ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Routing Algorithm For Computer Networks

[I]Ms. Anupama Tewari, [II]Dr. Anuj kumar,
[I]Research Scholar, Mewar University
[II]Professor, Accurate Institute of Science and Technology

Abstract
This paper gives the brief idea about the computer network routing algorithm. This is being done by analysing the limitations based on using algorithms in networks. It gives as idea of use of shortest path algorithm (Dijkstra's algorithm) indivially in networking and also the use of Bellman-Ford algorithm. Both work separately on routers, on RIP protocol. Combination of both together will give multiple ways of directing the edges together to have appropriate output. The framework of finding optimal solution for computer routing is solved here.

Key terms
Bellman Ford Algorithm, Shortest path Algorithm, algorithm in networking, OSPF in brief, Network is a bunch of users communicating among themselves, Algorithm is a sequence of logical instructions

I. Introduction

The routing algorithm is described by [2] as network layer protocol that guides packets (information stored as small strings of bits) through the communication subset to their correct destinations. Some reasons for the complexity of routing algorithms are: coordination between the nodes in the network; failures of the links and nodes; congestion of traffic links. Two types of algorithms are used for routing in networks: shortest path routing algorithms and Bellman - ford algorithm based on other measures. The efficiency of a routing algorithm depends on its performance, during congestions in the network. The routing algorithms must perform route choice and delivery of messages. The performance of the routing is assessed according to the throughput in the network (quantity of data transfer) and the average packet delay (quality of service).

II. Common Problems In Networking

1. The total path for delivering the packet is not defined in advance; rather each node decides which line to use in forwarding the packet to the next node.
2. Also, an instantaneous measurement of queue length does not accurately predict the average delay because there is a significant real time fluctuation in queue lengths at any traffic level. Certain variation may occur due to the high average delay of packet on CPU.

All three [7] defects are reflection of a single point, namely that the length of an output queue is only one of many factors that affect a packets delay.
The above mentioned routing algorithms is use for the betterment of using fewer network resources, operates on more realistic estimates of networks conditions, reacts faster to important network changes and does not suffer for long term loops or oscillations.

III. About Shortest Path Algorithm

In Internet environment, the routers compute the flow transmissions according to the shortest path algorithm. This algorithm is efficient in finding optimal route, according to the link weights presenting the traffic load on them. The limitation of this algorithm is that it cannot route the flow along alternative paths. In common network structure it exists always several paths between the source and destination nodes. Now the OSPF protocol routes according to the shortest path criteria, but it does not estimate and apply alternative routing to available paths. Thus Quality of Services (QoS) is not supported only by shortest path management. The optimal routing under (QoS) requirements is a complex problem for implementation [4,6]. Such architecture insists routers to broadcast the local resource status and the local topology information to all routers. One manner of providing QoS in routers is to apply traffic prioritization. The idea is to classify the traffic to a multiple levels of priority queues. The priorities are assigned on packet peculiarities: the protocol uses packet type, source and destination networks. Enhancements are done by subdividing the link capacity into different classes. The traffic is assigned to each class and the routers serve each class with different priority. However the traffic prioritization improves the QoS by class of traffic on a given link, but that link is chosen by the shortest path routing mechanism, which is independent of the QoS requirements. The optimal routing algorithm must keep the delays low as the flow control increases. Thus the routing increases the throughput and restricts the delay for the packet, during high traffic conditions. Thus the average delay per packet is reduced also at steady or low traffic conditions.
Open Shortest Path First (OSPF) [3] is a well known real-world implementation of DA used in network routing. In real networks, particularly in Ethernet networks, the Spanning-Tree Protocol (STP) [7] runs on the network before the OSPF. In a general way, a spanning tree of a graph is a sub-graph which is also a tree that contains all the nodes. In other words, in a network environment, where redundant links are common, the STP causes these links to appear closed for the operation of the network elements, as to eliminate the appearance of duplicate messages, such as e.g. Neighbour discovery messages .Rings are a particular interesting class of topologies in optical networks, because they allow an additional level of connectivity for each node (there are now two possible paths to the destination node instead of one),with the cost of a single additional link. Rings are common elements in existing or planned networks, such as the European Optical Network (EON) or the NSF net. Short version of EON (also termed COST 239) and the NSF net network – in both figures, among others, several four node ring sub networks, can be detected, e.g. .Amsterdam, Berlin, Prague, Luxembourg for EON and Pittsburgh, Princeton, Boston and Ithaca for NFS net.

Description of Dijkstra short path algorithm

The algorithm performs several rules [4]:
Rule1: A graph of the network is built network and the adjacency matrix a [i, j] with the weight of links is defined. For the case when a direct link between node Vi and Vj is missing, [5]the weight

of the link is assumed as infinity. The source and the destination nodes are noted as NS and NT.
Rule2: A status record set is established for every node with three fields:
The first field that shows the previous node, named "predecessor" field. The second filed is named "Length" field and it shows the sum of weights from source to that node. The last field named "Label" filed, shows the status of the node. Each node can have one status mode: "Permanent" or "Tentative".
Rule3: Initialization of the status record set for all nodes and setting all "Length" to Infinity, and all "Label" as tentative.
Rule 4: Labelling node NS as t node and marking its "Label" as "Permanent". When a label changes to permanent, it never changes again. T node rules as a current chosen node.
Rule5: For all tentative nodes, directly linked to t node, status record set is updated.
Rule6: From all the tentative nodes, choose the one whose weight to NS is less and set it as t node.
Rule7: If this node is not the destination NT, then, go to step 5.
Rule8: If this node is NT, then extract its previous node from status record set and do this until return to NS. The nodes show the best route from NS to NV
Notation:
Di = Length of shortest path from node 'i' to node 1.
di,j = Length of path between nodes i and j .

Algorithm

Each node j is labelled with Dj, which is an estimate of cost of path from node j to node 1. Initially, let the estimates be infinity, indicating that nothing is known about the paths. We now iterate on the length of paths, each time revising our estimate to lower values, as we obtain them. Actually, we divide the nodes into two groups ; the first one, called set P contains the nodes whose shortest distances have been found, and the other Q containing all the remaining nodes. Initially P contains only the node 1. At each step, we select the node that has minimum cost path to node 1. This node is transferred to set P. At the first step, this corresponds to shifting the node closest to 1 in P. Its minimum cost to node 1 is now known. At the next step, select the next closest node from set Q and update the labels corresponding to each node using :

$$D_j = \min [D_j , D_i + d_{j,i}]$$

Finally, after N-1 iterations, the shortest paths for all nodes are known, and the algorithm terminates.

IV. About Bellman – Ford Algorithm

In comparison to Dijkstra's algorithm, the Bellman-Ford algorithm admits or acknowledges the [6] edges with negative weights. That is why, a graph can contain cycles of negative weights, which will generate numerous number of paths from the starting point to the final destination, where each cycle will minimize the length of the shortest path. Taking into consideration this fact let's assume that our graph does not contain cycles with negative weights .The array d[]will store the minimal length from the starting points to other vertices. The algorithm consists of several phases, where in each phase it needs to minimize the value of all edges by replacing d[b] to following statement d[a] + c; a and bare vertices of the graph, and c is the corresponding edge that connects them.
This algorithm iterates on the number of edges in a path to obtain the shortest path. Since the number of hops possible is limited (cycles are implicitly not allowed), the algorithm terminates giving the shortest path.
Notation:
$d_{i,j}$ = Length of path between nodes i and j, indicating the cost of the link.
h = Number of hops.
D[i,h] = Shortest path length from node i to node 1, with upto 'h' hops.
D[1,h] = 0 for all h .

Algorithm :

Initial condition : D[i, 0] = infinity, for all i (i != 1)
Iteration : $D[i, h+1] = \min \{ d_{i,j} + D[j,h] \}$ over all values of j
Termination : The algorithm terminates when D[i, h] = D [i, h+1] for all i .

V. Combining Both Shortest Path Algorithm And Bellman-Ford Algorithm

Both, the Bellman-Ford algorithm and Dijkstra's algorithm are used to calculate 'metrics' (distance/cost of traversing a link) in routing protocols. Both of them consider only hop count (the number of machines between the source of the message and the destination) as the metric between two nodes. Other factors such as bandwidth and delay can also be used to calculate the metric, but they are used by other complex protocols. The difference between these two algorithms lies in the type of protocols which use the respective algorithms.
The Bellman-Ford algorithm is used by DVR protocols like RIP and RIPv2. In a distance vector routing protocol each router on the network, on which the protocol is running, prepares routing update packets. The information in each routing update packet includes the list of all the nodes in the network and the corresponding metric costs. This packet is forwarded to all the neighbours of the router. Similarly the router receives an update from each neighbour and performs the required updates in its routing table. The distances are calculated using the Bellman-Ford algorithm.
On the other hand,[1] Dijkstra's algorithm is used by LSR protocols like OSPF. LSR protocols are different from the DVR protocols as routers implementing these protocols store the entire topology of the network in their memory. There are 2 stages in building a routing table in LSR protocols. First, a map of the entire network should be stored in every router, and then the shortest distance to each node must be calculated by each router. Even here, the routing updates are prepared periodically and when the topology changes. But here, the update, called a link state packet, is flooded (LSP) throughout the network, unlike in the above case, where it is sent only to the neighbours. Each LSP contains an ID for the source, a Sequence Number (to distinguish newer packets from older packets) and the distances from the sender to each of its neighbours. When each router has collected LSPs from each router, it starts creating the routing table including the shortest paths to each node, using Dijkstra's algorithm.

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

VI. Pseudo Code For Dijkstra's Algorithm

```
// INITIALIZATION

S = {all nodes except source node z};
For all nodes v {
    if (v adjacent to z) {
        D[v] = C(z, v);
        R[v] = { z };
    }
    else {
        D[v] = infinity;
        R[v] = 0;
}

//LOOP

While (set S is not empty) {
    Select a node u from S with D[u] as the minimum;
    If D[u] is infinity {
        Error: no path exists to nodes in S;
        Quit;
    }
    Delete u from set S;
    For each node v such that (u, v) is an edge {
        if (v is still in set S) {
            c = D[u] + c(u,v);
            if (c < D[v]) {
                R[v] = {merge elements (R[u], u)};
                D[v] = c;
            }
        }
    }
}
```

VII. Pseudo Code For Bellman-Ford Algorithm

BELLMAN-FORD (G,w,s)
INITIALIZE-SINGLE-SOURCE (G,s)
for i = 1 to |G.V| - 1
for each edge(u,v) ∈ G.E
RELAX (u,v,w)
for each edge(u,v) ∈ G.E
if v.d > u.d + w(u,v)
return FALSE
return TRUE

VIII. Combination Of Dijkstra Algorithm And Bellman-Ford Algorithm

Initialization
d(v) ←∞, for all v ∈ V
π(v)← nil , for all v ∈ V
d(s) ← 0

Relax (u, v)
If d(u) + c(u, v)< d (v)
d(v) ← d(u) +c(u, v)
π(v) ← u

P lain scan
for each edge (u, v) ∈ E
Relax(u, v)

S←∅
while(there is a vertex in V\S with d <∞) do
find vertex u in V\S with the minimal value ofd
S ← S ∪{u}
for each edge (u, v) ∈ E /* scanning u*/
Relax(u, v)

Dijkstra(G, s)
Initialization
Dijkstra, scan
return(d, π)

Bellman-Ford(G, s)
Initialization
I ← 0
do
i++
P lain scan
until((there was no change of d at P lain scan) or(i= |V|))
if (i < |V|) return(d, π)
else return(There exists a negative cycle reachable froms.")

Algorithm Bellman-Ford-Dijkstra (BFD) is as follows:

Bellman-Ford-Dijkstra(G, s)
Initialization
I ← 0
do
i++
Dijkstra scan
until ((there was no change of d at Dijkstra scan) or(i =|V|−1))
if(i <|V|−1)
return(d, π)
else return("There exists a negative cycle reachable from s.)"

The Bellman-Ford algorithm and Dijkstra's algorithm [9] proved to be much more efficient than brute-force, with Dijkstra proving to run in the least amount of time for very large networks. It appears that for complete, fully meshed networks, the Bellman-Ford algorithm actually ran faster than Dijkstra's algorithm for networks of size 425 or less. However, Dijkstra showed to be considerably more efficient beyond networks with more than 425 vertices.
In relation to computer networking, this will likely come as a shock to many network engineers. As a general rule, network engineers are told to stay away from the routing protocols that use the Bellman-Ford algorithm. Again, these protocols are RIP and EIGRP, while OSPF runs Dijkstra's algorithm. However, our studies show that for small and medium sized networks, EIGRP and OSPF calculate the shortest path in a relatively close amount of time .Therefore, it can be said that OSPF only needs to be chosen in very large networks. Otherwise, in most cases, EIGRP and OSPF are very similar in their efficiencies.

IX. Conclusion

Combination of both shortest path algorithm and bellman-ford algorithm provides useful problem solving method in networking approach by Coding of the algorithm is also included. This technique can be very useful to evaluate the shortest path in various networks. Dijkstra's algorithm assigns the label that determines the minimal length fromt he starting point of vertex itself, whereas Bellman-ford algorithm as compare to dijkstra's algorithm, acknowledges the edges with negative weight. Hence, we conclude both these algorithms were tested using pre-defined test cases and automated checking system available in the websites. These algorithms are acceptable in terms of their overall performance in solving the various problems related to computer networking, such as router communication and of routing protocol RIP.

References

[1] *http://www.cs.bgu.ac.il/~dinitz/Course/SS-12/BFD.pdf*
[2] *"Routing Algorithms for Computer Networks -- A Survey", J.M. McQuillan, 1977 National Telecommunications Conference, December 197*
[3] *Cisco Systems, OSPF Design Guide. Document ID 7039*
[4] *Dijkstra's Algorithm, Available at http://informatics.mccme.ru/moodle/mod/statements/view.php?id=193#1. 2012*
[5] *E.W. Dijkstra, "A note on two papers in connection with graphs," Numeriske Mathematics 1 pp.269-271 1959*
[6] *Bellman-Ford Algorithm, Available at http://informatics.mccme.ru/moodle/mod/statements/ view.php?id=260#1.2012*
[7] *Beaubrun R, Pierre S, "Routing algorithm for distributed communication networks," Proc 22nd IEEE Conference on Computer Networks,LCN 97 pp. 99- 105 Nov 1997*
[8] *Thomas H. Cormen, Charles E. Leiserson,Ronald L.Rivest, and Clifford Stein.Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 24.1: TheBellman-Ford algorithm, pp.588–592. Problem 24-1, pp.614–615.*
[9] *R. Bellman. On a routing problem.Quar . Appl. Math., 16:87-90, 1958.*