An Empirical Approach for Minimal Delay Routing of Distributed Computing

Titiphan Kitcharoensup
Peraphon Sophatsathit

Department of Mathematics, Faculty of Science
Chulalongkorn University, Thailand

Abstract
This paper presents a new empirical method based on the concepts of the Additive Approximate Distributed Bellman-Ford algorithm and a Fuzzy Set Delay Representation for Computer Network Routing algorithm. The efficiency of each algorithm depends on how well it adapts to the traffic and topology changes in the network. The major influential factor is the length of the connecting links, which may change dynamically either due to individual link failure and repair, or traffic conditions in the network. We propose a new empirical approach to find the minimum delay path (or equivalently the shortest path) in polynomial message complexity. Performance improvement of the algorithm is carried out by applying fuzzy sets to model the uncertainty of delay information.

Introduction

Numerous routing algorithms have been employed in modern network management, namely, flooding, static routing, centralized routing, isolated routing, hierarchical routing, broadcast routing, and distributed routing. In some distributed routing network applications, each node Interface Message Processor (IMP) periodically exchanges explicit routing information with its neighbors. Each IMP maintains a routing table containing a preferred outgoing path and the corresponding estimated time or distance to the destination. One such example is the distributed Bellman-Ford (DBF) algorithm. The DBF algorithm requires very little information to be stored at the network nodes since it suffices for a node to know the length of its outgoing links and the identity of other nodes in the network. The major advantages of this algorithm are that the basic computation steps can be executed in parallel at each node, where the initial conditions can be arbitrary non-negative numbers. Such advantages facilitate repeated executions of the algorithm at each node to accommodate for any changes in the network status without the need for the algorithm re-initiation or restart. However, there appears to be various problems in the iteration of the DBF algorithm such as the bouncing problem, counting to infinity problem, and routing table looping problem.

In this paper, we will address some asynchronous scenarios of these problems. First and foremost, computational complexity of the shortest path obtained from the DBF algorithm lies in the number of messages exchanged which can grow exponentially with the number of nodes. In some cases, this may make the algorithm unusable. The delay estimate values obtained from the algorithm are transient in nature and have inherent uncertainty. All nodes in the network do not immediately know any change in the status of the network since the information available at each node at any time is out-of-date. As such, the intervals for transmitting periodic updates which range from 1 second to 10 seconds are considered long in

terms of change in traffic patterns. Moreover, a single crisp number does not carry sufficient information about the delay estimates in the network.

Next, we will focus on three algorithms, namely, DBF, Additive Approximate Distributed Bellman-Ford (AADBF) algorithm [3], and a Fuzzy Set Delay Representation for Computer Network Routing algorithm (henceforth FSDR) [4]. Both AADBF and FSDR algorithms were designed to increase the efficiency of DBF, emphasizing delay complexity reduction and delay selection, respectively. To optimize distributed routing delay approximation, a new empirical method based on features of the AADBF and FSDR is proposed. The following sections present some fundamentals of the new method.

Definitions

Given a graph G = (V, E), where V denotes the set of vertices and E denotes the set of edges of G. Define | V | = n and | E | = m. Let s be the source node of a network configuration instant and d_{ij} be the length of edge (i, j) connecting nodes i and j. Furthermore, let D_i be the shortest path length from source node to node i, parent(i) denote the identity of the previous node that sends message to node i, and N(i) denote the set of current neighbors of node i.

The Distributed Bellman-Ford (DBF) algorithm

The DBF [1, 2] algorithm determines the shortest paths from node s to every other node in the network. The algorithm first finds the shortest path lengths subject to the constraints that the paths contain at most one edge (or link). It then finds the shortest path lengths with a constraint that the paths contain at most two edges, and so on. Thus, for an n-node network, a path contains at most (n - 1) links and the algorithm converges in at most (n - 1) iterations. There are two basic rules that govern the shortest path algorithm, namely, the adopting rule and the sending rule. The adopting rule determines how to update the current label according to a message from a neighboring node. The node s calculates the shortest path to every other node in the network using the distance from itself to all its neighbors. This information is updated periodically. Thus, the estimated shortest paths D_j can be obtained from its neighbors through periodic updates. The sending rule determines what values to propagate to the neighbors except its parent(i) and is applied whenever a node adopts a new label. The updates from a particular node are sent out asynchronously with respect to the updates generated by the other nodes. In the asynchronous environment, the number of messages sent in the worst-case is proportional to the number of possible paths from s to j which could be exponential in n. The message complexity is bounded by $nm\Delta$, where Δ is the maximum length over all edges in the network.

The Additive Approximate Distributed Bellman-Ford algorithm (AADBF)

The AADBF, denoted by AADBF(s, i, β) where parameter β is greater or equal to 0, computes a path from s to i. The path length is at most $\beta n + D_i$ and the message complexity is bounded by $nm\Delta/\beta$ (see Lemma 5.1 and 5.2 in [3], respectively). The main difference between DBF and AADBF algorithms is in the adopting rule, i.e.,

- For DBF, node i adopts D'_i only if $D'_i < D_i$
- For AADBF, node i adopts D'_i only if $D'_i < (D_i - \beta)$

It is clear that when $\beta = 0$, the AADBF is the same as DBF. The AADBF algorithm may not compute the shortest path, but reduce message complexity from DBF's exponential to AADBF's polynomial without affecting the response time of the algorithm. This is the main

advantage of the AADBF algorithm. Moreover, this algorithm is straightforward and easy to implement compare with the original DBF algorithm.

A Fuzzy Set Delay Representation for Computer Network Routing algorithm

The Fuzzy Set Delay Representation (FSDR) algorithm determines the value of delay at each node on a network by resetting the minimum length, the maximum length, and the end of each period length of the queues to the current queue length. The minimum queue length, current queue length, and maximum queue length form the fuzzy number (TFN), which is used as a basis for delay representation and estimation. The entries in the delay table are TFNs, which represent the minimum delay, most possible delay, and maximum delay for every routing path. At the end of a period, a node sends an update, which carries the delay table to its neighbor nodes. When the neighbor node receives an update, it re-computes the routing paths by comparing the current delay table with the delay table received from the next node. Comparison is carried out by means of an ordering method for the fuzzy subsets of the unit interval i for computing the fuzzy weighted means of both delays. The width of the interval spanning the fuzzy numbers gives an estimate of the delay and congestion at the nodes. This information is used to compare and update the routing table. In this routing algorithm, the looping aspects of the routing algorithm have not been dealt with because the emphasis has been on the effects of the fuzzy decision methods rather than on developing a routing algorithm. The FSDR algorithm results in comparable or better delay than the DBF algorithm at lower loads and superior delay performance at higher loads. Furthermore, the use of fuzzy comparison criteria improves the delays more in the dense network than in the light network condition.

An Empirical Approach

Since DBF, AADBF, and FSDR algorithms are based on the same basis, we will apply them to a new empirical approach for finding minimum delay path, covering wider schemes of problems. The procedures focus on the asynchronous aspect of the DBF. In a fixed network topology with varying link lengths, every node in the network maintains three basic tables, namely, an adjacent node table, a route table, and a delay table shown below in Figure 1, 2, and 3, respectively.

Adjacent Node	Link Length

Figure 1. An Adjacent Node Table (of node s).

Destination	Parent Node	Destination Delay(D_i)	Next Node

Figure 2. A Route Table (of node s).

Destination	Min Delay	Current Delay	Max Delay	Next Node	Parent Node

Figure 3. A Delay Table (of node s).

The Adjacent Node Table identifies all adjacent nodes to node s. The link length (or weight) represents the distance from node s to each adjacent node. The Route Table contains the estimated minimum delay (cost) from node s to the destination node given by the Next Node column. The Delay Table holds the minimum delay, the current delay, and the maximum delay estimates of the shortest routing path from the adjacent nodes of s to an arbitrary destination via the next node within an update period. The Parent Node of s is the preceding node that sends message to s, whereby allowing the construction of the routing path hierarchy.

The empirical approach makes use of the information containing the three tables. In a typical iteration of the algorithm, node i maintains the current known minimum delay path length from source node s to itself. The minimum delay path length is the sum of the minimum delay path length from source node s to some node j adjacent to node i and the weight of the edge (i, j) or the estimates of the queueing delay. Delay measurement is monitored from the transmission queues to determine the minimum and the maximum values of the transmission queue of each path within an update period. The minimum and maximum queue lengths of an update period are weight-averaged with those of the preceding update period. Furthermore, the current queue length is weight-averaged with the preceding queue length. These values are reset to the current queue length during update, forming a new TFN for the desired path. We employed an ordering method [4] for the fuzzy subset of the unit interval i to compute the fuzzy weighted mean of current delay and new delay that is received from an adjacent node. This information is then used in the comparison criteria for selecting the minimal delay path.

Denote the current delay for the current path CD = (a1, a2, a3) and the new delay for the new path ND = (b1, b2, b3), where a1, b1 represent the minimum delays, a2, b2 represent the weighted means delays, and a3, b3 represent the maximum delays, respectively. We first compute the values of a2 and b2 from any estimated delays CD and ND along the two paths. If the difference is greater than a threshold value, the minimum of the two is chosen. The threshold value is the amount of extra delay we would tolerate along the new path so that the delay along this path would result in better overall delays. If the difference is less than or equal to the threshold value, there are four possible cases to choose as follows:

- If $a1 \leq b1$ and $a3 \leq b3$, the current delay is chosen as the optimal routing path.
- If $a1 \leq b1$ and $a3 > b3$, the new delay is chosen as the optimal routing path.
- If $a1 - b1 \leq$ threshold value and $a3 \leq b3$, the current delay is chosen as the optimal routing path.
- If $a3 - b3 \leq$ threshold value, there is an uncertainty about the most possible optimal routing path. We can choose either the current delay or the new delay.

A graphical comparison of the above four cases taken from [4] is shown in Figure 4.

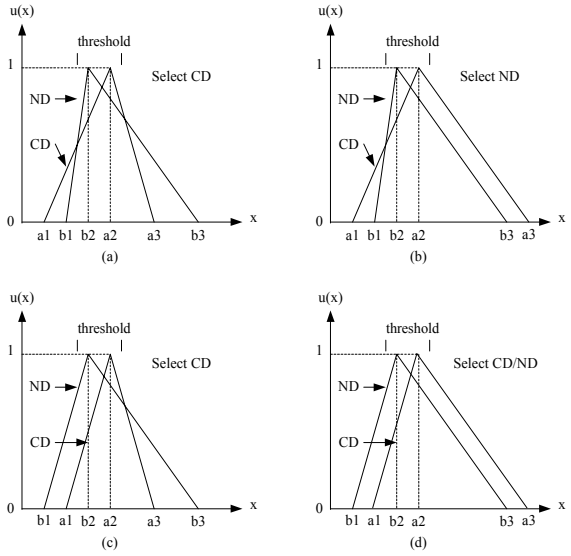

Figure 4. Comparison of Fuzzy Delay.

We will examine an adopting rule for the delay values based on these comparison criteria. Node i adopts the new delay values to its current delay values only if the new delay values are less than current delay values minus β. Otherwise node i retains the current delay values [3]. The new adopted delay values are then propagated to the neighbor nodes except its parent for subsequent update. Since the new delay values are smaller than the previous values by at least β. Thus, the number of delay values that could be sent over one link is at most $n\Delta/\beta$, which is in polynomial message complexity. This empirical approach is in the stage of analysis and simulation to determine the optimal result.

We developed a ***throwaway*** pseudo-code prototype to experiment with the empirical approach using a simple six-node network as shown in Figure 5. Note that node s and e denote the source and end node, respectively. The steps proceed as follows:

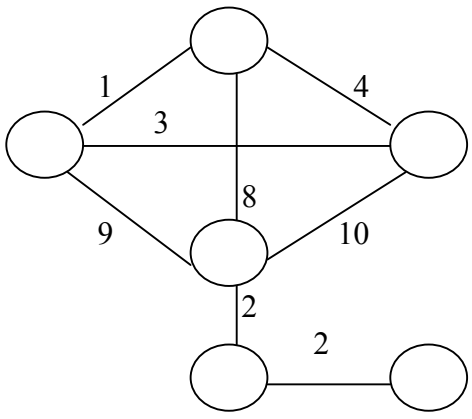

Figure 5. A six-node network.

1. ***initialize:***
   1.1. set $D_s = 0$ and $D_i = \infty$, for all $i \neq s$
   1.2. find $D_i = min_{j \in N(i)} [D_j + d_{ij}]$, for all $i \neq s$
   repeat step 1.2 until none of the delays (costs) changes

(Node s changes its routing table when it receives a Delay Table from its neighbors or the weight between node s and its neighbor changes.)

2. ***node s receives a Delay Table from its neighbors:***
   2.1. set bias to parent(i)
   2.2. compare the current delay values (CD(a1, a2, a3)) with the new delay values (ND(b1, b2, b3)) by using fuzzy ordering method
   2.3. if CD is selected, set a2 to be Current Delay(D_j)
   if ND is selected, set b2 to be Current Delay(D_j)
   2.4. set $D_i = D'_i$ and find $D'_i = min_{j \in N(i)} [D_j + d_{sj}]$, for all $i \neq s$
   if D'_i is less than $D_i - \beta$, set $D_i = D'_i$
   repeat step 2.4 for all $i \neq s$

3. ***node s sends a Delay Table to its neighbors:***
   3.1 find the minimum delay, the current delay, and the maximum delay estimates of the shortest routing path from the adjacent nodes of s to an arbitrary destination via the next node within an update period using delay measurement criteria
   3.2 send the Delay Table to the neighbor nodes except parent(s)

The results of the sample network are shown below.

Destination	Parent Node	Destination Delay(D_i)	Next node
s	-	0	-
A	s	1	A
B	s	4	B
C	s	8	C
D	C	10	C
e	D	12	C

Table 1. The initial routing table of node s.

Suppose the weight between each node of the sample network was changed within the update period as shown in Table 2. Each new value was then compared with its corresponding old value using Yager's ordering method and Fuzzy Delay comparison criteria depicted in Figure 4. The result yielded a new weight mean value between the node under consideration and its neighbor nodes. Assuming that the neighbors of node s (node A, B, and C) learned of the change before node s. Each of these nodes would then update its delay table and sent a copy to all of its neighbors, including node s. The updated routing table of node s is shown in Table 3. Since this empirical approach calculates the weight mean and updates all corresponding delay tables for every node in the network, regardless of their adjacency, the algorithm yields an exponential time complexity as shown in Table 4. Note that if only

adjacent nodes were considered, the performance (number of comparisons) would improve considerably.

Source node	Adjacent Node	Link Length (initial)	Updated Link Length (1)	Updated Link Length (2)	Updated Link Length (3)	Updated Link Length (4)
s	A	1	1	1	1	2
s	B	4	4	4	4	4
s	C	8	8	8	8	1
A	s	1	1	1	1	2
A	B	3	3	3	3	3
A	C	9	9	2	2	2
B	s	4	4	4	4	4
B	A	3	3	3	3	3
B	C	10	2	5	5	8
C	s	8	8	8	8	1
C	A	9	9	2	2	2
C	B	10	2	5	5	8
C	D	2	2	2	5	2
D	C	2	2	2	5	2
D	e	2	2	2	2	5
e	D	2	2	2	2	5

Table 2. The updated Link Length table.

Destination	Parent Node	Destination Delay(D_i)	Next node
s	-	0	-
A	A	1	A
B	B	4	B
C	C	5	A
D	C	7	A
e	D	9	A

Table 3. The final routing table of node s.

Number of nodes	Initial Iteration	Initial Comparison	Updated Iteration	Updated Comparison
1	400	65	480	15
2	800	130	960	30
3	1200	195	1440	50
4	1600	255	1920	60
5	2000	325	2400	65
6	2400	400	2880	80

Table 4. The iteration table.

Future Work

From the experiment, we found the complexity in step 2 and 3 resulted in exponential time mainly due to update process. We will first focus on how to improve the complexity from exponential time to polynomial time in accordance with the AADBF algorithm. Subsequent endeavors will emphasize on developing an approach for selecting minimal delay network routing.

Conclusion

This paper presents a new empirical method for the dynamic routing in the distributed network. The method focuses on the issue of message complexity and the uncertainty in the delay information. A simplified network configuration was devised to aid in analyzing the viability of the method. The delay estimate results were stable but yielded exponential complexity. Future work will incorporate this method to dynamically update network routing and, at the same time, arrive at polynomial complexity as part of a new distributed routing algorithm for larger networks. It is hoped that the new method will accommodate complex and widely dispersed organization of distributed computing.

References

[1] Andrew S. Tanenbaum, Computer Networks, Prentice-Hall Inc., Englewood Cliffs New Jersey, 1981.

[2] D. Bertsekeas and R. Gallager, Data Networks, Prentice-Hall Inc., Englewood Cliffs New Jersey, 1987.

[3] Baruch Awerbuch, Amotz Bar-Noy, and Madan Gopal, *Approximate Distributed Bellman-Ford Algorithm*, 1991 IEEE, pp. 1206-1213.

[4] Sridhar Pithani and Adarshpal S. Sethi, *A Fuzzy Set Delay Representation for Computer Network Routing Algorithms*, 1993 IEEE, pp. 286-293.