# On Self Adaptive Routing in Dynamic Environments *
## — An Evaluation and Design Using a Simple, Probabilistic Scheme

Haiyong Xie†, Lili Qiu‡, Yang Richard Yang†, and Yin Zhang§

†Computer Science Department, Yale University, New Haven, CT 06520
‡Microsoft Research, Redmond, WA 98052
§AT&T Labs – Research, Florham Park, NJ 07932

## Abstract

*Recently we have seen an emergent trend of self adaptive routing in both Internet and wireless ad hoc networks. Although there are previous methods for computing the traffic equilibria of self adaptive routing (*e.g.*, selfish routing), these methods use computationally demanding algorithms and require that a precise analytical model of the network be given. Also, it remains an open question how to design an adaptive routing scheme which ensures convergence to traffic equilibria in practice. In this paper we propose a simple, efficient, distributed probabilistic routing scheme for self adaptive routing in dynamic, realistic environments. Using both analysis and extensive simulations, we show that our scheme can converge to the desired traffic equilibrium (either user-optimal or network-optimal) very quickly. We find that user-optimal routing can achieve very close to optimal average latency in dynamic environments, but such performance often comes at the cost of seriously overloading certain links. To avoid link overloads, we improve adaptive routing by optimizing average user latency and link utilization simultaneously. Our evaluation shows that there is a trade-off between optimizing dual objectives, but the degradation in average latency is only marginal for typical link utilization requirements.*

## 1 Introduction

Recently we have seen an emergent trend of adaptive routing in both Internet and wireless ad hoc networks. In Internet, recent studies [30, 34] have shown that there is inherent inefficiency in IP routing from the user's perspectives. In response to this observation, we have seen a trend to allow end hosts to adaptively choose routes themselves either by using source routing (*e.g.*, Nimrod [14]) or by using overlay routing (*e.g.*, Detour [30] or RON [2]). Such end-to-end route selection is self adaptive, in that it allows end users to select routes to optimize their own performance without considering system-wide criteria [27]. In wireless ad hoc networks, source routing, such as DSR [19], allows wireless users to selfishly select low-latency routes (*e.g.*, [18]), thus resulting in self adaptive routing. This emergence of self adaptive routing poses challenging research questions regarding both design of routing protocols and evaluation of traffic equilibrium behavior.

*Haiyong Xie is supported by NSF grant ANI-0238038. Yang Richard Yang is supported in part by NSF grants ANI-0207399 and ANI-0238038.

In terms of design of routing protocols, it remains an open research question how to design an adaptive routing scheme that ensures convergence to traffic equilibria in realistic settings. Although there are several previous designs, some fundamental issues remain to be addressed. In particular, it is unclear how adaptive routing schemes should probe the network in order to effectively discover efficient routes. If a protocol uses an ineffective probing scheme, high-quality routes may not be discovered. Furthermore, it is unclear how the routing paths should be adjusted in a distributed way and still converge to the optimal paths without causing oscillations.

In terms of evaluation of traffic equilibrium behavior, an important yet challenging question is how to evaluate the performance of self adaptive routing in large networks. While there is previous work on this subject from both theoretical perspectives (*e.g.*, [23, 29]) and practical perspectives (*e.g.*, [27]), several issues remain to be addressed. First, how to efficiently compute traffic equilibria in large networks is an open problem. The traditional algorithms used to derive traffic equilibria (*e.g.*, the Frank Wolfe algorithm [15]) are computationally expensive, and hard to scale to large networks. Second, such algorithms require an analytical model of the network, *e.g.*, link latency functions, be known, which may not be the case. For example, there are no simple analytical expressions for packet delay in wireless networks, where delay is caused by queuing, MAC layer contention, and retransmissions. Moreover, such algorithms can derive equilibria only in a static environment. In reality, networks can be highly dynamic and routing works in a distributed fashion. So it is important to capture the dynamic process.

In this paper, we propose a routing scheme to address the above issues. Our scheme has the following requirements. First, viewed as a mechanism for computing equilibrium, the scheme should be simple and efficient, and thus can be used to evaluate large scale networks. It should not require a precise network model, and thus can be applied in various settings (*i.e.*, both wired and wireless networks). It should also be able to model adaptive routing in a dynamic environment, and be able to capture the potential overhead of adaptive routing. Second, viewed as a protocol for computing network routes, the scheme should be distributed and with low protocol overhead. The design should provide insight in designing adaptive routing protocols with different objectives (*e.g.*, user optimal or network optimal). The protocol should address the key issues of how to probe networks and how to adjust routing paths to guarantee convergence.

Specifically, the routing scheme we propose and study in this paper is a probabilistic routing scheme. On the data path, each packet is forwarded to a neighbor picked according to a probability distribution defined for each destination. Our scheme keeps states only for destinations with active traffic, and thus the state overhead is low. On the control path, a protocol resembling distance-vector routing maintains these probabilities. Upon receiving a routing update from a neighbor, a router updates its routing probabilities.

Our probability update scheme is motivated by the two time-scale stochastic approximation scheme proposed by Borkar and Kumar in [10] and the Q-routing scheme proposed by Littman and Boyan in [25]. However, these two schemes use path-based per-packet feedback and update. As a consequence, their protocols require the cooperation between the routing layer and the transport layer. Furthermore, the routing overhead of their protocols can be high. In comparison, our scheme aggregates routing updates; thus the overhead of our scheme is comparable to that of a load-adaptive routing scheme such as QoS routing. In a low-load environment (*e.g.*, one where the latency of a link is not sensitive to the amount of traffic), our update algorithm is equivalent to the distributed Bellman-Ford algorithm [5].

Our update scheme is also motivated by the distributed gradient projection algorithms, *e.g.*, see [7, 4, 17, 35]. However, these algorithms assume quasi-static environments, namely, the effect of a routing update can be observed immediately, while our scheme allows the effects of routing updates to settle down gradually. Thus our scheme captures network dynamics, and can be both a computing scheme and a realistic routing scheme. Also, our routing scheme is probabilistic and has lower complexity than the previous gradient projection algorithms, which are deterministic.

We use our scheme to implement two types of adaptive routing: user-optimal routing and network-optimal routing. The user-optimal routing converges to the Wardrop equilibria [36] in the Cesaro sense, where at Wardrop equilibria, users do not have incentives to unilaterally change their routes. The network-optimal routing converges to the minimum latency. The former is achieved by having neighbors exchange information about link latency, while the latter is achieved by having neighbors exchange information about *marginal link latency*.

We analyze the convergence of our scheme and evaluate its dynamics through extensive simulations. Our simulations show that our routing scheme responds to traffic stimuli (whether in the form of impulse, or step function, or linear function) and converges to new equilibria very quickly.

Utilizing our efficient routing scheme, we study how to choose routes to optimize end-user performance and link utilization simultaneously. This is an important problem in traffic engineering for adaptive networks [27], because optimizing end-user performance alone sometimes causes link overload (which is undesirable from network operators' point of view), while optimizing network utilization alone may degrade end-user performance. To achieve good user performance without overloading the network, we introduce a link utilization threshold. We update the routing probabilities as before when the utilization of a link is below the threshold; on the other hand, when the utilization of a link is above the threshold, we shift traffic to less loaded links. We evaluate the trade-off between user latency and link utilization and show that the degradation in end-user performance is only marginal for typical link utilization requirements.

In summary, our contributions are as follows.

- First, we develop a routing scheme to achieve user-optimal routing and network-optimal routing. This scheme can be used both as a simple, efficient computing mechanism to compute traffic equilibrium in a dynamic network without a precise analytical model, and as a routing scheme to determine network routes in a distributed way.

- Second, we prove the convergence of our routing scheme, and demonstrate its efficiency and responsiveness using extensive simulations.

- Third, we extend the routing scheme to optimize both end-user performance and link utilization simultaneously. Our evaluation shows that the routing scheme is able to achieve low link utilization while maintaining good end-user performance.

The remainder of this paper is organized as follows. In Section 2, we describe our routing scheme. In Section 3, we analyze the convergence of our scheme for implementing user-optimal routing and network-optimal routing. In Section 4, we describe our evaluation methodology. We present extensive evaluations on the performance and dynamics of the routing scheme in Section 5. We examine how to optimize end-user performance and link utilization simultaneously in Section 6. We discuss related work in Section 7 and conclude in Section 8.

## 2 Routing Scheme

The routing scheme we consider consists of a data-path component and a control-path component. The data-path component is common while the control-path component is different for different routing objectives.

### 2.1 Data path

We first present the data path. Consider node $i$ in the network. Assume that node $i$ has $n(i)$ neighbors, represented by the set $N(i)$.

| | | | *neigh. j* | | |
|---|---|---|---|---|---|
| ⋮ | | | | | |
| *dest. k* | | | $p_j^{ik}$ | | |
| ⋮ | | | | | |

**Figure 1. Forwarding table of node $i$.**

Similar to distance-vector routing, the routing scheme we consider maintains states for active destinations. In other words, the forwarding table of node $i$ consists of one row for each active destination. The active destinations may be all overlay nodes in an overlay network or all active sinks in a wireless ad hoc network. Below when we say destinations, we mean active destinations. Figure 1 illustrates the forwarding table of node $i$.

For destination $k$, node $i$ maintains a routing probability $p_j^{ik}$ for each neighbor $j$. Note that $\sum_j p_j^{ik} = 1$ and $p_j^{ik} \geq 0$. Whenever node $i$ receives a packet destined to node $k$, it forwards the packet to neighbor $j$ with probability $p_j^{ik}$.

Note that this probabilistic routing scheme generalizes the normal Internet routing. More specifically, if only one neighbor has a positive routing probability, the probability must be 1, and thus we have the traditional single-path routing. We can also implement the scheme of OSPF routing with equal-weight splitting by assigning equal probabilities to the neighbor(s) on the shortest path(s) to a given destination.

## 2.2 Control path

The forwarding table of a node is updated by the control path. We consider two implementations of the control path: the first one achieves user-optimal routing, while the second one achieves network-optimal routing.

### 2.2.1 User-optimal routing

User-optimal routing is also called Wardrop routing [36, 1, 3], which is defined in the context of transportation networks as follows [36]: "The journey times on all the routes actually used are equal and less than those which would be experienced by a single vehicle on any unused route." Wardrop routing is especially interesting since it achieves traffic equilibria when each user individually optimizes the performance of its traffic. In other words, at a Wardrop equilibrium, users do not have incentives to unilaterally change their routes.

In the context of computer networks, we define user-optimal routing in a similar way. For a given demand, *i.e.*, a source-destination pair with a given amount of traffic, the routes with positive traffic should have equal latency, no larger than those of the routes not used for this particular source-destination pair.

Note that although user-optimal routing is a multi-path routing scheme, in this situation the paths used have the same latency, so the chance of packet re-ordering is low, and therefore the potential performance penalty at the transport layer, *e.g.*, TCP, is small. In addition, several TCP enhancements have been proposed to cope with packet re-ordering under multi-path routing(*e.g.*, [8] and [38]).

To achieve user-optimal routing, we implement the control path asynchronously, where each node sends its updates to its neighbors after some delay. Note that our protocol is asynchronous. Below we describe the implementation at a given node $i$.

First, for destination $k$, node $i$ maintains the following data items for each neighbor $j$:

- The latency $l_j^i$ of the link from node $i$ to its neighbor $j$. This latency consists of propagation delay and the average queuing delay during the interval between the previous update and the current time. It is also possible that the receiver $j$ maintains $l_j^i$. We will further discuss how to measure $l_j^i$ at the end of this subsection.
- The estimated delay, $L_j^{ik}$, from $i$ to destination $k$ through node $j$. Node $i$ also receives the most recently reported latency $L^{jk}$ from neighbor $j$. Note that this report is generated by node $j$ at sometime in the past. Also note that destination $k$ always reports zero delay to itself.
- The internal probability $q_j^{ik}$ from node $i$ to destination $k$ through neighbor $j$. This probability is used mainly for internal update. As we will show later in Section 3.3, the set of internal probabilities $\{q_j^{ik}\}$ converges (in the Cesaro sense) with probability one to the set of Cesaro-Wardrop equilibria, which is an extension to the notion of a Wardrop equilibrium as introduced in [10].
- The routing probability $p_j^{ik}$ from node $i$ to destination $k$ through neighbor $j$. Note that the routing probability will change after each update and remain the same until next update. As we will show later, the set of routing probabilities $\{p_j^{ik}\}$ are ε-approximate of $\{q_j^{ik}\}$ and thus will converge to the set of ε-approximate Cesaro-Wardrop equilibria with probability one.

After receiving the $n$-th update $L^{jk}$ from neighbor $j$ at time $\sigma_j^{ik}(n)$, node $i$ first updates its delay estimation of $L_j^{ik}$, *i.e.*, the estimated delay to destination $k$ through node $j$. This update has two steps. Node $i$ first computes the value of a new sample:

$$\Delta_j^{ik} = l_j^i + L^{jk}. \tag{1}$$

Then it updates the new delay estimation as:

$$L_j^{ik} = (1 - \alpha(n))L_j^{ik} + \alpha(n)\Delta_j^{ik}, \tag{2}$$

where $\alpha(n) \in [0,1]$ is the *delay learning factor*. Note that we take $\Delta_j^{ik}$ as current delay sample with noise. Therefore, we use (2) to adapt delay estimation and make it robust in the presence of noise. When $\alpha(n)$ is larger, delay estimation is more sensitive to noise; however, $\alpha(n)$ cannot be too small. Otherwise, delay estimation is not responsive enough to network dynamics.

Node $i$ then computes its overall delay estimation $L^{ik}$ to destination $k$ as:

$$L^{ik} = \sum_{j' \in N(i)} p_{j'}^{ik} L_{j'}^{jk}, \tag{3}$$

where $N(i)$ represents the set of node $i$'s neighbors. Node $i$ reports $L^{ik}$ to its neighbors after some delay. This delay is a random value between $T/2$ and $T$ to avoid routing update synchronization, where $T$ is a constant.

Node $i$ then updates its internal routing probabilities as follows. Suppose that this is the $n'$-th time node $i$ updates its routing probabilities. For all neighbors $j$, node $i$ computes:

$$q_j^{ik} = q_j^{ik} + \beta(n')[q_j^{ik}(L^{ik} - L_j^{ik}) + \xi_j^{ik}], \tag{4}$$

where $\beta(n') > 0$ is the *routing learning factor* for the $n'$-th update, and $\xi_j^{ik}$ are i.i.d. random routing vectors distributed uniformly on the unit ball of dimension $N(i)$. The objective of adding the i.i.d. uniform random routing vectors is to add disturbance to avoid non-Wardrop solutions. Note that we use $\beta(n')$ and (4) to smooth out the noise in estimations.

After computing the above routing probabilities, node $i$ projects the routing vector consisting of the routing probabilities to the subspace of $[0,1]^{N(i)}$, where the sum of the routing probabilities is 1. The reason for the projection is to ensure that the vector is a valid probability vector. That is, node $i$ computes the projected value of the new routing probability by solving the following optimization problem:

$$\text{minimize} \quad \sum_{j \in N(i)} (x_j - q_j^{ik})^2 \tag{5}$$

$$\text{subject to} \quad \sum_{j \in N(i)} x_j = 1 \tag{6}$$

$$\text{over} \quad 0 \le x_j \le 1 \text{ for all } j. \tag{7}$$

The computed values of $x_j$ then become the new internal routing vector.

To ensure that the network probes all possible neighbors, node $i$ computes routing probabilities $p$ from the just updated internal routing probabilities $q$ by adding uniform routing probabilities to them:

$$p_j^{ik} = (1-\varepsilon) q_j^{ik} + \frac{\varepsilon}{N(i)}, \tag{8}$$

where $\varepsilon$ is a small constant number.

Finally, a few comments about measuring $l_j^i$. The preceding description specifies that the transmitter (*i.e.*, node $i$) measures $l_j^i$. To achieve this, node $i$ first measures the (fixed) propagation and transmission delay from node $i$ to node $j$. Then during the protocol run, node $i$ keeps track of its transmission queue to determine the queueing delay. An alternative is that the receiver $j$ measures $l_j^i$. Thus instead of sending $L^{jk}$, node $j$ sends $\Delta_j^{ik}$ to node $i$. An interesting advantage of this approach is that it does not need clock synchronization among the nodes. Specifically, in order to measure $l_j^i$, node $i$ time-stamps a packet (with its local clock), and node $j$ computes the link delay of the packet by computing the difference between the time it receives the packet (in its local clock) and that of the time stamp of the packet (when node $i$ receives the packet in its local clock). Thus the estimation of $l_j^i$ may have a constant offset due to the difference between the clocks of nodes $i$ and $j$, assuming constant clock drift. However, it can be checked that the values of $\Delta_j^{ik}$, $L^{ik}$ and $L_j^{ik}$ have an offset which is just the clock difference between node $i$ and the destination, which is independent of node $j$. Thus due to the structure of the routing probability update (*i.e.*, the update of (4) depends only on $L^{ik} - L_j^{ik}$), the offset is canceled and there is no need for clock synchronization. The overhead of this approach is that it needs to time-stamp packets. In the remaining of the paper we use the first approach for measuring $l_j^i$. It is easily extended to use the second approach. Figure 2 summarizes the protocol at node $i$ for destination $k$.

#### 2.2.2 Network-optimal routing

Our second implementation is network-optimal routing, which minimizes total latency over all traffic. In [3], Beckmann, McGuire, and Winsten showed that the total latency is minimized if and only if all traffic travels along paths with minimum marginal cost. In network settings, marginal cost is equivalent to marginal link latency, *i.e.*, $mc_j^i = l_j^i + f_j^i s_j^i$, where $s_j^i$ is the rate of change in the latency from node $i$ to node $j$ at traffic amount $f_j^i$. Compared with user-optimal routing where delays along different paths are minimized, we replace $l_j^i$ with $mc_j^i$ to achieve network-optimal routing. Similar to $l_j^i$, $mc_j^i$ is estimated without knowing the analytical expression.

$\triangleright$ Assume $p_j^{ik}$ is routing probability to neighbor $j$.

**repeat** after some random delay in a range
    send $L^{ik}$ to all neighbors

**repeat** after receiving an update $L^{jk}$ from neighbor $j$
    compute $L_j^{ik}$ for neighbor $j$ using (2)
    compute $L^{ik} = \sum_{j' \in N(i)} p_j^{ik} L_{j'}^{ik}$ using (3)
    update $q_{j'}^{ik}$ and $p_{j'}^{ik}$ for all neighbors $j'$
        update $q$ according to (4)
        do projection on $q$ as in (5)
        compute $p$ as in (8)

**Figure 2. Protocol to implement user-optimal routing.**

## 3 Convergence Analysis

In this section we analyze the convergence of our routing scheme.

### 3.1 Intuition

We first study Figure 3 to gain intuition. The figure shows the phase diagram of a node with two links leading to a given destination. The x-axis is the routing probability of link 2 while the y-axis is the routing probability of link 1. Note that the only valid probability vector will be $p_1 \ge 0$, $p_2 \ge 0$, and $p_1 + p_2 = 1$.

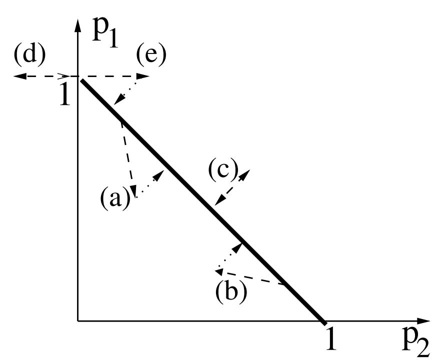



**Figure 3. Illustration of the update on routing probabilities.**

We identify five cases. In the first three cases, both links have traffic. In case (a), link 1 has higher latency. Thus the probability of link 1 is reduced while that of link 2 is increased. The dashed line points to the updated routing probabilities before projection. Then projection along the dotted line brings the routing probabilities back to a valid routing vector satisfying $p_1 \ge 0$, $p_2 \ge 0$, and $p_1 + p_2 = 1$. In case (b), link 1 has lower latency. Thus the probabilities are adjusted and then projected back to the space. In case (c), link 1 and 2 have the same latency and therefore their routing probabilities are not changed. Note that this is a stable state, *i.e.*, the state will no longer change. In the next two cases, one link has no traffic. Without loss of generality, we consider the case that link 1 has all of the traffic while link 2 has no traffic. If the latency of link 1 is smaller than that of link 2 (case (d)), then the algorithm will not change the state of the network. On the other hand, if the latency of link 1 is higher than that of link 2 (case (e)), then the probability of link 2 is increased and the network is on the correct trajectory to the final state.

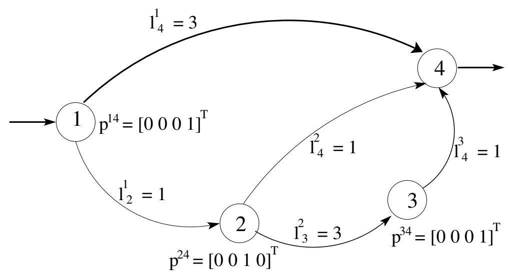



**Figure 4. An example of the need of multi-hop update.**

The above discussion applies to only a single node. A node may stay at a local equilibrium and wait for its downstream nodes to converge. Figure 4 shows an example. In this figure, traffic originates from node 1 to node 4. Routing probabilities, $p^{i4} = [p^{i4}_j]_{j=1,2,3,4}, i = 1,2,3,4$, are labeled along with nodes. Node 1 is at (local) equilibrium (namely case (d) if we think the link from node 1 to node 4 as link 1 in the above discussion). However, node 2 is not at equilibrium (namely case (e) if we think the link from node 2 to node 3 as link 1 in the above discussion). Thus the update of node 2 will eventually cause node 1 to update. Our small random probing will allow node 2 to measure the latency from node 2 to node 4.

### 3.2 Assumptions

We make the following assumptions in our convergence analysis. More technical assumptions on the stochastic process, such as the packet arrival process and network connectivity, can be found in the complete version of this paper [37]. Note that our assumptions are similar to those by Borkar and Kumar in [10], which are standard in the convergence analysis of two time-scale stochastic iterative algorithms [9, 11, 33]. Our delay assumption is similar to that of [35] on asynchronous distributed gradient algorithms.

A1 We assume that the latency of the link from node $i$ to node $j$ is represented by the function $l^i_j(x)$, where $x$ is the load of this link. We assume that this function is continuous, non-decreasing, and bounded. We also assume that the latency functions are chosen such that both the user-optimal and network-optimal settings satisfy the monotone condition to guarantee that the network has a unique Wardrop or optimal equilibrium. Note that although we make assumptions about the properties of the latency functions, our protocol does not need to know the analytical expressions.

A2 We assume that the Feller property holds, *i.e.*, the updates are frequent enough compared with the change rate in the underlying network states. In our protocol, the interval between two updates sent by one node to each of its neighbors is randomly distributed in $[T/2, T]$, where $T$ is a constant. Also, we assume that the number of packets sent in each interval is finite with a constant bound.

A3 We assume that $\{\alpha(n)\}$ used in delay estimation satisfy the following conditions:

$$\forall\, n\ :\ \alpha(n) \geq \alpha(n+1) > 0,$$

$$\sum_n \alpha(n) = \infty, \sum_n \alpha(n)^2 < \infty,$$

and

$$\sum_n ((\alpha(n) - \alpha(n+1))/\alpha(n))^r < \infty$$

for some $r \geq 1$.

A4 We assume that $\{\beta(n)\}$ used in routing probability update satisfy the following conditions:

$$\forall\, n\ :\ \beta(n) \geq \beta(n+1) > 0,$$

$$\sum_n \beta(n) = \infty, \sum_n \beta(n)^2 < \infty,$$

and

$$\sum_n (\beta(n)/\alpha(n))^s < \infty$$

for some $s \geq 1$.

Note that the above assumptions are common for most previous analyses. In particular, the last two assumptions (A3 and A4) are essential to guarantee convergence for stochastic iterative algorithms (see, *e.g.*, [6]). Intuitively, learning factors $\{\alpha(n)\}$ and $\{\beta(n)\}$ represent the step sizes of updating delay estimation and routing probability, respectively. The sum of step sizes ($\sum_n \alpha(n)$ or $\sum_n \beta(n)$) should be unbounded in order to reach equilibrium. On the other hand, the range of $\sum_n \alpha^2(n)$ and $\sum_n \beta^2(n)$ guarantees that the variance of delay estimation and routing probability is bounded. Therefore, diminishing step sizes satisfying the above assumptions guarantees that the stochastic iterative algorithms converge to the solution almost surely. Furthermore, the range of $\sum_n (\beta(n)/\alpha(n))^s$ guarantees that delay estimation has larger step sizes. In other words, the delay estimation should be relatively stabilized before the next routing probability update occurs. Our algorithm uses varying learning factors. It is possible to use constant learning factors as well. For stochastic approximation algorithms with constant learning factors, we refer interested readers to [33].

### 3.3 Convergence analysis

Consider valid routing probability vectors at all nodes. Let $H$ denote the subset: $\{y : y^{ik}_j > 0 \Rightarrow L^{ik}_j = \sum_{j' \in N(i)} y^{ik}_{j'} L^{ik}_{j'}\}$. Let $H_s$ denote the set $\{y \in H : y^{ik}_j > 0 \Rightarrow L^{ik}_j = \min_{j' \in N(i)} L^{ik}_{j'}\}$.

Note that we can use $L$ to denote either the true latency or the maintained state. When $L$ is the true latency, $H_s$ is the set of Wardrop equilibria. When $L$ is the maintained state, which is (properly normalized) time average of the true latency, $H_s$ becomes the set of so called Cesaro-Wardrop equilibria [10]. Under a Cesaro-Wardrop equilibrium, a link is assigned a positive flow only if the (properly normalized) time-averaged delay along the link is minimal.

Below we use $L$ to denote the maintained state and prove the convergence of the internal routing probabilities $\{q_j^{ik}\}$ to $H_s$, the set of Cesaro-Wardrop equilibria. More specifically, we have the following convergence result:

**Theorem 1** *If the assumptions are satisfied, the protocol in Figure 2 converges to the set $H$. Furthermore, the internal routing probabilities $q$ converge in $H_s$ almost surely.*

Due to space limitation, a complete proof of the above theorem is omitted here and can be found in a complete version of this paper [37]. The proof is modeled after [10] but adapted to our link-based aggregated update scheme. To give the readers some intuition about the proof, below we present the major steps of the proof. The readers can skip to the next section without loss of continuity.

In order to prove the above theorem, we need to show that (1) routing probability converges; (2) delay estimation converges under stationary routing probability; and (3) when routing probability and delay estimation converge, the internal routing probabilities converge in $H_s$ almost surely. Thus, our proof consists of three steps as follows.

In the first step of the proof, we show that the routing probabilities converge after the algorithm runs for a sufficiently long time. The major challenge in this step is that we need to find a converging sequence to bound the difference of routing probabilities. By combining and rewriting delay estimation and routing probability update equations, we derive a function of the two learning factors to bound the difference of routing probabilities at different times in a given small time interval. We then apply Borel-Cantelli Lemma and assumptions A1-A4 to show the convergence. In particular, the assumption that the range of $\sum_n(\beta(n)/\alpha(n))^s$ is bounded is crucial in order to apply Borel-Cantelli Lemma.

In the second step of the proof, we prove the convergence of the expected delay with respect to stationary routing probabilities (which is a result of the first step). The intuition behind the proof is that routing probability update has smaller step sizes in order for delay estimation to stabilize before the next routing probability update occurs. Again, the major challenge here is to find a converging bound for the expected delay. We consider delay estimation in a give small time interval. By rewriting the delay estimation equation (2), we derive a closed form expression for the difference of delays, which consists of terms of martingales and bounded delays. We then apply the convergence result in the first step, martingale convergence theorem and Gronwall's Lemma to derive the convergence of delay estimation.

In the last step of the proof, we derive the convergence of internal routing probabilities in $H_s$ based on the results of the previous two steps. Recall that we add the i.i.d. uniform random routing vectors in (4) and (8) to introduce disturbance to escape from non-Wardrop solutions $H \setminus H_s$. Specifically, we show that for any given demand, delays are equalized along paths with positive traffic. The major challenges in this step are to show that internal routing probabilities converge and that the expected delays converge to equalized delays along different paths with positive traffic. To simplify the analysis and make the problem more tractable, we adopt the standard O.D.E. approach to projected stochastic approximation algorithms and consider the continuous version of our discretized routing update scheme. We then prove that internal routing probabilities converge by taking the discrete routing update scheme (4) as an approximation of its continuous version in the O.D.E. approach. Similarly, we show that for any given demand, delays with respect to the converged positive routing probabilities converge to the same value. Afterwards, we show that non-Wardrop solutions in $H \setminus H_s$ are unstable and that the i.i.d. uniform random routing vectors allow us to avoid them.

As for the network-optimal routing scheme, a similar proof can be constructed.

## 4 Evaluation Methodology

We implement the above routing schemes in ns-2 [26] and evaluate their performance and dynamics through extensive simulations. Below we describe the network topologies, traffic demands, and performance metrics used in our evaluation.

**Network Topologies:** Rocketfuel applies effective techniques to obtain fairly complete ISP maps [31]. We use three POP-level topologies published by the authors: ATT, Sprint, and Tiscali. Link capacities of these topologies are derived by scaling up the OSPF weights of the links by a constant factor. To focus on the core of the network, we exclude all of the leaf nodes (*i.e.*, nodes with only one neighbor). Table 1 summarizes the three topologies.

| ISP | #Nodes | #Edges |
|---|---|---|
| ATT | 30 | 126 |
| Sprint | 19 | 100 |
| Tiscali | 32 | 140 |

**Table 1. ISP topologies as measured by Rocketfuel.**

**Traffic Demands:** We consider different ways of generating synthetic traffic demands for our evaluation. One possible approach is based on the *gravity model* [39], which has been shown to provide a reasonable approximation to real traffic demands.

However, we find that when using the gravity model, the network becomes stabilized too quickly to demonstrate evolution dynamics (*i.e.*, how convergence is reached). In addition, under the gravity model, we find it difficult to generate traffic demands that stress the entire network to a sufficient level — in most cases there are only a handful of congested links while most links are under-utilized.

Therefore, in order to better demonstrate the evolution dynamics of convergence, we use another way of generating synthetic traffic demands. We randomly pick two nodes as the source and destination and assign a Pareto traffic flow to them. The traffic rate $r_{ij}$ of the flow from node $i$ to $j$ is 20% of the minimum link bandwidth along the shortest hop-count path from node $i$ to $j$. We continue doing this until all nodes are assigned with an outgoing traffic flow. In our evaluation, the average link utilization ranges from 10% to 20% under the three network topologies we study.

**Traffic Stimuli:** To evaluate how a network converges, we introduce a traffic stimulus as follows. We first feed a given demand matrix to a network to let it converge. Then we introduce a traffic stimulus to the network. The maximum traffic rate of flow from node $i$ to $j$ during the traffic stimulus, $R_{ij}$, is three times the original rate $r_{ij}$. When the traffic rates achieve their maximum values, most low-capacity links are saturated and the average link utilization ranges from 20% to 50% with the network topologies in our evaluation. We apply the following three stimuli commonly used in control theory:

- *Traffic spike*: The traffic rate of each flow originating from node $i$ to $j$ increases suddenly to the highest rate $R_{ij}$ and lasts for only a short period of time; then it decreases to the original level $r_{ij}$. This represents a traffic burst and tests how the system adapts to the disturbance.
- *Step function*: The traffic rate of each flow originating from node $i$ to $j$ increases to $R_{ij}$ and remains at that level afterwards. This represents the transition of traffic levels in the network and tests how the system responds and evolves accordingly.
- *Linear function*: The traffic rate of each flow originating from node $i$ to $j$ increases linearly to the maximum rate $R_{ij}$ over a relatively long period of time, and remains at that maximum level afterwards. This represents the gradual transitions of traffic levels and tests how the system keeps up with the gradually changing traffic.

**Performance Metrics:** We consider the following three performance metrics: average latency, average convergence time, and link utilization.

- The *average latency* reflects the end-to-end user performance, which is the major concern for both user-optimal and network-optimal routing schemes. The average latency is computed for all source-destination pairs, weighted by the amount of traffic flowing from the source to destination.
- The *average convergence time* reflects the speed at which the network stabilizes. We consider a network as converged at time $t$+1 when $\sum_i \|x_{i,t+1} - x_{i,t}\| \leq 5\% \sum_i \|x_{i,t}\|$ where $x_{i,t}$ is the routing matrix at node $i$ during time $t$, and $\|A\|$ is $\ell_2$ norm of matrix A. When the network converges, the latency variance is small, and this can be used as a criterion for convergence.
- The *link utilization* reflects the objectives of network operators, who want to avoid link overloads in their networks.

# 5 Performance Evaluation: End-to-End Latency and Dynamics

## 5.1 Convergence under self-similar traffic demands

We first study the performance of adaptive routing schemes under realistic self-similar traffic demands and with links using drop-tail queues. Figure 5 shows the average latency for the three topologies under user-optimal and network-optimal routing. We make the following observations.

First, both user-optimal and network-optimal routing converge quickly to stable states, with comparable fluctuation during the learning stage.

Second, the performance of user-optimal routing is similar to that of network-optimal routing. This result further supports the observations in [27] that the performance degradation of user-optimal routing is not significant.

Third, network topologies play an important role in the speed of convergence. As shown in the figure, both ATT and Sprint topologies hurtle through the learning stage and converge almost immediately after the simulation starts. In contrast, the Tiscali network experiences a short period of learning stage with high latency before converging to a stable stage with fluctuating average latency.

Fourth, at stable states, the average latency of both routing schemes has small fluctuation (within about 10%). The fluctuation after convergence in both routing schemes are comparable.

Because all three topologies exhibit similar convergence properties, and the Tiscali topology has a distinct learning stage and stabilized stage, in the following subsections we focus on evaluation using the Tiscali topology.

## 5.2 Responsiveness to traffic stimuli

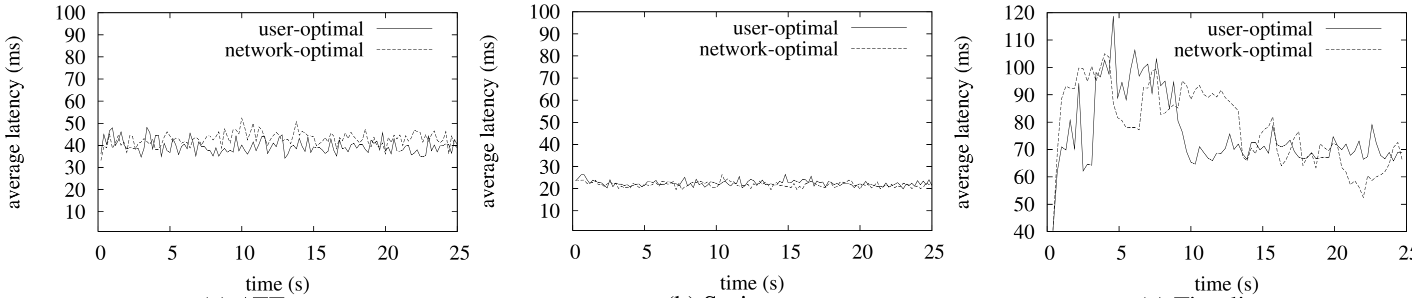
We now evaluate the responsiveness and stability of the routing schemes under different traffic stimuli in the forms of a spike, a step function, and a linear function.

We apply each of the traffic stimuli to the network at 13 second after the network has converged. The highest traffic rate during a stimulus is 3 times the original traffic rate. Both spike and step stimuli increase the traffic level to the highest rate at time 13 second. The spike stimulus maintains the highest traffic rate for 2 seconds and then decreases to the original level, while the step stimulus keeps the highest rate until the simulation ends. Linear stimulus increases the rate gradually to the highest rate from time 13 second to 20 second (with an increase interval of 0.2 ms) and then maintains that rate to the end of the simulation.

Figure 6 shows how the two routing schemes respond to the stimuli in the Tiscali network topology. As we can see, both user-optimal and network-optimal routing schemes react to the stimuli and stabilize very quickly. For the spike stimulus, the network returns to the original stable state almost immediately after the spike disappears. For the step stimulus, the network settles to the new steady state in a very short time. For the linear stimulus, the network follows the increasing traffic closely as the traffic rate increases gradually from 13 second to 20 second; after the linear stimulus stops increasing at 20 second, the network converges quickly.

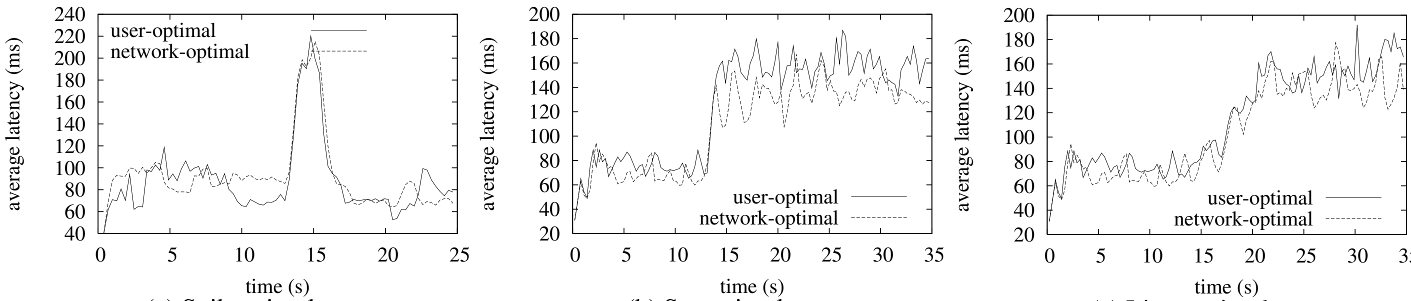
## 5.3 Adaptiveness to link failures

We have shown that our algorithms perform very well under realistic traffic demands and that the algorithms are responsive to various traffic stimuli in the preceding subsections. Next, we evaluate how our algorithms perform when the network topology is changing due to link failures.




(a) ATT (b) Sprint (c) Tiscali

**Figure 5. Dynamics of user-optimal and network-optimal Routing.**




(a) Spike stimulus (b) Step stimulus (c) Linear stimulus

**Figure 6. Responsiveness of routing schemes.**

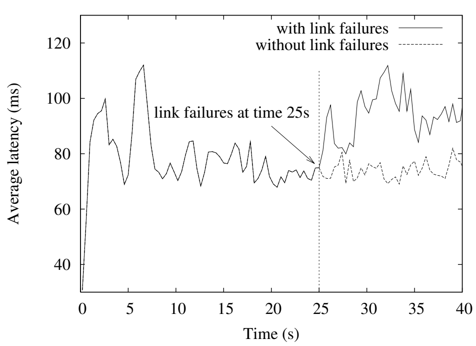



**Figure 7. Adaptiveness to link failures.**

When evaluating the adaptiveness of our algorithms to link failures, we run our algorithms with self-similar traffic demands and different network topologies until they converge; we then randomly disconnect 5% of the links from the network (the network is still connected).

Figure 7 shows the responsiveness of our user-optimal routing algorithm to link failures with the Tiscali network topology. The link failures occur at 25 second. We observe that the average latency increases immediately after the link failures. After a short period of time, the routing algorithm starts to converge. The converged average latency after the link failures is about 25% higher than that without link failures. The result shows that the algorithm adapts well to dynamic environments with link failures. The results using other routing algorithms and topologies are consistent.

## 5.4 Advantage over shortest hop-count routing

In this subsection, we evaluate the benefits of user-optimal routing over shortest hop-count routing (*i.e.*, rtProtoDV in ns-2) under both self-similar traffic and traffic stimuli. Figure 8 summarizes the results. As we can see, user-optimal routing out-performs shortest hop-count routing by 20% - 30% in most cases. This is consistent with our expectation, since shortest hop-count routing minimizes hop-count instead of user latency.

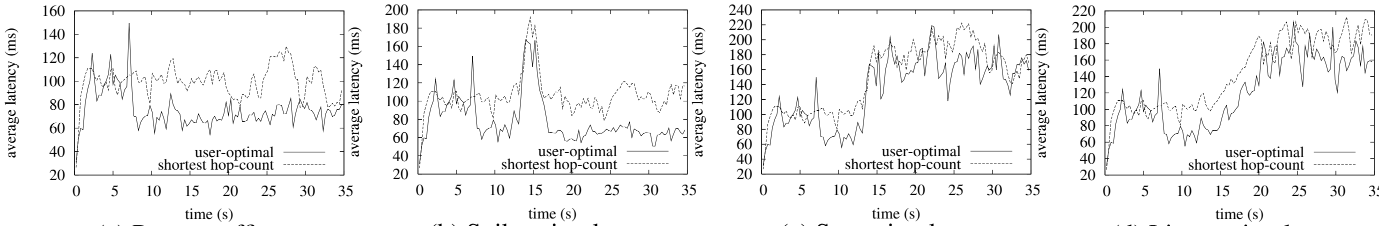
## 5.5 Improvement of convergence and responsiveness

In this subsection, we consider the following issue: how to make the network converge faster and be more responsive.

Recall that we implement the routing update in two steps. First, a node updates its delay estimations to all of the destinations using exponential averaging. The parameter used in exponential averaging is called the delay learning factor. Next, the node updates its routing probability using the results in the previous step; the parameter used in this step is called the routing learning factor. Note that in our analysis we use decreasing sequences while in our evaluation we use fixed values. We evaluate the impact of various control parameters, namely the delay learning factor, the routing learning factor, and the period of updating routing probability vectors.

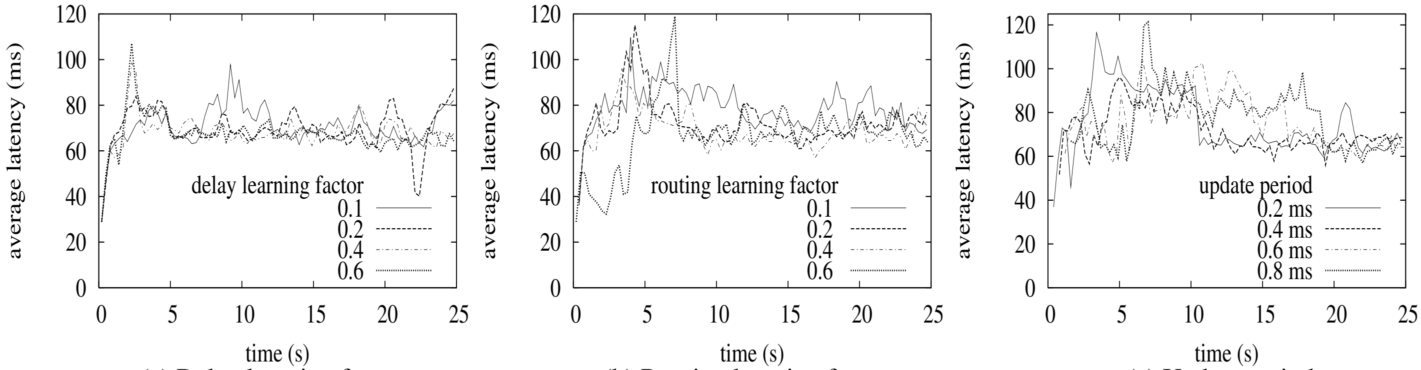
**Delay Learning Factor:** We first evaluate the effect of the delay learning factor, which is used in updating delay estimations. Recall that we adopt delay learning factor $\alpha(n)$ and Equation (2) to smooth out noise introduced in estimating link latency and marginal link latency.

Figure 9(a) summarizes the results. We find that a large delay learning factor leads to faster convergence but high fluctuation. In comparison, a small learning factor makes the algorithm not responsive to network dynamics, resulting in slow convergence. It is very important to choose appropriate




(a) Pareto traffic (b) Spike stimulus (c) Step stimulus (d) Linear stimulus

**Figure 8. Comparison between user-optimal and shortest path routing.**




(a) Delay learning factor (b) Routing learning factor (c) Update period

**Figure 9. Impacts of different parameters.**

values for delay learning factor in order for the routing algorithm to be both robust to noise and responsive to network dynamics. Our results suggest that a preferable learning factor would range between 0.4 and 0.6.

**Routing Learning Factor:** Figure 9(b) shows how the routing learning factor affects the convergence speed. The network starts to converge more quickly when using eager learning (with a higher learning factor) than lazy learning. However, we also observe that a higher factor leads to more fluctuation, *e.g.*, the curve corresponding to a factor of 0.6 has more fluctuation compared to that of 0.2 and 0.4. A preferable learning factor would range between 0.2 and 0.4. Note that similar to delay learning factor, a large routing learning factor makes the algorithm sensitive to noise, and leads to undesired fluctuations after convergence; and a small factor makes the algorithm less responsive and slows down convergence rate, as shown in the figure.

**Period of Routing Update:** The last parameter we evaluate is the period of updating the routing probability vectors. As Figure 9(c) shows, an eager update scheme (with shorter update period) leads to faster convergence and less fluctuations compared with lazy update. We have the same observation when evaluating the impact of larger update periods; therefore, only the results of shorter update periods are presented here in the interest of space. We observe that eager update scheme is preferable for fast convergence and high responsiveness. We also note that a shorter update period introduces more control traffic between neighboring routers. However, the overhead is low and localized, and the routing schemes with a shorter update period still scale very well.

In summary, the three parameters we evaluate have important impacts on convergence and fluctuation. A relatively short update period and medium routing learning factor can be combined to provide enough sensitivity to the dynamics of the network. Empirical evaluation can help serve as basis for setting values for these parameters.

### 5.6 Effects of transition process and probing probabilities on path lengths

The two routing schemes studied in this paper are both probabilistic routing. After our schemes converge, if there are no probing probabilities, no packet will visit the same node twice. More specifically, after convergence and without the small probing probabilities, a node will not assign a positive routing probability to a neighbor with a higher latency. Assuming no link has zero latency (which is true in almost all scenarios), we have that no packet can visit the same node twice. However, during the transition process and with the small probing probabilities, it is possible for a packet to take detours; therefore a packet may visit the same node multiple times.[1] Note that even if a packet visits the same node multiple times, it will not loop forever because of the probabilistic nature of the scheme.

In this subsection, we quantify the effects of the transition process and the probing probabilities on path lengths as fol-

[1]The routing schemes proposed in [4, 17] guarantee loop-freeness, *i.e.*, no node is visited twice by a packet. However, they require synchronous update, which may slow down convergence, especially in large networks. It is also possible to avoid any loops during the convergence process by imposing an acyclic structure on the network nodes, *e.g.*, the shortest hop-count path tree to a given destination. However such restrictions may reduce the search space and thus cause the network to converge to less efficient equilibria, *e.g.*, only short paths are searched. Overall, trade-offs among convergence speed, efficiency of converged equilibrium, and loops during transition process need to be made.

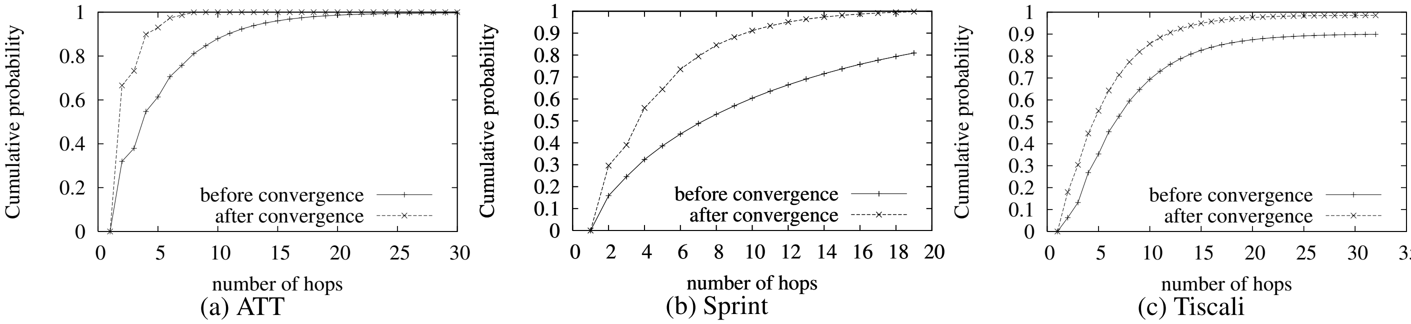



**Figure 10. Cumulative probability of packets being delivered in a number of hops.**

lows. Consider a given destination $d$. We take all of the routing probability vectors at all of the nodes at time $t$ for destination $d$. These vectors altogether represent a snapshot of the state of the network at that time. We construct a Markov transition matrix $P_d$ as follows. The $i$-th row of the matrix $P_d$ is the routing probability vector for destination $d$ at node $i$. Note that the $d$-th row of $P_d$ will have a value 1 at the $d$-th entry and 0 otherwise, indicating that node $d$ is an absorbing state. Consider a source node $s$. Let $\pi_s$ be the row probability vector with 1 for source node $s$ and 0 for all other nodes. Then according to the property of Markov matrices [21], the probability that a packet starting from source node $s$ arrives at the destination node $d$ in $h$ hops is the $d$-th entry of the row vector $\pi_s P_d^h$.

Figure 10 shows the results of the cumulative distribution of the lengths of routing paths. In our evaluations, we have a constant probing probability 0.08 at all nodes. We randomly choose a source node $s$ and a destination node $d$. We take snapshots of the complete routing probabilities $P_d$ at 5 second after simulation starts (the network has not converged yet) and at a random time after the network converges. We then compute the cumulative distribution of the number of hops that a packet sent from $s$ to $d$ traverses in the network. As we can see, the number of hops a packet traverses from the source to the destination is significantly lower after the network has converged. For example, before the network converges, the probability of delivering packets within 10 hops is 88%, 60%, and 70% in ATT, Sprint, and Tiscali topologies, respectively; in comparison, the corresponding probabilities increase to 100%, 91%, and 86% after the network converges.

## 6 Optimization for both User Latency and Link Utilization

So far, we have considered routing for optimizing latency. As shown in [27], optimizing the average user latency alone sometimes causes link overload, which is undesirable from the network operators' point of view [16]. Ideally, we would like to achieve low user latency while avoiding link overload. In this section, we study how to optimize routing for both metrics simultaneously.

Our method is to introduce a link utilization threshold. Whenever a link's utilization exceeds the threshold, the routing scheme will shift some traffic from the highly utilized link to other under-utilized links. This is done by updating the routing probability vector for the corresponding destination. When all of the outgoing links experience higher utilization than the threshold (*i.e.*, there are no under-utilized links), the routing scheme distributes the traffic evenly among all outgoing links.

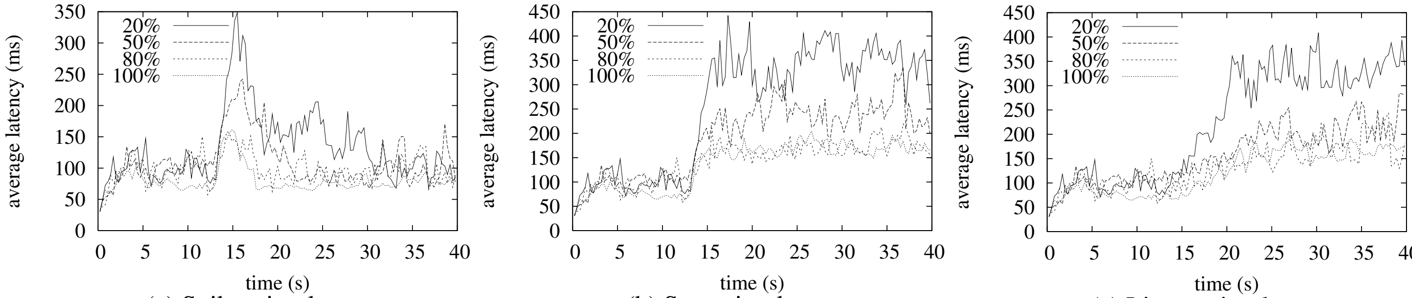
In our experiments, we use several utilization thresholds: 20%, 50%, 80%, and 100%. We apply spike, step, and linear stimuli to evaluate the responsiveness and stability of different routing schemes. Figure 11 shows the average latency of user-optimal routing scheme with the Tiscali network topology.

We make the following observations. First, both routing schemes are very responsive to the traffic stimuli: they closely track the changes in traffic and become stabilized as soon as the traffic stops changing. Second, comparing the results with those in Figure 6, which are obtained solely by optimizing user latency, we observe that by trying to minimize the maximum link utilization, the network experiences higher latency; this is especially clear when the utilization threshold is below 20%. In comparison, the latency increase is only marginal when we increase link utilization threshold to 50% or higher. This is because when the threshold is high, only a few links are above the threshold; as a result, only a small portion of traffic needs to be re-routed through less loaded paths.

Next we compare the user- and network-optimal routings. Figure 12 summarizes the results. As it shows, user-optimal routing and network-optimal routing exhibit similar latency and convergence speed, both adapting quickly to changes in traffic.

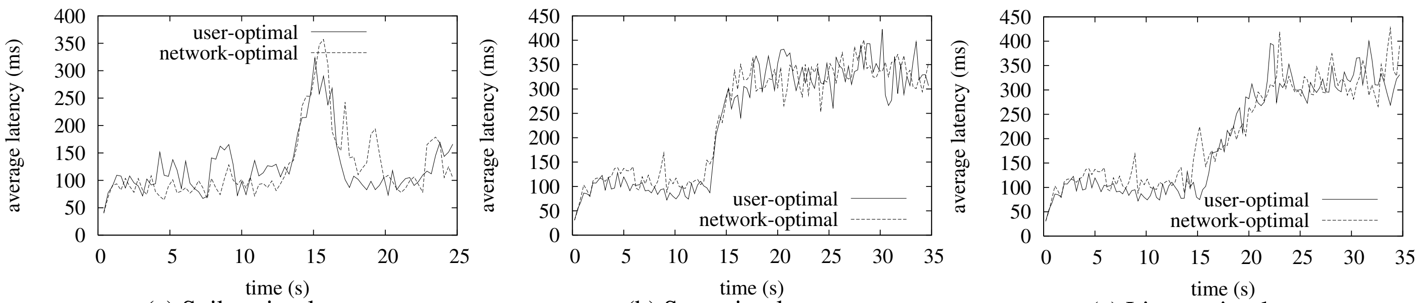
## 7 Related Work

User-optimal routing achieves Wardrop equilibrium [36]. In [3, 36], the authors showed that uncooperative traffic can be modeled as network flows, and the flow paths between any source and destination pair have the same latency. Based on the observation that such an equilibrium flow is an optimal solution to a related convex program, Beckmann *et al.* [3] proved the existence and uniqueness of traffic equilibrium for user-optimal routing. Most previous studies have been concerned with user-optimal routing with an infinite number of users, *i.e.*, infinitesimal demand. In [22], Korilis, Lazar, and Orda considered a finite number of users and studied the conditions for the existence of Nash equilibria. They showed that there may exist multiple Nash equilibria and that although it is possible to use Rosen's Diagonal Strict Concavity [28] to establish uniqueness, this condition generally does not apply. In [12], Boulogne, Altman, Pourtallier, and Kameda studied the conditions for the existence of a Nash equilibrium




(a) Spike stimulus (b) Step stimulus (c) Linear stimulus

**Figure 11. User-optimal routing combined with load optimization: average user latency for various link utilization thresholds.**




(a) Spike stimulus (b) Step stimulus (c) Linear stimulus

**Figure 12. Comparison between user-optimal and network-optimal routing schemes: average latency when the link utilization threshold is 20%.**

in a mixed network, *i.e.*, a network where some users control a substantial amount of traffic and other users generate infinitesimal amount of traffic.

Network-optimal routing in a centralized setting has been studied previously, and many optimization techniques have been proposed. For a survey, please see [5]. There are also previous distributed algorithms for computing optimal traffic equilibrium, *e.g.*, see [7] for a complete survey; however, they do not use the probabilistic routing scheme.

The authors in [27] study the performance of self adaptive routing and its impact on traffic engineering under traffic equilibria.

Our routing scheme is based on reinforcement learning [20]. In [13, 25], Littman and Boyan first applied reinforcement learning to routing and proposed the Q-routing scheme. The Q-routing scheme is further revised in [24, 32]. Their scheme is per-packet based, however, and only supports single path routing. The work closest to ours is [10] by Borkar and Kumar. In this significant work, they formally prove the convergence of their scheme and our proof models after theirs. We avoid per-packet feedback to have a more efficient distributed implementation.

## 8 Conclusion and Future Work

In this paper, we develop routing schemes to achieve user-optimal and network-optimal routing. Viewed as a mechanism for computing equilibrium, our scheme is simple and efficient; thus it can be used to evaluate the performance of large scale networks. Moreover, it does not require analytical network models, and is able to model user-optimal and network-optimal routing in a dynamic environment. It is also able to capture the potential overhead of the routing convergence process. Viewed as a protocol for determining routes in a network, our scheme is simple and distributed; and the scheme has low protocol overhead. We analyze the convergence of the routing scheme, and demonstrate its efficiency and responsiveness through extensive simulations.

In addition, we adapt the routing scheme to optimize end-user performance and link utilization simultaneously. We evaluate the trade-off between the two objectives and show that the degradation in end-user performance is only marginal for typical link utilization requirements.

There are a number of avenues for future work. First, by having routers compute user-optimal routing, we know that the users will have no incentives in deviating from the routing protocol. However, when the routers also consider traffic engineering objectives (*i.e.*, jointly optimize for both user latency and link utilization), the resulting routing will deviate from user-optimal routing. A thorough understanding of the interactions between user's routing incentives and traffic engineering is needed. Second, it remains an interesting question how to design routing protocols and compute traffic equilibria when users are optimizing for other end-to-end performance metrics, such as loss and throughput. In order to optimize latency, our scheme reduces contention, and thus has the potential to increase throughput. Also, the probabilistic routing scheme investigated here may offer a way to efficiently compute traffic equilibria for such metrics.

## Acknowledgments

We thank Sekhar Tatikonda for insightful discussions. We also thank Theodore Jewell and Sheng Zhong for many discussions.

## References


[1] E. Altman, T. Boulogne, R. E. Azouzi, and T. Jimenez. A survey on networking games. *Telecommunication Systems*, Nov. 2000.

[2] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of the 18th Annual ACM Symposium on Operating Systems Principles*, Banff, Canada, Oct. 2001.

[3] M. Beckmann, C. B. McGuire, and C. B. Winsten. *Studies in the Economics of Transportation*. Yale University Press, 1956.

[4] D. P. Bertsekas, E. M. Gafni, and R. G. Gallager. Second derivative algorithms for minimum delay distributed routing in networks. *IEEE Transactions on Communications*, COM-32(8):911–919, 1984.

[5] D. P. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, Second Edition, 1992.

[6] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic Programming*. Athena Scientific, 1996.

[7] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[8] S. Bohacek, J. Hespanha, J. Lee, C. Lim, and K. Obraczka. TCP-PR: TCP for persistent packet reordering. In *Proceedings of the IEEE 23rd International Conference on Distributed Computing Systems*, May 2003.

[9] V. Borkar. Stochastic approximation with two time scales. *Systems and Control Letter*, 29:291–294, Feb. 1997.

[10] V. Borkar and P. R. Kumar. Dynamic Cesaro-Wardrop equilibration in networks. *IEEE Transactions on Automatic Control*, 48(3):382–396, Mar. 2003.

[11] V. Borkar and S. Meyn. The O.D.E. method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control*, 38(2):447–469, 2000.

[12] T. Boulogne, E. Altman, O. Pourtallier, and H. Kameda. Mixed equilibrium for multiclass routing games. *IEEE Transactions on Automatic Control*, 47(6):903–916, June 2002.

[13] J. A. Boyan and M. L. Littman. *Advances in Neural Information Processing Systems*, volume 6, chapter Packet routing in dynamically changing networks: A reinforcement learning approach, pages 671–678. Morgan Kaufmann, San Francisco, CA, 1993.

[14] I. Castineyra, N. Chiappa, and M. Steenstrup. *The Nimrod Routing Architecture, RFC 1992*, Aug. 1996.

[15] M. Florian and D. Hearn. *Network Routing*, chapter 6, Network equilibrium models and algorithms, pages 485–550. Elsevier Science, 1995.

[16] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communication Magazine*, Oct. 2002.

[17] R. G. Gallager. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communications*, COM-25(1):73–85, 1977.

[18] P. Gupta and P. R. Kumar. A system and traffic dependent adaptive routing algorithm for ad hoc networks. In *Proceedings of IEEE 36th Conference on Decision and Control*, San Diego, CA, 1997.

[19] D. B. Johnson and D. A. Malt. *Mobile Computing*, chapter Dynamic Source Routing in Ad Hoc Wireless Networks, Chapter 5, (Tomasz Imielinski and Hank Korth, eds.). Kluwer Academic Publishers, 1996.

[20] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[21] S. Karlin and H. M. Taylor. *A First Course in Stochastic Processes*. Academic Press, second edition, 1975.

[22] Y. A. Korilis, A. A. Lazar, and A. Orda. Architecting noncooperative networks. *IEEE Journal of Selected Areas in Communications*, 13(7):1241–1251, Sept. 1995.

[23] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, 1999.

[24] S. Kumar. Confidence based dual reinforcement Q-routing: an on-line adaptive network routing algorithm. Technical Report AI98-267, Department of Computer Sciences, The University of Texas, Austin, Texas, U.S.A., 1998.

[25] M. L. Littman and J. A. Boyan. A distributed reinforcement learning scheme for network routing. In *Proceedings of the 1993 International Workshop on Applications of Neural Networks to Telecommunications*, pages 45–51, Hillsdale NJ, 1993.

[26] Network Simulator – ns-2. `http://www.isi.edu/nsnam/ns/`.

[27] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in Internet-like environments. In *Proceedings of ACM SIGCOMM '03*, Karlsruhe, Germany, Aug. 2003.

[28] J. B. Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica*, 33:520–534, July 1965.

[29] T. Roughgarden and E. Tardos. How bad is selfish routing? *Journal of ACM*, 49(2):236–259, 2002.

[30] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: a case for informed Internet routing and transport. In *IEEE Micro*, volume 19, pages 50–59, Jan. 1999.

[31] N. Spring, R. Mahajan, and D. Wetherall. Rocketfuel: An ISP topology mapping engine. Available from `www.cs.washington.edu/research/networking/rocketfuel/`.

[32] D. Subramanian, P. Druschel, and J. Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In *IJCAI (2)*, pages 832–839, 1997.

[33] V. Tadic and S. Meyn. Asymptotic properties of two time-scale stochastic approximation algorithms with constant step sizes. In *Proceedings of the 2003 American Control Conference*, June 2003.

[34] H. Tangmunarunkit, R. Govindan, S. Shenker, and D. Estrin. The impact of routing policy on Internet paths. In *Proceedings of IEEE INFOCOM '01*, Anchorage, AK, Apr. 2001.

[35] J. N. Tsitsiklis and D. P. Bertsekas. Distributed asynchronous optimal routing in data networks. *IEEE Transactions on Automatic Control*, 31:325–332, 1986.

[36] J. G. Wardrop. Some theoretical aspects of road traffic research. In *Proceedings of the Institute of Civil Engineers, Part II*, volume 1, pages 325–378, 1952.

[37] H. Xie, L. Qiu, Y. R. Yang, and Y. Zhang. On self adaptive routing in dynamic environments — an evaluation and design using a simple, probabilistic scheme. Technical Report YALEU/DCS/TR1289, Computer Science Department, Yale University, May 2004.

[38] M. Zhang, B. Karp, S. Floyd, and L. Peterson. RR-TCP: A reordering-robust TCP with DSACK. In *Proceedings of ICNP 2003*, Nov. 2003.

[39] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *Proceedings of ACM SIGMETRICS*, June 2003.