Volume 3, Issue 8, August 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Processing Delay Consideration in Dijkstra's Algorithm

Charika Jain* **, Jitendra Kumawat**
Computer Science, Amity University Rajasthan
Rajasthan, India

Abstract: This paper studies how to select a path with the minimum cost in terms of expected end-to-end delay in a network. Different from the previous efforts, the new end-to-end delay takes the queuing delay and transmission delay into account with propagation delay, since the end-to-end delay consist of not only the propagation delay but also the transmission delay and queuing delay in buffer. As now the network become complex so queuing delay has reached to the magnitude of propagation delay or even more.

Keywords- EED(end-to-end delay), Router, Network, Cost, Shortest Path

I. Introduction

In today's network shortest path routing has been widely used. In shortest path routing, each node attempts to route packet to destinations over paths of minimum cost. There are two main classes of algorithm: distance-vector algorithm and link-state algorithm. In a distance-vector algorithm, each node sustains a routing table having the distance of the shortest path to every destination in a network. A node only informs its immediate neighbours of any distance changes to any particular destinations. Example of distance-vector algorithms include the old ARPANET routing algorithm[1], NETCHANGE algorithm of the MERIT network [2], and cisco's IGRP [3]. In a link-state algorithm, each node keeps track of the entire network topology and computes the routing table based on the link distance information broadcast by every node in the network. Link-state algorithms have been used in the new ARPANET routing protocol (SPF) [4], OSPF [5] and ISO's IS-IS[6]. Shortest-path routing algorithms have served remarkably well in the network environment where traffic is light and network conditions change slowly. Shortest-path algorithms are able to respond to topological changes automatically and adjust routing decisions when traffic changes. In the presence of congestion, shortest-path routing algorithms can reduce the traffic away from the overloaded the paths.

However, as the networking speed boosts and new applications proliferate, the network environment becomes much more dynamic and the traffic patterns less predictable. The range of traffic rates which the network has to deal with is much wider and the distribution of traffic can also be extremely uneven. In a network environment where traffic approaches the capacity of paths and changes dynamically, shortest path routing algorithm, particularly those that attempt to adapt to traffic conditions, frequently exhibit oscillatory behaviours and cause performance degradation [7,8]. In this paper first Dijkstra's Algorithm is examined then optimized dijkstra's algorithm is applied considering queuing delay and transmission delay with propagation delay. Then finally a comparison between the two algorithms is given.

II. Overview

Routing algorithms can be categorized as static, quasi-static and dynamic according to how adaptive they are. In a static routing, the choice of route is predetermined and fixed for relatively long time period (months or even years). Static routing algorithms are simple to implement but vulnerable to resource failures and traffic changes. A dynamic routing algorithm, in contrast, allows constant changes in routing decisions to reflect the current traffic and topological changes. But such routing algorithms are often very complex and call for large amounts of information exchanges and computation. The shortest-path routing algorithms extensively used today fall approximately into the quasi-static category, in which the link distances remain constant for short period of time (eg. 10 seconds in SPF) but can be updated when significant changes occur. A quasi-static routing algorithm has two key procedures: distance estimation and route computation. The distance estimation predicts the link distances for the next route updating period based on the information collected in the past. The estimated link distances are then propagated over the network and used to derive the routing table for packet forwarding. In shortest path routing algorithms, the link distances are decided based on the particular routing metric used (eg delay, queue length, throughput, hops). With traffic-sensitive routing metrics (eg. delay), each node estimates the link distances for the next route updating period by averaging the link distances in this period.

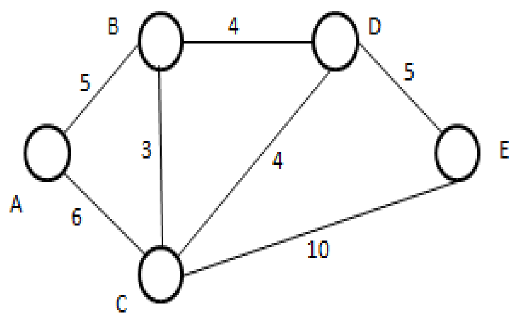
III. Earlier Work

Dijkstra's Algorithm was created in 1959 by Dutch computer scientist Edsger Dijkstra. While employed at the Mathematical Centre in Amsterdam, Dijkstra was asked to demonstrate the powers of ARMAC, a sophisticated computer system developed by Mathematical Centre. Part of his presentation involved illustrating the best way to travel between two points and in doing so, the shortest path algorithm was created. It was later renamed Dijkstra's Algorithm in recognition of its creator.

Dijkstra's Algorithm is a graph search algorithm that solves the single –source shortest path problem for a graph with non negative edge path costs, producing a shortest path tree. This algorithm is often used in routing and other network related protocols. For a given source vertex(node) in the graph, the algorithm finds the path with lowest cost(i.e. the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined.

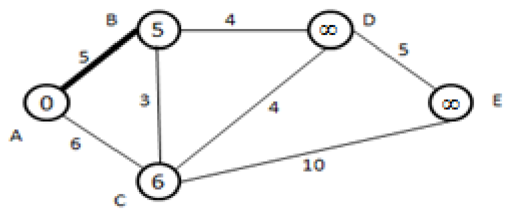
Algorithm:

```
function Dijkstra(Graph, source):
    for each vertex v in Graph:
        dist[v] := infinity ;

        previous[v] := undefined ;
    end for

    dist[source] := 0 ;
    Q := the set of all nodes in Graph ;

    while Q is not empty:
        u := vertex in Q with smallest distance in dist[] ;
        remove u from Q ;
        if dist[u] = infinity:
            break ;
        end if

        for each neighbor v of u:

            alt := dist[u] + dist_between(u, v) ;
            if alt < dist[v]:
                dist[v] := alt ;
                previous[v] := u ;
                decrease-key v in Q;
            end if
        end for
```

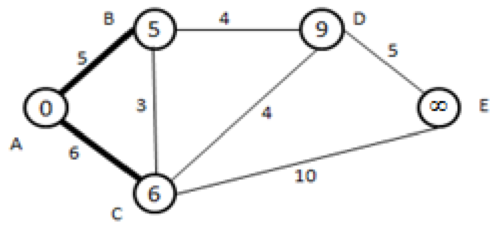


We want to find shortest path between A to E using Dijkstra's algorithm. Node A is destignated as current node. We set cost(A)=0 which indicates that we have found a path from s to s of total weight 0(the path with no edges). For any other vertex v we set distance(v)=∞ since we havn't even verified that an A-v path exists.

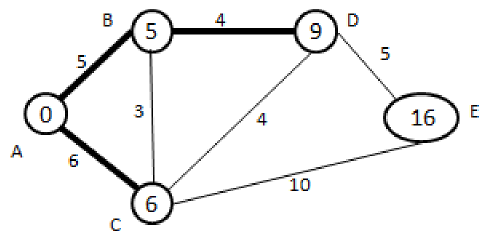
The smallest cost in cost(A)=0, we remove A from Q, indicating that cost(A)=0 represents the smallest possible total weight of a path from A to A:



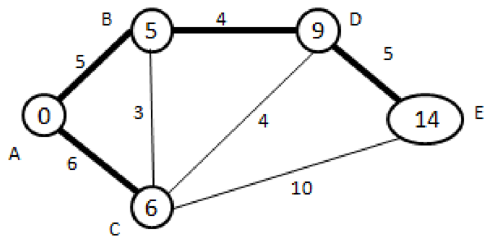
We examine the edge that leave A. The edge AB gives us a path cost 5 from A to B so we change cost(B)=5. Likewise, we change cost(C) from ∞ to the smaller value 6. The smallest value of distance is 5, which happens at B so we follow A to B path. Thus darkening AB edge.



The edge BD has weight 4, tells us we can get freom A to D for a cost of Cost(B) + distance_between(BD) = 5 + 4 = 9 which is less then ∞ so change the cost(C) = 9. The edge BC has weight 3, tells that cost(C) through B is Cost(B) + distance_between(BC) = 5 + 3 = 8 which is more than 6, so no change in cost(C). The smallest distance is 6, so egde AC is darken.



Now we examine the distance of neighbour of C. calculate CD and CE. The minimum occur at BD. Thus following that path.



Calculate DE. The minimum is DE so following that path.

IV. Proposed Work

Let us now consider what happen to a packet as it travels from a host (the source), passes through a series of routers, and ends its journey in another host (the destination). As a packet travels from one node to another node along this path, the packet suffers from several different types of delays at each node along the path. The most important of these delays are the nodal processing delay, transmission delay, queuing delay and propagation delay; together these delay accumulate to give a total nodal delay. This paper proposed an algorithm in which the processing delay, which was neglected by dijkstra's algorithm, is also added for calculating the cost at each router. Thus, this algorithm chooses the path with less number of hops.

Algorithm:

Initialization and everything is same in this algorithm except the formula to calculate the distance to each node.

```
for each neighbor v of u:
alt := dist[u] + dnodal;
// where dnodal is the processing delay at node
            if alt < dist[v]:
                dist[v] := alt ;
                previous[v] := u ;
                decrease-key v in Q;
            end if
        end for
```

where,

dnodal = dproc + dqueue + dtrans + dprop

To simplify the analysis of network delay times, the packet delay is broken up into sequence of nodal delays. Each nodal delay is the time between the arrival of a packet at a node and its arrival at the next node. The above equation decomposes the nodal delay into components that are simpler to analyze.

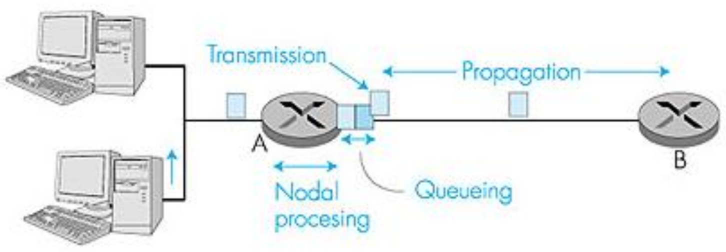


Fig 1: Nodal Processing Delay

Delay components:

1. The nodal processing delay dproc is the time that a node spends processing a packet. This includes times for error checking, time includes time for error checking, time for reading the packet header, and time for looking up the link to the next node, based on the destination address. Although the processing may sound complicated, the nodal processing delay is usually negligible compared to other terms in the delay equation.

2. Queuing delay: The queuing delay dqueue is the time that a packet spends in a queue at a node while waiting for other packets to be transmitted. If the node is a high-speed router then there is one queue for each outgoing link, so a packet

waits only for other packets that are going across the same link. The queuing delay is related to transmission delay dtrans by the following approximate equation:

dqueue = dtrans * lqueue

where lqueue is length of queue

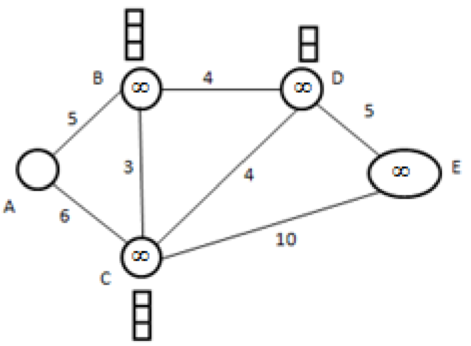
3. Transmission delay: Getting the entire packet "out of door". Let packet contain L bits and link transmission rate R b/s. Transmission delay is then L/R

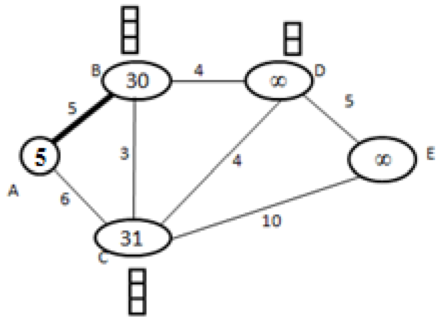
4. Propagation delay: Time for one bit to traverse the medium between two switches.

Example:

Now the same example is taken so that a caparison can be made. But here queue length at each router is also specified. So as to calculate dqueue delay of each router. Firstly the cost of A is L/R i.e. the transmission time taken by a packet.

Here L = 7.5 Mbits
R = 1.5 Mbps
dtrans = L/R = 5 sec





We examine the edges that leave A.

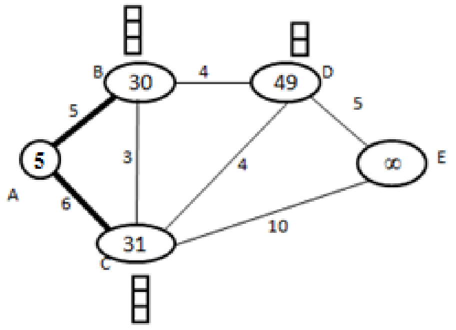
The edge AB has cost 5 from A to B, the queue length is 3 so we change

cost(B) = cost(A) + dnodal = cost(A) + dprop + dqueue + dtrans = 5 + 5 + 3 * 5 + 5 = 30.

Compare it with previous cost which was ∞. The new cost i.e. 30 is less than ∞. Thus cost(B) will change to 30.

Likewise, we change cost(C) from ∞ to the smaller value 31.

The smallest cost is 30, which is of B so follow A to B path. Thus darkening AB edge.



Examine neighbour of B:

The edge BD has weight 4, so the cost of D from B is

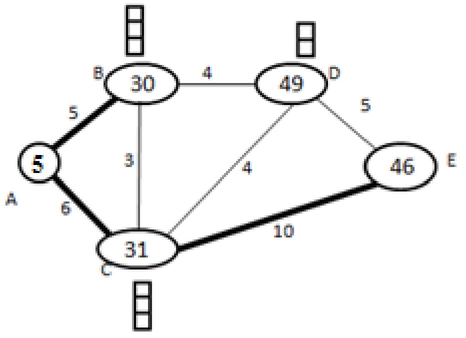
cost(B) + dnodal = 30 + 4 + 2 * 5 + 5 = 49.

Compare it with cost(D) earlier which was ∞, 49 is less than ∞ so change cost(D) to 49.

Likewise there is a edge from B to C so the cost of C from B is cost(B) + dnodal = 30 +3 + 3 * 5 + 5 = 53.

Compare it with cost(C) earlier which was 31, it is more than that so no change in it.

The smallest cost is of 31, which is of C. Thus darkening AC edge.



Now examine neighbours of C:

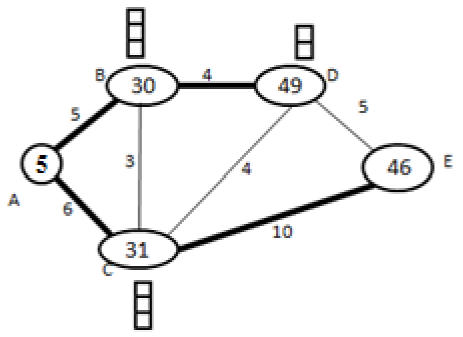
The edge CD has weight 4, so the cost(D) from C is

cost(C) + dnodal = 31 + 4 + 2 * 5 + 5 = 50.

Compare it with earlier cost(D) which is less than it. Thus no change in cost.

The edge CE has weight 10, so the cost(E) from C is cost(C) + dnodal = 31 + 10 + 5 = 46.

The smallest cost is 46 which is of E. Thus darkening the edge CE.



Now examine neighbour of D:
The edge DE has weight 5, so the cost(E) from D is cost(D) + dnodal = 49 + 5 + 5 = 59.
Compare it with earlier weight which was 46. Thus no change in cost(E).
The smallest cost is 49 which is of D from B. Thus darkening the edge BD.

The darken edges gives the route from source to destination i.e. from A to E with the cost of 46. The path is A -> C -> E.

If the route of Dijkstra’s algorithm has followed i.e. A -> B -> D -> E then the cost would be
cost(E) = cost(D) + dnodal = 59.
Thus we have optimized the Dijkstra’s Algorithm.

Comparison between Dijkstra’s Algorithm and optimized algorithm

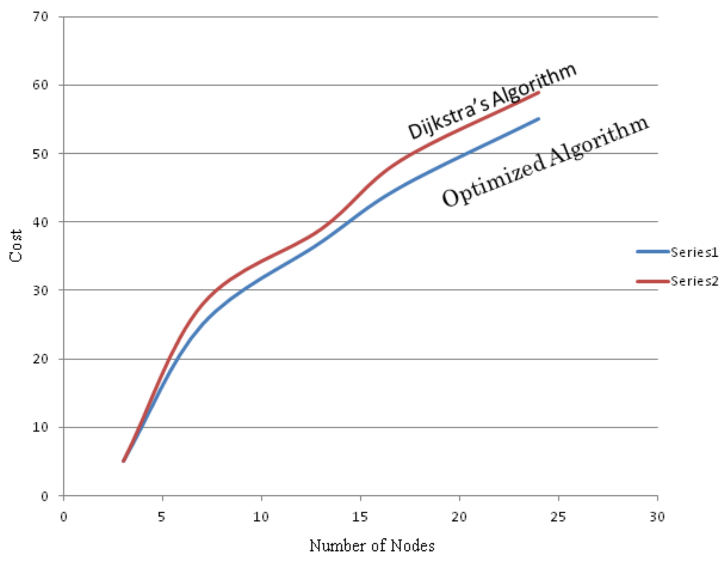


Fig 2: Comparison between Dijkstra’s Algorithm and Optimized Algorithm

VI. Conclusion

This paper has include the nodal delay(queuing delay and transmission delay) which was neglected by Dijkstra’s algorithm as it has been reached to a magnitude more than that of the propagation delay. Thus, one has to consider these delays for computing the optimized solution. By reducing the number of hops between the path followed from source to destination the total amount of network resourses are also reduced. Thus this optimizes dijkstra’s Algorithm.

References

[1] J. M. McQuillan and D. C. Walden, "The ARPA Network Design Decision", Computer Networks, Vol.1, pp. 243-389, 1977.

[2] W. D. Tajibnapis, "A Correctness Proof of a Topology Information Maintenance Protocol for a Distributed Computer Network", Communication of the ACM, Vol.20, pp. 477-485, 1977.

[3] C. L. Hedrick, "An Introduction to IGRP", Preprint, RUTGERS, Centre for Computers and Information Services, The State University of New Jersey, Oct. 1989.

[4] J. M. McQuillan, et al., "The New Routing Algorithm for the ARPANET", IEEE Transactions on Communications, Vol. COM-28, May 1980.

[5] J. Moy, "The OSPF Specification", RFC 1131, SRI International, Menlo Park, Calif., Oct. 1989.

[6] International Standards Organization, "Intra-Domain IS-IS Routing Protocol", ISO/IEC JTC1/SC6 WG2 N323, Sept. 1989.

[7] D. Bertsekas, "Dynamic Behavior of Shortest Path Routing Algorithms for Communication Networks", IEEE Transactions on Automatic Control, Vol. AC-27, No.1, Feb. 1982.

[8] A. Khanna, J. Zinky, "The Revised ARPANET Routing Metric", in Proc. of ACM SIGComm’89, pp 45-56, Sept. 1989.