מבוא לתקשורת מחשבים – PA2

איתמר שור 315129551
עומר גיל 211453907

Defines

- **#define MAX_ROW_LEN 500 –** הגדרת אורך שורת קלט מקסימלי, הנחנו ארביטרירית כי לכל היותר 500.
- **#define MAX_LINKS 9999 –** הגדרת כמות חיבורים מקסימלית, נובע מהדרישה לתמיכה בסדר גודל של אלפי חיבורים.
- **#define IP_ADDR_MAX_LEN 16 –** הגדרת אורך מקסימלי למחרוזת המייצגת כתובת IP חוקית, נובע ישירות מהגדרת כתובת IP חוקית (בתוספת null terminator).

מבני נתונים

- queue.h/queue.c:
  - מימוש תור ע"י רשימה מקושרת, תומך בפונקציות הבאות:
    - push_back()
    - pop_front()
- Minheap.h/minheap.c:
  - מימוש ערימת מינימום, נעזרנו במימוש של ערימת מקסימום מהמקור הבא https://algorithmtutor.com/Data-Structures/Tree/Binary-Heaps/.
    הערימה ממומשת בתור ערימה בינארית בעזרת מערך בגודל קבוע – MAX_LINKS.
    תומכת בפונקציות הבאות:
    - extract_min(), get_min_key()
- hashtable.h/hashtable.c/uthash.h:
  - השתמשנו בספריית open source של hash table (uthash.h) להלן קישור לuser guide בו השתמשנו https://troydhanson.github.io/uthash/userguide.html#_structure_keys.
    בקובץ hashtable.c, יצרנו פונקציות עוטפות לפונקציות הספרייה שייבאנו (לשימוש נוח יותר)
    תומך בפעולות הבאות:
    - search_link()
    - remove_link()
    - insert_link()

חלוקה למודלים

- **GPS_simulator.h/GPS_simulator.c -**

מודול עזר (שמבצע את רוב הheavy lifting), שמסמלץ את תהליך תיעדוף סדר שליחת הפקטות, ע"י מימוש גרסת online של אלגוריתם הGPS (שראינו בתרגול).
התיעדוף נעשה ע"י חישוב זמנים וירטואליים – last ו- round, כאשר החישוב תלוי במצב המערכת עד הרגע הנוכחי בלבד (סכום משקלי הקשרים הפעילים), ובפרמטרי הפקטה הנוכחית (גודל, משקל הקשר וכו).
המודול מתחזק את מצב המערכת בעזרת מבני נתונים ומשתנים שונים:
משתנים:

- **double weights-** משתנה המחזיק את סכום משקלי הקשרים הפעילים כרגע.
- **double next_vir_departure_time-** משתנה המחזיק את זמן העזיבה הוירטואלי הקרוב ביותר. משמש להכרעת המצב הבא – שליחת פקטה או הגעה של פקטה (בזמנים וירטואלים).
- **double round_t-** משתנה המחזיק את זמן ההגעה הוירטואלי של הפקטה האחרונה שעובדה במערכת.
- **double last_real_time-** משתנה המחזיק את זמן ההגעה (האמיתי) של הפקטה האחרונה שעובדה במערכת.
- **int nof_packets-** משתנה המחזיק את מספר הפקטות שנמצאות כרגע במערכת (שטרם נשלחו).

מבני נתונים:

- **hash_table-** טבלת hash שגודלה דינאמי. כל תא בטבלה מייצג קשר ומכאן המפתח הינו הרביעייה שמגדירה קשר (**struct local_link**). הערך של תא הינו אובייקט מסוג **struct queue,** שמחזיק את הפקטות מאותו הקשר, לפי סדר הגעה.
- **2 ערימות מינימום:**
  - **Virt_departure_heap (נסמן בvdh לשם נוחות)**
  - **real_departure_heap (נסמן בrdh לשם נוחות)**

מאופי האלגוריתם, יתכנו מצבים בהם נשלחה פקטה, אך עדיין משפיעה על חישוב הזמנים הוירטואלים של פקטות אחרות (ולהפך- פקטה טרם נשלחה, אך לא משפיעה על חישובי זמנים). לכן, בחרנו לתחזק 2 ערימות מינימום של פקטות, שבכל אחת מהן המפתח הוא הlast_time של הפקטה.
בvdh הערך של כל איבר הינו התור שמיצג את הקשר של אותה פקטה, וbrdh הערך הינו הפקטה עצמה.
הvdh משמשת לחישובי הזמנים הוירטואלים ולקביעת האירוע הבא שיתרחש (לפי האיבר המינימלי בערימה), ע"י עדכון של המשתנה next_vir_departure_time.
הrdh משמשת לשליחת הפקטות בפועל (בכל שליחה נבחר את הפקטה עם המפתח המינימלי מהערימה ונשלח אותה)

תומך בפעולות הבאות:

- receive_packet()- מקבל פקטה ומעבד אותה – מחשב את זמניה הוירטואלים לפי מצב המערכת הנוכחי. בעת הצורך, מעדכן את מצב המערכת הנוכחי בהתאם לאירוע המערכת הבא.
- Send_packet()- מדפיס לstdout את נתוני הפקטה שנבחרה לשליחה (לפי הrdh).

- **WFQ_scheduler.c-**
מודול המשמש כmain של הפרויקט. אחראי על קריאת פקטות מהstdin, ולפי מצב המערכת (יפורט בהמשך) קורא לפונקציות receive_packet()/send_packet() של הGPS_simulator.
מצב המערכת נקבע עפ"י ערך המשתנה next_departure_time.
בהינתן פקטה שהגיעה בזמן t:
  - אם $t \leq \text{next\_departure\_time}$:
כלומר בזמן t ישנה פקטה שנשלחת ולכן נקרא לrecieve_packet().
  - אחרת, המערכת פנויה לשליחת פקטה חדשה ולכן נקרא קודם לsend_packet() (כדי לתעדף פקטות שכבר עובדו במערכת וטרם נשלחו) ולאחר מכן לrecieve_packet() כדי לעבד את הפקטה החדשה שהגיעה.

ניתוח סיבוכיות אסימפטוטית

נסמן את סך כל הפקטות שמגיעות למערכת במשך כל התהליך בn.
כל פקטה מעובדת פעם אחת בדיוק (receive_packet) ונשלחת פעם אחת בדיוק (send_packet).
סך הפעולות לכל עיבוד פקטה:

- חיפוש הקשר המתאים בhash_table- אסימפטוטית **O(1).**
- הוספה לתור הקשר – **O(1).**
- הוספה ל2 ערימות המינימום – **O(log(n))** בworst case.
- חישובי זמנים וירטואליים ועידכון מצב המערכת –amortized **O(1)** (בשיטת המטבעות, כל פקטה שומרת מטבע בצד, "שמשלם" על הוצאתה מהערימה)

סה"כ – **O(log(n))** אסימפטוטית.

סך הפעולות לכל שליחת פקטה:

- הוצאה מערימת הrdh– **O(log(n))** בworst case.

סה"כ – **O(log(n))** אסימפטוטית.

מכאן, סיבוכיות זמן הריצה של התוכנית היא $\boldsymbol{O(nlog(n))}$.