International Journal of Engineering & Technology, 7 (4.10) (2018) 753-754

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Routing Algorithm for Fastest Path – a Queuing Approach

Nandhini S[1]*, Shajitha Begum S[2], Sanjay M S[3]

[1] *Department of Mathematics, School of Advanced Sciences, VIT, Vellore 632 014, India*
[2] *P.G and Research Department of Mathematics, Jamal Mohamed College, Tiruchirappalli 620 020, India*
[3] *School of Computer Science & Engineering, VIT, Vellore 632 014, India*
**Corresponding author E-mail: nandhini.s@vit.ac.in*

Abstract

Finding a fastest route for data packets is a complex process as far as large scale network is considered. Fastest path helps to minimize the delay, data loss and in turn minimizes the total cost of the network. Dijkstra's algorithm is one of the most commonly used algorithms for finding the shortest path from one source node to another destination node, which pertains to positive weights only. In this paper, Dijkstra's Algorithm combined with the features of single server queuing model is presented to find the fastest route between any two given nodes. In this improvised algorithm, every edge along with its end vertices is considered as M/M/1 queuing model. The Waiting time of a packet, number of packets in the system and the probability that the queue length exceeds 'n' packets are considered along with the weight of the particular link, for the calculation of the fastest route.

Keywords: *Dijkstra's Algorithm; M/M/1 Queuing model; Optimized shortest path; shortest path algorithms.*

1. Introduction

Shortest path algorithms play a vital role in robust networks with large number of nodes and edges. In recent years many algorithms are available to find the shortest path or to find the optimum path. Dijkstra's algorithm is an efficient algorithm to find minimum distances even in situations where the volume of data is huge and the velocity rate of data arrival large. The data is reduced used techniques like similarity computation, maximum solidarity clique and then the shortest path is found [1]. Dijkstra's algorithm is also used widely for routing in computer networks. This will reduce the cost of network establishment especially in wired networks. Along with Dijkstra's we have various other shortest path algorithms like Floyd-Warshall Algorithm, Bellman-Ford Algorithm and Genetic algorithm [3]. The shortest path problem also finds application in Global Positioning System (GPS) in which, after the position of a node has been retrieved, shortest path algorithms like Dijkstra's algorithm is applied to find the quickest and shortest route amongst its neighbours [3].

A queue is used to store traffic until it can be processed or serialized. In networking scenario, by default every node is modelled as M/M/1 queuing model, in which arrival of packets follow Poisson distribution and service pattern follows Exponential distribution and single server model. But M/M/1 model may be inefficient for real-time applications, especially during the time of congestion, since it will never distinguish or give preference to higher-priority packets. To avoid this type of congestion and packet drops, advanced queuing models are implemented.

In this paper, Dijkstra's algorithm along with M/M/1 queue is considered to find the fastest path, which will improve the efficiency. The organization of the paper is as follows (i) Dijkstra's algorithm structure (ii) Improved Dijkstra's algorithm (iv) Experimental study & Result (v) Concluding remarks.

2. Dijkstra's Algorithm Structure

Dijkstra's algorithm is a based on labelling the vertices of the given graph. At each iteration in the algorithm some vertices have permanent labels and others temporary labels, the algorithm begin by assigning a permanent label 0 to the starting vertex 's' and a temporary label ∞ to the remaining n-1 vertices [4]. From then on, in each iteration another vertex gets a permanent label, according to the following rules:

Every vertex 'j' that is not yet permanently labelled gets a new temporary label whose value is given by

$$\min\left[\text{old lable of j}, \left(\text{old lable of i} + d_{ij}\right)\right]$$

where 'i' is the latest vertex permanently labelled, in the previous iteration and d_{ij} is the direct distance between vertices 'i' and 'j'. If 'i' and 'j' are not joined by an edge, then $d_{ij} = \infty$.

The smallest value among all the temporary labels is found, and this becomes the permanent label of the corresponding vertex (say i'). In case of a tie, select any one of the vertex for permanent labeling

$$Plabel_{(j)} = \min\{\, temporary\ labels\ of\ j\} \qquad (1)$$

Steps 1 and 2 are repeated alternatively until the destination vertex 't' gets a permanent label.

3. Enhancement in Dijkstra's Algorithm

In this enhanced model, each link is considered as a single server queuing model. In a single-server queue, customers arrive

according to a Poisson stream of rate $\lambda > 0$. Customer who finds the server free is served immediately. The service times are exponentially distributed with a rate $\mu > 0$ [5]. This model is represented as $M/M/1$, where *'M'* represents the Poisson and Exponential distribution and $'1'$ represents the single server.

When a data packet enters a node $'i'$ (except the starting node), all the adjacent nodes are considered. Each adjacent node along with the node $'i'$ is assumed as a single server queuing model. In our improvised version of Dijikstra's algorithm, the fastest route is calculated as follows.

Let *'j'* be an adjacent node to *'i'* which is not permanently labelled.

The Average waiting time of a packet in the jth node is calculated by

$$WT_{avgj} = \frac{1}{\mu-\lambda} \quad (2)$$

The Average number of packets waiting in the jth node is given by

$$NC_{avgj} = \frac{\lambda}{\mu-\lambda} \quad (3)$$

Probability that the number of packets exceeds 'S' at jth node is

$$PR_{sj} = P(n > s) = \left(\frac{\lambda}{\mu}\right)^{s+1} \quad (4)$$

It has to be noted that for M/M/1 queuing model, in the steady state, the traffic intensity,

$$\frac{\lambda}{\mu} < 1. \quad (5)$$

$$dist(i,j) = \min\{\text{old label of } 'j', (\text{old lable of } 'i' + (WT_{avgj} * NC_{avgj} * PR_{sj}) + dij)\}$$

where *dij* represents the weight of the link (i,j), $dist(I,j)$ is then considered as the label for the node *'j'* for finding the further route.

Hence the minimum of $dist\ (i,j)^{th}$ link is considered as

$$mindist(i,j) = \min\{dist(i,j)\} \quad (6)$$

The j^{th} node pertaining to minimum length is chosen as the next node for the fastest route.

4. Experimental study & Results

The network topology we have considered is depicted in Fig.1 which has 8 nodes, in which we have to find the fastest route between $node\ 0$ and $node\ 7$. The weight of an edge in Figure.1 can be considered as the cost of traversal from one node to another. The result analysis is done by taking the value of $s = 5$. The service rate and arrival rate of all the nodes are presented in Table 1.

Table 1: λ and μ values

Node	$\lambda(sec)$	$\mu(sec)$
1	0.002	0.007
2	0.002	0.008
3	0.001	0.003
4	0.003	0.009
5	0.002	0.006
6	0.001	0.004
7	0.003	0.008

The improvised Dijkstra's algorithm gives the fastest route as $0-1-6-4-7.$

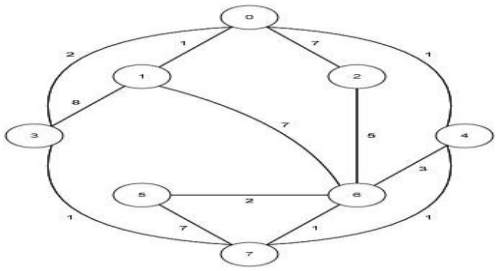

Fig. 1: Network Topology

5. Concluding Remarks

This paper represents an improvised version of Dijkstra's algorithm for finding the fastest path from a source node to a destination node. This model considers the queuing metrics along with the weight of the link to calculate the total length and the minimum of all the total length is chosen for that link. Since average waiting time of a packet, number of packets waiting in a node and Probability that the packet number exceeds are all considered for calculating the path, the path given will be the fastest route from given source to destination.

References

[1] Artalejo J R, Gomez-Corral, Performance Analysis of a Single-Server Queue with Repeated Attempts, Mathematical and Computer Modelling, 30 (1998), 79-88.

[2] Haysam Selim , Justin Zhan, Towards Shortest Path Identification on Large Networks, Journal of Big Data, (2016), 3-10.

[3] Kairanbay Magzhan , Hajar Mat Jani, A Review and Evaluations of Shortest Path Algorithms, International Journal of Scientific & Technology Research, 2, no.6 (2013), 99-104.

[4] Narsing Deo, Graph Theory with Applications to Engineering and Computer Science (2014), Prentice-Hall, Inc.

[5] Palaniammal, S, Probability and Queueing Theory (2012), PHI Learning Private Ltd, New Delhi.

[6] Wang Shu-Xi , The Improved Dijkstra's Shortest Path Algorithm and Its Application, Procedia Engineering , 29, (2012), 1186 – 1190.

[7] Yong Deng, Yuxin Chen, Yajuan Zhang ,Sankaran Mahadevan, Fuzzy Dijkstra Algorithm for Shortest Path Problem under Uncertain Environment Applied Soft Computing, 12, no.3 (2012) 1231-1237.