Δίκτυα Υπολογιστών Ι
Εργαστήρια

Άσκηση $3^{η}$

Δρομολόγηση

Πανεπιστήμιο Ιωαννίνων
Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Διδάσκων: Παπαπέτρου Ευάγγελος

1 Εισαγωγή

Σκοπός της άσκησης αυτής είναι η κατανόηση της έννοιας της δρομολόγησης. Για το σκοπό αυτό θα μελετηθεί ο κατανεμημένος αλγόριθμος διανυσμάτων απόστασης (Distance Vector - DV) που είναι γνωστός με το όνομα Bellman-Ford[1]. Ο αλγόριθμος αυτός προδιαγράφει ότι κάθε κόμβος του δικτύου διατηρεί έναν πίνακα δρομολόγησης με στοιχεία σχετικά με τις διαδρομές προς κάθε άλλο κόμβο του δικτύου. Κάθε στοιχείο του πίνακα αυτού περιέχει (α) το αναγνωριστικό του επόμενου κόμβου (nexthop) στο μονοπάτι προς τον αντίστοιχο τελικό κόμβο και (β) το κόστος (cost) της διαδρομής μέχρι τον τελικό κόμβο. Σύμφωνα με τον αλγόριθμο αυτό, κάθε κόμβος:

1. Αποστέλλει σε όλους τους γείτονές του τον πίνακα δρομολόγησης που διατηρεί. Με τον τρόπο αυτό όλοι οι κόμβοι του δικτύου λάμβάνουν τους πίνακες δρομολόγησης των γειτονικών τους κόμβων.
2. Εισέρχεται σε ένα βρόχο, στον οποίο:
   α. Χρησιμοποιεί την πληροφορία που έλαβε στο προηγούμενο βήμα για να βελτιώσει τον δικό του πίνακα δρομολόγησης, δηλαδή να βρει διαδρομές με μικρότερο κόστος.
   β. Αν προέκυψε κάποια μεταβολή στον πίνακα δρομολόγησής του, τότε ενημερώνει ξανά τους γείτονές του στέλνοντας τον νέο πίνακα δρομολόγησης.

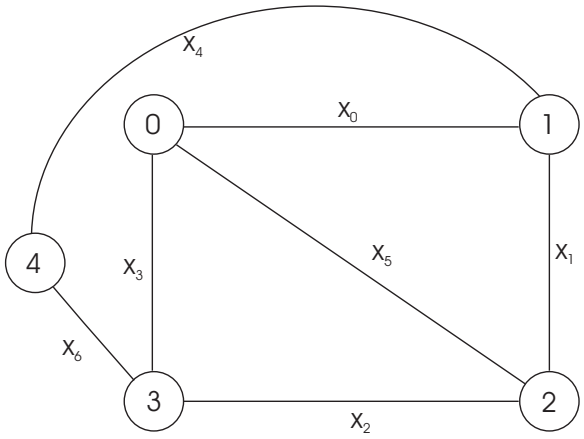
Όταν όλοι οι κόμβοι του δικτύου σταματήσουν να στέλνουν ενημερώσεις τότε ο αλγόριθμος έχει συγκλίνει και οι συντομότερες διαδρομές έχουν υπολογιστεί.

Η διαδικασία που περιγράφηκε αφορά μια εκτέλεση του αλγόριθμου δρομολόγησης που οδηγεί στον υπολογισμό των καλύτερων διαδρομών για τη δεδομένη χρονική στιγμή. Ωστόσο, σε μεταβαλλόμενα δίκτυα η παραπάνω διαδικασία επαναλαμβάνεται περιοδικά ώστε ο αλγόριθμος να είναι σε θέση να αντιλαμβάνεται τις μεταβολές στην κατάσταση του δικτύου. Είναι λοιπόν κατανοητό ότι ο χρόνος σύγκλισης του αλγόριθμου πρέπει να είναι μικρότερος από την περίοδο επανάληψης ώστε να μπορεί να θεωρηθεί ότι ο αλγόριθμος έχει καταλήξει στον υπολογισμό των καλύτερων διαδρομών.

2 Οδηγίες για την εκτέλεση της άσκησης

Για να μελετηθεί ο αλγόριθμος Bellman-Ford έχει υλοποιηθεί μία event-driven προσομοίωση που περιγράφει τη λειτουργία του δικτύου που εικονίζεται στο σχήμα 2. Το δίκτυο αυτό αποτελείται από 5 κόμβους και 7 αμφίδρομες συνδέσεις με κόστη $x_0, x_1, \ldots, x_6$ αντίστοιχα. Θεωρούμε ότι τα κόστη είναι ίδια και για τις δύο κατευθύνεις κάθε σύνδεσης. Ο πυρήνας της προσομοίωσης αποτελείται

[1] Η περιγραφή του αλγόριθμου βρίσκεται στο βιβλίο θεωρίας.



Σχήμα 1: Η τοπολογία του δικτύου στο οποίο θα υλοποιηθεί ο αλγόριθμος

από το αρχείο `dv-network.c` και υλοποιεί τους μηχανισμούς για τη μετάδοση χωρίς σφάλματα των πακέτων μεταξύ των κόμβων του δικτύου. Με άλλα λόγια, το αρχείο `dv-network.c` υλοποιεί όλους τους μηχανισμούς του δευτέρου επιπέδου κατά OSI. Να σημειωθεί ότι η προσομοίωση είναι δομημένη με τέτοιο τρόπο ώστε δεν χρειάζεται να διαβάσετε τον πηγαίο κώδικα του αρχείου αυτού. Η προσομοίωση συμπληρώνεται από τα αρχεία `node.h` και `node.c`. Η τοπολογία του δικτύου, δηλαδή το κόστος κάθε σύνδεσης[2], ορίζεται από τον πίνακα `connectcosts` στο αρχείο `node.c`. Στο ίδιο αυτό αρχείο ορίζονται οι ρουτίνες που σχετίζονται με τον αλγόριθμο δρομολόγησης και **τις οποίες θα πρέπει να υλοποιήσετε**. Περισσότερες πληροφορίες σχετικά με τις ρουτίνες αυτές θα δοθούν στην επόμενη ενότητα. Τα προαναφερθέντα αρχεία, καθώς και ένα τυπικό makefile, διατίθενται μαζί με την παρούσα εκφώνηση. Για τη μεταγλώττιση του κώδικα μπορείτε να χρησιμοποιήσετε, από τον φάκελο στον οποίο βρίσκονται τα παραπάνω αρχεία, την εντολή:

```
user@host:...$ make
```

Το εκτελέσιμο (`sim`) που θα προκύψει, δέχεται ως όρισμα γραμμής εντολών (`-t <number>`) το "επίπεδο" εκτύπωσης διαγνωστικών μηνυμάτων που αφορούν τα πακέτα που ανταλλάσσονται μεταξύ των κόμβων του δικτύου. Οποιαδήποτε τιμή μικρότερη ή ίση του 1 συνεπάγεται μη εκτύπωση διαγνωστικών, ενώ αυξανόμενες τιμές μεγαλύτερες του 1 συνεπάγονται και αύξηση του αριθμού των μηνυμάτων που θα βλέπετε. Για τους σκοπούς του εργαστηρίου, χρησιμοποιήστε την τιμή 2 ώστε να εκτυπώνονται κάποια βασικά διαγνωστικά μηνύματα:

```
user@host:...$ ./sim -t 2
```

[2]Το κόστος μιας σύνδεσης που δεν υπάρχει θεωρείται στην προσομοίωση ίσο με 999.

3 Ρουτίνες υλοποίησης του αλγόριθμου

Παρακάτω περιγράφονται οι ρουτίνες που υλοποιούν τον αλγόριθμο δρομολόγησης και τις οποίες θα πρέπει να συμπληρώσετε εσείς:

- `initRT(Node *n)`: η ρουτίνα αυτή καλείται μία φορά στην αρχή της προσομοίωσης και δέχεται ως όρισμα έναν δείκτη προς μια δομή κόμβου την οποία και θα αρχικοποιήσει. Σκοπός της είναι να δώσει αρχικές τιμές στον πίνακα δρομολόγησης (`rt`) του κόμβου αυτού. Οι αρχικές τιμές πρέπει να είναι τα κόστη[2] των συνδέσεων προς του γειτονικούς κόμβους, σύμφωνα με τον πίνακα `connectcosts`. Κατόπιν πρέπει να στείλει προς τους γειτονικούς του κόμβους τον πίνακα δρομολόγησης. Η αποστολή της πληροφορίας γίνεται με τη ρουτίνα `tolayer2(RtPkt packet)`, η οποία είναι ήδη υλοποιημένη στο αρχείο `dv-network.c` και μπορεί να χρησιμοποιηθεί ως έχει. Δέχεται ως όρισμα ένα πακέτο το οποίο πρέπει να κατασκευάσετε και το οποίο αποτελείται από τα πεδία:
  - `sourceid`: το αναγνωριστικό του κόμβου αποστολέα.
  - `destid`: η διεύθυνση του κόμβου παραλήπτη.
  - `mincost[ ]`: ένας πίνακας που περιέχει το μικρότερο κόστος που γνωρίζει ο κόμβος που δημιούργησε το πακέτο προς κάθε άλλο κόμβο του δικτύου.

  Στο σημείο αυτό πρέπει να σημειωθεί ότι η θέση κάθε στοιχείου του πίνακα `mincost` υποδηλώνει το αναγνωριστικό (δηλ. τη "διεύθυνση") του κόμβου προς τον οποίο αναφέρεται το αντίστοιχο κόστος. Με άλλα λόγια, η τιμή που περιέχεται στη θέση q του πίνακα `mincost` που κατασκευάζει ο κόμβος p, περιέχει το κόστος της διαδρομής που γνωρίζει ο κόμβος p προς τον κόμβο q. Η χρησιμοποίηση της ρουτίνας `tolayer2(...)` έχει ως αποτέλεσμα η προσομοίωση να "παραδώσει" το πακέτο `packet` στον κόμβο που αντιστοιχεί στο πεδίο `destid` του πακέτου. Η "παράδοση" του πακέτου γίνεται με κλήση της ρουτίνας `updateRT(...)` του κόμβου `destid`.

- `updateRT(Node *n, RtPkt* rcvdpkt)`: η ρουτίνα αυτή είναι η ουσία του αλγόριθμου δρομολόγησης. Καλείται από την προσομοίωση όταν πρέπει να παραδοθεί ένα πακέτο σε κάποιον κόμβο. Το πρώτο όρισμα περιέχει έναν δείκτη προς τον κόμβο παραλήπτη, ενώ το δεύτερο όρισμα περιέχει ένα δείκτη στο πακέτο που πρέπει να λάβει ο κόμβος αυτός. Με άλλα λόγια, κλήση της ρουτίνας `updateRT(n, rcvdpkt)` θα σημάνει παράδοση ενός πακέτου `rcvdpkt` στον κόμβο `n`. Με βάση τα περιεχόμενα του πακέτου αυτού ο κόμβος πρέπει να ενημερώσει τον πίνακα δρομολόγησής του για να πετύχει πιθανές μειώσεις στο κόστος

ή/και αλλαγή του επόμενου βήματος προς κάποιο κόμβο του δικτύου. Αν υπάρξουν μεταβολές στον πίνακα δρομολόγησης του κόμβου, τότε πρέπει να ενημερώσει όλους τους γειτονικούς κόμβους κατασκευάζοντας και αποστέλλοντας προς κάθε γείτονα ένα κατάλληλο πακέτο με τον ανανεωμένο πίνακα δρομολόγησής του.

- `PrintRT(Node *n)`: η ρουτίνα αυτή καλείται μία φορά στο τέλος της προσομοίωσης και σκοπό έχει να τυπώσει τον πίνακα δρομολόγησης του κόμβου στον οποίο δείχνει ο δείκτης `n`.

4 Παραδείγματα ζητούμενων

Ζητείται να γράψετε τις ρουτίνες που υλοποιούν τον αλγόριθμο δρομολόγησης (δηλαδή να συμπληρώσετε το αρχείο node.c) ώστε αυτός να βρίσκει τις συντομότερες διαδρομές στο δίκτυο για κάθε ζεύγος αρχής/προορισμού. Τα κόστη των συνδέσεων μπορείτε να τα υπολογίσετε από τις σχέσεις:
x_0=2+(αριθμός_μητρώου MOD 3)
x_1=1+(αριθμός_μητρώου MOD 3)
x_2=4+(αριθμός_μητρώου MOD 3)
x_3=3+(αριθμός_μητρώου MOD 3)
x_4=5+(αριθμός_μητρώου MOD 3)
x_5=6+(αριθμός_μητρώου MOD 3)
x_6=2+(αριθμός_μητρώου MOD 3)

Ερωτήσεις

1. Πόσα μηνύματα ανταλλάσσονται από τους κόμβους του δικτύου μέχρι ο αλγόριθμος να συγκλίνει;

2. Από ποιους παράγοντες εξαρτάται ο αριθμός των μηνυμάτων που ανταλλάσσονται μέχρι την σύγκλιση;

3. Από ποιους παράγοντες εξαρτάται ο χρόνος που μεσολαβεί μέχρι την σύγκλιση;

4. Η σειρά εκπομπής των κόμβων του δικτύου επηρεάζει το χρόνο σύγκλισης του αλγόριθμου δρομολόγησης; Δικαιολογήστε την απάντησή σας με ένα παράδειγμα.

5. Η σειρά εκπομπής των κόμβων του δικτύου επηρεάζει το αποτέλεσμα του αλγόριθμου δρομολόγησης; Δικαιολογήστε την απάντησή σας.

6. Τί πιστεύετε ότι συμβαίνει σχετικά με τη σύγκλιση του αλγόριθμου όταν το μέγεθος ενός δικτύου (αριθμός κόμβων και συνδέσμων) αυξάνει;

7. Πόσες ενημερώσεις κατ' ελάχιστο χρειάζεται να λάβει ένας κόμβος από τους γείτονές του μέχρι να μάθει μια διαδρομή για έναν κόμβο που βρίσκεται k άλματα μακριά;

8. Αν καταρρεύσει ένας κόμβος ή ένας σύνδεσμος στο δίκτυο πως μπορεί ο αλγόριθμος δρομολόγησης να αντιμετωπίσει το φαινόμενο αυτό;

9. Ποιες πιστεύετε ότι είναι οι επιπτώσεις στην λειτουργία του αλγόριθμου δρομολόγησης αν υπάρχει η πιθανότητα να χαθεί μια ενημέρωση πριν αυτή φτάσει στον παραλήπτη της;

10. Τροποποιήστε τον κώδικα που γράψατε ώστε να προσομοιώσετε μια πιθανότητα p απώλειας μιας ενημέρωσης.