Γιακουμής Γιώργος & Αναστασάς Αναστάσιος
csd3157 csd3166

MicroTCP protocol

create socket

Κατά την δημιουργία γίνεται έλεγχος στα argument που μας έχουν δοθεί δηλαδή αν είναι τα σωστά για TCP protocol. Αν δεν είναι επιστρέφουμε το socket με κατάσταση INVALID και sd -1. Επίσης αν δεν γίνει σωστά η δημιουργία του socket επιστρέφεται και πάλι sd με -1 και κατάσταση INVALID. Σε κάθε άλλη περίπτωση επιστρέφεται το socket με sd>0 και κατάσταση UNKNOWN.

bind socket

Εάν το bind είναι επιτυχημένο τότε επιστρέφεται αριθμός μεγαλύτερος το 0 αλλιώς -1.

3-way handshake

Κατά το **connect** γίνεται έλεγχος στο socket εάν το state είναι ESTABLISHED τότε απλά επιστρέφουμε το socket που πήραμε. Αλλιώς ξεκινάει η πρώτη επικοινωνία με τον server. Κατά την επικοινωνία αυτή θα καθοριστεί το max_win_size. Εάν γίνουν σωστές ανταλλαγές πακέτων τότε γίνεται το state ESTABLISHED και δεσμεύουμε μνήμη για τον recvbuf. Σε κάθε άλλη περίπτωση που συμβεί κάποιο λάθος κατά την αποστολή/λήψη πακέτων ή λάβουμε κάποιο λάθος πακέτο(λάθος control, λάθος checksum, λάθος ACK) τότε ορίζουμε νέα κατάσταση στο socket = INVALID και επιστρέφουμε το socket αυτό.

Κατά το **accept** γίνονται ανταλλαγές πακέτων και έλεγχοι για την ορθότητα των πακέτων. Εάν συμβεί κάποιο error στο checksum, ή το control, sequence number δεν είναι σωστό, τότε γίνεται return το socket με νέα κατάσταση INVALID.

microtcp_shutdown

Υπάρχουν δύο καταστάσεις σε αυτήν την συνάρτηση:

- Στην πρώτη κατάσταση περιμένουμε να λάβουμε το FIN,ACK και στην συνέχεια απαντάμε με ένα ACK. Σε αυτήν την περίπτωση μπαίνουμε είτε όταν έχει καλεστεί η συνάρτηση από τον server και δεν έχει λάβει το fin,ack κατά την microtcp_receive, είτε όταν η κατάσταση του socket είναι closing_by_host που σημαίνει ότι είμαστε από την μεριά του client επομένως περιμένουμε fin,ack και απαντάμε και ορίζουμε κατάλληλα το νέο state στο socket.
- Στη δεύτερη κατάσταση στέλνουμε το FIN,ACK και στην συνέχεια περιμένουμε να πάρουμε ACK. Εδώ μπαίνουμε είτε όταν είμαστε client και καλούμε για πρώτη φορά την συνάρτηση shutdown είτε όταν είμαστε server και έχουμε κατάσταση στο socket closing_by_peer. Αφού γίνουν όλες οι απαραίτητες αλλαγές στο socket τότε καλούμε ξανά την συνάρτηση αυτήν. Αυτό γίνεται γιατί εάν ήμασταν client θα έχουμε στείλει fin,ack και θα έχουμε λάβει ack, επομένως θα πρέπει να περιμένουμε να λάβουμε fin,ack από τον server και να στείλουμε ack και έτσι να κλείσουμε το socket. Αν ήμασταν server τότε απλά θα καλούσαμε την shutdown αλλά το state θα είναι CLOSED επομένως δεν μπαίνουμε σε κάποια από τις δύο καταστάσεις και έτσι επιστρέφουμε το ίδιο socket.

Γιακουμής Γιώργος & Αναστασάς Αναστάσιος
csd3157 csd3166

microtcp_recv

Εάν δεν έχουμε λάβει fin,ack τότε ορίζουμε ένα μικρό timeout και ξεκινάμε να λαμβάνουμε πακέτα. Εάν δεν συμβεί το timeout τότε κάνουμε έλεγχο για το checksum, ελέγχουμε το sequence number του πακέτου που λάβαμε και αν αυτό είναι αυτό που έπρεπε να λάβουμε με βάση το ack number που έχουμε στο socket. Εάν είναι σωστοί όλοι οι έλεγχοι τότε βάζουμε τα δεδομένα που λάβαμε στον recvbuf του socket μας. Απαντάμε με ένα ACK πακέτο και τότε εάν τα length του buffer που μας έδωσε ο server είναι μεγαλύτερος από το buf_fill_level που έχουμε για το socket μας περιμένουμε για νέο πακέτο. Όταν έχουμε στον buffer μας όσα δεδομένα μας ζήτησε ο χρήστης ή γίνει timeout στην recv τότε προωθούμε όσα δεδομένα μπορούμε στην εφαρμογή. Ακόμα εάν λάβουμε το fin,ack τότε αφού αδειάσουμε τον recvbuf και προωθήσουμε αυτά τα δεδομένα στην εφαρμογή τότε επιστρέφουμε -1 ώστε να τερματιστεί η microtcp_recv.

microtcp_send

Για κάθε νέο segment ελέγχουμε εάν πρέπει να αλλάξουμε από slow start σε congestion avoidance. Υπολογίζουμε το πλήθος των byte που πρέπει να στείλουμε και σε πόσα πακέτα θα τα στείλουμε. Ξεκινάμε να στέλνουμε back to back τα πακέτα που έχουμε υπολογίσει ότι πρέπει να στείλουμε. Στην συνέχεια ελέγχουμε το window size που έχουμε λάβει από το τελευταίο ack πακέτο. Αν είναι μηδέν τότε στέλνουμε κενά πακέτα και ελέγχουμε το window που λαμβάνουμε από το ACK πακέτο, αυτό επαναλαμβάνεται μέχρι να αυξηθεί το window αυτό. Στην συνέχεια για όσες αποστολές έχουμε κάνει περιμένουμε και απαντήσεις. Κάνουμε τους απαραίτητους ελέγχους στο πακέτο που λάβαμε και αν δεν πάει κάτι στραβά τότε ελέγχουμε αν το ACK που λάβαμε είναι αυτό που περιμέναμε. Αν ήταν τότε αλλάζουμε το min_ack(ελάχιστο αναμενόμενο ack) με βάση το επόμενο ack που θα πρέπει να έχει η επόμενη επιβεβαίωση. Αυξάνουμε το congestion window κατάλληλα αν χρειάζεται. Αν πάρουμε κάποιο ACK μεγαλύτερο κάνουμε παρόμοιες αλλαγές. Αν λάβουμε μικρότερο τότε αυξάνουμε έναν μετρητή που μετράει πόσα duplicate ack έχουμε λάβει. Τέλος ελέγχουμε αν έχουμε λάβει επιβεβαιώσεις για όσα πακέτα έχουμε στείλει. Αν έχει συμβεί κάποιο timeout(μπαίνουμε σε slow start) ή 3 duplicate ACK τότε ορίζουμε κατάλληλα τα remaining,data_sent με βάσει τα επιβεβαιωμένα δεδομένα που έχουν σταλθεί. Με αυτό τον τρόπο κάνουμε και retransmit εάν κάτι έχει πάει στραβά.

Άλλες συναρτήσεις:

convert_to_Network_order : μετατρέπουμε ένα header σε network order
convert_to_Local_order : μετατρέπουμε ένα header σε local order
init_header_packet: κάνουμε initialize ένα header με νέα δεδομένα
errorChecking: εξάγουμε από έναν buffer το header, υπολογίζουμε και ελέγχουμε εάν είναι σωστό το checksum του πακέτου.
init_socket: κάνουμε initialize ένα socket
min : υπολογίζουμε τον ελάχιστο αριθμό από τα 3 ορίσματα που λαμβάνει
wait_packet: κάνουμε recvfrom ελέγχουμε αν έχει ληφθεί σωστά και επιστρέφουμε το control του πακέτου.
set_timeout: ορίζουμε νέο timeout χρόνο για το socket μας

Γιακουμής Γιώργος & Αναστασάς Αναστάσιος
csd3157 csd3166

send_segment: κάνουμε αποστολή ενός segment

Performance = $\frac{15,719861}{90,800371}$ = 0.1731255150928844