

Gnutella

Εντοπισμός πόρων

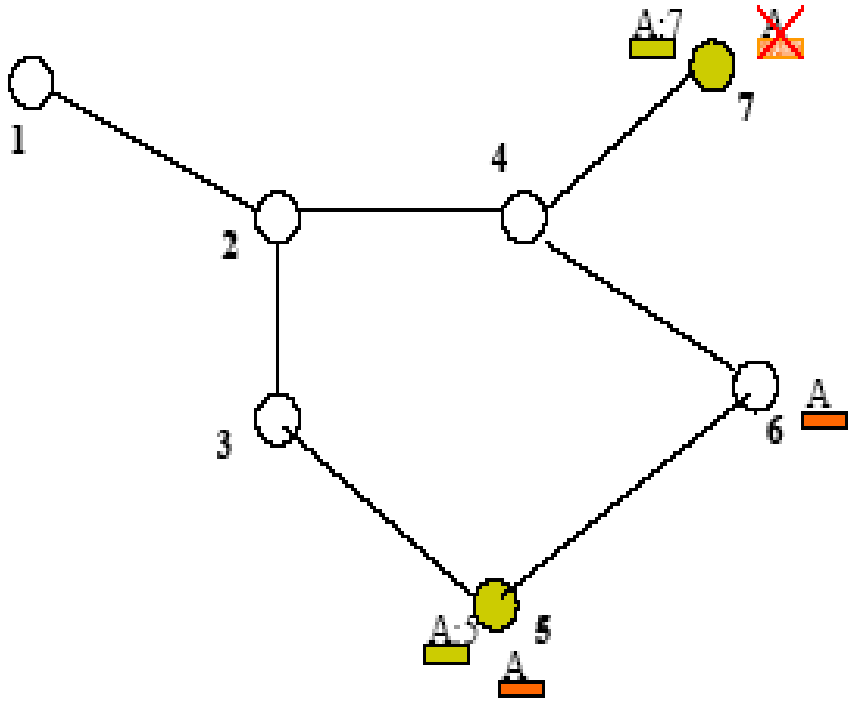


Βήματα:

- Ο κόμβος 2 αρχίζει μια αναζήτηση για το αρχείο A.
- Στέλνει μήνυμα σε όλους τους γείτονες.
- Οι γείτονες προωθούν το μήνυμα.

Gnutella

Εντοπισμός πόρων

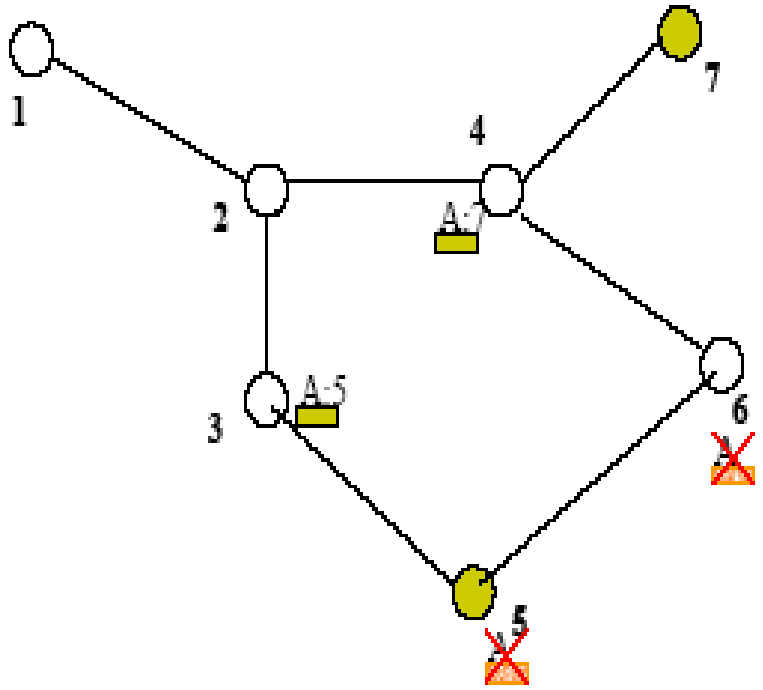


Βήματα:

- Ο κόμβος 2 αρχίζει μια αναζήτηση για το αρχείο Α.
- Στέλνει μήνυμα σε όλους τους γείτονες.
- Οι γείτονες προωθούν το μήνυμα.
- Οι κόμβοι που έχουν το αρχείο Α στέλνουν απάντηση.

Gnutella

Εντοπισμός πόρων

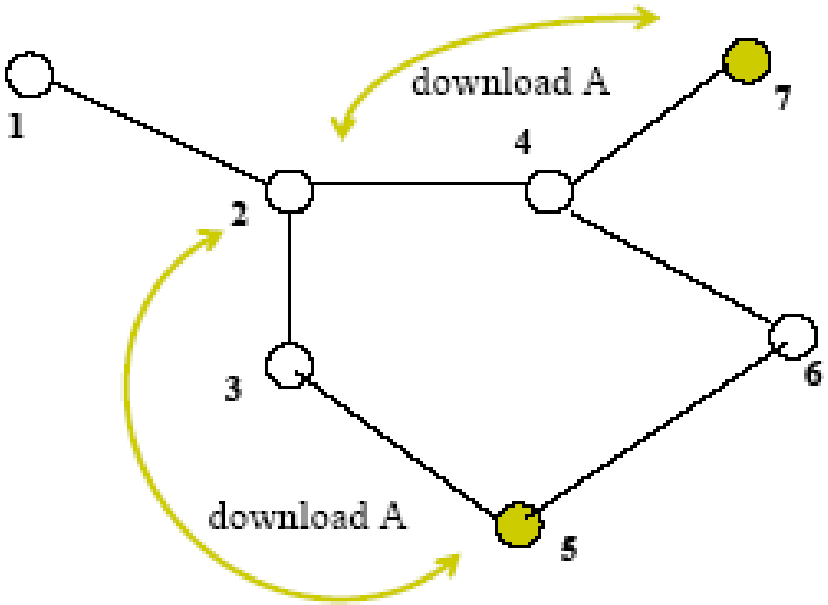


Βήματα:

- Ο κόμβος 2 αρχίζει μια αναζήτηση για το αρχείο A.
- Στέλνει μήνυμα σε όλους τους γείτονες.
- Οι γείτονες προωθούν το μήνυμα.
- Οι κόμβοι που έχουν το αρχείο A στέλνουν απάντηση.
- Η απάντηση διαδίδεται προς τα πίσω (back-propagated).

Gnutella

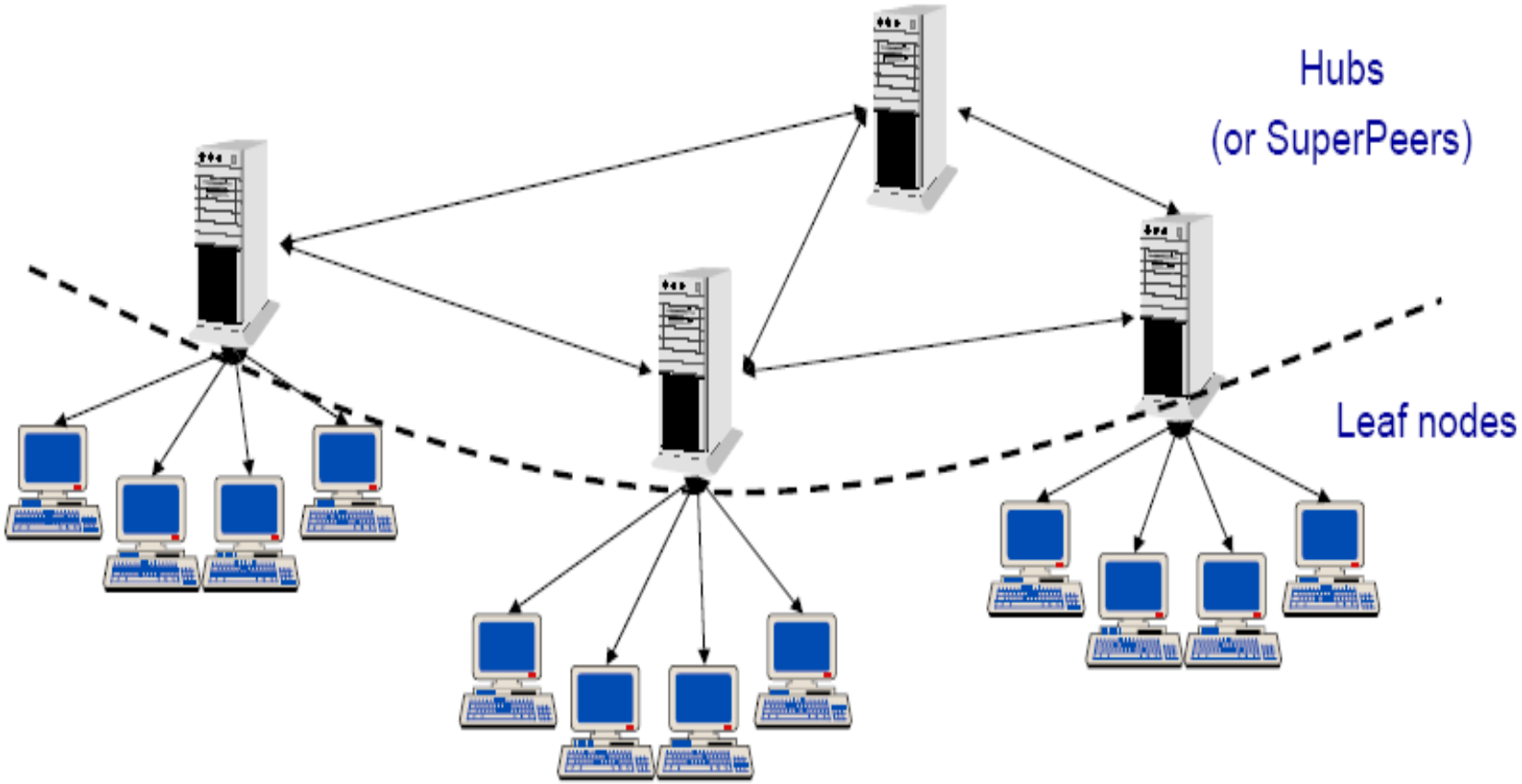
Εντοπισμός πόρων



Βήματα:

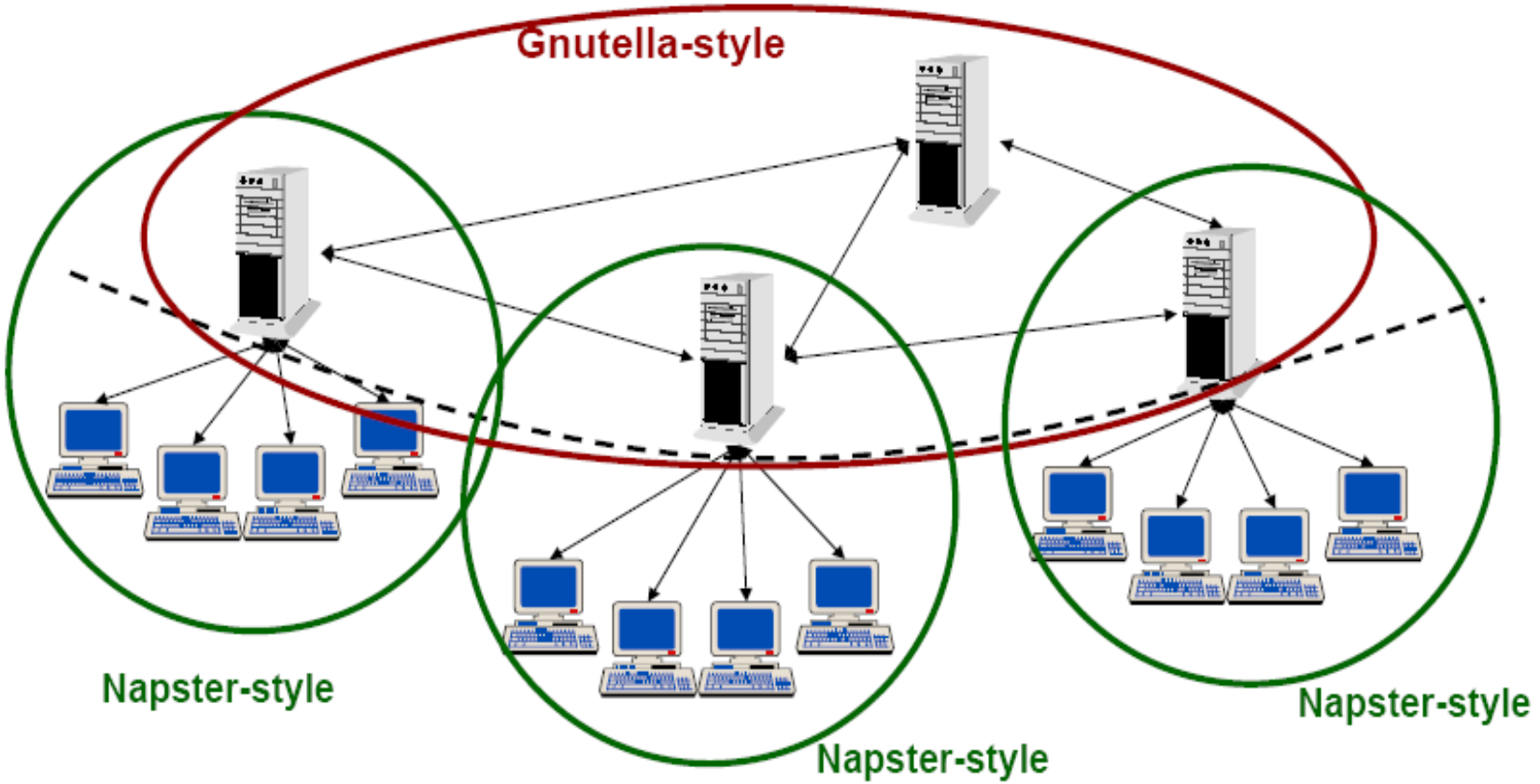
- Ο κόμβος 2 αρχίζει μια αναζήτηση για το αρχείο A.
- Στέλνει μήνυμα σε όλους τους γείτονες.
- Οι γείτονες προωθούν το μήνυμα.
- Οι κόμβοι που έχουν το αρχείο A στέλνουν απάντηση.
- Η απάντηση διαδίδεται προς τα πίσω (back-propagated).
- Download "A"

Ιεραρχικά Ομότιμα Συστήματα **(Hierarchical** P2P Systems**)**



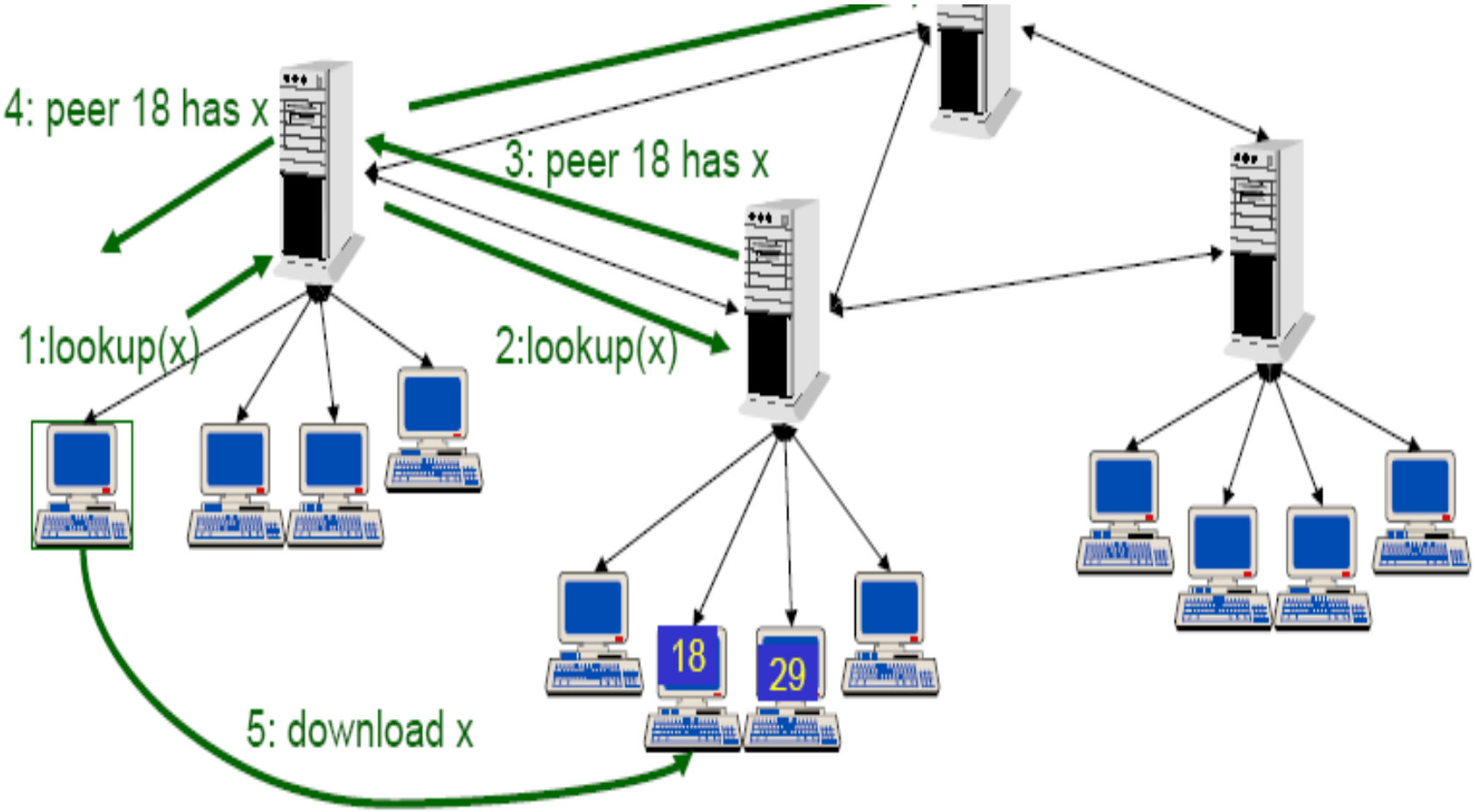
Ιεραρχικά Ομότιμα Συστήματα:

Συνδυασμός υβριδικών και αδόμητων συστημάτων

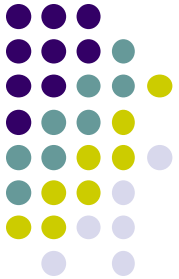


Ιεραρχικά Ομότιμα Συστήματα:

Εντοπισμός Πόρων



Δομημένα Ομότιμα Συστήματα (**Structured** P2P Systems)



Σκοπός:

Γρήγορος εντοπισμός πόρων χωρίς τη χρήση κεντρικού διακομιστή και ανταλλάσσοντας λίγα μηνύματα

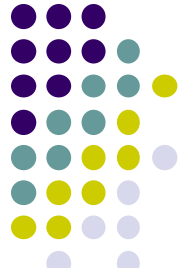
Κοινά χαρακτηριστικά των δομημένων ομότιμων συστημάτων

– κάθε κόμβος διατηρεί ένα μικρό τμήμα του καθολικού ευρετηρίου (πίνακας δρομολόγησης)

– οι αναζητήσεις γίνονται με προώθηση μηνυμάτων προς τη «σωστή» κατεύθυνση

Δομημένα Ομότιμα Συστήματα

Εντοπισμός πόρων



"Singing in the Rain" SR.mp3
«Υπάρχω» stelios.mp3

"Singing in the Rain" SingRain.mp3

p	k	d
121.111.86	"Singing in the Rain"	SR.mp3
121.111.86	«Υπάρχω»	stelios.mp3
222.18.78	"Singing in the Rain"	SingRain.mp3

Έστω peer με δνση **p** που αποθηκεύει στοιχείο **d** που χαρακτηρίζεται από το κλειδί **k**

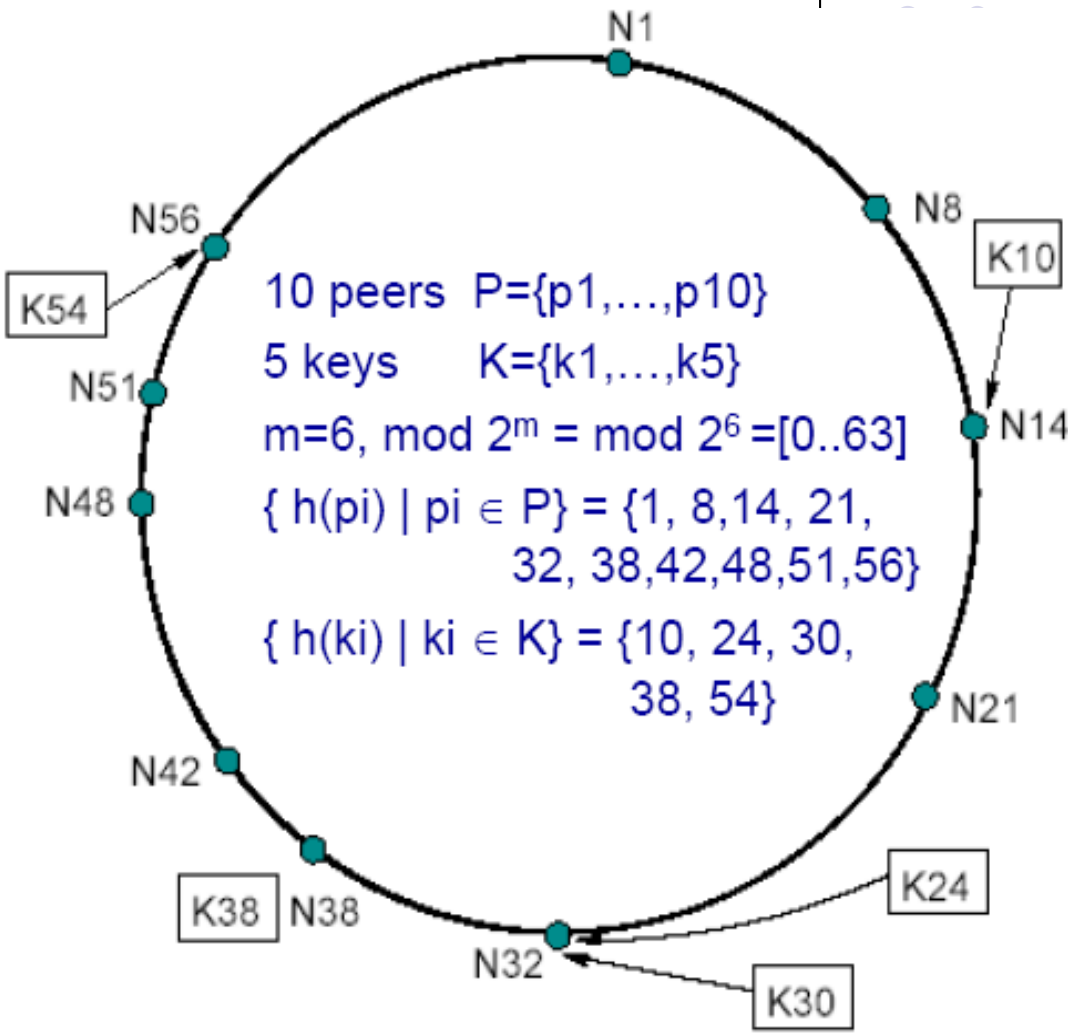
Ζητούμενο: Δοθέντος **k** (ή συνθήκης πάνω στο **k**) εντόπισε τον peer που έχει το **d**, δηλαδή βρες το ζεύγος ευρετηρίου **(k,p).**

(άρα το ευρετήριο μας αποτελείται από ζεύγη της μορφής **(k,p)**)

*Κρίσιμο ερώτημα: Πως μπορούμε να (α) **φτιάξουμε**, (β) **συντηρήσουμε** και (γ) να **χρησιμοποιήσουμε** ένα τέτοιο ευρετήριο χωρίς κεντρικό έλεγχο;*

Chord
Χρήση Distributed Hash Tables (DHT)

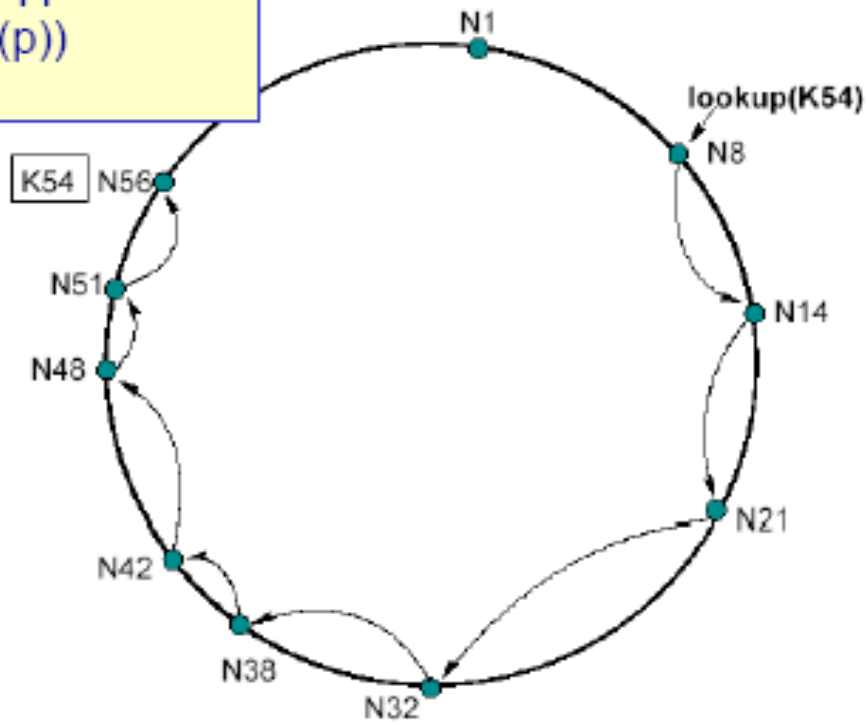
- Κατακερματισμός (Hashing) κλειδιών (k)και διευθύνσεων (p) σε δυαδικά κλειδιά με **m**-bits
  - π.χ. m=6, **h**(«υπάρχω»)=**11**, **h**(196.178.0.1)=**3**
- Τα δυαδ. κλειδιά τοποθετούνται σε έναν κύκλο modulo 2^m
  - Για m=8, κυκλική διάταξη των αριθμών 0…255
- Ένα κλειδί **k** εκχωρείται στον πρώτο κόμβο **p** τ.ω. h(**p**) >= h(**k**)
- Αυτός ο κόμβος λέγεται **successor(k)**



Chord
Απλός τρόπος εντοπισμού κόμβων

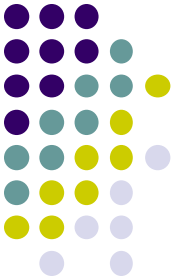
Έστω ότι κάθε κόμβος p ξέρει την δνση μόνο του επόμενου του (του p' με h(p') > h(p))

```
// ask node n to find the successor of id
n.find_successor(id)
  if (id in (n; successor])
    return successor;
  else
    // forward the query around the circle
   return successor.find_successor(id);
```



=> Number of messages linear in the number of nodes !

Chord
Εντοπισμός πόρου με χρήση πίνακα δρομολόγησης



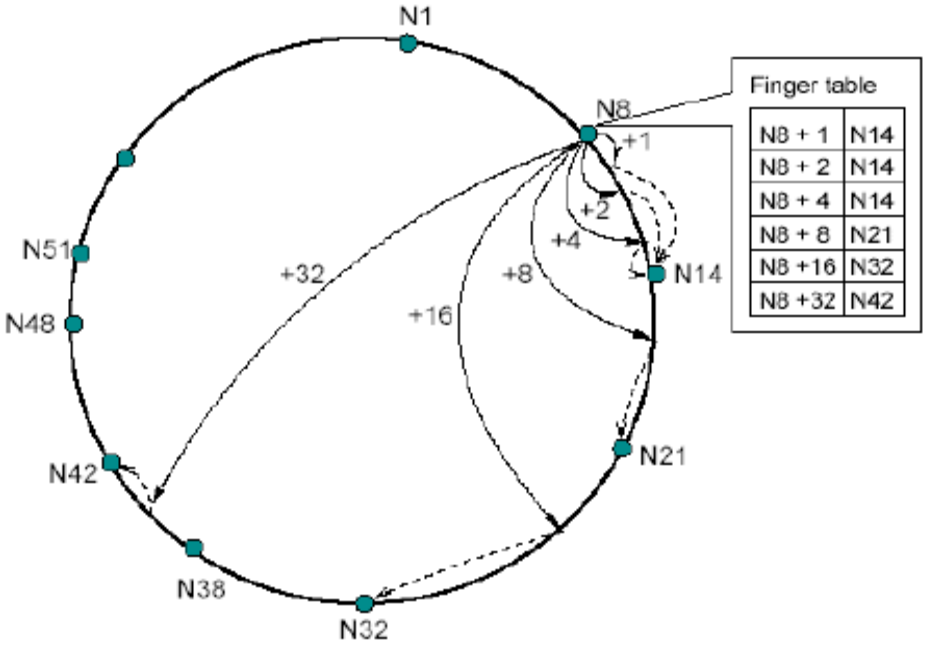
- Επιπλέον πληροφορία δρομολόγησης για επιτάχυνση
- Κάθε κόμβος n έχει έναν **πίνακα δρομολόγησης** με m εγγραφές
  - οι m αυτοί κόμβοι έχουν εκθετικά αυξανόμενη απόσταση από τον n
- Η i εγγραφή του πίνακα έχει την δνση του πρώτου κόμβου με κλειδί μεγαλύτερο ή ίσο με $n+2^{i-1}$

$$finger[i] = successor\ (n + 2^{i-1})$$

Chord
Εντοπισμός πόρου με χρήση πίνακα δρομολόγησης

Finger table:

finger[i] = successor (n + 2^{i-1})



n=8

finger[1] = succ(n+1) = succ(9) = 14

finger[2] = succ(n+2) = succ(10) = 14

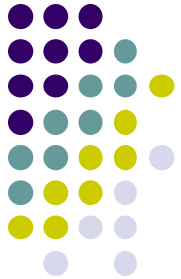
finger[3] = succ(n+4) = succ(12) = 14

finger[4] = succ(n+8) = succ(16) = 21

finger[5] = succ(n+16) = succ(24) = 32

finger[6] = succ(n+32) = succ(40) = 42

Chord
Εντοπισμός πόρου με χρήση πίνακα δρομολόγησης



Έστω μια επερώτηση k προς έναν κόμβο n

Ο n κοιτάζει τον πίνακα δρομολόγησης του και

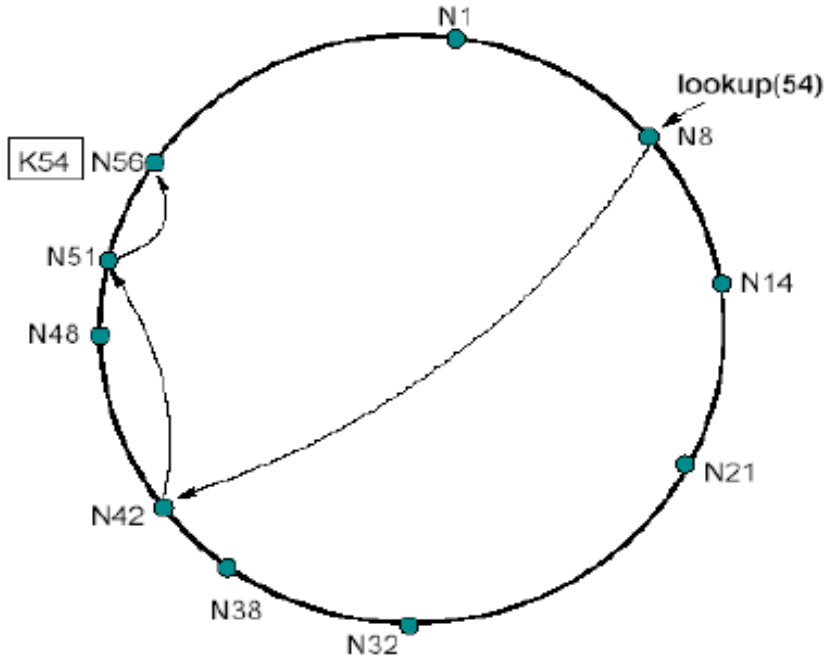
βρίσκει τον μικρότερο peer με κλειδί μεγαλύτερο αυτού της επερώτησης.

Αν δεν υπάρχει τέτοιος peer, τότε ο ίδιος είναι υπεύθυνος για το κ (και άρα το ζητούμενο βρέθηκε)

Αλλιώς προωθεί την επερώτηση

Αφού οι εγγραφές των πινάκων δρομολόγησης είναι εκθετικά αύξουσες, η αναζήτηση (με μεγάλη πιθανότητα) λαμβάνει λογαριθμικό χρόνο.

Chord
Εντοπισμός πόρου με χρήση πίνακα δρομολόγησης



N8

Finger table	
N8 + 1	N14
N8 + 2	N14
N8 + 4	N14
N8 + 8	N21
N8 +16	N32
N8 +32	N42

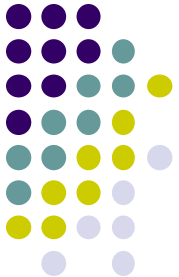
N42

Finger table	
N42+ 1	N48
N42+ 2	N48
N42+ 4	N48
N42+ 8	N51
N42+16	N1
N42+32	N14

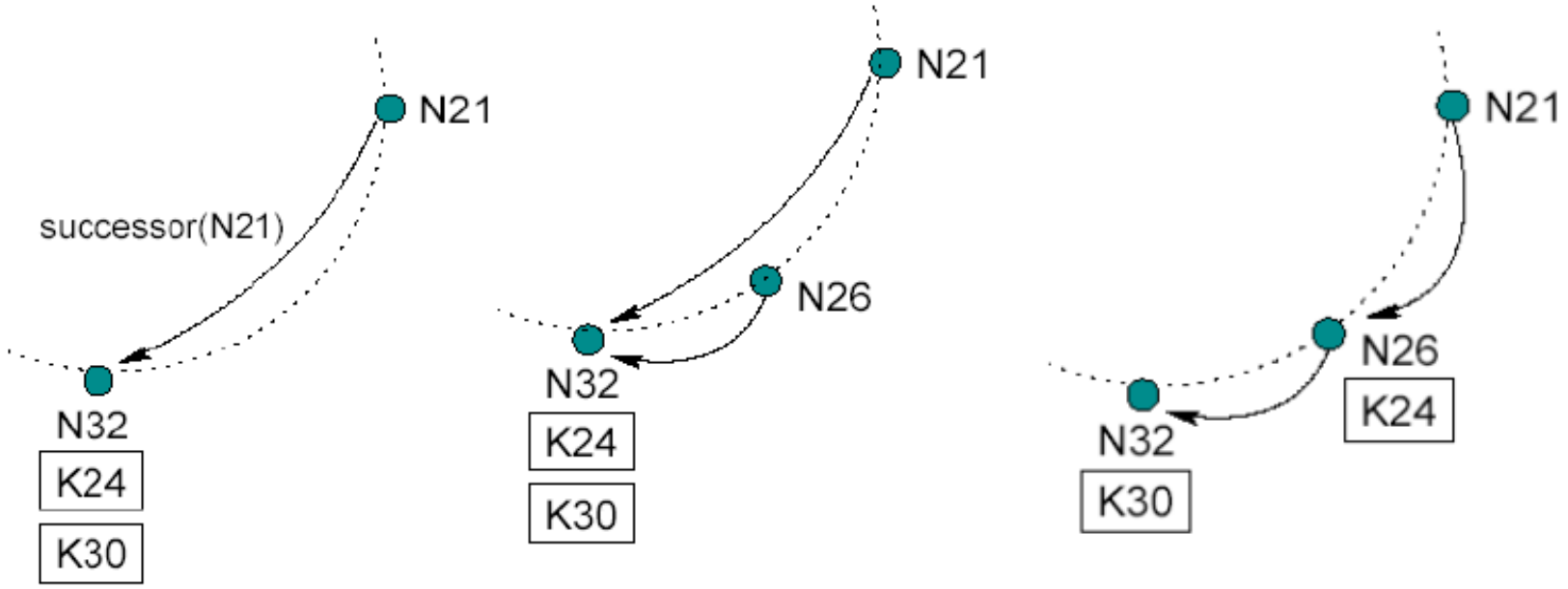
Εύρεση με ανταλλαγή τριών μηνυμάτων

- Search in finger table for the nodes which most immediatly precedes id
- Invoke find_successor from that node
- **=> Number of messages O(log N)**

Chord
Εντοπισμός πόρου με χρήση πίνακα δρομολόγησης – Εισαγωγή νέου κόμβου



- Ο νέος κόμβος πρέπει να φτιάξει τον πίνακα δρομολόγησης του
- Το κόστος κατασκευής του είναι αυτό της αναζήτησης
- Οι άλλοι κόμβοι πρέπει να ενημερώσουν τους δικούς τους πίνακες



Chord

Βασικά σημεία:

- **Κάθε κόμβος αποθηκεύει πληροφορία για μικρό αριθμό κόμβων (m)**

– **Κάθε κόμβος ξέρει περισσότερα για τους κοντινούς του όρους (απ'ότι για τους μακρινούς)**

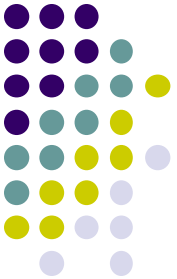
Επιδόσεις

- **Χρόνος αναζήτησης: O(log n) (με μεγάλη πιθανότητα)**
- **Πλήθος Μηνυμάτων: O(log n) (επιλεκτική δρομολόγηση)**
- **Κόστος αποθήκευσης: O(log n) (πίνακας δρομολόγησης)**
- **Κόστος εισόδου/εξόδου κόμβου: O(log^2 n)**
- **Κόστος ενημέρωσης: μικρό (περίπου σαν το κόστος αναζήτησης)**

Chord software

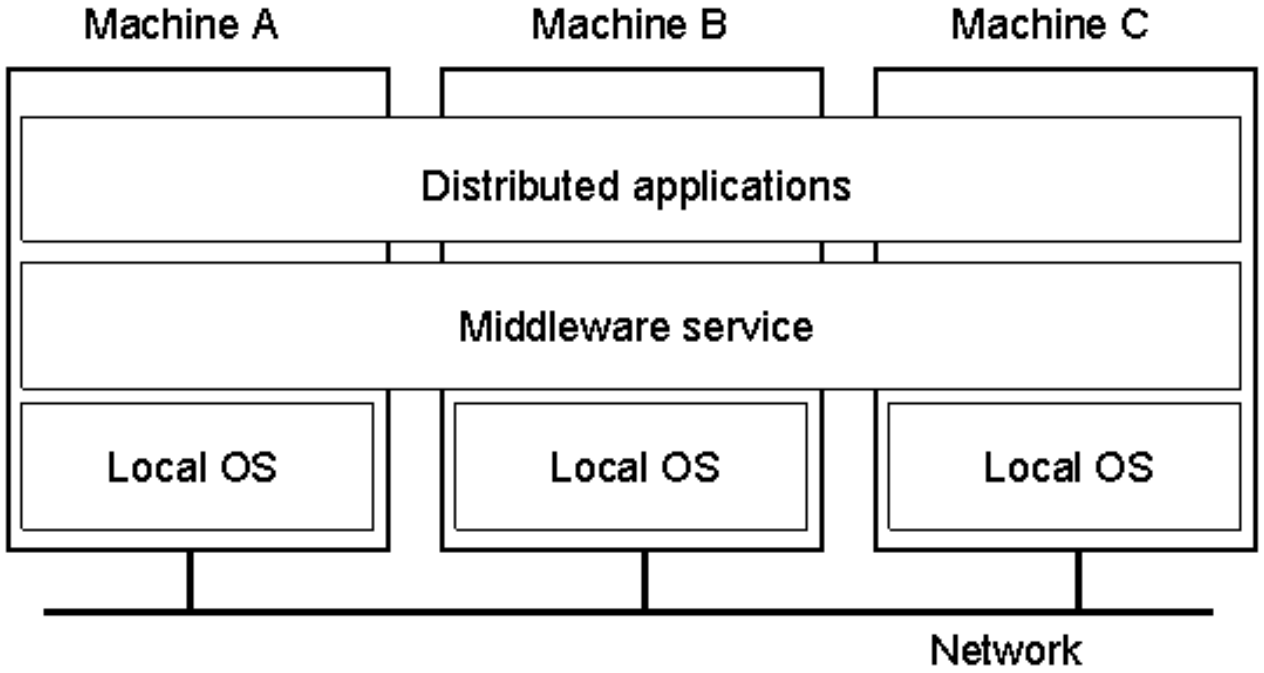
– **3000 lines of C++ code, Library to be linked with the application, provides a lookup(key) – function: yields the IP address of the node responsible for the key. Notifies the node of changes in the set of keys the node is responsible for**

Middleware – Ενδιάμεσο Λογισμικό



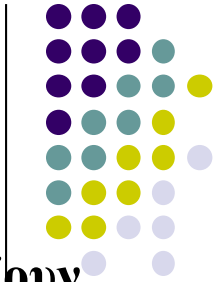
- **Προκειμένου να υποστηριχτούν διαφορετικά είδη υπολογιστών και δικτύων και παράλληλα να προσφέρουν μια ομοιόμορφη εικόνα στους τελικούς χρήστες, τα ΚΣ συνήθως αποτελούνται από ένα επίπεδο λογισμικού, το middleware**
- **Το λογισμικό αυτό τοποθετείται λογικά μεταξύ ενός υψηλότερου επιπέδου που αποτελείται από τους χρήστες και τις εφαρμογές και ενός χαμηλότερου επιπέδου που αποτελείται από λειτουργικά συστήματα**

Middleware



Παράδειγμα ΚΣ οργανωμένου ως middleware, το επίπεδο του οποίου καλύπτει πολλά διαφορετικά υπολογιστικά συστήματα

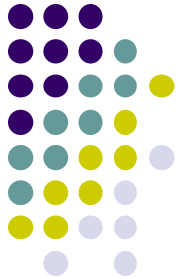
Πρωτόκολλα ενδιάμεσου λογισμικού



Τα πρωτόκολλα επικοινωνίας του ενδιάμεσου λογισμικού υποστηρίζουν ποικιλία υπηρεσιών ενδιάμεσου λογισμικού:

- **Πρωτόκολλα πιστοποίησης ταυτότητας (Authentication)**
- **Πρωτόκολλα εξουσιοδότησης πρόσβασης (Authorization) σε πιστοποιημένους χρήστες**
- **Πρωτόκολλα κατανεμημένης δέσμευσης (Distributed commit) ή ατομικότητας**
- **Πρωτόκολλα κατανεμημένου κλειδώματος (προστασία πόρων από ταυτόχρονη πρόσβαση)**

Μοντέλα ενδιάμεσου λογισμικού



- **Ενδιάμεσο λογισμικό που βασίζεται στα κατανεμημένα συστήματα αρχείων:**

  Η επικοινωνία ανάγεται στην προσπέλαση των διαμοιραζόμενων αρχείων.

  Η διαφάνεια της κατανεμημένης λειτουργίας περιορίζεται στα αρχεία (π.χ. συχνά απαιτείται η ρητή εκκίνηση διεργασιών σε συγκεκριμένες μηχανές).

- **Ενδιάμεσο λογισμικό που βασίζεται στις Κλήσεις Απομακρυσμένων Διαδικασιών (Remote Procedure Calls, RPC):**

  Μια διεργασία μπορεί να καλεί διαδικασία που υλοποιείται σε απομακρυσμένη μηχανή.

  Η καλούσα διεργασία δεν αντιλαμβάνεται ότι πραγματοποιήθηκε επικοινωνία μέσω δικτύου (πέρα από κάποια απώλεια απόδοσης)

- **Ενδιάμεσο λογισμικό που διαθέτει την έννοια του κατανεμημένου αντικειμένου:**

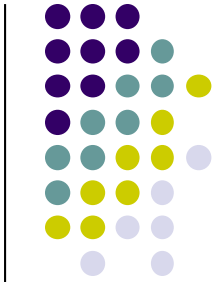
  Κάθε αντικείμενο τοποθετείται σε μία μηχανή και η διασύνδεσή του (μέθοδοι που υλοποιεί) γίνεται διαθέσιμη σε πολλές άλλες μηχανές. Όταν μια διεργασία καλεί μια μέθοδο, η υλοποίηση της διασύνδεσης στη μηχανή όπου εκτελείται η διεργασία μετασχηματίζει την κλήση της μεθόδου σε μήνυμα στο αντικείμενο. Το αντικείμενο εκτελεί τη μέθοδο και στέλνει πίσω το αποτέλεσμα.

Μοντέλα ενδιάμεσου λογισμικού

Παγκόσμιος Ιστός: **Χρήση του μοντέλου κατανεμημένων εγγράφων ήτοι οι πληροφορίες είναι οργανωμένες ως έγγραφα τα οποία είναι αποθηκευμένα σε ένα μηχάνημα κάπου στον κόσμο. Τα έγγραφα περιέχουν συνδέσμους τους οποίους αν ακολουθήσουμε, τα έγγραφα στα οποία αυτοί παραπέμπουν, προσκομίζονται στην οθόνη μας**

Remote Procedure Call (RPC)

- Κλήση Απομακρυσμένων Διαδικασιών



Όταν η επικοινωνία γίνεται με ανταλλαγή μηνυμάτων π.χ. κλήσεις send/receive χάνεται η διαφάνεια του συστήματος

Birrell and Nelson (1984)

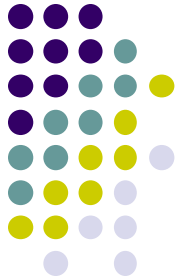
"Να επιτρέπεται στα προγράμματα να καλούν διαδικασίες που βρίσκονται σε άλλες μηχανές."

Η μέθοδος αυτή λέγεται Remote Procedure Call - (RPC)

Προβλήματα

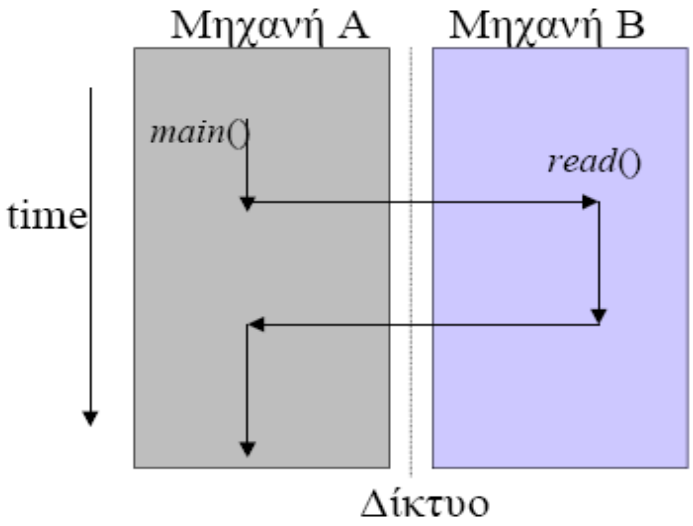
- Two machine architectures may not (or need not) be identical.
- Each machine has a different address space.
- How are parameters (of different, possibly very complex, types) passed to/from a remote procedure?

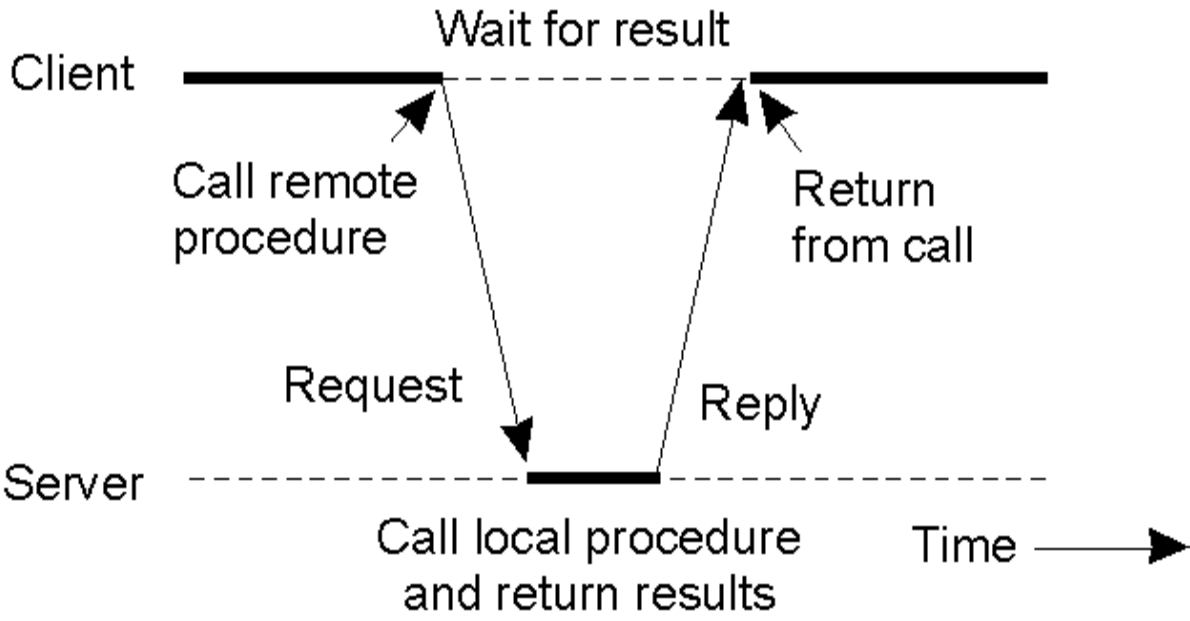
Επικοινωνία με χρήση απομακρυσμένων διαδικασιών



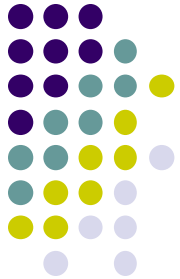
- Τα προγράμματα μπορούν να καλούν διαδικασίες που εκτελούνται σε διαφορετικές μηχανές (Birrell & Nelson 1984)
- Παρόμοια με κλήσεις τοπικών διαδικασιών
- Απόκρυψη της πραγματικής επικοινωνίας - Δεν λαμβάνεται γνώση της μεταβίβασης των μηνυμάτων και της λειτουργίας εισόδου/εξόδου
- Όταν μία διαδικασία στη μηχανή Α καλέσει μία διαδικασία στη μηχανή Β, η εκτέλεση της διεργασίας στην Α αναστέλλεται και η εκτέλεση της καλούμενης διαδικασίας συνεχίζεται στη μηχανή Β
- Πληροφορίες δίνονται από την καλούσα και την καλούμενη με τη μορφή παραμέτρων

Κλήση απομακρυσμένων διαδικασιών





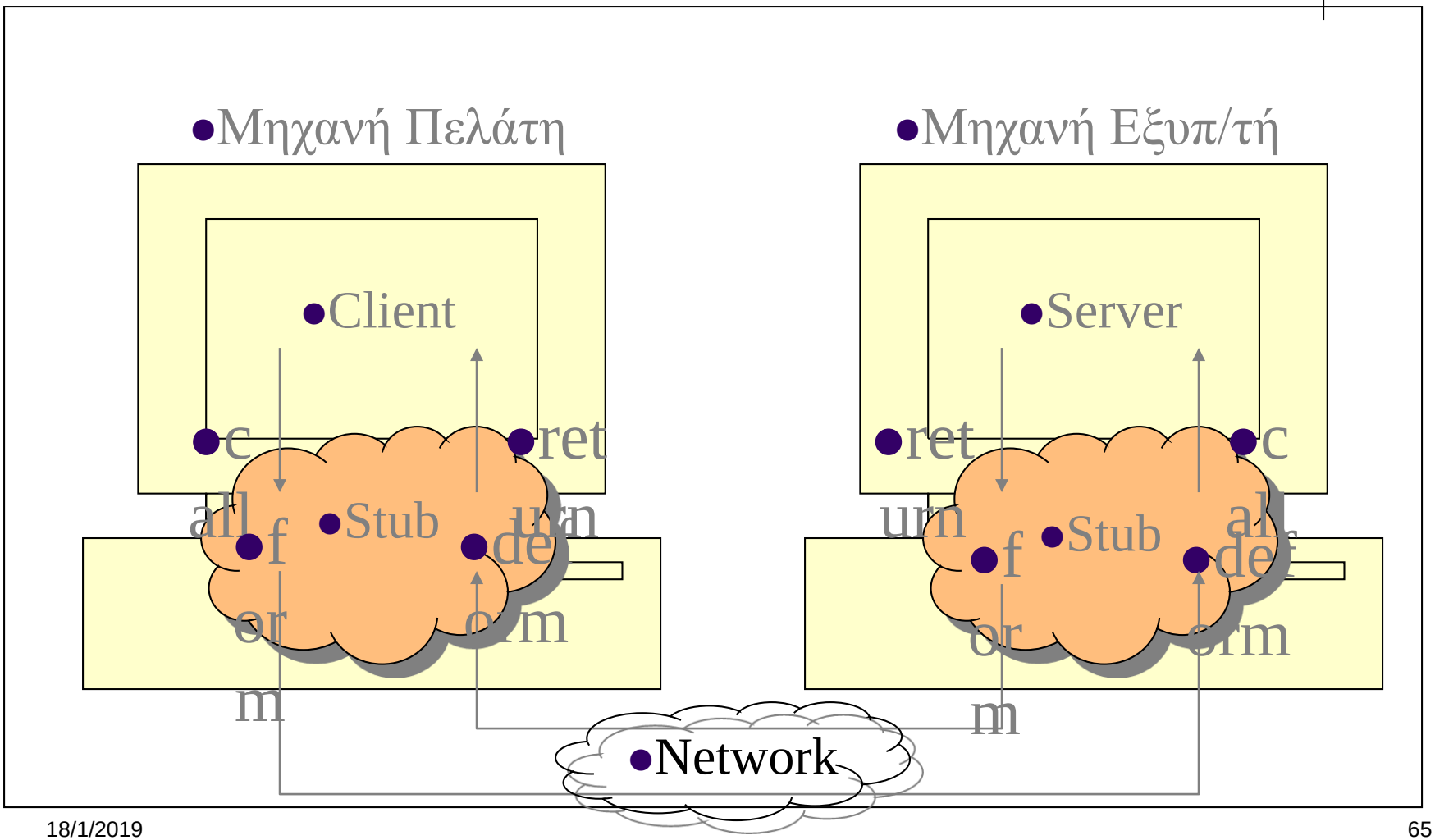
Λειτουργία μηχανισμού RPC



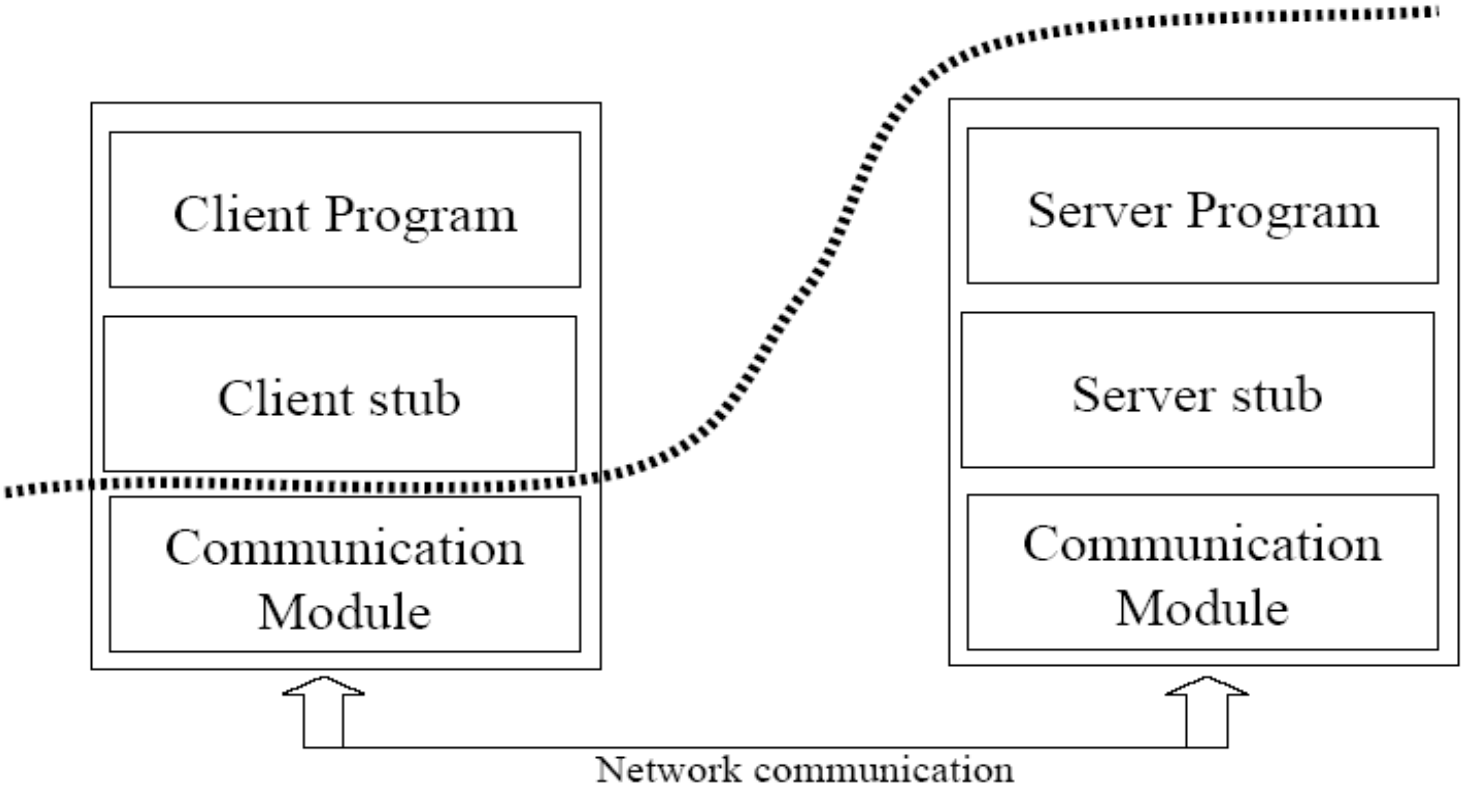
Η διαφάνεια της λειτουργίας του μηχανισμού RPC επιτυγχάνεται με την χρήση στελεχών-υποκατάστατων (stubs) client & server της διαδικασίας που καλείται

- **Στέλεχος πελάτη (client stub)**
  - **Ίδια κλητική ακολουθία με διαδικασία**
  - **Εκδοχή της απομακρυσμένης διαδικασίας στον πελάτη**
- **Στέλεχος διακομιστή (server stub)**
  - **Εκτελεί την κλήση της διαδικασίας**
  - **Μετατρέπει τις εισερχόμενες κλήσεις σε τοπικές κλήσεις**
- **Υλοποίηση της απομακρυσμένης κλήσης με 2 τοπικές κλήσεις**
  - **Κλήση στελέχους πελάτη τοπικά από πελάτη**
  - **Κλήση διαδικασίας τοπικά από το στέλεχος διακομιστή**

Κλήση Απομακρυσμένων Διαδικασιών

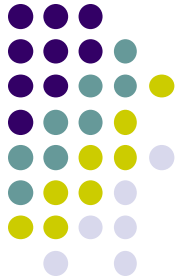


Client και Server



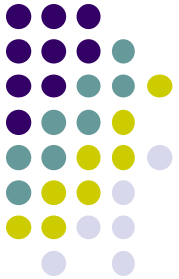
Client program only communicates with the client stab

Βήματα για απομακρυσμένη κλήση



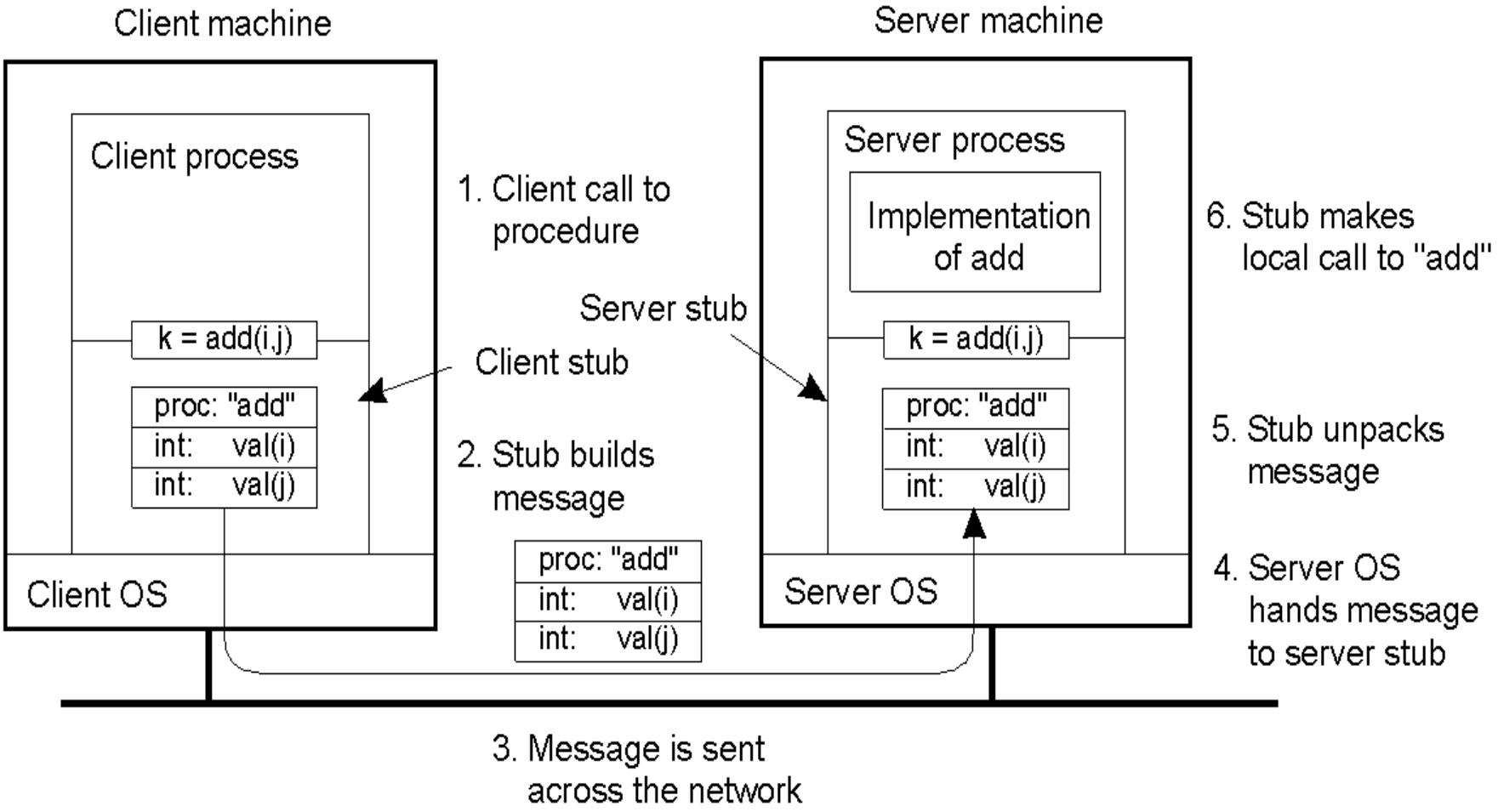
- **Η διαδικασία του πελάτη καλεί το στέλεχος πελάτη**
- **Το στέλεχος πελάτη σχηματίζει ένα μήνυμα και καλεί το τοπικό ΛΣ που στέλνει το μήνυμα στο απομακρυσμένο ΛΣ**
- **Το απομακρυσμένο ΛΣ παραδίδει το μήνυμα στο στέλεχος διακομιστή**
- **Το στέλεχος διακομιστή ξεπακετάρει τις παραμέτρους από το μήνυμα και καλεί τον διακομιστή για να το εκτελέσει**
- **Ο διακομιστής επιστρέφει τα αποτελέσματα στο στέλεχος του διακομιστή, το οποίο σχηματίζει μήνυμα με τα αποτελέσματα και καλεί το τοπικό ΛΣ**
- **Το ΛΣ του διακομιστή στέλνει το μήνυμα στο ΛΣ του πελάτη**
- **Το ΛΣ του πελάτη παραδίδει το μήνυμα στο στέλεχος πελάτη που ξεπακετάρει και επιστρέφει τα αποτελέσματα στον πελάτη**

Μεταβίβαση Παραμέτρων



Πληροφορίες δίνονται από την καλούσα και την καλούμενη διεργασία με τη μορφή παραμέτρων

Μεταβίβαση παραμέτρων κατ' αξία



Μεταβίβαση παραμέτρων κατ' αναφορά

- **Ένας δείκτης έχει νόημα μόνο μέσα στο χώρο διευθύνσεων της διεργασίας όπου χρησιμοποιείται.**
- **Επομένως δεν μπορούν να μεταβιβαστούν δείκτες.**

Τι μπορεί να γίνει?

Ας υποθέσουμε ότι έχουμε ως παράμετρο της διαδικασίας, έναν δείκτη σε έναν πίνακα χαρακτήρων (buf) και ότι γνωρίζουμε το μέγεθος του πίνακα.

Τότε μπορεί να λάβει χώρα η ακόλουθη διαδικασία η οποία αντικαθιστά την μεταβίβαση παραμέτρων κατ' αναφορά με αντιγραφή-επαναφορά.

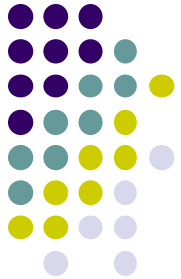
Μεταβίβαση παραμέτρων με αντιγραφή-επαναφορά.

1. **Ο πίνακας αντιγράφεται μέσα στο μήνυμα και στέλνεται στο διακομιστή.**
2. **Το στέλεχος διακομιστή καλεί το διακομιστή με παράμετρο ένα δείκτη σε αυτόν τον πίνακα** (ο δείκτης θα έχει διαφορετική τιμή από τον αρχικό). **Οι αλλαγές που κάνει ο διακομιστής, επηρεάζουν τον χώρο προσωρινής αποθήκευσης του μηνύματος στο στέλεχος του διακομιστή.**
3. **Όταν ο διακομιστής τελειώσει, το αρχικό μήνυμα στέλνεται στο στέλεχος πελάτη.**
4. **Το στέλεχος πελάτη αντιγράφει τον πίνακα πίσω στον πελάτη.**

Βελτιστοποίηση: η μία αντιγραφή μπορεί να παραλειφθεί.

- **Αν ο πίνακας είναι είσοδος για το διακομιστή δε χρειάζεται να σταλεί στο στέλεχος πελάτη.**
- **Αν είναι έξοδος δε χρειάζεται να σταλεί αρχικά στο διακομιστή.**

Μεταβίβαση παραμέτρων με αντιγραφή-επαναφορά



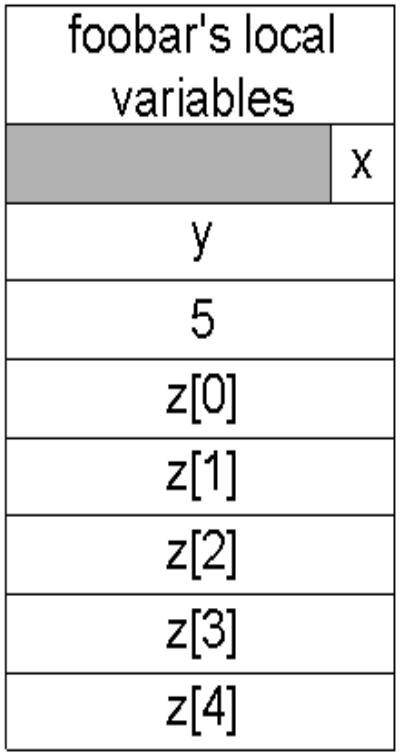
- **Είναι δυνατός ο χειρισμός δεικτών σε πίνακες με γνωστό μέγεθος.**

- **Δεν είναι δυνατός ο χειρισμός δεικτών σε μια οποιαδήποτε δομή δεδομένων. Για το χειρισμό τέτοιων δεικτών επιχειρούνται λύσεις όπως: μεταβιβάζεται ο δείκτης στο στέλεχος διακομιστή και στέλνεται πίσω στον πελάτη αίτηση για την αποστολή των αναφερόμενων δεδομένων.**

Προσδιορισμός παραμέτρων

```
foobar( char x; float y; int z[5] )
{
   ....
}
```

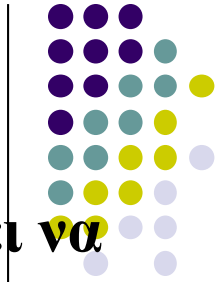
(a)



(b)

(α) μια διαδικασία (β) το αντίστοιχο μήνυμα

●Ορισμός πρωτοκόλλου RPC



Και οι δύο πλευρές μιας απομακρυσμένης κλήσης θα πρέπει να ακολουθούν το ίδιο πρωτόκολλο.

- **Ορισμός μορφής μηνυμάτων – προσδιορισμός παραμέτρων**
- **Ο πελάτης και ο διακομιστής θα πρέπει να συμφωνούν στην αναπαράσταση των απλών δομών δεδομένων (int, float, char) ώστε τα μηνύματα να ερμηνεύονται χωρίς ασάφειες.**
- **Ο πελάτης και ο διακομιστής θα πρέπει να συμφωνούν για το πώς θα γίνει η ίδια η ανταλλαγή μηνυμάτων (π.χ. χρήση TCP/IP ή UDP σε συνδυασμό με ένα μηχανισμό ελέγχου σφαλμάτων που υλοποιούν οι ίδιοι στο πλαίσιο του πρωτοκόλου RPC).**

Υλοποίηση στελεχών

- **Τα στελέχη για το ίδιο πρωτόκολλο αλλά για διαφορετικές διαδικασίες διαφέρουν μόνο στη διασύνδεσή τους με τις εφαρμογές.**
- **Μία διασύνδεση αποτελείται από μια συλλογή διαδικασιών που μπορούν να καλούνται από έναν πελάτη και υλοποιούνται από έναν διακομιστή.**
- **Οι διασυνδέσεις ορίζονται σε μια γλώσσα ορισμού διασυνδέσεων (Interface Definition Language – IDL)**
- **Μία διασύνδεση μεταγλωττίζεται σε ένα στέλεχος πελάτη και ένα στέλεχος διακομιστή.**

Όλα τα συστήματα ενδιάμεσου λογισμικού που βασίζονται σε RPC παρέχουν μια γλώσσα IDL για την υποστήριξη της ανάπτυξης εφαρμογών.

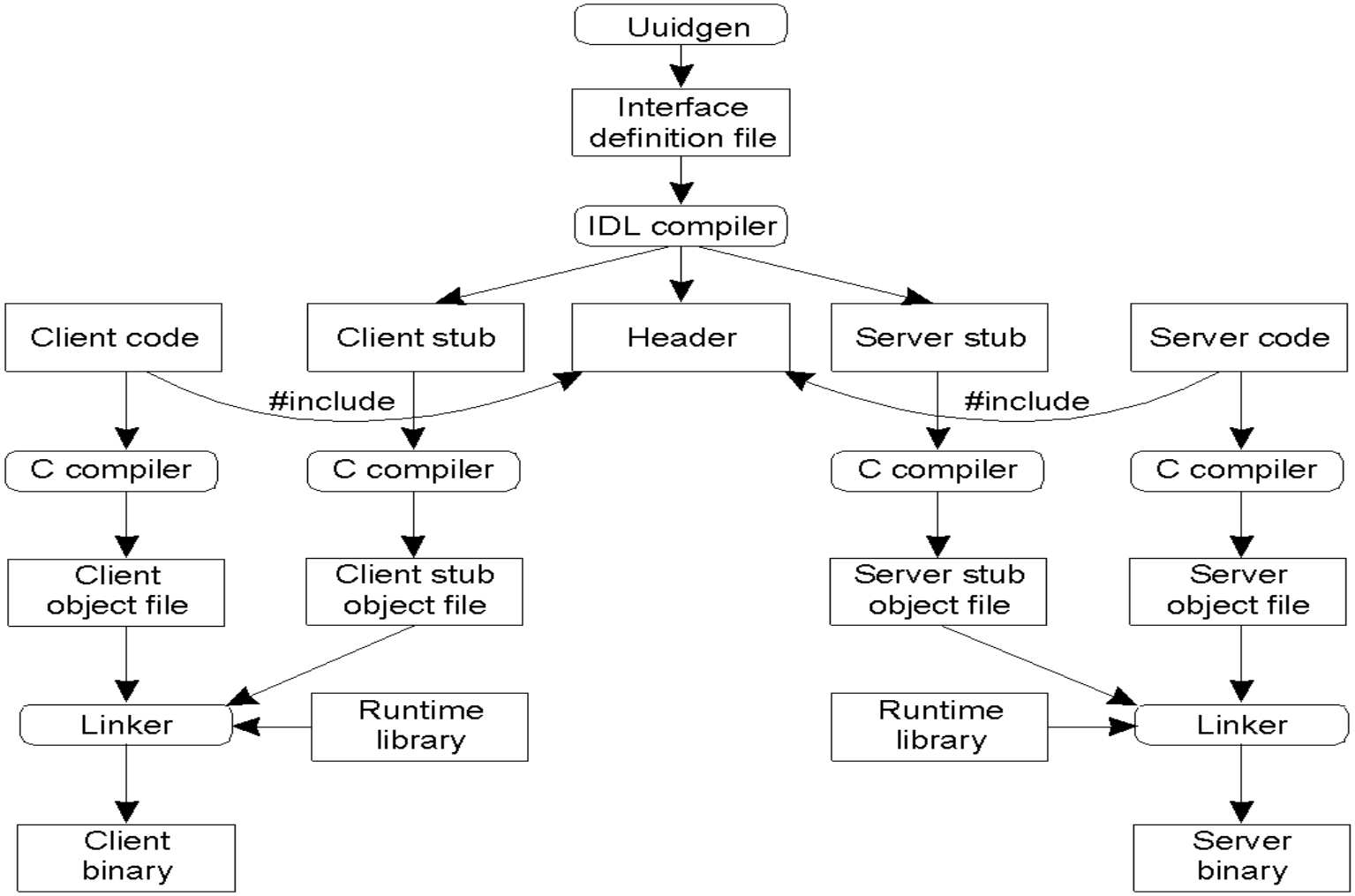
Διαφάνεια RPC

- Συντακτική (syntactic) και σημασιολογική (semantic) διαφάνεια
- Η συντακτική διαφάνεια δεν είναι δύσκολο να επιτευχθεί, αφού οι απομακρυσμένες κλήσεις ακολουθούν το ίδιο μοντέλο με τις τοπικές
  - Κλήση απομακρυσμένης διαδικασίας από μία διεργασία και αναστολή της διεργασίας μέχρι την επιστροφή της κλήσης
  - Πέρασμα παραμέτρων - επιστροφή αποτελεσμάτων στην κλήση
- Η πλήρης σημασιολογική διαφάνεια είναι δύσκολο να επιτευχθεί αφού οι απομακρυσμένες διαδικασίες
  - εκτελούνται σε ξεχωριστό address space και απαιτούν ειδικούς μηχανισμούς (call-by-copy/restore) αντί call by value/reference
  - παρουσιάζουν σφάλματα λόγω διαφορετικών μηχανών, δικτύου
  - χρειάζονται 10-1000 φορές περισσότερο χρόνο, γεγονός που πρέπει να ξέρει η διεργασία πελάτη

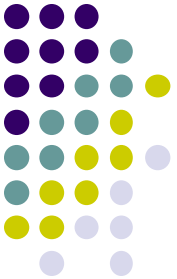
Distributed Computing Environment (DCE RPC)

- **Είναι σύστημα ενδιάμεσου λογισμικού που σχεδιάσθηκε από την OSF (Open Software Foundation – Open Group). Σχεδιάσθηκε για το UNIX**
- **Μπορεί να εντοπίζει αυτόματα το σωστό διακομιστή και να εγκαθιστά την επικοινωνία μεταξύ πελάτη και διακομιστή (συνδυασμός – binding).**
- **Χειρίζεται αυτόματα μετατροπές τύπων δεδομένων, ακόμη και αν ο πελάτης και ο διακομιστής εκτελούνται σε διαφορετικές αρχιτεκτονικές**
- **Αποτελείται από ένα πλήθος στοιχείων (γλώσσες, βιβλιοθήκες, βοηθητικά προγράμματα) προκειμένου να είναι δυνατή η συγγραφή πελατών και διακομιστών.**

Δημιουργία πελάτη και διακομιστή στο DCE RPC

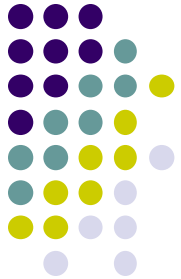


Δημιουργία πελάτη και διακομιστή στο DCE RPC



- **Τα αρχεία IDL περιλαμβάνουν τις δηλώσεις των διαδικασιών, ορισμούς τύπων, δηλώσεις σταθερών και άλλες πληροφορίες για τη σωστή παράταξη των παραμέτρων.**
- **Κάθε αρχείο IDL περιέχει ένα καθολικά μοναδικό αναγνωριστικό για τη διασύνδεση που ορίζεται. Ο πελάτης στέλνει αυτό το αναγνωριστικό με το πρώτο μήνυμα RPC και ο διακομιστής επαληθεύει αν είναι σωστό.**

Βήματα συγγραφής εφαρμογής πελάτη διακομιστή



- **Κλήση προς το πρόγραμμα uuidgen το οποίο παράγει αρχείο IDL με περιέχει μοναδικό κωδικό**
- **Επεξεργασία του αρχείου IDL με τη συμπλήρωση των ονομάτων των απομακρυσμένων διαδικασιών και των παραμέτρων τους**
- **Η έξοδος του μεταγλωττιστή IDL αποτελείται από:**

  (α) αρχείο κεφαλίδων (μοναδικό αναγνωριστικό, ορισμούς τύπων, κ.λπ.)

  (β) στέλεχος πελάτη (περιλαμβάνει τις διαδικασίες που είναι υπεύθυνες για τη συλλογή και συσκευασία των παραμέτρων στο εξερχόμενο μήνυμα και την κλήση του runtime system για την αποστολή του μηνύματος)

  (γ) στέλεχος διακομιστή (περιλαμβάνει τις διαδικασίες που είναι υπεύθυνες για την κλήση των διαδικασιών διακομιστή που διεκπεραιώνουν την εργασία)
- **Συγγραφή του κώδικα του πελάτη και του διακομιστή**

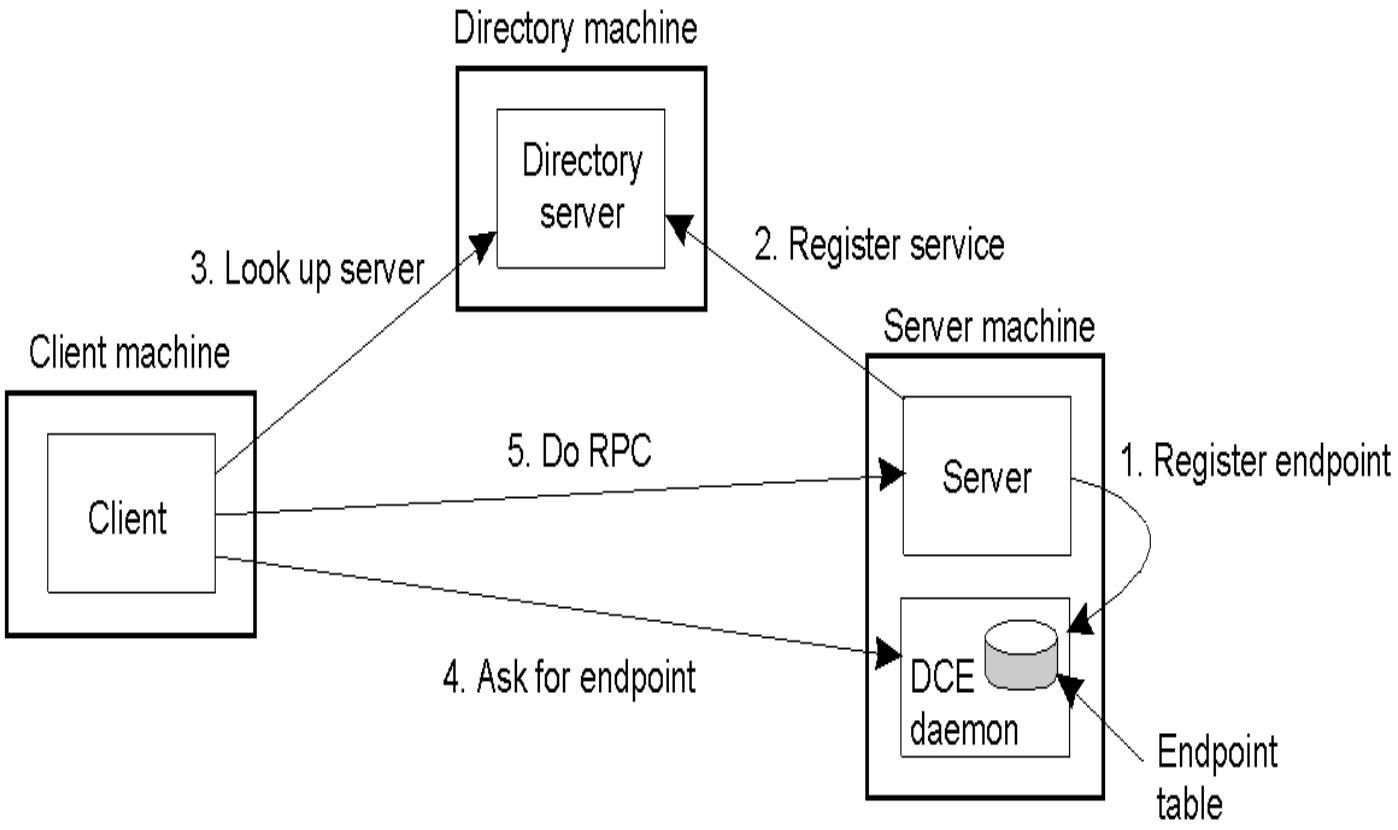
Συνδυασμός (binding) πελάτη και διακομιστή

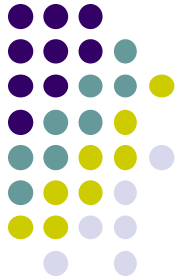
Εντοπισμός του διακομιστή (διεργασίας):

1. **Εντοπισμός της μηχανής του διακομιστή μέσω της υπηρεσίας ευρετηρίου**
   - **Η διεύθυνση της υπηρεσίας ευρετηρίου πρέπει να είναι γνωστή (π.χ. fixed ή μέσω αρχικού broadacst)**
   - **Ο διακομιστής (διεύθυνση δικτύου και όνομα διακομιστή) καταχωρίζεται στην υπηρεσία ευρετηρίου**
   - **Ο διακομιστής μπορεί να διαγραφεί από την υπηρεσία ευρετηρίου όταν δεν επιθυμεί να προσφέρει πλέον υπηρεσίες**

2. **Εντοπισμός του διακομιστή (διεργασίας) στη μηχανή διακομιστή**
   - **Για να επικοινωνήσει ο πελάτης με έναν διακομιστή πρέπει να γνωρίζει ένα ακραίο σημείο (θύρα) στη μηχανή του διακομιστή όπου θα στείλει το μήνυμα (τα ακραία σημεία χρησιμοποιούνται από το Λ.Σ. για τη διάκριση των μηνυμάτων που απευθύνονται σε διαφορετικές διεργασίες).**
   - **Ο Δαίμονας DCE σε κάθε διακομιστή κρατάει πίνακα ζευγών *(διεργασία,ακραίο_σημείο)***

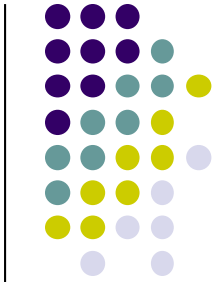
Συνδυασμός (binding) πελάτη και διακομιστή στο DCE





Κλήση απομακρυσμένων μεθόδων - Remote Method Invocation (RMI)

Αντικείμενα



- Οι **κλάσεις (classes)** ορίζουν τύπους αντικειμένων.
- Τα **αντικείμενα** (objects) είναι **στιγμιότυπα (instances)** των κλάσεων
- Ένα αντικείμενο ενθυλακώνει (encapsulates): δεδομένα (**κατάσταση**) και λειτουργίες (**μέθοδοι**)
- Οι μέθοδοι γίνονται διαθέσιμες μέσω μιας **διασύνδεσης (interface)**
- Ένα αντικείμενο μπορεί να υλοποιεί πολλές διασυνδέσεις.
- **Κατανεμημένο αντικείμενο:** ένα αντικείμενο του οποίου οι μέθοδοι μπορούν να κληθούν εξ' αποστάσεως. Το αντικείμενο και η διασύνδεσή του βρίσκεται σε ένα μηχάνημα αλλά η διασύνδεσή του μπορεί να γίνεται διαθέσιμη σε απομακρυσμένες διεργασίες

Κλήση απομακρυσμένων μεθόδων (RMI)

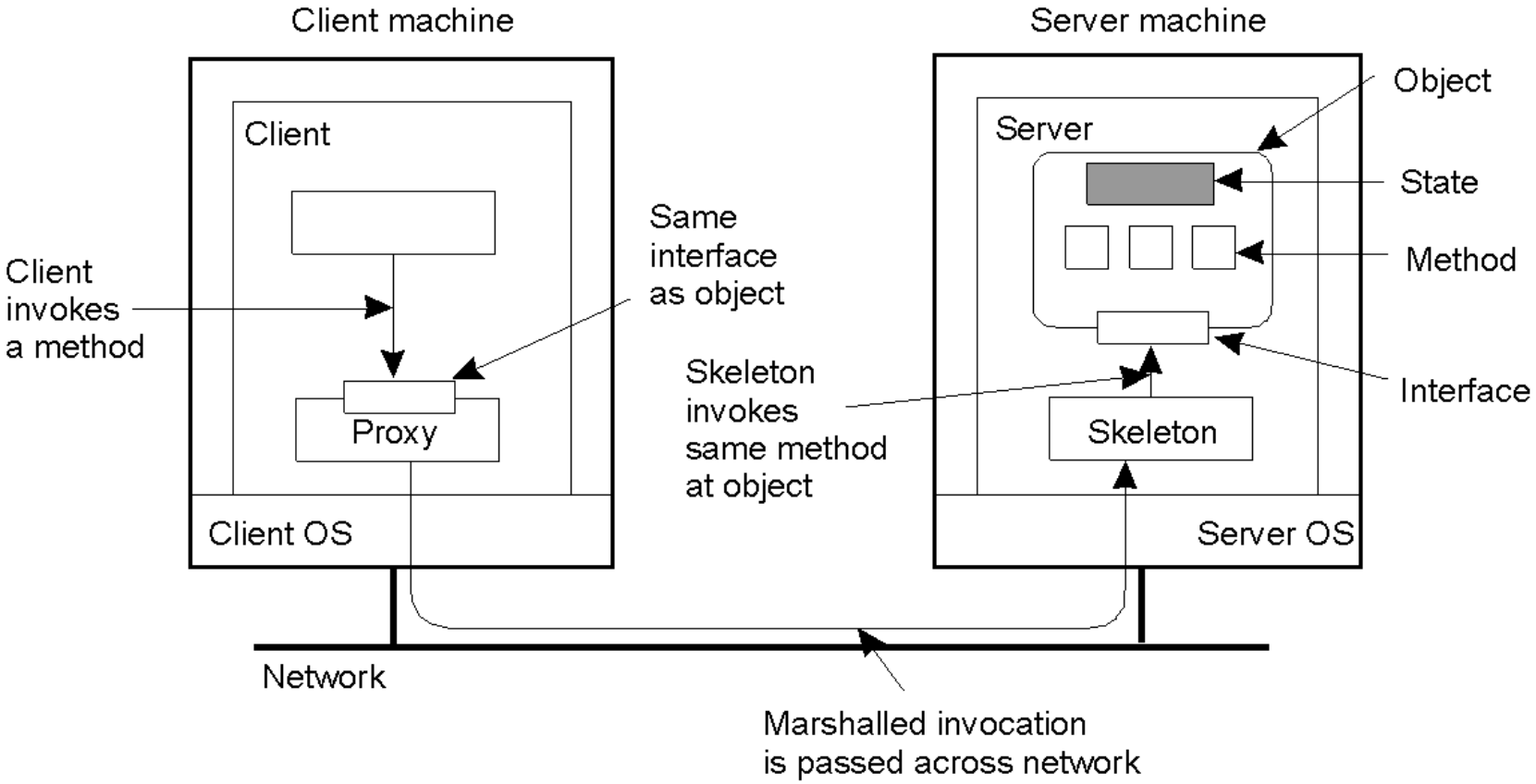
RMI: Αν ένας πελάτης **«συνδυαστεί με ένα αντικείμενο» τότε μπορεί** να καλέσει τις μεθόδους του αντικειμένου μέσω του **εξουσιοδότη** ή **διαμεσολαβητή**.

Αντικείμενο Εξουσιοδότης ή Διαμεσολαβητής (proxy): μια υλοποίηση της διασύνδεσης ενός αντικειμένου που φορτώνεται στο χώρο διευθύνσεων του πελάτη όταν ο πελάτης συνδυάζεται (bind) με το αντικείμενο.

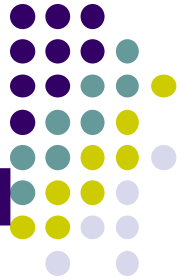
Ο διαμεσολαβητής παρατάσσει αιτήσεις κλήσεων μεθόδων σε μηνύματα και αποπαρατάσσει τις απαντήσεις

Αντικείμενο Σκελετός (skeleton): στέλεχος διακομιστή που παραλαμβάνει τις αιτήσεις, τις αποπαρατάσσει σε κλήσεις μεθόδων της διασύνδεσης που βρίσκεται στον διακομιστή και παρατάσσει και προωθεί τις απαντήσεις στον διαμεσολαβητή.

Κατανεμημένα Αντικείμενα

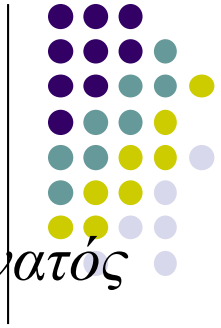


ΘΕΜΑΤΑ ΚΑΤΑΝΕΜΗΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ



- **Πρότυπο Κατανεμημένου Υπολογισμού**
- **Λογικά Ρολόγια /Χρονοσφραγίδες Lamport**
- **Διανυσματικές χρονοσφραγίδες**
- **Διασφάλιση αιτιοκρατικής παράδοσης μηνυμάτων**
- **Εκλογή Αρχηγού**
- **Αμοιβαίος Αποκλεισμός**
- **Ανοχή σε βλάβες / Μοντέλα Αστοχιών**
- **Συγκάλυψη αστοχιών μέσω Υπερεπάρκειας/Αναπαραγωγής**
- **Συμφωνία σε κατανεμημένα συστήματα με σφάλματα**
- **Κατανεμημένη Δέσμευση**

Λογικά Ρολόγια



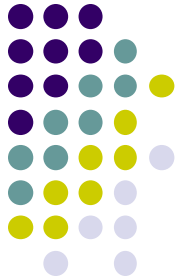
Lamport (1978): *Παρόλο που ο συγχρονισμός ρολογιών είναι δυνατός δεν είναι απαραίτητο να είναι απόλυτος*

Συνήθως, έχει **σημασία η συμφωνία όλων των διεργασιών ως προς τη σειρά με την οποία λαμβάνουν χώρα τα διαφορετικά γεγονότα** και όχι η συμφωνία τους για την ακριβή ώρα.

Λογικά ρολόγια: τα ρολόγια που σχετίζονται με αλγορίθμους για τους οποίους σημασία έχει η εσωτερική συνέπεια των ρολογιών και όχι αν αυτά είναι κοντά στην πραγματική ώρα.

Βασική Ιδέα: **Αν το γεγονός A συμβαίνει πριν το B, τότε θα πρέπει: ο χρόνος LT (Logical Time) του A να είναι μικρότερος από τον χρόνο LT (Logical Time) του B, ήτοι LT(A) < LT(B).**

Η σχέση συμβαίνει-πριν

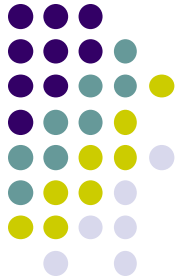


- Αν α και β είναι δύο γεγονότα μέσα στην ίδια διεργασία και το α λάβει χώρα πριν από το β τότε LT(α) < LT(β).

- Αν α είναι η αποστολή μηνύματος από μια διεργασία και β είναι η λήψη του μηνύματος από μια άλλη διεργασία τότε τα LT(α), LT(β) πρέπει να οριστούν έτσι ώστε κάθε διεργασία να συμφωνεί ότι LT(α) < LT(β).

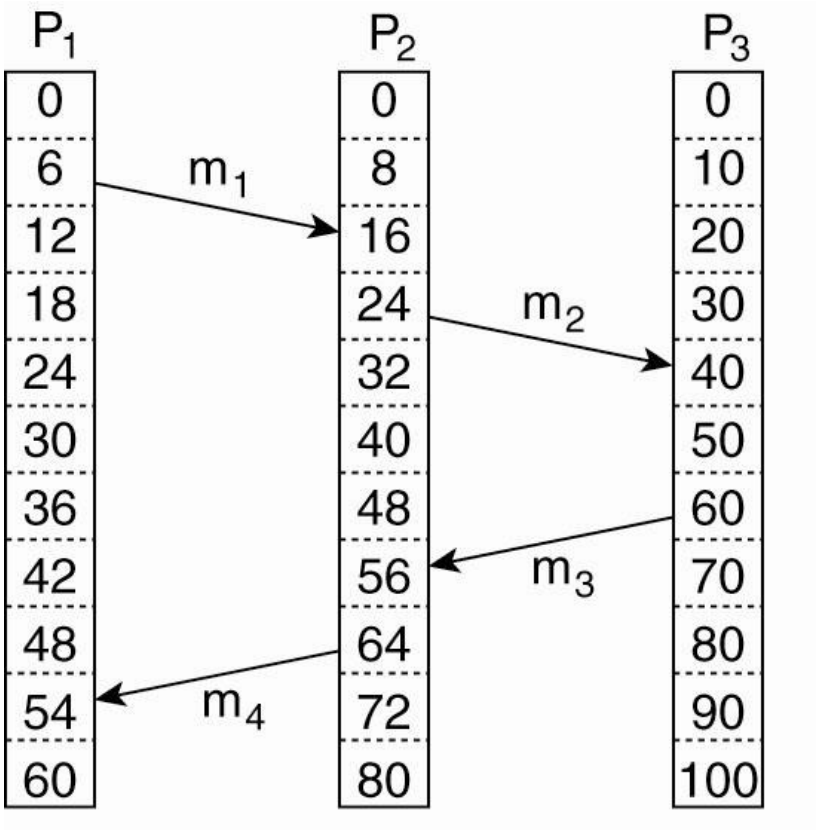
- Αν τα α και β είναι ταυτόχρονα τότε δεν ισχύει αναγκαστικά κάποια σχέση μεταξύ των LT(α), LT(β) .

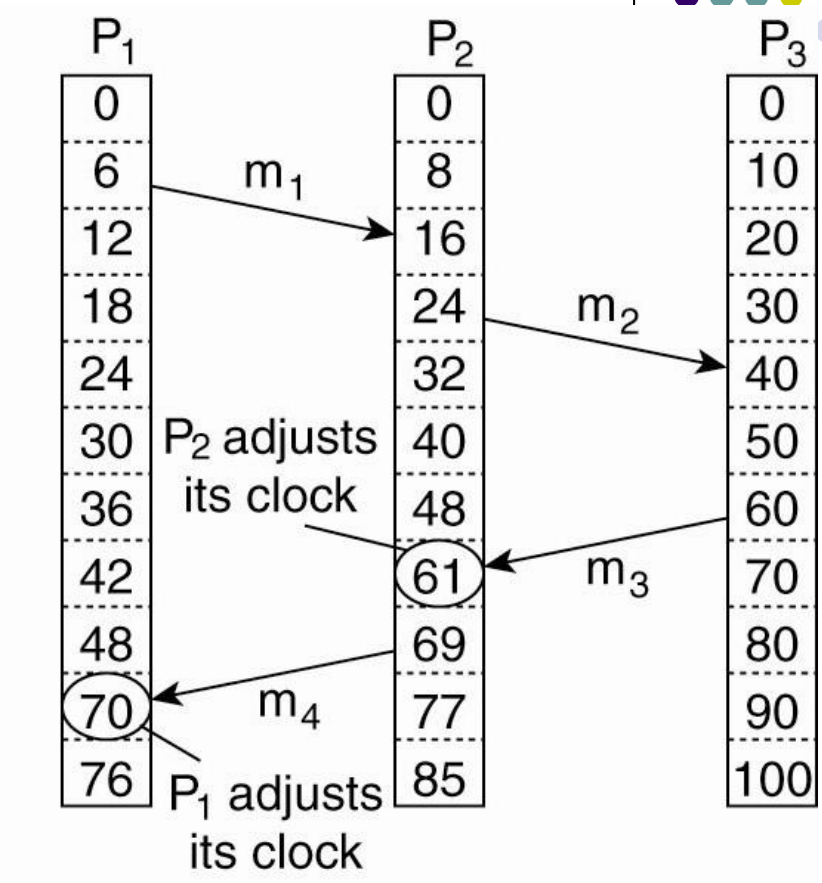
Χρονοσφραγίδες Lamport



- How can we maintain a clock such that if event A **happens-before** B, then LT(A) < LT(B)?
  - If same process, just use machine clock.
  - If different process, we don’t care if there is no message.
  - If there is a message, use that to update the clock somehow.

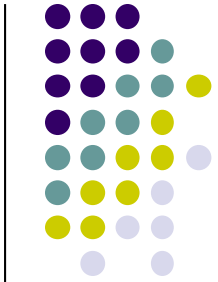
Ο Αλγόριθμος του Lamport





- Three processes, each with its own clock. The clocks run at different rates.
- After receiving a message, adjust clock if necessary.

Ο Αλγόριθμος του Lamport

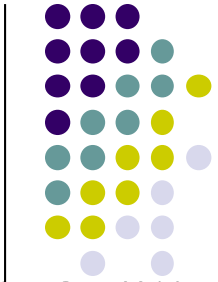


Κάθε διεργασία P_i διατηρεί μια τοπική μεταβλητή LT_i η οποία καλείται λογικό ρολόι και αποθηκεύει μη αρνητικούς ακέραιους και έχει αρχική τιμή 0.

1. Πριν από την εκτέλεση ενός γεγονότος η P_i εκτελεί $LT_i \leftarrow LT_i + 1$.
2. Όταν η διεργασία P_i στέλνει μήνυμα m στην P_j, θέτει την χρονοσφραγίδα *ts(m)* (timestamp) του *m* ίση με LT_i .
3. Όταν η διεργασία P_j λάβει το *m* προσαρμόζει το διάνυσμά της θέτοντας

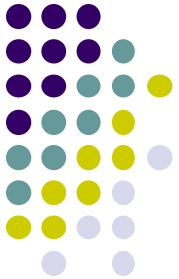
   $LT_j \leftarrow \max\{LT_j , ts\ (m)\}$.

   Στη συνέχεια η P_j εκτελεί το πρώτο βήμα και παραδίδει το μήνυμα στην εφαρμογή.



Ο Αλγόριθμος του Lamport μας δίνει ένα τρόπο να αποδίδουμε την ώρα σε όλα τα συμβάντα ενός κατανεμημένου συστήματος έτσι ώστε:

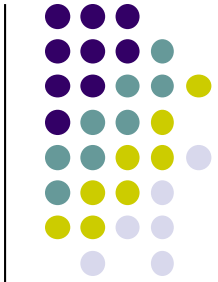
- Αν το α συμβεί πριν από β στην ίδια διεργασία τότε $LT(\alpha) < LT(\beta)$.
- Αν τα α και β αντιπροσωπεύουν την αποστολή και τη λήψη ενός μηνύματος αντίστοιχα, τότε $LT(\alpha) < LT(\beta)$.
- Για όλα τα διαφορετικά συμβάντα $LT(\alpha) \neq LT(\beta)$.



Πρόβλημα Εκλογής Αρχηγού: Επιλογή μιας μόνο διεργασίας αρχηγού/συντονιστή προκειμένου να εκτελέσει ένα συγκεκριμένο καθήκον (εκτέλεση συγκεντρωτικού αλγορίθμου, επαναφορά από αδιέξοδο)

- **Δεν μπορεί αυθαίρετα μια διεργασία να αυτοανακηρυχθεί αρχηγός. Θα πρέπει να προηγηθεί η εκτέλεση ενός αλγορίθμου εκλογής.**
- **Αλγόριθμοι Εκλογής Αρχηγού**

Αλγόριθμοι Εκλογής Αρχηγού



Αρχικά

– ένα αυθαίρετο μη κενό σύνολο διεργασιών

– όλες οι διεργασίες ξεκινάνε από την ίδια κατάσταση (candidate)

Κάθε διεργασία εκτελεί τον ίδιο αλγόριθμο

Υπάρχουν δύο είδη τερματικών καταστάσεων για μια διεργασία:

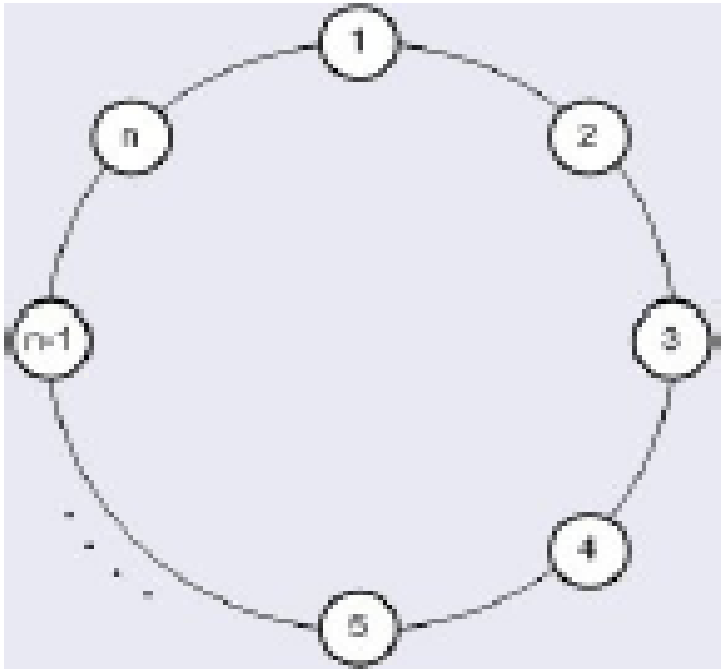
- **η τερματική κατάσταση στην οποία η διεργασία έχει εκλεγεί αρχηγός (*κατάσταση ισχύος ή αρχηγού*) και**
- **η τερματική κατάσταση στην οποία η διεργασία δεν είναι ο αρχηγός (*κατάσταση μη-ισχύος ή χαμένου*).**

Επιτρεπτές εκτελέσεις:

- **Κάθε διεργασία τελικά εισέρχεται σε μια τερματική κατάσταση, ισχύος ή μη.**
- **Μόνο μια διεργασία, ο αρχηγός, μπαίνει σε τερματική κατάσταση ισχύος.**

Εκλογή Αρχηγού σε Δακτύλιο

Έστω ένα κατανεμημένο σύστημα από *n* διεργασίες τοποθετημένες σε ένα δίκτυο δακτυλίου.

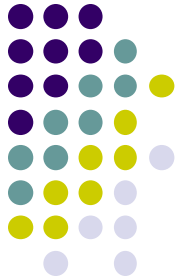


- **Οι διεργασίες έχουν μοναδικές ταυτότητες (IDs).**
- **Οι διεργασίες δεν γνωρίζουν τις ταυτότητες των υπόλοιπων διεργασιών**

Κάθε διεργασία γνωρίζει μόνο ποιος είναι ο αριστερός *(σύμφωνα με τους δείκτες του ρολογιού)* και ποιος ο δεξιός *(αντίθετα με τους δείκτες του ρολογιού)* της γείτονας.

Θεωρούμε ότι οι διεργασίες είναι αριθμημένες από 1 έως n χωρίς οι ίδιες να γνωρίζουν αυτή την αρίθμηση

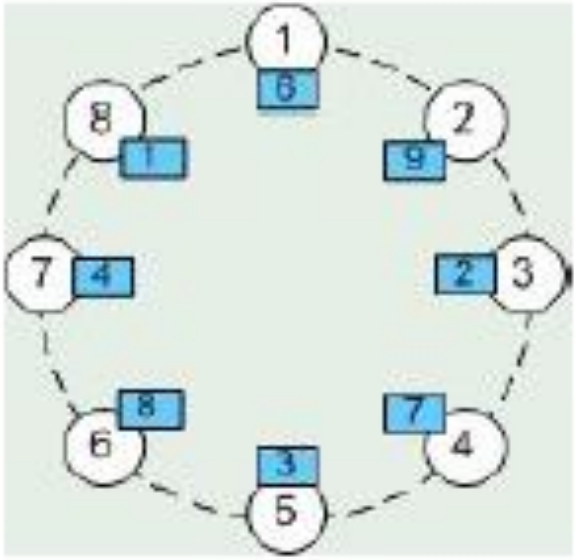
Αλγόριθμος Εκλογής Αρχηγού σε Δακτύλιο

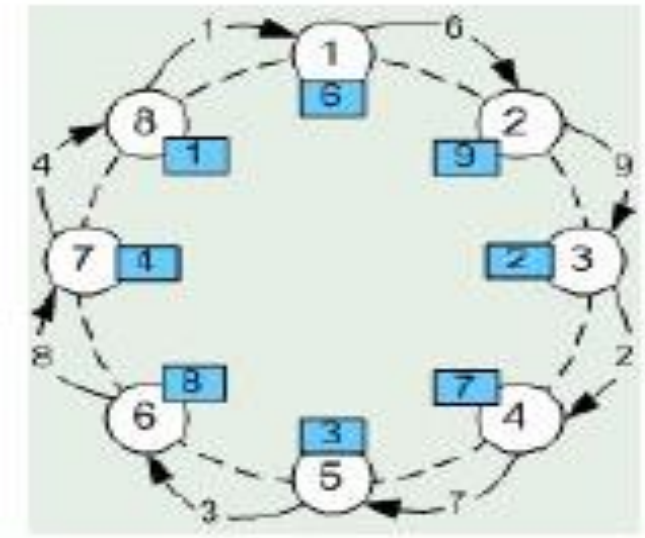


Ενέργειες κάθε διεργασίας p:

- **Αποστολή του ID της στα αριστερά.**
- **Όταν η p λάβει ένα ID (από δεξιά) κάνει τα εξής:**
  - **αν είναι μεγαλύτερο από το δικό της, το προωθεί προς τα αριστερά**
  - **αν είναι μικρότερο από το δικό της, το αγνοεί (και δεν το προωθεί)**
  - **αν είναι ίσο με το δικό της, αποφασίζει πως αυτή είναι ο αρχηγός στο σύστημα και στέλνει ένα μήνυμα τερματισμού προς τα αριστερά**
- **Όταν η p λάβει μήνυμα τερματισμού, το προωθεί προς τα αριστερά και εισέρχεται σε τερματική κατάσταση μη-ισχύος.**

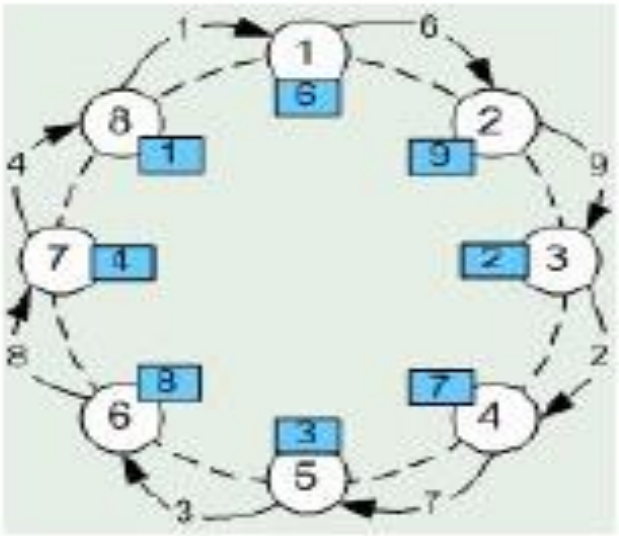
Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο



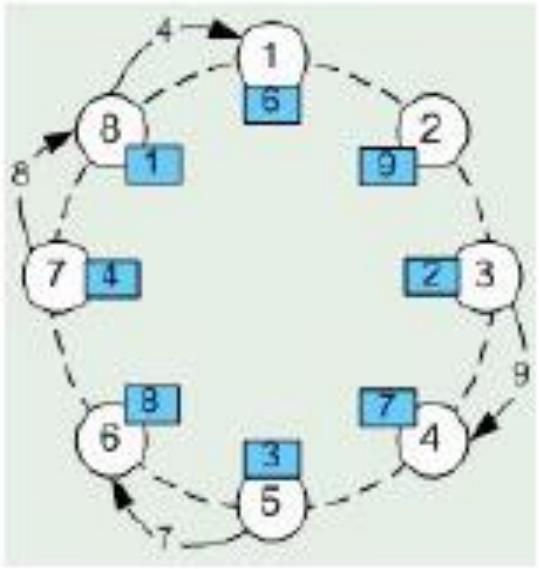


Βήμα 1

Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο

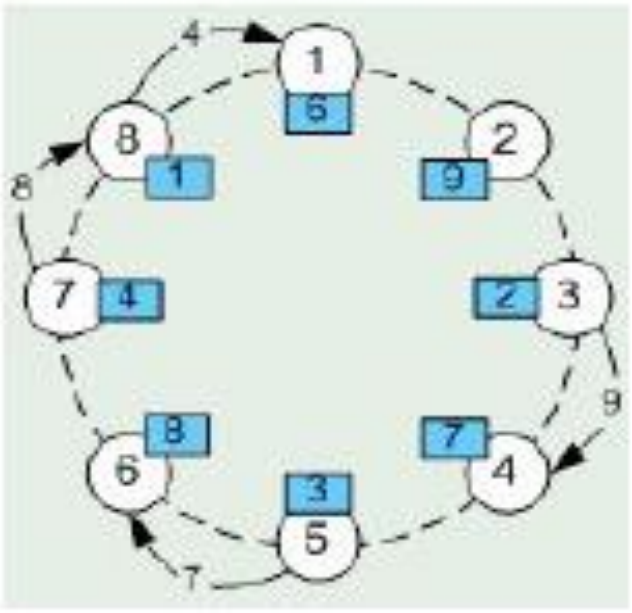


Βήμα 1

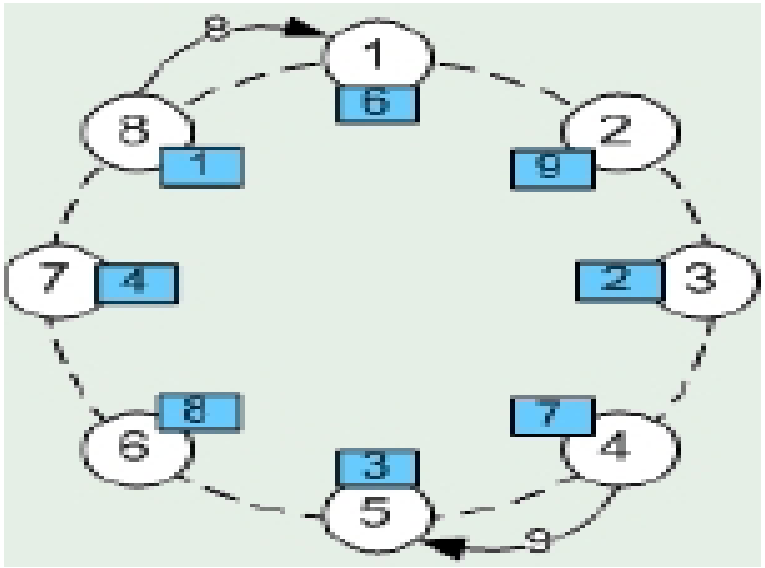


Βήμα 2

Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο

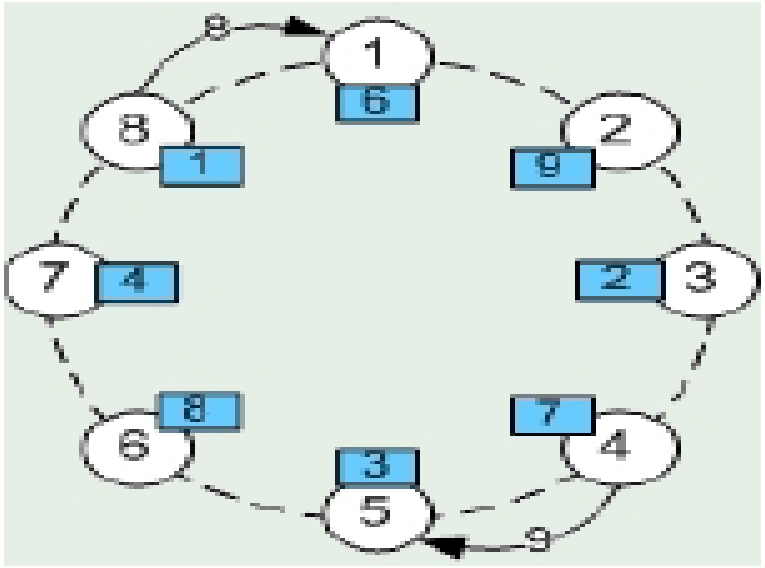


Βήμα 2

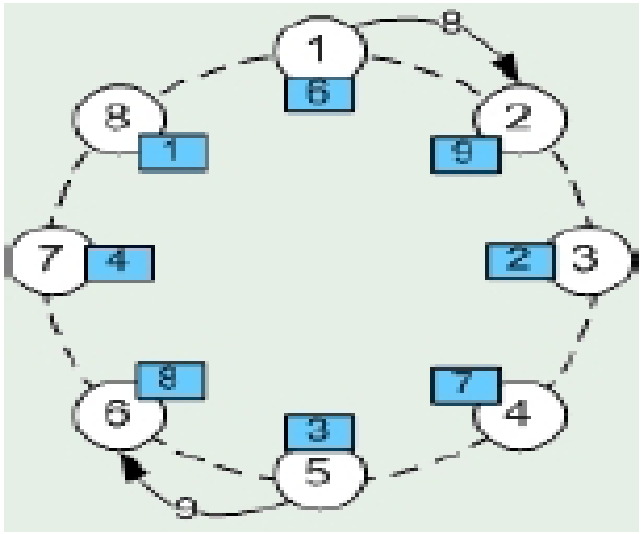


Βήμα 3

Εκτέλεση αλγορίθμου εκλογής αρχηγού σε **σύγχρονο** δακτύλιο

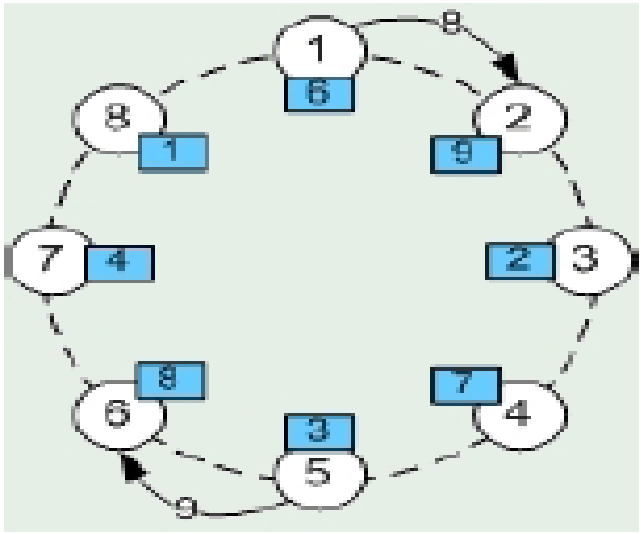


Βήμα 3

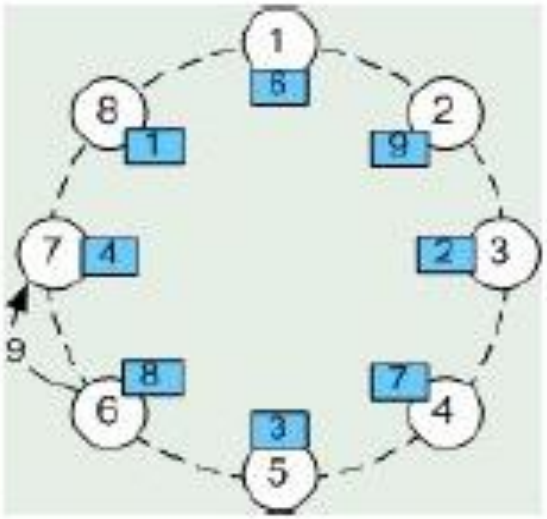


Βήμα 4

Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο

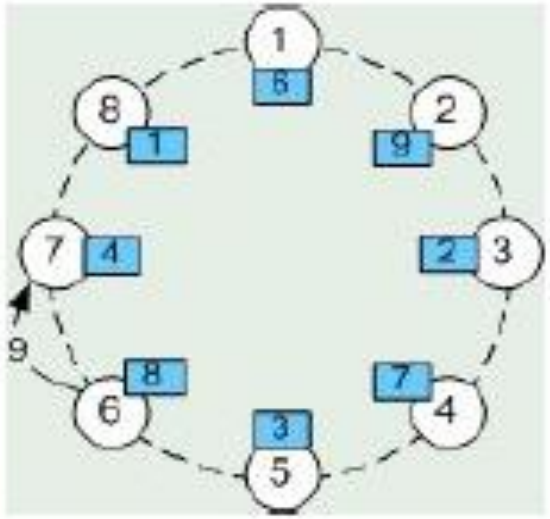


Βήμα 4

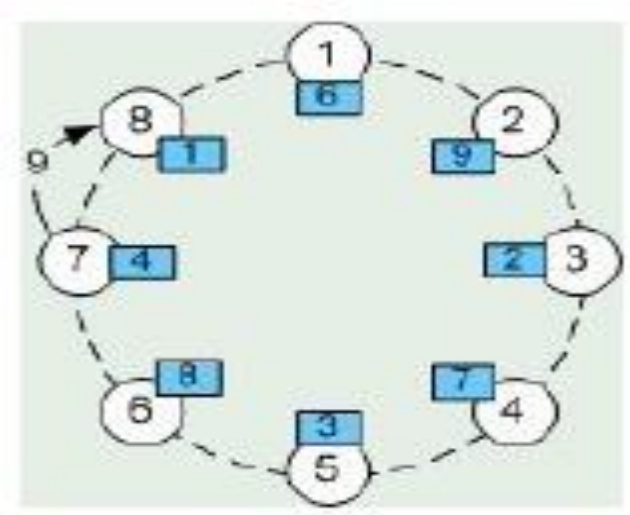


Βήμα 5

Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο

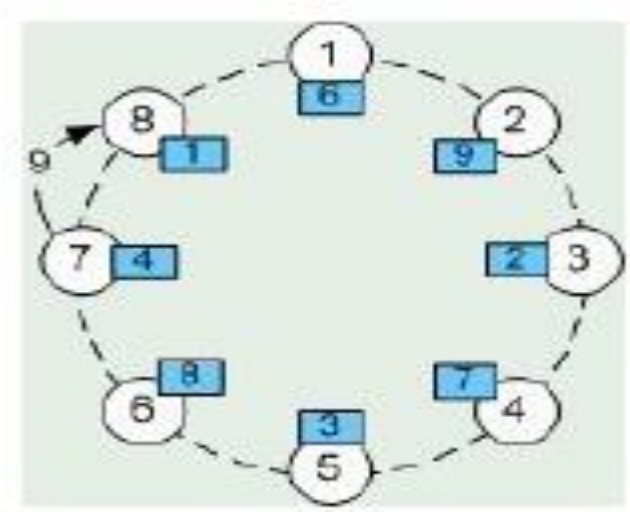


Βήμα 5

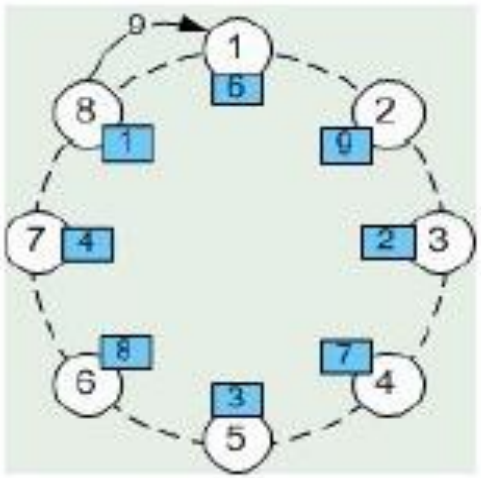


Βήμα 6

Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο

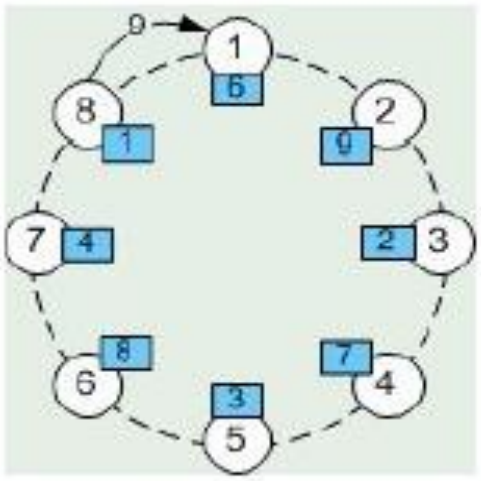


Βήμα 6

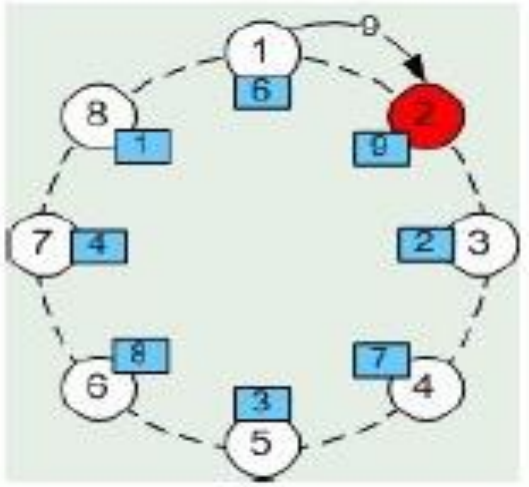


Βήμα 7

Εκτέλεση αλγορίθμου εκλογής αρχηγού σε σύγχρονο δακτύλιο



Βήμα 7



Βήμα 8

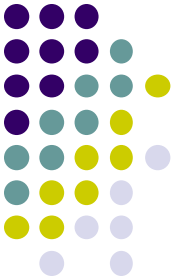
Ορθότητα - Πολυπλοκότητα Επικοινωνίας

Ορθότητα αλγορίθμου: Θα **εκλεγεί ως αρχηγός** η διεργασία με **το μεγαλύτερο ID**. Το μήνυμα με αυτό το ID θα περάσει από όλες τις διεργασίες.

Η Πολυπλοκότητα Επικοινωνίας **εξαρτάται από τη διάταξη των διεργασιών**

- Το μέγιστο ID θα περάσει από όλες τις διεργασίες (*n* μηνύματα)
- Το δεύτερο μέγιστο ID θα ταξιδέψει έως ότου συναντήσει το μέγιστο ID
- Το τρίτο μέγιστο ID θα ταξιδέψει έως ότου συναντήσει το μέγιστο ή το δεύτερο μέγιστο ID.
- κ.ο.κ

Πολυπλοκότητα Επικοινωνίας: απαιτούνται $O(n^2)$ μηνύματα



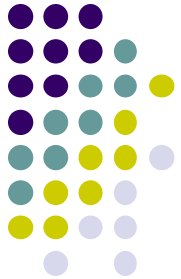
Η **χειρότερη διάταξη των IDs** είναι σε φθίνουσα σειρά κατά την ωρολογιακή φορά

Τότε:

- το δεύτερο μέγιστο ID συνεισφέρει *n-1* μηνύματα
- το τρίτο μέγιστο ID συνεισφέρει *n-2* μηνύματα
- το τέταρτο μέγιστο ID συνεισφέρει *n-3* μηνύματα
- κ.οκ.

Άρα συνολικά απαιτούνται $\sum_{i=1}^{n}(n-i+1) = \sum_{i=1}^{n} i = \Theta(n^2)$ μηνύματα

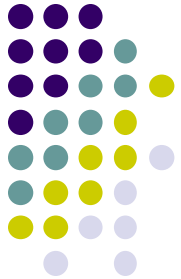
Εκλογή αρχηγού σε Γενικά Δίκτυα
Αλγόριθμος FloodMax



- Ο αλγόριθμος **FloodMax** έχει το προτέρημα της εύκολης υλοποίησης, και της **καλής απόδοσης** κάτω από ορισμένες συνθήκες χρονισμού του δικτύου (σύγχρονα συστήματα).
- Υποθέτει ότι οι διεργασίες **έχουν μοναδικές ταυτότητες** και οι διεργασίες **γνωρίζουν μόνο την δικιά τους** ταυτότητα.
- Υποθέτει ότι οι διεργασίες **συνθέτουν ένα ισχυρά συνεκτικό δίκτυο *n* διεργασιών και *m* καναλιών** (οποιασδήποτε τοπολογίας) και **γνωρίζουν τη διάμετρο *d*** του δικτύου.

- Ο αλγόριθμος **δεν απαιτεί** την ταυτόχρονη εκτέλεση των βημάτων από όλες τις διεργασίες (ασύγχρονα συστήματα). Στην περίπτωση ασύγχρονης εκτέλεσης επιτυγχάνει συγκρίσιμα αποτελέσματα με εκείνα της σύγχρονης.
- Ο αλγόριθμος βασίζεται σε λειτουργίες σύγκρισης των ταυτοτήτων των διεργασιών. Η βασική ιδέα του αλγορίθμου βρίσκεται στην επιβίωση και προώθηση της μεγαλύτερης ταυτότητας που υπάρχει στο δίκτυο. Στο τέλος της εκτέλεσης κάθε διεργασία είναι σε θέση να γνωρίζει αυτή τη μέγιστη ταυτότητα.

Εκλογή αρχηγού σε Γενικά Δίκτυα
Αλγόριθμος FloodMax



Οι διεργασίες διατηρούν τις εξής μεταβλητές:

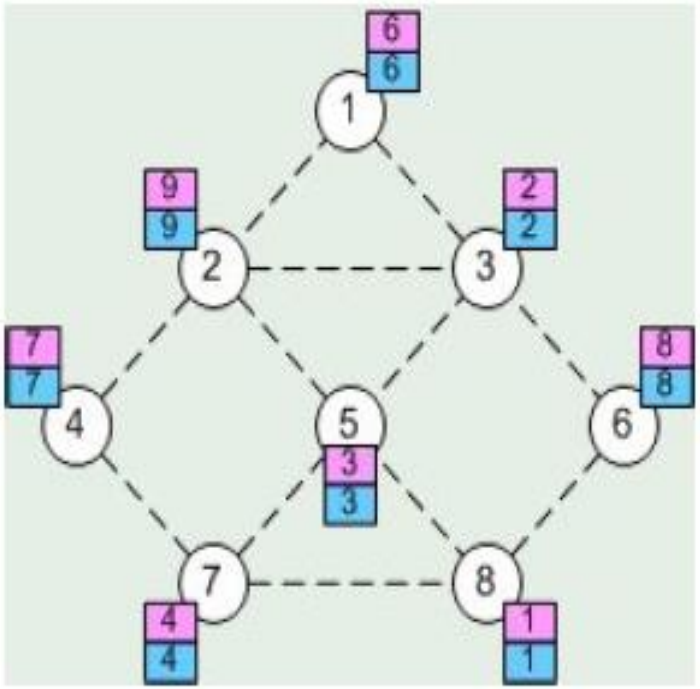
- την **μεταβλητή max_ID** με αρχική τιμή το ***ID*** της διεργασίας
- την **μεταβλητή *leader ={true,false}*** με αρχική τιμή ***false***

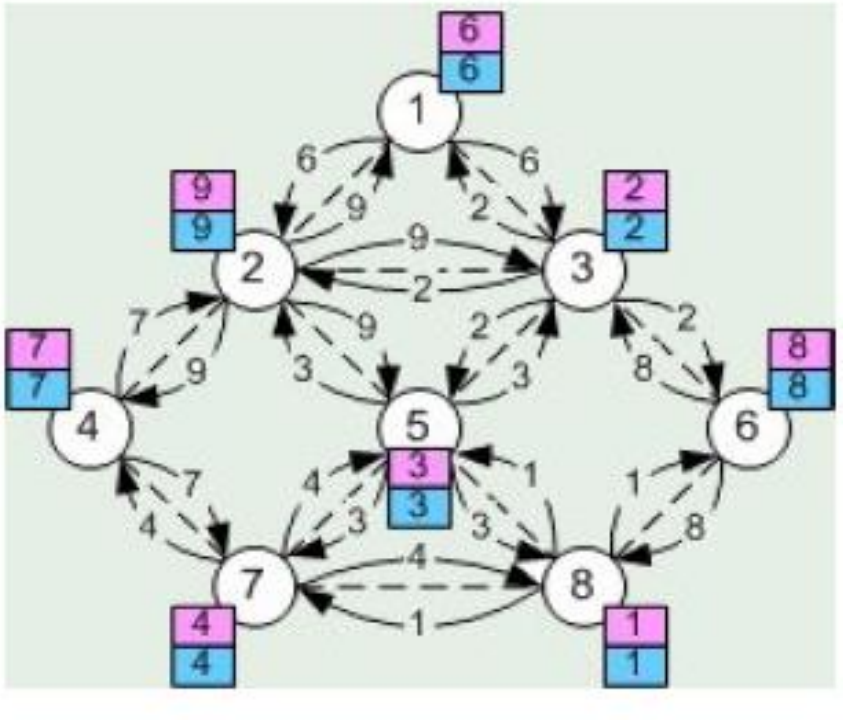
Αλγόριθμος FloodMax

- **Σε κάθε γύρο, οι διεργασίες εκπέμπουν την max_ID σε όλους τους γείτονες.**
- **Μόλις λάβουν μία ταυτότητα από κάποιον γείτονα, την συγκρίνουν με την δική τους max_ID.**
- **Αν είναι μεγαλύτερη, θέτουν την μεταβλητή στην νέα τιμή.**
- **Μετά από *d* γύρους**

  αν η μεταβλητή max_ID ισούται με την ταυτότητα της διεργασίας, η διεργασία μεταβαίνει στην κατάσταση ισχύος θέτοντας την μεταβλητή *leader* στην τιμή *true*

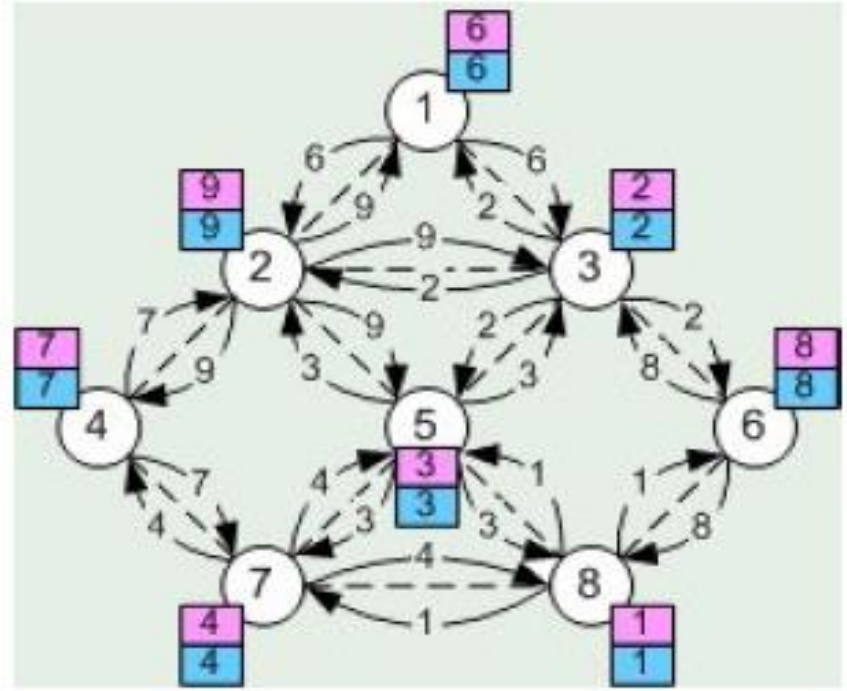
Παράδειγμα Εκτέλεσης του Αλγορίθμου FloodMax



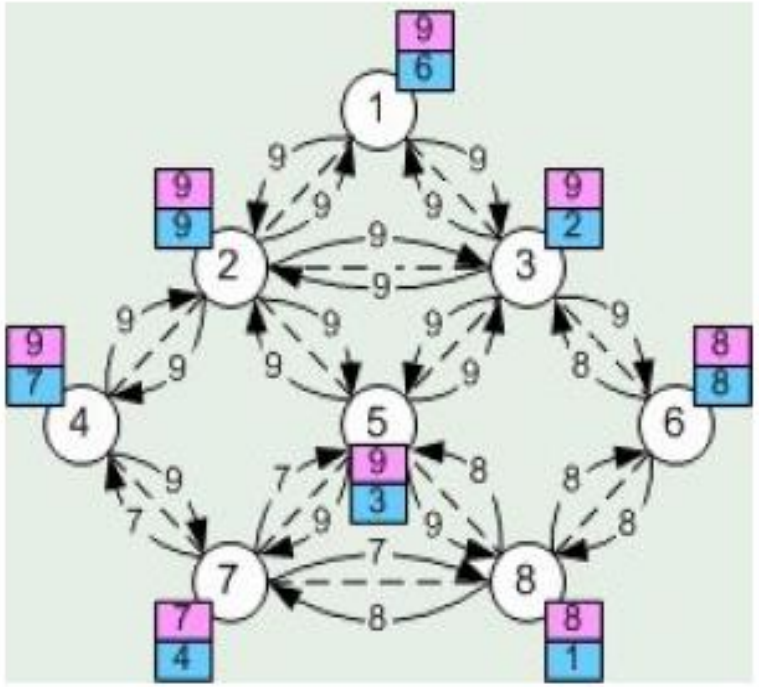


Πρώτος Γύρος

Παράδειγμα Εκτέλεσης του Αλγορίθμου FloodMax

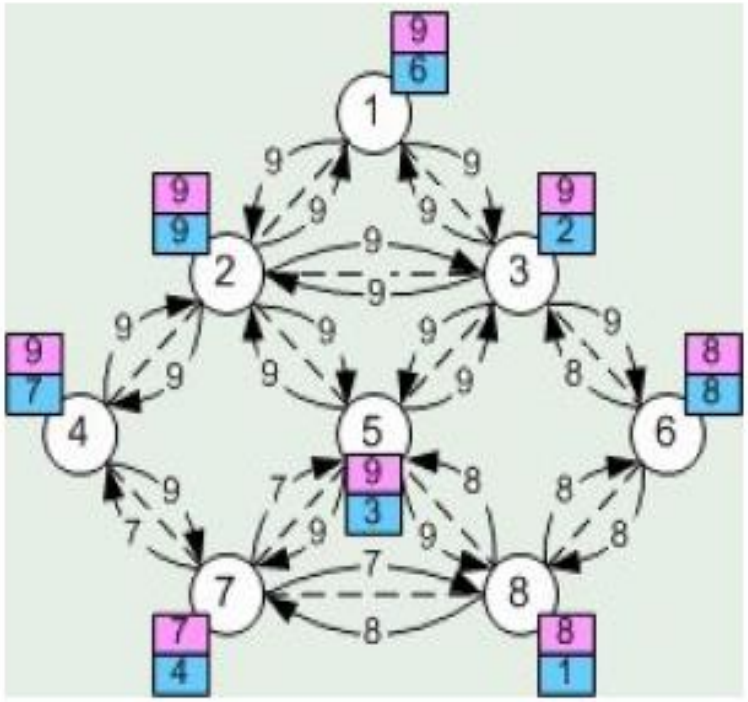


Πρώτος Γύρος

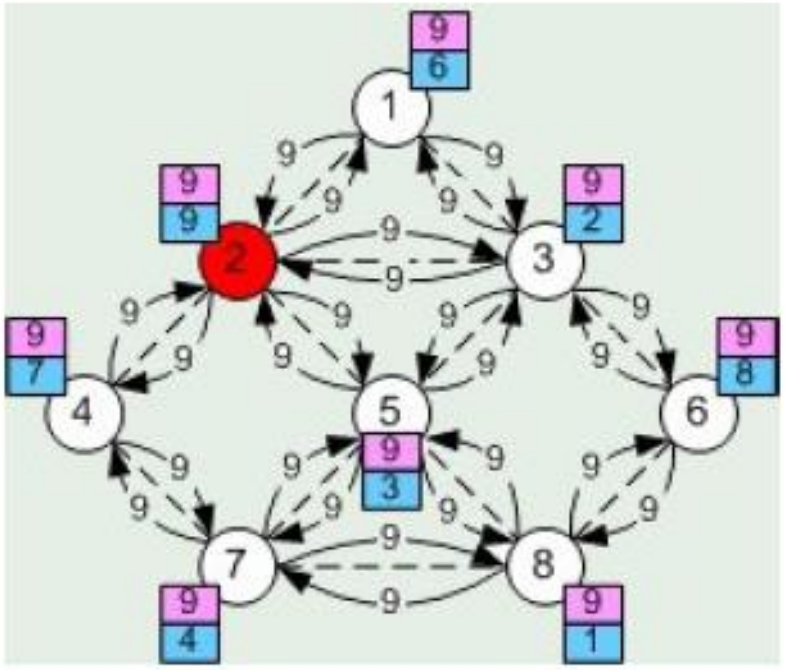


Δεύτερος Γύρος

Παράδειγμα Εκτέλεσης του Αλγορίθμου FloodMax

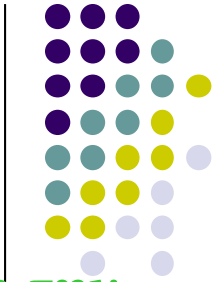


Δεύτερος Γύρος



Τρίτος Γύρος

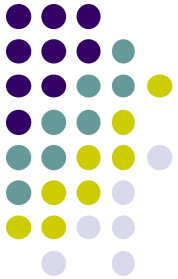
Τερματισμός Αλγορίθμου FloodMax



Συνήθως, επιθυμούμε μετά την εκλογή της διεργασίας με την μεγαλύτερη ταυτότητα, όλες οι διεργασίες στο δίκτυο να ενημερωθούν για την ταυτότητα της.

- Στον αλγόριθμο FloodMax αυτό επιτυγχάνεται στο τέλος του γύρου *d* όπου όλες οι διεργασίες θα έχουν αποθηκευμένη την ίδια max_ID. Θα γνωρίζουν δηλαδή τη διεργασία αρχηγό. Επίσης, για μία και μόνο διεργασία θα ισχύει ότι η max_ID θα ισούται με την ταυτότητα της διεργασίας.
- Στο τέλος, δηλαδή, της εκτέλεσης, θα υπάρχει μία διεργασία αρχηγός και όλες οι υπόλοιπες θα γνωρίζουν ποια είναι αυτή.

Απόδειξη ορθότητας του Αλγορίθμου FloodMax



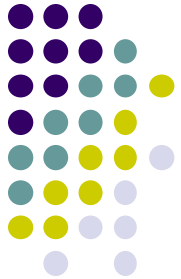
Έστω ID_k η ταυτότητα της διεργασίας k και ID_{max} η ταυτότητα της διεργασίας max με τη μεγαλύτερη ταυτότητα.

Τότε, στο τέλος του γύρου d η διεργασία max εκλέγεται αρχηγός και καμία άλλη διεργασία εκτός από την max δεν είναι σε κατάσταση *ισχύος*.

- Η χρονική πολυπλοκότητα είναι $O(d)$ - *είναι ο χρόνος που χρειάζεται η ταυτότητα της διεργασίας με τη μέγιστη ταυτότητα να φτάσει και στη πιο απομακρυσμένη διεργασία.*

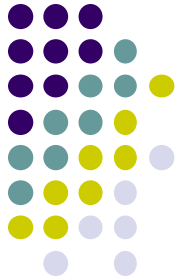
- Η πολυπλοκότητα επικοινωνίας είναι $O(dm)$ - *Τα μηνύματα που ανταλλάσσονται εξαρτώνται από τον αριθμό m των καναλιών επικοινωνίας και τον αριθμό d των γύρων εκτέλεσης*

Απόδειξη ορθότητας του Αλγορίθμου FloodMax



- **Λήμμα:** Η διεργασία με ταυτότητα ID_{max} τίθεται σε κατάσταση ισχύος θέτοντας *leader=true* στο τέλος του γύρου *d*.

- **Απόδειξη**:
- Η διαδικασία με ταυτότητα ID_{max} έχει αρχικά θέσει στη μεταβλητή max_ID την ταυτότητά της που ισούται με ID_{max} (έναρξη του αλγορίθμου).
- Μετά απο *d* γύρους η διεργασία δεν έχει κάνει καμία αλλαγή στην max_ID καθώς για κάθε άλλη διεργασία ισχύει $ID_k < ID_{max}$. Έτσι μετά και από *d* γύρους για τη διεργασία με ταυτότητα ID_{max} θα ισχύει max_ID = ID_{max}. Επομένως, ισχύουν οι συνθήκες για να τεθεί σε κατάσταση ισχύος θέτοντας *leader=true*.

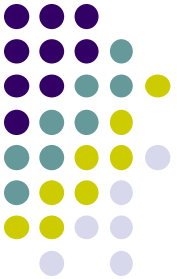


Το πρόβλημα του αμοιβαίου αποκλεισμού

- Πρόσβαση **πολλών** διεργασιών σε **κοινό** πόρο (π.χ., κοινή μεταβλητή, κοινή δομή δεδομένων, κοινό αρχείο)
- Οι διεργασίες πρέπει να αποκτήσουν **αποκλειστική πρόσβαση**

Αλγόριθμοι Αμοιβαίου Αποκλεισμού

Ο αλγόριθμος του Lamport (LamportME)



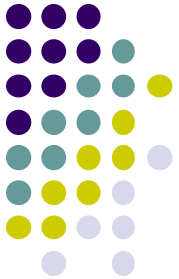
Lamport: διατύπωσε το 1978 τον πρώτο κατανεμημένο αλγόριθμο για την λύση του προβλήματος του αμοιβαίου αποκλεισμού.

Αλγόριθμος LamportME

- **Κάθε διεργασία κρατάει ένα λογικό ρολόι και μία ουρά για εισερχόμενες αιτήσεις.**
- **Κάθε διεργασία που επιθυμεί να μπει στην ΚΠ στέλνει ένα μήνυμα request σε όλες τις άλλες διεργασίες. Το μήνυμα περιέχει και την χρονοσφραγίδα του λογικού ρολογιού της διεργασίας. Η αίτησή της θα μπει και στην δική της ουρά.**
- **Όταν μία διεργασία παραλάβει μία αίτηση, την τοποθετεί στην ουρά και απαντάει στην αιτούσα διεργασία ότι η παραλαβή έγινε επιτυχώς.**
- **Αν η ουρά δεν είναι κενή και ο πόρος είναι ελεύθερος οι διεργασίες αποφασίζουν να μπει στην ΚΠ η διεργασία με την μικρότερη χρονοσφραγίδα, δηλαδή αυτή που έκανε πρώτα την αίτηση σύμφωνα με την σχέση "συνέβη πριν".**

  (Όλες οι διεργασίες έχουν ίδιες ουρές άρα όλες αποφασίζουν το ίδιο – στην πραγματικότητα η διεργασία που θα δει ότι στην ουρά της η αίτηση με την μικρότερη χρονοσφραγίδα είναι δικιά της θα μπει στην ΚΠ).
- **Για την απελευθέρωση του πόρου η διεργασία που τον έχει, στέλνει μήνυμα release σε όλες τις υπόλοιπες και διαγράφει την αίτησή της από την ουρά της. Όταν και οι άλλες διεργασίες παραλάβουν το μήνυμα θα διαγράψουν και αυτές την αίτηση της διεργασίας από την ουρά τους και θα συνεχιστεί η διαδικασία.**

Ο αλγόριθμος του Lamport (LamportME)



Αν η Pi θέλει να εισέλθει στην ΚΠ

- **εισάγει την αίτησή της στην ουρά της**
- **στέλνει <request, i, TS> σε όλες τις υπόλοιπες διεργασίες**

Αν η Pi λάβει <request, i, TS> από την Pj

- **βάζει την Pj στην ουρά προτεραιότητας με βάση το TS**
- **στέλνει <reply> στην Pj**

Αν η Pi θέλει να εξέλθει από την ΚΠ

- **διαγράφει την αίτησή της από την ουρά της**
- **στέλνει <release> σε όλες τις υπόλοιπες διεργασίες**

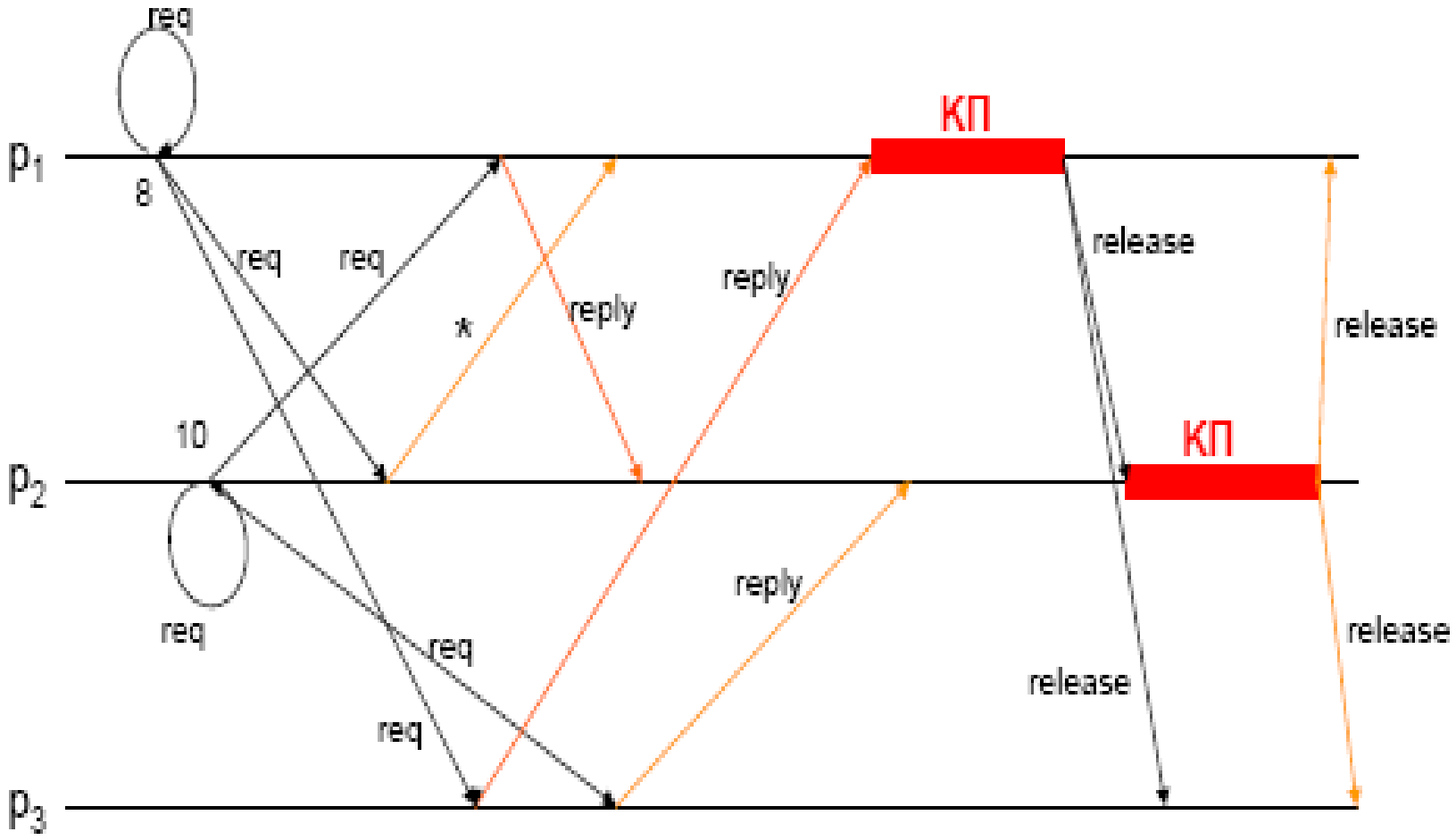
Αν η Pi λάβει <release> από την Pj

- **διαγράφει την Pj από την ουρά της**

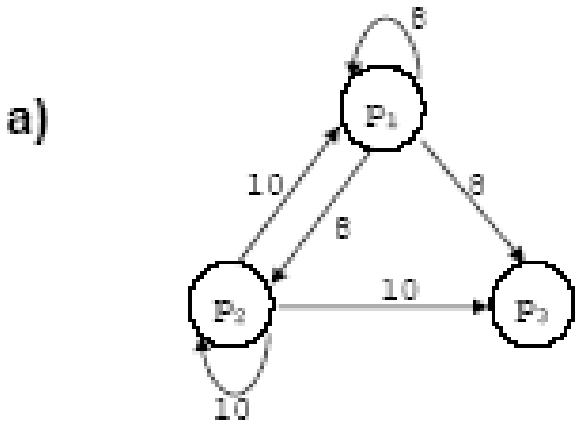
Η Pi εισέρχεται στην ΚΠ αν και μόνο αν:

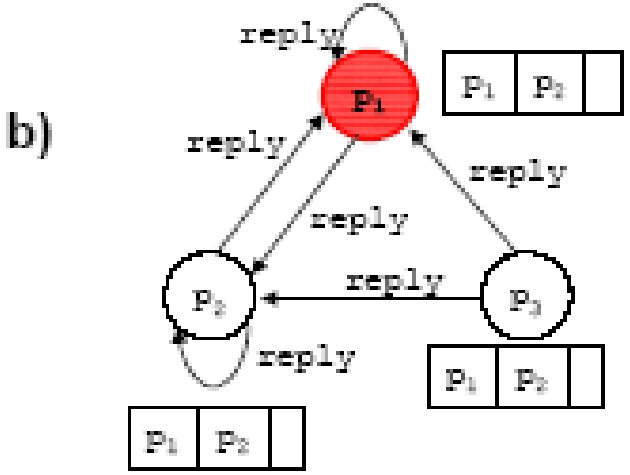
1. **έχει λάβει <reply> από όλες τις υπόλοιπες διεργασίες**
2. **η αίτησή της είναι στην κορυφή της ουράς της**

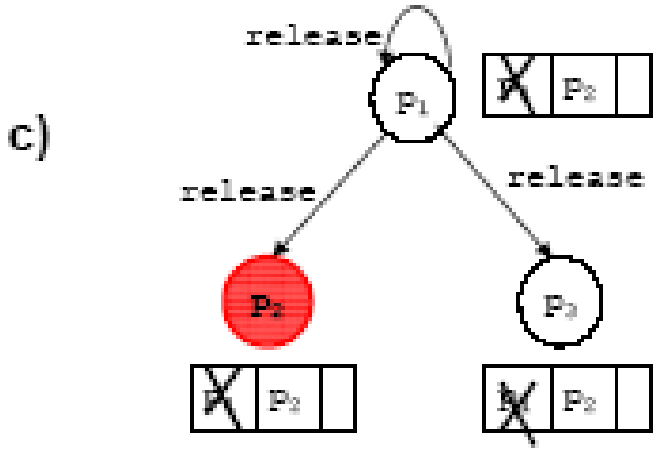
Ο αλγόριθμος του Lamport (LamportME) Παράδειγμα Εκτέλεσης

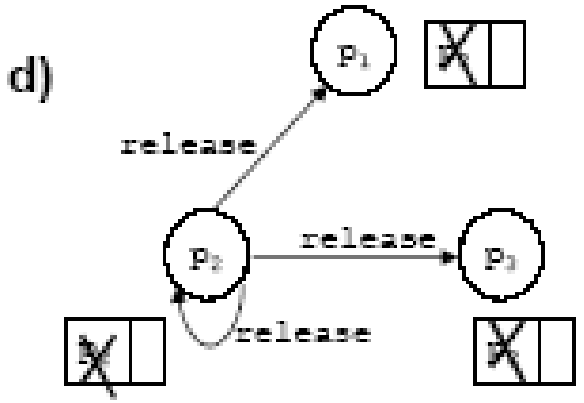


Ο αλγόριθμος του Lamport (LamportME)
Παράδειγμα Εκτέλεσης

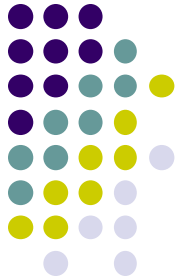








Ορθότητα αλγορίθμου LamportME



Ασφάλεια: **Έστω ότι κατά την εκτέλεση του αλγόριθμου υπάρχουν 2 διεργασίες η P1 και η P2 στην ΚΠ.**

Έστω ότι η P1 έστειλε το μήνυμα για request με TS μικρότερο από ότι η P2.

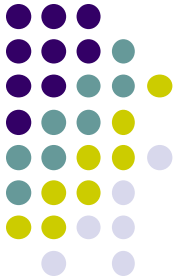
Σύμφωνα με τον αλγόριθμο, η P1 έχει δει την αίτηση της P2 και την έχει κρατήσει στην ουρά της και όμοια η P2 έχει δει την αίτηση της P1 και την έχει κρατήσει στην ουρά της.

Ωστόσο, αφού το TS της P1 είναι μικρότερο από ότι της P2, η P2 για να μπει στην ΚΠ θα περιμένει οπωσδήποτε το μήνυμα που θα ενημερώνει για την απελευθέρωση του πόρου από την P1.

Αυτό σημαίνει ότι η P1 θα έχει βγει από την ΚΠ πριν μπει η P2. Άρα δεν θα είναι και οι δύο μαζί στην ΚΠ πράγμα άτοπο.

Συμφωνία σε κατανεμημένα συστήματα με σφάλματα

Πρόβλημα των Βυζαντινών στρατηγών οι οποίοι πρέπει να συντονίσουν την επίθεση τους έναντι ενός κοινού εχθρού. Κάποιοι από τους στρατηγούς μπορεί να είναι προδότες.



Κατανεμημένη δέσμευση: αφορά την εκτέλεση μιας λειτουργίας είτε από όλες τις εργασίες μιας ομάδας είτε από καμία.

Μία απλή μέθοδος στην κατανεμημένη δέσμευση: η χρήση συντονιστή ο οποίος ειδοποιεί όλες τις διεργασίες να εκτελέσουν (τοπικά) ή όχι την εν λόγω λειτουργία

- **Πρωτόκολλο Δέσμευσης Δύο Φάσεων (2PC)**
- **Πρωτόκολλο Δέσμευσης Τριών Φάσεων (3PC)**