Διάλεξη 12: Διάχυση Μηνυμάτων

ΕΠΛ 432: Κατανεμημένοι Αλγόριθμοι

Τι θα δούμε σήμερα

- Ορισμός Προσομοίωσης
- Προσομοίωση Υπηρεσίας Διάχυσης Μηνυμάτων
- Ιδιότητες Διάταξης Μηνυμάτων

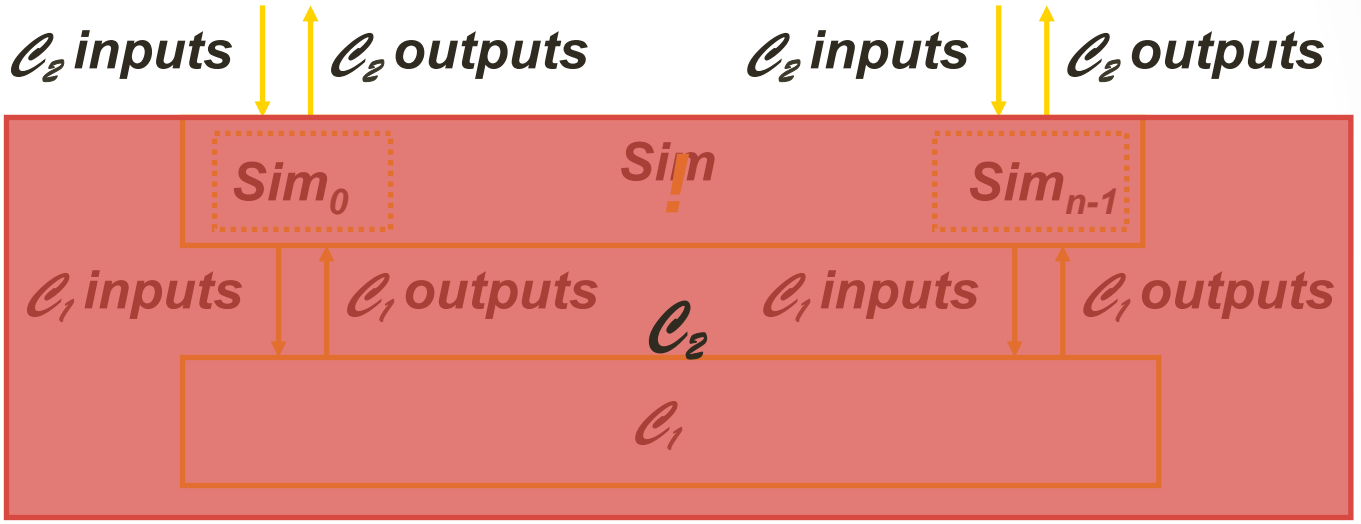
Είδη Επικοινωνιακών Συστημάτων

- Ανταλλαγή Μηνυμάτων vs Κοινόχρηστη Μνήμη
  - Διαφορά στην διεπαφή
    - αποστολή/παραλαβή vs πρόσβαση/απάντηση
- Στην ανταλλαγή μηνυμάτων μας ενδιαφέρει
  - Αξιοπιστία παράδοσης μηνυμάτων
  - Διάταξη μηνυμάτων
  - Εγγυήσεις στην διατήρηση του περιεχομένου του μηνύματος
- Στην κοινόχρηστη μνήμη μας ενδιαφέρει
  - Σημασιολογία των κοινόχρηστων μεταβλητών
    - Ανάγνωση/εγγραφή, ανάγνωση-μετατροπή-εγγραφή, κτλ.

Προσομοίωση

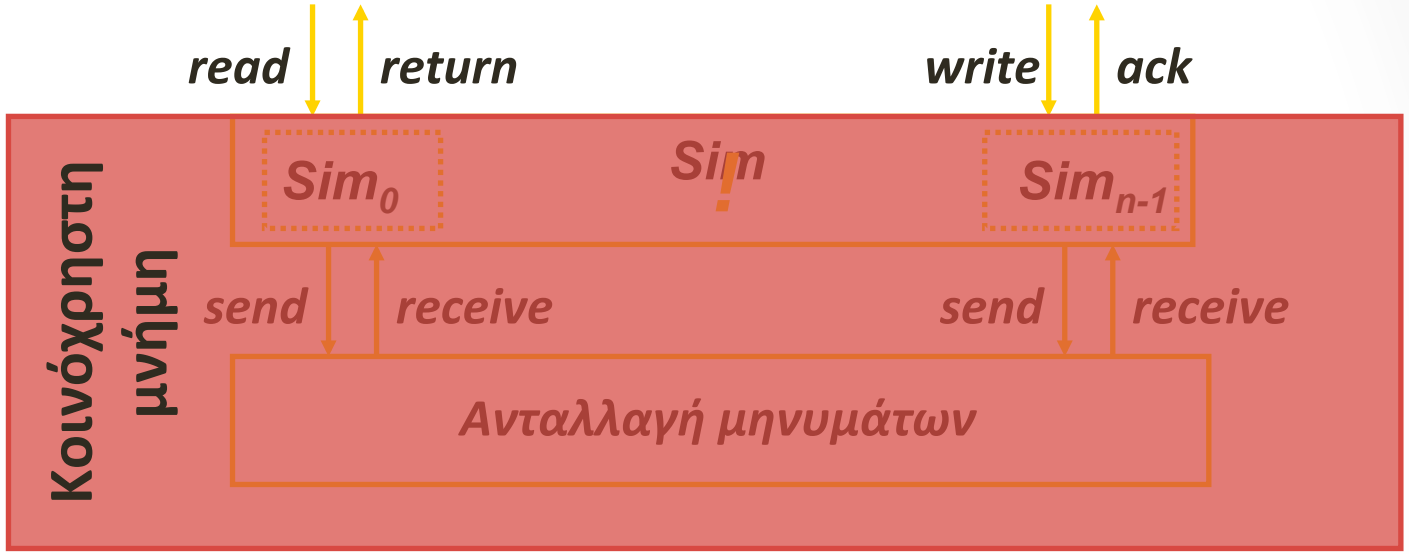
- **Ορισμός**: Κάποιο επικοινωνιακό σύστημα C1 **προσομοιώνει** ένα άλλο επικοινωνιακό σύστημα C2 αν κάθε κόμβος του συστήματος τρέχει μια *διεργασία προσομοίωσης*, Sim τ.ω.
  - Η άνω διεπαφή του Sim είναι η ίδια με την διεπαφή του C2
  - Η κάτω διεπαφή του Sim είναι η ίδια με την διεπαφή του C1
  - Για κάθε νόμιμη εκτέλεση α του Sim, υπάρχει μια ακολουθία γεγονότων του C2 που αντιστοιχούν στα γεγονότα του α στην άνω διεπαφή
    - Με άλλα λόγια η προσομοίωση παρουσιάζει την ίδια εξωτερική διεπαφή με το C2. Άρα η προσομοίωση είναι διαφανής στον χρήστη ο οποίος νομίζει ότι αλληλεπιδρά με το σύστημα C2.

Προσομοίωση



"# $% &'&$(µ) *)+ (, - .&($%/ 01#)+ &2&$., $3$0 (
4 -%&%µ%12&(Sim 055/6$)+ 3$+ % , - .&$(7 $%/ $\mathcal{C}_2$ #%µ180+ 3$+
, - (&+µ%9%+01 $% $\mathcal{C}_2$(*)+ 3,+ $% $\mathcal{C}_1$µ0 $% 09190:% 9 -%&%µ%12&(7)

Παράδειγμα



Προσομοιώνει

Διεργασίες

- Ένα επικοινωνιακό σύστημα προσομοιώνεται από μια **διεργασία** σε κάθε επεξεργαστή

- Για μια προσομοίωση μπορεί να τρέχουν πολλοί αλγόριθμοι (διεργασίες) σε ένα επεξεργαστή. Για παράδειγμα:
  - Μια διεργασία (αλγόριθμος) χρησιμοποιεί την υπηρεσία C2
  - Μια άλλη διεργασία (αλγόριθμος) υλοποιεί την προσομοίωση πάνω από την υπηρεσία C1

- Άρα δεν θα μιλούμε για επεξεργαστές που υλοποιούν ένα αλγόριθμο αλλά για διεργασίες

Ασύγχρονη Ανταλλαγή Μηνυμάτων Σημείου-Προς-Σημείο

Διεπαφή Επικοινωνιακού Συστήματος:

- inputs: $send_i(M)$
  - Μοντελοποιεί την αποστολή συνόλου μηνυμάτων M από τον p_i
  - Κάθε μήνυμα προσδιορίζει τον αποστολέα και τον παραλήπτη

- outputs: $recv_i(M)$
  - Μοντελοποιεί την παραλαβή συνόλου μηνυμάτων M από τον p_i
  - Κάθε μήνυμα στο *M* πρέπει να έχει το p_i ως τον παραλήπτη

Νόμιμη Ακολουθία Γεγονότων

- Μια ακολουθία send, recv γεγονότων είναι **νόμιμη** εαν υπάρχει μια αντιστοιχία κ από τα μηνύματα που στάλθηκαν στα μηνύματα που παραλήφθηκαν τ.ω.:
  - κ είναι καλά-ορισμένο: κάθε μήνυμα που παραλαμβάνεται είχε προηγουμένως σταλεί
    - Δεν δημιουργούνται μηνύματα τυχαία και ούτε χάνονται μηνύματα
  - κ είναι ένα-προς-ένα: δεν έχουμε πολλαπλασιασμό μηνυμάτων
  - κ είναι επί-τα-αυτά: κάθε μήνυμα που αποστέλλεται τελικά παραλαμβάνεται

Προδιαγραφές Διάχυσης Μηνυμάτων

- inputs: bc_send$_i$(*m*)
  - διάχυση του μηνύματος m από τον p_i προς όλους τους επεξεργαστές

- outputs: bc_recv$_i$(*m, j*)
  - Η υπηρεσία διάχυσης παραδίδει το μήνυμα m, που στάλθηκε από τον p_j, στον p_i

Νόμιμη Ακολουθία Γεγονότων

- Μια ακολουθία bc_send και bc_recv γεγονότων είναι **νόμιμη** εαν και μόνο εαν υπάρχει μια αντιστοιχία κ από τα μηνύματα που στάλθηκαν στα μηνύματα που παραλήφθηκαν τ.ω.:
  - κ είναι καλά-ορισμένο: κάθε μήνυμα που παραλαμβάνεται είχε προηγουμένως σταλεί (**Ακεραιότητα** μηνυμάτων)
  - κ περιορισμένο στα bc_rcv_i γεγονότα είναι ένα-προς-ένα: κανένα μήνυμα δεν παραλαμβάνεται περισσότερες από μια φορές σε κάθε επεξεργαστή. (**Όχι Πολλαπλασιασμό** μηνυμάτων)
  - κ περιορισμένο στα bc_rcv_i γεγονότα είναι επί-τα-αυτά: κάθε μήνυμα που αποστέλλεται τελικά παραλαμβάνεται σε κάθε επεξεργαστή. (**Ζωτικότητα** μηνυμάτων)

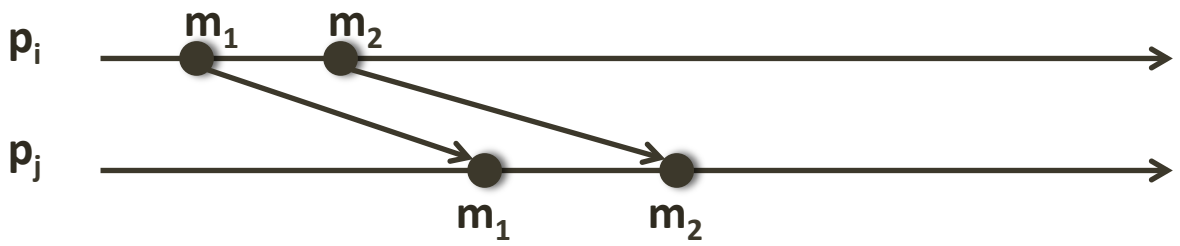
Διάταξη Μηνυμάτων

- Μερικές φορές θα θέλαμε η υπηρεσία διάχυσης να μασ παρέχει κάποιες εγγυήσεις για την διάταξη με την οποία τα μηνύματα θα παραληφθούν

- Πρέπει να προσθέσουμε επιπρόσθετουσ περιορισμούς στην αντιστοιχία κ για να επιτύχουμε:
  - FIFO μιας-πηγής (single-source FIFO), ή
  - ολική διάταξη (totally-ordered), ή
  - αιτιοκρατική διάταξη (causally ordered)

FIFO Μιας-Πηγής

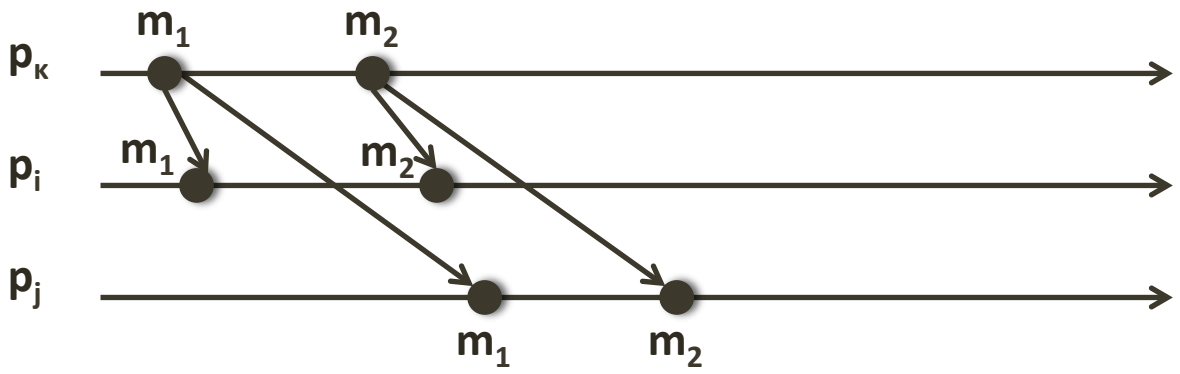
- Για όλα τα μηνύματα m_1 και m_2 και όλα τα p_i και p_j
  - Αν ο p_i στείλει m_1 πριν στείλει το m_2, και
  - Αν ο p_j λάβει τα m_1 και m_2, τότε ο p_j λαμβάνει το m_1 πριν λάβει το m_2



- **Προσοχή**: Δεν εγγυάται ότι και τα δυο μηνύματα θα ληφθούν
  - Αυτό πρέπει να το εγγυηθεί η ιδιότητα ζωτικότητας

Ολική Διάταξη

- Για όλα τα μηνύματα m_1 και m_2 και όλα τα p_i και p_j
  - Αν και οι δυο p_i και p_j λάβουν τα m_1 και m_2, τότε λαμβάνουν με την ίδια σειρά



- **Προσοχή**: Δεν εγγυάται ότι και τα δυο μηνύματα θα ληφθούν από τους δυο επεξεργαστές
  - Αυτό πρέπει να το εγγυηθεί η ιδιότητα ζωτικότητας

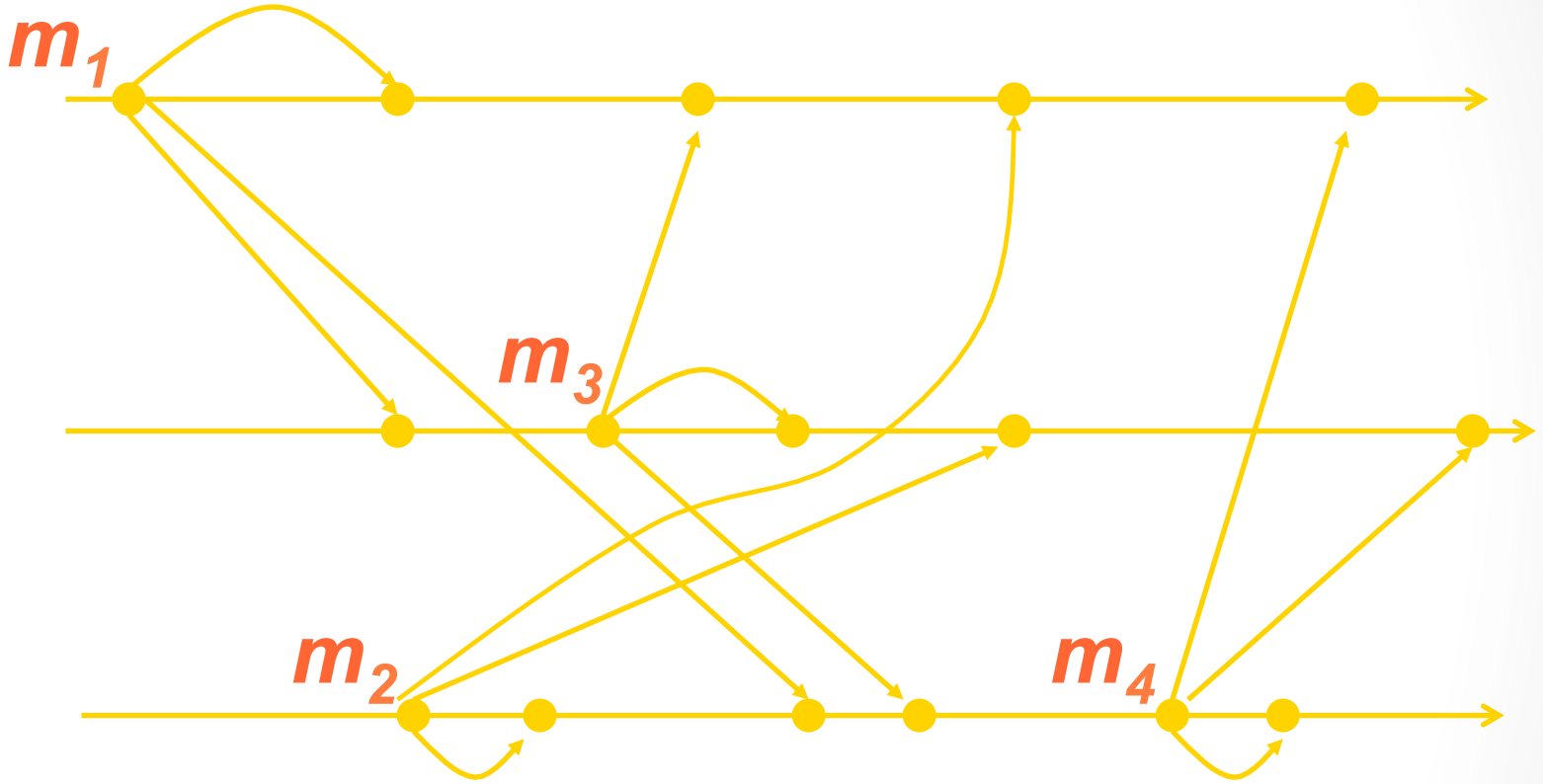
Σχέση «πριν-από» για μηνύματα

- Ας υποθέσουμε ότι όλη η επικοινωνία γίνεται με διάχυση μηνυμάτων

- Λέμε ότι ένα μήνυμα m_1 **συμβαίνει πριν-από** το μήνυμα m_2 αν:
  - Το γεγονός bc_recv(m_1, j) συμβαίνει πριν-από το γεγονός bc_send(m_2) (σύμφωνα με τον ορισμό του «πριν-από» για γεγονότα)

  - m_1 και m_2 στέλνονται από τον ίδιο επεξεργαστή και m_1 στέλνεται πριν το m_2

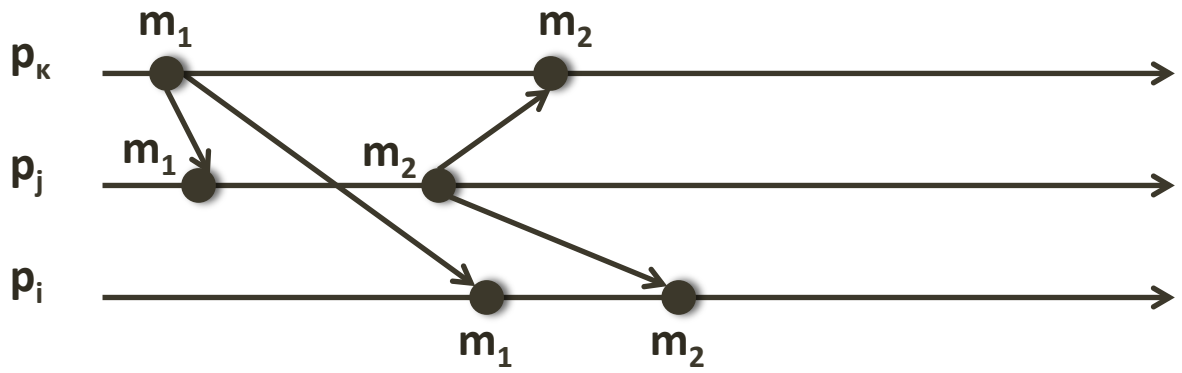
Παράδειγμα



m_1 συμβαίνει πριν-από τα m_3 και m_4
m_2 συμβαίνει πριν-από το m_4
m_3 συμβαίνει πριν-από το m_4

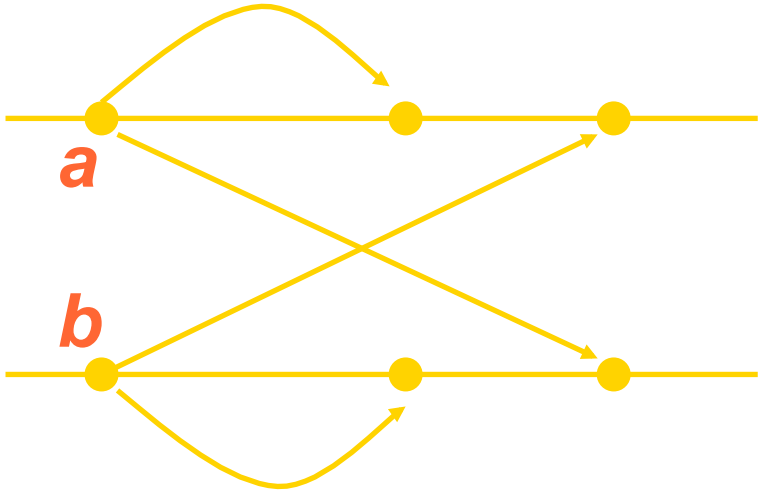
Αιτιοκρατική Διάταξη

- Για όλα τα μηνύματα m_1 και m_2 και όλα τα p_i
  - Αν το m_1 συμβαίνει πριν-από το m_2 και ο p_i λάβει και το m_1 και το m_2, τότε ο p_i λαμβάνει το m_1 πριν λάβει το m_2



- **Προσοχή**: Δεν εγγυάται ότι και τα δυο μηνύματα θα ληφθούν
  - Αυτό πρέπει να το εγγυηθεί η ιδιότητα ζωτικότητας

Παράδειγμα

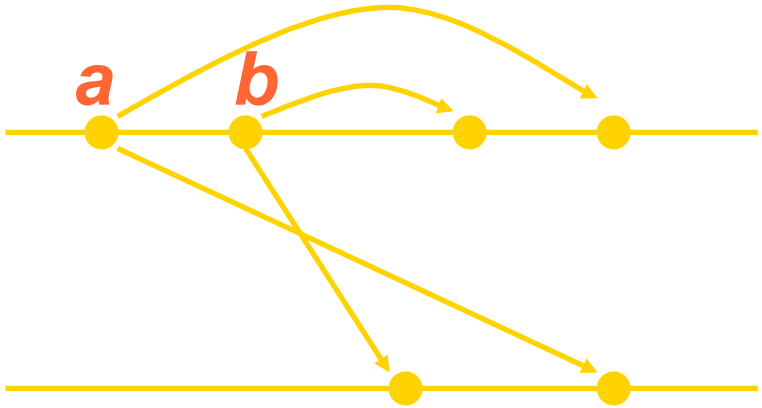


FIFO μιας-πηγής? ☑

Ολικά διατεταγμένο? ☒

Αιτιοκρατικά διατεταγμένο? ☑

Παράδειγμα

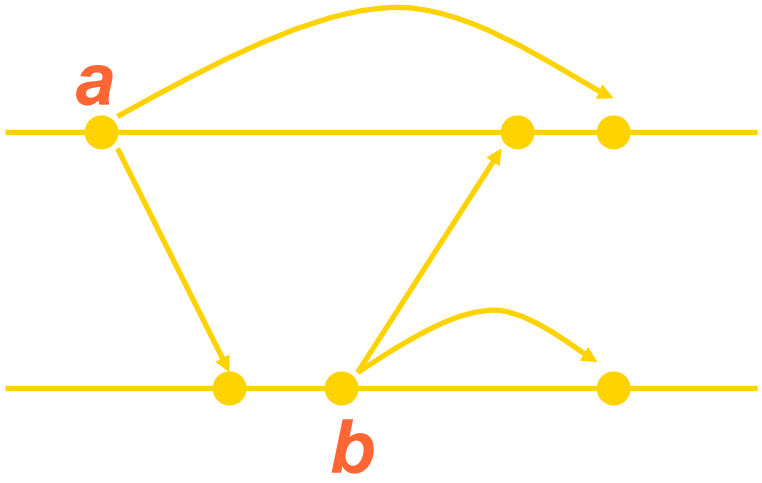


FIFO μιας-πηγής? ☒

Ολικά διατεταγμένο? ☑

Αιτιοκρατικά διατεταγμένο? ☒

Παράδειγμα



FIFO μιας-πηγής? ☑

Ολικά διατεταγμένο? ☒

Αιτιοκρατικά διατεταγμένο? ☒

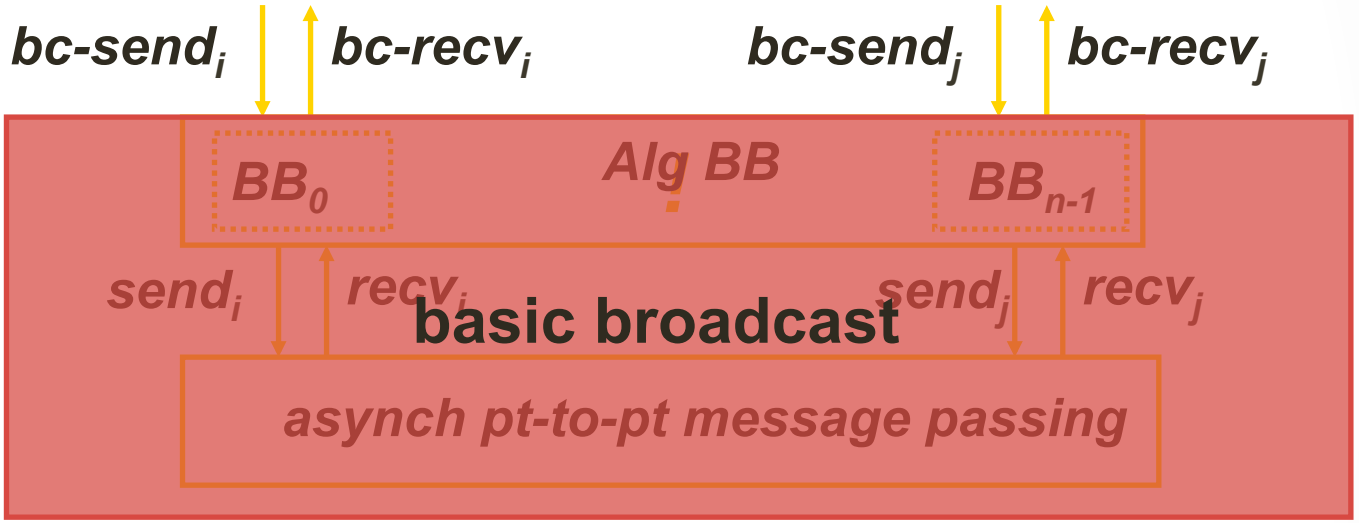
Βασικός Αλγόριθμος Διάχυσης

- Αλγόριθμος BB: Βασικός αλγόριθμος διάχυσης μηνυμάτων πάνω από επικοινωνιακό σύστημα ΣπΣ

- Όταν συμβεί το $bc\_send_i(m)$:
  - p_i στέλνει ένα ξεχωριστό αντίγραφό του m σε κάθε επεξεργαστή (συμπ. και του ιδίου) χρησιμοποιώντας το επικοινωνιακό σύστημα ΣπΣ ανταλλαγής μηνυμάτων

- Ο p_i εκτελεί το γεγονός $bc\_recv_i(m)$:
  - Όταν παραλάβει το m από το ΣπΣ επικοινωνιακό σύστημα του χαμηλότερου επιπέδου

Προσομοίωση BB



Ορθότητα ΒΒ

- Δεδομένου ότι το επικοινωνιακό σύστημα ΣπΣ ανταλλαγής μηνυμάτων είναι σωστό ο αλγόριθμος ΒΒ εγγυάται
  - Ακεραιότητα μηνυμάτων
  - Όχι πολλαπλασιασμό
  - Ζωτικότητα μηνυμάτων

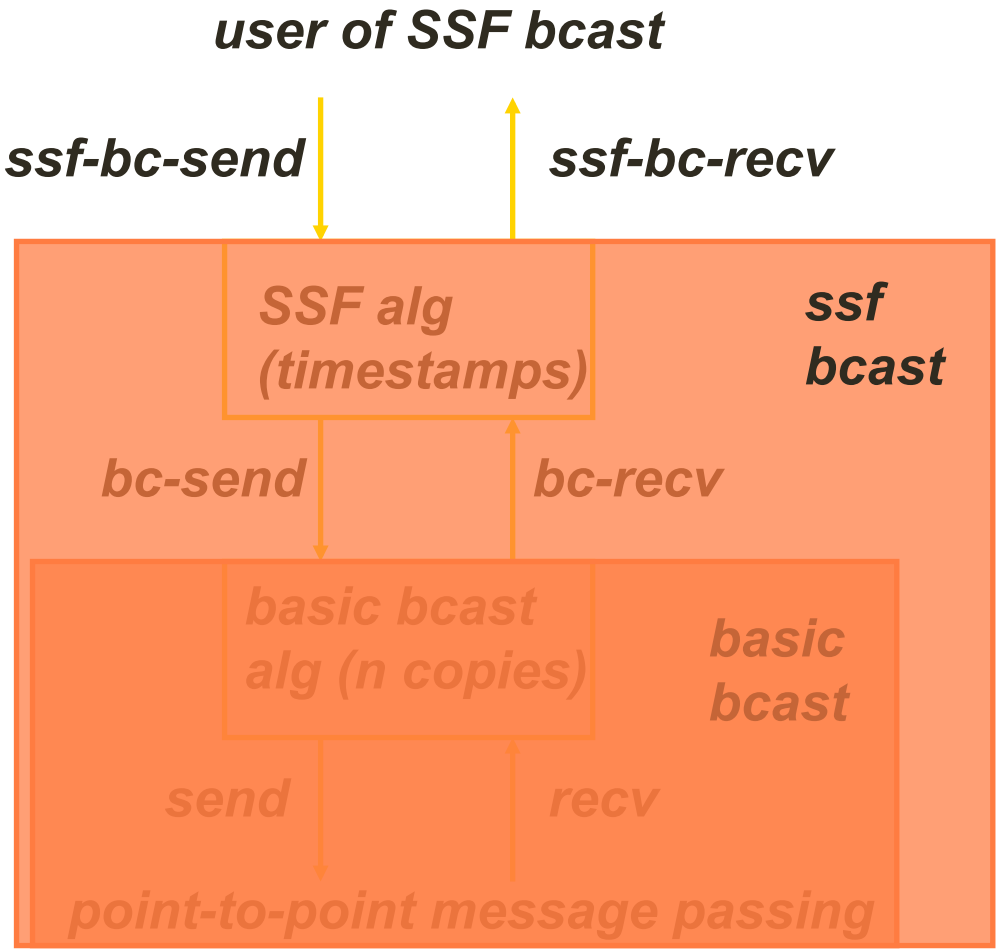
Αλγόριθμος Μιας-Πηγής FIFO

- Υποθέτουμε ότι κάτω επικοινωνιακό σύστημα υλοποιεί βασική διάχυση

- Όταν συμβεί το $ssf\_bc\_send_i(m)$:
  - p_i αυξάνει ένα τοπικό αριθμό κατα 1 και τον διανέμει μαζί με το μήνυμα m χρησιμοποιώντας την βασική διάχυση

- Ο p_i εκτελεί το γεγονός $ssf\_bc\_recv_i(m)$:
  - Όταν ο p_i παραλάβει το m και ένα αριθμό T μέσα σε ένα μήνυμα της βασικής διάχυσης από τον p_j και όλα τα άλλα μηνύματα που έλαβε από τον p_j είχαν μικρότερο αριθμό.

Αλγόριθμος Μιας-Πηγής FIFO



Συμμετρικός Αλγόριθμος Ολικής Διάταξης

- Υποθέτουμε ότι το κάτω επικοινωνιακό επίπεδο υλοποιεί διάχυση FIFO μιας-πηγής.

Κάθε επεξεργαστής

- Στέλνει μαζί με το μήνυμά του και μια χρονοσφραγίδα
  - Για αποφυγή ισοτήτων χρησιμοποιούμε τις ταυτότητες

- Κρατά ένα διάνυσμα με εκτιμήσεις των χρονοσφραγίδων των άλλων επεξεργαστών
  - Οι εκτιμήσεις ανανεώνονται βάση των μηνυμάτων που λαμβάνονται και βάση των μηνυμάτων «ενημέρωσης χρονοσφραγίδων»

- Προωθεί ένα μήνυμα στο ανώτερο επίπεδο εαν όλες οι χρονοσφραγίδες στο διάνυσμα είναι μικρότερες από τη δική του

Συμμετρικός Αλγόριθμος Ολικής Διάταξης

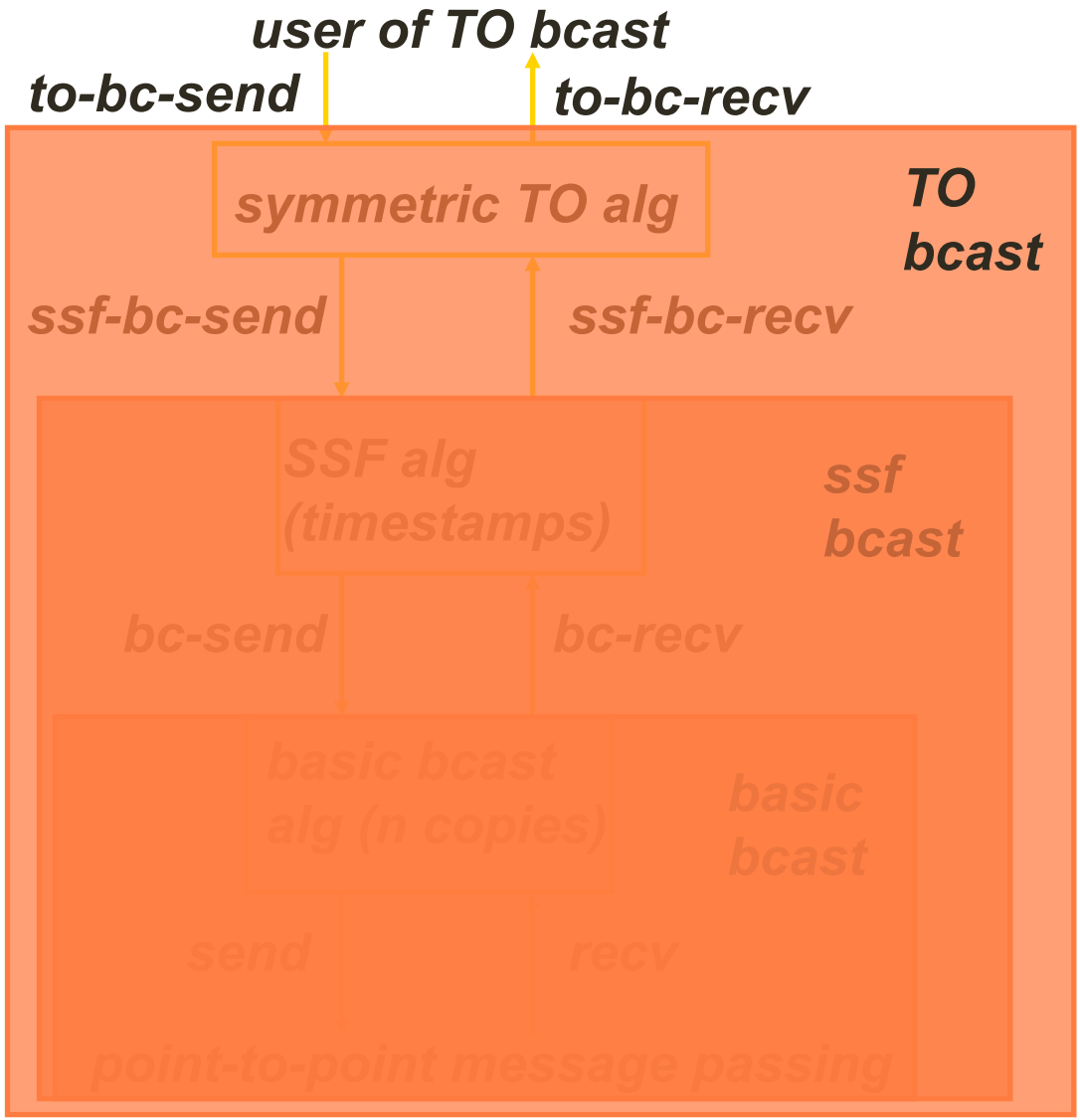
when to-bc-send$_i$(*m*) occurs:
  ts[i]++
  add (*m,ts[i],i*) to *pending*
  invoke ssf-bc-send$_i$((*m,ts[i]*))

when ssf-bc-recv$_i$((*m,T*)) from p_j occurs:
  ts[j] := T
  add *(m,T,j)* to *pending*
  if *T* > *ts[i]* then
    ts[i] := T
    invoke ssf-bc-send$_i$("ts-up",*T*)

invoke to-bc-recv$_i$(*m,j*) when:
  (*m,T,j*) is entry in *pending* with smallest (*T,j*)
  T ≤ *ts[k]* for all *k*
result: remove (*m,T,j*) from *pending*

when ssf-bc-recv$_i$("ts-up",*T*) from p_j occurs:
  ts[j] := T

Συμμετρικός Αλγόριθμος Ολικής Διάταξης



Αλγόριθμος Αιτιοκρατικής Διάταξης

- Ο αλγόριθμος βασίζεται στα διανυσματικά ρολόγια

Code for p_i :

when co-bc-send$_i$(*m*) occurs:
  vt[i]++
  invoke co-bc-recv$_i$(*m*)
  invoke bc-send$_i$((*m,vt*))

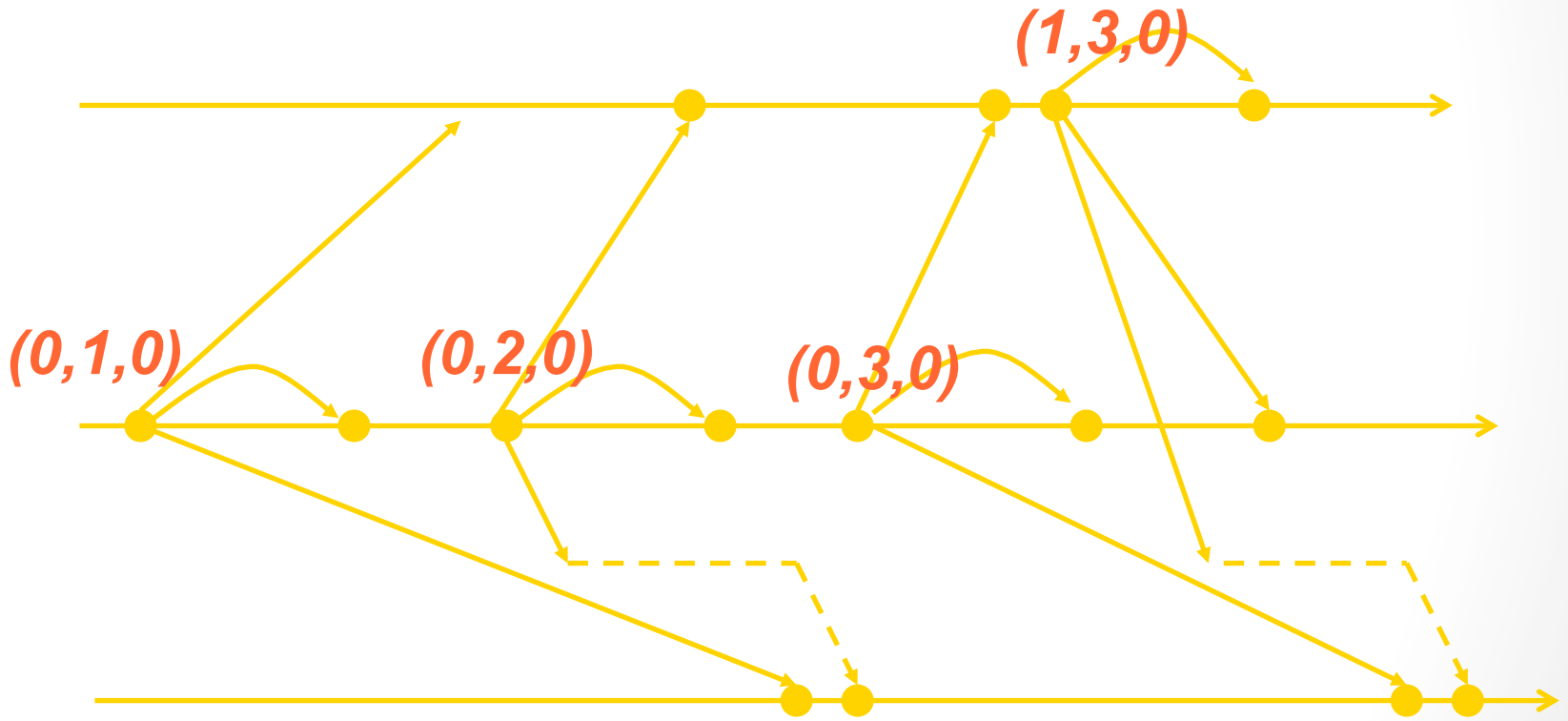
when bc-recv$_i$((*m,w*)) from p_j occurs:
  add *(m,w,j)* to *pending*

invoke co-bc-recv$_i$(*m,j*) when:
  (*m,w,j*) is in *pending*
  w[j] = *vt[j]* + *1*
  w[k] ≤ *vt[k]* for all *k* ≠ *j*
result:
  remove (*m,w,j*) from *pending*
  vt[j]++

Note: *vt[j]* καταγράφει πόσα μηνύματα του p_j έχουν co-bc-recv'ed από τον p_i

Παράδειγμα Αλγορίθμου

- Ο αλγόριθμος καθυστερεί την παράδοση των Α.Δ. Μηνυμάτων ώσπου να μπορούμε να διασφαλίσουμε την μη παραβίαση της αιτιοκρατικής διάταξης!



Ερωτήσεις;