Overview

מבנה התוכניות:

הקוד מחולק ל-3 מודולים: לקוח, שרת ועזרים. מודולי השרת והלקוח אחראים למימוש הפונקציות של כל רכיב. הם בלתי תלויים אחד בשני ונעזרים במודול העזרים. מודול זה מכיל את הפונקציות האחראיות להעברת ההודעות והקבצים בין הצדדים השונים. כמו כן, הוא מכיל את ערכי הקבועים השונים המשמשים לחסמים עבור הודעות המערכת.

התוכניות וכיצד להריץ אותן:

בכדי שהשרת יפעל, הוא חייב לקבל קובץ tab-delimited המכיל בעמודה הראשונה שמות משתמשים ובעמודה השנייה את סיסמתם. בכדי להפעיל את השרת יש לכתוב בשורת הפקודה

./file_server users_file dir_path [port]

כאשר users_file מייצג את הקובץ המכיל את שמות המשתמשים החוקיים וסיסמתם, dir_path מציין את התיקייה בא יישמרו הקבצים עבור כל משתמש. תהליך אתחול השרת מצריך שבקובץ המשתמשים והסיסמאות תהיה שורה ריקה בסופו. במידה ולא קיימת תיקייה שכזו השרת ייצור אותה. כמו כן, בעת עלייתו, תיווצר תיקייה לכל יוזר חוקי במערכת בתוך אותה תיקייה. הארגומנט השלישי שניתן להכניס הוא הפורט אליו יאזין השרת. הפורט הדיפולטי הוא 1337.

כדי להפעיל את הלקוח יש להריץ את השורה הבאה

./file_client [hostname [port]]

כאשר hostname מייצג את שם או כתובת השרת. לא ניתן להזין פורט מבלי להזין את שם או כתובת השרת. הערך הדיפולטי לשם השרת הוא hostname והפורט הדיפולטי הוא 1337.

פרוטוקול התקשורת:

התקשורת בין השרת ללקוח מתחלקת להודעות משני סוגים - הודעות מערכת והודעות מידע. גודל הודעות המערכת קבוע, ידוע מראש ומוגדר לפקודות השונות. על כן, הודעות המערכת נשלחות ללא הודעה מקדימה אך בסדר מוגדר מראש. לעומתן, הודעות המידע הנוגעות לריצת השרת ויישום הפעולות אינן מגיעות בגודל קבוע. לדוגמא, גודל הארגומנטים הנשלחים מהלקוח אינו זהה. על כן, לפני שליחת כל הודעה מידע כזו, תישלח הודעת מערכת המכילה מחרוזת בינארית המייצגת את אורך ההודעה שעתידה להגיע. להודעות המידע הוגדרו חסמים מקסימליים (ראה נספח 1) שבהתאם להם נקבע גודל הודעת המערכת הרלוונטית - מכילה ייצוג בינארי של גודל הודעת המידע ומרופדת באפסים ב-MSB עד לגודל שהוגדר להודעת המערכת הנ"ל (ראה נספח 2). כך, נמנע המצב בו אחד הצדדים מחכה לקבל מידע נוסף מהצד השני וקופא.

התקשורת בין השרת ללקוח מתבצעת בעזרת הפונקציות recv_data ו-send_data אשר מימושן דומה לפונקציה sendall שהוצגה בתרגול. העברת הקבצים בין השרת ללקוח מתבצעת בעזרת receive_file ו-send_file שעוטפות את פונקציית השליחה והקבלה הרגילות וקוראות/כותבות לתוך קובץ את הפלט. שתי הפונקציות מחזירות 0 במקרה של הצלחה, 1 במקרה של כישלון הנובע מתקלה עם הקבצים כדוגמת שגיאה בכתיבה לקובץ או 2 במידה וקפצה תקלה הקשורה לתקשורת בין השרת ללקוח.

השיפור שנעשה למערכת בתרגיל השני מאפשר עבודה של השרת במקביל על מספר לקוחות. העבודה מתבצעת ע"י שימוש בפונ' select אשר מנפה לשרת את החיבורים המעוניינים לשלוח מידע לעומת אלו שלא, דבר המונע קפיאה של השרת. בעליית השרת הוא יחכה לחיבורים, והפונ' select תחזיר את החיבורים אשר מוכנים לעבודה. בתחילה תהיה בדיקה האם ישנם לקוחות חדשים המעוניינים

להתחבר. לאחר מכן, השרת יעבור על החיבורים המוכנים לשליחת פקודה, ועבור כל לקוח יבצע את הפעולה המבוקשת. לאחר שסיים לעבור על כלל הלקוחות, יבצע פקודת select מחדש. באופן זה, לא יהיו לקוחות אשר יורעבו. מלבד האפשרות לעבודה של מספר לקוחות באותו הזמן נוספה אפשרות שליחת הודעות בין משתמשים שונים של המערכת. במקרים מסוימים, אשר יפורטו בהמשך, ההודעות ישמרו בקובץ בשם Messages_received_offline.txt. אנו מתייחסים לקובץ זה כשקוף למשתמש ומניחים, בהתאם לאישור המתרגל, כי לא מתבצעות עליו פעולות של הלקוח.

בגרסה הקודמת של המערכת, השרת והלקוח קרסו בשגיאה הקשורה למערכת הקבצים (file system) ובשגיאות הקשורות לרשת. כעת, במידה והסיבה לקריסה הינה בעיית רשת, לדוגמא שליחה או קבלה של הודעה שנכשלה, השרת מפסיק את העבודה מול הלקוח הנוכחי ועובר ללקוח הבא. לעומת זאת, במידה ומקור השגיאה הוא במערכת הקבצים, לדוגמא, מחיקת קובץ שנכשלה, השרת יקרוס. הסיבה לשינוי נובעת בעובדה שכעת ישנם לקוחות נוספים אשר מחוברים לשרת ונקודת ההנחה היא שמקור בעיית הקשת בלקוח ולא בשרת.

תהליך ההזדהות:

כאשר לקוח מתחבר לשרת מודפסת הודעה המברכת אותו ומבקשת ממנו להכניס שם משתמש וסיסמא בפורמט הבא :

User: current_user's_username
Password: its_password

במידה ושם המשתמש והסיסמא תקינים, תישלח ללקוח הודעה עם מספר קבציו המאוחסנים בשרת, מבלי להתייחס לקובץ ההודעות. במידה והודעות ההזדהות לא מתאימות לפורמט תודפס הודעה למשתמש עם דוגמא לפורמט הרצוי ויתאפשר ניסיון חיבור נוסף. כמו כן, במידה ושם המשתמש או הסיסמא שגויים, תישלח ללקוח תגובה ותודפס ההודעה :

"Wrong credentials - try again"

חשוב לציין כי אין הגבלה על מספר ניסיונות החיבור של הלקוח לשרת.

כאמור, במסגרת השינוי של המערכת והתמיכה במשתמשים מרובים בו זמנית הפרדנו למקרים את תהליכי התחברות לשרת והרצת פקודות. לכן, כאשר משתמש חדש יהיה בתהליך חיבור, השרת יתנהל מולו ולקוחות אחרים ימתינו עד אשר תהליך ההזדהות של אותו לקוח יסתיים. לדעתנו, על אף שתהליך ההזדהות במטלה דורש שליחה בנפרד של נתונים, לוגית מדובר בתהליך המתבצע באופן אטומי מול השרת.

לאחר ההזדהות:

מודפסת בקשה ללקוח להכניס את הפקודה הרצויה. כרגע, הפקודות הנתמכות ע״י השרת הן :

- list_of_files - מדפיסה ללקוח רשימה עם שמות קבציו בשרת.
- delete_file - מקבלת כארגומנט שם קובץ ומוחקת אותו מהשרת.
- add_file - מוסיפה קובץ לשרת.
- get_file - מביאה קובץ מהשרת.
- users_online - הצגת הלקוחות המחוברים לשרת.
- msg - העברת הודעות בין לקוחות שונים של המערכת דרך השרת.
- read_msgs - קריאת ההודעות שהמשתמש קיבל בזמן שהיה מנותק.
- quit - יציאה וניתוק הלקוח.

בהתאם לפקודה שהלקוח מכניס, תישלח הודעה באורך COMMAND_MSG_LEN לשרת עם אחת מבין האותיות {L,D,A,G,U,M,R,Q} ולפי האות שתתקבל, השרת יידע איזו פקודה להריץ וכמה ארגומנטים הוא צריך לקבל לטובתה. לאחר מכן, השרת יחכה שהלקוח ישלח אותם. מכיוון שגדלי הארגומנטים משתנים, קודם יועבר גודל הארגומנטים בהודעת מערכת ואח"כ הארגומנטים יעברו כהודעות מידע.

במידה והוכנסה פקודה שלא קיימת תודפס ללקוח הודעה שהפקודה לא מוכרת ויתאפשר לו להכניס פקודה חדשה - "un-recognizable command. try again".

לאחר שהשרת מקבל את התו הבודד המציין את הפקודה ואח"כ את הארגומנטים הנחוצים להרצתה, במידה ויש כאלו, הוא מבצע אותה ומחזיר ללקוח הודעת מערכת עם חיווי בהתאם לסוג הפקודה.

מבנה הנתונים בשרת:

בשרת יתוחזק מערך בגודל 15 שמכיל את המשתמשים השונים. כאשר כל משתמש הוא מבנה הכולל את השם, הסיסמא, התיקייה בשרת המשמשת אותו, חיווי האם הוא מחובר, מספר ה-socket המשמש אותו לתקשורת עם השרת ומספר הקבצים שלו ששמורים בשרת. בעת עליית השרת, המערך ייטען מתוך קובץ המשתמשים שמועבר בפקודת העלאת השרת ובמסגרת תהליך הקמת התיקיות של המשתמשים ייספרו מספר הקבצים ויעודכן השדה הנ"ל. בהמשך פעילות השרת, בעת פעולות הוספת ומחיקת קובץ יעודכן שדה הקבצים של המשתמש בהתאם.

הפקודות:

הטיפול במקרי הקצה מתואר בהסבר עבור כל פקודה.

פקודת list_of_files:

צד לקוח:

- שולח הודעת מערכת באורך COMMAND_MSG_LEN המכילה את התו L.
- מצפה לקבל הודעת מערכת באורך LIST_OF_FILES_MSG_LEN המייצגת בבסיס בינארי את אורך המחרוזת שמכילה את רשימת הקבצים - נקרא לאורך זה x.
- מצפה לקבל הודעת מידע באורך x המכילה את המחרוזת עם רשימת הקבצים, כל קובץ בשורה נפרדת.
- במידה ו-x הוא 0, קרי הרשימה ריקה, תודפס ההודעה "You have no files stored", אחרת מדפיס את הרשימה שקיבל מהשרת.

צד שרת:

- מצפה לקבל הודעת מערכת באורך COMMAND_MSG_LEN המכילה תו פקודה. מזהה את התו L.
- נכנס לתיקייה של הלקוח ומשרש למחרוזת את שמות כל הקבצים. אנו החלטנו כי קובץ ההודעות לא יופיע למשתמש בקריאה לפונ' זו ועל כן אנו מדלגים עליו בתהליך השרשור. בודק את גודל המחרוזת וממיר למספר בינארי בעל LIST_OF_FILES_MSG_LEN ספרות.
- שולח הודעת מערכת באורך LIST_OF_FILES_MSG_LEN המייצגת בבסיס בינארי את אורך המחרוזת שמכילה את רשימת הקבצים.
- שולח בהודעת מידע את רשימת הקבצים המשורשרת.

פקודת delete_file:

צד לקוח:

- שולח הודעת מערכת באורך COMMAND_MSG_LEN המכילה את התו D.
- שולח הודעת מערכת באורך FILENAME_MSG_LEN המכילה קידוד בינארי לאורך שם הקובץ שיש למחוק, שתשלח מייד לאחר מכן.
- מצפה להודעה מערכת באורך DELETE_FILE_MSG_LEN מהשרת אשר תציין האם הקובץ נמחק או לא. במידה והשרת שלח "0" תודפס ההודעה "No shuch file exists!". אם התקבלה ההודעה "1" יודפס "File removed".

צד שרת:

- מצפה לקבל הודעת מערכת באורך COMMAND_MSG_LEN המכילה תו פקודה. מזהה את התו D.
- מצפה לקבל הודעת מערכת באורך FILENAME_MSG_LEN, המקדדת בבינארי את אורך שם הקובץ שיש למחוק.
- מצפה לקבל הודעת מידע באורך המתאים עם שם הקובץ וניגש למחוק את הקובץ ולעדכן את מספר הקבצים של הלקוח. אם המחיקה הצליחה, שולח הודעת מערכת ללקוח עם התוכן "1". במידה והקובץ לא קיים נשלח "0". אחרת, קורס ומביא לקריסת הלקוחות. [*אנו מניחים, באישור המתרגל, כי הלקוח לא יקרא למחיקת קובץ ההודעות]

פקודת add_file:

צד לקוח:

- שולח הודעת מערכת באורך COMMAND_MSG_LEN המכילה את התו A.
- בודק האם הקובץ קיים ובמידה ולא, תודפס למשתמש ההודעה "The File doesn't exist".
- אחרת, שולח הודעת מערכת באורך FILENAME_MSG_LEN המכילה קידוד בינארי לאורך השם של הקובץ העתיד להגיע, שתישלח מייד לאחר מכן.
- שולח בהודעת מידע, בעזרת send_file, את הקובץ ששמור אצל הלקוח בכתובת שהתקבלה ב-arg1.
- מצפה להודעת מערכת באורך ADD_FILE_MSG_LEN עם חיווי על הצלחת הפקודה. אם היא "0" הפעולה התבצעה בהצלחה ויודפס "File added". אם ההודעה תהיה "1" תודפס למסך ההודעה "There was a file operation error" ואם ההודעה שתתקבל תהיה "2" נדפיס למסך "There was a network error". במידה ונקבל הודעה אחרת יודפס למסך הודעה המציינת שההודעה לא ידועה ותוכנה יודפס, ולאחר מכן הלקוח ייסגר.

צד שרת:

- מצפה לקבל הודעת מערכת באורך COMMAND_MSG_LEN המכילה תו פקודה. מזהה את התו A.
- מצפה לקבל הודעת מערכת באורך FILENAME_MSG_LEN, המקדדת בבינארי את אורך השם שתחתיו הקובץ יישמר בשרת ואז יצפה לקבל את הקובץ.
- יקבל את הקובץ בעזרת receive_file וישמור אותו בתיקיית הלקוח תחת השם שקיבל. במקרה ופונ' העזר תחזיר שגיאה אשר מקורה במערכת הקבצים, השרת יקרוס ולאחריו הלקוחות המחוברים. במידה ומקור השגיאה ברשת, השרת יפסיק את העבודה מול הלקוח, ינתק אותו וימשיך הלאה.
- במידה ומדובר בשם קובץ חדש, מספר הקבצים עלה ונעדכן את המונה. במידה והקובץ החדש דרס קובץ קיים, נשאיר את ערך המונה.

- שולח הודעת מערכת באורך ADD_FILE_MSG_LEN עם התוצאה שקיבלנו מהפונקציה receive_file.
[*אנו מניחים, באישור המתרגל, כי הלקוח לא יבקש להוסיף קובץ באותו שם כמו קובץ ההודעות]

פקודת get_file:

צד לקוח:

- שולח הודעת מערכת באורך COMMAND_MSG_LEN המכילה את התו G.
- מצפה להודעת מערכת באורך GET_FILE_MSG_LEN עם חיווי מהשרת האם הוא הצליח לאתר את הקובץ.
- אם התשובה היא "0" הפעולה התבצעה בהצלחה, והלקוח יצפה לקבל הודעת מידע עם הקובץ ואז ישמור אותו בכתובת שהתקבלה ב-arg2 וידפיס "File recived" ; אם התשובה היא "1" תודפס למסך ההודעה "There was a file operation error" ; אם התשובה היא "2" נדפיס למסך "There was a network error".

צד שרת:

- מצפה לקבל הודעת מערכת באורך COMMAND_MSG_LEN המכילה תו פקודה. מזהה את התו G.
- מצפה לקבל הודעת מערכת באורך FILENAME_MSG_LEN, המקדדת בבינארי את אורך שם הקובץ שאותו על השרת לשלוח ללקוח ואז יצפה לקבל הודעת מידע הכוללת את שם הקובץ.
- שולח ללקוח הודעת מערכת באורך GET_FILE_MSG_LEN עם חיווי על הצלחה או כישלון באיתור הקובץ.
- במידה והצליח ישלח את הקובץ בעזרת send_file.
- ידפיס בצד השרת האם הפעולה הצליחה או לא - אם התשובה היא "0" הפעולה התבצעה בהצלחה ויודפס "File sent". אם ההודעה תהיה "1" תודפס למסך ההודעה "There was a file operation error" ואם ההודעה שתתקבל תהיה "2" נדפיס למסך " There was a network error".
[*אנו מניחים, באישור המתרגל, כי הלקוח לא יבקש לקבל את קובץ ההודעות]

פקודת users_online:

צד לקוח:

- שולח הודעת מערכת באורך COMMAND_MSG_LEN המכילה את התו U.
- מצפה לקבל הודעת מערכת באורך USERS_ONLINE_MSG_LEN המייצגת בבסיס בינארי את אורך המחרוזת שמכילה את רשימת המשתמשים המחוברים - נקרא לו x.
- מצפה לקבל הודעת מידע באורך x המכילה את המחרוזת עם רשימת המשתמשים המחוברים, מופרדים בפסיקים וללא רווחים.

צד שרת:

- מצפה לקבל הודעת מערכת באורך COMMAND_MSG_LEN המכילה תו פקודה. מזהה את התו U.
- עובר על מבנה הנתונים של השרת ומשרשר את שמות הלקוחות המסומנים כמחוברים, מופרדים בפסיקים וללא רווחים. נציין כי גם הלקוח ששלח את הפקודה מופיע ברשימה.
- שולח הודעת מערכת באורך USERS_ONLINE_MSG_LEN המייצגת בבסיס בינארי את אורך המחרוזת שמכילה את רשימת המשתמשים המחוברים.
- שולח בהודעת מידע את הרשימה.

פקודת msg:

צד לקוח שולח:

- שולח הודעת מערכת באורך COMMAND_MSG_LEN המכילה את התו M.
- מצפה להודעת מערכת באורך MSG_MSG_LEN עם חיווי מהשרת האם הוא הצליח לאתר את המשתמש.
- אם התשובה היא "0", תודפס למסך ההודעה "Client name passed doesn't exist". ; אם התשובה היא "1" תודפס למסך ההודעה "Message was received by the recipient client" ; אם התשובה היא "2" תודפס למסך ההודעה " Client isn't online, message was saved on the server" ; אם התשובה היא "3" תודפס למסך ההודעה "There was a network error".

צד לקוח מקבל:

- במהלך קליטת הוראות מהמשתמש, קולט הודעת מערכת באורך COMMAND_MSG_LEN המכילה את התו N.
- מצפה להודעת מערכת באורך CLIENT_INTERACTION_LEN, המקודדת בבינארי את אורך ההודעה שהשרת רוצה לשלוח אליו.
- מצפה להודעת מידע באורך x המכילה את ההודעה שנשלחה אליו.
- מדפיס למסך את ההודעה שהתקבלה.
- שולח לשרת הודעת מערכת באורך COMMAND_MSG_LEN המכילה את התו N (זהה לפקודה שלקוח המקור שלח).

צד שרת:

- מצפה לקבל הודעת מערכת באורך COMMAND_MSG_LEN המכילה תו פקודה. מזהה את התו M.
- בודק האם הלקוח שהוא יעד ההודעה קיים במערך הלקוחות. אם לקוח היעד לא קיים, תישלח ללקוח הודעת מערכת באורך MSG_MSG_LEN עם התו "0". (חיווי לכך שיעד ההודעה לא קיים בשרת). אחרת (לקוח היעד קיים בשרת):
- אם לקוח היעד מחובר:
  - השרת ישלח ללקוח היעד הודעת מערכת באורך COMMAND_MSG_LEN המכילה את התו N.
  - שולח ללקוח היעד הודעת מערכת באורך CLIENT_INTERACTION_LEN, המקדדת בבינארי את אורך ההודעה שלקוח המקור רוצה לשלוח (בתוספת זיהוי לקוח המקור).
  - שולח ללקוח היעד הודעת מידע המכילה את ההודעה.
  - מצפה לקבל מלקוח היעד הודעת מערכת באורך COMMAND_MSG_LEN המכילה את התו N (זהה לפקודה שלקוח המקור שלח). אם לא מקבל הודעה כזאת, או מקבל הודעה שונה, מתייחס לזה כאילו לא הצליח לשלוח את ההודעה ללקוח היעד.
- אם לקוח היעד לא מחובר או אם בשלב הקודם לא הצלחנו לשלוח ללקוח היעד את ההודעה:
  - הולך לתיקיית לקוח היעד ופותח את קובץ ה-Messages_received_offline.txt
  - כותב לקובץ את ההודעה של לקוח המקור כולל זיהוי של השולח. התו המפריד בין זיהוי השולח להודעה הוא התו '_'.
- שולח ללקוח המקור הודעת חיווי באורך MSG_MSG_LEN לגבי תוצאת הפקודה.

פקודת read_msgs:

צד לקוח:

- שולח הודעת מערכת באורך COMMAND_MSG_LEN המכילה את התו R.
- כל עוד לא קיבל הודעת מידע המכילה את התוכן - END_OF_OFLINE_MESSAGES
  - מצפה לקבל הודעת מערכת באורך OFFLINE_MESSAGE_LEN המייצגת בבסיס בינארי את אורך המחרוזת שמכילה הודעה שמורה.
  - מצפה לקבל הודעת מידע באורך x המכילה את המחרוזת עם ההודעה.
  - מדפיס למסך את ההודעה שקיבל.

צד שרת:

- מצפה לקבל הודעת מערכת באורך COMMAND_MSG_LEN המכילה תו פקודה. מזהה את התו R.
- פותח את הקובץ Messages_received_offline.txt שנמצא בתיקייה הלקוח.
- עד שלא מסיים לקרוא את הקובץ :
  - קורא תו תו.
  - אחרי כל שורה שנקראת (= הודעה שנשמרה בקובץ), שולח הודעת מערכת באורך OFFLINE_MESSAGE_LEN המייצגת בבסיס בינארי את אורך המחרוזת שמכילה את ההודעה שנקראה.
  - שולח בהודעת מידע את ההודעה.
- אחרי שסיים לקרוא את הקובץ. שולח באותו אופן (קודם הודעת מערכת באורך OFFLINE_MESSAGE_LEN) בהודעת מידע את ההודעה END_OF_OFLINE_MESSAGES, המסמנת ללקוח שאין יותר הודעות שמורות.

פקודת quit:

צד לקוח:

- שולח הודעת מערכת באורך COMMAND_MSG_LEN המכילה את התו Q.
- סוגר את ה-socket ומסיים את ההתקשרות עם השרת.

צד שרת:

- מצפה לקבל הודעת מערכת באורך COMMAND_MSG_LEN המכילה תו פקודה. מזהה את התו Q.
- סוגר את ה-socket הנפתח עבור החיבור מול הלקוח ומסיים את התקשורת מולו. מחכה לקבלת הקישור מהלקוח הבא.

  נציין כי במידה והשרת מקבל מהשרת שגיאת peer shutdown הוא משנה את סטטוס הלקוח להיות לא מחובר על אף שלא הגיעה פקודת quit.

נספח 1:

הגדרות הגדלים שהחלטנו עליהם ומהם נגזרו גדלי הודעות המערכת:

USERNAME_MAX_LEN 25

PASSWORD_MAX_LEN 25

MAX_USERS 15

MAX_FILES_PER_USER 15

MAX_FILE_SIZE 512

MAX_FILENAME 50

MAX_PATH_LEN 50

MAX_MESSAGE_SIZE 100

ובהתאם להם נגזרו אורכי פקודות המערכת:

COMMAND_MSG_LEN 1

USERNAME_MSG_LEN 5 - derived from USERNAME_MAX_LEN

PASSWORD_MSG_LEN 5 - derived from PASSWORD_MAX_LEN

IS_AUTHORIZED 1

NUMBER_OF_FILES_MSG_LEN 3 - derived from MAX_FILES_PER_USER

LIST_OF_FILES_MSG_LEN 10 - derived from MAX_USERS*MAX_FILES_PER..

DELETE_FILE_MSG_LEN 1

ADD_FILE_MSG_LEN 1

GET_FILE_MSG_LEN 1

FILENAME_MSG_LEN 8 - derived from MAX_FILENAME

FILE_SIZE_MSG_LEN 12 - derived from MAX_FILE_SIZE

MESSAGE_SIZE_LEN 7 // derived from MAX_MESSAGE_SIZE

USERS_ONLINE_MSG_LEN 9 // derived from USERNAME_MAX_LEN * MAX_USERS

MSG_MSG_LEN 1

CLIENT_INTERACTION_LEN 14 // derived from USERNAME_MSG_LEN + MESSAGE_SIZE_LEN

OFFLINE_MESSAGE_LEN 7 // derived from USERNAME_MAX_LEN + MAX_MESSAGE_SIZE + 3

נספח 2:

דוגמא לצורך בריפוד ולאופן הביצוע שלו:

מספר הקבצים המקסימלי האפשרי ללקוח הוא 15 ונניח שללקוח מסוים יש בשרת רק 4 קבצים.
הוא לא יודע זאת ולא ברור לו לכמה תווים עליו לצפות – אחד או שניים.
המספר 15 ניתן לייצוג בינארי ע"י 3 תווים, לכן כל מספר תווים יקודד בקידוד בינארי של 3 ביטים. כך, 4 בייצוג בינארי הוא '10' וכדי להשלים אותו ל-3 תווים נוסיף ת '0' ב-MBS ונקבל '010'. כך נשמר ונשלח הערך.