Κινητός και διάχυτος υπολογισμός

Project εξαμήνου 2018-2019

‘On-demand data broadcast with deadlines for avoiding conflicts in wireless networks’

Φοιτητές: Κανελλοπούλου Αικατερίνη 2074

Ράντου Καλλιόπη 2004

Υπεύθυνος καθηγητής: Κατσαρός Δημήτριος

Εισαγωγή

Σε on-demand data broadcast systems ο server τοποθετεί τα requests των clients σε κανάλια, μέσω των οποίων αυτοί θα πάρουν τελικά τα δεδομένα. Κάθε ένα request έχει ένα συγκεκριμένο deadline το οποίο χρησιμοποιείται για την απόρριψή του ή όχι. Στόχος είναι η αποφυγή των conflict, τα οποία δημιουργούνται κατά την μετάδοση δεδομένων την ίδια χρονική στιγμή.

Upf algorithm

Ο upf algorithm προσεγγίζει το πρόβλημα αυτό εκπέμποντας τα πιο δημοφιλη ως προς τη ζήτηση δεδομένα, προσπαθώντας έτσι να μειώσει το ρυθμό αστοχίας.

Συγκεκριμένα, παίρνει ως είσοδο ένα σύνολο από requests (Rs) και υπολογίζει το slack time τους. Επιλέγει το κανάλι με τα λιγότερα στοιχεία και εκπέμπει τελικά το request με το μικρότερο slack time, ξεκινώντας από τα δεδομένα με τη μεγαλύτερη ζήτηση. Το request αφαιρείται από τη λίστα.

Το slack time υπολογίζεται έτσι ώστε να αποφεύγεται το 2-conflicts ανάμεσα στα data. Όταν εκπέμπεται κάποιο Request, το νέο broadcast time διαμορφώνεται ως $t_{brod\text{-}time} = t_{brod\text{-}time} + 2$. Σε περίπτωση αρνητικού slack time, όπου το αίτημα δεν μπορεί να εκπεμφθεί μέσα στο deadline του τοποθετείται στη λίστα Ru, την οποία ο upf θα χρησιμοποιήσει μόλις τελειώσει η Rs. Έτσι για κάθε ένα αίτημα θα εκπέμψει στα όρια των deadline ώστε να μειωθεί η αστοχία (deadline miss ratio).

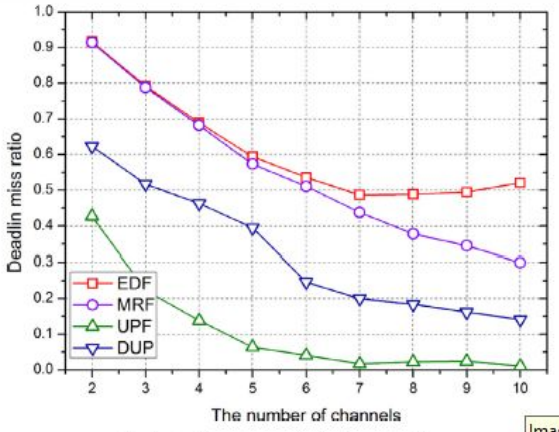

Fig. 9. Deadline miss ratio for database size 100.

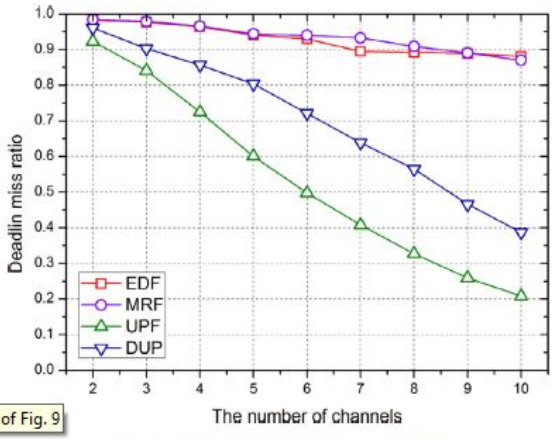

Image of Fig. 9

Fig. 11. Deadline miss ratio for database size 500.

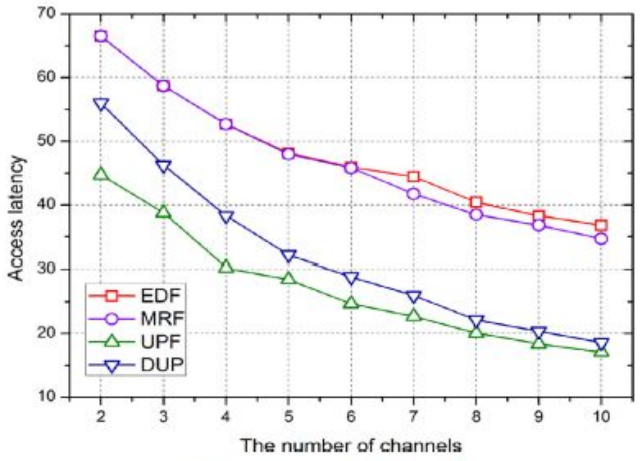

Fig. 13. Access latency for database size 100.

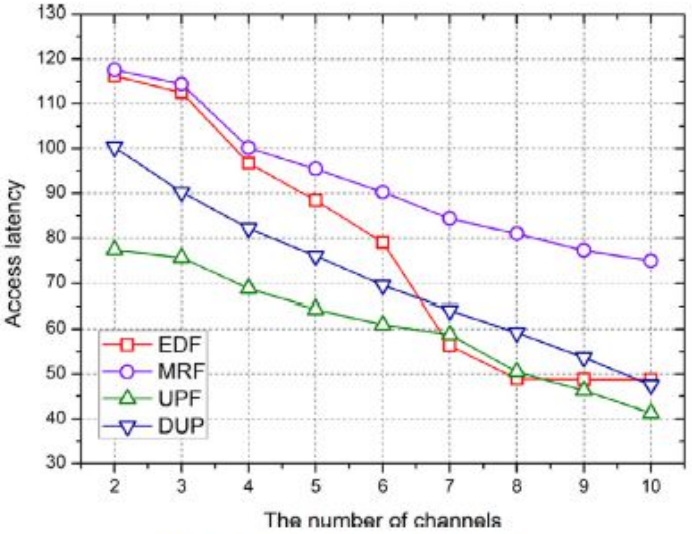

Fig. 15. Access latency for database size 500.

Υλοποίηση του αλγορίθμου

Κλάση **dataObj** με **attributes**:

- id: id για κάθε data του Rs.
- channel: κανάλι όπου έχει δεσμευτεί το data.
- t_alloc (time of allocation) : χρονική στιγμή δέσμευσης.

Συναρτήσεις:

- **create_Req_obj**: Δημιουργία αντικειμένου - request. Παίρνει ως όρισμα το id του request.
- **items_of_requests**: Υπολογισμός μη δεσμευμένων αντικειμένων του κάθε R. Παίρνει ως όρισμα ένα request.
- **min_items_channel**: Εντοπισμός καναλιού με τα λιγότερα στοιχεία. Αν όλα τα κανάλια είναι κενά ή έχουν τον ίδιο αριθμό δεσμευμένων αντικειμένων επέλεξε αυτό με το μικρότερο broadcast time.
- **find_min_slacktime**: Υπολογισμός του μικρότερου slacktime από τα R. Παίρνει ως όρισμα το επιλεγμένο κανάλι με τα λιγότερα στοιχεία και τα requests.
- **num_of_data**: Εύρεση του αριθμού των εμφανίσεων του ζητούμενου δεδομένου στο R. Παίρνει ως όρισμα το data του οποίου ζητάμε τον αριθμό εμφανίσεων.
- **channel_filling**: Εισαγωγή ‘αδέσμευτων’ δεδομένων στο επιλεγμένο κανάλι. Παίρνει ως όρισμα το επιλεγμένο κανάλι, το request προς εκπομπή και το τρέχον time.
- **channel_filling_Ru**: Εισαγωγή ‘αδέσμευτων’ δεδομένων από Ru στο επιλεγμένο κανάλι. Ίδια συνάρτηση η οποία χρησιμοποιεί την Ru.
- **initialize_globals**: Αρχικοποίηση πινάκων.
- **example1**, **example 2**: Οι συναρτήσεις example1, example2 υλοποιούν τα παραδείγματα του paper με τα συγκεκριμένα requests.

Το πρόγραμμα ξεκινάει εμφανίζοντας το Menu στο χρήστη για το παράδειγμα 1, παράδειγμα 2 ή έξοδο. Τερματίζει μόνο με εντολή του χρήστη.

```
------------MENU------------
Choose  1  for first example with no missed deadlines.
Choose  2  for second example with missed deadlines.
Choose  3  for exiting.
----------------------------
```

Example 1:

```
----------------------------
Initialized requests
----------------------------
R1
d1 d2 d3
with deadline: 6

R2
d2 d3 d4 d5
with deadline: 9

R3
d3 d4
with deadline: 3

----------------------------
    PROGRAM STARTS
----------------------------
Selected channel is 1

Selected R is
R3
d3 d4
with slacktime: 0 and current channel time 1

AFTER FILLING CHANNEL1:['d3', 'd4']

REMAINING REQUESTS:
R1 R2

--------NEXT BROADCAST-------
Selected channel is 2

Selected R is
R1
d1 d2
with slacktime: 1 and current channel time 3

AFTER FILLING CHANNEL2:[None, None, 'd2', 'd1']
```

```
REMAINING REQUESTS:
R2

--------NEXT BROADCAST-------
Selected channel is 1

Selected R is
R2
d5
with slacktime: 3 and current channel time 5

AFTER FILLING CHANNEL1:['d3', 'd4', None, None, 'd5']

REMAINING REQUESTS:
There are no more remaining requests!

--------NEXT BROADCAST-------

--------Ru HANDLING-------
There are no more remaining requests in Ru!
----------------------------
          PROGRAM ENDS
----------------------------
```

Example 2:

```
-----------------------------
Initialized requests
-----------------------------
R1
d1 d2 d3
with deadline: 4

R2
d4 d5 d6 d2
with deadline: 5

R3
d7 d8
with deadline: 9

-----------------------------
    PROGRAM STARTS
-----------------------------
Selected channel is 1

Selected R is
R2
d4 d5 d6 d2
with slacktime: 0 and current channel time 1

AFTER FILLING CHANNEL1:['d2', 'd4', 'd5', 'd6']

REMAINING REQUESTS:
R1 R3

--------NEXT BROADCAST-------
Selected channel is 2

Negative slacktime for R1, add R to Ru!

Selected channel is 2

Selected R is
R3
d7 d8
```

```
with slacktime: 0 and current channel time 1

AFTER FILLING CHANNEL2:['d7', 'd8']

REMAINING REQUESTS:
There are no more remaining requests!

--------NEXT BROADCAST-------

--------Ru HANDLING-------
Selected channel is 2

Selected R is
R1
d1 d3
with slacktime: -1 and current channel time 3

AFTER FILLING CHANNEL2:['d7', 'd8', 'd1']

-----------------------------
          PROGRAM ENDS
-----------------------------
```