Optimal Cache-based Route Repair for Real-time Traffic

Chuan Han, Yaling Yang and Yongxiang Peng
Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA, 24061
Email: {hanc, yyang8}@vt.edu, pengyx78@gmail.com

Abstract—Real-time applications in ad hoc networks require fast route repair mechanisms to minimize the interruptions to their communications. Cache-based route repair schemes are popular choices since they can quickly resume communications using cached backup paths after a route break. In this paper, through thorough theoretical modeling of the cache-based route repair process, we derive the optimal cache-based route repair policy. This optimal policy considers both the overhead of the route repair schemes and the promptness of the repair action. The correctness and advantages of our optimal policy are validated by extensive simulations.

I. INTRODUCTION

In mobile ad-hoc networks (MANETs), the constant movement of wireless nodes and the fading of channels can cause frequent route breaks for on-going communications. Real-time applications, on the other hand, require stable routes to ensure satisfactory performance. Hence, fast route repair mechanisms that can quickly reestablish new routes upon route breaks must be used in MANETs to minimize the interruption to on-going real-time communications.

Among the existing designs of ad hoc routing protocols, on-demand protocols, such as DSR [1] and AODV [2], have been proven to be able to quickly find paths in highly mobile networks and hence are very popular for supporting real-time flows. In these protocols, a source node re-floods the entire network to search for a new route when the current route breaks. Such flooding-based route discovery can return up-to-date new paths. However, flooding-based route discovery is usually very expensive in terms of message overhead. Hence, it can potentially cause network-wide congestion. When congestion happens, it may require a fairly long time to find a path. Therefore, path repair schemes that solely rely on flooding-based route discovery cannot satisfy the need of many time critical real-time applications.

As a solution to the above problem, cache-based route repair schemes [3], [4], [5], [6] have been proposed to reduce the frequency of flooding. These schemes exploit the fact that a round of flooding-based route discovery usually finds multiple paths to the destination. The source node can cache these paths as backups. When the route in use breaks, the source first tries to use the cached paths to continue its communications before it resorts to flooding. Hence, the cached paths may provide quicker route repair and impose less message overhead. However, aggressive use of the cached paths for route repair does not always bring benefit since there is always a chance that the cached paths are also broken due to the mobility of nodes [7]. Sending packets along broken cached paths can only cause excessive packet losses and delay. Therefore, there exists an interesting trade-off point between the flooding-based route discovery and the cache-based route repair schemes. Most of existing works on cache-based route repair [3], [4], [5], [6], [8], unfortunately, only focus on heuristic designs and simulation-based evaluation. There is only one work that provides theoretical analysis on cache-based route repair [9]. This work, however, only computes the interruption and overhead caused by using the cached paths and ignores the interruption and overhead caused by flooding-based route discovery. This omission is not tolerable since the flooding-based route discovery often imposes much higher overhead and incurs much larger interruption duration than cache-based route repair. Therefore, there is still a severe lack of correct insights into the trade-off point between cache-based route repair and flooding-based route discovery.

In this paper, we provide a mathematical analysis on this trade-off point and derive an optimal cache policy that is the best strategy between flooding and cache-based repair. Our unique contributions are listed as follows:

• While using cached paths for route repair has been proposed in many existing literature, we are the first to formulate an analytical framework to identify the optimal trade-off point between cache-based route repair and flooding-based route discovery in MANETs. This analytical framework accurately captures the factors that affect the performance of route repair strategies. These factors include path length, mobility of nodes and network size.
• Based on our analytical framework, an easy-to-implement optimal cache policy is proposed and proved for cache-based route repair schemes. This policy finds the optimal trade-off point between flooding and cache-based route repair. Its correctness and advantages are analytically proved and then validated by simulations.

The remaining part of this paper is organized as follows.
The system model is presented in Section II. The optimization objective is developed in Section III and its relationship with the characteristics of cached paths is studied in Section IV. The optimal cache policy is derived in Section V, and the policy is generalized to cover more optimization objectives in Section VI. The performance advantages of the optimal cache policy are validated by simulations in Section VII. Finally, Section VIII concludes the whole paper.

II. SYSTEM MODEL

To derive the optimal strategy for cache-based route repair, we first need to have a model of the network mobility, a model of the network, and a model for the routing protocol. Then, we can use these models to derive the optimal cache policy.

A. Mobility Model

In this paper, the mobility of nodes is modeled by the Semi-Markov Smooth (SMS) mobility model [10]. We use SMS mobility model because it has been shown in [10] to be able to accurately capture realistic node movement. For example, it can capture physical motion’s smooth nature, can adapt to diverse network application scenarios and can ensure the node distribution does not change over time. All of these realistic features are not present in other more traditional but more naive mobility models such as the random-way-point model.

Briefly speaking, the SMS model works as follows. In the SMS model, node movement consists of four consecutive phases: speed up phase (α phase), middle smooth phase (β phase), slow down phase (γ phase), and pause phase.

At the beginning of the speed up phase, the node is stationary. Then, it accelerates from the stationary state to the target speed \( v_α \). The direction \( φ_α \) does not change in this process. The target speed \( v_α \in [v_{min}, v_{max}] \), the direction \( φ_α \in [0, 2π] \), and the total number of time steps \( α \in [α_{min}, α_{max}] \).

The parameters \( v_α, φ_α, \) and \( α \) are independent and uniformly distributed.

After the speed up phase, the node transits into the middle smooth phase. In this phase, the initial speed \( v_0 = v_α \) and the initial direction \( φ_0 = φ_α \). The speed and direction change from \( v_α \) and \( φ_α \) at each time step based on a memory level parameter \( ζ \in [0, 1] \). This memory level parameter captures the temporal correlation of velocity between consecutive steps. This phase lasts \( β \) steps, where \( β \in Z^+ \) is uniformly distributed over [\( β_{min}, β_{max} \)].

After the middle smooth phase, the node enters the slow down phase. In this phase, the node comes to a complete stop within \( γ \in Z^+ \) steps, where \( γ \) is uniformly distributed over [\( γ_{min}, γ_{max} \)]. The direction \( φ_γ \) is randomly chosen, but is correlated to node movement direction at the end of the middle smooth phase. The direction \( φ_γ \) does not change in this phase.

B. Network Model

Consider a MANET, where there are \( N \) mobile nodes in the network, and each node’s movement is independent and identically distributed (i.i.d.). For each node, the average speed is \( \bar{v} \), and the radio coverage radius is \( R \). When two nodes are within the communication range of \( R \), we define that there exists a link between the two nodes, i.e., these two nodes can communicate with each other. When there is a set of links which can connect the source node with the destination node, we define that there exists a path between the source node and the destination node.

C. Protocol Model

Our model of cache-based route repair can capture any routing protocol that uses flooding-based route discovery, regardless if the routing protocol is source routing (e.g., DSR) or hop-by-hop routing (e.g., AODV). In our model, we assume that a source node first uses flooding-based route discovery to find a path to its destination. It sends out a broadcast route request (RREQ) message. Each intermediate node may receive multiple copies of the RREQ message and only selects one to forward to its own neighbors to limit the message overhead. When the destination receives multiple route request messages, it can discover multiple paths and return this information to the source node through a route reply (RREP) message. The best path among the returned paths is then set up as the path to relay data packets to the destination as follows. In the hop-by-hop routing case (e.g., AODV), the routing tables at the relaying nodes along the best path are setup by the RREP message that travels back to the source. In the source routing case (e.g., DSR), after the source receives the RREP message, the entire selected best path is copied to the headers of the data packets to the destination. The other paths discovered by the flooding process are returned by the RREP to the source and are cached as backup paths for path repair.

We denote this best path as \( p_1 \) and the set of cached paths as \( I = \{p_1, p_2, p_3, \ldots, p|I|\} \), where \( I \) is a sorted list of paths and is called the path cache. We assume that only the node-disjoint paths to the source node are cached. This is a common practice widely used in multipath routing and path repair to ensure that the cached backup paths are independent of the status of the path in use. Essentially, this means that \( p_0, p_1, p_2, \ldots, p|I| \) are node-disjoint. The flooding process has a large routing message overhead, especially when the network scale is relatively large or the network resource (e.g. bandwidth and energy) is limited. The routing messages sent during the flooding process are called the flooding packets in this paper.

Upon a route break, if \( I \) is not empty, the source picks the first cached path \( p_1 \in I \) for route repair. It sends out an exploration packet to the destination along \( p_1 \). If \( p_1 \) is already broken, a route error message is generated when the exploration packet reaches the first broken link. This route error message travels back to the source to notify the failure of \( p_1 \). Then, \( p_1 \) is removed from the path cache \( I \) and the next path \( p_2 \) in \( I \) is explored for path repair. This cache-based path exploration process continues until finally a valid path \( p_1 \) is found. In such a case, the destination receives the exploration packet and returns the confirmation to the source. The confirmation message will trigger the right routing table information to be set up for the paths \( p_1 \). In the case of hop-by-
hop routing (e.g., AODV), all the intermediate nodes on $p_i$ will have their routing table updated by the confirmation message. In the case of source routing (e.g., DSR), only the routing table in the source will be updated. Then, the source removes $p_i$ from the path cache $\mathcal{I}$ and resumes its communication using $p_j$. The cache-based path exploration is successful in this case. If $\mathcal{I}$ becomes empty before a valid path is found, the source node resorts to flooding-based route discovery for repairing the broken route.

III. PROBLEM FORMULATION

To formulate the problem of optimal cache-based route repair strategy, let us first take a detailed analysis of the cache-based route repair process. Upon a route break, a source node has two possible actions to repair the route. The first action is flooding-based route discovery, which happens when the source node has no cached path in $\mathcal{I}$. The source gets the up-to-date set of paths from the flooding process but pays for the high message overhead of flooding.

The second action is to explore the head-of-line path in the path cache $\mathcal{I}$. The paths in $\mathcal{I}$ are learned from the last flooding process. The exploration action happens when $\mathcal{I}$ is not empty. If the explored path is already broken, the exploration process incurs message overhead and increases the duration of communication interruption.

Essentially, the frequency, overhead and delay of the above two path repair actions greatly affect the interruption duration and the routing overhead of a real-time session. These two actions are in turn determined by the cache policy, which determines the set of paths cached in $\mathcal{I}$ and these paths’ sequence in $\mathcal{I}$. For example, if the number of paths in $\mathcal{I}$ is small, the frequency of flooding is large. If the paths and their sequence in $\mathcal{I}$ is not appropriately selected, the exploration action may often fail. Therefore, the problem of optimal cache-based route repair is essentially a design problem for the optimal cache policy, which jointly minimizes the interruptions to communications and the message overhead.

We formulate this goal for the optimal cache policy as an objective function that captures both the message overhead and the time interruption of the two path repair actions. For a real-time communication session that lasts $T$ seconds, the objective function is defined as:

$$\max_{\pi(\mathcal{I})} \frac{N_d^T}{\pi(\mathcal{I}) N_o^T}, \quad (1)$$

where $\pi(\mathcal{I})$ is a cache policy that sets the path cache $\mathcal{I}$, $N_d^T$ is the number of data packets transmitted during the lifetime of the real-time session, and $N_o^T$ is the number of overhead packets for establishing and repairing routes for the real-time session. The physical meaning of the objective function is to maximize the ratio between the number of real-time data packets delivered to the destination and the number of overhead packets for repairing and establishing routes. This objective function captures both the time and message overhead for establishing and repairing routes. If a cache policy creates long interruptions to the real-time session, it reduces the number of real time packets that can be delivered during time $T$ and hence decreases the value of the objective function. If a cache policy creates too much routing overhead, it increases the denominator in the objective function and hence also reduces the objective function’s value.

It is important to note that we do not claim that (1) is the only possible formulation of objective functions. Actually, our analysis for solving (1) can be generalized to cover a large range of possible objective functions. In Section VI, we will discuss these other possible formulations. Also note that when packet sizes are different for different types of packets, we can also take into account the packet size in the objective function (1) by multiplying the packet transmission time before the packet numbers. This packet transmission time for the same packet type can be approximated as a constant. This is because the variations in control packet (e.g., RREQ) sizes usually are negligible compared to the dominant packet transmission overhead in physical and MAC layers. Hence, considering packet sizes simply means multiplying constant packet transmission time to the variables that we want to optimize in (1). This modification of the optimization problem in (1) does not affect our analytical process. Therefore, we focus on the objective in (1) in the rest of this paper.

IV. RELATIONSHIP BETWEEN OBJECTIVE FUNCTION AND CACHED PATHS

In this section, we identify the relationship between objective function in (1) and the characteristics of cached paths, so that an optimal cache policy can be designed based on this relationship. First, a mathematical model of the cache-based route repair process is established. Based on this model, the objective function in (1), which is defined over the entire lifetime of a session, is converted to a tractable formulation over a short time period. Finally, by analyzing this short time period, the objective function is expressed by the characteristics of cached paths.

A. Renewal Model of the Cache-based Route Repair Process

To establish a mathematical model of the cache-based route repair process, note that when a flooding-based route discovery is completed at time $t$, the cache $\mathcal{I}$ in the source is filled with newly discovered paths and all the old cached paths are removed. This removal of old cached paths means that the future communication states of the real-time session has no dependence on its states before time $t$. Hence, the communication states of the real-time flow can be modeled as a renewal process, where each flooding-based route discovery is a renewal event. We define the interval between two consecutive flooding events, i.e. the inter-flooding period, as a repair cycle. As shown in Fig. 1, each repair cycle begins with a flooding-based route discovery that lasts $t_f$ time. For each occurrence of route break in the cycle, the source node explores the cached paths one by one until it finds one valid path to resume the communication session. Each exploring period is the duration for exploring one cached path for route repair and $t_c$ denotes this duration for exploring path $p_i \in \mathcal{I}$.
The number of exploring periods in the repair cycle equals the number of paths initially cached in \( I \) after the flooding at the beginning of the cycle.

**B. Objective Function Defined over the Repair Cycle**

With the above renewal model of the path repair process, we can translate the objective function in (1), which is defined over the entire duration of a real-time session, to an objective function that is defined over a repair cycle. The purpose of this step is to simplify the mathematical analysis since it enables us to focus on a relatively short period.

To translate the objective function in (1), we first divide the duration of the real-time session \( T \) as the combination of multiple repair cycles. Assuming there are \( L \) repair cycles over the time interval \( T \), the objective function in (1) becomes:

\[
\max_{\pi(I)} \frac{N_d}{N_o} = \max_{\pi(I)} \frac{\sum_{l=1}^{L} N_d^l}{\sum_{l=1}^{L} N_o^l}, \tag{2}
\]

where \( N_d^l \) is the number of data packets transmitted during the \( l \)th repair cycle, and \( N_o^l \) is the number of overhead packets transmitted during the \( l \)th repair cycle.

Assume that \( T \) is much larger than the inter-flooding period, i.e., \( L \) is a very large number. Since each inter-flooding period is independent and identically distributed, by the law of large numbers, the objective function in (2) becomes

\[
\max_{\pi(I)} \frac{\sum_{l=1}^{L} N_d^l}{\sum_{l=1}^{L} N_o^l} \approx \max_{\pi(I)} \frac{E[N_d]}{E[N_o]} = \max_{\pi(I)} \frac{N_d}{N_o}, \tag{3}
\]

where \( N_d \) is the expected number of data packets transmitted during a repair cycle, and \( N_o \) is the expected number of overhead packets transmitted during a repair cycle. Essentially, the objective function in (3) is defined over a repair cycle and hence we can further analyze it by closely examining the formation of a repair cycle.

**C. Objective Function v.s. Path Characteristics**

In this section, we derive the relationship between the objective function in (3) and the characteristics of cached paths in \( I \). First, denoting the real-time session data rate as \( r \) packets per unit time, the objective function in (3) becomes

\[
\max_{\pi(I)} \frac{N_d}{N_o} = \max_{\pi(I)} \frac{T_d r}{N_f + N_e}, \tag{4}
\]

where \( T_d \) is the expected time used for data transmission during a repair cycle, \( N_f \) is the expected number of flooding packets transmitted during the flooding-based route discovery, and \( N_e \) is the expected number of exploring packets transmitted for exploring cached paths during a repair cycle. Note that exploring a cached path may either succeed when the path is alive or fail when the path is broken. Hence, \( N_e \) includes the exploring packets incurred in both cases.

Since \( r \) is a constant, (4) is equivalent to:

\[
\max_{\pi(I)} \frac{T_d}{N_f + N_e} = \max_{\pi(I)} \frac{E[T_d^\text{max} - \sum_{i \in I} t_i^e]}{N_f + \sum_{i \in I} N_i}. \tag{5}
\]

where the term \( T_d^\text{max} \) is the duration of a repair cycle minus the time spent for flooding the network (See Fig. 1). The term \( t_i^e \) is the time used for exploring the \( i \)th path in the set \( I \) during a repair cycle. The term \( N_i \) is the expected number of overhead packets transmitted when exploring the \( i \)th path in \( I \). Note that, in general, the time used for exploring paths is much less than the time length of the inter-flooding period minus the flooding period, i.e., \( \sum_{i \in I} t_i^e \ll T_d^\text{max} \). Therefore, the objective function in (5) can be approximated as follows

\[
\max_{\pi(I)} \frac{E[T_d^\text{max} - \sum_{i \in I} t_i^e]}{N_f + \sum_{i \in I} N_i} \approx \max_{\pi(I)} \frac{E[T_d^\text{max}]}{N_f + \sum_{i \in I} N_i}. \tag{6}
\]

To compute \( N_f \) in (6), note that flooding packets include the RREQ packets and the route reply (RREP) packets. The number of RREQ packet equals the total number of nodes \( N \) in a fully connected network. The number of RREP packets is determined by the hop count of the path used by the RREP message. Both the numbers of RREQ and RREP packets are not related to path cache \( I \). In large scale MANETs, in general, the number of RREQ packets is much larger than the number of RREP packets. To simplify the analytical model, we assume that the expected number of flooding packets can be approximated as a constant during a repair cycle, and we further use the total number of nodes \( N \) in the network to approximate the total number of flooding packets. With this approximation, the objective function becomes

\[
\max_{\pi(I)} \frac{E[T_d^\text{max}]}{N + \sum_{i \in I} N_i}. \tag{7}
\]
Hence, only the $E[T_I^{\text{max}}]$ and $\sum_{p_i \in I} N_i$ parts in (7) can be optimized through the control of the paths cached in $\hat{I}$ and the sequence of cached path exploration. We next identify their relationship with the characteristics of cached paths.

1) Relationship between $E[T_I^{\text{max}}]$ and $\hat{I}$: Denote $T_i$ as the path lifetime of the $i$th path (i.e. $p_i$) in $\hat{I} = \mathcal{I} \cup p_0$. Note that all the cached paths in $\hat{I}$ are node-disjoint. Based on rigorous theoretical analysis and simulation validation in [11], under the SMS mobility model, the lifetime of a path can be approximated as an exponentially distributed random variable with parameter

$$\lambda = \frac{n\bar{v}}{R},$$  \hspace{1cm} (8)

where $n$ is the hop count of the path, $\bar{v}$ is the average node speed, and $R$ is the node radio coverage radius. Therefore, $T_i$ is continuous, independent and exponentially distributed with a parameter $\lambda_i$. Using this property, the relationship between $E[T_I^{\text{max}}]$ and $\hat{I}$ can be identified as follows.

Lemma 1. \hspace{1cm} 1

The expected value of $E[T_I^{\text{max}}]$ is given by

$$E[T_I^{\text{max}}] = E \left[ \max_{p_i \in \hat{I}} \{ T_i \} \right]$$

$$= \sum_{p_i \in \hat{I}} \frac{1}{\lambda_i} - \sum_{i<j} \frac{1}{\lambda_i + \lambda_j} + \sum_{i<j<k} \frac{1}{\lambda_i + \lambda_j + \lambda_k} - \ldots + (-1)^{|\hat{I}|-1} \frac{1}{\sum_{p_i \in \hat{I}} \lambda_i}. \hspace{1cm} (9)$$

Proof: Note that $T_I^{\text{max}}$ is the duration of an inter-flooding period minus the duration for a flooding-based route discovery. As shown in Fig. 1, a $T_I^{\text{max}}$ duration starts at the end of a flooding-based route discovery process and lasts until all paths in the path cache $\hat{I}$ have been explored and turned out broken, which ends the repair cycle by triggering the next flooding-based route discovery. Hence, the duration of $T_I^{\text{max}}$ is the maximum lifetime among all the paths in $\hat{I}$, which can be expressed as

$$T_I^{\text{max}} = \max_{p_i \in \hat{I}} \{ T_i \}. \hspace{1cm} (10)$$

Since $T_i$ is independent and exponentially distributed with parameter $\lambda_i$,

$$P\{T_I^{\text{max}} \leq t\} = P\{T_i \leq t, \forall p_i \in \hat{I}\} = \prod_{p_i \in \hat{I}} P\{T_i \leq t\} = \prod_{p_i \in \hat{I}} (1 - e^{-\lambda_i t}).$$  \hspace{1cm} (11)

Therefore, we have

$$P\{T_I^{\text{max}} > t\} = 1 - P\{T_I^{\text{max}} \leq t\} = 1 - \prod_{p_i \in \hat{I}} (1 - e^{-\lambda_i t})$$

$$= \sum_{p_i \in \hat{I}} e^{-\lambda_i t} - \sum_{i<j} e^{-(\lambda_i + \lambda_j) t} + \sum_{i<j<k} e^{-(\lambda_i + \lambda_j + \lambda_k) t} - \ldots + (-1)^{|\hat{I}|-1} \sum_{p_i \in \hat{I}} \lambda_i t.$$  \hspace{1cm} (12)

Notice that $T_i \geq 0$ for any $p_i \in \hat{I}$. It follows that

$$E[T_I^{\text{max}}] = \int_0^\infty P\{T_I^{\text{max}} > t\} dt$$

$$= \sum_{p_i \in \hat{I}} \frac{1}{\lambda_i} - \sum_{i<j} \frac{1}{\lambda_i + \lambda_j} + \sum_{i<j<k} \frac{1}{\lambda_i + \lambda_j + \lambda_k} - \ldots + (-1)^{|\hat{I}|-1} \frac{1}{\sum_{p_i \in \hat{I}} \lambda_i}. \hspace{1cm} (13)$$

2) Relationship between $N_i$ and $\hat{I}$: We next compute the relationship between the $\sum_{p_i \in \hat{I}} N_i$ in (7) and the path cache $\hat{I}$.

When exploring the $i$th cached path $p_i$ in $\hat{I}$, there are two possible outcomes: $p_i$ is still valid or $p_i$ is already broken. When $p_i$ is still valid, the number of overhead packets for exploring $p_i$ is $2n_i$, where $n_i$ is the hop count of $p_i$. When $p_i$ is broken, the number of overhead packets can be calculated as follows. Assume that the broken link is uniformly distributed among the $n_i$ links. When there is a link break, the expected number of the overhead packets for the $i$th path is $2n_i(n_i - 1).$ Hence, the expected number of overhead packets for exploring the $i$th path is

$$2n_i a_i + n_i (1 - a_i) = n_i (1 + a_i),$$  \hspace{1cm} (14)

where $a_i$ is the probability that the $i$th path is still alive. Notice that the path lifetime is independent and exponentially distributed with parameters $\lambda_i = \frac{n\bar{v}}{R}$. As the $i$th path in $\hat{I}$, $p_i$ is explored only when paths $p_0, p_1, ..., p_{i-1}$ are all broken. Hence, based on (8), the probability that $p_i$ is still valid when it is explored is

$$a_i = P\{T_j \leq T_i, \forall 0 \leq j \leq i - 1\} = \prod_{j=0}^{i-1} P\{T_j \leq T_i\}$$

$$= \prod_{j=0}^{i-1} \frac{\lambda_j}{\lambda_j + \lambda_i} = \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i}. \hspace{1cm} (15)$$

Therefore, (7) can be derived to

$$\max_{\pi(\hat{I})} \frac{E[T_I^{\text{max}}]}{N + \sum_{p_i \in \hat{I}} [n_i (1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i})]} \hspace{1cm} (16)$$

where $E[T_I^{\text{max}}]$’s expression is defined in (9).

---

1Here, we implicitly assume that a broken route cannot be locally recovered.
Expression (16) is essentially what we are looking for: an objective function in terms of the characteristics of cached paths. In the next section, we design an optimal cache policy, which controls the paths in \( I \) to always have the optimal characteristics that maximize the objective function.

V. ANALYSIS OF OPTIMAL CACHE POLICY

To find the optimal cache policy to maximize the objective function in (16), we take a three-step approach. First, given that \( I \) only caches \( K \) paths, we show how to select the best \( K \) paths to be cached in \( I \). Second, we show what is the optimal sequence for exploring the paths in \( I \) to repair routes. Finally, we show how to compute the optimal cache size \( K \).

A. Optimal K Path Selection

**Proposition 1**: If the size of the path cache \( I \) is fixed at a constant \( K \) (i.e. \( |I| = K \)), the policy that caches the shortest \( K \) paths among all discovered paths is better than any other path selection policies.

**Proof**: Define \( D \) as the set of all paths discovered in a flooding-based route discovery process. Only \( K \) number of these paths are selected to be cached in \( I \) and later explored for route repair. We next prove that the policy that selects the shortest \( K \) paths in \( D \) is better than other selection policies.

Suppose \( I \) does not hold the \( K \) shortest paths in \( D \). Then, we can substitute a longer path \( p_i \) (\( 1 \leq i \leq K \)) in \( I \) with a shorter path \( p'_i \) in \( D - I \). This substitution decreases the denominator

\[
N + \sum_{p_i \in I} [n_i(1 + \prod_{j=0}^{i-1} n_j + n_i)]
\]

in (16). This is because by Lemma 2 in Appendix A, \( f(n_i) = n_i(1 + \prod_{j=0}^{i-1} n_j + n_i) \) is an increasing function of \( n_i \). Therefore, after substituting \( p_i \) with \( p'_i \), the term \( n_i(1 + \prod_{j=0}^{i-1} n_j + n_i) \) decreases. For all \( i < l \), \( f(n_i) \) does not change and for all \( i > l \), \( f(n_i) \) also decreases. Therefore, the denominator decreases.

In addition, the substitution increases the numerator in (16). This is because the expected value of \( T_f^{\text{max}} \) is the expected maximum lifetime among \( p_0 \) and the paths in \( I \) according to Lemma 1. Since stochastically a shorter path has a longer lifetime, the substitution of a shorter path into the set \( I \) increases the expectation of \( T_f^{\text{max}} \).

Since the numerator increases and the denominator decreases, the objective function increases after the substitution. Continue this substitution process until all paths in \( D - I \) is no shorter than the paths in \( I \). The resultant \( I \) consists of the \( K \) shortest paths in \( D \). Hence, the policy of choosing the shortest \( K \) paths is better than other policies.

B. Optimal Sequence of Path Exploration

**Proposition 2**: Given a path cache \( I \), the optimal policy for using the cached paths in \( I \) in route repair is to always explore the shortest path first. In other words, the paths in \( I \) should be sorted in an increasing order of their hop counts.

**Proof**: Assume that an optimal path exploration sequence

\[
S = \{p_1, p_2, \ldots \}
\]

maximizes the objective function in (16). Assume that this optimal sequence is not the shortest path first policy. Hence, in \( S \), there must exist two path \( p_i \) and \( p_{i+1} \) that are explored consecutively while their path lengths satisfy \( n_i > n_{i+1} \). We next show that the value of the objective function in (16) increases if we switch the order of \( p_i \) and \( p_{i+1} \) in \( S \) to get a new sequence \( S' = \{p_1, \ldots, p_{i-1}, p_{i+1}, p_i, p_{i+2}, \ldots \} \).

The numerator \( T_f^{\text{max}} \) in (16) is the same for both \( S \) and \( S' \). The denominator of (16) is

\[
g(S) = C + n_l(1 + \sum_{i=0}^{l-1} n_{i} + n_{i+1} + n_l) + n_{l+1}(1 + \sum_{i=0}^{l-1} n_{i} + n_{i+1} + n_l + n_{l+1})
\]

for \( S \) and

\[
g(S') = C + n_l(1 + \sum_{i=0}^{l-1} n_{i} + n_{i+1} + n_l) + n_{l+1}(1 + \sum_{i=0}^{l-1} n_{i} + n_{i+1} + n_l + n_{l+1})
\]

Divide both sides of the above equation with a positive number

\[
H = \frac{1}{(n_0 + n_1 + \cdots + n_l + n_{l+1})(n_0 + n_1 + \cdots + n_l) - (n_0 + n_1 + \cdots + n_{l+1})^2 - (n_0 + n_1 + \cdots + n_l)^2}
\]

we get:

\[
g(S) - g(S') = (n_l(n_0 + n_{l+1}) + n_{l+1} + n_i) + n_{l+1}(n_0 + n_{l+1}) + n_l(n_0 + n_{l+1}) - (n_0 + n_{l+1})(n_{l+1} + n_1)
\]

\[
= (n_l - n_{l+1}) \sum_{j \in I, i \neq j} n_i > 0 \text{ (since } n_l > n_{l+1}).
\]

Hence, \( g(S) > g(S') \). Therefore, switching \( n_l \) with \( n_{l+1} \) in \( S \) increases objective function in (16). This contradicts with the assumption that \( S \) is the optimal sequence for path exploration. Hence, the optimal policy must be the shortest path first policy.

C. Optimal Number of Cached Paths

By Propositions 1 and 2, given a path cache size \( K \), i.e., \( |I| = K \), the optimal cache policy that maximizes the objective function in (16) is to cache the shortest \( K \) paths in an increasing order of their hop counts. We next show how to compute the optimal value of \( K \).
**Proposition 3:** The objective function in (16) with respect to $|I|$ has the following three possible shapes.

1) The objective function is concave and monotonically non-decreasing;
2) The first section of the objective function is concave and non-decreasing. Then, the objective function turns to monotonically non-increasing;
3) The objective function is monotonically non-increasing.

The above three shapes are shown in Fig. 2.

**Proof:** Define $K := |I|$. Let $\hat{h}(K)$ and $\tilde{g}(K)$ be the numerator and denominator of the objective function in (16), respectively. Hence, the objective function becomes $f(K) = \frac{\hat{h}(K)}{\tilde{g}(K)}$. By Lemma 3 in Appendix B, $\hat{h}(K)$ is a positive discrete concave function. Hence, we can construct $\tilde{h}(K)$’s continuously differentiable counterpart function $h(x), x \geq 1$ such that $h(x)|_{x=K} = \hat{h}(K)$ and $h(x)$ is concave. Similarly, by Lemma 4 in Appendix C, $\tilde{g}(K)$ is a positive discrete convex function. Hence, we can construct $g(K)$’s continuously differentiable and convex counterpart function $g(x), x \geq 1$ that satisfies $g(x)|_{x=K} = \tilde{g}(K)$. Then the continuous version of the objective function becomes: $f(x) = \frac{\hat{h}(x)}{\tilde{g}(x)}$, where $f(x)$ is continuously differentiable. By studying the property of $f(x) = \frac{\hat{h}(x)}{\tilde{g}(x)}$, we can get the property of $\tilde{f}(K)$.

Note that

$$f'(x) = \frac{h'(x)g(x) - h(x)g'(x)}{[g(x)]^2}$$

and

$$f''(x) = \frac{-2g'(x)}{g(x)} f'(x) + \frac{h''(x)g(x) - h(x)g''(x)}{[g(x)]^2}.$$ 

Since $h''(x) < 0, g''(x) > 0, h'(x) > 0, g'(x) > 0, h(x) > 0$ and $g(x) > 0$, we get that when $f'(x) > 0$, we always have $f''(x) < 0$. This means that at the increasing section of $f(x)$, $f(x)$ has a concave shape. Since $f(x)$ is a continuously differentiable function, if at any point it turns from decreasing to increasing, its concave increasing part can only be followed by a concave decreasing part. In addition, whenever $f(x)$ starts decreasing, $f(x)$ can never turn to increasing again. Based on the above observation, there is only three possible shapes of $f(x)$ depicted in Fig. 2. In the first possible shape (Fig. 2 (a)), $f(x)$ is an non-decreasing concave function. In the second possible shape (Fig. 2 (b)), $f(x)$ is formed by two sections. In the first section, $f(x)$ is a concave function that increases with $x$. In the second section, $f(x)$ becomes a decreasing function. In the final possible shape (Fig. 2 (c)), $f(x)$ is a non-increasing function.

The correctness of Proposition 3 is validated by our numerical results in Appendix D. Proposition 3 indicates that the optimal path repair policy does not necessarily cache all paths discovered in a flooding-based route discovery. There exists a unique optimal number of paths to be cached in $I$, which is denoted as $K^*$ and is smaller or equal to the number of paths discovered in a flooding-based route discovery. If the shape of the objective function in (16) is similar to Fig. 2(a), then the more path cached, the better. Hence, $K^*$ equals the number of paths discovered in a flooding-based route discovery. If the shape of the objective function is similar to Fig. 2(b), $K^*$ can be computed by finding the turning point where the objective function turns from increasing to decreasing. If the shape of the objective function is similar to Fig. 2(c), $K^*$ equals 0, which means that no path should be cached.

**D. Optimal Cache Policy**

Based on our results in Propositions 1, 2 and 3, we can finally design the optimal cache policy as Algorithm 1 that picks the best set of paths and caches them in the optimal sequence in $I$. The input $D$ to Algorithm 1 is the set of paths discovered in a flooding-based route discovery.

**Algorithm 1** The Optimal Cache Policy

**Require:** Discovered path set $D \neq \emptyset$; $p_0 \leftarrow$ pop the shortest path in $D$; $I \leftarrow \emptyset$;

Calculate the objective function value $val$ in (16); $maxVal \leftarrow val$; $K^* \leftarrow 0$.

**while** $D \neq \emptyset$ **do**

$p \leftarrow$ pop the shortest path in $D$; Append $p$ at the tail of $I$;

Calculate the objective function value $val$ in (16);

**if** $val \leq maxVal$ **then**

Remove $p$ from the tail of $I$;

**return** $I$ and $K^*$;

**else**

$maxVal \leftarrow val$; $K^* \leftarrow K^* + 1$;

**end if**

**end while**

**return** $I$ and $K^*$;

Using Algorithm 1, an optimal cache-based repair scheme works as follows:

1) Determine the node-disjoint path set $D$ discovered by the latest flooding;

2) Following Algorithm 1, cache the optimal set of paths in path cache $I$;

3) Whenever there is a route break, explore the cached paths in $I$ following the sequence that they are stored in $I$. Paths that are found broken are removed from $I$.

Flood the whole network when $I$ becomes empty.

The running time of Algorithm 1 is $O(|D|)$, where $|D|$ is the number of discovered node-disjoint paths per flooding and
usually is a small number. Hence, the complexity of Algorithm 1 is fairly small.

VI. EXTENSION AND DISCUSSION

While the optimal cache policy in Algorithm 1 is developed with respect to the objective in (1), it is also applicable to other objectives as long as the objective can also make Propositions 1, 2, 3 hold. For example, consider the case of a real-time application that periodically generates delay-sensitive packets. If a real-time packet is generated when the current path to the destination is still valid, the packet can arrive at its destination with a fairly small delay. If the packet is generated when the current path to the destination is broken, the packet will experience a much longer delay and have a very high chance to miss its deadline while waiting for the route to be repaired. Hence, denoting the entire lifetime of the real-time application as $T$, the problem of maximizing the percentage of real-time packets that can meet their deadline can be approximated as

$$\max \frac{\text{Data transmission time}}{T}. \quad (23)$$

Dividing $T$ into repair cycles and following a method similar to the method in Sections III and IV, the objective function in (23) can be transformed to

$$\max \frac{T_d}{T_d + T_o}, \quad (24)$$

where $T_d$ is the expected data packet transmission time within a repair cycle, and $T_o$ is the expected overhead packet transmission time within a repair cycle. The objective function in (24) is equivalent to

$$\max \frac{T_d}{T_d + T_e} = \max \frac{T_d}{T_f + T_e}, \quad (25)$$

where $T_f$ is the expected flooding time within a repair cycle, and $T_e$ is the expected exploration time within a repair cycle. Suppose that $T_f$ and $T_e$ can be computed as follows: $T_f(N)$ is an increasing function of network node count $N$, and $T_e = \sum_{p_i \in T} n_i t_e(1 + a_i)$, where $n_i$ is the hop count of path $p_i$, $t_e$ the expected exploration time for one link, and $a_i$ is the probability that path $p_i$ is still alive when it is explored. It can be shown that for the objective in (25), Propositions 1, 2 and 3 hold. Therefore, the optimal cache policy for the objective in (25) can be computed using Algorithm 1.

Another possible groups of objective functions can be formed by making the formulation of the objective functions in (1) and (23) from shape $\max \frac{A}{B}$ to $\max(A - B)$. It is very easy to go through similar analysis in the previous sections to show that Propositions 1, 2 and 3 also hold for these objective functions.

VII. SIMULATION RESULTS

We conduct simulations to validate our work. We first implement our optimal route repair scheme in the original DSR protocol, and then run simulations in NS 2.32 [12] to prove the advantages of the proposed optimal cache policy. In our simulations, we compare the performance of the optimal cache policy with that of some heuristic policies, including the no-cache policy, 1-path policy, and 2-path policy. For the no-cache policy, no path is cached for path repair. For the 1-path policy, at most 1 path is cached for path repair. For the 2-path policy, at most 2 paths are cached for path repair.

In the simulation, 200 mobile nodes are uniformly distributed in a $1000 \times 1000$ m$^2$ rectangular region. The transmission range is 250 m, the interference range is 550 m, and the channel bandwidth is 1 Mbit/s. We measure the performance of a single CBR flow between a randomly selected pair of source node and destination node. The packet size of the CBR flow is 512 bytes, and the packet interval is 0.05 s. For the mobility of nodes, we choose the SMS mobility model [10] due to its capability to capture realistic characteristics of node movements. The mean pause time of each node is 0 s. The minimum number of phase steps is 6. The maximum number of phase steps is 30. The unit time slot is 1 s. The memory factor parameter is 0.5. The average speed of each node ranges from 8 m/s to 16 m/s. For each average speed, we do 20 runs, and each run lasts 1000s.

We compute the objective function in (1) in the simulations and show the result in Fig. 3 (I). As shown in the figure, the value of the objective function under the optimal policy is larger than that of other heuristic schemes. The number of flooding-based discovery in the simulations is shown in Fig. 3 (II). It can be seen that the optimal policy drastically reduces the frequency of flooding comparing to other heuristic schemes. We also show the average throughput and packet loss rate in Fig. 4 (I) and (II), respectively. It can be seen that the optimal policy outperforms other heuristic schemes in terms of both throughput and packet loss rate.

VIII. CONCLUSIONS AND FUTURE WORK

While there have already existed some routing protocols aiming at optimizing cache-based route repair in MANETs, their methods are mainly heuristics and do not provide provably optimal routing policies. This paper studies the optimal cache-based path repair problem in a rigorous mathematical approach. We focus on the problem of how to use cached node-disjoint paths to reduce the flooding frequency while maintaining reasonable control packet overhead. An optimal
path cache policy is proposed and its correctness is proved. Its performance advantages are validated by simulations. We have also shown that our mathematical approach can be applied to a wide range of optimization goals for cache-based route repair. In the future, we plan to apply this approach to various objectives for different application scenarios.

APPENDIX A
PROOF OF LEMMA 2

Lemma 2: The function
\[ f(n_i) = n_i(1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i}) \]  
(A.1)
is monotonically increasing with respect to \( n_i \).

Proof: We relax the original function to a continuous function of \( n_i \), and then calculate its derivative.

\[
f'(n_i) = \left( 1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i} \right) - n_i \sum_{k=1}^{i-1} \frac{n_k}{n_k + n_i} \prod_{j=0, j \neq k}^{i-1} \frac{n_j}{n_j + n_i}
\]
\[
= 1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i} - \sum_{k=1}^{i-1} \frac{n_k}{n_k + n_i} \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i}.
\]  
(A.2)
To complete the proof, we need to show that

\[
1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i} - \sum_{k=1}^{i-1} \frac{n_k}{n_k + n_i} \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i} > 0,
\]

\[\iff 1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i} > \sum_{k=1}^{i-1} \frac{n_k}{n_k + n_i} \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i},\]

\[\iff \prod_{j=0}^{i-1} \left( 1 + \frac{n_j}{n_j + n_i} \right) + 1 > \sum_{k=0}^{i-1} \frac{n_i}{n_k + n_i}.
\]
The above inequality is true since

\[
\prod_{j=0}^{i-1} \left( 1 + \frac{n_j}{n_j + n_i} \right) + 1 > 1 + \sum_{j=0}^{i-1} \frac{n_i}{n_j} + 1 > \sum_{j=0}^{i-1} \frac{n_i}{n_j} > \sum_{k=0}^{i-1} \frac{n_k}{n_k + n_i}.
\]
Therefore, \( f'(n_i) > 0 \). That is \( f(n_i) \) is a monotonically increasing function of \( n_i \).

APPENDIX B
PROOF OF LEMMA 3

Lemma 3: Using the optimal policies in Propositions 1 and 2, the denominator of the objective function in (16) is an increasing, discrete, concave function with respect to path cache size \( k \).

Proof: Denote the \( k \) paths in \( I \) as \( \{p_1, p_2, \cdots, p_k\} \) in an increasing order of their hop counts. According to Lemma 1, the numerator of the objective function in (16) is

\[ E[T_{I, k}^{\text{max}}] = E\left[ \max_{0 \leq i \leq k} \{ T_i \} \right]. \]  
(B.1)
Now increase the number of cached paths to \( k + 1 \) by adding an additional path \( p_{k+1} \) in \( I \). Note that the hop counts of paths \( p_k \) and \( p_{k+1} \) satisfy \( n_{k+1} \geq n_k \). \( E[T_{I, k}^{\text{max}}] \) becomes

\[
E[T_{I, k+1}^{\text{max}}] = E\left[ \max_{0 \leq i \leq k} \{ T_i \} \right] + a_{k+1}E\left[ \left( T_{k+1} - \max_{0 \leq i \leq k} \{ T_i \} \right)^+ \right] + E\left[ \max_{0 \leq i \leq k} \{ T_i \} \right],
\]
(B.2)
where \( a_{k+1} = P\{ T_{k+1} > \max_{0 \leq i \leq k} \{ T_i \} \} \) and \((f(x))^+\) denotes

\[
(f(x))^+ = \begin{cases} f(x), & f(x) \geq 0, \\ 0, & f(x) < 0. \end{cases}
\]  
(B.3)
Define \( U(k+1) := E[T_{I, k+1}^{\text{max}}] - E[T_{I, k}^{\text{max}}] \). We have

\[ U(k+1) = a_{k+1}E\left[ (T_{k+1} - \max_{0 \leq i \leq k} \{ T_i \})^+ \right]. \]  
(B.4)
Clearly, \( U(k+1) > 0 \). Hence, \( E[T_{I, k}^{\text{max}}] \) is an increasing function of \( k \). In addition, \( E[T_{I, k}^{\text{max}}] \) is a concave function if we can show that \( U(k+1) \) is a decreasing function.

Based on (15), \( a_{k+1} = \prod_{j=0}^{k+1} \frac{n_j}{n_j + n_{k+1}} \). Since \( n_k \leq n_{k+1} \), \( a_{k+1} \) decreases as \( k \) increases. In addition, note that

\[
E\left[ (T_{k+1} - \max_{0 \leq i \leq k} \{ T_i \})^+ \right] = E\left[ (T_{k+1} - \max_{0 \leq i \leq k-1} \{ T_i \})^+ \right] < E\left[ (T_k - \max_{0 \leq i \leq k-1} \{ T_i \})^+ \right].
\]  
(B.5)
Therefore, \( E\left[ (T_{k+1} - \max_{0 \leq i \leq k} \{ T_i \})^+ \right] \) also decreases as \( k \) increases. Hence, based on (B.4), we can conclude that \( U(k+1) \) decreases as \( k \) increases. Hence, the numerator of the objective function in (16) is an increasing, discrete, concave function with respect to \( k \).

APPENDIX C
PROOF OF LEMMA 4

Lemma 4: Using the optimal policies in Propositions 1 and 2, the denominator of the objective function in (16) can be approximated as a non-decreasing, discrete, convex function with respect to the path cache size \( k \).

Proof: Consider increasing the path cache size from \( k \) to \( k + 1 \) by adding a path \( p_{k+1} \). Note that the path lengths of paths \( p_k \) and \( p_{k+1} \) satisfy \( n_{k+1} \geq n_k \). This addition of \( p_{k+1} \) increases the denominator of the objective function in (16) by

\[ V(k+1) = n_{k+1} \left( 1 + \prod_{j=1}^{k} \frac{n_j}{n_j + n_{k+1}} \right) \approx n_{k+1}. \]  
(C.1)
The approximation holds for large \( k \) since \( \prod_{j=1}^{k} \frac{n_j}{n_j + n_{k+1}} \ll 1 \) for large \( k \). In addition, since \( n_{k+1} \geq n_k \), \( V(k+1) \) in (C.1) can be approximated as an non-decreasing function with respect to \( k \). Hence, the denominator or the objective function can be approximated as a non-decreasing, discrete, convex function.
Appendix D

Validation of Proposition 3

The proof of Proposition 3 is based on Lemma 4 in the Appendix, which uses some approximation to derive its result. In this section, we validate that this approximation does not affect the correctness of Proposition 3 by experiments.

In the experiments, the total number of nodes is 200, and the path lifetime parameter $\lambda$ is calculated by (8). We set $\bar{v} = 10$ m/s and $R = 250$ m. Note that the specific values of the average speed $\bar{v}$ and communication range $R$ do not affect the general trend of the objective function value. The curves are calculated by (16). To validate the correctness of the objective function in (16), we also do numerical calculation of the objective. In the calculation, the path lifetime follows the exponential distribution with the same parameters as those used in the theoretical formulas. However, the numerical calculation of the objective function does not have the approximations in Lemma 4. The numerical calculated values are averaged over $10^4$ trials.

We perform two experiments: the random hop count experiment, and the fixed hop count experiment. For the random hop count experiment, the path hop count is uniformly distributed within $[1, 20]$. The value of the objective function in (16) and its numerator and denominator with respect to the path cache size $|\mathcal{I}|$ are shown in Fig. 5. In the figures, the theoretical curves and the numerical curves matches perfectly, validating the correctness of our theoretical formulas. Also, as expected, the numerator is an increasing concave function and the denominator is an increasing convex function. The objective function value with respect to $|\mathcal{I}|$ is first increasing in a concave shape and then decreasing. Its shape is the same as Fig. 2(b), validating the correctness of Proposition 3. For the fixed hop count experiment, the hop counts of all paths discovered by flooding-based route discovery are fixed at 2, 3, and 4 respectively. The experiment results are shown in Fig. 6. As shown in the figure, the theoretical curves matches perfectly with the numerical curves, validating the correctness of our theoretical formula for the objective function. Also, the objective function is concave and monotonically increasing with respect to $|\mathcal{I}|$. The shape is the same as Fig. 2(a), validating the correctness of Proposition 3.

Acknowledgment

This work was supported in part by the US National Science Foundation under grant CNS-0831865 and the Institute for Critical Technology and Applied Science (ICTAS) of Virginia Tech.

References