פרוטוקול:  
פורמט שליחה מהלקוח:

המרת הודעה לבייטים:

נוצר אובייקט ההודעה מסוג Request, המייצג את ההודעה שתישלח. הפונקציה msg.toByteArray() נקראת להמיר את ההודעה לייצוג מערך בתים.





כל מחלקה מומרת באופן שונה לבינארי:



הסיבה לכך שלא השתמשתי בפיקל להכול היא שבמהלך הבדיקות, אורך ההודעה היה גבוה במידה שהייתה גורמת לאגים, אז ניסיתי ליצור פחות עומס.

אורך הודעה:

אורך מערך בתים של ההודעות נקבע באמצעות len(הודעה). ערך זה מייצג את גודל מטען ההודעה בבתים.

קידוד אורך הודעה:

ערך אורך ההודעה מומר למחרוזת באמצעות str(len(message)), ולאחר מכן מקודד לבייטים באמצעות פורמט הקידוד שצוין (FORMAT = 'utf-8' במקרה זה).

ריפוד אורך ההודעה:

אורך ההודעה המקודדת מרופד ברווחים (b' ') כדי להבטיח שהוא תופס את כל גודל הכותרת (HEADER). אם אורך אורך ההודעה המקודדת כבר שווה או גדול מגודל הכותרת, מועלית ValueError.

גודל כותרת:

המשתנה HEADER מייצג את גודל הכותרת בבתים. במקרה זה, הוא מוגדר ל-4 בתים.

שליחת ההודעה:

שיטת השליחה של הלקוח נקראת לשלוח את ההודעה. אורך ההודעה וההודעה עצמה משורשרים (msgLength + message), ויוצרים את ההודעה השלמה שתישלח דרך הרשת.

לסיכום, מבנה ההודעה של פונקציית השליחה כולל גודל כותרת (HEADER), אורך הודעה מקודדת (מרופד כך שיתאים לגודל הכותרת), ומטען ההודעה בפועל. משתנה אורך ההודעה מייצג את גודל מטען ההודעה ומקודד כמחרוזת בבתים לפני שליחתו.

הלקוח שולח הודעה לשרת אישי שנוצר פר לקוח, ולא פר משחק.

הרי רשימת ההודעות:



הערך נע מ 1 עד 6, כשערך גדל ב128 אם אמור לחזור ערך מעבר לסטטוס בריספונד. למרות שבפרויקט הסופי אין בפיצ'ר הזה שימוש, הוא עזר מאוד בדיבאגינג

שרת:

הפורמט של הRESPONSE זהה לאחד של הREQUEST רק שבמקום "request type" שמסביר מה לעשות עם האובייקט, ישנו "Response type" שמסביר אם יש שגיאה, ואם כן מה היא



סדר ההרצה הוא לקוח – שרת – לקוח. לקוח שולח הודעה לשרת, שרת מעבד את ההודעה ויוצר שינויים, ומחזיר הודעה ללקוח שכוללת סטטוס.