MINIMUM DELAY ROUTING

IN MULTISATELLITE NETWORKS†

Ariel Orda   Raphael Rom   Moshe Sidi

Department of Electrical Engineering and the S. Neaman Institute
Technion - Israel Institute of Technology
Haifa, Israel 32000.

ABSTRACT

We consider the problem of traveling with least expected delay in networks whose link delays change probabilistically according to Markov chains. This is a typical routing problem in dynamic multisatellite networks.

We formulate several optimization problems, posed on infinite and finite horizons, and consider them with and without using memory in the decision making process. We prove that all these problems are, in general, intractable. However, for networks with nodal stochastic delays, a simple polynomial optimal solution is presented. This is typical of networks in which the dominant delays are incurred by the nodes. For more general networks, a tractable ε-optimal solution is presented.

† This work was supported in part by the Technion V.P.R. fund and the fund for the promotion of research at the Technion.

I. INTRODUCTION

Routing has a major impact on the functionality, performance, and flexibility of computer networks in general and satellite networks in particular. This is why such a vast amount of research was conducted to devise efficient routing algorithms. Routing is generally defined as identifying a path in the network that optimizes a certain criterion. While many criteria have been suggested, minimum delay is the most natural and most common one.

In terms of analysis, minimum delay routing falls into the category of shortest path problems where the delays are modeled as weights whose sum is to be minimized. A large number of algorithms were proposed as a solution to the problem under a variety of conditions and constraints[1]. The vast majority of these algorithms deal with fixed networks, i.e., those with fixed topology and link delays. While these algorithms did solve the basic static problems, the advancement of computer networks created dynamic problems that could not be solved by static models.

There are several possible approaches to model the dynamic behavior of networks. The most popular one is the quasi-static model, in which it is assumed that link delays change from time to time but remain constant between these (infrequent) changes[2,3,4,5]. This approach is memoryless and does not make use (nor does it benefit from) the known dynamics of the network, i.e., the predicted or the observed manner in which the delays change.

Another approach is to consider a deterministic, time-dependent network, in which the exact behavior of link delays is described by a given set of delay functions. Shortest path problems for such a model were studied in several works[6,7,8,9] and recently in the context of computer networks [10,11]. The time-dependent approach is beneficial when the dynamics of the network is deterministic or can be predicted fairly well. However, in many computer networks (e.g., multisatellite networks) such knowledge is not readily available. More commonly, the network designer has a partial estimate of the network dynamics, in a structural form and a set of available parameters that describe the stochastic nature of the network, giving rise to the problem of finding minimum delay routes in stochastic network models, which is the subject of this paper.

Shortest path problems for stochastic models of dynamic networks were investigated in the past. In [12] a network is considered in which links may be "up" or "down" according to a stochastic rule. Each operational link has a fixed delay, and the problem is to travel with the least expected delay, allowing possible "detours". It is shown that the general problem is complex, and a solution for a simpler problem is given. In [13] a network is considered which is both random and time-dependent. It is shown that the optimal route is not a simple path but must be constructed as an adaptive decision rule. Mirchandani et al. investigate several versions of the general problem[14,15,16], of which the most relevant to this work is [16], where the delays in the network change according to a *global* Markov chain: each state of the chain describes the delays of all links in the network. It is assumed that the current state of the network (and thus, the delays of *all* links) is known. Since each state describes the delays of all links, the number of states in the chain may be very large. A recent work [17] investigates a network model in which links fail or recover according to discrete Markov chains. In that work, the memory problem (to be addressed later in this paper) is avoided by considering the special cases in which either the network is directed and acyclic or the states of the Markov chains are independent (i.e. memoryless) from slot to slot.

In this paper we also assume that the stochastic evolution of the network is Markovian. However, there are some key differences between this paper and [16]. We assume no global chain but rather a separate Markov chain per node. This approach better reflects the dynamics of the network and gives a better insight to complexity issues. In addition, we do not assume global knowledge regarding the states of links; in a computer network it is unreasonable to assume that the routing information at every node includes the exact current state of all other nodes. Rather, we assume that the routing decisions at a node are based on the current state of links emanating from that node and on the statistics of other links. A case where the travel history is recorded in the message is also considered.

Our aim is to devise policies for traveling between two nodes with the least expected delay. Analysis of the problem reveals radical differences from the regular shortest path problem. The basic difference is that the solution of the stochastic problem is not in the form of a "path", but rather as an adaptive decision policy. This means that an optimal policy may well route messages in loops and through paths of unbounded length. Another remarkable property is that past information regarding a travel is beneficial in minimizing the expected delay.

The above properties make the general problem highly complex, and we prove that even when posed on a finite horizon and excluding travel history considerations, it is an NP-Hard problem. Nonetheless, efficient and tractable solutions do exist for some interesting cases. Most notably, when the stochastic nature of delays is a nodal property, we present a simple algorithm for finding an optimal policy. For other networks, we apply sub-optimal approaches and present a polynomial algorithm that finds an ε-optimal policy.

The paper is structured as follows. The formal model is presented in Section II. In Section III the minimum delay path problem in a stochastic environment is formulated, and its intractability is established. The case of nodal delays is considered in Section IV. Section V presents a sub-optimal algorithm. In order to facilitate the reading of this paper, proofs that are either too lengthy or too technical appear in the Appendix.

II. THE MODEL

The network is represented by a directed graph $G(V,B)$, where V is the set of nodes and $B \subseteq V \times V$ is the set of links. At any time a positive number is associated with each link representing the delay of traversing the link at that time. Time is divided into unit slots and we assume that link delay remains constant for the duration of the slot. The delay of each link evolves according to a homogeneous discrete Markov chain with finite number of states and known transition probabilities, and transitions may occur only at slot boundaries. In other words, associated with each state is a discrete and strictly positive number, representing the link delay during that slot.

Focusing on the node, we can combine the state of the links emanating from that node into a single *node state*. Thus, each node i can be characterized by a discrete Markov chain with states E_1^i, E_2^i,..., $E_{m_i}^i$ (where m_i is the number of states of node i); when node i is in state E_j^i, the delay of link $(i,k) \in B$ is given by $d_{ik}(j) \in \{0,1,2, \cdots\}$. We assume that chains of different nodes are independent. Note that we do allow dependencies between delays of links outgoing from the same node. Such possible dependencies are reflected in the *nodal* Markov chain. In order to incorporate a stay in node i, we add self-loops of the form (i,i), for which $d_{ii}(j) \equiv 1$. Given for each i is also the state transition probability matrix $\boldsymbol{P^i} = \{p_{jl}^i\}$, such that p_{jl}^i is the probability of a transition from state E_j^i to E_l^i at node i (at the beginning of a slot). For simplicity, and without loss of generality, we assume that $\forall i \quad m_i \equiv m$ and denote $M \stackrel{\Delta}{=} \{1,2,...,m\}$. The incorporation of link states into a single nodal state should not be confused with the nodal delays model discussed in Section IV: in the latter, the delay of a link is composed of a stochastic delay value which, at each state, is identical for all links emanating from the same node, plus possibly a constant (deterministic) delay value for each link; whereas in the general model presented herein, each link takes a delay value at each (nodal) state which is independent of those of other links and other states.

Although our interest in the problem stems from the field of computer communication networks, the results reported here are of interest in other types of networks as well. Hence our use of the generic terms "traveler" and "traveling policy" instead of "message" and "routing policy".

We assume that a traveler in the network who is currently at node i is able to measure the current state of the node, and thus knows the exact delays of outgoing links from node i in the current slot. The traveler cannot measure the state of any other node (except node i). Furthermore, we assume that at the beginning of his journey, the traveler has no knowledge regarding the initial state of any Markov chain in the network; thus, when steady-states exist, he initially assumes that chains are in their steady-states.

A *path* is a (finite or infinite) sequence of nodes $(v_0, v_1, ...)$ such that for all i $(v_i, v_{i+1}) \in B$ ($v_i = v_{i+1}$ corresponds to parking at a node for one slot).

Without loss of generality, we assume that travelers enter the network at the beginning of a slot, and that their departure from each node along their paths is at the beginning of a slot; thus, arrival times at nodes also occur at the beginning of slots.

A *realization* is a (finite or infinite) sequence of node-state pairs $((v_0, E_{j_0}^{v_0}), (v_1, E_{j_1}^{v_1}), \ldots)$, such that $(v_0, v_1, \ldots)$ is a path and in addition for all i, k, $k > i$, such that $v_i = v_k = v$, and for all t_0, the following holds:

$Pr\{v \text{ in state } E_{j_k}^{v} \text{ at time } t_0 + \sum_{l=i}^{k-1} d_{v_l v_{l+1}}(E_{j_l}^{v_l}) \mid v \text{ in state } E_{j_i}^{v} \text{ at time } t_0\} > 0;$

in other words, a realization describes a possible sequence of node-state pairs that a traveler may encounter while traveling in the network. The pair (v, E_j^v) for some $v \in V$ and $j \in M$ is called a *realization component*. The set of all realizations is denoted by R.

III. PROBLEM FORMULATION AND TRACTABILITY ANALYSIS

In order to reduce traveling delays, the traveler should be allowed to adapt his routing decisions while traveling, rather than having to fix his entire route beforehand. The example depicted in Figure 1 makes this point clear. In this network $m = 2$ and all links but $(2,3)$ and $(2,4)$ have a deterministic delay of unity. Link $(2,3)$ has a deterministic delay of 2 units. For link $(2,4)$ we have in state E_1^2: $d_{24}(1) = 1$, and in state E_2^2: $d_{24}(2) = 10$. The transition probabilities are $p_{ij}^2 = 0.5$ for $i, j \in \{1,2\}$. Consider the problem of traveling from node 1 to node 5. Since the expected delay through link $(2,4)$ is 5.5 it is clear that the path with minimal expected delay is $(1,2,3,5)$, with a deterministic delay of 4 units. However, if upon arrival at node 2 the traveler measures the state to be E_1^2 then, if he is allowed to change his decision, he should go along link $(2,4)$, decreasing the delay to 3 units.

To cope with such circumstances it behooves us to define a *traveling policy* as a mapping from node-state pairs into nodes:

$$tp : V \times M \rightarrow V$$

A traveling policy dictates which node a traveler should turn to for each state he measures at each node in his journey. The policy may also make the traveler stay at a node for an additional slot(s). We denote by $tp(i,j)$ the identity of the next node when measuring state E_j^i at node i. A *proper* traveling policy *tp* is one for which for all i, j $(i, (tp(i,j))) \in B$ (note that, due to the self-loops, $(i,i) \in B$, and thus it is possible that $tp(i,j) = i$). We shall consider only policies that are proper.

For a given source node v_s and a destination node v_d the *delay D(tp) of a policy tp* is defined as the expected delay (over all realizations) of a traveler that leaves node v_s and travels according to *tp*, up to his arrival at v_d. Note that, in general, a journey according to a *tp* may result in an infinite path for certain realizations. For example, consider the network depicted in Figure 2 in which the delay of link $(i,3)$ $(i = 1,2)$ is 1 in state E_1^i and 10 in state E_2^i. All transition probabilities are equal to 0.5. Let the source node be 1 and

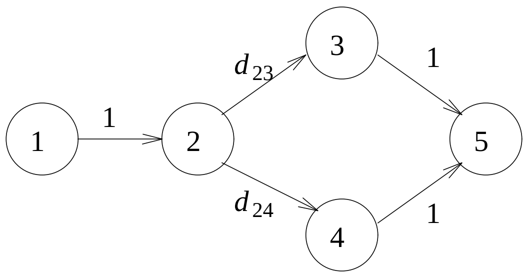

Figure 1: Advantage of Adaptive Routing

the destination be node 3.

It is easy to verify that the "best" *tp* is to move to node 3 in the state E_1, and in state E_2 either to stay at the node or else to move to the other node. Assume that the realization is such that the traveler always measures the state E_2, in both nodes 1 and 2. In this case he will either travel forever between the two nodes or else he will stay forever at the same node, resulting in both cases in an infinite path. Note, however, that the probability of this realization is 0, and the infinite delay incurred by it does not affect the delay of the policy.

To avoid an infinite path one might prefer a traveling policy *tp* that guarantees arrival at the destination within a bounded number of link traversals. We thus define a *q-stage traveling policy* as a mapping from node-state-stage triplets into nodes, i.e.

$$tp^q : V \times M \times Q \rightarrow V$$

where $Q = \{1,2,..,q\}$, and q is the maximal number of link traversals allowed. We consider staying at a node for an additional slot as a traversal and thus such a travel policy bounds the traveling time. We say that the traveler is at stage s if he has $q-s$ traversals left (the journey begins at stage 0). Also, when computing the delay of a tp^q, a realization for which the traveler does not arrive at the destination in q stages is considered having an infinite delay.

So far we have not considered traveling policies that make use of past information regarding node states gained while traversing the network. As the next example shows, the traveler can improve his performance by remembering past measurements. Consider the network depicted in Figure 2 in which link $(i,3)$ $(i=1,2)$ has delay 1 in state E_1^i and delay 10 in state E_2^i. The state transition probabilities are $p_{11}^i = p_{22}^i = 0.99$, $p_{12}^i = p_{21}^i = 0.01$ which converge to steady state probabilities of 0.5 (i.e., both nodes are equally likely to be in any of the two states). For simplicity, we consider 3-stage policies ($q=3$). Let node 1 be the source, node 3 the destination, and assume that at the beginning the traveler measures state E_2^1. If he goes directly along link $(1,3)$ his delay is 10; if he stays and waits for state E_1^1 (and then leaves on $(1,3)$) then his expected delay is $1 + 0.01 \cdot 1 + 0.99 \cdot (1 + 0.01 \cdot 1 + 0.99 \cdot 10) > 10$; and if he leaves to node 2

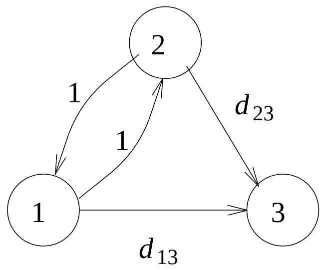

Figure 2: Example Network - infinite paths, advantage of memory.

his expected delay is at most $1+0.5\cdot 1+0.5\cdot 10=6.5$ because he will find node 2 in either of the two states with equal probability (the last calculation is the expected delay for a 2-stage policy, which is an upper-bound for that of a 3-stage policy). To optimize his delay the traveler should therefore move to node 2. Assume that once arrived at node 2 (having only 2 stages left) he measures the state E_2^2. Consider first the case where the traveler does not remember any previous measurement. If he departs immediately his delay is 10; if he waits (for at most one slot) at node 2 his expected delay is $1+0.99\cdot 10+0.01\cdot 1>10$; and if he goes to node 1 his expected delay is $1+0.5\cdot 10+0.5\cdot 1=6.5$ (these are in addition to the 1 slot delay incurred while traversing link $(1,2)$). Under these circumstances the best decision is to return to node 1. However, if he does remember previous measurements, then he realizes that the probability of finding node 1 again in state E_2^1 is $p_{22}^1\cdot p_{22}^1+p_{21}^1\cdot p_{12}^1=0.9802$ and therefore returning to node 1 he will incur an additional delay of $1+0.9802\cdot 10+0.0198\cdot 1>10$. Going directly along $(2,3)$ is therefore the preferred move.

The above example shows that the best *tp* must consider previous measurements. Denoting the set of (partial) realizations by *R*, such a *tp* will be denoted by $\tilde{tp}$, and is defined as

$$\tilde{tp}: V\times M\times R\to V$$

i.e., $\tilde{tp}(v,j,r)$ is interpreted as the next node along the path, where *v* is the current node, *j* is the index of the state currently measured, and $r\in R$ is the realization observed so far.

Similarly, we define a *q*-stage $\tilde{tp}$ by:

$$\tilde{tp}^q: V\times M\times R\times Q\to V$$

The sets of all possible traveling policies *tp*, tp^q, $\tilde{tp}$ and $\tilde{tp}^q$ shall be denoted, correspondingly, by *TP*, TP^q, $\tilde{TP}$ and $\tilde{TP}^q$.

We are now ready to define several problems:

- *The memoryless delay minimization problem (MDM).* Given a source node v_s and a destination node v_d, find a traveling policy $tp^*\in TP$ such that for all $tp\in TP$: $D(tp^*)\le D(tp)$.
- *The (general) delay minimization problem (DM).* Given a source node v_s and a destination node v_d, find a traveling policy $\tilde{tp}^*\in \tilde{TP}$ such that for all $\tilde{tp}\in\tilde{TP}$: $D(\tilde{tp}^*)\le D(\tilde{tp})$.
- *The q-stage memoryless delay minimization problem (QMDM).* Given a maximal stage value *q*, solve problem MDM within *q* stages.
- *The q-stage delay minimization problem (QDM).* Given a maximal stage value *q*, solve problem DM within *q* stages.

The policies tp^*, $\tilde{tp}^*$, tp^{q*} and $\tilde{tp}^{q*}$ shall be referred as *optimal* policies, each with respect to the corresponding problem.

We proceed to analyze the tractability of the above problems. We have seen that problems *DM* and *MDM* may need solutions posed on infinite time horizons, and thus they are in general intractable. We shall now show that *QDM* is likely to be intractable as well.

Theorem 1: Problem *QDM* is NP-Hard.

Proof: See the Appendix.

□

The intractability of *QDM* is typical of problems that require memorizing the entire history. It might be hoped that this would not be the case with *QMDM*, since in that problem the traveler forgets his measurement as soon as he leaves a node. Unfortunately, *QMDM* is also NP-Hard. In order to realize it, one should note that for a policy to be optimal (w.r.t. *QMDM*) it should consider the effect of the current measurement on future network states. For example, consider again Figure 2. Suppose that the traveler is in node 1 and measures the state E_2^1; when he considers a traversal to node 2, he should take into account that there he may measure the state E_2^2 and that as a consequence he may then return to node 1; however, since in the present time node 1 is in state E_2^1, the traveler knows that after that round-trip through node 2 the present node will very probably be in the same state (since $1 - p_{21} \cdot p_{11} - p_{22} \cdot p_{21} = 0.9802$), and thus going to node 2 is beneficial only if state E_1^2 will be measured there. In other words, for making his present decision he should "simulate" his future decisions in node 2 for every possible state there; although that simulation is memoryless in the sense that future decisions do not incorporate the present measurement, this measurement should be accounted for when evaluating the future performance; i.e, although the traveler is memoryless, he needs in fact to "remember" a lot when considering his future stages. This discussion is formalized by the following theorem.

Theorem 2: Problem *QMDM* is NP-Hard.

Proof: See the Appendix.

□

IV. NETWORKS WITH NODAL DELAYS

Assume that the stochastic nature of delays in the network is a nodal property, i.e.: the delay of a link at each (nodal) state is the sum of a stochastic (state-dependent) component, which is common to all links emanating from the same node, and of a deterministic (possibly zero) value, which varies from link to link. In other words, for every link (i,k) and for every state E^i_j, $d_{ik}(j) = d_i(j) + d_{ik}$, where $d_i(j)$ is the value of the random variable d_i at state E^i_j and d_{ik} is a deterministic value. For simplicity, we shall also assume that a link cannot have zero total delay, i.e.: $\forall\ (i,k) \in B,\ j \in M:\ \ d_i(j) + d_{ik} > 0$. For such a network, we show in this section that Problem *DM* is tractable. Define the *weight* of link (i,k) to be $\hat{d}_i + d_{ik}$, where $\hat{d}_i$ is a constant value to be defined in the sequel. We now show that an optimal *DM* policy for a network with nodal delays is one that navigates the traveler through a minimum-weight path, according to a certain departure-time decision strategy. We prove the above in two phases. First, we show that the minimum-weight path should be chosen independently of the departure-time decision and then we derive the departure times for the selected path.

The key-point in our proof is to show that the optimal traveler should never return to a previously visited node. Intuitively, a traveler may wish to return to a previously visited node when he realizes that the state of a link in his journey is worse than was expected, and a different path should be tried; however, in a network with nodal delays, the stochastic state affects equally all links emanating from a node, and thus the traveler incurs the same cost for moving to a previously visited node and moving to an unvisited node which will lead him closer to the destination. Obviously, this argument is far from satisfactory, since a *DM* policy is adaptive and may not choose entire paths beforehand, and it relies on the entire history as measured thus far and not only on the current measurement at a node. We proceed with a more convincing but still informal argument. A rigorous proof appears in the Appendix.

In general, a (optimal) traveling policy may indeed make the traveler return to a node that was previously visited. Nonetheless, there is at least one node from which such traversals are not possible -- the destination node v_d. Thus, backward movements (i.e., "away" from the destination) are possible from at most $|V| - 1$ nodes. Let *i* be a neighbor of the destination, for which the deterministic delay component of the link to the destination, d_{iv_d}, is minimal (among all neighbors of v_d). Suppose that the traveler is located at node *i* and measures the state E^i_j. The delay to the destination taking the direct link (i, v_d) is $d_i(j) + d_{iv_d}$ while the delay taking a path through any other neighbor, say through node *k*, will be at least $d_i(j) + d_{iv_d}$, since the delay on link (i,k) is at least $d_i(j)$, and the delay on any link that ends at the destination is, at all times, at least d_{iv_d}. We conclude that, when in node *i*, the traveler leaves only directly to the destination, meaning that there never are backward movements from node *i*. We are thus left with at most $|V| - 2$ nodes that may allow backward movements. We can now "contract" the network by merging v_d and i into a single new destination node, such that the new network is "equivalent" to the former, from the optimal *DM*-policy standpoint; such a contraction is made possible by the fact that any traversal through node i continues directly to the destination. We then apply the above argument to the new network, and identify another node which never allows backward movements. Repeating this procedure, we conclude that nowhere in the network will backward movements be made.

Lemma 1: In a network with nodal delays, a traveler using a *DM*-optimal policy does not return to a node which was previously visited.

Proof: See the Appendix.

□

Note that the lemma states that revisiting previously visited nodes is excluded by the optimal policy in *every* (nonzero probability) realization.

In fact, the proof of Lemma 1 shows that there is a *DM*-optimal policy that avoids returning to previously visited nodes even when zero delay values are allowed (see Appendix); however, in that case there may be other *DM*-optimal policies which do visit nodes more than once, since one might move around a loop of zero delay, without affecting optimality. It should be noted that the proof makes implicit use of the lack of correlation between nodes. When such a correlation exists, the above lemma may not hold (e.g., a small round-trip through a few nodes might reveal all that is necessary about future delay values at all nodes, making such a trip beneficial).

Suppose that the policy employed instructs a traveler situated at a node i that its next node is neighbor k, independently of the state measured at i and of the past history. Indeed, the next theorem shows that this is the case with an optimal *DM*-policy. With such a policy, it remains to decide at which state to leave node i. we are faced with a one-hop decision problem, namely: when to depart from the current node. We seek a departure time policy that minimizes the expected elapsed time from the arrival to the current node i and up to the arrival to the next node k. It is straightforward that the choice of the departure time depends solely on the stochastic component of the link delay. Thus, when calculating the departure policy we may ignore the deterministic component of link delays (i.e., in the following calculation we assume that $d_{ik} \equiv 0$). In order to specify the required policy, we define, for each node i, the values $\hat{d}_i(j)$ (for $1 \le j \le m$) through the following relations:

$$\hat{d}_i(j) = \min[d_i(j), 1 + \sum_{n=1}^{m} p_{jn} \cdot \hat{d}_i(n)]$$

Intuitively, $\hat{d}_i(j)$ is the expected delay, from the present slot and up to the arrival to the next node, of a traveler that is currently located at node i, measures state E_j^i and, departs only "when it is worthwhile". If a traveler measures state E_j^i and decides to wait, his expected delay is $1 + \sum_{n=1}^{m} p_{jn} \cdot \hat{d}_i(n)$; if he departs immediately his delay is $d_i(j)$. Thus, the traveler should depart only when $d_i(j) \le 1 + \sum_{n=1}^{m} p_{jn} \cdot \hat{d}_i(n)$. The above relations have a unique solution $\{\hat{d}_i(j)\}_{j=1}^{m}$ (see [18]). Consider a policy that instructs the traveler to depart in states j for which $\hat{d}_i(j) = d_i(j)$ and to wait otherwise; this policy is the optimal one we seek[18]. We call it the *Departure Policy*.

Next we show that the values $\hat{d}_i(j)$ can be calculated in a number of steps which is polynomial in *m*. We present the following iterative algorithm for making that calculation.

Departure Algorithm:

1. $n \leftarrow 0, \quad W_n \leftarrow \emptyset, \quad \forall j \ \ \hat{d}_i^0(j) \leftarrow d_i(j)$.
2. $n \leftarrow n+1$.
3. $W_n \leftarrow \{ j \mid 1 + \sum_l p_{jl} \cdot \hat{d}_i^{n-1}(l) < d_i(j) \}$.
4. $\forall j \notin W_n \ \ \hat{d}_i^n(j) \leftarrow d_i(j)$.
5. Solve the following system of nonhomogeneous linear equations (for $j \in W_n$):
$$\hat{d}_i^n(j) \ = \sum_{l \in W_n} p_{jl} \cdot \hat{d}_i^n(l) + C_i(j),$$
   where $C_i(j) \triangleq 1 + \sum_{l \notin W_n} p_{jl} \cdot \hat{d}_i^n(l)$.
6. If $W_n = W_{n-1}$ then stop.
7. Go to step 2.

The Departure Algorithm is a version of the approximation in policy space method [18]. It is known that the equations in step (5) have a unique solution, and that, for all *j*, $\hat{d}_i^n(j)$ converges to $\hat{d}_i(j)$ monotonically in *n*.

Lemma 2: The Departure Algorithm stops after at most $m+1$ iterations. Let *N* be the final value of *n*; then, $\forall j \ \ \hat{d}_i^N(j) = \hat{d}_i(j)$.

Proof: From the monotonicity of $\hat{d}_i^n(j)$ follows that $W_{n-1} \subseteq W_n$, for all *n*. Since W_n can have at most *m* elements, and since $W_n = W_{n-1}$ stops the algorithm, it follows that the algorithm runs for at most $m+1$ iterations.

Assume that, for some *j*, $\hat{d}_i^N(j) > \hat{d}_i(j)$. Suppose that we let the algorithm run indefinitely (i.e., we delete step (6)). Since *N* was the last iteration in the original algorithm, we have that $W_{N-1} = W_N$. Since the solution of the equations in step (5) is unique, it follows that $\hat{d}_i^N(j) = \hat{d}_i^{N-1}(j)$, and that for all $n > N$ we have $W_n = W_N$ and $\hat{d}_i^n(j) = \hat{d}_i^N(j)$; this is a contradiction, since $\hat{d}_i^n(j)$ converges monotonically to $\hat{d}_i(j)$.

Thus, $\hat{d}_i^N(j) \leq \hat{d}_i(j)$, for all *j*. On the other hand, $\hat{d}_i(j)$ corresponds to an optimal departure policy, and $\hat{d}_i^N(j)$ corresponds to *some* departure policy; thus, $\hat{d}_i^N(j) \geq \hat{d}_i(j)$. We conclude that $\hat{d}_i^N(j) = \hat{d}_i(j)$.

□

Since each iteration consists of $O(m^3)$ steps, we have:

Corollary: The Departure Algorithm stops after $O(m^4)$ steps.

□

Let $\hat{d}_i$ be the steady-state expected value of $\hat{d}_i(j)$ (over all *j*). Assign to every link $(i,k) \in B$ a weight $w_{ik} \stackrel{\Delta}{=} \hat{d}_i + d_{ik}$. We then have:

Theorem 3: In a network with nodal delays, a policy that instructs the traveler to move to the next node on a minimum-weight path to the destination according to the Departure Policy is an optimal *DM*-policy.

Proof: Denote by MW_i the minimal weight-distance from a node $i \in V$ to the destination. The proof is by induction and the inductive step is as follows. Suppose we proved that the claim is true for the *n* nodes whose weight-distance from the destination is at most the *n*-th lowest, and denote that set of nodes by *S*.

Suppose that the traveler visits for the first time some node $u \notin S$, from where he moves to a neighbor $v \in S$. Due to the inductive assumption, it is easy to see that, prior to measuring the state(s) of node *u*, such a travel incurs an expected delay of at least $\hat{d}_u + d_{uv} + MW_v = w_{uv} + MW_v \geq MW_u$. The argument $\hat{d}_u$ in the previous expression is due to the optimality of the Departure Policy for one-hop departure time decisions; an expected value is taken since node *u* was not visited before.

Let *i* be a node having the $n+1$'st lowest minimal weight-distance from the destination. We shall prove the inductive step by showing that the claim holds also for node *i*. Consider a traveler arriving to node *i* for the first time. Recall that, by Lemma 1, the traveler will not return to nodes that were previously visited. Denote by *j* the state at which the traveler departs from *i* (according to the optimal departure policy). Node *i* has some neighbor *k* such that $k \in S$ and $w_i + MW_k = MW_i$. Due to the inductive assumption, it is easy to see that a travel starting at node *i* (at the *j*-th state) and continuing through neighbor *k*, incurs an expected delay of $d_i(j) + d_{ik} + MW_k = MW_i + d_i(j) - \hat{d}_i$. It is easy to verify that every travel from node *i* to the destination must include a traversal from a node $u \notin S$ to a node $v \in S$. Due to Lemma 1, node *u* was not visited before. If $u \neq i$, then, according to the observation in the previous paragraph, the expected delay is at least $d_i(j) + MW_u$. According to the choice of *i*, we have that $MW_u \geq MW_i$, meaning that the travel beginning with neighbor *k* has a smaller expected delay. Thus $u = i$, and the inductive claim follows immediately.

We note that the base of the induction applies trivially to the destination node, and thus the theorem follows.

□

The above discussion and lemmas show that, for nodal delays, we have a *DM*-optimal policy that can be found in $O(|V|^2 + |V| \cdot m^4)$ steps.

V. A SUB-OPTIMAL SOLUTION

Ergodic Markov chains usually converge fairly rapidly to their steady state. We rely on that property (assuming that our chains are indeed ergodic) to obtain a solution to the *QDM* problem that is arbitrarily close to the optimal solution. We first present an optimal (and intractable) algorithm for obtaining a *QDM*-optimal policy and then modify it to a tractable, ε-optimal one. The algorithm is based on Dynamic Programming.

We begin with some notations. Let a realization history, r_n, be a sequence of up to n consecutive components observed by a traveler traversing the network according to some policy. Let R_n denote the set of all possible such realization histories. Let d be such that $d_{ik}(j) \le d$ (for all i,k,j). Let $p(i, E^i_j \mid r_n)$ be the probability of our traveler observing node i being in state E^i_j when r_n is his recorded history. Finally, denote by N_i the set of node i's neighbors (including node i itself, due to the self-loop). Recall that v_d is the destination node.

The algorithm uses several internal variables as follows. Consider a traveler located at some node i having passed n stages ($0 \le n \le q$) on its way to the destination node v_d and having recorded r_n as his realization history. For such a traveler $L^{(n)}_i(r_n)$ is the least expected delay to the destination (as known thus far in the execution of the algorithm). Similarly, $L^{(n)}_i(r_n, j)$ is the least expected delay (as known thus far) for a traveler under the same conditions and that is currently measuring state E^i_j. Finally, $L^{(n)}_{ik}(r_n, j)$ is the least expected delay (as known thus far) for a traveler under the previous conditions, and whose next move is to node k.

In the above, we implicitly assume that $L^{(n)}_i(r_n)$ is defined only for those r_n's which do not prohibit the traveler to be at that node at the n-th stage; similarly, it is implicitly assumed that $L^{(n)}_i(r_n, j)$ and $L^{(n)}_{ik}(r_n, j)$ are defined only for those r_n's which make it possible for the traveler to be at i at that stage and measure the state E^i_j.

The algorithm consists of q iterations, each representing a traversal stage, starting with the last one (i.e., stage q). At each iteration, the algorithm considers all nodes $i \in V$ at which the traveler may be found, all possible measurements E^i_j, and all possible realization histories r_n. In the description of the algorithm we use the operator "&" to signify the concatenation of a realization component to a history realization record.

Optimal QDM Algorithm:

1. Initialization: for all $i \in V$, $i \neq v_d$: $L_i^{(q)}(r_q) \equiv \infty$; for all $0 \leq n \leq q$: $L_{v_d}^{(n)}(r_n) \equiv 0$.
   $n \leftarrow q-1$.
2. For all $i \in V$, $i \neq v_d$, for all r_n:
   a. for all $j \in M$:
      i. $\forall k \in N_i$: $L_{ik}^{(n)}(r_n, j) \leftarrow d_{ik}(j) + L_k^{(n+1)}(r_n \& (i, E_j^i))$.
      ii. $L_i^{(n)}(r_n, j) \leftarrow \min_k L_{ik}^{(n)}(r_n, j)$.
   b. $L_i^{(n)}(r_n) \leftarrow \sum_{j=1}^{m} L_i^{(n)}(r_n, j) \cdot p(i, E_j^i | r_n)$.
3. $n \leftarrow n-1$.
4. If $n \geq 0$ then go to step (2).

In the first step of the algorithm we consider a traveler who has already passed q stages. Obviously, if he did not arrive at the destination, then his expected delay is infinite. On the other hand, the incremental delay of a traveler that arrives to the destination is zero, at all stages. The next steps consider previous stages in backward order. In step (2.a.i) we consider a departure from node i to node k, when the past measurements are recorded by r_n and the present state is E_j^i; here we make use of our previous computation for node k, stage $n+1$, and the history recorded by $r_n \& (i, E_j^i)$. Using this result, we identify in step (2.a.ii) the best decision to make under those circumstances. In step (2.b) expectation is taken over all possible states of node i, where the probabilities consider the history recorded.

Using standard Dynamic Programming techniques[19], one can show that the above algorithm indeed solves the *QDM* problem. The optimal policy for a traveler located at node i after n stages, whose past measurements are recorded by r_n and whose present measurement is E_j^i, is to move to a node k such that $L_i^{(n)}(r_n, j) = L_{ik}^{(n)}(r_n, j)$; if $k = i$, then the traveler should stay at node i. The final value of $L_i^{(n)}(r_n)$ is indeed the minimal expected delay to the destination of a traveler located at node i that passed n stages and whose realization history is recorded by r_n.

We proceed to count the number of operations of this algorithm. In step (2) we consider all r_n's (for $0 \leq n \leq q-1$), and their number is $O((m \cdot |V|)^n)$. For each r_n we perform $O(m \cdot |B|)$ operations in step (2.a). Step (2.b) involves the calculation of $p(i, E_j^i | r_n)$ for each node i, state E_j^i, and realization r_n; each such calculation involves up to $n \cdot d$ multiplications of an $m \times m$ probability matrix; thus, we have $O(|V| \cdot m \cdot n \cdot d \cdot m^3)$ operations for each r_n in step (2.b). The total number of operations of this algorithm is $O(m \cdot |B| \cdot \sum_{n=0}^{q-1} (m \cdot |V|)^n + m \cdot |V| \cdot d \cdot m^3 \cdot \sum_{n=0}^{q-1} n \cdot (m \cdot |V|)^n) = O((m \cdot |V|)^q \cdot (\frac{|B|}{|V|} + d \cdot m^3 \cdot q))$, which is intractable for a large number of stages. We point out that the length of r_n can be restricted to at most

$|V|-1$, since it is sufficient to record just the last visit at each node. This means that the value of q in the above complexity expression could be replaced with $min(q,|V|-1)$; nonetheless, the resulting complexity is still intractable.

We shall now modify this algorithm to obtain an efficient algorithm for finding an ε-optimal *DM*-policy. It is here that we rely on the convergence of Markov chains to their steady-states. The idea is that, since convergence is expected to be fast, the traveler can forget his oldest measurements, with only a negligible degradation in his performance.

For a given integer *d*, we restrict ourselves to networks whose maximal link delay value is *d*, that is, $d_{ik}(j) \le d$ (for all i,k,j). We also restrict ourselves to the class of (finite) Markov chains that are irreducible and aperiodic (and thus also ergodic), and assume that each transition probability matrix $\boldsymbol{P^i}$ has *m* linearly independent left eigenvectors. Such chains fulfill the following properties[20]: Each $\boldsymbol{P^i}$ has *m* eigenvalues, the dominant one (i.e., the unique maximal one) being $\lambda_0^i = 1$. Among all non-dominating eigenvalues of $\boldsymbol{P^i}$, let λ_1^i be one of maximal absolute value. Let $\boldsymbol{\pi^i}$ be the steady-state (row) probability vector of $\boldsymbol{P^i}$; let $\boldsymbol{x_n^i}$ and $\boldsymbol{y_n^i}$, for $1 \le n \le m$, be, respectively, the left- and right-eigenvectors of $\boldsymbol{P^i}$ ($\boldsymbol{x_n^i}$ is a row vector, and $\boldsymbol{y_n^i}$ is a column vector). Finally, let $\boldsymbol{x_0}$ be any (row) probability vector, i.e., a vector of order *m* with nonnegative components whose sum is 1. Then, the following relation holds [20]:

$$\|\boldsymbol{x_0}(\boldsymbol{P^i})^{\boldsymbol{k}} - \boldsymbol{\pi^i}\| \le |\lambda_1^i|^k \sum_{n=2}^{m} \|\boldsymbol{x_0}\boldsymbol{y_n^i}\boldsymbol{x_n^i}\| , \qquad (1)$$

meaning that the convergence of the *i*-th chain to its steady-state is not slower than geometric, with a rate of at least $|\lambda_1^i|$ ($\|\cdot\|$ denotes the standard Euclidean norm on R^m).

Let $\lambda \overset{\Delta}{=} \max_i |\lambda_1^i|$. We shall consider a class of networks for which, for all $i \in V$, all possible $\boldsymbol{x_0}$ and for predetermined positive α and β, $\lambda < \frac{1}{(m \cdot |V|)^{\alpha}}$ and $\sum_{n=2}^{m} \|\boldsymbol{x_0}\boldsymbol{y_n^i}\boldsymbol{x_n^i}\| < \beta$. As we shall see, α and β influence the complexity of the ε-optimal algorithm. Thus, given a maximum allowable number of operations, these values classify the set of networks that can be considered by this algorithm.

Suppose that the traveler records only the most recent *l* history components ($l \le n$). If r_n is his entire history, then he records just $\hat{r}_n$, which is defined as the last *l* components of r_n (for $n \le l$ $\hat{r}_n = r_n$). It is easy to verify that, from (1) and from the assumptions made on the Markovian chains, the following relation holds:

$$\forall i \in V, \;\; \forall r_n: \quad |p(i,E_j^i|r_n) - p(i,E_j^i|\hat{r}_n)| < \beta \cdot \lambda^l . \qquad (2)$$

Consider now the following algorithm, which is identical to the *QDM* Algorithm, with the exception that r_n is replaced by $\hat{r}_n$ (the value of the corresponding *l* shall be determined in the following).

Epsilon-Optimal QDM (EQDM) Algorithm:

1. Initialization: for all $i \in V$, $i \neq v_d$: $\hat{L}_i^{(q)}(\hat{r}_q) \equiv \infty$; for all $0 \leq n \leq q$: $\hat{L}_{v_d}^{(n)}(\hat{r}_n) \equiv 0$.

   $n \leftarrow q-1$;

2. For all $i \in V$, $i \neq v_d$, for all $0 \leq n \leq q-1$, for all $\hat{r}_n$:

   a. for all $j \in M$:

      i. $\forall k \in N_i$: $\hat{L}_{ik}^{(n)}(\hat{r}_n, j) \leftarrow d_{ik}(j) + \hat{L}_k^{(n+1)}(\hat{r}_n^{\,-} \& (i, E_j^i))$.

      ii. $\hat{L}_i^{(n)}(\hat{r}_n, j) \leftarrow \min_k \hat{L}_{ik}^{(n)}(\hat{r}_n, j)$.

   b. $\hat{L}_i^{(n)}(\hat{r}_n) \leftarrow \sum_{j=1}^{m} \hat{L}_i^{(n)}(\hat{r}_n, j) \cdot p(i, E_j^i | \hat{r}_n)$.

3. $n \leftarrow n-1$.

4. If $n \geq 0$ then go to step (2).

In step (2.a.i), $\hat{r}_n^{\,-}$ stands for the deletion of the first component from r_n, for $n \geq l$; in this way, we keep $\hat{r}_n$ within the allowed maximal size.

Lemma 3: Assume that algorithms *QDM* and *EQDM* are run both on the same network, and consider the final values of their corresponding variables. Let $\eta \overset{\Delta}{=} \beta \cdot \lambda^{l+1}$. Then, for all $0 \leq n \leq q$, $i \in V$, r_n, $\hat{r}_n$:

$$\left|\hat{L}_i^{(n)}(\hat{r}_n) - L_i^{(n)}(r_n)\right| \leq m \cdot (|V|-1) \cdot d \cdot (q-n) \cdot \eta \cdot (1 + m \cdot \eta)^{q-n}.$$

Proof: See the Appendix.

□

The corresponding *EQDM* policy is as follows: the traveler keeps a record $\hat{r}_n$ of his past l measurements; when located at node i after n stages and when measuring the state E_j^i, he moves to a neighbor k for which $\hat{L}_i^{(n)}(\hat{r}_n, j) = \hat{L}_{ik}^{(n)}(\hat{r}_n, j)$. It is easy to verify that the final value of $\hat{L}_i^{(n)}(\hat{r}_n)$ is the expected delay of a traveler who uses the *EQDM* policy, is located at node i having passed n stages, and whose l-truncated history is $\hat{r}_n$. Thus, the last lemma shows that his expected delay will be at most $m \cdot (|V|-1) \cdot d \cdot q \cdot \eta \cdot (1 + m \cdot \eta)^q$ away from the optimal value (i.e., it is an additive error). Let us consider l's which are large enough so that $\eta < \frac{1}{q \cdot m}$ (in fact, we shall soon restrict ourselves to even larger l's). For such values of η we have

$$m \cdot (|V|-1) \cdot d \cdot q \cdot \eta \cdot (1 + m \cdot \eta)^q < m \cdot |V| \cdot d \cdot q \cdot \eta \cdot (1 + \frac{1}{q})^q < m \cdot |V| \cdot d \cdot q \cdot \eta \cdot e \quad .$$

Thus, for an ε-optimal policy, we require that $m \cdot |V| \cdot d \cdot q \cdot \eta \cdot e \leq \varepsilon$. Since $\eta = \beta \cdot \lambda^l$, the required size of l is

$$l \geq \frac{\log\left(\frac{\varepsilon}{e \cdot \beta \cdot m \cdot |V| \cdot d \cdot q}\right)}{\log(\lambda)} - 1.$$

We proceed to count the number of operations of the EQDM Algorithm. In step (2) we consider all $\hat{r}_n$'s (for $0 \le n \le q-1$), and their number is $O((m \cdot |V|)^l)$. For each $\hat{r}_n$ we perform $O(m \cdot |B|)$ operations in step (2.a), thus having a total of $O(q \cdot m \cdot |B| \cdot (m \cdot |V|)^l)$ operations for step (2.a). Step (2.b) involves the calculation of $p(i, E_j^i | \hat{r}_n)$, for each node i, state E_j^i, and realization $\hat{r}_n$; each such calculation involves up to $l \cdot d$ multiplications of an $m \times m$ probability matrix; the values of $p(i, E_j^i | \hat{r}_n)$ (for all i,j, and $\hat{r}_n$) can be precalculated (instead of calculating them again for each n), needing a total number of $O(m^3 \cdot d \cdot l \cdot (m \cdot |V|)^{l+1})$ operations for this calculation. Thus, the number of steps performed by the *EQDM* Algorithm is

$$O((m \cdot |V|)^{l+1} \cdot (q \cdot \frac{|B|}{|V|} + m^3 \cdot d \cdot l))$$

This, together with the above expression for the required size of l, leads to the following result:

Lemma 4: The complexity of the EQDM Algorithm is

$$O\left[(q \cdot \frac{|B|}{|V|} + m^3 \cdot d \cdot MIN(|V|, \frac{log(\frac{\beta \cdot m \cdot |V| \cdot d \cdot q}{\varepsilon})}{\alpha \cdot log(m \cdot |V|)})) \cdot MAX(m \cdot |V|, (\frac{\beta \cdot m \cdot |V| \cdot d \cdot q}{\varepsilon})^{\frac{1}{\alpha}})\right]$$

Proof: See the Appendix.

□

Since d, α and β are predetermined, the above expression is polynomial in m, $|V|$, q, and ε. Thus, for the class of networks considered in this subsection, the *EQDM* Algorithm provides a tractable ε-optimal solution for the *DM* problem.

In order to gain some insight into the last result, note that for a deterministic network we have $m = 1$, $l = 0$, and the number of steps for $q = |V|$ is $O(|V| \cdot |B|)$, similar to Ford's shortest path algorithm (see [21]); this expression is a lower bound for the complexity expression found above, that is achieved asymptotically for $\lambda \to 0$ (since then we can choose $\alpha \to \infty$). As can be expected, the above expression shows that the algorithm works harder in order to get a better approximation, i.e., for smaller values of ε. Also, a smaller λ implies a greater α, which results in less computations; intuitively, a smaller λ promises faster convergence, which allows us to keep a smaller track of past events. We note that ε is the additive deviation of the algorithm's outcome from the optimal value (rather than a value related to the ratio between the two). This means that small integer values of ε would be good approximations; e.g., $\varepsilon = 1$ means that the expected delay of the EQDM policy is higher than that of the optimal policy by a single delay unit.

The EQDM Algorithm serves as a general scheme for obtaining efficient and tractable solutions when convergence is fast enough. It is possible to improve its performance further by exploiting specific network properties. For example, consider a network whose topology is such that any node is at most within h hops away from the destination (in fact, this is always the case when h is the network diameter). For such a network it is possible to change the bound of $(|V| - 1) \cdot d$ used in the proof of Lemma 3 by $h \cdot d$; this means that in the complexity expression of Lemma 4 the component $|V|\cdot d$ can be replaced with $h\cdot d$. An example of the performance of the EQDM Algorithm is presented in the Appendix.

VI. CONCLUSION

Traveling with the least expected delay in a stochastic network turns out to be an extremely complicated problem in general. Nonetheless, this problem merits attention, since we have shown that careful analysis of the specific class of networks considered may lead to the design of tractable solutions that are optimal or almost-optimal. In particular, an efficient optimal solution can be found in networks with nodal stochastic delays. This is typical of high-speed networks, in which the dominant delay is incurred by nodes and not by the high-speed links.

In this paper we assumed that the underlying Markov chains are uncontrolled i.e., routing decisions do not effect link delays. This is a proper framework for the case in which we aim to route efficiently (i.e., considering all that is known on the network) a portion of traffic that does not have a major effect on the total network loads. More generally, the results of this paper could serve as a baseline for the investigation of the case in which the effect of routing decisions on the network performance is taken into account. This, in fact, was the case with static-deterministic networks, for which the establishment of efficient load-dependent routing algorithms benefited from prior knowledge on the load-independent case.

Due to the complexity of the problem, it is of interest to evaluate the performance of different heuristics. An initial step in this vein is done in [22], where a version of a regular shortest-path algorithm is discussed. The algorithm considered the measured delays of links emanating from the current node, while all other links in the network are assumed to be in their steady states. It is shown that, for irreducible and aperiodic Markov chains, the algorithm guarantees arrival to the destination with probability 1; however, for periodic chains a counter example is given. The investigation of various heuristics, as well as possible enhancements of the model (e.g., incorporation of semi-Markov processes) are important topics left for further research.

ACKNOWLEDGEMENT

We would like to thank Israel Cidon for some insightful comments regarding this paper.

APPENDIX

Proof of Theorem 1

Theorem 1: Problem *QDM* is NP-Hard.

Proof: Assume that there is some algorithm in the class *P* that, when given a source node, a destination node and a (finite) maximal stage value *q*, produces an optimal q-stage traveling policy. We now show how this algorithm can solve the Hamiltonian Path Problem, which is known to be NP-Complete [23].

Given is a graph $\hat{G}=(\hat{V},\hat{B})$, for which it is required to decide whether it contains a Hamiltonian path beginning at some node $\hat{v}_s \in \hat{V}$. We derive from $\hat{G}$ a new graph *G* in the following way: we add to $\hat{V}$ a fictitious node v_d, and join each $\hat{v} \in \hat{V}$ to v_d by a fictitious link; thus, we get $V=\hat{V} \bigcup \{v_d\}$, $B=\hat{B} \bigcup \{(\hat{v},v_d) | \hat{v} \in \hat{V}\}$. We define for each link in *B* the following stochastic link delay: all links, except those that end in v_d, have a deterministic delay of unity; the delay of every link that ends in v_d is described by a homogeneous Markov chain with two states, E_1, E_2, such that $d(1)=1$, $d(2)=D=2\cdot|\hat{V}|$, $p_{11}=p_{22}=1$, $p_{12}=p_{21}=0$. Recall that the traveler does not have any knowledge regarding the initial states of links, and therefore he assumes that a link is in each state with probability of 0.5.

We now solve the *QDM* problem in *G*, for source node $v_s=\hat{v}_s$, destination node v_d and $q=|V|-1$, obtaining an optimal policy $\tilde{tp}^{q*}$. It is straightforward that $\tilde{tp}^{q*}$ will instruct the traveler to move straight to v_d on the first occasion that he measures the state E_1 at a node. We now show that, as long as there is an unvisited node in the network, the traveler should refrain from departing to the destination on a link whose state is E_2.

Consider a traveler that uses some *QDM*-optimal policy $\tilde{tp}^{q*}$. Suppose the traveler is at a node *v*, whose link to the destination is in state E_2. Let *n* ($n \geq 1$) be the number of unvisited nodes in the network excluding v_d. Suppose, by contradiction, that $\tilde{tp}^{q*}$ instructs the traveler to take the direct travel on the link (v,v_d), whose delay is *D*. We now construct a new policy that is better than $\tilde{tp}^{q*}$. Since $n \geq 1$ there is some unvisited node *w*; the new policy directs the traveler to *w* and then to take the direct link (w,v_d) (regardless of the measurements at *w*). The delay of the voyage from *v* to *w* is at most $|\hat{V}|-1$. Since *w* is unvisited, the probability of finding it in each state is 0.5. Thus, the expected delay on link (w,v_d) is $½(1+D)$. We conclude that the expected delay of the travel to v_d through node *w* is $|\hat{V}|-1+½(1+D) < D$. Thus, the decision taken by $\tilde{tp}^{q*}$ is not optimal, which is a contradiction.

Therefore, in a realization for which the state of every visited node is E_2, $\tilde{tp}^{q*}$ will first carry the traveler along all nodes in $\hat{V}$. On the other hand, visiting a node twice is wasteful, since it is certain to be in state E_2. Thus, we conclude that in such a realization $\tilde{tp}^{q*}$ will carry the traveler along an Hamiltonian path, if there is one. Thus, in order to solve the Hamiltonian Path Problem in $\hat{G}$, we compute an optimal travel in G in $|V|-1$ stages, assuming that state E_2 is measured at each stage; the answer to the Hamiltonian Path Problem is positive iff the obtained path is Hamiltonian. □

Proof of Theorem 2

Theorem 2: Problem *QMDM* is NP-Hard.

Proof: We make the same reduction from the Hamiltonian Path Problem as in Theorem 1. We note that the optimal policy in the problem outlined in Theorem 1 (where memory is allowed) does not use memory at all, since it can choose the Hamiltonian path beforehand (given that it exists), and then instruct the traveler to go along the path until he measures state E_1 when he should go directly to the destination. This policy would therefore be also an optimal policy for the memoryless case. This means that tp^{q^*} finds a Hamiltonian path if there is one, and thus one can obtain an answer to the Hamiltonian Path Problem by simulating a travel of $|V|-1$ stages, in which all links to v_d are found in state E_2. We conclude that tp^{q^*} solves the Hamiltonian Path Problem, proving the theorem. □

Proof of Lemma 1

Lemma 1: In a network with nodal delays, a traveler using a *DM*-optimal policy does not return to a node which was previously visited.

Proof: We start with the following definition. For a given *DM* problem and a policy that solves it, a *cluster* is a set of nodes, such that the following hold:

1. There exists a nonzero probability realization for which the policy causes a traversal from each node in the cluster to some other node in the cluster. (In that realization, every node in the cluster is visited at least once and at least one node in the cluster is visited at least twice.)
2. For all nonzero probability realizations, after leaving the cluster for the first time (i.e., after the first traversal from a cluster node to a non-cluster one), the traveler is not instructed by the policy to visit cluster nodes anymore.

It is emphasized that the definition of a cluster depends on the source-destination pair considered (by the given problem) and on the traveling policy. Condition (1) states that the traveler may return to a previously visited node. Condition (2) defines the cluster corresponding to Condition (1) to be the largest possible one. In fact, it is easy to verify that whenever a policy instructs the traveler to move from a node to a previously visited neighbor, then these two nodes belong to a cluster. In other words, considering, for a given policy, all possible link traversals under any (nonzero probability) realization, results in a (possibly empty) sequence of clusters, interconnected by acyclic sub-graphs. Clearly, our aim is to prove that a *DM*-optimal policy does not have clusters.

Consider a traveler that uses a *DM*-optimal policy: such a policy must bring the traveler to the destination for any nonzero probability realization. Since the destination cannot participate in clusters (due to condition (1)), it follows that, for a *DM*-optimal policy, every cluster has at least one *escape node* v with at least one *escape state* E_j^v, such that if the traveler arrives to v having measured some realization r and then measures the escape state E_j^v, he is instructed (by the optimal policy) to move to an *escape neighbor* w, which is outside the cluster. Escape neighbors have an important property: after reaching them the traveler will never return to the previous cluster(s), regardless of the realization measured.

Denote by $D_v(r)$ the expected delay from node v to the destination, when using a *DM*-optimal policy and the measured realization history is r. Suppose that a traveler moves from a cluster to an escape neighbor w. Since previous nodes will not be visited anymore, and since the stochastic chains of different nodes are independent, we conclude that the history r cannot affect the travel from w and on, i.e., $D_w(r) \equiv D_w$. We stress that the last relation holds since there is no backwards traversal from w for *any nonzero probability realization*, otherwise, by definition, w would have been part of the cluster. We are now ready to proceed with the proof.

Assume that the lemma is violated. Thus, there is at least one cluster C. Denote by $\{v_1, v_{2,} \cdots, v_n\}$ the (nonempty) set of C's escaping nodes, and by $\{w_1, w_2, \cdots, w_n\}$ a corresponding set of escaping neighbors (we point out that an escaping node may have several such neighbors, and vice-versa; hence, it may be that $v_i = v_j$ or $w_i = w_j$ for $i \neq j$; however, we construct the two sets so that $v_i = v_j$ implies $w_i \neq w_j$, and thus the two sets are finite). When the traveler moves from an escaping node v_i to an escaping neighbor w_i, at, say, some state $E_j^{v_i}$, the expected remaining delay (to conclude the journey) at that point is $d_{v_i}(j) + d_{v_i w_i} + D_{w_i}$. Thus, the cost of making such an escape is at least $K_i \stackrel{\Delta}{=} d_{v_i w_i} + D_{w_i}$. Denote $K \stackrel{\Delta}{=} \min_i K_i$. Since the traveler must escape eventually, after entering C we have, for all nodes v in C and all possible realizations r, $D_v(r) \geq K$. Suppose that l is such that $K_l = K$. Since v_l is in C, there is at least one nonzero probability realization r that brings the traveler to v_l; let $E_j^{v_l}$ be the state measured then at v_l. We point out that r is not necessarily such that the traveler is instructed to escape at v_l (the escape condition depends on the history measured and on the state measured at the escaping node). If the traveler does escape (to w_l), then his expected delay is

$$d_{v_l}(j) + K_l = d_{v_l}(j) + K$$

and if he moves to any neighboring node u within the cluster, his expected delay is

$$d_{v_l}(j) + d_{v_l u} + D_u(r') \geq d_{v_l}(j) + K \ ,$$

where r' is the "prefix" of r which corresponds to the realization measured up to the arrival to u. From the last two relations it follows that, when arriving to v_l, an escape to w_l is at least as good as continuing the travel in the cluster, *regardless of the history and present state measured*. Moreover, the last relation holds with equality only if all the following hold: (1) $d_{v_i u}=0$; (2) u is an escaping node, say $u=v_i$, for which $K_i=K$; (3) for at least one state E_j^u, $d_u(\hat{j})=0$ (otherwise, when escaping from u we always pay something for the nodal delay). Repeating the above discussion, but considering u as the escaping node, we again discover that regardless of history and state measured, escaping from u is at least as good as moving to any neighboring cluster node u'; however, equality here would again imply $d_{uu'}=0$, meaning that $d_u(\hat{j})+d_{uu'}=0$, which is forbidden in our model.

We conclude that there is at least one node in the cluster C, from which it is *always* preferable to move out of the cluster. Since we assume that an optimal policy is used, whenever the traveler arrives to that node he never returns to the cluster, meaning that that node is not in the cluster, which is a contradiction.

□

Proof of Lemma 3

Lemma 3: Assume that algorithms *QDM* and *EQDM* are both run on the same network, and consider the final values of their corresponding variables. Then, for all $0 \le n \le q$, $i \in V$, r_n, $\hat{r}_n$:

$$\left|\hat{L}_i^{(n)}(\hat{r}_n) - L_i^{(n)}(r_n)\right| \le m\cdot(|V|-1)\cdot d\cdot(q-n)\cdot\eta\cdot(1+m\cdot\eta)^{q-n}.$$

Proof: By (backward) induction on n. The claim is trivial for $n=q$ (adopting the convention that in the initial setting "$\infty-\infty=0$"). Assuming truth for $n+1$, we prove the lemma for n. By the inductive assumption on $n+1$ we have:

$$\begin{aligned}\left|\hat{L}_{ik}^{(n)}(\hat{r}_n,j) - L_{ik}^{(n)}(r_n,j)\right| &= \left|\hat{L}_k^{(n+1)}(\hat{r}_n{}^{-}\&(i,E_j^i)) \;-\; L_k^{(n+1)}(r_n\&(i,E_j^i))\right| \le \\ &\le m\cdot(|V|-1)\cdot d\cdot(q-n-1)\cdot\eta\cdot(1+m\cdot\eta)^{q-n-1}.\end{aligned}$$

Thus, from step (2.a.ii) of both algorithms:

$$\begin{aligned}\left|\hat{L}_i^{(n)}(\hat{r}_n,j) - L_i^{(n)}(r_n,j)\right| &= \left|\min_k \hat{L}_{ik}^{(n+1)}(\hat{r}_n,j) \;-\; \min_k L_{ik}^{(n+1)}(r_n,j)\right| \le \\ &\le m(|V|-1)\cdot d\cdot(q-n-1)\cdot\eta\cdot(1+m\cdot\eta)^{q-n-1} \quad .\end{aligned} \tag{3}$$

Denote: $p_i(r_n,\hat{r}_n,j) \triangleq \min\left[p(i,E_j^i|r_n),\; p(i,E_j^i|\hat{r}_n)\right]$ and $L_i^{(n)}(r_n,\hat{r}_n,j) \triangleq \min\left[L_i^{(n)}(r_n,j), \hat{L}_i^{(n)}(\hat{r}_n,j)\right]$. From (2) we have

$$\begin{aligned}p(i,E_j^i|r_n) &\le p_i(r_n,\hat{r}_n,j) \;+\; \eta \\ p(i,E_j^i|\hat{r}_n) &\le p_i(r_n,\hat{r}_n,j) \;+\; \eta \quad .\end{aligned} \tag{4}$$

Similarly, from (3) we have:

$$
\begin{aligned}
L_i^{(n)}(r_n,j) &\le L_i^{(n)}(r_n,\hat{r}_n,j) + m\cdot(|V|-1)\cdot d\cdot(q-n-1)\cdot\eta\cdot(1+m\cdot\eta)^{q-n-1} \\
\hat{L}_i^{(n)}(\hat{r}_n,j) &\le L_i^{(n)}(r_n,\hat{r}_n,j) + m\cdot(|V|-1)\cdot d\cdot(q-n-1)\cdot\eta\cdot(1+m\cdot\eta)^{q-n-1} \quad .
\end{aligned} \tag{5}
$$

For the next transition we use the following relation: given positive numbers a,a',b,b' such that $0\le\tilde{a}\le min(a,a')$, $max(a,a')\le A$, $0\le\tilde{b}\le min(b,b')$, and $max(b,b')\le B$, it holds that $|a\cdot b-a'\cdot b'|\le A\cdot B-\tilde{a}\cdot\tilde{b}$. This, together with (5) and step (2.b) in both algorithms yield:

$$
\begin{aligned}
|\hat{L}_i^{(n)}(\hat{r}_n)-L_i^{(n)}(r_n)| &\le \sum_{j=1}^{m}\{[L_i^{(n)}(r_n,\hat{r}_n,j) + m\cdot(|V|-1)\cdot d\cdot(q-n-1)\cdot\eta\cdot(1+m\cdot\eta)^{q-n-1}]\cdot[p_i(r_n,\hat{r}_n,j) + \eta] - \\
&\quad - L_i^{(n)}(r_n,\hat{r}_n,j)\cdot p_i(r_n,\hat{r}_n,j)\} \le \\
&\le \sum_{j=1}^{m}[L_i^{(n)}(r_n,j)\cdot\eta + m\cdot(|V|-1)\cdot d\cdot(q-n-1)\cdot\eta\cdot(1+m\cdot\eta)^{q-n-1}\cdot p(i,E_j^i|r_n) + \\
&\quad + m\cdot(|V|-1)\cdot d\cdot(q-n-1)\cdot\eta^2\cdot(1+m\cdot\eta)^{q-n-1}]
\end{aligned}
$$

Since $L_i(r_n,j)$ is the optimal expected delay for a *DM* policy, it cannot be larger than $(|V|-1)\cdot d$, which is the maximal delay for a policy that always chooses any simple path to the destination. Thus:

$$
\begin{aligned}
|\hat{L}_i^{(n)}(\hat{r}_n)-L_i^{(n)}(r_n)| &\le \sum_{j=1}^{m}(|V|-1)\cdot d\cdot\eta + m\cdot(|V|-1)\cdot d\cdot(q-n-1)\cdot\eta\cdot(1+m\cdot\eta)^{q-n-1}\cdot\sum_{j=1}^{m}p(i,E_j^i|r_n) + \\
&\quad + m\cdot(|V|-1)\cdot d\cdot(q-n-1)\cdot\eta\cdot(1+m\cdot\eta)^{q-n-1}\cdot m\cdot\eta = \\
&= m\cdot(|V|-1)\cdot d\cdot\eta\cdot[1+(q-n-1)\cdot(1+m\cdot\eta)^{q-n-1}+(q-n-1)\cdot(1+m\cdot\eta)^{q-n-1}\cdot m\cdot\eta] = \\
&= m\cdot(|V|-1)\cdot d\cdot\eta\cdot[1+(q-n-1)\cdot(1+m\cdot\eta)^{q-n}] \le \\
&\le m\cdot(|V|-1)\cdot d\cdot(q-n)\cdot\eta\cdot(1+m\cdot\eta)^{q-n}
\end{aligned}
$$

completing the inductive step. □

Proof of Lemma 4:

Lemma 4: The complexity of the EQDM Algorithm is

$$
O\left[(q\cdot\frac{|B|}{|V|}+m^3\cdot d\cdot MIN(|V|,\frac{log(\frac{\beta\cdot m\cdot|V|\cdot d\cdot q}{\varepsilon})}{\alpha\cdot log(m\cdot|V|)}))\cdot MAX(m\cdot|V|,(\frac{\beta\cdot m\cdot|V|\cdot d\cdot q}{\varepsilon})^{\frac{1}{\alpha}})\right]
$$

Proof: We recall that the number of steps performed by the *EQDM* Algorithm is $O((m\cdot|V|)^{l+1}\cdot(q\cdot\frac{|B|}{|V|}+m^3\cdot d\cdot l))$ and that l can be chosen as $l = MAX(0, \lceil \frac{\log(\frac{\varepsilon}{e\cdot\beta\cdot m\cdot|V|\cdot d\cdot q})}{\log(\lambda)} - 1\rceil)$. For $l=0$ the lemma is immediate. For $l>0$ we have that the complexity of the algorithm is

$$O\left[(q\cdot\frac{|B|}{|V|}+m^3\cdot d\cdot\frac{\log(\frac{\varepsilon}{e\cdot\beta\cdot m\cdot|V|\cdot d\cdot q})}{\log(\lambda)})\cdot(m\cdot|V|)\cdot(m\cdot|V|)^{\frac{\log(\frac{\varepsilon}{e\cdot\beta\cdot m\cdot|V|\cdot d\cdot q})}{\log(\lambda)}}\right] =$$

$$= O\left[(q\cdot\frac{|B|}{|V|}+m^3\cdot d\cdot\frac{log(\frac{e\cdot\beta\cdot m\cdot|V|\cdot d\cdot q}{\varepsilon})}{log(\frac{1}{\lambda})})\cdot(m\cdot|V|)\cdot(\frac{e\cdot\beta\cdot m\cdot|V|\cdot d\cdot q}{\varepsilon})^{\frac{\log(m\cdot|V|)}{\log(\frac{1}{\lambda})}}\right] \le$$

$$\le O\left[(q\cdot\frac{|B|}{|V|}+m^3\cdot d\cdot\frac{log(\frac{\beta\cdot m\cdot|V|\cdot d\cdot q}{\varepsilon})}{\alpha\cdot log(m\cdot|V|)})\cdot(m\cdot|V|)\cdot(\beta\cdot m\cdot|V|\cdot d\cdot q\cdot\frac{1}{\varepsilon})^{\frac{1}{\alpha}}\right]$$

In the last inequality we use the assumption that $\lambda<\frac{1}{(m\cdot|V|)^{\alpha}}$. Since it is sufficient to record just the last visit at each node, we have $l\le|V|$ and the lemma follows. □

Example of EQDM Algorithm Performance

We consider a network for which $m_i \equiv 2$, $d_i(1) \equiv 1$, $d_i(2) \equiv 10$. The transition probabilities are: $p_{11} = p_{22} = 0.55$, $p_{12} = p_{21} = 0.45$, to which correspond $\lambda = 0.1$, $\beta = \frac{1}{\sqrt{2}}$ and an expected link delay of 5.5 units.

We make the following assumptions regarding the topology. The number of edges is $|B| \approx |V|^{1.5}$, i.e. an intermediate value between a sparse network and a complete one. We assume that a node is within at most $2 \cdot \sqrt{|V|}$ hops away from the destination; this, for example, is the case with a rectangular grid. Following the remark in Section V, in the complexity expression of Lemma 4 the component $|V| \cdot d$ can be replaced with $2 \cdot \sqrt{|V|} \cdot d$. A typical value of q would be between $2 \cdot \sqrt{|V|}$ (the minimal number of stages in order to reach the destination) and $|V|$(which is far more than what is usually needed in a network with the $2 \cdot \sqrt{|V|}$ property).

Let $|V| = 50$. Then, $|B| \approx 350$ and $2 \cdot \sqrt{|V|} \approx 14$. We choose $q = 35$, i.e. an intermediate value in the range $2 \cdot \sqrt{|V|} \cdots |V|$. Let $\varepsilon = 2$ i.e., less than half of the steady-state expectation of a link delay. We then have $l = 3$ i.e., the EQDM Algorithm works on the last 3 realization components. The number of operations of the algorithm is in the order of 10^{10}, which is reasonable for off-line computation. Note that the number of operations for the optimal QDM Algorithm is in the order of 10^{73}.

We proceed to estimate the performance of the policy obtained by the EQDM Algorithm. To that end we obtain an approximation for the performance of an optimal policy. Consider a policy that navigates the traveler through a minimum-hop path to the destination, each time waiting for the state 1 to come about. The expected delay of such a policy (considering a path of 14 hops) is approximately 29.5 delay units; the expected elapsed time per hop is, approximately, 2.1 delay units. Note that such a policy seems as a good approximation for a *DM* policy, in view of the high symmetry in the network. Also, note that this policy has no bounds on the number of stages, but since $q = 35 > 29.5$, it is a reasonable approximation for our q-staged case. With this approximation, the expected delay of the EQDM Algorithm would be at most $29.5 + \varepsilon = 31.5$. A standard shortest path algorithm (which considers just the expected link delays) has an expected delay of *77* delay units (5.5 per link, on a path of 14 links). Thus, we have

$$D(\text{"}opt\text{"}) \approx 29.5 < D(EQDM) \approx 31.5 < D(SP) = 77$$

This example shows how EQDM may yield efficient solutions within a tractable number of operations when the Markov chains converge rapidly. For chains with a slower rate of convergence we would need more computations or else do with a higher ε. For example, consider the above network but with chains corresponding to $\lambda = 0.2$. Choosing $\varepsilon = 6$ (i.e., approximately the expected delay of a link) yields $l = 4$ and an order of 10^{12} operations (again, compare with 10^{73} for the optimal algorithm). We now have

$$D("opt") \approx 29.5 < D(EQDM) \approx 35.5 < D(SP) = 77$$

In considering the tradeoff between the quality of the result and the number of operations one should note that the EQDM Algorithm lends itself easily to an implementation on a parallel machine (e.g., by assigning a processor to each network node).

REFERENCES

1. N. Deo and C.Y. Pang, ‘‘Shortest Path Algorithms: Taxonomy and Annotation,’’ *Networks* **14** pp. 275-323 (1984).
2. P.M. Merlin and A. Segall, ‘‘A Failsafe Distributed Routing Protocol,’’ *IEEE Trans. on Communications* **COM-27**(9) pp. 1280-1287 (September 1979).
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. A. Segall, ‘‘Distributed Network Protocols,’’ *IEEE Trans. on Information Theory*, pp. 23-34 (January 1983).
5. J.M. McQuillan, G. Falk, and I. Richer, ‘‘A Review of the Development and Performance of the ARPANET Routing Algorithm,’’ *IEEE Trans. on Communications* **COM-26**(12) pp. 1802-1811 (December 1978).
6. E. Klafszky, ‘‘Determination of Shortest Path in a Network with Time-Dependent Edge-Lengths,’’ *Mathematische Operationsforschung u. Statistik* **3** pp. 255-257 (1972).
7. 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).
8. S.E. Dreyfus, ‘‘An Appraisal of some Shortest Path Algorithms,’’ *Operations Research* **17** pp. 395-412 (1969).
9. 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).
10. 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).

11. 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).

12. G. Andreatta and L. Romeo, ‘‘Stochastic Shortest Paths with Recourse,’’ *Networks* **18** pp. 193-204 (1988).

13. R.W. Hall, ‘‘The Fastest Path through a Network with Random Time-Dependent Travel Times,’’ *Transportation Science* **20**(3) pp. 181-188 (August 1986).

14. A. Eiger, P.B. Mirchandani, and H. Soroush, ‘‘Path Preferences and Optimal Paths in Probabilistic Networks,’’ *Transportation Science* **19**(1,) pp. 75-84 (February 1985).

15. P.B. Mirchandani and H. Soroush, ‘‘Optimal Paths in Probabilistic Networks: a Case with Temporary Preferences,’’ *Comput. & Ops. Res* **12**(4) pp. 365-381 (1985).

16. P.B. Mirchandani and M.H. Veatch, ‘‘`Hot Job´ Routing Through a Stochastic Network,’’ *Large Scale Systems* **11**(2) pp. 131-148 (1986).

17. R.C. Ogier and V. Ruthenburg, ‘‘Minimum-Expected-Delay Alternate Routing,’’ pp. 617-625 in *Proceedings of Infocom’92*, Florence, Italy (6-8 May 1992).

18. H. Kushner, *Introduction to Stochastic Control*, Holt, Rinehart and Winston, Inc. (1971).

19. S.E. Dreyfus and A.M. Law, *The Art and Theory of Dynamic Programming*, Academic Press (1977).

20. D.L. Isaacson and R.W. Madsen, *Markov Chains Theory and Applications*, R.E. Krieger Publishing Company, Malabar, Florida (1976).

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

22. A. Orda, R. Rom, and M. Sidi, ‘‘Minimum Delay Routing in Stochastic Networks,’’ EE Publication No. 810, Faculty of Electrical Engineering, Technion, Haifa, Israel (November 1992).

23. M.R. Garey and D.S. Johnson, *Computers and Intractability*, W.H. Freeman and Company, San Francisco (1979).