A Fast Distributed Shortest Path Algorithm for a Class of Hierarchically Structured Data Networks

John K. Antonio, Garng M. Huang, and Wei K. Tsai

Department of Electrical Engineering
Texas A&M University
College Station, TX 77843

Abstract

A new distributed algorithm is presented which finds the shortest paths from every node in the network to a given destination node. The network topology is assumed to be organizable into a generalized balanced-tree hierarchy (BH). The BH toplogy is introduced and characterized in this paper, and it is shown that most large interconnected data networks belong to this class. Assuming one processor is available per node in an n node BH topology, the computation and communication time complexities of the proposed algorithm are $O(\log n)$ and $O[(\log n)^2]$, respectively. (Actually, a communication complexity of $O(\log n)$ can be achieved, but due to the space limit the description of this more sophisticated communication scheme is not included in this paper.) It is also shown that the algorithm converges in an asynchronous environment. Therefore, some of the difficulties associated with synchronizing the order of events can be avoided in the actual implementation of the proposed algorithm.

I. Introduction

Solutions to shortest path problems have been the backbone of congestion control algorithms—many routing and flow control algorithms have to solve a form of the shortest path problem [4]. In this paper, we focus our attention on the shortest path problem whereby the objective is to find a path of minimum total length which joins two given nodes of a network. We assume the network is modeled as a connected and directed graph where each link is assigned a nonnegative length.

The shortest path problem has been studied extensively in the past. Dijkstra's algorithm [6] and the Bellman-Ford algorithm [7] are two classical examples. Most algorithms in the literature are variations of either the Dijkstra or Bellman-Ford algorithms [11]. The worst case computational time complexity for the serial version of Dijkstra's algorithm is known to be $O(n^2)$, for a network with n nodes. The corresponding complexity for the serial Bellman-Ford algorithm is $O(n^3)$.

The Dijkstra-type algorithms tend to be inherently serial and thus are not easily implemented as a distributed algorithm in a large data network. A parallel version of Moore's algorithm (a variant of Dijkstra's algorithm), however, was proposed and tested by Deo, Pang and Lord [5]. This algorithm was designed for a tightly coupled multiprocessor network where all of the nodes (i.e., processors) have access to shared global memory. This assumption is not valid in large data networks which span vast amounts of geographic area. Thus, a distributed version of this algorithm does not appear to be a staightforward extention.

The Bellman-Ford-type algorithms, on the other hand, can be implemented in a distributed data network. For instance, the original ARPANET routing algorithm [9] is based on a distributed version of the Bellman-Ford algorithm. For general networks, it can be shown that the distributed Bellman-Ford algorithm can find the shortest path from every node to a given destination node with computation and communication time complexities of $O(n^2)$ and $O(n)$ respectively. If the connectivity at all network nodes is $O(1)$, the computation time complexity of the distributed Bellman-Ford algorithm reduces to $O(n)$. This distributed Bellman-Ford algorithm is apparently the fastest known distributed algorithm for solving the so called single destination shortest path problem for general data networks.

One interesting fact associated with apparently all of the available shortest path algorithms is that the assumed network topology is general. A logical question then arises concerning the existence of even faster algorithms designed specifically for networks which possess certain structures in their topology. In the next section, we define a class of hierarchical topologies, called the balanced-tree hierarchical (BH) topology, for which we later develop a new single destination shortest path algorithm having computation and communication time complexities of $O(\log n)$ and $O[(\log n)^2]$, respectively. We then show that the standard single destination distributed Bellman-Ford algorithm can only achieve $O(n)$ computation and communication time complexities, even when the BH topology is assumed. That is, the BH topology assumption alone will not generically improve the time complexities associated with the standard distributed Bellman-Ford algorithm. Therefore, our new algorithm's time complexities are generically better than those associated with the standard Bellman-Ford algorithm. We should note that the BH topology is a reasonable topology for a data network to possess.

In order to illustrate the spirit of our new algorithm, consider the problem of solving the single destination shortest path problem for the simple 2-level hierarchical topology shown in Fig. 1. By blindly employing the distributed version of the Bellman-Ford algorithm, we know that the problem can be solved with a computational time complexity of cn, where c denotes the maximum connectivity at any node, and n is the total number of nodes in the network. Now, if we can assume that each cluster contains $O(\frac{n}{b})$ nodes, and no more than say g gates (i.e., the nodes connecting the cluster to the rest of the network), then our proposed algorithm operates as follows. First, shortest path sub-problems are solved from the nodes within each cluster to each of the associated g gates, in parallel at each cluster; then the solutions from all the clusters are merged

[1]This research was supported in part by the Texas Advanced Research Program under proposal no. 4659.

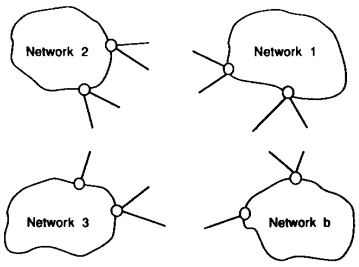

Fig. 1. The example 2-level hierarchical network showing $\frac{b}{g}$ speed-up.

together to obtain the complete solution for the entire network. The parallel solving of the cluster sub-problems requires $\frac{gc}{b}n$ time, while the merging takes $g(cb+1)$ time. If we assume that b, g and c are all independent of n (which is a reasonable assumption), then we note that both the distributed Bellman-Ford algorithm and our new algorithm have computational time complexities of $O(n)$. However, we point out that if $\frac{g}{b} < 1$ is small enough, then the actual amount of time required for our algorithm is less than that associated with the standard Bellman-Ford algorithm. The savings in computation time for this simple 2-level example is in the order of $\frac{b}{g}$ folds. This same basic idea is recursively applied to general hierarchically structured networks having more than two levels of hierarchy, yielding an $O(\log n)$ computational time complexity.

The key concept used in obtaining the complexity result of our new algorithm is that inter-cluster traffic must go through gates. If we assume the number of gates per cluster at all the levels of the hierarchical topology is $O(1)$, then our divide-and-conquer strategy capitalizes on this property to get a better overall time complexity. Also, it will be shown that even if the minimum hop distance between all pairs of nodes is $O(\log n)$ (in a BH topology), the Bellman-Ford algorithm still generically needs $O(n)$ iterations to converge. This result is of interest, since in a network having $O(\log n)$ minimum hop distances between any two nodes, one might intuitively expect that the Bellman-Ford algorithm would need only $O(\log n)$ iterations to converge. Several subclasses of the BH topologies are characterized in the next section, for which our algorithm applies. From these characterizations, the set of BH topologies appear to be quite general. We conjecture, in fact, that every connected graph with $O(1)$ connectivity can be characterized as a BH toplogy. An obvious way to prove this conjecture is to construct an algorithm to clusterize graphs as BH topologies. However, it turns out that such an algorithm is attempting to solve an an NP-complete problem (this fact can be deduced simply from a result in [17].) Nevertheless, our characterizations of the BH topology still strongly suggest the generality of the BH topology.

The remainder of the paper is organized in the following manner. In section II, we define the BH topology and justify why it is a realistic structure for data networks. In section III, we develop the proposed distributed algorithm which is designed specifically for data networks possessing the BH topology. In section IV, we show generically that the BH topology alone does not improve the time complexities associated with the standard distributed Bellman-Ford algorithm. Section V states that by making a slight modification to the new algorithm (introduced in section III), asynchronously convergence is possible. Finally, some final comments and conclusions are given in secton VI.

II. A Hierarchical Topology

In extremely large networks, the topology usually possesses a natural hierarchical structure. This hierarchical topology can be exploited to increase the efficiency of network control algorithms.

In studying network type problems/algorithms, most researchers assume a general network topology. As a result, the estimates on the complexities of their algorithms tend to be pessimistic—usually there exists a worst case topology for which the complexity bound is attained. However, the worst case topology may rarely occur in practice, therefore, more optimistic complexity bounds may be achievable for realistic networks. One straightforward way to obtain better complexities is to design algorithms around a restricted class of network topologies. We next characterize such a class, which we call the BH topology, that is based on the concept of a balanced-tree [1] hierarchy.[2]

A. *The Balanced-Tree Hierarchical (BH) Topology*

Before defining the BH topology, we first need the definition for a general balanced-tree.

Definition 2.1: A $(\underline{b}, \overline{b})$ balanced-tree is defined as a tree in which the number of children for each father is lower bounded by $\underline{b}$ and upper bounded by $\overline{b}$, where $\underline{b}$ and $\overline{b}$ are independent of n (i.e., the total number nodes in the tree) and $\underline{b} \geq 2$.

Definition 2.2: If all the nodes in a network graph can be organized into a $(\underline{b}, \overline{b})$ balanced-tree such that each leaf on the tree represents a node, each father on the tree represents a cluster which contains a group of connected children (where the children may represent either clusters or nodes) then the graph is said to have a $(\underline{b}, \overline{b})$ BH topology.

The actual nodes of the network are referred to as level 0 clusters, and the clusters whose members are nodes are referred to as level 1 clusters. In general, the clusters whose members are level j clusters are referred to as level $j+1$ clusters.

It is apparent that the $(\underline{b}, \overline{b})$ BH topology represents a large class of networks, including networks which were formed by interconnecting existing smaller networks. Even for those networks not created in this manner, there seems to be a general tendency for network designers to impose a hierarchical structure on the topology. In the following discussion, we introduce some notation and definitions to more precisely describe the $(\underline{b}, \overline{b})$ BH topology. For convenience, we will often refer to the general case of a $(\underline{b}, \overline{b})$ BH topology as simply a BH topology. Notice if a BH topology has k levels of hierarchy, then the total number of nodes in the network is bounded below by $(\underline{b})^k$ and above by $(\overline{b})^k$. We denote a BH topology having k levels of hierarchy as a k-level BH topology. Fig. 2 shows an example of a network which is characterized as a 3-level BH topology.

One way to precisely describe a k-level BH topology (and motivate the necessity for more notation) is with an inductive type construction procedure. Before stating this procedure, we have the following definition for characterizing the member nodes of a generic j-level BH topology.

Definition 2.3: Given a collection of j-level BH topologies, let N_j^i denote the node set of the i^{th} j-level BH topology.

[2]The balanced-tree was originally introduced to describe a class of hierarchical data structures, see for example [1].

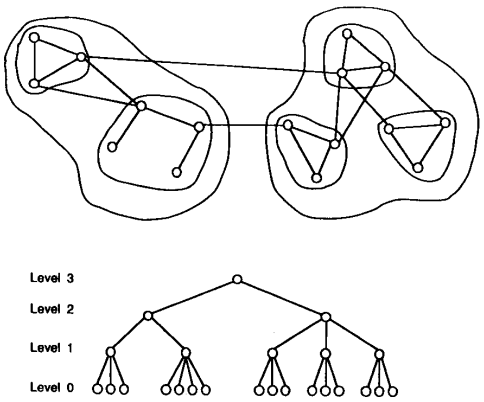

Fig. 2. The example network in (a) can be characterized as a 3-level BH topology, as shown in (b).

Construction Procedure for a k-level BH topology:

- Pick an integer b_k satisfying $\underline{b} \leq b_k \leq \bar{b}$.
- Connect b_k $(k-1)$-level BH topologies, having node sets $N_{k-1}^1, \cdots, N_{k-1}^{b_k}$ which satisfy $N_{k-1}^p \cap N_{k-1}^q = \emptyset$, for all $p \neq q$, so that the nodes in the set $N_k^1 = N_{k-1}^1 \cup \cdots \cup N_{k-1}^{b_k}$ form a connected graph.

From the above procedure, we see that a k-level BH topology is basically obtained by interconnecting b_k $(k-1)$-level BH topologies so as to form a connected graph. By recursively applying this procedure, a k-level BH topology can be constructed for any $k \geq 1$. (Recall that a single node is considered to be a 0-level BH topology, having only itself as a member.) It should be apparent that the BH topology definition describes a fairly general class of network topologies. In fact, we conjecture that nearly all large networks of practical interest can be characterized with a BH topology.

A key concept in establishing the complexity result of the new algorithm, is based on the fact that all inter-cluster traffic (i.e., paths) must pass through gates. Next, we define the notion of gates (and gate sets) for a BH topology.

Definition 2.4: For a given k-level BH topology, the i^{th} j-level gate set, denoted G_j^i, is defined as the set of all nodes contained in N_j^i which are directly connected to nodes in some other j-level node set N_j^p, where N_j^i and N_j^p belong to a common $(j+1)$-level node set, and $p \neq i$. Also, the elements of G_j^i are called j-level gates.

The next two definitions are basically counts for the number of children and grandchildren belonging to a given cluster. Recall from the construction procedure that each cluster is actually a BH topology itself.

Definition 2.5: Given a collection of j-level BH topologies, let b_j^i denote the number of $(j-1)$-level BH topologies contained in the i^{th} j-level BH topology.

Definition 2.6: Let b_j denote the total number of $(j-1)$-level BH topologies contained in a given k-level BH topology. So, by the previous definition we have

$$b_j = \sum_{i=1}^{b_{j+1}} b_j^i \quad \text{for all } j \in \{1, \cdots, k\},$$

where

$$b_{k+1} \stackrel{\triangle}{=} 1$$

and b_1 equals the total number of nodes in the network.

The next definition characterizes the set of $(j-1)$-level clusters which belong to a given parent j-level cluster.

Definition 2.7: Given a k-level BH topology, let M_j^q denote the q^{th} j-level cluster member set, which is defined to be the index set which satisfies

$$N_j^q = \cup_{i \in M_j^q} N_{j-1}^i \quad \text{for all } j \in \{1, \cdots, k\},\ q \in \{1, \cdots, b_{j+1}\}$$

where

$$N_0^i \stackrel{\triangle}{=} \{i\} \quad \text{for all } i \in \{1, \cdots, b_1\}.$$

Next, we define an upper bound for the number of gates contained within all levels of gate sets.

Definition 2.8: For a given k-level BH topology, let g denote the maximum number of elements contained in all sets G_j^i. That is, g is the integer which satisfies the following:

$$g = \max\{|G_j^i|\} \quad \text{for all } j \in \{1, \cdots, k-1\},\ i \in \{1, \cdots, b_{j+1}\},$$

where $|G_j^i|$ denotes the number of elements in G_j^i. (Note that it is necessary to have $g \geq 1$, in order for a BH topology to be connected.)

Finally, we define the intracluster gate sets, which are basically defined to be the union of all $(j-1)$-level gate sets belonging to a common j-level cluster.

Definition 2.9: Given a k-level BH topology, let H_j^q denote the q^{th} j-level intracluster gate set, defined as the set of nodes given by

$$H_j^q = \cup_{i \in M_j^q} G_{j-1}^i \quad \text{for all } j \in \{2, \cdots, k\},\ q \in \{1, \cdots, b_{j+1}\}.$$

To illustrate Definitions 2.4 through 2.9, consider the mesh[3] network clusterized as a 3-level BH topology depicted in Fig. 3. From Definition 2.4, we see that the gate sets are given by: $G_1^1 = \{3,5,7\}$, $G_1^2 = \{8,10,13\}$, $G_1^3 = \{16,17\}$, $G_1^4 = \{19,20\}$, $G_1^5 = \{30,31,32,33\}$, $G_1^6 = \{34,36,39\}$, $G_1^7 = \{42,43,44,45,49\}$, $G_2^1 = \{6,7,9,11,12,13\}$, $G_2^2 = \{14,16,18,21,22,24\}$, $G_2^3 = \{25,26,27,28,29,34,35,38\}$. By Definition 2.5, we have the following values: $b_1^1 = 7$, $b_1^2 = 6$, $b_1^3 = 4$, $b_1^4 = 7$, $b_1^5 = 9$, $b_1^6 = 8$, $b_1^7 = 8$, $b_2^1 = 2$, $b_2^2 = 2$, $b_2^3 = 3$ and $b_3^1 = 3$. Applying Definition 2.6, we have $b_1 = 49$, $b_2 = 7$, $b_3 = 3$ and $b_4 = 1$. From Definition 2.7, we get the following cluster member sets: $M_1^i = N_1^i$, for all $i \in \{1, \cdots, 7\}$, $M_2^1 = \{1,2\}$, $M_2^2 = \{3,4\}$, $M_2^3 = \{5,6,7\}$ and $M_3^1 = \{1,2,3\}$. By using the gate sets defined above, we have by Definition 2.8 that $g = 8$. Finally, Definition 2.9 yields the following intracluster gate sets: $H_2^1 = G_1^1 \cup G_1^2$, $H_2^2 = G_1^3 \cup G_1^4$, $H_2^3 = G_1^5 \cup G_1^6 \cup G_1^7$ and $H_3^1 = G_2^1 \cup G_2^2 \cup G_2^3$.

B. *Three Sub-Classes of the BH Topology*

We now state some definitions which describe three subclasses of BH topologies. The first of these definitions, Definition 2.10, is needed (in section III) primarily to insure

[3]The mesh network example is provided to show the generality of the BH topology. However, to obtain $O(\log n)$ computational time complexity, we require the number of gates per clusters at all levels to be $O(1)$, and this requirement motivates the three subclasses of BH topologies of the next sub-section. Nevertheless, in a shared memory parallel machine, an efficient parallel algorithm having a computational time complexity of $O[(\log n)^2]$ is achievable for any connected graph (using techniques similar to those introduced in this paper), which is faster than the standard parallel Bellman-Ford algorithm, see [16].

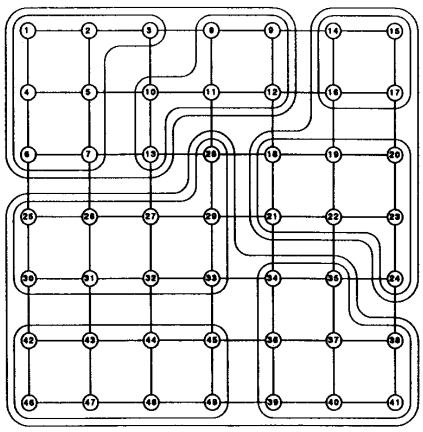

Fig. 3. A uniform mesh network characterized as a 3-level BH topology.

$O(1)$ computational time complexity at each iteration of the new shortest path algorithm.

Definition 2.10: A finite-gate k-level BH topology is a k-level BH topology in which the value of g (see Defintion 2.8) is independent of n.

Note that a mesh network cannot be classified as a finite-gate BH topology, since $g = O(\sqrt{n})$. This is easy to verify by noting that the number of gates, contained in at least one of the highest level gate sets, must be proportional to $\sqrt{n}$.

The finite-gate BH topology, alone, is useful if one is interested in a parallel implementation of our new algorithm (i.e., where no communication complexity exist.) However, it is not restrictive enough if one is iterested in achieving the $O(\log n)$-type communication complexity as claimed for the distributed implementation of our algorithm. The next definition, which describes a special type of finite-gate BH topology, is needed by the new algorithm to achieve the aforementioned communication time complexity.

Definition 2.11: A cross-level k-level BH topology is a finite-gate k-level BH topology in which the minimum hop distance between any $(j-1)$-level gate in G^i_{j-1} and any associated parent j-level gate in G^q_j is $O(1)$, for each $i \in M^q_j$, $j \in \{2, \cdots, k-1\}$, $q \in \{1, \cdots, b_{j+1}\}$.

Finally, Definition 2.12 describes a special type of cross-level BH topology (see Lemma 2.3).

Definition 2.12: A gate-connected k-level BH topology is a finite-gate k-level BH topology in which the following two conditions are satisfied:

(C1) $G^q_j \subset H^q_j$ for all $j \in \{1, \cdots, k-1\}$, $q \in \{1, \cdots, b_{j+1}\}$.

(C2) The sub-graph consisting only of nodes contained in G^i_j is connected, for each $j \in \{1, \cdots, k-1\}$, $i \in \{1, \cdots, b_{j+1}\}$.

The utility of the gate-connected BH topology definition, over the general cross-level BH topology definition, is one of notational convenience. Specifically, by assuming a gate-connected BH topology, the description of the new shortest path algorithm is less complicated (notationally) than the corresponding description for a general cross-level BH topology.

Note that the physical characteristics of both the finite-gate BH topology and the cross-level BH topology are easily interpreted by definitions 2.10 and 2.11, respectively. For example, if a given BH topology network has $O(1)$ gates in each gate set, then it is defined to be a finite-gate BH topology. Further, if a given finite-gate BH topology has the property that all $(j-1)$-level gates are "close" to their associated parent j-level gates [in the sense of $O(1)$ minimum hop distance], then it is defined to be a cross-level BH topology. On the other hand, the physical implications associated with condition (C1) of Definition 2.12, may not be quite as clear. Basically, the meaning of (C1) is that a parent j-level gate must also be a $(j-1)$-level gate. In order to clearly illustrate this concept, an example gate-connected BH topology network is given in Fig. 4. Note that the node sets for the network are included in the caption of the figure. The gate sets for Fig. 4 are given by: $G^1_1 = \{2,3,7\}$, $G^2_1 = \{8,12,13\}$, $G^3_1 = \{16,17\}$, $G^4_1 = \{18,23\}$, $G^5_1 = \{27,31,32,33\}$, $G^6_1 = \{34,35\}$, $G^7_1 = \{42,44\}$, $G^1_2 = \{7,13\}$, $G^2_2 = \{16,23\}$, $G^3_2 = \{27,34,44\}$. By Definition 2.8 we have that $g = 4$. Also, by Definition 2.9, we can compute the intracluster gate sets as: $H^1_2 = G^1_1 \cup G^2_1$, $H^2_2 = G^3_1 \cup G^4_1$, $H^3_2 = G^5_1 \cup G^6_1 \cup G^7_1$ and $H^1_3 = G^1_2 \cup G^2_2 \cup G^3_2$. Thus, with the above gate sets and intracluster gate sets, it can be confirmed that conditions (C1) and (C2) (of definition 2.12) are satisfied.

C. *Properties of the Three Sub-Classes of BH Topologies*

Out of the three sub-classes defined, the finite-gate BH topology is the most general. That is, both the cross-level BH topology and the gate-connected BH topology are special cases of the finite-gate BH topology (see definitions 2.11 and 2.12). Note also that the condition for a finite-gate BH topology (i.e., that the value of g is independent of n) is actually quite mild when considering realistic data networks. Next, we state some definitions which enable us to state an important theorem. The result of this theorem (Theorem 2.1), is that the minimum hop distance between any two nodes in a cross-level BH topology is $O(\log n)$. (Note, due to the space limit all theorems and corollaries are stated without proof.)

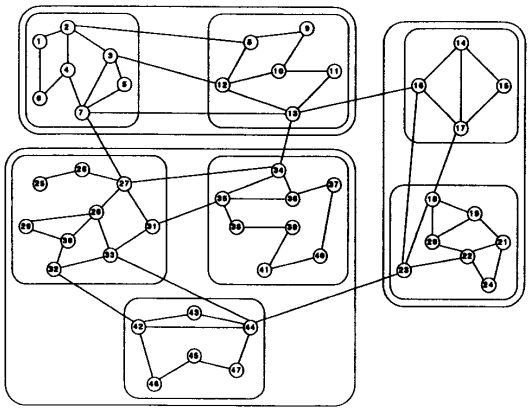

Fig. 4. An example of a gate-connected 3-level BH topology. The node sets are defined as: $N^1_1 = \{1, \cdots, 7\}$, $N^2_1 = \{8, \cdots, 13\}$, $N^3_1 = \{14, \cdots, 17\}$, $N^4_1 = \{18, \cdots, 24\}$, $N^5_1 = \{25, \cdots, 33\}$, $N^6_1 = \{34, \cdots, 41\}$, $N^7_1 = \{42, \cdots, 47\}$, $N^1_2 = N^1_1 \cup N^2_1$, $N^2_2 = N^3_1 \cup N^4_1$, $N^3_2 = N^5_1 \cup N^6_1 \cup N^7_1$, and $N^1_3 = N^1_2 \cup N^2_2 \cup N^3_2$.

Definition 2.13: Consider two nodes u and v, which belong to a common k-level BH topology. The least common level between u and v, denoted $lcl(u,v)$, is defined to be the level of the lowest level BH topology (within the common k-level BH topology) to which both u and v belong.

Definition 2.14: Let $MHD(u,v)$ be defined as the minimum hop distance between the nodes u and v.

Theorem 2.1: The MHD between any two nodes in a cross-level k-level BH topology is $O(\log n)$.

Next, we show (in Lemma 2.3) that a gate-connected BH topology is also a cross-level BH topology. Therefore, Theorem 2.1 holds for gate-connected BH topologies as well. First, we state a simple preliminary lemma.

Lemma 2.2: In a gate-connected k-level BH topology, the sub-graph consisting only of nodes within each intracluster gate set H_j^q is connected, for each $j \in \{1,\cdots,k\}$, $q \in \{1,\cdots,b_{j+1}\}$.

Lemma 2.3: If a given BH topology is a gate-connected k-level BH topology, then it is also a cross-level k-level BH topology.

Corollary 2.4: In a gate-connected k-level BH topology, the MHD between any two nodes is $O(\log n)$.

D. *Characterizing Realistic Data Networks as BH Topologies*

Now, we turn our attention toward showing that most large data networks can be characterized as having some kind of BH topology. A straightforward approach is to design an algorithm which exhaustively checks many different node set descriptions until one is found that satisfies the definition of a BH topology. However, to find such a node set description, an NP-complete problem has to be solved [17]; therefore, hueristic clustering algorithms are the viable alternative.

We should stress that a multi-level clustering algorithm does not have to be completed in order to reap the benefit of the proposed algorithm. Even if the algorithm stops with a 2-level clusterization, the speed-up can be substantial, as shown by the example provided in the introductory section.

The clustering problem can often be simplified by taking advantage of the physical characteristics of most large data networks. Quite often, the BH topology is obvious upon inspection. Most large data networks exhibit some sort of hierarchical structure, and this premise has been confirmed by other researchers. For example, in [8] a method is introduced for interconnecting existing networks, via gateways, to form larger networks. Thus, nodes belonging to a geographical area or a particular existing sub-network form natural clusters. The concept of area routing is also discussed, whereby routing within an area is managed seperately from routing between areas. (We utilize this same basic idea in the development of the proposed hierarchical shortest path algorithm presented in the next section.) Also, in [2], the use of multilevel gateways is shown to be an effective way of interconnecting similar (or dissimilar) networks. In [10], several approaches to interconnecting local networks to long-distance networks, are discussed. Overall, the network descriptions assumed or developed in [2,8,10] have obvious hierarchical structures which are accurately modeled with the BH topology.

There may exist some networks, however, in which an underlying hierarchical structure may not be altogether obvious. For example, consider Fig. 5 (a) which depicts a simple "linear" network. In what follows we show that this network can actually be considered as a finite-gate 3-level BH topology. The level 1 node and gate sets for this network are: $N_1^1 = \{1,2,3,4\}$, $N_1^2 = \{5,6,7,8\}$, $N_1^3 = \{9,10,11,12\}$, $N_1^4 = \{13,14,15,16\}$, $N_1^5 = \{17,18,19,20\}$, $N_1^6 = \{21,22,23,24\}$, $N_1^7 = \{25,26,27,28\}$, $N_1^8 = \{29,30,31,32\}$, $G_1^1 = \{4\}$, $G_1^2 = \{5\}$, $G_1^3 = \{12\}$, $G_1^4 = \{13\}$, $G_1^5 = \{20\}$, $G_1^6 = \{21\}$, $G_1^7 = \{28\}$, $G_1^8 = \{29\}$. The level 2 sets are then given by: $N_2^1 = N_1^1 \cup N_1^2$, $N_2^2 = N_1^3 \cup N_1^4$, $N_2^3 = N_1^5 \cup N_1^6$, $N_2^4 = N_1^7 \cup N_1^8$, $G_2^1 = \{8\}$, $G_2^2 = \{9,16\}$, $G_2^3 = \{17,24\}$, $G_2^4 = \{25\}$. Finally, we have $N_3^1 = \cup_{i=1}^4 N_2^i$. (Note that the above BH topology description does not qualify as a cross-level BH topology.) In Fig. 5 (b), the clusters of the linear network are circled to emphasize its hierarchical structure, as described above. In Fig. 5 (c), it is shown that by adding some links to the linear network, a gate-connected BH topology is produced. For this modified BH topology network, assume the same node sets as defined for the finite-gate 3-level BH topology of Fig. 5 (b). The new gate sets for the network of Fig. 5 (c) are as follows: $G_1^1 = \{1,4\}$, $G_1^2 = \{5,8\}$, $G_1^3 = \{9,12\}$, $G_1^4 = \{13,16\}$, $G_1^5 = \{17,20\}$, $G_1^6 = \{21,24\}$, $G_1^7 = \{25,28\}$, $G_1^8 = \{29,32\}$, $G_2^1 = \{8\}$, $G_2^2 = \{9,16\}$, $G_2^3 = \{17,24\}$, $G_2^4 = \{25\}$. By using this new gate set description, it is easy to verify that the network of Fig. 5 (c) is indeed a gate-connected 3-level BH topology. Next, notice that there are actually many valid choices for the node and gate sets of

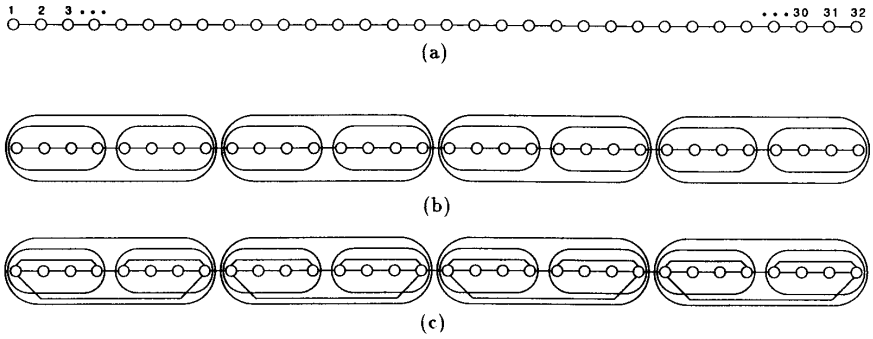

Fig. 5. (a) A 32 node "linear" network. (b) Viewing the network of (a) as a finite-gate 3-level topology. (c) By adding some links to the linear netowk of (a), a gate-connected 3-level topology is produced.

the network shown in Fig. 5 (a). For example, by using the same N_1^i's and G_1^i's as defined for Fig. 5 (b), we could consider the entire network to be only a 2-level BH topology, where we define $N_2^1 = \cup_{i=1}^{8} N_1^i$. This clearly implies that, in general, a given network may have many different possible BH topology descriptions.

III. A New Distributed Synchronous Shortest Path Algorithm for BH Topologies

We now specify the types of shortest path problems to be solved in the context of congestion control algorithms. There can be many variations of the shortest path problem depending on the update (computing) procedures of the underlying network algorithms. However, these variations only affect minor modifications in the hierarchical algorithm to be presented below. We will consider the single destination shortest path problem, which involves finding for every node in the network, the shortest distance and a shortest path to a given destination node. The solution to such a problem is typically needed in optimal routing algorithms. We note that a single origin version of our algorithm can be easily obtained from the single destination description given below.

The main idea of the proposed algorithm is twofold. First, shortest path problems are solved for a collection of small sub-graphs (i.e, clusters). There are no common nodes between these sub-graphs, thus, the entire collection of problems are solved in parallel. Second, the solution of the sub-graphs are recursively merged together until the solution for the entire network is acquired.

We should point out that the algorithm described in this section requires a certain amount of synchronicity for its operation. In particular, it is assumed that the iterations of the main step are done in a particular order, i.e., the j^{th} iteration must be complete before the $(j+1)^{th}$ iteration may begin. By knowing the bounds for communication and computation delays, an upper bound can be found for the maximum amount of time required for any iteration, thus, a straightforward synchronization scheme can be used. (We note that a uniform bound is assumed for computation and communication delays for the synchronous algorithms presented in this and the next section.)

Next, we note that a mechanism is needed for getting all the nodes to agree to start the algorithm. However, in this section we will not discuss this initialization problem, for several reasons. First, a straightforward initialization algorithm is possible due to our assumption on uniformly bounded delays. Also, the fact that the minimum hop distance between any pair of nodes is $O(\log n)$ for a cross-level BH topology, such a procedure should not increase the overall time complexity of our new algorithm. Finally, in section V we show that by making some minor modifications to the proposed synchronous algorithm, an asynchronous version is easily obtained.

A. *Network Assumptions*

(A1) The network has a gate-connected k-level BH topology, with $n \leq (\bar{b})^k$ nodes.

(A2) Each node has one processor.

(A3) Each node has a unique identification (ID) number taken from the set of integers $\{1, \cdots, n\}$. (For convenience, we assume that the given destination node is number 1.)

(A4) Each node u has available the link lengths ℓ_{uw}, for all $w \in N(u)$, which are assumed positive and constant after some initial time t_o, where $N(u)$ denotes the neighbors of node u.

(A5) Each node u has stored in its memory two $(1+gk)$-length vectors d_u and e_u. The first component of d_u, denoted d_{u1}, is the current estimate of the shortest distance from node u to the given destination node. The remaining gk components of d_u, denoted d_{uv} for the v^{th} component, are used to store estimates of the distances from node u to each of its (at most) gk logical parent gates (a physical node can be the site of multiple logical gates). The v^{th} component of e_u, denoted e_{uv}, is the ID of the next node (from node u) along the current estimate of the shortest path associated with d_{uv}.

(A6) Besides the memory required by (A5), each logical gate and each regular node has extra memory allocated to store an additional g d_u vectors.

(A7) The processing, transmission, queueing, and propagation delays, associated with sending a single $O(1)$ length control message across any link in the network, are assumed independent of n.

Assumptions (A1) through (A6) will prove useful primarily in deriving the computational time complexity component of the proposed algorithm. With the exception of (A1), these first seven assumptions are fairly standard for most distributed shortest path algorithms.

Assumption (A7) is related to the communication complexity involved with sending control messages in the network. Note that it is reasonable to assume that the processing and transmission delay (associated with sending an $O(1)$ length control message) is independent of n, since the length of each message is independent of n. Also in (A7), the assumption of the queueing delay being independent of n is justified by simply allowing the control messages to have the highest possible priority at all queues. Finally in (A7), note that the propagation delay associated with most data network links is considered negligible and therefore independent of n.

B. *Description of the New Distributed Shortest Path Algorithm*

In this sub-section, we first give a general description of the new shortest path algorithm, which has the intent of showing only the most fundamental tasks performed. Later, a detailed description is given which shows how to implement each step. Of prime importance is step (2). In particular, we will show how to "merge" shortest path information. The merging procedure is not obvious and our implementation of this task is the foundation of the entire algorithm.

In the following general description of the algorithm, note that steps (0) and (1) are executed once and step (2) is done recursively $k-1$ times.

General Description:

(0) Each component[4] of d_u is set to ∞ and each component of e_u is set to zero for all $u = 1, \cdots, n$.

(1) Each of the b_2 1-level BH topologies solves at most $g+1$ versions of the single destination shortest path

[4]Except for the first components of d_1 and e_1 which are set to 0 and 1 respectively.

problem—one version for the given destination node (if applicable) and at most g versions for each of the level 1 gates for each 1-level BH topology (i.e., cluster). So after this step, each node has an estimate (possibly non-infinite) of the shortest distance to the given destination node and an estimate of the shortest distance to each associated level 1 gate.

DO $j = 1,\ k - 1$.

(2) Parallelly merge shortest path information of the b_{j+1} j-level BH topologies into shortest path information for b_{j+2} $(j+1)$-level BH topologies.

END DO

Next, we give the detailed description of each step in the algorithm, along with the time complexities associated with these steps. Time complexities are measured by using two seperate time units—one for computation time complexity, the other for communication time complexity. In the detailed description, step (2) is divided into four "sub-steps." Furthermore, some of these sub-steps rely on a standard distributed version of the Bellman-Ford algorithm to solve a shortest path problem for small (i.e., $O(1)$) clusters or groups of nodes. As mentioned earlier, it can be shown that the for a general network (or sub-network) with say N nodes, a distributed version of the Bellman-Ford algorithm can solve the single destination shortest path problem with computation and communication time complexities of $O(N^2)$ and $O(N)$ respectively. These time complexity results are used in estimating the time complexities for the applicable sub-steps. Due to the space limit, the proof of correctness is not included in this paper.

Detailed Description (with complexity computation):

(0) Initialize the vectors d_u and e_u. This can be done in $1 + gk \leq 1 + g\log_{\underline{b}} n$ computation time units.

(1) At this point, the network is divided into b_2 1-level BH topologies (or clusters) containing at most $\bar{b}$ nodes each. Every cluster solves the single destination shortest path problem for the given destination node (if the destination node happens to belong to their cluster). Next, each cluster solves the single destination shortest path problem for each gate in G_1^i, for all $i \in \{1, \cdots, b_2\}$. These tasks are accomplished using the standard distributed Bellman-Ford algorithm. Since each node is a member of only one cluster, the b_2 clusters can solve their shortest path problems in parallel. Since each cluster contains at most $\bar{b}$ nodes, this step will require at most $\bar{b}^2(g+1)$ computation time units and $\bar{b}(g+1)$ communication time units.

DO $j = 1,\ k - 1$.

(2.1) All pairs of gates $g_a, g_b \in G_j^i$, replace the value of $\ell_{g_a g_b}$ with the current estimate of the shortest distance between g_a and g_b, for each $i \in \{1, \cdots, b_{j+1}\}$. (Note: For those pairs of gates in G_j^i which are not directly connected, a *virtual* link length is stored.) Thus, each gate must update at most g values of $\ell_{g_a g_b}$, which requires g computation time units.

(2.2) The gates in each intracluster gate set, denoted H_{j+1}^q, are considered as a connected sub-graph, for each $q \in \{1, \cdots, b_{j+2}\}$. The single destination problem is solved from every gate in H_{j+1}^q to the given destination node, using a "modified" distributed Bellman-Ford algorithm.[5] The current estimates from each gate to the destination node (from the previous iteration) are used as a virtual link length to the given destination node. Next, the single destination problem is solved for each parent $(j+1)$-level gate in H_{j+1}^q, again using a modified distributed Bellman-Ford algorithm. Note that there are b_{j+2} of these sub-graphs and since they are disjoint, the computation is done in parallel. Since there are at most $\bar{b}g$ nodes in each H_{j+1}^q, the shortest path computation for the given destination node and all parent $(j+1)$-level gates requires at most $(g+1)(\bar{b}g)^2$ computation time units and $(g+1)\bar{b}g$ communication time units. So, at this point we have the following situation: (i) Every regular node has an estimate of the shortest distance to the given destination node and to each associated j-level gate. (ii) Every gate in the set H_{j+1}^q has a new estimate of the shortest path distance to the given destination node and to every other parent $(j+1)$-level gate belonging to the same intracluster gate set, for all $q \in \{1, \cdots, b_{j+2}\}$.

(2.3) Every gate in G_j^i transmits its new estimates (of the distance from itself to the given destination node and the distance from itself to each $(j+1)$-level parent gate), to every other node belonging to the same j-level BH topology, for all $i \in \{1, \cdots, b_{j+1}\}$. This broadcast can be done in parallel for each i, with at most $2\bar{b}g\log_{\underline{b}} n$ hops (see Theorem 2.1 and Lemma 2.3). Thus, the broadcast of at most $g+1$ such messages requires no more than $2\bar{b}g(g+1)\log_{\underline{b}} n$ hops. Since transmitting $O(1)$ length messages across any link in the network requires $O(1)$ communication time units, this step will require no more than $2\bar{b}g(g+1)\log_{\underline{b}} n$ communication time units. Note that these messages are stored in the extra memory allotted to each regular node described by (A6).

(2.4) Finally, every node (within each $(j+1)$-level BH topology) can compute a new estimate of the shortest path distance to the given destination node, and a new estimate of the shortest path distance to each parent $(j+1)$-level gate, by adding its distance to each of its j-level gates to the estimates from each of these gates to the destination node (i.e., the given destination node or one of the $(j+1)$-level gates). Since there are at most g gates per j-level BH topology, this requires at most $g(g+1)$ additions and $(g+1)^2$ comparisons or $(g+1)(2g+1)$ computation time units. Note that the one extra comparison comes from the fact that a comparison must also be made with the current estimate stored in the distance vector at each origin node.

END DO

[5] By modified distributed Bellman-Ford algorithm, we mean the following. The gates in each j-level gate set (i.e., G_j^i) act as if they are directly connected to every other gate in G_j^i, by using the virtual link lengths determined in sub-step (2.1). Clearly, this modified Bellman-Ford algorithm requires some extra communication complexity in order to simulate the gates in each G_j^i as being completely connected. Due to the fact that each pair of gates in G_j^i are at most g (i.e., $O(1)$) hops apart, it can be shown that the communication complexity associated with the modified Bellman-Ford is of the same order as the standard Bellman-Ford algorithm. Thus, we will use the standard Bellman-Ford time complexities.

Since step (2) is done $k-1 \leq (\log_{\underline{b}} n) - 1$ times, the overall computation and communication time complexities, denoted $T_{\text{comp}}(n)$ and $T_{\text{comm}}(n)$ respectively, are given by

$$T_{\text{comp}}(n) = [2g + (g+1)(\bar{b}^2 + \bar{b}^2 g^2 + 2g + 1)][\log_{\underline{b}} n] \quad (3.1)$$

and

$$T_{\text{comm}}(n) = [(g+1)(\bar{b} + \bar{b}g + 2\bar{b}\log_{\underline{b}} n)][\log_{\underline{b}} n]. \quad (3.2)$$

Therefore, since we assume that $\bar{b}$ and g are independent of n, we have

$$T_{\text{comp}}(n) = O(\log n) \quad (3.3)$$

and

$$T_{\text{comm}}(n) = O[(\log n)^2]. \quad (3.4)$$

C. *Operation of the Algorithm Under More Relaxed Assumptions*

Recall from network assumption (A1) that the network topology is assumed to be that of a gate-connected BH topology. It should be noted, however, that by making some modifications to the above algorithm description, a similar distributed algorithm can be developed for the more general class of cross-level BH topologies. The main reason for assuming a gate-connected BH topology is that the associated algorithm description is less complicated than the general cross-level BH topology version. For example, note that in a general cross-level BH topology, the gates contained in the intracluster gate sets do not necessarily form a connected sub-graph. Thus, accomplishing sub-step (2.2) becomes more complicated.

In addition to a new distributed version of the algorithm, as described in the previous paragraph, a parallel algorithm is also possible which could solve the single destination shortest path problem for general finite-gate BH topologies. Such an algorithm could, for example, be used at a centralized parallel computing site having n processors which operate in a shared memory environment. By assuming shared memory between all processors, the communication complexity associated with sending control messages and the initialization procedure are eliminated. Thus, a centralized version of the algorithm would have an overall time complexity roughly the same as the computational time complexity associated with the distributed algorithm.

Before concluding this section, we first briefly discuss a connectivity requirement which is imposed by the conditions used in the previous section to define a gate-connected BH topology. (Note: the connectivity of a given node is defined to be the number of links connected to that node.) Specifically, because of conditions (C1) and (C2) of Definition 2.12, it can be shown that at least one of the $(k-1)$-level gates, of a k-level BH topology, has $O(\log n)$ connectivity. This is easy to verify, by noting from (C1) that every j-level gate must also be a $(j-1)$-level gate. Then by (C2), since each j-level gate must form a connected graph with other j-level gates, which belong to a common $(j+1)$-level BH topology, then generically a j-level gate has connectivity of $O(j)$. Thus, there exist at least one $(k-1)$-level gate which has connectivity of $O(\log n)$. If we allow the more relaxed definition of a cross-level BH topology, however, this $O(\log n)$ connectivity property can be eliminated. This is important since it is typically assumed that the connectivity at every node in a data network is $O(1)$.

As an example of how to relax a gate-connected BH topology to a cross-level BH topology so as to eliminate the $O(\log n)$ connectivity property, we present the following connection scheme for the gates: Let each j-level gate be directly connected to one of its member $(j-1)$-level gates; as opposed to requiring one of the member $(j-1)$-level gates to also play the role of a parent j-level gate. If this procedure is done for all j, then the characteristic that the $(k-1)$-level gates have $O(\log n)$ connectivity is removed. Fig. 6 depicts this type of connection scheme. For clarity, detailed connections are shown within only one of the clusters. Note that the MHD between any two nodes is still $O(\log n)$, since the MHD between $(j-1)$-level gates and their parent j-level gates is still $O(1)$, as required by Theorem 2.1. The network representation of Fig. 6 should be interpreted conceptually and not as an actual description of a physical network. That is, the level 2 gates actually belong to some physical level 1 cluster. Likewise, the level 3 gate belongs to one of the four level 2 clusters.

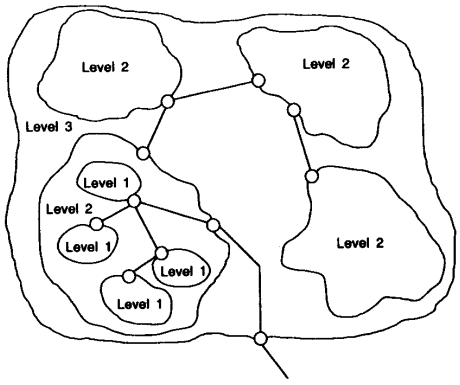

Fig. 6. An example of a connection scheme for cross-level BH topologies, which eliminates the $O(\log n)$ connectivity property associated with gate-connected BH topologies.

IV. Time Complexity of the Distributed Synchronous Bellman-Ford Algorithm Assuming a BH Topology

In this section, we show that the time complexities associated with the standard distributed Bellman-Ford algorithm do not improve by assuming the network topology is a BH topology.

As in the previous section, we will denote the length of link (u,w) as ℓ_{uw}. The current estimate of the shortest distance from node u to the given destination node is stored at node u as d_{u1}. The Bellman-Ford algorithm is then given by:

(0) $d_{u1}^{(0)} = \infty, \quad u \neq 1,$

$d_{11}^{(0)} = 0.$

(1) $d_{u1}^{(h+1)} = \min_{w \in N(u)} [\ell_{uw} + d_{w1}^{(h)}], \quad u \neq 1,$

$d_{11}^{(h+1)} = 0,$

where $N(u)$ denotes the set of current neighbors of node u, and h is the iteration count.

The time complexities for the distributed Bellman-Ford algorithm assuming a general network topology[6] with $O(1)$ connectivity at every node, can be simply derived as:

$$T_{\text{comp}}(n) = O(n) \tag{4.1}$$

and

$$T_{\text{comm}}(n) = O(n). \tag{4.2}$$

Next, we will show that the computation and communication time complexities for the distributed Bellman-Ford algorithm remain $O(n)$, even when the underlying network topology is a BH topology. This is shown by constructing a class of example gate-connected BH topologies in which a shortest path contains $n-1$ links. Such a network is shown in Fig. 7, which depicts a simple 2-level BH topology. The links are assumed bi-directional and their lengths are labeled on the figure. It is easy to verify that the shortest

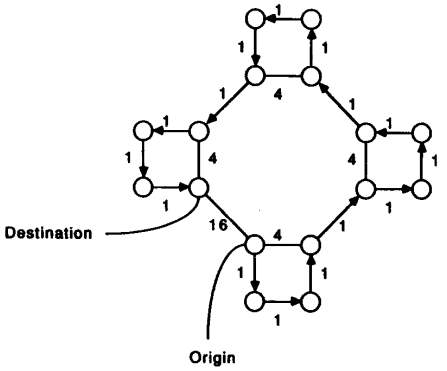

Fig. 7. A 2-level BH topology with 16 nodes and 15 links in a shortest path.

path (between the nodes labeled origin and destination) is the path denoted by the "arrowed" links. Note that there are 16 nodes in this 2-level BH topology network and 15 $(= n-1)$ links in the aforementioned shortest path. It is easy to see that 4 copies of this 2-level BH topology can be connected together to form a 64 node 3-level BH topology network, as shown in Fig. 8, with a shortest path containing 63 $(= n-1)$ links. Inductively, this procedure can be repeated to produce similar such k-level BH topologies, where k can be made arbitrarily large. So, we have a construction procedure which produces a class of BH topologies for which the distributed Bellman-Ford algorithm has $O(n)$ computation and communication time complexities. Obviously, there exist a multitude of such examples where a shortest path in a BH topology contains $O(n)$ links. The specific topologies presented here are included to verify the existence of a class of such examples.

V. Distributed Asynchronous Operation of Shortest Path Algorithms

It can be shown that our new algorithm can actually operate in an asynchronous manner, however, due to the space

[6]We assume that network assumptions (A2) through (A7) of the previous section are satisfied for a general network containing n nodes [note that assumption (A6) does not apply when assuming the case of a general network topology].

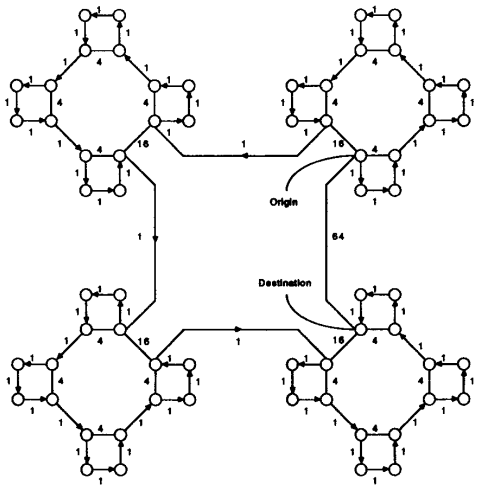

Fig. 8. A 3-level BH topology with 64 nodes and 63 links in a shortest path.

limit a detatiled proof is not included in this paper. Note that by asynchronous, we mean that the iterations required do not have to be executed in any particular order, even when there exists no uniform bound for computing and communication delays. We note that the algorithm of section III does require some slight modifications in order to achieve asynchronous convergence.

In [3,4], an asynchronous distributed computation model is introduced. This model has very weak assumptions on the ordering of computations, the timing of information exchange, the amount of information needed at each node, and the initial conditions for the algorithm. The distributed Bellman-Ford shortest path algorithm is given as an example algorithm which converges in this computation model. We should mention that the convergence of distributed asynchronous Bellman-Ford algorithm has been known and discussed by McQuillan, et. al. [9] and Schwartz [12], and the rigorous proof of convergence was provided by Tajibnapis [13] and Bertsekas [3,4]. By using similar techniques, it can be shown that by making some slight modifications on our new algorithm, it too converges asynchronously. The essential element involved in proving this type of convergence is centered around the fact that our new algorithm consists of basically solving an ordered series of dynamic programming problems.

VI. Summary and Conclusions

In conclusion, we have demonstrated that a properly chosen (natural as well as artificial) organization of the node sets for a graph can significantly improve the time complexity of solving the shortest path problem in either a distributed or parallel environment. The key concept has been that inter-cluster shortest paths must contain gates (nodes through which a cluster is connected to other clusters). If the number of gates of each cluster is independent of n at all levels of the hierarchy, then the properly chosen hierarchy (which we call the BH topology) can be exploited to obtain an $O(\log n)$-type time complexity. We have also shown a somewhat surprising result that even if the minimum hop distance between all pairs of nodes in a

BH topology is $O(\log n)$, the Bellman-Ford algorithm still generically requires $O(n)$ iterations to converge. The proposed algorithm is also shown to converge in a distributed asynchronous environment.

The clustering problem, i.e., the problem of organizing the node set of a graph into a $O(\log n)$-level BH topology, turns out to be an NP-complete problem [17]. However, we have characterized several sub-classes of the general BH topologies and pointed out that for many large wire data networks, the hierarchical structure is obvious by inspection or is not difficult to obtain if one is satisfied with only a few levels of hierarchy. We have also provided a simple example showing that even if two levels are involved, the speed-up of our algorithm over the existing algorithm can still be very significant.

As a final note, we also mention two closely related works. A hierarchical routing protocol and hierarchical topology update and maintenance protocols have been designed and shown to perform efficiently in [17]. Following this work, our algorithm can be readily adapted into efficient protocols, to be used in a realistic data network control scheme. The hierarchical concept has also been adapted by us [14,15] to improve the convergence rate of optimal routing algorithms. Since most optimal routing algorithms (including the one in [14,15]) require shortest path solutions at each iteration, our proposed algorithm is readily applicable.

References

[1] A.V. Aho, J.E. Hoptcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.

[2] E. Benhamou and J. Estrin, "Multilevel Internetworking Gateways: Architecture and Applications," *IEEE Computer Magazine*, Vol. 16, No. 9, September 1983, pp. 27–34 .

[3] D. Bertsekas, "Distributed Dynamic Programming," *IEEE Trans. on Auto. Control*, Vol. AC-26, 1982, pp. 610–616.

[4] D. Bertsekas, "Distributed Asynchronous Computation of Fixed Points," *Math Prog.*, Vol. 27, 1983, pp. 107–120.

[5] N. Deo, C. Pang, and R.E. Lord, "Two Parallel Algorithms for Shortest Path Problems," *Proceedings of the 1980 International Conference on Parallel Processing* (August), *IEEE*, New York, pp. 244–253.

[6] E. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, Vol. 1, 1959, pp. 269–271.

[7] L.R. Ford, Jr. and D.R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[8] R. Hinden, J. Haverty, and A. Sheltzer, "The DARPA Internet: Interconnecting Heterogeneous Computer Networks with Gateways," *IEEE Computer Magazine*, Vol. 16, No. 9, September 1983, pp. 38–48.

[9] J.M. McQuillan, G. Falk and I. Richer, "A Review of the Development and Performance of the ARPANET Routing Algorithm," *IEEE Trans. on Comm.*, Vol. COM-26, No. 12, Dec. 1978, pp. 1802–1811.

[10] N.F. Schneidewind, "Interconnecting Local Networks to Long-Distance Networks," *IEEE Computer Magazine*, Vol. 16, No. 9, September 1983, pp. 15–24.

[11] M. Schwartz and T.E. Stern, "Routing Techniques Used in Computer Communication Networks," *IEEE Trans. on Comm.*, Vol. COM-28, No. 4, April 1980, pp. 539–552.

[12] M. Schwartz, "Analysis of Congestion Control Techniques in Computer Routing and Flow Control in Data Networks," *NATO Advanced Study Inst.: New Concepts in Multi-User Communications*, Norwich, U.K., Aug. 4–16, 1980; Sijthoof and Nordhoof, Neth.

[13] W.D. Tajibnapis, "A Correctness Proof of a Topology Information Maintenance Protocol for Distributed Computer Networks," *Comm. ACM*, Vol. 10, July 1977, pp. 477–485.

[14] W.K. Tsai, G.M. Huang, J.K. Antonio, and W.T. Tsai, "Distributed Iterative Aggregation Algorithms for Box-Constrained Minimization Problems and Optimal Routing in Data Networks," in the January 1989 issue of *IEEE Trans. on Auto. Control.*

[15] W.K. Tsai, G.M. Huang, J.K. Antonio, and W.T. Tsai, "Distributed Aggregation/Disaggregation Algorithms for Optimal Routing in Data Networks," *Proceedings of the Automatic Control Conference*, June 1988, Atlanta, GA.

[16] W.K. Tsai, J.K. Antonio, and G.M. Huang, "Fast Parallel Hierarchical Aggregation and Disaggregation Algorithms for Multistage Optimization Problems and the Shortest Path Problems," under preparation.

[17] W.T. Tsai, "Control and Management of Large and Dynamic Networks," Ph.D. Thesis, Department of EECS, UC Berkeley, 1985.