On Delay-Adaptive Routing in Wireless Networks

Vivek Raghunathan and P. R. Kumar

Abstract— **We address the routing problem in multi-hop wireless networks and present a multipath delay-adaptive loop-free protocol with low overhead and elegant implementability. In earlier work regarding Wardrop routing, a distributed adaptive scheme using delay feedback was proposed and analyzed. In equilibrium, it results in all utilized paths from source to destination having the same delay, which is less than that over unutilized paths.**

In this paper, we attempt to reduce loops when Wardrop routing is used. We also present a completely distributed distance-vector like delay measurement scheme for Wardrop routing. The result is a protocol that is completely distributed, delay-adaptive, and guarantees loop-freedom.

Simulations indicate that the protocol is able to re-route flows to "avoid" each other, and boosts throughput performance in wireless networks. A working implementation of the protocol on the Linux 2.4.20-6 kernel is undergoing testing.

I. INTRODUCTION

In this paper, we investigate multipath Wardrop routing in wireless networks. The Wardrop equilibrium [1] was first studied in a transportation context:

1) Wardrop's first principle: The packet delays along all used paths between source and destination are the same.

2) Wardrop's second principle: The delay along used paths is smaller than or equal to the delays along unused paths.

The protocol presented here is based on earlier work for multipath routing using the STARA routing algorithm [2], where every node in the network maintains a probability vector for routing packets to the destination through each of its neighbors. The probabilities are updated using delay estimates obtained in packet acknowledgments. The probability update scheme is stateless and fully distributed: nodes update their probabilities to a destination independent of the source of the packet. In [3], the probability update law was modified to preserve the sum of components at unity, and Cesaro convergence to the set of (ϵ-approximate) Cesaro-Wardrop equilibria was proved.

The STARA algorithm presented in both papers suffers from practical problems which render its implementation difficult:

1. In equilibrium, STARA produces loop-free routes. However, packets can loop around before convergence.

This material is based upon work partially supported by DARPA/AFOSR under Contract No. F49620-02-1-0325, AFOSR under Contract No. F49620-02-1-0217, USARO under Contract Nos. DAAD19-00-1-0466 and DAAD19-01010-465, and NSF under Contract Nos. NSF ANI 02-21357 and CCR-0325716, and DARPA under Contact Nos. N00014-0-1-1-0576 and F33615-0-1-C-1905.

Vivek Raghunathan and P. R. Kumar are with the Department of Electrical and Computer Engineering, and the Coordinated Science Laboratory, University of Illinois, 1308 W. Main St, Urbana, IL 61801, USA. Email: {vivek, prkumar}@control.csl.uiuc.edu

2. The proposed delay measurement scheme relies on the use of per-packet ACKs to carry delay measurements back to sources (and intermediate nodes). Network layers in the Internet do not have end-to-end ACKs.

We propose the following modifications to STARA to make it easier to deploy:

1. We propose four variants of STARA to provide better control of packet paths while the algorithm is converging. These variants provide different bounds on the packet path length and one of them is loop-free. All these variants converge to the appropriately defined Wardrop equilibrium.

2. We propose a completely distributed delay measurement mechanism to replace the ACK-based scheme proposed in STARA. The mechanism retains the immunity to clock offsets present in STARA. It consists of a light-weight link delay measurement protocol and a distance-vector like neighborhood broadcast of average delay information.

To understand the behavior of the protocol, we have carried out extensive ns-2 simulations of STARA and its variants. In a broadcast medium like wireless, contention for the channel is the bottleneck to performance. Our simulations indicate that in such environments, STARA is able to cause interfering flows to "avoid" each other and boosts performance. STARA is thus an attractive choice for large highly-utilized static wireless networks.

Finally, we have built a working implementation of the STARA routing protocol as a userspace daemon that runs on a custom modified Linux 2.4.20 kernel.

II. RELATED WORK

A large number of protocols have been proposed to find loop-free routes to destinations in mobile environments with minimal routing overhead. These include pro-active [10] and reactive ([8], [9]) routing protocols.

Various multipath extensions of existing reactive protocols have been proposed in the literature ([11], [12], [13], [14], [15]). Packet delivery ratio is the most commonly used metric and throughput-delay performance of the schemes has not been investigated thoroughly. On the other hand, there is a substantial body of literature on dynamic routing and its optimization formulation ([5], [6]). An approximation approach to minimum delay routing [7] provides loop-free invariant conditions and uses link-state routing with a multiple-path topology dissemination algorithm to discover routes while distributing traffic to approximate the algorithm in [6].

III. STARA

We model the wireless network as a graph $G = (V, E)$. For node i and destination d, let $N(i, d)$ = set of neighbors

of i for destination d. For $j \in N(i,d)$, $p_{id}^j[n]$ = probability of routing packets at node i to destination d through neighbor j, $D_{id}^j[n]$ = average delay for packets from node i to destination d through neighbor j, and $\overline{D}_{id}[n]$ = average delay from node i to destination d.

STARA consists of two components [2], [3]: an iterative delay averager, and an iterative routing probability update scheme that uses running estimates of average delays to increase the probability of using paths with lower than average delay.

1) Delay Estimation Scheme: The source timestamps the data packet. On receiving the data packet, the destination sends an ACK with the delay information. The source uses the ACK to record the value $\hat{D}_{id}^j[n]$ = measured delay of the packet sent at time n. In order to produce an averaged estimate of the delay, STARA uses exponential forgetting (with $0 < \gamma < 1$):

$$D_{id}^j[n] = \gamma \ . \ D_{id}^j[n-1] + (1-\gamma) \ . \ \hat{D}_{id}^j[n] \ \ \forall \ j \in N(i,d). \tag{1}$$

2) Probability Update Scheme: STARA uses the probability update scheme:

$$p_{id}^j[n+1] = [p_{id}^j[n] + \alpha[n] \ . \ q_{id}^j[n] \times (\overline{D}_{id}[n+1] - D_{id}^j[n+1])]^+ \ \ \forall \ j \in N(i,d), \text{ where}$$

$$\overline{D}_{id}[n+1] = \sum_{j=1}^{N(i,d)} (D_{id}^j[n+1] \ . \ q_{id}^j[n]), \text{ and}$$

$$q_{id}^j[n] = (1-\epsilon) \ . \ p_{id}^j[n] + \epsilon \ . \ 1/N(i,d) \ \ \forall \ j \in N(i,d).$$

When a packet for destination d arrives at node i, the node routes it to one of $j \in N(i,d)$ with probability q_{id}^j. Thus, the actual probabilities used are a convex combination of p_{id}^j's and a uniform distribution on all neighbors. Observe that the probability update scheme only uses the difference between delays ($\overline{D}_{id} - D_{id}^j$) along different neighbors for a source-destination pair, thereby ensuring that it works even when clocks in the network are not synchronized.

A. Equilibrium point and convergence

Definition An allocation $q = \{q_{id}^j\}$ is Wardrop if

$$q_{id}^j > 0 \ \ \Rightarrow \ \ D_{id}^j = \sum_{j \in N(i,d)} q_{id}^j D_{id}^j = \ min_{\{j \in N(i,d)\}} D_{id}^j \ \ \forall \ \ d,i,j \in N(i,d).$$

If STARA converges, the steady state value satisfies the above condition which implies a delay-equalizing path-flow allocation from every source to every destination [2]. In [3], the Cesaro-convergence of the scheme to the set of $\epsilon-$approximate Wardrop equilibria for a modified variant of the scheme is proved.

IV. CONTROLLING THE PATH LENGTH

In equilibrium, STARA produces loop-free routes. However, packets can loop around while the algorithm is converging. These loops are temporary and the packet has a finite probability of escaping the loop; nevertheless, we would like a degree of control over packet paths during convergence. To do this, we can use two degrees of freedom:

1) N(i,d): The network model in [3] allows us to choose $N(i,d)$ differently for different destinations and still preserve the convergence and equilibrium properties of STARA.

2) Packet state: On the other hand, we might wish to modify the neighborhood set of node i based on the destination as well as some state variable contained in the packet (for example, the path that the packet took before it reached node i). If the state for packets with destination d takes one of $K(d)$ finite values at any node in the network, the problem can be cast in terms of the original problem by replacing each source-destination pair (s,d) with $K(d)$ different source-destination pairs $(s,(d,1)),...,(s,(d,K(d)))$. We can then define separate neighborhood sets $N(i,(d,F))$ at node i for each possible value of the incoming packet state $F = 1,...,K(d)$ at node i. Thus, at node i, we will need to maintain separate probability and delay estimate vectors, parameterized by the possible values of the packet state and update them independently based on corresponding delay measurements from the network.

In both the above cases, the equilibrium is a delay equalizing allocation with respect to the restriction of the network to the appropriate paths, given the rates of each source-destination pair. Thus, in the second case, the delay equalization is with respect to each (source, (destination, state)) tuple.

Based on these ideas, we have the following variants:

A. A-STARA (Additive STARA)

The additive variant (A-STARA) guarantees that all packets from source i to destination d follow paths with length upper-bounded by $S(i,d)+k$, where $S(i,d)$ = shortest-path distance in hops from node i to destination d.

When a source s transmits a packet p to destination d, it marks a field $F(p)$ in the packet with the value $F(p) = S(s,d) + k$. Every node that receives packet p decrements the field $F(p)$ by 1. Thus, k represents a budget of excess hops that the packet can take along its path from s to d. The basic idea of A-STARA is that when a node i receives a packet p with field $F(p)$, node i only considers the neighbors $j \in N(i,d)$ which can reach the destination in $F(p)-1$ hops as active for routing.

A-STARA consists of a delay estimation scheme and a routing probability update rule:

1) Delay Estimation Scheme: We maintain a list of delay estimate vectors $(D_{id}^j)_F[n]$, indexed by the possible values for field $F(p)$ at node i, $F \in \{S(i,d), S(i,d)+1, S(i,d)+2,..., S(i,d)+k\}$. Let $(\hat{D}_{id}^j)_F[n]$ = measured delay of the packet that arrived at node i with field F and was sent to node j at time n. Then, $(D_{id}^j)_F[n]$'s are updated as:

$$(D_{id}^j)_F[n] = \gamma * (D_{id}^j)_F[n-1] + (1-\gamma)(\hat{D}_{id}^j)_F[n], \ \ \forall \ F \in \{S(i,d), S(i,d)+1, ..., S(i,d)+k\}.$$

The delay measurement procedure is cumbersome and can be implemented using protocol generated probe packets (proposed in [4]). The probe packet mechanism can be appropriately modified so that all intermediate nodes can reconstruct the value of field at the time they had routed the packet to the destination. This allows them to update their corresponding delay estimate.

2) Probability Update Rule: Every node i maintains a list of probability vectors $(q_{id}^j)_F$ for destination d, indexed by the possible values of field $F(p)$, $F \in \{S(i,d), S(i,d)+1, ..., S(i,d)+k\}$. In the probability vector corresponding to F, we set $(q_{id}^j)_F = 0 \ \ \forall\ j \in N(i,d)\ s.t.\ S(j,d) > F-1$. The remaining probabilities are updated as follows:

$$N(i,d,F) = \{j \in N(i,d) : S(j,d) \leq F-1\},$$
$$(p_{id}^j)_F[n+1] = [(p_{id}^j)_F[n] + \alpha[n] * (q_{id}^j)_F[n] \times ((\overline{D}_{id})_F[n+1] - (D_{id}^j)_F[n+1])]^+, \quad (2)$$
$$(\overline{D}_{id})_F[n+1] = \sum_{j=1}^{N(i,d)} ((D_{id}^j)_F[n+1] * (q_{id}^j)_F[n]), \quad (3)$$
$$(q_{id}^j)_F[n] = (1-\epsilon) * (p_{id}^j)_F[n] + \epsilon * 1/N(i,d,F), \quad \forall\ j \in N(i,d,F). \quad (4)$$

On receiving a packet for destination d with field $F(p)$, node i routes the packet to one of its neighbors $jinN(i,d,F(p))$ using the probabilities $(q_{id}^j)_{F(p)}$, decrementing the field $F(p)$ in the packet by one.

We now present some properties of A-STARA.

Claim: When packet p arrives at node i, $S(i,d) \leq F(p) \leq S(i,d)+k$. Also, path P between s and d is considered for routing iff the path length $L(P) \leq S(s,d)+k$.

Proof: (Outline) Both proofs rely on the A-STARA rule used by nodes to decide which neighbors are ineligible as next hop neighbors for forwarding packets to a particular destination. Combining the rule with the optimality property of shortest paths gives both results. □.

Corollary: For all packets from s to d, the length of path taken by packets $\leq S(s,d)+k$.

B. M-STARA (Multiplicative STARA)

The A-STARA scheme provides a great deal of freedom in controlling path selection, but is hard to implement:

1. Delay measurement procedure is cumbersome.

2. The routing table and delay estimation complexity at each node scales by a multiplicative factor k.

If simplicity of implementation is the key design parameter, then the multiplicative variant of STARA should be used. M-STARA simply sets the neighborhood set $N'(i,d)$ at node i for destination d as $N'(i,d) = \{j \in N(i,d) : S(j,d) \leq S(i,d)\}$ and uses STARA on this restricted graph. Thus, we always make non-negative progress toward the destination. The advantage is that it can be built on top of STARA by running a distance vector protocol to produce hop distances, and adjusting the neighborhood sets allowed for routing.

C. M-STARA δ-variant

While M-STARA is simple to implement, it does not provide any bound on the average path length. To obtain such a bound, we use the $\delta-$variant of M-STARA. Here, when a packet arrives at node i, we flip a biased coin such that with probability δ, we use the neighborhood $N^1(i,d) = \{j \in N(i,d) : S(j,d) = S(i,d)-1\}$, and with probability $(1-\delta)$, we use the neighborhood $N^0(i,d) = \{j \in N(i,d) : S(j,d) \leq S(i,d)\}$. Since this variant uses an extra state variable to decide which probabilities to use, the probability update and delay estimation rules are similar to A-STARA (2, 3, 4), except that now we only have two possible values for the packet state, with the corresponding neighborhood sets defined above.

We can provide an upper bound on $V(i,d)$, the mean path length of packets from node i to destination d.

Claim: $V(i,d) \leq \frac{1}{\delta} S(i,d)\ \ \forall\ i$.

Proof: (Outline) The proof is by induction on $S(i,d)$. Let $(q_{id}^j)_1$ and $(q_{id}^j)_0$ be the probabilities of routing packet for destination d at node i through neighbor j, parametrized by the state of the coin flip, 1 or 0. Among all nodes i' with $S(i',d) = k$, we consider a node i with maximum value for $V(i,d)$. Then, the dynamic programming recursion for the average path length at node i can be combined with the induction hypothesis to obtain the result. □

D. P-STARA (Parity STARA)

None of the above variants produce loop-free paths at every instant, although A-STARA and the $\delta-$variant of M-STARA produce paths with bounded lengths. On the other hand, both of them need cumbersome delay measurement using probe packets. The M-STARA variant is easier to implement, but does not provide any guarantees on path lengths. The Parity variant of STARA that we describe next is an attempt to obtain loop-freedom and path-length bounds while allowing the design of a delay measurement scheme that can be implemented in a completely distributed fashion.

1. As in A-STARA, we include a field $F(p)$ in the packet which is decremented by 1 by every node along the path.

2. As in M-STARA (δ-variant), we define $N^1(i,d) = \{j \in N(i,d) : S(j,d) = S(i,d)-1\}$, and $N^0(i,d) = \{j \in N(i,d) : S(j,d) \leq S(i,d)\}$.

3. We define the packet state X as $X(F(p)) = (1 + F(p))\ mod\ 2$. Since X has only two possible values at any node, we need to maintain two separate probability vectors and delay estimate vectors for each destination.

4. $F(p)$ is decremented at each node. Note that successive nodes along a packet's path to the destination have a different value of state $X(F(p))$.

The probability update and delay estimation rules for P-STARA are similar to A-STARA (2, 3, 4), with two possible values for the state X and neighborhood sets as defined above. At this point, we have not justified why P-STARA allows us to implement a completely distributed delay estimation scheme. This will be discussed in the next

section. On the other hand, we can show that P-STARA is loop-free at every instant.

Claim: Let packet p with source s and destination d follow path P with length $L(P)$. Then $L(P) \leq 2S(s,d)$.

Proof: Since $F(p)$ is decremented by one at every step, $F(p)$ is alternately even and odd at every node along P. Thus, after every two hops, we are closer to the destination by at least one more hop (we are forced to choose a shortest-path neighbor at one of these two hops). Thus, $L(P) \leq 2 * S(s,d)$. □

Claim: P-STARA produces loop-free paths from every source s to every destination d, provided that the distance estimates $S(s,d)$ are accurate.

Proof: Let P be a path from s to d. P-STARA never forwards packets upstream, i.e., from i to j if $S(j,d) > S(i,d)$. Thus, if P contains a cycle $(i_1, i_2, ...i_n)$, then $S(i_1,d) = ... = S(i_n,d)$. However, if i_1, i_2, i_3 are three consecutive nodes along P, then $S(i_1,d) - S(i_3,d) \geq 1$. This implies that P has no cycle of size $n \geq 2$. □

We can modify P-STARA to generate loop-free paths by using the sequence number technique of [10] even if the distances $S(s,d)$ are not accurate.

Claim: P-STARA produces loop-free paths from every source s to every destination d, even when the distance accurate estimates $S(i,d)$ are not accurate or are converging.

Proof: (Outline) We define the neighborhood sets $N^1(i,d)$ and $N^0(i,d)$ in such a way that sequence numbers do not decrease along a path P to the destination. Thus, a loop can be formed only if the sequence numbers are the same. In this case, we obtain a contradiction by using the fact that the distance estimates $S(i,d)$ decrease by at least one after every two hops along the path P. □

V. AVERAGE DELAY ESTIMATION

STARA and the variants described above rely on obtaining delay measurements using ACKs (or probe packets). This makes delay measurement highly impractical:

1) Architectural issues: The Internet was built on the end-to-end principle, which conceived of the network as merely providing a packet delivery service, while leaving reliability (using ACKs) to higher layers. STARA requires per-packet ACKs to update (network layer) delay estimates at sources (frequently) and routing-protocol generated probe packets to update delay estimates at intermediate nodes (infrequently). If we must use per-packet ACKs, then these would have to be transport layer ACKs, meaning that routing will operate correctly only if all traffic is (TCP-type) reliable traffic which generates ACKs. Even in this case, we will need to add a TCP option to piggyback the forward delay to the source. Modifying transport layer headers to implement a network layer routing protocol violates the architectural separation of functionality.

2) Routing Overhead: We could potentially do away with per-packet ACKs and instead use routing protocol generated probes to update delay estimates everywhere on a frequent basis. This is impractical because of the extra routing overhead, something that is even more acute in low bandwidth wireless 802.11 environments.

A. Distributed Delay Measurement for STARA and M-STARA

We wish to design a distributed delay measurement scheme that measures end-to-end delay by using local delay measurements and local information exchange. Let $[q_{id}^j]$ be the quasi-static routing probabilities in effect during some phase when we wish to estimate the average delays. Suppose that the link delay d_{ij} from node i to its neighbor j is available. (In practice, we have to run a link delay measurement protocol to obtain the d_{ij}; see [16] for details). We observe that the average delay from node i to destination d through $j \in N(i,d)$, D_{id}^j, and the average delay from node i to destination d, $\overline{D}_{id}$, satisfy the equation:

$$D_{id} = \sum_{j \in N(i,d)} q_{id}^j (d_{ij} + D_{jd}). \tag{5}$$

This suggests a rather simple distributed technique to obtain the delay estimates from the measurements of the link delays d_{ij}. On a periodic basis, node i broadcasts its average delay to a destination $\overline{D}_{id}[n]$. On receiving the value of the average delay to d from node j, $\overline{D}_{jd}[n]$, node i updates its estimates as:

$$D_{id}^j[n] = d_{ij} + \overline{D}_{jd}[n], \tag{6}$$

$$\overline{D}_{id}[n] = \sum_{k \in N(i,d)} q_{id}^k D_{id}^k[n]. \tag{7}$$

Claim: If the destination always advertises its average delay to itself as 0, i.e., $\overline{D}_{dd}[n] = 0 \ \ \forall\ n$, then all nodes obtain the correct delay estimates to the destination. That is, $\exists\ n_0 \ \ s.t. \ \ D_{id}[n] = D_{id} \ \ \forall\ n > n_0$, signifying that the delay estimates converge to the correct values.

Proof: The proof of convergence is a standard adaptation of the proof of the asynchronous Bellman-Ford iteration in [5]. Observe that the delay estimation iteration satisfies a monotonicity property, i.e., if $\hat{D_{jd}}$ and $\tilde{D_{jd}}$ are such that $\hat{D_{jd}} \geq \tilde{D_{jd}} \ \ \forall\ j \in N(i,d)$, then

$$\sum_{j \in N(i,d)} q_{id}^j (d_{ij} + \hat{D_{jd}}) \geq \sum_{j \in N(i,d)} q_{id}^j (d_{ij} + \tilde{D_{jd}}). \tag{8}$$

□

B. Distributed Delay Measurement for P-STARA

The P-STARA variant requires us to separately measure and estimate delays for packets arriving at node i with odd and even value of field. Let d_{ij}^e and d_{ij}^o be the measured (quasi-static) values of link delay corresponding to even and odd values of field in incoming packet at node i. Then, since packets with even field at node i have odd field at the next downstream node j and vice versa, the average delays at node i through neighbor j for even field $D_{id}^{j,e}[n]$, for odd field $D_{id}^{j,o}[n]$ and the corresponding average delays $\overline{D_{id}^e}$ and $\overline{D_{id}^o}$ must satisfy:

$$D^e_{id} = \sum_{j \in N(i,d)} q^{j,e}_{id}(d^e_{ij} + D^o_{jd}), \quad (9)$$

$$D^o_{id} = \sum_{j \in N(i,d)} q^{j,o}_{id}(d^o_{ij} + D^e_{jd}). \quad (10)$$

This provides a rather simple distributed technique to obtain the delay estimates given the measured link delays d^o_{ij}. On a periodic basis, node i broadcasts its average delay to a destination for even field, $\overline{D}^e_{id}[n]$, and odd field, $\overline{D}^o_{id}[n]$. On receiving the value of the average delay to d from node j for even field $\overline{D}^e_{jd}[n]$, node i updates its estimates as:

$$D^{j,o}_{id}[n] = d^o_{ij} + \overline{D}^e_{jd}[n], \quad (11)$$

$$\overline{D}^o_{id}[n] = \sum_{k \in N(i,d)} q^{k,o}_{id} D^{k,o}_{id}[n]. \quad (12)$$

On receiving the value of average delay to d from node j for odd field $\overline{D}^o_{jd}[n]$, node i updates its even estimates as:

$$D^{j,e}_{id}[n] = d^e_{ij} + \overline{D}^o_{jd}[n], \quad (13)$$

$$\overline{D}^e_{id}[n] = \sum_{k \in N(i,d)} q^{k,e}_{id} D^{k,e}_{id}[n]. \quad (14)$$

C. Immunity to Clock Offsets

We can show that the distributed delay measurement procedure detailed above retains the immunity to clock offsets of the original STARA routing algorithm. Consider a single destination d. Let f_i be the offset of node i's clock from some reference clock.

Let D_i be the equilibrium measured averages, D^j_i be the equilibrium measured averages through neighbor j and let d_{ij} be the corresponding link delays with clock offsets present. Let D^{real}_i , $D^{real,j}_i$ and d^{real}_{ij} be the corresponding quantities if clocks were synchronized throughout the network Then, $D^{real,j}_i, D^{real}_i$ and d^{real}_{ij} satisfy (6) and (7). Similarly for D^j_i, D_i and d_{ij}. Furthermore, these parameters are related as $d_{ij} = d^{real}_{ij} + f_i - f_j$.

Claim: For $j \in V(G)$, the estimates D_i and D^j_i are offset from their real values by the difference between the clock at node i and the clock at the destination node d, $f_i - f_d$. Hence, the STARA algorithm which relies on the difference between these two terms is immune to clock offsets (as before).

Proof: (Outline) By induction on the size $n(G) = |V(G)|$ of the graph containing the destination node d. We know that the link delay d_{ij} is offset from its real value by $f_i - f_j$. We combine this with the observation that node j is in $G' = G - i$ with $n(G') = k - 1$ and thus, (by the induction hypothesis) D_j differs from its real value by $f_j - f_d$. Thus, the estimate D^j_i differs from its real value by $f_i - f_d$, proving the result for nodes in a graph of size $n(G) = k$. □

VI. SIMULATION RESULTS

In this paper, we only present a few illustrative plots from an exhaustive ns-2 simulation analysis (the detailed results will be separately reported).

A. Flow "avoidance"

The primary metric we use to evaluate routing performance is the throughput-delay curve. In a broadcast medium like wireless, transmissions from neighboring nodes contend with each other for access, placing an interference bound on performance. Furthermore, a flow traversing multiple hops suffers from "self-interference", i.e., its transmissions on adjacent hops interfere with each other. It is intuitive that routing performance over wireless networks will be improved if the routing protocol generates routes that are two-hop separated from each other, i.e., flows stay away from each others' two hop neighborhoods to the extent possible.

To test if this is true, we compared the performance of a shortest-path protocol (DSDV), manual routing (preconfigured routing so that one flow routes around another) and STARA for two scenarios (see Figure 1):

1. 2lines_inter: consists of two intersecting lines, 24-39 and 28-36.

2. 2lines_A: consists of two one-hop separated parallel lines, 3-59 and 4-60.

In the 2lines_inter scenario, shortest-path routing forces the flow from 24 to 39 and the flow from 28 to 36 to use the middle of the network. The throughput-delay curve for flow 24-39 is shown in Figure 2. As expected, (shortest-path) DSDV performs poorly compared to manual routing. On the other hand, STARA routes the longer flow around the bottleneck and manages to improve performance (although it does not out-perform manual routing).

An examination of ns-2 tracefiles reveals why STARA is unable to out-perform manual routing in this scenario. As the input rate increases, STARA switches from using the shortest path from 24 to 39 to using multipaths that are two-hop separated. However, at high input rates, nodes near the source and destination are congested and the link delay measurement protocol at these nodes is not successful in obtaining accurate delay estimates. Thus, STARA ends us using multiple nodes in the neighborhood of the destination (and source), causing intra-multipath interference and preventing it from outperforming manual routing.

While attempting to equalize delay along utilized multipaths, STARA can cause flows to avoid each other. For example, in scenario 2lines_A, where we have two flows 3-59 and 4-60 parallel to each other and DSDV is forced to use these parallel adjacent shortest paths. STARA discovers and uses two-hop separated paths (except at the source and close to the destination), outperforming DSDV as can be seen in Figure 3.

A more exhaustive simulation analysis of STARA and its variants along with a description of the working Linux 2.4.20 implementation is detailed in [16].

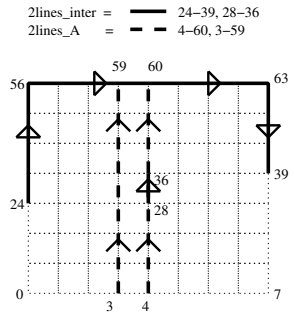

Fig. 1. Test scenarios

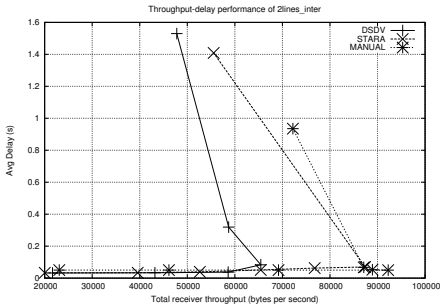

Fig. 2. Flow 24 to 39 in 2lines_inter

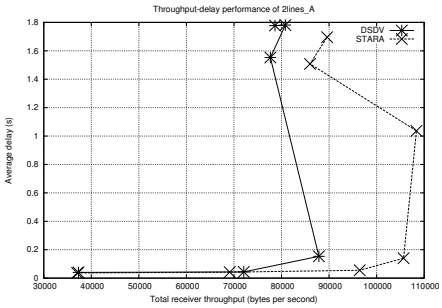

Fig. 3. 2lines_A throughput-delay

REFERENCES

[1] J. G. Wardrop, Some theoretical aspects of road traffic research, Proc. Inst. Civ. Eng. Part 2, pp325-378, 1952.

[2] Piyush Gupta and P. R. Kumar, A system and traffic independent adaptive routing algorithm for ad hoc networks, Proceedings of the IEEE CDC, pp 2375-2380, San Diego, Dec 1997.

[3] Dynamic Cesaro-Wardrop Equilibration in Networks, V. S. Borkar and P. R. Kumar, IEEE Transactions on Automatic Control, pp 382-396, vol 48, no. 3, March 2003.

[4] Piyush Gupta, Design and Performance Analysis of Wireless Networks, PhD. Thesis, University of Illinois, 2000.

[5] D. Bertsekas and R. Gallager, Data Networks, 2nd edition, Prentice Hall Inc., 1992.

[6] R. Gallager, A Minimum Delay Routing Algorithm Using Distributed Computation, IEEE Transactions on Communications, vol. 25, pp. 73-85, Jan. 1977.

[7] Srinivas Vutukury and J. J. Garcia-Luna-Aceves, A Simple Approximation to Minimum-Delay Routing, Proceedings of ACM SIGCOMM, pp. 227-238, 1999.

[8] David B. Johnson and David A. Maltz, Dynamic Source Routing in Ad Hoc Wireless Networks, Mobile Computing, Kluwer Academic Publishers, Chapter 5, pp. 153-181, 1996.

[9] Charles E. Perkins and Elizabeth M. Royer and Samir Das, Ad Hoc On Demand Distance Vector Routing, Proc. of 2nd IEEE Workshop on Mobile Computing Systems and Applications, pp 90-100, 1999.

[10] Charles E. Perkins and Pravin R. Bhagwat, Highly Dynamic Destination-Sequenced Distance Vector Routing for Mobile Computers, Proceedings of ACM SIGCOMM, pp 234-244, 1994.

[11] Alvin Valera, Winston Seah and SV Rao, Cooperative Packet Caching and Shortest Multipath Routing in Mobile Ad-hoc Networks, Proceedings of IEEE INFOCOM, pp. 260-269, 2003.

[12] Seoung-Bum Lee and Andrew T. Campbell, HMP: Hotspot Mitigation Protocol for Mobile Ad-hoc Networks, Proceedings of IWQoS, pp. 266-286, 2003.

[13] Mahesh Marina and Samir R. Das, On-demand Multipath Distance Vector Routing in Ad-hoc Networks, Proceedings of IEEE ICNP, pp. 14-23, 2001.

[14] Asis Nasipuri and Samir R. Das, On-demand Multipath Routing for Mobile Ad Hoc Networks, Proceedings of the IEEE ICCCN, Boston, MA, pp. 64-70, October 1999.

[15] Sung-Ju Lee and M. Gerla, Dynamic Load-Aware Routing in Ad-hoc Networks, Proceedings of IEEE ICC, pp. 3206-3210, June 2001.

[16] Vivek Raghunathan, Wardrop routing in wireless networks, Masters Thesis, University of Illinois, 2004.