Αναφορά πρώτης εργασίας
ΠΛΗ511

Αικατερίνη Τσιμπιρδώνη 2018030013
Φώτης Κοτσέλης:2018030155

Βοηθητικό πρόγραμμα:

Αρχικά μας ζητήθηκε να κατασκευάσουμε ένα πρόγραμμα που θα δημιουργεί αυτόματα τα αρχεία τροπολογίας για να λειτουργήσει το πρόγραμμά μας. Για να κατασκευάσουμε αυτό το αρχείο χρησιμοποιήσαμε τη γλώσσα JAVA. Το πρόγραμμα κατασκευάζει αρχικά έναν πίνακα μεγέθους DxD όπου D είναι η παράμετρος που δίνουμε εμείς και καθορίζει το μέγεθος του. Η γραμμή για να μπει το στοιχείο J καθορίζεται από τον τύπο J/D ενώ η στήλη από τον τύπο J%D ,οπότε παίρνουμε ένα αποτέλεσμα ενός πίνακα που έχει την παρακάτω μορφή:

$$B = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

Αυτό είναι ένα παράδειγμα ενός πίνακα πού μπορούμε να φτιάξουμε με αυτό το πρόγραμμα σε αυτή την περίπτωση έχουμε έναν 3 X 3 πίνακα αλλά μπορούμε να φτιάξουμε ότι πίνακα θέλημα ανάλογα με τον D πού θα βάλουμε, το οποίο το έχω περιορίσει να φτάνει ως 8 οπότε μπορεί να φτιάχνει μέγιστα 64 κόμβους.

Αμέσως μετά το πρόγραμμα βρίσκει τους γείτονες με βάση την απόσταση και την εμβέλεια την οποία θα δώσουμε εμείς. Δηλαδή παίρνει κάθε στοιχείο και υπολογίζει την απόσταση από τα υπόλοιπα στοιχεία του πίνακα όταν η απόσταση αυτή είναι μικρότερη της εμβέλειας τότε αποθηκεύει και τα δύο στοιχεία αυτά μέσα στο αρχείο topology.txt συν ενα τριτο στοιχείο -50.0. Αυτό γίνεται μέσα στη συνάρτηση findneigh η οποία δουλεύει αναδρομικά και το κάνει για όλα τα στοιχεία του πίνακα. Οπότε μέσα στο αρχείο έχουμε κάτι του τύπου 0 1 -50.0 αν ας πούμε η απόσταση του μηδέν κόμβου από τον ένα είναι μικρότερη της εμβέλειας που έχουμε ορίσει.
ο τύπος της απόστασης που χρησιμοποιούμε είναι:

distance=$\sqrt{(n - line)^2 + (i + column)^2}$.

Πρώτο μέρος της εργασίας:

Ο βασικός κώδικας βρίσκεται μέσα στο αρχείο SRTreeC.nc ενώ χρησιμοποιούμε και το αρχείο SRTreeAppC.nc για να συνδέσουμε τα διάφορα modules μεταξύ του δηλαδή τους timers, τα packets κτλ.

Αφαίρεση κωδικα:

Στο αρχείο SRTreeC.nc αρχικά υπήρχε πολύ κώδικα που δεν χρησιμεύει στην εργασία οπότε αφαιρέθηκαν τα περιττά κομμάτια. Πρώτον αφαιρέθηκε όλος ο κώδικας που αφορά την σειριακή επικοινωνία δηλαδή όλα εκείνα που είχαν τα labels SERIAL_EN και PRINTFDBG_MODE. Το PRINTFDBG_MODE αφαιρέθηκε διότι υπάρχει η dbg.Επίσης αφαιρέθηκαν τα τμήματα που αφορούν τη διαχείριση ενός χαμένου task δηλαδή set Lost Task και set Send Busy καθώς και ο ίδιος ο μετρητής. Τέλος αφαιρέθηκαν όλα τα τμήματα που αφορούν τα leds.

Αρχικά ασχοληθήκαμε με το routing ώστε κάθε Κόμβος να συνδεθεί με έναν γονέα το οποίο στον σχεδιασμό TINA δεν είναι πολύ περίπλοκο αφού όποιος Κόμβος μπει πρώτος θα επιλέξει πρώτος τον γονέα του.Δηλαδη οταν ενας κομβος λαμβάνει μύνημα routing από κάποιον, τότε απευθείας τον θέτει ως πατέρα του. Το rooting στην άσκηση γίνεται μόνο στην πρώτη εποχή και το ξεκινάει ο Κόμβος 0 με το routingMsgTimer.fired().
Αξίζει να σημειωθεί οτι σβήστηκαν κομμάτια του Κώδικα που αφορά το προορισμό και το μέγεθος του μηνύματος(routingmsg) και φτιάχτηκαν νέα εφόσον υπάρχουν δύο εναλλακτικοί τρόποι εκτέλεσης(TINA ,TINA with two arguments), αυτό αφορά τις συναθροιστικές συναρτήσεις όταν έχουμε Max and count τότε έχουμε δύο πεδία ένα για το Max και ένα για το count ενώ όταν έχουμε ή Max ή Count έχουμε ένα πεδίο.Επίσης το TINA έχει ένα επιπλέον πεδίο το tct το οποίο πρέπει να συμπεριληφθεί μέσα στο μήνυμα.Όλα τα παραπάνω λήφθηκαν υπόψη και φτιάξαμε τα μηνύματα κατάλληλα ώστε να μη στέλνετε περιττή πληροφορία ή λιγότερη. Καθως σβήστηκαν και τα κομματια του NotifyParent διοτι δεν ειδοποιεί τον γονέα ότι τον επέλεξε. Τέλος αντικαταστάθηκαν όλα τα κομμάτια του notify με άλλες ονομασίες επειδή αυτή η έννοια δεν υπάρχει στην υλοποίηση.

Επεξήγηση:

Οπότε στην αρχή της πρώτης εποχής έχουμε δημιουργήσει ένα δέντρο το οποίο μας δείχνει τη σύνδεση μεταξύ των κόμβων. Στη συνέχεια κάθε κόμβο λαμβάνει μία τυχαία μέτρηση με εύρος από 0 έως 80 ελέγχει να δει αν ξεπερνάει το tct. Έπειτα τις συναθροίζει με αυτές των παιδιών τους και τις προωθεί στο γονέα του.Στο τέλος ο Κόμβος 0 παίρνει το τελικό αποτέλεσμα από όλες τις συναθροίσεις και το τυπώνει .Αυτό συμβαίνει σε κάθε εποχή δηλαδή κάθε 30 δευτερόλεπτα και όλο το πρόγραμμα διαρκεί 40 γύρους εφόσον τελειώνεις τα 1200 δευτερόλεπτα (1200/30=40) .Σε κάθε νέα εποχή παίρνουμε νέες μετρήσεις πού όμως δεν πρέπει να απέχουν από το 10% των προηγούμενων μετρήσεων στις προηγούμενες εποχές.

TIMERS:

Η συνάθροιση αυτή που γίνεται μεταξύ του γονέα και του παιδιού χρονικά μας προϋποθέτει ότι πρέπει να στέλνουν πρώτα τα παιδιά και στην συνέχεια οι γονείς.Για αυτό δημιουργήσαμε ένα νέο timer τον (SendMsgTimer) το όποιο για να στείλει το μήνυμα στην κατάλληλη στιγμή θα πρέπει να ξέρει το current depth.Για να το γνωρίζει αυτό όμως πρέπει να έχει τελειώσει η διαδικασία του routing για αυτό και όταν καλούμε πρώτη φορά τον μετρητή αυτον τον καλούμε με startOneShot(5000). Και αυτό μας λέει ότι θα ξεκινήσει να στέλνει τα μηνύματα μετά το πέρας το routing του δέντρου. Εμείς βάλαμε ότι το routing θα διαρκέσει 5 second. Για αυτό και βάζουμε ότι θα ξεκινήσει μετά από 5.000ms. Οπότε μόλις τελειώσουν τα πέντε δευτερόλεπτα ο κάθε κόμβος ορίζει το μετρητή του και για το πότε θα στέλνει δεδομένα από δω και στο εξής.

Η διαδικασία αυτή γίνεται πάλι με τον ίδιον timer SendMsgTimer με το startPeriodicAt. Αυτή η εντολή είναι σημαντικής σημασίας για το πρόγραμμα γιατί δίνει την περιοδικότητα ώστε να αλλάζουμε εποχή κάθε 30 δευτερόλεπτα και αυτό δηλώνεται στο δεύτερο όρισμα της συνάρτησης αυτής που το έχουμε ορίσει ως TIMER_PERIOD_MILI=30000.

Όσο αναφορά το πρώτο όρισμα το οποίο καθορίζει ποιος θα είναι ο ακριβής χρόνος που κάθε κόμβος θα στέλνει μηνύματα.

Αρχικά μέσα στο .h αρχείο που μας δίνετε έχουμε έναν χρόνο που ονομάζεται TIMER_FAST_PERIOD πού μας τον δίνει 200 ms και είναι ο χρόνος ώστε να μεταδοθεί μία πληροφορία στο παραπάνω επίπεδο.

Επιπλέον γνωρίζουμε ότι για να μεταδοθεί μία πληροφορία στο σωστό χρόνο δηλαδή η πληροφορία των παιδιών να μεταδοθεί πρώτη και ύστερα των γονέων πρέπει να γνωρίζουμε το βάθος. Οπότε καταλαβαίνουμε ότι για να μεταδοθεί πρώτα το πιο βαθύ επίπεδο, θα πρέπει να συμπεριλαμβάνεται μέσα στην έκφραση το curdepth και να το πολλαπλασιάσουμε με το χρόνο που χρειάζεται για να μεταδοθεί η πληροφορία από τον κόμβο στο επόμενο επίπεδο(curdepth*TIMER_FAST_PERIOD).

Επίσης έχουμε έναν χρόνο που χρησιμοποιεί οι κόμβοι ώστε να κάνει boot την πληροφορία οπότε έχουμε -(curdepth*TIMER_FAST_PERIOD)-10000 (όπου 10s είναι αυτός ο χρόνος).

Το μείον από μπροστά από την έκφραση (curdepth*TIMER_FAST_PERIOD) μπαίνει διότι θέλουμε αρνητικό αριθμό ώστε να είναι TIMER_PERIOD_MILI-(curdepth*TIMER_FAST_PERIOD)-10000.

Άρα κανονικά θα έπρεπε SendMsgTimer.startPeriodicAt(-(10000)-(curdepth*TIMER_FAST_PERIOD),TIMER_PERIOD_MILI).Όμως παρατηρούμε ότι έτσι χάνουμε μία περίοδο διότι οταν το curdepth=0 τότε η ρίζα δεν προλαβαίνει να γράψει ακριβώς στα 1200 οποτε για αυτό βάζουμε (curdepth+1).

Οπότε έχουμε SendMsgTimer.startPeriodicAt(-(10000)-((curdepth+1)*TIMER_FAST_PERIOD),TIMER_PERIOD_MILI).

Τέλος με αυτό τον τρόπο οι κόμβοι πού βρίσκονται στο ίδιο βάθος θα στέλνουν μία συγκεκριμένη χρονική στιγμή όμως μπορεί σε ένα βάθος να υπάρχουν παραπάνω κόμβοι από ένα οπότε για να μη δημιουργηθούν προβλήματα μας μένει να στέλνουν κόμβοι με το ίδιο βάθος και εκείνοι σε διαφορετική στιγμή. Για αυτό πρόσθεσα το ID(TOS_NODE_ID) του κάθε κόμβου πολλαπλασιαζόμενο με 2 ώστε να ξεχωρίσει κάθε χρονική στιγμή αποστόλης του κάθε κόμβου .Το 2 το επέλεξαμε μετά από δοκιμές σε μεγάλους πίνακες με πολλές συγκρούσεις και κατέληξα ότι είναι το καλύτερο ωστε να μην ξεπεράσει το TIMER_FAST_PERIOD και να μην υπαρχουν πολλες συγκρουσεις.

πχ. Εάν έβαζα 3 αντι για 2 σε ένα πίνακα D=7 με εμβέλεια 1.5 τότε χάναμε κάποιους γύρους.

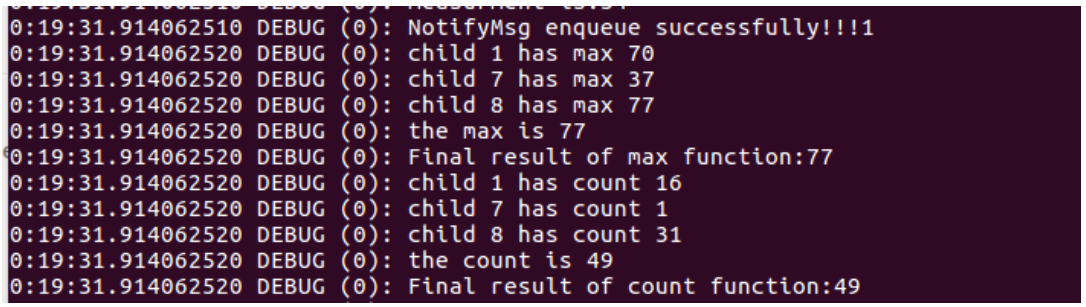
```
0:19:31.745162950 DEBUG (7): Package sent True
0:19:31.914062510 DEBUG (0): NODE_ID=0, curdepth=0
0:19:31.914062510 DEBUG (0): measurment is:34
0:19:31.914062510 DEBUG (0): NotifyMsg enqueue successfully!!!1
0:19:31.914062520 DEBUG (0): child 1 has max 9
0:19:31.914062520 DEBUG (0): child 8 has max 77
0:19:31.914062520 DEBUG (0): child 7 has max 37
0:19:31.914062520 DEBUG (0): the max is 77
0:19:31.914062520 DEBUG (0): Final result of max function:77
0:19:31.914062520 DEBUG (0): child 1 has count 1
0:19:31.914062520 DEBUG (0): child 8 has count 45
0:19:31.914062520 DEBUG (0): child 7 has count 1
0:19:31.914062520 DEBUG (0): the count is 48
0:19:31.914062520 DEBUG (0): Final result of count function:48
0:19:32.109375010 DEBUG (0): ############ROUND  41 ########
Node 0 connected with node 1 True True
```

Το count είναι 48 αντί για 49 που θα έπρεπε που σημαίνει ότι υπάρχει κάποια σύγκρουση και κάποιος κόμβος δεν έστειλε ενώ όταν έβαλα το 2 όλοι οι κόμβοι στείλαν κανονικά.

Το ιδιο και αν βαξαμε 1:

```
0:19:31.914062510 DEBUG (0): NODE_ID=0, curdepth=0
0:19:31.914062510 DEBUG (0): measurment is:34
0:19:31.914062510 DEBUG (0): NotifyMsg enqueue successfully!!!1
0:19:31.914062520 DEBUG (0): child 1 has max 9
0:19:31.914062520 DEBUG (0): child 8 has max 72
0:19:31.914062520 DEBUG (0): child 7 has max 46
0:19:31.914062520 DEBUG (0): the max is 72
0:19:31.914062520 DEBUG (0): Final result of max function:72
0:19:31.914062520 DEBUG (0): child 1 has count 1
0:19:31.914062520 DEBUG (0): child 8 has count 43
0:19:31.914062520 DEBUG (0): child 7 has count 2
0:19:31.914062520 DEBUG (0): the count is 47
0:19:31.914062520 DEBUG (0): Final result of count function:47
```

Το δύο δίνει το καλύτερο αποτέλεσμα:



Οπότε τελικά έχουμε κάτι της μορφής SendMsgTimer.startPeriodicAt(((-(10000)-((curdepth+1)*TIMER_FAST_PERIOD))+(TOS_NODE_ID*2)),TIMER_PERIOD_MILI).
Οπότε με αυτόν τον τρόπο εξασφαλίσαμε πότε κάθε Κόμβος Θα στέλνει μηνύματα σε κάθε εποχή. Επίσης μέσα SendMsgTimer γίνονται και Οι μετρήσεις με την συνάρτηση rand. Το **Onemessage** είναι μία δομή φτιάχτηκε για να παίρνει τη μέτρηση και αποτελείται από μόνο ένα στοιχείο unint8_t αφού έτσι κι αλλιώς δεν χρειαζόμαστε παραπάνω bit για μία μέτρηση από 0 έως 80.

Έχουμε έναν ακόμη timer τον routingMsgTimer.fired() ο οποίος με ένα random αποφασίζει αν θα έχουμε μία ή δύο συναθροιστικες συναρτήσεις. Ακομη υπολογιζει ποιο θα είναι το tct της συναρτήσης με random.
Αν επιλέξει μία συναθροιστικη συναρτήση φτιάξαμε μία δομή που λέγεται **routingwith1func,** η οποία έχει τρία παιδιά ένα για την αποθήκευση του depth , του tct και τέλος του αριθμού της τυχαίας επιλογής της συναθροιστικης συνάρτησης 1ή2. Την περίπτωση που έχουμε δύο συναθροιστικες συναρτήσεις έχουμε το πεδίο **routingwith2func** που κάνει ακριβώς ότι το από πάνω αλλά Απλά αποθηκεύει και τον κωδικό της δεύτερης συναθροιστικης συνάρτησης 1 και 2. Και έπειτα δρομολογείται η διαδικασία sendrouting task()
Η sendroutingtask() Στέλνει το μήνυμα ανάλογα με το αν το σύστημα με επέλεξε μία ή δύο συναντήσεις διότι τότε αλλάζει το size του μηνύματος για μια ειναι **routingwith1func** και για δυο **routingwith2func** .
Η receiveroutingtask() ξεκιναει όταν λαμβάνει το routingmsg αν δεν έχει πατέρα τότε θέτει ως πατέρας τον αποστολέα και στη συνέχεια παίρνει την υπόλοιπη πληροφορία. Την υπόλοιπη πληροφορία την παίρνει χρησιμοποιώντας τα routingwith2func για όταν έχουμε δύο συναρτήσεις και το routingwith1func αν έχουμε μία. Παίρνει την πληροφορία εφόσον εμείς την έχουμε αποθηκεύσει αρχικά εκεί όταν έτρεξε ο timer.

Τώρα θα ασχοληθούμε με την αποστολή, λήψη και επεξεργασία του μηνύματος για την αποστολή αυτή μετάρυθμίστηκαν όλα τα παιδιά που προϋπήρχαν notify και τα κάναμε message

δηλαδή όλα τα notifyAMpacket,NotifyAMsend κτλ.Εγιναν messageAMpacket,messageAMsend κτλ.
Αφού τρέξουμε λοιπόν το μετρητή (SendMsgTimer) που φτιάξαμε πάνω με τον τρόπο που εξηγήσαμε παραπάνω κάθε κόμβος έχει μία μέτρηση από 0 έως 80 στην πρώτη εποχή και από 0 έως 80 στις επόμενες εποχές με την προϋπόθεση ότι η νέα μέτρηση δεν θα ξεπερνάει το 10% της παλιάς. Έτσι στη συνέχεια δρομολογείται η διαδικασία sendmesstask().
Μέσα στο task αρχικά λαμβάνουμε με ένα dequeue μέτρηση του κόμβου και το κανει και στη συνέχεια ξεκινάει ένας έλεγχος για το αν βρισκόμαστε σε TINA ή σε TINA with two arguments.
Οπότε φτιάξαμε μια νέες δομη και Twomess.
Ξανά χρησιμοποιούμε την **Onemessage** για να παίρνει τη μέτρηση όταν έχουμε απλό TINA δηλαδή Ή το Max ή το count (οι συναθροιστικές συναρτήσεις που επιλέγεται τυχαία).
Το **Twomess** φτιάχτηκε για να παίρνει την μέτρηση όταν έχουμε TINA with two arguments. Και έχει δύο πεδία στα οποία παίρνει τη μέτρηση και του max και του count αλλά και έναν πίνακα ο οποίος αποθηκεύει αν ήρθε πρώτα το Max ή αν ήρθε πρώτα το count.
Μέσα στο task αυτό καλείται επίσης και η calculateF(με ορισμένα τη μέτρηση που είχαμε πάρει πριν και την είχα αποθηκεύσει στη δομή **Onemessage,** Και τον κωδικό το max ή του count) συνάρτηση που φτιάξαμε για να υπολογίζει το max και το count, που είναι στην συναθροιστικές συναρτήσεις οπότε παίρνει αποτελέσματα των παιδιών και και το δικό του και βγάζει ένα αποτέλεσμα για να το στείλει στον πατέρα του. Παίρνει λοιπόν το αποτέλεσμα της συνάρτησης και ελέγχει αν η καινούργια μέτρηση ξεπερνάει την παλιά σύμφωνα με το tct που έχει υπολογιστει μία φορά στην αρχή αν το ξεπερνάει τότε η η νέα μέτρηση στέλνετε ενώ αν όχι κρατάμε την παλιά.
Το ίδιο ισχύει και για τις δύο περιπτώσεις απλά όταν έχει TINA with two arguments κάνει ταυτόχρονο υπολογισμό του max και του count και χρησιμοποιούμε τη δεύτερη δομή Twomess για στείλουμε και τα δύο μηνύματα στο πατερα. Και όταν φτάσει στο τελευταίο επίπεδο βγάζει και το τελικό αποτέλεσμα.

Εφόσον οι μετρήσεις παίρνουμε έχουν Range από 0 έως 80 και το count δεν θα ξεπεράσει ποτέ το 255 μπορούμε να χρησιμοποιήσουμε μόνο uint8_t σε αυτές τις δομές.
Στην ουσία μετά το πέρας αυτό το task έχουμε στείλει το μήνυμα στον πατέρα του κάθε κόμβου και Όταν βρισκόμαστε στη ρίζα τυπώνουμε απλά το τελικό αποτέλεσμα.

Τέλος έχουμε το receivemesstask που πάλι ελέγχει αν έχουμε μία ή δυο συναρτησεις και με τη βοήθεια των δομων **Onemessage ,Twomess** που φτιάξαμε πιο πάνω λαμβάνει την πληροφορία των παιδιων του.

Αυτές οι δομές(**Twomess ,Onemessage ,routingwith1func ,routingwith2func**) που κατασκευάσαμε μας βοηθάει ώστε να στείλουμε σωστά την πληροφορία χωρίς να υπάρχουν περιττά κομμάτια.

Παραδειγμα 1:

Αρχικά το πρόγραμμα εμφανίζει το τη tct και αν θα δουλέψουμε με μία ή με δύο συναρτήσεις και ποια είναι αυτή:

```
0:0:10.195312510 DEBUG (0): RoutingMsgTimer fired!0:0:10.195312510 DEBUG (0): we have max&count
0:0:10.195312510 DEBUG (0): tct is:5
0:0:10.195312510 DEBUG (0): Sending RoutingMsg...
0:0:10.195312510 DEBUG (0): SendTask() posted!!
0:0:10.195312510 DEBUG (0): RoutingMsg enqueued successfully in SendingQueue!!!
0:0:10.195312520 DEBUG (0): sendRoutingTask(): Send returned success!!!
0:0:10.203964202 DEBUG (4): ### RoutingReceive.receive() start #####
0:0:10.203964202 DEBUG (4): ### RoutingReceive.receive() end #####
0:0:10.203964202 DEBUG (1): ### RoutingReceive.receive() start #####
0:0:10.203964202 DEBUG (1): ### RoutingReceive.receive() end #####
0:0:10.203964202 DEBUG (3): ### RoutingReceive.receive() start #####
0:0:10.203964202 DEBUG (3): ### RoutingReceive.receive() end #####
0:0:10.203964212 DEBUG (4): ReceiveRoutingTask(): len=4
```

Σε αυτό το παράδειγμα έχουμε Max and count και tct=5%

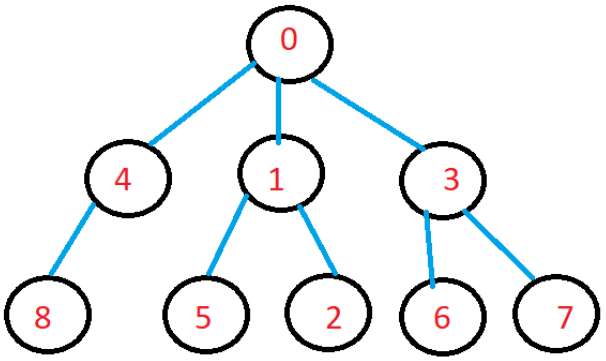
```
0:0:10.203964212 DEBUG (4): ReceiveRoutingTask(): len=4
0:0:10.203964212 DEBUG (4): Now the node 4 have a perent with parentID 0 and currentdepth 10:0:10.203964212 DEBUG (1): ReceiveRouting
Task(): len=4
0:0:10.203964212 DEBUG (1): Now the node 1 have a perent with parentID 0 and currentdepth 10:0:10.203964212 DEBUG (3): ReceiveRouting
Task(): len=4
0:0:10.203964212 DEBUG (3): Now the node 3 have a perent with parentID 0 and currentdepth 10:0:10.204132048 DEBUG (0): Package sent T
rue
0:0:10.398437510 DEBUG (1): RoutingMsgTimer fired!0:0:10.398437510 DEBUG (1): Sending RoutingMsg...
0:0:10.398437510 DEBUG (1): SendTask() posted!!
0:0:10.398437510 DEBUG (1): RoutingMsg enqueued successfully in SendingQueue!!!
0:0:10.398437510 DEBUG (3): RoutingMsgTimer fired!0:0:10.398437510 DEBUG (3): Sending RoutingMsg...
0:0:10.398437510 DEBUG (3): SendTask() posted!!
0:0:10.398437510 DEBUG (3): RoutingMsg enqueued successfully in SendingQueue!!!
0:0:10.398437510 DEBUG (4): RoutingMsgTimer fired!0:0:10.398437510 DEBUG (4): Sending RoutingMsg...
0:0:10.398437510 DEBUG (4): SendTask() posted!!
0:0:10.398437510 DEBUG (4): RoutingMsg enqueued successfully in SendingQueue!!!
0:0:10.398437520 DEBUG (1): sendRoutingTask(): Send returned success!!!
0:0:10.398437520 DEBUG (3): sendRoutingTask(): Send returned success!!!
0:0:10.398437520 DEBUG (4): sendRoutingTask(): Send returned success!!!
0:0:10.403976428 DEBUG (7): ### RoutingReceive.receive() start #####
0:0:10.403976428 DEBUG (7): ### RoutingReceive.receive() end #####
0:0:10.403976428 DEBUG (6): ### RoutingReceive.receive() start #####
0:0:10.403976428 DEBUG (6): ### RoutingReceive.receive() end #####
0:0:10.403976428 DEBUG (4): ### RoutingReceive.receive() start #####
0:0:10.403976428 DEBUG (4): ### RoutingReceive.receive() end #####
0:0:10.403976428 DEBUG (1): ### RoutingReceive.receive() start #####
0:0:10.403976428 DEBUG (1): ### RoutingReceive.receive() end #####
0:0:10.403976428 DEBUG (0): ### RoutingReceive.receive() start #####
0:0:10.403976428 DEBUG (0): ### RoutingReceive.receive() end #####
0:0:10.403976438 DEBUG (7): ReceiveRoutingTask(): len=4
0:0:10.403976438 DEBUG (7): Now the node 7 have a perent with parentID 3 and currentdepth 20:0:10.403976438 DEBUG (6): ReceiveRouting
Task(): len=4
0:0:10.403976438 DEBUG (6): Now the node 6 have a perent with parentID 3 and currentdepth 20:0:10.403976438 DEBUG (4): ReceiveRouting
Task(): len=4
0:0:10.403976438 DEBUG (4): the node 4 has already parent with parentID 0 and curdepth 10:0:10.403976438 DEBUG (1): ReceiveRoutingTas
k(): len=4
0:0:10.403976438 DEBUG (1): the node 1 has already parent with parentID 0 and curdepth 10:0:10.403976438 DEBUG (0): ReceiveRoutingTas
k(): len=4
0:0:10.403976438 DEBUG (0): the node 0 has already parent with parentID 0 and curdepth 00:0:10.404144274 DEBUG (3): Package sent True
```

```
0:0:10.406890839 DEBUG (4): ### RoutingReceive.receive() end #####
0:0:10.406890839 DEBUG (3): ### RoutingReceive.receive() start #####
0:0:10.406890839 DEBUG (3): ### RoutingReceive.receive() end #####
0:0:10.406890839 DEBUG (2): ### RoutingReceive.receive() start #####
0:0:10.406890839 DEBUG (2): ### RoutingReceive.receive() end #####
0:0:10.406890839 DEBUG (0): ### RoutingReceive.receive() start #####
0:0:10.406890839 DEBUG (0): ### RoutingReceive.receive() end #####
0:0:10.406890849 DEBUG (5): ReceiveRoutingTask(): len=4
0:0:10.406890849 DEBUG (5): Now the node 5 have a perent with parentID 1 and currentdepth 20:0:10.406890849 DEBUG (4): ReceiveRouting
Task(): len=4
0:0:10.406890849 DEBUG (4): the node 4 has already parent with parentID 0 and curdepth 10:0:10.406890849 DEBUG (3): ReceiveRoutingTas
k(): len=4
0:0:10.406890849 DEBUG (3): the node 3 has already parent with parentID 0 and curdepth 10:0:10.406890849 DEBUG (2): ReceiveRoutingTas
k(): len=4
0:0:10.406890849 DEBUG (2): Now the node 2 have a perent with parentID 1 and currentdepth 20:0:10.406890849 DEBUG (0): ReceiveRouting
Task(): len=4
0:0:10.406890849 DEBUG (0): the node 0 has already parent with parentID 0 and curdepth 00:0:10.407058685 DEBUG (1): Package sent True

0:0:10.408325158 DEBUG (8): ### RoutingReceive.receive() start #####
0:0:10.408325158 DEBUG (8): ### RoutingReceive.receive() end #####
0:0:10.408325158 DEBUG (7): ### RoutingReceive.receive() start #####
0:0:10.408325158 DEBUG (7): ### RoutingReceive.receive() end #####
0:0:10.408325158 DEBUG (6): ### RoutingReceive.receive() start #####
0:0:10.408325158 DEBUG (6): ### RoutingReceive.receive() end #####
0:0:10.408325158 DEBUG (5): ### RoutingReceive.receive() start #####
0:0:10.408325158 DEBUG (5): ### RoutingReceive.receive() end #####
0:0:10.408325158 DEBUG (3): ### RoutingReceive.receive() start #####
0:0:10.408325158 DEBUG (3): ### RoutingReceive.receive() end #####
0:0:10.408325158 DEBUG (2): ### RoutingReceive.receive() start #####
0:0:10.408325158 DEBUG (2): ### RoutingReceive.receive() end #####
0:0:10.408325158 DEBUG (1): ### RoutingReceive.receive() start #####
0:0:10.408325158 DEBUG (1): ### RoutingReceive.receive() end #####
0:0:10.408325158 DEBUG (0): ### RoutingReceive.receive() start #####
0:0:10.408325158 DEBUG (0): ### RoutingReceive.receive() end #####
0:0:10.408325168 DEBUG (1): ReceiveRoutingTask(): len=4
0:0:10.408325168 DEBUG (1): the node 1 has already parent with parentID 0 and curdepth 10:0:10.408325168 DEBUG (0): ReceiveRoutingTas
k(): len=4
0:0:10.408325168 DEBUG (0): the node 0 has already parent with parentID 0 and curdepth 00:0:10.408325168 DEBUG (2): ReceiveRoutingTas
k(): len=4
0:0:10.408325168 DEBUG (2): the node 2 has already parent with parentID 1 and curdepth 20:0:10.408325168 DEBUG (3): ReceiveRoutingTas
k(): len=4
0:0:10.408325168 DEBUG (3): the node 3 has already parent with parentID 0 and curdepth 10:0:10.408325168 DEBUG (8): ReceiveRoutingTas
k(): len=4
0:0:10.408325168 DEBUG (8): Now the node 8 have a perent with parentID 4 and currentdepth 20:0:10.408325168 DEBUG (7): ReceiveRouting
Task(): len=4
0:0:10.408325168 DEBUG (7): the node 7 has already parent with parentID 3 and curdepth 20:0:10.408325168 DEBUG (5): ReceiveRoutingTas
k(): len=4
0:0:10.408325168 DEBUG (5): the node 5 has already parent with parentID 1 and curdepth 20:0:10.408325168 DEBUG (6): ReceiveRoutingTas
k(): len=4
0:0:10.408325168 DEBUG (6): the node 6 has already parent with parentID 3 and curdepth 20:0:10.408493003 DEBUG (4): Package sent True

0:0:10.598632823 DEBUG (6): RoutingMsgTimer fired!0:0:10.598632823 DEBUG (6): Sending RoutingMsg...
0:0:10.598632823 DEBUG (6): SendTask() posted!!
0:0:10.598632823 DEBUG (6): RoutingMsg enqueued successfully in SendingQueue!!!
0:0:10.598632823 DEBUG (7): RoutingMsgTimer fired!0:0:10.598632823 DEBUG (7): Sending RoutingMsg...
```

Μετα φτιάξαμε ένα παράδειγμα με έναν πίνακα D=3 και εμβέλεια 1.5 και μας την παρακάτω σχέση μεταξύ των κόμβων.



Στην πρώτη εποχή όπως φαίνεται :

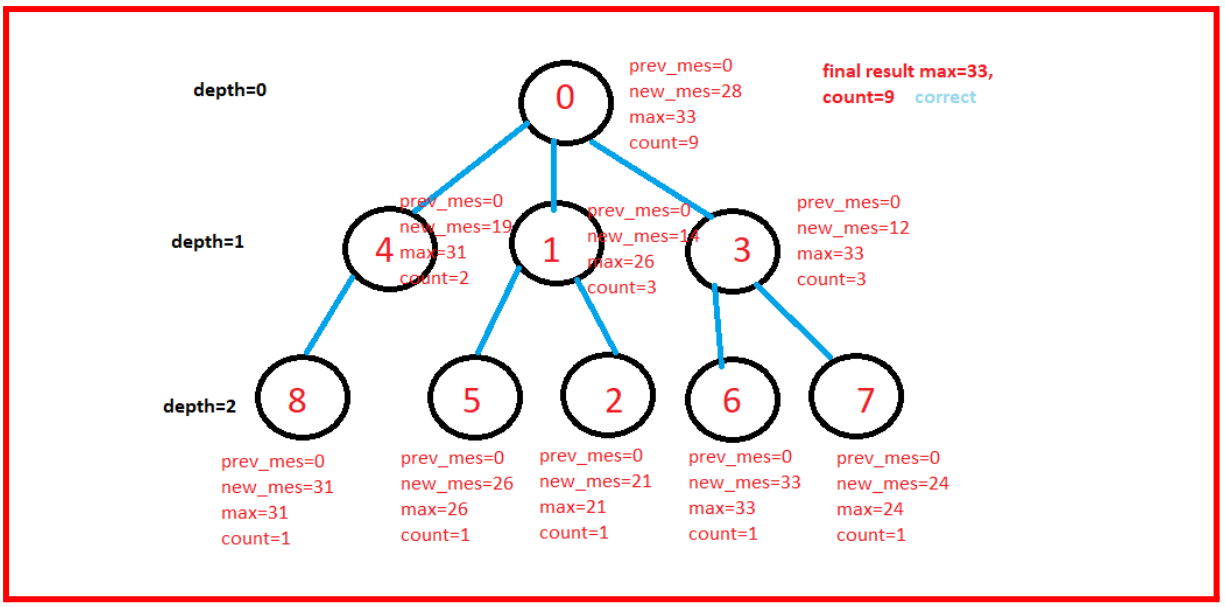
```
0:0:33.832031260 DEBUG (2): NODE_ID=2, curdepth=2
0:0:33.832031260 DEBUG (2): Starting data transmition to parent!
0:0:33.832031260 DEBUG (2): prev_measurment is:0
0:0:33.832031260 DEBUG (2): New measurment is:21
0:0:33.832031260 DEBUG (2): NotifyMsg enqueue successfully!!!1
0:0:33.832031270 DEBUG (2): the max is 21
0:0:33.832031270 DEBUG (2): the count is 1
0:0:33.832031270 DEBUG (2):  max function:measurments pass the tct!New measurment:21 prev:0
0:0:33.832031270 DEBUG (2):  count function:measurments pass the tct!New measurment:1,prev:0
0:0:33.832031270 DEBUG (2): SendMessTask(): Send returned success!!!
0:0:33.837890635 DEBUG (5): NODE_ID=5, curdepth=2
0:0:33.837890635 DEBUG (5): Starting data transmition to parent!
0:0:33.837890635 DEBUG (5): prev_measurment is:0
0:0:33.837890635 DEBUG (5): New measurment is:26
0:0:33.837890635 DEBUG (5): NotifyMsg enqueue successfully!!!1
0:0:33.837890645 DEBUG (5): the max is 26
0:0:33.837890645 DEBUG (5): the count is 1
0:0:33.837890645 DEBUG (5):  max function:measurments pass the tct!New measurment:26 prev:0
0:0:33.837890645 DEBUG (5):  count function:measurments pass the tct!New measurment:1,prev:0
0:0:33.837890645 DEBUG (5): SendMessTask(): Send returned success!!!
0:0:33.838088974 DEBUG (1): ### MessageReceive.receive() start #####
0:0:33.838088984 DEBUG (1): receiveMessTask(): len=4
0:0:33.838088984 DEBUG (1): New child
0:0:33.838088984 DEBUG (1): Received from Child :2 max:21
0:0:33.838088984 DEBUG (1): Received from Child :2 count:1
0:0:33.838088984 DEBUG (1): message receined from 2
0:0:33.838256819 DEBUG (2): Package sent True
0:0:33.839843760 DEBUG (6): NODE_ID=6, curdepth=2
0:0:33.839843760 DEBUG (6): Starting data transmition to parent!
0:0:33.839843760 DEBUG (6): prev_measurment is:0
0:0:33.839843760 DEBUG (6): New measurment is:33
0:0:33.839843760 DEBUG (6): NotifyMsg enqueue successfully!!!1
0:0:33.839843770 DEBUG (6): the max is 33
0:0:33.839843770 DEBUG (6): the count is 1
0:0:33.839843770 DEBUG (6):  max function:measurments pass the tct!New measurment:33 prev:0
0:0:33.839843770 DEBUG (6):  count function:measurments pass the tct!New measurment:1,prev:0
0:0:33.839843770 DEBUG (6): SendMessTask(): Send returned success!!!
0:0:33.841796885 DEBUG (7): NODE_ID=7, curdepth=2
0:0:33.841796885 DEBUG (7): Starting data transmition to parent!
0:0:33.841796885 DEBUG (7): prev_measurment is:0
0:0:33.841796885 DEBUG (7): New measurment is:24
0:0:33.841796885 DEBUG (7): NotifyMsg enqueue successfully!!!1
0:0:33.841796895 DEBUG (7): the max is 24
0:0:33.841796895 DEBUG (7): the count is 1
0:0:33.841796895 DEBUG (7):  max function:measurments pass the tct!New measurment:24 prev:0
```

```
0:0:33.841796895 DEBUG (7):  max function:measurments pass the tct!New measurment:24 prev:0
0:0:33.841796895 DEBUG (7):  count function:measurments pass the tct!New measurment:1,prev:0
0:0:33.841796895 DEBUG (7): SendMessTask(): Send returned success!!!
0:0:33.843750010 DEBUG (8): NODE_ID=8, curdepth=2
0:0:33.843750010 DEBUG (8): Starting data transmition to parent!
0:0:33.843750010 DEBUG (8): prev_measurment is:0
0:0:33.843750010 DEBUG (8): New measurment is:31
0:0:33.843750010 DEBUG (8): NotifyMsg enqueue successfully!!!1
0:0:33.843750020 DEBUG (8): the max is 31
0:0:33.843750020 DEBUG (8): the count is 1
0:0:33.843750020 DEBUG (8):  max function:measurments pass the tct!New measurment:31 prev:0
0:0:33.843750020 DEBUG (8):  count function:measurments pass the tct!New measurment:1,prev:0
0:0:33.843750020 DEBUG (8): SendMessTask(): Send returned success!!!
0:0:33.844680790 DEBUG (3): ### MessageReceive.receive() start #####
0:0:33.844680800 DEBUG (3): receiveMessTask(): len=4
0:0:33.844680800 DEBUG (3): New child
0:0:33.844680800 DEBUG (3): Received from Child :7 max:24
0:0:33.844680800 DEBUG (3): Received from Child :7 count:1
0:0:33.844680800 DEBUG (3): message receined from 7
0:0:33.844848635 DEBUG (7): Package sent True
0:0:33.847869835 DEBUG (1): ### MessageReceive.receive() start #####
0:0:33.847869845 DEBUG (1): receiveMessTask(): len=4
0:0:33.847869845 DEBUG (1): New child
0:0:33.847869845 DEBUG (1): Received from Child :5 max:26
0:0:33.847869845 DEBUG (1): Received from Child :5 count:1
0:0:33.847869845 DEBUG (1): message receined from 5
0:0:33.848037681 DEBUG (5): Package sent True
0:0:33.849441515 DEBUG (4): ### MessageReceive.receive() start #####
0:0:33.849441525 DEBUG (4): receiveMessTask(): len=4
0:0:33.849441525 DEBUG (4): New child
0:0:33.849441525 DEBUG (4): Received from Child :8 max:31
0:0:33.849441525 DEBUG (4): Received from Child :8 count:1
0:0:33.849441525 DEBUG (4): message receined from 8
0:0:33.849609361 DEBUG (8): Package sent True
0:0:33.850952104 DEBUG (3): ### MessageReceive.receive() start #####
0:0:33.850952114 DEBUG (3): receiveMessTask(): len=4
0:0:33.850952114 DEBUG (3): New child
0:0:33.850952114 DEBUG (3): Received from Child :6 max:33
0:0:33.850952114 DEBUG (3): Received from Child :6 count:1
0:0:33.850952114 DEBUG (3): message receined from 6
0:0:33.851119949 DEBUG (6): Package sent True
0:0:34.025390635 DEBUG (1): NODE_ID=1, curdepth=1
0:0:34.025390635 DEBUG (1): Starting data transmition to parent!
0:0:34.025390635 DEBUG (1): prev_measurment is:0
0:0:34.025390635 DEBUG (1): New measurment is:14
0:0:34.025390635 DEBUG (1): NotifyMsg enqueue successfully!!!1
0:0:34.025390645 DEBUG (1): child 2 has max 21
0:0:34.025390645 DEBUG (1): child 5 has max 26
0:0:34.025390645 DEBUG (1): the max is 26
0:0:34.025390645 DEBUG (1): child 2 has count 1
0:0:34.025390645 DEBUG (1): child 5 has count 1
0:0:34.025390645 DEBUG (1): the count is 3
0:0:34.025390645 DEBUG (1):  max function:measurments pass the tct!New measurment:26 prev:0
0:0:34.025390645 DEBUG (1):  count function:measurments pass the tct!New measurment:3,prev:0
```

```
0:0:34.025390645 DEBUG (1): SendMessTask(): Send returned success!!!
0:0:34.029296885 DEBUG (3): NODE_ID=3, curdepth=1
0:0:34.029296885 DEBUG (3): Starting data transmition to parent!
0:0:34.029296885 DEBUG (3): prev_measurment is:0
0:0:34.029296885 DEBUG (3): New measurment is:12
0:0:34.029296885 DEBUG (3): NotifyMsg enqueue successfully!!!1
0:0:34.029296895 DEBUG (3): child 7 has max 24
0:0:34.029296895 DEBUG (3): child 6 has max 33
0:0:34.029296895 DEBUG (3): the max is 33
0:0:34.029296895 DEBUG (3): child 7 has count 1
0:0:34.029296895 DEBUG (3): child 6 has count 1
0:0:34.029296895 DEBUG (3): the count is 3
0:0:34.029296895 DEBUG (3):  max function:measurments pass the tct!New measurment:33 prev:0
0:0:34.029296895 DEBUG (3):  count function:measurments pass the tct!New measurment:3,prev:0
0:0:34.029296895 DEBUG (3): SendMessTask(): Send returned success!!!
0:0:34.029556270 DEBUG (0): ### MessageReceive.receive() start #####
0:0:34.029556280 DEBUG (0): receiveMessTask(): len=4
0:0:34.029556280 DEBUG (0): New child
0:0:34.029556280 DEBUG (0): Received from Child :1 max:26
0:0:34.029556280 DEBUG (0): Received from Child :1 count:3
0:0:34.029556280 DEBUG (0): message receined from 1
0:0:34.029724115 DEBUG (1): Package sent True
0:0:34.031250010 DEBUG (4): NODE_ID=4, curdepth=1
0:0:34.031250010 DEBUG (4): Starting data transmition to parent!
0:0:34.031250010 DEBUG (4): prev_measurment is:0
0:0:34.031250010 DEBUG (4): New measurment is:19
0:0:34.031250010 DEBUG (4): NotifyMsg enqueue successfully!!!1
0:0:34.031250020 DEBUG (4): child 8 has max 31
0:0:34.031250020 DEBUG (4): the max is 31
0:0:34.031250020 DEBUG (4): child 8 has count 1
0:0:34.031250020 DEBUG (4): the count is 2
0:0:34.031250020 DEBUG (4):  max function:measurments pass the tct!New measurment:31 prev:0
0:0:34.031250020 DEBUG (4):  count function:measurments pass the tct!New measurment:2,prev:0
0:0:34.031250020 DEBUG (4): SendMessTask(): Send returned success!!!
0:0:34.031768804 DEBUG (0): ### MessageReceive.receive() start #####
0:0:34.031768814 DEBUG (0): receiveMessTask(): len=4
0:0:34.031768814 DEBUG (0): New child
0:0:34.031768814 DEBUG (0): Received from Child :3 max:33
0:0:34.031768814 DEBUG (0): Received from Child :3 count:3
0:0:34.031768814 DEBUG (0): message receined from 3
0:0:34.031936650 DEBUG (3): Package sent True
0:0:34.041366538 DEBUG (0): ### MessageReceive.receive() start #####
0:0:34.041366548 DEBUG (0): receiveMessTask(): len=4
0:0:34.041366548 DEBUG (0): New child
0:0:34.041366548 DEBUG (0): Received from Child :4 max:31
0:0:34.041366548 DEBUG (0): Received from Child :4 count:2
0:0:34.041366548 DEBUG (0): message receined from 4
0:0:34.041534384 DEBUG (4): Package sent True
0:0:34.218750010 DEBUG (0): NODE_ID=0, curdepth=0
0:0:34.218750010 DEBUG (0): prev_measurment is:0
0:0:34.218750010 DEBUG (0): New measurment is:28
0:0:34.218750010 DEBUG (0): NotifyMsg enqueue successfully!!!1
0:0:34.218750020 DEBUG (0): child 1 has max 26
0:0:34.218750020 DEBUG (0): child 3 has max 33
0:0:34.218750020 DEBUG (0): child 4 has max 31
```

```
0:0:34.218750010 DEBUG (0): New measurment is:28
0:0:34.218750010 DEBUG (0): NotifyMsg enqueue successfully!!!1
0:0:34.218750020 DEBUG (0): child 1 has max 26
0:0:34.218750020 DEBUG (0): child 3 has max 33
0:0:34.218750020 DEBUG (0): child 4 has max 31
0:0:34.218750020 DEBUG (0): the max is 33
0:0:34.218750020 DEBUG (0): Final result of max function:33
0:0:34.218750020 DEBUG (0): child 1 has count 3
0:0:34.218750020 DEBUG (0): child 3 has count 3
0:0:34.218750020 DEBUG (0): child 4 has count 2
0:0:34.218750020 DEBUG (0): the count is 9
0:0:34.218750020 DEBUG (0): Final result of count function:9
```

Θα βγει το σωστό αποτέλεσμα αλλά πάμε να δούμε όλη τη διαδικασία που έχει αναπαρασταθεί σε ένα δέντρο:



Όπως φαίνεται στον κώδικα και στο παράδειγμά μας από πάνω το πρόγραμμα βρίσκει σωστά MAX τιμή καθώς και τη count.Ξεκίνα Αρχικά να στελνουν οι κόμβοιμε depth = 2 μετά πηγαίνει depth = 1 και στο τέλος τυπώνει το τελικό αποτέλεσμα οταν φτάνει στο depth = 0.Για την ακρίβεια ξεκινάει το Node = 2 όπως μπορούμε να δούμε και από τον κωδικα μετά στέλνει ο 5 αμέσως μετά ο 6 7 και 8.Μετά ο 1οποίος αθροίζει το count των παιδιών του και βρίσκει την max τιμή ανάμεσα στη δικιά του και των παιδιών του και ακριβώς το ίδιο κάνουν και οι κόμβοι 1 και 3. Τέλος φτάνουμε στη ρίζα όπου και εκείνη με τη σειρά της αθροίζει το τελικό count και βρίσκει το Max ανάμεσα στη δική της τιμή και των παιδιών της και τα τυπώνει get τελειώνει μία εποχή.

Στην πρώτη εποχη πάντα περνάει το tct προηγούμενο αποτέλεσμα.
Για να βεβαιωθείτε ότι δουλεύει καλά το tct και όταν υπάρχει προϋπάρχουσα μέτρηση σας παραθέτω τα παρακάτω αποτελέσματα:

```
0:19:31.333984385 DEBUG (2): NODE_ID=2, curdepth=3
0:19:31.333984385 DEBUG (2): Starting data transmition to parent!
0:19:31.333984385 DEBUG (2): measurment is:12
0:19:31.333984385 DEBUG (2): NotifyMsg enqueue successfully!!!1
0:19:31.333984395 DEBUG (2): the max is 12
0:19:31.333984395 DEBUG (2): the count is 1
0:19:31.333984395 DEBUG (2):  max function:measurments pass the tct!New measurment:12 prev:13
0:19:31.333984395 DEBUG (2): count function:measurments  dont pass the tct!old measurment:1
```

```
0:18:32.941406260 DEBUG (4): NODE_ID=4, curdepth=2
0:18:32.941406260 DEBUG (4): Starting data transmition to parent!
0:18:32.941406260 DEBUG (4): measurment is:14
0:18:32.941406260 DEBUG (4): NotifyMsg enqueue successfully!!!1
0:18:32.941406270 DEBUG (4): the max is 14
0:18:32.941406270 DEBUG (4): the count is 1
0:18:32.941406270 DEBUG (4): max function:measurments  dont pass the tct!old measurment:14
0:18:32.941406270 DEBUG (4): count function:measurments  dont pass the tct!old measurment:1
```

Αν εχουμε μονο count:

```
0:0:28.951171885 DEBUG (2): NODE_ID=2, curdepth=2
0:0:28.951171885 DEBUG (2): Starting data transmition to parent!
0:0:28.951171885 DEBUG (2): measurment is:21
0:0:28.951171885 DEBUG (2): NotifyMsg enqueue successfully!!!1
0:0:28.951171895 DEBUG (2): the count is 1
0:0:28.951171895 DEBUG (2): measurments pass the tct!New measurment:1
0:0:28.951171895 DEBUG (2): sendNotifyTask(): Send returned success!!!
0:0:28.958145121 DEBUG (1): ### NotifyReceive.receive() start #####
0:0:28.958145121 DEBUG (1): Something received!!!  from 421   2
0:0:28.958145121 DEBUG (1): ### NotifyReceive.receive() end #####
0:0:28.958145131 DEBUG (1): ReceiveNotifyTask(): len=1
0:0:28.958145131 DEBUG (1): New child0:0:28.958145131 DEBUG (1): message receined from 2
0:0:28.958312966 DEBUG (2): Package sent True
0:0:28.959960948 DEBUG (5): NODE_ID=5, curdepth=2
0:0:28.959960948 DEBUG (5): Starting data transmition to parent!
0:0:28.959960948 DEBUG (5): measurment is:26
0:0:28.959960948 DEBUG (5): NotifyMsg enqueue successfully!!!1
0:0:28.959960958 DEBUG (5): the count is 1
0:0:28.959960958 DEBUG (5): measurments pass the tct!New measurment:1
0:0:28.959960958 DEBUG (5): sendNotifyTask(): Send returned success!!!
0:0:28.962890635 DEBUG (6): NODE_ID=6, curdepth=2
0:0:28.962890635 DEBUG (6): Starting data transmition to parent!
0:0:28.962890635 DEBUG (6): measurment is:33
0:0:28.962890635 DEBUG (6): NotifyMsg enqueue successfully!!!1
0:0:28.962890645 DEBUG (6): the count is 1
0:0:28.962890645 DEBUG (6): measurments pass the tct!New measurment:1
0:0:28.962890645 DEBUG (6): sendNotifyTask(): Send returned success!!!
0:0:28.965515142 DEBUG (3): ### NotifyReceive.receive() start #####
0:0:28.965515142 DEBUG (3): Something received!!!  from 421   6
0:0:28.965515142 DEBUG (3): ### NotifyReceive.receive() end #####
0:0:28.965515152 DEBUG (3): ReceiveNotifyTask(): len=1
0:0:28.965515152 DEBUG (3): New child0:0:28.965515152 DEBUG (3): message receined from 6
0:0:28.965682987 DEBUG (6): Package sent True
0:0:28.965820323 DEBUG (7): NODE_ID=7, curdepth=2
0:0:28.965820323 DEBUG (7): Starting data transmition to parent!
0:0:28.965820323 DEBUG (7): measurment is:24
0:0:28.965820323 DEBUG (7): NotifyMsg enqueue successfully!!!1
0:0:28.965820333 DEBUG (7): the count is 1
0:0:28.965820333 DEBUG (7): measurments pass the tct!New measurment:1
0:0:28.965820333 DEBUG (7): sendNotifyTask(): Send returned success!!!
0:0:28.967544531 DEBUG (1): ### NotifyReceive.receive() start #####
0:0:28.967544531 DEBUG (1): Something received!!!  from 421   5
0:0:28.967544531 DEBUG (1): ### NotifyReceive.receive() end #####
0:0:28.967544541 DEBUG (1): ReceiveNotifyTask(): len=1
0:0:28.967544541 DEBUG (1): New child0:0:28.967544541 DEBUG (1): message receined from 5
0:0:28.967712377 DEBUG (5): Package sent True
0:0:28.968750010 DEBUG (8): NODE_ID=8, curdepth=2
0:0:28.968750010 DEBUG (8): Starting data transmition to parent!
0:0:28.968750010 DEBUG (8): measurment is:31
0:0:28.968750010 DEBUG (8): NotifyMsg enqueue successfully!!!1
0:0:28.968750020 DEBUG (8): the count is 1
0:0:28.968750020 DEBUG (8): measurments pass the tct!New measurment:1
```

```
0:0:29.143554707 DEBUG (1): child 2 has count 1
0:0:29.143554707 DEBUG (1): child 5 has count 1
0:0:29.143554707 DEBUG (1): the count is 3
0:0:29.143554707 DEBUG (1): measurments pass the tct!New measurment:3
0:0:29.143554707 DEBUG (1): sendNotifyTask(): Send returned success!!!
0:0:29.149414072 DEBUG (3): NODE_ID=3, curdepth=1
0:0:29.149414072 DEBUG (3): Starting data transmition to parent!
0:0:29.149414072 DEBUG (3): measurment is:12
0:0:29.149414072 DEBUG (3): NotifyMsg enqueue successfully!!!1
0:0:29.149414082 DEBUG (3): child 6 has count 1
0:0:29.149414082 DEBUG (3): child 7 has count 1
0:0:29.149414082 DEBUG (3): the count is 3
0:0:29.149414082 DEBUG (3): measurments pass the tct!New measurment:3
0:0:29.149414082 DEBUG (3): sendNotifyTask(): Send returned success!!!
0:0:29.151336644 DEBUG (0): ### NotifyReceive.receive() start #####
0:0:29.151336644 DEBUG (0): Something received!!!  from 933   1
0:0:29.151336644 DEBUG (0): ### NotifyReceive.receive() end #####
0:0:29.151336654 DEBUG (0): ReceiveNotifyTask(): len=1
0:0:29.151336654 DEBUG (0): New child0:0:29.151336654 DEBUG (0): message receined from 1
0:0:29.151504490 DEBUG (1): Package sent True
0:0:29.152343760 DEBUG (4): NODE_ID=4, curdepth=1
0:0:29.152343760 DEBUG (4): Starting data transmition to parent!
0:0:29.152343760 DEBUG (4): measurment is:19
0:0:29.152343760 DEBUG (4): NotifyMsg enqueue successfully!!!1
0:0:29.152343770 DEBUG (4): child 8 has count 1
0:0:29.152343770 DEBUG (4): the count is 2
0:0:29.152343770 DEBUG (4): measurments pass the tct!New measurment:2
0:0:29.152343770 DEBUG (4): sendNotifyTask(): Send returned success!!!
0:0:29.156829828 DEBUG (0): ### NotifyReceive.receive() start #####
0:0:29.156829828 DEBUG (0): Something received!!!  from 677   4
0:0:29.156829828 DEBUG (0): ### NotifyReceive.receive() end #####
0:0:29.156829838 DEBUG (0): ReceiveNotifyTask(): len=1
0:0:29.156829838 DEBUG (0): New child0:0:29.156829838 DEBUG (0): message receined from 4
0:0:29.156997673 DEBUG (4): Package sent True
0:0:29.158477750 DEBUG (0): ### NotifyReceive.receive() start #####
0:0:29.158477750 DEBUG (0): Something received!!!  from 933   3
0:0:29.158477750 DEBUG (0): ### NotifyReceive.receive() end #####
0:0:29.158477760 DEBUG (0): ReceiveNotifyTask(): len=1
0:0:29.158477760 DEBUG (0): New child0:0:29.158477760 DEBUG (0): message receined from 3
0:0:29.158645596 DEBUG (3): Package sent True
0:0:29.335937510 DEBUG (0): NODE_ID=0, curdepth=0
0:0:29.335937510 DEBUG (0): measurment is:28
0:0:29.335937510 DEBUG (0): NotifyMsg enqueue successfully!!!1
0:0:29.335937520 DEBUG (0): child 1 has count 3
0:0:29.335937520 DEBUG (0): child 4 has count 2
0:0:29.335937520 DEBUG (0): child 3 has count 3
0:0:29.335937520 DEBUG (0): the count is 9
0:0:29.335937520 DEBUG (0): Final result of count function:9
```

Φαίνεται η διαδικασία που παίρνω μόνο το count του δέντρου ως αποτέλεσμα το οποίο είναι 9 όπως και το περιμέναμε.

Αν έχουμε μόνο Max τότε πάλι φαίνεται όλη η διαδικασία παρακάτω:

```
0:0:28.951171885 DEBUG (2): NODE_ID=2, curdepth=2
0:0:28.951171885 DEBUG (2): Starting data transmition to parent!
0:0:28.951171885 DEBUG (2): measurment is:21
0:0:28.951171885 DEBUG (2): NotifyMsg enqueue successfully!!!1
0:0:28.951171895 DEBUG (2): the max is 21
0:0:28.951171895 DEBUG (2): measurments pass the tct!New measurment:21
0:0:28.951171895 DEBUG (2): sendNotifyTask(): Send returned success!!!
0:0:28.959579438 DEBUG (1): ### NotifyReceive.receive() start #####
0:0:28.959579438 DEBUG (1): Something received!!!  from 5499   2
0:0:28.959579438 DEBUG (1): ### NotifyReceive.receive() end #####
0:0:28.959579448 DEBUG (1): ReceiveNotifyTask(): len=1
0:0:28.959579448 DEBUG (1): New child0:0:28.959579448 DEBUG (1): message receined from 2
0:0:28.959747284 DEBUG (2): Package sent True
0:0:28.959960948 DEBUG (5): NODE_ID=5, curdepth=2
0:0:28.959960948 DEBUG (5): Starting data transmition to parent!
0:0:28.959960948 DEBUG (5): measurment is:26
0:0:28.959960948 DEBUG (5): NotifyMsg enqueue successfully!!!1
0:0:28.959960958 DEBUG (5): the max is 26
0:0:28.959960958 DEBUG (5): measurments pass the tct!New measurment:26
0:0:28.959960958 DEBUG (5): sendNotifyTask(): Send returned success!!!
0:0:28.962890635 DEBUG (6): NODE_ID=6, curdepth=2
0:0:28.962890635 DEBUG (6): Starting data transmition to parent!
0:0:28.962890635 DEBUG (6): measurment is:33
0:0:28.962890635 DEBUG (6): NotifyMsg enqueue successfully!!!1
0:0:28.962890645 DEBUG (6): the max is 33
0:0:28.962890645 DEBUG (6): measurments pass the tct!New measurment:33
0:0:28.962890645 DEBUG (6): sendNotifyTask(): Send returned success!!!
0:0:28.965820323 DEBUG (7): NODE_ID=7, curdepth=2
0:0:28.965820323 DEBUG (7): Starting data transmition to parent!
0:0:28.965820323 DEBUG (7): measurment is:24
0:0:28.965820323 DEBUG (7): NotifyMsg enqueue successfully!!!1
0:0:28.965820333 DEBUG (7): the max is 24
0:0:28.965820333 DEBUG (7): measurments pass the tct!New measurment:24
0:0:28.965820333 DEBUG (7): sendNotifyTask(): Send returned success!!!
0:0:28.966445922 DEBUG (3): ### NotifyReceive.receive() start #####
0:0:28.966445922 DEBUG (3): Something received!!!  from 8571   6
0:0:28.966445922 DEBUG (3): ### NotifyReceive.receive() end #####
0:0:28.966445932 DEBUG (3): ReceiveNotifyTask(): len=1
0:0:28.966445932 DEBUG (3): New child0:0:28.966445932 DEBUG (3): message receined from 6
0:0:28.966613768 DEBUG (6): Package sent True
0:0:28.968750010 DEBUG (8): NODE_ID=8, curdepth=2
0:0:28.968750010 DEBUG (8): Starting data transmition to parent!
0:0:28.968750010 DEBUG (8): measurment is:31
0:0:28.968750010 DEBUG (8): NotifyMsg enqueue successfully!!!1
0:0:28.968750020 DEBUG (8): the max is 31
0:0:28.968750020 DEBUG (8): measurments pass the tct!New measurment:31
0:0:28.968750020 DEBUG (8): sendNotifyTask(): Send returned success!!!
0:0:28.969375575 DEBUG (1): ### NotifyReceive.receive() start #####
0:0:28.969375575 DEBUG (1): Something received!!!  from 6779   5
0:0:28.969375575 DEBUG (1): ### NotifyReceive.receive() end #####
0:0:28.969375585 DEBUG (1): ReceiveNotifyTask(): len=1
0:0:28.969375585 DEBUG (1): New child0:0:28.969375585 DEBUG (1): message receined from 5
0:0:28.969543421 DEBUG (5): Package sent True
```

```
0:0:29.143554697 DEBUG (1): NODE_ID=1, curdepth=1
0:0:29.143554697 DEBUG (1): Starting data transmition to parent!
0:0:29.143554697 DEBUG (1): measurment is:14
0:0:29.143554697 DEBUG (1): NotifyMsg enqueue successfully!!!1
0:0:29.143554707 DEBUG (1): child 2 has max 21
0:0:29.143554707 DEBUG (1): child 5 has max 26
0:0:29.143554707 DEBUG (1): the max is 26
0:0:29.143554707 DEBUG (1): measurments pass the tct!New measurment:26
0:0:29.143554707 DEBUG (1): sendNotifyTask(): Send returned success!!!
0:0:29.149414072 DEBUG (3): NODE_ID=3, curdepth=1
0:0:29.149414072 DEBUG (3): Starting data transmition to parent!
0:0:29.149414072 DEBUG (3): measurment is:12
0:0:29.149414072 DEBUG (3): NotifyMsg enqueue successfully!!!1
0:0:29.149414082 DEBUG (3): child 6 has max 33
0:0:29.149414082 DEBUG (3): child 7 has max 24
0:0:29.149414082 DEBUG (3): the max is 33
0:0:29.149414082 DEBUG (3): measurments pass the tct!New measurment:33
0:0:29.149414082 DEBUG (3): sendNotifyTask(): Send returned success!!!
0:0:29.150054913 DEBUG (0): ### NotifyReceive.receive() start #####
0:0:29.150054913 DEBUG (0): Something received!!!  from 6779   1
0:0:29.150054913 DEBUG (0): ### NotifyReceive.receive() end #####
0:0:29.150054923 DEBUG (0): ReceiveNotifyTask(): len=1
0:0:29.150054923 DEBUG (0): New child0:0:29.150054923 DEBUG (0): message receined from 1
0:0:29.150222759 DEBUG (1): Package sent True
0:0:29.151489266 DEBUG (0): ### NotifyReceive.receive() start #####
0:0:29.151489266 DEBUG (0): Something received!!!  from 8571   3
0:0:29.151489266 DEBUG (0): ### NotifyReceive.receive() end #####
0:0:29.151489276 DEBUG (0): ReceiveNotifyTask(): len=1
0:0:29.151489276 DEBUG (0): New child0:0:29.151489276 DEBUG (0): message receined from 3
0:0:29.151657111 DEBUG (3): Package sent True
0:0:29.152343760 DEBUG (4): NODE_ID=4, curdepth=1
0:0:29.152343760 DEBUG (4): Starting data transmition to parent!
0:0:29.152343760 DEBUG (4): measurment is:19
0:0:29.152343760 DEBUG (4): NotifyMsg enqueue successfully!!!1
0:0:29.152343770 DEBUG (4): child 8 has max 31
0:0:29.152343770 DEBUG (4): the max is 31
0:0:29.152343770 DEBUG (4): measurments pass the tct!New measurment:31
0:0:29.152343770 DEBUG (4): sendNotifyTask(): Send returned success!!!
0:0:29.162567099 DEBUG (0): ### NotifyReceive.receive() start #####
0:0:29.162567099 DEBUG (0): Something received!!!  from 8059   4
0:0:29.162567099 DEBUG (0): ### NotifyReceive.receive() end #####
0:0:29.162567109 DEBUG (0): ReceiveNotifyTask(): len=1
0:0:29.162567109 DEBUG (0): New child0:0:29.162567109 DEBUG (0): message receined from 4
0:0:29.162734945 DEBUG (4): Package sent True
0:0:29.335937510 DEBUG (0): NODE_ID=0, curdepth=0
0:0:29.335937510 DEBUG (0): measurment is:28
0:0:29.335937510 DEBUG (0): NotifyMsg enqueue successfully!!!1
0:0:29.335937520 DEBUG (0): child 1 has max 26
0:0:29.335937520 DEBUG (0): child 3 has max 33
0:0:29.335937520 DEBUG (0): child 4 has max 31
0:0:29.335937520 DEBUG (0): the max is 33
0:0:29.335937520 DEBUG (0): Final result of MAX function:33
```

Και βγάζει 33 όπως και περιμέναμε.

Το δοκίμασα και για μεγάλο πίνακα D=7 και και εμβέλεια 1,5 και το αποτέλεσμα ήταν ικανοποιητικό ειδικά στο count που τους μέτρησε όλους τους κόμβους και δεν έχασε κανέναν. Οπότε καταλάβαμε ότι το θέμα των Συγκρούσεων το έχουμε κάνει σχεδόν βέλτιστο .

```
0:19:31.914062510 DEBUG (0): NODE_ID=0, curdepth=0
0:19:31.914062510 DEBUG (0): measurment is:34
0:19:31.914062510 DEBUG (0): NotifyMsg enqueue successfully!!!1
0:19:31.914062520 DEBUG (0): child 1 has max 70
0:19:31.914062520 DEBUG (0): child 7 has max 37
0:19:31.914062520 DEBUG (0): child 8 has max 79
0:19:31.914062520 DEBUG (0): the max is 79
0:19:31.914062520 DEBUG (0): Final result of max function:79
0:19:31.914062520 DEBUG (0): child 1 has count 16
0:19:31.914062520 DEBUG (0): child 7 has count 1
0:19:31.914062520 DEBUG (0): child 8 has count 31
0:19:31.914062520 DEBUG (0): the count is 49
```

Κατανομή εργασιών:

Το πρόγραμμα όπως και την αναφορά την δουλέψαμε και τα δύο μέλη της ομάδας μαζί.