Wardrop routing in wireless networks

Vivek Raghunathan and P. R. Kumar

Abstract— **Routing protocols for multi-hop wireless networks have traditionally used shortest-path routing to obtain paths to destinations, and do not consider traffic load or delay as an explicit factor in the choice of routes. We focus on static mesh networks, and formally establish that if the number of sources is not too large, then it is possible to construct a perfect flow-avoiding routing, which can boost the throughput provided to each user over that of shortest-path routing by a factor of four when carrier sensing can be disabled, or a factor of 3.2 otherwise.**

So motivated, we address the issue of designing a multi-path, load adaptive routing protocol that is generally applicable even when there are more sources. We develop a protocol that adaptively equalizes the mean delay along all utilized routes from a source to destination, and does not utilize any routes that have greater mean delay. This is the property satisfied by a system in Wardrop equilibrium. We also address the architectural challenges confronted in the software implementation of a multi-path, delay feedback based, probabilistic routing algorithm. Our routing protocol is (i) completely distributed, (ii) automatically load balances flows, (iii) uses multiple paths whenever beneficial, (iv) guarantees loop-free paths at every time instant even while the algorithm is still converging, and (v) amenable to clean implementation.

A ns-2 simulation study indicates that the protocol is able to automatically route flows to "avoid" each other, consistently out-performing shortest-path protocols in a variety of scenarios. The protocol has been implemented in userspace with a small amount of forwarding mechanism in a modified Linux 2.4.20 kernel. Finally, we discuss a proof-of-concept measurement study of the implementation on a six node testbed.

Index Terms— **Wireless networks, Wardrop routing, delay-adaptive multipath routing, performance study, implementation.**

I. Introduction

Routing research in wireless multi-hop networks has traditionally focused on shortest path routing protocols. A lot of effort has gone into designing protocols that route packets efficiently in mobile networks with minimal overhead, e.g., [1], [2], [3], [4], [5].

This paper is addressed towards static or at best networks with slow mobility, as for example, mesh networks or adhoc networks in office environments where users do not move around with their laptops. Our focus is on wireless networks, e.g., mesh networks, where the focus is on boosting the throughput-delay performance, and energy limitations are not a constraining factor. Looking ahead to the next generation of wireless routing protocols for such quasi-static networks, users may want their routing to be adaptive to the load on the network. For example, users may want to improve their throughput-delay performance by intelligently routing flows in a manner to avoid interference from other paths as far as possible. Users may also want to utilize multiple paths so that they obtain more throughput. In addition, looked at from a network standpoint, one may want the routing algorithms implemented at each node to combine together to balance load across the network. At the same time, one would like to retain some of the key features of current protocols, including the completely distributed operation, loop-free paths and ease of implementability.

In this paper, we present a completely distributed, load adaptive, multi-path routing protocol for quasi-static wireless mesh networks. For every source-destination pair, the protocol adaptively equalizes mean delays along all utilized routes, and avoids using any paths with greater or equal mean delay. This is the property satisfied by a system in Wardrop equilibrium. Such an equilibrium is potentially useful in practice for a variety of reasons:

1) Adaptive delay-based routing can automatically route around hotspots in interference-constrained wireless networks.
2) Equalizing the average delay along used paths can reduce re-sequencing delays for packets in receiver socket buffers.
3) TCP congestion control reacts adversely to re-ordered packets, and thus misbehaves when TCP is used over multiple paths. Equalizing the average delay along used routes can reduce packet re-ordering and potentially improve TCP behavior when it is used on top of multi-path routing.

The work in this paper builds on earlier work on design [6] and convergence [7] of delay-adaptive routing algorithms in wireless networks. The goal of this paper is to bridge the gap between the theory of delay-adaptive routing and its implementation and use in practice as a routing protocol for 802.11-based wireless mesh networks. The issues addressed range from theoretical characterization and algorithmic properties, to a detailed simulation study and architectural challenges in implementing multi-path delay-adaptive routing protocols. This has resulted in our userspace implementation on a custom-modified Linux 2.4.20-6 kernel, and a measurement study of the implementation on a testbed. This paper is an extended version of the work presented in [8], [9].

The main results of this paper are as follows:

1) We prove a new result that formalizes the potential performance improvement for interference avoiding multipath routing in wireless networks. We show that when the number of active source-destination (SD) pairs is appropriately small in comparison with the number of nodes in the network, and the SD pairs are randomly distributed, then it is possible to choose two paths per each SD pair such that no two paths from different SD pairs cross or interfere with each other, and that the two paths for the same SD pair meet only at their end-points. The throughput benefit

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. Vivek is now with Google Inc. Email: {vivek, prkumar}@control.csl.uiuc.edu

This material is based upon work partially supported by NSF under Contract Nos. ECCS 07-01604, CNS 05-19535 and CCR-0325716, AFOSR under Contract No. F49620- 02-1-0217, DARPA/AFOSR under Contract No. F49620-02-1-0325, and Oakridge-Battelle under Contract 239-DOE BATT 4000044522. Vivek Raghunathan is also supported by a Vodafone graduate fellowship. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the above agencies.

realizable by such path disjoint multi-path routing depends on whether carrier sensing (with twice the transmission range) can or cannot be turned off. The throughput that can be furnished to each SD pair is four times what would be furnished by minimum hop shortest-path routing if carrier sensing can be disabled, or 3.2 times otherwise.

2) The algorithm used to construct the flow-avoiding routing only illustrates the feasibility of such an approach and is not amenable to distributed implementation in a wireless network. Further, in practice, we would like our routing protocol to be load adaptive even when there are a large number of flows in the network which necessarily cross each other. So motivated, we design a multi-path routing protocol with the following properties:

a) The routing algorithm converges to a set of admissible routes for each SD pair with the following properties: for each SD pair, the mean delays experienced along the multiple paths are all 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. This is the property satisfied by a system in Wardrop equilibrium.

b) The admissible paths all have no more than twice the number of hops of the minimum hop path.

c) Even during the transient phase while the algorithm is in the process of converging, the utilized paths are guaranteed to be loop free.

3) We demonstrate, via a theoretical example, how our delay adaptive routing can help automatically achieve flow avoiding routing whenever possible. We conduct a detailed ns-2 simulation study of the algorithm to study the throughput delivered, the number of hops along the paths, the delays experienced, the rate at which the algorithm converges, and the overhead consumed by the routing algorithm. The simulation study indicates that Wardrop routing is effective in routing flows to "avoid each other" and minimize interference.

4) We propose a software architecture for implementing the multi-path, delay feedback based, Wardrop equilibrating protocol. The architecture respects the IP stack layering. Even though it is adaptive to the end-to-end delay, it does not utilize transport layer mechanisms to implement network layer routing.

5) We implement the above protocol in the Linux 2.4.20-6 kernel. The routing policy is implemented completely in userspace, with a small amount of forwarding mechanism in the kernel itself.

6) We conduct a proof-of-concept measurement study of the protocol on a six node testbed.

The rest of this paper is organized as follows: in Section II, we survey related work. In Section III, we demonstrate, by theoretical proof and simulation, the regime in which shortest path routing can lead to a loss of throughput performance in large wireless networks. In Section IV, we introduce the traffic-adaptive routing algorithm, the challenges to be faced in developing a practical protocol, and our solutions resulting in the P-STARA protocol. In Section V, we describe the protocols for distributed delay estimation and link delay measurement. In Section VI, we present the results of a detailed simulation study. In Section VII, we present the implementation architecture, and in Section VIII, we carry out a measurement study of the protocol. Finally, we conclude in Section IX.

II. Related Work

The problem of finding loop-free routes to destinations in mobile environments has been well studied [1], [2], [3], [4], [5], [10], [11]. Pro-active protocols ([1], [4], [10]) maintain routes to all destinations, and incur a large routing overhead. Reactive 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 recovery from route failures. For example, CHAMP [12], built on top of DSR, uses cooperative packet caching and shortest multipath routing to reduce packet loss due to frequent link breakdowns. AOMDV [13], built on top of AODV, computes multiple link-disjoint paths to a destination by using an advertised hop-count. [14] proposes an extension to DSR where they provide sources and intermediate nodes with alternate paths during the route discovery phase. A technique to allow AODV to install backup routes at the neighbor nodes of a primary route is investigated in [15].

Multi-path routing can also alleviate congestion in wireless networks. The hotspot mitigation protocol (HMP) [16] 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) [17] carries congestion information forward in ROUTE REQUEST packets, allowing destinations to choose the least congested path.

Finally, there is an extensive literature on optimal dynamic routing and its formulation as the solution of an optimization problem [18], [19]. The problem can be equivalently cast in terms of routing variables at each node, and a completely distributed algorithm can be derived [19]. An approximation approach to minimum delay routing [20] provides a set of loop-free invariant conditions, while distributing traffic to approximate [19].

The Wardrop equilibrium can be interpreted as the equilibrium point of an optimization problem that minimizes the sum of the integral of delays on all network links. On the other hand, the Wardrop equilibrium differs from a pure delay minimization policy because it takes re-sequencing delay at the receiver into consideration. As mentioned in the introduction, this work builds on earlier work on design and convergence of delay-adaptive routing algorithms [6], [7]. This paper aims to bridge the gap between the theory of Wardrop routing and its implementation and use as a routing protocol in 802.11-based wireless mesh networks, and is an extended version of our work described in [8], [9].

III. A few sources regime in which perfect flow-avoiding routing is feasible

Traditional routing protocols for networks have been based on distributed algorithms that compute shortest paths in a graph. The original ARPANET routing naively used a delay-based metric as the link weight for the shortest path computation. This resulted in route flapping and instability [21], and consequently, most routing protocols have used a hop count-based link metric. The use of hop count as the metric to compute paths in a network makes a few implicit assumptions about the network:

1) The cost of a link is constant and does not vary with the load on the link.

2) The network attains global optimality when each flow individually minimizes its own path length. In wired networks, the capacities of links are huge and this assumption is reasonable. In wireless networks, a single flow passing through a shortest

path can completely load the nodes along the path, and shortest hop-count routing can lead to formation of hot spots.

Thus, hop-count based shortest path routing can result in a lot of flows interfering with each other and adversely affecting each others' throughput. It seems reasonable to expect that performance will improve by routing flows to avoid each others' path as far as possible. Note first that in IEEE 802.11, if the carrier sensing range is set to twice the reception range, then only one in a contiguous sequence of five links can transmit at a time, with the other links detecting physical carrier sense and deferring packet transmission. Thus, if the data rate capability of a node is W bits/sec (bps), then any path with more than five hops can only attain a throughput of at most $W/5$ bps. (This throughput can be achieved by pipelining transmissions). If carrier sensing could be disabled, the only constraint is the primary interference constraint arising from half-duplex radios. This constraint, which requires that no two simultaneously transmitting edges have a common source or destination, restricts the achievalbe rate to $W/3$ bps. In either case, we represent the maximum throughput obtainable by any long path by T_{max}. Now, we note that if any node on a path has another interfering flow passing through it, then (assuming both flows share the node equally), the throughput of each flow is at most $T_{max}/2$.

Theorem 1: (Regime for perfect flow-avoiding routing)
1) Consider N^2 nodes on the vertices of a $N \times N$ grid. Suppose $k(N)$ of these nodes are randomly chosen as sources, and $k(N)$ nodes as destinations corresponding to each of the sources. Suppose $\lim_{N\to\infty} \frac{k(N)}{N^{1/3}} = 0$.

If shortest path routing is used, then

$$\lim_{N\to\infty} P(\text{A given s-d pair obtains thruput} \geq T_{max}/2) = 0. \quad (1)$$

However, there exists a routing where flows can perfectly avoid each other, so that

$$\lim_{N\to\infty} P(\text{Every s-d pair obtains thruput} = T_{max}) = 1. \quad (2)$$

2) If the nodes are instead randomly located in a square of unit area, then (1, 2) continue to hold if

$$\lim_{N\to\infty} k(N)/(N^{1/3}/\log\ N) = 0 \quad (3)$$

Proof: (Outline) The proof of (1) follows simply from the fact that as $k(N)$ increases, the probability of a flow's path intersecting one of the other $(k-1)$ flows converges to one. Thus, we focus on (2). To prove it, we construct an optimal routing in which flows avoid each other.

Let $\{s_j = (a_j, b_j)\}$ be the sources and $\{d_j = (x_j, y_j)\}$ be the corresponding destinations. Around each source s_j, draw a set of $2k$ lines along the x-axis and y-axis. Repeat this procedure for each destination d_j. This set of $2k$ (horizontal and vertical) lines around any source (or destination) will be referred to as a "thick" line and used to construct an optimal routing.

The first step is to note that in the limit, no parallel "thick" lines intersect. The proof uses a counting argument. We map the problem to a bins and balls problem. Suppose we had N bins and $(2k)^2$ balls and we filled the bins with balls in the following manner: for the first $2k$ balls, choose the bin j in which ball i is placed independently and with uniform distribution on $(1, N - 2k + 1)$. Once ball i is placed in a bin j, fill the next $2k-1$ bins $(j+1, j+2, ..., j+2k-1)$ with $2k-1$ balls out of the remaining balls. Let X_j = number of balls in bin j. The number of unique strings $\{X_j\}$ is $= (N-2k+1).(N-2k+1)...(N-2k+1)/(2k)! = (N-2k+1)^{2k}/(2k)!$.

To count the number of valid strings $\{X_j\}$ with no "thick" lines intersecting, we can draw an equivalence between each such valid string and a valid $2k$-partition of the number $N-(2k)^2$. Thus, the number of valid strings is $= (N-(2k)^2+2k).(N-(2k)^2+2k-1).(N-(2k)^2+2k-2)...(N-(2k)^2+1)/(2k)! = \frac{(N-(2k)^2+2k)!}{(N-(2k)^2)!(2k)!}$.

Finally, P(no "thick" lines intersect) = number of unique valid strings / total number of unique strings = $((N-(2k)^2+2k)!/(N-(2k)^2)!(2k)!)/((N-2k+1)^{2k}/(2k)!)$. By using Stirling's approximation and the assumption on $k(N)$, we obtain $\lim_{N,k\to\infty} P$(no "thick" lines intersect) $= \lim_{N,k\to\infty} e^{-\frac{2k.((2k)^2-2k-2k+1)}{N-(2k)^2+2k}} = 1$.

We now construct a routing using L^1-optimal (i.e., Manhattan-metric) paths consisting of these non-intersecting sets of "thick" lines. The basic idea is to inductively construct a flow-avoiding route for each new flow by routing it around pre-existing routes, in such a way that it uses at most two additional lines in any "thick" line (We say that the new route "fattens" any segment of a pre-existing route by at most two additional lines); see Figure 1(a).

For source-destination pair 1: We start from s_1 and move parallel to the x-axis till we hit the column corresponding to d_1. Then, we move down/up till we hit d_1. This is an L^1-optimal path from source to destination, and since this is the first source-destination pair, there can be no conflicts with any pre-existing paths.

For source-destination pair r: Consider the L^1-optimal path of the r^{th} source-destination pair and suppose it intersects some of the paths for the first $r-1$ source-destination pairs that have already been drawn without intersections. Consider the drawing formed by the first $r-1$ ("fattened") paths in the Euclidean plane. This drawing is the planar drawing of a forest. Let $T_1, T_2, ..., T_n$ be the connected components of this forest. Let $(n_f)_i, (n_l)_i$ be the first and last points of intersection of the r^{th} source-destination pair's L^1 path with T_i. We progressively construct the path from s_r to d_r as follows:

(i) Start from s_r and follow the L^1 path from s_r to d_r till some T_i is hit at $(n_f)_i$. Then, consider the drawing formed by the segment of the L^1 path from s_r to $(n_f)_i$, the drawing of T_i (with "fattened" paths compressed to lines) and the segment of L^1 path from $(n_l)_i$ to d_r. This is the planar drawing of a tree. An $\epsilon-$neighborhood of this planar drawing is topologically equivalent to a disk, and thus its boundary can be oriented, and a re-route can be found from $(n_f)_i$ to $(n_l)_i$ along this oriented boundary. This new route "fattens" any pre-existing path segments in T_i by at most two.

(ii) Next, consider the L^1 path from $(n_l)_i$ to the destination and let T_j be the next component encountered. Repeat the above procedure, noting that T_i and T_j share no points in common and thus, the new route still "fattens" any pre-existing path segment by at most two. The procedure is repeated till we reach the destination d_r.

Note that at most two more lines are used in every "thick" line for routing the r^{th} source-destination pair, and therefore, r source-destination pairs can be routed using perfect flow-avoiding routes.

For the random case (3), one can tessellate the domain into $\theta(N^2/\log\ N)$ equal sized squares. All contain $O(\log\ N)$ nodes with probability approaching one as $N \to \infty$. Now, we can simply treat each such small square as one point in the earlier grid.

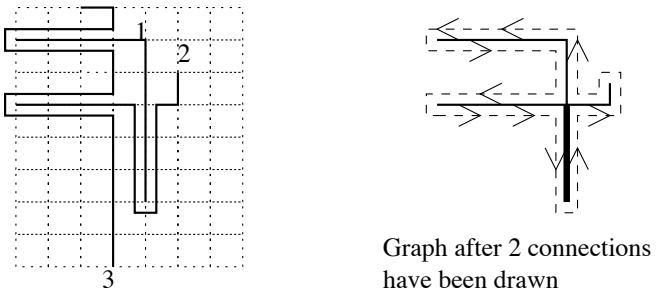


Fig. 1. Algorithm to optimally route k flows

TABLE I
SHORTEST-PATH VERSUS FLOW-AVOIDING ROUTING

Mean L^2 path len.	DSDV Tput (Kbps)	"Manual" Tput (Kbps)	% increase (total)
4	255.407	307.315	20.31
5	134.507	182.200	35.45
6	117.719	179.852	52.78
7	143.572	215.253	48.55

■

A simulation study of flow avoidance using ns-2 [22] with the Monarch wireless extensions [23] confirms the observation that flow-avoiding routing can outperform min-hop routing, even in more general environments, provided the number of flows is not too high. Scenarios consisting of two flows on a 8×8 grid were generated by placing sources and destinations randomly. Diagonally adjacent nodes on the grid are within one hop of each other. The shortest-path routing protocol used was the DSDV routing protocol [10]. Instead of implementing the optimal routing strategy detailed above, we extended the ns-2 simulator to include a new "Manual" routing agent. The "Manual" routing agent implements static routing and exports (oTcl) interfaces to add and remove routes from an (initially empty) routing table at every node. We then configured flow-avoiding routing by hand for each of the random scenarios using the "Manual" routing agent.

The throughput-delay performance of two example random scenarios (see Figure 2) is shown in Figures 3 and 4. Table I shows the average percentage improvement in throughput over six random scenarios with the average connection length same for both connections. The percentage improvement refers to the saturation value of the throughput as the delay becomes very large, when the network is loaded to the maximum.

Observe that the random scenarios shown in Figures 3 and 4 (and Table I) show (upto) a factor of two improvement in throughput-delay performance resulting from using re-routed paths, as opposed to using the L^2 shortest-path along the diagonals. (We have corrected for the increased throughput due to lack of routing overhead in "Manual".)

Finally, we note that the routes produced by the above algorithm can be improved by doing a greedy local optimization: for each path, pick two points on the path randomly and if possible, replace the segment between them by the L^2-optimal segment. Repeat till this is no longer possible (see Figure 1(b)). While the algorithm (with this greedy optimization) can be used to construct flow-avoiding routes, it only illustrates the feasibility of such an approach, and is not amenable to distributed implementation. To achieve that goal, we turn to a more practical approach.

IV. THE ROUTING ALGORITHM

The theoretical result in the previous section indicates that traffic-aware routing can provide considerable benefits. However, the scheme used to prove the result is centralized and should only be taken as proof of existence. Instead, we examine a more general adaptive approach. For every source-destination pair, we will attempt to drive routes towards an equilibrium where the mean delay along all utilized paths is equalized, and all unutilized paths have greater or equal potential mean delay. In a communication network, such an equilibrium has attractive properties vis-a-vis multi-path routing:

1) When packets have to be re-sequenced at the receiver and delivered in-order to the application, equalizing the average delay along utilized paths reduces receiver socket buffer space requirements and receiver socket buffer re-sequencing delays.

2) Equalizing the average delay along utilized paths mitigates TCP congestion misbehavior that results from TCP's adverse reaction to multiple paths and re-ordered packets.

3) Route adaptation using delay feedback allows re-routing of flows around traffic bottlenecks in wireless environments. This allows flows to automatically "avoid" each other and minimize interference.

A. Connection to the Wardrop equilibrium

The above equilibrium can be interpreted as a routing solution for a global optimization problem that minimizes the sum of the integral of delays on all links in the network [24]. In this sense, it is similar to the minimum-delay optimization approach of [18],[19]. There is another interpretation - as the property of a system in Wardrop equilibrium. Suppose that wireless connectivity is represented by a graph $G = (V, E)$, and let $SD = \{s, d\}$, where $s, d \in V$, represent the set of source-destination pairs. Consider the problem of routing packets over such a network where the performance measure we are interested in minimizing is the expected delay. The setting is a non-cooperative one in which each packet wishes to minimize the time taken to get from source to destination. The route chosen by each packet affects the latency experienced by other packets along its route, as well as in the vicinity of its route due to wireless interference. Since each packet has an infinitesimally small impact on the load of the network, the solution of this non-cooperative game corresponds to the Nash equilibrium when the number of agents goes to infinity. For each $\{s, d\}$ pair in SD, this corresponds to a solution where all flow paths have equal latency, which is lower than the latency experienced on any unutilized path. In the absence of this property, it would be possible for some packet to reduce its latency by switching to the unutilized path [25]. This is the "Wardrop equilibrium", defined in [26]:

1) Wardrop's first principle: All utilized paths from a source to a destination have equal mean delays.

2) Wardrop's second principle: Any unutilized path from a source to a destination has greater potential mean delay than that along utilized paths.

B. Why Wardrop routing can automatically lead to flow avoiding routes

Before describing our algorithm, we illustrate a simple example to show that routing towards a Wardrop equilibrium can automatically result in flow avoiding routes in wireless networks.

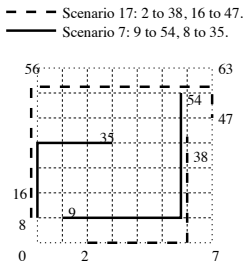


Fig. 2. Two random scenarios on a 8×8 grid

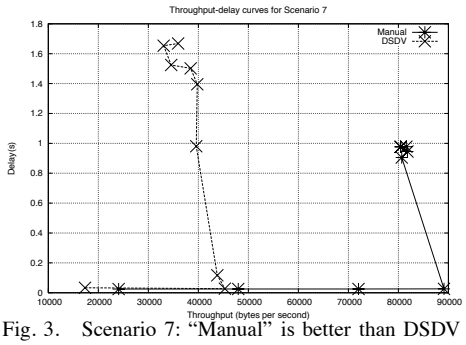


Fig. 3. Scenario 7: "Manual" is better than DSDV

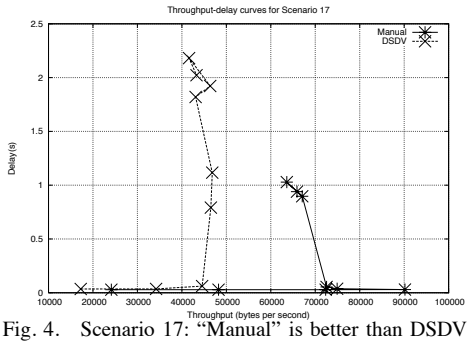


Fig. 4. Scenario 17: "Manual" is better than DSDV

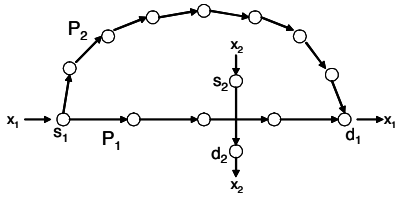


Fig. 5. Flow avoidance with Wardrop routing: a simple two flow example

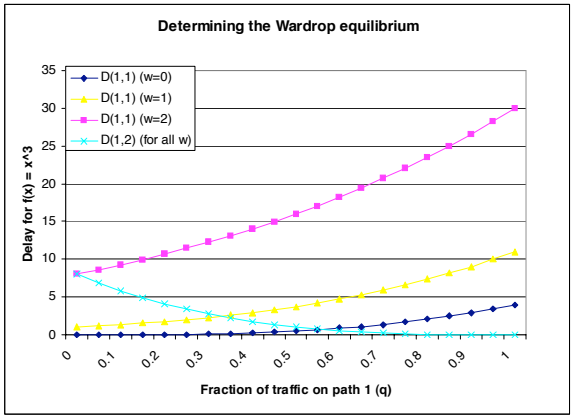


Fig. 6. Determining the Wardrop equilibrium: the equilibrium for a given w is attained at the value of q for which the delays $D_{1,1}$ and $D_{1,2}$ are equalized.

Consider the example in Figure 5, where there are two flows traversing a wireless network. The first flow f_1, from s_1 to d_1 has two available paths: path P_1 of length $l_1 = 4$, and path P_2 of length $l_2 = 2l_1 = 8$. The second flow f_2, from s_2 to d_2, is a one-hop flow that interferes with exactly one link of the path P_1. Suppose the input rates for the two flows are x_1 and $x_2 = wx_1$ respectively. Let us determine the Wardrop equilibrium for this system as a function of the parameter w, which captures the relative magnitude of the interference from the second flow on the first flow. In general, the link delay along a directed edge in a wireless network is a complex function of the topology, traffic on interfering edges, and the routing and scheduling algorithms used in the wireless network. As a first approximation, we assume that the delay along each directed edge is a convex non-decreasing function $f(x + y)$ of the traffic x on that edge, and the cumulative traffic y on interfering edges. Since this example is merely illustrative, we will use the function $f(x) = x^n$, with $n = 3$.

Suppose, in Wardrop equilibrium, that the optimal routing for flow f_1 directs q fraction of the traffic along path P_1 and $1 - q$ fraction of the traffic along path P_2. The delay along path P_1 is given by $D(1,1) = (l_1 - 1)(qx_1)^n + (qx_1 + x_2)^n = x_1^3(3q^3 + (q + w)^3)$. The delay along path P_2 is given by $D(1,2) = (l_2)((1 - q)x_1)^n = 8x_1^3(1 - q)^3$. In Figure 6, we have plotted the values of $\frac{D(1,1)}{x_1^3}$ and $\frac{D(1,2)}{x_1^3}$ as a function of the fraction of traffic q on the first path P_1 for different values of w. For $w < 2$, it is easy to see that the solution where all the traffic is on one of the two paths (i.e, $q = 0$, or $q = 1$) are not Wardrop equilibria, since the delay on the unutilized path in these cases is strictly lower than the delay on the utilized path. Thus, for a given $w < 2$, the Wardrop equilibrium is obtained by solving $D(1,1) = D(1,2)$ for q. On the other hand, when $w \geq 2$, the delay $D(1,1)$ on path P_1 is always greater or equal the delay $D(1,2)$ on path P_2, irrespective of what fraction of traffic q flows along P_1. This implies that for $w \geq 2$, the unique equilibrium consists of routing all traffic along path P_2, and completely avoiding the intereference bottleneck along path P_1 caused by the presence of flow f_2. This equilibrium routing is plotted as a function of w in Figure 7. This example shows the potential of traffic-adaptive routing to automatically lead to flow avoiding routing whenever possible in a wireless network.

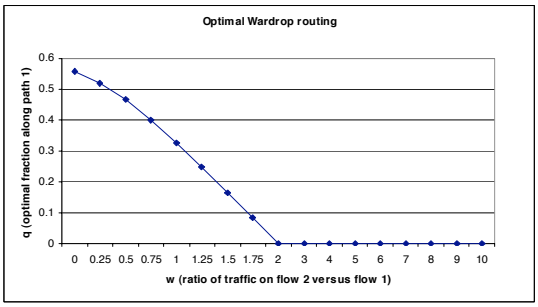


Fig. 7. The optimal routing as a function of w, the ratio of traffic betweens flows f_1 and f_2.

C. The STARA algorithm

Given these attractive features of routing towards a Wardrop equilibrium, our focus in the rest of this paper will be on the design and implementation of a practical distributed algorithm that equalizes the mean delays along all utilized paths in the network, and ensures that all unutilized paths have greater or equal mean delay. We start off by describing earlier work on the design [6] and convergence [7] of delay-adaptive routing algorithms in wireless networks. The problem addressed there was to construct an algorithm for reducing or increasing the flows along paths that would equilibrate to a Wardrop equilibrium. The basic idea 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. 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 a packet destined for 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 $j \in N(i,d)$. The routing probabilities $[p_{id}^j : j \in N(i,d)]$ are adjusted based on delay feedback.

Let D_{id}^j denote the average delay experienced by packets going from node i to d via the immediate neighbor j. This information is obtained by delay feedback using ACKs from the destination. Also, denote by $\overline{D}_{id}$ the average delay experienced by all packets from node i to d, without regard to what next node they were forwarded to. STARA iteratively adjusts the routing probabilities $[p_{id}^j : j \in N(i,d)]$. It increases 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 decreases it otherwise.

The algorithm consists of two components: an iterative scheme for delay estimation, and an iterative scheme for updating routing probabilities [6]:

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

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

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

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

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

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

The above algorithm features two modifications from the scheme discussed above. The $[.]^+$ is the projection map onto the simplex of probability vectors, and $\alpha[n]$ are appropriate step-sizes. When a packet for destination d arrives at node i, the node routes the packet to one of its neighbors $j \in N(i,d)$ with probability q_{id}^j rather than p_{id}^j. The actual routing probabilities used are thus a convex combination of the p_{id}^j's and a uniform probability distribution on all neighbors. This ensures a positive probability of probing to obtain delay information on unutilized routes [7], and is a standard feature in adaptive control; see [27]. Note that the Wardrop equilibrium is now defined with respect to $q = \{q_{id}^j\}$, i.e., $q_{id}^j > 0$ only if $D_{id}^j = \sum_{k \in N(i,d)} q_{id}^k D_{id}^k = \min_{\{k \in N(i,d)\}} D_{id}^k$.

An important issue in using delay information is that no two clocks are synchronized. The above algorithm works even when clocks in the network are not synchronized [6]. The probability update scheme only uses the difference between delays $(\overline{D}_{id} - D_{id}^j)$ for a source-destination pair. Thus, even if the clocks at i and d differ by an offset, by taking the difference $\overline{D}_{id} - D_{id}^j$, the offset is eliminated since it is present in both $\overline{D}_{id}$ as well as D_{id}^j.

D. The challenges

Our work on rendering Wardrop routing practical will build on the above algorithm. We will preserve all the valuable features, such as immunity to clock offsets at the nodes. However, to obtain a practical protocol that in fact delivers improved performance, we need to make several modifications. We begin by identifying three problems that render difficult the use of the algorithms presented in [6], [7]:

1) The routes followed by packets are unrestricted. They can be arbitrarily long. This causes several problems.

a) There are many bad routes that packets can go out on, and delays experienced by such packets can be exceedingly long.

b) Since the algorithm requires feedback on all these bad routes before it can adapt, its convergence can be very slow, rendering it impractical.

2) After the routing probabilities have converged to the correct estimates, the algorithm produces loop-free routes. However, packets can follow loopy paths while the algorithm is converging, which, in practice, is always the case.

3) The delay measurement in [6] relies on acknowledgments to carry measurements back to sources and intermediate nodes on a per-packet basis. This poses problems in implementation:

a) A scheme relying on transport layer ACKs to solve the network layer routing problem violates layering, and does not extend to unreliable transport layers (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 address these problems and obtain a practical protocol amenable to deployment, we re-designed the algorithm in [7] to obtain two protocols M-STARA and P-STARA.

1) We propose new mechanisms to control the length of paths followed by packets while the algorithm is converging. These mechanisms preserve the attractive properties of load adaptation and multi-path utilization, while retaining the property of convergence toward the appropriately defined Wardrop equilibrium.

2) We propose a mechanism which guarantees that routes followed by packets are loop-free even while the algorithm is still converging. This mechanism is not only compatible with the path length control mechanism above, but also continues to preserve convergence towards the appropriately defined Wardrop equilibrium for the resultant load adaptive multi-path routing protocol.

3) We propose a completely distributed delay measurement mechanism to replace the ACK-based scheme with the attendant problems discussed earlier. The new mechanism retains immunity to clock offsets. It consists of a light-weight link delay measurement protocol, and a distance-vector like neighborhood broadcast of average delay information. This mechanism is described in Section V.

E. Controlling paths and eliminating loops

Our first objective is to control paths so that the algorithm does not have to investigate all possible paths from source to destination, which would render the convergence very slow. 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 d, *M-STARA*. (The reason for

the name, denoting Multiplicative-STARA, is that with one additional modification, it can be used to strongly control path lengths. For brevity we omit discussion on this, referring the reader to [28].) The advantage of M-STARA is that it can be easily built on top of STARA by running a distance vector protocol to produce distances to all destinations. One other important property is that by merely suitably defining the notion of "neighborhood," we can prove convergence to a suitably defined Wardrop equilibrium; see Theorem 2 below.

2) Eliminating loops and controlling path lengths: P-STARA (Parity-STARA): The above mechanism restricts the set of paths investigated by M-STARA, and is easy to implement, but it does not provide any guarantees on path lengths. In fact, it does not eliminate routing loops.

Note that under any algorithm that converges to a Wardrop equilibrium with respect to the delay, after the routing probabilities have converged to the correct estimates asymptotically, the resulting probabilistic routes are guaranteed to be loop-free. However, while the adaptive algorithm is still converging, which in practice is *always*, packets can indeed follow loopy paths. Thus we need to guarantee loop-freedom of the algorithm at every instant.

To solve these two problems of controlling the path lengths as well as eliminating routing loops, we propose an additional mechanism, resulting in P-STARA.

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. In this way, we will show that over every two hops 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.

An equally important advantage, which we will expound on in the next section, is that this scheme admits a simple and completely distributed delay estimation algorithm matched to the definitions of the two neighborhoods used. To understand the simplicity of the solution provided by P-STARA, one should note the difficulties of controlling route length as well as killing loops while preserving Wardrop equilibration, all without sacrificing simplicity of implementation. For example, if one just allows a budget of k over the shortest path length by initially setting an "excess budget" packet field of k, and then progressively decrements it by one after each hop, then one can guarantee that the path length does not exceed the shortest path length by more than k hops. However, one would then have to use k separate routing tables in order to assure the Wardrop property since packets would be differently treated depending on the excess budget with which they arrive at a node. In comparison, the parity feature of P-STARA allows us to get away with just two tables, kills all loops, and yet allows us to control path lengths by a factor of two proportional to the shortest path length.

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 IV-E.1.

1) We include a field $F(p)$ in each packet, which is decremented by one by every node along the path. The source initially sets $F(p)$ to an arbitrary value. In practice, a field like the IP TTL field already behaves in this manner and can be used as $F(p)$.

2) We define the state of the packet as $X(F(p)) = (1 + F(p)) \; mod \; 2$.

2) 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 (4, 5, 6) below:

$$(p^j_{id})_F[n+1] = [(p^j_{id})_F[n] + \alpha[n] * (q^j_{id})_F[n] * ((\overline{D}_{id})_F[n+1] - (D^j_{id})_F[n+1])]^+, \quad (4)$$

$$(\overline{D}_{id})_F[n+1] = \sum_{j=1}^{N(i,d)} ((D^j_{id})_F[n+1] * (q^j_{id})_F[n]), \quad (5)$$

$$(q^j_{id})_F[n] = (1-\epsilon) * (p^j_{id})_F[n] + \epsilon * 1/N^{X(F)}(i,d). \quad (6)$$

We now prove that P-STARA has all the desired properties of controlling path lengths and eliminating all routing loops even during the transient phase, while providing the load adaptation of the multi-path routing protocol so that it converges to a Wardrop equilibrium. In fact we will prove that it provides loop-free routes even when the distance estimates are not accurate, or are still converging, provided only that the distance estimation scheme uses the destination originated sequence numbering technique of DSDV [10]. For brevity, we avoid an explanation of the sequence numbering technique used by DSDV, as well as the technical definitions of Cesaro convergence in [7], and provide only a brief outline of the corresponding proofs.

Theorem 2: (i) Suppose packet p with source s and destination d follows path P with length $L(P)$. Then $L(P) \leq 2S(s,d)$.

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

(iii) P-STARA produces loop-free paths from every source s to every destination d, even when the distance estimates $S(i,d)$ are not accurate or are still converging, provided only that the distance vector scheme uses sequence numbering to avoid loops.

(iv) The routing probabilities $q = \{q^j_{id}\}$ produced by P-STARA converge almost surely to the set of ϵ-Cesaro-Wardrop equilibria with respect to the restriction to the allowed routes.

Proof: (i) Since $F(p)$ is decremented by 1 at every step, $F(p)$ is alternately even and odd at every node along the path P. Thus, after every two hops along the path P, a packet is closer to the destination by at least one more hop (since we are forced to choose a shortest-path neighbor at one of these two hops). Thus, $L(P) \leq 2S(s,d)$.

(ii) 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 contains no cycle of size n for any $n \geq 2$.
(iii) We can redefine the neighborhood sets $N^1(i,d)$ and $N^0(i,d)$ (see [28] for details) so that the 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 can obtain a contradiction by using the fact that the distance estimates $S(i,d)$ decrease by at least one after every two successive hops along the path P.
(iv) The last property capitalizes on the fact that all we have changed is the definition of admissible neighborhoods with respect to a destination. This preserves the convergence properties of the original scheme. Thus the proof paralleling that in [7] can be used. ■

V. From Algorithm to Protocol: Delay Estimation

The Wardrop routing algorithms described in Section IV rely on obtaining delay measurements using (transport layer) ACK schemes. A faithful implementation of such a scheme in a network would violate the architectural separation of functionality between transport and network layers [8]. An alternative is to use to estimate end-to-end-delays using routing-protocol generated probe packets; such a scheme suffers from the prohibitive communication overhead required for accurate estimation.

A. Distributed end-to-end delay estimation for M-STARA and STARA

Let d_{ij} denote the measured link delay from node i to its neighbor j during some phase in which we wish to estimate average delays. Then, the average delay from node i to destination d, $\overline{D}_{id}$, must satisfy the equation:

$$\overline{D}_{id} = \sum_{j \in N(i,d)} q_{id}^{j} (d_{ij} + \overline{D}_{jd}).$$

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:

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

B. Distributed end-to-end delay estimation for P-STARA

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_{ij}^e and d_{ij}^o 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 for even field $\overline{D}_{id}^{j,e}[n]$ and for odd field $\overline{D}_{id}^{o}[n]$ must satisfy:

$$\overline{D}_{id}^{e} = \sum_{j \in N(i,d)} q_{id}^{j,e} (d_{ij}^{e} + \overline{D}_{jd}^{o}),$$
$$\overline{D}_{id}^{o} = \sum_{j \in N(i,d)} q_{id}^{j,o} (d_{ij}^{o} + \overline{D}_{jd}^{e}).$$

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

$$\overline{D}_{id}^{o}[n] = \sum_{k \in N(i,d)} q_{id}^{k,o} (d_{ij}^{o} + \overline{D}_{jd}^{e}[n]).$$

and similarly for the odd field.

C. Immunity to clock offsets

The distributed delay measurement procedure retains the immunity to clock offsets of the original STARA routing algorithm [6]. We prove this for M-STARA, the proof for P-STARA is identical. Consider a single destination, dropping the subscript d in the notation. Let f_i be the offset of node i's clock from some reference clock. Let D_i^{real} be the equilibrium averages, and let $D_i^{real,j}$ be the equilibrium averages through neighbor j assuming that the clocks were synchronized throughout the network. Let d_{ij}^{real} be the corresponding link delay from node i to node j. With clock offsets present, let D_i be the equilibrium measured averages, D_i^j be the equilibrium measured averages through neighbor j, and let d_{ij} be the corresponding link delays. Then, $D_i^{real,j}, D_i^{real}$ and d_{ij}^{real} satisfy (7). Similarly for D_i^j, D_i and d_{ij}. Furthermore, these parameters are related as $d_{ij} = d_{ij}^{real} + f_i - f_j$.

Claim 1: For $j \in V(G)$, the estimates D_i and D_i^j 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.

Proof: (Outline) By induction on the size $n(G) = |V(G)|$ of the graph containing the destination node d. We combine the fact that the link delay (i,j) is offset from its real value by $f_i - f_j$, and that node j is in $G' = G - i$ with $n(G') = k - 1$. Then, by induction, the estimate D_j differs from its real value by $f_j - f_d$, which proves the result for all nodes in a graph of size $n(G) = k$. ■

D. Link delay estimation

The average delay measurement scheme detailed in the previous section assumes that the link delays d_{ij}'s are available instantaneously. In practice, we need to measure and average these out in real time at a faster rate than the average delay update algorithm. Such a link delay measurement protocol operates more frequently than other components of the routing protocol (e.g., average delay estimation), and thus must have as little communication overhead as possible. We use a lightweight protocol to perform link delay measurement to obtain the delay estimates d_{ij} from node i to node j. As packets cross a link (i,j), the protocol records timestamps at node i and j. These statistics are averaged and exchanged periodically so that both ends of each link (i,j) can determine the average link delay d_{ij} across the link up to clock offsets as the difference between the average sender timestamp and the average receiver timestamp. The protocol operates correctly even without reliable delivery of every message. The state machines for the protocol implementing the scheme are shown in Figures 8 and 9.

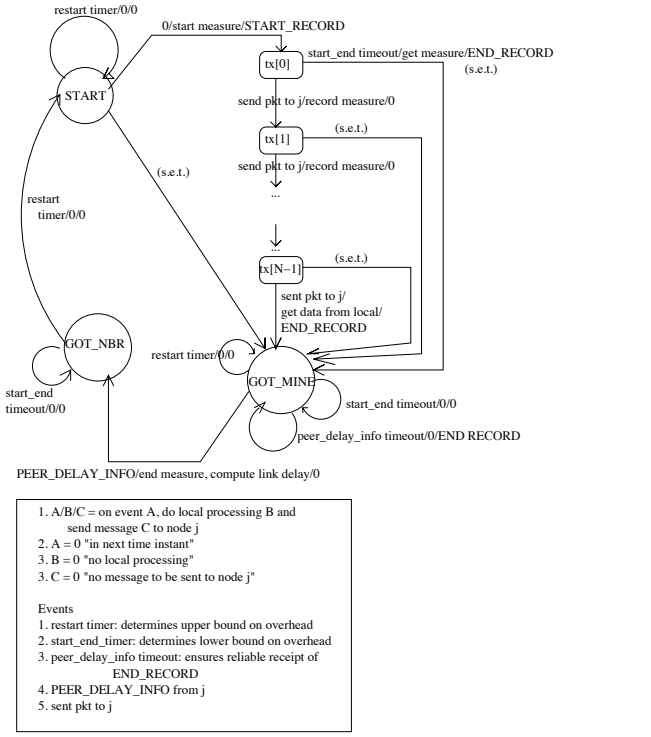


Fig. 8. Sender side state machine.

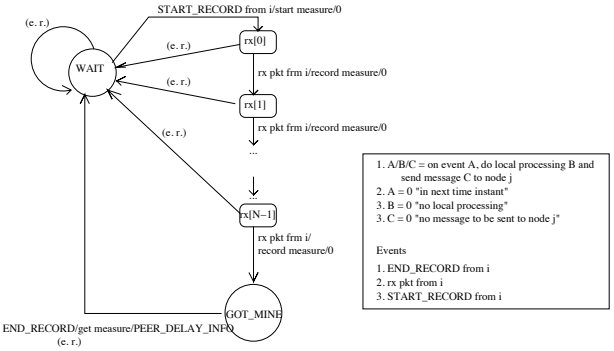


Fig. 9. Receiver side state machine.

VI. Simulation Study

In order to understand the protocol behavior, we have carried out an extensive simulation analysis for wireless environments using the ns-2 simulator with the Monarch wireless extensions [23]. In all our simulations, we use the two ray ground propagation mode, the IEEE 802.11 DCF MAC, a radio model based on the Lucent WaveLAN 2 Mbps radio with a radio range of 250m. We have implemented the Wardrop routing protocols in ns-2, using DSDV [10] as a base protocol to obtain distance information. All of our simulations (except in the section on topology impact) were carried out with N^2 nodes on the intersections of a $N \times N$ grid. The interface queue length is set to 50 packets, and the retry parameters of the MAC are left at their default values MAC_ShortRetryLimit = 7, and MAC_LongRetryLimit = 4. We use CBR sources with a constant 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. Throughput and delay statistics are obtained from ns-2 tracefiles, while other parameters like routing probabilities, average delay, link delay, routing overhead etc. are obtained through a custom oTcl interface. For each scenario, we repeated the simulations for 3-5 runs to confirm that the algorithm converged to the same equilibrium.

We use the following metrics to evaluate performance: (i) throughput-delay performance, (ii) path length and quality, (iii) routing overhead, and (iv) convergence speed. The throughput-delay performance of a routing protocol is a reliable metric in determining its performance in real networks. While auxiliary metrics like routing overhead, path length, packet delivery ratio 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. Throughput-delay performance

To obtain throughput-delay curves for a particular scenario, we load the network progressively till overload and for each level of network loading, measure the average delay and system throughput. We compare the performance of Wardrop routing to that of the shortest-path DSDV protocol. In all throughput-delay simulations, ADP is set to 15.0, LDP to 5.0 and $LDPF$ to 25.0. (These parameters are explained later in this section.) In examining the throughput-delay performance of Wardrop routing, we make the following observations:

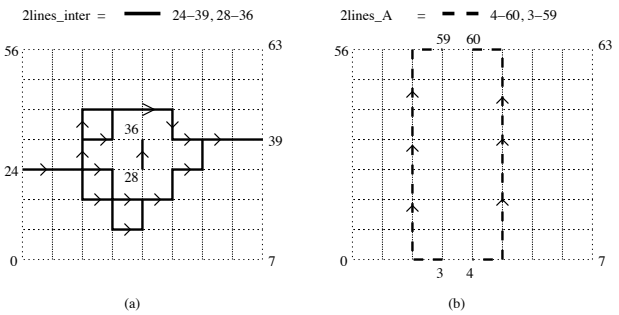


Fig. 10. Few sources: Wardrop flow-avoiding multipaths

1) *Flows "avoid" each other in "few source" environments*

Wardrop routing adjusts flow probabilities in such a manner as to equalize the delays along utilized paths. In the steady state, unutilized paths have greater delay than the delay along utilized paths, and thus we expect it to be able to route around traffic hot spots. In IEEE 802.11 wireless environments, congestion results from two-hop interference in the neighborhood of nodes and routing performance will be improved over the IEEE 802.11 MAC if the routing protocol generates routes that are two-hop separated from each other. To test whether the routes generated by Wardrop routing in the few source regime have this property, we compared the performance of the shortest-path DSDV routing protocol, "Manual" routing (in which we pre-configured routing so that one flow routes around another) and Wardrop routing for some canonical "few source" scenarios.

The scenarios and the flow-avoiding paths followed are shown in Figure 10. In the 2lines_inter scenario (Figure 10(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 11). Wardrop routing routes

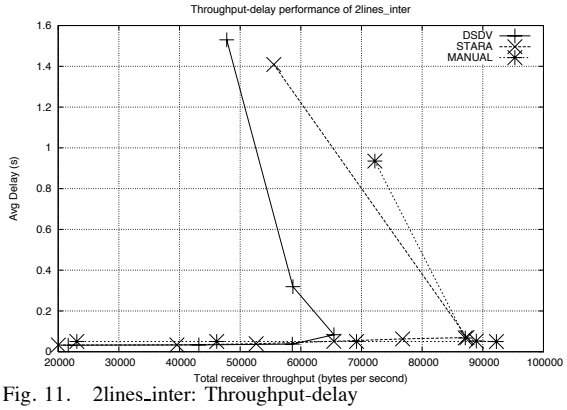


Fig. 11. 2lines_inter: Throughput-delay

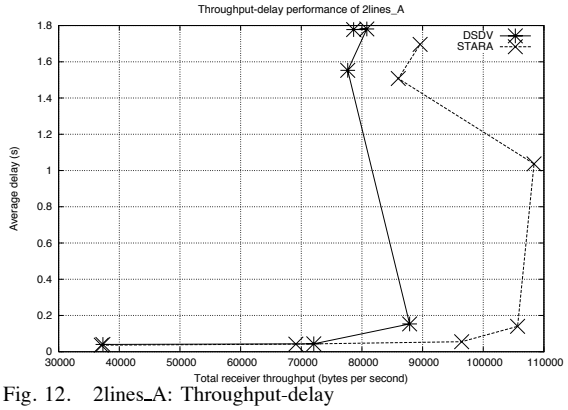


Fig. 12. 2lines_A: Throughput-delay

the longer flow around the bottleneck, and manages to achieve reasonable performance.

Another interesting scenario is Scenario 2lines_A (Figure 10(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. At higher network loading, these paths become the bottleneck and Wardrop routing switches to two-hop separated paths that improve throughput performance (Figure 12).

An examination of the tracefiles shows that at low rates, Wardrop routing primarily uses the shortest path from node 24 to node 39. As the rate is increased, it diffuses flow slightly around the flow 28-36, but the multiple paths interfere with each other. As the rate is still further increased, we would expect it to converge to multiple paths that are two-hop separated from each other except in the region close to nodes 24 and 39, and thus be able to out-perform "Manual" routing. However, this is not always the case, because at very high input rates the source and destination nodes are very congested, and the link delay mechanism close to the source and destination nodes is not always successful in obtaining accurate link delay estimates. Moreover, the average delay packets sent using broadcast suffer a high drop rate, and as a result Wardrop routing uses inaccurate delay estimates, ending up using multiple nodes in the neighborhood of the destination (or the source). (Wardrop routing is unable to outperform "Manual" in most scenarios.)

While attempting to equalize delay along utilized multi-paths, Wardrop routing also causes flows to avoid each other. This can be clearly seen in scenario 2lines_A (Figure 10(b)), where we have two flows 3-59 and 4-60 parallel to each other, but not intersecting. The performance is clearly better than that of DSDV, as the throughput-delay performance in Figure 12 clearly shows. Again, DSDV is forced to use adjacent parallel paths that interfere with each other, and thus limit its performance. On the other hand, Wardrop routing uses the shortest paths at low rates but as the network is made more and more congested, it switches to paths that are two-hop separated except at the source and close to the destination.

2) *If optimal avoidance is not possible, Wardrop routing equilibrates to paths that avoid each other as far as possible*.

Since it is not always possible to route all flows to avoid each other, a good routing protocol should produce routes that minimize the interference due to nodes within two hops of each other. In simulations, we observe that Wardrop routing generates routes with this property in the majority of scenarios, e.g., Figures 13 and 14 which show the routes for scenarios with three and four sources respectively.

TABLE II
WARDROP IMPROVEMENT OVER DSDV

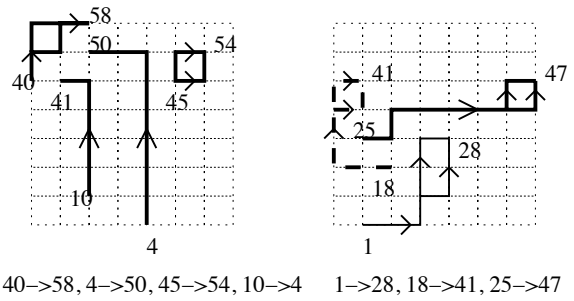
No. of connections	% of scenarios with improvement	Avg. % increase in sat. thruput
2	75	13.94
3	87	23.47
4	62	14.33
7	100	23.16

3) *Wardrop routing improves throughput-delay performance even when the "few sources" assumption is violated*. Even with more source-destination pairs in the network, Wardrop routing is able to obtain throughput performance gains by minimizing interference, thereby efficiently using the scarce wireless resource. For example, Figure 15 shows a scenario with three flows, each of length five, and Figure 16 shows a scenario with seven flows, each of length four.

4) *Wardrop routing's throughput-delay performance is consistently better than DSDV*. To verify the effectiveness of Wardropian delay-adaptive routing, we generated random scenarios on a 8×8 grid, and measured the saturation throughput. For a majority of scenarios considered, the saturation throughput of Wardrop routing, i.e., the maximum throughput supportable by Wardrop routing, was better than that of DSDV (see Table II). The percentage improvement for scenarios with seven connections is shown in Figure 17.

B. Path length and quality

It is important to ensure that the paths followed by packets are loop-free at every instant, or failing that, that packets do not follow wild paths to the destination. To check this, we ran the STARA, M-STARA and P-STARA variants of Wardrop routing on an 8×8 grid with direct connections between diagonally adjacent grid nodes. We simulated ten scenarios representing varying degrees of network congestion and traffic burstiness, 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). M-STARA drastically reduces the percentages of packets following loops during convergence (see Figure 18 for scenario $r2\_20$), while P-STARA completely avoids loops. The performance improvements in average path length and packet loops is shown in Table III.



40–>58, 4–>50, 45–>54, 10–>4 1–>28, 18–>41, 25–>47

Fig. 13. 4random Scenario 15: Wardrop multipaths

Fig. 14. 3random Scenario 18: Wardrop multipaths

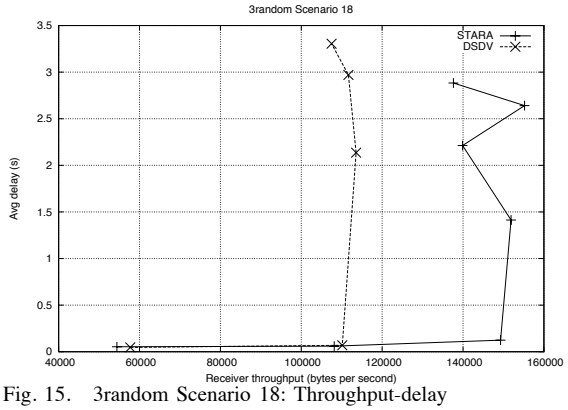


Fig. 15. 3random Scenario 18: Throughput-delay

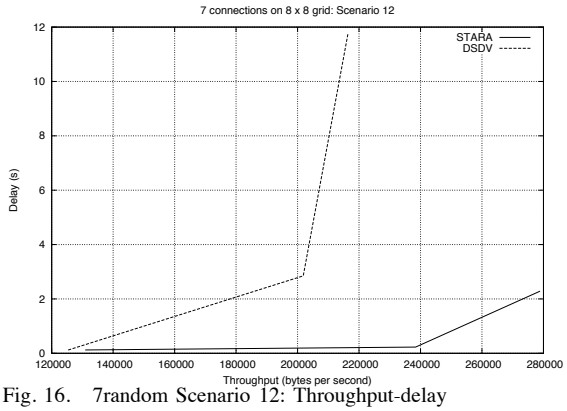


Fig. 16. 7random Scenario 12: Throughput-delay

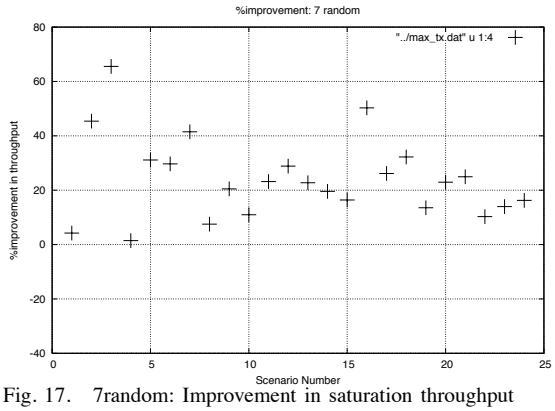


Fig. 17. 7random: Improvement in saturation throughput

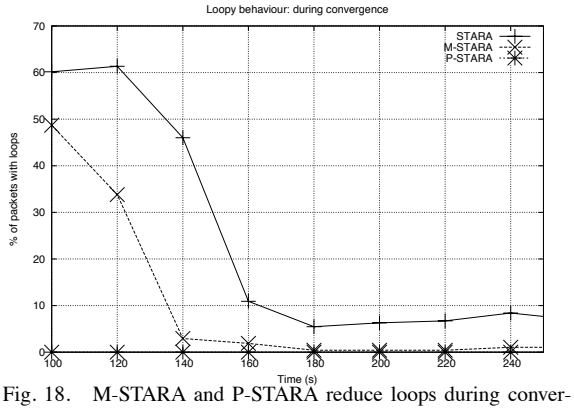


Fig. 18. M-STARA and P-STARA reduce loops during convergence

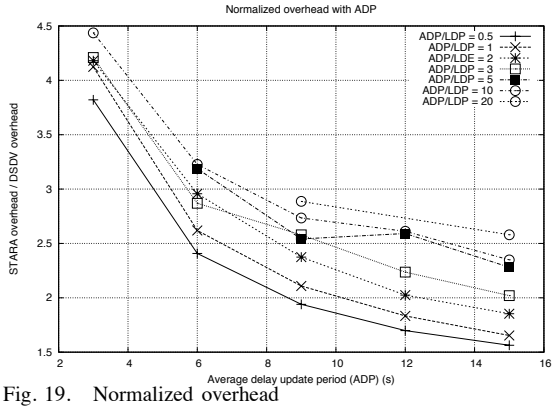


Fig. 19. Normalized overhead

C. Routing overhead and parameter selection

The Wardrop protocol implementation can be tuned by adjusting the following parameters:

1) *Average delay update period (ADP)*: The rate at which nodes broadcast average delay information to neighbors.

2) *Link delay re-start timer (LDP)*: Wardrop routing attempts to re-start the link delay protocol once every LDP seconds.

3) *Link delay forced re-start timer ($LDPF$)*: Wardrop routing forcibly re-starts the link delay protocol once every $LDPF$ seconds to ensure that link delay information is obtained.

4) *DSDV periodic update period*: 15s.

The optimal choice of parameters $ADP, LDP, LDPF$ represents a trade-off between convergence speed and routing overhead. A plot of routing overhead (normalized with respect to DSDV overhead) in a 10×10 grid is shown in Figure 19. Observe that the effect of increasing ADP for a fixed ADP/LDP, or decreasing ADP/LDP for a fixed ADP is to reduce the routing overhead. However, convergence time depends largely on ADP, provided that a few link delay measurements are obtained in every ADP time period, i.e, $ADP/LDP > 1$ (for a convergence time analysis, see [28]). A good heuristic to pick ADP and LDP is to set $ADP/LDP = 2$ or 3, and then select ADP for satisfactory convergence time. With our choice of parameters, Wardrop overhead (with DSDV overhead included) is roughly twice the DSDV overhead.

TABLE III

PATH LENGTH AND QUALITY

Scenario	% decrease in path len (M-STARA)	% decrease in loops (M-STARA)
$r1\_20$	17.9	53.02
$r1\_40$	6.3	39.72
$r1\_60$	17.2	42.04
$r2\_20$	11.8	70.64
$r2\_40$	19.9	69.03
$r2\_60$	14.8	68.27
$r3\_20$	16.1	62.17
$r3\_40$	11.4	45.09
$r3\_60$	7.2	33.53

D. Impact of topology

We expect performance improvements provided by Wardrop routing to depend on the existence of sufficiently diverse paths. The grid topology used has sufficient path diversity, allowing our traffic-adaptive routing algorithm to route around interference bottlenecks whenever possible. In order to investigate the impact of topology, we conducted simulations on a set of 500 randomly chosen topologies. The topologies were generated by placing nodes randomly on an area of size 2500×2500, taking care to ensure that the connectivity graph did not have any partitions. The number of nodes used in each topology varies from 50 to 250, and the number of flows used varies from 10 to 20. We use the saturation system throughput to characterize the performance. The cdf of the improvement in system throughput from using Wardrop routing is shown in Figure 20. The average improvement over all topologies is 18.85%. As the node density increases, the number of available paths increases. Intuitively, we expect this to increase the performance of traffic-adaptive routing. This is seen in Figure 21, where the performance improvement of Wardrop routing over DSDV increases as the number of nodes in the topology increases, saturating when there are 150 nodes and higher.

These simulations, carried out over random topologies, seem to indicate that Wardrop routing's benefits are roughly correlated with the availability of sufficient path diversity for interference-avoiding routing. It is possible that some practical networks may not have such path diversity, as for example, in the case of backhaul mesh networks that have a hub-and-spoke topological structure consisting of "backbone" and "access points". In such topologies, we expect the benefits of Wardrop routing to decrease.

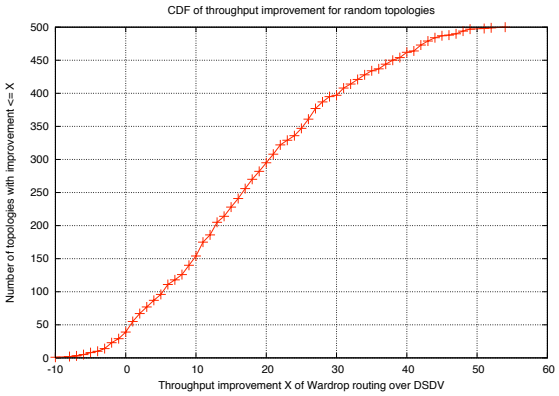


Fig. 20. Random topologies: CDF of throughput improvement

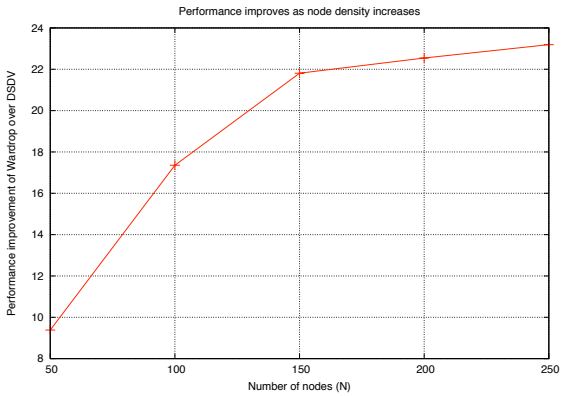


Fig. 21. Random topologies: throughput improvement with density

VII. IMPLEMENTATION ARCHITECTURE

One of the most important considerations in the design of the Wardrop routing protocol described in this paper was implementability. In implementing the protocol, we have taken care to ensure that:

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 extensible.

The architecture of the implementation is shown in Figure 22. The key architectural choice we made was to separate probabilistic packet forwarding from delay estimation. We provide an in-kernel per-packet probabilistic forwarding mechanism consisting of multiple in-kernel forwarding tables. Each 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 userspace library provides an API interface to this in-kernel probabilistic forwarding mechanism. This separation of routing policy from forwarding mechanism allows us to cleanly implement different probabilistic route adaptation policies in userspace.

The protocol logic is almost completely in user-space. The link delay measurement module implements the state machines in Figures 8 and 9. The average delay measurement module implements a "distance vector"-like protocol to exchange average delay information with neighbors on a periodic basis. The distance vector routing module uses an implementation of DSDV developed at UIUC [29] to produce hop-count information that is used in P-STARA to guarantee loop-freedom. The probability update policy module implements various Wardrop route adaptation algorithms, including P-STARA, M-STARA and STARA.

Our implementation fully utilizes advanced kernel routing features like equal cost multipath routing, netfilter, iptables, policy routing and MARK target routing to minimize the need for kernel modifications. Nevertheless, the implementation needed a small amount of in-kernel mechanism, which we added to the Linux 2.4.20 kernel:

1) *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 forwarding engine caches routes on a per-connection basis, and all packets of a connection end up using the cached route. Instead of disabling the route cache and incurring a huge performance hit, we tricked the route cache into simulating per-packet multipath routing by randomizing one of the keys used for the cache lookup in the Linux kernel. This

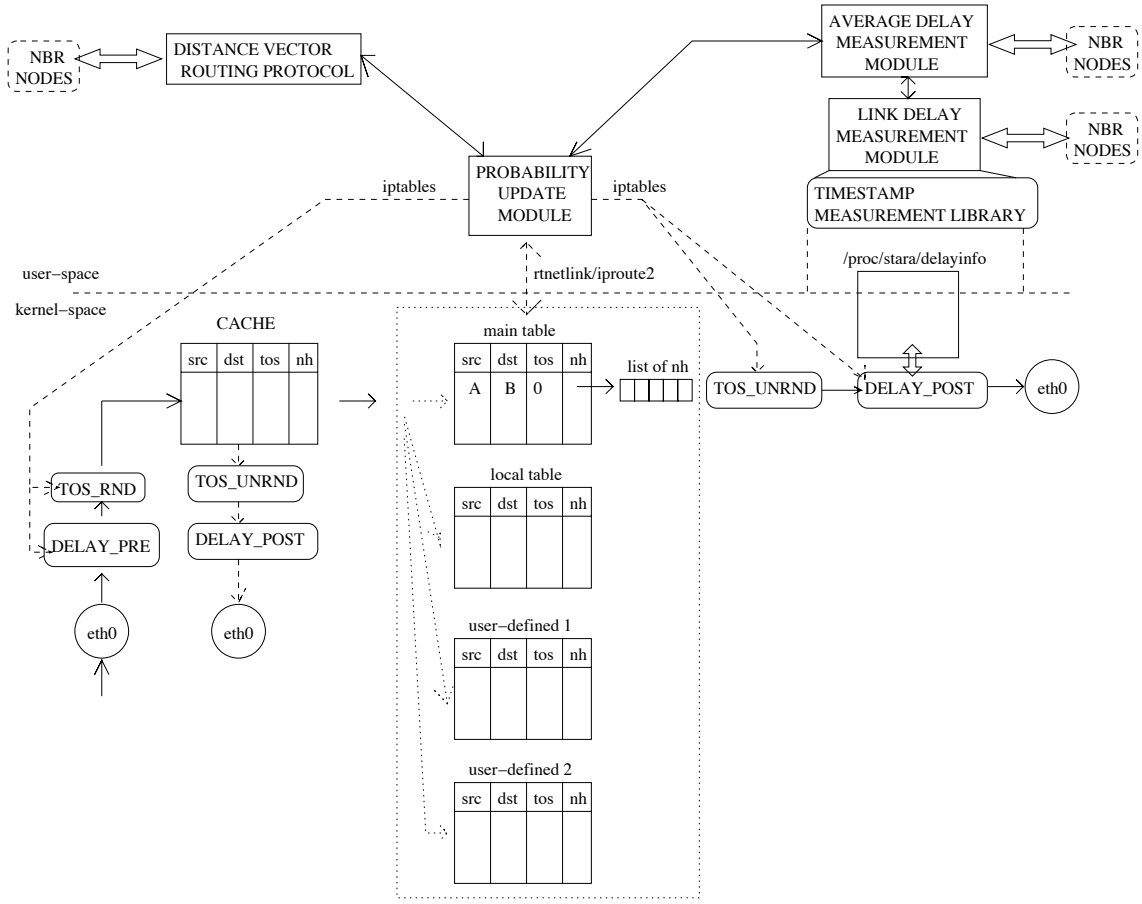


Fig. 22. Implementation architecture.

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. This technique has the added benefit of speeding up the convergence of Wardrop routing in practice as a side-effect.

This randomization procedure is implemented through the use of two iptables/netfilter target modules [30]. The TOS_RND netfilter module is attached to the PRE_ROUTING and OUTPUT hooks, while the TOS_UNRND netfilter module is attached to the POST_ROUTING and INPUT hooks respectively.

2) *Timestamp measurement*: This module timestamps and classifies packets based on incoming and outgoing MAC addresses as they pass through the kernel forwarding engine and exports the collected information to userspace using the $/proc$ filesystem. It is written as two netfilter/iptables target modules. DELAY_PRE is attached to the PRE_ROUTING and OUTPUT hooks, while DELAY_POST is attached to the POST_ROUTING hooks.

VIII. Measurement Study

We have carried out a measurement study of the Wardrop implementation on a six node testbed to verify the practical effectiveness of Wardrop route adaptation. 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*. Throughout our experiments, the following MAC parameters are fixed at every node as follows: (i) RTSThreshold = 100 (ii) FragThreshold = OFF (iii) Retry = 8 (iv) Channel = 10 (v) Rate = 11Mbps. (vi) Transmit Power = 30mW. We used experimentation experience gained during an earlier study of TCP [31] to resolve common problems:

1) *MAC filtering*: Topologies were generated by collocating all nodes and using iptables MAC filtering to create virtual multi-hop scenarios. This approach assists 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 room 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" [31]. We use copper tape to carry out the experiments.

A. Performance data

The topologies used are shown in Figure 23. The throughput performance of DSDV and Wardrop routing for the topologies is summarized in Table IV and Table V. 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. The performance results in this section should be treated as a proof-of-concept; further large scale testing is still needed.

Both protocols see an (expected) drop in throughput performance as the number of hops along a path increases (see Table IV) due to increased self-interference. Even if we attribute all

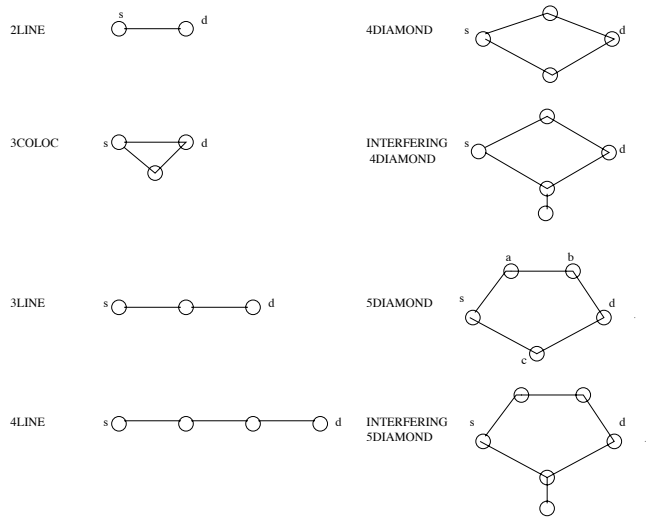


Fig. 23. Experimental topologies

the difference in performance to our protocol's additional routing overhead, (in practice, heating effects and in-building wireless interference also matter), the worst-case impact of overhead on performance is approximately 12.3%.

TABLE IV
BASIC TOPOLOGIES

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

Further, there is a significant benefit to using Wardrop routing 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 IEEE 802.11 frames (no RTS/CTS handshake, no ACKs and the ICMP ECHO REQUEST is dropped at the target node); thus, only the target node is affected by the interference. (If the target node sent any packets in the reverse direction; then these packets would be sent out at 30mW and would silence every other node in the topology.)

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, Wardrop routing is able to completely route around the interference and achieves better performance. (Observe that Wardrop routing also loses performance in INTERFERING 4DIAMOND, due to a head-of-line blocking artifact of our setup).

The 5DIAMOND scenario is designed to accentuate the bad performance of DSDV. 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. (DSDV's sequence numbering mechanism allows DSDV to use the 3-hop path on rare occasions, but the route oscillates back to the 2-hop path almost immediately.) On the other hand, Wardrop routing is able to route around the interference and use the longer 3-hop path and achieve better performance.

TABLE V
WARDROP ROUTE ADAPTATION

Topology	DSDV (Kbps)	Wardrop routing (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 by generating ping traffic from s to d in the 5DIAMOND topology. As a precaution against address spoofing, the Linux TCP/IP implementation drops packets arriving at a node whose source address matches that of the node. Since the only loops possible in 5DIAMOND are of the form s-a-s, all loopy packets will be dropped by Linux. In our experiments, P-STARA is loop-free at every instant, while 4.76% of packets follow loops with STARA.

B. Lessons learned

Our performance study provided additional insight into Wardrop routing not available through analysis or simulation:

1) Since the average delay estimation protocol uses a distance-vector like update to propagate average delay information through the network, it suffers from an "average delay counting-to-infinity" problem similar to the counting-to-infinity problem faced by distance vector protocols. This problem can be solved, as in distance vector protocols like RIP [32], by limiting the maximum allowable average delay to some large constant.

2) Clock offsets, in theory, do not affect the protocol correctness. In practice, the detection of "average-delay-counting-to-infinity" is slowed down by clock offsets, since the maximum allowable average delay must be a value larger than the largest possible clock offset in the network.

3) Kernel transmit buffering renders userspace link delay measurement tricky by preventing the userspace code from synchronizing to the actual departure of the START_RECORD message from the "head" node to its "tail" neighbor. This can result in head and tail collecting timestamps for different sets of packets, and incorrect delay measurement with a naive implementation. For example, suppose there are N packets in the kernel transmit buffer at the head. A naive implementation of userspace link delay measurement at the head issues a START_RECORD message to the tail, and uses /proc to record the next N kernel packet transmission timestamps, corresponding to the N packets in the transmit buffer *ahead of* the START_RECORD message. On the other hand, the tail records N receive timestamps corresponding to the START_RECORD message and the $N-1$ packets *after* it.

Our implementation solves this problem by allowing for synchronization across the userspace-kernelspace boundary in spite of kernel buffering. We add a *delayprocid* field to the sk_buff data structure and a socket option to set/retrieve the value of this field from userspace. This field is exported to userspace along with timestamp information by the timestamp measurement module, allowing userspace to synchronize to the precise instant at which the START_RECORD message leaves the node.

IX. Conclusion

In this paper, we have formally established a few sources regime where flow-avoiding routing provides a four-fold improvement over the throughput of shortest path routing when carrier sensing can be disabled, and a 3.2 factor improvement otherwise. Focusing on static wireless networks, we have developed a multipath, load adaptive routing protocol that (i) is completely distributed, (ii) does load balancing, (iii) can produce loop-free paths, and (iv) is elegantly implementable in the operating system kernel. We propose mechanisms to control the lengths of paths and provide loop freedom at all times. We also propose a completely distributed delay estimation procedure that eliminates the need for ACK-based delay estimation.

Simulations indicate that the protocol appears to be effective in obtaining improved throughput-delay performance gains over shortest-path routing in static networks. We have implemented the protocol in userspace on a modified Linux 2.4.20 kernel based on an architecture that separates probabilistic multipath routing from delay estimation. A proof-of-concept measurement study indicates the effectiveness of Wardrop route adaptation.

Our longitudinal work on Wardrop routing at the theoretical, simulation, implementation and testing levels shows that there is scope for traffic adaptive routing that outperforms minimum hop routing. Second generation protocols for wireless networks may well benefit from such adaptive routing.

References

[1] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, and P. Muhlethaler, "Optimized link state routing protocol (OLSR)," RFC 3626.

[2] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Kluwer Academic Publishers, 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.

[4] V. Park and S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proc. IEEE INFOCOM*, 1997.

[5] M. P. Zygmunt Haas and P. Samar, "The zone routing protocol for ad hoc networks," 1999, IETF MANET Internet Draft, available at http://www.ietf.org/.

[6] P. Gupta and P. R. Kumar, "A system and traffic independent adaptive routing algorithm for ad hoc networks," in *Proceedings of IEEE Conference on Decision and Control (CDC)*, 1997.

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

[8] V. Raghunathan and P. R. Kumar, "On delay adaptive routing in wireless networks," in *Proceedings of IEEE Conference on Decision and Control (CDC)*, 2004, pp. 4661–4666.

[9] ——, "Issues in Wardrop routing in wireless networks," in *Proceedings of the first International conference on wireless Internet (WICON)*, 2005, pp. 34–41.

[10] C. E. Perkins and P. R. Bhagwat, "Highly dynamic destination-sequenced distance vector routing for mobile computers," in *Proceedings of ACM SIGCOMM*, 1994.

[11] E. Royer and C.-K. Toh, "A review of routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communication*, vol. 6, pp. 46–55, 1999.

[12] W. S. Alvin Valera and S. Rao, "Cooperative packet caching and shortest multipath routing in mobile ad-hoc networks," in *Proc. IEEE INFOCOM*, 2003.

[13] M. Marina and S. R. Das, "On-demand multipath distance vector routing in ad-hoc networks," in *Proceedings of IEEE ICNP*, 2001.

[14] A. Nasipuri and S. R. Das, "On-demand multipath routing for mobile ad hoc networks," in *Proceedings of IEEE ICCN*, 1999.

[15] S. J. Lee and M. Gerla, "AODV-BR: Backup routing in ad hoc networks," in *Proceedings of IEEE WCNC*, 2003.

[16] S.-B. Lee and A. T. Campbell, "HMP: Hotspot mitigation protocol for mobile ad-hoc networks," in *Proceedings of IwQOS*, 2003.

[17] S.-J. Lee and M. Gerla, "Dynamic load-aware routing in ad-hoc networks," in *Proceedings of IEEE ICC*, 2001.

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

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

[20] S. Vutukury and J. J. Garcia-Luna-Aceves, "A simple approximation to minimum-delay routing," in *Proceedings of ACM SIGCOMM*, 1999.

[21] A. Khanna and J. Zinky, "The revised arpanet routing metric," *SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 4, pp. 45–56, 1989.

[22] K. Fall and K. Varadhan, "ns manual," September 2003, http://www.isi.edu/nsnam/ns/.

[23] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proc. ACM MOBICOM*, 1998, pp. 85–97.

[24] R. E. Azouzi, E. Altman, and O. Pourtallier, "Properties of equilibria in routing problems in networks with several types," in *Proceedings of IEEE Conference on Decision and Control (CDC)*, 2002.

[25] T. Roughgarden, "Designing networks for selfish users is hard," in *FOCS '01: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, 2001, p. 472.

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

[27] P. R. Kumar and P. Varaiya, *Stochastic systems: Estimation, identification and adaptive control*. New Jersey: Prentice Hall Inc., 1986.

[28] V. Raghunathan, "Wardrop routing in wireless networks," Master's thesis, University of Illinois at Urbana-Champaign, 2004.

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

[30] R. Russell and H. Welte, "Linux netfilter HOWTO," http://www.netfilter.org/.

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

[32] C. Hedrick, "Routing information protocol," 1988, RFC 1058, http://www.ietf.org/rfc/rfc1058.txt.