תיאור פרוטוקול התקשורת

תקשורת בין השרת ללוקוח מתבצעת בפרוטוקול מבוסס פרוטוקול TCP.

ניתן להעביר ביניהם 5 סוגי הודעות המוגדרות ע"י ה- struct הבא:

/\*\*

\* structure for the message in the system

\* type - type of the message, can be one of four types defined by msgtype\_t

\* payload - relevant data accordingly to the message type

\*\*/

typedef struct game\_msg {

msgtype\_t type;

payload\_t payload;

} game\_msg\_t;

המבנה הזה מכיל את השדות הבאים:

type מטיפוס msgtype\_t – מציין את סוג ההודעה. גודל השדה הינו 4 בתים.

\* definition of message types:

\* WELCOME - message from server to recently connected client

\* contains gameType, playersCnt, clientId and clientStatus

\* STATUS - message from server with current game status

\* contains heapStatus, clientStatus and endGame status

\* TURN\_REQ - message from client to server with new move received from a user

\* message contains heapIndex and amount of cubes to be taken from that heap

\* TURN\_RESP - message with server response to the move received from client

\* response can be that move is LEGAL or ILLEGAL or NOT\_YOUR\_TURN

\* CHAT - chat message from client to client

\* contains srcId, dstId and text

\*\*/

typedef enum {

WELCOME, STATUS, TURN\_REQ, TURN\_RESP, CHAT

} msgtype\_t;

WELCOME – הודעה מהשרת ללקוח המועברת עם התחברות הלקוח אל השרת. מכילה את סוג המשחק שיכול להיות MISERE, REGULAR או REJECTED במקרה והתחברות הלקוח לא הצליחה. בנוסף מועברים ללקוח כמות השחקנים המקסימלית במשחק הנוכחי,ID של הלקוח הניתן לו ע"י השרת וסטטוס הלקוח המוגדר לו ע"י השרת.

STATUS – הודעה מהשרת ללקוח עם פרמטרי המשחק. מכילה את מצב הערימות, עדכוני סטטוס של הלקוח וסטטוס המשחק (האם כבר הסתיים או עדיין לא).

TURN\_REQ – הודעה המועברת מהלקוח אל השרת המכילה את המהלך שהתקבל מהלקוח.

TURN\_RESP – הודעה המועברת מהשרת אל הלקוח בתור התגובה למהלך של הלקוח.

CHAT – ההודעה המועברת מהלקוח כלשהו ללקוח מסוים או כלל הלקוחות האחרים (broadcast). ההעברה מתבצעת דרך השרת.

payload מטיפוס payload\_t - הינו Union המאפשר לקרוא את המידע בהתאם לסוג ההודעה. גודל השדה כגודל המקסימלי של ההודעה 62 בתים. גודל זה מוכתב ע"י גודל הודעת טקסט מקסימלית היכולה להישלח במערכת והיא מוגבלת ל-60 תווים.

/\*\*

\* message data - can be read as one of five types

\* accordingly to the message type

\*\*/

typedef union payload {

chat\_t chat;

welcome\_msg\_t welcomeMsg;

status\_t status;

turn\_req\_t turnReq;

turn\_resp\_t turnResp;

} payload\_t;

msgtype\_t הינו משתנה המכיל את 5 סוגים האפשריים של ההודעות:

WELCOME, STATUS, TURN\_REQ, TURN\_RESP, CHAT.

**WELCOME MESSAGE**

/\*\*

\* welcome message data

\* gameType - current game type of game\_type\_t type, can be one of defined game types

\* playersCnt - number of players (p) in current game

\* clientId - ID received by client

\* clientStatus - current client status of client\_status\_t, can be one of defined client statuses

\*\*/

typedef struct welcome\_msg {

game\_type\_t gameType;

char playersCnt;

char clientId;

client\_status\_t clientStatus;

} welcome\_msg\_t;

ההודעה מכילה את המידע הנשלח ע"י השרת אל הלקוח מהשרת בזמן התחברות אל השרת.

gameType - סוג המשחק הנוכחי. יכול להיות אחד מבין הסוגים המוגדרים ב-enum הבא:

/\*\*

\* definition of game types:

\* MISERE - looser of game is player who takes last cube from heap

\* REGULAR - winner of game is player who takes last cube from heap

\* REJECTED - connect attempt not succeeded

\*\*/

typedef enum {

MISERE, REGULAR, REJECTED

} game\_type\_t;

MISERE – המפסיד הוא האחרון שלוקח קובייה מהערימות.

REGULAR – המנצח הוא האחרון שלוקח קובייה מהערימות.

REJECTED – נשלחת ללקוח שלא הצליח להתחבר מפאת חוסר מקום פנוי.

גודל השדה הינו 4 בתים.

playersCnt – מספר מקסימלי של שחקנים במשחק הנוכחי. גודל השדה הינו בית אחד.

clientId – ID הניתן ללקוח ע"י השרת. גודל השדה הינו בית אחד.

clientStatus – סטטוס הלקוח. יכול להיות אחד מבין הסוגים המוגדרים ב-enum הבא:

/\*\*

\* definition of client status:

\* PLAYING - client is player in current game

\* SPECTATOR - client is spectator in current game

\* YOUR\_TURN - client can make move now

\* UNKNOWN - status not determined yet

\*\*/

typedef enum client\_status {

PLAYING, SPECTATOR, YOUR\_TURN, UNKNOWN

} client\_status\_t;

PLAYING – הלקוח הינו שחקן במשחק הנוכחי.

SPECTATOR – הלקוח רק צופה במשחק ללא אפשרות ביצוע מהלכים.

YOUR\_TURN – התור לביצוע מהלך במשחק שייך ללקוח נוכחי.

UNKNOWN – סטטוס הלקוח עוד לא הוגדר.

גודל השדה הינו 4 בתים.

**STATUS MESSAGE**

/\*\*

\* status data

\* heapStatus - of heap\_status\_t type, contains current state of heaps array

\* clientStatus - current client status

\* endGame - current game state

\*\*/

typedef struct status {

heap\_status\_t heapStatus;

client\_status\_t clientStatus;

end\_game\_t endGame;

} status\_t;

ההודעה מכילה את המידע הנשלח ע"י השרת אל הלקוח בתור העדכון בזמן המשחק.

heapStatus – מצב נוכחי של הערימות. מוגדר ע"י הטיפוס הבא:

/\*\*

\* heap data

\* heap[4] - current state of the heaps

\*\*/

typedef struct heap\_status {

short heap[4];

} heap\_status\_t;

גודל השדה הינו 8 בתים.

clientStatus – סטטוס הלקוח. יכול להיות אחד מבין הסוגים המוגדרים ב-enum מטיפוס client\_status\_t. גודל השדה הינו 4 בתים.

endgame – סטטוס המשחק הנוכחי. יכול להיות אחד מבין הסוגים המוגדרים ב-enum הבא:

/\*\*

\* definition of end game statuses

\* YOU\_WIN - player win

\* YOU\_LOSE - player lose

\* YOU\_WATCHED - player watched

\* NOT\_FINISHED - game not finished yet

\*\*/

typedef enum {

YOU\_WIN, YOU\_LOSE, YOU\_WATCHED, NOT\_FINISHED

} end\_game\_t;

YOU\_WIN – סטטוס זה נשלח למנצחים במשחק.

YOU\_LOSE – סטטוס זה נשלח למפסידים במשחק.

YOU\_WATCHED – סטטוס זה נשלח לצופים במשחק.

NOT\_FINISHED – סטטוס זה נשלח לכולם כל עוד המשחק טרם הסתיים.

גודל השדה הינו 4 בתים.

**TURN REQUEST MESSAGE**

/\*\*

\* user move data

\* heapIndex - index of a heap chosen by user

\* amount - amount of cubes to take from chosen heap

\*\*/

typedef struct turn\_req {

char heapIndex;

short amount;

} turn\_req\_t;

ההודעה מכילה את המידע הנשלח ע"י הלקוח אל השרת. מכילה את המהלך שמבצע הלקוח שזה תורו כעת.

heapIndex – שם הערימה ממנה הלקוח בוחר לקחת את הקוביות. גודל השדה הינו בית אחד.

amount – כמות הקוביות שהלקוח בוחר לקחת מהערימה המצוינת. גודל השדה הינו 2 ביתם.

**TURN RESPONSE MESSAGE**

/\*\*

\* definition of move types

\* LEGAL - move accepted by the server

\* NOT\_YOUR\_TURN - turn is out of player's order

\* ILLEGAL - move rejected by the server (invalid move)

\*\*/

typedef enum {

LEGAL, NOT\_YOUR\_TURN, ILLEGAL

} turn\_resp\_t;

ההודעה מכילה את המידע הנשלח ע"י השרת אל הלקוח בתור תגובה למהלך שהתקבל. התגובה מוגדרת ע"י enum המכיל את האפשרויות הבאות לתגובה:

LEGAL – המהלך שהתבצע הינו חוקי והתקבל בשרת.

NOT\_YOUR\_TURN – המהלך שהתקבל בשרת בוצע ע"י לקוח שזה אינו תורו כעת לבצע מהלך. לכן נדחה ע"י השרת.

ILLEGAL – המהלך שהתבצע הינו לא חוקי מבחינת צורת הקלדתו ולכן התור יעבור ללקוח הבא בתור.

גודל ההודעה הינו 4 בתים.

**CHAT**

/\*\*

\* chat message data

\* srcId - sender ID

\* dstId - receiver ID

\* text - message data

\*\*/

typedef struct chat {

char srcId;

char dstId;

char text[MAX\_CHAT\_TEXT];

} chat\_t;

ההודעה מכילה את המידע הנשלח ע"י הלקוח אל השרת. מכילה מחרוזת טקסט שהלקוח מעוניין לשלוח ללקוח אחר.

srcId – ID הלקוח השולח את ההודעה. גודל שדה זה בית אחד.

dstId – ID הלקוח אליו נשלחת ההודעה. גודל שדה זה בית אחד.

text – תוכן ההודעות טקסט הנשלחת. גודל שדה זה הינו כערך הקבוע MAX\_CHAT\_TEXT השווה ל-60. זהו חסם עליון על אורך הודעת טקסט במערכת. לכן גודל שדה זה הינו 60 בתים.

גודל מקסימלי של ההודעות המועברות בשני הכיוונים בין השרת והלקוח הינו 66 בתים.

תיאור התוכנית

פתרון התרגיל ממש את אפליקציית הרשת במבנה שרת/לקוח. האפליקציה מממשת גרסה פשוטה של משחק ה-Nim. ומרחיבה את המימוש של תרגיל המעשי מס' 1.

הפתרון מורכב משתי תוכנות: הלקוח(client) ממוש בקובץ בשם nim.c והשרת בקובץ בשם nim-server.c. הלקוח מאפשר משחק אינטראקטיבי בין מספר שחקנים שונים ברשת.

השרת מנהל את המשחק מול מספר שחקנים שונים. כל שחקן משתמש בתוכנת לקוח משלו ע"מ לתקשר עם השרת.

התוכנות משתמשות בקובץ עזר משותף בשם transport.c הנמצא יחד עם התוכנות באותה תיקייה. אל הפליקציה מצורף קובץ ה-makefile המאפשר להכין את 2 התוכנות להרצה בעזרת פקודת make all מה-terminal.

האפליקציה נבדקה על שרת ה- Nova של האוניברסיטה.

סדר הפעלת התוכנות:

קודם נדרש להפעיל את תוכנת השרת nim-server ואז את תוכנת הלקוח nim.

הפעלת תוכנת השרת (nim-server):

את התוכנה של השרת נדרש להפעיל עם 3 פרמטרי חובה:

p – פרמטר מטיפוס int המציין את מספר שחקנים מקסימלי האפשרי במהלך המשחק הנוכחי.

M – פרמטר מטיפוס int המציין את הגודל התחלתי של הערימות.

gameType – פרמטר מטיפוס int המציין באופן בוליאני את סוג המשחק.

אם רוצים להפעיל את תוכנת השרת תוך ציון ה-port הספציפי עליו היא תפעל, קיימת אפשרות להעביר לתוכנה את הפרמטר השלישי המציין את מספר ה-port. פרמטר זה הינו פרמטר אופציונלי ואינו חובה להעברה. במקרה ולא מעבירים אותו לתוכנה, היא תשתמש ב-port ברירת המחדל שמספרו 6325.

הפעלת תוכנת הלקוח (nim):

תוכנת הלקוח אינה דורשת פרמטרי חובה להפעלת. לפי הצורך ניתן ניתן להפעיל אותה עם 2 פרמטרים אופציונליים:

hostname- פרמטר מטיפוס char\* המציין את כתובת השרת אליו מעוניינים להתחבר. ערך ברירת מחדל של ה- hostname הינה localhost.

port – פרמטר מטיפוס int המציין את מספר ה-port. ערך ברירת מחדל של פרמטר זה הינו 6325.

מקרי קצה שלא היו מוגדרים בהנחיות

1. כאשר אחד הלקוחות מתנתק מהשרת ע"י הזנת Q ל-stdin שלו, ההודעה על מצב הערימות הנוכחי נשלחת לכל המחוברים ולא רק לצופה שהפך לשחקן (אם היה כזה) או זה שהתור עבר אליו מהשחקן המתנתק.
2. הודעה על דחיית הלקוח נשלחת ללא שימוש ב-buffer השליחה כיוון שלקוח נדחה אינו מוקצה משתנה מטיפוס Client עם buffer משלו. כמו כן ההנחה היא שהודעה שתשלח מיד אחרי ניסיון הלקוח להתחבר, כלומר הלקוח זמין לשליחה ולכן סיכויים לתקלה בשליחה זניחים.
3. ע"מ למנוע סגירות חיבורים עם הלקוח החווה בעיות זמניות בקו שלו, ל-socket עבורו פונקציית receive מחזירה 0 ניתנים מספר ניסיונות נוסף המוגדר ע"י קבוע RX\_TIMEOUT כדי לאפשר לו לסיים את השליחה ורק אחרי זה החיבור שלו נסגר.

במקרה של שגיאות באחת הפונקציות בהן משתמשת התוכנית ושלא הוגדרה לגביהן התנהגות מסוימת בהנחיות, התוכנית תדפיס הודעת אזהרה על המסך ותפסיק את ריצתה.