Part 1 - Answers

.

c) For implementations in a) and b) in the worst case the lookup time is equal to the maximal prefix length. Assume that you have an explicit constraint on a number of lookups. For instance, 8 lookups in the table containing 32-bit prefixes. Suggest your ways how to address a fundamental tradeoff between lookup time and the required memory to represent this table.

הצעה ראשונה:

route caching in conjunction with linear search to speed up the lookup time. A cache is a fast buffer for storing recently accessed data. The main use of the cache is to speed up subsequent access to the same data if there is a sufficient amount of locality in data access requests.

For lookup the cache stores the recently seen 32-bit destination addresses and the associated next-hop information. When a packet arrives at the router, the destination address is extracted and the route cache is consulted.

Profs: we have 8 lookups only to find, and there is more chance to find it in the Cache of storing the recently accessed data.

Cons: locality exhibited by flows in the backbone has been observed to be very poor. This leads to a much lower cache hit ratio and degenerates to a linear search for every lookup.

In summary, caching can be useful when used in conjunction with other algorithms, but that precludes the need for fast prefix lookups.

הצעה שנייה עדיפה:

מספר החיפושים במימוש שבו כל צומת בעץ שומרת רק ביט אחד יכול להגיע במקרה הגרוע ל32 קפיצות.

כדיי להקטין את מספר הקפיצות ואת עומק העץ, נשמור בכל נוד 4 ביטים במקום 1, וככה נוכל למצוא את המבוקש בלא יותר מ8 קפיצות.

העומק המקסימלי הוא 4 לעומת 16.

d) Think about alternative ways to represent a table of prefixes with the desired balance between lookup time and memory requirements. Compare cons and pros of various proposals. More creative proposals are better valued. Not necessary binary trie should be considered.

הצעה:

שמירת כל prefix וה action ב hashmap , מתאים לרשתות שאינן מורכבות ובעלות מספר קטן של prefixes\actions, שכן, בעזרת O(1) ישר נוכל למצוא לפרפיקס את הAction המתאים, חשוב לקחת בחשבון שאם יהיו לנו הרבה כאלה אז יהיה שימוש בכמות זיכרון גבוהה.

Part 1 – input example

ADD 255.255.128.0/17

11111111.11111111.10000000.00000000

ADD 255.255.0.0/17

11111111.11111111.00000000.00000000

FIND 255.255.5.5

11111111.11111111.00000101.00000101

FIND 254.255.255.255

11111110.11111111.11111111.11111111

FIND 255.255.128.0

11111111.11111111.10000000.00000000

Prefix_table

```
shaharnik@ShaharNik:/mnt/c/Users/Shahar/Desktop/316416668/1_316416668$ ./prefix_table sample_input.txt
Added 255.255.128.0/17 A at the depth 17, total Nodes 17
Added 255.255.0.0/17 A at the depth 17, total Nodes 18
Found 255.255.5.5 A at the depth 17
Not Found
Found 255.255.128.0 A at the depth 17
```

Prefix_table_opt

```
shaharnik@ShaharNik:/mnt/c/Users/Shahar/Desktop/316416668/1_316416668$ ./prefix_table_opt sample_input.txt
Added 255.255.128.0/17 A at the depth 17, total Nodes 17
Added 255.255.0.0/17 A at the depth 16, total Nodes 16
Found 255.255.5.5 A at the depth 16
Not Found
Found 255.255.128.0 A at the depth 16
```