ΗΜΜΥ Πολυτεχνειου Κρητης

ΠΛΗ 516-Επεξεργασια και Διαχειριση Δεδομενων σε Δικτυα Αισθητηρων

Χειμερινο Εξαμηνο 2018-2019

Εργασία TinyOS: Φάση 2

Φοιτητές
Καλογερακης Στεφανος
Παγκαλος Μανωλης

Διδάσκων
Α. Δεληγιαννακης

Εισαγωγή

Το δεύτερο μέρος της εργασίας αποτελεί μια επέκταση του πρώτου με περισσότερες συναρτήσεις για το TAG καθώς και την προσθήκη του TiNA. Πιο συγκεκριμένα, υλοποιήσαμε τέσσερις επιπλέον επιλογές πέραν των **MAX** και **AVG** που υπολογίζονται στο πρώτο μέρος. Οι συναρτήσεις που προσθέσαμε είναι οι **MIN**, **COUNT**, **SUM** και **VARIANCE**. Στο πρόγραμμα επιλέγεται τυχαία αν θα εκτελεστεί μια ή δύο από τις παραπάνω συναρτήσεις. Σε περίπτωση επιλογής δύο συναρτήσεων θα πρέπει αυτές να υπολογιστούν ταυτόχρονα. Διαφορετικά αν επιλεγεί μια μόνο συνάρτηση, είτε θα γίνει η συνάθροιση όπως προτάσσει το TAG είτε θα γίνει με εφαρμογή του TiNA για περιορισμό μηνυμάτων. Αν θα εφαρμοστεί το TiNA ή όχι επιλέγεται επίσης τυχαία από το πρόγραμμα.

Διορθώσεις φάσης 1

Σχετικά με την πρώτη φάση δεν αφαιρέσαμε κάποιο επιπλέον κομμάτι κώδικα. Πραγματοποιήσαμε μια σημαντική όμως αλλαγή στην λογική της υλοποίηση μας μετά από παράληψη στην πρώτη φάση. Πιο συγκεκριμένα, οι πράξεις και οι έλεγχοι υλοποιούνται μέσα σε tasks και όχι απευθείας σε events. Αυτό θέλουμε να συμβαίνει καθώς τα events είναι καλή πρακτική να είναι όσο το δυνατό πιο μικροσκελή.

Παρατήρηση: Αξίζει να σημειωθεί ότι η επιλογή του χρονισμού αποστολής πληροφοριών των κόμβων έγινε με βασικό κριτήριο την τυχόν αποφυγή των συγκρούσεων στα μηνύματα. Αναφέρουμε αυτό, καθώς θα μπορούσαμε να στέλνουμε σε πιο πυκνό διάστημα την πληροφορία και να έχει σταλεί σε μικρότερο χρονικό διάστημα αλλά με αρκετά μεγαλύτερη απώλεια πληροφορίας. Μετά από πολλές δοκιμές, παρατηρήσαμε ότι ο χρόνος που εξοικονομούνταν σε σχέση με τον αριθμό απώλειας μηνυμάτων ήταν μεγάλος, γεγονός που μας οδήγησε στην υλοποίηση από την στιγμή μάλιστα που δεν υπήρξε κάποια σχετική υπόδειξη από την εκφώνηση.

Πρόγραμμα φάσης 2

Επέκταση TAG σε περισσότερες συναρτήσεις

Για την επέκταση του TAG, το δίκτυο πρέπει να υλοποιηθεί με βασικό στόχο τη βέλτιστη εξοικονόμηση ενέργειας. Για να το πετύχουμε αυτό πρέπει η πληροφορία που στέλνεται σε κάθε συνάθροιση να είναι η ελάχιστη απαραίτητη που χρειάζεται για τον υπολογισμό των εκάστοτε συναρτήσεων. Φυσικά, υπάρχει το πλεονέκτημα ότι κάποιες συναρτήσεις εμπεριέχονται σε άλλες, αυτό μας επιτρέπει να εκμεταλλευτούμε την επικάλυψη δεδομένων και να μην στείλουμε παρόμοια πληροφορία. Στην συνέχεια κάνουμε την ανάλυση των σύνθετων συναρτήσεων.

$$AVG = \frac{SUM}{COUNT},$$

$$VARIANCE = \frac{SUM\_OF\_SQUARES}{COUNT} - (\frac{SUM}{COUNT})^2$$

Παρατηρούμε ότι για την υλοποίηση των AVG και $VARIANCE$ χρειαζόμαστε τις συναρτήσεις $COUNT$ και SUM τις οποίες πρέπει να υλοποιήσουμε καθώς και μια καινούρια παράμετρο, την $SUM\_OF\_SQUARES$ η οποία είναι το άθροισμα των τετραγώνων των SUM όλων τον κόμβων ενός υπό-δέντρου. Επομένως, αν θέλουμε για παράδειγμα να υπολογίσουμε την AVG και την $VARIANCE$ θα χρειαστεί να στείλουμε μόνο αυτές τις τρεις παραμέτρους τις οποίες θα χρησιμοποιήσουν αμφότερες. Αν συλλέξουμε όλες τις παραμέτρους που στέλνονται από όλες τις συναρτήσεις καταλήγουμε στις εξής

$$AggregationMap = \begin{cases} MIN \\ MAX \\ COUNT \\ AVG \\ SUM \\ VARIANCE \end{cases}$$

Με βάση αυτές μπορούμε να υπολογίσουμε οποιαδήποτε συνάρτηση μας ζητηθεί. Δεν είναι όμως καλή υλοποίηση να υπολογίζονται και να στέλνονται όλες οι παράμετροι σε κάθε εποχή ανεξαρτήτου των συναρτήσεων που ζητούνται.Η βέλτιστη επιλογή είναι να αναλύσουμε όλους τους πιθανούς συνδυασμούς και να θέσουμε σε κάθε περίπτωση τον αντίστοιχο αριθμό πληροφοριών που θα στέλνονται.

Η υλοποίησης μας έχει ως βασικό στόχο την γενίκευση του προβλήματος και ακολουθώντας την παραπάνω λογική καταλήξαμε στο συμπέρασμα να δημιουργήσουμε τέσσερις διαφορετικές abstract δομές με την κάθε μια να περιέχει από ένα έως τέσσερα πεδία αντίστοιχα. Δεν υπάρχει περίπτωση που να απαιτούνται περισσότερα των τεσσάρων πεδίων. Στην συνέχεια θα παραθέσουμε τις περιπτώσεις σύμφωνα με τις οποίες επιλέγεται το κάθε structure.

- **Structure 1 (1 πεδίο πληροφορίας)**

  Αυτή η περίπτωση συναντάται όταν υλοποιούμε μόνο μια συνάρτηση και η οποία δεν χρειάζεται περισσότερες από μια παραμέτρους για να υπολογιστεί. Αυτές είναι οι $MIN, MAX, COUNT$ και SUM.

- **Structure 2 (2 πεδία πληροφορίας)**

  Αυτή η δομή χρησιμοποιείται στις περιπτώσεις όπου δύο παράμετροι είναι αρκετές για τη συνάρτηση ή τις συναρτήσεις που θα υπολογιστούν. Αυτές οι περιπτώσεις είναι οι εξής:

  1. AVG
  2. $AVG \,\&\&\, (COUNT \,||\, SUM)$
  3. $MAX \,\&\&\, (MIN \,||\, COUNT \,||\, SUM)$
  4. $MIN \,\&\&\, (COUNT \,||\, SUM)$
  5. $COUNT \,\&\&\, SUM$

  Οι παραπάνω περιπτώσεις είναι 9.

- **Structure 3 (3 πεδία πληροφορίας)**

  Εδώ καλύπτονται οι περιπτώσεις όπου είναι απαραίτητες ακριβώς τρεις παράμετροι.

  1. $VARIANCE$
  2. $VARIANCE \,\&\&\, AVG$
  3. $AVG \,\&\&\, (MIN \,||\, MAX)$
  4. $VARIANCE \,\&\&\, (COUNT \,||\, SUM)$

  Οι παραπάνω περιπτώσεις είναι 6.

- **Structure 4 (4 πεδία πληροφορίας)**

  Τέλος, οι περιπτώσεις όπου στέλνεται ο μέγιστος αριθμός πληροφοριών, δηλαδή 4, καλύπτονται από την τέταρτη δομή. Αυτές οι περιπτώσεις είναι 2.

  1. $VARIANCE \,\&\&\, (COUNT \,||\, SUM)$

Παραθέτουμε εδώ το κομμάτι του κώδικα όπου δημιουργούμε τα structures.

```
38 /**
39     Created the most abstract structs to generalize the problem.
40     Our main issue was to minimize the messages send. The most
41     messages demanded are when variance and max/min are combined(4 fields).
42     The least is in the case where only one message is sent. In every case,
43     i must explain in comments where each field is refered.
44 */
45 typedef nx_struct DistrMsg4{
46     nx_uint16_t field4a;
47     nx_uint16_t field4b;
48     nx_uint16_t field4c;
49     nx_uint16_t field4d;
50 } DistrMsg4;
51
52 typedef nx_struct DistrMsg3{
53     nx_uint16_t field3a;
54     nx_uint16_t field3b;
55     nx_uint16_t field3c;
56 } DistrMsg3;
57
58 typedef nx_struct DistrMsg2{
59     nx_uint16_t field2a;
60     nx_uint16_t field2b;
61 } DistrMsg2;
62
63 typedef nx_struct DistrMsg1{
64     nx_uint16_t field1a;
65 } DistrMsg1;
66
```

Σχήμα 1: Κώδικας υλοποίησης δομών

Επισημαίνουμε και πάλι ότι τα πεδία είναι δηλωμένα αφηρημένα καθώς η πληροφορία που μεταφέρεται από κάθε πεδίο διαφέρει για τους διάφορους συνδυασμούς.

Στην πρώτη εποχή επιλέγεται τυχαία ποια ή ποιες συναρτήσεις να υπολογίσει και να εμφανίσει το πρόγραμμα. Στέλνεται το αντίστοιχο μήνυμα από τη ρίζα προς τα κάτω και σε αυτό το σημείο ελέγχουμε για ποια περίπτωση πρόκειται, συνεπώς και ποια δομή θα χρησιμοποιηθεί(Βλ. Παράρτημα).

```
chooseProg = call RandomGen.rand16() % 2 + 1;

dbg("SRTreeC","Program chose to execute 2.%d\n",chooseProg);

if(chooseProg == 1){    //case 2.1 question is chosen

    numFun = call RandomGen.rand16() % 2 + 1;
    dbg("SRTreeC","\n\nFunction num: %d\n\n\n",chooseProg);
    if(numFun == 2){
        chooseFun1 = call RandomGen.rand16() % 6 + 1;
        chooseFun2 = call RandomGen.rand16() % 6 + 1;

        //TODO must check that random numbers always differ
        if((chooseFun1 == 1 || chooseFun2 == 1) || (chooseFun1 == 2 || chooseFun2 == 2)){   /** Case that one of the g

            if(chooseFun1 == 6 || chooseFun2 == 6){ //case that one of the choices is VARIANCE
                //case 4
                //numMsgSent = 4;
                chooseQues = chooseFun1 * 1000 + chooseFun2;
            }
            else if(chooseFun1 == 4 || chooseFun2 == 4){    //case that one of the choises is AVG
                //case 3
                //numMsgSent = 3;
                chooseQues = chooseFun1 * 100 + chooseFun2;
            }
            else if((chooseFun1 == 3 || chooseFun2 == 3) || (chooseFun1 == 5 || chooseFun2 == 5)){  //case that one of
                //case 2
                //numMsgSent = 2;
                chooseQues = chooseFun1 * 10 + chooseFun2;
            }
```

Σχήμα 2: Ενδεικτικό σημείο τυχαίας επιλογής συναρτήσεων

Στη συνέχεια, ανάλογα με την περίπτωση που υπάρχει θα εισάγουμε στα πεδία τις αντίστοιχες απαραίτητες πληροφορίες. Ακόμη, κάναμε προσθήκες στο σημείο του κώδικα όπου γίνεται η συνάθροιση των πληροφοριών αυτών ελέγχοντας πάντα να γίνεται σωστή συνάθροιση για όλα τα πιθανά σενάρια. Ενημερώνουμε επίσης κάθε φορά των πίνακα με τα παιδιά όπως και στην πρώτη φάση της άσκησης. Στο σχήμα 3 , παραθέτουμε ενδεικτικά σημείο του κώδικα που γίνεται συνάθροιση δεδομένων ανάλογα με την κάθε περίπτωση.

```
}else if(numMsgSent == 3){
    /**
    Check all possible combinations where we need three attributes
    */
    mrpkt3 = (DistrMsg3*) (call DistrPacket.getPayload(&tmp, sizeof(DistrMsg3)));

    if(numFun==2 && (chooseFun1==1 || chooseFun2==1 || chooseFun1==2 || chooseFun2==2)){  //Definitely case AVG + (MIN or MAX)
        atomic{
        mrpkt3->field3a = randVal;  /*used as sum*/
        mrpkt3->field3b = 1;    /*used as count*/
        mrpkt3->field3c = randVal; /*used as min or max */
        }

        /*Aggregation of values*/
        for(i = 0 ;i < MAX_CHILDREN && childrenArray[i].senderID!=0 ; i++){
            mrpkt3->field3a += childrenArray[i].sum;
            mrpkt3->field3b += childrenArray[i].count;
            if(chooseFun1==1 || chooseFun2==1) // case min
               mrpkt3->field3c = minFinder(childrenArray[i].min, mrpkt3->field3c);
            else // case max
               mrpkt3->field3c = maxFinder(childrenArray[i].max, mrpkt3->field3c);
        }
    }else{   // cases VARIANCE, VARIANCE + SUM, VARIANCE + COUNT, VARIANCE + AVG all need the same attributes
        atomic{
        mrpkt3->field3a = randVal;  /*used as sum*/
        mrpkt3->field3b = 1;    /*used as count*/
        mrpkt3->field3c = randVal * randVal; /*used as sumofSquares*/
        }

        /*Aggregation of values*/
        for(i = 0 ;i < MAX_CHILDREN && childrenArray[i].senderID!=0 ; i++){
        mrpkt3->field3a += childrenArray[i].sum;
        mrpkt3->field3b += childrenArray[i].count;
        mrpkt3->field3c += childrenArray[i].sumofSquares;
        }
    }
```

Σχήμα 3: Συνάθροιση συναρτήσεων 3ης δομής

Υλοποίηση του TiNA

Στην δεύτερη φάση υλοποιήσαμε την λειτουργικότητα του TiNA για τον υπολογισμό των συναρτήσεων SUM, MAX, MIN και $COUNT$. Σκοπός αυτής της υλοποίησης ήταν η μείωση του πλήθους των μηνυμάτων που στέλνονται σε κάθε εποχή διατηρώντας όμως την ποιότητα των αποτελεσμάτων. Η επιλογή της συνάρτησης προς υπολογισμό είναι τυχαία. Σε αυτό το σημείο να επισημάνουμε ότι η βελτιστοποίηση για αποκοπή μηνυμάτων συμβαίνει και στους ενδιάμεσους κόμβους, όχι μόνο σε κόμβους φύλλα, καθώς όπως επισημάναμε και στο μάθημα έχουμε αυτή την δυνατότητα σε περίπτωση που οι τιμές που συναθροίζονται είναι ομόσημες(στην περίπτωση μας συναθροίζονται σε κάθε εποχή τιμές μεταξύ του 0 και 50). Αυτό σημαίνει ότι κάθε κόμβος πατέρας που συναθροίζει τις τιμές των παιδιών του και την δική του θα συγκρίνει αυτό το αποτέλεσμα με την προηγούμενη τιμή που είχε αποστείλει προς τα πάνω. Όπως προτείνει το TiNA ο έλεγχος που κάναμε σε κάθε κόμβο για το αν θα στείλει τη νέα του τιμή είναι

$$\frac{|Vnew - Vold|}{|Vold|} > tct$$

Στην περίπτωση που η παραπάνω σχέση επιβεβαιωθεί για ένα προκαθορισμένο tct ο κόμβος θα στείλει τη νέα τιμή, διαφορετικά δεν θα στείλει τίποτα και ο πατέρας του θα χρησιμοποιήσει την παλιά τιμή που έχει διατηρήσει.

```
/**
    In that section, we implement TINA. We know as a fact that both numbers
    are positive,so we can compare their aggregate values not only in the
    leaves but in the whole tree. So if the statement below is true, just keep
    the old values to minimize messages sent. TCT is defined in SimpleRoutingTree.h
*/
if(chooseProg == 2 && !(abs(mrpkt1->field1a - Vold) > abs(Vold) * ((double)TCT/(double)PERCENTAGE))){   //don't change value
    oldFlag = 1;
    dbg("SRTreeC","Don't send message with new value %d and old value %d\n", mrpkt1->field1a, Vold);
}else if( chooseProg == 2){ //change value and update old
    oldFlag = 0;
    Vold = mrpkt1->field1a;
}

//Check that the value can be updated (2.2)
if(oldFlag == 0){


    call DistrPacket.setPayloadLength(&tmp, sizeof(DistrMsg1));

    if(chooseFun == 1){ //min case
        dbg("SRTreeC", "Node: %d , Parent: %d, min: %d, depth: %d\n",TOS_NODE_ID,parentID, mrpkt1->field1a,curdepth);
    }else if( chooseFun == 2){  //max case
        dbg("SRTreeC", "Node: %d , Parent: %d, max: %d, depth: %d\n",TOS_NODE_ID,parentID, mrpkt1->field1a,curdepth);
    }else if(chooseFun == 3){   //count case
        dbg("SRTreeC", "Node: %d , Parent: %d, count: %d, depth: %d\n",TOS_NODE_ID,parentID, mrpkt1->field1a,curdepth);
    }else{  // sum count
        dbg("SRTreeC", "Node: %d , Parent: %d, sum: %d, depth: %d\n",TOS_NODE_ID,parentID, mrpkt1->field1a,curdepth);
    }
```

Σχήμα 4: Κώδικας υλοποίησης TiNA

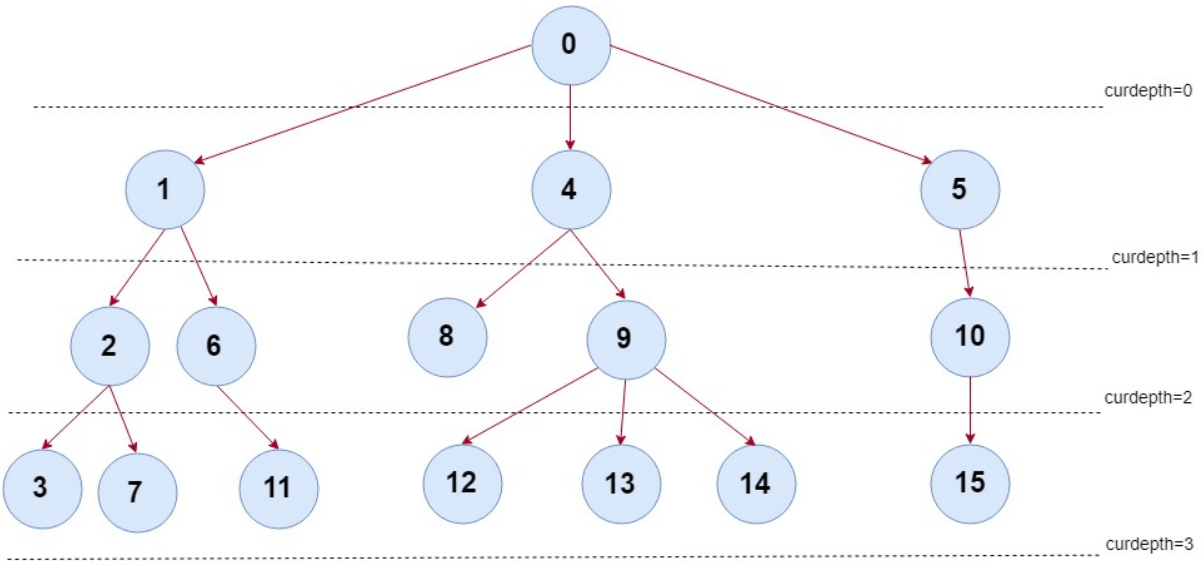
Επαλήθευση αποτελεσμάτων

Στην συνέχεια θα επιβεβαιώσουμε την ορθή λειτουργία του TiNA σε μια τυχαία εκτέλεση του προγράμματος. Για το παράδειγμα χρησιμοποιήθηκε τοπολογία κόμβων $4 * 4$.

```
0:0:10.202362049 DEBUG (5): NodeID= 5 : curdepth= 1 , parentID= 0
0:0:10.202362049 DEBUG (4): NodeID= 4 : curdepth= 1 , parentID= 0
0:0:10.202362049 DEBUG (1): NodeID= 1 : curdepth= 1 , parentID= 0
0:0:10.400741588 DEBUG (6): NodeID= 6 : curdepth= 2 , parentID= 1
0:0:10.400741588 DEBUG (2): NodeID= 2 : curdepth= 2 , parentID= 1
0:0:10.403610224 DEBUG (9): NodeID= 9 : curdepth= 2 , parentID= 4
0:0:10.403610224 DEBUG (8): NodeID= 8 : curdepth= 2 , parentID= 4
0:0:10.406631446 DEBUG (10): NodeID= 10 : curdepth= 2 , parentID= 5
0:0:10.598052995 DEBUG (7): NodeID= 7 : curdepth= 3 , parentID= 2
0:0:10.598052995 DEBUG (3): NodeID= 3 : curdepth= 3 , parentID= 2
0:0:10.600875854 DEBUG (11): NodeID= 11 : curdepth= 3 , parentID= 6
0:0:10.603225712 DEBUG (14): NodeID= 14 : curdepth= 3 , parentID= 9
0:0:10.603225712 DEBUG (12): NodeID= 12 : curdepth= 3 , parentID= 9
0:0:10.603225712 DEBUG (13): NodeID= 13 : curdepth= 3 , parentID= 9
0:0:10.605896001 DEBUG (15): NodeID= 15 : curdepth= 3 , parentID= 10
```

Σχήμα 5: Routing στο συγκεκριμένο Simulation

Με τα παραπάνω δεδομένα έχουμε πλήρη γνώση για το δέντρο που έχει σχηματιστεί. Παραθέτουμε τον σχεδιασμό αυτού του δέντρου στην παρακάτω εικόνα.



Σχήμα 6: Δέντρο Routing

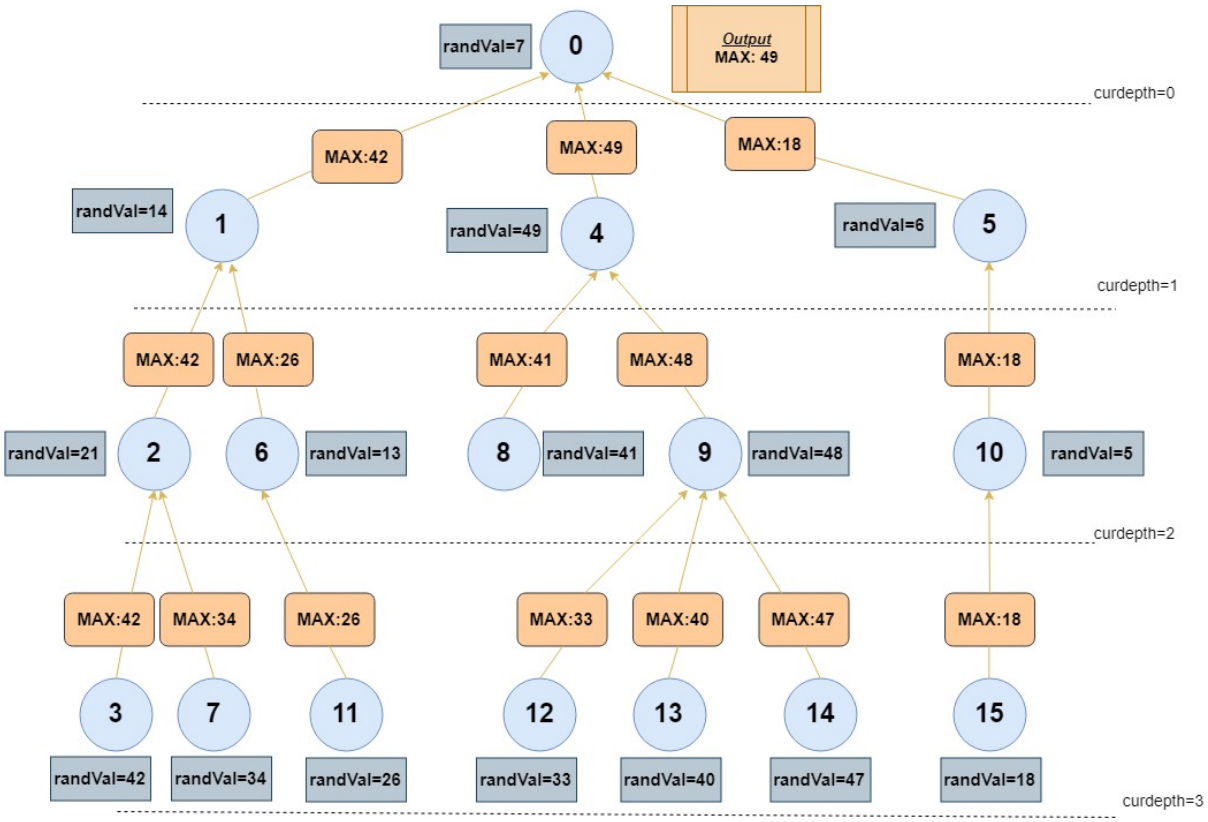
Στην πρώτη εποχή θα αποσταλούν όλα τα μηνύματα αφού δεν υπάρχουν συγκρούσεις, ενώ το TiNA δεν εφαρμόζεται αφού δεν υπάρχουν προηγούμενες τιμές προς σύγκριση.

```
0:1:11.728515635 DEBUG (3): Random value generated 42
0:1:11.728515635 DEBUG (3): Node: 3 , Parent: 2, max: 42, depth: 3
0:1:11.826171885 DEBUG (7): Random value generated 34
0:1:11.826171885 DEBUG (7): Node: 7 , Parent: 2, max: 34, depth: 3
0:1:11.923828136 DEBUG (11): Random value generated 26
0:1:11.923828136 DEBUG (11): Node: 11 , Parent: 6, max: 26, depth: 3
0:1:11.948242198 DEBUG (12): Random value generated 33
0:1:11.948242198 DEBUG (12): Node: 12 , Parent: 9, max: 33, depth: 3
0:1:11.972656261 DEBUG (13): Random value generated 40
0:1:11.972656261 DEBUG (13): Node: 13 , Parent: 9, max: 40, depth: 3
0:1:11.982421885 DEBUG (2): Random value generated 21
0:1:11.982421885 DEBUG (2): Node: 2 , Parent: 1, max: 42, depth: 2
0:1:11.997070323 DEBUG (14): Random value generated 47
0:1:11.997070323 DEBUG (14): Node: 14 , Parent: 9, max: 47, depth: 3
0:1:12.021484386 DEBUG (15): Random value generated 18
0:1:12.021484386 DEBUG (15): Node: 15 , Parent: 10, max: 18, depth: 3
0:1:12.080078135 DEBUG (6): Random value generated 13
0:1:12.080078135 DEBUG (6): Node: 6 , Parent: 1, max: 26, depth: 2
0:1:12.128906260 DEBUG (8): Random value generated 41
0:1:12.128906260 DEBUG (8): Node: 8 , Parent: 4, max: 41, depth: 2
0:1:12.153320323 DEBUG (9): Random value generated 48
0:1:12.153320323 DEBUG (9): Node: 9 , Parent: 4, max: 48, depth: 2
0:1:12.177734386 DEBUG (10): Random value generated 5
0:1:12.177734386 DEBUG (10): Node: 10 , Parent: 5, max: 18, depth: 2
0:1:12.236328135 DEBUG (1): Random value generated 14
0:1:12.236328135 DEBUG (1): Node: 1 , Parent: 0, max: 42, depth: 1
0:1:12.309570322 DEBUG (4): Random value generated 49
0:1:12.309570322 DEBUG (4): Node: 4 , Parent: 0, max: 49, depth: 1
0:1:12.333984385 DEBUG (5): Random value generated 6
0:1:12.333984385 DEBUG (5): Node: 5 , Parent: 0, max: 18, depth: 1
0:1:12.490234385 DEBUG (0): Random value generated 7
0:1:12.490234385 DEBUG (0):

##########################Epoch 1 completed######################
0:1:12.490234385 DEBUG (0): #### OUTPUT:
0:1:12.490234385 DEBUG (0): #### [MAX] = 49
```

Σχήμα 7: Τιμές πρώτης εποχής

Παρατηρούμε ότι κάθε κόμβος στέλνει στον πατέρα του την μέτρηση που έλαβε και οι τιμές συναθροίζονται καθώς ρέουν προς τη ρίζα. Το τελικό αποτέλεσμα που προέκυψε είναι 49.



Σχήμα 8: Ροή πληροφορίας κατά την πρώτη εποχή

Από την δεύτερη εποχή και μετά θα ξεκινήσουν οι συγκρίσεις με την εκάστοτε προηγούμενη εποχή. Σαν παράδειγμα θα παραθέσουμε την συνάθροιση που συνέβη κατά την δεύτερη εποχή της ίδιας εκτέλεσης. Η τιμή του tct καθορίζεται από τον χρήστη πριν την εκτέλεση και ισχύει για όλες τις εποχές. Εμείς θέσαμε αρχικά μια μεγάλη τιμή ώστε να παρατηρήσουμε πολλές περικοπές μηνυμάτων. Συγκεκριμένα, επιλέξαμε $tct = 0.8$, αυτό σημαίνει ότι μια τιμή θα σταλεί μόνο αν διαφέρει από την προηγούμενη που στάλθηκε κατά 80%. Τα αποτελέσματα των μετρήσεων που προέκυψαν και των τιμών που εν τέλει στάλθηκαν κατά την δεύτερη αποχή παραθέτονται παρακάτω.

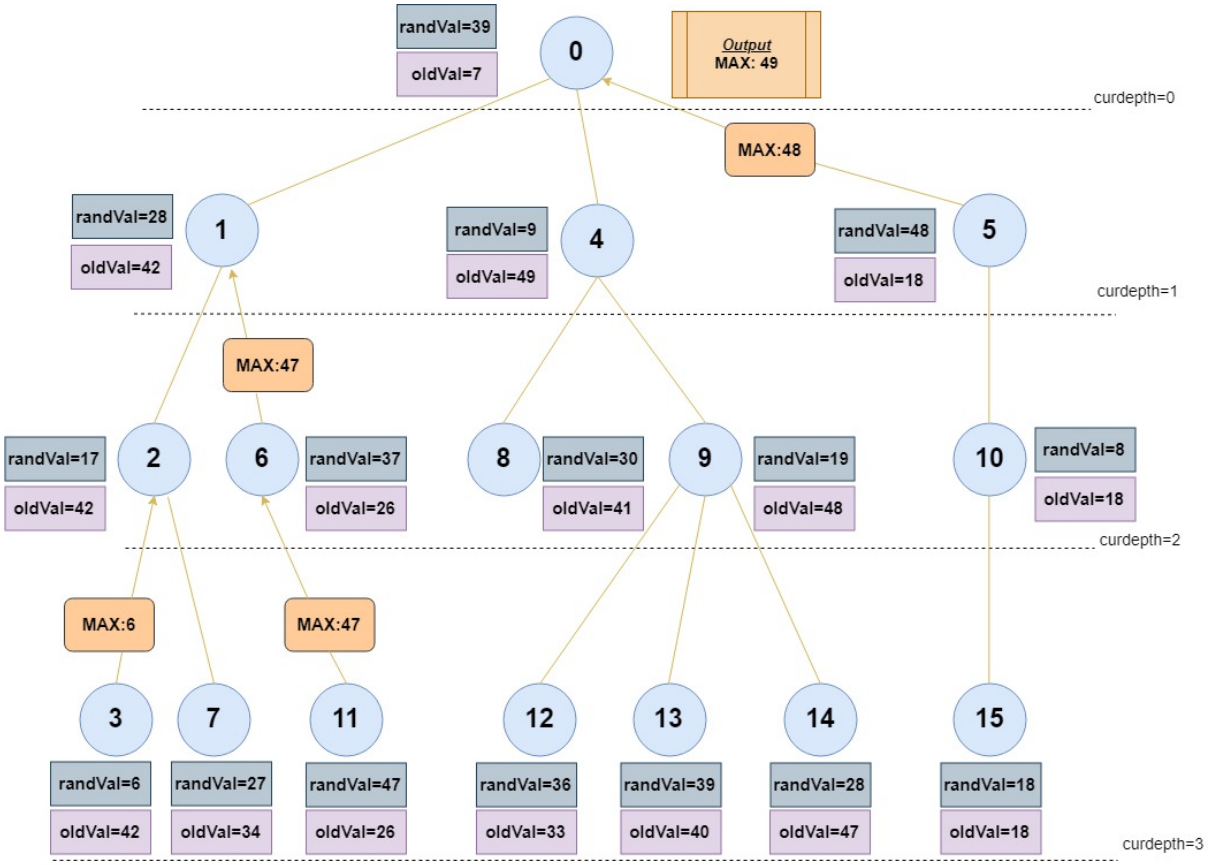
```
0:2:10.322265635 DEBUG (3): Random value generated 6
0:2:10.322265635 DEBUG (3): Node: 3 , Parent: 2, max: 6, depth: 3
0:2:10.419921885 DEBUG (7): Random value generated 27
0:2:10.419921885 DEBUG (7): Don't send message with new value 27 and old value 34
0:2:10.517578136 DEBUG (11): Random value generated 47
0:2:10.517578136 DEBUG (11): Node: 11 , Parent: 6, max: 47, depth: 3
0:2:10.541992198 DEBUG (12): Random value generated 36
0:2:10.541992198 DEBUG (12): Don't send message with new value 36 and old value 33
0:2:10.566406261 DEBUG (13): Random value generated 39
0:2:10.566406261 DEBUG (13): Don't send message with new value 39 and old value 40
0:2:10.576171885 DEBUG (2): Random value generated 17
0:2:10.576171885 DEBUG (2): Don't send message with new value 34 and old value 42
0:2:10.590820323 DEBUG (14): Random value generated 28
0:2:10.590820323 DEBUG (14): Don't send message with new value 28 and old value 47
0:2:10.615234386 DEBUG (15): Random value generated 18
0:2:10.615234386 DEBUG (15): Don't send message with new value 18 and old value 18
0:2:10.673828135 DEBUG (6): Random value generated 37
0:2:10.673828135 DEBUG (6): Node: 6 , Parent: 1, max: 47, depth: 2
0:2:10.722656260 DEBUG (8): Random value generated 30
0:2:10.722656260 DEBUG (8): Don't send message with new value 30 and old value 41
0:2:10.747070323 DEBUG (9): Random value generated 19
0:2:10.747070323 DEBUG (9): Don't send message with new value 47 and old value 48
0:2:10.771484386 DEBUG (10): Random value generated 8
0:2:10.771484386 DEBUG (10): Don't send message with new value 18 and old value 18
0:2:10.830078135 DEBUG (1): Random value generated 28
0:2:10.830078135 DEBUG (1): Don't send message with new value 47 and old value 42
0:2:10.903320322 DEBUG (4): Random value generated 9
0:2:10.903320322 DEBUG (4): Don't send message with new value 48 and old value 49
0:2:10.927734385 DEBUG (5): Random value generated 48
0:2:10.927734385 DEBUG (5): Node: 5 , Parent: 0, max: 48, depth: 1
0:2:11.083984385 DEBUG (0): Random value generated 39
0:2:11.083984385 DEBUG (0):

##########################Epoch 2 completed######################
0:2:11.083984385 DEBUG (0): #### OUTPUT:
0:2:11.083984385 DEBUG (0): #### [MAX] = 49
```

Σχήμα 9: Τιμές δεύτερης εποχής

Με μία πρώτη ματιά βλέπουμε όπως και αναμέναμε, ότι δεν στέλνονται πολλές συναθροίσεις καθώς δεν διαφέρουν από τις προηγούμενες κατά ποσοστό tct.

Σημειώνουμε ότι $randVal$ είναι η μέτρηση που έλαβε κάθε κόμβος στην τρέχουσα εποχή. Η τιμή $oldVal$ που αναγράφεται είναι η τιμή που έστειλε ο κόμβος στην προηγούμενη εποχή, όχι δηλαδή η μέτρηση που είχε αλλά το αποτέλεσμα της συνάθροισης από τις τιμές των παιδιών του και την δικιά του. Συνεπώς, οι παράμετροι που συμμετέχουν στον έλεγχο για αποστολή νέας τιμής είναι το $oldVal$ και το αποτέλεσμα της νέας συνάθροισης.



Σχήμα 10: Ροή πληροφορίας δεύτερη εποχή

Παρατηρούμε ότι ο κόμβος 3 αποστέλλει νέα τιμή. Αυτό συμβαίνει γιατί επιβεβαιώνεται η σχέση του TiNA για τις παραμέτρους της.

$$Node\ 3 : \frac{|Vnew-Vold|}{|Vold|} = \frac{|6-42|}{|42|} = 0.85 > tct = 0.8$$

Ομοίως και ο κόμβος 11

$$Node\ 11 : \frac{|Vnew-Vold|}{|Vold|} = \frac{|47-26|}{|26|} = 0.807 > tct = 0.8$$

Στο κόμβο 2 αυτό που συμβαίνει είναι ότι θα δεχτεί από τον 3 την νέα τιμή 6. Από τον κόμβο 7 δεν θα λάβει κάτι άρα θα προχωρήσει σε συνάθροιση θεωρώντας την παλιά του τιμή, το 34. Το αποτέλεσμα της συνάθροισης στον κόμβο 2 θα είναι $max(17, 6, 34) = 34$. Η προηγούμενη τιμή όμως που είχε στείλει είναι $oldVal = 42$, έτσι προκύπτει το εξής,

$$Node\ 2 : \frac{|Vnew-Vold|}{|Vold|} = \frac{|34-42|}{|42|} = 0.19 < tct = 0.8$$

Άρα ο κόμβος 2 δεν θα στείλει καινούρια μέτρηση και ο πατέρας του, ο κόμβος 1, θα προχωρήσει σε συνάθροιση θεωρώντας την παλιά του τιμή, το 42.Ο κόμβος 6 θα λάβει την νέα τιμή από τον κόμβο 11 και το αποτέλεσμα προς αποστολή θα είναι $max(47, 37) = 47$. Η προηγούμενη τιμή που είχε στείλει ήταν το 26. Έτσι από την σχέση του TiNA έχουμε

$$Node\ 6 : \frac{|Vnew-Vold|}{|Vold|} = \frac{|47-26|}{|26|} = 0.807 > tct = 0.8$$

Συνεπώς, ο κόμβος 6 πρέπει να στείλει τη νέα τιμή. Στο υπό-δέντρο με ρίζα τον κόμβο 4 δεν υπήρξε κάποια περίπτωση μετάδοσης νέας τιμής και έτσι ο κόμβος 4 κάνει συνάθροιση με τις προηγούμενες τιμές που είχε λάβει από τα παιδιά του και την νέα του μέτρηση, $max(41, 48, 9) = 48$. Η τιμή που είχε στείλει στην προηγούμενη εποχή ήταν 49. Επομένως, εφόσον η διαφορά είναι πολύ μικρή δεν θα στείλει ξανά.

Τέλος, στον κόμβο 0 που είναι η ρίζα και όπου γίνεται η τελική συνάθροιση θα σταλεί νέα τιμή μόνο από τον κόμβο 5. Ο κόμβος 0 θα υπολογίσει το τελικό αποτέλεσμα με βάση τις προηγούμενες τιμές από τους κόμβους 1 και 4, την νέα τιμή που έλαβε από τον κόμβο 5 και την δική του μέτρηση. Το αποτέλεσμα που θα προκύψει είναι $max(42, 49, 48, 39) = 49$. Βλέπουμε λοιπόν ότι το τελικό αποτέλεσμα που θα εμφανιστεί είναι 49 παρόλο που η μέγιστη μέτρηση που παρατηρήσαμε σε αυτή την εποχή ήταν το 48. Υπάρχει μικρή αλλοίωση του τελικού αποτελέσματος, όμως η εξοικονόμηση ενέργειας που έγινε κατά την συνάθροιση είναι πολύ σημαντική. Συγκεκριμένα, στάλθηκαν 4 μηνύματα έναντι 15 που θα αποστέλλονταν χωρίς την εφαρμογή του TiNA. Γενικά, σε πολλές περιπτώσεις αυτή η μικρή απόκλιση στο αποτέλεσμα είναι μικρό τίμημα συγκριτικά με την επιθυμητή εξοικονόμηση ενέργειας που επιτυγχάνεται.

Καταμερισμός Εργασιών

Στο μεγαλύτερο κομμάτι της άσκησης υπήρξε δίκαιος καταμερισμός εργασιών και συνεργασία στα στάδια της λύσης και από τα δύο μέλη της ομάδας.Η τεχνική της κωδικοποίησης/αποκωδικοποίησης κατά την επιλογή συναθροιστικών συναρτήσεων υλοποιήθηκε από τον φοιτητή Στέφανο Καλογεράκη.

Παράρτημα

Επιλογή ερωτήματος και συναθροιστικών συναρτήσεων

Κομμάτι του προγράμματος στην φάση 2 ήταν και η τυχαία επιλογή ερωτήματος και συναθροιστικών συναρτήσεων για την αποστολή των κατάλληλων δεδομένων. Οι επιλογές αυτές γινόταν πριν στείλει ο κόμβος 0 πληροφορία(πριν το Routing). Κάθε κόμβος όμως πρέπει να είναι ενήμερος σχετικά με αυτές τις τιμές κάθε στιγμή για να γίνεται η επιλογή των κατάλληλων δομών όπως αναλύθηκε παραπάνω. Για αυτό τον λόγο για υλοποίηση μας στάλθηκε ένα μήνυμα επιπλέον κατά το Routing των κόμβων που περιλάμβανε αυτές τις πληροφορίες. Αυτό το μήνυμα ήταν ένας μη προσημασμένος ακέραιος αριθμός.

Στο πρακτικό κομμάτι υλοποίησης του κώδικα, δημιουργήσαμε ένα Aggregation Map στο οποίο είχε ανατεθεί μια τιμή από το 1 έως το 6 σε όλες τις πιθανές συναθροιστικές συναρτήσεις όπως φαίνεται παρακάτω

$$AggregationMap = \begin{cases} MIN & \text{αν } 1; \\ MAX & \text{αν } 2; \\ COUNT & \text{αν } 3; \\ AVG & \text{αν } 4; \\ SUM & \text{αν } 5; \\ VARIANCE & \text{αν } 6. \end{cases}$$

Η πληροφορία που αποστέλλεται, βασίζεται σε μεγάλο βαθμό στην τεχνική της κωδικοποίησης/αποκωδικοποίησης. Για την μέθοδο που γίνεται η κωδικοποίηση λάβαμε υπόψιν πληροφορίες σχετικά με τον αριθμό των πεδίων που αποστέλλονται καθώς και τις συναθροιστικές συναρτήσεις βάσει του Aggregation Map. Αρχικά το μήκος του ακεραίου είναι σε κάθε περίπτωση ίδιο με τον αριθμό των πεδίων που αξιοποιούνται κατά την αποστολή πληροφορίας. Το επόμενο βήμα είναι η ανάθεση του αριθμού της κάθε επιλεγμένης συνάρτησης στην αρχή και στο τέλος του αριθμού που αποστέλλεται. Στην περίπτωση αποστολής ενός πεδίου, αναθέσαμε τις τιμές από 1-4 για το ερώτημα 2.1 και τις τιμές 6-9 για το ερώτημα 2.2. Στην περίπτωση πάλι, που για μία συναθροιστική συνάρτηση απαιτούνται παραπάνω από ένα πεδία, η τιμή της επιλεγμένης συνάρτησης τοποθετείται στην αρχή και στο τέλος τοποθετείται το 0.

Παράδειγμα: Η τιμή 6002 θα σταλεί στην περίπτωση που επιλεγούν οι $VARIANCE$ και MAX καθώς απαιτούν τέσσερα πεδία για την αποστολή όλης της πληροφορίας τους. Αντίστοιχα η τιμή 40 θα σταλεί στην περίπτωση της AVG συνάρτησης καθώς απαιτούνται δυο πεδία για την αποστολή της πληροφορίας και είναι η μόνη συνάρτηση που μας ενδιαφέρει.