Distributed Systems 236351 Home Assignment 2

**Decentralized uber**

Shai Porath 312434863 [shai.porath@campus.technion.ac.il](mailto:shai.porath@campus.technion.ac.il)

Hagar Sheffer 205814627 [hagar.s@campus.tecnion.ac.il](mailto:hagar.s@campus.tecnion.ac.il)

**המערכת שבנינו תומכת בשתי פונקציות:**

1. Post ride – משתמש יכול להירשם כנהג במערכת עבור נסיעה בין שתי ערים בתאריך מסויים.
2. Post path planning request – משתמש יכול לפרסם מסלול העובר ברשימת ערים בתאריך מסויים. אם קיימת קבוצת נסיעות המספקת את המסלול, ישמר עבורו מקום בכל אחת מהנסיעות ונחזיר את פרטיהן. אחרת נחזיר שאין אפשרות לספק את המסלול.

**המערכת שלנו עומדת בדרישות הבאות:**

**Atomicity**

**Fault tolerance**

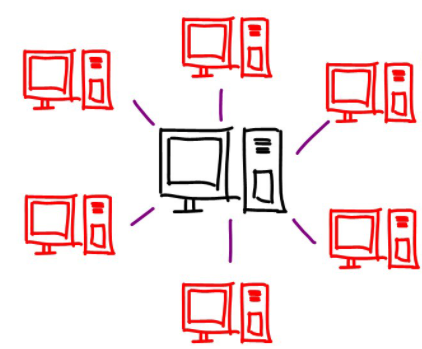
**Linearizability**

**Reliability**

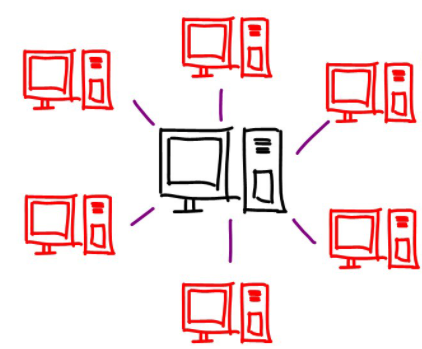
**Timeliness**

**consistency**

**מבנה הרשת**

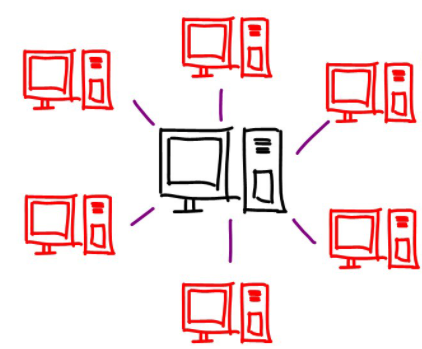


Shard A

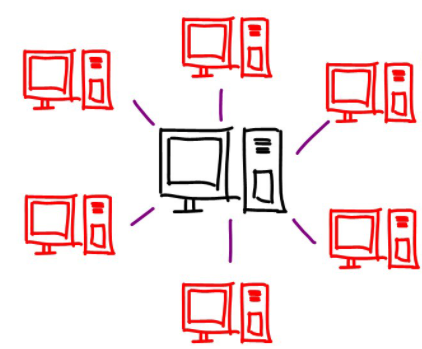


Shard C

Shard B



Load balancer



client

REST

gRPC



**מאזן עומסים (load balancer)**

שירות זה אחראי לקבל בקשות ממשתמשים באמצעות REST ולהעביר כל בקשה לשרת האיזורי המתאים, תוך התחשבות בשני פרמטרים:

1. השרת שייך לאיזור האחראי על העיר ממנה יוצאת הנסיעה המבוקשת.
2. מבין שרתי האיזור המתאים, הבקשה תישלח לשרת הכי פחות עמוס.

איזון העומסים יתבצע בשיטת round robin. יתרונות בשיטה זו:

* השיטה פשוטה למימוש
* יכולות החישוב של השרתים השונים זהה
* עלות החישוב בין בקשות path planning דומה וכן ובין בקשות post ride ואנו מניחים פיזור אחיד של הבקשות. שי ?
* תקורה נמוכה – כיוון שמאזן העומסים מקבל את כל הבקשות מהמשתמשים ומעביר אותם לשרתים האיזוריים הוא מהווה צוואר-בקבוק במערכת. לכן העדפנו שהחישובים שהוא מבצע יהיו קצרים ככל הניתן.

מימוש מאזן עומסים תורם לפיזור העומסים, לשקיפות פרטי המערכת מול המשתמש וכן מהווה נקודת כניסה יחידה למערכת בה ניתן להציב חומת אש, ובכך משפר את הבטיחות. בנוסף, ע"י שימוש ב-zookeeper מאזן העומסים מעודכן בכל רגע ברשימת השרתים החיים ולא יפנה בקשות לשרת שנפל. בכך הוא תורם להתאוששות מהירה של המערכת מנפילות.

Fault tolerance – כדי להבטיח עמידות בפני כשלונות, לload balancer קיימים גיבויים המוכנים להחליף אותו במקרה שיפול באופן שקוף למשתמש. הרפליקציות ממומשות בשיטת primary-backup replication כאשר הbackup הוא ב-hot standby – כלומר שהוא מוכן בכל רגע להחליף את ה-lb הנוכחי שנפל. זאת כדי לספק זמינות גבוהה.

**צביר איזורי (shard cluster)**

כל איזור מחזיק תחתיו מספר ערים כך שהוא אחראי לבצע בקשות היוצאות מהן. הרפליקציות בצביר האיזורי מבוססות total order – לכל צביר יש מנהיג יחיד, וכל שינוי בstate של הצביר חייב לעבור דרכו כדי שיקבע את הסדר שבו הגיעו הבקשות. כלומר, כל שרת בצביר יכול לטפל בבקשה ולפני שהוא שומר את התוצאה הוא מעביר אותה למנהיג כך שיפיץ אותה לכל חברי הצביר. הרחבה על האופן שבו מתרחש הcommit והוכחה שהוא שומר על מצב תקין בכל רגע גם במקרה של נפילות וכשלנות, בהמשך.

Fault tolerance – נניח שבכל צביר חברים n שרתים. כיוון שכל חברי הצביר הם רפליקציות, אנו יכולים להתמודד עם נפילה של n-1 שרתים מכל צביר.

Linearizability – ע"י הבטחת total order באמצעות השימוש במנהיג אנו מב

**Zookeeper**

אנו משתמשים בzookeeper כדי לממש failure detector, group membership