Εργασία 1 – Ασύγχρονο request-reply με σημασιολογία at most once και πολλούς εξυπηρετητές

Αναπτύξτε λογισμικό για την υποστήριξη του ασύγχρονου σχήματος request-reply, στο πνεύμα μιας βιβλιοθήκης που υποστηρίζει το παρακάτω ενδεικτικό API (βλέπε διαλέξεις).

Διασύνδεση πελάτη	`int sendRequest (int svcid, void *buf, int len);`
	`int getReply (int reqid, void *buf, int *len, int block);`
Διασύνδεση εξυπηρετητή	`int register (int svcid);`
	`int unregister (int svcid);`
	`int getRequest (int svcid, void *buf, int *len);`
	`void sendReply (int reqid, void *buf, int len);`

Η υλοποίηση πρέπει να είναι σχεδιασμένη ως ξεχωριστό τμήμα λογισμικού, η εσωτερική λειτουργία του οποίου δεν είναι γνωστή στην εφαρμογή ούτε είναι στενά συνδεδεμένη με το νήμα της εφαρμογής. Επίσης, η πλευρά (API) του πελάτη πρέπει να υλοποιηθεί σε διαφορετική γλώσσα από αυτήν του εξυπηρετητή.

Η ανακάλυψη του εξυπηρετητή από τον πελάτη πρέπει να γίνει με UDP/IP multicast, και η αλληλεπίδραση αίτησης-απάντησης ανάμεσα στις δύο πλευρές πρέπει να γίνει πάνω από UDP/IP. Το πρωτόκολλο που θα σχεδιάσετε πρέπει να είναι ανεξάρτητο γλώσσας προγραμματισμού / πλατφόρμας εκτέλεσης. Για απλότητα, μπορείτε να υποθέσετε ότι κάθε αίτηση/απάντηση χωράει σε ένα UDP datagram. Όμως, δεν μπορείτε να υποθέσετε κάποιο άνω όριο για τον χρόνο επεξεργασίας των αιτήσεων του πελάτη από τον εξυπηρετητή. Η παρεχόμενη σημασιολογία πρέπει να είναι at most once ακόμα και σε περίπτωση επανεκκίνησης του εξυπηρετητή μετά από βλάβη. Αν η πλευρά του πελάτη, μετά από προσπάθειες, δεν λάβει δείγμα ζωής από τον εξυπηρετητή, υποθέτει ότι αυτός παρουσίασε βλάβη και εγκαταλείπει. Αντίστοιχα, η πλευρά του εξυπηρετητή πρέπει να εντοπίζει και να χειρίζεται πιθανές βλάβες του πελάτη.

Δοκιμάστε την υλοποίηση σας μέσω μιας απλής εφαρμογής για τον έλεγχο πρώτων αριθμών (primality test), όπου ο χρόνος επεξεργασίας μπορεί να διαφέρει σημαντικά αναλόγως με τον αριθμό ελέγχεται κάθε φορά. Ο κώδικάς της εφαρμογής πρέπει να είναι ξεχωριστός από την υλοποίηση του API. Ο εξυπηρετητής στο επίπεδο της εφαρμογής πρέπει να είναι πολυνηματικός με άνω όριο 3 worker threads. Φυσικά, αν έχετε κέφι, μπορείτε να δοκιμάσετε την υλοποίηση σας και με άλλες εφαρμογές.

Επεκτείνετε την υλοποίηση έτσι ώστε να υποστηρίζει πολλούς εξυπηρετητές που προσφέρουν την ίδια υπηρεσία και ο φόρτος επεξεργασίας να κατανέμεται ομοιόμορφα στους διαθέσιμους εξυπηρετητές. Αυτό μπορεί να επιτευχθεί με ανταλλαγή πληροφορίας απ' ευθείας ανάμεσα στους εξυπηρετητές ή μέσω ενός ξεχωριστού διαχειριστή που μεσολαβεί ανάμεσα στους πελάτες και τους εξυπηρετητές. Πρέπει να υποστηρίζεται η δυναμική προσθήκη/αφαίρεση εξυπηρετητών. Ιδανικά, δεν πρέπει να γίνουν αλλαγές στην διεπαφή προγραμματισμού, και η εφαρμογή πρέπει να τρέχει πάνω από την νέα έκδοση του λογισμικού request-reply χωρίς καμία αλλαγή στον κώδικα της.

Σχεδιάστε και κάντε πειράματα για: (α) να μετρήσετε την καθυστέρηση μιας «μηδενικής» αλληλεπίδρασης όπου τα μηνύματα σε επίπεδο εφαρμογής είναι κενά και η επεξεργασία μιας αίτησης σε επίπεδο εφαρμογής είναι ακαριαία, (β) να επιβεβαιώσετε την δυνατότητα ασύγχρονης εξυπηρέτησης αιτήσεων του ιδίου πελάτη από τον ίδιο εξυπηρετητή, (γ) να επιβεβαιώσετε την δυνατότητα εξυπηρέτησης πολλών πελάτων από τον ίδιο εξυπηρετητή, και (δ) να καταγράψτε την κατανομή των αιτήσεων (ακόμα και του ίδιου πελάτη) στους διαθέσιμους εξυπηρετητές ανάλογα με τον φόρτο που αυτές επιφέρουν στους εξυπηρετητές.

Μπορείτε να χρησιμοποιήστε όποια γλώσσα προγραμματισμού επιθυμείτε, αρκεί να έχει ρητή υποστήριξη για νήματα. Ακολουθήστε τις οδηγίες παράδοσης εργασιών (υπάρχουν στο eclass).

Φροντιστήριο/συζήτηση: **Πέμπτη 4 Μαρτίου 2021**
Ημερομηνία παράδοσης: **Σάββατο 27 Μαρτίου 2021, 22:00**