**Τμήμα Ψηφιακών Συστημάτων**



Ακαδημαϊκό Έτος: 2020-2021

Μάθημα: Πρωτόκολλα Διαδικτύου

3η Εργαστηριακή Άσκηση

Πανώριος Μιχαήλ Ε18127

Email: [michaelpanorios@gmail.com](mailto:michaelpanorios@gmail.com)

**Να φτιάξετε ένα πρωτόκολλο επικοινωνίας μεταξύ ενός server και ενός client.**

Το πρωτόκολλο που επέλεξα να φτιάξω βασίζεται στην επικοινωνία μεταξύ του server και του client. Για αυτό τον λόγο έφτιαξα δύο python scripts TCPHeaderClient και TCPHeaderServer. Ο server δέχεται ως δεδομένα δύο αριθμούς number1 & number2 από το 0 εως το 30.000 καθώς και ένα string που θα είναι ο operator ανάλογα την πράξη που επιθυμούμε να κάνουμε. Η δομή του μηνύματος που θα ανταλλάσεται:

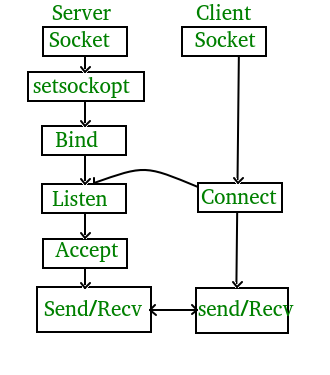
# 0------------16-------------------31  
# | Type | Length1 , Length2 | # type:2 byte, length1:1byte, length2:1byte  
# 0-------------+-------------------31  
# | operator | # operator: 1 byte, padding 3 bytes  
# 0---------------------------------31  
# | number1(+pad) | # number1: 4 byte  
# 0---------------------------------31  
# | number2(+pad) | # number2: 4 byte  
# 0---------------------------------31

Ο client στέλνει τα παρακάτω στοιχεία με bytes:

* Type: τύπος του header με μια σταθερή τιμή 0 που θα καταλαμβάνει 2 bytes (σταθερός αριθμός 0) (unsigned short)
* Operator: μια τιμή τύπου string που προσδιορίζει την πράξη καταλαμβάνει 1 byte (τιμή string +,-,/,\*)
* Length1: μήκος πρώτου αριθμού που καταλαμβάνει 1 byte (unsigned char)
* Length2: μήκος δεύτερου αριθμού που καταλαμβάνει 1 byte (unsigned char)
* Number1: ο πρώτος αριθμός που εισάγουμε που καταλαμβάνει 4 byte + padding αν χρειαστεί (unsigned int)
* Number2: ο δεύτερος αριθμός που εισάγουμε που καταλαμβάνει 4 byte + padding αν χρειαστεί (unsigned int)

Συνεπώς διαμορφώνεται και το μήνυμα για το packing & unpacking ως εξής: ***HBBs3x***

**Γενικά για TCP επικοινωνία:**

Εάν δημιουργούμε σύνδεση μεταξύ πελάτη και διακομιστή χρησιμοποιώντας TCP, τότε έχει λίγες λειτουργίες όπως, το TCP είναι κατάλληλο για εφαρμογές που απαιτούν υψηλή αξιοπιστία και ο χρόνος μετάδοσης είναι σχετικά λιγότερο κρίσιμος. Χρησιμοποιείται από άλλα πρωτόκολλα όπως HTTP, HTTPs, FTP, SMTP, Telnet. Το TCP αναδιατάσσει τα πακέτα δεδομένων με την καθορισμένη σειρά. Υπάρχει απόλυτη εγγύηση ότι τα δεδομένα που μεταφέρονται παραμένουν άθικτα και φτάνουν στην ίδια σειρά με την οποία στάλθηκαν. Το TCP κάνει Flow Control και απαιτεί τρία πακέτα για τη ρύθμιση μιας σύνδεσης υποδοχής, πριν από την αποστολή δεδομένων χρήστη. Το TCP χειρίζεται την αξιοπιστία και τον έλεγχο συμφόρησης. Κάνει επίσης έλεγχο σφαλμάτων και ανάκτηση σφαλμάτων. Τα λανθασμένα πακέτα αναμεταδίδονται από την πηγή στον προορισμό. Η όλη διαδικασία μπορεί να χωριστεί σε βήματα που εμφανίζονται στην εικόνα.

Ανάλυση του κώδικα:

Ξεκινώντας την ανάλυση του κώδικα ξεκινάω από την μεριά του Server. Αρχικά ορίζω την IP όπου το host θα ακούει και στην συνέχεια την πόρτα μαζί με ένα flag close που προσδιορίζει αν η διαδικασία έχει τελειώσει. Για αυτόν τον λόγο την αρχικοποιώ με false.

serverIP = '127.0.0.1'

serverPort = 1000  
close = False

Στην συνέχεια ορίζω το socketing που θα γίνει μεταξύ client και server. Αφόυ ολοκληρωθεί η γεφύρωση της επικοινωνίας εμφανίζω μήνυμα κατάφασης πως ο server είναι έτοιμος για επικοινωνία στην πόρτα που όρισα.

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as serverSocket:

serverSocket.bind((serverIP, serverPort))  
print ("The server is ready to receive at port", str(serverPort))

Με την παρακάτω εντολή μεταφερόμαστε στην μεριά του client αφού εκεί θα γίνει η διαδικασία προετιμασίας του μηνύματος.

serverSocket.listen()

Και στο αρχείο του client οι πρώτες εντολές ακούν στην διεύθυνση και στην πόρτα. Γεφυρώνεται η επικοινωνία μεταξύ client και server με την ίδια εντολή και τέλος γίνεται connect με το IP και το δοσμένο port. Για τα επόμενα βήματα, αφού υπάρχει σε commented section το πρόσχημα του μηνύματος ετοιμάζω τα δεδομένα όπως msg\_number1, msg\_number2, operator.

msg\_type = 0  
print("Please enter the first number: ")  
msg\_number1 = input()  
print("Please enter the second number: ")  
msg\_number2 = input()  
print("Please enter an operator: ")  
operator = input()

Με μια επανάληψη προσπαθώ να εξασφαλήσω την εγκυρότητα του operator, για το αν είναι δηλαδή σύμβολο των αναμενόμενων πράξεων.

while operator != "+" and operator != "-" and operator != "\*" and operator != "/":  
 print("Please re-enter an operator: ")  
 operator = input()

Το msg\_name καθορίζει το μέγεθος των bytes που καταλαμβάνει ο operator έχοντας ως σταθερό encoding το UTF-8

msg\_name = bytes(operator, 'utf-8')

Έπειτα, υπολογίζω το μήκος των δοσμένων αριθμών καθώς και τα bytes που αυτοί καταλαμβάνουν.

msg\_length1 = len(msg\_number1)  
msg\_length2 = len(msg\_number2)

nb1 = bytes(msg\_number1, 'utf-8')  
nb2 = bytes(msg\_number2, 'utf-8')

Στην συνέχεια με την γνωστή φόρμουλα του υπολογισμού padding υπολογίζω τα padding των αριθμών 1 και 2 που έδωσε ο χρήστης, οποίοι έχουν οριστεί ως unsigned integer.

msg\_padSize\_number1 = (4 - len(msg\_number1) % 4) % 4  
msg\_padSize\_number2 = (4 - len(msg\_number2) % 4) % 4

Εν συνεχεία, χρειάζεται να διακρίνω όλες τις περιπτώσεις.

***1. Number1 and Number2 padding equal to zero***

Σε αυτή την περίπτωση ετοιμάζω το packString με βάση τα δεδομένα που έχω ετοιμάσει για το πακέτο και επίσης εισάγω και δύο s που προσδιορίζουν το μήκος των αριθμών

if msg\_padSize\_number1 == 0 and msg\_padSize\_number2 == 0:  
 packString = 'HBBs3x'+str(len(msg\_number1))+'s'+str(len(msg\_number2))+'s'

***2. Number1 non equal to zero and Number2 equal to zero***

Σε αυτή την περίπτωση ετοιμάζω το packString με βάση τα δεδομένα που έχω ετοιμάσει για το πακέτο και επίσης εισάγω και δύο s που προσδιορίζουν το μήκος των αριθμών καθώς και το padSize του number1 συνοδευόμενο από το x.

elif msg\_padSize\_number1 != 0 and msg\_padSize\_number2 == 0:  
 packString = 'HBBs3x'+str(len(msg\_number1))+'s'+str(msg\_padSize\_number1)+'x'+str(len(msg\_number2))+'s'

***3. Number1 equal to zero and Number2 non equal to zero***

Σε αυτή την περίπτωση εφαρμόζω την ίδια ιδέα με την 2η περίπτωση με μόνη διαφορά ότι εισάγω το padSize για το number2

elif msg\_padSize\_number1 == 0 and msg\_padSize\_number2 != 0:  
 packString = 'HBBs3x' + str(len(msg\_number1))+'s'+str(len(msg\_number2))+'s'+str(msg\_padSize\_number2)+'x'

***4. Number1 and Number2 non equal to zero***

Και για τους δύο αριθμούς δημιουργώ το pack με το padSize και των δύο

else:  
 packString = 'HBBs3x' +str(len(msg\_number1))+'s'+str(msg\_padSize\_number1)+'x'+ str(len(msg\_number2))+'s'+str(msg\_padSize\_number2)+'x'

Στην συνέχεια πρέπει να κάνω pack το string που δημιούργησα στις παραπάνω περιπτώσεις, η οποία πρέπει να ανταποκρίνεται στα δεδομένα που έχω θέσει για το μήνυμα ‘HBBs3x’ .

message = pack(packString, msg\_type, msg\_length1, msg\_length2, msg\_name, nb1, nb2)

Στην συνέχεια γυρνάω το pack στον server για να τον κάνει unpack.

clientSocket.sendall(message)

Πλέον ο server λαμβάνει το μήνυμα με 8 bytes δεσμευμένα και το κάνει unpack, αφού πρώτα το εμφανίζει σε δεκαεξαδική μορφή.

msg = conn.recv(8)  
print(binascii.hexlify(msg))  
msg\_type, msg\_length1, msg\_length2, msg\_name = unpack('HBBs3x', msg)

Για να βρώ το τελικό padding εφαρμόζω τις παρακάτω πράξεις για τον κάθε αριθμό:

msg\_padSize\_number1 = (4 - msg\_length1 % 4) % 4  
msg\_padSize\_number2 = (4 - msg\_length2 % 4) % 4

Επίσης, εμφανίζω το μέγεθος του μηνύματος σε bits χωρίς padding:

print('Total message length without pad='+str(msg\_length1+msg\_length2))

Υπολογίζω επίσης το μέγεθος του μηνύματος, τα υπόλοιπα bytes του μηνύματος που απομένουν:

msg = conn.recv(msg\_length1+msg\_padSize\_number1+msg\_length2+msg\_padSize\_number2)

Όπως διαχώρισα και τις περιπτώσεις από την μεριά του client έτσι διαχωρίζω και της περιπτώσεις για το unpack στην μεριά του server. Λαμβάνοντας υπόψη, το padding του κάθε αριθμού αν είναι ίσο με το μηδέν δεν κάνω unpack το pad του αν δεν είναι όμως ίσο με το μηδέν προχωρώ στο απλό unpack του μηνύματος χωρίς ‘x’.

if msg\_padSize\_number1 == 0 and msg\_padSize\_number2 == 0:  
 unpackString = str(msg\_length1)+'s'+str(msg\_length2)+'s'  
elif msg\_padSize\_number1 != 0 and msg\_padSize\_number2 == 0:  
 unpackString = str(msg\_length1)+'s'+str(msg\_padSize\_number1)+'x'+str(msg\_length2)+'s'  
elif msg\_padSize\_number1 == 0 and msg\_padSize\_number2 != 0:  
 unpackString = str(msg\_length1)+'s'+str(msg\_length2)+'s'+str(msg\_padSize\_number2)+'x'  
else:  
 unpackString = str(msg\_length1)+'s'+str(msg\_padSize\_number1)+'x'+ str(msg\_length2)+'s'+str(msg\_padSize\_number2)+'x

Κάνω unpack τα εναπομείναντα στοιχεία του μηνύματος και τα ορίζω αντίστοιχα στους αριθμούς 1 και 2.

msg\_number1, msg\_number2 = unpack(unpackString, msg)

Φτάνοντας στο τέλος κάνω decode και εμφανίζω τα δεδομένα στην κονσόλα του χρήστη.

msg\_name = msg\_name.decode('utf-8')  
msg\_number1 = msg\_number1.decode('utf-8')  
msg\_number2 = msg\_number2.decode('utf-8')  
print('Name received is '+msg\_name)  
print('Number 1 received is ' + msg\_number1)  
print('Number 2 received is ' + msg\_number2)

Πλέον χρειάζεται να διαχωρίσω το response code σε περιπτώσεις. Αρχικά το ορίζω σε 0 που σημαίνει ότι όλα πήγαν καλά στην επικοινωνία του μηνύματος. Όμως, διαχωρίζω άλλες τρεις περιπτώσεις όπως αν ένας εκ των δύο αριθμών είναι <0 και >30000, όπου τότε επιστρέφω άλλον κωδικό. Άλλη μία περίπτωση που πρέπει να διαχωρίσω είναι ο number2 να είναι ίσος με το 0 και ο operator “/”. Αν είναι τότε δεν μπορεί να πραγματοποιηθεί διαίρεση και επιστρέφεται κωδικός 3. Αν όμως όλα πάνε καλά ο κωδικός απάντησης θα είναι το 0. Μέσα σε δομές if κάνω τους υπολογισμούς ανάλογα το είδος του msg\_name (+, -, \*, /) και κάνω τις απαραίτητες πράξεις.

Κώδικος απάντησης:

msg\_response\_code = 0  
if "0" > msg\_number1 > "30000":  
 msg\_response\_code = 1  
if "0" > msg\_number2 > "30000":  
 msg\_response\_code = 2  
if msg\_number2 == "0" and msg\_name == "/":  
 msg\_response\_code = 3

Κώδικας υπολογισμών:

if msg\_response\_code == 0:  
 if msg\_name == "+":  
 response = int(msg\_number1) + int(msg\_number2)  
 elif msg\_name == "-":  
 response = int(msg\_number1) - int(msg\_number2)  
 elif msg\_name == "\*":  
 response = int(msg\_number1) \* int(msg\_number2)  
 elif msg\_name == "/":  
 response = int(msg\_number1) / int(msg\_number2)

Τέλος κάνω πάλι pack τα αποτελέσματα και τα στέλνω στον client. Eκεί υπολογίζω το τελικό padding υπολογίζω τα bytes του μήκους της απάντησης από τον σέρβερ καθώς και το pad. Ξανακάνω unpack το τελικό μήνυμα και ανάλογα το response code που υπάρχει εμφανίζω κατάλληλο μήνυμα. Αν όλα πάνε καλά εμφανίζω το τελικό μήκος size.

if msg\_response\_code == 0:  
 size\_pad = (4 - response\_length % 4) % 4  
 modifiedMessage = clientSocket.recv(response\_length+size\_pad)  
 if (size\_pad == 0):  
 unpackString = str(response\_length) + 's'  
 else:  
 unpackString = str(response\_length) + 's' + str(size\_pad) + 'x'  
 size = unpack(unpackString, modifiedMessage)

Σε κάθε άλλη περίπτωση ανάλογα τον κωδικό εμφανίζω αντίστοιχα μηνύματα λάθους.

elif msg\_response\_code == 1:  
 print("Number 1 out of bounds")  
elif msg\_response\_code == 2:  
 print("Number 2 out of bounds")  
elif msg\_response\_code == 3:  
 print("Both numbers are out of bounds")

Τέλος κλείνω το socket από τον client και επιστρέφω στον server.

clientSocket.close()

Εμφανίζω κάθε πιθανό error αν και εφόσον υφίσταται. Αν δεν υπάρχει αλλάζω την λογική τιμή του flag close σε True και κλείνω το connection μεταξύ server και client κλείνοντας και την διαδικασία του socket.

print(err)  
# Signal (with the flag) to close the socket  
close = True  
# And closing  
conn.close()  
serverSocket.close()

**Εφαρμογή και αποτέλεσματα.**

Παίρνω ως number1 το 10000 και ως number2 το 5000 με την πράξη της αφαίρεσης.

**Server response:**

*The server is ready to receive at port 1000*

*Connected by: ('127.0.0.1', 7614)*

*Server Socket port: ('127.0.0.1', 1000)*

*Client Socket port: ('127.0.0.1', 7614)*

*b'000005042d000000'*

*Total message length without pad=9*

*Total message length with pad=12*

*Name received is -*

*Number 1 received is 10000*

*Number 2 received is 5000*

**Client response:**

*Please enter the first number:*

*10000*

*Please enter the second number:*

*5000*

*Please enter an operator:*

*-*

*b'0100000004000000'*

*5000*