LOCATION OF CENTRAL NODES IN TIME VARYING COMPUTER NETWORKS

Ariel Orda

Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa, Israel 32000

Raphael Rom[†]

Sun Microsystems Inc.
Mountain View, CA 94043

March 1989
Revised April 1990

ABSTRACT

Some computer and communication networks require the assignment of certain nodes to perform special, network wide functions such as routing, monitoring etc. The natural question that arises is which is the best network node to perform the function. This is the essence of the location problem.

Research in network location problems concentrated on static environments, whereas many problems, in particular those related to computer networks, are dynamic in nature. In this work we consider the single facility dynamic-network location problem in a continuous domain of time.

We first formulate the problem, make some general observations, and then solve it for a discrete time domain. Having become acquainted with the specificities of the problem we then offer a solution to the continuous time domain. Finally, a distributed version of the algorithm is outlined and discussed.

Keywords: Networks, location, time-dependency.

[†] On leave from the Technion - Israel Institute of Technology. This research was supported in part by the Foundation for Research in Electronics, Computers and Communications, Administered by the Israel Academy of Science and Humanities

I. INTRODUCTION

The control and operation of some computer and communications networks sometimes requires the existence of a central node which assumes "leadership" for execution of special network tasks, e.g., by having specific hardware or processing capabilities. For example, choosing one node in a computer network to serve as a "routing center" that offers routing services to its subscribers [1] or that performs network monitoring functions[2]. The network designer is therefore faced with a location decision namely where to locate the "leader" from among a set of alternative locations and, if the function can change its location, when and how should this be done. Henceforth, we shall use the term "facility" to refer to the central node. Consider for example a computer network in which all nodes report periodically the condition of their outgoing links to some routing center which then disiminates a digest of this information to the individual nodes of the network (as currently done in the ARPANET[3]). Obviously, it is preferable to locate the routing center within a node which is as close as possible to all others. However, because of time-varying load conditions, each potential node may be overloaded at different times of the day or may not be `sufficiently close' to all other nodes all the time. In other words, each potential node is "close" to the rest at certain times but "far" at others. Thus, we may prefer to change the identity of the routing center based on time. This change consists of transmitting the network state between the old and new centers (which, for a network with m links, is an $O(m)$ table); in many circumstances the improvement due to the new location of the center exceeds the penalty of having to relocate it.

The above are instances of a class of problems known as network location problems that were the subject of extensive research during the past few years[4,5,6,7]. Apart from computer networks, these problems are of interest in the fields of transportation, city planning, and transmission.

A location problem can be described as the selection of one or more locations from among a set of possible choices, in order to satisfy requirements imposed by a set of users or "customers". To solve a location problem one must define a *performance criterion* that depends on the "transportation" costs between users and facilities such as travel distances or response time. Once defined the problem is then to choose the locations that optimize the performance criterion. Two typical examples of the location problem are finding the (single) center i.e., a point in the network whose maximal distance to any customer is minimal (the "minimax" criterion) or, finding the "median", i.e., a location whose average distance to all customers is minimal (the "minisum" criterion)[4].

Many location problems are presented as graph problems in which the vertices of the graph represent the users while the arcs (and vertices) are the points of potential facility location. When referring to nodes and arcs we actually refer to the location problem as presented on a graph. Noteworthy is the fact that the optimal location of a facility does not necessarily coincide with the network nodes; in such cases the facility is said to be located on an arc and the distance to the nearest node is the fraction of the total arc length.

One of the first and most fundamental results in network location theory was that of Hakimi[8] who showed that under some simple assumptions there is always at least one node which is a median. Hakimi's result was expanded to a large variety of location problems (see [4]). In fact, a class of problems having a solution in the network nodes are referred to as having the Hakimi property.

Research in network location problems concentrated mainly on static environments, i.e., fixed topology and fixed distances between nodes. Dynamic location problems were hardly treated and are clearly of interest in many networks--computer networks in particular. In these networks topology and distances are subject to constant change with time. Analysis of dynamic networks is facilitated since quite often the way a network changes with time may be predicted or observed or recorded (see [9] for a discussion of some examples related to computer networks).

Of the few works that investigated dynamic location problems most considered a discrete time domain. Wesolowsky's work[10] is one such example. There, the problem is to locate a facility at each "time period" so as to minimize the sum of the costs (during all periods) associated with the distance between the customers and the facility as well as the costs incurred by relocating the facility. In Wesolowsky and Truscott[11] a related problem is treated with more than a single facility and locations restricted to nodes of a network. These two works use integer programming and dynamic programming techniques to reach the solution. Berman and Odoni [12] treated a problem close to that of [11] but assumed network distances to be Markovian variables and tried to minimize the expected cost. It is proved that Hakimi's result (see above) is valid in this case too. However, as observed by the authors, their method of solution becomes overly complex for large networks or for a large time domain. In fact, their problem is NP-hard (this can be proved using a reduction from the clique problem). They do, however, offer an optimal solution for the limited case of a single facility and a tree shaped network.

In this work we consider a single-facility dynamic-network location problem with a *continuous* (rather than discrete) domain of time. We assume the network state changes constantly and thus at each moment a different point may be the best choice for facility location. We take into account the cost of switching the facility from one node to another since this switching consumes resources (although switching does not necessarily involve physical relocation, the cost results from the need to relocate the function). Moreover, we assume these costs to be time dependent. We point out that we deal with a deterministic environment, and thus all functions are deterministic. Introducing time dependency into network parameters was previously treated in several works[9,13]. Results from these works are used in this paper to define the model as well as the problem.

The paper is organized as follows. In Section II we define and formulate the problem. Section III presents an algorithm for a discrete time domain followed by Section IV where an algorithm for the general problem is presented, validated, and some special cases discussed. A distributed version is discussed in Section V.

II. PROBLEM FORMULATION

We consider a directed network $G(V,E,L)$ with $V=\{1,2,...n\}$ being the set of nodes, $E \subseteq V \times V$ the set of links, and $L=\{l_{ik}(t) \mid (i,k) \in E\}$ a set of time dependent link "lengths". Each function $l_{ik}(t)$ is a strictly positive function of time defined for $[0,\infty)$ that describes some transportation cost on link (i,k) at time *t*. For example, the cost might be the delay of a message leaving node *i* on link (i,k) at time *t*. Networks of this type were previously studied[9] from which it is known that the time-distance between any two nodes can be efficiently found and we denote by $\tilde{l}_{ik}(t)$ the time distance between two nodes $i,k \in V$.

As stated in the introduction there are several plausible criteria by which optimal locations can be defined. These criteria define the relation between the objective function and the link costs. Since we deal with a time varying environment the performance measure itself will be a function of time denoted by $C_i(t)$. In other words, $C_i(t)$ is the cost of having the facility located at node *i* at time *t* (we do assume the function $C_i(t)$ to be integrable). For example, the minisum (or median) strategy is represented by a cost function $C_i(t) = \sum_{k \in V} \tilde{l}_{ik}(t)$ which must be minimized; the "minimax" strategy measuring the longest distance from the center to any other node is represented by the cost function $C_i(t) = \max_{k \in V} \{\tilde{l}_{ik}(t)\}$. Also, since in general we do not require $\tilde{l}_{ik}(t) \neq \tilde{l}_{ki}(t)$ we can define directed strategies such as the "in-minisum", "out-minisum", a combination of the two, and so on.

To gain insight into the problem consider the simple three-node example depicted in Figure 1 where the arc lengths are denoted along the arcs and are considered to be the delay functions of traversing the arcs. Denote $x(t)$ the fractional distance from node 2 of a point on link $(2,1)$ at time *t*, that is, the delay of the point to node 2 is $l_{12} \cdot x(t)$ and the delay to node 1 is $l_{21} \cdot [1-x(t)]$. Then, it can be easily verified that the location of the median (i.e., optimal location according to the minisum criterion) is given by

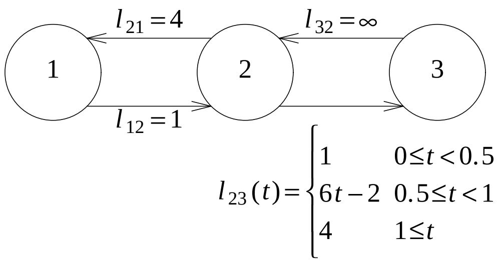

Figure 1: Example of minisum location with continuous functions

$$x(t) = \begin{cases} 0.5-t & 0 \le t < 0.5 \\ 0 & 0.5 \le t < 0.666 \\ 1 & 0.666 \le t \end{cases}$$

Two interesting observations result. First, we note that at times the facility *must* be located on the link, meaning that Hakimi's property does not hold. The second is that although all functions involved are continuous the location function $x(t)$ is not, meaning that the median "jumps" (in our example at $t=0.666$). In the context of computer networks these observations are not of particular interest since locating facilities on arcs is meaningless. They are of interest in the general context of network problems. In the context of computer networks it is noteworthy to point out that even with continuous functions it is not guaranteed that the optimal location jumps between neighboring nodes.

Having defined the cost functions $C_i(t)$, we can now measure the total weight of having the facility located at a node *i* during some interval $[t,\tau]$. This is given by

$$W_i(t,\tau) \stackrel{\Delta}{=} \int_t^\tau C_i(y)\,dy$$

Switching the location, as we mentioned, consumes resources. Hence the switching must come at a cost which must depend on the following:

- The identity of the current and the new location.
- The duration of time available for making the switch (making a faster switch is costlier); the available time is measured from the moment the first node starts to "host" the facility until the new node takes over.
- The time in which the switch takes place. In a time dependent system there may be times in which the switching is costlier than in others.

Thus, for each pair of nodes $i,k \in V$ we define a switching cost function $S_{ik}(t,\tau)$ to account for switching the facility from node *i* to node *k* ($k \ne i$) when node *i* hosts the facility during the period $[t,\tau)$. The following assumptions are made regarding these functions:

1. Switching always comes at a cost thus it must be that $\forall\, i,k \ne i, t, \tau > t \quad S_{ik}(t,\tau) > 0$.
2. For each t_0, $S_{ik}(t_0,\tau)$ is non increasing with respect to τ and for each τ_0, $S_{ik}(t,\tau_0)$ is non decreasing with respect to *t*.
3. Since switching the facility must take time we require that $\lim_{\tau - t \to 0} S_{ik}(t,\tau) = \infty$.
4. For all practical purposes we may assume that $S_{ik}(t,\tau)$ is continuous or piecewise continuous in *t* and τ. Also, in order to guarantee the existence of an optimal solution to the problem (see

below) we require that for each τ_0 and for all t $S_{ik}(t,\tau_0) \le \min\{S_{ik}(t^-,\tau_0), S_{ik}(t^+,\tau_0)\}$ and for each t_0 and for all τ $S_{ik}(t_0,\tau) \le \min\{S_{ik}(t_0,\tau^-), S_{ik}(t_0,\tau^+)\}$.

To formulate the problem we need two more definitions:

Definition 1: A *Location Sequence* (*ls*) for an interval $[0,t]$ is a finite sequence of ordered pairs $ls = \left[(v_0,t_0),(v_1,t_1),\ldots,(v_m,t_m)\right]$ such that $v_0,v_1,\ldots,v_m \in V$ is a sequence of facility locations and $0 = t_0 < t_1 < \cdots < t_m < t$ are the moments of facility relocation. We say that this sequence is of *length* $m+1$. The set of all location sequences for the interval $[0,t]$ is denoted $LS(t)$.

Definition 2: The *Location Sequence Cost* ($C(ls,t)$), namely the cost of locating the facility during the interval $[0,t]$ as dictated by the location sequence *ls* is given by

$$C(ls,t) \triangleq \sum_{i=0}^{m-1}\left[W_{v_i}(t_i,t_{i+1}) + S_{v_i,v_{i+1}}(t_i,t_{i+1})\right] + W_{v_m}(t_m,t)$$

With the above definitions the Time Varying Location (TVL) Problem is defined as follows:

Problem TVL: For an interval $[0,T]$ find a location sequence $ls^* \in LS(T)$ such that

$$\forall\, ls \in LS(T) \quad C(ls^*,T) \le C(ls,T) \quad .$$

Using standard methods, it can be verified that the fourth property of the functions $S_{ik}(t,\tau)$ and the continuity of the functions $W_i(t,\tau)$ guarantee the existence of a solution to problem TVL.

The third property of the functions $S_{ik}(t,\tau)$ rules out an infinite optimal location sequence, and therefore the solution of problem TVL will give us an optimal location policy.

III. A DISCRETE DOMAIN PROBLEM

In order to gain insight into the continuous case of TVL problems and their solutions we present first a solution to a discrete version of the problem. Let the time axis be divided into slots, or periods, that are numbered sequentially. The facility can be switched between nodes only at the beginning of a slot. We consider a time interval $[0,T]$ where T is an integer and define for each slot m ($m=0,1,\ldots,T-1$) the following variables, similar to those defined for TVL:

- The weight of node i at slot m: $\overline{W}_i(m) \stackrel{\Delta}{=} W_i(m,m+1)$.
- The cost of switching the facility from node i to node k at the end slot m: $\overline{S}_{ik}(m) \stackrel{\Delta}{=} S_{ik}(m,m+1)$ for $i \neq k$ and $\overline{S}_{ii}(m) \equiv 0$.
- A discrete location sequence ($\overline{ls}$) which is a finite sequence $\overline{ls}=(v_0,v_1,\ldots,v_{T-1})$ where $v_m \in V$ is the location of the facility during slot m. The set of all discrete location sequences is denoted $\overline{LS}$.
- The cost of a discrete location sequence given by

$$\overline{C}(\overline{ls}) \stackrel{\Delta}{=} \sum_{m=0}^{T-2} \left[\overline{W}_{v_m}(m) + \overline{S}_{v_m,v_{m+1}}(m) \right] + \overline{W}_{v_{T-1}}(T-1)$$

With the above the discrete time varying location (DTVL) problem can be stated as follows:

Problem DTVL: For an interval $[0,T]$ find a discrete location sequence $\overline{ls}^* \in \overline{LS}$ such that

$$\forall\, \overline{ls} \in \overline{LS} \quad \overline{C}(\overline{ls}^*) \leq \overline{C}(\overline{ls}) \quad .$$

Our solution is based on constructing a graph $\hat{G}(\hat{V},\hat{E},\hat{L})$ derived from $G(V,E,L)$ which we do in two steps. First we derive the graph $\overline{G}(\overline{V},\overline{E},\overline{L})$ in the following manner:

- The set of nodes $\overline{V}$ is derived by having each node of V split into T nodes, that is

  $$\overline{V} = \{v_0,v_1,\ldots,v_{T-1} \mid \forall\, v \in V\} \quad .$$

  Note that there is a parenthood relation $P:\overline{V} \rightarrow V$ that associates with each node in $\overline{V}$ one node in V. We refer to node $v_i \in \overline{V}$ as being the *level i* descendent of node $P(v_i)$.

- The set of links is defined as

  $$\overline{E} = \{(v_i,u_{i+1}) \mid v_i,u_{i+1} \in \overline{V} \quad 0 \leq i \leq T-2\}$$

  that is, every node of level i is connected with every node of level $i+1$, even if they correspond to the same parent. In other words, in the definition above it is possible that $P(v_i)=P(u_{i+1})$ (this will correspond to the facility being hosted in a node for more than a single slot).

- The set of link costs $\overline{L} = \{ \overline{l}(\overline{e}) | \overline{e} \in \overline{E} \}$ where the cost of the individual link is given by

$$\overline{l}(\ (v_i, u_{i+1})\ ) \stackrel{\Delta}{=} \overline{W}_{P(v_i)}(i) + \overline{S}_{P(v_i),P(u_{i+1})}(i)$$

Next we derive $\hat{G}$ from $\overline{G}$ by introducing two additional nodes referred to as "source" (s) and "destination" (d) and several links. We add zero-cost links between node s and every node of $\overline{V}$ of level 0. Also, we add a link between every node of $\overline{V}$ of level $T-1$ to node d. The cost of a link (v_{T-1}, d) is given by $\overline{l}((v_{T-1}, d)) \stackrel{\Delta}{=} \overline{W}_{P(v_{T-1})}(T-1)$.

There is a one-to-one correspondence between the set of paths in $\hat{G}$ between nodes s and d and the set of discrete location sequences $\overline{LS}$. The sequence of nodes correspond to the nodes along the path and the sequence of times correspond to the level of the nodes. The cost of a sequence, $\overline{C}(\overline{ls})$, is clearly the sum of the costs of the links along the path. Thus, solving the DTVL problem on graph G is equivalent to solving a regular shortest path problem on graph $\hat{G}$ which can be done in any of the many available algorithms[14]. A typical and simple algorithm will require $O(|V|^2 T^2)$ operations.

It is interesting to compare this result with those of Berman and Odoni[12] since in both cases a discrete time varying environment is investigated. In our case the variations in time are assumed known whereas in[12] only the invariant switching probabilities are known. This difference in models leads to the ability to find a polynomial solution (in both $|V|$and $|T|$) for our problem compared to the inability of doing so in the other case.

IV. THE ALGORITHM

The discussion in the previous chapter suggests that problem TVL may be solved by shortest path algorithms. Since we deal with a time varying environment, the shortest path algorithm should be one of a class described in [9]. In this section we first specify an algorithm and then prove that it solves problem TVL. For brevity we divide the algorithm into two parts: the first and main one, called Algorithm Cost, computes the cost of the optimal location sequence while the second (and much simpler one), called Algorithm Sequence, calculates the sequence itself. Some special cases are discussed at the end.

A. Specification

Algorithm Cost gets as input an instance of the TVL problem namely, the interval $[0,T]$ and the two sets of functions $W_i(t,\tau)$ and $S_{ik}(t,\tau)$ all defined for that interval. It computes two sets of functions $X_i(t)$ and $Y_{ik}(t)$ from which the cost of the optimal location sequence is computed and from which the optimal location sequence can be derived by Algorithm Sequence.

Notably different from typical algorithms, these algorithms operate on functions i.e., assignment, addition, or any other operation is performed on entire functions. In particular, we need to be able to detect functions that change during the execution of an algorithm step. We use the term "just changed" to mean that there exists at least one time instance t for which a function changed its value during the most recent operation performed on that function.

The algorithms make use of two internal functions of time defined for $[0,T]$ as follows:

- For each node $i \in V$, $X_i(t)$ is the best known cost (at that stage of algorithm execution) of a location sequence in the interval $[0,t]$ in which i is the last node.
- For each pair of nodes $i,k \in V$, $i \neq k$, $Y_{ik}(t)$ is the best known cost (at that stage of algorithm execution) of a location sequence in the interval $[0,t]$ in which the last switch is from node i to k and takes place at time t.

We shall prove that at the end of execution the functions $X_i(t)$ and $Y_{ik}(t)$ are the best respective cost functions, from which the optimal location sequence can be derived.

Algorithm Cost

1. $\forall\, i \in V \;\; X_i(t) \leftarrow W_i(0,t)$
2. $\forall\, i,k \in V,\;\; k \neq i$

   $$Y_{ik}(t) \leftarrow \min_{0 \le \tau < t} \{ X_i(\tau) + W_i(\tau,t) + S_{ik}(\tau,t) \}$$

3. $\forall\, k \in V\, X_k(t) \leftarrow \min_{i \in V, i \neq k, \tau < t} \{ Y_{ik}(\tau) + W_k(\tau, t) \}$

4. If for some $i \in V\, X_i(t)$ just changed then go to step 2, else stop.

Some explanation of the computation is called for. In step 1 we assume that there will be no switch and node *i* will host the facility for the entire period. In step 2 we consider a switch from node *i* to *k* to take place at time *t*. Assuming $X_i(\tau)$ to represent the optimal cost (known so far) of a location sequence in the interval $[0, \tau]$ with i the last node we have that the cost of making a switch at time $t > \tau$ is $X_i(\tau) + W_i(\tau, t) + S_{ik}(\tau, t)$ incorporating the additional cost of staying at node *i* during the interval $[\tau, t]$ and the cost of switching in that interval. We then choose the optimal τ to yield the proper value of $Y_{ik}(t)$. In step 3 we use the new values of $Y_{ik}(t)$ to update the values of $X_k(t)$.

In analogy to the solution of the discrete problem, the algorithm presented above resembles a shortest path algorithm in a time varying environment between fictitious source and destination nodes. Paths and sequences correspond to one another as in the discrete algorithm.

Having computed the functions $X_i(t)$ and $Y_{ik}(t)$ for all i and *k*, Algorithm Sequence calculates the sequence itself. In the description we use the operator "&" to mean concatenation, i.e., prepending a component to a sequence.

Algorithm Sequence

1. $k \leftarrow \min\{ j \,|\, \forall\, i \in V\ X_j(T) \leq X_i(T) \}$, $t \leftarrow T$, $ls \leftarrow \emptyset$.
2. If $X_k(t) = W_k(0, t)$ then
   a. $ls \leftarrow (k, 0)\ \&\ ls$.
   b. Stop.
3.
   a. Find a node *l* and a time $\tau < t$ such that $X_k(t) = Y_{lk}(\tau) + W_k(\tau, t)$.
   b. $ls \leftarrow (k, \tau)\ \&\ ls$.
   c. $k \leftarrow l$, $t \leftarrow \tau$.
   d. Go to step 2.

The algorithm builds the location sequence backwards starting with an empty list. We know the identity of an optimal location at time *T*: this is a node for which $X_j(T)$ is minimal. Given that at time *t* the facility is located at node *k* we look, in step 3, for the time τ and node *l* in which the most recent node switch ocurred (step 3.a). Having identified these we prepend the component (k, τ) to the location sequence. Step 2 assures termination and the addition of the first component to the sequence.

B. Validation

Algorithm Cost generates as its output the functions $X_i(t)$ and $Y_{ik}(t)$ yet we are interested in location sequences; hence it is necessary to define a correspondence between location sequences and the algorithm's output. As a first step we establish a correspondence between the function $X_i(t)$ and the cost of a location sequence.

From the structure of Algorithm Cost it is clear that at any stage of execution for each $i \in V$ and time $\hat{t} \in [0,T]$ there is at least one location sequence $ls = [(v_0,t_0),(v_1,t_1),\ldots,(v_m = i, t_m \le \hat{t})]$, whose cost with respect to interval $[0,\hat{t}]$ equals $X_i(\hat{t})$. In other words there is a location sequence the last node of which is node *i*, the last switching time takes place not after $\hat{t}$, and the cost of the sequence is exactly $X_i(\hat{t})$.

Suppose that after Algorithm Cost stops node *j* is such that $\forall i \in V\ X_j(T) \le X_i(T)$, i.e., $X_j(T)$ is minimal. The optimal location sequence ls^* is the location sequence that corresponds to $X_j(T)$. We prove this claim via a main theorem that we precede by auxiliary lemmas.

Observing the execution of Algorithm Cost we note that it executes step 1 once and then steps 2 through 4 repeatedly. We thus define every passage through step 2 as an "iteration" of the algorithm, and the execution of step 1 (the initialization) as the zeroth iteration.

Lemma 1: For each node $j \in V$ and time $\hat{t} \in [0,T]$, after the *N*-th iteration, $X_j(\hat{t})$ equals the cost of an optimal location sequence for the interval $[0,\hat{t}]$ among all location sequences of length at most $N+1$ whose last node is *j*.

Proof: We note that for any given value of *t* the value of the functions $X_i(t)$ and $Y_{ik}(t)$ (for all i,k) cannot increase from one iteration to the next. We proceed to prove the lemma by induction on *N*. The claim is clearly true for $N = 0$. Assuming truth for $N-1$ we prove for *N*. Let $ls_N = [(v_0,t_0),(v_1,t_1),\ldots,(v_m = j, t_m)]$ ($m \le N$) be an optimal location sequence for interval $[0,\hat{t}]$ among all location sequences of length no more than $N+1$ whose last node is *j*. Denote $ls_{N-1} = [(v_0,t_0),(v_1,t_1),\ldots,(v_{m-1},t_{m-1})]$. By the inductive assumption after $N-1$ iterations $X_{v_{m-1}}(t_{m-1}) \le C(ls_{N-1},t_{m-1})$. Thus after performing step 2 at the *N*th iteration we have

$$Y_{v_{m-1},v_m}(t_m) \le X_{v_{m-1}}(t_{m-1}) + W_{v_{m-1}}(t_{m-1},t_m) + S_{v_{m-1},v_m}(t_{m-1},t_m) \le$$

$$\le C(ls_{N-1},t_{m-1}) + W_{v_{m-1}}(t_{m-1},t_m) + S_{v_{m-1},v_m}(t_{m-1},t_m) = C(ls_{N-1},t_m) + S_{v_{m-1},v_m}(t_{m-1},t_m)$$

In the above inequality the first transition stems from the fact that *Y* is chosen as the minimum in step 2; the second transition incorporates the inductive assumption; the last transition is merely a time extension (without switching) at the last node. After performing step 3 at that iteration we have

$$X_{v_m}(\hat{t}) \le Y_{v_{m-1},v_m}(t_m) + W_{v_m}(t_m,\hat{t}) \le C(ls_{N-1},t_m) + S_{v_{m-1},v_m}(t_{m-1},t_m) + W_{v_m}(t_m,\hat{t}) = C(ls_N,\hat{t}) \quad .$$

In this inequality the first transition is due to the minimality of *X* (step 3), the second transition incorporates the previous inequality, and the last transition is by the definition of $C(ls_N, \hat{t})$. Since ls_N is optimal we have $X_{v_m}(\hat{t}) = C(ls_N, \hat{t})$. □

Lemma 2: Algorithm Cost stops after a finite number of iterations; when this happens there is a node *j* for which $X_j(T) = C(ls^*, T)$ where ls^* is an optimal location sequence (for $[0, T]$) whose last node is *j*.

Proof: Let $LS_i(t)$ denote the set of location sequences for interval $[0, t]$ each having *i* as its last node. As was previously explained, an optimal location sequence for any finite interval is finite. Thus, there is a finite *N* such that $\forall i \in V$, $\hat{t} \in [0, T]$, there is an $\hat{ls} \in LS_i(\hat{t})$ such that $\forall ls \in LS_i(\hat{t})$ $C(\hat{ls}, \hat{t}) \leq C(ls, \hat{t})$ and the length of $\hat{ls}$ is at most *N*. It follows from the algorithm and from Lemma 1 that after *N* iterations $X_i(t)$ stops changing for all $i \in V$. It also follows from Lemma 1 that if *j* is the last node in an optimal location sequence ls^* (for $[0, T]$) then when the algorithm stops $X_j(T) = C(ls^*, T)$. □

Lemma 3: Algorithm Sequence stops after a finite number of steps and produces (in the variable *ls*) an optimal location sequence.

Proof: As a result of Lemmas 1 and 2 the value of $X_i(T)$ equals the cost of an optimal location sequence for the interval $[0, T]$ with *i* the last node of the sequence. Algorithm Sequence chooses (step 1) that node for which $X_j(T)$ is minimal. To prove the lemma we need therefore to prove that for a given *j* Algorithm Sequence computes, in a finite number of steps, a location sequence whose cost is $X_j(T)$.

Denote each passage through step two as an "iteration", and assume that at a certain iteration $k = k_0$ and $t = t'$. Then, either the location sequence $[(k_0, 0)]$ corresponds to $X_{k_0}(t')$, in which case step 2a is performed and the algorithm stops in step 2b, or there is a location sequence $ls' = [(v_0, t_0), \ldots, (v_{m-1}, t_{m-1}), (k_0, t_m)]$ such that ls' corresponds to $X_{k_0}(t')$ and step 3 is executed. In this case we find v_{m-1} (called *l* in this step of the algorithm) and t_m (called τ in this step) and make the appropriate setting. We are now left with a new value $X_{v_{m-1}}(t_m)$ which corresponds to the location sequence $[(v_0, t_0), \ldots, (v_{m-1}, t_{m-1})]$. The above arguments indicate that the algorithm does not deadlock, that is, either step 2a or 3 are performed in each iteration and that a location sequence corresponding to $X_j(T)$ is constructed. Since (from Lemma 1) such a sequence is finite and since each iteration increases the length of the location sequence by unity, it follows that the algorithm is finite. □

The properties of the algorithm proven in the above lemmas can be therefore summarized in a single theorem whose proof follows directly from the lemmas:

Theorem: A solution to problem TVL can be found within a finite number of steps by executing Algorithm Cost followed by Algorithm Sequence. □

The above claims and theorem show that the algorithm has a "universality" property: after completing the computation for an interval $[0, T]$ we can identify an optimal location sequence for any subinterval $[0, t]$, $t \leq T$ by running Algorithm Sequence with *t* replacing *T*. However, the solution itself is not universal, i.e., an optimal location sequence for $[0,t]$ is not necessarily a prefix to any optimal one for a larger time interval.

C. Special cases

The solution presented above to problem TVL can be easily modified to deal with some common special cases, as follows:

- *Transition between neighbors only*. In many cases it makes sense to move the facility only between neighbors in the graph. By setting $S_{ik}(t,\tau)=\infty$ for $(i,k)\notin E$ the algorithm will exclude the undesired transitions.
- *Restricted locations*. Assume that the location of the facility is restricted to a few nodes. Then, setting $W_i(t,\tau)=\infty$ for all nodes *i* which may not host the facility will ensure that the algorithm will not dictate locating there the facility.
- *Bounded switching time*. It may be useful to restrict the rate of switching which can be done by ensuring that the facility stays at a node for a minimum time period θ ($0<\theta\leq T$). Setting $S_{ik}(t,\tau)=\infty$ for each $i,k\in V$ and all $\tau<t+\theta$ will enforce this requirement.
- *Limited number of switches*. It may also be desirable to restrict the number of switches during the period in question. This can be had by running Algorithm Cost for at most N iterations. Lemma 1 guarantees that at that stage we have an optimal sequence among those limited to length $N+1$.

V. A DISTRIBUTED PROTOCOL

Since we deal with computer communication networks it is of interest to consider a distributed execution of the algorithm. The structure of the algorithm resembles the shortest path algorithms presented in [15] that lend themselves easily to distributed computation. In the following we outline one possible distributed implementation of the protocol.

We assume that each node $i \in V$ recognizes the values of T, $W_i(t,\tau)$, and $S_{ik}(t,\tau)$ (for all $k \in V$) in the interval $[0,T]$. The protocol has to find an optimal location sequence for an interval $[0,T]$ and it is assumed that it begins well before $t=0$ so that it terminates in time for the results to be used. The protocol operates in phases on a predefined spanning tree rooted at some node *s*. The function of the root, beyond the regular computation related to the algorithm, is to determine the beginning and end of phases; consequently, any node can be selected as root. It is assumed that every node recognizes its father and sons on the spanning tree. Finally, we assume the existence of an underlying routing mechanism with no particular restriction except that it ensures message delivery in a bounded time.

The protocol begins by having node *s* receive an external impetus. It then calculates the values of $Y_{sk}(t)$ for all $k \in V$ and sends each such function in a Y-message to node *k*. Every node *i* upon receiving a Y-message from its father, considers it as an indication that a new phase has begun. Node *i* will then perform a calculation similar to that done by the root namely, for each $k \in V$ except its father on the tree it calculates the value of $Y_{ik}(t)$ and sends a Y-message containing this function to node *k*.

Every node then waits until it receives a Y-message from all other nodes in the network. The leaves of the tree will clearly be the ones to get it first. Node *i* Having received the Y-message from all other nodes in the network sends a Y-message to its father. When node *s* receives Y-messages from all its sons it knows that the previous iteration ended and a new iteration may begin.

The protocol has to identify the end of execution of the algorithm. This is done by sending "no-change" signals down the tree; this is implemented by setting a special bit in the Y-message that node *i* sends its father. When in a given iteration node *i* receives the "no-change" signal from all its sons, and it notes that its $X_i(t)$ did not change in the current iteration it sends a "no-change" signal to its father (in the Y-message it is about to send). When node *s* receives a no-change signal from all its sons the protocol terminates i.e., *s* would not start a new iteration.

The remaining task is to identify an optimal location sequence. This is done in accordance with the procedure to derive the location sequence from the functions $X_i(t)$ and $Y_{ik}(t)$. First, with a single search along the tree the node with the minimal value of $X_i(T)$ is identified. This will be the last node of the location sequence. Then the backtracking procedure begins whereby every node on the location sequence identifies its corresponding interval and notifies the previous one, until the entire sequence is constructed.

The operation of this protocol is based on the operation of the PIF protocol of [15] with the Y-messages serving as both the control and data carrying message, and on the distribution concepts presented in [13]. Because of this similarity we omit here the formal specification and validation.

The protocol presented above computes the location sequence for one finite interval $[0,T]$. To be applicable to a computer network it must be extended so that the entire time domain is considered. Our approach is to divide the time axis into successive intervals of duration T and perform the computation for each interval separately. The value of T is assumed to be large enough so that computation ends in time.

Consider the nth interval $[nT,(n+1)T]$ during which the following is performed: (1) a location protocol as described above is run for the $n+1$st interval; (2) a second, data-setting protocol is run for the $n+2$nd interval[13]. This data-setting protocol initializes at each node i the values of $W_i(t,\tau)$ and $S_{ik}(t,\tau)$ for that interval (recall that these functions depend, through the performance measure, on the link functions $l_{ik}(t)$ assumed known at the individual nodes). Thus, at the termination of the data setting protocol every node has sufficient data to start the location protocol for the next interval.

The choice of T is not quite arbitrary. On one hand, as indicated, T is assumed large enough so that these protocol finish in time. On the other hand, too large a T is inconvenient since it hampers the compact representation of functions in messages. it should be noted though, that if T happens to be somewhat too small and one of the protocols does not finish in time, the partial output obtained so far from the algorithm can be used as an approximation to the exact values required.

One last issue is the seam between consecutive intervals. As mentioned in the previous section an optimal location sequence for the nth interval is not necessarily the prefix of the one in the next interval. Thus, the location protocol for consecutive intervals identifies an optimal solution for the $n+1$st under the constraint that the first node should be the last one chosen for the nth interval. We remark here that such a solution is not necessarily optimal but that the difference should not be considerable if network state does not change too fast. The nature of the problem does not permit computing an optimal solution for all intervals.

REFERENCES

1. M. Schwartz and T.E. Stern, ‘‘Routing Techniques used in Computer Communication Networks,’’ *IEEE Trans. on Communications* **COM-28**(4) pp. 539-555 (April 1980).

2. B.M. Leiner, D.L. Nielson, and F.A. Tobagi, ‘‘Issues in Packet Radio Network Design,’’ *Proceedings of the IEEE* **75**(1) pp. 6-20 (January 1987).

3. J.M. McQuillan, I. Richer, and E.C. Rosen, ‘‘The New Routing Algorithm for the ARPANET,’’ *IEEE Trans. on Communications* **COM-28**(5) pp. 711-719 (May 1980).

4. G.Y. Handler and P.B. Mirchandani, *Location on Networks, Thoery and Algorithms,* MIT Press, Cambridge (1979).

5. R.L. Francis and J.A. White, *Facility Layout and Locations: An Analytical Approach,* Prentice Hall, Englewood Cliffs, New Jersey (1974).

6. B.C. Tansel, R.L. Francis, and T.J. Lowe, ‘‘Location on Networks: A Survey. Part I: The p-Center and p-Median Problems,’’ *Management Science* **29**(4) pp. 482-497 (April 1983).

7. B.C. Tansel, R.L. Francis, and T.J. Lowe, ‘‘Location on Networks: A Survey. Part II: Exploiting Tree Networks Structure,’’ *Management Science* **29**(4) pp. 498-511 (April 1983).

8. S.L. Hakimi, ‘‘Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph,’’ *Operations Research* **12** pp. 450-459 (1964).

9. A. Orda and R. Rom, ‘‘Shortest-Path and Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length,’’ *Journal of the ACM* **37**(3) pp. 607-625 (July 1990).

10. G.O. Wesolowsky, ‘‘Dynamic Facility Location,’’ *Management Science* **19**(11) pp. 1241-1248 (1973).

11. G.O. Wesolowsky and W.G. Truscott, ‘‘The Multiperiod Location-Allocation Problem with Relocation of Facilities,’’ *Management Science* **22**(1) pp. 57-65 (1975).

12. O. Berman and A.R. Odoni, ‘‘Locating Mobile Servers on a Network with Markovian Properties,’’ *Networks* **12**(1) pp. 73-86 (Spring 1982).

13. A. Orda and R. Rom, ‘‘Distributed Shortest-Path Protocols for Time-Dependent Networks,’’ pp. 439-445 in *Proceedings of ICCC 88,* Tel Aviv, Israel (November 1988).

14. N. Deo and C.Y. Pang, ''Shortest Path Algorithms: Taxonomy and Annotation,'' *Networks* **14** pp. 275-323 (1984).

15. A. Segall, ''Distributed Network Protocols,'' *IEEE Trans. on Information Theory*, pp. 23-34 (January 1983).