# תכנות סוקטים

## הקדמה

נעסוק בתקשורת של Client-Server, שבה השרת תמיד מחובר, בעל כתובת IP קבוע, ומאזין בפורט כלשהו לבקשות של לקוחות, ולרוב אינו יכול ליזום תקשורת באופן עצמאי. הלקוח לעומת מחובר לפרקי זמן ויכול להיות עם כתובת משתנה. הלקוח יוזם חיבור עם השרת, שולח בקשות ומקבל תשובות.

כדי שהשרת/לקוח יוכלו לקבל ולשלוח הודעות הם צריכים צינור, הנקרא Socket, שדרכו הם ישלחו/יקבלו הודעות. לאחר יצירת הסוקט כל אחד מהם צריך לחבר לו את ה-Socket Address שלו, המורכב מכתובת IP ופורט. כעת כל סוקט מסוגל לשלוח ולקבל הודעות שעליהן יוטבעו כתובת ופורט אלו. סוקט הוא בעצם הממשק המתווך בין שכבת האפליקציה לשכבת תעבורה.

תכנות הסוקט וקבלת/שליחת הודעות משתנה בהתאם לסוג פרוטוקול התעבורה שאנו רוצים להשתמש בו: TCP המספק חיבור אמין, או UDP שאינו אמין אך מהיר ודורש פחות זיכרון.

נלמד כיצד ליצור ולהשתמש בסוקטים בשתי שפות תכנות: Java ו-C.

# תכנות סוקט ב-JAVA

חבילת java.net מספקת שתי מחלקות לייצוג קשר לפי פרוטוקול TCP בין תוכנית לקוח (Socket) ותוכנית שרת (ServerSocket).

## TCP - לקוח

1. בשלב ראשון צריך ליצור את הסוקט אצל הלקוח, ולהכניס לו כתובת IP ופורט.

Socket socket = new Socket(IP, PORT);

1. לאחר מכן צריך ליצור אובייקט מסוג PrintWriter שיקבל אובייקט זרימה החוצה של ה-Socket שיצרנו באמצעות השיטה getOutputStream(). וכן ניצור אובייקט מסוג BufferedReader שיקבל אובייקט זרימה פנימה של ה-Socket באמצעות השיטה getInputStream(). באמצעות שני אובייקטים אלו נוכל לכתוב/לקבל הודעות מהשרת.

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

BufferedReader in = new BufferedReader(

new InputStreamReader(

socket.getInputStream()));

1. כדי לכתוב הודעה לשרת שהכתובת והפורט שלו מצוינים בבנאי של הסוקט נשתמש בפקודה:

out.println(String str);

1. כדי לקבל הודעה מהשרת נשתמש בפקודה למטה. התהליך יעצור עד שתתקבל הודעה מהשרת.

in.readLine();

1. לאחר שנסיים את כל העברת ההודעות עם השרת, נצטרך לסגור את הסוקט ושתי הזרימות תווים:

out.close();

in.close();

socket.close();

## TCP - שרת

1. בשלב ראשון צריך ליצור שרת שמאזין בפורט מסוים:

ServerSocket serverSocket = new ServerSocket(PORT);

1. לאחר מכן צריך לחכות שהלקוח יבקש להתחבר, נעשה זאת באמצעות השיטה accept() על השרת. שיטה זו מחזירה אובייקט מסוג Socket שבעזרתו נתקשר עם הלקוח.

Socket clientSocket = serverSocket.accept();

ניתן גם להגדיר כמה זמן נרצה שהשרת יחכה לחיבור. הפרמטר בסוגריים מייצג את הזמן באלפיות שניה שהשרת יחכה. אם זמן זה עבר ולא היה חיבור תיזרק השגיאה SocketTimeoutException, ולכן צריך לעטוף פקודה זו ב-try ו-catch. אם לא הוגדר זמן או שהפרמטר 0 השרת ימתין לעולם.

serverSocket.setSoTimeout(1000);

1. ניצור אובייקט מסוג PrintWriter ואובייקט מסוג BufferedReader כמו שעשינו בסעיף קודם. באמצעות אובייקטים אלו נוכל לקבל ולשלוח הודעות לשרת.

PrintWriter out = new PrintWriter(

clientSocket.getOutputStream(), true);

BufferedReader in = new BufferedReader(

new InputStreamReader(

clientSocket.getInputStream()));

1. כדי לקבל הודעה מהלקוח נשתמש בפקודה למטה. התהליך יעצור עד שתתקבל הודעה מהלקוח.

in.readLine();

1. כדי לכתוב הודעה ללקוח נשתמש בפקודה:

out.println(String str);

1. לאחר שנסיים את כל העברת ההודעות עם הלקוח, נצטרך לסגור את הסוקט של הלקוח ושתי הזרימות:

out.close();

in.close();

clientSocket.close();

## UDP

חבילת java.net מספקת שתי מחלקות לייצוג קשר לפי פרוטוקול UDP. DatagramPacket מייצגת חבילת מידע שמועברת בפרוטוקול, ו-DatagramSocket מייצג קשר שדרכו ניתן לקבל ולשלוח DatagramPacket.

## DatagramPacket

אובייקט מסוג DatagramPacket מייצג חבילה להעביר ברשת, ומכיל נתונים ומידע טכני על החבילה. כשרוצים לקבל חבילה ולהכניס אותה לאובייקט מסוג DatagramPacket, נשתמש בבנאי המקבל מערך של בתים שהם ה-Data של החבילה ואת גודל המערך.

DatagramPacket packet = new DatagramPacket(

byte[] arr,

int arr.length);

כאשר נרצה לבנות אובייקט המייצג חבילה שנרצה לשלוח אותה, נשתמש בבנאי כמו הקודם רק שמכיל עוד שתי שדות נוספים של כתובת ופורט של היעד שאליו אנו רוצים לשלוח.

DatagramPacket packet = new DatagramPacket(

byte[] arr,

int arr.length,

InetAddress address,

int port);

על כל אובייקט מסוג DatagramPacket נוכל להפעיל את השיטות:

packet.getAddress();

packet.getPort();

packet.getLength();

packet.getData();

## DatagramSocket

אובייקט מסוג DatagramSocket מייצגים סוקט שאליו ניתן להכניס אובייקט DatagramPacket המיועד לשליחה או קבלה.

### לקוח

לקוח יוצר סוקט באמצעות בנאי ריק:

DatagramSocket socket = new DatagramSocket();

כאשר הוא רוצה לשלוח הודעה לשרת, הוא יוצר אובייקט DatagramPacket packet שמיועד לשליחה והכתובת והפורט בו מתאימים לשרת, ואז משתמש בפקודה:

socket.send(DatagramPacket packet);

כאשר הוא רוצה לקבל חזרה הודעה מהשרת, הוא הוא יוצר אובייקט DatagramPacket packet שמיועד לקבל מידע (ללא כתובת ושרת) ואז משתמש בפקודה למטה. ניתן להגדיר גם שימתין זמן מסוים כמו TCP.

socket.receive(DatagramPacket packet);

לבסוף, צריך לסגור את הסוקט:

socket.close();

### שרת

בדיוק כמו הלקוח אלא שביצירת הסוקט צריך להכניס את הפורט שלו הוא מאזין:

DatagramSocket socket = new DatagramSocket(PORT);

## כתובות

המחלקה InetAddress מייצגת כתובת IP. באמצעות מחלקה זו השרת יכול לשמור את כתובת הלקוח:

InetAddress addr = clientSocket.getInetAddress();

השיטה הסטטית getByName במחלקה זו מאפשר להכניס כתובת URL או כתובת IP שאינה בינארית, ולתרגם את הפרמטר שהוכנס לאובייקט מסוג InetAddress.

InetAddress address = InetAddress.getByName(“www.google.com”);

InetAddress address = InetAddress.getByName(“192.168.0.1”);

# תכנות סוקט ב-C

## סוגי סוקטים

קיימים 3 סוגי סוקטים:

1. Stream Socket – סוקט היוצר קשר אמין מקצה לקצה, משתמש בפרוטוקול TCP.
2. Datagram Socket – הקשר לא אמין, משתמש בפרוטוקול UDP.
3. Raw Socket – סוקט שבו ניתן לגשת ולערוך פרוטוקולים בכל השכבות.

## ספריות

כדי לתכנת סוקטים ב-C צריכים לכלול את הספריות הבאות:

#include <sys/types.h>

#include <sys/socket.h>

#include <string.h>

#include <arpa/inet.h>

## מבנה כתובת

ב-C כדי לבנות סוקט של שרת, צריך לחבר את הסוקט עם מבנה (struct) המכיל את המידע הדרוש כדי שלקוח יוכל ליצור קשר עם השרת. מידע זה הוא: גרסת פרוטוקול ה-IP של הסוקט, הפורט לו השרת מאזין, וכתובת ה-IP של השרת. יש שני סוגי מבנים מרכזיים המחזיקים מידע זה.

### sockaddr

struct sockaddr {

unsigned short sa\_family;

char sa\_data[14];

};

sa\_family - גרסת פרוטוקול ה-IP. INET עבור IPv4, ו-AF\_INET6 עבור IPv6.

sa\_data - 14 בתים המכילים את כתובת ה-IP והפורט של השרת.

### sockaddr\_in

מבנה המיועד לIPv4 בלבד. עבור IPv6 נשתמש במבנה sockaddr\_in6.

struct sockaddr\_in {

short int sin\_family;

short int sin\_port;

struct in\_addr sin\_addr;

unsigned char sin\_zero[8];

};

sin\_family - גרסת פרוטוקול ה-IP. ערכו תמיד AF\_INET.

sin\_port - מספר הפורט. יש להמיר באמצעות פונקציה htons().

sin\_addr - מבנה מסוג in\_addr המכיל כתובת IP. יש להמיר באמצעות פונקציה inet\_pton().

### איפוס המבנה

לאחר שיצרנו מבנה מסוג sockaddr\_in יש לאפס את כל הערכים בו לפני שמכניסים חדשים. נעשה זאת באמצעות הפונקציה memset. הפונקציה מוחקת n תווים ראשונים של מחרוזת str אשר המצביע אליה מתקבל כפרמטר, ובמקומם רושמת c.

void memset(void \*str, int c, size\_t n);

נשתמש בה כך:

struct sockaddr\_in serverAddress;

memset((char \*)&serverAddress, 0, sizeof(serverAddress));

### המרת כתובת ופורט לייצוג בינארי

במבנה sockaddr\_in יש להמיר את הכתובת והפורט לייצוג בינארי כשמכניסים אותם למבנה.

struct sockaddr\_in serverAddress;

* כדי להמיר פורט נשתמש בפונקציה htons שתקבל את מספר הפורט.

serverAddress.sin\_port = htons(SERVER\_PORT);

* כדי להמיר כתובת נשתמש בפונקציה inet\_pton, שנכניס אליה את כתובת ה-IP.

inet\_pton(AF\_INET, "10.12.110.57", &(serverAddress.sin\_addr));

### המרת ייצוג בינארי לכתובת ופורט

עבור sockaddr\_in אם נרצה לחלץ ממנו את הכתובת והפורט נשתמש בפונקציות הבאות:

* כדי להמיר sin\_port לפורט נשתמש בפונקציה ntohs.

Short port = ntohs(serverAddress.sin\_port);

* כדי להמיר sin\_addr לכתובת נשתמש בפונקציה inet\_ntop. נצטרך קודם ליצור מחרוזת שתוכל להכיל את הכתובת.

char addr[ADDRS\_LEN];

inet\_ntop(AF\_INET, &(serverAddress.sin\_addr), addr, ADDRS\_LEN);

## UDP - שרת

השלבים הנצרכים כדי ליצור סוקט עבור השרת המוכן לקבל ולשלוח הודעות הם:

### יצירת סוקט

int socket(int af, int type, int protocol);

**af** - גרסת פרוטוקול ה-IP של הסוקט. AF\_INET עבור IPv4, ו-AF\_INET6 עבור IPv6.

**type** - סוג הסוקט (בסעיף א'). SOCK\_DGRAM עבור UDP, ו-SOCK\_STREAM עבור TCP.

**protocol** - סוג פרוטוקול התעבורה. עבור UDP, ו-SOCK\_STREAM עבור TCP. בדרך כלל כותבים 0 שנותן את פרוטוקול התעבורה המתאים לסוג הסוקט משדה קודם (ברירת מחדל).

**ערך חזרה** - במידה ולא הוקם socket הפונקציה תחזיר את הערך -1.

### יצירת הכתובת והפורט

ניצור מבנה מסוג sockaddr\_in. תחילה נאפס אותו באמצעות פונקציה memset. לאחר מכן נכניס לו את גרסת פרוטוקול IP, הפורט, וכתובת ה-IP שאנו רוצים באמצעות פונקציות המרה לייצוג בינארי.

### קישור כתובת ופורט עם סוקט - bind

לאחר שיצרנו סוקט עבור השרת, צריך לקשר את הסוקט עם הכתובת והפורט של השרת, כך שאם משהו ינסה לפנות לכתובת ופורט אלו הוא יתחבר לסוקט. נעשה זאת באמצעות הפונקציה הסטטית bind:

int bind(int sockfd, struct sockaddr \*my\_addr, int addrlen);

**sockfd** - הסוקט שיצרנו בשביל השרת.

**my\_addr** - מצביע למבנה sockaddr המכיל כתובת ופורט של שרת אליו מתחברים. בדרך כלל נעשה casting על מבנה מסוג sockaddr\_in. (struct sockaddr \*)&sockaddr\_in.

**addrlen** - גודל מבנה sockaddr בבתים. בדרך כלל sizeof(my\_addr).

**ערך חזרה** - הפונקציה מחזירה ערך -1 אם יש שגיאה, אחרת 0.

## UDP - לקוח

הלקוח הרבה יותר פשוט. יש רק ליצור סוקט כמו השלב הראשון אצל השרת. כבר לאחר מכן ניתן לשלוח ולקבל הודעות באמצעות הסוקט. המיוחד בלקוח הוא שלאחר סיום ההתקשרות צריך לסגור את הסוקט.

### סגירת סוקט

בסיום ההתקשרות עם השרת, הלקוח צריך לסגור את הסוקט שהוא פתח.

int close(int sockfd);

הפונקציה מקבלת סוקט, ומחזירה -1 אם יש שגיאה, אחרת 0.

## UDP - שליחה וקבלה

### שליחת הודעה - sendto

כדי לשלוח הודעה בסוקט של UDP נשתמש בפונקציה sendto.

int sendto(int sockfd, const void \*msg, int len, unsigned int flags,

const struct sockaddr \*to, socklen\_t tolen);

**sockfd** - הסוקט שממנו שולחים.

**msg** - מצביע למידע שרוצים לשלוח. בדרך כלל מערך של char. char \*msg = "message here".

**len** - גודל המידע הנשלח בבתים. נוח להשתמש בפונקציה len = strlen(msg).

**flags** - יקבל בדרך כלל את הערך 0.

**to** - מצביע למבנה sockaddr המכיל כתובת ופורט של מי שרוצים לשלוח אליו. בדרך כלל נעשה casting על מבנה מסוג sockaddr\_in. (struct sockaddr \*)&sockaddr\_in.

**tolen** - מצביע לגודל מבנה to בבתים. בדרך כלל sizeof(to).

**ערך חזרה** - הפונקציה מחזירה את גודל המידע שנשלח בבתים, אם יש שגיאה מחזירה 1-. יש לוודא שאכן הגודל שרצינו לשלוח נשלח, ואם לא להשלים מה שחסר.

### קבלת הודעה - recvfrom

כדי לקבל הודעה בסוקט של UDP נשתמש בפונקציה recvfrom. הפונקציה תחכה עד שתתקבל הודעה.

int recvfrom(int sockfd, void \*buf, int len, unsigned int flags,

struct sockaddr \*from, int \*fromlen);

**sockfd** - הסוקט שאליו מקבלים הודעה.

**buf** - מצביע לבאפר אשר קוראים לתוכו. בדרך כלל מערך של char. char \*msg[size].

**len** - אורך הבתים שאנו רוצים לקרוא. בדרך כלל בגודל של buf.

**flags** - יקבל בדרך כלל את הערך 0.

**from** - מצביע למבנה sockaddr המכיל כתובת ופורט של מי שרוצים לקבל ממנו הודעה. בדרך כלל נעשה casting על מבנה מסוג sockaddr\_in. (struct sockaddr \*)&sockaddr\_in.

**fromlen** - מצביע לגודל מבנה from בבתים. בדרך כלל sizeof(from).

**ערך חזרה** - הפונקציה מחזירה את גודל המידע שהתקבל בבתים, אם יש שגיאה מחזירה 1-. אם מחזירה 0 משקף שהחיבור נסגר.

## TCP - שרת

בסוקט שרת של TCP יצירת השרת מתחילה כמו בסוקט של UDP. יש ליצור את הסוקט, את מבנה הכתובת, ולקשר את הסוקט עם הכתובת באמצעות פונקציית bind. אלא שב-TCP יש להוסיף עוד שני שלבים.

### הקשבה לחיבורים - listen

אחרי פעולת bind צריך להעביר את השרת למצב המתנה שבו הוא מחכה שלקוח יצור איתו קשר.

int listen(int sockfd, int backlog);

**sockfd** - הסוקט שיצרנו בשביל השרת.

**backlog** - גודל מקסימלי של תור הבקשות לחיבור. תור הבקשות לחיבור מוגבל על ידי כמות החיבורים שניתן להקים.

**ערך חזרה** - הפונקציה מחזירה ערך -1 אם יש שגיאה, אחרת 0.

### אישור חיבור - accept

לאחר שהשרת קיבל בקשה לחיבור הוא צריך לאשר את אחת הבקשות הנמצאות בתור (פתיחת קשר).

int accept(int sockfd, struct sockaddr \*addr, socklen\_t \*addrlen);

**sockfd** - הסוקט שיצרנו בשביל השרת.

**addr** - מצביע למבנה sockaddr שיקבל את כל הפרטים על הכתובת של הלקוח. באמצעות כתובת זו נוכל לשלוח תגובה ללקוח. בדרך כלל נעשה casting על מבנה מסוג sockaddr\_in.

**socklen\_t** - גודל מבנה sockaddr בבתים. בדרך כלל sizeof(addr).

**ערך חזרה** - אם החיבור לא הוקם, הפונקציה מחזירה ערך 1-, אחרת 0.

## TCP - לקוח

בסוקט לקוח של TCP גם כן צריך ליצור סוקט כמו ב-UDP, אלא שיש להוסיף שלב נוסף של התחברות לשרת. לאחר שלב זה הלקוח מוכן לשלוח ולקבל הודעות מהשרת. חשוב לזכור בסיום ההתקשרות יש לסגור את הסוקט.

### הקמת חיבור עם השרת - connect

לאחר שהלקוח יוצר את הסוקט הוא צריך לחבר אותו עם השרת

int connect(int sockfd, struct sockaddr \*serv\_addr, int addrlen);

**sockfd** - הסוקט שיצרנו בשביל הלקוח.

**serv\_addr** - מצביע למבנה sockaddr המכיל כתובת ופורט של שרת אליו מתחברים. בדרך כלל נעשה casting על מבנה מסוג sockaddr\_in. (struct sockaddr \*)&sockaddr\_in.

**addrlen** - גודל מבנה serv\_addr בבתים. בדרך כלל sizeof(serv\_addr).

**ערך חזרה** - גם כאן, אם החיבור לא הוקם, הפונקציה מחזירה ערך 1-, אחרת 0.

## TCP - שליחה וקבלה

### שליחת הודעה - send

כדי לשלוח הודעה בסוקט של TCP נשתמש בפונקציה send.

int send(int sockfd, const void \*msg, int len, int flags);

**sockfd** - הסוקט שממנו שולחים.

**msg** - מצביע למידע שרוצים לשלוח. בדרך כלל מערך של char. char \*msg = "message here".

**len** - גודל המידע הנשלח בבתים. נוח להשתמש בפונקציה len = strlen(msg).

**flags** - יקבל בדרך כלל את הערך 0.

**ערך חזרה** - הפונקציה מחזירה את גודל המידע שנשלח בבתים, אם יש שגיאה מחזירה 1-. יש לוודא שאכן הגודל שרצינו לשלוח נשלח, ואם לא להשלים מה שחסר.

### קבלת הודעה - recv

כדי לקבל הודעה בסוקט של TCP נשתמש בפונקציה recv. הפונקציה תחכה עד שתתקבל הודעה.

int recv(int sockfd, void \*buf, int len, int flags);

**sockfd** - הסוקט שאליו מקבלים הודעה.

**buf** - מצביע לבאפר אשר קוראים לתוכו. בדרך כלל מערך של char. char \*msg[size].

**len** - אורך הבתים שאנו רוצים לקרוא. בדרך כלל בגודל של buf.

**flags** - יקבל בדרך כלל את הערך 0.

**ערך חזרה** - הפונקציה מחזירה את גודל המידע שהתקבל בבתים, אם יש שגיאה מחזירה 1-. אם מחזירה 0 משקף שהחיבור נסגר.

# Raw Socket