Load-Balanced Routing in Ad hoc Networks

Sudharsan Rangarajan*
Email: sudrang@soe.ucsc.edu
*Computer Engineering Department
University of California,Santa Cruz
Santa Cruz,CA 95064.

J.J. Garcia-Luna-Aceves*+
Email: jj@soe.ucsc.edu
+Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto,CA 94304.

Abstract— **Many multipath routing protocols proposed to date have used destination sequence numbers to provide multiple loop-free paths to destinations. We present the first on-demand multipath routing protocol for ad hoc networks that uses source sequence numbers to maintain loop-free routes. We propose a novel load-balancing scheme that incurs very little control overhead. Extensive simulations illustrate that the proposed multipath protocol performs better than single-path approaches and that the proposed load-balancing scheme performs better than basic round-robin scheduling.**

I. Introduction

Existing approaches to routing can be classified as on-demand or proactive. Proactive routing protocols maintain routing entries for all destinations. On-demand routing protocols maintain routing information for only those destinations for which there is traffic. The focus of this paper is on the latter type of approach.

There has been considerable work on multipath routing, with most work focusing on multipaths consisting of multiple link- or node-disjoint paths [6] [4]. However, it has been shown [2] that disjoint paths are not more reliable than non-disjoint paths. To achieve disjoint paths, these protocols rely on destination-only replies, which incurs more overhead by forcing the destinations to respond to RREQs. Furthermore, the protocols proposed to date are based on destination sequence numbers or source routes. The limitation with the way in which destination sequence numbers have been used is that the resulting protocols may incur loops if local repairs are allowed and routing state is lost in relays [3]. The limitation with source routes is that packet headers are much longer and source routes become stale very quickly as nodes move.

In any on-demand routing scheme, each route request (RREQ) is uniquely identified by the identifier of the source and a sequence number assigned by the source, which we call the source-sequenced label (SSL). Recently, is has been shown that robust on-demand loop-free routing protocols can be implemented based solely on SSLs [1], and that such protocols can perform much better than those based on destination sequence numbers or source routes. However, the loop-free conditions provided in this prior work apply only to single-path routing. This motivates the need for a multipath loop-freedom condition based entirely on SSLs. We present the first on-demand multipath routing algorithm based on SSLs. Section II provides an overview of destination-controlled, source-sequenced labeling (DLSR) [1] proposed by Rangarajan and Garcia-Luna-Aceves. We then introduce a new loop-freedom condition for DLSR, and show how the loop-freedom of DLSR can be extended to obtain multiple loop-free paths.

Section III reviews the labeled source-sequenced distance label (LSR-D), which was first proposed in [1]. In DLSR, only the destination can reply to RREQs, and LSR-D builds on DLSR to allow intermediate nodes to reply to RREQs. We extend the loop-freedom condition of LSR-D and present MLSR-D, an on-demand multipath routing algorithm that allows intermediate nodes to reply to RREQs. Section IV illustrates the need for the new loop-freedom condition by means of an example.

There are no prior on-demand routing schemes that monitor the network state and change the multipaths used accordingly. Existing protocols are concerned with the network state only during route establishment [8] or incur excessive control traffic to cope with changing network conditions [7]. Section V presents a novel scheme for load-balancing multiple loop-free paths. We utilize two kinds of information to schedule packets over different next hops for a destination: path information and link information. The metric we use is updated at the rate that is necessary, and without introducing extra control packets. Section VI presents simulation results indicating that the multipath routing scheme we propose performs better than single-path approaches. Table 1 summarized the terminology and variables used in the rest of this paper.

II. On-Demand Multipath Routing with Destination-Only Replies

A. DLSR

In the Destination-controlled, Source-sequenced Labeling (DLSR) algorithm, only the destination of a route request (RREQ) is allowed to reply. DLSR makes every node associate with every RREQ that it processes a unique tuple (node id, sequence number) known as the relay-sequenced label (RSL). For the source of the RREQ, the RSL and SSL are the same. The RREPs traverse the reverse path to the source built by the RREQ and cause nodes to switch successors

[1]This work was supported in part by the Baskin Chair of Computer Engineering at UCSC, the National Science Foundation under Grant CNS-0435522, and the U.S. Army Research Office under grant No.W911NF-041-1-0224. Any opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the funding agencies.

TABLE I

TERMINOLOGY

ID_A	A strictly increasing hardware counter that node A possesses
$ID_D^{\star A}$	The value of ID_A when node A last updated its routing table entry for destination D
SID_D^A	Is the same as $ID_D^{\star A}$
CID_D^A	The last known value of node A's RSL when it updated its routing table for destination D
S_D^A	Node A's successor for destination D
SSL	A unique $[source, id]$ tuple associated with every RREQ by its source
RSL	A unique $[node, id]$ tuple associated with every RREQ a node processes
$SSDL$	It is a tuple of $[SSL, distance]$
$id(B)_D^A$	The id in RSL(B, id) from the RREP that A receives from or transmits to a neighbor B
$id(A)_{DB}^A$	The id in RSL(A,id) from the RREP for destination D sent by node B to node A
rt_D^A	The set of routes that node A has for destination D.
$rsl(x)$	The RSL contained in x
$ssl(x)$	The SSL contained in x

for the destination. If a node takes part in multiple route computations at a time for a destination, the aggregate DAG is not necessarily acyclic. Accordingly, a loop-freedom condition (SSC) was proposed [1] taking this into account.

Each node A has a hardware clock that is used to derive a strictly increasing 64 bit source sequence number ID_A. SID_D^A is equal to $ID_D^{\star A}$, which is the last known value of $ID_A(t)$ when node A updated its routing table entry for D at time t. If there is no routing state, SID_D^A is set to $ID_A(t)$, where t is the time that A initiates or relays a RREQ for D. CID_D^A is the last known value of node A's RSL when it updated its routing table for destination D. If there is no routing state, it is set to invalid.

If node A contains a routing table entry for destination D, it must maintain the successor S_D^A, the start identifier SID_D^A and CID_D^A, which are used to check if a certain neighbor is a loop-free successor for D. If $CID_D^A(t)$ is invalid at time t, where t is the time that A initiates or relays a RREQ for D, it is set to $SID_D^A(t)$. $id(A)_{DB}^A$ represents the id in the RSL(A,id) obtained from the RREP for destination D sent by node B to node A. $id(B)_D^A$ represents the id in RSL(B, id) from the RREP that A receives from or transmits to a neighbor. rt_D^A is used to represent the routes node A has for destination D. The operators $rsl(x)$ and $ssl(x)$ are used to represent the RSL and SSL of x, where x can be a routing table entry, a RREP or a SSDL (will be explained later).

The following rules are required for the processing of RREQs and RREPs at node A:

Rule 1: Node A must increment ID_A every time it relays or originates a RREQ.

Rule 2: If node A needs a route for destination D, it issues a RREQ identified by SSL(A, ID_A). The RSL and the SSL are the same if node A is the node originating the RREQ.

Rule 3: If node A receives a RREQ identified by SSL (O, id_O^{req}) from neighbor C for which A is not the destination, node A caches the RSL(C, id_C^{req}) for this SSL. Node A processes each RREQ only once, and forwards a RREQ to its neighbors with SSL (O, id_O^{req}) and its own RSL(A, ID_B).

Rule 4: If node A is the destination of a RREQ received from neighbor I with SSL(A, id_A^{req}), it sends a RREP carrying the same SSL of the RREQ, and the RSL(I, id_I^{req}).

Rule 5: When node A receives a RREP for destination D identified by SSL (O, id_O^{rep}) and carrying RSL(A, id_I^{rep}), it must use xSSC to determine if the reply can be accepted and relayed. If the RREP can be relayed, node A must find the pair (B, id_B^{req}) it cached for this (O, id_O^{rep}), and send a RREP to neighbor B with RSL(B, id_B^{req}) and SSL(O, id_O^{rep}).

As we have stated, the original loop-freedom condition based on SSLs [1] was meant for single-path routing. For convenience, we restate it below.

Source sequence-number condition (SSC): Node A can accept node B as a successor for destination D at time t if $id(A)_{DB}^A(t) \geq SID_D^A(t)$, where $SID_D^A(t) = ID_D^{\star A}(t)$.

We refer the reader to [1] for a proof. To address loop-freedom over multipaths, we modify SSC as follows, and introduce conditions for relaying and updating.

Extended source sequence-number condition (xSSC): Node A can accept node B as a successor for destination D at time t if one of the following conditions hold:

Condition 1 : $id(A)_{DB}^A(t) \geq SID_D^A(t)$, where $SID_D^A(t) = ID_D^{\star A}(t)$.

Condition 2 : $id(A)_{DB}^A(t) = CID_D^A(t) < SID_D^A(t)$. $SID_D^A(t)$ is not updated to $ID_A(t)$.

Source sequence-number relay condition (xSRC): If condition 1 of xSSC is satisfied, node A must relay a RREP received from neighbor B to neighbor C and invoke the CID update rule (stated below). If condition 2 of xSSC is satisfied, the node A can relay the RREP. It is not necessary to do so though.

CID update rule:. If node A accepts node B as a successor for destination D at time t, then node A must set $CID_D^A(t)$ to $id(A)_{DB}^A(t)$.

The following two lemmas are taken from [1] and stated without proof.

Lemma 1: $SID_D^A(t_1) \leq SID_D^A(t_2)$, where $t_1 < t_2$.

Lemma 2: There exists a causal relation $\rightsquigarrow$ between the event of reporting the value of $id(B)_D^A$ at time t to neighbor B and the event that node A uses a value of $id(A)_D^A$ to update its routing table at time t^-, where $t = t^- + \epsilon$ and ϵ is the time to update the route table.

We establish the following two lemmas when nodes follow rules 1 to 5, and use xSSC to change successors.

Lemma 3: $CID_D^A(t1) \leq SID_D^A(t2)$ when $t1 \leq t2$.

Proof: Consider the case where $t1 = t2 = t$. Say that node A receives a reply for destination D from node B at time t^-. If the RREP satisfies condition 1 of xSSC, then, by lemma 2, $id(A)^A_{DB}(t^-) \geq SID^A_D(t^-)$. By the CID update rule, $CID^A_D(t) = id(A)^A_{DB}(t^-)$. Note that $id(A)^A_{DB}(t^-) < SID^A_D(t)$. We therefore have $CID^A_D(t) < SID^A_D(t)$. If the RREP satisfies condition 2 of xSSC, then it must be true that $id(A)^A_{DB}(t^-)$ was processed before, and hence there exists a time $t1 < t$ at which SSC was satisfied, i.e., $CID^A_D(t1) = id(A)^A_{DB}(t^-) < SID^A_D(t)$. The acceptance of this RREP implies no change in the values of CID^A_D and SID^A_D since $t1$ and hence $CID^A_D(t) < SID^A_D(t)$.

Let $t1 < t2$. We know that $CID^A_D(t1) \leq SID^A_D(t1)$. By Lemma 1, $SID^A_D(t1) \leq SID^A_D(t2)$ and hence we have $CID^A_D(t1) \leq SID^A_D(t2)$. ■

Lemma 4: $CID^A_D(t1) \leq CID^A_D(t2)$ if $t1 < t2$.

Proof: If node A does not relay a RREP for D after relaying a RREP for D at time $t1$ until time $t2$, then by the CID update rule $CID^A_D(t1) = CID^A_D(t2)$. If a RREP for D at some time $t2$ satisfies condition 1 of xSSC, by lemma 2 there exists a neighbor B such that $id(A)^A_{DB}(t2^-) \geq SID^A_D(t2^-)$. Hence, by the CID update rule, we have $CID^A_D(t2^-) <= SID^A_D(t2^-) \leq id(A)^A_{DB}(t2^-) = CID^A_D(t2)$, i.e., $CID^A_D(t2^-) <= CID^A_D(t2)$. Note that this is true even on reboots. ■

Theorem 1: If nodes use xSSC to change successors, no routing table loops can form.

Proof: Assume that a loop $L_D(G)$ is formed at time t and the directed successor graph for destination D, which we denote by $S_D(G)$, was loop-free at every prior instant. A loop can be formed only if at least one node changes its successor at time t to its ancestor in $S_D(G)$. When all the nodes follow xSSC, we show by contradiction that a loop cannot be formed.

Let the loop be formed when node i processes an input event at time t and makes node a its new successor $s^i_D(t)$, where $b = s^i_D(t_b) \neq a$ and $t_b < t$. Clearly $P_{aD}(t)$ must include $P_{ai}(t)$. Let $P_{ai}(t)$ consist of the nodes {a= s[1,new], s[2,new], ..., s[k,new],...,i}.

The notation s[k, new] indicates the k^{th} hop in the path $P_{ai}(t)$ at time t, and s[k+1,new] its successor at time t. The last time that node s[k,new] updates its routing table entry up to time t and sets $s^{s[k,new]}_D$=$s[k+1, new]$ is denoted by $t_{s[k+1,new]}$, where $t_{s[k+1,new]} \leq t$. Therefore $s^{s[k,new]}_D(t_{s[k+1,new]})$= $s^{s[k,new]}_D(t)$. Because nodes joining P_{aD} do not switch to any new successors subsequently, $CID^{s[k,new]}_D(t_{s[k+1,new]}) = CID^{s[k,new]}_D(t)$. Let $t_{s[k+1,old]}$ denote the time at which node s[k,new] sent a reply that constitutes the last reply from such a node that is processed by node s[k-1, new] up to time t. We denote the successor of node s[k, new] at time $t_{s[k+1,old]}$ by s[k+1, old]. Note that$t_{s[k+1,old]} \leq t_{s[k+1,new]} \leq t$, and that $s[k+1, old]$ is not necessarily $s[k+1, new]$.

If node A accepts a reply from node B for destination D at time t, then $id(A)^A_{DB}(t) \geq CID^A_D(t)$. If the reply satisfies condition 1, this is evident from lemma 3. If the reply satisfies condition 2, then $id(A)^A_{DB}(t) = CID^A_D(t)$. Also note that if a reply satisfies condition 1, then by the CID update rule we have $id(A)^A_D(t^-) = CID^A_D(t)$. If it satisfies condition 2, this is obviously true.

For a loop to be formed after time t, $P_{aD}(t)$ must exist. We now derive the following inequality along the path $P_{ai} \subset P_{aD}$ at time t, when nodes use xSSC to switch successors.

$$CID^i_D(t) \leq id(i)^i_{Da}(t) = id(i)^a_D(t_{s[2,old]}) \rightsquigarrow$$
$$CID^a_D(t^-_{s[2,old]}) \leq id(a)^a_D(t^-_{s[2,old]}) = CID^a_D(t_{s[2,old]})$$
$$\leq CID^a_D(t_{s[2,new]}) \leq id(a)^a_{Ds[2,new]}(t)$$

$$= id(a)^{s[2,new]}_D(t_{s[3,old]}) \rightsquigarrow ... \rightsquigarrow CID^{s[k,new]}_D(t^-_{s[k,old]})$$
$$\leq id(s[k,new])^{s[k,new]}_D(t^-_{s[k+1,old]})$$

$$= CID^{s[k,new]}_D(t_{s[k+1,old]}) \leq CID^{s[k,new]}_D(t_{s[k+1,new]})$$
$$\leq id(s[k,new])^{s[k,new]}_{Ds[k+1,new]}(t)$$

$$= id(s[k,new])^{s[k,new]}_D(t_{s[k+1,old]}) \rightsquigarrow ... \rightsquigarrow$$
$$CID^i_D(t^-_b) \leq id(i)^i_D(t^-_b) = CID^i_D(t_b) \leq CID^i_D(t)$$

This leads us to the conclusion that $CID^i_D(t) \leq CID^i_D(t)$. If by the chain of inequalities we have $CID^i_D(t) = CID^i_D(t)$, then $CID^{s[k,new]}_D(t_{s[k+1,old]}) = CID^{s[k,new]}_D(t_{s[k+1,new]}) = CID^{s[k+1,new]}_D(t_{s[k+2,old]})$ for every node s[k,new] in the loop. This implies that every node's routing entries are from the same RREQ. Because the RREQ establishes a DAG and there exists a loop, it must be true that $CID^i_D(t) < CID^i_D(t)$, which is erroneous. Hence, no loops can be formed when xSSC is applied. ■

B. Multipath DLSR

1) Sufficient condition for loop-freedom: We add another constraint to xSSC to obtain loop-freedom when nodes maintain multiple paths to a destination.

Same RSL condition (SRL): $id(A)^A_{DB}(t)$ is the same for all successors B that node A maintains for destination D at time t.

Source Sequence-number condition (MxSSC): Node A must satisfy SRL at every instant of time and can accept node B as a successor for destination D at time t if xSSC is satisfied.

Theorem 2: If a node uses MxSSC to choose successors no loops can be formed

Proof: The proof is similar to the proof for xSSC and hence is not presented here. We note that, because SRL must be satisfied at every instant of time, all the paths are equivalent and hence we can reason in terms of a single CID and SID for every destination that a node has paths for. Observe that setting CID to SID at every instant of time t gives the proof for SSC. ■

2) An Algorithm for On-demand Multipath Routing: We propose an algorithm that obtains a set of shortest paths for a destination placing no restrictions on the nature of the paths. Mosko and Garcia-Luna-Aceves [2] showed that having a mesh structure (non-disjoint paths) improves path reliability and we do the same.

Route Establishment : We say that node A is active for destination D if it is currently taking part in a computation to find a route for D. If Node A has packets for destination D and it does not have a valid route for it, it initiates a route request with SSL(A,id) and buffers data packets. If node A is not currently active for destination D, it becomes active and relays the RREQ. Node A, which is active for destination D and initiated a RREQ for D, will perform a expanding ring search and maintains a RREQ timer set to 2.ttl.latency, where ttl is the time-to-live of the RREQ packet and latency is an estimate of the per-hop latency of the network.

To get shortest paths, destination D can do one two things. It can queue RREQs identified by a certain SSL for a predefined period of time and reply to the RREQs that carry the shortest hopcounts. Alternatively, it can reply to a RREQ identified a certain SSL if the hopcounts associated with the RREQ are monotonically decreasing. Intermediate nodes must relay an RREP if it satisfies condition 2 of xSSC. We follow the second approach for our simulations.

Route Maintenance: We use MAC-layer acknowledgments to maintain paths. The precursor list of a node A for destination D is defined as the set of nodes that use node A to reach D. On transmission of a data packet, if the intended recipient does not acknowledge after a certain number of retries, a node may assume that a link failure has occurred.

Node A performs two steps after detecting a link failure to node B: (a) It removes node B from the precursors list of all the routing entries it maintains; and (b) if it does not have any valid routes for destination D other than via B, node A sends a route error(RERR) for D to the precursors for destination D.

After receiving a RERR for D, node A sends a RERR to its precursors for D if it does not have any valid routes.

III. On-Demand Multipath Routing with Intermediate Node Replies

A. LSR-D

LSR-D allows RREQ's to be answered by intermediate nodes and can hence substantially reduce the control overhead as compared to DLSR. This is done by combining SSL with distances to the destination to create Source-sequenced distance labels (SSDL). SSDL's create a relative ordering of nodes involved in the computation for a particular RREQ SSL. We use $ssdl(rt_D^A)$ to represent the SSDL of rt_D^A.

An SSDL is a tuple $[SSL, distance]$. The components of an $SSDL$ are referred to by $d(SSDL)$ and $SSL(SSDL)$. We define the freshness operator ($\succ$) between two SSDL's as:

$SSDL[(src1, id1), d1] \succ SSDL[(src2, id2), d2]$ if

$$(src1 = src2 \wedge id1 > id2) \vee$$

$$(src1 = src2 \wedge id1 = id2 \wedge d1 > d2)$$

We restate the source sequenced distance labeling condition (SSDLC) from [1] here: Let $SSDL_{DB}^A$ be the SSDL reported to node A for destination D by node B. $SSDL_D^A$ denotes the SSDL node A has for destination D. Node A can switch to node B for destination D at time t, if $SSDL_{DB}^A(t) \succ SSDL_D^A(t)$

It should be noted that the only time the SSDL can be changed is when a destination reset occurs. If there exists no valid SSDL, then a destination reset has to be done.

B. Multipath LSR-D

We present a condition similar to what Marina and Das [4] proposed for destination sequence numbers, to obtain multiple loop-free paths for a destination.

MSSDLC: : Node A maintains a single SSDL for all routes it has to destination D. On a RREP originated from the destination, Node B is a potential successor for destination D if xSSC is satisfied. If node A is the destination of the RREP then

- If node A has a label for D and is the source of the label,the label is not changed.
- Otherwise, $SSDL_D^A$ is updated to $[ssl(ssdl(rt_D^A)), hopcount_{RREP}]$.

On the other hand, if node A is not the destination of the RREP then

- If $\neg\exists rt_D^A$ s.t $ssl(ssdl(rt_D^A)) \geq ssl(RREP)$, node A accepts node B as a successor.
- If $\exists rt_D^A$ s.t $ssl(ssdl(rt_D^A)) = ssl(RREP)$, node A accepts node B as a successor if $d(ssdl(rt_D^A)) \geq hopcount_{RREP}$.
  - If $d(ssdl(rt_D^A)) > hopcount_{RREP}$, then node A must delete all route entries such that $d(ssdl(rt_D^A)) > hopcount_{RREP}$. Node A must relay the RREP and set $d(SSDL_D^A)$ to $hopcount_{RREP}$.
  - If $d(ssdl(rt_D^A)) = hopcount_{RREP}$, node A does not need to relay the RREP

On receiving a RREP not originated from the destination, Node A can accept node B as a next hop for destination D if SSDLC is satisfied.

Lemma 5: If nodes use MSSDLC to choose successors and hence change labels, $SSDL(t1) \succ SSDL(t2)$ if $t1 < t2$.

Proof: Assume that node A was engaged in two computations for a destination from the same source at some time. Let the SSLs associated with them be $SSL(X, xid1)$ and $SSL(X, xid2)$. The RSLs associated with them have the ids $id1$ and $id2$, where $id1 < id2$. When node A accepts a reply, i.e., must satisfy SSC, it sets $SID_D^A > max(id1, id2)$. Node A will change its SSDL only once per RSL, i.e., the first time. Hence $SID_D^A(t1) <= SID_D^A(t2)$. This is true even after reboots as a result of 1. ■

Theorem 3: If nodes use MSSDLC to change successors, no routing table loops can form.

Proof: The proof of LSR-D applies here. By MSSDLC we know that if node A has node B as a successor then $SSDL_D^A \succ SSDL_D^B$. We only need to prove that the label a

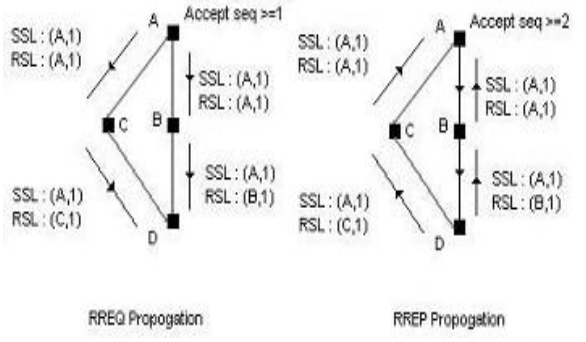

Fig. 1. SSC cannot accept more than one path per sequence number

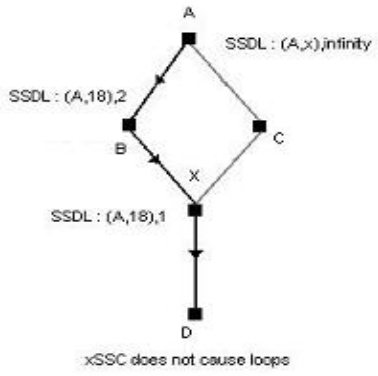

Fig. 2. xSSC

node possesses for a destination increases monotonically when we use xSSC. We proved this in Lemma 5 ■

1) Shortest Paths Multipath Routing: We now propose a multipath routing protocol that uses MSSDLC to establish paths.

Route Establishment : Each RREQ carries the freshest of the SSDL's stored at the nodes along the path traversed by the RREQ, which is denoted by FSSDL. We define an in-order operator ($\cap$) between two SSDLs, SSDL1 and SSDL2, as follows:

$$\cap(SSDL1, SSDL2) = \begin{cases} \text{if (SSDL2} \succ \text{SSDL1)} \\ SSDL2 \\ \text{else null} \end{cases}$$

To get shortest paths, destination D can follow the same approach as in MDLSR. Node B can accept node C as a successor for destination D if it satisfies MSSLDC.

IV. Examples

We illustrate the need for xSSC with examples. Consider Fig. 1. Assume that all sequence numbers are initially equal to 1. Node A initiates a RREQ for destination D and the destination replies to all RREQs it receives (the intermediate nodes will however discard duplicate RREQs). Assume that node A receives the RREP via node B first. By SSC, node A sets its accept sequence for destination D to the current value of ID_A and that is greater than the value of the id in the RSL that node A has for D. SSC maintains a window of computations, but once a reply is obtained for the window no more replies are accepted from that window. Clearly, simply using SSC can accept only a single RREP per sequence number. One way to obtain multiple paths would be to delay relaying RREPs (and hence not move the window immediately). With no constraints on the path selection, all intermediate nodes have to delay relaying RREPs, which would cause considerable delay in obtaining routes.

Consider the example in Fig. 2. Node A initiates a RREQ for destination D with SSL(A,18). Say the RREQ is queued at node B and node C did not receive the RREQ at all. Node A will timeout and initiate another RREQ with SSL(A,19). Say the RREQ with SSL(A,19) arrives before the RREQ with SSL(A,18) at node X. Node X will then assign the RSLs(X,18) and (X,19) to the RREQs respectively. If node X accepted the RREP with SSL(A,18), the only RREPs it can accept are those with SSL(A,18) or whose SSL id is greater than 19. In essence, if a node follows xSSC, it either follows SSC or can choose successors from the computation it last accepted.

V. Load Balancing

We introduce a framework for the exchange of two kinds of information between nodes to attain load balancing. Path information is used to characterize the capability of nodes on the path to a destination, and link information characterizes the capability of the next hop. We first discuss how path and link capability are measured. We then present a mechanism to disseminate at a small cost, the information at the proper rate (we define this term below). This section assumes that MLSR-D is the underlying routing protocol.

A. Metric

We utilize the notion of instantaneous free-buffer space to characterize the capabilities of nodes and hence paths. The choice of this metric is based on the observation that the lesser the free space, the higher the service time and more likely the packet drops.

We assume that all nodes have the same buffer size F. Let N_D^A=$\{n_1,n_2,...,n_k\}$ be the set of next hops node A has for destination D. At time t, Let $S_{n_1}^A(t)$ and $S_{Dn_1}^A(t)$ be values stored at node A that represent the capability of node n_1 and the path capability via node n_1 to destination D respectively.

V_A represents the value reported by node A of its own capability. V_A^D represents the value reported by node A that characterizes the capability of the path to destination D. Let $F_A(t)$ be the instantaneous free buffer space at time t. Node A computes $V_A(t) = F_A(t)/F$ and $V_A^D(t) = \min(V_A(t), N(t))$, where $N(t) = \frac{\sum_{n \in N_D^A} S_{Dn}^A(t)}{|N_D^A|}$. Note that V_A and V_A^D take values between 0 and 1. V_D^D and hence S_{DD}^A always takes the value 1.

B. MAC Layer Acknowledgments

We investigated the use of Hellos to carry metrics that can be utilized for load balancing. We chose to piggyback information on MAC layer acknowledgments. Notice that at node b, both $V_b^D(t)$ and $V_b(t)$ can be represented using 1 byte each. We also use a two byte sequence number field to uniquely identify the destination. This information is present in the original MAC frame and is just copied onto the acknowledgment. The overhead is small and the rate of information flow is as necessary.

C. Scheduling

Each node attempts to schedule a group of k packets (a predefined constant). For node A, let $P_{An_1}^D(t)$ denote the goodness of utilizing node n1 to reach destination D and let $P_{An_1}^D(t)$. Let $N_{An_1}^D(t)$ denote the number of packets (out of k) scheduled for destination D via n_1. We then have:

$$P_{An_1}^D(t) = \frac{(1-\beta) * S_{n_1}^A(t) + \beta * S_{Dn_1}^A(t)}{\sum_{n \in N_D^A} (1-\beta) * S_n^A(t) + \beta * S_{Dn_1}^A(t)}$$

$$N_{An_1}^D(t) = P_{An_1}^D(t) * k$$

where β is the inverse of the hop count to D.

When the hop count is 1, node capability of the next hop (or destination) is completely ignored. The idea is to penalize other flows via the destination. It is important that we weight the information because path information takes a longer time to propagate than link information. A good weighting function will make unnecessary the need to deal with congestion. The current weighting function is bad because it penalizes all path information equally.

D. Dealing with Congestion

Nodes can utilize the value of link information to agree on a notion of congestion. When node A perceives all its next hops to D being congested, it tends to delay packets to D to relieve spots of congestion and performs a local repair. This local repair can only be answered by nodes having sufficient free buffer space(50%). Inability to clear congestion will result in the queuing of packets getting noticed by predecessors of A for D and if necessary they can perform the same set of actions. Path information will be utilized to send lesser packets over this path. If necessary, the source can perform a route discovery process, although we did not check out this possibility. We delay packets exponentially as $j * 2^m$, where j is a predefined constant and m starts with 0 and is incremented by 1 for every schedule of packets sent.

VI. Performance Comparison

We present results over varying loads and mobility for AODV [5], DLSR, MDLSR, LSR-D and MLSR-D. Simulations were carried out in Qualnet 3.5.2 [9]. The network consists of 50 nodes and the terrain has dimensions of 1500m x 300m. Traffic is generated by CBR sources with a data packet size of 512 bytes. The number of flows at any instant of time in the network is either 10 or 30, with each flow generating 4 packets per second. We used 802.11 as the MAC layer with a transmission range of 275m and throughput 2 Mbps. Each simulation was run for 900 seconds for a total of 9 seeds. We present results for MLSR-D using round-robin (MLSR-D-RR) only for the 30 flows case. The AODV protocol can perform local repair if necessary.

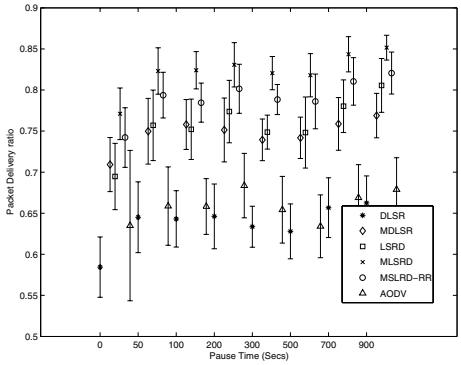

Fig. 3. Packet Delivery Ratio

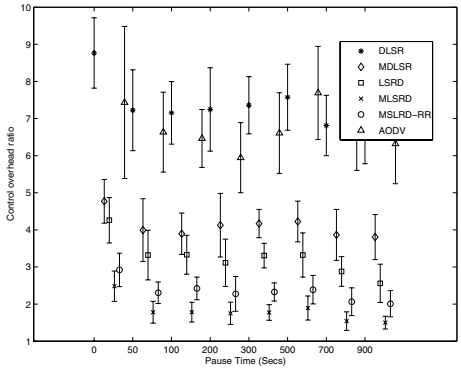

Fig. 4. Control Overhead

We used four metrics as the basis for comparison. Delivery ratio is defined as the ratio of packets delivered to the packets to be delivered. Latency is the end to end delay measured for data packets that reach the destination. Network Load is the ratio of control packets to data packets delivered. Data hops is the ratio of the total number of data transmissions made to the number of data packets received. The last three metrics are to be minimized while we expect to maximize the first metric.

With multipath routing, the number of destination replies and local replies (in the case of MLSR-D) can be large. This could lead to excessive control traffic and hence have adverse effects on the E2E delay. We believe that broadcasting RREPs is a good solution and do so for simulation purposes.

Table II summarizes the results for different metrics by averaging across all pause times. The columns show the mean value and the 95 confidence interval. The packet delivery, control overhead, E2E delay and Data Hops for the 30 flows scenario is shown in 3, 4, and 5. The vertical bars represent the confidence intervals.

Under very light loads, all the protocols perform similarly. This is because the network is not saturated with enough traffic

TABLE II

PERFORMANCE AVERAGE OVER ALL PAUSE TIMES FOR 50 NODES NETWORK FOR 10-FLOWS AND 30-FLOWS

Protocol	Flows	Delivery Ratio	Latency (sec)	Net Load	Data Hops
AODV	10	0.966±0.013	0.068±0.024	0.474±0.167	2.767±0.107
DLSR	10	0.961±0.016	0.078±0.027	0.556±0.187	2.784±0.111
MDLSR	10	0.951±0.017	0.110±0.036	0.599±0.173	2.839±0.114
LSR-D	10	0.971±0.010	0.056±0.013	0.275±0.073	2.682±0.097
MLSR-D	10	0.966±0.012	0.080±0.052	0.256±0.069	2.749±0.115
AODV	30	0.63.9±0.083	1.812±0.618	5.786±1.797	3.550±0.319
DLSR	30	0.637±0.037	2.279±0.373	7.357±0.970	3.778±0.211
MDLSR	30	0.748±0.032	1.231±0.265	4.100±0.636	3.315±0.148
LSR-D	30	0.758±0.040	1.118±0.264	3.262±0.600	3.282±0.178
MLSR-D-RR	30	0.791±0.031	1.102±0.259	2.339±0.387	3.216±0.148
MLSR-D	30	0.823±0.028	1.071±0.308	1.814±0.329	3.077±0.127

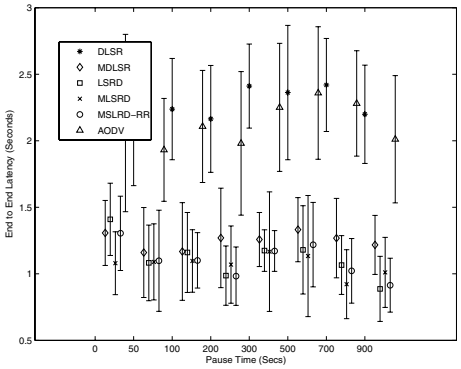

Fig. 5. End to End delay

for the control traffic introduced by the single path routing protocols to cause substantial performance degradation.

As the number of flows increase, we see that the variations of LSR-D have significantly better performance than the variations of DLSR. With the constraint of destination-only replies, DLSR introduces substantial control traffic. Notice that MDLSR convincingly outperforms DLSR. When more routes are obtained in a RREQ computation, the necessity to flood RREQs becomes lesser. Introducing control traffic in a network on a contention based MAC protocol has significant effects. They compete with data packets, causing them to be queued increasing E2E delay and broadcast packets can collide with unicast transmissions causing loss of packets.

MLSR-D has the best performance of the set of protocols proposed. It has excellent packet delivery ratio and as a consequence, the lowest control overhead. However, there is no significant improvement in the end-to-end delay. However, the delays in the case of LSR-D and MLSR-D are of different nature. LSR-D needs to frequently recompute paths. MLSR-D queues packets in the case of congestion. MLSR-D-RR performs better than LSR-D as expected. However, it does not take into account the node characteristics while scheduling packets and hence does not perform as well as MLSR-D.

VII. CONCLUSION

We have shown how loop-freedom conditions previously introduced for single-path routing using source sequence numbers [1] can be extended to provide multiple loop-free paths. We proposed a load-balancing scheme that utilizes path and link information and a novel approach to disseminate such information with minimal control overhead. The results of simulation experiments illustrate that the proposed multipath approach outperforms single-path schemes.

REFERENCES

[1] H. Rangarajan and J.J. Garcia-Luna-Aceves, "On-Demand Loop-free Routing In Ad hoc Networks Using Source-Sequence Numbers," *Proc. IEEE MASS 2005*, Nov. 7–10, 2005 ,Washington D.C.

[2] M. Mosko and J.J.Garcia-Luna-Aceves, "Multipath Routing in Wireless Mesh Networks," *Proc. IEEEE WiMesh 2005*, September 6, 2005, Santa Clara, CA.

[3] J.J. Garcia-Luna-Aceves and H. Rangarajan, "A New Framework for Loop-Free On-Demand Routing Using Destination Sequence Numbers," *Proc. IEEE MASS 2004*, Oct. 25–27, 2004, Fort Lauderdale, FL.

[4] M. Marina and S. Das, "On-demand Multipath Distance Vector Routing in Ad Hoc Networks," *Proc. IEEE ICNP 2001*, Nov. 2001, Riverside CA.

[5] C.E. Perkins and E.M. Royer. "Ad hoc On-Demand Distance Vector Routing," *proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999, New Orleans, LA.

[6] S. Lee and M. Gerla, "SMR: Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks," *Proc. IEEE ICC 2001*, Helsinki, Finland, June 2001.

[7] S.J. Lee and M. Gerla, "Dynamic Load-Aware Routing in Ad hoc Networks," *Proc. IEEE ICC 2001*, Helsinki, Finland, June 2001.

[8] H. Hassanein and A. Zhou, "Routing with Load Balancing in Wireless Ad Hoc Networks," *Proc. MSWiM 01*, Rome, Italy, July 21, 2001.

[9] Scalable Technologies,Qualnet, http://www.qualnet.com.