Situation-Aware Caching Strategies in Highly Varying Mobile Networks

Yong Wang, Margaret Martonosi, and Li-Shiuan Peh
Princeton University, Princeton, NJ 08544
{yongwang, mrm, peh}@princeton.edu

Abstract

Some emerging applications in mobile ad hoc networks (MANETs) and mobile sensor networks (MSNs) have varying mobility patterns that entails different routing strategies at different times to maintain high performance. This requires the routing protocol to adapt to different situations for better overall routing performance. We propose a model-based approach to enable such situation-awareness for the Dynamic Source Routing (DSR) protocol in such challenging environments. Our model captures the access behavior of route cache (hit, miss, and false hit) and is simple enough to be used in real-world settings. We also present a feedback-based architecture that uses the model outputs to cope with mobility changes by adjusting caching strategy on-the-fly. We validate our model against ns-2 simulations for a variety of scenarios, including a real-world mobility. Our results show that the model can aptly drive adaptive routing that leads to consistent performance improvement over DSR for the scenarios considered.

1. Introduction

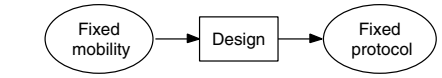
The dynamics of mobile networks make efficient protocol design extremely challenging as mobility causes network topology to constantly change in unpredictable ways. In some emerging real-world applications [1, 2], atypical mobility patterns such as one that alternates between highly mobile and very static movements are prevalent. Since routing is governed by complex interactions between node mobility and protocol behavior, small changes in either of them may have significant impact on the overall routing performance. To maintain routing performance under such varying mobility, a routing protocol needs to *adjust its behavior on-the-fly* to adapt to mobility dynamics. However, previous studies mainly focus on typical mobilities [3], with key routing components hand-tuned for expected mobility patterns.

In this paper, we explore situation-aware routing strategies in highly varying mobile networks. In particular, we study DSR-like protocols where the effectiveness of route caching is of critical importance [4, 5]. In this context, the problem narrows down to understanding how node mobility impacts route cache access behavior and how to adjust the caching strategy to cope with mobility phase shifts.

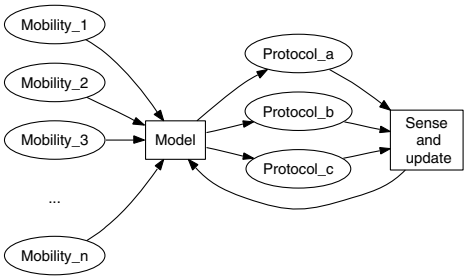
Understanding mobility and adapting to it is difficult after a protocol is deployed because some metrics are difficult to capture at run-time. Simulation can be used to collect such metrics. However, simulation speed becomes a significant problem when applied to such scenarios. We performed several simulations of a 50-node mobile network for 1000 seconds on a machine with 2.2GHz Pentium 4 processor and 512MB RAM under different mobility scenarios; they took from 15 minutes to 1 hour. Further, to explore the design space, we may need many such simulation runs.

To overcome such shortcomings, we take a more analytical approach and develop a route cache model that we use dynamically to facilitate timely decision-making. Our model is a Discrete Time Markov Chain model that describes the access behavior of a route cache structure derived from DSR. The model accepts as input metrics collected from online measurement and outputs various cache access rates. As node mobility and protocol behavior are both incorporated into the model, the overall routing performance is projected as a function of both of them. A route cache access has three states: hit, miss, and false hit (wherein the route stored in the cache no longer exists due to node mobility). A false hit leads to extra processing time, network bandwidth waste and even packet drops.

Our approach is best understood in the context of a specific example. Consider a mobile sensor network for wildlife tracking across large regions with no communications infrastructure [6]. Since ZebraNet is in essence a MANET with resource-constrained sensor nodes, we do not make any distinctions between MSNs and MANETs for the following discussion. In such a network, nodes move throughout an environment to gather information about their surroundings. Periodically, logged data is aggregated to the base station, which is also constantly moving to increase the probability of data homing success. In this scenario, node movements are unpredictable and highly varying. By feeding collected mobility metrics at the base sta-



(a) Previous approach.



(b) Our model-driven approach.

Figure 1. A model-driven adaptive routing architecture. Our approach can be applied to different protocols with every protocol customized on-the-fly to mobility changes.

tion into our model, however, one can predict routing performance and dynamically adjust protocol configurations when necessary.

Previously, as shown in Figure 1(a), once a network is deployed, the protocol remains fixed and can not significantly change when mobility varies. Our approach, shown in Figure 1(b), introduces a dynamic feedback loop. Periodically, mobility data is collected through the data collection process and sent to the base station. Metrics, such as route lifetime, are extracted from the data and fed into our model which then outputs the current route cache access behavior. Based on model outputs, proper protocol adjustment decisions are then disseminated to each participating node in the system via an updating mechanism such as that proposed in [7]. This process continues as new mobility data is collected. The result is a robust system that can run autonomously for a long period of time.

The major contributions of this work are summarized as follows: First, we present a realistic route cache model to assess the impact of mobility on overall routing performance in a protocol based on route caching. Second, our model can be used across a range of mobilities, and has error rates less than 5% even for real-world mobility models, validated against the *ns-2* simulator, with order of magnitude savings in running time. Third, we demonstrate through a case study how a feedback-driven architecture (Figure 1(b)) can be realized using our model to drive adaptive routing under highly varying mobility. This serves as a proof of concept of how analytical models can be leveraged to drive adaptive routing in realistic applications.

The rest of the paper is organized as follows. Section 2 reviews related work. In Section 3, we give an overview of our model. In Section 4, we present the mathematical construction and validation results of our model. We then present a case study using a wildlife tracking application in Section 5. Finally, we conclude the paper with future directions.

2. Related Work

MANETs have become a significant research focus in recent years, with emerging application studies including wildlife tracking, vehicular ad hoc networks [8] and others. Many mobility datasets from such real-world applications have been collected and archived at CRAWDAD (the Community Resource for Archiving Wireless Data At Dartmouth) [9] and are publicly available to the research community. In addition, we also consider a real-world zebra mobility model from our own research group [10]. Next, we divide our discussion of related work into two categories, those related to analytical techniques and those related to mobility characterization.

Analytic techniques. In terms of prior work with analytic techniques, the bulk of modeling has concentrated on the analysis of MAC protocols, for either single-hop [11] or multi-hop networks [12]. These works provide solid understanding of behavior in the MAC layer. However, overall routing behavior cannot be explained without reference to higher layer protocols. A model beyond the MAC layer is critical for understanding end-to-end routing behavior. Several related analytical work in higher networking layers have been proposed. Zhou et al.[13] developed performance models of reactive routing in the network layer for an unreliable static sensor network. Their main analytical results are with regard to control overhead, while ours are with regard to overall performance metrics. Viennot et al. [14] also proposed a model for analyzing protocol control overhead, but as the aim is to be general, many important details are missing and it becomes difficult to isolate the impact of node mobility, not to mention leveraging the model for protocol optimizations.

Research by Shah et al. [15] has modeled data delivery rates in a mobile sensor network. Their model uses an asynchronous store-and-forward communication pattern suitable for Delay Tolerant Networks [1]. Therefore, no end-to-end route semantics are considered. Samar et al. [16] develop an analytical framework to investigate the timing behavior of the communication links, while our study is based on route lifetime. They evaluate their framework in a synthetic random environment, while we use realistic mobility traces.

Mobility Characterization. There is a large body of literature in this direction and we only discuss the most relevant work here. Work by Bai et al. [17] studies various mobility statistics in a mobile network, while we focus on mobility metrics that have a direct impact on routing per-

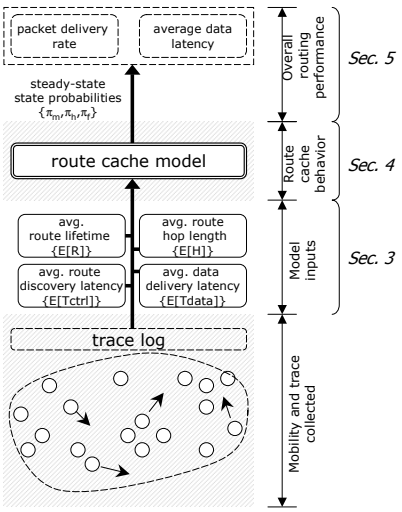


Figure 2. Modeling framework and data flow.

formance. Follow-up work by the same authors provides a detailed study of how mobility impacts path duration statistics in MANETs [18]. None of the above work, however, ties mobility to protocol behavior as ours does.

3. Model Overview

In this section, we present an overview of the route cache model in the context of DSR routing. A more detailed mathematical construction is presented in Section 4. Figure 2 sketches all quantities of interest and their relationships.

3.1. Routing Overview

DSR is a source routing protocol consisting of two major components: *route discovery* and *route maintenance*, both of which operate entirely on-demand. During route discovery, a node actively scouts through the network to find routes to an intended destination. Route maintenance is the process by which the sending node determines if the route used is broken and takes repair actions when necessary.

When a data packet arrives, a request is made to the route cache for its intended destination. If a route is found (a cache hit), the packet is forwarded to the next node along the route. We call the selected route the *candidate* route. All other routes with the same destination are *auxiliary* routes. If the request misses (a cache miss), a route discovery is initiated, with the packet forwarded along the newly discovered route if found. If the request has a hit but the candidate route is stale, which is called a "false hit" in our case, delays may be introduced at intermediate hops to fix the broken route. Even worse, the packet may be dropped if the error cannot be fixed en route.

3.2. Model Outputs

Our route cache model essentially outputs three steady-state probabilities, the probability of cache hit (π_h), miss (π_m), and false hit (π_f). Since route cache access is on the critical path of routing, its access behavior is tightly connected to and reflective of the overall routing performance. This is illustrated in the upper dashed frame in Figure 2. In this study, we consider two performance metrics: *packet delivery rate*, defined as the fraction of successfully delivered data packets and *average data latency*, defined as data latency averaged among all data packets delivered.

Table 1. Model input.

Notation	Description
$E[R]$	Average route lifetime
$E[T_{\text{ctrl}}]$	Average route discovery latency
$E[T_{\text{data}}]$	Average data delivery latency
$E[L]$	Average route hop length

Intuitively, π_h has a positive effect on both packet delivery rate and data latency. The higher the π_h, the higher the two performance metrics since packets are delivered along good routes and do not need to wait for re-discoveries and error recoveries. π_m, on the contrary, has a negative effect on data latency because a route discovery has to be followed that adds to the total latency. For packet delivery rate, π_m's impact depends on the current traffic conditions and the queueing behavior of the protocol. If no packets are dropped due to these factors, the packet can be successfully delivered after a route discovery with valid replies. π_f, however, always has a negative impact on routing performance because following a stale route will always incur more processing overhead to repair such errors. Very likely, packets will be dropped by failing to fix such errors en route. π_f can happen during packet forwarding, route reply, and packet salvaging by providing a stale route, causing poor packet delivery rate and increased data latency.

However, π_f is hard, if not impossible, to collect in practice after a protocol is deployed. Since our model exposes the probability of being in the *false hit* state, it can be used as an indicator of such *non-sensical* protocol behaviors. By factoring such knowledge into a system, these otherwise unachievable metrics can be leveraged at system runtime.

3.3. Model Inputs

Table 1 lists the input parameters for our route cache model. The parameter $E[R]$ denotes the average lifetime of routes in a network and its inverse denotes the rate at which valid routes become stale. The shorter the average route lifetime, the more frequently a route breaks. Since a route breakage triggers route maintenance in a reactive protocol[1], $E[R]$ is used to characterize node mobility. Previous work [19] has shown that route lifetime is useful in capturing mobility properties. We choose to use an average here

[1]There are other factors, such as having multiple paths in the route cache, that influence the triggering rate of route maintenance operations. They are further discussed in Section 4.

Table 2. Model parameters.

Notation	Description
κ	Rate of a route becoming stale. Also the transition rate out of state *H*.
μ_1	Stale route detection and invalidation rate. Also the transition rate out of state *F*.
μ_2	Route recovery service rate. Also the transition rate out of state *M*.
p_{xy}	The transition probability from state x to state y. Both x and y can be one of the following: h, m, or f.

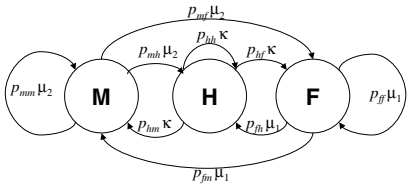


Figure 3. The Discrete Time Markov Chain model for a single node.

for its simplicity and amenability to analysis. The use of route lifetime *distributions* will improve model accuracy, as further discussed in Section 4.3.

The parameters $E[T_{\text{ctrl}}]$ and $E[T_{\text{data}}]$ are used to capture the timing behavior of two critical protocol-related operations. $E[T_{\text{ctrl}}]$ denotes the average latency of a broadcast-based route discovery process and $E[T_{\text{data}}]$ denotes the average data packet delivery latency. The route discovery process populates the route cache and provides routes for data packets. Because data packets are used implicitly for signaling routing errors in DSR, the data delivery latency determines how quickly a stale route in cache is detected.

The parameter $E[L]$ denotes the average route length in terms of hop count. It depends on node mobility and traffic pattern and is useful in determining the delay for detecting a route error, as discussed in Section 4.2.4.

We estimate $E[R]$ by bookkeeping route creation and dead events in the route cache. We only track routes that have existed at least once in the cache for the following reasons: First, only routes stored in the cache can influence the route cache behavior. Second, naively bookkeeping all potential routes is computationally expensive. For a network with n nodes, the number of potential routes is on the order of $n!$, which grows exponentially as n becomes larger.

$E[T_{\text{data}}]$ can be measured by timestamping data packet departure and arrival events. $E[T_{\text{ctrl}}]$ can be collected in a similar way as $E[T_{\text{data}}]$. However, since it measures the latency from when a route request is sent out until a **valid** reply is received in our model, we need to check the validity of discovered routes. This is possible with off-line trace processing wherein omniscient knowledge of route validity is available. $E[L]$ can be measured by recording the number of hops traversed for each packet successfully delivered.

4. Route Cache Model

In this section, we describe the construction of our route cache model. Model parameters are listed in Table 2.

4.1. Assumptions

A1. (Cold start miss) We assume no cold start misses once the route cache reaches steady state.

A2. (Capacity miss) We assume no capacity misses. This is reasonable because capacity misses are independent of mobility-induced misses and can be eliminated easily by increasing cache size.

A3. (Channel models) A noisy channel may reduce the actual route lifetime due to transmission failures. We do not consider route breakage related to this factor and leave it as a future research direction.

A4. (Traffic pattern) We only consider saturated traffic workload with all nodes continuously pumping data to the base station. This assumption matches a large range of real-world traffic patterns, such as the one used in ZebraNet. For on-demand protocols whose operations depend on traffic distribution, the correlation between route cache behavior and traffic is lowered.

A1-3 allow us to assume that the average route breakage rate κ depends only on node mobility, which is abstracted as route lifetime timers. We will show later in this section that even with such simplifications, our model still produces accurate results.

4.2. Model Mechanics

4.2.1. Virtual Detection State (F)

Figure 3 illustrates the three-state Discrete Time Markov Chain model for a route cache. In state **M**, the node has no candidate path for an initiating packet and a cache miss occurs. In state **H**, the node has a valid candidate path for an initiating packet and a normal hit occurs. In state **F**, the node has a candidate route that is stale due to mobility and a false hit occurs.

One major contribution of our model is that state F, the false hit state, actually does not exist in a realistic protocol, hence the name *virtual* detection state. For many reactive ad hoc routing protocols, including DSR, it is impossible to detect route failures instantaneously and they will inevitably enter this artificial. Having such a virtual detection state expose valuable information that is essential to routing performance. Such information is not possible with traditional approaches.

4.2.2. Rate of Cache Staleness (κ)

The rate of cache staleness, or the rate transiting out of state H, is κ. Intuitively, for some period proportional to $E[R]$, the current candidate route will break which leads the node to state *F* or *M*, depending on factors such as the route

discovery and maintenance mechanism used, node mobility and traffic workload. The average actual route lifetime should be smaller than $E[R]$ because it only represents the average lifetime in the route cache. Therefore, we need some adjustment to calculate κ using $E[R]$. In our model, we estimate κ as $\frac{1}{\gamma E[R]-\frac{1}{\mu 1}}$ with γ a constant denoting the ratio between the actual lifetime to $E[R]$. We use a γ of 0.5 in our study, assuming that when a route is selected for routing, its residual lifetime is uniformly distributed between $(0, E[R])$. We subtract $1/\mu_1$ from $E[R]$ because during route recovery, the elapsed route lifetime cannot be used for routing.

4.2.3. Route Discovery (μ_2)

The average route discovery rate is denoted as μ_2, which is also the transition rate out of state M. We estimate μ_2 simply as $\frac{1}{E[T_{\text{ctrl}}]}$.

4.2.4. Stale Route Detection and Recovery (μ_1)

With the introduced virtual detection state *F*, we denote the rate that a node detects an invalid route as μ_1, which is also the transition rate out of state F.

Route error detection in a reactive routing protocol is divided into two phases: a *negative detection phase* and an *active error notification phase*. In the negative detection phase, data traffic is used implicitly to detect a link error when the packet reaches the broken link. In the active error notification phase, a control packet is sent to notify the source of this error.

Therefore, the negative detection latency $E[T_{\text{negd}}]$ depends on the number of hops to traverse along the broken route until the packet reaches the broken link. If we assume that the probability of route breakage is distributed uniformly over all hops from the source to the destination, we can estimate the average hop count traversed before a route breakage as $\sum_{i=1}^{E[L]} i = \frac{1}{2}(E[L]+1)$. Therefore, the negative detection latency can be estimated as $E[T_{\text{data}}] \times \frac{E[L]+1}{2E[L]}$. We can calculate the active error notification latency $E[T_{\text{actn}}]$ in a similar way. The only difference is that $E[T_{\text{ctrl}}]$ is a two-way delay that spans $2E[L]$ hops in total. Therefore, $E[T_{\text{actn}}]$ should be calculated as $\frac{1}{2}E[T_{\text{ctrl}}] \times \frac{E[L]+1}{2E[L]}$. These two latencies add up to the average route recovery latency and μ_1 is estimated as $\frac{1}{E[T_{\text{negd}}]+E[T_{\text{actn}}]}$.

4.2.5. Influence of Protocol Designs

In this section, we discuss the influence of protocol designs on state transition probabilities.

In our model a transition out of state M only happens after a route is discovered, so p_{mm} should be 0. For discussion convenience, we directly denote the probability from state M to state H as p_m and the probability from state M to state F as $1-p_m$. Thus, p_m represents the probability that a route reply is valid and $1-p_m$ the probability that a route reply is stale. Since we estimate μ_2 using only valid route replies (i.e., only a valid route reply finishes a route request), p_m is 1 in our case.

Since a route will always enter the virtual detection state due to the reactive maintenance mechanism, p_{hh} and p_{hm} are 0 and we denote the probability from state *H* to state *F* as p_h. Thus, $p_h \cdot \kappa$ represents the rate of a broken route not being detected immediately. In our model, p_h is approximated as 1 because DSR mainly depends on data traffic for route error detection.

The parameters p_{fm}, p_{fh} and p_{ff} denote the three transition probabilities out of the false hit state. They all depend on the number of backup routes available when the candidate route breaks because only when there is no route to the destination does the node enter state M. Otherwise, the breakage of the candidate route will not incur a new route request and the state may transit to either H or F, depending on the validity of the new candidate route selected. Therefore, p_{fm} represents the probability of having no backup routes when the route being used is invalidated. Hence, $p_{fm} = 1$ is the case where there is only one route for each destination. Whenever this route is broken, it enters state M. On the other hand, $p_{fm} = 0$ is the case where there are always valid auxiliary routes when the route being used is invalidated.

4.2.6. State Probabilities

In this section, we calculate the equilibrium state probabilities of the model, denoted as $\boldsymbol{\pi} = [\boldsymbol{\pi_m}, \boldsymbol{\pi_h}, \boldsymbol{\pi_f}]$.

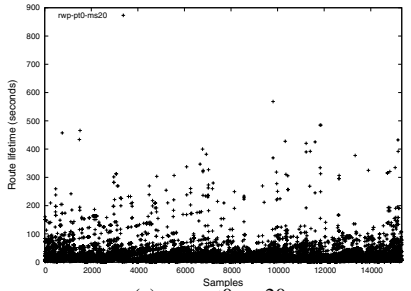
We simplify the mathematical calculation by predetermining parameters that can directly be estimated from the protocol behavior, which are p_h and p_m. From earlier discussions, both of them should be equal to 1 for DSR. Therefore, we have the following global balancing equations for the steady state of our Markov chain model, which can be solved to produce the limiting state probabilities:

$$\begin{cases} \pi_h \kappa &= \pi_f(1-p_{ff})\mu_1 \\ \pi_f p_{fm} \mu_1 &= \pi_m \mu_2 \\ \pi_m + \pi_h + \pi_f &= 1 \end{cases}$$

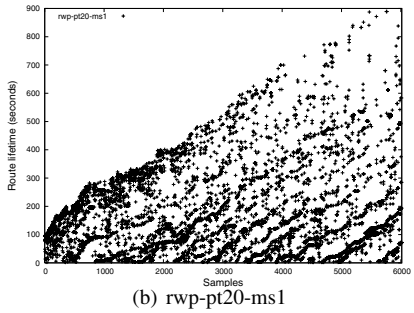
4.3. Route Cache Model Validation

In this section, we present model validation results against *ns-2* simulations using the Random WayPoint (RWP) model. Our approach can also be used with other mobility models since only high-level mobility metrics are used in our model. In Section 5, we will validate our model on a real-world mobility. We seek to study (i) how close our analytical model results are to simulated outcomes of DSR, under different mobility scenarios, and (ii) how our model parameters affect its accuracy.

We simulate a network of 50 nodes in a 1100m×1100m grid. Each node has a radio range of 250m. Initially, nodes



(a) rwp-pt0-ms20



(b) rwp-pt20-ms1

Figure 4. Route lifetime distributions for different mobility scenarios.

are randomly distributed across the defined area. We generate 30 communication pairs randomly and use a packet rate fixed at 2pkt/s. We do not adopt a higher injection rate because we need a network sufficiently provisioned such that the effects of mobility are isolated from effects of congestion [20]. We use UDP traffic in packets of 512 bytes. Traffic is injected from 900s to populate the route cache and all metrics are measured starting from 1000s. The first 900s is used for the mobility model to reach its steady state, a method proposed in [21] to fix one deficiency in RWP. Other simulation setups, including the radio propagation model, the MAC protocol used, and link bandwidth are the same as those used in [22].

4.3.1. Validation Results

Table 3 compares the analytical results against simulation results for four mobility scenarios. Each scenario is represented as rwp-pt$x$-ms$y$ with x its pause time and y its maximum speed. For a moderately- to highly-mobile network, our model is reasonably accurate with error rates less than 10%. This indicates that in steady-state, our model successfully captures the state of the route cache in a mobile environment. For static scenarios such as rwp-pt20-ms1, however, our model has a much higher error rate.

This could be explained by looking at the model inputs. Figure 4 illustrates the distribution of route lifetimes observed by a randomly selected node (14) for both rwp-pt0-ms20 and rwp-pt20-ms1. The observations for other nodes are similar. The samples are collected by post-processing the simulation trace. For a moderately- to highly-mobile network, route lifetimes tend to be in the same order of magnitude, as shown in Figure 4(a). For a less mobile network, however, route lifetimes have very high variability, as shown in Figure 4(b).[2] In such a network, skew will be introduced by representing route lifetimes as an average. This problem can be solved by using separate estimations for short and long lifetimes. More generally, one could include distributions of route lifetimes in the model.

Table 3 also illustrates the savings in running time to derive the same quantities of interest. Model computation time is negligible, while input gathering time dominates. It should be noted that when the model is dynamically deployed, input gathering time is spent on running scripts on trace files, which normally takes less than 1 minute. Therefore, the total elapsed time using our model is less than 1 minute. On the contrary, it usually takes more than 15 minutes to finish one simulation run.

4.3.2. Future Refinements of the Model

Overall, the model's accuracy is already quite good. Nonetheless, there is still room for improvement. The modest disagreements between analytical and simulated results can be explained by certain simplifying assumptions regarding state transition probabilities. We discuss these below.

The first source of error has to do with the transition probability from state *H* to *M*. For the implementation of DSR in *ns-2*, this probability is greater than 0, due to various protocol optimizations not considered in the model. One such optimization is route error propagating, which spreads route error messages aggressively to suppress their propagation. Thus, a route recovery process can be finished without incurring the two-phase operation. Another optimization is cache purging that times out a route after some duration. This also may lead a node in state *H* directly to *M*.

The second source of error arises in the presence of a false route reply. In our model, we only account for valid route replies when calculating $E[T_{\mathrm{ctrl}}]$. This approach may under-estimate the probability entering state *F*. In other words, we assume there is no state transition from *M* to *F*. In contrast, such effects are present in our simulations. This explains why for most scenarios, π_f calculated by our model is smaller than that from simulation.

It would be interesting to study whether our model still works under those cases. We are also working on extending the model such that it will not be limited by a specific cache policy.

[2] While there appears to be a correlation between simulation time and route lifetimes, this is an artifact of our statistics gathering method: routes that last longer than the simulation time can not be tracked.

Table 3. Route cache model validation.

Scenario	π_m			π_h			π_f			Running Time		
	Sim	Model	%diff	Sim	Model	%diff	Sim	Model	%diff	Sim	Model	sim/model
rwp-pt0-ms20	0.124	0.136	9.6%	0.526	0.538	2.3%	0.349	0.326	6.6%	22:14m	49s	27
rwp-pt10-ms20	0.243	0.238	2.1%	0.354	0.358	1.1%	0.403	0.404	0.2%	13:22m	54s	15
rwp-pt20-ms20	0.166	0.154	7.2%	0.456	0.475	4.2%	0.378	0.371	1.9%	24:37m	57s	26
rwp-pt20-ms1	0.075	0.053	29.3%	0.884	0.906	2.5%	0.042	0.41	2.4%	17:47m	39s	27

Table 4. Validation results for a real-world mobility.

Category	Sim	Model	%diff
π_m	0.130	0.133	2.3%
π_h	0.509	0.486	4.7%
π_f	0.361	0.376	4.0%

5. Case Study

In this section, we validate our model using a real-world mobility. We also evaluate its effectiveness in a wildlife tracking application.

5.1. Validation Using Real-world Mobility Data

Our mobility trace is collected from a mobile sensor network deployed in January 2004 by the ZebraNet group. A number of collars (sensor nodes) are attached to the body of zebras. Each collar recorded its GPS data every 8 minutes for a total of 32 hours. Due to extreme weather and waterproofing issues, as well as antenna problems, only one tracking collar returned uninterrupted movement data for the whole 32-hour duration. Due to such limitations, we extended the collected data to create a semi-synthetic mobility model as follows. We collect node speed and turn angle *distributions* from the observed data. Then we create other node movements by uniformly selecting from the node speed and turn angle distribution collected in the first step. Next, we cast the trace data into a RWP model that can fit into the *ns-2* simulator. Although this approach may miss some temporal correlation information between zebras, it is one step closer to reality.[3]

Originally, the nodes move in an area of 6km×6km. We scale the area size to 1km×1km and randomly distribute the nodes in the defined area. In order to calculate metrics like cache false hit rate, we also incorporate other needed information about connectivity and shortest route length at any instance between all communicating pairs, so that the trace file can be directly used in *ns-2* simulations. The rest of the simulation configuration is the same as that described in the previous section.

Table 4 shows that all three probabilities of interest have error rates below 5%.

[3]From extended data collection in a second, June 2005 deployment, we found that there is little node correlation in movements, and thus our assumption here is valid.

5.2. A Case for a Model-Driven Dynamic Protocol

In this section, we present a case study demonstrating how to leverage our model to drive adaptive routing decisions on-the-fly.

DSR uses route caching extensively in both route discovery and route reply. It adopts a passive route maintenance mechanism for fixing stale routes. The problem with such a scheme is its slow response to mobility changes. Given that all routing decisions are based on route cache state, the performance may suffer from using stale information. By exposing the route cache states, our model helps to predict route caching performance in a timely fashion and guide protocol adjustment when necessary.

Specifically, we show how route cache reply options can be switched on and off dynamically to improve routing performance by leveraging π_f. This idea can be used for other optimizations, such as route discovery backoff, given proper models for those components. The mobility used is derived from the zebra trace with node speeds varied. We divide the mobility trace into three phases, with 1000s to 1300s phase 1, 1300s to 1600s phase 2, and 1600s to 1900s phase 3. Traffic starts at 900s to populate route cache. We reduce the node speed to $\frac{1}{10}th$ of the original speed for phase 1, increase the node speed by 3 times for phase 2, and increase the node speed by 6 times for phase 3. The trace produced this way demonstrates a significant variation from one phase to another and is fairly realistic as zebras normally move in walk-run-walk phases [6]. Moreover, we expect such phases of varying mobility to typify many other mobile network scenarios as well.

The set of experiments we performed uses a similar configuration as described in the last section, with a total of 50 nodes and 30 CBR flows. We use a radio range of 150m here because a 250m radio range for this mobility trace results in severe radio interferences in our simulation. We study the instantaneous packet delivery rate and normalized routing overhead for three configurations listed in Table 5 and compare their performance with the original DSR. By *instantaneous*, we mean that results shown in the y-axis are not aggregated from the start of the simulation. They demonstrate instant behavior for that period. This allows for a better observation of the adaptation behavior.

The three configurations differ from DSR in their decisions as to when to switch *route cache reply* on/off for the

Table 5. Configuration options studied.

	1^{st} phase	2^{nd} phase	3^{rd} phase
DSR (always-on)	on	on	on
Strategy 1 (off-low)	off	on	on
Strategy 2 (always-off)	off	off	off
Strategy 3 (off-high)	on	off	off

Table 6. Data latency comparison.

Traffic rates	DSR	Strategy 3	%improvement
2pkts/s	6.2s	5.2s	16%
4pkts/s	4.1s	3.0s	27%
8pkts/s	2.7s	2.1s	22%

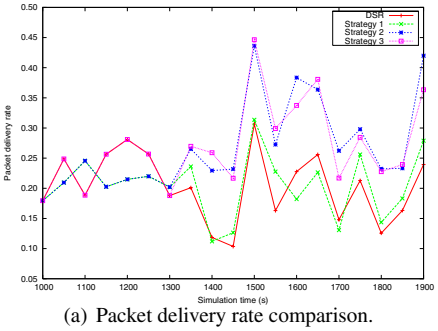
three phases. Intuitively, for a highly-mobile scenario, route cache replies should be disabled because the information stored in the cache is prone to be invalid; using a route cache for answering route requests can lead to inaccurate routing decisions. For a less mobile scenario, as route cache knowledge is normally accurate, enabling route cache replies will increase locality, reduce latency and save resources. The decision is based on π_f.

Since phase 1 is very static and phases 2 and 3 are both highly mobile, the strategy that disables route cache replies for phases 2 and 3 and enables route cache replies for phase 1, i.e., Strategy 3, should have the best performance, the highest packet delivery rate and the lowest routing overhead and energy consumption. Strategy 1, which has the opposite configuration options to Strategy 3, should have the worst performance. Strategy 2 should stay in the middle because most of the time, it has the right option (for phases 2 and phase 3). The original DSR just switches on route cache replies all the time.

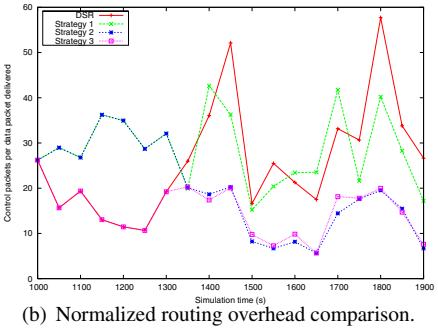
For such an adaptive scheme to work, a node needs to be able to detect the mobility phase changes. For this section, we pre-program such information into the simulation for them to make decisions in a distributed manner. We will discuss a practical phase detection method in Section 5.3.

Table 6 compares the average data latency between DSR and Strategy 3 using the common set of packets they successfully delivered. Intuitively, Strategy 3 saves route repair time by not following stale routes. However, by switching off route cache replies, it needs more time to get a route because it only accepts replies from the intended destination. On the contrary, DSR saves time by getting a route from other nodes' route cache. However, if the route obtained is stale, it incurs additional delays fixing errors at intermediate hops. Table 6 shows that Strategy 3 has a smaller average data latency for all traffic rates. This indicates that for the mobility trace we studied, the utility of switching off route cache replies is higher than keeping it on.

The latency improvement for 4pkts/s and 8pkts/s are both higher than that for 2pkts/s. When more packets are injected into the network, the penalties of using stale routes become higher because the contention for the medium is more severe than at lower packet rates. As a result, the benefits of using dynamic configurations become more salient.



(a) Packet delivery rate comparison.

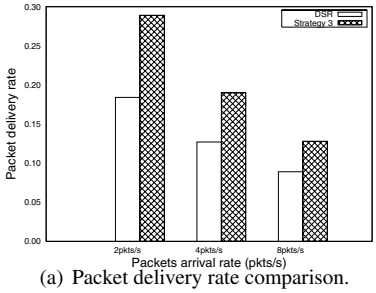


(b) Normalized routing overhead comparison.

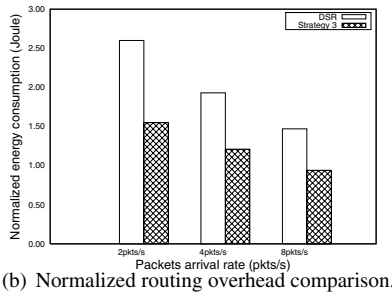
Figure 5. Performance comparison of different strategies. Results are collected from simulation and the packet injection rate is 2pkts/s.

Figure 5(a) shows the instantaneous packet delivery rate at an interval of 50s. Figure 5(b) shows the routing overhead averaged among all data packets delivered. Routing overhead refers to control packets sent for route discovery and route maintenance. The normalized routing overhead is used as a measure of routing efficiency, including energy efficiency.

For the first 300s from 1000s to 1300s, Strategies 1 and 2 have the lowest packet delivery rates and the highest average overhead. This is because phase 1 is very static with all the nodes moving very slowly. For such an environment, stale information is very rare and route cache replies should be enabled to maximize locality. For phases 2 and 3, as nodes move really fast, stale information begins to flood the network. In this case, a route cache reply should be avoided because there is a higher probability that the benefits of employing route caching can be overwhelmed by the disadvantages it brings. Strategies 2 and 3, which switch off route cache replies for phases 2 and 3, have a higher packet delivery rates and lower routing overhead than Strategy 1. Because Strategy 3 adapts to varying mobility correctly, it achieves the best of both worlds and has the best performance compared to all other options, including DSR. The



(a) Packet delivery rate comparison.



(b) Normalized routing overhead comparison.

Figure 6. Impact of traffic rate.

improvement in packet delivery rate is consistently higher than 40% and the maximum improvement is up to 120%. The reducton in routing overhead is consistently higher than 40% and the maximum reduction is up to 66%.

Figure 6 illustrates the impact of packet arrival rate on the performance of our dynamic optimization. As the injection rate increases, the demand for bandwidth increases too and we believe that our approach should still outperform a scheme that is unaware of mobility changes. Simulation results shown in Figure 6 affirm this.

5.3. Detection of Mobility Changes Using π_f

In this section, we propose a practical phase detection method using π_f. It uses a feedback loop as introduced in Section 1. When enough mobility data are collected at the sink, we extract mobility metrics as input to the route cache model. Since all input parameters can be obtained by running `awk` and `python` scripts on the collected mobility trace, this process takes only tens of seconds on a modern PC. We then use our model to output π_f, which takes only a couple of seconds at most. This obtained information is disseminated to all nodes in the network through a data dissemination protocol. Decision is then made at each node regarding its caching strategy.

Figure 7 shows instantaneous π_f in our simulation, with different sampling intervals (epochs). Each point represents the π_f re-evaluated at the end of each epoch. Figure 7(a) shows π_f with a sampling interval of 20s. There are two jumps with the first one starting at 1300s and the second starting at 1600s. They conform to the mobility changes in 1300s and 1600s, respectively and are emphasized using

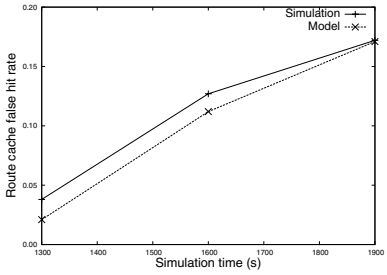


Figure 8. Comparison of π_f changes detection (simulation vs. model). The epoch length is 300s and the packet injection rate is 2pkts/s.

two dotted lines. Because the sampling rate is very high, the variation is pretty high compared to variations using longer sampling intervals. Figure 7(b) shows the results for a sampling interval of 100s. The variation is much smaller and the changes in π_f are consistent with changes in mobility. Figure 7(c) shows the results for a sampling interval of 300s, which exactly matches the three mobility phases. This in turn indicates that there is a salient change in π_f in response to mobility changes. The results demonstrate that π_f can be used for predicting changes in mobility with reasonable accuracy.

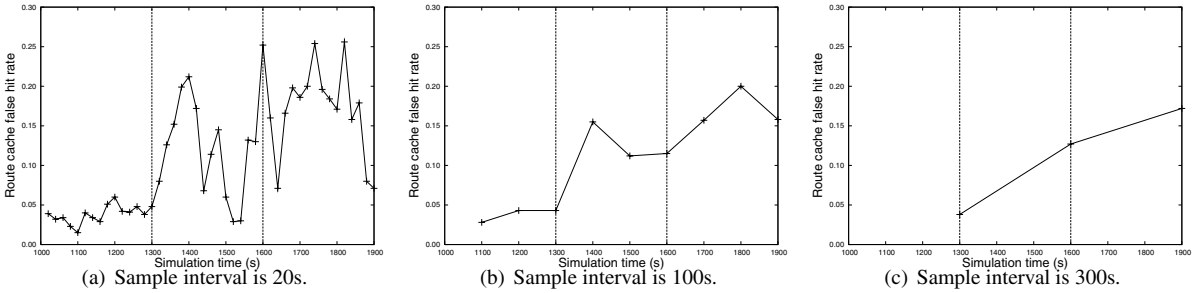
Figure 8 compares the estimation of instantaneous π_f using our model to that using simulation for a sampling interval of 300s. As the figure shows, the estimation from our model matches that from simulation very well. This demonstrates that our model can aptly capture the changes in mobility at runtime with high accuracy.

Finally, we compare the running time to derive π_f by model to that by simulation using the semi-synthetic mobility trace as described in Section 5.1. The simulation took **13:05 minutes** on a machine with 2.2GHz Pentium 4 processor and 512MB RAM. However, it took only **25 seconds** to output π_f using our model. Simulation time becomes even longer when dynamically trading off between different parameter configurations because several simulation runs are then necessary.

5.4. Discussions

While our model and its use already demonstrate significant and useful performance improvements, there is still room for future refinements. We discuss some of them here.

First, our current approach requires the sink to collect mobility traces from all nodes in the network. This is a challenging task for a MSN, even a modest-size one. Second, our current model only derives steady-state probabilities, which requires a certain mobility phase to be long enough to be observed. We do not view this as a significant weakness, since short-lived mobility changes are not likely to be worth optimizing for. Third, our model tries to use a sin-



(a) Sample interval is 20s. (b) Sample interval is 100s. (c) Sample interval is 300s.

Figure 7. π_f changes for DSR in response to mobility changes.

gle metric (the false hit) to capture the route cache behavior for nodes distributed across the network, which works for a homogeneous network wherein nodes experience similar mobility patterns. However, for other realistic mobility patterns, a distributed algorithm may perform better. Again, we see this as a topic for future work.

Although our approach has such limitations, it offers new opportunities for using analytical models in real-world setting and further work is anticipated to improve on this.

6. Conclusions and Future Work

In this paper, we presented an analytical model of route cache for DSR-like reactive protocols. We also illustrated how to leverage the model for dynamic protocol configuration to adapt to varying mobility, using a real-world mobility. When validated against detailed network simulations, our model produces fairly accurate results with typical errors less than 5% for a real-world mobility and less than 10% for synthetic RWP-based mobilities. To the best of our knowledge, our work is the first to model the behavior of a route cache in MANETs. Model-driven adaptation can improve instantaneous packet delivery rate by up to 120% and data latencies by 16-27%.

As part of our on-going work, we are investigating ways to estimate model parameters such as route discovery latency using analytical models too. This allows an adaptive model-driven scheme that depends purely on node mobility and is thus easy to deploy in real systems. We are also looking at ways for adaptively selecting the sampling interval used in mobility changes detection, trying to capture all potential optimization opportunities.

References

[1] Delay Tolerant Networking Research Group Webpage, http://www.dtnrg.org/.

[2] J. J. Blum, A. Eskandarian, and L. J. Hoffman, "Challenges of intervehicle ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, 2004.

[3] M. Ilyas, Ed., *The Handbook of Ad hoc Wireless Networks*. CRC Press, 2002.

[4] Y.-C. Hu and D. B. Johnsoan, "Caching strategies in on-demand routing protocols for wireless ad hoc networks," in *Proc. ACM MobiCom*, 2000.

[5] M. Marina and S. Das, "Performance of route caching strategies in dynamic source routing," in *Proc. IEEE WNMC*, 2001.

[6] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," in *Proc. ACM ASPLOS*, 2002.

[7] T. Liu and M. Martonosi, "A middleware system for managing autonomic, parallel sensor systems." in *Proc. ACM PPOPP*, 2003.

[8] H. Wu, R. Fujimoto, R. Guensler, and M. Hunter, "MDDV: a mobility-centric data dissemination algorithm for vehicular networks," in *Proc. ACM VANET*, 2004.

[9] D. Kotz and T. Henderson, "CRAWDAD: A community resource for archiving wireless data at dartmouth," *IEEE Pervasive Computing*, vol. 4, no. 4, 2005.

[10] P. Zhang, C. M. Sadler, S. Lyon, and M. Martonosi, "Hardware design experiences in ZebraNet," in *Proc. ACM SenSys*, 2004.

[11] M. Carvalho and J. Garcia-Luna-Aceves, "Delay analysis of IEEE 802.11 in single-hop networks," in *Proc. IEEE ICNP*, 2003.

[12] M. M. Carvalho and J. Garcia-Luna-Aceves, "A scalable model for channel access protocols in multihop ad hoc networks," in *Proc. ACM MobiCom*, 2004.

[13] N. Zhou, H. Wu, and A. A. Abouzeid, "Reactive routing overhead in networks with unreliable nodes," in *Proc. ACM MobiCom*, 2003.

[14] L. Viennot, P. Jacquet, and T. H. Clausen, "Analyzing control traffic overhead versus mobility and data traffic activity in mobile ad hoc network protocols," *ACM WINET Journal*, vol. 10, no. 4, 2004.

[15] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: Modeling a three-tier architecture for sparse sensor networks," in *Proc. IEEE SNPA*, 2003.

[16] P. Samar and S. B. Wicker, "On the behavior of communication links of a node in a multi-hop mobile environment," in *Proc. ACM MobiHoc*, 2004.

[17] F. Bai, N. Sadagopan, and A. Helmy, "IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of RouTing protocols for Adhoc NeTworks," in *Proc. IEEE Infocom*, 2003.

[18] N. Sadagopan, F. Bai, B. Krishnamachari, and A. Helmy, "PATHS: analysis of path duration statistics and their impact on reactive MANET routing protocols," in *Proc. ACM MobiHoc*, 2003.

[19] Y. Wang, M. Martonosi, and L.-S. Peh, "MARio: Mobility-adaptive routing using route lifetime abstractions in mobile ad hoc networks," *ACM MC2R*, vol. 8, no. 4, 2004.

[20] Y.-C. Hu and D. B. Johnson, "Exploiting MAC layer information in higher layer protocols in multihop wireless ad hoc networks," in *Proc. IEEE ICDCS*, 2004.

[21] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," in *Proc. IEEE Infocom*, 2003.

[22] 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," in *Mobile Computing and Networking*, 1998.