ΜΥΕ017 ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ
Άσκηση #2

Ονοματεπώνυμο: Χρύσα Τεριζή
ΑΜ: 2553
Ομάδα: Παναγιώτης Κουζουγλίδης, ΑΜ:2276
Ημερομηνία: 11/12/2016

Ο κώδικας είναι βασισμένος πάνω στον κώδικα της άσκησης #1.

Main(): Γίνονται οι συνδέσεις όπως θέλει η εκφώνηση.

Class PeerFactory(ClientFactory): Έχω κρατήσει τον κώδικα ίδιον με αυτόν της άσκησης #1.

Def parse_args(): Δεν έχουν γίνει αλλαγές σε αυτήν την συνάρτηση.

Έχω κάποιες global μεταβλητές όπως,

- **connectionList = []** όπου χρησιμοποιείται για την κάθε διεργασία και αποθηκεύει μέσα τις τιμές από τις άλλες διεργασίες που είναι συνδεδεμένη,
- **vectorClock = [0, 0, 0]** όπου είναι το vector clock για την κάθε διεργασία [0 διεργασία, 1 διεργασία, 2 διεργασία],
- **counterQuestions = 0** όπου δείχνει τον αριθμό της ερώτησης που στέλνεται,
- **f = None** όπου είναι η μεταβλητή για το αρχείο,
- **notDelivered = ""** όταν έρχεται μια απάντηση πρώτα πριν ακόμα έρθει η ερώτηση κρατάω το περιεχόμενο της απάντησης σε μία μεταβλητή ώστε να την γράψω σωστά όταν έρθει η ώρα,
- **X = 0, Y = 0, Z = 0** είναι μεταβλητές που βοηθάνε όταν μία απάντηση έρθει πιο πριν από μία ερώτηση και ουσιαστικά κρατάει τις τιμές του vector clock.

Class Peer(Protocol):

- **def __init__(self, factory, process_id)**: δημιουργούνται τα *.txt αρχεία
- **def connectionMade(self)**: Γίνετε η σύνδεση μεταξύ των συνδέσεων, και ξεκινάει η μετάδοση των ερωτήσεων όταν έχουν συνδεθεί όλες οι διεργασίες με όλες.
- **def wrongSendQuestion(self)**: Είναι μία συνάρτηση που έφτιαξα ώστε να δημιουργήσουμε το θέμα να στελεί πρώτα η απάντηση πριν σταλεί η ερώτηση πρώτα.
- **def sendQuestion(self)**: Χρησιμοποιείται από την διεργασία 0 η οποία στέλνει τις ερωτήσεις. Το vector clock της διεργασίας 0 αλλάζει. Η μορφή της ερώτησης είναι η εξής: **question = 'question ' + str(counterQuestions) + ' ' + str(vectorClock[int(self.process_id)]) + ' ' + str(vectorClock[int(self.process_id) + 1]) + ' ' + str(vectorClock[int(self.process_id) + 2])**, δηλαδή στέλνετε η λέξη “question” έπειτα ακολουθεί ο αριθμός της ερώτησης μετά η τιμή του vector clock στην θέση 0, έπειτα στην θέση 1 και τέλος στην θέση 2. Γράφετε στο αρχείο της. Και στέλνετε στις υπόλοιπες διεργασίες. Η συνάρτηση *wrongSendQuestion(self)* καλείται για την ερώτηση 1. Η διεργασία 0 καλεί την συνάρτηση *sendQuestion(self)* ανά 3 seconds μέχρι να ολοκληρώσει την αποστολή 20 ερωτήσεων. Όταν τις στείλει εμφανίζει ότι συνολικά έστειλε 20 ερωτήσεις.
- **def sendAnswer(self, numberOfQuestion)**: Η συνάρτηση αυτή χρησιμοποιείται από τις διεργασίες 1 και 2 για να στείλουν τις απαντήσεις για τις ερωτήσεις που δέχονται. Αλλάζει το vector clock της, δίνεται +1 στην αντίστοιχη θέση(ποια διεργασία είναι). Αν είναι η διεργασία 1 που στέλνει την απάντηση, η απάντηση έχει την μορφή:

answer = 'answer for ' + str(numberOfQuestion) + ' question from ' + str(self.process_id) + ' ' + str(vectorClock[int(self.process_id)]) + ' ' + str(vectorClock[int(self.process_id) - 1]) + ' ' + str(vectorClock[int(self.process_id) + 1]), δηλαδή στέλνεται πρώτα για το ποια ερώτηση στέλνουμε απάντηση και από ποια διεργασία στέλνετε η απάντηση. Έπειτα η τιμή του vector clock στην θέση 1, έπειτα στην θέση 0 και μετά στην θέση 2. Έπειτα εκτυπώνετε η απάντηση στο αρχείο. Αντίστοιχα αν είναι η διεργασία 2 η οποία στέλνει την απάντηση, η απάντηση έχει την μορφή:
answer = 'answer for ' + str(numberOfQuestion) + ' question from ' + str(self.process_id) + ' ' + str(vectorClock[int(self.process_id)]) + ' ' + str(vectorClock[int(self.process_id) - 2]) + ' ' + str(vectorClock[int(self.process_id) – 1]), δηλαδή ερώτηση και αριθμός ερώτησης και από ποια διεργασία έρχεται η απάντηση. Έπειτα η τιμή του vector clock στην θέση 2, έπειτα στην θέση 0 και τέλος στην θέση 1. Γράφεται τέλος στο αρχείο. Τέλος στέλνονται οι απαντήσεις στις υπόλοιπες διεργασίες

- **def dataReceived(self, data):** Αν είναι η διεργασία 0, το data που δέχεται όπου θα είναι σίγουρα απάντηση γίνεται split για να πάρουμε τα μέρη της απάντησης. Αν την απάντηση την έλαβε από την διεργασία 1 τότε ανανεώνει το vector clock της με βάση την max τιμή. Αντίστοιχα το ίδιο αν έλαβε την απάντηση από την διεργασία 2. Εκτυπώνονται οι απαντήσεις και γράφεται η απάντηση στο αρχείο. Τώρα αν δεν είναι η διεργασία 0, διαχωρίζω πάλι με το αν οι διεργασίες 1, 2 έλαβαν ερώτηση ή απάντηση. Αν η διεργασία 1 έλαβε ερώτηση ανανεώνει το vector clock της και το γράφει στο αρχείο. Εάν το *notDelivered* είναι διάφορο του κενού, σημαίνει ότι για την συγκεκριμένη ερώτηση είχε έρθει πρώτα η απάντηση της. Την γράφουμε τ αρχείο με την σωστή σειρά. Η αντίστοιχη διαδικασία γίνεται και στην περίπτωση για την διεργασία 2 αν έλαβε ερώτηση. Ακόμη γίνεται η εναλλαγή για το ποια στέλνει την απάντηση. Αρχικά απαντάει η διεργασία 1 και έπειτα η 2 και μετά γίνονται εναλλάξ. Τώρα γίνεται ο διαχωρισμός η διεργασία 1 ή 2 να έλαβαν απάντηση. Α διεργασία είναι η 1 τότε ελέγχουμε αν έχει έρθει η ερώτηση πρώτα, αν όχι την κρατάμε. Διαφορετικά, ανανεώνεται το vector clock και η απάντηση γράφεται στο αρχείο. Το αντίστοιχο γίνεται για την διεργασία 2.
- **def connectionLost(self, reason):** Κλείνει το αρχείο.
- **def done(self):** Δεν κάνει κάτι.