Abstract—Timely and efficient message transmission through intermittently and sparsely connected networks is a problem of significant interest to the mobile networking community. Although the long-term statistics describing the time-varying connectivity in such networks can be characterized systematically and can be used for selecting good routes, it may be possible to achieve better performance by intelligently using the actual link states at the time of routing, in conjunction with these statistical dynamics models. In this paper, we investigate a family of minimum expected latency routing methods for such dynamic networks, spanning purely model-based and state-oblivious source routing; state-based source routing; and various flavors of dynamic (or hop-by-hop) routing, with increasing amounts of current link state knowledge around the source.

First, we give a heuristic and an approximation scheme for the model-assisted source routing problem, as well as heuristics for the dynamic routing problem. Then we show using extensive simulations on both synthetic and MIT Reality Mining traces that although dynamically sampling link states helps to improve expected routing latency compared to source routing, the marginal improvements decline rapidly for knowledge of current link states beyond 2 hops. To the best of our knowledge, this is the first thorough characterization of the performance of the entire spectrum of model-assisted routing algorithms ranging from little knowledge to complete knowledge of link dynamics.

I. INTRODUCTION

Routing in dynamic networks has been a popular topic of research for decades in the networking research community. This problem is relevant in a plethora of practical situations, such as wireless/mobile ad hoc networks (or MANETs), mesh or sensor networks, disruption tolerant networks (DTNs), and even transportation networks. In the case of wireless networks, network topology could vary over time due to link outages caused by node mobility, jamming, or interference, or due to node outages caused by duty cycling or, in the extreme case, battery exhaustion. In the case of transportation networks, the state of roads in a network changes over time, thus affecting the optimal routing of vehicles through the network. One version of this problem studied in the computer science literature is the Canadian Traveler Problem (CTP) [21], where drivers may experience random road blockages due to snowfall but discover the road is out only upon reaching it. The goal is to minimize the expected travel time.

In MANETs, shortest paths can be computed through networks by collecting topology information periodically or in an event-triggered manner. This can be done on demand by flooding followed by source routing [12] or reverse path forwarding [22], or proactively by flooding topology updates followed by each source computing a shortest path to destinations using standard shortest path algorithms [5], [24]. A chosen path can then be used until a link failure prompts its revision. Alternatively, routing can be done dynamically, through local broadcast [4] or repetition [25].

Our focus here is on “single-copy” store-and-advance routing schemes, which often apply to DTNs and transportation networks. We study a number of problem variants of this form.

Source vs. dynamic routing: Routing in dynamic networks is typically done hop-by-hop, the key benefit of which is responsiveness to topology changes. In our setting, this means either choosing a next-hop link or waiting for a better link to become available. An alternative approach is (store-and-advance) source routing, in which a complete, immutable path is chosen for a packet to travel, from source to destination, within a network [2]. One advantage of source routing is that once the routing path is chosen, no routing decisions need be made online. This is especially advantageous for networks where the end-hosts are resource-rich but the intermediate core is resource-poor in terms of computing power.

Link state information: Different levels of knowledge of actual link state (initial link state in the case of source routing; current link state in the case of dynamic routing, at the time of each next-hop decision.) may be considered. At one extreme, only local link state information is available, i.e., for the outgoing links from the current node (in the case of dynamic routing) or the source node (in the case of source routing). (Any link whose state is not known will be assumed to be in steady-state.) At another extreme, the states of all links may be known at all times. There are two sorts of intermediate settings: a) global knowledge of all link states but only in the first time step, and b) local knowledge, after each hop, of the links within some k-hop neighborhood of the current node.

It may be feasible to inform the source (and intermediate nodes) about changes in network topology over alternate communication channels. An example of this is a “hybrid” network where each node is equipped with two transceivers: a high data rate / low transmission range, intermittently connected transceiver to carry “data” (e.g., WiFi) and a lower data rate / high transmission range, always connected transceiver for “control” (e.g., cellular data). When the future topology changes known, latency can be minimized by running a modified Dijkstra algorithm [11]. Alternately, one can minimize estimated expected delay based on observed contact schedules [13].

Link dynamics model: In addition to current or historical link state information, routing decisions can be based on a suitable model of the underlying link dynamics. This allows probabilistically projecting the state of a link forward in
time, thus alleviating the need to flood current topology state information every time a link’s state changes.

The simplest model of link dynamics is the one in which each link is, at any given time, up independently with a certain probability \(p\) (also referred to as the dynamic Erdős-Rényi (ER) random graph model). \(p\) is assumed to be known in advance or learnt, and the current state of a link is observable by sampling or exchanging messages between endpoints of the said link. The ER random graph is a basic model for studying dynamic graphs, but it does not capture temporal correlation in link dynamics. A more powerful and accurate model, and the focus in this paper, is the \((p, q)\) dynamic discrete-time Markovian model. At each discrete time step in this model, a link is either on or off; between one time step and the next, the link may transition between these states, with known probabilities of transitioning from off to on (\(p\)) and from on to off (\(q\)).

**Contributions.** To the best of our knowledge, this sort of systematic exploration of minimum latency link state routing schemes (see Fig. 1) has not previously appeared in the literature. Our main contributions are:

1) We verify using the MIT Reality Mining connectivity traces that the Markovian \((p, q)\) link dynamics model for link dynamics is accurate.

2) For Markovian networks with global or \(m\)-hop state info, we give an efficient heuristic and a PTAS for source routing, and two heuristics for dynamic routing. We also show that for a number of interesting special cases of \((p, q)\), the routing problems can be solved optimally.

3) We perform an extensive evaluation of these algorithms on both synthetically generated time-varying networks and real networks, investigating these questions:
   - How beneficial is having link state info beyond the current/source neighborhood, and how far?
   - How much advantage does the flexibility of dynamic routing offer over source routing?

We find that both provide benefits, but the state info benefits rapidly diminish beyond 2 hops.

The results of this paper are directly applicable to multiradio the WiFi/Cellular hybrid networks mentioned earlier, although we do not study actual link state dissemination issues in intermittently connected networks. These networks are assumed to be lightly loaded, and thus the network congestion effects are assumed to be negligible. Our goal in this paper is not to provide a complete routing protocol but to investigate fundamental aspects of the aforementioned algorithmic tradeoffs.

II. RELATED WORK

See [1] and references therein on routing in DTNs. In the transportation literature, Hall has studied stochastic, time-dependent networks (STDNs) [10], where edge delays are time-dependent random variables (r.v.), and a standard routing goal is to minimize expected travel time. Because of the dynamic nature of the network, the solution concept considered there not a fixed path but a dynamic routing policy. Such problems have been studied assuming global state info [8], in which the global network state is known throughout the routing process, as well as under weaker assumptions about on state knowledge.

Lott and Teneketzis [15] give a model and algorithms for routing in stochastic ad hoc networks which are somewhat similar in spirit to our work, but the models are quite different. In their work, for example, the network state is specified by which nodes have received the message so far, and it is the transition of this state (i.e., whether an additional node receives the message) that is Markov, not the states of individual links.

The work closest to ours is by Ogier and Rutenburg [19]. They study the scenario where each dynamic link \(\ell\) is assumed to obey a two-state Markov chain with parameters \((p_\ell, q_\ell)\), which are known to the routing algorithm, and one can observe the state (i.e., on or off) of links in the current node’s neighborhood, while links beyond that are assumed to be in steady-state—our local state info setting. They showed that this problem is #P-complete, even in the special case of \(p_\ell = 1 - q_\ell\) with zero-delay edges. However, if \(p_\ell = 1 - q_\ell\) and edge lengths are nonzero or if the graph is directed and acyclic (DAG), then the problem is optimally solvable by Dijkstra-style dynamic programming. Similar results were recently given in the AI community by Nikolova and Karger [18] for the stochastic edge-delay version of the Canadian Traveler Problem (CTP) [3]. The (simpler) stochastic edge-state version of CTP (interpret off as infinite delay) is a PSPACE-hard [7] and is a special case of our dynamic (local state info) routing problem. In the special case where state is node-based rather than edge-based, however, the routing problem (with stochastic edge delays) is optimally solvable on general graphs because an optimal route will never reverse [20], [23].

A polynomial-time algorithm was given by Nain et al. [17] for the related problem of computing the expected traversal time (ETT) of a fixed path in the Markovian model with global state info. Our source routing problem is the problem of finding a best path in this sense within a general graph. A “state-budgeted” variant of this problem has been studied [9] in restricted graph structures. In the current paper we allow for general graphs.
III. MARKOVIAN NETWORKS

A. Definitions and Model

We begin with basic assumptions and concepts. Time is measured in discrete time steps. Time $\tau$ refers to the beginning of the time step $\tau$, for $\tau = 0, ..., T$.

Definition 1 (Dynamic graph): $G[0,T] = \{G_\tau = (V_\tau, E_\tau)\}_{\tau=0}^T$ is a dynamic (or time-varying) graph indexed by discrete time step $\tau$ if each “graphlet” $G_\tau$ is a valid graph. $V_\tau$ and $E_\tau$ denote the vertex and link sets, respectively, at time $\tau$. A dynamic graph may have a finite or an infinite number of graphlets.

Definition 2 (Underlying graph): $G_{UL} = (V_{UL}, E_{UL})$ is the underlying graph of a dynamic graph $G[0,T]$ if $V_{UL} = \bigcup_{\tau=0}^T V_\tau$ and $E_{UL} = \bigcup_{\tau=0}^T E_\tau$.

Definition 3 (Markovian $(p,q)$ graphs): At time 0, each link is in some state (which we may or may not assumed to be known). The state of a given link $i$ in subsequent time steps is governed by a known two-state Markov chain specified by the probability transition matrix $P_i$:

$$P_i = \begin{pmatrix} 1 - p_i & p_i \\ q_i & 1 - q_i \end{pmatrix},$$

with $p_i$ (resp. $q_i$) the probability that link $i$ transitions from state 0 (resp. state 1) into state 1 (resp. state 0) in one time step. Note that the nontrivial eigenvalue of $P_i$ is $\beta_i = 1 - p_i - q_i \in [-1,1]$, a quantity which will be very useful later.

Definition 4 (Steady-State probabilities): $\pi_0 = q_i/(p_i + q_i)$ and $\pi_1 = p_i/(p_i + q_i)$ are the stationary probabilities that link $i$ is in states off or on, respectively.

We note a few important special cases of $(p, q)$ and their resulting steady state probabilities:

- $(1, 1)$: Link states alternate deterministically in every time step, with $\pi_0 = .5$.
- $(p, 1 - p)$: The Erdős-Renyi setting, where a link’s state is independent of its previous state, with $\pi_0 = p$.
- $(p, 0)$: A “densifying network” where edges appear but never disappear.
- $(0, 0)$: Link states never change, which is consistent (in the unknown initial state setting) with any specified steady-state (i.e., initial) probability $\pi_0$.

Definition 5 (Transient probabilities): Let $P_{a,b}^\tau(\tau) = \Pr(X_\tau = b | X_0 = a)$ be the probability that link $\ell$ is in state $b \in \{0, 1\}$ at time $\tau$ given that it was in state $a \in \{0, 1\}$ at time $\tau = 0$ for initial state $X$. In cases where all links have the same parameters $p, q$, we omit the superscript $\ell$.

It is known [14] that for any $t \geq 0$:

$$P_{1,0}(t) = \pi_0(1 - \beta^t); \quad P_{1,1}(t) = \pi_1 + \pi_0\beta^t$$

$$P_{0,1}(t) = \pi_1(1 - \beta^t); \quad P_{0,0}(t) = \pi_0 + \pi_1\beta^t$$

Transmission assumptions. If the message reaches a link’s entry node when the link is on, then it immediately starts traversing the link; if the link is off, the message waits there until the link appears. For each link $\ell$ its delay $d(\ell)$ is the number of time steps it takes to traverse the link (when on), which is nonzero unless otherwise stated. We also assume that transmission finishes within a single time step and is not interrupted by a link-down event.

Notation. The global state of all links at time $t$ is indicated by $X(t)$, which represents an assignment of each edge of the underlying graph to a bit. The state (0 or 1) of a particular link $i$ at time $t$ is indicated by $X_i(t)$, sometimes written $X_i^t$ when notationally convenient.

B. Validation of $(p,q)$ Model

Although the $(p, q)$ model has been used before in networking literature [6], [16], we could not find an empirical study that attempts to validate the model. We now demonstrate that the model of dynamic graphs illustrated in Sect. III-A is realistic, by reference to the MIT Reality Mining connectivity traces. In this data set, there are 106 people who were given smartphones equipped with Bluetooth proximity sensors. We use these proximity traces as a proxy for an intermittently connected network.

We divide time into 6-minute slots and declare a link $(u,v)$ up in a time step $\tau$ if node $v$ was contained in $u$’s scan in time step $\tau$. We consider a dynamic intermittently connected graph over $T = 48000$ time steps (or 200 days between $\tau = 73200$ to $\tau = 732400$). For each link $\ell$ in the underlying graph, we analyze the time series of up/down events and attempt to estimate the parameters $(p_\ell, q_\ell)$. Let $n_{0\ell}$ and $n_{1\ell}$ denote the number of 0 – 1 and 1 – 0 transitions, respectively, during $T$ time steps. If the number of down time steps is given by $n_{0\ell}$ and number of up time steps is given by $n_{1\ell}$, then $p_\ell \sim n_{0\ell}/n_{1\ell}$ and $q_\ell \sim n_{0\ell}/n_{1\ell}$.

We verify the validity of this estimation by comparing the mean latency of the estimated $(p_\ell, q_\ell)$ dynamics model with the average “run length” of 0s before a 1 is encountered in the data. The mean latency is given by $1 + \frac{q_\ell}{p_\ell(1 + q_\ell)}$, whereas the run length is computed from the data. For a majority of links, these two values are close to each other; therefore we conclude (in the sense of the first moment) that the two-state Markovian model is reasonable for capturing the time-varying connectivity of a realistic data set such as Reality Mining.
Figure 3 shows the distribution of $p_\ell$ and $q_\ell$ values across the network. We observe that almost all links have a low value of $p$, which means that they are unlikely to come up, once down. This is unsurprising since this is a time-varying network with severe intermittent connectivity over a period of months. The values of $q$ on the other hand fall in two clusters: the first one contains all links with $q \approx 1$, i.e., links that may be up only once or twice in the entire data set. There are very few such links. The second cluster corresponds to links that are better connected with the rest of the network, with $q$ values appearing to be normally distributed.

Based on this exercise, we conclude that the Markovian dynamics model is a reasonable one for modeling real intermittent networks. We will use these dynamics model is a reasonable one for modeling real intermittent networks. We will use these dynamics to represent each node $(1)$ noted in Sect. III. All but the last global or local state info) appears to be normally distributed.

IV. SOURCE ROUTING IN MARKOVIAN GRAPHS

We are given an underlying graph $G_\cup$, initial link state configuration $G_0$, source node $s$, target node $t$, and dynamics model parameters $(p, q)$. Let $\mathcal{P}_{s,t}$ be the set of all simple paths from $s$ to $t$ in $G_\cup$. Then the objective of source routing is:

$$\arg\min_{P_{s,t} \in \mathcal{P}_{s,t}} ETT(P_{s,t})|G_0|$$

$ETT(P_{s,t})|G_0|$ is the expected traversal time in the $(p, q)$ model of a path $P_{s,t}$ with the initial link states given by $G_0$. This value can be computed in $O(n^2)$ time for an $n$-hop path [17]. The complexity remains $O(n^2)$ when $(p_\ell, q_\ell)$ vary by link $\ell$, as long as there is only a constant number of distinct $(p_\ell, q_\ell)$ pairs. Our algorithms hence also generalize in this way, but we assume uniform $(p, q)$ values for simplicity.

A. Special cases

We begin by sketching results for the special cases (with global or local state info) noted in Sect. III. All but the last are optimally solvable by running Dijkstra on the specified graph.

$(1,1)$ global: An expanded graph where two nodes $v_0, v_1$ represent each node $v$ of $G_\cup$, in even and odd timesteps, respectively.

$(1,1)$ local: A weighted graph where the weight of each edge $\ell$ is set to $d(\ell)$ if it is outgoing from $s$ and $t$, or to $d(\ell) + 1$ otherwise.

$(p, 1-p)$ local and global $(0 < p < 1)$: The weighted graph $G_w$, which, for each edge $\ell$ of $G_\cup$, has a corresponding edge of weight $d_w(\ell) = d(\ell) + p \frac{q_\ell}{p_\ell + q_\ell}$.

We begin by sketching results for the special cases (with global or local state info) noted in Sect. III. All but the last are optimally solvable by running Dijkstra on the specified graph.

$(1,1)$ global: An expanded graph where two nodes $v_0, v_1$ represent each node $v$ of $G_\cup$, in even and odd timesteps, respectively.

$(1,1)$ local: A weighted graph where the weight of each edge $\ell$ is set to $d(\ell)$ if it is outgoing from $s$ and $t$, or to $d(\ell) + 1$ otherwise.

$(p, 1-p)$ local and global $(0 < p < 1)$: The weighted graph $G_w$, which, for each edge $\ell$ of $G_\cup$, has a corresponding edge of weight $d_w(\ell) = d(\ell) + p \frac{q_\ell}{p_\ell + q_\ell}$.

We begin by sketching results for the special cases (with global or local state info) noted in Sect. III. All but the last are optimally solvable by running Dijkstra on the specified graph.

$(1,1)$ global: An expanded graph where two nodes $v_0, v_1$ represent each node $v$ of $G_\cup$, in even and odd timesteps, respectively.

$(1,1)$ local: A weighted graph where the weight of each edge $\ell$ is set to $d(\ell)$ if it is outgoing from $s$ and $t$, or to $d(\ell) + 1$ otherwise.

$(p, 1-p)$ local and global $(0 < p < 1)$: The weighted graph $G_w$, which, for each edge $\ell$ of $G_\cup$, has a corresponding edge of weight $d_w(\ell) = d(\ell) + p \frac{q_\ell}{p_\ell + q_\ell}$.

B. A modified-Dijkstra heuristic

We first show that the “subpath optimality property” does not hold in this setting, which precludes using standard Dijkstra-style dynamic programming techniques to optimally solve the problem. Nonetheless, we give a Dijkstra-based heuristic and analyze its approximation factor.

Definition 6: A Markovian dynamic graph $G$ with its underlying graph $G_\cup$ satisfies the subpath property if for any three different nodes $u, v, w \in G$, assuming there are two different paths connecting $u, v, w$, namely $X(u, v)$ and $Y(u, v)$, and one path connecting $v, w$, namely $Z(v, w)$, if $ETT(X) \leq ETT(Y)$, then $ETT(XZ) \leq ETT(YZ)$, or vice versa.

As illustrated in Figure 4, consider two paths in $G_\cup$ with initial states denoted by bit strings $X, Y$. Let $\Delta = ETT(X) - ETT(Y) = E_X[T] - E_Y[T]$ and $\Delta_1 = ETT(X1) - ETT(Y1) = E_X[T1] - E_Y[T1]$. Examples can be found (see Table 4) of strings $X, Y$ violating the subpath property. In this case, the difference $\Delta_1$ can grow arbitrarily large. Nonetheless, we obtain the following lemma and a subsequent observation.

Lemma 1: If $|X| = n$ and $\Delta \leq 0$, then $\frac{\Delta}{EX[T]} \leq \alpha$, where $\alpha = \min\{\frac{q}{p}, \frac{q_\ell}{p_\ell + q_\ell}\}$.

Proof: Applying Eqs. (1,2), we can get the following recurrence ($P_n(\tau|X)$ is the probability that a path $X$ of length $n$ can be traversed in $\tau$ time slots, and thus the message will
be at the beginning of the \((n+1)\)-st link in path \(X1\):

\[
\begin{align*}
E_{X1}[T] &= E_X[T] + \frac{q}{p(p+q)}(1 - \sum_{\tau=1}^{\infty} \beta^\tau P_n(\tau|X)) \\
E_{Y1}[T] &= E_Y[T] + \frac{q}{p(p+q)}(1 - \sum_{\tau=1}^{\infty} \beta^\tau P_n(\tau|Y))
\end{align*}
\]

From the above equations, we get:

\[
\Delta_1 = \Delta + \frac{q}{p(p+q)}(E_Y[\beta^T] - E_X[\beta^T]) \\
\leq \frac{q}{p(p+q)}(1 - E_X[\beta^T]), \text{(since } \sum_{\tau=1}^{\infty} \beta^\tau P_n(\tau|Y) \leq 1) \\
\leq \frac{q}{p(p+q)}(1 - \beta^{2x[T]}), \text{(Jensen's inequality)}
\]

\[
\Delta_1 \leq \frac{q}{p(p+q)}(1 - \beta) \frac{q}{p} \leq \left(1 - \frac{\beta^x}{x}\right) \leq 1 - \beta
\]

Also, since \(E_X[T] \geq n\) (because \(d(\ell) \geq 1\) for all \(\ell\)), we have:

\[
\frac{\Delta_1}{E_X[T]} \leq \frac{\Delta_1}{E_X[T]} \leq \frac{q}{p(p+q)} \frac{1}{E_X[T]} \leq \left(\frac{q}{xp(p+q)}\right)
\]

The lemma follows.

**Observation 1:** The ratio \(\frac{\Delta_1}{ETT(X1)}\) remains small and is upper bounded by \(\alpha\) as seen in Figure 4.

Thus we give Algorithm 1, which is simply Dijkstra’s algorithm except with the change that when considering a candidate successor \(v\) for the current node \(u\), we evaluate it not by adding the edge value to the cost of the path to \(v\) but by computing the ETT of the path to \(v\) concatenated with this additional edge.

**Claim 2:** The approximation factor of any \(k\)-hop solution found by Algorithm 1 is at most \((1 + \alpha)^k\).

**C. Approximation Algorithms**

Now we give a Polynomial-time Approximation Scheme (PTAS) for the source routing problem, based on some bounds on the ETT of a path. For any \(\epsilon > 0\), this algorithm finds a path whose cost is within a factor \(1 + \epsilon\) of the optimal, and has polynomial time complexity.

First we need some bounds on the expected time to traverse a path of known state.

**Theorem 3 (General bounds for ETT):** Let \(x\) be an an initial state of an \(n\)-hop path, and let \(B_x(k)\) be the indicator function for the set of bit positions in \(x\) equaling 1. Let \(SS = n + n\pi_0/p\) be the expected traversal time in steady-state. Then ETT\((x)\) is bound by the following:

\[
|SS - ETT(x)| \leq \frac{1}{p + q} \sum_{k=0}^{n-1} \left(\frac{q}{p}\right)^k |1 - p - q|^k \quad (3)
\]

**Proof:** Let \(P_k(t|X)\) be defined as in the proof of Lemma 1. Then for any string \(X(|X| = k)\) (using \(p_{01}(t)\) and \(p_{00}(t)\) from Eqs. (1) and (2)), we have:

\[
ETT(X0) = \sum_{\tau=0}^{\infty} P_k(\tau|X)[\tau + p_{01}(\tau) \cdot 1 + p_{00}(\tau) \cdot (1 + 1/p)] = ETT(X0) + 1 + \frac{\pi_0}{p} + \frac{\sum_{\tau=0}^{\infty} P_k(\tau|X)\beta^\tau}{p+q}
\]

Similarly, we have:

\[
ETT(X1) = ETT(X1) + 1 + \frac{\pi_0}{p} - \frac{q}{p} \sum_{\tau=0}^{\infty} P_k(\tau|X)\beta^\tau
\]

Note that \(\sum_{\tau=0}^{\infty} P_k(\tau|X)\beta^\tau < |\beta|^k\) since \(P_k(\tau|X) = 0\) for \(\tau < k\). Thus we obtain the following bounds:

\[
|ETT(X) + 1 + \frac{\pi_0}{p} - ETT(X0)| \leq \frac{\beta^k}{p+q}
\]

\[
|ETT(X) + 1 + \frac{\pi_0}{p} - ETT(X1)| \leq \frac{q\beta^k}{p+q}
\]

By summing the differences \(ETT(X) - ETT(Xb)\) for all \(n\), the result follows.

We now prove a technical lemma.

**Lemma 2:** For any \(b \in (0,1)\) and \(e \in (0,b]\), we have that \(\forall x \geq \log_6 e, b^x \geq e x\).

**Proof:** Since \(e x\) grows with \(x\) and \(b^x\) decreases with \(x\), it suffices to show that for \(x = \log_6 e\), we have \(e x \geq b^x \Leftrightarrow e \log_6 e \geq b^{\log_6 e} \Leftrightarrow e \log_6 e \geq 1 \Leftrightarrow b \geq e\).

Armed with the above bound and lemma, we derive now derive the PTAS. We assume an error parameter \(\epsilon > 0\) that is sufficiently small, specifically that \(\epsilon \leq |1 - p - q| = |\beta|\).

**Theorem 4:** Let \(X = (x_1, x_2, ..., x_n)\) be the initial state of a path and \(SS\) be the path’s steady-state ETT, and let \(ETT(X^d)\) and \(SS^d\) be the corresponding values restricted to the edges \(d+1, ..., n\). Finally, let \(ETT_d(X) = ETT(x_1, ..., x_d) + SS^d\).

Then for any \(e \in (0, 1 - p - q]\), there exists a constant \(d = (d(p, q, n, e))\) such that \(ETT(X) - ETT_d(X) \leq \epsilon\).

**Proof:** Observe that:

\[
|ETT(X) - ETT_d(X)| = |ETT(X^d) - SS^d|
\]
Restricting Ineq. (3) to edges \(d + 1, \ldots, n\), we obtain:

\[
\frac{|\text{ETT}_d(X) - \text{ETT}(X)|}{\text{ETT}(X)} \leq \frac{\max \{q/p, 1\}}{p + q} \sum_{k = d + 1}^{n} |\beta|^k
\]

\[
\frac{|\text{ETT}_d(X) - \text{ETT}(X)|}{\text{ETT}(X)} < \frac{\max \{q/p, 1\}}{(p + q)(1 - |\beta|)} \cdot |\beta|^d
\]

The last inequality follows because \(\text{ETT}(X) \geq n > 0\).

Let \(c\) be the constant for which the RHS of (4) equals \(c \cdot |\beta|^d / d\) (recall \(p, q\) are fixed). Then \(c |\beta|^d / d \leq \epsilon\) iff \(\beta^d \leq c \epsilon / d\), which, by Lemma 2, holds when \(d \geq \log_\beta (\epsilon / c)\).

**Corollary 5:** There exists a PTAS (Algorithm 2) for source-routing in the \((p, q)\) model, with running time \(O(n^\gamma \cdot f(n))\), where \(\gamma = \log_\beta \frac{p + q}{q} (1 - |\beta|)\epsilon', \epsilon' = \epsilon / (2 + \epsilon)\), and \(f(n)\) is the running time of Dijkstra.

**Proof:** For fixed \(d\) chosen as in Theorem 4, \(\text{ETT}_d(\cdot)\) can be interpreted as an alternative cost function, approximating \(\text{ETT}(\cdot)\). For any \(X\) and \(\epsilon' > 0\), Theorem 4 gives the bounds:

\[(1 - \epsilon') \text{ETT}(X) \leq \text{ETT}_d(X) \leq (1 + \epsilon') \text{ETT}(X)\]

Let \(\text{OPT}\) be an optimal path according to the cost function \(\text{ETT}(\cdot)\), and let \(\text{ALG}\) be an optimal path according to \(\text{ETT}_d(\cdot)\) (i.e., a path chosen by Algorithm 2). By Ineq. (5, left), we have:

\[\text{ETT(\text{ALG})} \leq \frac{1}{1 - \epsilon'} \text{ETT}_d(\text{ALG})\]

Also observe that

\[\text{ETT}_d(\text{ALG}) \leq (1 + \epsilon') \text{ETT(\text{OPT})}\]

since otherwise (using Ineq. (5, right)) we would have \(\text{ETT}_d(\text{ALG}) > (1 + \epsilon') \text{ETT(\text{OPT})} \geq \text{ETT}_d(\text{OPT})\), which contradict the choice of \(\text{ALG}\).

Combining Ineqs. (6,7), we obtain:

\[\text{ETT(\text{ALG})} \leq \frac{1 + \epsilon'}{1 - \epsilon'} \cdot \text{ETT(\text{OPT})} = (1 + \epsilon) \text{ETT(\text{OPT})}\]

\[\Box\]

Note that the running time of Algorithm 2 depends the constants \(\epsilon, p, q\). Figure 5 illustrates how the running time exponent \(\gamma\) varies with \(p, q\) for \(\epsilon = 0.1\). We observe the running time is low for low \(q\) and low-to-moderate \(p\), but can be high when \(q >> p\). We observe that the “pole” in the curve is at \(p = q = 0.5\), the configuration where the network is the least predictable and has the maximum entropy.

**V. Dynamic Routing**

Now we turn to dynamic routing, where the decision is not made at the source node, but instead is made at each intermediate node as the message passes through them. That is, a solution to the dynamic routing problem takes the form not of a path but as a policy. The objective once again is to minimize the expected time it takes to reach the target.

**A. Special cases**

We begin by sketching results for the special cases (with global or local state) noted in Sect. III. All but the last are optimally solvable.

1. **(1, 1)** **global:** Solvable as the corresponding source routing problem.
2. **(1, 1)** **local:** Solvable by Ogier’s algorithm [19] since it can be seen that an optimal policy in this case will never reverse.
3. **(p, 1)** **local and global** \((0 < p < 1)\): Solvable by Ogier’s algorithm [19].
4. **(p, 0)** **global** \((p > 0)\): In the special case that \(d(\ell) = 0\) for every link \(\ell\), trivially solvable by waiting until there exists a path from \(s\) to \(t\), and then taking it.
5. **(0, 0)** **local** \((p > 0)\): Solvable as the corresponding source routing problem, since in steady-state every edge not local to \(s\) is on with probability 1.
6. **(0, 0)** **global:** Solvable as the corresponding source routing problem.
7. **(0, 0)** **local:** This is the PSPACE-Complete CTP [7].

Because the dynamic routing problem subsumes a PSPACE-Complete problem, we focus on heuristics.

**B. Ogier’s algorithm**

The most well known policy for routing in Markovian networks is due to Ogier and Rutenburg [19] (see Algorithm 3 with \(m = 1\)) and it operates in the setting where the current node can observe the state of links within its one-hop neighborhood. Links beyond the one-hop neighborhood are assumed to be in steady-state. The routing policy is as follows: for each neighbor \(u\) of the current node \(v\), first calculate the expected delay if the link \((u, v)\) is up and it is selected; then find the maximum \(k\) such that waiting for one more time slot due to the \(k\) lowest expected-delay links are currently all down is preferable to the expected delay of the \(k + 1\)-st link. Then the policy will be, for such \(k\), if at least one of the best \(k\) links is up, then take the best of them; otherwise wait a time slot and check again. This policy provides an optimal expected routing time under certain restrictions, such as nonzero-weight edges and an acyclic graph.

**C. Multi-hop extensions to Ogier**

For settings where we can assume current link state information beyond the one-hop neighborhood, we make a natural extension to Ogier’s algorithm. If the link states beyond one hop are far from steady-state, Ogier could perform poorly. If a node can obtain current link state knowledge from links up to \(m\) hops away \((m > 1)\), perhaps routing performance could improve, especially in sparsely connected networks.
performance evaluation. Enforcing loop freedom may require though, the loops are likely to be transient, not persistent. In cycling in loops. Since the graph is dynamic, works on the undirected underlying graph. There is a danger, DAG on the underlying graph to avoid loops, this algorithm node routing PTAS (Algorithm 2) as a subroutine. At each current paths could guarantee the minimum expected delay through (expected delay from the end of path to the destination. Then from link with (Algorithm 3) is calculating the expected delay through each link (Algorithm 4)

1) PTAS-based Heuristic: A second algorithm we give using multihop state info is a heuristic that calls the source-routing PTAS (Algorithm 2) as a subroutine. At each current node $u$, the algorithm runs the PTAS starting from $u$ and then chooses the first link of the resulting path to be its next hop.

We note that unlike Ogier’s algorithm which imposes a DAG on the underlying graph to avoid loops, this algorithm works on the undirected underlying graph. There is a danger, therefore, of cycling in loops. Since the graph is dynamic, though, the loops are likely to be transient, not persistent. In fact, we have not encountered any persistent loops during our performance evaluation. Enforcing loop freedom may require maintaining some additional state during route computation, and that is a topic of future research.

VI. PERFORMANCE EVALUATION

In this section we explore several tradeoffs between source routing and dynamic routing in some representative dynamic Markovian network configurations. We compare their relative performance by means of numerical simulations on a variety of parameters that are listed below. Since we are only studying the flow of a single message through the dynamic network, we did not use a discrete event simulation for our simulations.

A. Random Geometric Graphs

We begin with simulations on an idealized network with uniform $(p, q)$ parameters for each link. We vary several aspects of the setup: a) Structure and density of underlying graph, which is a random geometric graph (RGG) denoted by $G(n, r)^1$, modeling mesh network topologies where links can be intermittent due to outages. Figure 6 depicts the three underlying graphs with varying densities that were used in simulations. b) Dynamics parameters $(p, q)$ are varied from 0 to 1, with special emphasis on the left end of the spectrum. c) We test the different routing schemes of Sects. IV and V.

Figure 7 (left and middle) compares the performance among various source and dynamic routing protocols. From Figure 7 (left), we observe that even for networks with low $p$ and $q$ (stable but dynamic networks), source routing performs poorly compared to dynamic routing. Also, we can observe a decisive advantage towards using $m$ hop knowledge as seen by the better performance of Ogier-$m$ algorithms compared to Ogier. However, there is no advantage in using $m > 2$ in routing decisions. The best performing algorithm is DynHeuristic which in this case ends up performing a lookahead of up to 3 hops but uses a different policy than Ogier-3. Its performance is not much worse than (multi-copy) Epidemic Flooding.

Figure 8 plots the relative performance of the worst routing algorithm in the list compared to epidemic flooding for a wide range of $p, q$. The ratios of latencies of flooding and the worst algorithm are plotted in each case. The general message is that in “reddish” regions of the plot (low $p$, high $q$), even source routing algorithms perform quite well. In contrast, the lower portions of the plots are “blueish”, which means that source routing performs worse than dynamic routing.

Results for the 5 dynamic routing algorithms are shown in Fig. 9. We found that DynHeuristic is the best overall

1In an RGG, nodes are dropped uniformly at random in a square and edges are drawn between nodes whose distance is less than $r$.
algorithm and omit plots comparing it to the Ogier. We observe that Ogier-$m$ does outperform Ogier, especially in the bottom half of the plots. There is little difference in performance of the various multi-hop Ogier algorithms, however. We conclude that $m = 2$ is a reasonable choice, which gives good performance with only a modest burden of state information.

Fig. 7 (right) plots the performance of the schemes as $p = q$ varies. As $p = q \to 0$, the network becomes more stable. We find that for very low $p$, source routing schemes perform much worse than dynamic. Note that the SS-Dijkstra curve has a shape $\propto 1/p$. This is not surprising since each link is in steady-state with expected delay $1 + 1/p$. MapRTAppx performs slightly better but is outperformed by the dynamic schemes. Among these, Ogier does the worst, probably because the underlying graph is sparse ($r = 15$ is close to the critical radius of connectivity), which limits the chances for course correction. The schemes using two or more hops of link state perform better, with DynHeuristic best among them.

B. MIT Reality Mining Data

We used the MIT Reality Mining data set (intermittent connectivity traces) to evaluate the performance of our algorithms. Some salient characteristics of this network were discussed in Sect. III. Fig. 10 illustrates the routing performance between nodes belonging to 3 different classes of node degrees in the underlying graph (low (L), medium (M), and high (H)). We plot two different metrics: a) delivery failure rate and b) expected latency of delivered packets. We observe that all flows going towards high-degree nodes get delivered while a large fraction of flows going towards low-degree nodes do not arrive within the simulation time. In the extreme L-L case, about 20% flows do not make it due to the lack of a non-contemporaneous path. However, Fig. 10(b) indicates that the latency of the delivered messages is not too bad even in the L-L case. This means for some cases, source routing is not a bad choice if one is prepared to tolerate a certain fraction of lost messages. If not, then dynamic routing is preferable.

We observe asymmetries in the results. For example, a node with low degree (in $G_L$) has lower failure rate as a source than as a destination. This is because as a source, a low-degree node can avail any of the few opportunities to push the message out, which then gets routed easily through better connected neighborhoods. In contrast, as a destination, a low-degree node with the same number of contact opportunities is less likely to receive the message before the opportunity passes. We also observe that Ogier-2 outperforms Ogier in all situations. The fractional gains are the highest in H-H, M-H, H-M and L-H scenarios, where the overall latency is low to begin with.

As in the scenarios of Section VI-A, DynHeuristic outperforms the Ogier algorithms. Since $G_L$ is quite dense for this network, however, there are many different 2-hop or 3-hop paths emanating from a source. DynHeuristic must compute ETT for all of them, which makes it slower than the other dynamic routing algorithms. Hence we do not plot the performance of DynHeuristic in Figure 10 and instead show the performance of source-destination pairs in a subset of the categories below (i.e., those with lower latency).

<table>
<thead>
<tr>
<th>Category</th>
<th>Average</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>H-H</td>
<td>1183.51</td>
<td>41</td>
<td>4316</td>
</tr>
<tr>
<td>H-M</td>
<td>8582.55</td>
<td>52</td>
<td>29911</td>
</tr>
<tr>
<td>M-H</td>
<td>6265.05</td>
<td>117</td>
<td>23024</td>
</tr>
</tbody>
</table>

VII. CONCLUSION AND FUTURE WORK

In this paper, we proposed a collection of model-assisted link-state routing protocols spanning both source routing and dynamic routing for intermittently connected networks including a source routing PTAS. We found that dynamic routing generally outperforms source routing significantly in highly intermittently connected networks. We also found that lookahead beyond 1-hop is useful, particularly in sparse networks where mistakes can be hard to correct. Much more lookahead, however, did not buy much further improvement. The dynamic heuristic outperformed the source routing algorithms...
as well as Ogier’s dynamic routing algorithm. Seeking possible performance guarantees for that heuristic is a topic of future research.

This paper constitutes only a first step towards a theory of routing in Markovian networks (e.g., we restricted ourselves to time-correlated dynamics). Natural interesting generalizations include spatial correlations between edge states. A simple example is that of node failures causing adjacent edge failures (in which case source and dynamic routing become tractable [20]), but of course more complicated sorts of correlations could occur. Other topics include online periodic estimation of \((p_i,q_i)\) parameters and development of intelligent dissemination strategies to fuel the routing algorithms.

REFERENCES


