MP-DSR: A QoS-aware Multi-path Dynamic Source Routing Protocol for Wireless Ad-Hoc Networks

Roy Leung, Jilei Liu, Edmond Poon, Ah-Lot Charles Chan, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto
{*leungr, chanah*}*@comm.toronto.edu,* {*jliu, epoon, bli*}*@eecg.toronto.edu*

Abstract

Routing in wireless ad-hoc networks has received significant attention from recent literature due to the fact that the dynamic behavior of these networks poses many technical challenges on the design of an effective routing scheme. Though on-demand routing approaches have been shown to perform well, they generally lack the support for Quality-of-Service (QoS) with respect to data transmission. In order to select a subset of end-to-end paths to provide increased stability and reliability of routes, a new QoS metric, end-to-end reliability, is defined and emphasized in this paper. We present a distributed multi-path dynamic source routing protocol (MP-DSR) for wireless ad-hoc networks to improve QoS support with respect to end-to-end reliability. Our protocol forwards outgoing packets along multiple paths that are subject to a particular end-to-end reliability *requirement. A simulation study is performed to demonstrate the effectiveness of our proposed protocol, particularly the fact that MP-DSR achieves a higher rate of successful packet delivery than existing best-effort ad-hoc routing protocols, such as the Dynamic Source Routing (DSR).*

1 Introduction

Wireless ad-hoc networks are autonomous systems composed of mobile hosts that are free to move around arbitrarily. These mobile hosts are referred to as nodes. Rather than relying on a network infrastructure to perform routing in an ad-hoc network, each mobile host serves as a router to forward packets originated from other hosts. Such characteristics allow an ad-hoc network to be established on-the-fly with built-in fault tolerance and unconstrained connectivity. For such networks, an effective routing protocol is critical for adapting to node mobility as well as possible channel error to provide a feasible path for data transmission. In addition to basic routing, for mission-critical applications, Quality-of-Service (QoS) routing protocols are needed to search for a path that can satisfy certain QoS requirements and constraints, such as bandwidth or data reliability. The focus of this paper is to propose a QoS routing protocol in wireless ad-hoc networks.

In this paper, we design a QoS-aware multi-path source routing protocol (MP-DSR) focusing on a new QoS metric, *end-to-end reliability*. End-to-end reliability is used to reflect the probability of sending data successfully from the source node to the destination node within a time window. Note that it is not the focus of MP-DSR to provide strict *end-to-end reliability* guarantees. Rather, MP-DSR provides routes that satisfy a specific *end-to-end reliability* requirement and such routes persist with a high probability. Thus, it is possible to have a transient QoS disruption even with such a guarantee.

Our major contributions in this paper are the following. First, we define our QoS parameter of interest, *end-to-end reliability*. Second, we propose a fully distributed QoS routing protocol, MP-DSR, with respect to this QoS parameter. MP-DSR is based on the existing Dynamic Source Routing protocol (DSR) [1, 4] and takes advantage of its distributed on-demand nature. It seeks to compute a set of unicast routes that can satisfy a minimum *end-to-end reliability* requirement; it then maintains this requirement throughout the life time of transmission. Packets transmitted from the source node will arrive at the destination node with a higher successful probability than existing best-effort ad-hoc routing protocols. Finally, we evaluate the performance of our routing protocol using extensive network simulations, and compare this to the performance of DSR.

The rest of this paper is organized as follows. The system model is presented in Section 2. Our MP-DSR protocol is shown and analyzed in Section 3. Additional

optimizations are discussed in Section 4. Extensive simulation results of MP-DSR against the original DSR protocol are presented in Section 5, including detailed analysis and evaluations. Section 6 compares our proposed MP-DSR with related work. Section 7 concludes the paper.

2 System Model

In this paper, we propose an end-to-end reliability model that derives end-to-end reliability between the source and destination nodes by evaluating the path reliabilities of all existing feasible paths. The path reliability is calculated based on the link availabilities of all the links along a path. According to Jiang et al. [3] and McDonald et al. [6, 7], link availability is defined as the probability that a link is available until time $t_0 + t$, given that it is an active link at time t_0. The calculation of link availability is based on the current node's movement [3]. A path is defined as a sequence of links from the source node to the destination node. Path reliability is the product of link availabilities of all the links along the path under the assumption that all links are independent. If we denote the *link availability* of the link connecting node m and node n in a time period of t as $A_{m,n}(t)$, the *path reliability* of kth path between the source node S and the destination node D as $\Pi^k_{S,D}(t)$ in a time period of t, we have:

$$\Pi^k_{S,D}(t) = \prod_{(m,n)\in k} A_{m,n}(t) \qquad (1)$$

Although we adopt the link availability model and path reliability model from previous work [3, 6, 7], our end-to-end reliability model is independent from the implementation of link availability and path reliability models. Our model is ready to accommodate any link availability and path reliability models without any modification on any part but the link availability and path reliability themselves.

We make three assumptions in this paper: (a) Mobile nodes in ad-hoc networks cannot move too fast to render QoS routing impossible. (b) An individual mobile node has an uniform transmission range by its omnidirectional antenna. Thus, all links are bi-directional in this network model. (c) There exists a neighbor discovering protocol; such neighbor discovery can be easily achieved by having each node periodically transmit a BEACON packet identifying itself [9], so that each node can learn about its neighbors.

2.1 End-to-End Reliability Model

We define *end-to-end reliability* based on the link availability and path reliability model. Given a specific application's requirement of *end-to-end reliability*, MP-DSR seeks to discover multiple disjoint paths for data transmission to satisfy such a requirement. We formally define the term *disjoint path* in an ad-hoc network as:

Definition 1: If two acyclic paths , P_1 and P_2, share common source and destination nodes, but do not share common intermediate nodes, they are *Disjoint Paths*.

In multiple path routing, data transmission fails if and only if all disjoint paths fail at the same time. Thus, the probability that transmission fails is less than the probability that any path fails individually.

Definition 2: *End-to-end reliability*, $P(t)$, is the probability of having a successful data transmission between two mobile nodes within the time period from t_0 to $t_0 + t$, where t_0 is any time instant.

Proposition: *The end-to-end reliability from the source node S to the destination node D, $P(t)$, to route data along multiple disjoint paths either in parallel or as alternatives, given that K is a set of disjoint paths, is derived as:*

$$P(t) = 1 - \prod_{k\in K} (1 - \Pi^k_{S,D}(t)) \qquad (2)$$

3 Multi-Path DSR (MP-DSR)

3.1 Overview

We now present our proposed multi-path dynamic source routing algorithm, referred to as MP-DSR, which provides soft QoS guarantees with respect to end-to-end reliability by discovering a set of multiple disjoint paths and transmitting data along these paths. When an application uses MP-DSR for a route discovery, it supplies an end-to-end reliability requirement, P_u, where $0 \leq P_u \leq 1$. Given this requirement, MP-DSR determines two parameters for the route discovery: (1) the number of paths it needs to discover; and (2) the lowest path reliability requirement that each search path must be able to provide in order to satisfy P_u. We refer to these two parameters as m_0 and Π_{lower}, respectively. These two parameters are decided based on the available state information.

The relationship between m_0 and Π_{lower} is straightforward: when there are fewer paths between the source and destination nodes, more reliable paths are preferable and therefore, a higher Π_{lower}, and vice versa. Once the source node makes this decision, it sends m_0 Route Request (RREQ) messages to search for feasible paths.

Each message contains information such as Π_{lower}, the path it has traversed (T), the corresponding path reliability (Π_{acc}), etc. When an intermediate node receives the RREQ message, it checks whether this message meets the path reliability requirement (i.e. $\Pi_{acc} > \Pi_{lower}$). If this RREQ message fails to meet such a requirement, the node will discard the message. Otherwise, the intermediate node updates the RREQ message to include itself in T as well as in Π_{acc}, and then forwards multiple copies of this message to its neighbors. The number of copies is based on the number of neighbors that can receive this RREQ message without failing the path reliability requirement. This number of copies is also bounded by m_0 to restrict the degree of message forwarding inside the network. When the destination collects the RREQ messages, it selectively chooses multiple disjoint paths from these messages, and sends Route Rely (RREP) messages back to the source node via these selected paths. Upon the arrival of these RREP messages at the source node, the source node begins to send data along these paths.

```
Procedure RouteDiscovery (P_u)
begin
    for (t_w = t_wMax; t_w > t_wMin; t_w * 0.9)
        set A ∈ {A_{s,1}(t_w), ..., A_{s,l}(t_w)};
        m_0 = 0;
        while (m_0 ≤ l and m_0 ≤ m_max)
            path = 0;
            m_0 = m_0 + 1;
            Π_lower = 1 − ∛(1 − P_u);
            for (each neighbor node, j, in set A) {
                if (A_{s,j}(t_w) > Π_lower)
                    path = path + 1;
                }
            }
            if (m_0 ≤ path)
                return m_0, Π_lower, t_w
            }
        }
    }
    return error;
end
```

Figure 1. Initializing route discovery

3.2 Route Discovery

To illustrate the MP-DSR algorithm, we assume that at time t_0, an application requests a connection between source node S and destination node D, with end-to-end reliability denoted as P_u. Before sending the route request (RREQ) message, MP-DSR needs to determine the following parameters using the route discovery algorithm in Figure 1 (details explained later): (a) m_0; (b) Π_{lower}; and (3) the time window that this end-to-end reliability guarantee holds, denoted as t_w.

In order to guarantee that the end-to-end reliability is greater than P_u, the objective of the route discovery algorithm is to determine reasonable values for these three parameters with the given P_u and local link availability information. To accomplish this, the algorithm needs to determine the value of Π_{lower} such that in the worst case, when all paths have their path reliability equal to Π_{lower}, the resulting end-to-end reliability, denoted as $P(t)$, can still satisfy the end-to-end reliability requirement (i.e. $P(t) = P_u$). This implies that given a fixed value of m_0 and P_u, Π_{lower} can be determined by substituting P_u into $P(t)$ in Equation (2) as follows:

$$\Pi_{lower} = 1 - \sqrt[m_0]{1 - P_u}$$

In addition, the route discovery algorithm needs to determine the value of m_0, such that there are at least m_0 neighbors whose link availabilities to the source node are greater than the Π_{lower} computed from Equation (3.2). Since it is preferable to keep the data and RREQ traffic at a minimum while still meeting the P_u requirement, the route discovery algorithm attempts to find the minimum value of m_0. Thus, the algorithm begins by setting the value of m_0 to 1. If there is no neighbor that can satisfy Π_{lower}, then the algorithm increments m_0 by 1, and then examines again whether the condition can be met. The value of m_0 is upper-bounded either by the number of neighbors of node S denoted as l_S or by a system parameter referred to as m_{max}, depending on which parameter has a lower value. The purpose of m_{max} is to restrict the maximum amount of network traffic from RREQ messages.

In some scenarios, the route discovery algorithm may not be able to find a suitable value for m_0 because the neighbors' link availabilities are too low at a specific time window (t_w). To resolve this issue, the route discovery algorithm reduces t_w to raise the values of link availability between node S and its neighbors. Nevertheless, such reduction of the time window also increases the overhead of route maintenance messages, since source node is required to send route check (RCHK) message every t_w to validate if the end-to-end reliability can still satisfies P_u (details shown later in Section 3.3). Due to this overhead, it is preferable to have a larger t_w. However, the tradeoff of using a large t_w is that validation takes place with a longer time interval, and the reliability guarantee is less stringent as a result. Thus, there are two system parameters in place corresponding to the minimum and maximum values for t_w, in order to restrict the overhead of RCHK message

Table 1. System parameters

P_u	end-to-end reliability requirement
m_0	the number of paths that the source node aims to discover
Π_{lower}	path reliability requirement for each RREQ message
T	path that a RREQ message has traversed
Π_{acc}	accumulated path reliability of path that a RREQ message has traversed
t_w	time window that this end-to-end reliability guarantee holds. It dictates how often the route maintenance takes place during the lifetime of data transmission
$P(t)$	end-to-end reliability of multiple paths
m_{max}	Upper bound of m_0
l_i	number of neighbors of node i
t_{wMin}	lower bound for time window
t_{wMax}	upper bound for time window

while maintaining the guarantees at a reasonable level. The minimum t_w is denoted as t_{wMin}, while the maximum t_w is denoted as t_{wMax}. t_{wMin} and t_{wMax} are application-dependent parameters. For example, HTTP requests may have a smaller windows size while the FTP application may need a larger window size. The strategy of our route discovery algorithm is to reduce the amount of RCHK messages, thus it begins by setting t_w to t_{wMax}. If this t_w raises the link availabilities to such values that no neighbor can satisfy Π_{lower}, then the algorithm lowers t_w with a multiplier of 0.9. This continues until m_0 and its corresponding Π_{lower} are satisfied. If t_w is reduced to t_{wMin} and no appropriate m_0 value is found, the route discovery algorithm may then notify the application with an error. Figure 1 formally presents the algorithm explained above. For the sake of clarity, Table 1 summarizes all the notations previously introduced.

Once the route discovery algorithm has determined Π_{lower} and m_0, the source node sends an RREQ message to the m_0 neighbors that have the highest link availability. Among the neighbors in the local broadcast range, only the intended receivers keep the RREQ message, while other neighbors discard the message.

Table 2 illustrates the fields inside a RREQ message. In addition to the system parameters (i.e. m_0, Π_{lower}, t_w, and P_u), a RREQ message stores the path that the RREQ message has traversed into the *path* field, as well as the corresponding accumulated path reliability into the $\Pi_{acc}(t_w)$ field. Each RREQ message is associated with an unique request ID; together with the source node ID, a network node can uniquely identify a RREQ message. We refer to this pair of IDs as an *identifier pair*. When the source node sends a RREQ message, its $\Pi_{acc}(t_w)$ field is set to 1 while its path field is set to be empty.

Table 2. RREQ in MP-DSR

1. Source ID
2. Request ID
3. Destination ID
4. RREQ
5. m_0
6. Π_{low}
7. t_w
8. P_u
9. *path:* T
10. $\Pi_{acc}(t_w)$

When an intermediate node receives an RREQ message, the node begins by inspecting the message's identifier pair to check whether the message is a duplicate. In MP-DSR, an intermediate node is allowed to forward at most m_0 RREQ messages with the same identifier pair. After forwarding m_0 RREQ messages, any subsequent RREQ message of this identifier pair is discarded by that intermediate node. To keep track of the number of times a particular RREQ message has been forwarded, each node maintains a table of counters for each of the RREQ message the node has forwarded. This counter is incremented by the number of neighbors to which the node has sent in each forwarding procedure.

If the RREQ message satisfies the above criterion, the node needs to update the $\Pi_{acc}(t_w)$ field of the message, by multiplying this field with the link availability of the link the message has just traversed. Assume that the node ID of the intermediate node is i and the ID of the previous node is j, then the update procedure is as follows:

$$\Pi_{acc}(t_w) = \Pi_{acc}(t_w) * A_{j,i}(t_w)$$

Once the update is complete, the intermediate node makes its forwarding decision by choosing a subset of its neighbors to receive the RREQ message. First, it determines the minimum link availability ($L_i(t_w)$) required for the outgoing links to obey, defined as:

$$L_i(t_w) = \frac{P_u}{\Pi_{acc}(t_w)}$$

$L_i(t_w)$ is used to ensure that neighbors with their link availability above $L_i(t_w)$ can satisfy P_u, when they receive the RREQ message. Thus, the intermediate node excludes any neighbors that have its link availability below $L_i(t_w)$ to receive this RREQ message. Furthermore, among the candidates of these RREQ message receivers, the intermediate nodes excludes neighbors that have already appeared in the *path* field of the message; this

ensures that the RREQ message does not traverse in a loop inside the network. Once the validation is complete, the intermediate node chooses at most m_0 neighbors that have the highest link availability to be the intended receivers of the RREQ message. Before sending the RREQ message, the intermediate node appends its node ID to the path field of the message. To forward the RREQ message to the selected neighbors, the intermediate node deploys the same local broadcast strategy as that of the source node, in which only the intended receivers keep the RREQ message.

3.2.1 Path Selection Algorithm

After the route discovery begins, the destination node collects multiple route request (RREQ) messages. From the *path* fields stored inside these messages, the destination node uses a *path selection algorithm* to pick the set of disjoint paths that can provide end-to-end reliability greater than P_u. To limit the time that the destination node waits for the RREQ messages, the destination node sets a timer when it receives the first RREQ message. When this timer time-outs, the destination node executes the path selection algorithm based on the RREQ messages that are received during the interval between the arrival of the first RREQ message and the time-out. Upon the expiration of the timer, any subsequent RREQ message are dropped by the destination node. Once the destination node completes executing the path selection algorithm, it replies to the source node about the result of its selection by a set of *Route Reply* (RREP) messages. Each of the RREP messages stores a single disjoint path; it follows such a path to traverse back to the source node.

The path selection algorithm is composed of two major steps. The first step is path-sorting algorithm, which is simply a sorting procedure that sorts all feasible paths (gathered from the RREQ messages) in a descending order according to their accumulated path reliabilities. We refer to the set of sorted paths as the *candidate set*. The second step is a *disjoint path selection algorithm* that selects a group of disjoint paths from the *candidate set*, such that this group of disjoint paths may collectively satisfy P_u. We refer to this group of disjoint paths as the *trace set*. Figure 2 formally shows our disjoint path selection algorithm. It takes the outputs of the path-sorting algorithm as its inputs, which include the *candidate set* and a set of path reliabilities corresponding to the *candidate set* (denoted as the *reliability set*). The main responsibility of this algorithm is to selectively copy disjoint paths from the *candidate set* to the trace set, according to the reliability information in the *reliability set*.

The disjoint selection algorithm is generally a recursive algorithm. In each recursive step, the algorithm attempts to add a new path to the *trace set*, and the new

```
/* Returns a disjoint set that satisfies P_u */

set CandidateSet ∈ {P_0, P_1, ..., P_{n-1}}
set PathReliabilitySet ∈ {Π^0_{S,D}(t), Π^0_{S,D}(t), ..., Π^0_{S,D}(t)}
set TraceSet ∈ {}

int DisjointPathSelection(si, failProbability) {
    if (1 - failProbability ⩾ P_u) {
        return success;
    }
    for (each feasible path P_i in subset
                {P_si, ... P_{n-1}}) {
        if (P_i does not contains node in any paths of
                TraceSet) {
            TraceSet = TraceSet ⋃ P_i
            if (( DisjointPathSelection (i + 1,
                    failProbability * (1 − Π^i_{S,D}(t)) = 1)
                    and (|TraceSet| < m_max)) {
                return success;
            }
        }
        Remove P_i from TraceSet
    }
    return failure; /* Unable to find a set so far */
}
```

Figure 2. Disjoint path selection algorithm

path must be disjoint to all of the existing paths in the *trace set*. After the addition of the new path, the algorithm continues by calling itself to begin the next recursive step. As each recursive step begins, it obtains two pieces of information from the previous step:

- Which path in the *candidate set* that the current step should start to consider? This is denoted as the *start index* (si). This *start index* normally points to the path where all the preceding paths before the *start index* have already been considered by previous recursive steps.

- The end-to-end failure probability of the paths in the current trace set, denoted as *failProbability* in Figure 2. The end-to-end failure probability may be straightforwardly computed by multiplying path failure probabilities, each of which equals to one minus the corresponding path availability.

If the current recursive step discovers that none of the paths in the *candidate set* starting from the *start index* is disjoint with those in the *trace set*, it then removes the paths that was previously added, and returns the control to the previous step (backtracking). In this case, the previous step continues by examining the paths after the one that has just been removed, and makes a recursive

call when it finds another path that is disjoint to the *trace set*. The algorithm stops when an intermediate step discovers that the paths in the *trace set* have satisfied the P_u requirement, which is equivalent to finding the end-to-end failure probability less than the required reliability level (i.e. $1 - P_u$). Otherwise, the algorithm returns a failure.

Once the destination nodes have completed executing the path selection algorithm, it sends a set of RREP message through each disjoint path. As indicated in Table 3, each RREP message stores its corresponding identifier pair, destination ID, as well as a disjoint path selected in the path selection algorithm. Each RREP message traverses back to the source node using the *path* field of the message.

Table 3. RREP in MP-DSR

1. Source ID
2. Request ID
3. Destination ID
4. RREP
5. Path: $S, ..., D$

When the first RREP message arrives at the source node, the source node adds the *path* field into its route cache and immediately sends data using that path. When subsequent RREP messages arrives at the source node, the source node also adds their *path* fields into the route cache and may use these paths to send data packets. In fact, our MP-DSR protocol is fully detached from data transmission schemes; the application has the freedom to choose among various schemes to transmit data along the discovered feasible paths. For example, one possible transmission scheme is to deliver data to multiple paths redundantly; another possibility is to use one of the paths to send data, and in the case of path failure, the source node can immediately switch to an alternative path. Refer to [2] for details on other alternatives. Notice that packets may arrive out-of-order at the destination node, packet re-ordering is assumed to be carried out at the transport layer.

3.2.2 Route Discovery Failure Handling

There are two possible scenarios in which the source node may fail to discover a path that meets the end-to-end reliability requirement, P_u.

1. The RREQ message fails to reach the destination node because all the intermediate nodes fail to meet path reliability requirement, that is, $\Pi_{acc(t_w)}$ is below Π_{lower}. In this case, all RREQ messages are dropped and a timeout event takes place at the source node.
2. The path selection algorithm in Figure 2 is unable to determine a set of feasible paths that can satisfy P_u. In this case, the destination node does not send a reply to the source node.

In either case, the source node needs to initiate another route discovery with a different system parameter setting. There are two options: one is to decrease the time window size, t_w, which results in fewer number of paths discovered in the next route discovery but there is a higher probability that these discovered paths have higher path reliability. The other is to increase the number of initial path discovery, m_0, which in turn lowers Π_{lower}. Thus, the destination node is likely to obtain more feasible paths for path selection algorithm.

Our approach is a combination of both techniques. We reduce the value of t_w to obtain better link availability at the source node and initiate the value of m_0 to be the same as or greater than its previous value. Nevertheless, it is often the case that the value of m_0 is constrained by l_S and m_{max}. If the discovery still fails after several attempts, we set m_0 to l_S and set t_w to t_{wMin} to compute a new value for Π_{lower}. For simplicity, we limit the maximum number of retries to 2 in our simulation. If the route discovery fails in the last attempt, the source node triggers an exception back to the application to notify this failure. The application may wish to lower the end-to-end reliability requirement since the discovery failure indicates that the network cannot provide a high reliability of service. Alternatively, the application may wish to retry the transmission at a later time.

3.3 Route Maintenance

After each successful route discovery takes place, the source node can deliver its data to the destination node through a set of paths. However, these paths may break at any time instant due to the dynamic nature of network topology in ad-hoc wireless networks. In the worst case, all paths may break and the source node can no longer transmit data to the destination node. In order to maintain a reliable and seamless network connection, route maintenance is necessary to ensure the up-to-date end-to-end reliability is at an acceptable level relative to P_u.

We assume that in the case of single path failures, the MAC layer protocol is able to notify the network layer. The source node may then respond simply by stop sending data through the broken path. However, such failure does not trigger the route maintenance.

The route maintenance takes place under two scenarios. The first is when the time window t_w at the source node expires; the second is at the time instant when all paths are broken. Route maintenance in the first scenario requires examination of up-to-date reliability be-

fore deciding whether a new route discovery is necessary, whereas in the second scenario, the source node immediately initiates a new route discovery without any examination.

When t_w at the source node expires, the source node validates the end-to-end reliability by sending a route check (RCHK) message to each of the existing paths. The end-to-end reliability is valid if its reliability is above the acceptance level defined as $v * P_u$, where v is an application-dependent parameter less than 1.0, and is referred to as *tolerance*. The objective of providing a *tolerance* parameter is to balance the amount of control messages resulting from route discoveries during a data transmission and the degree of end-to-end reliability satisfying the P_u requirement. The preference of choosing a right *tolerance* is application-dependent. The closer the tolerance to 1.0, the stricter the requirement is and the greater amount of control message traffic may be triggered.

Table 4 indicates the fields inside the RCHK message. The *path* field of the RCHK message indicates the path that this message traverses to collect the accumulated path availability in the $\Pi_{acc}(t_w)$ field. When the intermediate node receives a RCHK message, it updates the $\Pi_{acc}(t_w)$ field by multiplying its value with the link availability of the next link along the path. Once the destination node receives the RCHK message, it immediately replies this message with a route check reply (RCHK-RP) message sending back to the source node as shown in Table 5. The source node collects the RCHK-RP packets to validate the end-to-end reliability. If the validation fails, the source node immediately performs a route discovery while it continues to deliver data through existing paths.

Table 4. RCHK in MP-DSR

1. Source ID
2. Destination ID
3. RCHK
4. t_w
5. P_u
6. $\Pi_{acc}(t_w)$
7. Path: S, ..., D

Table 5. RCHK-RP in MP-DSR

1. Source ID
2. Destination ID
3. RCHK-RP
4. Π
5. Path: S, ..., D

4 Optimization

In the previous section, we have presented the basic operations of our MP-DSR protocol. Further optimizations may be performed to improve the routing performance and reduce routing overhead with MP-DSR. We discuss some alternatives to optimize the proposed MP-DSR in this section.

4.1 Timeout Control

In Section 3.2.1, we have presented a mechanism for restricting the number of RREQ messages in the destination node by setting up a timeout period. The destination node waits until the timeout before it proceeds to execute the path selection algorithm as specified in Figure 2.

To improve the efficiency of the path selection process at the destination node, rather than waiting for the timer to expire in order to begin the selection algorithm, a better approach is to allow the destination node to compute the selection as soon as a RREQ message arrives and to finish when it collects sufficient RREQ messages to satisfy the requirement. In this approach, when the destination receives a RREQ message, the destination triggers an event to invoke the path selection algorithm in Figure 2 based on all the arrived RREQ messages. Once the destination node is able to obtain a set of paths that can satisfy the end-to-end reliability requirement, P_u, it replies to the source node and discards any subsequent arriving RREQ messages.

This approach could also take advantage of a timeout mechanism: if the timeout expires and the destination node still cannot discover multiple paths to satisfy the end-to-end reliability requirement, the destination node sends a failure message back to the source node. Hence, there is no fixed time that the destination node needs to wait, and the maximum waiting time is a fixed timeout period. This can significantly increase the overall throughput of the network. However, the trade off is that the computing power needed to perform the path selection algorithm can be exponentially increased, especially when the destination node is able to find multiple paths just upon timeout.

4.2 Routing Reliability Repair

When end-to-end reliability is below the P_u requirement during the transmission, route maintenance needs to be performed. Rather than executing a route discovery again, MP-DSR can select additional paths based on history information.

With this approach, the destination node may refer to the current disjointed set calculated in the path selec-

tion phase for the purpose of end-to-end reliability repair. The destination node selects the set U in $Y - N[i]$ where $N[i]$ denotes the currently transmitting paths. The source node then sends RCHK message through the paths in U, and sets a timeout period to wait for any RCHK-RP messages. A set Q is defined as the paths in U in which the source node receives the RCHK-RP message during the timeout period. After this timeout, the source node re-calculates the end-to-end reliability using paths in Q. If the requirement can be satisfied, source node may then update the route cache and use those paths in Q for data transmission. Otherwise, a route discovery still need to be performed as defined in Section 3.2.

4.3 Shorter Route Discovery

Another possible optimization is to allow the intermediate node to unicast RREQ message if the destination node is its neighbor and the link between them satisfies the path reliability requirement. Such RREQ message handling reduces the number of hops a RREQ message needed to travel and it also eliminates unnecessary broadcast. Further, a path with more intermediate nodes means lower end-to-end reliability. Thus, eliminating additional RREQ relay significantly improves the routing performance. However, if the destination node is a neighbor of the source node, multicast is still adopted for route discovery in case the direct route is unable to satisfy the P_u requirement and multiple paths are required.

5 Performance Evaluation

In order to demonstrate the effectiveness of MP-DSR, we evaluate our proposed protocol and compare its performance to the Dynamic Source Routing protocol (DSR). We have implemented MP-DSR using the Global Simulation (GloMoSim) Library [10]. Our simulation environment consists of 20 mobile nodes in a rectangular region of size 1000 meters by 1000 meters. The nodes are randomly placed in the region and each of them has a radio propagation range of 250 meters. The channel bandwidth is assumed to be 1 Mb/sec. 20 constant bit rate (CBR) flows are deployed for data transmission. Simulation time is 4 hours for every session.

We choose the random waypoint model as our node mobility model. All mobile nodes have their minimum mobility speed fixed at 0 m/s, and their pause time is 0 seconds after each node reaches its epoch destination. We simulate MP-DSR that includes the "shorter route discovery" optimization method, presented in the previous section.

5.1 Performance Metrics

Three performance metrics are introduced for performance evaluations: *Success Delivery Rate (SDR)*, *Control Overhead Ratio (COR)* and *Error Ratio*. They are defined as follows:

$$\text{SDR} = \frac{\text{Number of Data Received}}{\text{Number of Data Originated}}$$

$$\text{COR} = \frac{\text{Number of Control Data Sent}}{\text{Number of Data Received}}$$

$$\text{Error Ratio} = \frac{\text{Number of Error Packets}}{\text{Number of Data Received}}$$

Note that all the performance metrics are measured in the network layer and data is sent redundantly once multiple paths are established.

5.2 Simulation Results

In this section, we present our simulation results for both DSR and MP-DSR protocols, followed by performance analysis and comparisons.

We begin by examining the effects of the maximum mobility speed on the performance of MP-DSR and DSR. The results are shown in Figures 3, 4, and 5. The epoch length is 30 seconds, and $P_u = 0.6$ in these simulations. Figure 3 shows the success delivery ratio for both DSR and MP-DSR. It illustrates that our proposed MP-DSR outperforms the DSR at any mobility speed ranging from 1 to 4 meters/second. We notice that at low mobility speeds (e.g. 1 m/s), MP-DSR performs similarly to DSR due to the relative stationary node movement. In addition, the simulation results demonstrate the ability of MP-DSR to obtain consistent success delivery ratio regardless of the change in node mobility speed. In contrast, DSR suffers in its success delivery ratio when the maximum mobility speed increases. Figure 4 and Figure 5 demonstrate the packet overhead and the error ratio at different level of node mobility, respectively. Again, MP-DSR achieves a lower error ratio compared to DSR. In average, MP-DSR has a lower control overhead ratio than that of DSR.

Figures 6, 7, and 8 illustrate how MP-DSR responds to the end-to-end reliability requirement, P_u. The epoch length is 30s and the maximum mobility speed is 2.5m/s. The results in Figure 6 show that MP-DSR achieves a higher success delivery rate than DSR. Figure 7 shows a lower control overhead ratio in MP-DSR. In addition, the control overhead ratio of MP-DSR remains consistent as routing reliability increases. Similarly, the error

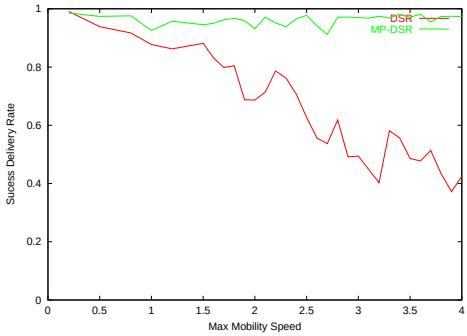

Figure 3. Success delivery rate as a function of max mobility speed

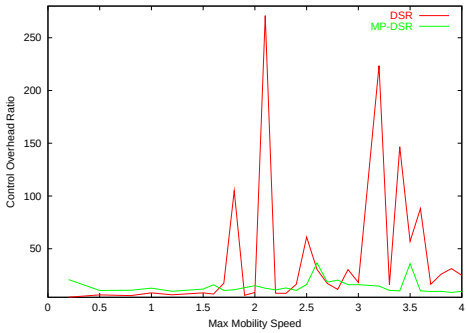

Figure 4. Control overhead ratio as a function of max mobility speed

ratio of MP-DSR is significantly lower than that of DSR as indicated in Figure 8, and this ratio reduces gradually as P_u increases. These observations show the effectiveness of MP-DSR to reduce the probability of path failure.

6 Related Work

On-demand routing protocols generally perform well for wireless ad-hoc networks, since the flooding of route request messages is only performed when a route is needed, rather than periodically as in proactive routing protocols. The degree of flooding is further reduced by using multi-path routing protocols, which have been proposed to discover multiple paths for data transmission. Such protocols can be considered as a hybrid of proactive and on-demand routing, because route discovery is invoked on-demand while route maintenance is done on a proactive basis. Examples of such multi-path protocols include Temporally-Ordered Routing Algorithm (TORA) [8] and Split Multi-path Routing (SMR) [5]. In TORA, the source node constructs mul-

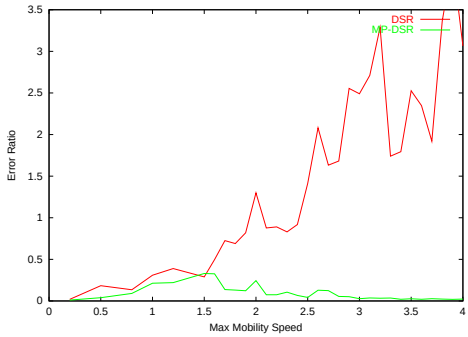

Figure 5. Error ratio as a function of max mobility speed

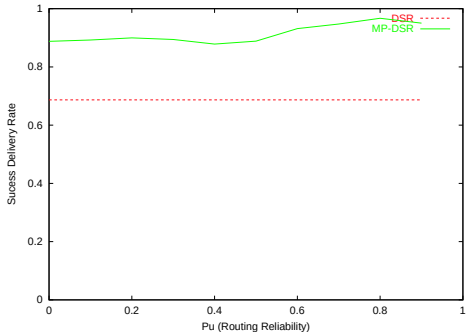

Figure 6. Success delivery ratio as a function of P_u

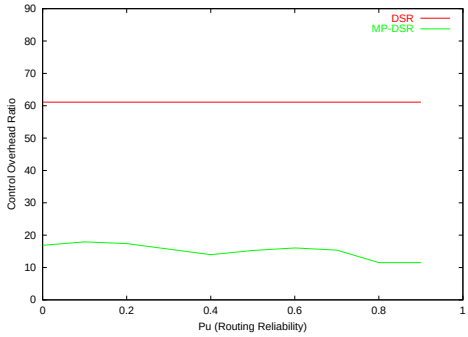

Figure 7. Control overhead ratio as a function of P_u

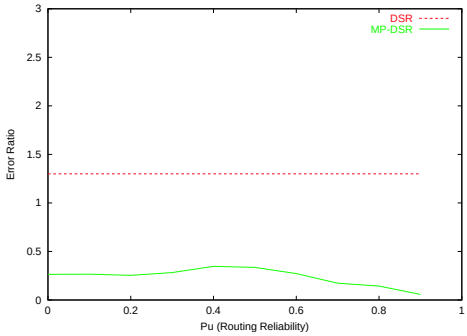

Figure 8. Error ratio as a function of P_u

tiple routes by flooding a query message followed by a set of update messages. However, TORA does not have any mechanisms to evaluate the quality of these multiple paths and this leads to its poor performance. MP-DSR overcomes this problem by selectively choosing more reliable paths and by providing soft guarantees on the end-to-end reliability. SMR [5] extends DSR in the way that the destination can discover two paths for each route request, in which one is the shortest path, and the other is the maximum disjoint path. There is no explicit enforcement of disjoint paths and this differs from our work, because our algorithm enforces the use of disjoint paths in its route discovery in order to use the definition of path reliability to provide end-to-end reliable service.

Previous work in QoS routing for ad-hoc wireless networks focuses on guarantees with respect to bandwidth, cost and delay. One of such routing protocols is the ticket-based QoS routing protocol [2]. It considers two kinds of routing criteria: the delay-constrained least-cost routing and the bandwidth-constraint least-cost routing. It uses ticket-based probing to control the number of route queries and to find multi-path in parallel. In comparison, our MP-DSR considers the dynamic nature of network topology as well as the importance to offer continuous network connection in certain mission-critical applications. Thus, the objective of our protocol is to improve the level of service by providing guarantee with respect to *end-to-end reliability*, and to probabilistically guarantee the required connection lifetime. In addition, our MP-DSR differs in the way of searching multiple paths; the route discovery in our protocol relies only on local link availability information at each intermediate node to perform the route request (RREQ) message forwarding, without resorting to any global information as was used in [2].

7 Conclusion

In this paper, we have proposed a multi-path dynamic source routing (MP-DSR) protocol to provide data transmission with higher end-to-end reliability in wireless ad-hoc networks. The objective is to provide a reliable route for packet transmission with a minimum network overhead. We introduce a QoS parameter, *end-to-end reliability*, which is used for path selections. Applications can specify their end-to-end reliability requirements to control the routing failure probability. With our algorithm, data transmission can then be soft provisioned with limited extra overhead. End-to-end reliability is also maintained throughout the whole transmission life time. Simulation results show that our MP-DSR can offer higher and more consistent success delivery ratio than DSR. In addition, the lower error ratio of MP-DSR illustrates that its end-to-end transmission is more reliable. Finally, the control message overhead in MP-DSR is almost identical to that of DSR in average cases.

References

[1] J. Broch, D. B. Johnaon, and D. A. Maltz. Dynamic Source Routing (dsr). *Internet Draft, draft-ietf-manet-dsr-03.txt*, 1999.

[2] S. Chen and K. Nahrstedt. Distributed Quality of Service Routing in Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 17(8), August 1999.

[3] S. Jiang, D. He, and J. Rao. A Prediction-based Link Availability Estimation for Mobile Ad Hoc Networks. In *INFOCOM*, pages 1745–52, 2001.

[4] D. B. Johnaon and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, pages 153–181, 1996.

[5] S. Lee and M. Gerla. Split Multipath Routing with Maximally Disjoint paths in Ad Hoc Networks. In *Technical Report in University of California*, 2000.

[6] A. B. McDonald and T. F. Znati. A Mobility-Based Framework for Adaptive Clustering in Wireless Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 17(8), August 1999.

[7] A. B. McDonald and T. F. Znati. A Path Availability Model for Wireless Ad-Hoc Networks. In *Proceedings of IEEE Wireless Communications and Networking Conference*, 1999.

[8] V. Park and S. Corson. Temporaly-ordered routing algorithm. *Internet Draft*, August 1998.

[9] C. Toh. Associativity-Based Routing for Ad-Hoc Mobile Networks. *Wireless Personal Communications*, 4:103–139, 1997.

[10] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: a Library for Parallel Simulation of Large-Scale Wireless Networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulations*, May 1998.