Network Processors
Using SPLAY Tree's for state-full packet classification

Sepehr Rezaei
Instructor: Dr. Mahdi Abbasi
dept. of Computer Engineering
Bu-Ali Sina University
Hamedan, Iran
sepehr.rezaei@eng.basu.ac.ir

1. Splay Trees

Splay trees were built to use the notion that in most real world applications, 80% of the accesses to a dataset occur on 20% of the data. In order to take advantage of this splay trees work by splaying the node that was most recently accessed to the root of the tree. Splaying a node means moving the node to the root using tree rotations so we maintain the BST properties. Doing this not only modes the node that was most recently accessed to the top of the tree, but also moves nodes that are close to that one further up the tree, so accesses to keys that are close to each other will in theory get better performance. The operations are implemented like a regular BST, except we splay the node we accessed at the end of the access (this is done in find, insert and remove).

1.1. Splaying a node

To splay a node, splay trees use simple rotations similar to the rotations previously viewed in Red Black trees and AVL trees. However in Splay trees they are named differently, Zig for a right rotation and Zag for a left rotation. We can also have combinations of the two with Zig-Zig, Zag-Zag, Zig-Zag and Zag-Zig which combines the two simple rotations into a more complex rotation. To splay a node, we must recursively apply combinations of these rotations until the accessed node has reached the root of the tree. If the node has a grand-parent, that means it's still at least 2 rotations away from becoming the root, so we apply 2 rotations. When the node does not have a grandparent it means that it's at most 1 rotation away from becoming the root of the tree. To decide which rotation to use, we first check whether we have a grandparent or not.

If we do have a grandparent, we have to decide which of the 4 rotations (Zig-Zig, Zag-Zag, Zig-Zag, Zag-Zig) is going to be chosen. The 4 possibilities are:

1. **Left left case**: When the parent is a left child of the grandparent and the node is a left child of the parent. In this situation, we will apply a **Zig-Zig** rotation.
2. **Left right case**: When the parent is a left child of the grandparent and the node is a right child of the parent. In this situation we will apply a **Zag-Zig** rotation.
3. **Right left case**: When the parent is a right child of the grandparent and the node is a left child of the parent. In this situation we will apply a **Zig-Zag** rotation.
4. **Right right case**: When the parent is a right child of the grandparent and the node is a right child of the parent. In this situation we will apply **Zag-Zag** rotation.

If we do not have a grandparent, then there are just two possibilities:

1. We are a left child of the root, which means we will need a Zig rotation.
2. We are a right child of the root, which means we will need a Zag rotation.

By recursively applying these rotations to the node we want to splay it will rotate the tree until we have the node at the root.

1.2. Complexity of splaying

By splaying a node, we can come to a situation where we have a tree that does not have a height of $O(\log(N))$, but instead $\theta(n)$. In this case, accessing and splaying the deepest node in the tree would take $\theta(n)$ time, resulting in a worst case complexity of $O(n)$ for insert,

contains and remove. However, when we perform a splay operation on the deepest node, we will then get a tree that has an average node depth of roughly half compared to the tree before the access.
9To get an accurate measurement of the complexity in splay trees, we must use Amortized Analysis. We will not discuss how this analysis is done, but by employing it the complexity we obtain for m operations on a given splay tree will result in a complexity of O(mlog(n)) which averages O(log(n)) complexity per operation.

1.3. Advantages and disadvantages

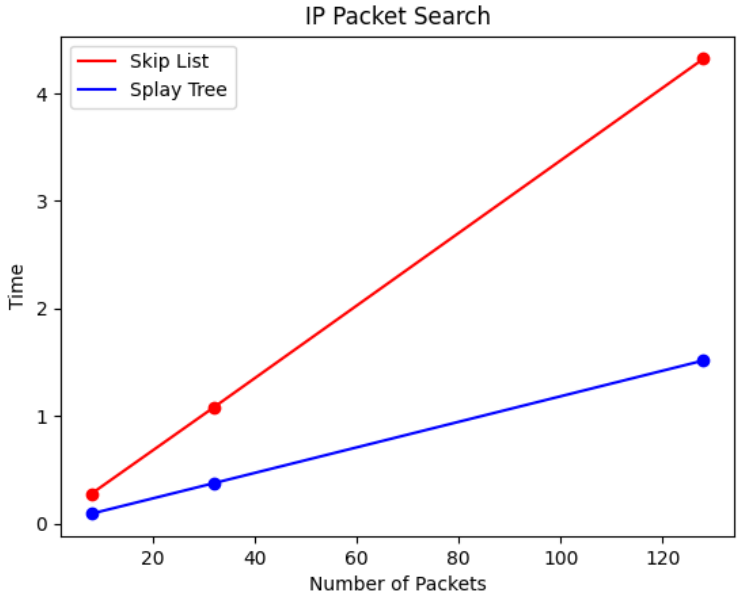
Some advantages of using Splay trees:

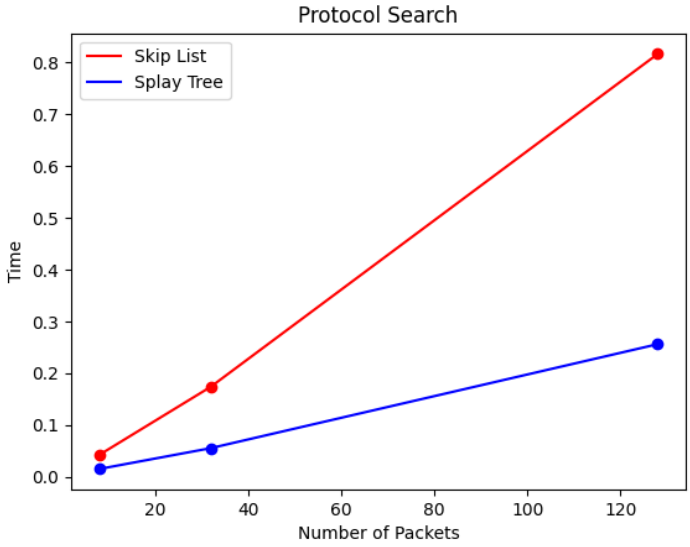
- No memory overhead on each node for storing state.
- If our use case follows the 80% of accesses accessing 20% of the dataset, we will see visible performance improvements.

Some disadvantages of using Splay trees:

- Randomized accesses might lead to very bad (Near linear) performance.
- We can get situations where we have θ(N) operations, instead of a constant worst case O(log(n)).

2. Results




Protocol Search
Skip List
Splay Tree
Time
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
20
40
60
80
100
120
Number of Packets