HY335α – Δικτυα Υπολογιστων (Project report)

Ομαδα: 60

Άγγελος Γκογκόσης (csd3730)

Καράντζης Κωνσταντίνος (csd5274)

Ο σκοπός του project είναι η υλοποίηση μιας βιβλιοθήκης η οποία θα παρέχει μια απλή έκδοση του TCP. Παρόλα αυτά, θα είναι ικανή να παρέχει αξιόπιστη επικοινωνία μεταξύ συσκευών με περιορισμένους υπολογιστικούς πόρους χρησιμοποιώντας το διαθέσιμο UDP πρωτόκολλο μεταφοράς.

microtcp_sock_t
microtcp_socket(int domain, int type, int protocol);

Αρχικοποιει ενα microTCP socket, πανω απο UDP. Ειναι ουσιαστικα το ακρο επικοινωνιας μεταξυ client/server μεσω δικτυου. Εχει IP, port για να γνωριζει ο peer που θα στελνει πακετα, πρωτοκολλο (UDP στην συγκεκριμενη περιπτωση) και state πχ LISTEN, για οταν ειναι ετοιμο να συνδεθει με καποιο αλλο socket (connection μεταξυ client και server), ESTABLISHED για οταν γινει το connection και αλλα οπως CLOSING_BY_PEER, CLOSING_BY_HOST, CLOSED, INVALID.

Στην συναρτηση μεσα δημιουργειται ενα νεο sock με γνωρισματα-τυπο του struct microtcp_sock_t. Σε πρωτη φαση γινεται ελεγχος για το αν το socket υποστηριζει UDP και οχι δηλαδη ετοιμο TCP. Στην περιπτωση που δεν υποστηριζει UDP επιστρεφει ως error με state = INVALID. Αν ειναι VALID, τοτε το state αρχικοποιειται ως CLOSED και γενικα αρχικοποιειται ολη η δομη του microtcp_sock_t.

int
microtcp_bind(microtcp_sock_t *socket, const struct sockaddr *address, socklen_t, address_len);

Κανει bind το socket σε μια συγκεκριμενη IP και PORT για να μπορει ο server να δεχεται πακετα που ερχονται σε αυτο το port της IP του.

Την καλει-χρειαζεται μονο ο server διοτι στην πρωτη φαση του connection στελνει πρωτος ο client, οποτε ο server πρεπει να εχει γνωστο σε ποιο port και διευθυνση ακουει και δεχεται πακετα.

Μετα το bind το state του socket του server μετατρεπεται σε LISTEN ωστε να ξεκινησει η διαδικασια της συνδεσης μεταξυ client/server, δηλαδη ο server να περιμενει καποιον να συνδεθει.

int
microtcp_connect(microtcp_sock_t *socket, const struct sockaddr *address, socklen_t address_len);

Ξεκιναει την διαδικασια 3-way handshake του TCP (πανω απο UDP). Το καλει ο client γιατι ξεκιναει απο αυτον η διαδικασια (handshake) οπου στελνει SYN στον server και seq = x. Περιμενει απο τον server SYN+ACK και seq = y, ack = x + 1 και επειτα στελνει το τελικο ACK και ack = y + 1.

Πιο αναλυτικα στην συναρτηση αρχικοποιουμε με εναν random 32bit αριθμο το αρχικο sec (του client). Δημιουργουμε το header ενος πακετου που θα στειλει SYN (14^{th} bit of control = 1) και seq = x. Υπολογιζεται το checksum του πακετου για την ορθοτητα του, οταν αυτο ληφθει και επειτα το στελνουμε. Ο server αν δει πως το checksum ειναι σωστο, μαζι με κατι extra ελεγχους τοτε μας στελνει πισω ενα πακετο SYN-ACK (14^{th} and 12^{th} bits of control = 1) και seq = y, ack = x + 1. Ελεγχουμε το checksum του πακετου που λαβαμε, αν ειναι SYN-ACK και αν το ack ειναι σωστο, και επειτα αρχικοποιουμε τα window για control flow και στελνουμε το τελικο ACK (μετα τους καταλληλους υπολογισμους checksum, seq, ack). Τελος το state του socket αλλαζει σε ESTABLISHED και γινεται εγκαθιδρυση της συνδεσης.

int
microtcp_accept(microtcp_sock_t *socket, struct sockaddr *address, socklen_t address_len);

Ειναι η server-side συναρτηση που ολοκληρωνει το 3-way handshake οταν ενας client ζηταει συνδεση. Περιμενει καποιον client να συνδεθει, δηλαδη περιμενει SYN και seq = x. Απανταει με SYN+ACK και seq = y, ack = x + 1 και επειτα λαμβανει το τελικο ACK και ack = y + 1 και αλλαζει το state σε ESTABLISHED, δηλαδη οτι εγινε εγκαθιδρυση της συνδεσης.

Πιο αναλυτικα στην συναρτηση, περιμενουμε απτον client ενα πακετο SYN (14^{th} bit of control = 1) και seq = x. Υπολογιζεται το checksum του πακετου για επιβεβαιωση οτι ηρθε το σωστο πακετο. Ελεγχεται επισης αν το πακετο ειναι οντως SYN πακετο. Αν περασει τον ελεγχο, τοτε αποθηκευει το window του client για flow control και στελνει στον client ενα πακετο SYN-ACK (14^{th} and 12^{th} bits of control = 1) και seq = y, ack = x + 1. Περιμενει να λαβει το τελικο ACK και επειτα ελεγχει το checksum του, seq, ack, control. Τελος το state του socket αλλαζει σε ESTABLISHED και γινεται εγκαθιδρυση της συνδεσης.

int
microtcp_shutdown(microtcp_sock_t *socket, int how);

Η microtcp_shutdown υλοποιεί την ομαλή τερματική διαδικασία σύνδεσης (connection teardown) του TCP πάνω από UDP, ακολουθώντας λογική αντίστοιχη του handshake του TCP. Γινεται απο την μερια του client, ξεκινοντας με FIN+ACK κιαλλαγη state σε CLOSING_BY_PEER. Παιρνει απο τον server απαντηση ACK, κιεπειτα γινεται αλλη μια φορα.
Αρχικά γίνονται έλεγχοι εγκυρότητας του socket και της κατάστασής του. Αν το socket βρίσκεται ήδη σε κατάσταση CLOSING_BY_PEER, δηλαδή έχει προηγηθεί FIN από τον peer, η συνάρτηση ολοκληρώνει άμεσα το κλείσιμο στέλνοντας FIN+ACK, αλλάζει το state σε CLOSED και κλείνει το socket. Αν το socket δεν βρίσκεται σε κατάσταση ESTABLISHED, η διαδικασία τερματισμού απορρίπτεται, καθώς δεν υπάρχει ενεργή σύνδεση.
Στη συνέχεια ορίζεται timeout στο socket, ώστε να αποφεύγεται deadlock σε περίπτωση απώλειας πακέτων. Δημιουργείται πακέτο FIN+ACK με τα κατάλληλα seq_number και ack_number, υπολογίζεται το checksum και αποστέλλεται στον peer. Το socket μεταβαίνει σε κατάσταση CLOSING_BY_HOST. Ο sender περιμένει ACK για το FIN. Αν δεν ληφθεί εντός του timeout, γίνεται επαναμετάδοση του FIN μέχρι να φτάσει το μέγιστο πλήθος retries. Αν ξεπεραστεί το όριο, η σύνδεση εγκαταλείπεται. Μετά τη λήψη του ACK, η συνάρτηση περιμένει FIN από τον peer. Αν το πακέτο που ληφθεί δεν είναι FIN ή δεν φτάσει εντός του timeout, γίνεται εκ νέου προσπάθεια λήψης μέχρι να εξαντληθούν τα retries.
Όταν ληφθεί το FIN του peer και επιβεβαιωθεί η ορθότητά του μέσω checksum και flags, αποστέλλεται το τελικό ACK με ack = server_seq + 1, ολοκληρώνοντας το shutdown handshake. Τέλος, το state του socket αλλάζει σε CLOSED, ο file descriptor κλείνει και απελευθερώνονται οι πόροι, ολοκληρώνοντας με ασφάλεια τον τερματισμό της σύνδεσης.

ssize_t
microtcp_send(microtcp_sock_t *socket, const void *buffer, size_t length, int flags);

Υλοποιει TCP αξιοπιστη μεταδοση δεδομενων πανω απο UDP, εφαρμοζοντας δηλαδη αποφυγη deadlock μεσω timeout,acknowledgments, retransmissions, flow control, congestion control, ελεγχους checksum.

Οταν ο sender στελνει ενα πακετο στον receiver, γινεται receive απο τον receiver για να λαμβανει το πακετο και επειτα απανταει πισω στελνωντας ACK. Αν δεν λαβει ο sender πακετο μεσα σε X ms τοτε αποτυγχανει και εχουμε deadlock (μπλοκαρει για παντα). Γιαυτο θετουμε αρχικα ενα timeout, οπου αν το πακετο δεν ερθει εγκαιρως κανει retransmission.

Σε κάθε επανάληψη της while υπολογίζεται το effective window, το οποίο είναι το ελάχιστο μεταξύ του congestion window (cwnd) και του παραθύρου του peer (peer_win_size). Με αυτόν τον τρόπο εφαρμόζονται ταυτόχρονα flow control και congestion control.

Στη συνέχεια υπολογίζεται πόσα bytes επιτρέπεται να σταλούν στο συγκεκριμένο βήμα (bytes_to_send), χωρίς να ξεπεραστεί ούτε το window ούτε το συνολικό μήκος των δεδομένων.

Τα δεδομένα κατακερματίζονται σε chunks μεγέθους MICROTCP_MSS. Ο αριθμός των chunks υπολογίζεται έτσι ώστε να καλύπτονται όλα τα bytes, ακόμη και αν το τελευταίο chunk είναι μικρότερο του MSS.

Για κάθε chunk δημιουργείται ένα ενιαίο πακέτο (header + payload). Αρχικοποιείται το header με τα κατάλληλα seq_number, ack_number, flags (ACK), window και μήκος δεδομένων. Τα πεδία future_use τίθενται σε 0 καθώς δεν χρησιμοποιούνται στην παρούσα υλοποίηση. Ακολουθεί αντιγραφή των δεδομένων στο πακέτο και υπολογισμός checksum για έλεγχο ακεραιότητας.

Το πακέτο αποστέλλεται με sendto. Ο sender στη συνέχεια περιμένει ACK από τον receiver. Αν το ACK είναι έγκυρο (σωστό checksum, σωστό ack_number), τότε ενημερώνεται το seq_number, αυξάνεται το data_sent και συνεχίζεται η αποστολή του επόμενου chunk.
Αν δεν ληφθεί ACK εντός του timeout, γίνεται επαναμετάδοση του ίδιου chunk, μέχρι να ξεπεραστεί ο επιτρεπτός αριθμός retransmissions.

Η διαδικασία συνεχίζεται μέχρι να σταλούν όλα τα δεδομένα. Όταν ολοκληρωθεί επιτυχώς η αποστολή, η συνάρτηση επιστρέφει το συνολικό πλήθος bytes που μεταδόθηκαν.

ssize_t
microtcp_recv(microtcp_sock_t *socket, void *buffer, size_t length, int flags);

Η microtcp_recv υλοποιει την αξιοπιστη ληψη δεδομενων TCP πανω απο UDP, διαχειριζομενη segmentation, acknowledgments, flow control, ελεγχο checksum και σωστη σειρα πακετων.
Αρχικα γινονται ελεγχοι εγκυροτητας των παραμετρων και της καταστασης του socket. Η συναρτηση μπορει να λειτουργησει μονο αν το socket βρισκεται σε καταστση ESTABLISHED, δηλαδη υπαρχει ενεργη συνδεση.
Αρχικοποιουνται οι βασικες μεταβλητες, δεσμευεται buffer για την ληψη πακετων και ξεκιναει μια while λουπα, η οποια συνεχιζει μεχρι να παραδοθουν στον χρηστη τα ζητουμενα bytes.
Σε καθε επαναληψη λαμβανεται ενα πακετο (header + data) με recvfrom. Αν το πακετο ειναι μικροτερο απο το header, απορριπτεται. Καταγραφονται στατιστικα για packets και bytes που εληφθησαν.
Γινεται ελεγχος των flags του πακετου. Αν εντοπιστει FIN, η συνδεση σημειωνεται ως CLOSING_BY_PEER, στελνεται ACK για το FIN και η συναρτηση επιστρεφει τα δεδομενα που εχουν ηδη ληφθει.
Στη συνεχεια εξαγεται το μηκος των δεδομενων και γινονται ελεγχοι συνεπειας (packet size, MSS). Ακολουθει ελεγχος checksum για την επιβεβαιωση της ακεραιοτητας του πακετου. Αν καποιος ελεγχος αποτυχει, το πακετο απορριπτεται και στελνεται duplicate ACK.
Επειτα ελεγχεται ο αριθμος ακολουθιας (sequence number). Αν το πακετο δεν ειναι το αναμενομενο (out-of-order), δεν αποθηκευεται και αποστελλεται duplicate ACK.

Αν το πακετο ειναι σωστο, τα δεδομενα αποθηκευονται στο receive buffer του socket. Ενημερωνεται το fill level του buffer, το current window size και το επομενο αναμενομενο ACK number.
Στελνεται κανονικο ACK προς τον sender με ενημερωμενο window, υλοποιωντας flow control.
Στη συνεχεια δεδομενα προωθουνται απο το internal receive buffer προς το buffer της εφαρμογης, με σωστο χειρισμο μετακινησης (memmove) για τα υπολοιπα δεδομενα.
Η διαδικασια συνεχιζεται μεχρι να παραδοθουν στον χρηστη οσα bytes ζητηθηκαν ή μεχρι να ξεκινησει η διαδικασια τερματισμου της συνδεσης.
Τελος, αποδεσμευονται οι δεσμευμενοι ποροι και η συναρτηση επιστρεφει το συνολικο πληθος bytes που παραδοθηκαν επιτυχως στην εφαρμογη.

Bandwidth measurement

Για την αξιολογηση της αποδοσης της υλοποιησης, μετραται το bandwidth της μεταδοσης δεδομενων τοσο για το microTCP οσο και για την TCP υλοποιηση του λειτουργικου συστηματος.
Το bandwidth εκφραζει τον ρυθμο μεταφορας δεδομενων και υπολογιζεται ως το συνολικο πληθος bytes που μεταδοθηκαν δια του συνολικου χρονου μεταδοσης.
Ο χρονος μετρησης ξεκινα απο την αρχη της μεταφορας δεδομενων και ολοκληρωνεται με το τελος της συνδεσης. Με βαση αυτον τον χρονο και το συνολικο μεγεθος του αρχειου, υπολογιζεται το πραγματικο throughput της συνδεσης.
Η συγκριση μεταξυ TCP και microTCP επιτρεπει την αξιολογηση της συμπεριφορας της custom υλοποιησης ως προς την αξιοπιστια, τον ελεγχο ροης και τον ελεγχο συμφωρησης. Τυχον διαφορες στο bandwidth αντανακλουν το κοστος των μηχανισμων αξιοπιστιας και retransmissions που υλοποιουνται σε επιπεδο εφαρμογης.
Κατω απο αυτην την ενοτητα παρατιθενται οι χρονοι μεταδοσης για 500MB data:

TCP time: 3.376512 seconds
microTCP time: 28.522 seconds

TCP Throughput: 148.081812 MB/s
microTCP Throughput: 17.53 MB/s

Performance: TCP Throughput / microTCP Throughput = 148.08 / 17.53 = 8.446

Αρα το TCP ειναι περιπου 8.446 φορες ποιο γρηγορο απο το microTCP που φτιαξαμε.