HY486 - Αρχές Κατανεμημένου Υπολογισμού

Εαρινό Εξάμηνο 2015-2016

Δεύτερη Προγραμματιστική Εργασία

Γενική περιγραφή

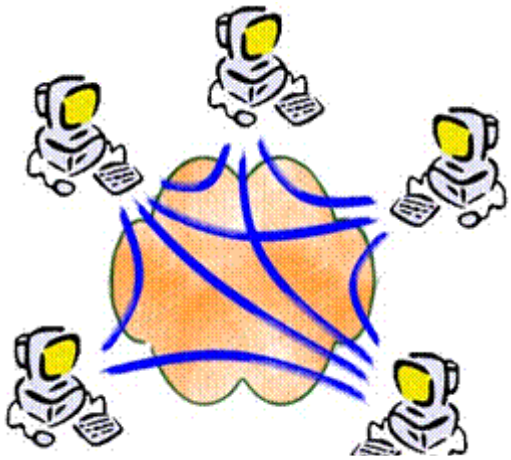
Στη δεύτερη προγραμματιστική εργασία καλείστε να υλοποιήσετε ένα διομότιμο σύστημα (Peer-to-Peer) με τη χρήση της βιβλιοθήκης Message Passing Interface (**MPI**).

Η εργασία θα πρέπει να υλοποιηθεί στη γλώσσα C. Επίσης, θα πρέπει να γίνει χρήση της έκδοσης του MPI που είναι προεγκατεστημένη στα μηχανήματα του Τμήματος.

Περιγραφή διομότιμου συστήματος

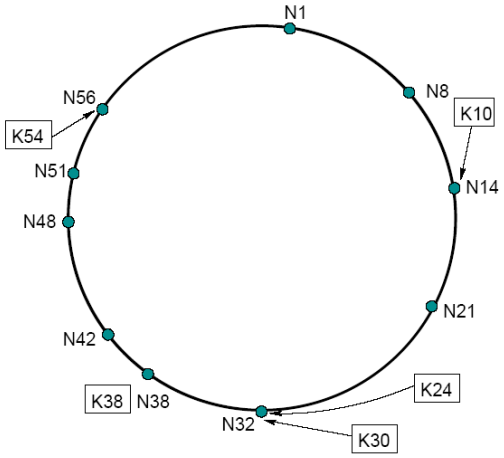
Ένα διομότιμο (Peer-to-Peer ή P2P) σύστημα είναι ένα κατανεμημένο σύστημα (Σχήμα 1) που αποτελείται από κόμβους που είναι ομότιμοι (δηλαδή κόμβους που έχουν την ίδια λειτουργικότητα).

Θεωρούμε ένα διομότιμο σύστημα στο οποίο το πλήθος των κόμβων που συμμετέχουν κάθε χρονική στιγμή είναι αυθαίρετο αλλά όχι μεγαλύτερο από 2^m-1, όπου m είναι κάποιος ακέραιος. Θεωρούμε πως κάθε κόμβος που συμμετέχει στο σύστημα έχει ένα **μοναδικό** αναγνωριστικό, το οποίο καθορίζει τη θέση του πάνω σε έναν λογικό δακτύλιο που αναπαριστά το σύνολο τιμών {0, …, 2^m-1}. Ένα διομότιμο σύστημα εξελίσσεται δυναμικά, δηλαδή υποστηρίζει την εισαγωγή και την αποχώρηση κόμβων.



Σχήμα 1: Ένα Διομότιμο Σύστημα

Στο Σχήμα 2 παρουσιάζεται ένα διομότιμο σύστημα για το οποίο ισχύει ότι m=6 (δηλαδή ο μέγιστος αριθμός κόμβων που μπορούν να συμμετέχουν στο σύστημα είναι 2^6 = 64). Την τρέχουσα χρονική στιγμή μόνο 10 κόμβοι συμμετέχουν στο σύστημα, οι οποίοι έχουν αναγνωριστικά 1, 8, 14, 21, 32, 38, 42, 48, 51 και 56. Οι κόμβοι αυτοί, στο σχήμα εμφανίζονται ως N1, N8, N14, N21, N32, N38, N42, N48, N51 και N56, αντίστοιχα.



Μία από τις σημαντικότερες εφαρμογές ενός συστήματος P2P είναι η κατανεμημένη αποθήκευση και διαχείριση αρχείων. Θεωρούμε πως κάθε αρχείο έχει (όπως και κάθε κόμβος) ένα μοναδικό αναγνωριστικό στο σύνολο {0,…,2^m-1}. Έστω N_{min} και N_{max} οι κόμβοι με το μικρότερο και το μεγαλύτερο αναγνωριστικό στο δακτύλιο αντίστοιχα,.

Ο κανόνας σύμφωνα με τον οποίο αποθηκεύονται τα αρχεία στους κόμβους του συστήματος είναι ο ακόλουθος:

Συνθήκη (1): *«(α) Κάθε κόμβος* $\neq N_{min}$ *αποθηκεύει τα αρχεία των οποίων τα αναγνωριστικά είναι μεταξύ του αναγνωριστικού του κόμβου και του αναγνωριστικού του προηγούμενου κόμβου στο δακτύλιο και (β) ο* N_{min} *αποθηκεύει κάθε αρχείο με αναγνωριστικό Κ τέτοιο ώστε* $N_{max} \leq K \leq 2^m-1$ *ή* $0 \leq K \leq N_{min}$*».* *(1)*

Γιά παράδειγμα, στο Σχήμα 2, έχουν αποθηκευτεί 7 αρχεία στο σύστημα, τα οποία έχουν αναγνωριστικά 10, 24, 30, 38, 54, 58 και 62. Το αρχείο με αναγνωριστικό 10 αποθηκεύεται στον κόμβο με αναγνωριστικό 14 (αφού το 10 είναι μεταξύ του αναγνωριστικού 8 που έχει ο προηγούμενος κόμβος του 14 στο σύστημα και του 14), τα αρχεία με αναγνωριστικά 24 και 30 αποθηκεύονται στον κόμβο με αναγνωριστικό 32, το αρχείο με αναγνωριστικό 38 αποθηκεύεται στον κόμβο με αναγνωριστικό 38, εκείνο με αναγνωριστικό 54 αποθηκεύεται στον κόμβο με αναγνωριστικό 56, ενώ τα αρχεία με αναγνωριστικά 58 και 62 αποθηκεύονται στον κόμβο με αναγνωριστικό 1.

Αν στο σύστημα εισαχθεί κόμβος με αναγνωριστικό 26, ο κόμβος 32 θα συνεχίσει να αποθηκεύει μόνο το αρχείο με αναγνωριστικό 30 ενώ εκείνο με αναγνωριστικό 24 θα αποθηκευθεί στο νέο κόμβο (δηλαδή σε εκείνο με αναγνωριστικό 26) ώστε να ισχύει η ιδιότητα (1). Αντίστοιχα με τον αριθμό των κόμβων, έτσι και τα αρχεία τα θα πρέπει να προστίθονται και να αφαιρούνται κατά βούληση. Επίσης θα πρέπει να υποστηρίζεται και η λειτουργία της αναζήτησης ενός αρχείου εντός του διομότιμου συστήματος.

Ζητείται να υλοποιηθεί ένα πρόγραμμα που προσομοιώνει τη λειτουργία ενός τέτοιου διομότιμου συστήματος.

Message Passing Interface

Για την υλοποίηση της εργασίας θα πρέπει να χρησιμοποιήσετε τη βιβλιοθήκη Message Passing Interface (MPI). Η βιβλιοθήκη αυτή παρέχει τη δυνατότητα ανταλλαγής μηνυμάτων μεταξύ διεργασιών (που μπορεί να εκτελούνται στο ίδιο ή σε διαφορετικά μηχανήματα) μέσω ενός καθιερωμένου API, ανεξαρτήτου γλώσσας και υλοποίησης. Το API αυτό παρέχει τη δυνατότητα ανταλλαγής μηνυμάτων μεταξύ των διεργασιών όχι μόνο με τη χρήση μηνυμάτων σημείο-προς-σημείο (point-to-point messaging), αλλά και με συλλογικό τρόπο, π.χ. υποστηρίζει επικοινωνία μεταξύ ομάδων διεργασιών (multicast), καθώς και καθολική επικοινωνία (broadcasts). Σε κάθε μηχάνημα μπορούν να εκτελούνται μία ή περισσότερες MPI διεργασίες**. Για την προσομοίωση που ζητείται, κάθε κόμβος του διομότιμου συστήματος θα προσομεινώνεται από μια διεργασία MPI.** Είναι αξιοσημείωτο ότι περισσότερες από μια διεργασίες MPI μπορεί να τρέχουν στο ίδιο μηχάνημα (παρότι προσωμοιώνουν διαφορετικούς κόμβους του διομότιμου συστήματος).

Το σύστημα MPI είναι εγκατεστημένο στα μηχανήματα του Τμήματος. Οι δύο βασικές εντολές που απαιτούνται για τη λειτουργία του είναι οι **mpicc** και **mpirun**. Η εντολή mpicc χρησιμοποιείται για τη μεταγλώττιση προγραμμάτων που χρησιμοποιούν το API του MPI. Με την εντολή mpirun μπορείτε να τρέξετε το εκτελέσιμο ταυτόχρονα σε περισσότερα του ενός μηχανήματα του Τμήματος τα οποία έχουν εγκατεστημένο το MPI. Ένα παράδειγμα χρήσης παρατίθεται στη συνέχεια:

```
$ mpirun -np <count> --hostfile <file containing hostnames> <executable>
```

Η επιλογή **np** καθορίζει το συνολικό αριθμό των MPI διεργασιών που θα εκτελούν το εκτελέσιμο αρχείο που δίνεται σαν τελευταίο όρισμα στην παραπάνω γραμμή εντολών. Στην επιλογή **hostfile** δίνεται το όνομα του αρχείου στο οποίο περιέχονται τα hostnames των μηχανημάτων όπου θα εκτελεστούν διεργασίες MPI.

Κάθε διεργασία που εκκινείται με τη χρήση της εντολής mpirun έχει ένα ξεχωριστό αναγνωριστικό στο σύστημα MPI, το οποίο ονομάζεται βαθμός (rank) MPI. Επίσης, μπορεί να μάθει πόσες άλλες διεργασίες MPI τρέχουν στο σύστημα και να τους στείλει μηνύματα, χρησιμοποιώντας το rank τους σαν αναγνωριστικό. Είναι αξιοσημείωτο πως μία διεργασία μπορεί να στείλει μηνύματα μόνο σε διεργασίες που έχουν ξεκινήσει από την ίδια εντολή mpirun και είναι κατά συνέπεια μέλη του ίδιου "κόσμου" MPI.

Για παραπάνω πληροφορίες μπορείτε να διαβάσετε το υλικό που βρίσκεται στους παρακάτω συνδέσμους:

- https://computing.llnl.gov/tutorials/mpi/
- http://mpitutorial.com/tutorials/

Οργάνωση διεργασιών

Το διομότιμο σύστημα που περιγράφεται παραπάνω μπορεί να προσομοιωθεί με διάφορους τρόπους, αλλά για λόγους απλότητας η προσομοίωση θα γίνει μέσω μιας διεργασίας **συντονιστή**. Η επιλογή της διεργασίας συντονιστή μπορεί να γίνει απλά, π.χ. αναθέτοντας στη διεργασία η οποία θα λάβει το αναγνωριστικό (rank) 0, να εκτελέσει τις απαραίτητες εργασίες συντονισμού (δηλαδή να είναι ο συντονιστής).

Η διεργασία συντονιστής θα πρέπει να κρατάει πληροφορίες για το ποιοι κόμβοι συμμετέχουν στο διομότιμο σύστημα κάθε χρονική στιγμή (δηλαδή ποιοι κόμβοι έχουν προσέλθει στο σύστημα και δεν έχουν ακόμη αποχωρήσει από αυτό) και για το ποια διεργασία αντιστοιχεί σε κάθε έναν από αυτούς τους κόμβους. Η πληροφορία αυτή θα πρέπει να αποθηκεύεται σε έναν πίνακα κατακερματισμού (hash table). Το κλειδί που θα αποτελεί το όρισμα της συνάρτησης κατακερματισμού θα είναι το αναγνωριστικό του κόμβου στο διομότιμο σύστημα. Το αναγνωριστικό αυτό παρέχεται ως όρισμα στους διάφορους τύπους γεγονότων που περιγράφονται στην ενότητα Τύποι Γεγονότων.

Είναι αξιοσημείωτο ότι το συνολικό πλήθος διεργασιών που είναι ενεργές στο MPI σύστημα ανά πάσα στιγμή δεν αλλάζει και είναι ίσο με <np> (ανεξάρτητα από το πόσοι κόμβοι είναι ενεργοί στο διομότιμο σύστημα την εκάστοτε χρονική στιγμή). Οι διεργασίες που αντιστοιχούν σε κόμβους που έχουν αποχωρήσει από το διομότιμο σύστημα παραμένουν ενεργές αλλά έχουν περιορισμένη λειτουργικότητα (συγκεκριμένα, δεν πραγματοποιούν καμία ενέργεια αν δεν λάβουν εκ νέου κάποιο μήνυμα από το συντονιστή που να σηματοδοτεί την επανείσοδό τους στο διομότιμο σύστημα).

Προσέξτε πως κάθε διεργασία μπορεί να έχει **δύο** αναγνωριστικά. Το πρώτο αναγνωριστικό καθορίζεται από το rank της διεργασίας στο σύστημα MPI και χρησιμεύει στην ανταλλαγή μηνυμάτων μεταξύ των διεργασιών. Κάθε μια από τις <np> ενεργές διεργασίες στο σύστημα έχει ένα μοναδικό τέτοιο αναγνωριστικό (ανεξάρτητα από το αν η διεργασία αυτή αντιστοιχεί σε κάποιο ενεργό κόμβο του διομότιμου συστήματος ή όχι). Η διεργασία μπορεί να έχει και δεύτερο αναγνωριστικό σε περίπτωση που αντιστοιχεί σε κάποιο ενεργό κόμβο του διομότιμου συστήματος. Σε αυτή την περίπτωση, το δεύτερο αναγνωριστικό αυτό είναι το αναγνωριστικό του κόμβου στο διομότιμο σύστημα και καθορίζει τη θέση του κόμβου στο δακτύλιο που προσομοιώνεται.

Τα μηνύματα μπορούν να είναι δύο τύπων: **μηνύματα γεγονότων** τα οποία αποστέλλονται μόνο από τη διεργασία συντονιστή, και **μηνύματα επικοινωνίας** τα οποία μπορούν να έρθουν από οποιαδήποτε άλλη διεργασία εντός του διομότιμου συστήματος.

Η διεργασία συντονιστής διαβάζει γεγονότα από ένα αρχείο εισόδου. Ανάλογα με τον τύπο του γεγονότος στέλνει ένα μήνυμα γεγονότος στον κατάλληλο κόμβο του διομότιμου συστήματος (δηλαδή στη διεργασία που αντιστοιχεί στον κόμβο αυτό), ο οποίος θα εκκινεί την προσομοίωση του γεγονότος. Η προσομοίωση ενός γεγονότος μπορεί να απιτεί την αποστολή διαφόρων μηνυμάτων επικοινωνίας. Η κατάλληλη διεργασία παραλήπτης για κάθε μήνυμα γεγονότος εξαρτάται απο τον τύπο του γεγονότος και από το ποιοι κόμβοι βρίσκονται στο σύστημα τη χρονική στιγμή της προσομοίωσης. Αφού στείλει ένα μήνυμα γεγονότος και λάβει acknowledgement για την ολοκληρωσή του, η διεργασία συντονιστής διαβάζει και επεξεργάζεται το επόμενο γεγονός από το αρχείο εισόδου. Επομένως, η επεξεργασία των γεγονότων γίνεται σειριακά. Για κάθε ένα γεγονός που προσομοιώνεται επιτυχώς, η διεργασία συντονιστής θα πρέπει να ενημερώνεται από τον κόμβο που εκκίνησε την προσομοίωση του γεγονότος αυτού. Όταν όλα τα γεγονότα του αρχείου εισόδου έχουν προσομοιωθεί, η διεργασία συντονιστής θα πρέπει να ειδοποιεί τις υπόλοιπες διεργασίες πως πρέπει να τερματίσουν την εκτέλεση τους και μετά να τερματίζει και η ίδια.

Οι υπόλοιπες διεργασίες, δηλαδή εκείνες που δεν εκτελούν χρέη συντονιστή, θα πρέπει να περιμένουν μηνύματα και να τα εξυπηρετούν με τη σειρά που έρχονται. Κάθε διεργασία κόμβος θα πρέπει να διατηρεί τοπικά την πληροφορία για το ποιές διεργασίες αντιστοιχούν σε ενεργούς κόμβους του διομότιμου συστήματος με έναν πίνακα κατακερματισμού παρόμοιο με αυτόν που διατηρεί και η διεργασία συντονιστής. Επίσης, κάθε τέτοια διεργασία θα πρέπει να διατηρεί έναν ακόμα πίνακα κατακερματισμού ο οποίος θα περιέχει τα αναγνωριστικά των (εικονικών) αρχείων που αποθηκεύει

αυτός ο κόμβος τοπικά. Για να ενημερώνονται σωστά αυτές οι δομές, χρειάζεται να χρησιμοποιηθεί ο μηχανισμός του broadcast στο σύστημα MPI.

Τύποι Γεγονότων

Η προσομοίωση του διομότιμου συστήματος απαιτεί να υποστηρίζονται τα παρακάτω γεγονότα:

J <id>: Γεγονός τύπου Join το οποίο σηματοδοτεί την εισαγωγή ενός νέου κόμβου στο διομότιμο σύστημα με αναγνωριστικό <id>. Μια διεργασία πρέπει να αντιστοιχιστεί με το νέο κόμβο. Η διεργασία αυτή δεν πρέπει να αντιστοιχεί σε κάποιον ήδη υπάρχον κόμβο στο διομότιμο σύστημα Πληροφορίες για το νέο κόμβο πρέπει να αποθηκευτούν στον πίνακα κατακερματισμού του συντονιστή. Επίσης, ο πίνακας κατακερματισμού αρχείων του νεοεισαχθέντος κόμβου πρέπει να αρχικοποιηθεί κατάλληλα. Πιο συγκεκριμένα, τα αρχεία που βρίσκονται στον πίνακα κατακερματισμού αρχείων του επομένου κόμβου στο δακτύλιο μοιράζονται ώστε να εξακολουθήσει να ισχύει η Συνθήκη (1) (δηλαδή κάθε κόμβος να έχει στον πίνακα κατακερματισμού αρχείων του εκείνα τα αρχεία για τα οποία ισχύει πως το αναγνωριστικό τους είναι μεταξύ του αναγνωριστικού του κόμβου και του αναγνωριστικού του προηγουμένου του κόμβου στο δακτύλιο). Για παράδειγμα, έστω ο κόμβος με αναγνωριστικό 50 και έστω ότι ο προηγούμενός του στον πίνακα κατακερματισμού των κόμβων είναι ο 20. Έστω ότι ο 50 έχει στον πίνακα κατακερματισμού αρχείων του τα αρχεία με αναγνωριστικά 25, 33, 36, 41, 46, 49. Αν στο σύστημα εισαχθεί ο κόμβος με αναγνωριστικό 40, ο 40 είναι πλέον ο προηγούμενος του 50 και τα αρχεία με αναγνωριστικά 25, 33 και 36 πρέπει να μεταφερθούν από τον πίνακα κατακερματισμού αρχείων του 50 στον πίνακα κατακερματισμού αρχείων του 40. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

J <id> DONE, PRED = <pid>, SUCC = <sid>, DocList =<fid1, fid2, …, fidr>

όπου <pid> και <sid> είναι τα αναγνωριστικά του προηγούμενου και του επόμενου, αντίστοιχα, του προς εισαγωγή κόμβου στο διομότιμο σύστημα και fid1, fid2, …, fidr είναι τα αναγνωριστικά των αρχείων που έχουν αποθηκευτεί στον πίνακα κατακερματισμού αρχείων του προς εισαγωγή κόμβου.

L <id>: Γεγονός τύπου Leave το οποίο σηματοδοτεί την αποχώρηση του κόμβου με αναγνωριστικό <id> από το διομότιμο σύστημα. Το αντίστοιχο μήνυμα γεγονότος πρέπει να σταλεί στον κατάλληλο κόμβο. Η διεργασία που αντιστοιχεί σε αυτόν τον κόμβο βρίσκεται από τον πίνακα κατακερματισμού του συντονιστή. Ο κόμβος θα πρέπει να παραχωρήσει όλα τα αρχεία του στον επόμενό του κόμβο πριν αποχωρήσει. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

L <id> DONE, PRED = <pid>, SUCC=<sid>, DOCLISTSUCC = <fid1,fid2,…,fidr>

όπου <pid> και <sid> είναι τα αναγνωριστικά του προηγούμενου και του επόμενου, αντίστοιχα, του κόμβου με αναγνωριστικό <id> που απεχώρησε από το σύστημα, ενώ fid1, fid2, …, fidr είναι τα αναγνωριστικά των αρχείων που είναι αποθηκευμένα στον πίνακα κατακερματισμού αρχείων του επομένου του κόμβου με αναγνωριστικό <id>.

I <id>: Γεγονός τύπου Insert το οποίο σηματοδοτεί την εισαγωγή ενός αρχείου με αναγνωριστικό <id> στο σύστημα. Το μήνυμα γεγονότος μπορεί να σταλεί σε οποιονδήποτε κόμβο που είναι ήδη μέρος του διομότιμου συστήματος. Το αρχείο θα πρέπει να τοποθετηθεί, βάσει του αναγνωριστικού του, στον πίνακα κατακερματισμού αρχείων του κατάλληλου κόμβου. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

I <id> DONE, NODEID = <nid> DOCLIST = <fid1,fid2,…,fidr>

όπου <nid> είναι το αναγνωριστικό του κόμβου στου οποίου τον πίνακα κατακερματισμού αρχείων αποθηκεύτηκε το αρχείο και ενώ fid1, fid2, …, fidr είναι τα αναγνωριστικά των αρχείων που είναι αποθηκευμένα στον πίνακα κατακερματισμού αρχείων του.

F <id>: Γεγονός τύπου Find το οποίο σηματοδοτεί την αναζήτηση ενός αρχείου με αναγνωριστικό <id> στο σύστημα. Το μήνυμα γεγονότος μπορεί να σταλεί σε οποιονδήποτε κόμβο που είναι ήδη μέρος του διομότιμου συστήματος. Το αρχείο θα πρέπει να ευρεθεί και να τυπωθεί ο κόμβος στον οποίο είναι αποθηκευμένο. Πιο συγκεκριμένα, θα πρέπει να τυπωθεί η ακόλουθη πληροφορία:

F <id> DONE, NODEID = <nid> DOCLIST = <fid1,fid2,…,fidr>

όπου <nid> είναι το αναγνωριστικό του κόμβου στου οποίου τον πίνακα κατακερματισμού αρχείων βρέθηκε το αρχείο, ενώ fid1, fid2, …, fidr είναι τα αναγνωριστικά των αρχείων που είναι αποθηκευμένα στον πίνακα κατακερματισμού αρχείων του.

D <id>: Γεγονός τύπου Delete το οποίο σηματοδοτεί τη διαγραφή του αρχείου με αναγνωριστικό <id> από το σύστημα. Το μήνυμα γεγονότος μπορεί να σταλεί σε οποιονδήποτε κόμβο που είναι ήδη μέρος του διομότιμου συστήματος. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

D <id> DONE, NODEID = <nid> = <fid1,fid2,…,fidr>

όπου <nid> είναι το αναγνωριστικό του κόμβου στου οποίου τον πίνακα κατακερματισμού αρχείων ήταν αποθηκευμένο το αρχείο πριν τη διαγραφή του, ενώ fid1, fid2, …, fidr είναι τα αναγνωριστικά των αρχείων που είναι αποθηκευμένα στον πίνακα κατακερματισμού αρχείων του.

P: Γεγονός τύπου Print που σηματοδοτεί την εκτύπωση των πινάκων κατακερματισμού των κόμβων και για κάθε έναν από αυτούς την εκτύπωση των αρχείων του. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

P DONE

NODEID = <id1>, DOCLIST = <fd_1^1, fd_2^1, …, fd_{r1}^1>

NODEID = <id2>, DOCLIST = <fd_1^2, fd_2^2, …, fd_{r2}^2>

…

NODEID = <idk>, DOCLIST = <fd_1^k, fd_2^k, …, fd_{rk}^k>

NO_NODES = <number of nodes> No_FILES = <number of files>

όπου <number of nodes> και <number of files> είναι το συνολικό πλήθος κόμβων και αρχείων, αντίστοιχα, στο διομότιμο σύστημα (προσέξτε ότι k = <number_of_nodes>), για κάθε i, $1 \leq i \leq k$, $<id_i>$ είναι το αναγνωριστικό του τρέχοντος προς εκτύπωση κόμβου, r_i είναι το πλήθος αρχείων που είναι αποθηκευμένα στον κόμβο με αναγνωριστικό $<id_i>$ και για κάθε j, $1 \leq j \leq k$, και fd_j^i είναι το αναγνωριστικό του j-οστού αρχείου του πίνακα κατακερματισμού αρχείων του κόμβου με αναγνωριστικό $<id_i>$.

Είναι αξιοσημείωτο ότι όταν ένα μήνυμα γεγονότος παραλαμβάνεται απο έναν κόμβο, δεν είναι απαραίτητο πως το μήνυμα αυτό αφορά το συγκεκριμένο κόμβο που το παρέλαβε. Πιο συγκεκριμένα, τα γεγονότα τύπου Join και Leave θα πρέπει <u>πάντα</u> να στέλνονται στη διεργασία που έχει αντιστοιχιστεί με τον κόμβο ο οποίος θα πρέπει να μπεί ή να βγεί αντίστοιχα απο το διομότιμο σύστημα. Ωστόσο, τα γεγονότα τύπου Insert, Delete και Find μπορούν να σταλούν σε οποιονδήποτε κόμβο είναι ήδη μέρος του διομότιμου συστήματος. Για παράδειγμα, θεωρήστε το διομότιμο σύστημα του Σχήματος 2. Έστω πως ο κόμβος N14 παραλαμβάνει ένα μήνυμα γεγονότος **F 38**. Όπως

φαίνεται και απο το σχήμα, ο κόμβος N14 δεν είναι αυτός που έχει το αρχείο με το αναγνωριστικό 38. Θα πρέπει λοιπόν να βρεθεί ποιος κόμβος είναι αυτός που αποθηκεύει το συγκεκριμένο αρχείο και να επιστραφεί το κατάλληλο αναγνωριστικό του κόμβου.

Προετοιμασία περιβάλλοντος

Για να λειτουργήσει σωστά το MPI σε παραπάνω απο 2 μηχανήματα, θα πρέπει να μπορείτε να κάνετε login μεταξύ των μηχανημάτων χωρίς να χρειάζεται password (passwordless login). Η διαδικασία είναι σχετικά απλή και περιγράφεται στον ακόλουθο σύνδεσμο http://www.linuxproblem.org/art_9.html .

Παράδοση Άσκησης

Η παράδοση των προγραμματιστικών ασκήσεων γίνεται μέσω του προγράμματος turnin. Το όνομα του παραδοτέου για την άσκηση είναι **project2.** Η εντολή turnin που πρέπει να χρησιμοποιηθεί είναι η εξής:

turnin project2@hy486 <dir>

όπου <dir> είναι ο φάκελος στον οποίο βρίσκονται τα αρχεία που πρέπει να παραδοθούν. Επειδή το πρόγραμμα turnin δεν καταχωρεί συμπιεσμένα αρχεία, τα αρχεία προς παράδοση δεν πρέπει να είναι binary/compressed/gzipped.