SHORTEST-PATH AND MINIMUM-DELAY ALGORITHMS
in
NETWORKS WITH TIME-DEPENDENT EDGE-LENGTH

Ariel Orda
Raphael Rom

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

August 1989

ABSTRACT

We consider in this paper the shortest-path problem in networks in which the delay (or weight) of the edges changes with time according to arbitrary functions. We present algorithms for finding the shortest-path and minimum-delay under various waiting constraints and investigate the properties of the derived path. We show that if departure time from the source node is unrestricted then a shortest path can be found that is simple and achieves a delay as short as the most unrestricted path. In the case of restricted transit, it is shown that there exist cases where the minimum delay is finite but the path that achieves it is infinite.

This research was supported by the Foundation for Research in Electronics, Computers and Communications, Administered by the Israel Academy of Science and Humanities

I. INTRODUCTION

Shortest paths algorithms have been the subject of extensive research for many years resulting in a large number of algorithms for various conditions and constraints[1]. The vast majority of these deal with fixed graphs i.e., fixed topology and fixed link weights.

The advancement of computer networks and distributed processing has brought renewed interest in the subject with a new twist: time dependency. Several works have been published dealing with topological changes in which links may occasionally become unavailable (i.e., infinite weight) and others deal with quasi static models, that is, link weights that change from time to time but remain constant in between these (infrequent) changes[2,3].

Time dependent shortest path problems have been studied for the case of discrete delay functions whose domain and range are the positive integers. Such problems were addressed both directly [4,5] and indirectly in the context of maximal flow[6,7].

In this paper we address the shortest path problem without these restrictions, i.e., we allow arbitrary functions for link delays. In this respect this is the broadest generalization. Such a problem was briefly treated by Dreyfus[8] and Ling et.al.[9] which address only limited cases. The most direct treatment to date was done by Halpern[10] where arbitrary waiting times are also considered. In this latter work an algorithm is proposed for various waiting constraints, but this algorithm cannot be bounded by network topology (i.e., the number of operations cannot be bounded by a function of the number of nodes or edges) nor are the properties of the resulting path investigated (e.g., whether it is a simple path). All the above works avoid the treatment of functions by addressing the problem for a single instance of time and not for time ranges. In this paper we present algorithms for finding the shortest-path and minimum-delay for all instances of time and under various waiting constraints and investigate properties of the derived path. We show that if messages can be arbitrarily delayed at the source node then a shortest path can be found that is simple and achieves a delay as short as the most unrestricted path, without having to wait enroute.

Our interpretation of time dependency of links is that of message traversal. For example, one interpretation might be the delay incurred by a message traversing the links. We note that time dependency may not be a continuous function. Consider a dial-up link between two nodes which is established and disestablished periodically. A message arriving while the link is established will suffer a relatively short delay while a message arriving immediately after the link is disestablished will suffer a much greater delay.

We also do not restrict ourselves to FIFO links only, since in some potential cases the FIFO assumption is invalid. For example, consider a link composed of two physical communication channels one being faster than the other. If the policy of link management is to send a message over the first available channel, then a message sent over the slower one may arrive later than another message, sent later on the faster channel, meaning that messages arrive in a non-FIFO order.

Our interest in the problem stems from related problems in computer communication networks; hence our reference to "messages." Nonetheless, the results reported here hold for a general graph. In particular, some transportation networks may serve as good examples. One such example (following [9]) is a traveler standing on a platform in a railway station wondering whether to take the local train stopping in front of him or to wait for the express train to his destination. Here again, we have time dependency of delays, with possible non-FIFO behavior.

This paper is structured as follows. After presenting a formal model in section II, we present shortest-path algorithms for various types of node behavior and several classes of delay functions. Section IV concludes these results and points at some further research problems to extend this work.

II. MODEL

We consider a bi-directed network $G(V,E,D)$, with $V = \{1,2,...,n\}$ being the set of nodes, $E \subseteq V \times V$ the set of links (with $(i,k) \in E$ implying $(k,i) \in E$), and $D = \{d_{ik}(t) | (i,k) \in E\}$ a set of time-dependent link delays i.e., $d_{ik}(t)$ is a strictly positive function of time defined for $[0,\infty)$ which describes the delay of a message over link (i,k) at time t. (Several results presented in this paper are valid only for a smaller class of functions, e.g., continuous or piecewise continuous; we shall indicate these restrictions when the need arises).

Several interpretations of the link delay functions are possible resulting in somewhat different models. In one model, referred to as the *frozen link* model, the delay of a message is fixed at the time a message starts traversing it. In another, referred to as the *elastic link* model $d_{ik}(t)$ is the instantaneous link length at time t. According to this latter model messages that start traversing the link from i to k at time t_0 will arrive at k at the first instance of time $t_1 > t_0$ for which $t_1 - t_0 \geq d_{ik}(t_1)$.

As a matter of fact the elastic link model is a submodel of the frozen link one. Consider the following transformation from a function $d_{ik}^{(e)}(t)$ to $d_{ik}(t)$:

$$d_{ik}(t) = \min_{\tau \geq 0} \{ \tau | \tau \geq d_{ik}^{(e)}(t+\tau) \}$$

then, $d_{ik}^{(e)}(t)$ interpreted as the delay function according to the elastic link and $d_{ik}(t)$ interpreted as the delay function according to the frozen link model will result in exactly the same network behavior. Note that in the elastic link we always have FIFO behavior, while in the frozen link model non-FIFO behavior is possible. In the rest of the paper we therefore focus our attention on the more general frozen link model.

Because of the possible non-FIFO characteristics of the $d_{ik}(t)$ it may sometimes be preferable to wait a certain amount of time at the sending node before embarking on the link traversal. Such waiting consumes buffer space and may not be permitted by all nodes. We therefore consider three different network traversal policies:

- *Unrestricted waiting* (UW) in which unlimited waiting is allowed everywhere along the message path through the network.
- *Forbidden waiting* (FW) in which waiting is disallowed everywhere along the message path through the network.
- *Source waiting* (SW) in which waiting is disallowed everywhere along the message path through the network except at the source node which permits unlimited waiting.

Note that waiting time is a means of flow control[11]. The UW model implies a hop-by-hop flow control, the SW an end-to-end flow control mechanism, and the FW rules out flow control.

We now define the terminology used in analyzing the above models. Consider a message that arrives at node i at time t, waits for a period of τ and then departs on link (i,k) at time $t+\tau$. This message arrives at node k at time $t+[\tau+d_{ik}(t+\tau)]$. We define $D_{ik}(t,\tau) \stackrel{\Delta}{=} \tau+d_{ik}(t+\tau)$ which is the combined waiting time and link delay for traversing link (i,k). To minimize traversal time we are interested in finding , for link (i,k) and time t, an optimal waiting time, i.e., a waiting time $\tau^* \geq 0$ such that for any other waiting time $\tau \geq 0$, $D_{ik}(t,\tau^*) \leq D_{ik}(t,\tau)$. Note that in some cases such a value may not exist. Consider, for example, a link (i,k) for which

$$d_{ik}(t) = \begin{cases} 100 & t \leq 10 \\ 1 & t > 10 \end{cases}$$

Then, for $t=0$, $\inf_{\tau \geq 0} D_{ik}(0,\tau) = 11$ but this value cannot be achieved for any $\tau \geq 0$. This point was overlooked in previous works[8,9,10]. In the following we shall implicitly assume that delay functions are such that for all (i,k) and every time instant t a proper optimal waiting time τ^* does exist. This property clearly holds for continuous functions while for piecewise continuous functions a sufficient condition for this property to hold is that for all i,k,t: $d_{ik}(t) \leq \min\{d_{ik}(t^+), d_{ik}(t^-)\}$.

Denote by N_k the set of node k's neighbors; then, a *topological path* through the network is a sequence of nodes $(v_0, \ldots, v_m)$ such that a link exists between every pair (v_i, v_{i+1}) i.e., $v_{i+1} \in N_{v_i}$. These definitions of N_k and of a topological path are only topology dependent and not time dependent. A simple topological path is one in which no node appears more than once.

A *waiting schedule* is an m-tuplet of waiting times $\boldsymbol{\tau} = (\tau_0, \tau_1, \ldots, \tau_{m-1})$ with $\tau_i \in [0,\infty)$ signifying the duration of waiting at node v_i. A *traversal path* is an ordered pair $(\boldsymbol{\pi}, \boldsymbol{\tau})$ of a topological path $\boldsymbol{\pi}$ and a waiting schedule $\boldsymbol{\tau}$, $\boldsymbol{\tau}$ having one less component than $\boldsymbol{\pi}$. A traversal path $(\boldsymbol{\pi}, \boldsymbol{\tau})$ is called simple if $\boldsymbol{\pi}$ is a simple topological path. When no ambiguity exists, we shall use the term "path" to refer to both topological and traversal paths.

Let t_S be the earliest time in which a given message can start traversing the network (this is usually the message creation time). t_S is referred to as the *starting time*. Suppose that a given message travels from a source node s to its destination node w along some traversal path $(\boldsymbol{\pi}, \boldsymbol{\tau})$, with $\boldsymbol{\pi} = (v_0, v_1, \ldots, v_m)$, $v_0 = s$, $v_m = w$, $\boldsymbol{\tau} = (\tau_0, \tau_1, \ldots, \tau_{m-1})$. Define:

$$t_A(0) \stackrel{\Delta}{=} t_S$$

$$t_A(i) \stackrel{\Delta}{=} t_A(i-1) + D_{v_{i-1},v_i}(\, t_A(i-1), \tau_{i-1} \,) \qquad 0 < i \leq m$$

$$t_D(i) \stackrel{\Delta}{=} t_A(i) + \tau_i \qquad 0 \leq i < m$$

For starting time t_S, $t_A(i)$ is the arrival time at node v_i and $t_D(i)$ the departure time from that node when traversing the network according to $(\boldsymbol{\pi}, \boldsymbol{\tau})$. The *path delay* $PD(\,(\boldsymbol{\pi}, \boldsymbol{\tau}), t_S\,)$ of a path $(\boldsymbol{\pi}, \boldsymbol{\tau})$ for starting time t_S is given by

$$PD(\ (\boldsymbol{\pi},\boldsymbol{\tau}),t_S\ )=t_A(m)-t_A(0)$$

The shortest path problem for a graph $G(V,E,D)$ can now be formulated. Given nodes s and w in V and a time $t_S \in [0,\infty)$, find a traversal path $(\boldsymbol{\pi},\boldsymbol{\tau})$, $\boldsymbol{\pi}=(s,v_1, \ldots, v_{m-1},w)$, $\boldsymbol{\tau}=(\tau_0,\tau_1, \ldots, \tau_{m-1})$ such that for any other traversal path $(\boldsymbol{\pi}',\boldsymbol{\tau}')$ with $\boldsymbol{\pi}'=(s,u_1, \ldots, u_{l-1},w)$

$$PD((\boldsymbol{\pi},\boldsymbol{\tau}),t_S) \le PD((\boldsymbol{\pi}',\boldsymbol{\tau}'),t_S) \quad .$$

Such a traversal path is a minimum delay path to which we shall also refer as a "shortest path." The topological-path component of a minimum delay traversal path is referred to as a shortest topological path. A shortest path as defined above is denoted $SP(s,w,t_S)$ and since there may be several shortest traversal paths for given s,w,t_S, we denote by $\boldsymbol{SP}(s,w,t_S)$ the set of them all.

A shortest topological path may have the property that each of its subpaths is also a shortest topological path between the source and the intermediate nodes for the same starting time. Such a topological path is said to be *concatenated*. Formally, a shortest topological path $\boldsymbol{\pi}=(v_0,v_1, \ldots, v_m)$ between nodes v_0 and v_m for starting time t_S is said to be concatenated if for each i, $0 \le i \le m-1$ there is a schedule $\boldsymbol{\tau}^{(i)}$ such that $((v_0,v_1, \ldots, v_i)\ ,\boldsymbol{\tau}^{(i)})$ is a shortest traversal path between v_0 and v_i for starting time t_S. Note that the schedule for each of the subpaths may be different. Consider now a shortest traversal path whose corresponding topological path is concatenated. If the optimal schedule of each subpath is the corresponding sub-schedule of the entire traversal schedule we say that the shortest traversal path is also concatenated. Formally, a shortest traversal path $(\boldsymbol{\pi},\boldsymbol{\tau})$, $\boldsymbol{\pi}=(v_0,v_1, \ldots, v_m)$, $\boldsymbol{\tau}=(\tau_0,\tau_1, \ldots, \tau_{m-1})$, for v_0, v_m, and starting time t_S is said to be concatenated if for each i, $0 \le i \le m-1$, the traversal path $((v_0,v_1, \ldots, v_i)\ ,(\tau_0,\tau_1, \ldots, \tau_{i-1}))$ is a shortest traversal path for v_0, v_i, and t_S.

Finally, while all the shortest paths in $\boldsymbol{SP}(s,w,t)$ have the same delay they may differ by the number of hops (i.e., edges). Considering the subset of paths of $\boldsymbol{SP}(s,w,t)$ which are simple and concatenated, and assuming this subset is nonempty we define the hop-index $H(s,w,t)$ as the minimal number of hops among these simple and concatenated shortest paths.

III. SHORTEST PATH ALGORITHMS

A. The Unrestricted Waiting Model

1. A Shortest-Path Algorithm for a Given Starting Time

Given a source node *s* and a starting time t_S we look for shortest paths and minimum delays between *s* and all other nodes for that starting time. As noted in Dreyfus[8] this is a straightforward extension to such algorithms as Dijkstra's[12] or Ford's[6]. We present the algorithm here to become familiar with some notions and differences from the standard version that will be helpful later.

We start by making some observations regarding optimal waiting times. Define $D_{ik}(t) \overset{\Delta}{=} \min_{\tau \geq 0} \{ D_{ik}(t,\tau) \}$ and consider Figure 1. In Figure 1.a we observe a message departing node *i* at time t_1 when the link delay is $d_{ik}(t_1)$, meaning that the message arrives at node *k* at $t_1{}' = t_1 + d_{ik}(t_1)$.

Figure 1: Message departure and arrival times
(a) Immediate departures
(b) Arrival times for several departure times
(c) Optimal departure time

Figure 1.b depicts arrival times at node *k* for several departing times from node *i*. Noteworthy is the fact that a delayed departure at t_3 results in an earlier arrival.

The reader will easily convince himself that to minimize the delay, a message arriving at t_1 should depart at t_4 (see Figure 1.c). The point t_4 can be characterized as the rightmost intersection point in $[t_1, t_1 + d_{ik}(t_1)]$ between $d_{ik}(t)$ and the 45° cord closest to the origin. (If $d_{ik}(t)$ is differentiable then its derivative at t_4 equals -1). Note that all messages arriving in $[t_0, t_4]$ should depart at t_4 to achieve minimum delay. $D_{ik}(t)$ is a well defined function of *t*. Figure 2 shows the function $D_{ik}(t)$ resulting from $d_{ik}(t)$ of Figure 1.

We proceed now to present the algorithm, which is a version of a labeling algorithm where each node is labeled by the earliest possible arrival time at that node with the given starting time at the source node. Following the terminology of labeling algorithms, X_k is the permanent label of the node (*NULL* indicating the node is not permanently labeled), and Y_k is its temporary label. The algorithm also builds the shortest topological-paths spanning tree which can be constructed by the values of f_k computed by the algorithm. f_k is the identity of node *k*'s father in that tree. The functions $D_{ik}(t)$, as defined above, are part of the algorithm's input.

Algorithm UW1

1. Initialization:
   $X_s \leftarrow t_S; f_s \leftarrow NIL; \forall k \neq s \quad Y_k \leftarrow \infty, \quad X_k \leftarrow NULL, \quad f_k \leftarrow NIL;$
   $j \leftarrow s;$
2. For all neighbors *k* of *j* for which $X_k = NULL$ do:

Figure 2: Optimal delay function (including optimal waiting)
for the delay function of Figure 1

a. $Y_k \leftarrow \min\{Y_k, X_j + D_{jk}(X_j)\}$

b. If Y_k changed in step 2.a then set $f_k \leftarrow j$.

3. If all nodes have nonnull *X*-value, then stop.
   Otherwise, let *l* be a node for which $X_l = NULL$ and such that $Y_l \le Y_k \ \forall k$ for which $X_k = NULL$. Set $X_l \leftarrow Y_l, j \leftarrow l$, and proceed with step 2.

The main difference between this and conventional shortest path algorithms is the calculation of $D_{jk}(X_j)$ in step 2.a. However, this is a simple operation (see next section) and thus all operations performed by the algorithms are simple.

Theorem 1: Algorithm UW1 terminates after $O(|V|^2)$ operations. After execution the relation $j \to f_j$ forms a spanning tree of *G* rooted at *s* on which each path from *s* to any node *j* is a shortest topological path for starting time t_S whose delay is given by $X_j - t_S$.

Proof: The proof follows closely that of Dijkstra's algorithm (see, for example, Even[13]).

□

Corollary 1: In the UW model, for any source-destination pair and any starting time, there exists a simple and concatenated shortest path. □

Note that optimal waiting times are implicitly given via the functions $D_{ik}(t)$ and $d_{ik}(t)$: a message arriving at node *i* at time *t* and having to depart on link (i,k) waits at node *i* for a period of τ, where τ is such that $\tau + d_{ik}(t+\tau) = D_{ik}(t)$. Thus the algorithm enables construction of shortest traversal paths.

2. A Shortest-Path Algorithm for All Starting Times

The algorithm we present is a generalization of UW1. The main difficulty stems from the fact that the values X_k are now functions of time. In addition, constructing the spanning tree is now more complicated since the tree changes with time in a manner that is harder to capture.

To overcome this latter problem we define, instead of a single value, a set of functions $Y_{kl}(t)$ that is defined for every node *k* and for each of its neighbors $l \in N_k$. $X_k(t)$ is, as before, the earliest arrival time at node *k* for starting time *t* (as known at a certain step of execution). $Y_{kl}(t)$ is, similarly, the earliest arrival time at node *l* through its neighboring node *k* of a message started at time *t*. The double index on *Y* is used to determine the spanning tree. This tree is defined by choosing a father *k* for node *l* and time t_0 such that $X_l(t_0) = Y_{kl}(t_0)$; if more than a single such node exists we choose (arbitrarily) the one with the smallest index (this is done throughout the paper whenever a selection is not unique).

Algorithm UW2

1. $\forall k$: (a) $X_k(t) \leftarrow \infty$; (b) $\forall l \in N_k$ $Y_{kl}(t) \leftarrow \infty$.
2. $X_s(t) \leftarrow t$.
3. For each $(k,l) \in E$ set $Y_{kl}(t) \leftarrow X_k(t) + D_{kl}(X_k(t))$.
4. For each $l \in V$ set $X_l(t) \leftarrow \min_{k \in N_l} \{Y_{kl}(t)\}$.
5. If no $X_l(t)$ just changed[1] (for any l) stop; otherwise, proceed with step 3.

Before we proceed to the next theorem some explanations are in order. All the assignments are to functions. Thus, for example, in step 3 the function $Y_{kl}(t)$ is set to the sum of two functions $X_k(t)$ and $D_{kl}(X_k(t))$. In order to analyze and compare algorithms that work with functions, we first introduce terminology of operations performed on functions. We refer to one dimensional functions with domain $[0,\infty)$ and range $(0,\infty)$. A *simple operation* on functions is one of the following:

- Function assignment: $f(t) \leftarrow g(t)$.
- A linear combination of two functions: $c_1 g_1(t) + c_2 g_2(t)$.
- Minimum of two functions: $\min\{g_1(t), g_2(t)\}$.
- Compounding two functions: $f(g(t))$.
- Computing the maximal right neighborhood of a function in which it is constant, i.e.: $f(t) = \max\{\tau \,|\, \forall\theta,\ t \le \theta \le \tau,\ g(\theta) = g(t)\}$[2]

We say that an algorithm operating on functions has a *functional complexity* of order $\alpha(n)$, denoted $O^f(\alpha(n))$, if there is a constant $k > 0$ such that for an input of length n (that is, n functions to be operated on) the number of simple function operations performed by the algorithm is bounded by $k\alpha(n)$. Clearly, the execution of a function operation may sometimes be quite involved. Nonetheless, the definition is useful for several reasons: (1) assuming that such operations can be performed in finite time, an algorithm of finite functional complexity is finite; (2) algorithms that operate on functions can be compared; (3) algorithms that operate on functions can be compared with those that do not (e.g., UW1 with UW2). This definition of functional complexity leads to the following theorem.

Theorem 2:

i. Algorithm UW2 terminates after at most $O^f(|V||E|)$ function operations.

---

[1] "Just changed" means that there exists at least one instance t for which the function changed its value during the most recently executed step.

[2] Such operations are not used by algorithm UW2, and are introduced for future reference.

ii. After termination, $X_k(t) - t$ is the minimum delay from node *s* to node *k* for starting time *t*.

iii. For a given time *t* and for every node *l* let *k* be the father of *l* (i.e., *k* is the lowest index for which $Y_{kl}(t) = X_l(t)$), a relation denoted by $l \rightarrow k$. Then, after termination, the relation $l \rightarrow k$ defines a spanning tree of *G* rooted at *s* for which the topological path from *s* to every node along the tree is a simple and concatenated shortest (topological) path for starting time *t*.

Proof: See Appendix A.

□

Note that the functional complexity of UW2 is $O^f(|V||E|)$ whereas the complexity of UW1 is $O(|V|^2)$. The difference is that when all starting times are considered at once, we can no longer identify at each iteration a single node that serves as the center of operation for that (and only that) iteration and be disregarded in subsequent iterations (this is done in UW1 by labeling a node permanently). Rather, in each iteration of UW2 all edges must be considered in an attempt to improve the earliest arrival times. In other words, we cannot use a Dijkstra-type algorithm, and must resort to a Ford-type one (see [13]).

B. The Forbidden Waiting Model

It is possible that in the FW model none of the shortest paths is simple or concatenated. A simple four node example is shown in Figure 3 where $d_{12}(t) = d_{13}(t) = 1$, $d_{23}(t) = d_{32}(t) = 2$, and $d_{34}(t) = 1 + (t - 5)^2$. Here $SP(1,4,t=0) = (1,3,2,3,4)$ is the only shortest path and contains a loop (since in the FW model the waiting schedule is all zeroes a traversal path is fully described by the topological path; we thus make no distinction between these two concepts when dealing with this model). The shortest path from node 1 to node 2 is $SP(1,2,t) = (1,2)$. Thus $SP(1,4,0)$ is neither simple nor concatenated.

The above fact rules out the use of Ford- or Dijkstra-type algorithms and suggests that no polynomial algorithm exists, i.e., the number of operations cannot be bounded by a polynomial in $|V|$. Indeed, if shortest paths are not concatenated, "partial results" cannot be used and the calculation of the shortest path between two nodes must consider all possible paths between them. (The authors suspect that the problem is

Figure 3: Nonsimple and nonconcatenated shortest-path for the Forbidden Waiting (FW) model

NP-Hard and are currently working on a proof).

Under some circumstances it is even possible to have an infinite shortest path (i.e., containing an infinite number of hops) although the minimum delay is finite. Figure 4 depicts such a case for the shortest path between nodes 1 and 3. Here

$$d_{12}(t) = d_{21}(t) = \begin{cases} (1-t)/2 & 0<t<1 \\ 1 & elsewhere \end{cases} \qquad d_{13}(t) = \begin{cases} 3-2t & 0\le t<1 \\ 1 & elsewhere \end{cases}$$

Starting in node 1 at some time $0<t_S<1$ node 1 can be revisited at a sequence of times $t_k = 1-(1-t_S)/4^k$ meaning that possible arrival times at node 3 are $T_k = t_k + d_{13}(t_k) = 2+(1-t_S)/4^k > 2$. However, for $k\rightarrow\infty$ one can arrive at node 3 at time $T=2$.

In Halpern[10] an algorithm is presented that attempts to find a (finite) shortest path in a time-dependent network. Waiting at every node is limited to a predetermined (possibly empty) set of intervals and delay functions are nonnegative and piecewise continuous. The FW model is a special case of that model. From the above example of an infinite shortest path we conclude that Halpern's algorithm cannot perform its task in all cases for which it was supposed to. A close examination of Halpern's proof reveals that slightly tighter constraints must be imposed on the delay functions for the algorithm to be correct (such as the constraints discussed below).

There are only a few facts that can be stated generally regarding this model. One obvious characteristic is that if optimal waiting time is always zero (in which case $D_{ik}(t)\equiv d_{ik}(t)$) the behavior of the algorithm in the FW and UW will be identical. This is a characteristic of a network with only FIFO links which, for differentiable delay functions means that the slope of $d_{ik}(t)$ never decreases below -1. Another general fact that can be stated is that if for every i, k and every finite T there exists a $\delta > 0$ (depending on i, k, and T) such that for every $t \le T$ holds $d_{ik}(t) > \delta$, then the shortest path between any two nodes is finite.

Figure 4: Example of infinite shortest-path with finite minimum delay

C. The Source Waiting Model

We observe an immense gap between the UW and FW models in finding shortest paths: the UW model behaves just like the time-independent case (i.e., when delays are constant) whereas in the FW model it seems that such efficient algorithms do not exist. However, by just alleviating slightly the constraint on waiting, namely by permitting source waiting, efficient algorithms can be found. In particular if all the delay functions are of the class described below the following theorem proves equivalence (from the shortest path standpoint) between the UW and SW models.

Theorem 3: If $\forall (k,j) \in E$ $d_{kj}(t)$ is continuous or piecewise continuous with only negative discontinuities (i.e., $\forall t \quad d_{kj}(t^-) \geq d_{kj}(t^+)$) and such that for all t either $d_{kj}(t) = d_{kj}(t^-)$ or $d_{kj}(t) = d_{kj}(t^+)$, then every shortest topological path in the UW model is also a shortest topological path in the SW model, having the same delay. In other words

if $(\boldsymbol{\pi}, \tau) = ((v_0, v_1, \ldots, v_m), (\tau_0, \tau_1, \ldots, \tau_{m-1})) \in \boldsymbol{SP}_{UW}(s,w,t)$

then for some τ_0':

$(\boldsymbol{\pi}, \boldsymbol{\tau}') = (v_0, v_1, \ldots, v_m), (\tau_0', 0, \ldots, 0)) \in \boldsymbol{SP}_{SW}(s,w,t)$ and $PD((\boldsymbol{\pi}, \tau), t) = PD((\boldsymbol{\pi}, \tau'), t)$.

Proof: Assume first that all delay functions are continuous.

Let $(\boldsymbol{\pi}_{UW}, \tau_{UW}) = ((v_0, \ldots, v_m), (\tau_0, \tau_1, \ldots, \tau_{m-1})) \in \boldsymbol{SP}_{UW}(s,w,t_S)$, $s = v_0$, $w = v_m$, and let $t_A(i)$ and $t_D(i)$ be, respectively, the arrival and departure times at node $v_i \in \boldsymbol{\pi}_{UW}$ when traveling along $(\boldsymbol{\pi}_{UW}, \tau_{UW})$. To prove the theorem we show that for any given $v_i \in \boldsymbol{\pi}_{UW}$ and for any given time $T \geq t_A(i)$ there is a source departure time $T_D \geq t_S$ such that if we leave s at T_D and travel along $\boldsymbol{\pi}_{UW}$ according to the SW model we arrive at v_i at time T. In other words, we can adjust the departure times from s so that we arrive at the node v_i at any desired preset time T as long as it is not earlier than $t_A(i)$.

We prove this claim by induction on the nodes in $\boldsymbol{\pi}_{UW}$. For $i = 0$ the claim is trivially true. Assuming truth for the ith node we prove for the $i+1$st.

Choose $T \geq t_A(i+1)$. To arrive at node v_{i+1} at time T we have to depart node v_i at time θ such that $\theta + d_{v_i, v_{i+1}}(\theta) = T$. The question is whether such θ exists and whether we can arrive at node v_i at that time. Consider therefore the function $f(x) = x + d_{v_i, v_{i+1}}(x)$. $f(x)$ is continuous, $f(x) \rightarrow \infty$ as $x \rightarrow \infty$ and $f(t_D(i)) = t_A(i+1) \leq T$. Therefore, according to the Intermediate Value Theorem of infinitesimal calculus, there exists a θ such that $f(\theta) = T$ and $\theta \geq t_D(i) \geq t_A(i)$. By the inductive assumption there exists a departure time T_D that will cause arrival in node v_i at time θ and therefore at node v_{i+1} at time T.

Suppose now that delay functions have a countable number of negative discontinuities. Up to the point where the continuity of $f(x)$ is assumed the proof remains the same. Next, for this case, $f(x)$ is piecewise continuous with negative discontinuities, $f(x) \rightarrow \infty$ as $x \rightarrow \infty$ and $f(t_D(i)) = t_A(i+1) \leq T$. It can be verified that the Intermediate Value Theorem holds for this case as well, so that there exists a θ such that $f(\theta) = T$ and $\theta \geq t_D(i) \geq t_A(i)$; the proof follows. □

It should be noted that while the same path is being traversed, the departure times for each intermediate node may be different. Indeed, as we shall see in Algorithm SW1 that follows, a separate calculation is needed for each source-destination pair.

From Theorem 3 and Corollary 1 we get immediately:

Corollary 2: If all $d_{ik}(t)$ are as in Theorem 3 then, for a given source and destination pair and a starting time there exists a shortest traversal path in the SW model that is simple and whose corresponding topological path is concatenated. □

1. A Shortest-Path Algorithm for Continuous Functions and a Given Starting Time

As in the UW model, we first present an algorithm to find the shortest path between some node *s* and all other nodes for a given starting time t_S. The addition here is the computation of the departure times for every destination node (we assume the delay functions are those for which Theorem 3 holds). In the algorithm we make use of the results of Theorem 3 and Corollary 2.

For a given destination we first determine the minimum delay and shortest topological path for the UW model. By Theorem 3 we know that the same topological path can be used for the SW model with the same arrival time at the destination. We thus start at the destination node for the known arrival time and compute the departure time from its predecessor on the shortest topological path. This process is then repeated for all nodes in the path. When this backtracking is complete, i.e., when the source node is reached in the computation process, we have computed the departure time from the source node which trivially yields the source waiting time.

Algorithm SW1

1. Compute a shortest topological path using Algorithm UW1 for node *s* and starting time t_S (after execution both the spanning tree relation $j \rightarrow f_j$ and the minimum delay $X_j - t_S$ are known for every node *j*).
2. For each $i \in V$ do
   a. Set $t_A \leftarrow X_i, j \leftarrow i$.
   b. Until $j = s$ do
      i. Find t_D such that $t_D \geq X_{f_j}$ and $t_D + d_{f_j,j}(t_D) = t_A$.
      ii. $t_A \leftarrow t_D, j \leftarrow f_j$.
   c. $t_W(s,i,t_S) \leftarrow t_A - t_S$.

Theorem 4: The following is true for Algorithm SW1:

i. It stops after $O(|V|^2)$ operations.

ii. After termination the relation $j \rightarrow f_j$ forms a spanning tree of G rooted at s on which each path from s to any node j is a shortest topological path for time t_S whose delay is given by $X_j - t_S$.

iii. After termination for each $i \in V$ $t_W(s,i,t_S)$ is a (nonnegative) waiting time such that departing node s at time $t_S + t_W(s,i,t_S)$ and moving without waiting along the tree will result in arrival at node i with minimum delay.

Proof: By Theorem 1, the relation $j \rightarrow f_j$ defines a spanning tree meaning that step 2.b terminates (in fact is executed $O(|V|)$ times) provided the value of t_D, as appears in step 2.b.i of the Algorithm, can always be computed. We now demonstrate this fact.

For continuous or piecewise continuous functions with only negative discontinuities the equation $t_D + d_{kj}(t_D) = t_A$ has at least one solution $t_D \in (-\infty, \infty)$ for any given t_A. The solutions are the intersection points between $d_{kj}(t)$ and the $-45°$ line $t_A - t$. Moreover, Theorem 3 guarantees that one of these solutions fulfills $t_D > X_{f_j}$, proving claim (iii).

The two other claims follow immediately. Step 2.b involves $O(|V|)$ operations and is executed $O(|V|)$ times resulting in a total of $O(|V|^2)$ operations for step 2. By Theorem 1 claim (ii) holds. Claim (i) follows directly from the above and Theorems 1 and 3. □

As an example of the above computation consider a network all of whose links have the same delay functions, that is, $d_{ik}(t) = d(t)$. Starting at a given time t_S we seek minimum delay from the source node to a node 4 hops away traversing nodes 1,2,3 on its way. Figure 5 shows the function $d(t)$ and the respective arrival and departure times in the UW model (the dashed lines). Having determined t_{a_4} we compute backwards the visiting times according to the SW model (marked t'_i). These are computed as follows. We determine t'_3 as the departure time for arrival at t'_4. We then compute t'_2, the departure time for arrival at t'_3, and so on (the dot-dashed lines in Figure 5). Finally t'_S determines the departure time from the source according to the SW model, and $t_W(s,w,t_S) = t'_S - t_S$ is the waiting time.

2. A Shortest-Path Algorithm for All Starting Times

We assume in this section that the delay functions are those for which Theorem 3 holds. The following observations show how Algorithm UW2 is used for solving the same problem in the SW model. As was proved, when UW2 terminates $X_k(t)$ is the earliest arrival time at node k from node s for starting time t in the UW model. Since waiting is allowed in the UW model, $t_2 > t_1$ implies $X_k(t_2) \geq X_k(t_1)$. With each time instant t_0 and destination node k we may thus associate the latest starting time, $L_k(t_0)$, that results in the same arrival time at the destination. Formally, for each t_0 and node k: (a) $L_k(t_0) \geq t_0$; (b) $X_k(L_k(t_0)) = X_k(t_0)$; (c) $\forall t > L_k(t_0): \quad X_k(t) > X_k(t_0)$. In fact, $L_k(t_0)$ is the latest departure time from node s in the UW model that enables arrival at node k at the earliest possible arrival time for starting time

Figure 5: Computation of waiting times

t_0. It is fairly easy to determine $L_k(t_0)$ for given k and t_0, since from the above follows that $\forall t \in [t_0,\ L_k(t_0)]$ we have $X_k(t) = X_k(t_0)$. The following lemma establishes a useful property of the values of $L_k(t)$.

Lemma 1: For source node *s*, destination node *w*, starting time t_S, and $L_w(t_S)$ as defined above, let $P = (\boldsymbol{\pi}, \tau)$ be a shortest traversal path for $s, w, L_w(t_S)$. Then $\tau = (0, 0, \ldots, 0)$ i.e, traversing is without waiting (neither at the source nor enroute).

Proof: Assume the contrary, i.e., there is a shortest path

$$P = (\boldsymbol{\pi}, \tau) = (\ (s, v_1, \ldots, v_{m-1}, w), (\tau_0, \ldots, \tau_{m-1})\ )$$

for $s, w, L_w(t_S)$ such that for some $0 \le i \le m-1$, $\tau_i \ne 0$. We distinguish two cases depending on the value of τ_0. Assume first that $\tau_0 \ne 0$. Since $X_w(t)$ is the earliest arrival time at node *w* for starting time *t* and since *P* is a shortest path we have

$$X_w(\ L_w(t_S)\ ) = PD(P, L_w(t_S)) + L_w(t_S) \quad .$$

Consider now the traversal path $P' = (\boldsymbol{\pi}, \tau')$ where $\tau' = (0, \tau_1, \ldots, \tau_{m-1})$ and consider the starting time $L_w(t_S) + \tau_0$. It is straightforward to notice that the arrival time at node *w* for these two paths is the same, namely

$$PD(P'\ ,\ L_w(t_S) + \tau_0\ ) + L_w(t_S) + \tau_0 = PD(P,\ L_w(t_S)) + L_w(t_S)$$

hence

$$X_w(L_w(t_S)) = PD(P',\ L_w(t_S) + \tau_0) + L_w(t_S) + \tau_0 \quad .$$

On the other hand, again because $X_w(\cdot)$ is the earliest arrival time, we have

$$PD(P',\ L_w(t_S)+\tau_0)\ +\ L_w(t_S)+\tau_0\ \geq\ X_w(L_w(t_S)+\tau_0)$$

and we conclude that $X_w(L_w(t_S)) \geq X_w(L_w(t_S)+\tau_0)$. But $L_w(t_S)+\tau_0 > L_w(t_S) \geq t_S$, thus property (c) in the definition of $L_w(t_S)$ implies that $X_w(L_w(t_S)) < X_w(L_w(t_S)+\tau_0)$, contradicting the above conclusion.

Assume now that $\tau_j \neq 0$ for some $j>0$ and let i be the smallest such j (i.e., $\tau_0=\tau_1= \cdots =\tau_{i-1}=0$). Denote by $t_D(i)$ the departure time from node v_i on path P for starting time $L_w(t_S)$. Theorem 3 guarantees that there is some $\tau>0$ such that if we leave node s at time $L_w(t_S)+\tau$ traversing the topological path $\boldsymbol{\pi}$, we arrive at node v_i at time $t_D(i)$ without waiting enroute. Thus for $P'=(\boldsymbol{\pi},\tau')$, $\tau'=(\tau,0,\ldots,0,\tau_{i+1},\ldots,\tau_{m-1})$, we have $PD(\ P',\ L_w(t_S)\ )=PD(\ P,\ L_w(t_S)\ )$, and thus P' is also a shortest path for $s,w,L_w(t_S)$ but with $\tau_0=\tau>0$, and we have already seen that a contradiction follows. □

Corollary 3: For source node s, destination node w, and starting time t_S, $L_w(t_S)-t_S$ is an optimal source waiting time for the SW model and any shortest topological path for $s,w,L_w(t_S)$ in the UW model is also a shortest topological path for s,w,t_S in the SW model.

Proof: Let $\lambda_w(t_S)$ (where $\lambda_w(t_S) \geq t_S$) be the latest departure time for starting time t_S that assures earliest arrival time at node w in the SW model. From Theorem 3 it follows that $\lambda_w(t_S)=L_w(t_S)$, establishing the first part of the corollary. The second part follows directly from Lemma 1. □

The above corollary makes our approach clear. We first run Algorithm UW2 and obtain for each $w \in V$ the function $X_w(t)$ which is the minimal delay from node s to w for all times t in both the UW and SW models. To construct a shortest traversal path for starting time t we calculate the source waiting time $WAIT(s,w,t)$ according to

$$WAIT(s,w,t)\ =\ \max\{\tau \mid X_w(t+\tau)=X_w(t)\}$$

i.e., we want to leave at $L_w(t)$ and therefore wait for $L_w(t)-t$. We then traverse the topological path defined by the values of $X_k(t+WAIT(s,w,t))$ and $Y_{kl}(t+WAIT(s,w,t))$.

The above discussion is summarized by the following formal specification of the algorithm, along with the corresponding theorem.

Algorithm SW2

1. Execute Algorithm UW2.
2. For each $w \in V$ do $WAIT(s,w,t)\ \leftarrow\ \max\{\tau \mid X_w(t+\tau)=X_w(t)\}$.

Theorem 5: The following is true for Algorithm SW2 and for the SW model:

i. It stops after $O^f(|V||E|)$ function operations.
ii. After termination, $X_w(t)-t$ is the minimum delay from node s to node w for starting time t, and $WAIT(s,w,t)$ is an optimal source waiting time.

iii. For a time t and for every node w, the topological path between s and w on the spanning tree defined (in the way described in Theorem 2) by $X_l(t+WAIT(s,w,t))$, $Y_{kl}(t+WAIT(s,w,t))$ is a shortest topological path for s,w,t.

Proof: According to Theorem 2, claim (i) is true for the first part of the algorithm. The second part of the algorithm involves $O^f(|V|)$ function operations. Thus, we conclude that the algorithm terminates after executing $O^f(|V||E|)$ function operations, proving claim (i). Claim (ii) follows from Theorems 2 and 3, together with Corollary 3. Claim (iii) follows from Corollary 3. □

Note that the only functions that should be stored are $X_k(t)$ and $Y_{kl}(t)$. $WAIT(s,w,t)$ need not be computed as a function for all t. Rather, the calculation of $WAIT(s,w,t_0)$ can be done only when the need arises for a message generated at s at time t_0 and whose destination is w. A shortest topological path to node w for the SW model can then be constructed from the spanning tree defined by $X_l(t_0+WAIT(s,w,t_0))$, $Y_{kl}(t_0+WAIT(s,w,t_0))$ as described in the theorem. In this way we avoid using the last type of simple function operations, and restrict the definition of functional complexity to much simpler types of operations.

3. Extension to General Piecewise-Continuous Functions

To gain some insight regarding the problematic aspect of positive discontinuities consider the four-node network in Figure 6 where

$$d_{12}(t) = d_{23}(t) = 400 \quad d_{13}(t) = \begin{cases} 1 & t \le 1 \\ 1000 & t > 1 \end{cases} \quad d_{34}(t) = \begin{cases} 1000 & t < 10 \\ 1 & t \ge 10 \end{cases}$$

In the UW model for $t_S=0$ we get $SP(1,4,t=0)=((1,3,4),(0,9))$ with 11 units delay. In the SW model $SP(1,4,0)=((1,2,3,4),(0,0,0))$ with 801 units delay, demonstrating that the shortest path and minimum delay in the SW model are different from those in the UW model.

Figure 6: Example network for computing
the shortest-path for noncontinuous functions

Let us make a slight change in the problematic delay function as follows:

$$d_{13}(t) \; = \; \begin{cases} 1 & t \leq 1 \\ 999t - 998 & 1 < t \leq 2 \\ 1000 & t > 2 \end{cases}$$

This function is "almost" the same as the one before except for being continuous. With this change, one could, in the SW model, leave node 1 at $t_S = 1.008$, go through node 3 arriving in node 4 at $t = 11$ as we would have in the UW model.

While the above demonstrates that corollary 2 does not always hold for general noncontinuous functions, the following demonstrates the reason. Consider, again, a network all of whose link delay functions are the same, noncontinuous function $d(t)$ (see Figure 7). For starting time t_S the UW arrival time t_{a_4} is computed. However, going backwards from $t'_4 = t_{a_4}$ we notice that there exists no time t'_2 that will cause arrival at t'_3. Graphically, the $-45°$ line at t'_3 does not intersect $d(t)$.

Although corollary 2 does not hold for general noncontinuous functions we can expand the previous results to cover a wider class of functions by slightly relaxing the waiting constraints. Clearly, if we allow unrestricted waiting at nodes whose incoming links have noncontinuous functions we can do as well as in the UW model since each such node can be considered a source for traversal of a subnetwork all of whose delay functions are continuous. However, for piecewise continuous functions, we shall demonstrate in the following that to achieve minimum delay equal to that of the UW model, the amount of waiting in such intermediate nodes may be bounded in terms of when and for how long waiting must be allowed.

Figure 7: Computation of waiting times
for discontinuous functions

Consider a network whose delay functions are piecewise continuous, and we require also that for all i,k,t either $d_{ik}(t)=d_{ik}(t^-)$ or $d_{ik}(t)=d_{ik}(t^+)$. Our SW policy is relaxed to allow limited waiting in nodes whose incoming links have noncontinuous delay functions (in addition to the unrestricted waiting at the source). Since negative discontinuities are treated in Theorem 3 we focus here on positive discontinuities. Let $d_{ik}(t)$ have a positive discontinuity at t_0, i.e., $d_{ik}(t_0^+)>d_{ik}(t_0^-)$. To such a discontinuity we assign a *relaxation instant* τ_0 around the time $t_0+d_{ik}(t_0^-)$ when waiting restrictions are relaxed in the following manner. Messages arriving at node k from node i at the relaxation instant τ_0 may wait at node k for an amount of time not exceeding $d_{ik}(t_0^+)-d_{ik}(t_0^-)+\varepsilon$ for any predetermined $\varepsilon>0$ (ε can be arbitrarily small). Moreover, if $d_{ik}(t_0)=d_{ik}(t_0^-)$ then we may even choose $\varepsilon=0$. We refer to this policy as the relaxed SW policy.

Figure 8 illustrates a simple example of the above method. In the figure $d_{ik}(t)$ has a positive discontinuity at $t=t_0$ of the kind $d_{ik}(t_0^-)=d_{ik}(t_0)$. We choose $\tau_0=t_1=t_0+d_{ik}(t_0^-)$ and allow messages departing on link (i,k) at time t_0 to wait at node k an amount of time limited to $d_{ik}(t_0^+)-d_{ik}(t_0^-)$. Doing so we guarantee that messages arriving at node k on link (i,k) at time $\tau_0=t_1$ may depart from k at any time during the interval $[t_1,t_2]$.

Lemma 2: In the relaxed SW model the results of Theorem 3 hold for piecewise continuous delay functions.

Proof: See Appendix B.

□

Figure 8: Example for treating delay-function discontinuity

IV. CONCLUSION

In this paper we have investigated the shortest path problem for networks in which link delays are functions of time. It is shown that the problem can be solved efficiently when no constraints are imposed on waiting times at the nodes. On the other hand, the examples presented for the FW model indicate that general waiting constraints render the problem very complex. For a wide class of delay functions, it is shown that an efficient solution exists if we allow waiting just at the source node. It should be pointed out that delaying the departure at the source node is commonplace in practical cases. Moreover, it is proved that performance in this case, from the minimum delay standpoint, is equivalent to that of the most unrestricted case.

The relaxation needed in order to cover all piecewise continuous delay functions is slight, since it changes the waiting constraints in only a countable number of points in a continuum. Since in most practical cases delay functions are continuous or piecewise continuous, we conclude that waiting restrictions at intermediate nodes do not really impose limitations on minimal delay nor do they increase the complexity of computing these delays.

In this paper we restricted ourselves to delay functions for which an optimal waiting time can be defined for all times. One can avoid this restriction by considering paths that are ε-optimal, i.e., their delay is within a predetermined and arbitrary small range of the infimum delay of all paths. For a wide family of delay functions (including the piecewise continuous ones) the algorithms presented in this paper can be applied to the ε-optimal paths problem after making some modifications e.g., instead of an optimal waiting time we consider an ε-optimal one, defined for time t by $D_{ik}(t,\tau^{\varepsilon}) \le D_{ik}(t,\tau)+\varepsilon$ for all $\tau \ge 0$; the function $D_{ik}(t)$ is then defined accordingly.

We have presented algorithms (UW2 and SW2) that handle all starting times at once. These are useful in a scenario where messages may be created at the source node at any time and an optimal path should be found for each of them. An efficient way is to run an algorithm for all starting times prior to the time range being considered; then, for a message created at time t_S, an optimal path is immediately found from the values of $X_k(t_S)$ and $Y_{kl}(t_S)$ that are computed by the algorithm (along with the value $L_k(t_S)$ computed in the SW model). Those algorithms were also presented to make a conceptual point: they demonstrate the difference between the cases having a single, given, starting time, and those that consider all starting times at once. Their study offers more insight into the nature of the time-dependent environment (e.g., the functions $L_w(t)$) and they serve as building blocks for extending these algorithms to a distributed environment.

There are several possible extensions to these results, two of which we describe here. The drive for this paper came from the field of computer communication networks for which it is interesting to consider distributed versions of network control algorithms. Indeed, for the static or quasi-static environment, several distributed shortest path algorithms were developed based on centralized versions. (See for example Segall[14]). Such distributed versions were developed and investigated also for the algorithms presented in this paper[15].

This paper considers minimum-delay paths. In a general case, the "length" of a link may be characterized by factors other than delay such as reliability, traversal cost, etc. Since these factors may also be time-dependent it is of interest to investigate the shortest path problem for general time-dependent link weights. This problem is far more complicated than the minimum-delay one to the extent that the least restricted UW model does not guarantee the existence of simple, concatenated, or even finite optimal paths. The interested reader is referred to [16] for discussion and analysis of this problem.

APPENDIX A

Proof of Theorem 2

We start with some notations. Denote each execution of step 3 of algorithm UW2 as an "iteration". Let VAR_n be the value of *VAR* after the *n*th iteration. Let $f_k(t)_n$ be the neighbor *l* of node *k* with the smallest index for which $X_k(t)_n = Y_{lk}(t)_n$ (clearly such a neighbor exists). Finally, for given i,t,n let $PATH(s,i,t)_n = (v_m, v_{m-1}, \ldots, v_1, v_0)$, where $s = v_m$ and $i = v_0$, be defined by $v_l = f_{v_{l-1}}(t)_n$. The theorem is proven through the following two lemmas.

Lemma A1: For all i,t,n if $X_i(t)_n$ is defined (not ∞) then $PATH(s,i,t)_n$ is a simple topological path and $X_i(t)_n - t$ is its delay for starting time *t* (when choosing an optimal waiting time at each node).

Proof: From the way $X_i(t)$ and the *Y*'s are calculated in all nodes it is clear that $PATH(s,i,t)_n$ is a well defined and finite topological path, and that $X_j(t)_{n+1} \le X_j(t)_n$. Denote $f \stackrel{\Delta}{=} f_j(t)_n$; thus

$$X_j(t)_n = Y_{f,j}(t)_n = D_{f,j}(X_f(t)_n) + X_f(t)_n > X_f(t)_n$$

meaning that the *X* value is strictly increasing along the path, and since it is impossible to move around a loop with monotonically decreasing *X*'s it follows that $PATH(s,i,t)_n$ is simple for all i,t,n.

By the way X_k is constructed along the topological path, it is clear that $X_k(t)_n - t$ is the delay of $PATH(s,i,t)_n$. □

In Section II we defined the hop-index $H(s,w,t)$ as the minimal number of hops (edges) among all the simple and concatenated shortest paths for s,w,t. We use this definition to prove the following lemma.

Lemma A2: Let $h = H(s,i,t)$. Then $\forall m \ge h$ $PATH(s,i,t)_m$ is a simple and concatenated shortest topological path for s,i,t whose delay (choosing an optimal waiting time at each node) is $X_i(t)_m - t$.

Proof: By induction on *h*. For $h = 0$ the assumption is trivially correct since then $s = i$. Assuming it is correct for all times *t* and for all nodes *k* for which $H(s,k,t) = h$ we prove it for all times *t* and for all nodes for which the hop index (corresponding to that time) is $h+1$. Let t_0 and *i* be such that $H(s,i,t_0) = h+1$. Node *i* has at least one neighbor *k* whose hop index for t_0 equals *h*. By the inductive assumption, after the $m = h$ iteration $PATH(s,k,t_0)_m$ is a simple and concatenated shortest topological path whose delay is $X_k(t_0)_m - t_0$. Thus, at its $(h+1)$st iteration (at most) the algorithm sets Y_{ki} at step 3 and subsequently sets X_i (at step 4) to the earliest arrival times. This operation appends node *i* to $PATH(s,k,t_0)$ to yield $PATH(s,i,t_0)$ which is therefore a simple and concatenated shortest topological path.

Henceforth, once again by the inductive assumption, X_i will remain unchanged. Since t_0 is arbitrary the assumption holds for all t, completing the proof. □

Lemma A2 shows that after at most $O(|V|)$ iterations the algorithm terminates. Since step 3 involves $O^f(|E|)$ functional operations the algorithm requires a total of $O^f(|V||E|)$ functional operations. The rest of the claims of the Theorem follow directly from the two lemmas. □

APPENDIX B

Proof of Lemma 2

Let $SP_{UW}(\boldsymbol{\pi}_{UW},\boldsymbol{\tau}_{UW}) = ((v_0,v_1,\ldots,v_m), (\tau_0,\tau_1,\ldots,\tau_{m-1})) \in \boldsymbol{SP}_{UW}(s,w,t_S)$, $v_0=s$, $v_m=w$, and let $t_A(i)$ and $t_D(i)$ be respectively the arrival and departure times for node $v_i \in \boldsymbol{\pi}_{UW}$ when traveling along SP_{UW}. Without loss of generality, we assume that $\boldsymbol{\tau}_{UW}$ is such that $t_A(i+1)$ is the earliest possible arrival time at v_{i+1} when departing from v_i at $t \geq t_A(i)$ (this rule applies recursively to $i=0,1,\ldots,m-1$). We show that for any $v_i \in \boldsymbol{\pi}_{UW}$ and any time $T \geq t_A(i)$, there is a departure time $T_D \geq t_S$ such that if we leave s at T_D and travel along $\boldsymbol{\pi}_{UW}$ according to the relaxed SW policy we may depart from node v_i at time T. We prove this claim by induction on the nodes of $\boldsymbol{\pi}_{UW}$. For $i=0$ the claim is trivially true; assuming truth for the ith node we prove for the $i+1$st.

Choose $T \geq t_A(i+1)$. Since $t + d_{v_i,v_{i+1}}(t) \underset{t\to\infty}{\rightarrow} \infty$, the set $S_T \stackrel{\Delta}{=} \{t \mid t + d_{v_i,v_{i+1}}(t) > T, t_A(i) \leq t\}$ is nonempty, and let $t_1 \stackrel{\Delta}{=} \inf S_T$. If $t_1 + d_{v_i,v_{i+1}}(t_1) = T$, then we are done, since by the inductive assumption we may depart from node v_i at time t_1, arrive at v_{i+1} at time T, and depart immediately. Otherwise, since $t_1 = \inf S_T$, it follows that $T_1 \stackrel{\Delta}{=} t_1 + d_{v_i,v_{i+1}}(t_1^-) \leq T$ and $T_2 \stackrel{\Delta}{=} t_1 + d_{v_i,v_{i+1}}(t_1^+) \geq T$; moreover, if $T_1 = T$ then $T_2 > T$ (otherwise we have $t_1 + d_{v_i,v_{i+1}}(t_1) = T$). Thus, there is a positive jump of $d_{v_i,v_{i+1}}(t)$ at t_1.

Assume first that $d_{v_i,v_{i+1}}(t_1) = d_{v_i,v_{i+1}}(t_1^-)$. Since $t_1 \geq t_A(i)$, it follows that $T_1 \geq t_A(i+1)$ otherwise $t_A(i+1)$ would not be the earliest arrival time at v_{i+1} for departure time $t \geq t_A(i)$ from v_i contrary to the assumption made on $\boldsymbol{\tau}_{UW}$. We choose T_1 as the relaxation point for t_1. Since $t_1 \geq t_A(i)$ it follows from the inductive assumption that in the relaxed SW model one may depart from node v_i at time t_1 and therefore arrive at node v_{i+1} at time T_1. Since

$$T - T_1 = T - [t_1 + d_{v_i,v_{i+1}}(t_1)] = T - [t_1 + d_{v_i,v_{i+1}}(t_1^-)] \leq T_2 - [t_1 + d_{v_i,v_{i+1}}(t_1^-)] =$$

$$= [t_1 + d_{v_i,v_{i+1}}(t_1^+)] - [t_1 + d_{v_i,v_{i+1}}(t_1^-)] = d_{v_i,v_{i+1}}(t_1^+) - d_{v_i,v_{i+1}}(t_1^-)$$

one may wait at v_{i+1} for a period of $T - T_1$ and then depart at time T.

Assume now that $d_{v_i,v_{i+1}}(t_1) \neq d_{v_i,v_{i+1}}(t_1^-)$. Since $d_{v_i,v_{i+1}}(t)$ is piecewise continuous, for any $\varepsilon > 0$ there is a value $\hat{t}_1 < t_1$ close enough to t_1 such that

a. $\hat{t}_1 \geq t_A(i)$.

b. $|d_{v_i,v_{i+1}}(t_1^-) - d_{v_i,v_{i+1}}(\hat{t}_1)| < \varepsilon/2$.

c. $t_1 - \hat{t}_1 < \varepsilon/2$.

Denote $\hat{T}_1 \overset{\Delta}{=} \hat{t}_1 + d_{v_i,v_{i+1}}(\hat{t}_1)$. We choose $\hat{T}_1$ as the relaxation point for t_1. Since $\hat{t}_1 \geq t_A(i)$ one may depart from node v_i at time $\hat{t}_1$ and arrive at node v_{i+1} at time $\hat{T}_1$. We have

$$T - \hat{T}_1 = T - [\hat{t}_1 + d_{v_i,v_{i+1}}(\hat{t}_1)] < T - [\hat{t}_1 + d_{v_i,v_{i+1}}(t_1^-) - \varepsilon/2] < T - [t_1 + d_{v_i,v_{i+1}}(t_1^-) - \varepsilon] \leq T_2 - [t_1 + d_{v_i,v_{i+1}}(t_1^-) - \varepsilon] =$$

$$= [t_1 + d_{v_i,v_{i+1}}(t_1^+)] - [t_1 + d_{v_i,v_{i+1}}(t_1^-) - \varepsilon] = d_{v_i,v_{i+1}}(t_1^+) - d_{v_i,v_{i+1}}(t_1^-) + \varepsilon.$$

Thus, one may wait at v_{i+1} for a period of $T - \hat{T}_1$ and then depart at time *T*. □

ACKNOWLEDGMENT

The authors would like to thank the reviewers for an extraordinary reviewing effort that considerably improved the presentation of the work.

REFERENCES

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

2. 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).

3. S. Even and H. Gazit, ‘‘Updating Distances in Dynamic Graphs,’’ *Methods of Operations Research* **49** pp. 371-387 (May 1985).

4. E. Klafszky, ‘‘Determination of Shortest Path in a Network with Time-Dependent Edge-Lengths,’’ *Mathematische Operationsforschung u. Statistik* **3** pp. 255-257 (1972).

5. K.L. Cooke and E. Halsey, ‘‘The Shortest Route Through a Network with Time Dependent Internodal Transit Times,’’ *Journal Math. Analysis Applications* **14** pp. 493-498 (1966).

6. L.R. Ford Jr. and D.R. Fulkerson, *Flows in Networks,* Princeton University Press, Princeton, New Jersey (1962).

7. L.R. Ford Jr. and D.R. Fulkerson, ‘‘Constructing Maximal Dynamic Flows from Static Flows,’’ *Operations Research* **6** pp. 419-433 (1958).

8. S.E. Dreyfus, ‘‘An Appraisal of some Shortest Path Algorithms,’’ *Operations Research* **17** pp. 395-412 (1969).

9. S.T. Ling, K. Furuno, and Y. Tezuka, ‘‘Optimal Path in Networks with Time-Varying Traverse Time and Expenses Branches,’’ Technological Reports of Osaka University, Japan (1972).

10. J. Halpern, ‘‘The Shortest Route with Time Dependent Length of Edges and Limited Delay Possibilities in Nodes,’’ *Zeitschrift fuer Operations Research* **21** pp. 117-124 (1977).

11. M. Gerla and L. Kleinrock, ‘‘Flow Control: A Comparative Survey,’’ *IEEE Trans. on Communications* **COM-28**(4) pp. 553-574 (April 1980).

12. E.W. Dijkstra, ‘‘A Note on Two Problems in Connection with Graphs,’’ *Numerical Analysis* **1** pp. 269-271 (October 1959).

13. S. Even, *Graph Algorithms,* Computer Science Press, New York (1979).

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

15. 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).

16. A. Orda and R. Rom, ‘‘Minimum Weight Paths in Time-Dependent Networks,’’ *Networks* **21**(3) pp. 295-319 (May 1991).