Different Routing Algorithm for computer Networks

Anupama Tewari*1,
Dr. Anuj Kumar2

1Research Scholar, Mewar University, rajasthan
2Supervisor, Professor, Accurate Institute of Science & Technology, G.Noida

ABSTRACT
This paper is the study about the different computer network routing algorithms available today. This is being done by analysing the limitations based on using algorithms in networks. It gives the idea of use of shortest path algorithm (Dijkstra's algorithm) individually in networking and also the use of Bellman-Ford algorithm. Both routing algorithms work differently 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.

Keywords: 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
INTRODUCTION

Computer networks have been growing at an enormous rate ever since the concept was first proposed, for example, networks were first designed with two different purposes which later on merged to form the internet. The military network (MILNET) was designed to keep military communications working under a war. The American University Networks (National Science Foundation Network, NSFNET / Advanced Research Projects Agency Network, ARPANET) were designed to make the exchange of research results easier.

The basic function of a data network is very simple: delivering data from one network node to another. Data networks can be viewed as huge graph with routers as vertices and transmission lines as edges. On such a huge graph the data packets must find their way from the source to the destination. Routing can be described as the process of creating a logical connection between nodes in a network so that packets sent by a node can reach their destination.

The routing algorithm is described by [2] as network layer protocol that guides the packets (information stored as small strings of bits) through the communication layer to their correct destinations. There are some reasons for the complexities 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).

COMMON PROBLEMS IN NETWORKING

“How do I best get from here to there?” This simple question is the essence of the routing problem, but it belies the considerable complexity embedded in modern intra-domain routing protocols. At the heart of this complexity is the issue of topology change. Routing in a static network is trivial, a simple table of directions calculated once for each destination. However, most real networks are dynamic—network links go up and down—and thus some nodes may need to be notified to recalculate their routes in response. This problem in turn can be boiled down to the question, “Who needs to know?” In the traditional approach, enshrined in the family of link-state protocols, is to tell everyone; flood the topology change throughout the network and have each node then recomputed its table of best routes.
However, as a network grows, this requirement to universally communicate and act on each topology change can become problematic. This is because a larger network also generates routing updates more often, necessitating more frequent route updates and route recomputation. Worse yet, these costs are incurred by every router in the network, meaning that the most resource-constrained router effectively determines the maximum network size that can be served by a routing algorithm. Thus, link-state protocols are frequently said to "not scale well."

Below are few common problems in networking:

- 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.
- 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.
- Low latency for end to end communication. Latency is the time taken by the packet to reach its destination from its source.
- Low latency jitter for end to end communication. Latency jitter is the variation in latency, for real time applications such as streaming video, the requirement for low latency jitter is more important than the requirement of low latency.
- High throughput for end to end communication. Throughput can be defined as the number of data packets delivered per second. Throughput is affected by packets being dropped, and protocol data units that are used by protocols to setup communication with peers.
- Low packet loss or High Reliability. Packet loss causes decrease in throughput and increases latency.
- Low convergence time in case of changes in network topology. It is necessary for routing algorithm to adapt to changes in network as quickly as possible, so that utilization of network resources is maximized.
- Low routing overhead. Routing overhead is caused by the update packets that are exchanged by routing protocols to convey network information to its peers. Routing overhead decreases throughput.
Routing Techniques for packet-switched networks can be broadly classified into static and adaptive routing policies. In Static routing, routing tables are setup at a certain time before the data are being transmitted and routing table is not changed thereafter. In adaptive routing, network conditions are continuously monitored and routing tables are changed dynamically to adapt to the changing network conditions. Adaptive routing can be further subdivided into centralized and distributed routing, depending on the storage of the routing information. Henceforth, we refer to adaptive, distributed routing simply as routing.

Packet routing in the Internet is divided into two general groups: interior and exterior routing. Interior routing happens inside or interior to an independent network system. In TCP/IP terminology, these independent network systems are called autonomous systems. Within an autonomous system (AS), routing information is exchanged using an interior routing protocol chosen by the autonomous system’s administration. The exterior routing protocols, on the other hand are used between the autonomous systems.

**Link State Algorithms**

In the link-state approach, each router maintains a complete view of the network topology with a cost for each link. A router broadcasts regularly the link cost information of its entire outgoing links to all other routers. Typically, this is done by flooding. That is, a router sends link cost information to all its neighbouring routers, who in turn forward the same information to their neighbours and so on. When a router receives information about the change in a link cost, it updates its view of the network topology and applies a shortest path algorithm to choose its next hop for each destination.

Routers may not always have a consistent view of the network topology, because of the time updates take to reach all routers. This inconsistent view of the network can lead to the formation of loops, which are temporary and disappear in the time it takes for all routers to have the same topological information.

**Shortest Path First (SPF)** is a link-state protocol in which each node computes and broadcast the cost of its outgoing links periodically and applies Dijkstra’s shortest path algorithm to determine the next hop; other routing protocols that work on the same link state approach are IS-IS and OSPF.
Distance-Vector Algorithms

In a distance-vector algorithm, a router knows the length of the shortest-path (distance) from each of its neighbors to every destination in the network, and uses this information to compute its own distance and next router (successor) to each destination. Well known examples of routing protocols based on distance-vector algorithms, which we call distance vector algorithms (DVA), are the routing information protocol (RIP), the HELLO protocol, the gateway-to-gateway protocol (GGP), the exterior gateway protocol (EGP) and old ARPANET routing protocol. All these DVAs have used variants of distributed Bellman-Ford algorithm (DBF) for shortest path computation. The primary disadvantage of DBF are routing table loops and counting to infinity. A routing table loop is a path specified in the routers' routing tables at a particular point in time, such that the path visits the same router more than once before reaching the intended destination until it reaches a predefined maximum distance value.

Because of the poor performance of DVAs implemented using DBF, DVAs were not considered to be viable approach to supporting routing in the large networks or internets. Recently, however, a number of efficient distance-vector algorithms have been proposed to eliminate the counting-to-infinity problem and routing table loops.

Distance vector algorithms use the Bellman–Ford algorithm. This approach assigns a cost number to each of the links between each node in the network. Nodes will send information from point A to point B via the path that results in the lowest total cost (i.e. the sum of the costs of the links between the nodes used).

The algorithm operates in a very simple manner. When a node first starts, it only knows of its immediate neighbors, and the direct cost involved in reaching them. (This information — the list of destinations, the total cost to each, and the next hop to send data to get there — makes up the routing table, or distance table.) Each node, on a regular basis, sends to each neighbor node its own current assessment of the total cost to get to all the destinations it knows of. The neighboring nodes examine this information and compare it to what they already 'know'; anything that represents an improvement on what they already have, they insert in their own routing table(s). Over time, all the nodes in the network will discover the best next hop for all destinations, and the best total cost.
The table below shows the categorization of different routing protocols with categorization of group and type of algorithm:

<table>
<thead>
<tr>
<th>Routing Protocols</th>
<th>Group</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGP/BGP4: Border Gateway Protocol</td>
<td>Exterior</td>
<td>Link State</td>
</tr>
<tr>
<td>EGP: Exterior Gateway Protocol</td>
<td>Exterior</td>
<td></td>
</tr>
<tr>
<td>EIGRP: Enhanced Interior Gateway</td>
<td>Interior</td>
<td>Distance Vector</td>
</tr>
<tr>
<td>Routing Protocol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSRP: Hot Standby Routing Protocol</td>
<td>Interior</td>
<td></td>
</tr>
<tr>
<td>IDRP: InterDomain Routing Protocol</td>
<td>Exterior</td>
<td>Link State</td>
</tr>
<tr>
<td>IGRP: Interior Gateway Routing Protocol</td>
<td>Interior</td>
<td>Distance Vector</td>
</tr>
<tr>
<td>IS-IS: Intermediate System to Intermediate System</td>
<td>Interior</td>
<td></td>
</tr>
<tr>
<td>Mobile IP for IPv4 and IPv6</td>
<td>Interior</td>
<td>Link State</td>
</tr>
<tr>
<td>NARP: NBMA Address Resolution Protocol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NHRP: Next Hop Resolution Protocol</td>
<td>Exterior</td>
<td>Distance Vector</td>
</tr>
<tr>
<td>OSPF: Open Shortest Path First</td>
<td>Interior</td>
<td>Link State</td>
</tr>
<tr>
<td>RIP (RIP2): Routing Information Protocol</td>
<td>Interior</td>
<td>Distance Vector</td>
</tr>
<tr>
<td>RIPng: RIP for IPv6</td>
<td>Interior</td>
<td>Distance Vector</td>
</tr>
<tr>
<td>RSVP: Resource ReSerVation Protocol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRRP: Virtual Router Redundancy Protocol</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Categorization of Routing Protocols

Here we will discuss the two algorithms from both categories – Link State & Distance Vector.

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.
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 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 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.

Rule 2: 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 field 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".

Rule 3: 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.

Rule 5: For all tentative nodes, directly linked to t node, status record set is updated.

Rule 6: From all the tentative nodes, choose the one whose weight to NS is less and set it as t node.

Rule 7: If this node is not the destination NT, then, go to step 5.

Rule 8: 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:

\[ D_i \] = Length of shortest path from node 'i' to node 1.

\[ d_{i,j} \] = Length of path between nodes i and j.

Algorithm: Each node j is labelled with \( D_j \), 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.

- Routers send out update messages whenever the state of a link changes. Hence the name: “Link State” algorithm.
- Each router calculates lowest cost path to all others, starting from itself.
- At each step of the algorithm, router adds the next shortest (i.e. lowest-cost) path to the tree.
- Finds spanning tree routed on source router.

**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 \( D[i,j] \) 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: \( D[i, 0] = \text{infinity}, \text{for all } i \neq 1 \)

Iteration: \( D[i, h+1] = \min \{ di, j + D[j, h] \} \) over all values of \( j \)

Termination: The algorithm terminates when \( D[i, h] = D[i, h+1] \) for all \( i \).

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. Every hour, 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 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.

PSEUDO CODE FOR DIJKSTRA’S ALGORITHM

```plaintext
// INITIALIZATION
S = {all nodes except source node s};
For all nodes v {
    if {v adjacent to s} {
        D[v] = C(s, v);
        R[v] = {s};
    }
    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 {
        c = D[u] + c(u, v);
        if (c < D[v]) {
            R[v] = {merge elements R[u], v};
            D[v] = c;
        }
    }
}
```
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

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

Plain 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 of d
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 from s.)

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.

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 from the 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: