University of Dhaka

Department of Computer Science and Engineering

CSE-3111 : Computer Networking Lab

Lab Report 8:

Implementation of Distance Vector Algorithm

Submitted By:

1. Syed Mumtahin Mahmud, Roll: 50

2. Nazira Jesmin Lina, Roll: 55

Submitted On :

April 6, 2023

Submitted To :

Dr. Md. Abdur Razzaque

Md Mahmudur Rahman

Md. Ashraful Islam

Md. Fahim Arefin

1 Introduction

Routing algorithms play a critical role in enabling communication between devices on a network. The Distance Vector Algorithm is a dynamic routing algorithm that computes the shortest path to each destination in a network. It is based on the Bellman-Ford algorithm and is used by distance-vector routing protocols such as RIP (Routing Information Protocol). In this lab, we will explore the operation of the Distance Vector Algorithm by simulating a network of routers and observing how routing tables are updated. The objective of the lab experiment is to implement the Distance Vector Algorithm and evaluate its performance in various network scenarios. We will also compare the Distance Vector Algorithm with Link State Algorithm to gain a deeper understanding of their strengths and weaknesses.

1.1 Objectives

- To explore the operation of the Distance Vector Algorithm and its ability to compute the shortest path to each destination in a network.
- To simulate a network of routers and observe how routing tables are updated as the algorithm runs.
- To implement the Distance Vector Algorithm, including initializing the network topology and routing tables, defining the update function, exchanging distance vectors, and checking for convergence.
- To introduce changes to the network topology and observe how the algorithm responds and reconverges.

2 Theory

The Distance Vector Algorithm is a routing algorithm used in computer networks. It is a distributed algorithm that uses a distance-vector approach to determine the shortest path for data packets in a network. The algorithm works by having each router maintain a table of the shortest known distance to every other router in the network. This table is called a routing table.

The routing table consists of a list of destination networks and the next hop router to reach that network. Initially, each router has a routing table that only contains information about its directly connected networks. Each router periodically broadcasts its routing table to its neighbors. When a router receives the routing table from its neighbor, it updates its own routing table by combining its current knowledge of the network with the information provided by its neighbor.

The Distance Vector Algorithm uses the Bellman-Ford equation to calculate the shortest path between two points. The Bellman-Ford equation is as follows:

d(v) = min c(u,v) + d(u)

Where d(v) is the distance from the source node to node v, c(u,v) is the cost to move from node u to node v, and d(u) is the shortest known distance from the source node to node u.

To compute the shortest path to all other routers, each router maintains a distance vector that represents the cost to reach all other nodes in the network. The distance vector is updated when a router receives new information from its neighbors, and the process continues until the routing tables of all routers converge, i.e., the distances to all destinations stabilize and no further changes occur.

The Distance Vector Algorithm has some advantages and disadvantages. One advantage is that it is easy to implement and requires less memory and processing power compared to other routing algorithms such as Link State Algorithm. However, it is prone to count-to-infinity problem, where incorrect distance vectors can cause routing loops and slow convergence. Also, the algorithm may not converge to the optimal solution in certain network scenarios.

3 Methodology

1. **Design and Implementation of the Program:** A program was designed and implemented , that simulates a network of routers. Each router is represented as a node, and each link is represented as an edge with a given cost. Each router has a mapping of port and router names. The program initializes the network topology, initializes the routing tables, defines the update function, simulates the exchange of distance vectors, updates routing tables, and checks for convergence.

2. **Initialization of Network Topology:** A data structure, such as an adjacency matrix or list, is created to represent the network topology. Nodes (routers) and edges (links) are defined with their corresponding costs or metrics. The distance to directly connected networks is set to the link cost, and the distance to non-directly connected networks is set to infinity.

3. **Initialization of Routing Tables:** The routing tables for each router are initialized. The routing table for each router contains entries for all other routers in the network, with the initial distances set based on direct connections.

4. **Definition of Update Function:** A function is defined that updates the routing table of a router based on the Bellman-Ford equation: new_distance = min(old_distance, neighbor_distance + link_cost). The function takes distance vector updates from neighboring routers and updates the routing table accordingly.

5. **Exchange of Distance Vectors:** The periodic exchange of distance vector updates between neighboring routers is simulated. Each router sends its current distance vector (routing table) to its directly connected neighbors.

6. **Updating of Routing Tables:** After each exchange, the update function is called to recalculate the distances in the routing tables based on the received distance vectors from neighbors. The next-hop information for each destination is also updated.

7. **Checking for Convergence:** The program determines whether the network has converged by checking if the routing tables have stabilized . If the network has not converged, the exchange and update steps are repeated. Gradual updates of paths in each node are printed.

8. **Analysis of Results:** The final state of the routing tables is analyzed, the correctness of the computed shortest paths is verified, and the algorithm's behavior in terms of convergence and response to network changes is observed.

9. **Optional Changes to the Network:** Optionally, changes to the network, such as modifying link costs or simulating link failures, can be introduced to observe how the algorithm responds and reconverges.

10. **Testing Functionality of the Algorithm:** The functionality of the algorithm is tested in various network scenarios by changing the network topology and evaluating the calculated

shortest paths. A 30-second timer can be used to randomly update an edge. The initial network topology should be configurable from a file.

4 Experimental Result

The experimental results of the implementation of the Distance Vector Algorithm were successful. The program was able to simulate a network of routers and compute the shortest path to each destination in the network using the Distance Vector Algorithm.

Screenshots and console output were obtained during the testing of the program in various network scenarios. These screenshots and console output demonstrated the correct functionality of the program in different situations, including changes in link costs and network failures. The output showed the gradual updates of the path in each node, which helped to verify the correctness of the computed shortest paths.

Initial Graph: This is the initial graph for this problem

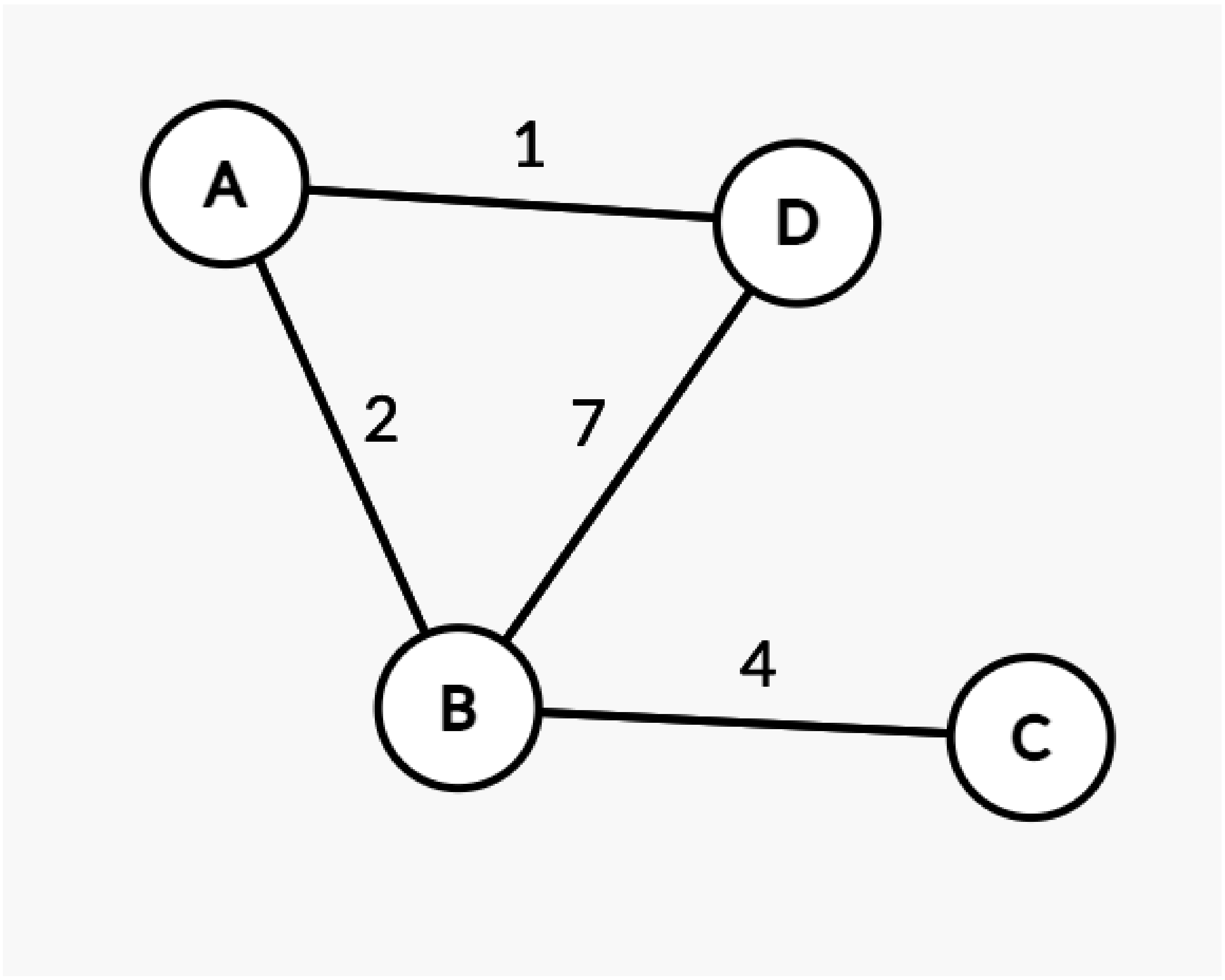
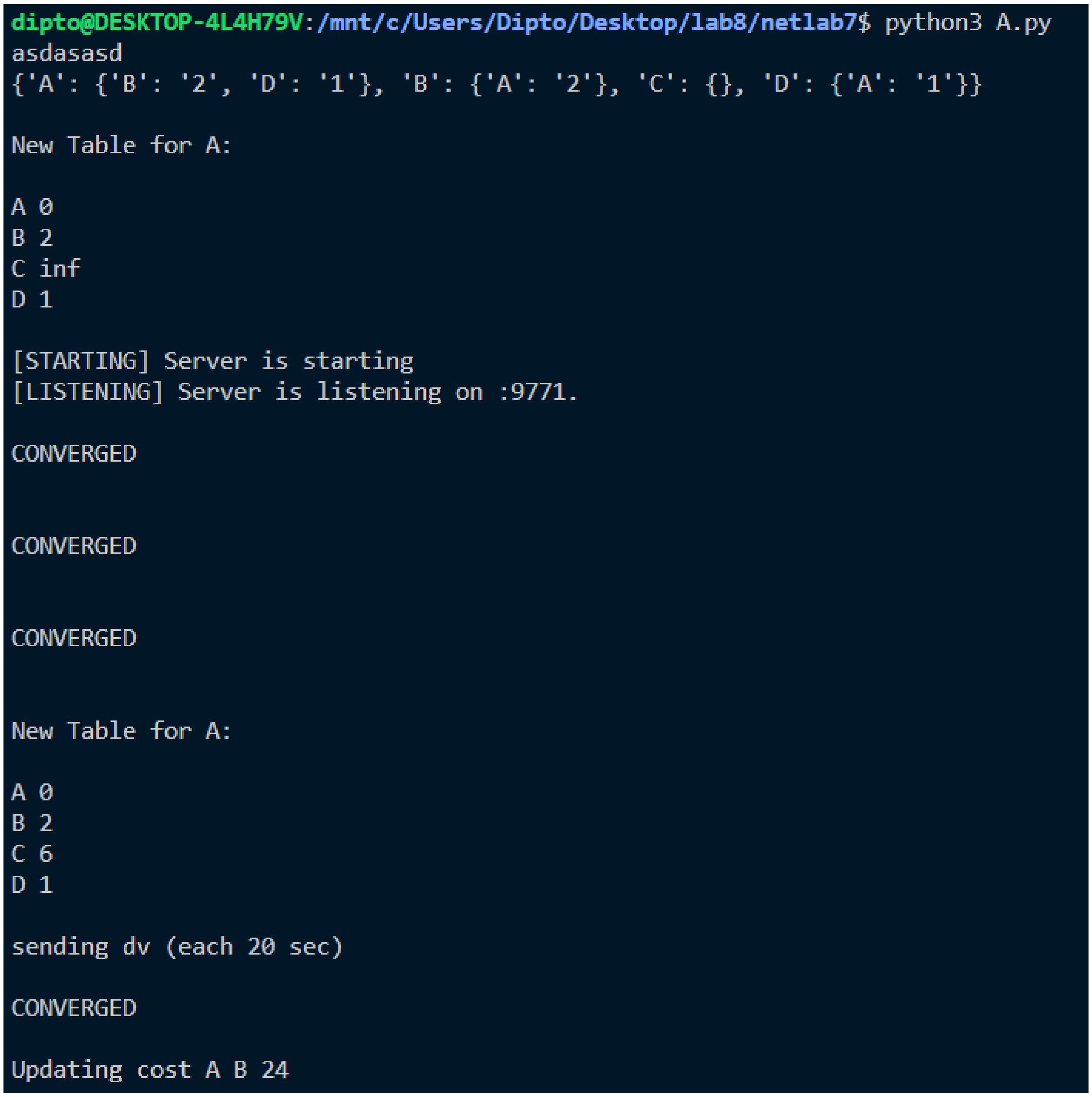


Figure 1: Initial Graph

4.1 Snapshots:

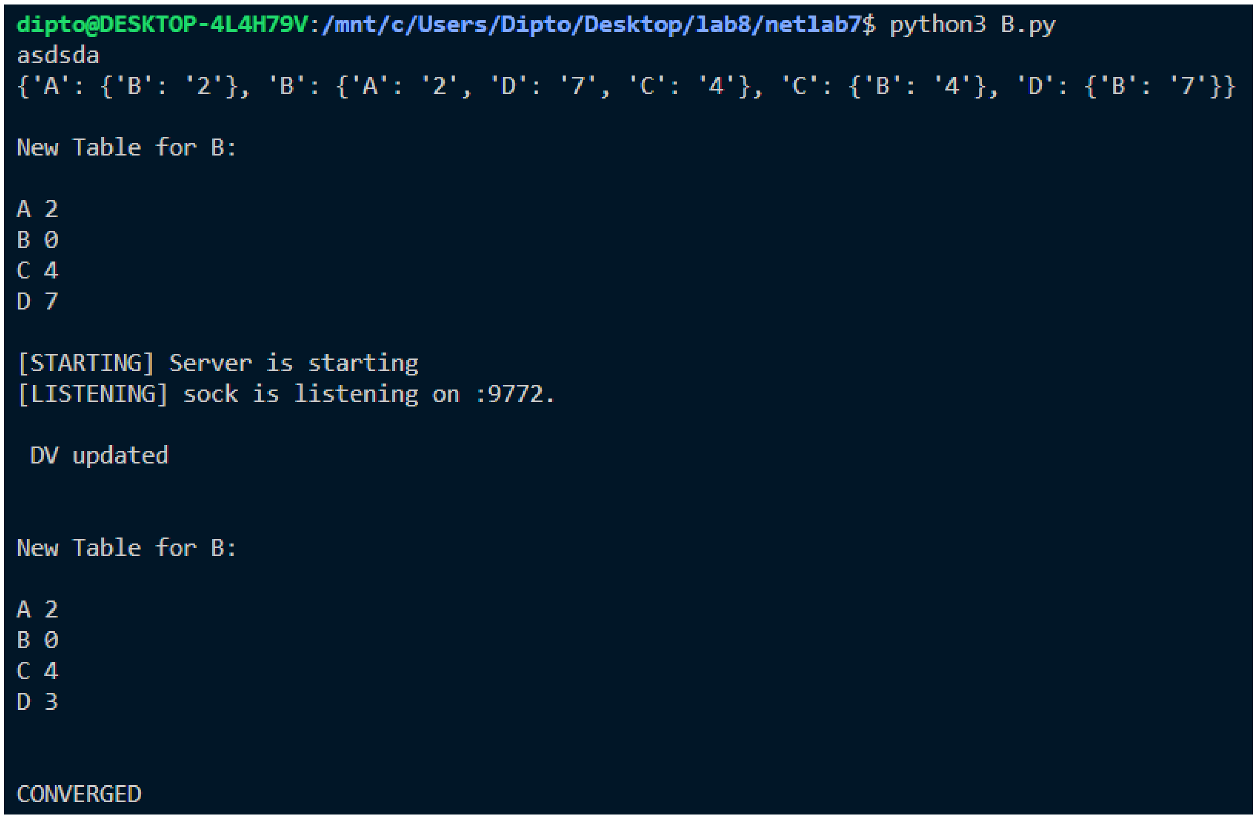
For Router A: This the experimental result(path cost) of A router for the initial graph



```
dipto@DESKTOP-4L4H79V:/mnt/c/Users/Dipto/Desktop/lab8/netlab7$ python3 A.py
asdasasd
{'A': {'B': '2', 'D': '1'}, 'B': {'A': '2'}, 'C': {}, 'D': {'A': '1'}}

New Table for A:

A 0
B 2
C inf
D 1

[STARTING] Server is starting
[LISTENING] Server is listening on :9771.

CONVERGED


CONVERGED


CONVERGED


New Table for A:

A 0
B 2
C 6
D 1

sending dv (each 20 sec)

CONVERGED

Updating cost A B 24
```

Figure 2: A router's Initial path cost

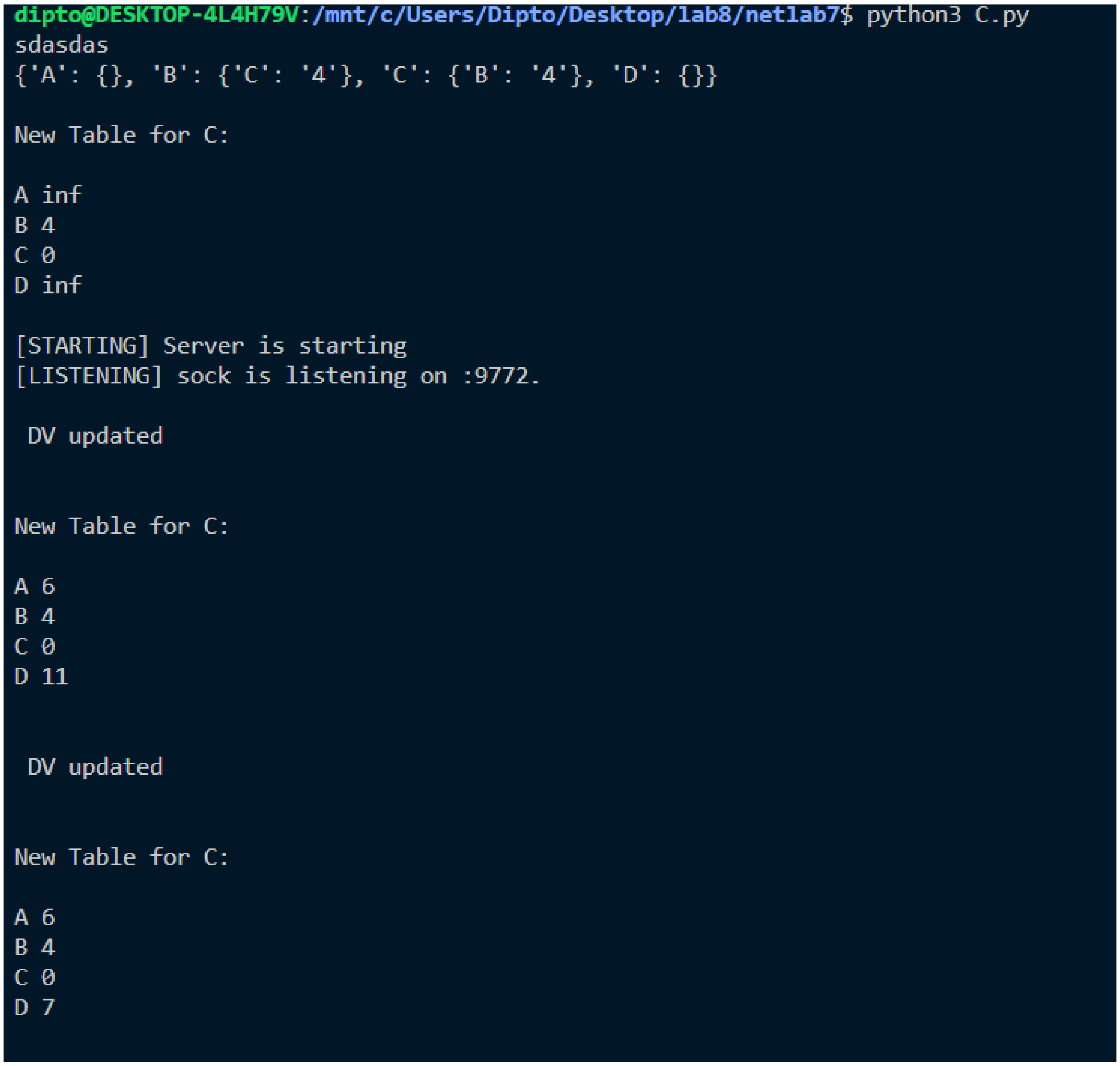
For Router B: This the experimental result(path cost) of B router for the initial graph



```
dipto@DESKTOP-4L4H79V:/mnt/c/Users/Dipto/Desktop/lab8/netlab7$ python3 B.py
asdsda
{'A': {'B': '2'}, 'B': {'A': '2', 'D': '7', 'C': '4'}, 'C': {'B': '4'}, 'D': {'B': '7'}}

New Table for B:

A 2
B 0
C 4
D 7

[STARTING] Server is starting
[LISTENING] sock is listening on :9772.

 DV updated


New Table for B:

A 2
B 0
C 4
D 3


CONVERGED
```

Figure 3: B router's Initial path cost

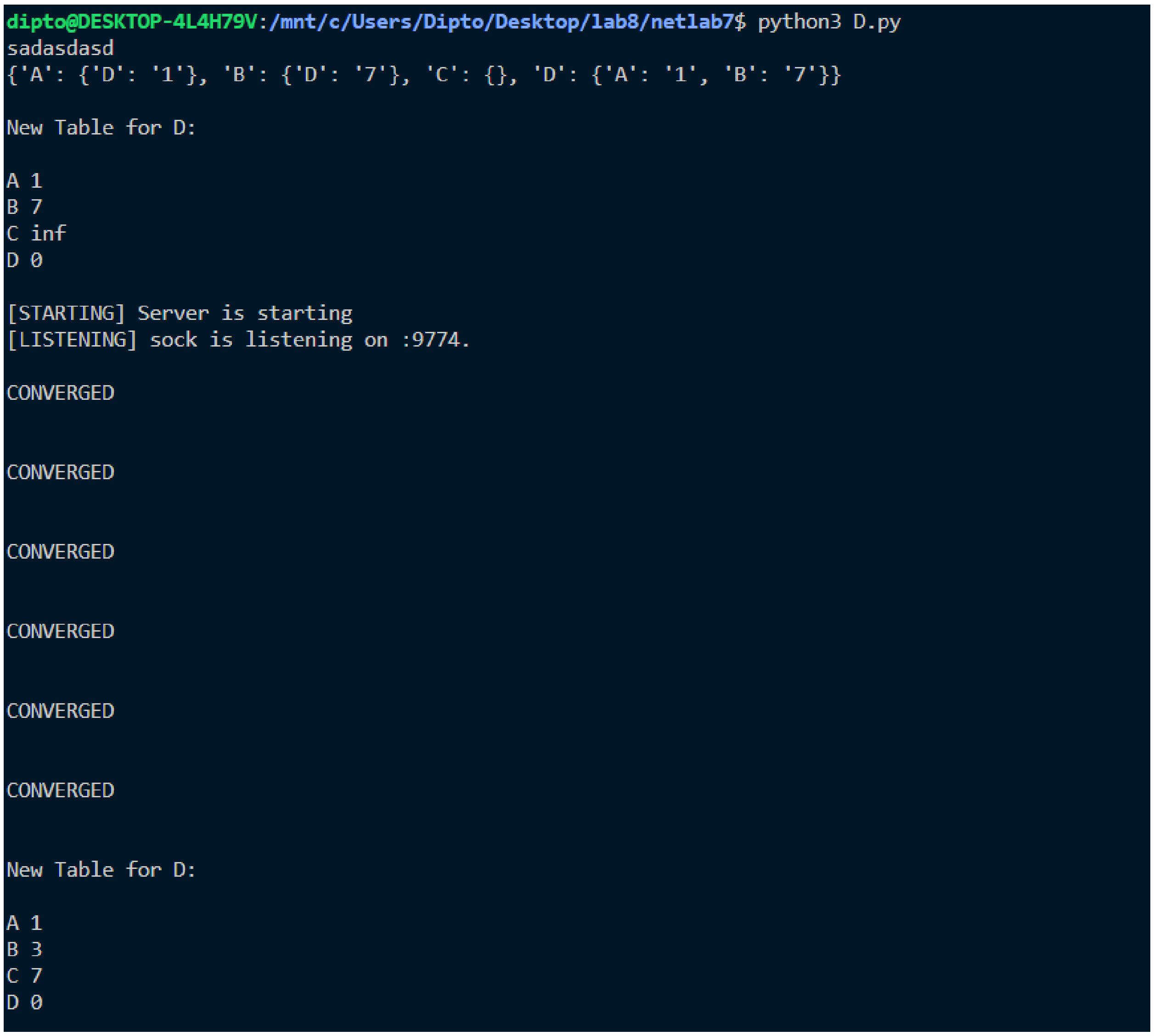
For Router C: This the experimental result(path cost) of C router for the initial graph



```
dipto@DESKTOP-4L4H79V:/mnt/c/Users/Dipto/Desktop/lab8/netlab7$ python3 C.py
sdasdas
{'A': {}, 'B': {'C': '4'}, 'C': {'B': '4'}, 'D': {}}

New Table for C:

A inf
B 4
C 0
D inf

[STARTING] Server is starting
[LISTENING] sock is listening on :9772.

 DV updated


New Table for C:

A 6
B 4
C 0
D 11


 DV updated


New Table for C:

A 6
B 4
C 0
D 7
```

Figure 4: C router's Initial path cost

For Router D: This the experimental result(path cost) of D router for the initial graph

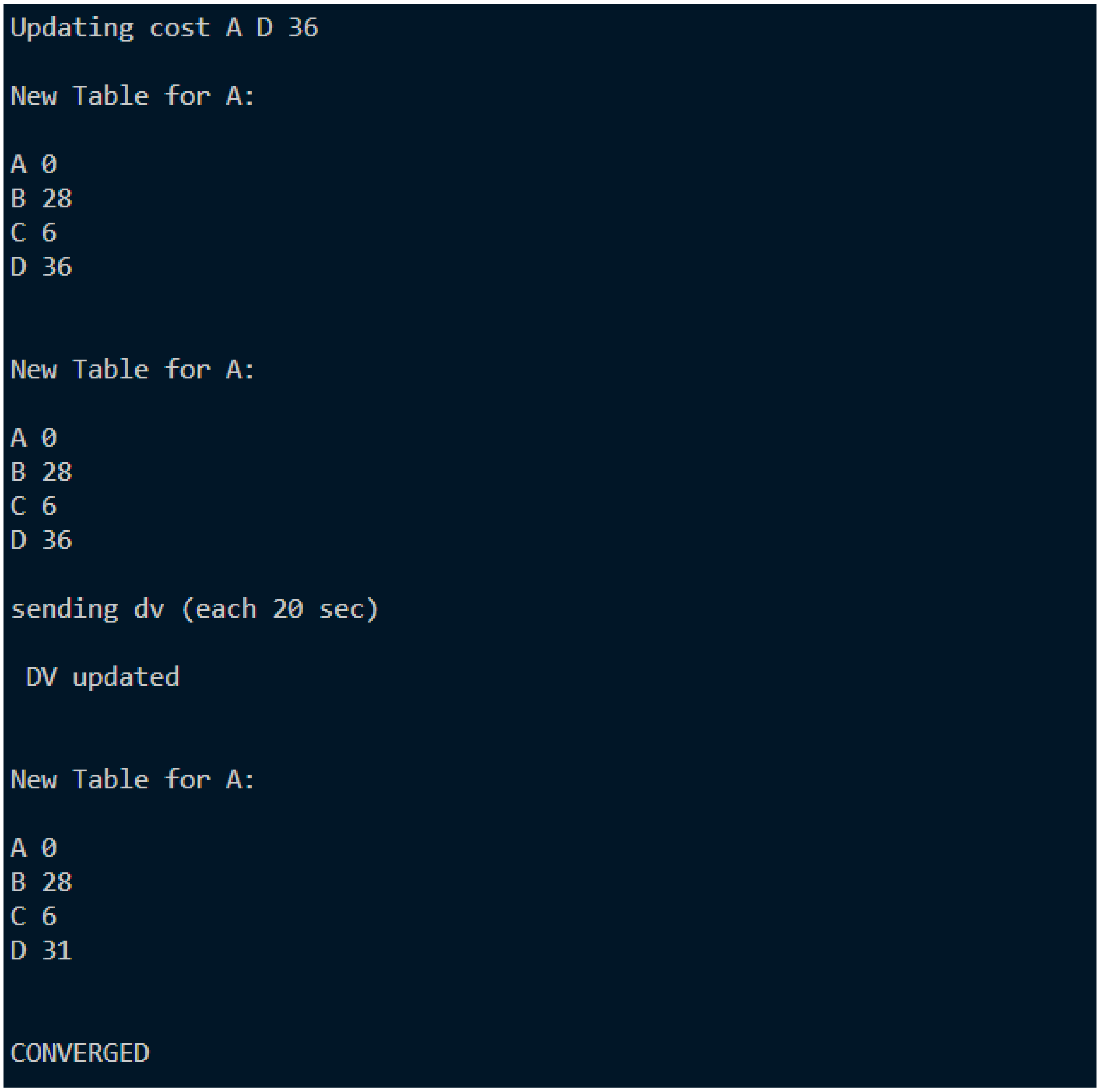


```
dipto@DESKTOP-4L4H79V:/mnt/c/Users/Dipto/Desktop/lab8/netlab7$ python3 D.py
sadasdasd
{'A': {'D': '1'}, 'B': {'D': '7'}, 'C': {}, 'D': {'A': '1', 'B': '7'}}

New Table for D:

A 1
B 7
C inf
D 0

[STARTING] Server is starting
[LISTENING] sock is listening on :9774.

CONVERGED


CONVERGED


CONVERGED


CONVERGED


CONVERGED


CONVERGED


New Table for D:

A 1
B 3
C 7
D 0
```

Figure 5: D router's Initial path cost

After 30 second, the path costs are randomly updated for each router or the new edges are added to the graph.

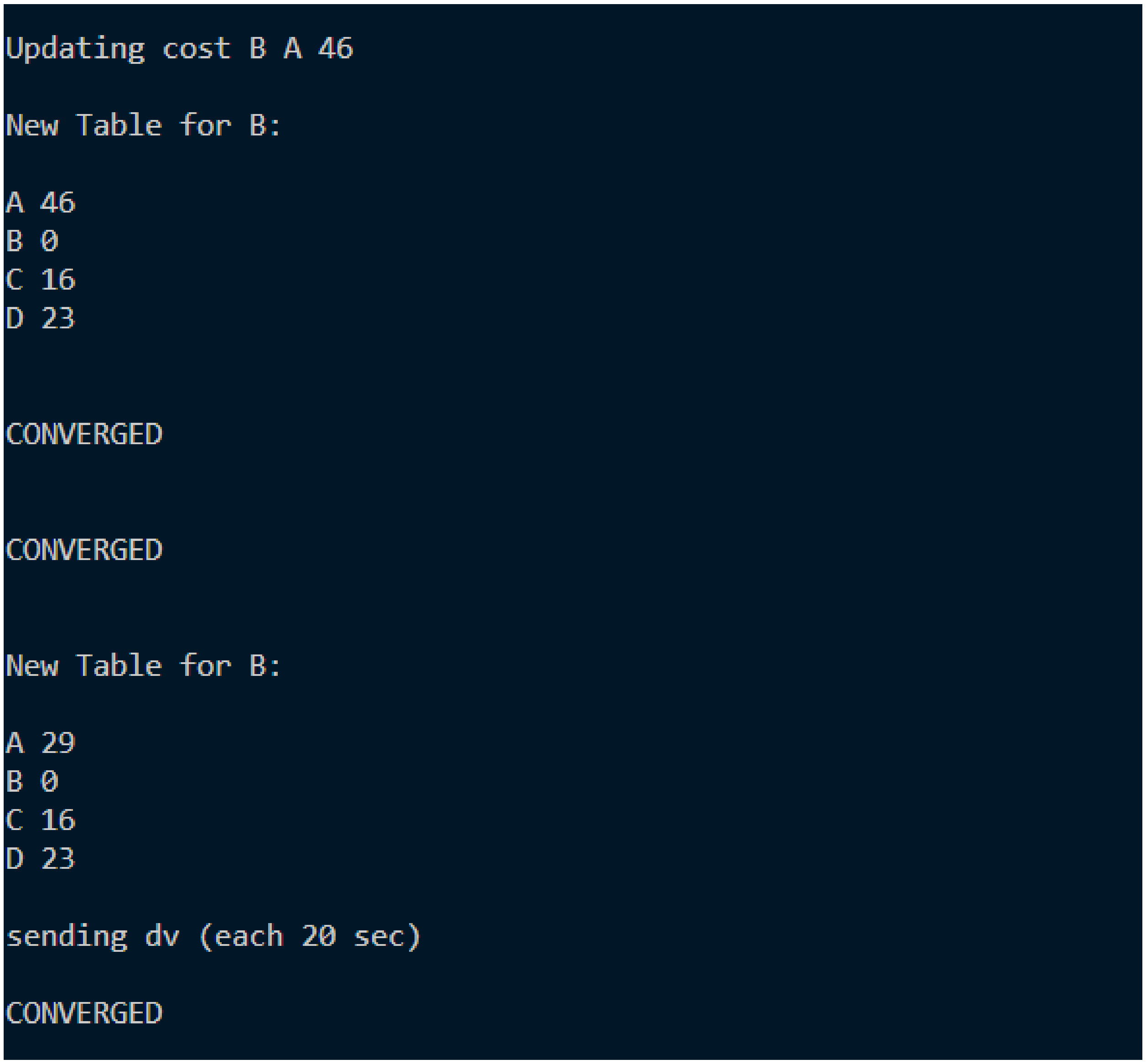
For Router A:



```
Updating cost A D 36

New Table for A:

A 0
B 28
C 6
D 36


New Table for A:

A 0
B 28
C 6
D 36

sending dv (each 20 sec)

 DV updated


New Table for A:

A 0
B 28
C 6
D 31


CONVERGED
```

Figure 6: A router's path cost after 30 sec

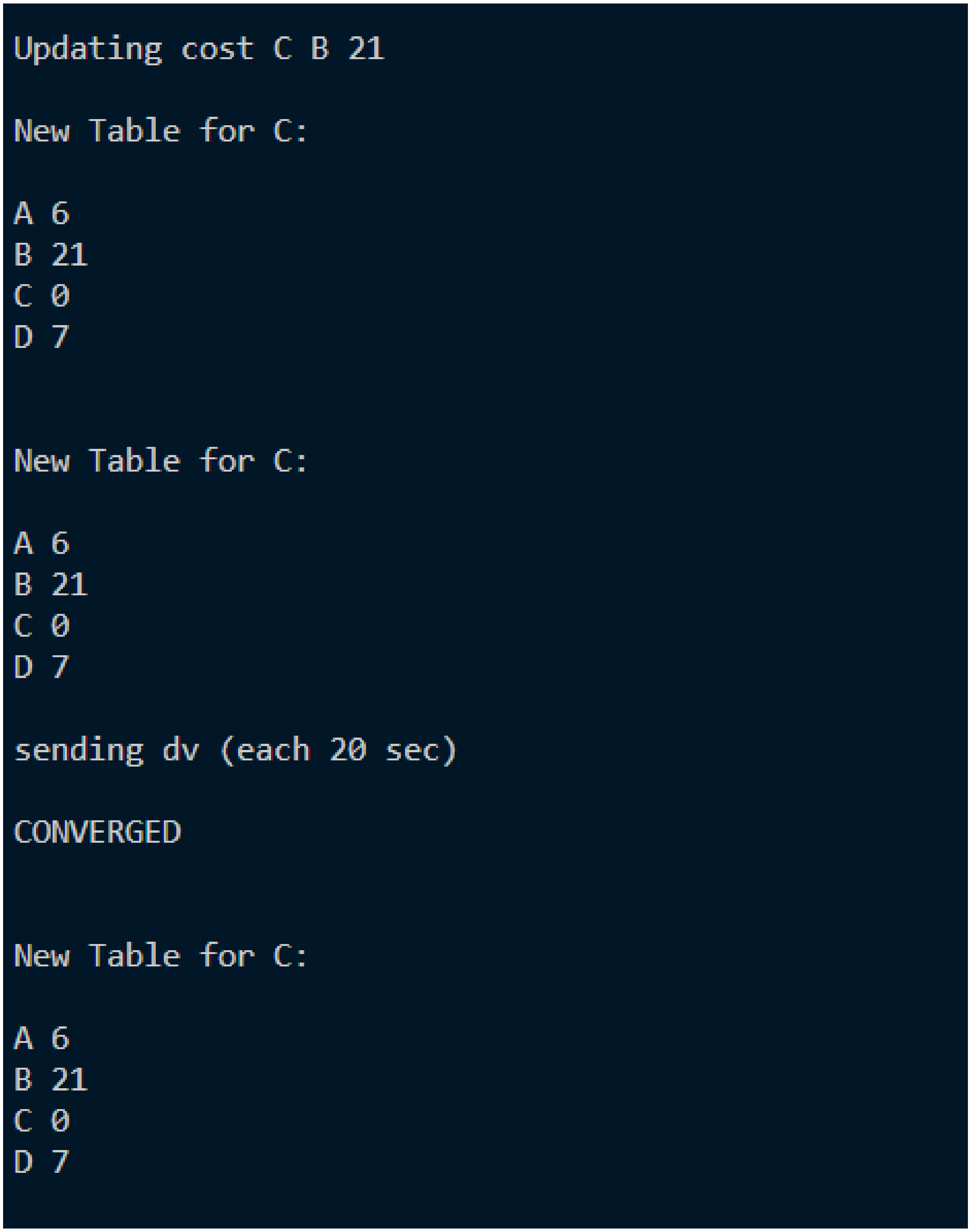
For Router B:



```
Updating cost B A 46

New Table for B:

A 46
B 0
C 16
D 23


CONVERGED


CONVERGED


New Table for B:

A 29
B 0
C 16
D 23

sending dv (each 20 sec)

CONVERGED
```

Figure 7: B router's path cost after 30 sec

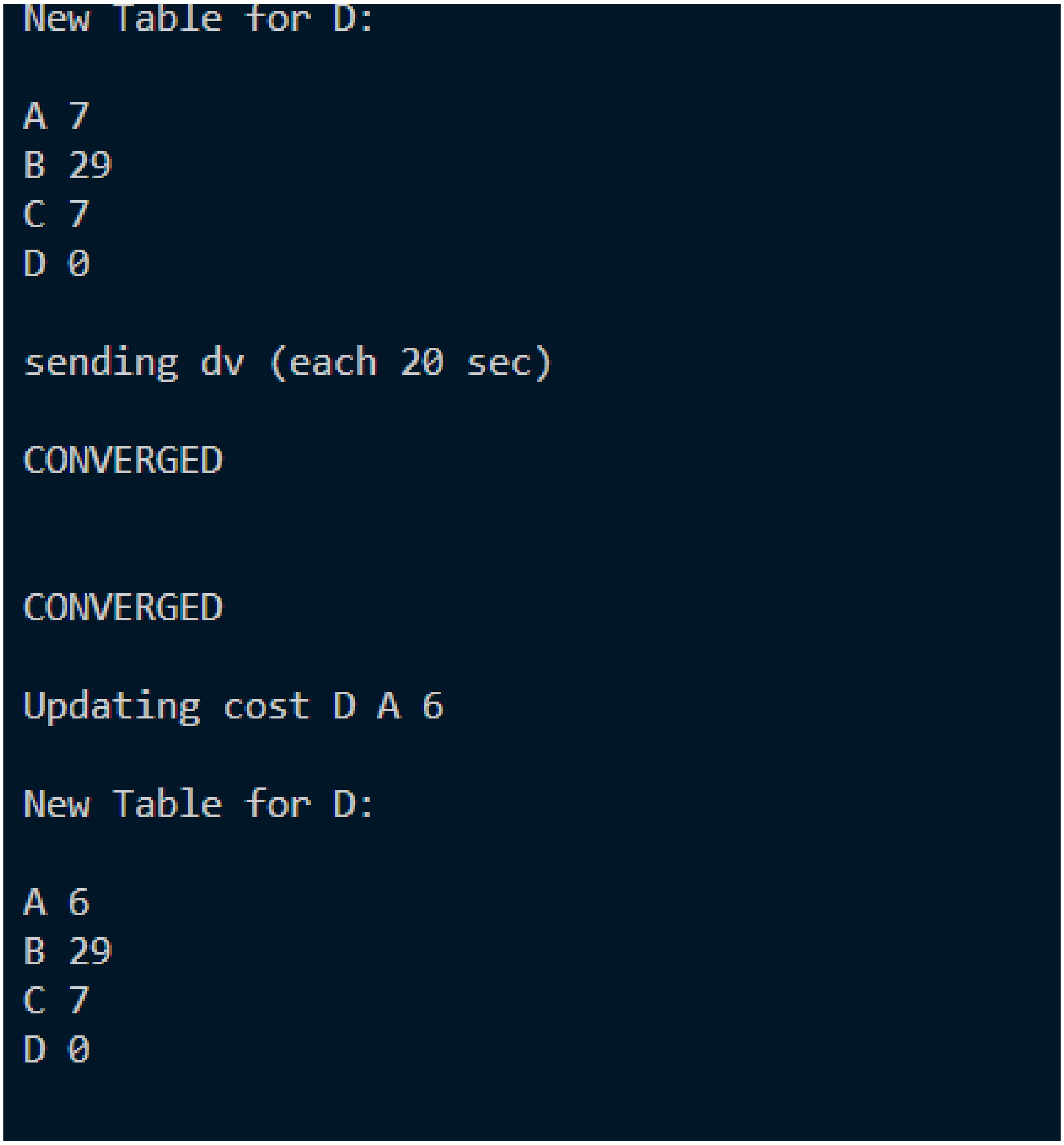
For Router C:



```
Updating cost C B 21

New Table for C:

A 6
B 21
C 0
D 7


New Table for C:

A 6
B 21
C 0
D 7

sending dv (each 20 sec)

CONVERGED


New Table for C:

A 6
B 21
C 0
D 7
```

Figure 8: C router's path cost after 30 sec

For Router D:



```
New Table for D:

A 7
B 29
C 7
D 0

sending dv (each 20 sec)

CONVERGED


CONVERGED

Updating cost D A 6

New Table for D:

A 6
B 29
C 7
D 0
```

Figure 9: D router's path cost after 30 sec

4.2 Issues or challenges

During the implementation of the Distance Vector Algorithm, some challenges and issues were encountered.

One of the challenges was designing a program that simulates a network of routers, where each router is represented as a node and each link is represented as an edge with a given cost. This required a good understanding of data structures and algorithms.

Another challenge was initializing the routing tables for each router in the network. Initially, the routing tables were not properly initialized, which led to incorrect path calculations. This was resolved by carefully checking the code and ensuring that the routing tables were properly initialized.

Additionally, testing the program in various network scenarios and verifying the correctness of the computed shortest paths was another challenge. This required generating different network topologies and making changes to the network to observe how the algorithm responds and reconverges.

Furthermore, understanding the Bellman-Ford equation and implementing it in the program was another challenge. The equation computes the new distance to a particular destination by taking the minimum value between the current distance and the sum of the neighbor's distance and link cost.

4.3 Comparison between Link state and Distance Vector algorithm

Both Link State (LS) and Distance Vector (DV) algorithms are used in dynamic routing protocols to calculate the shortest path between nodes in a network.

The main difference between the two algorithms is how they propagate routing information throughout the network. In the Link State algorithm, each node floods the entire network with information about its directly connected links, and the entire network topology is constructed based on this information. This creates a complete view of the network and enables more efficient path calculations, but at the cost of higher memory and processing requirements.

On the other hand, in the Distance Vector algorithm, each node maintains only its own view of the network and periodically sends this information to its neighbors. The neighbors then use this information to update their own view of the network, and the process continues until all nodes have a consistent view. This approach is simpler and requires less memory and processing power, but may result in slower convergence and routing loops if not configured properly.

In terms of performance, Link State algorithm is generally considered more efficient in large and complex networks with many nodes and high traffic. This is because it is more accurate in calculating the shortest path between nodes, and can quickly adapt to changes in the network topology.

On the other hand, Distance Vector algorithm is more suitable for small and simple networks with fewer nodes and lower traffic. This is because it is simpler to implement and requires less overhead, but it may take longer to converge and may not perform as well in larger and more complex networks.

Overall, the choice between the two algorithms depends on the specific requirements and characteristics of the network being deployed. In general, the Link State algorithm is preferred for large and complex networks, while the Distance Vector algorithm is preferred for small and simple networks.

References

[1] Difference between Distance vector routing and Link State routing. https://www.geeksforgeeks.org/difference-between-distance-vector-routing-and-link-state-routing/amp/. [Online; accessed 2023-04-05].

[2] Snehal Jadhav. Distance vector routing algorithm. https://www.scaler.com/topics/computer-network/distance- oct 7 2022. [Online; accessed 2023-04-05].

[3] Estefania Cassingena Navone. Dijkstra's shortest path algorithm - A detailed and visual introduction. *freeCodeCamp.org*, sep 28 2020. [Online; accessed 2023-03-23].