Γενική περιγραφή

Στο οικοσύστημα του Internet of Thinks (IoT) υπάρχει μια ευρεία γκάμα από δικτυακές συσκευές. Πολλές από αυτές έχουν ελάχιστο μέγεθος και λειτουργούν με μπαταρίες. Για αυτόν τον λόγο, οι υπολογιστικοί τους πόροι είναι αρκετά περιορισμένοι. Στις περισσότερες περιπτώσεις, η υλοποίηση ολόκληρου του TCP πρωτοκόλλου είναι αδύνατη και η μεταφορά της πληροφορίας γίνεται με UDP. Υπάρχουν όμως εφαρμογές που απαιτούν αξιόπιστη επικοινωνία και μεταφορά δεδομένων.

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.

1. Δομή του project - Προθεσμίες

Το project αποτελείται από **2 φάσεις**.

Στην πρώτη φάση ζητείται να υλοποιήσετε την δομή των microTCP πακέτων, το 3-way handshake (χειραψία) και το finalize (τερματισμό) της microTCP σύνδεσης. Επίσης, θα υλοποιήσετε ενα εργαλείο μεταφοράς αρχείων με TCP το οποίο θα χρησιμοποιήσετε στο τέλος για την σύγκριση της απόδοσης της υλοποίησή σας, τόσο σε σχέση με την πλήρη υλοποίηση του TCP, όσο και με υλοποιήσεις άλλων ομάδων.

Στην δεύτερη φάση θα πρέπει να υλοποιήσετε την βασική λειτουργία του TCP η οποία περιλαμβάνει τους μηχανισμούς για acknowledgements, retransmissions, error checking, TCP windowing, Congestion control και Slow start.

Η προθεσμία παράδοσης της πρώτης φάσης είναι **05 Δεκεμβρίου 23:59**.

Απαγορεύεται να αλλάξετε τα συμβόλαια των συναρτήσεων που σας έχουν δοθεί και το structure του header που σας δίνονται. Η αλλαγή τους θα σημαίνει και τον μηδενισμό του project. Παρόλα αυτά είναι επιθυμητές οι αλλαγές/προσθήκες στα υπόλοιπα structs.

Αναλυτική περιγραφή

1.1 Δομή Κώδικα

Στο https://github.com/surligas/microTCP θα βρείτε ένα μέρος έτοιμου κώδικα που θα πρέπει να χρησιμοποιήσετε. Υπάρχουν σχόλια στον κώδικα που αναφέρουν που ακριβώς πρέπει να κάνετε αλλαγές με δικό σας κώδικα.

1. **Φάκελος *lib*:** Περιέχει την δομή και τις συναρτήσεις της microTCP βιβλιοθήκης.
2. **Φάκελος *test*:** Περιέχει το εργαλείο μέτρησης του bandwidth τόσο με χρήση της κανονικής TCP υλοποίησης όσο και της microTCP.
3. **Φάκελος *utils*:** Περιέχει χρήσιμες συναρτήσεις. Σε αυτή την φάση περιέχει την συνάρτηση υπολογισμού του checksum.

Για το compile και την παραγωγή του εκτελέσιμου μπορείτε να χρησιμοποιήσετε το *cmake* (https://cmake.org/). Το εργαλείο αυτό παράγει αυτόματα τα κατάλληλα makefiles σε ένα ξεχωριστό φάκελο ώστε ο φάκελος με τον πηγαίο κώδικα να παραμένει καθαρός από ενδιάμεσα αρχεία. Για να γίνει αυτό μέσα στο φάκελο με τον source code δημιουργείστε ένα φάκελο με το όνομα *build*. Μέσα σε αυτόν το φάκελο τρέξτε την εντολή *cmake* όπως φαίνεται στο παρακάτω παράδειγμα:

```
mkdir build
cd build
cmake ..
```

Μετά από αυτό μπορείτε να κάνετε compile δίνοντας την εντολή:

```
make
```

1.2 Δομή microTCP πακέτων

Η δομή κάθε microTCP πακέτου βασίζεται αρκετά στην δομή των TCP πακέτων με την διαφορά ότι ο header κάθε microTCP πακέτου είναι πιο απλός και σταθερού μεγέθους. Ο header έχει την εξής μορφή:

1. **Sequence Number, 32bit:** Το πεδίο αυτό περιέχει το sequence number του πακέτου. Αρχικά κατά την έναρξη μιας microTCP σύνδεσης επιλέγεται ένας τυχαίος αριθμός. Μετά από κάθε επιτυχημένη μετάδοση, ο αριθμός αυτός αυξάνεται κατά τον αριθμό των data bytes που στάλθηκαν επιτυχημένα.

2. **Acknowledgment Number, 32bit:** Το πεδίο αυτό περιέχει το sequence number που περιμένει ο αποστολέας να λάβει στο επόμενο πακέτο.

Bit-0		Bit-16		Bit-31
Sequence Number				
ACK Number				
Control			**Window**	
Data Length				
Future use field 0				
Future use field 1				
Future use field 2				
CRC32 Checksum				

3. **Control, 16bit:** Σε αυτό το πεδίο περιέχεται η πληροφορία για τον τύπο του κάθε πακέτου καθώς και κάποια flags που θα χρησιμοποιηθούν κυρίως στην επόμενη φάση.

Bit 0	Bit-12	Bit-13	Bit 14	Bit-15
0	**ACK**	**RST**	**SYN**	**FIN**

   - **ACK:** Αν είναι 1, το Acknowledgment Number θα πρέπει να ληφθεί υπόψιν. Στην ουσία μόνο το πρώτο πακέτο μιας TCP σύνδεσης δεν κάνει 1 αυτό το bit. Μετά το πρώτο SYN πακέτο, όλα τα υπόλοιπα θα πρέπει να έχουν το ACK bit 1.
   - **RST:** Αν είναι 1, η σύνδεση θα πρέπει να γίνει reset.
   - **SYN:** Synchronize sequence numbers. Χρησιμοποιείται για να δηλώσει την έναρξη μιας νέας σύνδεσης. Θα πρέπει να σταλθεί 1, μόνο μια φορά από κάθε πλευρά κατά το πρώτο πακέτο. Στις υπόλοιπες περιπτώσεις θα πρέπει να είναι 0.
   - **FIN:** Αν είναι 1, δηλώνει πως ο αποστολέας δεν έχει άλλα δεδομένα να στείλει. Χρησιμοποιείται για τον τερματισμό μιας σύνδεσης.

4. **Window 16bit:** Το window ορίζει τον αριθμό των bytes που είναι πρόθυμος να λάβει ο αποστολέας αυτού του πακέτου.

5. **Data Length, 32bit:** Το μέγεθος των δεδομένων του τρέχοντος πακέτου σε bytes. Προσοχή σε αυτόν τον αριθμό δεν θα πρέπει να προσμετρούνται τα bytes του header.

6. **Future use 0, 32bit:** Θα χρησιμοποιηθεί στην επόμενη φάση. Σε αυτή τη φάση θα πρέπει να είναι 0.

7. **Future use 1, 32bit:** Θα χρησιμοποιηθεί στην επόμενη φάση. Σε αυτή τη φάση θα πρέπει να είναι 0.

8. **Future use 2, 32bit:** Θα χρησιμοποιηθεί στην επόμενη φάση. Σε αυτή τη φάση θα πρέπει να είναι 0.

9. **CRC32 Checksum 32bit:** Περιέχει το CRC32 checksum του header και των δεδομένων του κάθε πακέτου. Για τον υπολογισμό του CRC32, υπάρχει υλοποιημένη η συνάρτηση *crc32(const uint8_t *buf, size_t len)* στο αρχείο *utils/crc32.h*.

   Ο υπολογισμός του checksum γίνεται ως εξής:

   - Όλα τα πεδία του header συμπληρώνονται με την κατάλληλη πληροφορία και τα δεδομένα τοποθετούνται ακριβώς μετά τον header.

   - Η μνήμη των πεδίων που δεν χρησιμοποιούνται καθώς και τα 32-bits του CRC32 checksum πεδίου θα πρέπει να είναι 0.

   - Η συνάρτηση *crc32()* εφαρμόζεται από το πρώτο byte του πακέτου μέχρι το τελευταίο συμπεριλαμβανομένων των data bytes.

   - Τα 32-bits που επιστρέφει η συνάρτηση τοποθετούνται κατάλληλα στο αντίστοιχο πεδίο.

   Κατά το receive, ο receiver εξάγει το checksum και το αποθηκεύει κατάλληλα τοπικά. Στο checksum πεδίο τοποθετεί μηδενικά bits και εφαρμόζει την ίδια *crc32()* συνάρτηση όμως ακριβώς έκανε ο sender. Αν το checksum πεδίο που λήφθηκε είναι ίδιο με αυτό που υπολογίστηκε από την συνάρτηση *crc32()*, το πακέτο περιέχει σωστά δεδομένα και μπορεί να συνεχιστεί η επεξεργασία του. Διαφορετικά το πακέτο θεωρείται κατεστραμμένο.

2. Ζητούμενα Α' Φάσης

Στο αρχείο *lib/microtcp.c* θα βρείτε μια σειρά από συναρτήσεις που θα πρέπει να υλοποιήσετε. Η ονοματολογία τους σχετίζεται αρκετά με τις αντίστοιχες συναρτήσεις του TCP και η λειτουργικότητά τους θα πρέπει να είναι αντίστοιχη.

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

  Η συνάρτηση αυτή δημιουργεί και επιστρέφει ένα socket microTCP socket σε αντιστοιχία με την συνάρτηση *socket()*. Με την δημιουργία του socket
  ```

το state της microTCP σύνδεσης θα πρέπει να ορίζεται ως *UKNOWN*. Σε περίπτωση λάθους κατάλληλο πεδίο του *microtcp_sock_t* θα πρέπει να ορίζεται κατάλληλα.

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

  Η συνάρτηση αυτή θα πρέπει να υλοποιεί ακριβώς την ίδια λειτουργικότητα με την *bind()*.

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

  Η συνάρτηση αυτή μπλοκάρει έως ότου ένας client συνδεθεί στο microTCP socket που έχει ανοίξει ο server. Η *microtcp_accept()* επιστρέφει το socket που πήρε ως παράμετρο, με την διαφορά ότι το πεδίο state θα πρέπει να έχει την τιμή *ESTABLISHED*. Μετά την επιτυχημένη κλήση της συνάρτησης αυτής, στην επόμενη φάση ο client και ο server μπορούν να ανταλλάξουν δεδομένα.

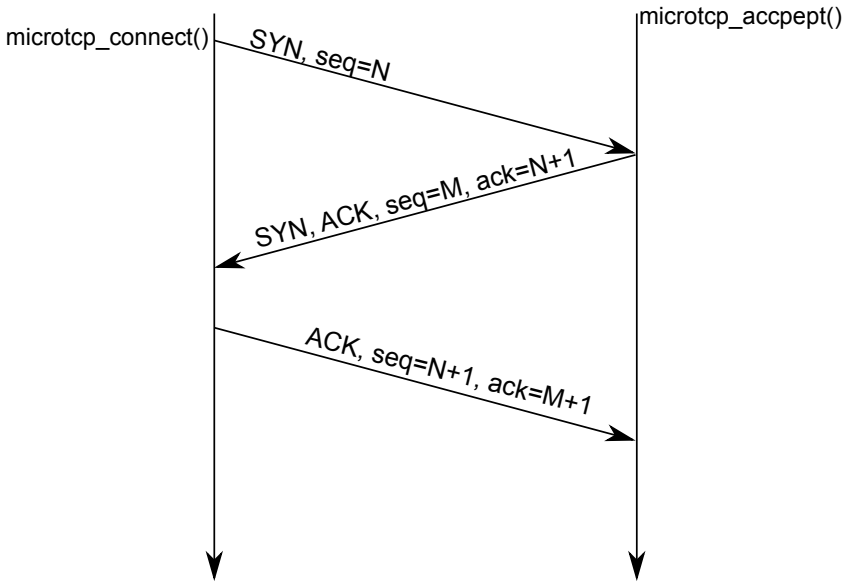
- ```
  int
  microtcp_connect(microtcp_sock_t *socket
  ,                const struct sockaddr *address,
                   socklen_t address_len);
  ```

  Η *microtcp_connect()* συνδέεται σε ένα microTCP socket το οποίο περιμένει για συνδέσεις έχοντας καλέσει την *microtcp_accept()*. Η εγκαθίδρυση της σύνδεσης επιτυγχάνεται χρησιμοποιώντας την τεχνική 3-way handshake που περιγράφεται παρακάτω. Η *microtcp_connect()* επιστρέφει το socket που πήρε ως παράμετρο, με την διαφορά ότι το πεδίο state θα πρέπει να έχει την τιμή *ESTABLISHED*. Σε περίπτωση λάθους κατάλληλο πεδίο του *microtcp_sock_t* θα πρέπει να ορίζεται κατάλληλα.

- ```
  int
  microtcp_shutdown(microtcp_sock_t *socket, int how)
  ```

- Η συνάρτηση αυτή τερματίζει την σύνδεση μεταξύ server και client επιστρέφοντας το socket που πήρε ως παράμετρο, με τροποποιημένο κατάλληλα το πεδίο state. Η τεχνική για τον τερματισμό της σύνδεσης περιγράφεται παρακάτω αναλυτικά.

2.1 3-Way handshake

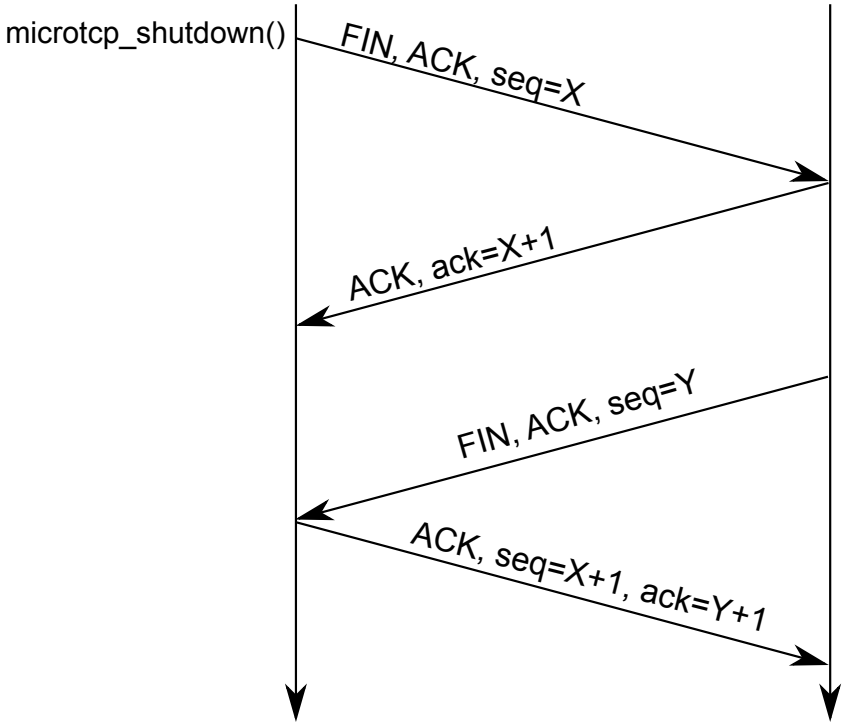


Με το 3-way handshake (χειραψία) γίνεται η εγκαθίδρυση της σύνδεσης. Για να γίνει αυτό ο client και ο server πρέπει να ανταλλάξουν πακέτα με τα κατάλληλα sequence numbers.

Το πρώτο πακέτο που στέλνει ο client είναι το SYN (Synchronize) πακέτο, το οποίο περιέχει το sequence number N που διάλεξε τυχαία. Όπως περιγράφτηκε και προηγουμένως, το SYN πακέτο δηλώνεται θέτοντας σε 1 το κατάλληλο bit στο control πεδίου του header. Ο server λαμβάνει το SYN πακέτο και αποθηκεύει το sequence number του client. Έπειτα διαλέγει τυχαία ένα δικό του sequence number M, και στέλνει ένα πακέτο SYN+ACK πίσω στον client το οποίο περιέχει στο πεδίο Sequence Number τον αριθμό M και στο πεδίο ACK Number τον αριθμό N+1. Το SYN+ACK πακέτο δηλώνεται θέτοντας σε 1 τα αντίστοιχα bits στο control πεδίου του header. Ο client λαμβάνει το SYN+ACK πακέτο, αποθηκεύει το sequence number M του server και στέλνει ένα ACK με ACK number M+1. Με το τέλος της διαδικασίας η σύνδεση θεωρείται εγκαθιδρυμένη.

Σημείωση: Κατά την διάρκεια της χειραψίας, υπάρχει πάντα η περίπτωση κάποιο πακέτο να χαθεί ή να περιέχει λάθος δεδομένα. Ο έλεγχος τέτοιων περιπτώσεων θα γίνει στην επόμενη φάση.

2.2 Connection termination



Για τον τερματισμό μιας microTCP σύνδεσης ακολουθείται μια παρόμοια διαδικασία με το 3-way handshake. Αρχικά ο client στέλνει ένα FIN πακέτο, δηλώνοντας στον server πως επιθυμεί την διακοπή της σύνδεσης. Ο server απαντάει στο FIN πακέτο με ένα ACK και θέτει την σύνδεση σε state *CLOSING_BY_PEER*. Μόλις ο client λάβει το ACK, θέτει την σύνδεση σε state *CLOSING_BY_HOST*. Ο Server εκτελεί όποιες λειτουργίες έχουν απομείνει και στέλνει ένα FIN πακέτο στον client. Μόλις λάβει το FIN πακέτο ο client, απαντάει στον server με ένα ACK και θέτει το state της σύνδεσής του σε *CLOSED*. Όταν αντίστοιχα λάβει ο server το ACK, θέτει το state της σύνδεσής του και αυτός σε CLOSED.

Σημείωση: Για λόγους απλότητας μπορείτε να θεωρήσετε ότι μόνο ο client μπορεί να ξεκινήσει την διαδικασία τερματισμού της σύνδεσης.

2.3 Bandwidth Test

Σαν μέρος του project είναι και η υλοποίηση ενός εργαλείου μέτρησης του bandwidth που επιτυγχάνεται μεταξύ ενός server και ενός client τόσο με την χρήση του microTCP όσο και με το κανονικό TCP.

Στο αρχείο *test/bandwidth_test.c* θα πρέπει να υλοποιήσετε τις απαραίτητες συναρτήσεις

που ζητούνται. Το πρόγραμμα παίρνει ως παράμετρο ένα όνομα αρχείου, το πρωτόκολλο μεταφορά (TCP ή microTCP) καθώς και αν θα λειτουργήσει ως server ή client. Ο client θα πρέπει να στείλει σωστά το αρχείο στον server και στο τέλος να εκτυπώνει την διάρκεια της μεταφοράς και το συνολικό bandwidth που επιτεύχθηκε.

Σημείωση: Η πλήρης υλοποίηση του εργαλείου δεν είναι απαραίτητη για την πρώτη φάση, αλλά συνιστάται ιδιαίτερα να υλοποιήσετε το μεγαλύτερό του μέρος ώστε να είστε σε θέση να λάβετε feedback στην συμβουλευτική εξέταση της πρώτης φάσης.