COMPUTER NETWORKS PROJECT

DYNAMIC NETWORK ROUTING

PERFORMANCE ANALYSIS AND IMPLEMENTATION

NIMISHA THALLAPALLY
CS22B1082

ABSTRACT

In today's increasingly complex networking environments, the need for efficient and adaptive routing protocols is important. This project aims to develop a dynamic network routing protocol that optimizes data paths in real-time by utilizing shortest path algorithms, specifically Dijkstra's and Bellman-Ford.

The implementation will adapt to changes in network topology, such as varying traffic loads and node failures, ensuring reliable and efficient data transmission. Dijkstra’s algorithm and Bellman-Ford will be utilized for networks that may include negative weights, offering greater flexibility in handling dynamic changes.

Although the Floyd-Warshall algorithm could theoretically be used for all-pairs shortest path calculations, its implementation in dynamic network routing is costly and not suitable for real-time applications where efficiency, responsiveness to changes, and memory usage are critical factors..

AIM OF THE PROJECT

- The project focuses on the implementation and comparison of three key shortest path algorithms.
  - Dijkstra’s Algorithm
  - Bellman-Ford Algorithm
  - Floyd-Warshall Algorithm
- Simulate and compare the performance of Dijkstra's, Bellman-Ford, and Floyd-Warshall algorithms.
- Evaluate the adaptability and efficiency of each algorithm in dynamic network scenarios.
- Investigate the algorithms' response to network topology changes, such as node failures.

FLOYD WARSHALL ALGORITHM

- Traditional algorithms like Dijkstra's and Bellman-Ford are designed for single-source shortest path calculations, which are ideal for situations requiring paths from one node to all others.
- However, in scenarios involving multiple path queries (e.g., network routing, real-time systems), these algorithms may not be as efficient.

Floyd-Warshall Advantage:

- All-Pairs Shortest Path Algorithm: Computes the shortest paths between all pairs of nodes simultaneously, making it suitable for networks where many path queries are needed.
- Provides a comprehensive view of the entire network's connectivity.

CHALLENGES

Computational Complexity:

- **Time Complexity:** Floyd-Warshall operates with a time complexity of $O(n^3)$, which can be highly computationally expensive for networks with large numbers of nodes (n).
- **Space Complexity**: The space complexity is also high, requiring $O(n^2)$ space for storing the distance matrix, which becomes challenging for networks with many nodes.

Scalability Issues:

Large-scale Networks:

- In very large networks, the algorithm’s cubic complexity can lead to significant delays in computation. This is especially problematic for real-time systems or large-scale applications like internet backbone networks or global-scale traffic routing.

Limited Adaptability to Dynamic Networks:

- While Floyd-Warshall gives a complete snapshot of the network's state, it is not optimized for frequently changing network topologies. The algorithm may need to be re-run after each network change (e.g., when a node fails or a new link is added), adding overhead in dynamic environments.

IMPLEMENTATION DETAILS

- Tools and Technologies:
  - Programming Language: Python
  - Networking: Socket programming for client-server communication
  - Testing Framework: Python’s time module for measuring response times.

- Servers:
  - Each algorithm (Dijkstra, Bellman-Ford, Floyd-Warshall) runs on a separate server.
  - The servers listen for incoming requests and compute the shortest path.
- Client:
  - A common client sends requests to different servers for shortest paths and measures round-trip times.

https://github.com/nimishathallapally/networkRouting

ANALYSIS

- **Testing and Results**
  - **Testing Strategy:**
    - Each algorithm was tested with multiple scenarios, including normal and failure cases.
    - Metrics such as round-trip time and performance under node failure were measured.
  - **Key Results:**
    - **Dijkstra's Algorithm:** Fast and efficient.
    - **Bellman-Ford:** Moderate performance, with resilience to node failures.
    - **Floyd-Warshall:** Slow but comprehensive, suitable for all-pairs shortest path calculations.

OBSERVATIONS

- Key Observations:
  - Dijkstra’s algorithm is optimal for simple routing but lacks flexibility in dynamic scenarios.
  - Bellman-Ford’s ability to handle node failures gives it an advantage in real-time routing.
  - Floyd-Warshall, while comprehensive, is not practical for large networks due to its high computational cost.
- Challenges Encountered:
  - Handling network failures and adjusting algorithms accordingly.
  - Computational delays, especially with Floyd-Warshall.

FUTURE ENHANCEMENTS

- **Future improvements could include optimizing Floyd-Warshall for larger graphs and incorporating additional dynamic topology handling features or using Algorithms like A* to find the shortest path.**

- **Improved Algorithms:** Explore optimizations to make Floyd-Warshall more efficient for larger networks.

- **Real-Time Applications:** Implement the algorithms in real-time network simulation software to test scalability.

- **Node Failure Scenarios:** Enhance failure simulations to handle multiple simultaneous node failures.

REFERENCES

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009). Introduction to Algorithms, 3rd Edition.
- Forouzan, B. A. (2007). Data Communications and Networking with TCP/IP Protocol Suite (6th ed.). McGraw-Hill.
- Relevant Research Papers and Articles related to shortest path algorithms and their network applications.
  Link : https://ieeexplore.ieee.org/document/5212662

THANK YOU