Yale University
Department of Computer Science

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

Haiyong Xie
Yale University

Lili Qiu
Microsoft Research

Yang Richard Yang
Yale University

Yin Zhang
AT&T – Research

YALEU/DCS/TR-1289
May 2004

On Self Adaptive Routing in Dynamic Environments

— An Evaluation and Design Using a Simple, Probabilistic Scheme

Haiyong Xie
Yale University

Lili Qiu
Microsoft Research

Yang Richard Yang
Yale University

Yin Zhang
AT&T – Research

Abstract

Recently we have seen an emergent trend of self adaptive routing in both Internet and wireless ad hoc networks. Although there is previous work on studying the traffic equilibria of self adaptive routing (e.g., selfish routing), the previous 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 scheme that ensures convergence to a traffic equilibrium in practice. In this paper we propose a simple, distributed, probabilistic routing scheme for scalable, efficient, 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 [34, 38] have shown that there is inherent inefficiency in IP routing from the user's perspective. In response to these observations, 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 [34] or RON [3]). 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 ramifications [29]. In wireless ad hoc networks, DSR [20] allows wireless users to selfishly select low-latency routes (*e.g.*, [19]), thus resulting in self adaptive routing. This emergence of adaptive routing, in particular, self adaptive routing, poses challenging research questions in both design and evaluation.

In terms of evaluation, a particular important and challenging question is how to evaluate the performance and impacts of self adaptive routing in a large network. This question is particularly important given the increasingly wide deployment of self adaptive traffic and the theoretical work (*e.g.*, [24, 33]) which shows that the worst-case performance degradation of self adaptive routing (also called selfish routing or user-optimal routing in the literature) can be unbounded. Motivated by the theoretical worst-case analysis, researchers start to study the performance of self adaptive routing in Internet-like environments. In order to carry out the evaluations, the authors of [29] compute traffic equilibria using the Frank-Wolfe based algorithms [16]. However, these algorithms are computationally expensive, and cannot scale to large networks. In addition, the computational method requires that a precise analytical model of the network, *e.g.*, link latency functions, be known. However, analytical expressions may not be always available. For example, there are no simple analytical expressions for the queueing delay in wireless networks where delay is due to MAC layer 802.11 contention and retransmission, or in the Internet with active queue management schemes. Thus the performance of self adaptive routing in important scenarios cannot be evaluated. Moreover, the computational method can only capture the equilibrium of a static environment. In reality, networks can be highly dynamic and routing works in a distributed fashion. So far there is no model to capture the dynamic process.

In terms of design, it remains an open research question how to design an adaptive routing scheme that ensures convergence to traffic equilibria in realistic settings. Although previous designs exist, some fundamental design issues are still not 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 but still converge to the optimal routing paths without causing oscillations.

In this paper, we propose a routing scheme to address both the evaluation and the design 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 adaptive networks. It should not require a precise network model, and thus can be applied in various settings such as both the Internet 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 provable 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. This probability distribution is defined for each destination. Our scheme keeps states only for destinations with active traffic, and thus can be applied to many settings (*e.g.*, the direction diffusion approaches). On the control path, a protocol resembling distance-vector routing maintains these probabilities, *i.e.*, routing neighbors exchange aggregated routing updates. Upon receiving a routing update from a neighbor, a router computes its new 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 [28]. However, these two schemes use path-based per-packet feedback and update; therefore their protocol overhead 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.

Our update scheme is also motivated by the distributed gradient projection algorithms, *e.g.*, see [8, 5, 18, 39]. 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 to reduce complexity while the previous gradient projection algorithms are deterministic.

We use our scheme to implement two types of adaptive routing: user-optimal routing and network-optimal routing, where the former converges to the Wardrop equilibria [40], and the latter converges to the minimum latency. The user-optimal routing is achieved by having neighbors exchange information about link latency, while the network-optimal routing is achieved by having neighbors exchange information about *marginal link latency*.

We formally 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 [29], because optimizing end-user performance alone sometimes results in overloaded links (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 formally 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 set $N(i)$.

Dest.		j	
⋮			
k		p_j^{ik}	
⋮			

Figure 1. Forwarding table of node i.

Similar to distance-vector routing, the routing scheme we consider maintains a state for each active destination. 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 to 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 [40, 4, 2], which is defined in the context of transportation networks as follows [40]: “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, smaller than those of the routes not used for this particular source-destination pair. As an example, the demand in a sensor network running directed diffusion may be the periodical reports generated at the specified rate. (Although user-optimal routing is a multi-path routing scheme, in this situation the paths used have the same latency, so out-of-order packet arrival is not very likely and therefore the potential performance penalty at the transport layer, *e.g.*, TCP, is small.)

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 and there is no need for clock synchronization. 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 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, it is this set of probabilities that will converge to Cesaro-Wardrop equilibrium.
- 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.
- The latency l_j^i of the link from node i to its neighbor j. This latency is the average of the sum of propagation delay and queuing delay during the time between the previous update and the current time.
- The most recently reported latency L^{jk} from neighbor j. Note that this report is generated by node j at some time in the past.

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 Δ_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 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 ξ^{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:

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

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

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

The computed x_j then becomes 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:

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

where ϵ is a small constant number.

Figure 2 summarizes the protocol at node i.

```
▷ 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.

2.2.2 Network-optimal routing

Our second implementations is network-optimal routing, which minimizes total latency over all traffic. In [4], 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 minimize marginal link latency and achieve network-optimal routing. Note that mc_j^i can be estimated without knowing the analytical expressions.

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 network with two nodes connected by two direct links from the source to the destination. The x-axis is the routing probability of link 1 while the y-axis is the routing probability of link 2. Note that the only valid probability vector will be $q_1 \geq 0$, $q_2 \geq 0$, and $q_1 + q_2 = 1$.

In case (a), the latency of link 1 is lower while that of link 2 is higher; thus the vertical 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 $q_1 \geq 0$, $q_2 \geq 0$, and $q_1 + q_2 = 1$. In case (b), the latency of link 1 is higher than that of link 2; 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 they stay on the line. Note that this is a stable state, *i.e.*, the state will no longer change. In cases (d) and (e), link 1 has no traffic and all traffic goes to link 2. Although this is a stable state according to our update rule, a slight disturbance may or may not change the state, depending on whether or not the state is globally stable. If the latency of link 2 is smaller than that of link 1 (case (d)), then a slight disturbance will not change the state of the network. On the other hand, if the latency of link 2 is higher than that of link 1 (case (e)), then the probability of link 1 is increased and the network is on the correct trajectory to the final state.

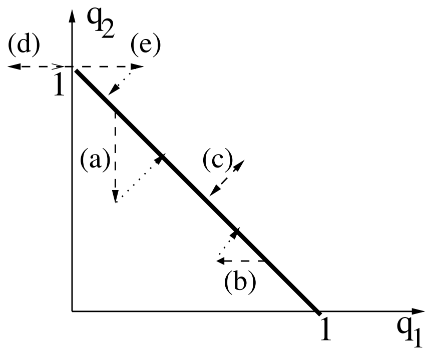


Figure 3. Illustration of the update on routing probabilities.

3.2 Assumptions

We make the following assumptions in our convergence analysis. More technical assumptions on the stochastic process, such as the continuity and the Feller properties, can be found in the more completely version of this paper [41]. 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, 37]. Our update delay assumption is similar to that of [39] on asynchronous distributed gradient algorithms.

A1 We assume that the latency at each link is a continuous, non-decreasing, and bounded function of the load on the link. In particular we assume that the latency functions are chosen such that both the user-optimal and network-optimal settings satisfy the monotone condition; thus the network has a unique Wardrop or optimal equilibrium. Note that although we make assumptions about the properties of the link 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 the $\alpha(n)$ factors 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 the $\beta(n)$ factors 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.*, [7]). Intuitively, delay learning factor $\alpha(n)$ and routing learning factor $\beta(n)$ 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 probability update is bounded. Therefore, diminishing step sizes satisfying the above assumptions guarantee 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 [37].

3.3 Convergence analysis

Let H denote the set $\{y : y_j^{ik} > 0 \Rightarrow L_j^{ik} = \sum_{j'} y_{j'}^{ik} L_{j'}^{ik}\}$. Let H_s denote the set $\{y \in H : y_j^{ik} > 0 \Rightarrow L_j^{ik} = \min_{j'} L_{j'}^{ik}\}$. Note that here we abuse notation a bit because we use L to denote the true function instead of maintained states or samples. It is clear from the definitions that H_s is what we need while H is a larger set. Figure 4 shows an example of network configuration that is in H but not in H_s. In this figure, traffic originates from node 1 to node 4. Routing probabilites, $p^{i4} = [p_j^{i4}]_{j=1,2,3,4}, i = 1, 2, 3, 4$, are labeled along with nodes. Apparently, the state of the network shown in the figure is in H but not in H_s.

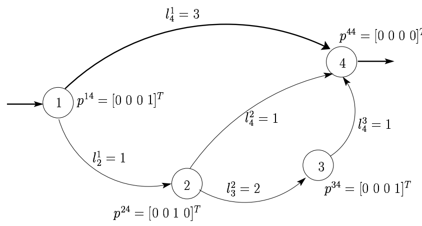

$l_4^1 = 3$
$p^{44} = [0\ 0\ 0\ 0]^T$
4
1
$p^{14} = [0\ 0\ 0\ 1]^T$
$l_4^2 = 1$
$l_4^3 = 1$
$l_2^1 = 1$
3
2
$l_3^2 = 2$
$p^{34} = [0\ 0\ 0\ 1]^T$
$p^{24} = [0\ 0\ 1\ 0]^T$

Figure 4. An example of network configuration in H but not in H_s.

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.*

Please see the appendix for our complete proof of the above theorem. The proof is motivated by [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 a sufficiently long time. The major challenge in this step is that we need 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 to the proof by applying 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 Lemma to derive the convergence of delay estimation.

In the last step of the proof, we derive the convergence of internal routing probability in H_s based on the results of the previous two steps. 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 probability converges and that the expected delays converge to equalized delays along different paths with positive traffic. In order to make the challenge more attackable and simplify the analysis, 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 probability converges by taking the discrete routing update scheme (4) as an approximate 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 prove by contradiction that the paths with zero traffic have higher delay.

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 [1] 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 [35]. 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* [42], 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 its 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 the synthetic traffic demands. We randomly pick two nodes as the source and destination and assign a Pareto traffic flow to them. The traffic rate of flow from node i to j, r_{ij}, is 20% of minimum link bandwidth along the shortest hop-count path from node i to j. We continue doing this until all of the 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 Stimulus: To evaluate how a network converges, we introduce 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 of low-capacity links are saturated and the average link utilization ranges from 20% to 50% with the network topologies in our evaluation. We consider the following three stimulus models 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 in the network 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 ℓ_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 User latency 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 [29] that the performance degradation of user-optimal routing is not significant. Note that the conclusion in this paper is based on realistic demands under realistic queuing, while the result from [29] is based on simple analytical expressions.

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 types has small fluctuation (within about 10%). The fluctuation after convergence in both routing schemes are comparable.

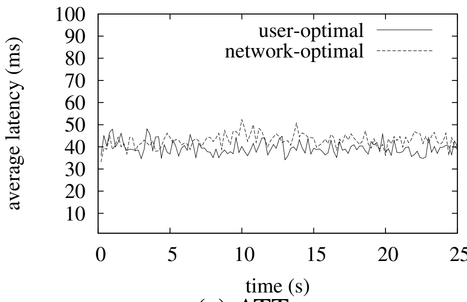
Because all three topologies exhibit similar convergence properties, and the Tiscali topology has a distinct learning stage and stabilized stage, in the following subsections we focus on evaluation using the Tiscali topology.

5.2 User latency under traffic stimulus

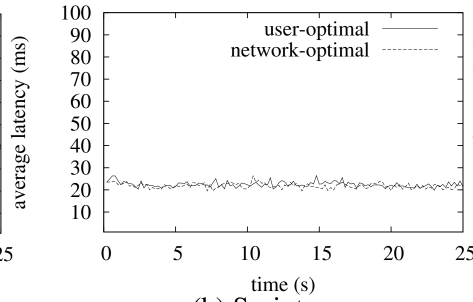
We now evaluate the responsiveness and stability of the routing schemes under different traffic stimuli in the forms of spike, step function, and linear function.

We apply the traffic stimulus to the network at 13 second after the network has converged. The highest traffic rate during the stimulus is 3 times the original traffic rate. Both the spike and the 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 0.2ms) and then maintains that rate to the end of simulation.

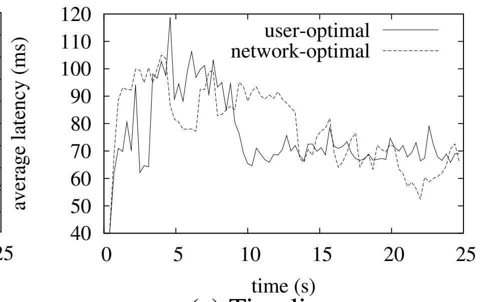
Figure 6 shows how the two routing schemes respond to the stimuli in the Tiscali network topology. As we can see, both the user-optimal and the network-optimal routing schemes react to the stimuli and stabilize very quickly. For traffic bursts such as spikes, the network returns to the original stable state almost immediately after the spike disappears. For a step stimulus, the network begins to stabilize in a very short time. For gradually increasing traffic levels (*i.e.*, a linear function), the network follows the changing traffic closely and starts to converge as soon as the traffic rate stabilizes.



(a) ATT

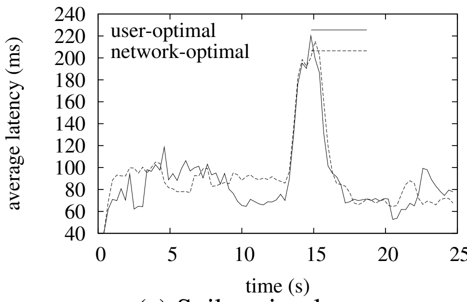


(b) Sprint

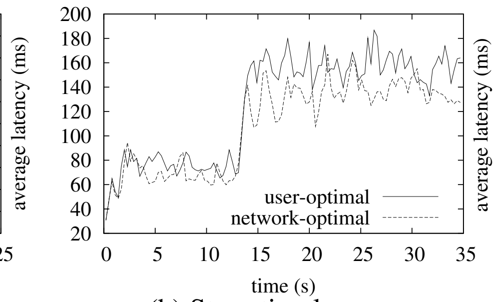


(c) Tiscali

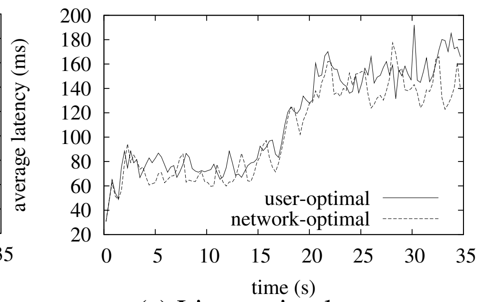
Figure 5. Dynamics User-optimal and Network-optimal Routing.



(a) Spike stimulus



(b) Step stimulus



(c) Linear stimulus

Figure 6. Responsiveness of routing schemes.

5.3 Comparison with shortest hop-count routing

In this subsection, we compare the performance of user-optimal with shortest hop-count routing (*i.e.*, rt-ProtoDV in ns-2) under both self-similar traffic and traffic stimuli. Figure 7 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.

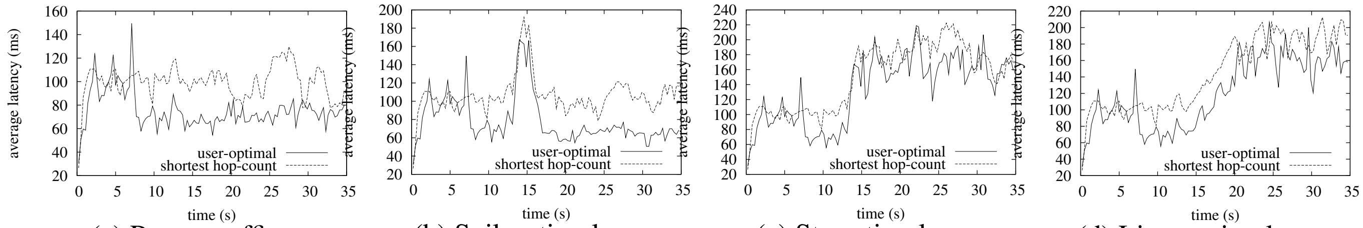
5.4 Improving network convergence and responsiveness

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.

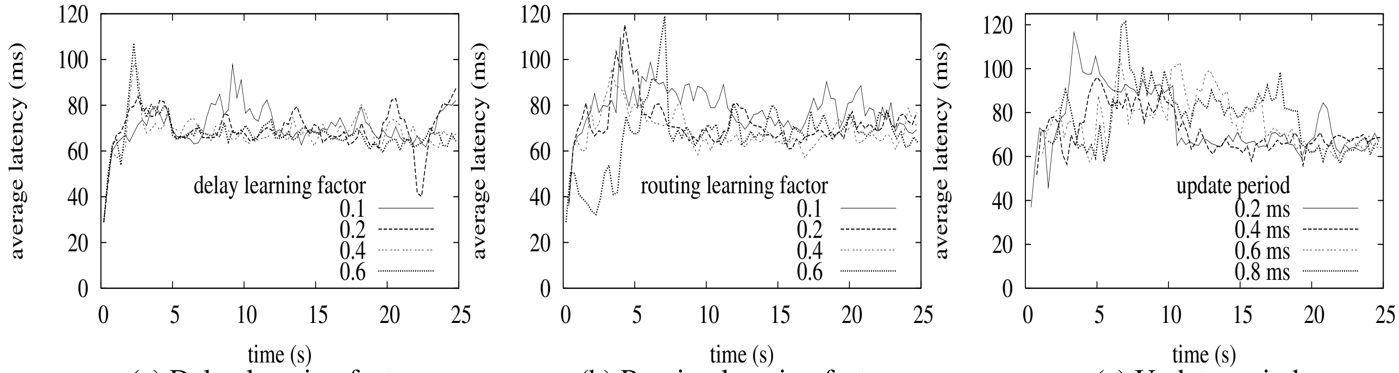
Delay Learning Factor: We first evaluate the effect of the delay learning factor, which is used in updating delay estimations. As shown in Figure 8(a), a higher delay learning factor leads to faster convergence and higher fluctuation. However, neither a too-low factor(*e.g.*, 0.1 and 0.2) nor a too-high factor (*e.g.*, 0.8, which is not plot for the sake of clarity) leads to a stable state with low fluctuation. A preferable learning factor would range between 0.4 and 0.6.

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. High learning factors make the algorithm too sensitive to noise (therefore, more and larger fluctuations in latency after convergence), while small factors make the algorithm not responsive to network dynamics (therefore, slower convergence rate). It is very important to choose appropriate values for delay learning factor



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

Figure 7. Comparison between user-optimal and shortest path routing.



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

Figure 8. Impacts of different parameters.

in order for the routing algorithm to be both robust to noise and responsive to network dynamics.

Routing Learning Factor: Figure 8(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, routing learning factor of large values makes the algorithm sensitive to noise and leads to undesired fluctuations after convergence; and 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 8(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 to fit the 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 very low and localized, and the routing schemes with a shorter update period still scale very well. In practice, we can use much larger update periods to further reduce the overhead when slower convergence and lower responsiveness are allowed.

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. Finding the optimal values for these parameters in real networks may be difficult, and studies based on simulations and empirical experiments will play an important role.

5.5 Routing loops

The two routing schemes studied in this paper are both probabilistic routing. Unlike a deterministic routing such as a distance-vector or link-state routing scheme, a probabilistic routing cannot be guaranteed

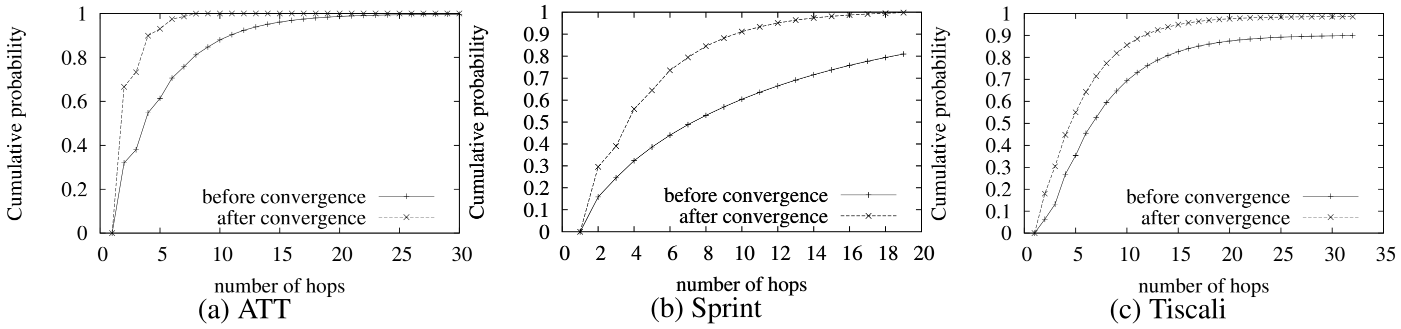


Figure 9. Cumulative probability of packets being delivered in a number of hops.

to be free of loops.[1] In this section, we examine how likely routing loops are formed under the above routing schemes.

We compute the probability of forming routing loops as follows. We take all of the routing probability vectors from all of the nodes at time t. These vectors altogether represent the complete state of the network at that time. For any packet sent to a particular destination, a Markov transition matrix can be constructed from the above vectors obtained from all of the routing nodes. Let A_d denote such a complete routing probability matrix for a destination d. The probability that a packet starting from a particular source node s (this node routes the packet based on its own routing probability vector) arrives at the destination node in h hops can be represented as $(A_d^h)_{sd}$, *i.e.*, the d-th entry in the s-th row in the matrix A_d^h.

Figure 9 compares the cumulative distribution of a packet delivered to the destination in h hops before and after the network is converged. As we can see, the probability of forming routing loops 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.

In addition, we observe that topologies have a significant impact on how likely routing loops are formed. The probability of having routing loops is much lower in the ATT topology than in Sprint or Tiscali. After taking a closer look at the topologies, we find that both Sprint and Tiscali are highly connected, so a packet has more routing choices at every hop, increasing the chance of traveling more hops. In comparison, for a less well connected topology, such as ATT, the chance of forming routing loops is lower.

6 Optimization for both User Latency and Link Utilization

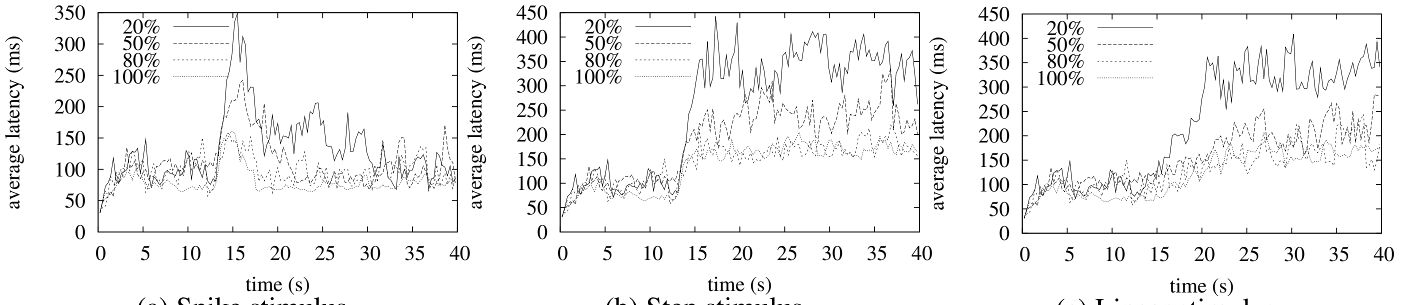
So far, we have considered routing for optimizing latency. As shown in [29], optimizing the average user latency alone sometimes leads to overloaded network links, which is undesirable from the network operators' traffic engineering objective [17]. Ideally, we would like to achieve low user latency while avoiding overloaded links. 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 overloaded 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 how responsive and stable the different routing schemes are. Figure 10 shows the average latency for the Tiscali network when user-optimal routing is used.

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 Fig-

[1]The routing schemes proposed in [5, 18] guarantee loop-freeness. However, they require synchronous update, which may be undesirable in a large network.



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

Figure 10. User-optimal routing combined with load optimization: average user latency for various link utilization thresholds.

ure 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 we restrict link utilization to be 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 11 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 [40]. In [4, 40], 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.* [4] 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 [30] 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 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.

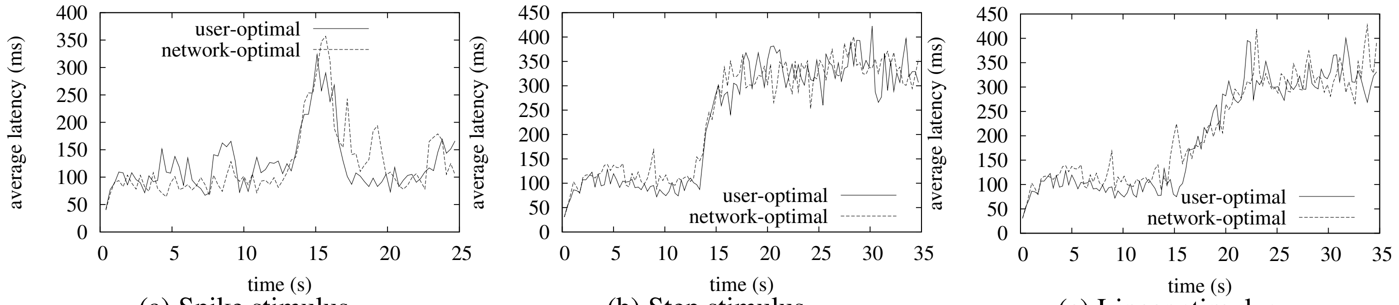
Network-optimal routing in a centralized setting has been studied previously, and many optimization techniques have been proposed. For a survey, please see [6]. There are also previous distributed algorithms for computing optimal traffic equilibrium, *e.g.*, see [8] for a complete survey; however, they do not use the probabilistic routing scheme.

Our routing scheme is based on reinforcement learning [21]. In [13, 28], Littman and Boyan first applied reinforcement learning to routing and proposed the Q-routing scheme. The Q-routing scheme is further revised in [25, 36]. Their scheme is per-packet based, however, and only supports single path routing. The work closest to ours is [10], but (like Q-routing) the scheme is per-packet based and considers only user-optimal routing.

The authors in [29] study the interaction between end users' route selection and network-level routing. In [23, 26, 31, 32], the interactions between routing and traffic engineering are studied, but in the context of Stackelberg routing, which is different from the traffic engineering objective [17] we studied in this paper.

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



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

Figure 11. Comparison between the user- and network-optimal routing schemes: average latency when the link utilization threshold is 20%.

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, we are interested in a better understanding of the incentive issues of our routing scheme. By having routers compute user-optimal routing, we know that the users will have no incentives in deviating from the routing protocol. The major issue, however, is the interaction among routing, traffic engineering, and flow control. How to design routing schemes with incentive-compatibility [15] and high efficiency while interacting well with traffic engineering and flow control is an open issue. Second, it remains an open problem how to compute traffic equilibria when users are optimizing for other end-to-end performance metrics, such as loss and throughput. The probabilistic routing investigated here may offer a way to efficiently compute traffic equilibria for such cases.

9 Appendix

Let $\gamma(n,t) = \min\{l \geq n : \sum_{k=n}^{l} \alpha(k) \geq t\}$, where $t > 0$. Define interval $I_n = [\sigma_j^{ik}(n), \sigma_j^{ik}(\gamma(n,t))]$.

Let P^{ik} denote the simplex of probability vectors in R^{N_i}, where N_i is the number of neighbors of node i. Let $p^{ik}(n) = [p_1^{ik}(n), ..., p_1^{ik}(N_i)]^T \in P^{ik}$ denote the probability vector of node i for destination node k. We define Q^{ik} and $q^{ik}(n)$ similarly. Define $P = \prod_{i,k} P^{ik}$ and $Q = \prod_{i,k} Q^{ik}$. Let $q(t) \in Q$ denote the vector of all the routing probabilities at all nodes at time t, and the value of $q(t)$ is computed using Equation (4). $p(t) \in P$ is defined similarly as well.

Lemma 1

$$\lim_{n\to\infty} \sup_{t\in I_n} \|q(t) - q(\sigma_j^{ik}(n))\| = 0, a.s.$$

Proof Let $\{\sigma^{ik}(n)\} = \bigcup_{j\in N(i)}\{\sigma_j^{ik}(m)\}$, rearranged in increasing order. Suppose that $\sigma_j^{ik}(n) = \sigma^{ik}(\hat{n})$ and $\sigma_j^{ik}(\gamma(n,t)) = \sigma^{ik}(\hat{n}_h)$ with $\hat{n} = \hat{n}_0 < \hat{n}_1 < ... < \hat{n}_h$ denoting the integers for which $\sigma^{ik}(n+l) = \sigma^{ik}(\hat{n}_l), 0 \leq l \leq h$. Let X_l be the number of updates received by node i during the interval $[\sigma_j^{ik}(n+l), \sigma_j^{ik}(n+l+1)]$. Let M_l be the set of m such that the m-th update takes place during this interval. Note that $|M_l| = X_l$. Therefore, $\beta(\hat{n}_l)$ is the most recent step size used to update routing probabilities at time $\sigma_j^{ik}(n+l)$ at node i. Since $\{\beta(m)\}$ is non-increasing, we have $\beta(\hat{n}_l) \geq \beta(m)$ for all $m \in M_l$. We also have $\hat{n}_l \geq n+l$, implying $\beta(\hat{n}_l) \leq \beta(n+l)$.

Hence

$$\begin{aligned}\sup_{t\in I_n}\|q(t)-q(\sigma_j^{ik}(n))\| &\leq C_1\sum_{l=0}^{h}\sum_{m\in M_l}\beta(m)\\ &\leq C_1\sum_{l=0}^{h}X_l\beta(\hat{n}_l)\\ &= C_1\sum_{l=0}^{h}a(n+l)X_l(\frac{\beta(\hat{n}_l)}{a(n+l)})\\ &\leq C_1\sum_{l=0}^{h}a(n+l)X_l(\frac{\beta(n+l)}{a(n+l)})\\ &\leq C_2 t\frac{\sum_{l=0}^{h}\alpha(n+l)X_l(\frac{\beta(n+l)}{\alpha(n+l)})}{\sum_{l=0}^{h}\alpha(n+l)},\end{aligned}$$

where C_1 is a suitable positive constant in the above derivation. The facts that $\beta(\hat{n}_l)\geq\beta(m)$, $|M_l|=X_l$, and $\beta(\hat{n}_l)\leq\beta(n+l)$ are used in deriving the above inequalities. In the last step, the fact that $\sum_{l=0}^{h}\alpha(n+l)=\sum_{l=n}^{\gamma(n,t)}\alpha(l)\in[t,t+a(\gamma(n,t))]$ is used in the derivation as well. We choose $C_2\geq C_1(t+\alpha(1))/t$.

The right hand side of the above inequality will tend to zero almost surely if $X_l(\beta(n+l)/\alpha(n+l))\to 0$ almost surely. However, for any $\delta>0$ and $s>1$, and a suitable constant $K>0$,

$$\sum_l P\left(X_l\left(\frac{\beta(n+l)}{\alpha(n+l)}\right)\geq\delta\right)\leq\delta^{-s}K\sum_l\left(\frac{\beta(l)}{\alpha(l)}\right)^s<\infty.$$

Note that $E[(X_l)^s]<\infty$. By applying Borel-Cantelli Lemma, we can see that the desired claim holds. □

Lemma 2

$$\lim_{n\to\infty}\sum_{m=n}^{n+l-1}\alpha(m)[\Delta_j^{ik}(m+1)-\Delta_j^{ik}(n)]=0, a.s.$$

Proof Delay estimations are propagated backward from neighbors of destination node k to node j and then to node i along the reverse paths from i through j to k. At any instant when node i receives updates from j, the updates are aggregation of a series of earlier updates received by j's neighbors. Let T_n be the maximum time lag in the updates, namely, the earliest estimation at some node en route to k from i is $n-T_n$. Let $D_j^{ik}(n)$ be the end to end delay, it can be seen that $\Delta_j^{ik}(m+1)-\Delta_j^{ik}(n)\leq C_1\alpha(n-T_n)D_j^{ik}(n)$, where C_1 is a suitably chosen constant (note that $\{\alpha(n)\}$ is a non-increasing sequence). Therefore,

$$\begin{aligned}&\sum_{m=n}^{n+l-1}\alpha(m)[\Delta_j^{ik}(m+1)-\Delta_j^{ik}(n)]\\ &\leq C_2\sum_{m=n}^{n+l-1}\alpha(m)\alpha(n-T_n)D_j^{ik}(n).\end{aligned}$$

Notice that

$$\begin{aligned}&\sum_{m=n}^{n+l-1}P(\alpha(m)\alpha(n-T_n)D_j^{ik}(n)\geq\delta)\\ &\leq \delta^{-2}\sum_{m=n}^{n+l-1}(\alpha(m)\alpha(n-T_n))^2E[(D_j^{ik}(n))^2] \quad (9)\\ &\leq C_3\sum_{m=n-T_n}^{n+l-T_n-1}(\alpha(m))^2<\infty.\end{aligned}$$

In (9), we assume that $E[(D_j^{ik}(n))^2]<\infty$. Therefore, by applying Borel-Cantelli Lemma, we can see that the right hand side of the above inequality tends to zero almost surely. □

Define

$$\bar{D}_j^{ik}(\hat{p}(n))=E_{\hat{p}(n)}[\Delta_j^{ik}(n)]$$

where $\hat{p}(n)=p(\sigma_j^{ik}(n))$. We have the following Lemma:

Lemma 3

$$\lim_{n\to\infty}\sum_{m=n}^{n+l-1}a(m)[\bar{D}_j^{ik}(p(\sigma_j^{ik}(n)))-\Delta_j^{ik}(n)]=0, a.s.$$

Proof Let $Z(t)$ denote the state of the underlying network at time t. $Z(t)$ can be seen as a continuous-time controlled Markov chain with state space $S=\prod_{l=1}^{L}S_l$ for some $L\geq 1$, where S_l is a locally compact topological space with the randomized control given by $p(t)$, which is the vector of all the routing probabilities at time t. $Z(t)$ will be a time-homogeneous Markov chain on S if $p(\cdot)$ is held fixed at a specific state. Let F_t denote the right-continuous completion of the corresponding σ-field. Set $\hat{F}_n=F_{\sigma_j^{ik}(n)}$, for $n\geq 1$, the sum

$$\sum_{m=1}^{n}\alpha(m)\left(\Delta_j^{ik}(m)-E_{p(\sigma_j^{ik}(n))}[\Delta_j^{ik}(n)|\hat{F}_{m-1}]\right)$$

is seen to be a zero mean martingale. Using standard martingale analysis techniques, we can prove that the above martingale converges almost surely. The desired claim then follows Borel-Cantelli Lemma. Note that $\bar{D}_j^{ik}(p(\sigma_j^{ik}(n)))$ represents the expected delay from i through j to k, with respect to the unique limiting stationary law for $\{Z(\sigma_j^{ik}(m))\}$ under $p(n)$. □

In order to make delay estimation a continuous function, we take a linear interpolation approach. We define $\bar{L}(t)$ to linearly interpolate $L_j^{ik}(n)$ by letting $\bar{L}(t) = L_j^{ik}(n)$ on $[t(n), t(n+1)]$, where $t(n) = \sum_{m=1}^{n} \alpha(m)$ and $t(0) = 0$, for $n \geq 0$. Let T be a constant and $T > 0$, then we have the following lemma:

Lemma 4

$$\lim_{n\to\infty} \sup_{t(n)\le t\le t(n)+T} \|\bar{D}_j^{ik}(\hat{p}(n)) - \bar{L}(t)\| = 0, a.s.$$

Proof Let $A_n = \sum_{m=n}^{n+l-1} \alpha(m)[\Delta_j^{ik}(m+1) - \Delta_j^{ik}(n)]$, and $B_n = \sum_{m=n}^{n+l-1} \alpha(m)[\Delta_j^{ik}(n) - \bar{D}_j^{ik}(p(\sigma_j^{ik}(n)))]$, for $t(n) \leq t(n+l) \leq t(n) + t$, rewrite $\bar{L}(t(n+l))$ as

$$\begin{aligned}
&\bar{L}(t(n+l)) \\
=\ & \bar{L}(t(n)) + \textstyle\sum_{m=n}^{n+l-1} \alpha(m)[\Delta_j^{ik}(m+1) - \bar{L}(t(m))] \\
=\ & \bar{L}(t(n)) + \textstyle\sum_{m=n}^{n+l-1} \alpha(m)[\Delta_j^{ik}(m+1) - \Delta_j^{ik}(n)] \\
& + \textstyle\sum_{m=n}^{n+l-1} \alpha(m)[\Delta_j^{ik}(n) - \bar{D}_j^{ik}(p(\sigma_j^{ik}(n)))] \\
& + \textstyle\sum_{m=n}^{n+l-1} \alpha(m)[\bar{D}_j^{ik}(p(\sigma_j^{ik}(n))) - \bar{L}(t(m))].
\end{aligned}$$

Therefore,

$$\begin{aligned}
\bar{L}(t(n+l)) &= \bar{L}(t(n)) + A_n + B_n + o(1) \\
&\quad + \int_{t(n)}^{t(n+l)} (\bar{D}_j^{ik}(\hat{p}(n)) - \bar{L}(t))dt.
\end{aligned}$$

Along with Lemma 1, Lemma 2, and Lemma 3, the desired claim can be derived by a standard argument based on Gronwall Lemma. □

Define $\hat{\beta}(n) = \sum_{m=0}^{n-1} \beta(m), \hat{\beta}(0) = 0$, and $\bar{q}_j^{ik}(\hat{\beta}(n)) = q_j^{ik}(n), n \geq 0$. Therefore, $\bar{q}(\cdot)$ is a linear interpolation of $q(\cdot)$ on each interval $[\hat{\beta}(n), \hat{\beta}(n+1)]$. By using standard O.D.E. approach to projected stochastic approximation algorithms (see, *e.g.*, [27]), we can derive the following P-valued O.D.E.:

$$\dot{y}_j^{ik}(t) = y_j^{ik}(t)[\sum_{m=1}^{N_i} y_m^{ik}(t) D_m^{ik}(y(t)) - D_j^{ik}(y(t))] \tag{10}$$

where $m \in N(i)$. Therefore, $\bar{q}(\cdot)$ tracks (10).

Let P^* denote the space of probability measures on P with Prohorov topology. A pair (q, v^*) in $P \times P^*$ is a Cesaro-Wardrop equilibrium if both of the following two conditions are satisfied:

1. v^* is invariant under O.D.E. (10);
2. $E^*[D_j^{ik}(p)] = \min_m E^*[D_m^{ik}(p)]$, if $q_j^{ik} > 0$. Here $E^*[D_m^{ik}(p)] = \int D_m^{ik}(y)v^* dy$, *i.e.*, $E^*[\cdot]$ denotes the expectation with respect to v^*.

Define empirical measures $v(t)$ as follows:

$$\int_P f dv(t) = \frac{1}{t}\int_0^t f(\bar{q}(\tau))d\tau$$

where $t > 0$ and $f \in C(P)$. Then we have the following lemmas to show that both of the above conditions are satisfied asymptotically.

Lemma 5 *Every limit point of* $v(t) \in P^*$ *is invariant under O.D.E. (10) as* $t \to \infty$, $a.s.$

Proof Suppose that $v(t_n) \to \tilde{v} \in P^*$ as $t_n \to \infty$. Let $\Phi_t : P \to P$ denote the map mapping $y(0)$ to $y(t)$ for O.D.E. (10) for $t > 0$. Consider function $f \in C(P)$ satisfying

$$\frac{1}{t_n}\int_0^{t_n} f(\bar{q}(\tau))d\tau \to \int f d\tilde{v}.$$

For $t > 0$,

$$\begin{aligned}
\lim_{n\to\infty} \frac{1}{t_n}\int f(\bar{q}(\tau)d\tau &= \lim_{n\to\infty} \frac{1}{t_n}\int_t^{t_n+t} f(\bar{q}(\tau))d\tau \\
&= \lim_{n\to\infty} \frac{1}{t_n}\int_0^{t_n} f \circ \Phi_t(\bar{q}(\tau))d\tau \\
&= \int f \circ \Phi_t d\tilde{v}.
\end{aligned}$$

Therefore, for any limit point of $v(t) \in P^*$ is invariant as $t \to \infty$ since $t > 0$ is arbitrary in the above equations. □

According to Lemma 5, we denote by (q^*, v^*) a limit point of $(\bar{q}, v(t))$ as $t \to \infty$. Note that v^* is invariant under (10) by the previous lemma.

Lemma 6 *$E^*[D_j^{ik}(p)] = \min_m E^*[D_m^{ik}(p)]$, if $q_j^{ik} > 0$.*

Proof Suppose $\bar{q}(t_n) \to q^*, v(t_n) \to v^*$ in P^* as $t_n \to \infty$. Since $\bar{q}(\cdot)$ approximates (10), we have

$$\sum_m \rho_m^{ik} n(\frac{\bar{q}_m^{ik}(t_n)}{\bar{q}_m^{ik}(0)}) = \int_0^{t_n} (\bar{q}^{ik}(\tau))^T D^{ik}(\bar{q}(\tau)) d\tau - \int_0^{t_n} (\rho^{ik})^T D^{ik}(\bar{q}(\tau)) d\tau + o(t_n). \quad (11)$$

Note that the term $o(t_n)$ collects the error terms that are asymptotically negligible. Without loss of generality, we assume that $\bar{q}_m^{ik} > 0$. The right-hand side becomes

$$t_n(E^*[(p^{ik})^T D^{ik}(p)] - (\rho^{ik})^T E^*[D^{ik}(p)]) + o(t_n) \quad (12)$$

as $n \to \infty$; while the left-hand side remains bounded by our choice of ρ. Therefore, by dividing both sides by t_n, we have

$$E^*[(p^{ik})^T D^{ik}(p) - (\rho^{ik})^T E^*[D^{ik}(p)] = 0 \quad (13)$$

as $t_n \to \infty$, which is true for all ρ mutually absolutely continuous with respect to q^*. Therefore, $E^*[D_m^{ik}(p)]$ is independent of m if $(q^*)_m^{ik} > 0$. □

Now we prove Theorem 1 as follows. Let (q^*, v^*) be a limit point of $(\bar{q}(t), v(t))$ as $t \to \infty$ and (q^*, v^*) satisfies the following two conditions: (1) v^* is invariant; (2) $E^*[D_j^{ik}(p)] = min_m E^*[D_m^{ik}(p)]$ if $(q^*)_j^{ik} > 0$, according to Lemma 5 and 6.

Given that we have proved Lemma 5 and 6, we need only to prove that if $(q^*)_j^{ik} = 0$, then

$$E^*[D_j^{ik}(p) \geq min_m E^*[D_m^{ik}(p)]$$

for all j. We prove the theorem by contradiction.

Assume that there exists a j such that $(q^*)_j^{ik} = 0$ and $E^*[D_j^{ik}(p)] < E^*[D_{m_0}^{ik}] - \gamma$ where $(q^*)_{m_0}^{ik} > 0$ and $E^*[D_{m_0}^{ik}(p)] = min_m E^*[D_m^{ik}(p)]$, for some $\gamma > 0$. Let t_n be defined as in the proof of Lemma 6. Let $\rho = (q^* + \delta_{m_0})/2$, where δ_{m_0} is a point mass at m_0. Similarly to the proof of Lemma 6, we have (11) and (12). (12) is positive for sufficiently large n in view of choice of ρ and Equation (13). Therefore, (12) is actually $O(\gamma)t_n$ and increasing to $+\infty$. However, the left-hand side of Equation (11) is approaching to $-\infty$ because $\bar{q}_{m_0}^{ik}(t_n) \to (q^*)_{m_0}^{ik} = 0$ according to our assumption. Therefore our previous assumption yields a contradiction.

References

[1] Network simulator – ns-2. http://www.isi.edu/nsnam/ns/.

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

[3] 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.

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

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

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

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

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

[9] V. Borkar. Stochastic approximation with two time scales. *Systems and Control Letter*, 29:291–294, 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] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. In *Proceedings of the 21st Symposium on Principles of Distributed Computing*, pages 173–182, 2002.

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

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

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

[19] 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.

[20] 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.

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

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

[23] Y. A. Korilis, A. A. Lazar, and A. Orda. Achieving network optima using Stackelberg routing strategies. *IEEE/ACM Transactions on Networking*, 5(1):161–173, February 1997.

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

[25] S. Kumar. Confidence based dual reinforcement Q-routing: an on-line adaptive network routing algorithm. Technical Report AI98-267, 1, 1998.

[26] V. A. Kumar and M. V. Marathe. Improved results for Stackelberg scheduling strategies. Technical report, Los Alamos National Laboratory, November 2001.

[27] H. J. Kushner and G. G. Yin. *Stochastic Approximation Algorithms and Applications*. Springer Verlag, New York, NY, 1997.

[28] 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.

[29] 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.

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

[31] T. Roughgarden. Designing networks for selfish users is hard. In *Proceedings of the 42nd Annual Symposium on Foudations of Computer Science 2001*, pages 472–481, Las Vegas, Nevada, Oct. 2001.

[32] T. Roughgarden. Stackelberg scheduling strategies. In *Proceedings of the 33rd Annual Symposium on Theory of Computing 2001*, pages 104–113, 2001.

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

[34] 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, January 1999.

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

[36] 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.

[37] 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.

[38] 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.

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

[40] 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.

[41] 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, Computer Science Department, Yale University, May 2004.

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