

Ευαγγελία Χαμιλάκη AM: 4261

Θεοδώρα Σαμπάλου AM: 4306

## MICROTCP

Η υλοποίηση του πρότζεκτ γίνεται μεταξύ του `test_microtcp_server` και του `test_microtcp_client`.

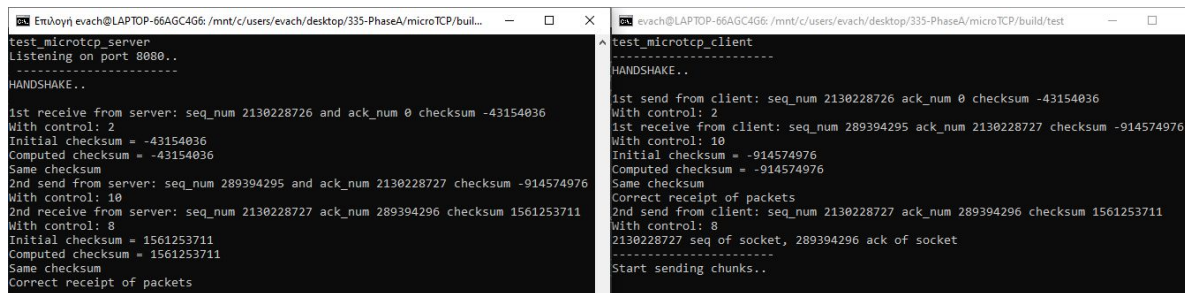
Δημιουργούμε στον `client` ένα `socket` και μια `address` για τον `client` όπου ορίζουμε το `port,family,address`. Έπειτα θέλουμε να γίνει η σύνδεση μεταξύ `client` και `server`, οπότε καλούμε την `microtcp_connect` με ορίσματα το `socket` που φτιάξαμε και το `address`.

Απο την μεριά του `server` δημιουργούμε επίσης ένα `socket` και ένα `address` για τον `server`. Πρέπει το `socket` να γίνει `bind` οπότε καλούμε την `microtcp_bind`. Αν αποτύχει το `bind` σταματάει το πρόγραμμα. Αν πετύχει καλούμε την `microtcp_accept` (με ορίσματα το `socket` και το `address` του `server` που μόλις φτιάχτηκε) ώστε να εγκαθιδρυθεί η σύνδεση με τον `client`.

### Υλοποίηση

`microtcp_connect`: Η τεχνική γίνεται με το 3-way handshake. Δημιουργούμε ένα header και εκεί του βάζουμε ως `sequence number` ένα τυχαίο αριθμό. Το `control` του γίνεται 2 με την βοήθεια μιας συνάρτησης που μετατρέπει το 2ο bit απ τα δεξιά σε 1 όπου αυτο είναι SYN. Του ορίζουμε το `window`, `checksum` (υπολογίζοντας το με την `crc32()`) και το στέλνουμε με την `microtcp_send_header`. Αφού λάβει το καινούριο header απο τον `server` κάνει τα κατάλληλα `error checking` ορίζει τις νέες τιμές στο header και το στέλνει στον `server` και το state γίνεται ESTABLISHED.

`microtcp_accept`: Μόλις ο `client` στείλει με την `microtcp_connect` θα καλεστεί η `microtcp_accept` η οποία κάνει `receive` το header με την `microtcp_recv_header` που στάλθηκε από τον `client`. Όταν το λάβει θα του ορίσει το `ack number` στην τιμή που πρέπει και το `sequence number` θα πάρει μια τυχαία μεταβλητή. Το `control` θα γίνει 10 τώρα αφού του ορίζουμε το ACK σε ένα και έχει απο προηγουμένως το SYN. Τώρα στέλνει πίσω στον `client` το καινούριο header με τις νέες τιμές. Η `accept` λαμβάνει το τελευταίο header που χρειάζεται για το 3-way handshake, κάνει τα κατάλληλα `error checkings` και ορίζει στο `socket` τις τελικές τιμές του header.



```
test_microtcp_server
Listening on port 8080..
.....
HANDSHAKE..
1st receive from server: seq_num 2130228726 and ack_num 0 checksum -43154036
With control: 2
Initial checksum = -43154036
Computed checksum = -43154036
Same checksum
2nd send from server: seq_num 289394295 and ack_num 2130228727 checksum -914574976
With control: 10
2nd receive from server: seq_num 2130228727 ack_num 289394296 checksum 1561253711
With control: 0
Initial checksum = 1561253711
Computed checksum = 1561253711
Same checksum
Correct receipt of packets
.....

test_microtcp_client
.....
HANDSHAKE..
1st send from client: seq_num 2130228726 ack_num 0 checksum -43154036
With control: 2
1st receive from client: seq_num 289394295 ack_num 2130228727 checksum -914574976
With control: 10
Initial checksum = -914574976
Computed checksum = -914574976
Same checksum
Correct receipt of packets
2nd send from client: seq_num 2130228727 ack_num 289394296 checksum 1561253711
With control: 0
2130228727 seq of socket, 289394296 ack of socket
.....
Start sending chunks..
.....
```

Για να κλείσει η σύνδεση μεταξύ του client και server χρειάζεται η `microtcp_shutdown` όπου καλείται και από τις δύο μεριές. Υποθέτουμε ότι πρώτα καλείται από τον client όπου αυτός θέλει να κλείσει την σύνδεση.

### Υλοποίηση

**`microtcp_shutdown`:** Ελέγχουμε σε ποιά κατάσταση είναι το socket αν δεν είναι `CLOSING_BY_PEER` σημαίνει ότι στέλνει πρώτη φορά ο client. Αφού το στείλει ο server με την σειρά του το λαμβάνει και στέλνει πίσω ένα ack με το sequence που έλαβε +1. Όταν το στείλει ο server θα θέσει σε κατάσταση του state σε `CLOSING_BY_PEER`. Άρα την επόμενη φορά που θα μπει θα είναι το state `CLOSING_BY_PEER`. Σε αυτήν την περίπτωση θα στείλει ο server ένα header με ACK και FIN και αλλαγμένες τις τιμές του header. Θα το κάνει με την σειρά του receive ο client και θα στείλει πίσω στον server ένα header με τις κατάλληλες τιμές και ο server θα το κάνει receive και το state θα γίνει closed. Για να επιτυχθεί αυτή η λειτουργία γίνονται πάντα error checkings για το αν στάλθηκαν με την σωστή σειρά.

```
data received: 10000
HOST ENTERS SHUTDOWN..

Correct receipt of sequence
1st send server: seq_num 2130228726 and ack_num 2130228727
With control: 8
2nd send server: seq_num 289394295 and ack_num 0
With control: 9
2nd rcv server: seq_num 2130228727 and ack_num 289394296
With control: 8
Correct receipt of packets
evach@LAPTOP-66AGC466:/mnt/c/users/evach/desktop/335-PhaseA/microTCP/build/test$

CLIENT SHUTDOWN..
1st send client: seq_num 2130228726 and ack_num 0
With control: 9
1st rcv client: seq_num 2130228726 and ack_num 2130228727
With control: 8
Correct receipt of sequence
2nd rcv client: seq_num 289394295 and ack_num 0
With control: 9
Correct receipt of sequence
2nd send client: seq_num 2130228727 and ack_num 289394296
With control: 8
evach@LAPTOP-66AGC466:/mnt/c/users/evach/desktop/335-PhaseA/microTCP/build/test$
```

### SEND DATA

Για να στέλνουμε δεδομένα απο τον client στον server θα χρειαστούμε μόνο μια `microtcp_send` στον client. Τα ορίσματα που παίρνει είναι ο πίνακας με τα δεδομένα , το μέγεθος των δεδομένων, το socket που χρησιμοποιούμε σε όλη την main και το flag.

**`microtcp_send`:** Εδώ έχουμε μία λούπα η οποία τρέχει μέχρι να σταλθούν όλα τα bytes που χρειάζονται. Τα bytes που θα σταλθούν είναι το ελάχιστο μεταξύ του flow και congestion control και των remaining. Έπειτα τα χωρίζουμε σε πακέτα μικρά (chunks) όπου το καθένα θα στέλνει τα bytes που πρέπει να σταλθούν/ maximum segment size. Στην περίπτωση που ένα chunk έχει λιγότερο απο το όριο αυξάνονται τα chunks κατά 1 και στέλνονται με την ίδια διαδικασία που στέλνονται και τα υπόλοιπα chunks απλά αυτή την φορά το tempSeq αυξάνεται κατά τα λιγότερα data που υπάρχουν σε αυτο το chunk. Για καθένα απο τα chunks ορίζουμε στον header που θα σταλθεί το data length που είναι τα ποσα bytes υπολογίστηκαν να σταλθούν, το sequence number που στην αρχή έχει το sequence που έχουμε βάσει προηγουμένως στο socket και το ίδιο ισχυεί για το ACK. Μετά αντιγράφουμε το header που θα σταλθεί στο data και στην θέση των data και μετα τον header θα προσθέσουμε και αυτά που έχει ο buffer και τα data που στάλθηκαν. Γίνονται οι κατάλληλοι έλεγχοι και αφού περάσουν του ορίζουμε το checksum και στέλνουμε τα data αυτή την φορά αντι για το header. Αν αποτύχει το state γίνεται invalid και τερματίζεται με error. Αν σταλθεί κανονικά το tempSeq (όπου μπαίνει ως τιμή στο header.sequence number) αυξάνεται κατά `MICROTCP_MSS`, το tempAck (όπου μπαίνει ως τιμή στο header.ack number) αυξάνεται κατά 1.

Ο έλεγχος για τα ACK γίνεται για κάθε chunk. Ελέγχουμε αν έχει γίνει timeout. Αν έχει γίνει το ssthresh του socket γίνεται το μισό του cwnd του socket και το cwnd γίνεται το ελάχιστο μεταξύ του MICROTCP\_MSS και του ssthresh και συνεχίζει. Ο client λαμβάνει το ack που του έστειλε ο server (έλεγχος αν δεν σταλθούν) και αυξάνουμε το acks που έχουν ληφθεί. Γίνονται τα κατάλληλα error checkings(checksum). Μετά γίνεται ο έλεγχος για το αν έχουμε λάβει duplicate ack. Αν του socket το sequence είναι διάφορο από το ack που λάβαμε αυξάνουμε τα duplicate acks. Αν γίνουν 3 τότε πρέπει να κάνουμε retransmission. Τότε το ssthresh γίνεται το μισό του cwnd και το cwnd γίνεται το μισό του εαυτού του αυξημένο κατά 1. Τα data που θα αρχίσουμε να στέλνουμε ξανά στην λούπα θα ξεκινήσουν από τα τελευταία data που στάλθηκαν σωστά. Θα μηδενιστεί η μεταβλητή που μετρούσε τα duplicate acks για να χρησιμοποιηθεί την επόμενη φορά που θα χάσουμε δεδομένα. Χρησιμοποιώντας ένα flag για να δούμε αν μπήκε στο retransmission μειώνουμε τα πόσα δεδομένα έχουν απομείνει μειώνοντας αυτά που στάλθηκαν σωστά και θέτουμε στο socket τις τιμές του τελευταίου segment που πρέπει να ξαναστείλουμε.

```
evach@LAPTOP-66AGC4G6: /mnt/c/users/evach/desktop/335-PhaseA/microTCP/build/test
Start sending chunks..
-----
Packet case 1
SENDING(1): Sending Sequence = 1316113291
SENDING(1): Sending Ack = 860500416
SENDING(1): Sending Checksum = 2103603887
SENDING(1): Sending 1432 bytes.
-----
Packet case 1
SENDING(1): Sending Sequence = 1316114691
SENDING(1): Sending Ack = 860500417
SENDING(1): Sending Checksum = 4229206599
SENDING(1): Sending 1432 bytes.
-----
Packet case 1
SENDING(1): Sending Sequence = 1316116091
SENDING(1): Sending Ack = 860500418
SENDING(1): Sending Checksum = 4021412018
SENDING(1): Sending 1432 bytes.
-----
Checking for ACK..
IN SEND: Received ACK with Seq = 860500416, Ack = 1316113291, Checksum = 272276059
Initial checksum = 272276059
Computed checksum = 272276059
Same checksum
sock seq = 1316113291
-----
IN SEND: Received DUP ACK(Num of Dup Acks = 1)
-----
Checking for ACK..
IN SEND: Received ACK with Seq = 860500416, Ack = 1316113291, Checksum = 272276059
Initial checksum = 272276059
Computed checksum = 272276059
Same checksum
sock seq = 1316113291
-----
IN SEND: Received DUP ACK(Num of Dup Acks = 2)
```

```
evach@LAPTOP-66AGC4G6: /mnt/c/users/evach/desktop/335-PhaseA/microTCP/build/test
IN SEND: Received ACK with Seq = 860500416, Ack = 1316113291, Checksum = 272276059
Initial checksum = 272276059
Computed checksum = 272276059
Same checksum
sock seq = 1316113291
-----
IN SEND: Received DUP ACK(Num of Dup Acks = 1)
-----
Checking for ACK..
-----
IN SEND: Received ACK with Seq = 860500416, Ack = 1316113291, Checksum = 272276059
Initial checksum = 272276059
Computed checksum = 272276059
Same checksum
sock seq = 1316113291
-----
IN SEND: Received DUP ACK(Num of Dup Acks = 2)
-----
Checking for ACK..
-----
IN SEND: Received ACK with Seq = 860500416, Ack = 1316113291, Checksum = 272276059
Initial checksum = 272276059
Computed checksum = 272276059
Same checksum
sock seq = 1316113291
-----
IN SEND: Received DUP ACK(Num of Dup Acks = 3)
-----
RETRANSMISSION
Retransmitting packet with:
    Sequence = 1316111891
    Acknowledge = 860500416
    Bytes Sent = 0
Packets sent are equal to Acks received!
In Congestion Avoidance..
    Control window = 3501
    Ssthresh = 2100
^^^^^^socket sequence number that i will retrnsmismit 1316111891
Final sent correctly sock seq = 1316111891, ack = 860500416 , remaning : 10000
-----
Start sending chunks..

evach@LAPTOP-66AGC4G6: /mnt/c/users/evach/desktop/335-PhaseA/microTCP/build/test
Start sending chunks..
-----
Packet case 1
    SENDING(1): Sending Sequence = 1316111891
    SENDING(1): Sending Ack = 860500416
    SENDING(1): Sending Checksum = 315456231
    SENDING(1): Sending 1432 bytes.
-----
Packet case 1
    SENDING(1): Sending Sequence = 1316113291
    SENDING(1): Sending Ack = 860500417
    SENDING(1): Sending Checksum = 1050922066
    SENDING(1): Sending 1432 bytes.
-----
Packet case 2
    SENDING(2): Sending Sequence = 1316114691
    SENDING(2): Sending Ack = 860500418
    SENDING(2): Sending Checksum = 3915365373
    SENDING(2): Sending 733 bytes.
-----
Checking for ACK..
-----
IN SEND: Received ACK with Seq = 860500416, Ack = 1316113291, Checksum = 272276059
Initial checksum = 272276059
Computed checksum = 272276059
Same checksum
sock seq = 1316111891
IN SEND: Received correct ACK.
ACK: In Congestion Avoidance..
    Control window = 3502
    Ssthresh = 2100
    Data sent correctly = 0
-----
Checking for ACK..
-----
IN SEND: Received ACK with Seq = 860500417, Ack = 1316114691, Checksum = 908737110
Initial checksum = 908737110
Computed checksum = 908737110
Same checksum
sock seq = 1316111891
```

Αν τώρα δεν έχουμε duplicate ack κάνουμε ελέγχους για το slow start πρώτα όπου αν έχουμε slow start τότε το cwnd αυξάνεται κατά MICROTCP\_MSS ενώ αν έχουμε congestion avoidance το cwnd αυξάνεται κατά 1. Η μεταβλητή για τα δεδομένα που στάλθηκαν σωστά αυξάνεται κατά το μέγεθος των δεδομένων που υπάρχει στον header που λήφθηκε. Στο τέλος του κάθε chunk που στέλνεται το current window size και το ack number και το flow και control window παίρνουν τις αντίστοιχες τιμές που έχει ο header που λήφθηκε. Αν μέχρι τώρα όλα πάνε καλά και όσα ack έχουμε λάβει είναι ίσα με τα πακέτα που στείλαμε κάνουμε έλεγχο για κάθε RTT. Όταν έχουμε slow start τότε διπλασιάζουμε το cwnd αλλιώς αυξάνουμε το cwnd κατά MICROTCP\_MSS. Τέλος χρησιμοποιώντας το flag που είχαμε για τα 3 duplicate acks μειωνουμε τα data που απομένουν με αυτά που θα σταλθούν και αυξάνουμε τα δεδομένα που στάλθηκαν κατά αυτά που θέλουμε να στείλουμε.

Εχουμε βάλει μια λούπα στον server η οποία όσο λαμβάνει κανονικά data τα αυξάνει.

microtcp\_recv: Υπάρχει μια `buffer_recv` μεταβλητή όπου εκεί αποθηκεύουμε τον buffer που έγινε receive. Αν αποτύχει το state γίνεται invalid και γυρνάει error. Υπολογίζουμε πόσα bytes λαβαμε αφαιρώντας το μέγεθος του header. Αυτά που λάβαμε τα αντιγράφουμε σε μια μεταβλητή `header_recv`. Γίνεται ο έλεγχος αν ο receive buffer έχει άλλο χώρο για να λάβει αυτά που στάλθηκαν και αν δεν έχει το window του γίνεται 0. Αν έχει χώρο αυξάνουμε το τελευταίο χώρο που είχε με αυτά που λάβαμε. Γίνονται έλεγχοι για το control και το checksum. Μετά έχουμε τον έλεγχο για τα duplicate acks. Αν δεν λάβουμε αυτο που περιμέναμε ή αν το checksum δεν είναι αυτο που θέλουμε. Τότε η `microtcp_receive` στέλνει έναν header με το ack αυτουνου που στάλθηκε τελευταία φορά σωστά. Αν δεν σταλθει duplicate ack στέλνουμε πίσω ένα header που έχει ack όσο είναι το sequence number και τα data που λήφθηκαν. Αφαιρούμε και απο το window τα data που λήφθηκαν και το data length του header αυξάνεται κατα data που λήφθηκαν. Το socket παίρνει τις τιμές του header που θα σταλθεί και μειώνεται και το window size του ανάλογα με το πόσο γέμισε ο buffer του. Στέλνουμε τον header και επιστρέφουμε τα data που λήφθηκαν σωστά.

```
Επιλογή evach@LAPTOP-66AGC4G6: /mnt/c/users/evach/desktop/335-PhaseA/microTCP/build...
Sending ACK..
header data length 1400
SEQ NUMBER = 289394296 ACK NUMBER = 2130230127, CONTROL NUMBER = 8
Socket seq = 289394296, ack = 2130230127
IN RECEIVE: Sending Checksum = 799984369
data1 received : 1400
.....
Receiving chunks..
IN RECEIVE: 1432 bytes received and data received : 1400
IN RECEIVE: Seq = 2130230127, Ack = 289394297
Received in receive: checksum= 4196838265
Computed in receive: checksum= 4196838265
CRASH: x = 1
Header sequence 2130230127 , socket->ack 2130230127
.....
Sending ACK..
header data length 1400
SEQ NUMBER = 289394297 ACK NUMBER = 2130231527, CONTROL NUMBER = 8
Socket seq = 289394297, ack = 2130231527
IN RECEIVE: Sending Checksum = 940377232
data1 received : 1400
.....
Receiving chunks..
IN RECEIVE: 1432 bytes received and data received : 1400
IN RECEIVE: Seq = 2130231527, Ack = 289394298
Received in receive: checksum= 2923549396
Computed in receive: checksum= 2923549396
CRASH: x = 0
Header sequence 2130231527 , socket->ack 2130231527
.....
Sending ACK..
header data length 1400
SEQ NUMBER = 289394298 ACK NUMBER = 2130232927, CONTROL NUMBER = 8
Socket seq = 289394298, ack = 2130232927
IN RECEIVE: Sending Checksum = 3086632432
data1 received : 1400
.....
Receiving chunks..
IN RECEIVE: 1432 bytes received and data received : 1400

evach@LAPTOP-66AGC4G6: /mnt/c/users/evach/desktop/335-PhaseA/microTCP/build/test
Checking for ACK..
IN SEND: Received ACK with Seq = 289394296, Ack = 2130230127, Checksum = 799984369
Initial checksum = 799984369
Computed checksum = 799984369
Same checksum
sock seq = 2130228727
IN SEND: Received correct ACK.
ACK: In Slow Start..
Control Window = 5600
Ssthresh = 8192
Data sent correctly = 1400
.....
Checking for ACK..
IN SEND: Received ACK with Seq = 289394297, Ack = 2130231527, Checksum = 940377232
Initial checksum = 940377232
Computed checksum = 940377232
Same checksum
sock seq = 2130228727
IN SEND: Received correct ACK.
ACK: In Slow Start..
Control Window = 7000
Ssthresh = 8192
Data sent correctly = 2800
.....
Checking for ACK..
IN SEND: Received ACK with Seq = 289394298, Ack = 2130232927, Checksum = 3086632432
Initial checksum = -1208334864
Computed checksum = -1208334864
Same checksum
sock seq = 2130228727
IN SEND: Received correct ACK.
ACK: In Slow Start..
Control Window = 8400
Ssthresh = 8192
Data sent correctly = 4200
Packets sent are equal to Acks received!
```



```
-----
Receiving chunks..

IN RECEIVE: 1432 bytes received and data received : 1400
IN RECEIVE: Seq = 2130232927, Ack = 289394299
Received in receive: checksum= 263993073
Computed in receive: checksum= 263993073
CRASH: x = 1
Header sequence 2130232927 , socket->ack 2130232927
Sending ACK..

Header data length 1400
SEQ NUMBER = 289394299 ACK NUMBER = 2130234327, CONTROL NUMBER = 8
Socket seq = 289394299, ack = 2130234327
IN RECEIVE: Sending Checksum = 1547600060
data1 received : 1400
-----
Receiving chunks..

IN RECEIVE: 1432 bytes received and data received : 1400
IN RECEIVE: Seq = 2130234327, Ack = 289394300
Received in receive: checksum= 2318248715
Computed in receive: checksum= 2318248715
CRASH: x = 1
Header sequence 2130234327 , socket->ack 2130234327
Sending ACK..

Header data length 1400
SEQ NUMBER = 289394300 ACK NUMBER = 2130235727, CONTROL NUMBER = 8
Socket seq = 289394300, ack = 2130235727
IN RECEIVE: Sending Checksum = 1766551244
data1 received : 1400
-----
Receiving chunks..

IN RECEIVE: 1432 bytes received and data received : 1400
IN RECEIVE: Seq = 2130235727, Ack = 289394301
Received in receive: checksum= 2400239998
Computed in receive: checksum= 2400239998
CRASH: x = 0
Header sequence 2130235727 , socket->ack 2130235727
Sending ACK..

Packets sent are equal to Acks received!

In Congestion Avoidance..
Control Window = 9800
Ssthresh = 8192

Final socket->seq = 2130232927
Start sending chunks..

-----
Packet case 1
SENDING(1): Sending Sequence = 2130232927
SENDING(1): Sending Ack = 289394299
SENDING(1): Sending Checksum = 263993073
SENDING(1): Sending 1432 bytes.

-----
Packet case 1
SENDING(1): Sending Sequence = 2130234327
SENDING(1): Sending Ack = 289394300
SENDING(1): Sending Checksum = 2318248715
SENDING(1): Sending 1432 bytes.

-----
Packet case 1
SENDING(1): Sending Sequence = 2130235727
SENDING(1): Sending Ack = 289394301
SENDING(1): Sending Checksum = 2400239998
SENDING(1): Sending 1432 bytes.

-----
Packet case 1
SENDING(1): Sending Sequence = 2130237127
SENDING(1): Sending Ack = 289394302
SENDING(1): Sending Checksum = 1323256756
SENDING(1): Sending 1432 bytes.

-----
Packet case 2
SENDING(2): Sending Sequence = 2130238527
SENDING(2): Sending Ack = 289394303
SENDING(2): Sending Checksum = 2283203170
SENDING(2): Sending 232 bytes.
```

```
Header sequence 2130235727 , socket->ack 2130235727
Sending ACK..

Header data length 1400
SEQ NUMBER = 289394301 ACK NUMBER = 2130237127, CONTROL NUMBER = 8
Socket seq = 289394301, ack = 2130237127
IN RECEIVE: Sending Checksum = 2635418504
data1 received : 1400
-----
Receiving chunks..

IN RECEIVE: 1432 bytes received and data received : 1400
IN RECEIVE: Seq = 2130237127, Ack = 289394302
Received in receive: checksum= 1323256756
Computed in receive: checksum= 1323256756
CRASH: x = 0
Header sequence 2130237127 , socket->ack 2130237127
Sending ACK..

Header data length 1400
SEQ NUMBER = 289394302 ACK NUMBER = 2130238527, CONTROL NUMBER = 8
Socket seq = 289394302, ack = 2130238527
IN RECEIVE: Sending Checksum = 3697209447
data1 received : 1400
-----
Receiving chunks..

IN RECEIVE: 232 bytes received and data received : 200
IN RECEIVE: Seq = 2130238527, Ack = 289394303
Received in receive: checksum= 2283203170
Computed in receive: checksum= 2283203170
CRASH: x = 0
Header sequence 2130238527 , socket->ack 2130238527
Sending ACK..

Header data length 200
SEQ NUMBER = 289394303 ACK NUMBER = 2130238727, CONTROL NUMBER = 8
Socket seq = 289394303, ack = 2130238727
IN RECEIVE: Sending Checksum = 799038660
data1 received : 200
-----
Receiving chunks..

Checking for ACK..

IN SEND: Received ACK with Seq = 289394299, Ack = 2130234327, Checksum = 1547600060
Initial checksum = 1547600060
Computed checksum = 1547600060
Same checksum
sock seq = 2130232927
IN SEND: Received correct ACK.

ACK: In Congestion Avoidance..
Control Window = 9801
Ssthresh = 8192
Data sent correctly = 5600

-----
Checking for ACK..

IN SEND: Received ACK with Seq = 289394300, Ack = 2130235727, Checksum = 1766551244
Initial checksum = 1766551244
Computed checksum = 1766551244
Same checksum
sock seq = 2130232927
IN SEND: Received correct ACK.

ACK: In Congestion Avoidance..
Control Window = 9802
Ssthresh = 8192
Data sent correctly = 7000

-----
Checking for ACK..

IN SEND: Received ACK with Seq = 289394301, Ack = 2130237127, Checksum = 2635418504
Initial checksum = -1659548792
Computed checksum = -1659548792
Same checksum
sock seq = 2130232927
IN SEND: Received correct ACK.

ACK: In Congestion Avoidance..
Control Window = 9803
Ssthresh = 8192
Data sent correctly = 8400
```

```
Enukoyj evach@LAPTOP-66AGC4G6: /mnt/c/users/evach/desktop/335-PhaseA/microTCP/build...
Sending ACK..
header data length 1400
SEQ NUMBER = 289394302 ACK NUMBER = 2130238527, CONTROL NUMBER = 8
Socket seq = 289394302, ack = 2130238527
IN RECEIVE: Sending Checksum = 3697209447
data1 received : 1400
Receiving chunks..
IN RECEIVE: 232 bytes received and data received : 200
IN RECEIVE: Seq = 2130238527, Ack = 289394303
Received in receive: checksum= 2283203170
Computed in receive: checksum= 2283203170
CRASH: x = 0
Header sequence 2130238527 , socket->ack 2130238527
Sending ACK..
header data length 200
SEQ NUMBER = 289394303 ACK NUMBER = 2130238727, CONTROL NUMBER = 8
Socket seq = 289394303, ack = 2130238727
IN RECEIVE: Sending Checksum = 799038660
data1 received : 200
Receiving chunks..
IN RECEIVE: 32 bytes received and data received : 0
1st rcv server: seq_num 2130228726, ack_num 0
With control 9
data received : 10000

evach@LAPTOP-66AGC4G6: /mnt/c/users/evach/desktop/335-PhaseA/microTCP/build/test
Checking for ACK..
IN SEND: Received ACK with Seq = 289394302, Ack = 2130238527, Checksum = 3697209447
Initial checksum = -597757849
Computed checksum = -597757849
Same checksum
sock seq = 2130232927
IN SEND: Received correct ACK.
ACK: In Congestion Avoidance..
Control Window = 9804
Ssthresh = 8192
Data sent correctly = 9800
Checking for ACK..
IN SEND: Received ACK with Seq = 289394303, Ack = 2130238727, Checksum = 799038660
Initial checksum = 799038660
Computed checksum = 799038660
Same checksum
sock seq = 2130232927
IN SEND: Received DUP ACK(Num of Dup Acks = 1)
Packets sent are equal to Acks received!
In Congestion Avoidance..
Control Window = 11204
Ssthresh = 8192
Final socket->seq = 2130237327
Message sent from client.
```