DYNAMICS OF A LOOP-FREE PATH-FINDING ALGORITHM

Shree Murthy and J.J. Garcia-Luna-Aceves

Computer Engineering Department, 225, Applied Sciences Building
University of California, Santa Cruz, CA 95064
U.S.A

Abstract The dynamics of a loop-free path-finding algorithm (LPA) based on predecessor information and a single-hop internodal synchronization mechanism is investigated. LPA is compared with a loop-free algorithm based on diffusing computations, DUAL, and an ideal link-state (ILS) algorithm based on topology broadcast. Comparisons include the dynamic response of the algorithms to a single and multiple link-cost changes as well as single link and router failures and recoveries. The results show that LPA requires a significantly smaller number of messages than ILS and DUAL to update routing tables when multiple changes in link costs occur. LPA's performance is always significantly better than DUAL's and significantly better than ILS's after node failures and resource additions (in some instances, ILS requires almost four times as many messages). After a link failure, LPA requires approximately the same time to converge as ILS and at most twice as many messages.

I. INTRODUCTION

The shortest-path algorithms used in computer networks today, can be classified as *distance-vector* or *link-state* algorithms. Many of these distance-vector routing protocols are based on the distributed Bellman-Ford algorithm (DBF) for shortest-path computation [2]. In order to overcome the *counting-to-infinity* problem and *bouncing effect* of DBF, several shortest-path algorithms [1], [9], [10], [12] have been proposed. These algorithms, known as *path-finding algorithms* utilize information regarding the distance and the second-to-last hop (predecessor) of the shortest path to each destination. Although these algorithms provide a marked improvement in performance over DBF, they do not eliminate the possibility of temporary loops. The loop-free algorithms reported to date rely on mechanisms that require routers to either synchronize along multiple hops [5], [3], [11], or exchange path information that can include all the routers in the path from source to destination [7]. We have presented and verified elsewhere [8] the first path-finding algorithm that is loop-free at every instant, which we call the *loop-free path-finding algorithm* (LPA). This paper provides further insight into LPA by analyzing its dynamic behavior. We compare LPA with DUAL [5] and an ideal link-state algorithm (ILS).

Like previous path-finding algorithms, LPA eliminates the counting-to-infinity problem of DBF using the predecessor information. Since each router reports to its neighbors the predecessor to each destination, any router can traverse the path specified by a predecessor from any destination back to a neighbor router to determine if using that neighbor as its successor would create a path that contains a loop (i.e., involves the router itself). To block a potential temporary loop, a router sends a query to all its neighboring routers reporting an infinite distance to a destination before it changes its routing table; the router is free to choose a new successor only when it receives all the replies from its neighbors in response to its synchronization queries. To reduce the communication overhead incurred with the interneighbor coordination mechanism, routers use a *feasibility condition* to limit the number of times queries will be exchanged among neighbors. In contrast to many prior loop-free routing algorithms [5], [3], [11], queries propagate only one hop in LPA. Furthermore, updates and routing-table entries in LPA require a single node identifier as path information rather than a variable number of node identifiers as in previous algorithms [7]. The results show that LPA performs better for single and multiple changes. LPA's performance is always significantly better than DUAL's.

This work was supported in part by the Office of Naval Research under Contract No. N-00014-92-J-1807 and by the Advanced Research Projects Agency (ARPA) under contract F19628-93-C-0175

Section II gives a brief description of LPA with an example. Sections III and IV describe the design of the simulator and the instrumentation we do to analyze the routing algorithm respectively. In Section V, we discuss the dynamic behavior of LPA and compare its performance with that of DUAL and ILS, an ideal link-state algorithm.

II. LPA DESCRIPTION

LPA is built on two basic mechanisms: using *predecessor information* to eliminate counting-to-infinity problem and blocking temporary routing loops using *an inter-neighbor synchronization* mechanism similar to the one proposed in [6].

Each router maintains a distance table, a routing table and a link-cost table. The distance table at each router i maintains the distance and the predecessor information to each destination j through each of its neighboring nodes. The routing table contains information about the shortest-path to destination j, predecessor and successor to j along the shortest path. A *tag* entry for destination j (tag_j^i) specifies whether the entry corresponds to a simple path, a loop or a destination that has not been marked (correct, error and null respectively). The link-cost table lists the cost of each link.

Using the predecessor information, each router can infer if the path corresponding to a distance-table or a routing-table entry includes the router itself. This feature eliminates the counting-to-infinity problem of DBF. Furthermore, temporary loops can be detected by each router within a finite time. This depends on the speed with which correct predecessor information reaches the router and not on its distance (number of hops). When a router determines that a loop may be formed if it changes its current successor to a given destination, it blocks such a loop by reporting an infinite distance for that destination to all its neighbors by sending a query and waits till it receives reply from its neighbors about the path (distance and predecessor) information before changing its current successor. In order to reduce the overhead involved in sending a query every time a node wants to change its successor, the following *feasibility condition* is introduced.

Feasibility Condition (FC): If at time t router i needs to update its current successor, it can choose as its new successor $s_j^i(t)$ any router $n \in N_i(t)$ such that i is not present in the implicit path to j reported by neighbor n, $D_{jn}^i(t)+d_{in}(t) = D_{min}(t) = Min\{D_{jx}^i(t)+d_{ix}(t)|x \in N_i(t)\}$ and $D_{jn}^i(t) < FD_j^i(t)$. If no such neighbor exists and $D_j^i(t) < \infty$, router i must keep its current successor. If $D_{min}(t) = \infty$ then $s_j^i(t) = null$.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1995		**00-00-1995 to 00-00-1995**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Dynamics of a Loop-Free Path-Finding Algorithm	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**6**	

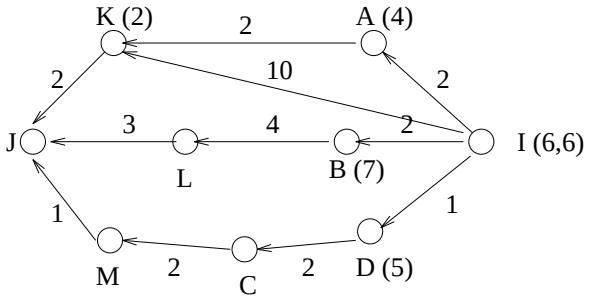

Fig. 1. Example: LPA's Operation

Before sending a query, a router compares the distances reported by its neighbors against its feasible distance. A *feasible distance* (FD) is defined as the smallest distance achieved by the router's own distance since the last query sent by the router. A router sends a query, thus blocking a potential loop when none of its neighbors reports a distance smaller than the router's own feasible distance. This feature accounts for the low overhead incurred in LPA to accomplish loop-free paths at every instant. LPA uses a *tagging* mechanism to ensure that it updates only those routing table entries which are affected by the input event.

The updating of the routing table for a given destination depends on whether the router is *active* or *passive* for that destination. A router is passive if it has a feasible successor. An active router cannot send an update regarding the destination; this is because an update during active state would have to report an infinite distance to ensure that the inter-neighbor synchronization mechanism used in LPA provides loop-freedom at every instant. LPA is verified to be correct in [8].

Figure 1 illustrates LPA's operation. In this network, links and nodes have the same processing time or propagation delays; the operation of the algorithm is discussed for the case when a link goes down. The number adjacent to the link indicates the cost of the link. The arrowhead from node x to node y indicates that node y is the successor of node x towards destination j (i.e., $s_j^x = y$). The label in parenthesis assigned to node x indicates the current distance (D_j^x) and the feasible distance from x to j (FD_j^x) respectively.

In Figure 1, the current path from node i to j is through node d. Assume that link (i, d) fails. Node i uses FC to decide whether it has a feasible successor or not. Applying FC, node i determines that node a is a feasible successor, because it reported a distance D_j^a=4 and that distance plus the cost of the link (i, a) (i.e., $D_j^a + l_a^i$) equals 6, which is the minimum distance that node i can achieve through any of its remaining neighbors (nodes a, b and k). If link (i, a) did not exist, i.e., if node a were not node i's neighbor, node i would have to send a query to all its remaining neighbors, because none of them satisfies FC. Node k has reported a distance $D_j^k = 2 < FD_j^i = 6$; however $D_j^k + l_k^i$=12 is larger than the minimum distance node i can achieve, which is 9 through node b. On the other hand, node b does not satisfy FC because it has reported a distance $D_j^b = 7 > FC_j^i = 6$.

III. SIMULATION DESIGN

We have developed simulations using an actor-based, discrete-event simulation language called *Drama* [13], together with a network simulation library. Link failures and recoveries are handled by sending a link-status message to the routers at the end points of the appropriate link. In the link models used in the simulation, each link responds to a packet by encapsulating the packet in another message and sending the message to the link itself. The link propagation time is an input parameter, although all runs were made with unit propagation time. If a link fails, packets in transit are dropped.

For routing algorithm simulations in our study, a router receives a packet and responds by running a routing algorithm, then queueing any outgoing updates, and finally by waiting for some processing time. If any incoming packets arrive before the processing time expires, the routing algorithm is run again and any newly generated packets are queued. Once the processing time for all events has expired, redundant updates are removed (algorithm dependent) and the message queues are sent over the links. In the runs actually made, processing time was set to zero. This assumption was made as the procedures used for each event are very simple. The link propagation time was set to unit time. *Drama*'s internal mechanisms (FCFS) ensure that all updates due to arrive at the current simulation time were processed before any new updates were generated. The advantage of this approach is that simulation puts multiple updates in the same packet. Therefore, the number of packets becomes a more critical measure of performance than the number of bits actually transmitted. The number of packets in the simulation is therefore a lower bound on the number that would actually be sent because of the possibility of packet fragmentation. In the simulation runs under study, the packet lengths are short so that, on an average, the fragmentation is not significant.

Three distributed shortest-path routing algorithms, Loop-free path-finding algorithm (LPA) [8], the Diffusing Update Algorithm (DUAL) [5] and an Ideal Link-State Algorithm (ILS) which uses Dijkstra's shortest-path algorithm at each node, were studied. In all cases, the algorithms produced a routing table giving a successor for each destination. The successor is either a neighboring node or null. A null successor entry indicates that a node does not have a path which satisfies FC to a given destination. We ensure that all redundant updates were removed in the algorithms simulated.

IV. INSTRUMENTATION

The simulation has been instrumented in two ways. A set of counters that can be reset at various points are maintained. These counters determine statistics such as the total number of times all the nodes in the simulation responds to a packet and the total number of messages sent. When the event queue empties indicating that the algorithm has converged, the counter values are noted. Some counters are associated with individual nodes and links in the simulation whereas others are associated with all nodes and links.

The simulation also gathers information after each event has been processed. In particular, a copy of the routing table is analyzed at each step of the simulation thereby allowing the characterization of the routes each algorithm has produced while the algorithm is still running. Each copy of the routing table was generated just before a node's script returned. For each algorithm, the routing table of each router is also analyzed at each step of the simulation for characterizing the routes produced while the algorithm was running.

To obtain the average figures, each link (router) in the network is made to fail, and the number of steps and messages needed for each algorithm to converge is counted. Then the same link (router) is made to recover and the process is repeated. The average is then taken over all link (router) failures and recoveries. The routing algorithm was allowed to converge after each such change. In all cases, routers were assumed to perform computations in zero time and links were assumed to provide one time unit of delay. For the failure and recovery runs, the costs were set to unity. Both the mean and the standard

deviation were computed for each counter; the four counters used are, the number of updates and changes in link status processed by routers (events), the number of packets transmitted over the network (packets), time taken for the algorithm to converge (duration) and the number of operations performed by the algorithm (operations). Several quantities were measured as a function of time and an ensemble average over link-cost changes was measured as a function of the time since the change in cost. These include the probability that there is a message in transit, the average number of packets and the average number of messages given that at least one message is in transit.

For the routing algorithms under consideration, there is only one shortest path between a source and a destination pair and we do not consider null paths from a router to itself. That is, a network with N routers when it is fully connected will have $N - 1$ paths. During each run, data was collected for a large number of topology changes to determine statistical distributions. In addition, multiple runs were run to determine the statistical errors.

V. SIMULATION RESULTS

The simulations were run on several network topologies such as *Los-Nettos*, *Nsfnet* and *ARPANET*. We choose these topologies to compare the performance of routing algorithms for well-known cases, given that we cannot sample a large enough number of networks to make statistically justifiable statements about how an algorithm scales with network parameters. In this paper, we focus on the results for ARPANET topology only.

For each network, we generated test cases consisting of all single failures and recoveries for both routers and links in which the routing algorithms were allowed to converge after each such change. We have also simulated the algorithms for random link cost changes; links were chosen at random, with link costs chosen from the interval (0,1] and with a Poisson distributed interarrival time. For link cost changes, five independent runs were made and the averages and standard deviations of all quantities measured were determined. Interarrival time between the link-cost changes was varied to simulate multiple link-cost changes.

In all cases, nodes were assumed to perform computation in zero time, and links were assumed to provide one time unit of delay. The link model allows link delay and link cost to be set independently. Each unit of time therefore represents a step in which all currently available packets are processed. Although the choice of input parameters causes the simulation to proceed synchronously, the node model treats each incoming packet asynchronously. Each input event is processed independently of other events received during the same simulation step.

A. Total Response to a Single Change

The performance of the routing algorithms for a single resource failure is presented in [8]. The results indicate that LPA outperforms DUAL for a single resource failure. The average performance of LPA and DUAL is better than ILS after a resource addition and LPA's performance is comparable to ILS after a resource failure.

The response of the algorithms for a single link-cost change is given in Table I. The table gives the average value and the standard deviation along with the statistical error for each link-cost change. The statistical errors were determined based on repeated trails. For a single link-cost change, LPA is faster and needs fewer messages and operations than both DUAL and ILS in all topologies. We can conclude from these results that LPA has a better average performance than ILS and DUAL after any single link-cost or topology change.

B. Dynamic Response to a Single Change

To study the dynamic behavior of the routing algorithms, we ran an exhaustive series of test cases for all the node and the link failures and recoveries and recorded the message related statistics. A statistical characterization of the performance of the routing algorithms was obtained by treating every node change as a separate case and by computing a distribution as a function of time. In this section, we present the results of the dynamic behavior of the routing algorithms for the *ARPANET* topology. In the instrumentation, we do not consider paths from a node to itself, because they do not require a network. In our simulations, we have taken care to handle these cases.

Figures 2–9 show the transient response of the routing algorithms (probability of packets in transit and the average number of messages that are exchanged) after a link failure, link recovery, node failure and node recovery respectively. These are shown as a function of time.

The results indicate that for a link failure, ILS performs better than DUAL and LPA in terms of the number of messages exchanged. LPA's average performance is much better than ILS for resource addition at any time after the change. The performance of LPA is comparable to ILS after a node failure. In all cases, LPA outperforms DUAL. The probability of packets being in transit for LPA after a resource recovery is less than ILS and DUAL. The average packet length for LPA after a resource change is much smaller than DUAL. This is because of the single hop interneighbor coordination mechanism and the tagging mechanism used in LPA.

C. Response to Multiple Link-Cost Changes

The steady-state behavior of the algorithms is more interesting with multiple link-cost changes than the transient response after each topology change. Figures 10–13 shows the average number of update messages when such messages are in transit, the average lengths of messages, the average number of messages in transit and the probability that the messages are in transit as a function of the interarrival times between link-cost changes for LPA, DUAL and ILS. This again is for the ARPANET topology. From [14], it has been observed that the behavior of DUAL and ILS for multiple link-cost changes is similar for different network topologies; our conjecture is that the same is true for LPA.

For very long interarrival times, the number of messages during busy periods is independent of the interarrival time because the probability of two topology changes occurring simultaneously is small. In this case, the performance approaches that of single link-cost change. When the interarrival time approaches the network diameter, this situation changes and the number of messages during the busy period increases because of multiple topology changes occurring simultaneously.

The average number of messages exchanged when messages are in transit is slightly less for LPA compared to DUAL and ILS. This again is because of the single-hop internodal synchronization mechanism and updating of the distance-table entries, which has been explained earlier. All curves are of roughly the same shape. For ILS, the average message length is close to 1. DUAL and LPA have longer messages as a message can contain multiple updates; this occurs when messages from the routers at both ends of a link that changes cost arrive at some router at the same time. With the increase in the interarrival time between changes, the average number of messages drops down as it now approaches

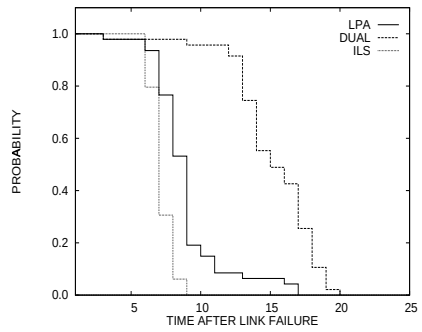

Fig. 2. Probability of packets in transit for Link Failure

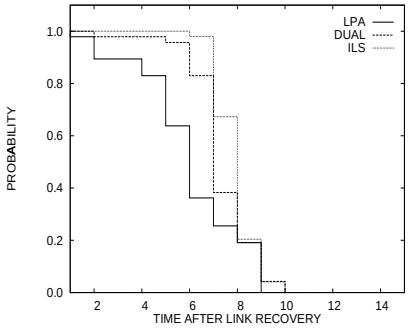

Fig. 3. Probability of packets in transit for Link Recovery

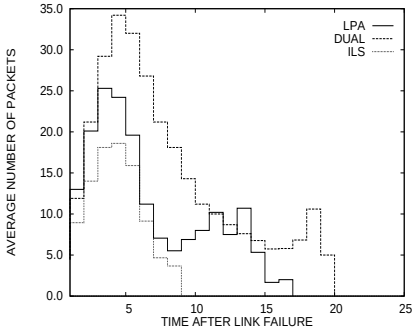

Fig. 4. Average number of packets for Link Failure

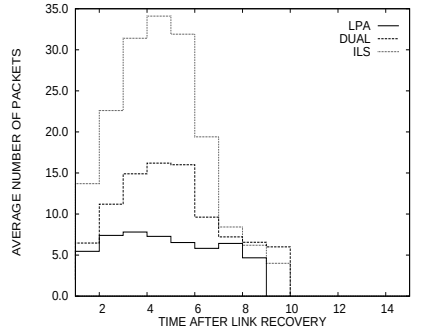

Fig. 5. Average number of packets for Link Recovery

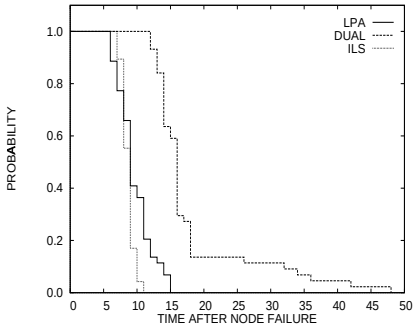

Fig. 6. Probability of packets in transit for Node Failure

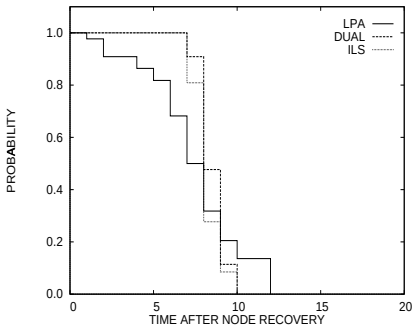

Fig. 7. Probability of packets in transit for Node Recovery

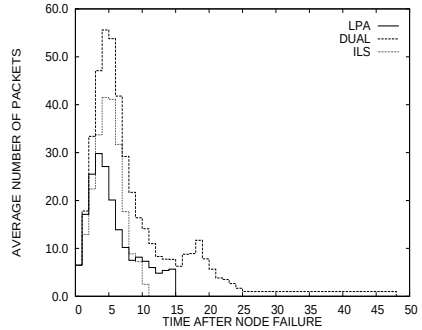

Fig. 8. Average number of packets for Node Failure

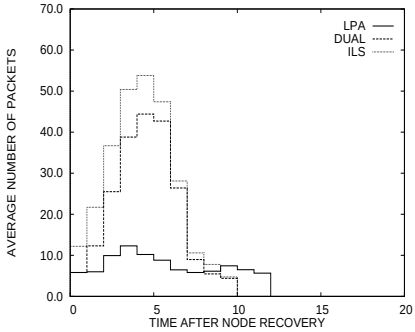

Fig. 9. Average number of packets for Node Recovery

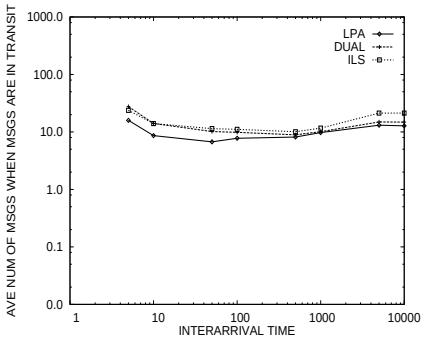

Fig. 10. Average Number of Messages when messages are in Transit

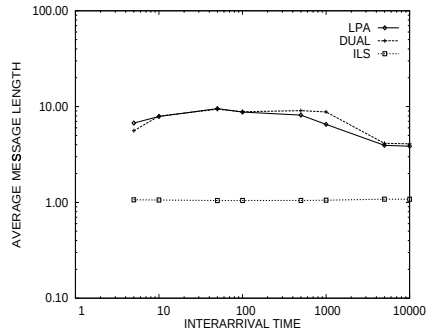

Fig. 11. Average Message Length

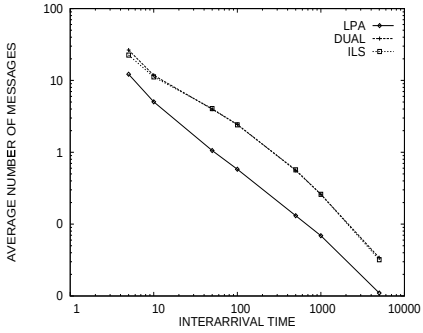

Fig. 12. Average Number of Messages in Transit

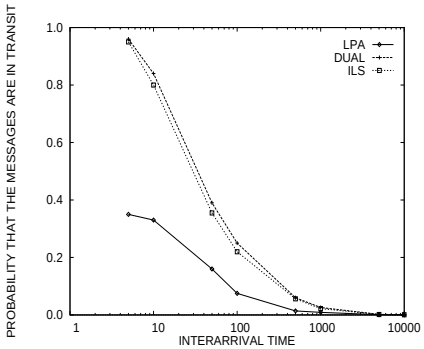

Fig. 13. Probability that Update Messages are in Transit

TABLE I
ROUTING ALGORITHM RESPONSE TO A CHANGE IN LINK COST

Parameter	LPA		DUAL		ILS	
	mean	sdev	mean	sdev	mean	sdev
Los-Nettos Cases						
Event Count	21.4± 0.47	35.6± 2.8	34.0± 0.24	45.0± 0.54	18.5± 0.05	18.5± 0.04
Packet Count	12.2±0.15	7.5± 0.41	15.45± 0.14	18.6± 0.25	17.5± 0.05	17.5± 0.04
Duration	3.95± 0.06	1.76± 0.07	4.9±0.06	2.17± 0.07	4.06± 0.01	0.46± 0.03
Operation Count	45.7± 0.23	23.1± 1.1.7	44.0± 0.24	52.9± 0.5	473.9± 3.08	475.5± 3.09
NSFNET Cases						
Event Count	34.65± 0.83	72.86± 2.7	50.97± 1.2	68.7± 2.5	28.5± 0.09	28.5± 0.09
Packet Count	14.97± 0.15	14.78± 0.47	21.81± 0.54	27.1± 0.82	27.5± 0.1	27.5± 0.09
Duration	4.624± 0.05	2.45± 0.05	5.516± 0.17	2.57± 0.15	4.7± 0.02	0.46± 0.01
Operation Count	55.323± 0.42	42.64± 1.21	63.97± 1.2	78.8± 2.35	1012.88± 3.9	1014.5± 3.76
ARPANET Cases						
Event Count	247.5± 16.09	679.7± 24.6	350.09± 15.08	501.4± 22.7	84.1± 0.23	84.6± 0.21
Packet Count	55.8± 1.86	66.8± 3.0	81.8± 2.99	102.9± 4.9	83.1± 0.23	83.6± 0.21
Duration	7.042± 0.22	4.9± 0.09	10.84± 0.45	4.75± 0.55	7.74± 0.028	0.74± 0.022
Operation Count	269.7± 8.05	359.2± 12.2	396.1± 15.1	534.6± 22.6	13613.1± 51.0	13691.3± 47.4

towards a single link-cost change. The average number of messages that are in transit and its probability are significantly smaller for LPA compared to DUAL and ILS. However, DUAL and ILS roughly follow the same curve. The results clearly indicate that LPA incurs smaller overhead traffic than either DUAL and ILS when multiple link-cost changes occur.

VI. CONCLUSION

We have presented a complete analysis of the dynamic behavior of a loop-free path-finding algorithm (LPA) that uses distance vectors with the distance and the predecessor to each destination, and a single hop internodal synchronization mechanism for achieving loop freedom. The dynamic behavior of this algorithm is compared with that of DUAL and ILS.

The statistical techniques used in our analysis provide a way of characterizing the performance of various algorithms, and can be used as a basis for a tradeoff analysis during network design. The time behavior of loop-free distance vector algorithms shows that parameters such as packet length can change as updates propagate, thereby suggesting the possibility of heuristics that can exploit local conditions. Our analysis indicates that overall, LPA has better performance among the three algorithms simulated. LPA's performance is always significantly better than that of DUAL. LPA requires a significantly smaller number of messages than ILS and DUAL to update routing table for multiple link-cost changes.

REFERENCES

[1] C. Cheng, R. Reley, S. P. R Kumar and J. J. Garcia-Luna-Aceves, "A Loop-Free Extended Bellman-Ford Routing Protocol without Bouncing Effect", *ACM Computer Communications Review*, Vol.19, No.4, 1989, pp.224–236.

[2] L.R. Ford and D.R. Fulkerson, *Flow in Networks*, Princeton University Press, Princeton, New Jersey, 1962.

[3] J.M. Jaffe and F.M. Moss, "A Responsive Routing Algorithm for Computer Networks", *IEEE Trans. Comm.*, Vol.30, July 1982, pp.1758–1762.

[4] J. J. Garcia-Luna-Aceves, "A Fail-Safe Routing Algorithm for Multihop Packet Radio Networks", *Proc. of IEEE INFOCOM*, Miami, Florida, 8–10 April, 1986, pp.434–443.

[5] J. J. Garcia-Luna-Aceves, "Loop-free Routing using Diffusing Computations", *IEEE/ACM Transactions on Networking*, Vol. 1, No. 1, Feb, 1993, pp.130–141.

[6] J. J. Garcia-Luna-Aceves, "Distributed Routing with Labeled Distances", *Proc. of IEEE INFOCOM*, Conference on Computer Commun. Florence, Italy, 4–8 May 1992, pp.633–643.

[7] J. J. Garcia-Luna-Aceves, "LIBRA: A Distributed Routing Algorithm for Large Internets", *Proc. of IEEE Globecom*, Orlando, Florida, 6–9 Dec 1992, pp.1465–1471.

[8] J.J. Garcia-Luna-Aceves and S. Murthy, "A Loop-Free Path-Finding Algorithm: Specification, Verification and Complexity", *Proc. of IEEE INFOCOM*, Boston, 4–6 April 1995.

[9] J. Hagouel, "Issues in Routing for Large and Dynamic Networks," IBM Research Report RC 9942 (No. 44055) Communications, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, April 1983.

[10] P.A. Humblet, "Another Adaptive Shortest-Path Algorithm", *IEEE Trans. Comm.*, Vol.39, No.6, June 1991, pp.995–1003.

[11] P.M. Merlin and A. Segall, "A Failsafe Distributed Routing Algorithm", *IEEE Trans. Comm.*, Vol.27, Sept. 1979, pp.1280–1288.

[12] B. Rajagopalan and M. Faiman, "A Responsive Distributed Shortest-Path Routing Algorithm within Autonomous Systems," *Internetworking: Research and Experience*, Vol.2, No.1, March 1991, pp. 51-69.

[13] W. T. Zaumen, "Simulations in Drama", *Network Information System Center, SRI International*, Menlo Park, California, January 1991.

[14] W. T. Zaumen and J. J. Garcia-Luna-Aceves, "Dynamics of Link-state and Loop-free Distance-vector Routing Algorithms", *Internetworking: Research and Experience*, Vol. 3, 1992, pp.161–188.