IP Address Finder

Chitti babu Yelugubandi

Title:

IP Address Finder

Aim:

To design an algorithm capable of searching data with maximum efficiency.

Objective:

To utilize splay trees the most efficient data structure for searching and insertion—to accomplish the aim.

Abstract:

This project uses the **splay tree data structure** to construct a tree where nodes store the IP addresses of devices connected to a network router. In this implementation, we consider 11 devices connected to one router, all sharing a common network prefix in their IP addresses. Only the unique segments of each device's IP address are stored in the nodes.

When the router receives data packets from the internet, it must identify the correct device using its IP address. To enhance this process, splay trees are employed. These trees dynamically adjust their structure, ensuring that frequently accessed nodes are moved closer to the root, optimizing future searches and insertions.

In this simulation, random values are generated to represent data packets, enabling a fully automated process without the need for manual input. The implementation effectively models how real-world routers forward data packets to multiple connected devices by quickly identifying the appropriate IP address using a splay tree.

Principle:

The splay tree operates on the principle that the most frequently accessed IP addresses are moved closer to the root, progressively reducing the time complexity of subsequent searches.

Methodology:

The splay operation rearranges the tree to ensure the most frequently searched or inserted IP addresses are positioned at the root, streamlining access.

Architecture:

This section includes a block diagram (as outlined in the original document) to illustrate the system's structure.

Code Modules:

1. **Starting Module:**
    - Includes the necessary library files (stdio.h, stdlib.h) and defines the structures for nodes and the splay tree.
2. **Process Module:**
    - Implements key functions such as node creation, tree creation, rotations, splaying, insertion, and searching.
3. **Implementation Module:**
    - Contains the main() function that initializes the tree, inserts nodes, assigns random data packets, and prints the final structure using an in-order traversal.

Time Complexity Analysis:

By dynamically adjusting its structure, the splay tree ensures logarithmic average-case complexity for search and insert operations, making it ideal for handling high volumes of data packets in real time

Conclusion:

In scenarios where multiple devices are connected to a single router and large volumes of data packets need to be routed in milliseconds, the splay tree is a highly effective data structure. Its ability to adapt dynamically ensures efficient data packet delivery even under heavy load.