Synchronous Distributed Path Computation for High Speed Networks

B. Gudarankaiah[#1], A. Rajesh[#2],

#1 Student, Sri Vasavi Engineering College, Thadepalli Gudem, West Godavari(dt),

#2 Asst. professor, Sri Vasavi Engineering College, Thadepalli Gudem, West Godavari(dt),

Abstract: Transient routing loops pose significant stability problems in networks. Distributed routing algorithms capable of avoiding such transient loops in network path are deemed efficient. Some earlier approaches like Shortest path routing (Dijkstra),Flooding, Flow-based routing, Distance vector routing(OSPF),Link state routing(Bellmen-Ford),Hierarchical routing, Broadcast routing, Multicast routing have problems maintaining the balance between node delays and link delays. In this paper, we propose a new algorithm, Distributed Path Computation with Intermediate Variables (DIV) that guarantees that no loops, transient or steady-state, can never downgrade network dynamics. Besides its ability to operate with existing distributed routing algorithms to guarantee that the directed graph induced by the routing decisions stays acyclic by implementing an update mechanism using simple message exchanges between neighboring nodes that guarantees loop freedom at all times. Specifically, the routing seeks robustness against failures by maximizing the number of next-hops available at each node for each destination. It provably outperforms existing loop prevention algorithms in several key metrics such as frequency of synchronous updates and the ability to maintain paths during transitions. DIV's routing capabilities that operates according to a non-shortest path objective will quantify its performance gains in simulations.

I INTRODUCTION

The multiple autonomous computers that communicate through computers is called Distributed computing. In this, the systems interact with each other to reach the destination. The computer program that runs in the distributed program and distributed programming. This will solve the computational problems which refer to Distributed computing.

The mode of information which is disseminated and subsequent computation is using the disseminated information, the new broad classes of algorithms are (i) Hierarchical routing, (ii) Broadcast routing. In this both algorithms, nodes will check the destination based on the information given by the algorithm. In the first algorithm the nodes will form in hierarchical model and the information from source to destination in the hierarchical model. In the Second routing algorithm, the data and packets from source to destination will reached through the broadcasting nodes. A **link-state routing protocol** is one of the two main classes of routing protocols used in packet switching networks for computer communications (the other is the distance-vector routing protocol). Examples of link-state routing protocols include OSPF and IS-IS.

The link-state protocol is performed by every *switching node* in the network (i.e. nodes that are prepared to forward packets; in the Internet, these are called routers). The basic concept of link-state routing is that every node constructs a *map* of the connectivity to the network, in the form of a graph, showing which nodes are connected to which other nodes. Each node then independently calculates the next best logical *path* from it to every possible destination in the network. The collection of best

paths will then form the node's routing table. This contrasts with distance-vector routing protocols, which work by having each node share its routing table with its neighbors. In a link-state protocol the only information passed between nodes is connectivity related. In computer communication theory relating to packet-switched networks, a distance-vector routing protocol is one of the two major classes of routing protocols, the other major class being the link-state protocol. Distance-vector routing protocols use the Bellman-Ford algorithm, Ford–Fulkerson algorithm, or DUAL FSM (in the case of Cisco Systems's protocols) to calculate paths.

II REALTED WORK

To the best of our knowledge, **Routing** is the process of selecting paths in a network along which to send network traffic. Routing is performed for many kinds of networks, including the telephone network (Circuit switching), electronic data networks (such as the Internet), and transportation networks. This article is concerned primarily with routing in electronic data networks using switching technology.

In packet switching networks, routing directs packet forwarding, the transit of logically addressed packets from their source toward their ultimate destination through intermediate nodes, typically hardware devices called routers, bridges, gateways, firewalls, or switches. General-purpose computers can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time, but multipath routing techniques enable the use of multiple alternative paths.

Routing, in a more narrow sense of the term, is often contrasted with bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Because structured addresses allow a single routing table entry to represent the route to a group of devices, structured addressing (routing, in the narrow sense) outperforms unstructured addressing (bridging) in large networks, and has become the dominant form of addressing on the Internet, though bridging is still widely used within localized environments.

III BACKGROUND

A. Distance Vector Routing

A distance-vector routing protocol requires that a router informs its neighbors of topology changes periodically. Compared to link-state protocols, which require a router to inform all the nodes in a network of topology changes, distance-vector routing protocols have less computational complexity and message overhead.

In computer communication theory relating to packet-switched networks, a **distance-vector routing protocol** is one of the two major classes of routing protocols, the other major class being the link-state protocol. Distance-vector routing protocols use the Bellman-Ford algorithm, Ford–Fulkerson algorithm, or DUAL FSM (in the case of Systems' protocols) to calculate paths.

A distance-vector routing protocol requires that a router informs its neighbors of topology changes periodically. Compared to link-state protocols, which require a router to inform all the nodes in a network of topology changes, distance-vector routing protocols have less computational complexity and message overhead. The term *distance vector* refers to the fact that the protocol manipulates *vectors* (arrays) of distances to other nodes in the network. Routers using distance vector protocol do not have knowledge of the entire path to a destination. Instead DV uses two methods:

1. Direction in which or interface to which a packet should be forwarded.
2. Distance from its destination.

The methods used to calculate the best path for a network are different between different routing protocols but the fundamental features of distance-vector algorithms are the same across all DV based protocols.

B. Count-to-Infinity Problem

The Bellman-Ford algorithm does not prevent routing loops from happening and suffers from the **count-to-infinity problem**. The core of the count-to-infinity problem is that if A tells B that it has a path somewhere, there is no way for B to know if the path has B as a part of it. To see the problem clearly, imagine a subnet connected like as A-B-C-D-E-F, and let the metric between the routers be "number of jumps". Now suppose that A is taken offline. In the vector-update-process B notices that the route to A, which was distance 1, is down - B does not receive the vector update from A. The problem is, B also gets an update from C, and C is still not aware of the fact that A is down - so it tells B that A is only two jumps from C (C to B to A) , which is false. This slowly propagates through the network until it reaches infinity (in which case the algorithm corrects itself, due to the "Relax property" of Bellman-Ford).

IV OVERVIEW OF DIV

Most previous distance-vector algorithms free from transient loops follow a common structure: Nodes exchange update-messages to notify their neighbors of any change in their own cost-to-destination (for any destination). If the cost-to-destination decreases at a node, the algorithms allow updating its neighbors in an arbitrary manner; these updates are called *local* (asynchronous) updates. However, after an increase in the cost-to-destination of a node, these algorithms require that the node potentially update all its upstream nodes *before* changing its current successor; these are *synchronous* updates. Algorithms differ in handling situations where during the propagation of a node's cost-to-destination changes. Note that the primary challenge in avoiding transient loops lies in handling inconsistencies in the information stored across different nodes. Otherwise, simple approaches can guarantee loop-free operations at each step. In this context, approaches that are "in-between" link-state and distance vector and avoid counting-to-infinity are also possible achieves this by having nodes learn the penultimate nodes in the shortest paths to each destination from its neighbors.

DIV lays down a set of rules on existing routing algorithms to ensure their loop-free operation at each instant. This rule-set is not predicated on shortest path computation, so DIV can be used with other path computation algorithms as well. For each destination, DIV assigns a *value* to each node in the network. To simplify our discussion and notation, we fix a particular destination and from now on, speak of the value of a node. The values can be arbitrary hence, the independence of DIV from any underlying path computation algorithm. However, usually the value of a node will be related to the underlying objective function that the routing algorithm attempts to optimize and the network topology.

Some typical value assignments are as follows: (i) in shortest path computations, the value of a node could be its cost-to-destination; (ii) as done in DUAL, the value could be the minimum cost-to-destination seen by the node from time ; (iii) as done in TORA , the value could be the *height* of this node; (iv) as illustrated in the value could be related to the number of next-hop neighbors for the destination, etc. We, however, impose one restriction on the value assignment: a node that does not have a path to a destination must assign a value of "infinity" (the maximum possible value) to itself. Intuitively, this restriction prevents other nodes from using it as a successor which is sensible since it does not have a path to the destination in the first place. This restriction turns out to be crucial for avoiding counting-to-infinity problems in shortest path environments.

V PATH SELECTION

Path selection involves applying a routing metric to multiple routes, in order to select (or predict) the best route. In the case of computer networking, the metric is computed by a routing algorithm, and can cover such information as bandwidth, network delay, hop count, path cost, load, MTU, reliability, and communication cost (see e.g. this survey for a list of proposed routing metrics). The routing table stores only the best possible routes, while link-state or topological databases may store all other information as well. Because a routing metric is specific to a given routing protocol, multi-protocol routers must use some external heuristic in order to select between routes learned from different routing protocols. Cisco's routers, for example, attribute a value known as the administrative distance to each route, where smaller administrative distances indicate routes learned from a supposedly more reliable protocol.A local network administrator, in special cases, can set up host-specific routes to a particular machine which provides more control over network usage, permits testing and better overall security. This can come in handy when required to debug network connections or routing tables.

VI CHALLENGES OF DIV

There are four approaches for implementation DIV which address a number of challenges.

1.) **The ip address stored at the each node** from the client point of view we have to access the each and node with the ip address. Here, Client-server computing or networking is a distributed application architecture that partitions tasks or workloads between service providers (servers) and service requesters, called clients. Often clients and servers operate over a computer network on separate hardware. A server machine is a high-performance host that is running one or more server programs which share its resources with clients. A client also shares any of its resources; Clients therefore initiate communication sessions with servers which await (listen to) incoming requests

2.) **Dynamic random approach** We choose the Random Waypoint (RWP) Model whose limiting distribution was derived in and shown to be independent of node velocity. Each node moves with a velocity of 10 m/s in each travel interval, and the pause time is 0 seconds. We choose a "sampling interval" Ts of 5 seconds, with which each node samples the energy field. Since the cell size is 50 m * 50 m, on average, a node will have moved to a new cell before it makes a new measurement. We assume that the transmit energy is dominant over receiver energy and that the path loss exponent is q = 2. The transmission range of each node is 120 meters.

3.) **The rules updating the ip address** means it will randomly select the each and every node for the transfer of data and packets. We can update the ip address of the each and every node for the efficient transfer of the packets.

4.) **Will update message delivery from each and every node i.e.,** here in this approach using DIV will send data or packets of data will be updated or not.

VII CONCLUSION

Among all the algorithms distance vector algorithms have most features over link state algorithms i.e., it will provide the high stability and it will affect the changes local. The inconsistent decisions of the nodes will affect the impact and duration between the nodes. These descriptions itself through transient loops and the counting to infinity problem described this paper. In this paper, finally comparing all the algorithms and selecting the efficient algorithms to update mechanism for enforcing rules. To implement we need distance vector algorithm, distributed path computation with intermediate (DIV). Here DIV not integrated with shortest path computations, it can used with any routing algorithm. Distance-vector algorithms have advantages over link-state algorithms, e.g., lower resource requirements and often greater stability by keeping the impact of changes local. When we integrated with shortest algorithms DIV with perform more efficient compare with other algorithms.

VIII REFERENCES

[1] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and K. vander Merwe, "The case for separating routing from routers," in *Proc. ACM SIGCOMM Workshop FDNA*, Portland, OR, Sep. 2004, pp. 5–12.

[2] A. Shankar, C. Alaettinoglu, K. Dussa-Zieger, and I. Matta, "Transient and steady-state performance of routing protocols: Distance-vector versus link-state," *Internetw.: Res. Exper.*, vol. 6, no. 2, pp. 59–87, Jun. 1995.

[3] A. Myers, E. Ng, and H. Zhang, "Rethinking the service model: Scaling Ethernet to a million nodes," presented at the ACM SIGCOMM Hot- Nets, San Diego, CA, 2004.

[4] Y. Ohba, "Issues on loop prevention in MPLS networks," *IEEE Commun. Mag.*, vol. 37, no. 12, pp. 64–68, Dec. 1999.

[5] P. Francois and O. Bonaventure, "Avoiding transient loops during IGP convergence in IP networks," in *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005, vol. 1, pp. 237–247.

[6] J. Moy, "OSPF version 2," Internet Engineering Task Force, RFC 2328, 1998 [Online]. Available: http://www.rfc-editor.org/rfc/rfc2328.txt

[7] A. Shaikh and A. Greenberg, "Experience in black-box OSPF measurement," in *Proc. ACMSIGCOMM IMW*, San Francisco, CA, Nov. 2001, pp. 113–125.

[8] G. Malkin, "RIP version 2," Internet Engineering Task Force, RFC 2453, 1998 [Online]. Available: http://www.rfc-editor.org/ rfc/ rfc2453.txt

[9] R. Albrightson, J. J. Garcia-Luna-Aceves, and J. Boyle, "EIGRP – A fast routing protocol based on distance vectors," presented at the Netw./ Interop. 1994.

[10] E. Gafni and D. Bertsekas, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," *IEEE Trans. Commun.*, vol. COM-29, no. 1, pp. 11–18, Jan. 1981.

[11] P. M. Merlin and A. Segall, "A failsafe distributed routing protocol," *IEEE Trans. Commun.*, vol. COM-27, pp. 1280–1288, Sep. 1979.

[12] J. M. Jaffe and F. M. Moss, "A responsive routing algorithm for computer networks," *IEEE Trans. Commun.*, vol. 30, pp. 1768–1762, Jul. 1982.

[13] J. J. Garcia-Lunes-Aceves, "Loop-free routing using diffusing computations," *IEEE/ACMTrans. Netw.*, vol. 1, no. 1, pp. 130–141, Feb. 1993.

[14] S. Vutukury and J. J. Garcia-Luna-Aceves, "A simple approximation to minimum-delay routing," in *Proc. ACM SIGCOMM*, 1999, pp. 227–238.

[15] S. Vutukury and J. J. Garcia-Luna-Aceves, "MDVA: A distance-vector multipath routing protocol," in *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001, vol. 1, pp. 557–564.