On Selfish Routing in Internet-Like Environments

Lili Qiu
Microsoft Research
liliq@microsoft.com

Yang Richard Yang [*]
Yale University
yry@cs.yale.edu

Yin Zhang
AT&T Labs – Research
yzhang@research.att.com

Scott Shenker [†]
ICSI
shenker@icir.org

ABSTRACT

A recent trend in routing research is to avoid inefficiencies in network-level routing by allowing hosts to either choose routes themselves (*e.g.*, source routing) or use overlay routing networks (*e.g.*, Detour or RON). Such approaches result in *selfish* routing, because routing decisions are no longer based on system-wide criteria but are instead designed to optimize host-based or overlay-based metrics. A series of theoretical results showing that selfish routing can result in suboptimal system behavior have cast doubts on this approach. In this paper, we use a game-theoretic approach to investigate the performance of selfish routing in Internet-like environments. We focus on intra-domain network environments and use realistic topologies and traffic demands in our simulations. We show that in contrast to theoretical worst cases, selfish routing achieves close to optimal average latency in such environments. However, such performance benefit comes at the expense of significantly increased congestion on certain links. Moreover, the adaptive nature of selfish overlays can significantly reduce the effectiveness of traffic engineering by making network traffic less predictable.

Categories and Subject Descriptors

C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet*

General Terms

Performance

Keywords

Selfish Routing, Overlay, Game Theory, Traffic Equilibrium, Traffic Engineering, Optimization, Relaxation

1. INTRODUCTION

For decades, it has been the responsibility of the network to route traffic. Recent studies [32, 40] have shown that there is inherent inefficiency in network-level routing from the user's perspective. In response to these observations, we have seen an emergent trend to allow end hosts to choose routes themselves by using either source routing (*e.g.*, Nimrod [8]) or overlay routing (*e.g.*, Detour [32] or RON [5]). These end-to-end route selection schemes are shown to be effective in addressing some deficiencies in today's IP routing. For example, measurements [10, 32, 33] from the Detour project show that in the Internet, a large percentage of flows can find better alternative paths by relaying among overlay nodes, thereby improving their performance. RON [5] also demonstrates the benefit of overlay routing using real implementation and deployment.

Such end-to-end route selection schemes are selfish by nature in that they allow end users to greedily select routes to optimize their own performance without considering the system-wide criteria. Recent theoretical results suggest that in the worst case selfish routing can result in serious performance degradation due to lack of cooperation. In particular, Roughgarden and Tardos prove that the *price of anarchy* (*i.e.*, the worst-case ratio between the total latency of selfish routing and that of the global optimal) for selfish routing can be unbounded for general latency functions [31].

Despite much theoretical advance, an open question is how selfish routing performs in Internet-like environments. This is a challenging question, since today's Internet is unique in the following respects.

First, topologies and traffic demands of the Internet are not arbitrary but have certain structures. The worst-case results may not be applicable to realistic topologies and traffic demands. A general open question is *whether selfish routing results in bad performance in Internet-like environments (*i.e., *under realistic network topologies and traffic demands)*.

Second, users in overlay networks do not have full flexibility in specifying their end-to-end paths. Due to limited availability of source routing support in the routers, the path between any two network nodes is dictated by the Internet routing protocols, such as OSPF [28], MPLS [26], or BGP [37]. While overlay networks provide another mechanism to enable users to control their routes by relaying through overlay nodes, the route between two overlay nodes is still governed by the underlying routing protocol. A natural question is *how to model such selfish overlay routing and whether selfish overlay routing results in bad performance*.

Third, even if selfish overlays (*i.e.*, overlays consisting of selfish traffic) yield good performance, they can only be deployed gradually. As a result, background traffic and overlay traffic will interact with each other. We call such interactions *horizontal interactions*. An important question is *how such selfish traffic affects the remaining traffic routed using the traditional routing protocols*. A related question is *whether multiple overlays result in bad performance*.

Fourth, the way in which selfish users choose their routes can interact with traffic engineering. We call such interactions *vertical interactions*, which can be viewed as the following iterative process. First, ISPs adjust network-level routing according to traffic demands, using schemes in [6, 14, 15, 42], to minimize *network cost*. Then selfish users adapt to changes in the underlying default routes by choosing different overlay paths to optimize their end-

[*] Supported in part by NSF grant ANI-0207399.
[†] Supported in part by NSF grants ITR-0205519, ANI-0207399, ITR-0121555, ITR-0081698, ITR-0225660 and ANI-0196514.

to-end performance. Such adaptation changes traffic demands and triggers traffic engineering to readjust the default routes, which in turn makes selfish users adapt to new routes. Given the mismatch between the objectives of selfish routing and traffic engineering, an interesting question is *whether selfish routing interacts badly with traffic engineering*.

In this paper, we seek to answer the above questions through extensive simulations. We take a game-theoretic approach to compute the traffic equilibria of various routing schemes and then evaluate their performance. We focus on *intra-domain* network environments because recent advances in topology mapping [36] and traffic estimation [44] allow us to use realistic network topologies and traffic demands for such scenarios. Understanding selfish routing in inter-domain environments is also of great interest but will be more challenging. First, we do not have realistic models for inter-domain traffic demands. Moreover, despite some recent progress towards understanding autonomous system relationships [17, 38], more research efforts are needed to develop realistic models for inter-domain routing policies. Finally, the large size of inter-domain topologies makes it computationally prohibitive to derive traffic equilibria. Due to these difficulties, we defer it to future work.

Our key contributions and results can be summarized as follows. First, we formulate and evaluate selfish routing in an overlay network. Selfish routing in an overlay network is different from traditional selfish source routing in that (i) the route between any two overlay nodes is dictated by network-level routing, and (ii) different overlay links may share common physical links and therefore traditional algorithms to compute traffic equilibria do not apply.

Second, we find that in contrast to theoretical worst cases, selfish routing in Internet-like environments yields close to optimal average latency, which can be much lower than that of default network-level routing. This is true for both source routing and overlay routing. Moreover, we show selfish routing achieves good performance without hurting the traffic that is using default network-level routing.

Third, we show that the primary impact of selfish routing on Internet-like environments is the fundamental mismatch between the objectives of selfish routing and traffic engineering. In particular, our results show that the low latency of selfish routing is often achieved at the expense of increased congestion on certain links. Moreover, the adaptive nature of selfish routing makes traffic demands less predictable and can significantly reduce the effectiveness of traffic engineering.

The rest of the paper is organized as follows. In Section 2, we review related work. In Section 3, we present our network model. In Section 4, we specify the routing schemes we evaluate and present the algorithms we use to compute their traffic equilibria. In Section 5, we describe our evaluation methodology. We study the performance of selfish source routing in Section 6 and that of selfish overlay routing in Section 7. In Section 8 and Section 9, we investigate horizontal and vertical interactions, respectively. We conclude in Section 10.

2. RELATED WORK

A number of recent studies have reported that network-level routing is inefficient from the user's perspective. For example, Savage *et al.* [33] use Internet measurements to show that the default routing path is often suboptimal in terms of latency, loss rate, and TCP throughput. The suboptimal performance of network-level routing is inevitable due to routing hierarchy and policy [40], as well as different routing objectives used by network operators, whose goal is to avoid high utilization. Moreover, stability problems with routing protocols, such as BGP [37], could make things even worse. As a result, there has been a movement to give users more autonomy in choosing their routes by using source routing (*e.g.*, Nimrod [8]) or overlay routing networks (*e.g.*, Detour [32, 33] and RON [5]).

Recently a series of theoretical results show that selfish routing can result in extremely suboptimal performance in worst cases. The pioneering work in this area is by Koutsoupias and Papadimitriou [22], who compare the worst-case Nash equilibrium with a global optimal solution in minimizing network congestion in a two-node network. Roughgarden and Tardos are interested in a different performance metric – latency. In [31], they prove that the price of anarchy (*i.e.*, the worst-case ratio between the average latency of a Nash equilibrium and that of the global optimal) depends on the "steepness" of the network latency functions. They show that the price of anarchy is unbounded for a general latency function such as M/M/1. In contrast to the theoretical studies, our study focuses on a practical setting, by using realistic network topologies and traffic demands; different from the measurement studies, our study considers a more general setting and investigates networks with a large amount of selfish traffic, under different network configurations (including both static and dynamic network controls).

Although the price of anarchy can be high in the worst-case, some theoretical studies have also shown that the degradation is less severe from some other perspectives. For example, Friedman shows that for "most" traffic rate vectors in a range, the price of anarchy is lower than that of the worst cases [16]. He also analyzes the effects of TCP rate adaptation in a parallel-link network and shows that the performance loss is small. Roughgarden and Tardos [31] show (essentially) that the performance degradation due to selfish routing can be compensated for by doubling the bandwidth on all links. However, this is often not a practical option for the Internet at least in the short-term.

There are also other ways in which end users can selfishly optimize the performance of their traffic. For example, a user can greedily inject traffic into a network. A number of papers (*e.g.*, [2, 35]) consider such a congestion game. In practice, it is possible to have a hybrid game that consists of a route selection game and a congestion game, but we defer it to future work.

3. NETWORK MODEL

In this section, we describe our network model, especially the network-level routing protocols. In the next section, we describe the schemes of how traffic demands are routed through the network. In Section 5, we describe the network topologies, traffic demands, and latency functions that we use to instantiate our network model.

Physical network: We study the performance of realistic physical networks. We model a physical network as a directed graph $G = (V, E)$, where V is the set of nodes, and E the set of directed links. We assume that the latency of each physical link is a function of its load. The exact latency functions we use will be described in Section 5.3.

Demands: We partition network traffic into demands. A demand represents a given amount of traffic from a source to a destination. In particular, we identify a special type of demand, called infinitesimal demand. A collection of infinitesimal demands models a large aggregation of independent, small transactions such as web transactions, and the generator of each transaction makes an independent decision.

Overlays: An overlay consists of overlay nodes, directed overlay links, and a set of demands originated from the overlay nodes. The overlay nodes agree to forward each other's traffic along one or more overlay links. The physical route for an overlay link is dictated by network-level routing and may involve multiple physical links. Different overlay links may share one or more physical links. The overlay nodes and overlay links form the overlay topology. To limit the parameter space, we only consider the *fully connected* overlay topology in this work. That is, we assume there is an overlay link between every pair of overlay nodes. We plan to investigate the effects of different overlay topologies in our future work.

Users: We assume that the network consists of a collection of users. Each user decides how its traffic should be routed. The objective of a user is to minimize the average latency of its traffic. We choose to use latency as the optimization objective of selfish routing for the following reasons: 1) many applications such as short Web transfers and IP telephony require low latency; 2) most previous theoretical analyses are based on latency, and one of the major objectives of this study is to investigate whether the theoretical worst-case results apply to Internet-like environments. We plan to investigate the effects of alternative routing objectives (*e.g.*, loss [3]) in our future work.

Route controller: Besides users, we also have a route controller, which controls the network-level routing in the physical network. (We use network-level routing and physical routing interchangeably in this paper.) We consider several types of network-level routing. We assume that the route controller uses a routing protocol based on either OSPF[28], which uses shortest-path with equal-weight splitting, or MPLS[26], which uses the more general multi-commodity flow routing. For OSPF routing, we consider three weight assignments:

- Hop-count OSPF routing, which assigns a unit weight to each physical link;
- Random-weight OSPF routing, which assigns a random weight to each physical link;
- Optimized-compliant OSPF routing, which has OSPF weights set to minimize network cost [14] (see Section 5.4), when assuming all traffic is compliant, following the routes determined by the network. The network cost is a piece-wise linear convex function over all links. This metric has been considered as a good objective for traffic engineering because it not only avoids overloading physical links, but also avoids taking very long paths [14, 15].

We represent network-level routing by a routing matrix R, where $R[p, e]$ specifies the fraction of traffic between the source-destination pair p that goes through the physical link e. The routing matrix R is computed by the routing protocol under study.

In our study, the route controller can change network routing to optimize overall network performance; in other words, it can perform traffic engineering. For MPLS, the route controller can directly adjust the routing matrix R; for OSPF, the route controller will adjust the weights of the physical links to influence network routing [14, 15].

4. ROUTING AND TRAFFIC EQUILIBRIA

We evaluate each selfish routing scheme by computing its performance at traffic equilibria. Using a game-theoretic approach, we define a traffic equilibrium as a state where no user can improve the latency of its traffic by unilaterally changing the amount of traffic it sends along different network paths. One possible way of computing traffic equilibria is through simulation. More specifically, one could simulate the moves of each individual user and wait until the system reaches equilibrium. However, given the size of the network we are considering (see Section 5.1), such simulation-based approach may take a prohibitively long time to converge. Instead, we compute traffic equilibria directly. Below we introduce the routing schemes, and specify the algorithms we use to compute the traffic equilibria. See Appendix for further details on the algorithms.

For a comprehensive study, we consider the following five routing schemes: (i) source routing, (ii) optimal routing, (iii) overlay source routing, (iv) overlay optimal routing, and (v) compliant routing. Below we describe these routing schemes in details.

4.1 Routing on the physical network

The first two routing schemes allow a user to route its traffic directly through any paths on the physical network.

Source routing: Source routing results in selfish routing, since the source of the traffic makes an independent decision about how the traffic should be routed. The selfish routing scheme studied in most previous theoretical work is source routing.

Optimal routing: Optimal routing refers to *latency* optimal routing. It models a scenario where a single authority makes the routing decision for all the demands to minimize the average latency.

A traditional algorithm to compute the traffic equilibria of source routing and optimal routing is the linear approximation algorithm, a variant of the well-known Frank-Wolfe algorithm [13, 29, 34] (see Appendix for more details).

4.2 Overlay routing

The next two routing schemes are the overlay versions of source routing and optimal routing.

Overlay source routing: Overlay source routing is selfish routing through overlay nodes. Similar to source routing, it is the traffic source that controls the routes.

Overlay optimal routing: Overlay optimal routing refers to overlay *latency* optimal routing. It models a scenario where the demands in the overlay have complete cooperation in minimizing the average latency.

As mentioned in Section 1, overlay routing is different from routing directly on the physical network. In particular, the physical route for an overlay link is dictated by network-level routing and may involve multiple physical links. Moreover, different overlay links may share common physical links and therefore may interfere with each other. Therefore, we cannot apply the traditional linear approximation algorithms to compute traffic equilibria for such schemes.

We use the following approach to compute traffic equilibria for overlay routing. For each overlay, we build a logical network from the physical network. The nodes in the logical network consist of the union of the nodes in the overlay and the nodes that are the destinations of nonzero demands in the overlay. The links in the logical network consist of all the overlay links, as well as a link from each overlay node to each node that is the destination of some traffic demands but does not belong to the overlay.

Given this model, each logical link can be mapped to a collection of physical links. More specifically, assume that the logical link p is for the source-destination pair p (we use the same symbol p to denote the logical link p and the source-destination pair p), then the logical link consists of all the physical links e such that $R[p, e] > 0$. If a demand sends f units of traffic through a logical link p, then each physical link e will carry $f \cdot R[p, e]$ amount of traffic for this demand. Figure 1 shows an example of a physical network, and the logical network for an overlay formed by nodes 2, 3, and 5.

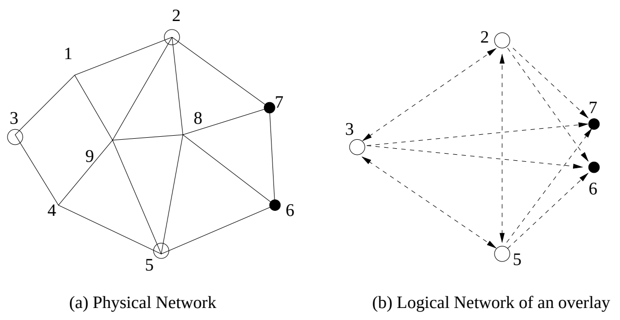


Figure 1: A physical network and the logical network for the overlay formed by nodes 2, 3, and 5. Nodes 6 and 7 are not overlay nodes but nodes 2, 3, and 5 have demands to them. The logical link from node 2 to 5 consists of two physical paths: 2 to 9 to 5, and 2 to 8 to 5, if hop-count OSPF routing is used.

Using such logical networks, we can compute the traffic equilibria of overlay routing by either a modified linear approximation algorithm or a relaxation algorithm (see Appendix for details). When there are multiple overlays, we use the relaxation framework proposed in [23, 41] to ensure convergence (see Appendix for details).

4.3 Compliant routing

For comparison, we also consider the default network-level routing, which we term compliant routing.

Compliant routing: Traffic demands using compliant routing follow the routes determined by the network-level routing protocol.

5. EVALUATION METHODOLOGY

In this section, we first describe the network topologies, traffic demands, and link latency functions used in our evaluation. Then we discuss the performance metrics that we use as a basis for comparing the efficiency of different routing schemes.

5.1 Network topologies

We use both real and synthetic topologies in our evaluation.

Real topology: We use a real router-level backbone topology from an operational tier-1 ISP, referred to as $ISPTopo$, with on the order of a hundred backbone routers connected by OC48 (*i.e.*, 2.48 Gbps) and OC192 (*i.e.*, 10 Gbps) links (the exact numbers are omitted for proprietary reasons). For each link in the real topology, we use the actual link capacity in our study. The propagation delay of each link is estimated using the actual fiber length divided by the speed of light.

Rocketfuel topologies: Rocketfuel applies several effective techniques to obtain fairly complete ISP maps [36]. We use the POP-level maps published by the authors, shown in Table 1, as part of our topologies. For each Rocketfuel topology, we use two bandwidth settings: all links are either OC3 (*i.e.*, 155 Mbps) or OC48 (*i.e.*, 2.48 Gbps). The propagation delay of each link is approximated using geographical distance divided by the speed of light.

ISP	Loc.	#Nodes	#Non-leaf Nodes	#Edges
ATT	US	108	30	282
Abovenet	US	22	13	160
Exodus	US	22	17	102
Level3	US	53	37	912
Sprint	US	44	21	212
Verio	US	122	82	620
EBONE	Intl.	28	25	132
Telstra	Intl.	58	8	120
Tiscali	Intl.	51	38	258

Table 1: ISP topologies as measured by Rocketfuel.

Random topologies: In addition to real topologies, for diversity we also randomly generate power-law topologies using BRITE [25], since a number of papers [11, 39] have shown that the power-laws capture the Internet structure quite well. We generate 100-node router-level topologies with edge density (*i.e.*, the number of neighboring nodes that each new node connects to) varying from 2 to 10. In the following sections, we use PowerDn to denote a power-law topology with edge density n. For each power-law topology, we use two bandwidth settings: all links are either OC3 or OC48. The propagation delay of each link is drawn uniformly between $0 - 10$ ms.

5.2 Traffic demands

We use both real and synthetic traffic demands in our evaluation.

Real traffic demands: Our real traffic demands are estimated from SNMP link data using the *tomogravity* method [44], which has been shown to yield accurate estimates especially for large traffic matrix elements. We use the backbone router to backbone router traffic matrices during three randomly chosen hours in November 2002.

Synthetic traffic demands: The real traffic demands are only available for $ISPTopo$. For the other topologies, we generate synthetic traffic demands as follows. For a Rocketfuel topology, we generate synthetic traffic by randomly mapping POPs in $ISPTopo$ to non-leaf nodes in the Rocketfuel topology, using several different random seeds. Specifically, let $m(.)$ denote a random mapping from the cities in $ISPTopo$ to those in a Rocketfuel topology. Let $T(s,d)$ denote the traffic demand from city s to city d in $ISPTopo$. Then the traffic demand from city $m(s)$ to city $m(d)$ in the topology under study is set to $T(s,d)$. For synthetic power-law topologies, we perform similar mappings at the router level to derive demands.

Load scale factor: To control system load, we scale up the demands so that when all the traffic is compliant and routed based on *shortest hop-count*, the maximum link utilization is $100 \cdot F\%$, where F is a *load scale factor* (sometimes abbreviated as LSF).

5.3 Link latency functions

As shown in [30], link latency functions play an important role in determining the effectiveness of selfish routing. In our evaluations, we use five representative latency functions: M/M/1, M/D/1 [18], P/M/1, P/D/1 [19], and BPR [9]. We also implement piecewise-linear, increasing, convex functions to approximate any other latency functions. In all latency functions, we include a term for propagation delay (Section 5.1 shows how we determine its value for each physical link).

Our first two latency functions belong to the general M/G/1 class of latency functions: M/M/1 and M/D/1. For a M/G/1 queue, the latency can be expressed as $l(x) = \frac{1}{\mu} + \frac{x \cdot (1+\sigma^2\mu^2)}{2\mu(\mu - x)} + prop$, where x is the traffic load, μ the link capacity, σ the standard deviation of the service time, and $prop$ the propagation delay. The M/M/1 latency function is M/G/1 with $\sigma = \frac{1}{\mu}$; therefore $l(x) = \frac{1}{\mu - x} + prop$. The M/D/1 latency function is M/G/1 with $\sigma = 0$; therefore $l(x) = \frac{0.5}{\mu - x} + \frac{0.5}{\mu} + prop$. To avoid the discontinuity when the load approaches capacity, we approximate the M/M/1 or M/D/1 function with a linear function beyond 99% utilization. To test sensitivity to the threshold, we also try 90% and 99.9%. The results are very similar, and in the interest of brevity we present the results using 99% as the threshold.

Our next two latency functions, P/M/1 and P/D/1, have heavy-tail inter-arrival times. Here P stands for Pareto. We set the shape parameter $\beta = 1.5$ so that the resulting distribution has infinite variance. Since there is no closed-form expression for either P/M/1 or P/D/1, we approximate each of them using a piecewise-linear, increasing, convex function. We use the results in [19] to approximate P/M/1. For P/D/1, we derive a linear approximation of its shape using ns-2 [27] simulations. Specifically, we generate Pareto traffic to compete for a single bottleneck link with a large FIFO drop-tail queue and observe the latency as we vary the load.

For comparison purposes, we also run some experiments with the latency function BPR [9], which is used as a standard latency function in transportation networks. The expression for this latency function is $l(x) = prop \cdot \left[1 + 0.15\,(\frac{x}{\mu})^4\right]$. Table 2 summarizes the above five latency functions.

Notation	Latency function
M/M/1	$l(x) = \frac{1}{\mu - x} + prop$
M/D/1	$l(x) = \frac{0.5}{\mu - x} + \frac{0.5}{\mu} + prop$
P/M/1	approx. with Pareto $\beta = 1.5$, see [19]
P/D/1	approx. with Pareto $\beta = 1.5$
BPR	l(x) = $prop \cdot \left[1 + 0.15\,(\frac{x}{\mu})^4\right]$

Table 2: Link latency functions.

5.4 Performance metrics

We use the following performance metrics to evaluate routing efficiency: (i) average latency, (ii) maximum link utilization, and (iii) network cost. The first metric reflects end-to-end user performance, while the next two reflect the perspective of network operators, who aim to avoid link overloads in their networks. These performance metrics are computed from traffic equilibria, as we discussed in the previous section.

The utilization of a link is the amount of traffic on the link divided by its capacity. When a link utilization is beyond 100%, the link is overloaded. The maximum link utilization is the maximum utilization over all links in a network.

The maximum link utilization is an intuitive metric; however, it is dominated by a single bottleneck, as pointed out in [14]. To get a more complete picture, we also adopt a metric to capture the overall network cost. According to [14, 15], the cost of a link can be modeled using a piecewise-linear, increasing, convex function with slopes specified as follows:

$$u_e(x/c) = \begin{cases} 1 & : \quad x/c \in [0, 1/3) \\ 3 & : \quad x/c \in [1/3, 2/3) \\ 10 & : \quad x/c \in [2/3, 9/10) \\ 70 & : \quad x/c \in [9/10, 1) \\ 500 & : \quad x/c \in [1, 11/10) \\ 5000 & : \quad x/c \in [11/10, \infty), \end{cases}$$

where x is the load on link e, and c its capacity. We refer to the points at which the slope changes (*e.g.,* $1/3$ and $2/3$) as the cut-points. The overall network cost is the sum of all links' costs. In [14], Fortz, Rexford, and Thorup showed that OSPF weights derived from one set of cut-points and slopes also tend to give good performance for other sets of cut-points and slopes. Therefore the above cost function is a general metric to consider.

For all three metrics, the lower values are preferred.

6. SELFISH SOURCE ROUTING

We first investigate the performance of selfish source routing; that is, all the demands are infinitesimal and the selfish traffic can use any routes in the physical network. This is the type of selfish routing scheme analyzed in most theoretical studies. As shown in [30], the worst-case latency degradation of selfish source routing compared with optimal routing can be unbounded due to lack of cooperation. In this section, we seek answer to the following question: how does selfish routing perform in Internet-like environments?

6.1 Are Internet-like environments among the worst cases?

Effects of network load: We begin our investigation of selfish routing by varying network load. Figure 2 shows the latency for three representative topologies, as we vary the network load scale factor from 0.2 to 2.

We make the following observations. First, under various loads, selfish routing yields lower latency than compliant routing, which is based on optimized-compliant OSPF weights. This result complements the previous findings, such as Detour [33] and RON [5], and shows that the performance benefit of selfish routing over compliant routing exists even in a single AS network; moreover such benefit does not disappear even if all traffic is selfish (as opposed to just having a small portion of selfish traffic in RON). It is not surprising that compliant routing results in higher latency, because the OSPF weights are optimized mainly to avoid link overloads rather than minimize end-to-end user latency. As we will see later, the lower latency of selfish routing comes at the cost of increased congestion on certain links.

Second, compared with optimal routing, selfish routing yields very similar average latency—the difference is close to 0 in most cases and is always within 30%. In other words, unlike the theoretical worst cases, the price of anarchy in Internet-like environments is close to 1. This is likely because under realistic network topologies and traffic demands, traffic is spread across the network and only a few links get congested even with selfish routing. As a result, the average latency under selfish routing is similar to that of optimal routing.

Effects of network topologies: Next we examine the effects of network topologies on the latency of selfish routing. Figure 3 compares the latency of different routing schemes when the link latency function is M/M/1, the load scale factor is 1.0, and the links' bandwidth is OC3.

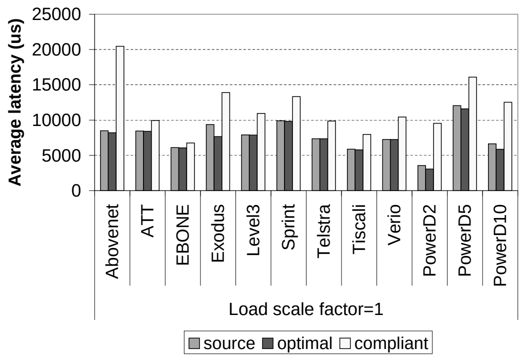


Figure 3: User latency for all topologies with the M/M/1 latency function and load scale factor 1. Selfish stands for selfish source routing; optimal stands for optimal routing; compliant stands for optimized-compliant OSPF routing. The other figures in this section use the same notation.

As Figure 3 shows, network topologies have a pronounced effect on the relative performance of selfish and compliant routing. For example, in the Abovenet and power-law topologies, the latency achieved by selfish routing is less than half of that incurred by compliant routing. A detailed look at these two topologies shows that these two topologies have mesh-like connectivity; therefore, selfish routing is likely to find more paths and therefore achieves much lower latency. However, in all topologies, we observe that selfish routing consistently yields close to optimal latency.

Effects of latency functions: Finally, we study how different latency functions affect the latency of selfish routing. From Figure 4, we observe similar latency across different latency functions. When comparing the latency achieved by different routing schemes, we see that the performance of selfish routing is close to that of optimal routing and noticeably better than that of compliant routing.

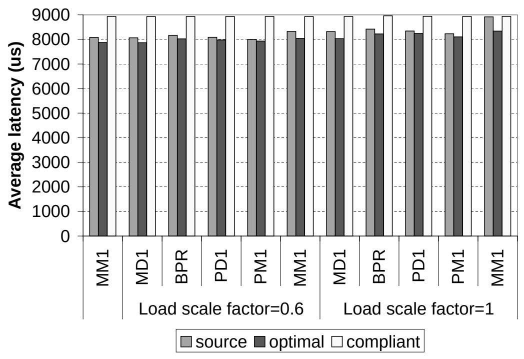
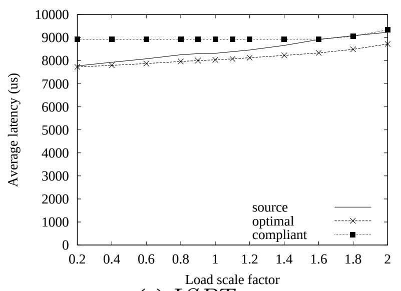


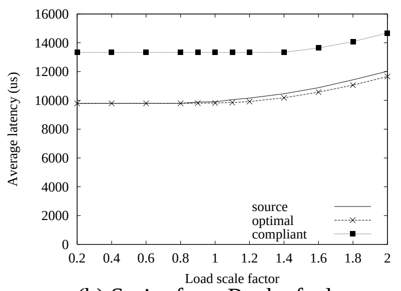
Figure 4: User latency for $ISPTopo$ under various latency functions.

6.2 What is the system-wide cost for selfish source routing?

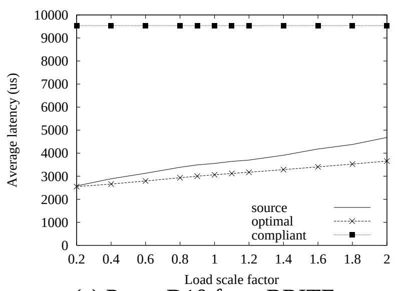
The previous subsection shows that unlike theoretical worst cases, selfish source routing in Internet-like environments incurs low latency. A natural question is whether the low latency comes at the expense of increased system-wide cost. We examine this issue by comparing different routing schemes based on two metrics: (i) max-



(a) $ISPTopo$

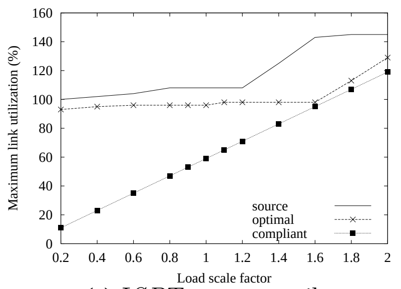


(b) Sprint from Rocketfuel

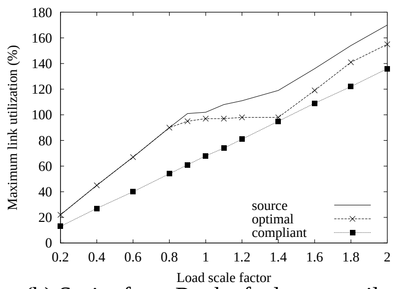


(c) PowerD10 from BRITE

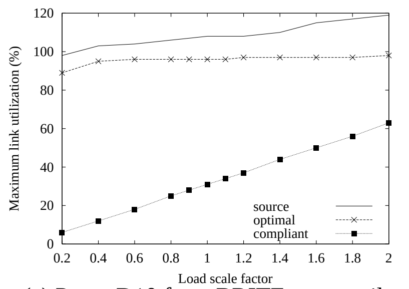
Figure 2: Selfish source routing: comparison of user latency using M/M/1 link latency under various network loads.



(a) $ISPTopo$, max. util.



(b) Sprint from Rocketfuel, max. util.



(c) PowerD10 from BRITE, max. util.

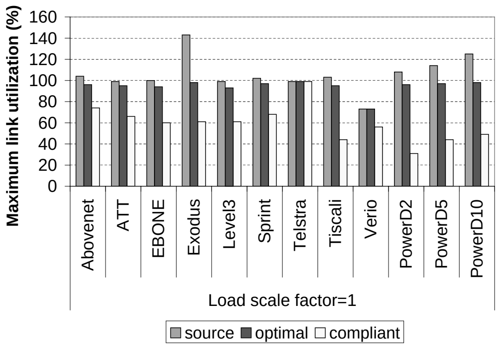
Figure 5: Selfish source routing: comparison of maximum link utilization using M/M/1 link latency under various network loads.

imum link utilization and (ii) network cost, both defined in Section 5.4.

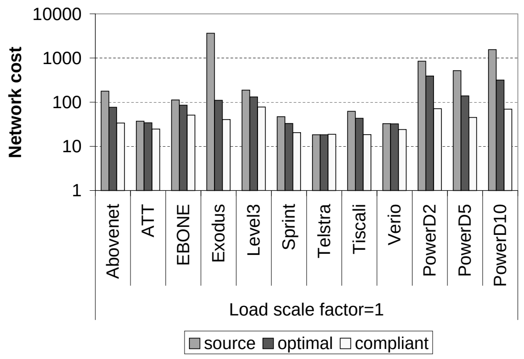
Effects of network load: We start by examining the impact of network load. Figure 5 shows the maximum link utilization for the same network configurations as those in Figure 2. From Figure 5, we observe that in compliant routing, maximum link utilization increases linearly with offered load. This is expected since we use the same set of weights to scale the traffic (see Section 5.2). In comparison, both optimal routing and selfish routing can cause high link utilization even when the overall offered load is low. For example, in both $ISPTopo$ and PowerD10 topologies, at a load factor of 0.2, the maximum link utilization of optimal routing is close to 90% and that of selfish routing is close to 100%. This result occurs because both optimal routing and selfish routing aim to choose shortest paths; thus they are more likely to cause congestion there, whereas compliant routing more uniformly spreads traffic across the entire network to avoid link overloads at the cost of longer end-to-end paths. The high network utilization is undesirable, since many backbone networks are kept at a load well below 50% so that there are enough backup paths during link or router failures [20].

Effects of network topologies: Next we verify the above observations by varying the network topologies. As shown in Figure 6, selfish routing consistently yields the highest maximum link utilization and network cost in all topologies. For example, in the Exodus network, the maximum link utilization achieved by selfish routing is 40% higher than that of optimal routing and 80% higher than that of compliant routing; for the same network, the network cost of selfish routing is over an order of magnitude higher than that of optimal routing or compliant routing. These results suggest that selfish routing may make a network much more vulnerable to overload, especially when failures occur.

Effects of latency functions: The results based on other latency functions are qualitatively the same, as shown in Figure 7. Since both latency and network cost/utilization are not very sensitive to latency functions for the topologies that we consider, in the following sections we focus on the M/M/1 latency function. Moreover, we show only the maximum link utilization, since it is more intuitive and it gives consistent results as network cost.



(a) Maximum link utilization (%)

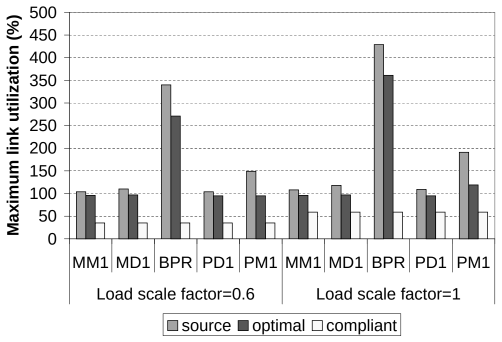


(b) Network cost

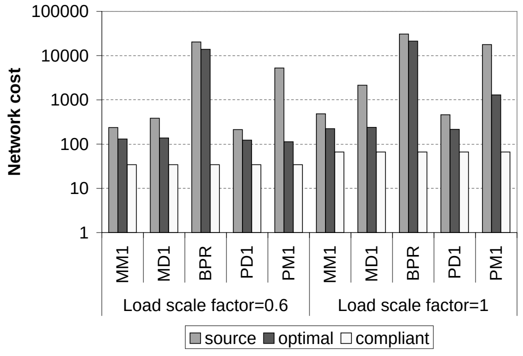
Figure 6: Selfish source routing: comparison of maximum link utilization and network cost using M/M/1 link latency across different network topologies.

6.3 Summary

To summarize, in this section we compare the performance of different routing schemes using realistic network topologies and traffic demands. Our results show that unlike the theoretical worst cases,



(a) Maximum link utilization



(b) Network cost

Figure 7: Selfish source routing: comparison of maximum link utilization and network cost across different latency functions.

selfish source routing in Internet-like environments is very effective in choosing shortest paths, and yields close to optimal average latency. On the other hand, this often comes at the cost of overloading links on the shortest paths. This suggests that selfish routing may potentially have a negative impact on traffic engineering. We will further investigate the issue in Section 9.

7. SELFISH OVERLAY ROUTING

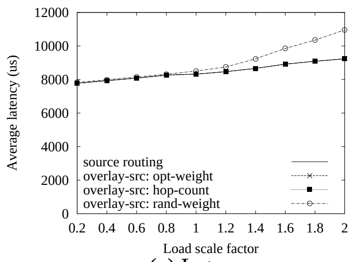
The previous evaluations consider selfish source routing. However, as we discussed in Section 1, in practice, end users often do not have complete routing control. We initially expected that reducing routing flexibility would increase both latency and link utilization, since users lose fine-grained control over routing. However, as we will see, this is often not the case.

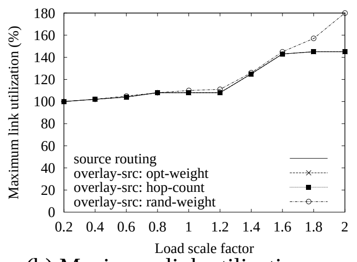
7.1 Does selfish overlay routing perform well when every node is in the overlay?

We first consider an overlay that consists of all network nodes. Note that even if the overlay includes all network nodes, routing on an overlay is still different from routing on the physical network in that the latter has access to all network resources, but this may not be the case for the former. For example, the network-level routing can easily prevent any overlay traffic from using a particular link by setting its corresponding column in the routing matrix to 0 (in OSPF this can be achieved by assigning a large weight to the link). As a result, certain physical routes cannot be implemented by any overlay routing schemes.

In our evaluation, we use the same network setting as before, except that the routes between any pair of overlay nodes are no longer determined by end users, but by the network-level routing. We adopt OSPF for network-level routing and use the three OSPF weight assignments as described in Section 3.

Figure 8 shows the performance of overlay source routing for the $ISPTopo$ network, as we vary network load. In both figures, three of the four curves overlap, namely source routing, overlay source routing when the network-level routing uses optimized-compliant OSPF weights, and overlay source routing when the network-level routing uses hop count. This suggests that routing constraints, whether based on hop-count or optimized-compliant weights, have little effect on user latency or system-wide cost. This result came as quite a surprise since our initial conjecture was that routing constraints would degrade performance. In contrast, when the network-level routing uses random weights, we observe much higher delay and link utilization. To understand this result, below we introduce a notion called *direct link shortest (DLS)*.





(a) Latency (b) Maximum link utilization

Figure 8: Selfish overlay routing: comparison of user latency and maximum link utilization for the $ISPTopo$ topology.

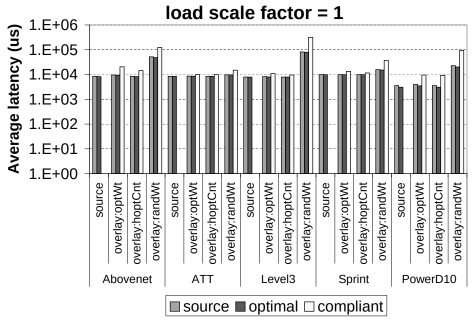
DEFINITION 1 (DIRECT LINK SHORTEST). We consider a network-level routing scheme to be *direct link shortest (DLS)*, if for any physically adjacent nodes A and B, all the traffic from A to B is routed through the direct link $\overline{AB}$ without involving any other links. As an example, hop-count-based OSPF is a DLS routing scheme.

Our key observation about DLS routing schemes is as follows. In an overlay that covers all network nodes and satisfies DLS, routing on the overlay has as much routing flexibility as directly routing on the underlying physical network. This is because, by definition of DLS, the overlay can force traffic to follow any given physical path $\overline{N_1 N_2 ... N_k}$ by specifying an overlay path with the same node sequence: $N_1 \rightarrow ... \rightarrow N_k$, where nodes N_i and N_{i+1} are physically adjacent. Given this observation, since hop-count-based OSPF satisfies DLS, it performs as well as source routing. As for optimized-compliant OSPF weights, our verification shows that such weights satisfy DLS to a large extent, thus it also performs well.

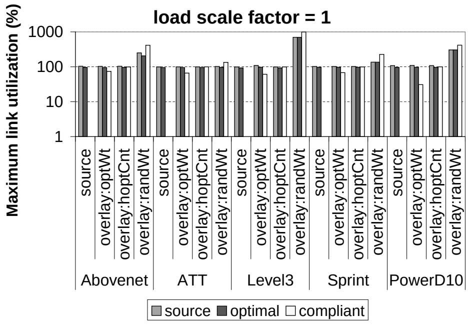
One implication of the above observation is that the only way in which a network-level routing scheme can affect the amount of selfish overlay traffic on a given link $\overline{AB}$ is by violating DLS. In the context of OSPF, this can only be achieved by choosing the weights so that an alternative path from node A to B has a total weight that is either lower than or equal to the OSPF weight of $\overline{AB}$. When the alternative path has a lower total weight, $\overline{AB}$ is effectively pruned from the network, since no overlay traffic can ever use it. When there is a tie, some load balancing can be achieved. However, such ties are very rare in our experiments. Therefore, such violations of DLS effectively reduce the network resources available to the selfish overlay and can lead to higher latency and link utilization.

With random OSPF weights, violations of DLS are common and therefore the network resources available to the overlay are significantly reduced. This explains why we see substantially higher latency and maximum link utilization with random OSPF weights. We will show later in Section 9 that selfish overlay routing interacts poorly with OSPF optimizer for exactly the same reason.

We further verify the above observations by using different network topologies; the results are shown in Figure 9. As before, random OSPF weights continue to yield substantially higher delay and maximum link utilization, while the performance of the other three is close to each other. This confirms our previous findings. When comparing the performance across different routing schemes, we observe that selfish routing continues to result in close to optimal average latency. Moreover, it yields noticeably lower latency than compliant routing in most cases. However, this lower latency often comes at the cost of higher maximum link utilization.



(a) Latency



(b) Maximum link utilization

Figure 9: Selfish overlay routing: comparison of user latency and maximum link utilization for different network topologies.

7.2 Does selfish overlay routing perform well when only some nodes are in the overlay?

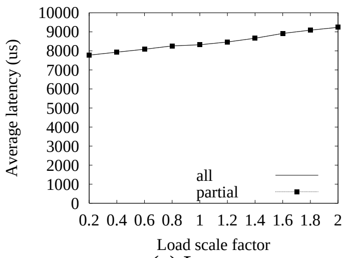
The previous evaluation includes all of the network nodes in an overlay. In practice, an overlay may only have *partial coverage*, *i.e.*, only a fraction of the nodes are in the overlay. In such a case, the routing choice is further constrained, which may have an impact on the performance. Below we investigate this issue in detail.

Effects of only covering edge nodes: In our first experiment, we form an overlay from all of the edge nodes in $ISPTopo$ and route all demands among these edge nodes through the overlay. We then compare the performance with what we achieve when the same set of demands is routed through an overlay that includes all of the network nodes. As shown in Figure 10, the curves of full overlay coverage almost completely overlap with those of partial coverage, in terms of both latency and maximum link utilization. These results are likely due to the fact that the Internet backbone is fairly well-connected and well-provisioned; therefore, even though end users can only forward traffic through edge nodes, they do not lose much flexibility in controlling their routes.

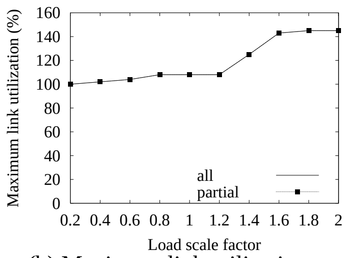
Effects of random partial coverage: In our second experiment, we uniformly choose a fraction of network nodes to form an overlay and vary the fraction from 20% to 100%. As before, partial overlay coverage yields similar latency compared to full overlay coverage. On the other hand, as shown in Figure 11, full overlay coverage incurs a slightly higher maximum link utilization than partial coverage, because as more nodes and links are included, it becomes more likely that the overlay has popular shortcuts, which get overloaded.

7.3 Summary

To summarize, in this section we investigate the effects of overlay routing constraints. We show that if the physical network uses a routing scheme that satisfies direct link shortest (DLS), the overlay



(a) Latency



(b) Maximum link utilization

Figure 10: Effects of partial coverage ion the performance of selfish overlay routing. Here edge nodes in $ISPTopo$ belong to an overlay, and OSPF weights are set according to hop count.

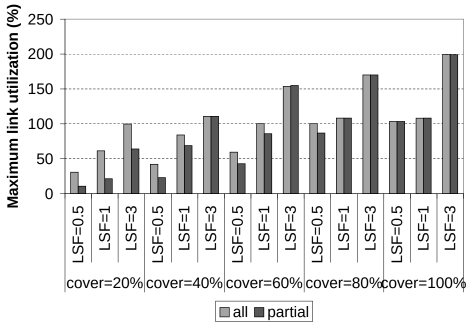


Figure 11: Effects of partial coverage in $ISPTopo$ with random node selection on maximum link utilization.

has full control over how its traffic is routed through the physical network. In the context of OSPF, the only way in which OSPF can affect overlay traffic is by violating DLS, which effectively reduces network resources and may therefore degrade both user and system-wide performance. We also show that like source routing, overlay source routing reduces latency at the expense of higher network cost. Finally, we observe that the effects due to partial coverage are small in backbone topologies.

8. INTERACTIONS AMONG COMPETING OVERLAYS

So far we have only considered either a large number of independent, small users using source routing (Section 6) or a single selfish overlay (Section 7). In practice, it is possible that multiple overlays and background traffic will share the same physical network, and these different traffic will compete against one another for the shared network resources. We call such interactions *horizontal interactions*.

8.1 What is the relative competitiveness of two routing schemes?

We start by looking at the interactions between any two types of traffic. The objective of this subsection is to evaluate the "friendliness" of different types of routing schemes. We use R_1/R_2 to denote that the routing scheme of the foreground traffic is R_1, and that of the background is R_2. Here R_i is either overlay source routing, overlay optimal routing, or compliant routing. We evaluate the interactions through four sets of experiments.

Effects of network topologies: First, we study how traffic using different routing schemes compete against each other in different topologies. In this set of experiments, we put the competing demands at the same nodes, and we set both the foreground and background traffic to be 50%. In other words, the two types of competing traffic have the same amount of traffic and the same set of overlay nodes. Figure 12 shows the results. We make two observations. First, the performance difference between compliant routing and the competing overlay routing scheme varies across different topologies. For example, the performance difference is larger in the Abovenet and power-law topologies. This is consistent with Figure 3 and can again be explained by the better connectivity of these topologies (see Section 6.1 for details). Comparing the results in Figure 12 with those in Figure 3, we observe that the latency of the compliant traffic is not substantially increased, which indicates that selfish routing does not hurt the performance of compliant routing in this environment. Second, overlay source routing achieves similar performance compared to overlay optimal routing. This suggests that the performance gain of cooperative overlay optimal routing over uncooperative overlay source routing is not significant.

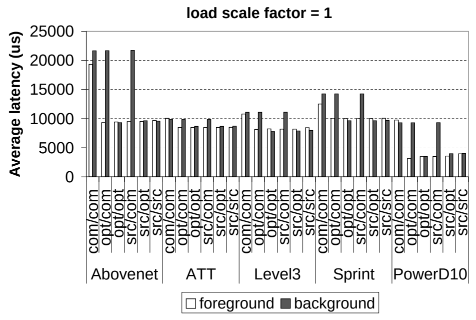


Figure 12: Coexistence of two routing schemes: varying network topologies.

Effects of network-level routing schemes: Second, we explore the impact of network-level routing schemes on the horizontal interactions as follows. We set both the foreground and background traffic in $ISPTopo$ to be 50%, and we vary how OSPF weights are set. As shown in Figure 13, the foreground and background traffic experience similar latency in most cases, except when OSPF weights are set randomly. When OSPF weights are set randomly, compliant traffic incurs about twice as much delay as that of the competing overlay source routing or overlay optimal routing. This indicates that inappropriate OSPF weights can significantly degrade the performance of compliant traffic. In comparison, a selfish overlay is able to reduce the latency of its traffic, as it looks for better alternative paths. Interestingly, this also has a positive side effect: it helps to reduce the load on the links used by the competing compliant traffic, thereby cutting the latency of the latter by half. When the network-level routing scheme is configured reasonably, different overlay routing schemes can coexist well.

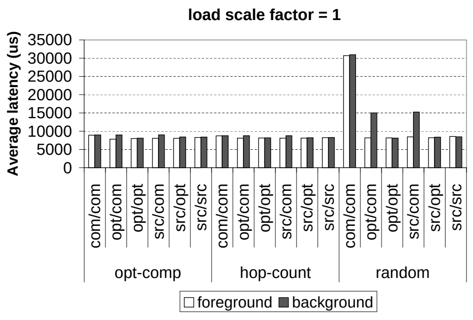
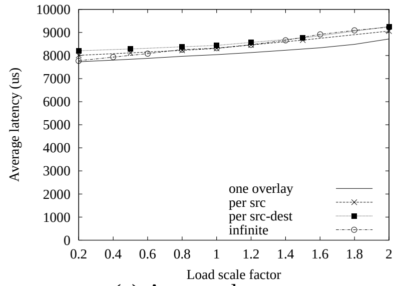


Figure 13: Coexistence of two routing schemes: varying OSPF weights in $ISPTopo$.

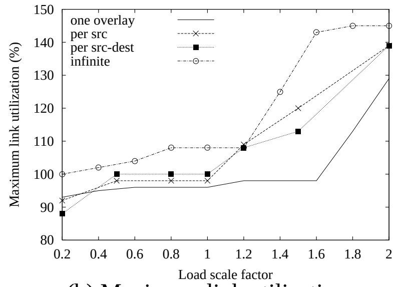
Effects of network load and traffic distribution among overlays: We further examine the performance of two competing overlays as we vary the network load, or vary the fraction of foreground traffic. In both cases, we observe consistent results: selfish routing out-performs compliant routing without hurting the latter.

8.2 Can many overlays coexist well?

Next we study horizontal interactions by varying the number of overlays. Each overlay uses overlay optimal routing and covers all network nodes. Figure 14 shows the result for $ISPTopo$, when the number of overlays is changed in the following ways: (i) one overlay, which includes all the demands; (ii) overlay per source, where each overlay includes all demands originated from a source; (iii) overlay per source-destination pair, where each overlay includes all demands between a source and destination pair; (iv) an infinite number of overlays, where each overlay has infinitesimal demand. We use the relaxation framework specified in the Appendix to compute the traffic equilibria for (ii) and (iii). For (iv), we note that having an infinite number of overlays with infinitesimal demands is equivalent to having all the infinitesimal demands on a single overlay, each of which tries to minimize its own latency. In other words, (iv) is equivalent to having a single overlay using overlay source routing. Thus we do not need to use the relaxation framework. From Figure 14, we observe that there is only a slight difference in user latency due to variations in the number of overlays. Results from other topologies confirm this finding, which suggests that performance degradation due to competition among overlays is not significant.



(a) Average latency



(b) Maximum link utilization

Figure 14: Coexistence of multiple overlays in $ISPTopo$.

8.3 Summary

To summarize, with reasonable OSPF weights (*e.g.*, hop-count), different routing schemes can share network resources reasonably well without hurting each other; with bad OSPF weights, selfish overlays improve performance both for themselves and for compliant traffic. Note that these results are consistent with previous findings (by Zegura *et al.* [43]) that selfish routing co-exists well with non-selfish routing in the context of server selection.

9. SELFISH ROUTING VS. TRAFFIC ENGINEERING

So far all of our experiments assume that the network-level routing is fixed. We find that while selfish routing can achieve close to optimal latency, it often increases maximum link utilization and network cost. In practice, the network-level routing may be constantly changing since one principal goal of traffic engineering is to reduce network cost by adapting the network-level routing in response to varying traffic patterns. This motivates us to examine the interactions between selfish routing and traffic engineering, which we term *vertical interactions*. More specifically, we ask the following basic question: *Will the system reach a state with both low latency and low network cost, as selfish routing and traffic engineering each tries to minimize its own cost function by adapting to the other process?*

Below we evaluate vertical interactions in the context of OSPF and MPLS route optimization. As we will see, OSPF route optimization provides little control over selfish traffic and as a result, the system performance, both in terms of user latency and network cost, is no better than using hop-count-based OSPF routing. In contrast, MPLS provides fine-grained control and can potentially lead to better performance.

9.1 Specification of vertical interactions

We specify vertical interactions as an iterative process between the two players: traffic engineering and selfish overlays.

Traffic engineering adjusts physical routing based on network traffic patterns, which are usually in the form of a traffic matrix. More specifically, let T_t denote the estimated traffic matrix for time slot t, then $T_t(s, d)$ represents the total traffic from source s to destination d during the time slot t. Traffic engineering takes T_t as input, and computes a routing matrix R_t to optimize network performance. For our study, we assume T_t is given. In reality, T_t can either be obtained through direct measurements [12] or be estimated based on link loads [44].

Selfish routing interferes with traffic engineering by changing the traffic matrix. More specifically, after traffic engineering installs the routing matrix R_t to the network, selfish routing will respond and redistribute traffic through overlay nodes, which leads to a new traffic matrix T_{t+1}. This process repeats.

Figure 15 specifies the process of vertical interactions. We also add a relaxation option in the hope of improving stability; however, our results show that it does not yield much performance improvement. Thus, in the interest of brevity, below we only present the results of traffic engineering without relaxation.

```
▷ $T_t$ is the estimated traffic matrix at time $t$.
▷ $T_t^*$ is the real traffic matrix at time $t$.
▷ $R_t$ is the routing matrix at time $t$.
▷ Assume $\sum_t \alpha_t \to \infty$; $\alpha_t \to 0$ as $t \to \infty$.

$T_t^*$ = Traffic matrix when routing matrix is $R_{t-1}$
if (relaxation)
    $T_t = (1 - \alpha_t)T_{t-1} + \alpha_t T_t^*$
else
    $T_t = T_t^*$
$R_t$ = OptimizedRoutingMatrix($T_t$)
Traffic engineering installs $R_t$ to network
Selfish routing redistributes traffic to form $T_{t+1}^*$
```

Figure 15: One round during vertical interaction.

9.2 Does selfish routing work well with OSPF optimizer?

We first evaluate vertical interactions when the route controller uses OSPF. In all of our experiments, the traffic engineering process uses an OSPF optimizer to optimize link weights as described in [14], and the starting routing matrix of the interactions is computed using hop-count-based OSPF. We choose this starting point to model a scenario in which selfish routing initially has full control over the routing of its traffic in the physical network (see Section 7), and then the network decides to start using traffic engineering.

Figure 16 shows the dynamics of vertical interactions for the Sprint topology. The results indicate that the response of OSPF traffic engineering could yield considerably worse performance than compliant routing using optimized-compliant OSPF weights (*i.e.*, traffic engineering without selfish traffic), and worse than overlay source routing on top of hop-count-based OSPF (*i.e.*, selfish routing without traffic engineering). We observe qualitatively similar results as we vary network topologies, the fractions of selfish traffic, and the sizes of selfish overlays.

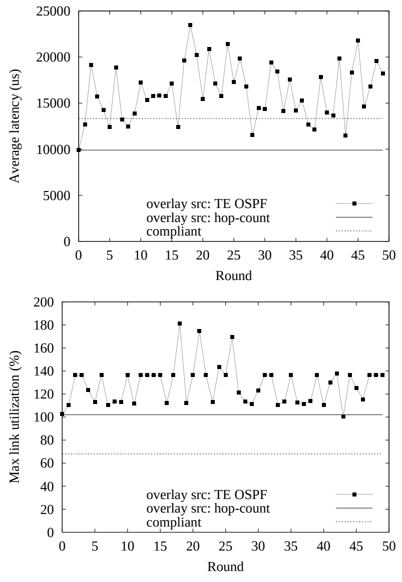


Figure 16: Vertical interaction with OSPF optimization for the Sprint topology.

These results suggest that the interactions between the two separate routing control processes is so ineffective that each individual control process, when applied alone, can yield better performance than having such interactions.

Such inefficiency is partly due to the fact that the adaptive nature of selfish traffic creates considerable variability in traffic demands and therefore makes it harder to do traffic engineering. Another important reason is the limited control of OSPF over selfish overlay traffic. Recall in Section 7 we have shown that when all network nodes belong to an overlay, the only way in which OSPF can affect the selfish overlay traffic is by violating DLS, which effectively reduces available network resources. As a result, both latency and network cost could be worse than those of hop-count-based OSPF, which gives the overlay full access to all available network resources.

9.3 Does selfish routing work well with MPLS optimizer?

The poor interactions between selfish routing and the OSPF optimizer motivates us to look for alternative solutions. In this subsection, we examine vertical interactions between selfish routing and the MPLS optimizer, which allows one to implement general multi-commodity routing. Given a traffic matrix and a piece-wise linear, increasing, convex network cost function, the MPLS optimizer can find the optimal routing matrix R that minimizes the network cost by solving a linear programming problem [1, Chapter 17]. We have implemented such an optimizer based on `lp_solve` [24].

Figure 17 shows the average latency and maximum link utilization for the Sprint topology. We observe that the routing performance is noticeably better than that of OSPF. It allows the system to reach a state in which the network cost is close to that of optimal traffic engineering without selfish routing, and the average latency is only marginally higher than what selfish routing can achieve in the absence of traffic engineering. This is important because the traffic engineering process can choose to stop at any moment and settle on a routing matrix that gives a satisfactory result; that is, the traffic engineering process can be considered as a type of Stackelberg game.

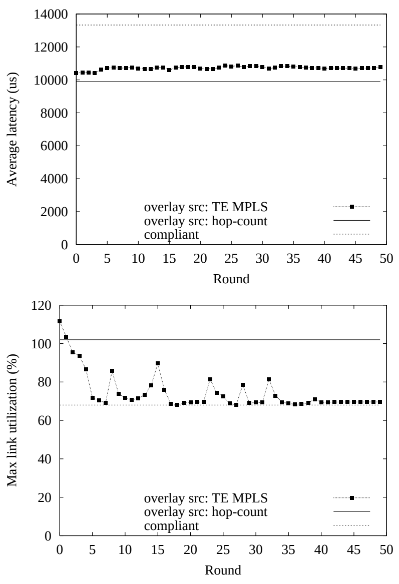


Figure 17: Vertical interaction with MPLS optimization for the Sprint topology.

We observe similar results on other topologies.

These results indicate that MPLS-based traffic engineering can interact much more effectively with selfish routing. This is likely due to the fact that MPLS has much more fine-grained control over selfish overlay traffic. Specifically, unlike OSPF, MPLS can adjust the routing matrix R without having to reduce available network resources.

Despite the encouraging results, however, we note that there are a number of practical challenges in applying MPLS-based traffic engineering, or traffic engineering in general, in the presence of selfish traffic. For example, in our evaluation we assume that we know the perfect traffic matrices, which need to be estimated in practice. The adaptive nature of selfish traffic can make it very difficult to accurately estimate traffic matrices. Another challenge is that MPLS-based traffic engineering requires solving a very large linear programming problem. For large networks, the problem may contain millions of unknowns, which is infeasible to solve using software available today. A thorough exploration of these subjects is outside the scope of this paper, so we defer it to future work.

9.4 Summary

To summarize, in this section we examine the interactions between selfish routing and traffic engineering. We find that OSPF route optimization interacts very ineffectively with selfish routing, largely due to its limited control over selfish traffic. In contrast, MPLS route optimization has more fine-grained control and therefore interacts with selfish traffic more effectively. However, further research is required to investigate such interactions in more detail.

10. CONCLUSIONS AND FUTURE WORK

In this paper, we use a game-theoretic approach to study the performance of selfish routing in Internet-like environments. Our results show that unlike the theoretical worst case, selfish routing in such environments achieves close to optimal average latency, when the network-level routing is static. On the other hand, such performance often comes at the cost of overloading certain links. Moreover, when selfish routing and traffic engineering each tries to minimize its own cost by adapting to the other process, the resulted performance could be considerably worse.

There are a number of avenues for future work. First, we would like to investigate how the multi-AS nature of the Internet affects the routing performance. There are a few challenges involved, including modeling inter-domain topologies, routing policies, and traffic demands, as well as handling larger topologies. Second, our study focuses on the performance at traffic equilibria. The dynamics of selfish routing, *i.e.*, how equilibria are reached, is an interesting question. In addition, we are interested in better understanding and improving the interactions between selfish routing and traffic engineering. Finally, we plan to study selfish routing with alternative performance metrics, such as loss and throughput.

Acknowledgments

We are grateful for helpful comments from Jennifer Rexford, Ellen Zegura, Jitendra Padhye, Ted Jewell, and the anonymous reviewers. We would also like to thank Stan Eisenstat, Joan Feigenbaum, Eric Friedman, and Yanbin Liu for helpful discussions.

11. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, New Jersey, 1993.

[2] A. Akella, S. Seshan, R. Karp, and S. Shenker. Selfish behavior and stability of the Internet: A game-theoretic analysis of TCP. In *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, Aug. 2002.

[3] E. Altman, R. E. Azouzi, and A. Vyacheslav. Non-cooperative routing in loss networks. In *Proceedings of Performance '02*, Rome, Italy, Sept. 2002.

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

[5] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of SOSP '01*, Banff, Canada, Oct. 2001.

[6] D. O. Awduche. MPLS and traffic engineering in IP networks. *IEEE Communication Magazine*, pages 42–47, Dec. 1999.

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

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

[9] A. Chen, D.-H. Lee, and R. Javakrishnan. Computational study of state-of-the-art path-based traffic assignment algorithms. *Mathematics and Computers in Simulation*, pages 509–518, 2002.

[10] A. Collins. The Detour framework for packet rerouting. PhD Qualifying Examination, Nov. 1998.

[11] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *Proceedings of ACM SIGCOMM '99*, Cambridge, MA, Aug. 1999.

[12] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford. Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE/ACM Transactions on Networking*, Jun. 2001.

[13] M. Florian and D. Hearn. *Network Routing*, chapter 6, Network equilibrium models and algorithms. Elsevier Science, 1995.

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

[15] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings of IEEE INFOCOM '00*, Tel Aviv, Israel, Mar. 2000.

[16] E. Friedman. Selfish routing on data networks isn't too bad: Genericity, TCP, and OSPF. Working paper. Available from http://www.orie.cornell.edu//~friedman/papers.html, Oct. 2002.

[17] L. Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking*, 9(6), Dec. 2001.

[18] D. Gross and C. Harris. *Fundamentals of Queueing Theory*. John Wiley, 3rd edition, 1998.

[19] C. M. Harris, P. H. Brill, and M. J. Fischer. Internet-type queues with power-tailed interarrival times and computational methods for their analysis. *INFORMS Journal on Computing*, pages 261–271, 2000.

[20] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot. An approach to alleviate link overload as observed on an IP backbone. In *Proccedings of IEEE INFOCOM '03*, San Francisco, CA, Apr. 2003.

[21] 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.

[22] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, 1999.

[23] J. B. Krawczyk and S. Berridge. Relaxation algorithms in finding Nash equilibria. In Computational Economics from Economics Working Paper Archive at WUSTL, Jul. 1997.

[24] lp_solve. ftp://ftp.ics.ele.tue.nl/pub/lp_solve/.

[25] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: Boston University representative Internet topology generator. Available from http://www.cs.bu.edu/brite.

[26] Multiprotocol label switching (MPLS). http://www.ietf.org/html.charters/mpls-charter.html.

[27] The network simulator: ns-2. http://www.isi.edu/nsnam/ns/.

[28] Open shortest path first (OSPF). http://www.ietf.org/html.charters/ospf-charter.html.

[29] M. Patriksson. Algorithms for computing traffic equilibria. In *Networks and Spatial Economics*. 2003. http://www.cs.chalmers.se/˜mipat/LATEX/NSE.ps.

[30] T. Roughgarden. *Selfish Routing*. PhD thesis, Cornell University, May 2002.

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

[32] 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(1), pages 50–59, Jan. 1999.

[33] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *Proceedings of ACM SIGCOMM '99*, pages 289–299, Cambridge, MA, Aug. 1999.

[34] Y. Sheffi. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice-Hall, 1985.

[35] S. Shenker. Making greedy work in networks: A game-theoretic analysis of switch service discipline. *IEEE/ACM Transactions on Networking*, 3, 1995.

[36] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, Aug. 2002.

[37] J. W. Stewart. *BGP4: Inter-Domain Routing in the Internet*. Addison Wesley, 1998.

[38] L. Subrmanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *Proceedings of IEEE INFOCOM '02*, New York, NY, June 2002.

[39] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topology generators: Degree-based vs. structural. In *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, Aug. 2002.

[40] 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.

[41] S. Uryas'ev and R. Y. Rubinstein. On relaxation algorithms in computation of noncooperative equilibria. *IEEE Transactions on Automatic Control*, 39(6):1263–1267, Jun. 1995.

[42] X. Xiao, A. Hannan, B. Bailey, and L. Ni. Traffic engineering with MPLS in the Internet. *IEEE Network Magazine*, Mar. 2000.

[43] E. Zegura, M. Ammar, Z. Fei, and S. Bhattacharjee. Application-layer anycasting: A server selection architecture and use in a replicated web service. *IEEE/ACM Transactions on Networking*, 8(4), Aug. 2000.

[44] 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 '03*, Jun. 2003.

APPENDIX

In this Appendix, we give more details about the algorithms we use to compute traffic equilibria.

Computing traffic equilibrium for non-overlay traffic: We use the linear approximation algorithm (a variant of Frank-Wolfe algorithm) [13] to compute traffic equilibrium. The linear approximation algorithm is a gradient algorithm for solving non-linear optimization problems. Specifically, in each iteration we compute shortest paths, and use them to construct the gradient direction. We then move towards that direction by taking a step size that optimizes the objective function. The number of iterations is controlled by the stopping condition from [13]. When the link latency functions satisfy the monotonicity condition, which is the case for our latency functions, there is a unique equilibrium.

Computing traffic equilibrium for selfish overlay routing: Using the logical networks we described in Section 4, we can compute the traffic equilibrium of overlay routing by either a relaxation algorithm or a modified linear approximation algorithm.

Specifically, for a logical network that is asymmetric (*i.e.*, there are two logical links that share the same physical link but send different fractions of traffic through the physical link), we use Jacob's relaxation algorithm on top of Sheffi's diagonalization method [34] to determine the traffic equilibrium, since in this case we cannot formulate the equilibrium problem as an optimization problem. For a logical network that is symmetric (*i.e.*, not asymmetric; an example of a symmetric logical network is OSPF routing without equal weight splitting), we still can formulate the problem as an optimization problem by using a line integral to replace the normal summation of cost on each link. As a result, we still can use the linear approximation algorithm. Figure 18 specifies the structure of our algorithm. Note that for overlay networks, the traffic equilibrium may not be unique [21, 4, 7] and our algorithm identifies only one equilibrium.

```
▷ Assume $l_e(x)$ is increasing and convex for any edge $e$.
▷ Assume $xl_e(x)$ is convex for any edge $e$.
▷ If the overlay is latency optimal, $f = \sum_e xl_e(x)$;
▷     otherwise, $f = \oint l_e(x)$;

set other overlay's traffic as background traffic
repeat
        assume the current traffic vector on each edge is $x_t$
        determine link latency according to $x_t$
        use Dijkstra's algorithm to find all-or-nothing
           traffic assignment $y_t$
        use line search to find optimal $\lambda$ so that
           $f(x_t + \lambda(y_t - x_t))$ is minimal.
until (best lower bound gap $<$ threshold)
```

Figure 18: The linear approximation algorithm to compute the best response of source routing or overlay routing, when the network is symmetric, assuming the other overlay's traffic is background.

Computing traffic equilibrium for multiple overlays: Guaranteeing convergence poses a major challenge in computing traffic equilibrium when there are multiple overlays. To this end, we use the relaxation framework proposed in [23, 41] to ensure convergence to one equilibrium. Figure 19 shows the algorithm. The basic structure of the algorithm is that in each round, each overlay computes its best response by considering the other's traffic as background traffic. Then the best response and the previous state are merged using the relaxation factor α_t.

```
▷ $N$ is the number of overlays.
▷ $x_t(i)$ is a vector of overlay $i$'s traffic at round $t$.
▷ $y_t(i)$ is the best response of overlay $i$ at round $t$.
▷ Assume $\sum_t \alpha_t \to \infty$; $\alpha_t \to 0$ as $t \to \infty$.

repeat
        assume the traffic state is $x_t(i)$ of overlay $i$
        for each $i$
            computes its best response $y_t(i)$,
               assuming other overlays as background.
        for each overlay $i$
            set $x_{t+1}(i) \leftarrow (1 - \alpha_t)x_t(i) + \alpha_t y_t(i)$.
until (change between round $<$ threshold)
```

Figure 19: The relaxation framework to compute the traffic equilibrium of N overlays.