Simulation of Dynamic Updating in a Distributed Network and Its Applications in Green Computing

Roshan Tapas Basu, Pranav Sarkar, Rishabh Kant Singh, Prof. Sharmila Banu K
Vellore Institute of Technology, Vellore

Abstract
One of the key issues that are faced in green computing is the amount of energy used in transmitting data either over a wired or wireless network. To reduce the amount of energy consumed, a protocol for the transmission of data over a network can be designed so that the least possible distance can be covered and also all the transmissions take place as expected. This will also be beneficial with respect to the speed of data transmission as the amount of time taken for a data packet to travel over a network decreases significantly and thus other stations can get a chance to transmit more packets over the network. Thus dynamic updating can be used to solve this problem. An implementation of the same with the output being the total distance covered for all the data transmissions will be used to validate whether dynamic uploading works in favour of reducing data transmissions or not. However there is a trade-off in this approach because dynamic updating may lead to an increased pressure with respect to data storage in the network. Thus certain parameters in a practical situation have to be set according to the use of the network.

Keywords: Dynamic updating, data transmission, green computing.

1. INTRODUCTION

The network being used in this simulation will actually be a graphical representation of a distributed database. In a distributed database, the different nodes of the database work together and the data is also stored separately for each node. Thus the data is locally available and there is no need for a node to waste time and energy in updating a centralised server. But when the data required to be processed in available in another node, then the data transmission takes place over the network. To minimise the distance needed for the data packet to travel from node to another for this purpose, a technique called dynamic updating is used. This can also lead to an increase in speed of completion of data requests. In most cases of dynamic updating, a node will make a note whether a required data packet is already present or not. If it is not present then it requests the original source to send the data packet from its localised server. But this can further be optimised if it is checked whether nearby nodes also have the data packet or not. Thus the maintenance of a global table for this purpose needs to updated for each dynamically uploading that takes place.

2. METHODOLOGY

The simulation program works on any type of a network with respect to its topology. The program is written using C++ programming language.

Steps for calculation used by the simulator:

- The network can be designed by giving a user input. A sparse matrix is used to implement the network as a graph.
- The program can handle as many data requests as the user wants to be simulated. The data requests will also have to given as an input.
- For each data request, the simulator, checks from the global table which nodes already have the requested data packet.
- The distance between the destination node and the potential sources (from the global table) is calculated using Dijkstra's algorithm. When the minimal distance between the destination source node and the potential source is calculated, the sparse matrix is updated. Also that particular pair of nodes is marked in the sparse matrix so that the distance is not calculated each time but can be taken directly from the sparse matrix.
- Amongst all the potential sources, the one with the minimum distance from the destination is chosen to transmit the data packet.
- The distance covered is added to a global variable which is shown as the final output for the total distance covered for all the data transmissions across the network.
- Also at the destination node, the history of requesting the latest data packet received is checked whether it is equal to the threshold value or not. If it equal to the threshold value, then the global table is updated.

Data structures used in the simulator:

- Sparse matrix for the graph implementation of a network.
- Sparse matrix for the global table which marks which data packet is present in which node.
- The graph used is undirected. Thus for each node, it has been assumed that a two way communication is possible with its neighbouring nodes.
- User structures were used for implementing the nodes of the network.

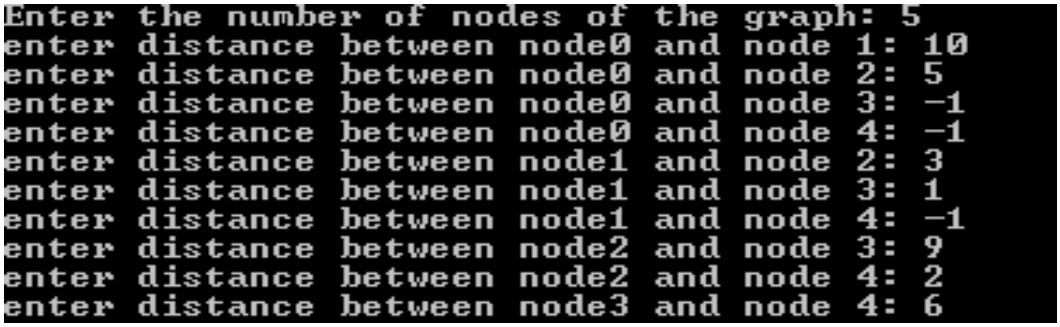
Using this method will lead to lesser wastage of energy for data transmission over a network as the distance needed to transmit a data packet is reduced drastically.

Errors encountered during implementation

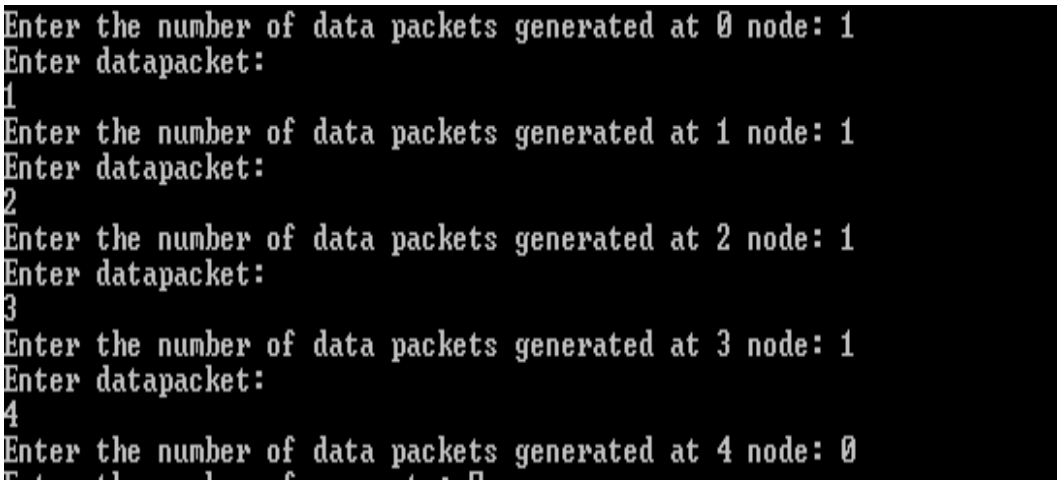
- The updating of the sparse matrix for the network. As each node has a bidirectional connection to its neighbour, the distance has updated diagonally in the matrix along the diagonal.
- Implementing the global table using linked lists. It was an inefficient manner to implement the table. The concept of in indexed 2-D array was used to rectify this as time complexity of updating the potential source nodes.
- Also while implementing the Dijkstra's algorithm, it was initially giving wrong distance calculations. These errors were corrected as and when they were encountered till the implementation became error free.

3. SIMULATION RESULTS

1.) Network creation:



2.) Initial state of network input:



3.) Transfer of data packet

```
Enter the number of requests: 7
Enter destination node number and datapacket respectively:
1
4
New request
Node 3 contains the requested datapacket
Distance from destination node = 1
Distance covered uptill now = 1
Enter destination node number and datapacket respectively:
1
4
Old request
Node 3 contains the requested datapacket
Distance from destination node = 1
Distance covered uptill now = 2
Enter destination node number and datapacket respectively:
1
4
Old request
Node 3 contains the requested datapacket
Distance from destination node = 1
Distance covered uptill now = 3
Enter destination node number and datapacket respectively:
1
4
Old request
Node 3 contains the requested datapacket
Distance from destination node = 1
Distance covered uptill now = 4
Enter destination node number and datapacket respectively:
1
4
Old request
Node 3 contains the requested datapacket
Distance from destination node = 1
Distance covered uptill now = 5
Enter destination node number and datapacket respectively:
1
4
Already exists
Distance covered uptill now = 5
Enter destination node number and datapacket respectively:
0
4
New request
Node 1 contains the requested datapacket
Distance from destination node = 8
Node 3 contains the requested datapacket
Distance from destination node = 9
Distance covered uptill now = 13
Total distance covered by data transmissions: 13
--------------------------------
Process exited after 58.71 seconds with return value 0
Press any key to continue . . .
```

In each transfer till the sixth transfer the total distance covered is incremented. In the sixth transfer, the destination node already had the required data packet. In the seventh transfer, we can see that the data packet was transferred from node 1 and not from the original generator of the data packet (node 3). Hence the distance covered is 8 units and not 9 units as it would have been without dynamic uploading.

4. CONCLUSION

The above proposed method was implemented with a fixed threshold value of 5 requests for same data packet for each node. However, the value of the threshold value in practical applications has to be fixed according to the usage pattern of the network. For lesser data transmission per transfer, it can be argued that a threshold of 1 would be the best value. But if each node of the network keeps a copy of the data packet it receives, then the servers used for data storage at each node would require more energy for maintenance which would again be undesirable for green computing. Thus a proper balance has to be maintained between saving power on data transmissions and data centre energy consumption.

REFERENCES

1.) Green Networks: Energy Efficient Design for Optical Networks by Balagangadhar G. Bathula, Jaafar M. H. Elmirghani, University of Leeds., IEEE,2009

2.) Dynamic Data Allocation Methods in Distributed Database System by Dejan Chandra Gope, American Academic & Scholarly Research Journal,2012.

3.) I. O. Hababeh , A Method for Fragment Allocation Design in the Distributed Database Systems, The Sixth Annual U.A.E. University Research Conference, 2005.

4.) T. Özsu, and P. Valduriez, 1991. Principles of Distributed Database Systems. Prentice-Hall Book Co., Englewood Cliff, USA.

5.) Chu, W.W., 1969. Optimal File Allocation in a Multiple Computer System, IEEE Transactions on Computers, C-18: 885-889.

6.) R.Azoulay-Schwartz, and S. Kraus, Negotiation on Data Allocation in Multi-Agent Environments, Autonomous Agents and Multi-Agent Systems, 5: 123-172, 2002.