Optimizing Route-Cache Lifetime in Ad Hoc Networks

Ben Liang† and Zygmunt J. Haas‡

Abstract—**On-demand routing reduces the control overhead in mobile ad hoc networks, but it has the major drawback of introducing latency between route-request arrival and the determination of a valid route. This paper addresses the issue of minimizing the delay in on-demand routing protocols through optimizing the Time-to-Live (TTL) interval for route caching. An analytical framework is introduced to compute the expected routing delay when a source node or an intermediate node has a cached route with any given TTL value. Furthermore, numerical methods are proposed to determine the optimal TTL of a newly cached route. We present simulation results that support the validity of our analysis. Using the proposed analytical framework, we study how the routing delay is affected by route length, route-request frequency, and the frequency of topology variation. We show that the proposed optimal route-cache TTL strategy can significantly reduce the routing delay over systems that either does not use route-cache or keeps route-cache indefinitely long. We further show that the performance gain of optimizing the route-cache TTL increases with increasing traffic pattern localization.**

I. Introduction

In the ad-hoc network architecture, there is no pre-existing fixed network infrastructure. Nodes of an ad-hoc network are mobile hosts, typically with similar transmission power and computation capabilities. Direct communication between any two nodes is allowed when adequate radio propagation conditions and network channel assignment exist. Otherwise the nodes communicate through multi-hop routing. The lack of fixed infrastructure suggests that some network functions, otherwise handled by the wireline backbone, must now be maintained by the nomadic nodes in an ad-hoc network.

Node mobility and the lack of topological stability make the routing protocols previously developed for wireline networks unsuitable for ad hoc networks[4][19][16][27][8]. The ad hoc network routing protocols can be roughly divided into three categories: *proactive*, *reactive*, and *hybrid*. A proactive routing protocol, also called a *table-driven* protocol, requires that each node maintains an up-to-date routing table, such that a route is readily available when data packets need to be send out. Routing protocols such as DSDV[25], FSR[12], OLSR[13], STAR[5], TBRPF[1], and WRP[22] are examples of proactive protocols. In reactive routing protocols, also called *on-demand* protocols, a node is not required to maintain a routing table (although route caches may be kept), but instead a route query process is initiated whenever it is needed. Routing protocols such as ABR[31], AODV[26], DSR[15], and TORA[24] are examples of reactive protocols. A reactive protocol avoids the control message overhead incurred in building routing tables, which may never be used before the routes become obsolete due to topology changes. However, when routes are requested, nodes need to send out route query messages into a large part of the network, which could lead to a large delay of route response, in addition to the potential penalty in network resources. A hybrid flat protocol, ZRP[7], was proposed to combine the benefit of both approaches. In this protocol, a node proactively maintains only the link information of nodes within a variable-sized local neighborhood, and it reactively sends out route queries to faraway destinations.

A. Route Caching

In an on-demand routing protocol, or in the on-demand component of a hybrid routing protocol, a newly discovered route should be cached, so that it may be reused the next time that the same route is requested. Two cases of route caching exists. In the basic case, a source node caches routes so that a route is available when an application, running within the same node, demands it. We call this *source route caching*. As an extension to the above, many on-demand routing protocols, such as AODV and DSR, allow an intermediate node (a non-destination node that has received a copy of the source node's route request) that has a cached route to the destination reply to the source with the cached route. We call this *intermediate route caching*.

The benefit of using route caches in the on-demand approach is two-fold. First and most importantly, when a route request arrives, provided that the cached route is not obsolete, a valid route is immediately available, leading to significantly smaller routing latency. This is particularly important in audio and video transmissions, where the successful play-back of the received information is delay sensitive. Second, route caching reduces the control traffic that is required in searching for a new route.

However, prolonged storage of a route cache may render it obsolete. When an invalid route cache is used, extra traffic overhead and routing delay is incurred to discover the broken links. Depending on the implementation details, data and/or control packets are delivered over part of the cached route that is still valid, before the broken link can be discovered.

One approach to minimize the effect of invalid route cache is to purge the cache entry after some Time-to-Live (TTL) interval.[1] If the TTL is set too small, valid routes are likely to be discarded, and large routing delay and traffic overhead may result due to the new route search. On the other hand, if the TTL is set too large, invalid route-caches are likely to be used, and additional routing delay and traffic overhead may result before the broken route is discovered. Thus, an algorithm that optimizes

† Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada; liang@comm.utoronto.ca

‡ School of Electrical and Computer Engineering, Cornell University, Ithaca, New York, USA; haas@ece.cornell.edu

[1] Another approach is to use a route-cache invalidation procedure, where the up-stream node of a broken link initiates route invalidation packets, whether or not the broken link is part of an active route presently delivering data. The route invalidation packets prompt the route-caching nodes to purge the affected route caches. This is essentially a proactive approach, which can lead to large control overhead when the network topology changes frequently. Such *proactive* route-cache invalidation procedures are outside the scope of this paper.

the TTL setting is necessary for the optimal performance of an on-demand routing protocol.

B. Routing Delay vs. Traffic Overhead

The impact of using an invalid route cache is not the same in terms of the traffic overhead or the routing delay.

In terms of the traffic overhead, this impact is relatively small. Since the number of hops in a route scales as the *square-root* of the number of nodes, it's easy to see that the traffic to discover a broken link scales as the square-root of the number of nodes. Meanwhile, the traffic to discover a new route scales *linearly* with the number of nodes. Therefore, the extra traffic overhead due to an invalid route cache is comparatively insignificant.

In terms of the routing delay, however, the impact of invalid route caches is large. The delay required to discover a broken link scales as the square-root of the number of nodes. This is of the same scaling law as the delay required to discover a new route. Therefore, invalid route caches can significantly prolong the routing delay.

Hence, in this paper, we concentrate on the routing delay of on-demand routing.

C. Outline of Presentation

In this work, we consider the problem of optimizing the TTL of a cached route in order to minimize the expected routing delay of the next request of the same route (i.e., the same source and destination node pair, or the same intermediate and destination node pair). In Section II, we survey the related work in literature. In Section III, we explain the network model under consideration. In Section IV, we concentrate on the analysis of *source* route-caching. We first introduce an analytical framework to compute the expected routing delay when the source node has a cached route with a given TTL value. We then propose a numerical method to determine the optimal TTL of a newly discovered route cached by the source node. In Section V, we extend the analysis in the previous section to the case of *intermediate* routing caching, presenting approximation methods to compute the expected routing delay and to determine the optimal route-cache TTL. In Section VI, we present simulation results that support the validity of our analysis. Using the proposed analytical framework, we further study how the routing delay is affected by the route length and the route-request frequency relative to the frequency of topology variation. In addition, we show that the optimal route-cache TTL gives the most performance gain when the traffic pattern is localized. Finally, concluding remarks are provided in Section VII.

II. Related Work

In wireline networks, such as the Internet, links are relatively stable. Hence, the routing protocols of choice are proactive[18], so that route caching is unnecessary. Even in wireless cellular networks, since the data traffic is routed through the wireline backbone network[28], route caching is irrelevant.

In ad hoc networks, however, as explained in the previous section, the proper selection of the route-cache TTL is particularly important. As far as we are aware, there is very little reported work in literature that addresses the issue of ad hoc route-cache TTL optimization.

Most existing on-demand protocols, such as AODV, DSR, and TORA, employ route caching in various forms, while others, such as ABR, do not consider route caching. In DSR and TORA, a cached route is kept indefinitely (i.e. TTL=∞), until a broken link in the route is detected during data transmission. In AODV, a discovered route is associated with an "active route time-out" value that dictates the duration within which the route can be used. This time-out value is static and identical throughout the network. In this work, we study the optimization of the route-cache TTL *adapted* to each cached route.

In [21], case study based on DSR has suggested that route caching can reduce the average latency of route discovery by more than 10-fold. Further simulation studies reported in [2], [14], [10], [3], and [11] have confirmed the effectiveness of route caching in on-demand routing protocols. However, [14], [10], [3], and [11] have also drawn the conclusion that the indefinite route-cache, as is employed in DSR, can lead to many stale routes and hence degrade the routing performance.

In addition, [3] and [11] have demonstrated the need for determining a suitable time for the route-cache expiration. The simulation results in [11] have further shown a case study of the optimal route-cache expiry time obtained by exhaustive search. In this work, we approach the problem of adaptive route-cache TTL optimization through analytical studies.

III. Network Model

We consider a mobile ad hoc network represented by a graph $G(V, E)$ consisting of a set V of nodes together with a set E of edges. At any time instant, an edge (u, v), where $u, v \in V$, exists if and only if node u can successfully transmit to node v. In this case, we say that the link from node u to node v is *up*. Otherwise, the link is *down* or has *failed*.

The topology of a mobile ad hoc network changes continuously. In the modeling of general communication networks, it is usually assumed that all edge failures are statistically independent [17]. The modeling of dependent link failures generally requires a large number of conditional probability distributions. Therefore, though unrealistic, the independence assumption greatly simplifies the analysis of network performance. In this paper, we assume that all links have identical up-time distribution $F_u(t)$ and down-time distribution $F_d(t)$. However, the proposed analytical framework can accommodate non-identical up-time and down-time distributions as well. In particular, if a node has somehow obtained the up-time and down-time distributions of each link in its cached route, this information can be applied to improved the estimation of route lifetime. However, collecting the statistics of each individual link can potentially lead to very large amount of control message overhead. Therefore, we do not assume the availability of such information in this work.

Source Route Caching: We assume that route requests to a destination node n_d arrive at the source node n_s as a stream that has identically distributed inter-arrival intervals with a general distribution $F_{a(s,d)}(t)$. When there is no ambiguity, this is

simply denoted $F_a(t)$. We consider only non-trivial networks where the average time between topology changes is smaller than the average route-search delay.[2] Therefore, we assume that the route-request inter-arrival time is much larger than the route-search delay, since, otherwise, a valid route is already found at the last route request. Namely, a burst of data packets sent to a common destination within a very small time frame would cause a single route request.

When a route request is made due to a data packet arrival, if n_s has a cached route to n_d, it immediately sends out the data packet using the cached route. If the cached route is valid, we assume that this operation does not incur any routing delay. However, if the cached route is invalid, the node on the up-stream end of a failed link notifies n_s via a route-error packet. In this case, and in the case that n_s does not have a cached route to n_d, the pre-defined routing protocol [3] is employed to search for a new route to n_d. We further assume that n_s renews or re-computes the TTL of a cached route to n_d each time a packet is successfully sent through the cached route. A cached route is purged when its TTL expires.

Intermediate Route Caching: We consider the case where a route request to a destination node n_d is originated from a source node n_s and received by an intermediate node n_i. As in the source route caching case, we only consider non-trivial networks where the average time between topology changes is smaller than the route-search delay.

When a route request is received by n_i, if n_i has a cached route to n_d, it computes the proper TTL of the cached route *for the n_s-to-n_d route request*[4], based on the method to be described in Section V. If the route-cache has expired for the route request from n_s to n_d, node n_i simply retransmits the route request to its neighbors, usually following a controlled flooding scheme [7]. If the TTL has not been exceeded, n_i sends the cached route to n_s, together with the newly discovered (or revalidated) route between n_s and n_i. If the cached route is valid, the total routing delay, for n_s, is the round-trip time between n_s and n_i. If the cached route is not valid, the node on the up-stream end of a failed link notifies n_s via a route-error packet. In this case, a new route-discovery process is initiated by n_s.

As explained in Section I, we are mostly interested in reducing the routing delay, by using the appropriate route-cache TTL. We assume that all data and control packet transmissions across a link incur an average delay of L seconds.[5] Although packets may be different in length, in a wireless ad hoc network operating at medium to high load, the predominant factor in the aggregate delay of packet transmission across a link is the queuing delay in th MAC layer due to the contention of the share wireless medium. This justifies the assumption that different types of packets incur the same delay over a link. If necessary, this assumption can be relaxed, requiring only minor modification of the proposed analytical framework.

n_s, n_i, n_d:	source, intermediate, and destination nodes
$F_u(t)$, $f_u(t)$:	probability distribution and density of link up-time
${f_u}^*(s)$:	Laplace transform of $f_u(t)$
$F_d(t)$, $f_d(t)$:	probability distribution and density of link down-time
$F_a(t)$, $f_a(t)$:	probability distribution and density of route request inter-arrival interval
${f_a}^*(s)$:	Laplace transform of $f_a(t)$
L:	average delay per packet transmission per link
S:	hop count to source node
D:	hop count to destination node
T:	route-cache TTL
t_a:	time between two consecutive route requests
$f_c(t)$:	probability density of t_a given $t_a < T$
${f_c}^*(s)$:	Laplace transform of $f_c(t)$
$F_r(t)$, $f_r(t)$:	probability distribution and density of backward residual lifetime of link in cached route
${f_r}^*(s)$:	Laplace transform of $f_r(t)$
$R_r(t)$:	$1 - F_r(t)$
X_i:	minimum of the residual lifetimes of the first i links in cached route
$F_{X_i}(t)$, $f_{X_i}(t)$:	probability distribution and density of X_i
${f_{X_i}}^*(s)$:	Laplace transform of $f_{X_i}(t)$
$R_{X_i}(t)$:	$1 - F_{X_i}(t)$
$Q_i(T)$:	probability that when a route request arrives before TTL$= T$ expires the first i links of the cached route have not failed
$p_i(T)$:	$Q_{i-1}(T) - Q_i(T)$

TABLE I
TABLE OF NOMENCLATURE: PART I

Tables I and II contain lists of the symbols, definitions, and variables used throughout this paper.

IV. Optimizing the Source Route-Cache TTL

Obviously, the optimal TTL of a route-cache depends on how stable the route topology is. In a hypothetical universe, if the status of all links at all times in the future were known, the TTL should be set to the first time instant that any link in the route fails. In reality, however, we can only estimate the route lifetime from statistics and optimize the TTL to minimize the *expected* delay.

This section deals with the problem of optimizing the TTL for

[2] Otherwise, the only suitable routing approach is to flood data packets throughout the network

[3] The exact mechanism of the on-demand routing protocol is not important here.

[4] It can be shown that, within any node, the cached route to a destination as a source route-cache (i.e., the node itself is the source) always has larger TTL value than that of the intermediate route-cache (i.e., the node serves as an intermediate node) to the same destination.

[5] For example, in the simulation environment considered in the case study of [21], the average delay is shown to be approximately 14.5 ms/hop.

$C_{x>T(T)}, C_{x<T(T)}$:	expected delays given TTL$= T$ and the comparison of x and T, due to source route caching
$C(T)$:	expected delay given TTL$= T$, due to source route caching
$C_{\lambda_u,\lambda_a}(T)$:	special case of $C(T)$ with exponential distributions
$q(\tau)$:	probability that a given link in the cached route is up at time τ after the last route request
$g(x)$:	auxiliary function used in TTL optimization, for source route caching
$C'_{...}(T)$, $g'(x)$:	counterparts of $C_{...}(T)$ and $g(x)$ for intermediate route caching
T_{opt}:	optimal TTL value
γ:	TTL optimality factor ($\gamma = 1$ for optimal TTL)
λ_a, μ_a:	rate and mean of route request arrival time
λ_u, μ_u:	rate and mean of link up-time
λ_d, μ_d:	rate and mean of link down-time
π_D:	distribution of D

TABLE II
TABLE OF NOMENCLATURE: PART II

source route caching, assuming that intermediate route caching is not performed. In the following two subsections, we first provide an analytical frameworks for computing the expected routing delay. We then describe a numerical method to determine the optimal TTL.

A. Computing the Expected Routing Delay

Suppose the source node n_s has a cached route to the destination node n_d, which is validated by the last route request and has a TTL value of T seconds. Let D be the number of hops in this route.

Let the next route request to n_d arrive at time t_a after the n_s-to-n_d route is cached. Then, from Section III, t_a has distribution $F_a(t)$. Let $f_a(t)$ be the density function of t_a, and let ${f_a}^*(s)$ be the Laplace transform of $f_a(t)$.

If $t_a > T$, node n_s searches for a new route to n_d. The routing delay is the time for one round-trip propagation of control packets between n_s and n_d. We assume that the two end nodes have not moved too far during the inter-arrival interval, so that the new route still has mean route length of D hops. In this case, the routing delay is

$$C_{t_a>T}(T) = 2LD \; . \tag{1}$$

If $t_a < T$, either there is no routing delay, in the case where the cached route is valid, or the routing delay is the time to discover the route-cache invalidity plus the time to search for a new route. The time to discover the route-cache invalidity depends on the distance between the source node and the nearest failed link from the source. In the following, we first derive the distribution of the distance between the source and the first failed link.

Let $f_c(t)$ be the density function of t_a given $t_a < T$. We have

$$f_c(t) = f_a(t \mid t < T) = \frac{1}{F_a(T)} f_a(t) P_T(t) \; ,$$

where

$$P_T(t) = \begin{cases} 1 & 0 \le t \le T \\ 0 & \text{otherwise} \; . \end{cases}$$

Thus, the Laplace transform of $f_c(t)$ is

$$\begin{aligned} {f_c}^*(s) &= \frac{1}{2\pi j F_a(T)} {f_a}^*(s) * \left[\frac{1 - e^{-sT}}{s} \right] \\ &= \frac{1}{2\pi j F_a(T)} \int_{c-j\infty}^{c+j\infty} \frac{1 - e^{-zT}}{z} {f_a}^*(s-z) dz \\ &= -\frac{1}{F_a(T)} \sum_{\xi \in \substack{\text{poles of} \\ {f_a}^*(s\,-\,z)}} \operatorname*{Res}_{z=\xi} \frac{1 - e^{-zT}}{z} {f_a}^*(s-z) \; , \end{aligned} \tag{2}$$

where $\operatorname*{Res}_{z=\xi}$ denotes the residue at the pole $z = \xi$.

Let $f_u(t) = dF_u(t)/dt$ be the density function of the link up-time and ${f_u}^*(s)$ be its Laplace transform. Since the route-request arrival process is independent of the network topology change, given the time that the cached route was determined due to the previous route request, the residual lifetime of a link in the cached route has the density function[23]

$$f_r(t) = \frac{1}{\mu_u}[1 - F_u(t)] \; , \tag{3}$$

where μ_u is the mean up-time of a link. The Laplace transform of $f_r(t)$ is

$${f_r}^*(s) = \frac{1}{\mu_u s}[1 - {f_u}^*(s)] \; .$$

Let X_i be the minimum of the residual lifetimes of the first i links in the cached route. Let $f_{X_i}(t)$ and $F_{X_i}(t)$ be its density and distribution functions, and let ${f_{X_i}}^*(s)$ and ${F_{X_i}}^*(s)$ be the corresponding Laplace transforms, respectively. Let $R_{X_i}(t) = 1 - F_{X_i}(t)$ and $R_r(t) = 1 - F_r(t)$, and ${R_{X_i}}^*(s)$ and ${R_r}^*(s)$ be their Laplace transforms, respectively. Then ${f_{X_i}}^*(s)$ can be determined through the following recursion. We have

$$R_{X_i}(t) = [R_r(t)]^i = R_{X_{i-1}}(t) R_r(t) \; ,$$

since the link failures are independent [23]. Thus,

$$\begin{aligned} {R_{X_i}}^*(s) &= \frac{1}{2\pi j}[{R_{X_{i-1}}}^*(s) * {R_r}^*(s)] \\ &= \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} {R_{X_{i-1}}}^*(z) {R_r}^*(s-z) dz \\ &= - \sum_{\xi \in \text{poles of } {R_r}^*(s-z)} \operatorname*{Res}_{z=\xi} {R_{X_{i-1}}}^*(z) {R_r}^*(s-z) \; . \end{aligned} \tag{4}$$

From this, we can compute

$$f_{X_i}{}^*(s) = sF_{X_i}{}^*(s) - F_{X_i}{}^*(0+) = 1 - sR_{X_i}{}^*(s) \ .$$

Let $Q_i(T)$ be the probability that, when a route request arrives before the TTL expires, the first i links of the cached route have not failed. We have

$$\begin{aligned} Q_i(T) &= \Pr\{t_a < X_i \mid t_a < T\} \\ &= \int_0^\infty \Pr\{t_a < t \mid t_a < T\} f_{X_i}(t)dt \\ &= \int_0^\infty \left[\frac{1}{2\pi j}\int_{c-j\infty}^{c+j\infty} \frac{f_c{}^*(s)}{s} e^{st} ds\right] f_{X_i}(t)dt \\ &= \frac{1}{2\pi j}\int_{c-j\infty}^{c+j\infty} \frac{f_c{}^*(s)}{s}\left[\int_0^\infty f_{X_i}(t)e^{st}dt\right] ds \\ &= \frac{1}{2\pi j}\int_{c-j\infty}^{c+j\infty} \frac{f_c{}^*(s)}{s} f_{X_i}{}^*(-s)ds \\ &= -\sum_{\xi \in \text{poles of } f_{X_i}{}^*(-s)} \operatorname*{Res}_{s=\xi} \frac{f_c{}^*(s)}{s} f_{X_i}{}^*(-s) \ . \end{aligned} \tag{5}$$

Due to the multi-node contention of common wireless medium, the appropriate value of the average number of neighbors per node in an ad hoc network is usually small.[6] Therefore, in the i.i.d. link model with sufficiently large number of nodes, the average duration that a link is down is far larger than the average duration that the link is up. Namely, it is safe to assume that, within the time scale of route-request arrivals, a link that has failed within the route-request arrival interval will not be up by the end of the interval. Therefore, the probability that, when a route request arrives before the TTL expires, the first $i-1$ links are up, but the i^{th} link is down is

$$p_i(T) = Q_{i-1}(T) - Q_i(T) \ ,$$

where we define $Q_0(T) = 1$.

From the above, the expected routing delay given $t_a < T$ is[7]

$$C_{t_a<T}(T) = 2L\sum_{i=1}^{D} i p_i(T) + 2L[1 - Q_D(T)]D \ . \tag{6}$$

Finally, combining the equations (1) and (6), we obtain the expected routing delay of the next route request, when the TTL of a D-hop cached route is set to T:

$$\begin{aligned} C(T) &= F_a(T)C_{t_a<T}(T) + [1 - F_a(T)]C_{t_a>T}(T) \\ &= 2L\left[D + F_a(T)\sum_{i=1}^{D} i p_i(T) - F_a(T)Q_D(T)D\right] \ . \end{aligned} \tag{7}$$

[6]For example, in [30] and [29] the optimal number of neighbors per node is found to be around six.

[7]Here, we have assumed that the node at the up-stream end of a failed link discovers the link failure after waiting for the round-trip packet-delivery time over the link. This is not necessary if the system under consideration employs a MAC layer protocol that constantly monitors the neighborhood connectivity and reports that information to the upper layers. In this case, it is a simple matter to subtract $2L$ from the cost $C_{t_a<T}(T)$.

As an example, if the link up-times are exponentially distributed with mean $\mu_u = 1/\lambda_u$ and if the route-requests form a Poisson process with rate λ_a, equation (7) can be reduced to

$$\begin{aligned} C_{\lambda_u,\lambda_a}(T) =& 2LD + 2\lambda_a L\sum_{i=0}^{D-1} \frac{1 - e^{-(i\lambda_u+\lambda_a)T}}{i\lambda_u + \lambda_a} \\ &- 4\lambda_a LD\frac{1 - e^{-(D\lambda_u+\lambda_a)T}}{D\lambda_u + \lambda_a} \ . \end{aligned} \tag{8}$$

The detailed derivation is provided in Appendix A. In Section VI, we verify the above through simulations.

The above analytical framework provides a means for evaluating the expected routing delay given the TTL value. However, it is very likely that the optimal TTL value is more important to a system designer. Due to the complexity of equation (7), or even in its simpler, special case of equation (8), where all distributions are memory-less, this is not easy to accomplish. In the next section, we provide a numerical method to compute the optimal TTL.

B. Determining the Optimal Route-Cache TTL

Let $q(\tau)$ be the probability that a given link in the cached route is still up at time τ after the last route request. Suppose a route request arrives at time τ after the last route request. Let T be the chosen TTL value for the cached route. If $\tau > T$, the cached route is not used, and the routing delay is

$$C_{\tau>T}(\tau, T) = 2LD \ . \tag{9}$$

If $\tau < T$, the routing delay is

$$\begin{aligned} C_{\tau<T}(\tau, T) &= 2L\sum_{i=1}^{D} i[1 - q(\tau)]q^{i-1}(\tau) + 2L[1 - q^D(\tau)]D \\ &= 2L\left[D + \frac{q^D(\tau) - 1}{q(\tau) - 1} - 2Dq^D(\tau)\right] \ . \end{aligned} \tag{10}$$

Therefore, the expected routing delay as defined in (7) has the following alternate from:

$$\begin{aligned} C(T) &= \int_T^\infty C_{\tau>T}(\tau, T)f_a(\tau)d\tau + \int_0^T C_{\tau<T}(\tau, T)f_a(\tau)d\tau \\ &= 2LD - 2L\int_0^T \left[2Dq^D(\tau) - \frac{q^D(\tau) - 1}{q(\tau) - 1}\right] f_a(\tau)d\tau \ . \end{aligned} \tag{11}$$

Since $q(\tau)$ is a decreasing function of τ and $0 \le q(\tau) < 1$, it is easy to verify that $C(T)$ is a concave function of T. Therefore, if we let

$$\begin{aligned} g[q(T)] &= -\frac{1}{2Lf_a(T)}\frac{dC(T)}{dT} \\ &= 2Dq^D(T) - \frac{q^D(T) - 1}{q(T) - 1} \ , \end{aligned}$$

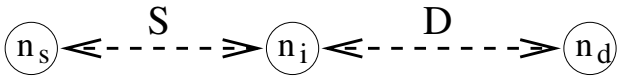

Fig. 1. Intermediate node route caching and route reply

the minimum of $C(T)$ is achieved when $g[q(T)] = 0$. Therefore, the optimal value of $q(T)$ is the root in $[0, 1)$ of a function of the form

$$g(x) = 2Dx^D - \frac{x^D - 1}{x - 1} .$$

Given any value of D, a numerical method such as bisection or the Newton's method[9] can be used to find this root. Since $q(T) = 1 - F_r(T)$, once the optimal value of $q(T)$ is determined numerically, the optimal TTL value can be found by reversing the density function of the residual lifetime of a link.

From the above analysis, one immediately makes an important observation that the optimal TTL value does not depend on $f_a(t)$, although the minimal routing delay does. Therefore, the optimization of the route-cache TTL can be carried out independently of the actual route-request inter-arrival statistics. This property of the optimal TTL significantly reduces the computational requirement of the adaptive, real-time route-cache TTL optimization performed by the individual nodes in an ad hoc network.

Following the example in the previous section, in the case that the link up-times are exponentially distributed with mean $\mu_u = 1/\lambda_u$ and the route-requests form a Poisson process with rate λ_a, the optimal TTL value T_{opt} satisfies

$$q(T_{opt}) = 1 - [1 - e^{-\lambda_u T_{opt}}] .$$

Therefore, in this case, we have

$$T_{opt} = -\frac{1}{\lambda_u} \log q_{opt} ,$$

where q_{opt} is the root of $g(x)$ in $[0, 1)$.

The validity of the above derivation is verified through simulation, and the results are shown in Section VI.

V. Optimizing the Intermediate Route-Cache TTL

In this section, we extend the analytical frameworks in Section IV to provide an heuristic scheme for the optimization of the route-cache expiry time for *intermediate route caching*.

As shown in Figure 1, suppose a copy of the route request, from the source node n_s to the destination node n_d, is received by an intermediate node n_i, and n_i has a cached route to n_d. Let S be the hop count from n_s to n_i, and D be the hop count from n_i to n_d. Clearly, the optimal TTL of the cached route depends on both S and D. In fact, since $S \leq 0$, it can be easily demonstrated that this TTL is always smaller than or equal to the TTL of the *source* route cache from n_i to n_d. Therefore, the selection of the optimal source route-cache TTL does not affect the determination of the intermediate route-cache TTL.

As in Section IV, we first present the analytical model for computing the routing delay, and then present a method for minimizing the routing delay. However, here we will combine both in an iterative framework.

Let t_a be the time of arrival of the next route request, from n_s to n_d, at node n_i, after the n_a-to-n_d route is cached, due to the previous route establishment or revalidation. Let T_{opt} be the optimal TTL value of the cached n_i-to-n_d route, *corresponding to the n_s-to-n_d route request*. Let $C'_{min}(S, D)$ be the associated *total* routing delay from n_s to n_d. We further assume that t_a is also the time of route request arrival after the last time the n_i-to-n_d route is cached. Since the n_i-to-n_d route-cache can be established or revalidated based on route searches from nodes other than n_s, this assumption leads to a *worst case* approximation of the actual routing delay.

Based on most existing on-demand routing protocols, if no route is cached in n_i, or if the cached route has expired, the route request continues to be forwarded to the adjacent nodes of n_i [27][8]. Therefore, in this case, the routing delay is $2LS$ plus the time of route discovery between n_i and n_d. Suppose all nodes in the network use the same optimal route-cache TTL strategy, we have the upper-bound

$$C'_{t_a > T_{opt}}(S, D) = C'_{min}(S + 1, D - 1) .$$

If the route cache at n_i is used and it is valid, the routing delay equals to the route reply delay, $2LS$. If the route cache at n_i is used and it is not valid, the routing delay is a combination of the route reply delay, the time for the source node to discover the broken link, and the time to search for a new route. Therefore, corresponding to equation (6), we have

$$\begin{aligned} C'_{t_a < T_{opt}}(S, D) =& 2LS + 2L \sum_{i=1}^{D} i p_i(T_{opt}) + [1 - Q_D(T_{opt})] \\ & \cdot [2LS + C'_{min}(S + 1, D - 1)] . \end{aligned}$$

Thus, the expected routing delay is

$$\begin{aligned} & C'_{min}(S, D) \\ =& F_a(T_{opt}) C'_{t_a < T_{opt}}(S, D) + [1 - F_a(T_{opt})] C'_{t_a > T_{opt}}(S, D) \\ =& [1 - F_a(T_{opt}) Q_D(T_{opt})] C'_{min}(S + 1, D - 1) \\ & + 2LF_a(T_{opt}) \left[2S - Q_D(T_{opt}) S + \sum_{i=1}^{D} i p_i(T_{opt}) \right] \end{aligned} \tag{12}$$

In order to determine the optimal intermediate route-cache TTL, we modify the analytical framework in Section IV-B as follows.

Corresponding to equations (9)-(10), the routing delay when route cache has expired is

$$C'_{\tau > T}(S, D) = C'_{min}(S + 1, D - 1) ,$$

and the routing delay when route cache has not expired is

$$\begin{aligned} C'_{\tau < T}(S, D) = & 2LS + 2L \sum_{i=1}^{D} i[1 - q(\tau)] q^{i-1}(\tau) \\ & + [1 - q^D(\tau)][2LS + C'_{min}(S + 1, D - 1)] \end{aligned}$$

Therefore, the total expected routing delay due to intermediate route caching can be expressed by

$$\begin{aligned} & C'_T(S,D) \\ =& \int_T^\infty C'_{\tau>T}(S,D) f_a(\tau) d\tau + \int_0^T C'_{\tau<T}(S,D) f_a(\tau) d\tau \\ =& C'_{min}(S+1,D-1) - 2L \int_0^T f_a(\tau) \Bigg[[D+S+\frac{1}{2L} \\ & \cdot C'_{min}(S+1,D-1)] q^D(\tau) - \frac{q^D(\tau)-1}{q(\tau)-1} - 2S \Bigg] d\tau \end{aligned}$$

If we let

$$g'[q(T)] = -\frac{1}{2Lf_a(T)} \frac{dC'_T(S,D)}{dT} ,$$

the minimum of $C'(T)$ is achieved when $g'[q(T)] = 0$. Therefore, the optimal value of $q(T)$ is the root in $[0,1)$ of a function of the form

$$\begin{aligned} g'(x) =& [D+S+\frac{1}{2L} C'_{min}(S+1,D-1)] x^D \\ & - \frac{x^D-1}{x-1} - 2S , \end{aligned} \tag{13}$$

From this, the optimal value of the intermediate route-cache TTL at node n_i can be determined as a function of S, D, and $C'_{min}(S+1,D-1)$.

Furthermore, we have the initial condition

$$C'_{min}(S+D,0) = 2L(S+D) .$$

Thus, iteratively combining (12) and (13), we can determine the optimal route-cache TTL at each intermediate node, and the corresponding routing delay between nodes n_s and n_d.

VI. Simulation and Numerical Evaluation

We use discrete-event simulation to validate the proposed analytical framework. Selected results are presented in this section. In addition, we apply the analytical framework to study the minimal routing delay given various system parameters. We further demonstrate the performance gain achieved by the TTL optimization, through comparison with systems employing *no route cache* or *never-expiring route cache*.

A. Simulation Model and Output Analysis

A simulation model is developed to represent the link establishments and breakages in an ad hoc network based on the network model described in Section III. In particular, we present the simulation results for the case where the link up and down times are exponentially distributed. The link between every pair of nodes is modeled to become up and down alternatively, with inter-event arrival time distributions $f_u(t) = \frac{1}{\mu_u} e^{-\frac{t}{\mu_u}}$ and $f_d(t) = \frac{1}{\mu_d} e^{-\frac{t}{\mu_d}}$. In this case, the state of a link can be modeled by a continuous-time Markov Chain as shown in Figure 2.

Given a source node, the destination node is chosen randomly with uniform distribution among all other nodes in the network.

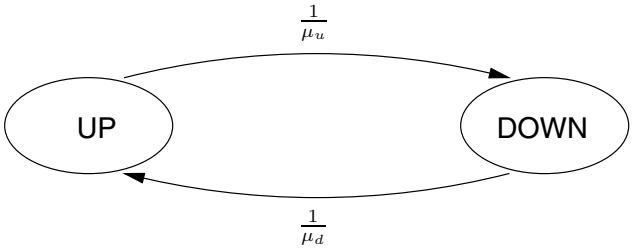

Fig. 2. Continuous-time Markov-Chain model of link states

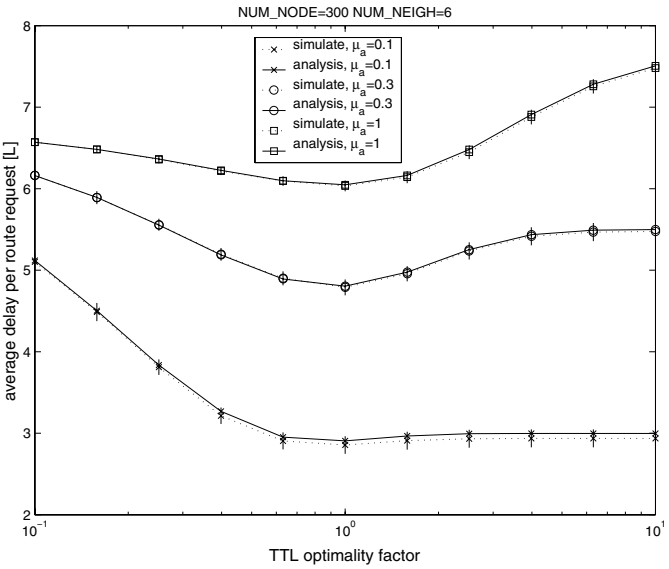

Fig. 3. Expected routing delay normalized to L vs. TTL optimality factor. Comparison between simulation and analysis on a network of 300 nodes. The average degree per node is 6. $\mu_u = 1$ and $\mu_d = 48.8$. The vertical lines represent the 99.95% confidence intervals.

For a chosen source and destination node pair, the route-request inter-arrival time has distribution $f_a(t) = \frac{1}{\mu_a} e^{-\frac{t}{\mu_a}}$, where $\mu_a = \frac{1}{\lambda_a}$.

Figure 3 illustrates the expected routing delay normalized to L, the average delay of packet delivery over each link, for a network of 300 nodes, with purely source route caching. The mean up-time is 1 and the mean down-time is 48.8. As a result, there are 6 neighbors per node on average.

We define a TTL optimality factor γ, such that, when a new route is cached, its TTL is set to γT_{opt}, where T_{opt} is the optimal TTL value found as described in Section IV-B. Each plot in Figure 3 represents the expected routing delay vs. γ, for $0.1 \leq \gamma \leq 10$. The plots show the three cases where μ_a is 0.1, 0.3, or 1, representing various route-request arrival frequencies. The 99.95% confidence intervals are also presented in the figure.

The analytical results are compared with the simulation results. Figure 3 shows that the analysis agrees very well with the simulation. In particular, the simulation results demonstrate that the minimal routing delay is indeed achieved at $\gamma = 1$, as expected from the analysis.

There is small (less than 2%) difference between the analytical and simulation results. This is due to the pessimistic assumption in the analytical model that once a link in a cached route fails, it does not become up again at the time of the next route request. In the simulation model and in reality, the re-establishment of a failed link in a cached route has non-zero probability P_{relink}, so that the probability of a valid cached route is larger than that dictated in the analytical model. Therefore, the analytical model gives a slightly larger estimation of the expected routing delay. However, as explained in Section IV-A, given a fixed node degree, P_{relink} is a decreasing function of the network size and, in general, is quite small for networks of size 100 or more, as demonstrated by this example.

When a route request arrives soon after the previous one, it is very likely that the cached route is still valid. As expected, Figure 3 shows that, for a given value of γ, the expected routing delay per route request is a decreasing function of the route-request frequency.

Figure 3 also demonstrates how the routing delay is affected by the TTL selection, for different route-request frequencies. As shown in Section IV-B, the route-request frequency does not affect the optimal TTL value. However, if the chosen TTL value is much larger than the mean inter-arrival time of the route requests, the TTL almost never expires, and its value does not affect the routing delay. This is demonstrated in Figure 3 where the plots level off when γ is large. In particular, when the route-request frequency is high (e.g., $\mu_a = 0.1$), the expected routing delay does not change much for TTL greater than T_{opt}. On the other hand, if the chosen TTL value is much smaller than the mean inter-arrival time of the route requests, the cached route is almost never used. In this case, neither the TTL value nor the route-request frequency affects the routing delay. This is demonstrated in Figure 3 where all three plots converge, when γ is small, to the routing delay of a non-caching system. In particular, when the route-request frequency is low (e.g., $\mu_a = 1$), the expected routing delay increases very little for TTL less than T_{opt}.

Based on the above observations, a conclusion can be drawn that the optimal TTL determination is the most important when the route-request inter-arrival time is moderate compared with the mean link-failure time.

The results for systems with other parameter values are similar to Figure 3 and are omitted.

B. Performance Gain of the Optimal Source Route-Cache TTL

Using the proposed analytical framework, we can quantitatively study the advantage of optimizing the route-cache TTL.

In the following, we have scaled time such that $\mu_u = 1$. Therefore, the route-request frequency presented here represents the relative frequency of the route requests to the frequency of topology variation.

Figure 4 illustrates the minimal routing delay given the optimal TTL, for routes of length $1 \leq D \leq 20$, and for route-request inter-arrival time $\mu_a \in [0.1, 0.3, 1, 3, 10]$. As expected, these plots show that the routing delay is an increasing function of both D and μ_a. In particular, when μ_a is large (i.e., $\mu_a > 3$), the optimized routing delay approaches $2LD$, the expected delay when no route cache is used.

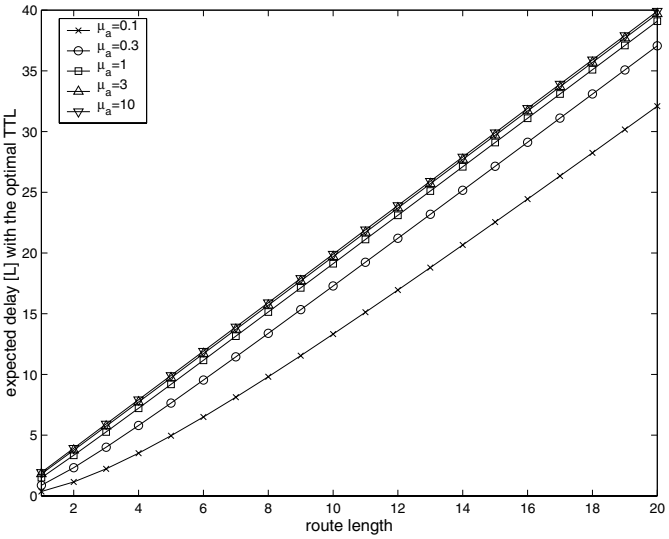

Fig. 4. Expected routing delay based on the optimal TTL vs. the distance between source and destination nodes, with source route caching.

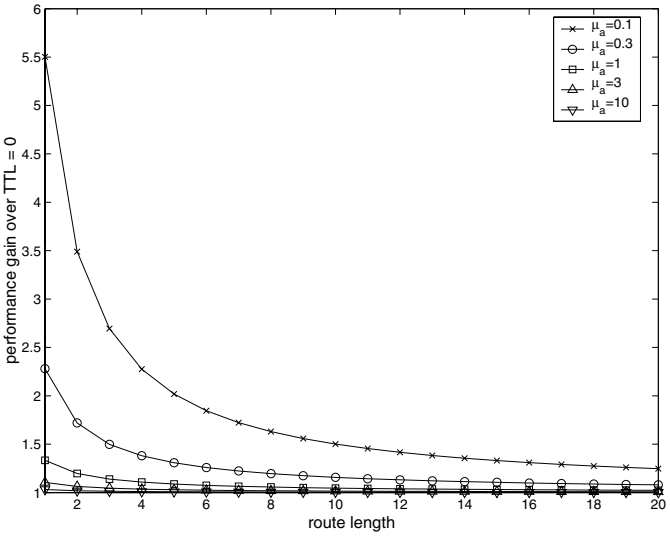

Fig. 5. Performance gain, achieved by using the optimal route-cache TTL, over systems not using route-cache (i.e., TTL=0), with source route caching.

We define the performance gain as the ratio between the expected delay of using a non-optimal TTL and the expected delay of using the optimal TTL. Figures 5 and 6 illustrate the performance gain over the no route-cache system (i.e., TTL=0) and the never-expiring route-cache system (i.e., TTL=∞), respectively, for various values of D and μ_a. Non-surprisingly, Figure 5 shows that the performance gain over TTL=0 decreases as the route length increases and as the route-request frequency decreases. Figure 6 shows that the performance gain relative to TTL=∞ is not a monotonic function of either the route length or the route-request frequency.[8] However, in the non-trivial cases

[8]When D is small, such that T_{opt} is comparable to μ_u, and if $\mu_a << \mu_u$,

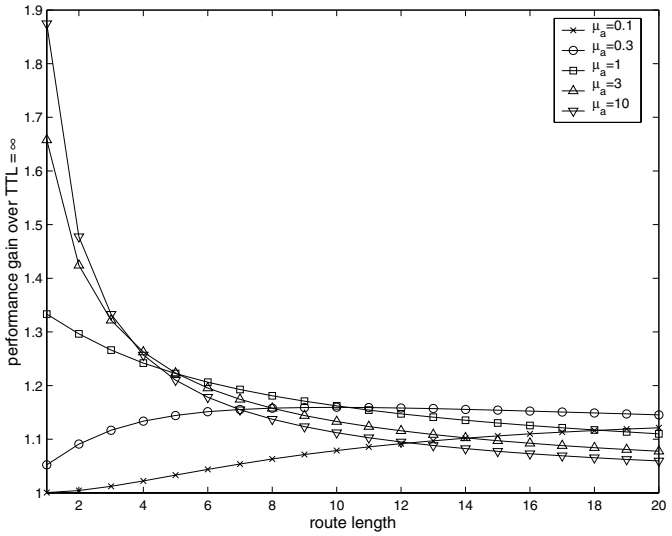

Fig. 6. Performance gain, achieved by using the optimal route-cache TTL, over using never-expiring route-cache (i.e., TTL=∞), with source route caching.

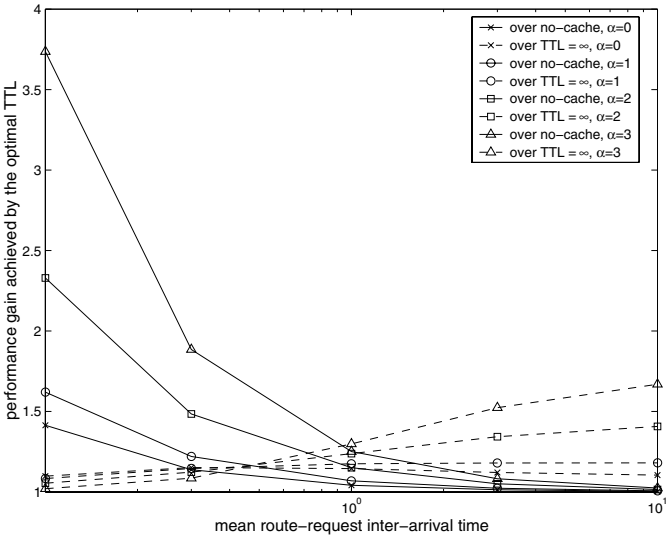

Fig. 7. Performance gain achieved by using the optimal source route-cache TTL, as a function of μ_a, for different traffic locality patterns.

where μ_a is not too small, the performance is indeed a decreasing function of the route length. This suggests that route-cache TTL optimization, in source route caching, is more advantageous in networks with localized traffic patterns such that the route lengths are not too long.

In [6] and [20], it was observed that the capacity of wireless ad hoc networks degrades catastrophically with the network size unless the network traffic is localized. Therefore, traffic localization is an essential characteristic of practical ad hoc networks that are scalable. Figure 7 illustrates the performance gain of the optimal route-cache TTL for different levels of traffic locality.

In describing the traffic locality, we have used a power distribution similar to what is described in [20]. Let π_D be the probability that a given route request is made to a destination of D hops away. If D is upper-bounded by D_{max}, the probability distribution function of D is defined as

$$\pi_D = \frac{D^{-\alpha}}{\sum_{i=1}^{D_{max}} i^{-\alpha}} , \tag{14}$$

where a larger value of α indicates a higher level of locality.

In Figure 7, we plot the performance gain of the optimal TTL, over the no route-cache system and the never-expiring route-cache system, for $\alpha \in \{0, 1, 2, 3\}$ and for $\mu_a \in [0.1, 10]$. We have assumed that $D_{max} = 20$. Figure 7 demonstrates that the performance gain achieved by using the optimal route-cache TTL is a fast increasing function of α. As a point of reference, when $\alpha = 3$ and $\mu_a = 1$, using the optimal TTL can reduce the routing delay of either a non-caching system or a never-expiring caching system by approximately 25%. Further improvement can be achieved in extreme cases. For example, when $\alpha = 3$ and $\mu_a = 0.1$, the performance gain of 3.7 is obtained over the non-caching system. As another example, when $\alpha = 3$ and $\mu_a = 10$, the performance gain of 1.7 is obtained over the never-expiring caching system. Therefore, route-cache optimization is especially important in the design of on-demand routing protocols where the traffic pattern must be localized in order to achieve scalability.

C. Performance Gain of the Optimal TTL with Intermediate Route Caching

Routing delay can be further decreased with intermediate route caching. Figure 8 illustrates the performance gain of using the optimal TTL versus a no route-cache system (i.e., TTL=0). Since the expected delay when TTL=0 is the same whether intermediate route-caches exist or not, compared with Figure 5, this figure shows that the expected performance achieved by using intermediate route caching can be much greater than that with using only source route caching. Furthermore, this benefit is the most prominent when the routes are long.

However, the potential penalty of keeping route caches for too long is also larger with intermediate routing caching. Figure 9 illustrates the performance gain of using the optimal TTL versus a never-expiring route-cache system (i.e., TTL=∞). Compared with Figure 6, this figure shows that keeping intermediate route-caches indefinitely can lead to expected delay that is more than 10-fold larger than if only source route-caches are keep indefinitely. It further shows that the performance gain of the optimal TTL is a monotonic function of both the route length and the route request frequency. When the routes are long, the negative effect of stale routes, due to large TTL, compounds with intermediate route caching, and hence the selection of optimal TTL values is more important. Likewise, when the mean time between route requests is long, the route caches are more likely to be invalid, and hence the penalty of using large TTL is more severe.

then $\frac{C(\infty)}{C(T_{opt})} \approx \frac{C(\infty)}{C(\infty)} = 1$. Furthermore, as μ_a increases from $0.1\mu_u$ to $10\mu_u$, the performance gain over $TTL = \infty$ increases from 1 to 2. When D is large, such that T_{opt} is small, $\frac{C(\infty)}{C(T_{opt})} \approx \frac{C(\infty)}{C(0)}$. Since $C(\infty)$ increases as μ_a decreases, the performance gain over $TTL = \infty$ is a decreasing function of μ_a.

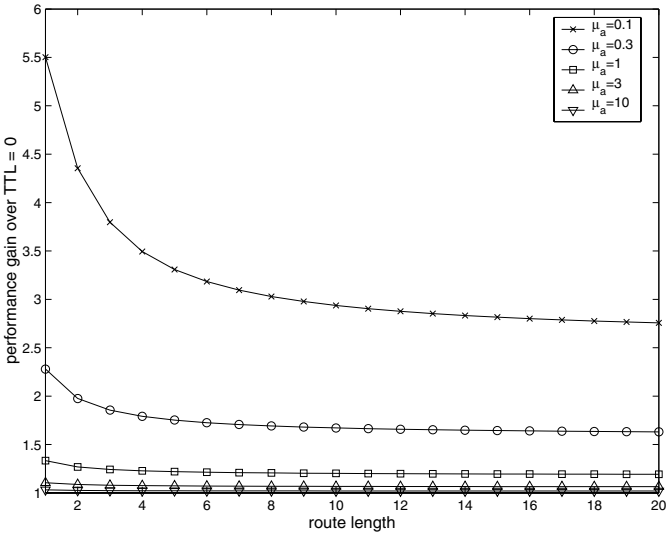

Fig. 8. Performance gain, achieved by using the optimal route-cache TTL, over systems not using route-cache (i.e., TTL=0), with intermediate route caching.

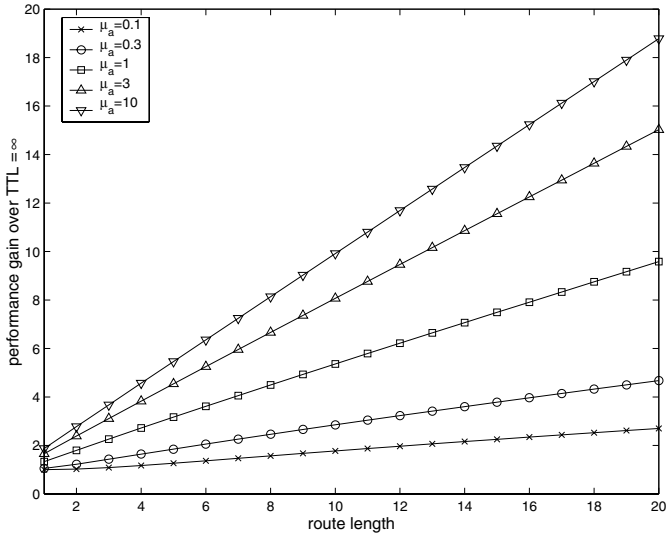

Fig. 9. Performance gain, achieved by using the optimal route-cache TTL, over using never-expiring route-cache (i.e., TTL=∞), with intermediate route caching.

To describe traffic locality, we again use the power distribution as in (14), where D now represents the *full* route length from n_s to n_d. In Figure 10, we plot the performance gain of the optimal TTL, over the no route-cache system and the never-expiring route-cache system, for $\alpha \in \{0, 1, 2, 3\}$ and for $\mu_a \in [0.1, 10]$. We have assumed that the maximal route length is 20 hops.

This figure shows that, with intermediate route caching, the performance gain of using optimally selected route caches values versus not using route caches is a fast increasing function of the traffic locality α. This is similar to the pure source routing case as shown in Figure 7. Furthermore, it shows that using intermediate route caching as oppose to only source route caching is more important when the network has less local traffic. For example, when $\mu_a = 0.1$ and the traffic pattern is uniform (i.e., $\alpha = 0$), using source route caching can reduce the routing delay by 1.5 times, while using intermediate route caching can reduce the routing delay by 3 times.

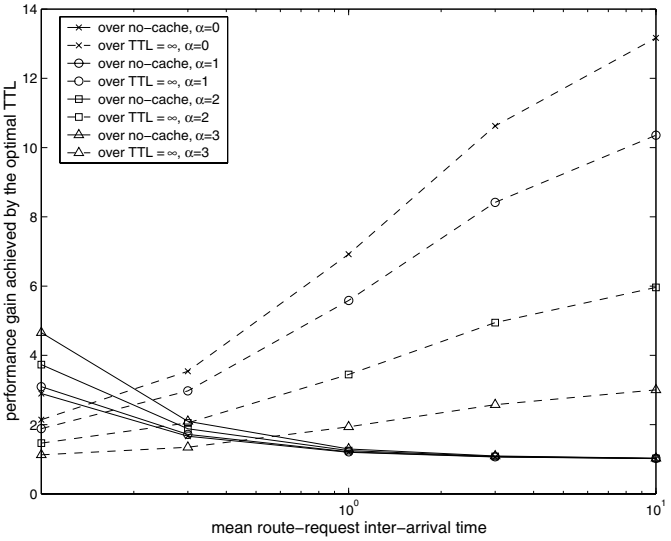

Fig. 10. Performance gain achieved by using the optimal intermediate route-cache TTL, as a function of μ_a, for different traffic locality patterns.

This figure also confirms the result shown in Figure 9 that, with intermediate route caching, there is a large penalty for keeping route caches for too long, especially when the network traffic flows have long spans. For example, in a network with $\mu_a = 1$ and $\alpha = 0$, compared with the optimal TTL strategy, when only purely source route caching is used, keeping route-caches indefinitely only increase the routing delay by 20%, but when intermediate route caching is used, keeping route-caches indefinitely increases the routing delay by 600%.

VII. Conclusions

Routing delay is one of the major drawbacks of on-demand routing protocols in mobile ad hoc networks. Route caching, with per-route lifetime limits, can significantly improve the performance of on-demand routing. However, the route-cache TTL needs to be properly chosen.

We have proposed analytical models to compute the expected routing delay when a source node or an intermediate node has a cached route with a given TTL value. We have also introduced numerical methods to determine the optimal TTL of a newly cached route. Our analytical results agree very well with the simulation results.

Through the proposed analytical and numerical frameworks, one can study the routing delay of a network given various operation parameters. The results of our analysis have shown that, compared with not using route-cache, the utilization of an optimal route-cache TTL strategy is the most effective when the traffic pattern is localized. As a point of reference, in networks where the requested routes have the inverse-cubic distance distribution, the optimal route-cache TTL can reduce the expected routing delay by a factor of 1.5 to 4 times. Furthermore, intermediate route caching can significantly out perform pure source

route caching when the traffic is not overly localized. For example, with uniform traffic and route requests ten times as frequent as link breakages, the optimal pure source route caching reduces the routing delay by a factor of 1.5, while the optimal intermediate route caching reduces the routing delay by a factor of 3.

In addition, our numerical results have demonstrated quantitatively the penalty for keeping indefinite route-caches versus using the proposed optimal route-cache TTL. The penalty is significantly more severe for intermediate route caching than for pure source route caching, with a more than 10-fold increase in routing delay. Therefore, when intermediate route caching is employed in ad hoc networks, the proposed route-cache TTL strategy provides an important framework for an overall efficient system design.

APPENDIX A

If $f_a(t) = \lambda_a e^{-\lambda_a t}$, then ${f_a}^*(s) = \frac{\lambda_a}{s+\lambda_a}$. From (2), we have

$$\begin{aligned} {f_c}^*(s) &= -\frac{1}{F_a(T)} \underset{z=s+\lambda_a}{Res} \frac{1-e^{-zT}}{z} \frac{\lambda_a}{s-z+\lambda_a} \\ &= \frac{\lambda_a}{1-e^{-\lambda_a T}} \frac{1-e^{-(s+\lambda_a)T}}{s+\lambda_a} \ . \end{aligned}$$

If $f_u(t) = \lambda_u e^{-\lambda_u t}$, then $f_r(t) = \lambda_u e^{-\lambda_u t}$, ${f_r}^*(s) = \frac{\lambda_u}{s+\lambda_u}$, and ${R_{X_1}}^*(s) = {R_r}^*(s) = \frac{1}{s+\lambda_u}$. From (4), we have

$$\begin{aligned} {R_{X_i}}^*(s) &= -\underset{z=s+\lambda_u}{Res} {R_{X_{i-1}}}^*(z) \frac{1}{s-z+\lambda_u} \\ &= {R_{X_{i-1}}}^*(s+\lambda_u) \ . \end{aligned}$$

It is easy to solve the above recursive equation to obtain ${R_{X_i}}^*(s) = \frac{1}{s+i\lambda_u}$, and, hence, ${f_{X_i}}^*(s) = \frac{i\lambda_u}{s+i\lambda_u}$.

Therefore, from (5), we have

$$\begin{aligned} Q_i(T) &= -\underset{s=i\lambda_u}{Res} \frac{\lambda_a}{1-e^{-\lambda_a T}} \frac{1-e^{-(s+\lambda_a)T}}{s+\lambda_a} \frac{1}{s} \frac{i\lambda_u}{-s+i\lambda_u} \\ &= \frac{\lambda_a}{i\lambda_u+\lambda_a} \frac{1-e^{-(i\lambda_u+\lambda_a)T}}{1-e^{-\lambda_a T}} \ . \end{aligned}$$

Substituting the above in (7), we have

$$\begin{aligned} C_{\lambda_u,\lambda_a}(T) =& 2LD + 2\lambda_a L \sum_{i=0}^{D-1} \frac{1-e^{-(i\lambda_u+\lambda_a)T}}{i\lambda_u+\lambda_a} \\ & - 4\lambda_a LD \frac{1-e^{-(D\lambda_u+\lambda_a)T}}{D\lambda_u+\lambda_a} \ . \end{aligned}$$

REFERENCES

[1] B. Bellur and R. G. Ogier, "A Reliable, Efficient Topology Broadcast Protocol for Dynamic Networks," *IEEE INFOCOM*, New York, March 1999.

[2] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," *ACM/IEEE MOBICOM*, 1998.

[3] S. R. Das, C. E. Perkins, and E. M. Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks," *IEEE INFOCOM*, 2000.

[4] J. J. Garcia-Luna-Aceves et al, "Analysis of routing strategies for packet radio networks," *Proceedings of IEEE INFOCOM'85*, Washington, DC, March 1985.

[5] J. J. Garcia-Luna-Aceves and M. Spohn, "Source-Tree Routing in Wireless Networks,", *Proc. IEEE ICNP 99: 7th International Conference on Network Protocols*, Toronto October 1999.

[6] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. Information Theory*, vol. 46, no. 2, March 2000.

[7] Z. J. Haas and M. R. Pearlman, "Providing Ad-hoc Connectivity with the Reconfigurable Wireless Networks," *Ad Hoc Networks,* Charles Perkins, ed., Addison Wesley Longman, 2000

[8] Z. J. Haas, J. Deng, B. Liang, P. Papadimitratos, and S. Sajama, "Wireless Ad Hoc Networks," John G. Proakis ed., *Wiley Encyclopedia of Telecommunications*, John Wiley & Sons, 2002.

[9] M. T. Heath, "Scientific Computing: An Introductory Survey," McGraw-Hill, 1997.

[10] G. Hollan and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *ACM/IEEE MOBICOM*, August, 1999.

[11] Y.-C. Hu and D. B. Johnson, "Caching strategies in on-demand routing protocols for wireless ad hoc networks," *ACM/IEEE MOBICOM*, 2000.

[12] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, "Scalable Routing Strategies for Ad Hoc Wireless Networks," *IEEE J. Selected Areas Commun.*, Special Issue on Wireless Ad Hoc Networks, vol. 17, no. 8, pp. 1369-1379, August 1999.

[13] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Lanouiti, L. Viennot, and T. Clausen, Draft "draft-ietf-manet-olsr-02.txt," July 2000.

[14] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based performance analysis of routing protocols for mobile ad-hoc networks," *ACM/IEEE Mobicom*, August 1999.

[15] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," in *Mobile Computing*, edited by T. Imielinski and H. Korth, Ch. 5, pp. 153-181, Kluwer Academic Publishers, 1996.

[16] J. Jubin and J. D. Tornow, " The DARPA packet radio network protocols," *Proceedings of IEEE (Special Issue on Packet Radio Networks)*, vol. 75, pp. 21-32, January 1987.

[17] J. J. Kelleher, "Tactical communications network modeling and reliability analysis: overview," JSLAI Report JC-2091-GT-F3 under contract DAAL02-89-C-0040, November 1991 (AD-A245339).

[18] S. Keshev, "An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network," Addison-Wesley, 1997

[19] B. M. Leiner, D. L. Nielson, and F. A. Tobagi, "Issues in packet radio network design," *Proceedings of the IEEE*, vol. 75, pp. 6-20, January 1987.

[20] J. Li, C. Blake, D.S.J. De Couto, H. I. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks," *ACM/IEEE MobiCOM*, Rome, Italy, July 2001.

[21] D. A. Maltz, J. Broch, J. Jetcheva, and D. B. Johnson, "The effect of on-demand behavior in routing protocols for multihop wireless ad hoc networks," *IEEE JSAC - Special Issue on Wireless Ad Hoc Networks*, vol. 17, no. 8, pp. 1439-1453, August 1999.

[22] S. Murthy and J.J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks," *ACM Mobile Networks and Applications Journal*, Special Issue on Routing in Mobile Communication Networks, October 1996.

[23] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, Third Edition, McGraw-Hill, 1991.

[24] V. Park and M. S. Corson, MANET Internet Draft "draft-ietf-manet-tora-spec-03.txt," November 2000.

[25] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *ACM SIGCOMM: Computer Communications Review*, vol. 24, no. 4, pp. 234-244, October 1994.

[26] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," *Proc. Second IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90-100, February 1999.

[27] C. E. Perkins, ed., *Ad Hoc Networking*, Addison-Wesley Longman, 2001.

[28] T. S. Rappaport, *Wireless Communications: Principles and Practics*, Prentice Hall, 1996.

[29] M. Sanchez, P. Manzoni, and Z. Haas, "Determination of critical transmission range in ad hoc networks," *Proc. IEEE Workshop on Multiaccess, Mobility and Teletraffic for Wireless Comm.*, October 1999.

[30] H. Takagi and L. Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals," *IEEE Trans. Communications*, vol. COM-32, pp. 246-257, March 1984.

[31] C-K. Toh, *Wireless ATM and Ad Hoc Networks: Protocols and Architectures*, Kluwer Academic Press, 1997.