Issues in Wardrop routing in wireless networks

Vivek Raghunathan and P. R. Kumar

Abstract— **Traditionally, traffic in wireless networks is routed along minimum hop paths from sources to destinations. This can result in hot spot formation and loss of performance. It can be theoretically shown that flow avoiding routing provides throughput gains over shortest path routing in random wireless networks by a factor of four when the sources are few enough.**

Motivated by the goal of traffic adaptive routing, we present an alternative approach to wireless network routing using delay measurements to adaptively route packets along multiple paths. The proposed protocol is completely distributed and distributes load along loop-free paths. When the protocol has equilibrated, the network achieves a Wardrop equilibrium, where all utilized paths from a source to destination have the same delay, which is less than that over any unutilized path.

We further discuss results from a ns-2 simulation study of the protocol. Our study indicates that the protocol is able to automatically route flows to "avoid" each other, consistently outperforming shortest-path protocols when the number of sources in the network is low. As the number of sources increases, the protocol's throughput-delay performance is still better than that of shortest-path routing.

We also address the architectural challenges confronted in the software implementation of such a multi-path, delay feedback based, probabilistic routing algorithm. A working implementation of the protocol has been built in userspace on a modified Linux 2.4.20 kernel. Finally, we discuss a measurement study of the implementation on a six node testbed.

Index Terms— **Wireless networks, Wardrop equilibrium, routing protocols, performance study**

I. INTRODUCTION

Traditionally, the design goal for routing in wireless networks has been correctness in the presence of mobility rather than optimal throughput performance. By and large, routing protocols for wireless networks typically route along the minimum hop path from sources to destinations ([1], [2], [3], [4]).

This paper is addressed toward static (or slowly changing networks), as for example community mesh networks ([5], [6]) or ad hoc networks in office environments. In such scenarios, the primary design goal is to optimize throughput performance. The interference constraints imposed by the wireless medium are the dominating bottleneck and minimum hop routing can result in the formation of traffic hot spots and loss of throughput.

As a network designer, we would like to re-design wireless networks for such environments to include the following (but not exhaustive) list of features:

1) Intelligent re-routing of flows "around" each other.
2) Intelligent use of multiple paths whenever beneficial.
3) Load-adaptive routing.
4) Uniform balancing of load across the network.

At the same time, we would like second generation routing solutions to retain some of the key properties of state-of-the-art wireless routing protocols: (i) loop-free paths at every instant. (ii) completely distributed operation. (iii) implementability.

In previous work on this problem [7], [8], [9], we have proposed and analyzed the STARA approach using delay estimation and load adaptive probabilistic routing. STARA [7] can be shown to converge to a Wardrop equilibrium [8], i.e., an equilibrium state in which all utilized paths from a source to a destination have the same delay, which is less than that over any unutilized path. In [9], we described modifications to STARA to make it more amenable to practical distributed implementation. These included path control using the IPv4 TTL field, distributed delay estimation and lightweight link delay measurement.

The main contributions of this paper are the following:

1) We present a multi-path routing algorithm with the following properties [7], [8], [9]:

(i) The algorithm converges to a Wardrop equilibrium with respect to the mean delay experienced by each source-destination (SD) pair. Specifically for each SD pair, the mean delays experienced along all the utilized paths is the same. The potential delay along any unutilized admissible path through the network is greater than or equal to the delay along each of the utilized paths.

(ii) Even during the transient phase while the algorithm is in the process of converging, the utilized paths are guaranteed to be loop free and have a path length no more than twice the number of hops of the minimum hop path.

2) We conduct a simulation study of the Wardrop routing algorithm to study the throughput delivered, path length, delays experienced, and the overhead consumed by the routing algorithm. The study indicates the utility of our approach, as the preliminary results in [9] indicated.
3) We propose a software architecture for implementing the multi-path, delay feedback based protocol.
4) We implement the above protocol in userspace on a modified Linux 2.4.20 kernel.
5) We present a measurement study of the protocol on a six node testbed.

The rest of the paper is organized as follows: In Section II, we recapitulate the Wardrop equilibrium, the challenges faced in developing a practical protocol, and our solutions resulting in the P-STARA protocol. In Section III, we conduct a detailed ns-2 simulation study of the protocol. In Sections IV and V, we present the implementation architecture and a measurement study of the protocol. We discuss related work in Section VI, and conclude in Section VII.

II. THE WARDROP EQUILIBRIUM AND THE M-STARA AND P-STARA ALGORITHMS

Traditionally, wireless networks have used minimum hop routing. More recently, new link metrics like ETX [10] have

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@control.csl.uiuc.edu, prkumar@uiuc.edu.`

been proposed to handle lossy links. While we do not compare our approach to these; we note in passing that they do not factor traffic patterns explicitly into their choice of routes.

In interference-constrained wireless environments, it can be shown that minimum hop routing leads to the formation of hot spots and loss of throughput performance. When the number of sources is not too large, it can be shown that it is possible to boost throughput performance over minimum hop routing by a factor of up to four by constructing a perfect flow-avoiding routing (for details, see [11]). This result shows that there are significant benefits to using traffic aware routing, and motivates the quest for a good load adaptive, multi-path routing protocol. The scheme used to prove the result is centralized and should only be taken as proof of existence.

Instead, we examine a more general approach to traffic-aware routing which aims to route toward a "Wardrop equilibrium". This equilibrium was first studied by J. G. Wardrop [12] in the context of transportation systems. In our context, we enunciate it through two principles:

1) If two or more paths from the same source to the same destination each carry positive flow, then the mean delays along all these utilized paths are equal.
2) The only reason that any path from the same source to the same destination is unutilized is because its potential mean delay is greater than or equal to the delay along every path.

In a communication network, the Wardrop equilibrium has the following attractive properties vis-a-vis multi-path routing:

1) In wireless environments, interference is the primary constraint on throughput performance. Route adaptation using delay feedback allows re-routing of flows around traffic bottlenecks.
2) When packets have to be delivered in order to the receiver application using a re-sequencing mechanism, equalizing the average delay along utilized paths reduces receiver re-sequencing delay and buffer space requirements.
3) TCP's congestion mechanism reacts adversely to re-ordered packets, and thus misbehaves when TCP is used over multiple paths. By equalizing the average delay along the used paths, this behavior can potentially be improved.

This section is a recap of work on Wardrop routing in wireless networks using the STARA approach; for details, see [7], [8], [9].

A. THE STARA ALGORITHM

The basic idea of the STARA algorithm [7] is to estimate end-to-end delay along various paths and adaptively shift traffic from higher delay paths to lower delay paths; equilibrating only when all utilized paths have the same average delay. More precisely, for each node i and destination d, let $N(i,d)$ denote the neighbors of i used to forward packets to destination d. Under the STARA algorithm, when node i receives or generates a packet destined for node d, it probabilistically sends it out to a neighbor in $N(i,d)$. Let p_{id}^{j} be the probability that it is forwarded to node $j \in N(i,d)$. Let D_{id}^{j} denote the average delay experienced by packets going from node i to destination d via the immediate neighbor j. Also, denote by $\overline{D}_{id}$ the average delay experienced by all packets from node i to destination d. STARA iteratively adjusts the routing probabilities $[p_{id}^{j} : j \in N(i,d)]$, increasing the probability p_{id}^{j} of sending a packet via node j if the delay D_{id}^{j} via j is less than the average delay $\overline{D}_{id}$ over all neighbors, and decreasing it otherwise.

STARA consists of two components: an iterative scheme for delay estimation, and an iterative scheme for updating routing probabilities [8]:

1) *Delay Estimation Scheme*: End-to-end ACKs are used to record the value of $\hat{D}_{id}^{j}[n]$, the measured delay of the packet sent at time n. Exponential forgetting is used to produce an averaged estimate of the delay:

$$D_{id}^{j}[n] = \gamma * D_{id}^{j}[n-1] + (1-\gamma)\hat{D}_{id}^{j}[n]. \qquad (1)$$

2) *Probability Update Scheme:* The routing probabilities are updated as follows:

$$\begin{aligned} p_{id}^{j}[n+1] &= [p_{id}^{j}[n] + \alpha[n] * q_{id}^{j}[n] * \\ &\quad (\overline{D}_{id}[n+1] - D_{id}^{j}[n+1])]^{+}, \text{ where} \\ \overline{D}_{id}[n+1] &= \sum_{j=1}^{N(i,d)} (D_{id}^{j}[n+1] * q_{id}^{j}[n]), \text{ and} \\ q_{id}^{j}[n] &= (1-\epsilon) * p_{id}^{j}[n] + \epsilon * 1/N(i,d). \end{aligned}$$

It is easy to see that the above algorithm works correctly even when clocks in the network are not synchronized since it only uses differences in times (for details, see [7]).

B. THE CHALLENGES

There are two problems that render difficult the use of the algorithms presented in [7], [8]:

1) Packets can follow loopy paths while the algorithm is converging. These paths can be arbitrarily long leading to possibly unbounded delays and slow convergence, since the algorithm needs feedback on all paths before it can adapt.
2) The delay measurement scheme proposed in [7] relies on ACKs to carry measurements back to sources and intermediate nodes on a per-packet basis. This poses two problems:
   a) A scheme relying on transport layer ACKs to solve the network layer routing problem violates the layering architecture, and does not extend to transport layers without end-to-end ACKs (e.g., UDP).
   b) Further, there is no guarantee that ACKs will follow the reversed path as the data packet. Thus, intermediate nodes will not be able to obtain delay information to the destination.

To address these problems and obtain a practical protocol amenable to deployment, we re-designed the algorithm in [8] to obtain two protocols M-STARA and P-STARA [9]:

1) New mechanisms to control the path lengths and guarantee loop-free routes during convergence preserve the attractive properties of load adaptation and multi-path utilization, even while retaining the property of convergence toward the appropriately defined Wardrop equilibrium.
2) A completely distributed delay measurement mechanism to replace the ACK-based scheme retains the immunity to clock offsets. It consists of a light-weight link delay measurement protocol, and a distance-vector like propagation of average delay estimates.

C. CONTROLLING PATH LENGTHS AND ELIMINATING LOOPS

The basic idea that we use is to modify the definition of the neighborhood $N(i,d)$, the set of neighbors to which node i is allowed to forward packets destined for node d.

1) Controlling paths with M-STARA: Let $S(i,d)$ denote the shortest-path distance in hops from node i to destination d. Set $N^0(i,d) := \{j \in N(i,d) : S(j,d) \leq S(i,d)\}$. We call this algorithm, which only allows the nodes in $N^0(i,d)$ to be used for forwarding packets at i destined for node d, *M-STARA*. It can be easily built on top of STARA by running a distance vector protocol to produce distances to all destinations.

2) Eliminating loops and controlling path lengths with P-STARA (Parity-STARA): The neighborhood control mechanism restricts the set of paths investigated by M-STARA, and is easy to implement, but does not eliminate routing loops or provide any guarantees on path lengths. To do so, we propose an alternative protocol called *P-STARA*. Here, the basic idea is to introduce a packet state which alternates between "odd" and "even," as a packet moves from hop to hop. When the packet state is "odd," only those neighbors which strictly decrease the distance in hops to the destination are considered for forwarding. When the packet state is "even," the neighborhood $N^0(i,d)$, defined above for M-STARA, is used. Observe that over every two hops the distance to the distance to the destination is decreased by at least one hop. This simultaneously eliminates all routing loops, as well as strictly upper bounds the path length to be no more than twice the shortest path length. In fact, it can provide loop-free routes even when the distance estimates are not accurate, or are still converging by using the sequence numbering technique of DSDV [4]. An equally important advantage (as we shall shortly see) is that this scheme admits a simple and completely distributed delay estimation algorithm matched to the definitions of the two neighborhoods used. We omit the proofs here for brevity; for details, see [9], [11].

We now describe the P-STARA algorithm. Define $N^1(i,d) := \{j \in N(i,d) : S(j,d) = S(i,d) - 1\}$, recalling the definitions of $N^0(i,d)$ and $S(j,d)$ from Section II-C.1.
1) We include a field $F(p)$ in each packet, which is initially set to an arbitrary value and decremented by one at every node along the path. In practice, we use the IPv4 TTL field as $F(p)$.
2) We define the state of the packet as $X(F(p)) = (1 + F(p)) \, mod \, 2$.
3) When the packet has state $X(F) = 0$, node i only consider neighbors in $N^0(i,d)$ as valid for routing. When the packet has state $X(F) = 1$, it considers neighbors in $N^1(i,d)$ as valid for forwarding.
4) Since $F(p)$ is decremented at each node, at successive nodes along its path the packet's state $X(F(p))$ has different values ("0" or "1").
5) Corresponding to the two values of $X(F)$, we maintain two separate probability vectors, $(p^j_{id})_F$, and delay estimate vectors $(D^j_{id})_F$, for $F = 0, 1$.
6) When a packet for destination d arrives at node i with field F, the node routes the packet to one of its neighbors $j \in N^{X(F)}(i,d)$ using the probabilities $(q^j_{id})_F$.
7) The probability update and delay estimation rules for P-STARA are given by equations (2, 3, 4) below:

$$\begin{aligned}(p^j_{id})_F[n+1] = [(p^j_{id})_F[n] + \alpha[n] * (q^j_{id})_F[n] * \\ ((\overline{D}_{id})_F[n+1] - (D^j_{id})_F[n+1])]^+, \end{aligned} \quad (2)$$

$$(\overline{D}_{id})_F[n+1] = \sum_{j=1}^{N(i,d)} ((D^j_{id})_F[n+1] * (q^j_{id})_F[n]), \quad (3)$$

$$(q^j_{id})_F[n] = (1-\epsilon) * (p^j_{id})_F[n] + \epsilon * 1/N^{X(F)}(i,d). \quad (4)$$

D. DISTRIBUTED DELAY ESTIMATION

Noting that we need to separately measure and estimate delays for packets arriving at node i depending on whether the value of the field is even or odd, let d^e_{ij} and d^o_{ij} denote the respective measured (quasi-static) values of link delay from i to j. 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 $D^{j,e}_{id}[n]$ and for odd field $D^{j,o}_{id}[n]$ must satisfy:

$$D^e_{id} = \sum_{j \in N(i,d)} q^{j,e}_{id}(d^e_{ij} + D^o_{jd}), \quad (5)$$

$$D^o_{id} = \sum_{j \in N(i,d)} q^{j,o}_{id}(d^o_{ij} + D^e_{jd}). \quad (6)$$

This provides a simple distributed technique to obtain the delay estimates given the measured link delays: 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:

$$D^{j,o}_{id}[n] = d^o_{ij} + \overline{D}^e_{jd}[n], \quad (7)$$

$$\overline{D}^o_{id}[n] = \sum_{k \in N(i,d)} q^{k,o}_{id} D^{k,o}_{id}[n], \quad (8)$$

and similarly for the odd field.

It can be shown that the distributed delay measurement procedure described above retains the immunity to clock offsets of the original STARA routing algorithm [9].

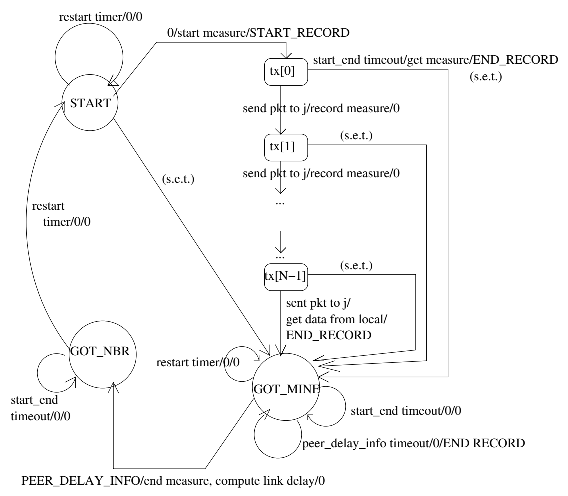
E. LINK DELAY MEASUREMENT

In practice, we need a link delay measurement protocol to measure and average the link delays d_{ij}'s out in real time. Such a link delay measurement protocol operates more frequently than other components of the routing protocol (e.g., average delay estimation), and must have as little communication overhead as possible. We use a lightweight protocol to measure the link delay d_{ij} from node i to node j.

The state machines for the protocol (without transitions to error states) implementing the above scheme are shown in Figures 1 and 2. We reduce the overhead of the protocol as follows:
1) The average delay is computed as the difference between the average receive time and the average send time (instead of transmitting the individual receiver timestamps back to the sender).
2) The protocol operates correctly even without reliable delivery of every message. For example, START_RECORD and PEER_DELAY messages are not re-transmitted. (However, the receiver must receive an END_RECORD message from the sender for correct operation of the state machine.)

III. SIMULATION STUDY

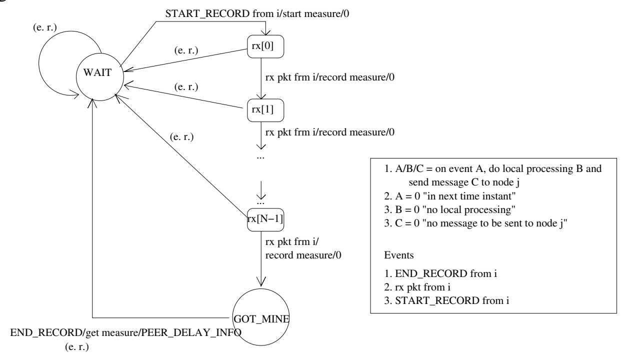
In order to understand the protocol behavior, we initially implemented it in the ns-2 simulator and carried out an extensive simulation analysis. In all our simulations, we use a two ray ground propagation model, IEEE 802.11 DCF MAC, and a radio model based on the Lucent WaveLAN 2 Mbps radio. All simulations were carried out with N^2 nodes on the vertices of a $N \times N$ grid. The interface queue length is set to 50 packets, and the retry parameters of the MAC are $MAC\_ShortRetryLimit = 7$, and


restart timer/0/0
0/start measure/START_RECORD
START
tx[0]
start_end timeout/get measure/END_RECORD
(s.e.t.)
send pkt to j/record measure/0
tx[1]
(s.e.t.)
send pkt to j/record measure/0
(s.e.t.)
...
restart
timer/0/0
...
tx[N−1]
(s.e.t.)
sent pkt to j/
get data from local/
END_RECORD
GOT_NBR
restart timer/0/0
GOT_MINE
start_end
timeout/0/0
start_end timeout/0/0
peer_delay_info timeout/0/END RECORD
PEER_DELAY_INFO/end measure, compute link delay/0

1. A/B/C = on event A, do local processing B and send message C to node j
2. A = 0 "in next time instant"
3. B = 0 "no local processing"
3. C = 0 "no message to be sent to node j"

Events
1. restart timer: determines upper bound on overhead
2. start_end_timer: determines lower bound on overhead
3. peer_delay_info timeout: ensures reliable receipt of END_RECORD
4. PEER_DELAY_INFO from j
5. sent pkt to j

Fig. 1. Sender side state machine


START_RECORD from i/start measure/0
(e. r.)
WAIT
(e. r.)
rx[0]
(e. r.)
rx pkt frm i/record measure/0
rx[1]
(e. r.)
rx pkt frm i/record measure/0
...
...
rx[N−1]
rx pkt frm i/
record measure/0
GOT_MINE
END_RECORD/get measure/PEER_DELAY_INFO
(e. r.)

1. A/B/C = on event A, do local processing B and send message C to node j
2. A = 0 "in next time instant"
3. B = 0 "no local processing"
3. C = 0 "no message to be sent to node j"

Events
1. END_RECORD from i
2. rx pkt from i
3. START_RECORD from i

Fig. 2. Receiver side state machine

$MAC\_LongRetryLimit = 4$. We use CBR sources with a constant packet size of 210 bytes running on top of UDP to stress the network. The simulations are run for times between 100 and 3000 seconds. The parameters for exponential forgetting of average delay measurement and link delay measurement are set to 0.8. The probability of uniform routing to all neighbors, ϵ, is set to 0.05.

We use the following metrics to evaluate performance:
1) Throughput-delay performance.
2) Path length and quality.
3) Routing overhead.

The throughput-delay performance of a routing protocol is the most reliable metric in determining its performance in real networks. (While auxiliary metrics like routing overhead or path length may provide an indicator of how good the performance is, these metrics are already factored into the throughput-delay curve, while the reverse is not true.) A convergence speed analysis of STARA is omitted here for brevity; see [11] for details.

A. Throughput-delay performance

We compare the performance of STARA to that of the shortest-path DSDV protocol. (The ns-2 AODV implementation was apparently buggy at high network load and we exclude it from our performance comparison). In all throughput-delay simulations, ADP is set to 15.0, LDP to 5.0 and $LDPF$ to 25.0. (These parameters and the choices of values are explained later in this section.) In examining the throughput-delay performance of STARA, we make the following observations:

No. of connections	% of scenarios with improvement	Avg. % increase in sat. thruput
2	75	13.94
3	87	23.47
4	62	14.33
7	100	23.16

TABLE I
STARA IMPROVEMENT OVER DSDV

1) *STARA does result in flows "avoiding" each other in environments with "few sources."*

To test whether the routes generated by STARA in the few sources regime indeed have the flow avoidance property, we compared it against the performance of the shortest-path DSDV routing protocol, and "Manual" perfect flow-avoiding routing. The scenarios and the flow-avoiding paths followed by STARA are shown in Figure 5. These two scenarios were described in [9] and are included here for continuity.

In the 2lines_inter scenario (Figure 5(a)), the short flow in the middle hogs the wireless medium in the center of the grid and DSDV obtains poor performance, as can be seen from the throughput-delay curve (Figure 3). STARA routes the longer flow around the bottleneck, and manages to achieve reasonable performance.

Another interesting scenario is Scenario 2lines_A (Figure 5(b)), where we have two flows 3-59 and 4-60 parallel to each other, but not intersecting. DSDV is forced to use adjacent parallel 1-hop separated paths that interfere with each other. STARA however uses these paths at low rates, but at higher network loading, when these paths become the bottleneck, STARA switches to paths that are two-hop separated except at the source and close to the destination. The performance of STARA is clearly better than that of DSDV (see Figure 4).

2) *If optimal avoidance is not possible, STARA does equilibrate to paths that avoid each other as far as possible*. Sometimes, it may not be possible to route all flows to avoid each other. In this case, a good multipath routing protocol should produce routes that minimize the interference due to nodes within two hops of each other. In simulations, we observe that the STARA protocol does generate routes with this property in the majority of scenarios, e.g., in Figures 6 and 7 which show the routes for scenarios with three and four sources respectively. The path shown for each source-destination pair is the primary path used with high probability.

3) *STARA improves throughput-delay performance even when the "few sources" assumption is violated*. STARA is able to obtain considerable throughput performance gains by minimizing interference even with more source-destination pairs in the network, e.g., Figure 8 showing a scenario with three flows, each of length five, and Figure 9, for a scenario with seven flows, each of length four. This is because STARA produces routes that minimize interference effects.

4) *STARA's throughput-delay performance consistently provides improvements over DSDV*. To verify the effectiveness of STARA's delay-adaptive routing in wireless environments, we tested a large number of randomly chosen scenarios. With the number of connections fixed, 24 random scenarios of

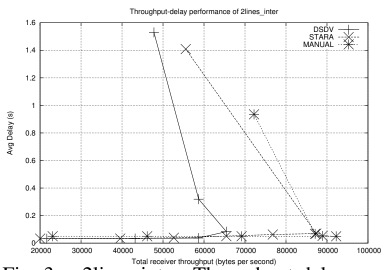


Fig. 3. 2lines_inter: Throughput-delay performance

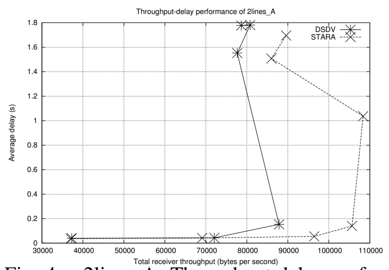


Fig. 4. 2lines_A: Throughput-delay performance

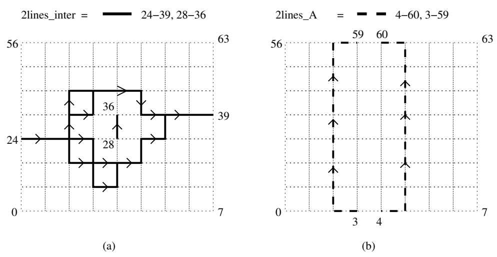


Fig. 5. Few sources regime: STARA generates flow-avoiding multipaths

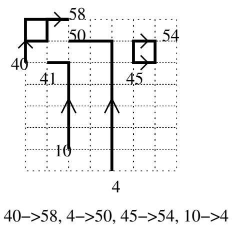


Fig. 6. 4random Scenario 15: STARA multipaths

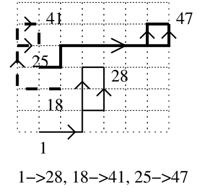


Fig. 7. 3random Scenario 18: STARA multipaths

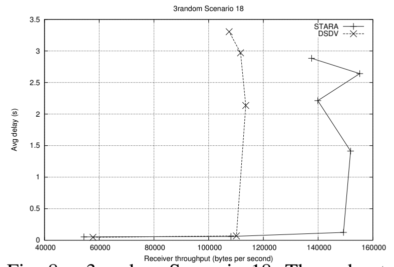


Fig. 8. 3random Scenario 18: Throughput-delay performance

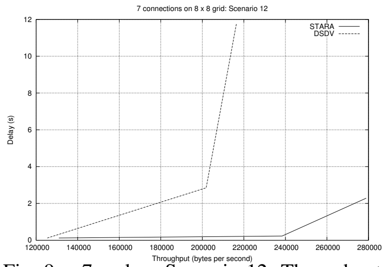


Fig. 9. 7random Scenario 12: Throughput-delay performance

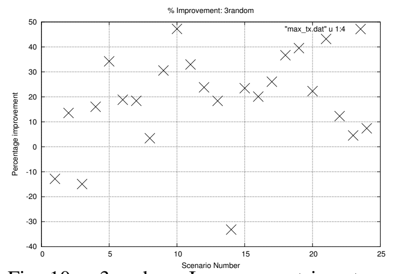


Fig. 10. 3random: Improvement in saturation throughput for several scenarios

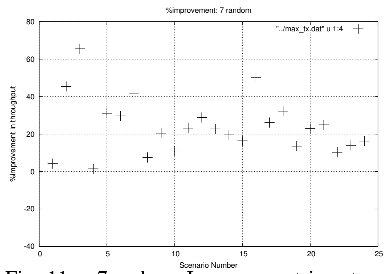


Fig. 11. 7random: Improvement in saturation throughput for several scenarios

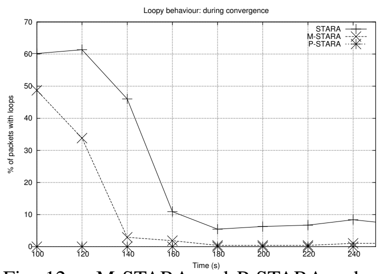


Fig. 12. M-STARA and P-STARA reduce loops during convergence

varying connection lengths were generated on a 8×8 grid, and the throughput-delay performance measured. For a majority of scenarios considered, the saturation throughput of DSDV, i.e., the maximum throughput that DSDV can support, was a lower-bound on that of STARA. The results are summarized in Table I, where the percentage improvement and the number of scenarios in which STARA outperformed DSDV are shown. The percentage improvements for the various scenarios are shown in Figures 10 and 11.

B. Path length and quality

It is important to ensure that the paths followed by packets at every instant are loop-free, or failing that, that packets do not follow wild paths to the destination. To check this, we ran STARA on an 8×8 grid with direct connections between diagonally adjacent grid nodes (in the absence of these, all the variants implemented will always use L^1-optimal, i.e., Manhattan metric, loop-free paths). We simulated ten scenarios representing varying degrees of network congestion and traffic burstiness. The scenarios are labeled as rk_n, where k represents the number of connections passing through the middle of the grid, and the n represents the input rate for each connection (in Kb). $r3$_burst is $r3\_60$ with connections randomly switching on and off to simulate burstiness.

M-STARA drastically reduces the percentages of packets following loops during convergence (see Figure 12 for scenario $r2\_20$), while P-STARA completely avoids loops. A performance comparison of average path length and packet loops is shown in Table II.

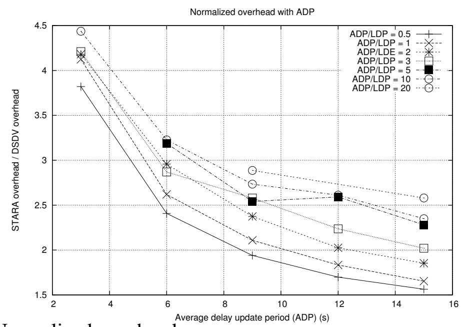


Fig. 13. Normalized overhead

TABLE II
M-STARA VERSUS STARA: PATH LENGTH AND QUALITY

Scenario	Avg. path length (STARA)	% of loopy pkts (STARA)	% decrease in path len (M-STARA)	% decrease in loops (M-STARA)
$r1\_20$	9.31	13.39	17.9	53.02
$r1\_40$	8.96	11.00	6.3	39.72
$r1\_60$	9.66	10.25	17.2	42.04
$r2\_20$	8.54	10.39	11.8	70.64
$r2\_40$	9.31	9.80	19.9	69.03
$r2\_60$	8.75	9.11	14.8	68.27
$r3\_20$	9.24	10.47	16.1	62.17
$r3\_40$	9.87	11.02	11.4	45.09
$r3\_60$	10.71	8.41	7.2	33.53
$r3$_burst	10.36	7.97	-4.1	44.54

C. Routing overhead

The routing protocol overhead for STARA has the following components (arranged in order of timescale): (i) Link delay measurement overhead (LDE) (fastest) (ii) Average delay measurement overhead (ADE) (iii) DSDV overhead (slowest).

The first and third component grow like N^2, while LDE overhead grows like the number of edges. The following protocol parameters provide us an additional degree of control over the overhead; their choice reflects a trade-off between convergence speed and routing overhead:

1) *Average delay update period (ADP)*: The rate at which nodes broadcast average delay information to their neighbors.
2) *Link delay measurement re-start timer (LDP)*: The frequency at which STARA attempts to re-start the link delay measurement protocol is once every LDP seconds (see Figure 1). The value of $1/LDP$ places an upper bound on LDE.
3) *Link delay measurement forced re-start timer ($LDPF$)*: STARA forcibly re-starts the link delay measurement protocol once every $LDPF$ seconds to ensure that link delay information is always obtained. The value of $1/LDPF$ places a lower bound on LDE.
4) *DSDV periodic update period*: set constant to $15s$.

The ADE and DSDV components of the overhead are independent of traffic load, while LDE overhead follows an on-demand pattern, i.e., overhead for nodes not along the active forwarding path is minimal (equal to the lower bound decided by $1/LDPF$), while the overhead for nodes along the active forwarding path is higher. For details on this behavior of the LDE overhead, see [11].

A normalized plot of the routing overhead with respect to the DSDV overhead in a 10×10 grid is shown in Figure 13 for various values of ADP and ADP/LDP. Observe that STARA overhead (with DSDV overhead included) is roughly twice the DSDV overhead (for an appropriate choice of parameters) As noted earlier, the routing overhead is an auxiliary metric and the increased goodput of STARA in spite of the increased routing overhead confirms this observation.

D. Parameter choice

As noted in [11], the optimal choice of parameters $ADP, LDP, LDPF$ represents a trade-off between convergence speed and routing overhead. The effect of increasing ADP for a fixed ADP/LDP, or decreasing ADP/LDP for a fixed ADP is to reduce the routing overhead (see Figure 13). The convergence time depends largely on ADP, provided that a few link delay measurements are obtained in every ADP time period, i.e, $ADP/LDP > 1$ (see [11]). A good heuristic to pick ADP and LDP is to set $ADP/LDP = 2$ or 3, and then select ADP for satisfactory convergence time.

IV. SOFTWARE ARCHITECTURE

When designing the STARA protocol, one of the most important design considerations was that the protocol be elegantly implementable in modern operating systems. We have implemented STARA based on the three traditional principles of IP routing implementations:

1) *Routing policy* is separated from *forwarding mechanism.*
2) Components are placed in-kernel only if the userspace performance hit is unacceptable.
3) Interfaces are generic, well-defined and cleanly extensible.

The key architectural choice we made in implementing STARA was to separate probabilistic packet forwarding from delay estimation. Probabilistic forwarding on a per-packet basis is thought of as simply a new mechanism provided by the kernel to forward packets. The probabilistic forwarding mechanism provided consists of multiple in-kernel forwarding tables. Each forwarding table consists of a list of routes to destination. Each route to a destination consists of a vector of two-tuples (next hop, probability of usage). A clean interface is provided to configure the probabilistic forwarding mechanism via a userspace library. (M-STARA and P-STARA are thus simply different probabilistic routing policies, and are implemented completely in user-space.)

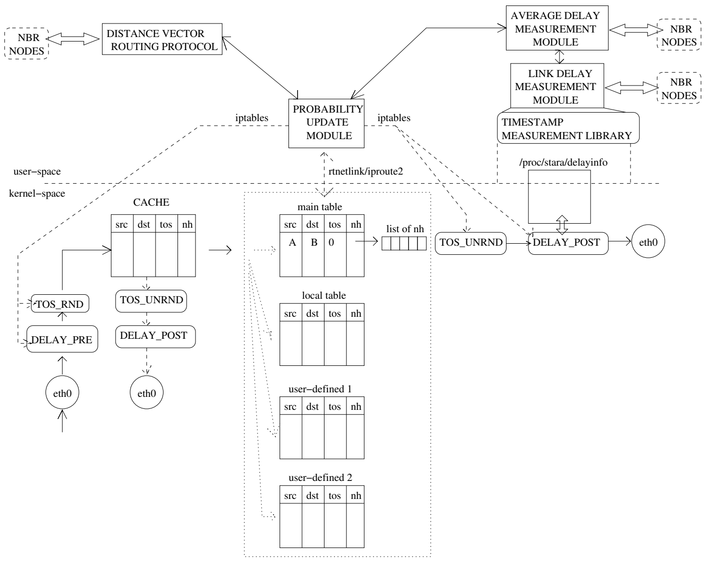


Fig. 14. Implementation Architecture

The architecture of the implementation is shown in Figure 14. We have used all the in-built features of Linux kernel routing to implement the STARA protocol almost completely in user-space. These include netfilter, equal cost multipath routing, policy routing, iptables and MARK target routing. The userspace code implements protocol primitives such as link delay measurement, average delay measurement, and probability update policy. To obtain distance estimates, we use a DSDV implementation [13]. The protocol needs some in-kernel support that was unavailable in the Linux 2.4.20 kernel:

1) *Timestamp measurement*:
This module timestamps and records incoming and outgoing MAC addresses as packets pass through the kernel and exports the collected information to userspace using $/proc$.
2) *Per-packet multipath routing*:
The Linux kernel implementation of multipath only allows next hop selection on a per-connection basis. This is because the kernel caches routes on a per-connection basis, and packets end up using the cached route to a destination through the first (randomly chosen) next hop to that destination.

Instead of disabling the route cache (and incurring a performance hit) (see equalize patch [14]), we trick the route cache into simulating per packet multipath routing by randomizing one of the keys used for the cache lookup. This ensures that the distribution of next hops to a destination in the cache is the same as that in the routing table (up to quantization error).

V. MEASUREMENT STUDY

We have carried out a measurement study of the STARA implementation on a six node testbed to verify the practical effectiveness of STARA route adaptation.

A. EXPERIMENTAL SETUP

The nodes in our testbed consist of Compaq Presario 2800T, Compaq Presario 1800T and HP Pavilion zv5240 laptops with Cisco Aironet 340 and 350 Series PCMCIA cards. All nodes run our custom-patched Linux 2.4.20 kernel and Red Hat Linux 9.0. UDP traffic is generated using *nttcp*. We used experimentation experience gained in-house during an earlier study of TCP [15] to resolve commonly encountered problems:
1) *MAC filtering for topology creation.* Topologies are generated by collocating all nodes and using iptables MAC filtering at individual nodes to create a virtual multi-hop scenario. This approach is useful in verifying implementation correctness.
2) *Copper tape for multi-hop topology creation.* The Cisco Aironet 350 Series cards have very high receive sensitivities and transmit range. In order to decrease the effective transmit range, we wrap the cards with 3M 1181 EMI copper shielding tape. This lets us create stable topologies in a single hall because the copper tape "changes the antenna impedance and causes an impedance mismatch between the card and the antenna, resulting in less power delivered to the air" [15]. This approach lets us observe STARA route adaptation to traffic conditions and obtain performance data.
3) *Heating effects.* The copper tape causes cards to heat up excessively, and degrade in throughput performance after a while. To handle this, we randomize over which protocol is run first, and abandon data from bad runs.

B. PERFORMANCE DATA

The copper tape attenuation approach is used to verify the effectiveness of STARA route adaptation. The performance results in this section should be treated as a proof-of-concept; further large scale testing is still needed. The topologies used are shown in Figure 15. The throughput performance of DSDV and STARA for the topologies is summarized in Table III and Table IV. Each data point is an average over ten runs of nttcp, and in each run, a UDP connection transfers 8 MB of data from source s to destination d.

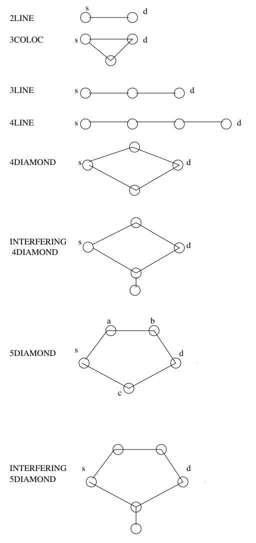


Fig. 15. Experimental topologies

Both protocols see an (expected) drop in throughput performance as the number of hops along a path increases (see Table III) due to the increased self-interference. Even if we attribute all the performance difference in these simple scenarios to the extra routing overhead, (which it clearly is not, since STARA actually out-performs DSDV in the 3LINE scenario; in practice, heating effects and in-building wireless interference also play a role), the worst-case impact of extra routing overhead on performance is approximately 12.3%.

TABLE III
THROUGHPUT PERFORMANCE FOR BASIC TOPOLOGIES

Topology	DSDV (Kbps)	STARA (Kbps)
2LINE	2.698	2.367
3COLOC	3.210	3.169
3LINE	2.011	2.347
4LINE	1.271	0.993

However, as the throughput performance of the DIAMOND scenarios shows, there is a significant benefit to using STARA over DSDV when there are alternative paths available in the network, and some of these paths are interference-bound. In the INTERFERING DIAMOND scenarios, an extra node interferes at a target node along one of the paths by issuing broadcast ping (ICMP ECHO REQUEST) packets at a very low power level (1mW). We turn off broadcast ICMP ECHO REPLY at the target node. This setup ensures that the target node does not send any IEEE802.11 frames (no RTS/CTS handshake is performed, no ACKs are sent and the ICMP ECHO REQUEST is dropped at the target node); thus, only the target node is affected by the interference.

For example, in the 4DIAMOND scenario, DSDV has two equivalent paths of length two between s and d for routing and can re-route around the bad path in the presence of interference. In practice, HELLO packets arrive along the interference-bound path too, and DSDV keeps oscillating between the two paths. On the other hand, STARA is able to completely route around the interference and achieves better performance. (Observe that STARA also loses throughput performance in INTERFERING 4DIAMOND. This is a head-of-line blocking artifact of our experimental setup).

The 5DIAMOND scenario is designed to accentuate the bad performance of DSDV since DSDV is designed to find shortest paths and would be forced to use the shortest 2-hop path even in the presence of interference. As a result, DSDV is unable to use the longer 3-hop path and suffers a performance hit. On the other hand, STARA is able to route around the interference and use the longer 3-hop path for routing.

TABLE IV
STARA ROUTE ADAPTATION LEADS TO BETTER THROUGHPUT

Topology	DSDV (Kbps)	STARA (Kbps)
4DIAMOND	2.134	2.231
INTERFERING 4DIAMOND	0.880	1.322
5DIAMOND	1.902	1.477
INTERFERING 5DIAMOND	0.475	1.326

Finally, we test whether P-STARA is able to eliminate loops in the network by generating traffic from s to d in the 5DIAMOND topology. With our implementation, the only loops possible in this topology are of the form $s - a - s$. The Linux TCP/IP implementation is designed to drop packets arriving at a node whose source address matches that of the node (as a precaution against address spoofing) and will thus drop all packets that use the path $s - a - s$. Thus, the drop rate of ping traffic from s to d gives us the percentage of packets following loops. As can be seen in Table V, P-STARA provides loop-freedom at every instant.

TABLE V
P-STARA IS LOOP-FREE

Protocol	ping drop rate
STARA	4.759
P-STARA	0.000

VI. Related Work

There is a large body of work on the problem of finding loop-free routes to destinations in mobile environments with minimal overhead [1], [2], [3], [4]. Pro-active routing protocols ([1], [4]) maintain routes to all destinations, thereby incurring a larger routing overhead. Reactive routing protocols, on the other hand, construct routes on an on-demand basis ([2], [3]) using a route discovery procedure.

Multipath extensions of existing on-demand protocols can provide faster recovery from route failures. For example, CHAMP [16], built on top of DSR, uses cooperative packet caching and shortest multipath routing to reduce packet loss due to frequent link breakdowns. AOMDV [17], built on top of AODV, computes multiple loop-free link-disjoint paths to a destination by using an advertised hop-count. An extension to DSR is proposed in [18] which provides sources and intermediate nodes with alternate paths during ROUTE DISCOVERY. A technique to allow AODV to install backup routes at the neighbor nodes of a primary route is investigated in [19].

Multi-path routing can also alleviate congestion in wireless ad hoc networks. The hotspot mitigation protocol (HMP) [20] can be integrated with AODV and DSR, and attempts to disperse new flows from being routed through hot spots and congestion-prone areas. The Dynamic Load Aware Routing protocol (DLAR) [21] carries congestion information forward in ROUTE REQUEST packets, allowing destinations to choose the least congested path.

On the other hand, there is a substantial body of literature on optimal dynamic routing and its formulation as the solution of an optimization problem ([22], [23]). The problem can be equivalently cast in terms of routing variables at each node, and a completely distributed algorithm can be derived [23]. An approximation approach to minimum delay routing [24] provides a set of loop-free invariant conditions, while distributing traffic to approximate [23].

Mesh networks are being built and deployed in communities across the United States [5], [6]. The ETX metric [10] has been used as a link metric for routing in these networks. As discussed earlier, ETX is useful for finding routes in environments with lossy links; our work focuses on the problem of finding traffic-aware routes in interference-constrained environments. Finally, interference-aware routing has been studied in [25], which uses a conflict graph to model interference, and computes bounds on the optimal throughput obtained under the given wireless configuration and traffic matrix. While it reaches a similar conclusion regarding the limitation of the hop count metric, it does not present a distributed algorithm that produces interference-avoiding routes.

VII. Conclusion

In this paper, we have described a multi-path load adaptive routing protocol that is completely distributed and loop-free. Path control mechanisms like the IPv4 TTL field and a completely distributed delay estimation procedure make the protocol highly practical and elegantly implementable.

Simulations indicate that the protocol appears to be effective in obtaining improved throughput-delay performance gains over shortest-path routing in static wireless networks. An architecture that separates probabilistic multipath routing from delay estimation has resulted in a clean implementation of the protocol in userspace on a modified Linux 2.4.20 kernel. A proof-of-concept measurement study indicates the effectiveness of STARA route adaptation.

Our work on STARA at the theoretical, simulation, implementation and testing levels shows that there is scope for traffic adaptive routing which can outperform minimum hop routing. Second generation protocols for wireless networks may well benefit from such adaptive routing. Clearly, a lot needs to be done to translate this initial feasibility study into practice. This is an enticing challenge for the systems community.

References

[1] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, and P. Muhlethaler, “Optimized link state routing protocol (OLSR),” 2003, RFC 3626, http://www.ietf.org/rfc/rfc3626.txt.

[2] D. B. Johnson and D. A. Maltz, “Dynamic source routing in ad hoc wireless networks,” *Mobile Computing*, pp. 153–181, 1996.

[3] C. E. Perkins, E. M. Royer, and S. Das, “Ad hoc on demand distance vector routing,” in *Proc. of 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999, pp. 90–100.

[4] C. E. Perkins and P. R. Bhagwat, “Highly dynamic destination-sequenced distance vector routing for mobile computers,” in *Proceedings of ACM SIGCOMM*, 1994, pp. 234–244.

[5] “Champaign-Urbana Community Wireless Network,” http://www.cuwireless.net.

[6] “Seattle Wireless,” http://www.seattlewireless.net.

[7] P. Gupta and P. R. Kumar, “A system and traffic independent adaptive routing algorithm for ad hoc networks,” in *Proceedings of the IEEE CDC*, 1997, pp. 2375–2380.

[8] V. S. Borkar and P. R. Kumar, “Dynamic Cesaro-Wardrop equilibration in networks,” *IEEE Transactions on Automatic Control*, vol. 48, pp. 382–396, 2003.

[9] V. Raghunathan and P. R. Kumar, “On delay adaptive routing in wireless networks,” in *Proceedings of the IEEE CDC*, 2004.

[10] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris, “A high-throughput path metric for multi-hop wireless routing,” in *Proceedings of ACM MOBICOM*, 2003.

[11] V. Raghunathan, “Wardrop routing in wireless networks,” Master’s thesis, University of Illinois at Urbana-Champaign, 2004.

[12] J. G. Wardrop, “Some theoretical aspects of road traffic research,” *Proc. Inst. Civ. Eng.*, vol. 2, pp. 325–378, 1952.

[13] B. Gupta, “Design, implementation and testing of routing protocol for mobile ad-hoc networks,” Master’s thesis, University of Illinois at Urbana-Champaign, 2002.

[14] P. McHardy, “Linux 2.4.18 equalize patch,” http://trash.net/ kaber/equalize/.

[15] V. Kawadia, “Protocols and architecture for wireless ad hoc networks,” Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2004.

[16] W. S. Alvin Valera and S. Rao, “Cooperative packet caching and shortest multipath routing in mobile ad-hoc networks,” in *Proceedings of IEEE INFOCOM*, 2003, pp. 260–269.

[17] M. Marina and S. R. Das, “On-demand multipath distance vector routing in ad-hoc networks,” in *Proceedings of IEEE International Conference on Network Protocols*, 2001, pp. 14–23.

[18] A. Nasipuri and S. R. Das, “On-demand multipath routing for mobile ad hoc networks,” in *Proceedings of the 8th Annual IEEE International Conference on Computer Comminications and Networks (ICCN)*, 1999, pp. 64–70.

[19] S. J. Lee and M. Gerla, “AODV-BR: Backup routing in ad hoc networks,” in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, 2003, pp. 266–286.

[20] S.-B. Lee and A. T. Campbell, “HMP: Hotspot mitigation protocol for mobile ad-hoc networks,” in *Proceedings of the IwQOS*, 2003, pp. 266–286.

[21] S.-J. Lee and M. Gerla, “Dynamic load-aware routing in ad-hoc networks,” in *Proceedings of the 3rd IEEE International Conference on Communications*, 2001, pp. 3206–3210.

[22] D. Bertsekas and R. G. Gallager, *Data Networks*. New Jersey: Prentice Hall Inc., 1992.

[23] R. Gallager, “A minimum delay routing algorithm using distributed computation,” *IEEE Transactions on Communications*, vol. 25, pp. 73–85, 1977.

[24] S. Vutukury and J. J. Garcia-Luna-Aceves, “A simple approximation to minimum-delay routing,” in *Proceedings of ACM SIGCOMM*, 1999, pp. 227–238.

[25] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu, “Impact of interference on multi-hop wireless network performance,” in *Proceedings of ACM MOBICOM*, 2003.