Understanding the Interaction Between Overlay Routing and Traffic Engineering

Honggang Zhang, Yong Liu, Weibo Gong, and Don Towsley

Abstract—

In this paper, we study the interaction between overlay routing and Multiprotocol Label Switching (MPLS) Traffic Engineering (TE) in a single Autonomous System (AS). We formulate this interaction as a two-player non-cooperative non-zero sum game, where the overlay tries to minimize the delay of overlay traffic and TE's objective is to minimize the network cost as a whole. Two types of games, a Nash game with best-reply dynamics and a static Stackelberg game, are studied. In a Nash routing game, overlay and TE are of equal status, and take turns to compute their optimal strategies based on the response of the other player in last round. We give analytical proof of existence, uniqueness and global stability of Nash equilibrium point (NEP) for a simple network. For general networks, we show that the selfish behavior of an overlay can cause huge cost increase and oscillations to the whole network. Even worse, we have identified cases, both analytically and experimentally, where the overlay's cost increases as the Nash routing game proceeds even though the overlay plays optimally based on TE's routing at each round. We propose that the overlay play as a leader in a Stackelberg game with TE to completely eliminate oscillations and optimize its own performance. We provide a gradient projection search heuristic to solve for Stackelberg strategy. We also discuss various practical issues, such as, time scale discrepancy between TE and overlay, overlay routing with limited information and the interaction between multiple overlay networks.

I. Introduction

There are two recent trends in network routing research. One is overlay routing, and the other one is Traffic Engineering (TE).

Overlay routing (*e.g.*, Detour [29], RON [7]) allows end hosts to choose routes by themselves. It is application level routing, where traffic is routed by application level routers (computers.) The logical paths and links of an overlay lie on top of physical paths set by intra-domain (*e.g.*, OSPF [1], MPLS [2], IS-IS [3]) and inter-domain routing protocols (*e.g.*, BGP [4].) It is shown that these overlay routing schemes are effective in dealing with some of the deficiencies in today's IP routing ([7][30] [29].) On the other hand, as pointed out by [18] and [8], Internet Service Providers (ISPs) are working towards better and robust intra-domain routings to adapt to the prevailing traffic through Traffic Engineering (TE).

In this paper, we are interested in overlay networks within a single ISP, and study the interaction between the routing of an overlay network and MPLS Traffic Engineering. Our work is motivated in part by the work of Qiu, Yang, Zhang, and Shenker [26], in which the interaction between overlay selfish routing and TE is brought up. However, our work is different in that [26] assumes each overlay user controls an infinitesimal amount of traffic demand and makes routing decisions independently. We study a single large scale and centrally controlled overlay network that controls a *non-negligible portion* of traffic demand and does optimal routing on application level. Akamai examplifies the type of overlay of interest to us. We further assume that the proportion of overlay traffic is significant enough to influence the routing decisions of TE. [1]

There is a fundamental mismatch between the objectives of an overlay routing and TE routing. An overlay is interested in the optimal routes for its own group of users. TE is interested in improving the whole network performance by considering all users including both overlay and non-overlay (or underlying) users. The overlay evaluates its cost or delay on each *logical* path and link, whereas, TE evaluates its cost on each *physical* path and link. The cost functions of overlay and TE could be different. Furthermore, the routing decisions of overlay (traffic flows on logical links) are essentially the input to TE (interpreted as traffic demands), and in turn, routing decisions of TE (traffic allocation on each physical link) will influence future routing decisions made by the overlay by affecting the costs or delays on logical links. Figure 1 shows conceptually how overlay and TE interacts with each other.

Since both overlay and TE optimize their routes over

Honggang Zhang, Yong Liu, and Don Towsley are with the Computer Science Department, University of Massachusetts at Amherst, MA 01003. E-mail: {honggang, yongliu, towsley}@cs.umass.edu.

Weibo Gong is with the Department of Electrical and Computer Engineering, University of Massachusetts at Amherst, MA 01003. E-mail: gong@ecs.umass.edu.

[1] There is no fixed cutoff value for this significant proportion, since it depends on how different the two objective functions are, and depends on the network topology and traffic patterns. Later in the paper, we will give more detailed discussions on this issue.

time, the interaction between their decisions can be understood as an iterative process ([26]). For example, initially, in the first round, overlay users allocate traffic among all logical paths (set routing decisions) based on current logical link delays. Each logical path might include several logical links. The traffic flow on a logical link between two nodes is interpreted by TE as traffic demand between these two nodes. In the second round, an ISP performs Traffic Engineering. It takes as input the traffic demand matrix (each demand pair includes traffic demand from underlying traffic and/or demand from overlay traffic), and computes a set of physical level routes using TE scheme such as [18] to minimize overall network cost or to minimize maximum link utilization. This process repeats itself. In this process, TE and overlay changes the input for each other in turn. Overlay routing decisions are the logical link traffic flows, which are in turn interpreted as traffic demands by TE, and on the other hand, TE changes the delays of logical paths by adjusting the actual physical-level routings of overlay traffic. This routing interaction process was first introduced by Qiu *et al* [26]. Some numerical results are given in [26], and the comment there is that MPLS TE interacts well with overlay routing, but OSPF TE interacts badly with overlay.

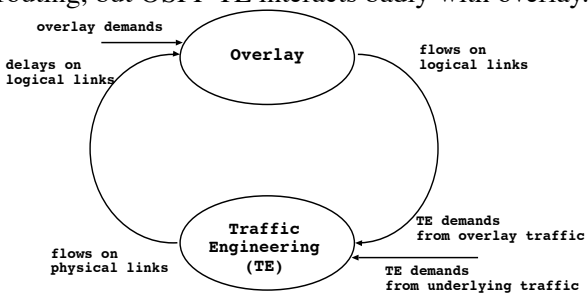

Fig. 1. Interaction between overlay optimizer and Traffic Engineering optimizer.

The central questions we address in this paper are on the *dynamics of this interaction process*. We formally model this interaction as a *non-cooperative non-zero sum two player game*. Overlay and TE are essentially two players with different objectives. From now on, we refer an overlay routing optimizer as *overlay*, and refer a TE routing optimizer as *TE*. In the interaction process, or best-reply dynamics, each player adjusts its response optimally based on the other player's decisions in the previous round. Given the mismatch between the objectives of overlay and TE, we ask questions such as: *does a Nash equilibrium exist in this game? If a Nash equilibrium exists, is it unique? How about the stability of Nash equilibriums? Does the interaction process always converge to a Nash equilibrium? What effects on the performance of TE and overlay can be caused by this interaction process? How does the information availability influences the strategies of both players and further influences the interaction process?*

In general, the objective of TE is to minimize the overall network cost or to minimize the maximal link utilization [18]. In this paper, we consider MPLS routing to achieve TE. Specifically, we consider minimizing overall network cost [18]. On the other hand, there are a variety of objectives for different overlay structures, we focus on an overlay optimizer to minimize the overall delay for its own traffic on top of the routings set by TE. We also address other overlay objectives, such as selfish routing ([28] and [26]) or user-equilibrium problem([17]).

The key contributions and results are summarized as follows.

First, we formulate and evaluate the routing interaction problem as a non-cooperative non-zero sum two player game. We focus on two types of games. In the first type of game, overlay and TE are equal in status, and the interaction process is best-reply dynamics. We call this type of game a *Nash routing game*. The general insights into this game are that overlay routing will never improve the performance of TE if TE uses MPLS (analytically proved), and in most cases, TE's cost will be increased a lot in the interaction with overlay, and the cost increase of TE is a function of the percentage of overlay traffic. If overlay traffic is about 50% of total traffic, then, overlay's influence on TE's performance achieves the largest. To illustrate the interaction process, for a simple network, we give an analytical proof on the existence and uniqueness of Nash equilibrium, and we prove that the interaction process always converges to the equilibrium. For general networks, the existence and uniqueness of Nash equilibria depend on network topology and traffic patterns. In addition, given that there is a Nash equilibrium, the game playing or interaction process may not converge to the that Nash equilibrium. The convergence property is problem specific.

One interesting finding is that under certain network condition, overlay's cost may increase when it plays a Nash routing game with TE. We give an analytical proof for a simple example. We have observed this also in experiments with a 14-node tier-1 ISP topology. Thus, it may not be wise for an overlay to always optimize its routes each time TE does physical routings, because in the long run, overlay's cost may be increased. This observation is of practical importance to an overlay routing structure, even though it is not surprising from a game-theoretic point of view because this is the inherent inefficiency characteristic of the NEP.

Second, in the Stackelberg game, we further assume that the overlay knows the optimization algorithms used

by TE, so that it can predict the response (physical level routings) from TE. For this game, we study the static Stackelberg strategy of overlay for the case that the overlay has higher status than TE. We call this type of game *Stackelberg routing game*. Our results show that, if an overlay plays (as a leader) a static Stackelberg game against TE, not only can oscillations be completely eliminated, but also the performance of overlay can be improved. Since solving a static Stackelberg game (a bi-level programming problem) is NP-hard, we give a heuristic to solve this Stackelberg routing game. This heuristic uses random start and gradient projection search. Our preliminary results show that it is very promising to use this heuristic to solve for the approximate Stackelberg routing strategy for an overlay network.

Third, in the previous two types of games, we assume overlay knows necessary information to play the Nash routing game and static Stackelberg routing games. In practice, an overlay most likely cannot access this information. So, we further study the interaction processes in which overlay has only limited information. According to the ways an overlay to measure and estimate necessary information and the ways to implement its routing decisions, we further study two types of overlay: a *one-step* overlay v.s. *incremental* overlay. We give a simple example to show that one-step overlay can lead bad oscillations and badly degraded performance for both players. For incremental overlay, we prove the existence of Nash equilibrium and the convergence of dynamic interaction process for a simple network. An numerical example is given to verify our results. In general, incremental algorithms are recommended to an overlay network routing optimizer. Even though oscillations could be decreased, they still exist in many cases because the fundamental problems are not solved by incremental algorithms. We discuss the tradeoffs between performance improvement and measurement costs to an overlay network.

Finally, we discuss issues on: the interaction between multiple overlays; frequency and time scale of game playing process; online learning in this interaction game.

The rest of the paper is organized as follows. In Section 2, related work is given. In Section 3, we formally model the interaction process as a two-person non-cooperative non-zero sum game. Nash routing game is given in Section 4, and static Stackelberg routing game is given in Section 5. In Section 6, we study games in which overlay has limited information. General discussions are given in Section 7. Conclusions are given in Section 8.

II. Related Work

Noncooperative games in the context of routing have been studied in the areas of transportation networks for a long time. In that framework, each user controls just an *infinitesimally small portion* of the network flow, and tries to minimize its own delay or cost. Dafermos and Sparrow [16] show that a simple transform of the cost function can make the routing game a standard network optimization problem, which is called *user equilibrium* model. On the contrary, a *system optimum* model has an objective to minimize the overall delay of the whole network. In the area of computer networks, the *user equilibrium* model is called selfish routing ([23][28][26].) Orda [25] and Korilis [22] studied a model in which users control *non-negligible portion* of flow. Orda [25] investigates the existence and uniqueness of Nash equilibrium in a routing game in which each user attempts to optimize its own performance by controlling its own portion of traffic. In [22], a a central manager is introduced into the model. Other related work can be seen in [6] and [27].

Our work differs from that on selfish routing ([26], [17], [28]) in that each user or player controls a *non-negligible portion* of flow in our work. Our work also differs from [25] and [22]. In our work, the view of network of each player (overlay and Traffic Engineering) is different. Overlay has a logical view of the network, whereas Traffic Engineering has a physical view of the network. In addition, each player's decision can change the input for the other player. The routing interaction problem was first studied by Qiu *et al* [26]. Two experimental studies were given to show the interaction between overlay and MPLS TE, and the interaction between overlay and OSPF TE. Our work starts from this base to formally model this interaction as a noncooperative game.

Solving for the static Stackelberg strategy is essentially a bi-level Programming problem, which in general is NP-Hard ([20], [9], [12], [33].) Inspired by decent method ([31], [21]) and projection methods for optimal routing in [14] and [13], we propose a gradient projection search heuristic.

III. Models of Interaction

A. Formulations of Traffic Engineering Optimizer and Overlay Routing Optimizer

Recently proposed Traffic Engineering, OSPF optimizer [18] or MPLS optimizer [26], are *load-sensitive* routing algorithms. They run periodically in response to load changes. In general, both OSPF and MPLS optimizers have the *same* objective, to reduce the total network cost. But they have different ways to reach that objective.

OSPF optimizer searches for a best set of link weights, and runs the OSPF protocol to achieve routings that can make the total network cost as close as possible to its objective. On the contrary, MPLS can directly set routings to reach its objective. MPLS can arbitrarily split traffic among available paths, but OSPF can only do all or nothing traffic allocation on available paths (traffic may be split evenly among paths which have the same cost.) Note, even though both optimizers have the same objective (i.e., same *expected* cost), they may end up with different *realized* costs because of the ways they use to set up routings. Theoretically speaking, MPLS can exactly achieve its objective, but OSPF cannot do so. We mostly focus on MPLS optimizer, and also discuss the effects if using OSPF optimizer.

We assume that the routing optimizer of an overlay minimizes the overall cost of overlay traffic.

In the following, starting from an example network ([26]) in Figure 2, we introduce our formal formulations of both the TE and overlay optimizers. All notations used in our formulations are given in Table I. There are three overlay nodes $1, 4, 7$. All overlay nodes in the overlay network are willing to forward traffic for demands originated from any other overlay node, even if the destination nodes are not overlay nodes. In this example, we have one overlay demand pair $d^{(1',9')}$ from 1 to 9. There are three logical paths for this overlay demand pair: path 1 is $1 \rightarrow 4 \rightarrow 9$; path 2 is $1 \rightarrow 9$; path 3 is $1 \rightarrow 7 \rightarrow 9$. Let $h_p^{(s',t')}$ denote the flow of overlay demand $d^{(s',t')}$ routed on logical path p. Then, overlay optimizer's job is to route this demand pair optimally among three logical paths, i.e., finding $\{h_1^{(1',9')}, h_2^{(1',9')}, h_3^{(1',9')}\}$. Note, logical paths are mapped into logical links. For example, path 1 is mapped onto logical links $(1, 4)$ and $(4, 9)$. Mapping coefficient $\delta_{(s',t',p)}^{(s,t)} = 1$ denotes that logical link (s, t) is on logical path (s', t', p) of overlay demand $d^{(s',t')}$.

Note, one logical path $h_a^{(s',t')}$ flow decided by overlay is mapped to logical link flows, which in turn are interpreted by TE as demand $d_{\text{overlay}}^{(i,j)}$, where (i, j) is any logical link on logical path a. In the sequel, *we represent the routing decisions of overlay by using either* $d_{overlay}^{(i,j)}$ *or* $h_a^{(s',t')}$. As a comparison, the demand seen by TE that comes from underlying traffic is denoted as $d_{\text{under}}^{(i,j)}$, and the total demand seen by TE at a node pair (i, j) is $d^{(i,j)} = d_{\text{overlay}}^{(i,j)} + d_{\text{under}}^{(i,j)}$. It is very important to keep these notations in mind while studying those games in the sequel.

In this example, if we assume at time t, overlay optimizer's routing decision is to put $d^{(1',9')}$ exclusively on path 1, then, $h_1^{(1',9')} = d^{(1',9')}$, $h_2^{(1',9')} = 0$, and $h_3^{(1',9')} = 0$.

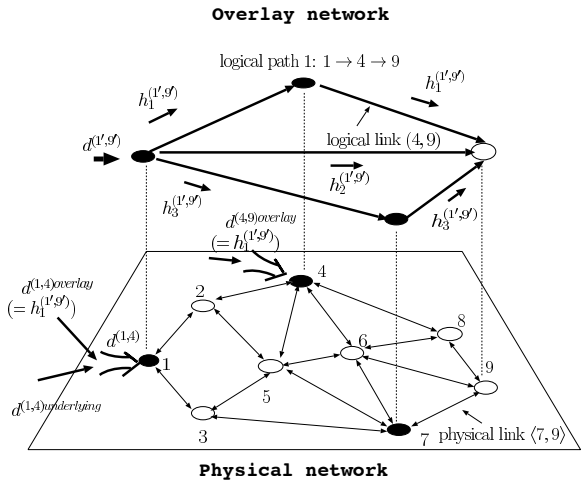

Fig. 2. An example overlay network.

Then, flow on logical path 1 is seen as two demand pairs by TE: $d^{(1,4)\text{overlay}}$ and $d^{(4,9)\text{overlay}}$. If there is an underlying traffic demand $d^{(1,4)\text{underlying}}$, then total TE demands are: $d^{(1,4)} = d^{(1,4)\text{overlay}} + d^{(1,4)\text{underlying}}$ and $d^{(4,9)} = d^{(4,9)\text{overlay}}$. TE optimizer's job is to route these two demand pairs optimally. Let $f_a^{(s,t)}$ denote the fraction of TE demand $d^{(s,t)}$ on physical link a, and let $v_a^{(s,t)}$ denote the flow on physical link a, then, $f_a^{(s,t)} = v_a^{(s,t)}/d^{(s,t)}$. Then, TE's job in this example network at time t is to find a flow fraction vector

$$(f_{\langle 1,2\rangle}^{(1,4)}, ..., f_{\langle 8,9\rangle}^{(1,4)}, f_{\langle 1,2\rangle}^{(4,9)}, ..., f_{\langle 8,9\rangle}^{(4,9)})$$

which is the collection of fractions of each demand pair on all physical links, or a link flow vector

$$(v_{\langle 1,2\rangle}^{(1,4)}, ..., v_{\langle 8,9\rangle}^{(1,4)}, v_{\langle 1,2\rangle}^{(4,9)}, ..., v_{\langle 8,9\rangle}^{(4,9)})$$

After introducing some notations and concepts through this example network, we will introduce cost functions of these two optimizers and their optimization objectives. Let a or $\langle i, j\rangle$ denote a physical link. Then, in general, traffic flow or load on a physical link a will be

$$l_a = \sum_{(s,t)} \{f_a^{(s,t)} \cdot (d^{(s,t)\text{overlay}} + d^{(s,t)\text{underlying}})\}$$

We assume that each link can be modeled as a $M/M/1$ queue with mean delay $\frac{1}{C_a - l_a}$. This is the cost seen by a single packet. We assume both overlay and TE use this cost function, then, the total cost seen by TE on link a is $\frac{l_a}{C_a - l_a}$, but the total cost seen by overlay on link a is $\frac{\sum_{(s,t)} f_a^{(s,t)} \cdot d^{(s,t)\text{overlay}}}{C_a - l_a}$.

Then, TE wants to minimize the overall cost

$$\sum_a \frac{l_a}{C_a - l_a} \tag{1}$$

$d^{(s',t')}$	:	Overlay demand on node pair (s,t).
$d^{(s,t)}$	:	Demand pair of TE at physical level.
$d^{(s,t)\text{overlay}}$	:	TE demand due to overlay flow on logical link (s,t), which are routing decisions of overlay.
$d^{(s,t)\text{underlying}}$	:	TE demand due to underlying traffic.
$d^{(\cdot,t)}$	:	Demand of TE to destination t.
$h_p^{(s',t')}$	:	Overlay flow on logical path p for $d^{(s',t')}$.
$P^{(s',t')}$	:	Set of logical paths of overlay demand $d^{(s'.t')}$.
$\delta^{(s,t)}_{(s',t',p)}$	:	Mapping coefficient, indicating logical path p of $d^{(s',t')}$ on a logical link (s,t).
C_a(or $C_{\langle i,j\rangle}$)	:	Capacity of a physical link a (or $\langle i,j\rangle$).
l_a(or $l_{\langle i,j\rangle}$)	:	Link traffic at a physical link a.
$f_a^{(s,t)} or f^{(s,t)}_{\langle i,j\rangle}$	:	Fraction of TE demand $d^{(s,t)}$ on link a (or $\langle i,j\rangle$).
$v_a^{(s,t)}$	:	Flow of $d^{(s,t)}$ on link a.
v_a^t	:	Flow destined to t on link a.
Φ_a	:	TE cost on a physical link a.
Φ_a^{overlay}	:	Overlay cost on a physical link a.
J^{overlay}	:	Total cost of an overlay network.
J^{TE}	:	Total cost of TE.
N	:	Number of physical network nodes.

TABLE I

NOTATIONS IN FORMULATIONS OF TE AND OVERLAY OPTIMIZERS.

The overall cost that an overlay wants to minimize is

$$\sum_a \frac{\sum_{(s,t)} f_a^{(s,t)} \cdot d^{(s,t)\text{overlay}}}{C_a - l_a} \tag{2}$$

Formally, we have the following non-linear programming formulation of an overlay optimizer. Note, we use a piece-wise linear version in experiments.

$$\min_{h_p^{(s',t')}} J^{\text{overlay}} = \sum_a \frac{\sum\limits_{(s,t)} f_a^{(s,t)} d^{(s,t)\text{overlay}}}{C_a - \sum\limits_{(s,t)} \{f_a^{(s,t)}(d^{(s,t)\text{overlay}} + d^{(s,t)\text{underlying}})\}} \tag{3}$$

where,

$$d^{(s,t)\text{overlay}} = \sum_{(s',t',p)} \delta^{(s,t)}_{(s',t',p)} h_p^{(s',t')} \tag{4}$$

subject to

$$\sum_{p \in P^{(s',t')}} h_p^{(s',t')} = d^{(s',t')}, \forall (s',t') \in N \times N \tag{5}$$

$$h_p^{(s',t')} \geq 0, \qquad \forall (s',t') \in N \times N \tag{6}$$

An overlay optimizer takes five inputs

$$\{d^{(s',t')}, \delta^{(s,t)}_{(s',t',p)}, C_a, f_a^{(s,t)}, d^{(s,t)\text{underlying}}\} \tag{7}$$

and computes out the set of flows on logical paths $h_p^{(s',t')}$ (or flows on logical links $d^{(s,t)\text{overlay}}$). Note, among five inputs, only $d^{(s',t')}, \delta^{(s,t)}_{(s',t',p)}$ are known by overlay. The other three inputs can only be estimated. Thus, it might be difficult for an overlay optimizer to compute out a set of optimal logical level routings. We will come back to this issue in the next section.

Similar to the formulation adopted by [18], we use a piece-wise linear version of (1) for TE optimizer. In order to scale down the linear programming problem, we only solve for v_a^t, the traffic allocation on each link for traffic destined to a particular destination t, instead of computing traffic allocation on each link for a particular source and destination pair (s,t). After we solve this LP problem, we can recover $v_a^{(s,t)}$ from v_a^t based on the assumption of proportional allocation of traffic. We use d^t to represent the total demand destined to destination t: $d^t = \sum\limits_s d^{(s,t)}$. We give the TE optimization formulation as follows,

$$\textbf{min} \quad J^{\text{TE}} = \sum_{a \in A} \Phi_a \tag{8}$$

subject to

$$G_{\text{TE}}(C_a, l_a, \Phi_a, v_a^t, d^{(s,t)\text{overlay}}, d^{(s,t)\text{underlying}}) \tag{9}$$

where, G_{TE} [2] is the constraint of Traffic Engineering optimizer; Φ_a is the TE cost on physical link a.

The solution given by the TE optimizer is $\{J^{\text{TE}*}, {v_a^t}^*, {\Phi_a}^*\}$ From this solution, we can recover $v_a^{(s,t)}$, l_a (flow/load on a physical link). and $f_a^{(s,t)}$ adopted by TE.

B. Non-cooperative Non-zero Sum Two-player Game

Based on the formulations of TE and overlay optimizers, it is clear now that the strategy used by overlay is a vector of logical link flows. Then, a strategy of overlay is one feasible flow configuration on the logical links for all overlay demand pairs:

$$\mathbf{d}^{(\mathbf{s},\mathbf{t})\text{overlay}} = (..., d^{(i,j)\text{overlay}}, ...) \tag{10}$$

[2] To save space, we use this notation to summarize the constraints of TE optimzier.

Recall that we use $d^{(s',t')}$ to denote overlay demand, but we use $d^{(s,t)\text{overlay}}$ to denote overlay routings which are interpreted as the demand by TE. As a comparison, the demand seen by TE that comes from underlying traffic is denoted as $d^{(s,t)\text{under}}$. The strategy space Γ^{overlay} of an overlay network is the set of all feasible flow configurations on logical links or paths.

A strategy of TE is one feasible flow configuration on the physical links for all TE demand pairs:

$$\mathbf{f}^{\text{TE}} = (..., f^{(s,t)}_{\langle i,j\rangle}, ...) \tag{11}$$

The strategy space Γ^{TE} of TE is the set of all feasible flow configurations on physical links.

A strategy profile is $\bar{\gamma} = (\mathbf{f}^{\text{TE}}, \mathbf{d}^{(\mathbf{s},\mathbf{t})\text{overlay}})$. The cost function of TE is $J^{\text{TE}}(\mathbf{f}^{\text{TE}}, \mathbf{d}^{(\mathbf{s},\mathbf{t})\text{overlay}})$ and the cost function of overlay is $J^{\text{overlay}}(\mathbf{f}^{\text{TE}}, \mathbf{d}^{(\mathbf{s},\mathbf{t})\text{overlay}})$.

We have the following definition of *Nash equilibrium* for this routing game.

Definition 1: Nash Equilibrium A strategy profile $\bar{\gamma}^*$ is a Nash equilibrium if, for both players, TE and overlay,

$$J^{\text{TE}}(\mathbf{f}^{\text{TE}^*}, \mathbf{d}^{(\mathbf{s},\mathbf{t})\text{overlay}^*}) \leq J^{\text{TE}}(\mathbf{f}^{\text{TE}}, \mathbf{d}^{(\mathbf{s},\mathbf{t})\text{overlay}^*}) \quad \forall \mathbf{f}^{\text{TE}} \in \Gamma^{\text{TE}} \tag{12}$$

$$J^{\text{overlay}}(\mathbf{f}^{\text{TE}^*}, \mathbf{d}^{(\mathbf{s},\mathbf{t})\text{overlay}^*}) \leq J^{\text{overlay}}(\mathbf{f}^{\text{TE}^*}, \mathbf{d}^{(\mathbf{s},\mathbf{t})\text{overlay}}) \quad \forall \mathbf{d}^{(\mathbf{s},\mathbf{t})\text{overlay}} \in \Gamma^{\text{overlay}} \tag{13}$$

For a TE optimizer, overlay's response is observed as part of the demand matrix. Since TE knows the physical network's topology and all link capacities, and if we assume TE can estimate its demand matrix accurately ([5]), then, TE can compute out its optimal strategy. As for the implementation, if TE uses MPLS, it can exactly realize its strategy; if TE uses OSPF optimizer, it can only approximately realize its optimal strategy. However, an overlay optimizer may *not* be able to compute out its optimal strategy because it might not know the necessary information mentioned in last section. In the following sections, we first assume overlay knows the necessary information to compute out its best response, and model this interaction as a *Nash routing game* [11] in the next section. Later on, we will address the situation in which overlay only has limited information.

Another situation of interest to us is when one player can predict the other player's response (equivalent to knowing the other player's optimization algorithm.) In this case, the player who has this information and can move faster may choose to play a Stackelberg game ([34], [11]) against the other player (follower.) For example, if an overlay optimizer knows the optimization algorithm used by TE optimizer, then it can predict TE's new physical routings in response to overlay's logical level routing decisions, and then choose an optimal set of logical level routings in consideration of TE's potential responses. We model this interaction as a *static Stackelberg routing game* [34].

Our Nash routing game model is a *discrete* time model. One basic assumption is that, during its turn, one player completes its optimization before the other player starts. It could be true of course that a player starts its turn even when the other player has not yet finished. For example, OSPF TE will take some time to converge to a steady state. An overlay network may start its flow re-allocation during this OSPF's transient phase. We are not concerned with such an interaction process in this paper. A similar process is studied in [10], [24], [15], and [19].

Our static Stackelberg routing game model is *static* in a sense that, the leader (overlay network) can compute off-line the best strategy (Stackelberg equilibrium strategy for the leader), and sets it strategy before TE responds. Once TE responds the game is done. The definition of leader's *Stackelberg equilibrium strategy* will be given in Section V.

In the following two sections, we study Nash routing game and static Stackelberg routing game. Then, in Section VI, we study games in which overlay only has limited information.

IV. Nash Routing Game

We first give an illustrative example to show the structure of Nash equilibrium (NEP). For different initial conditions, we give analytical proof on the existence of NEP. Then, for one particular scenario, we prove the global stability of NEP for the best-reply dynamics of the game-playing process. [3] For general networks and general traffic demand patterns, the existence and stability of Nash equilibrium are much harder problems. And, even if there exists Nash equilibrium, the interaction process may not converge to that equilibrium, as shown in our experiments. Then, the more important question we study is how the selfish behavior of an overlay routing network influences the performance of TE in this interaction game. We prove that TE's performance will never be improved in this Nash routing game [4], and this is demonstrated through experiments on a 9-node network given in [26], and a 14-node tier-1 POP network in [5].

[3] Since our focus is on analyzing the relationships between routings and delays or costs, and in order to demonstrate the problem more clearly, we do not consider propagation delays in these examples. It is easy to extend these analysis to cases including propagation delays.

[4] We refer to the best-reply dynamics of the game when both players are of equal status as Nash routing game.

A. An Illustrative Example

We use the topology in Figure 3 for this example. We assume the bandwidth on two physical links between node 2 and 3 is large enough such that the delay on both links is negligible. Without loss of generality, we assume link $\langle 1,2\rangle$ has higher capacity than link $\langle 1,3\rangle$. Note, TE has the physical view of the network. But, overlay has the logical view of the network, so, logical link $(1,2)$ is actually mapped into two physical paths $1 \rightarrow 2$ and $1 \rightarrow 3 \rightarrow 2$. In the following, we show the actual interaction process and obtain the NEP for this example.

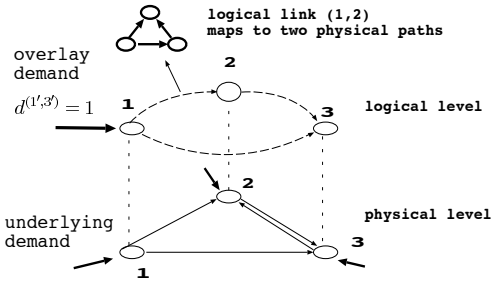

Fig. 3. Topology of a three-node network in example 2.

We assume overly only has demand between node 1 and 3. Then TE optimization can be written down as

$$\min_{\{l_{\langle 1,2\rangle}, l_{\langle 1,3\rangle}\}} J^{TE} = \frac{l_{\langle 1,2\rangle}}{C_{\langle 1,2\rangle} - l_{\langle 1,2\rangle}} + \frac{l_{\langle 1,3\rangle}}{C_{\langle 1,3\rangle} - l_{\langle 1,3\rangle}}$$

subject to

$$l_{\langle 1,2\rangle} + l_{\langle 1,3\rangle} = d^{(1,2)under} + d^{(1,3)under} + d^{(1,3)overlay} \tag{14}$$

The necessary condition for TE optimum is

$$\frac{\partial}{\partial l_{\langle 1,2\rangle}} J^{TE} = \frac{\partial}{\partial l_{\langle 1,3\rangle}} J^{TE} \tag{15}$$

$$\Rightarrow \frac{C_{\langle 1,2\rangle}}{(C_{\langle 1,2\rangle} - l_{\langle 1,2\rangle})^2} = \frac{C_{\langle 1,3\rangle}}{(C_{\langle 1,3\rangle} - l_{\langle 1,3\rangle})^2} \tag{16}$$

Based on (14) and (16), TE can calculate its optimal traffic assignment $\{l^*_{\langle 1,2\rangle}, l^*_{\langle 1,3\rangle}\}$. Obviously, the optimal routing fraction $\{f^{(1,2)}_{\langle 1,2\rangle}, f^{(1,3)}_{\langle 1,2\rangle}\}$ is not unique. To avoid ambiguity, we force TE to route traffic directly as much as possible. This is consistent with real case when the bandwidth on links between node 2 and 3 is finite. So we have:

$$f^{(1,2)}_{\langle 1,2\rangle} = \begin{cases} 1 & d^{(1,2)} \le l^*_{\langle 1,2\rangle} \\ \frac{l^*_{\langle 1,2\rangle}}{d^{(1,2)}} & d^{(1,2)} > l^*_{\langle 1,2\rangle} \end{cases} \tag{17}$$

and

$$f^{(1,3)}_{\langle 1,2\rangle} = max\{0, (l^*_{\langle 1,2\rangle} - f^{(1,2)}_{\langle 1,2\rangle} \times d^{(1,2)})/d^{(1,3)}\}. \tag{18}$$

Overlay optimization can be formulated as:

$$\min_{\{h^{(1,2)}, h^{(1,3)}\}} J^{overlay} = \frac{f^{(1,2)}_{\langle 1,2\rangle} h^{(1,2)} + f^{(1,3)}_{\langle 1,2\rangle} h^{(1,3)}}{\tilde{C}_{\langle 1,2\rangle} - f^{(1,2)}_{\langle 1,2\rangle} h^{(1,2)} - f^{(1,3)}_{\langle 1,2\rangle} h^{(1,3)}} + \frac{f^{(1,2)}_{\langle 1,3\rangle} h^{(1,2)} + f^{(1,3)}_{\langle 1,3\rangle} h^{(1,3)}}{\tilde{C}_{\langle 1,3\rangle} - f^{(1,2)}_{\langle 1,3\rangle} h^{(1,2)} - f^{(1,3)}_{\langle 1,3\rangle} h^{(1,3)}} \tag{19}$$

subject to $h^{(1,2)} + h^{(1,3)} = d^{(1,3)overlay}$, where $\{\tilde{C}_{\langle 1,2\rangle}, \tilde{C}_{\langle 1,3\rangle}\}$ is available bandwidth for overlay on link $\langle 1,2\rangle$ and $\langle 1,3\rangle$:

$$\tilde{C}_{\langle 1,2\rangle} = C_{\langle 1,2\rangle} - f^{(1,2)}_{\langle 1,2\rangle} d^{(1,2)under} - f^{(1,3)}_{\langle 1,2\rangle} d^{(1,3)under} \tag{20}$$

$$\tilde{C}_{\langle 1,3\rangle} = C_{\langle 1,3\rangle} - f^{(1,2)}_{\langle 1,3\rangle} d^{(1,2)under} - f^{(1,3)}_{\langle 1,3\rangle} d^{(1,3)under} \tag{21}$$

The necessary condition for overlay optimum with $h^{(1,2)} > 0$ and $h^{(1,3)} > 0$ is :

$$\frac{\partial}{\partial h^{(1,2)}} J^{overlay} = \frac{\partial}{\partial h^{(1,3)}} J^{overlay} \tag{22}$$

which is equivalent to

$$\frac{\tilde{C}_{\langle 1,2\rangle}}{(\tilde{C}_{\langle 1,2\rangle} - f^{(1,2)}_{\langle 1,2\rangle} h^{(1,2)} - f^{(1,3)}_{\langle 1,2\rangle} h^{(1,3)})^2} = \frac{\tilde{C}_{\langle 1,3\rangle}}{(\tilde{C}_{\langle 1,3\rangle} - f^{(1,2)}_{\langle 1,3\rangle} h^{(1,2)} - f^{(1,3)}_{\langle 1,3\rangle} h^{(1,3)})^2} \tag{23}$$

together with (20), (21), the necessary condition in terms of link rate is:

$$\frac{\tilde{C}_{\langle 1,2\rangle}}{(C_{\langle 1,2\rangle} - l_{\langle 1,2\rangle})^2} = \frac{\tilde{C}_{\langle 1,3\rangle}}{(C_{\langle 1,3\rangle} - l_{\langle 1,3\rangle})^2} \tag{24}$$

Comparing (24) with (16), we can see that for any NEP with $h^{(1,2)} \cdot h^{(1,3)} > 0$ we must have $\frac{\tilde{C}_{\langle 1,2\rangle}}{C_{\langle 1,2\rangle}} = \frac{\tilde{C}_{\langle 1,3\rangle}}{C_{\langle 1,3\rangle}}$. It can be satisfied for the trivial case when there is no underlying traffic. When underlying traffic is only between node 1 and 2, we have shown $\frac{\tilde{C}_{\langle 1,2\rangle}}{C_{\langle 1,2\rangle}} \neq \frac{\tilde{C}_{\langle 1,3\rangle}}{C_{\langle 1,3\rangle}}$. The possible NEPs are on the boundary, i.e., $h^{(1,2)} = 0$ or $h^{(1,3)} = 0$. Since the delay on logical path $(1,2)$ is always smaller than logical path $(1,3)$, $\{h^{(1,2)} = d^{(1,3)overlay}, h^{(1,3)} = 0\}$ is the only NEP. On the other hand, if $d^{(1,2)under} = 0$ and $d^{(1,3)overlay} < l^*_{\langle 1,2\rangle}$, there exists only one NEP with $h^{(1,2)} \cdot h^{(1,3)} > 0$.

Characteristics of NEPs. We are interested in the characteristics of those NEPs, namely, stability and effeciency

(for either overlay or TE), since these are of practical importance. To proceed, keep in mind that in this example, TE's cost will remain the same in this best-reply dynamics. It is easy to show that those NEPs happening on the boundary are stable and give overlay lower cost compared with initial cost.

One interesting NEP is for the scenario where $d^{(1,2)under} = 0$ and $d^{(1,3)overlay} < l^*_{\langle 1,2\rangle}$. This NEP is unique and globally stable, i.e. the overlay routing will always converges to the NEP regardless of overlay's initial routing. See Appendix A for the proof. One interesting observation is that this NEP is *inefficient* for overlay for some inital condition, namely, overlay's cost at NEP is higher that its initial cost at the beginning of the interaction process.

To illustrate, we present results from one experiment. We set $C_{\langle 1,2\rangle} = 1$, $C_{\langle 1,3\rangle} = 0.5$, $d^{(1,3)overlay} = d^{(1,3)under} = 0.5$. Overlay takes turn at even round, TE takes turn at odd round. We use Matlab optimization toolbox to solve TE and overlay's optimization (14) and (19). We did two experiments with different initial overlay routing: $h^{(1,2)}(0) = d^{(1,3)overlay}$; $h^{(1,2)}(0) = 0$. Figure 4 shows that in both cases overlay routing converges to the NEP. Figure 5 shows the trend of overlay cost as the Nash game proceeds.

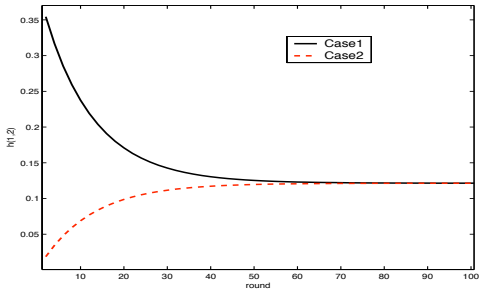

Fig. 4. Convergence of Overlay Routing

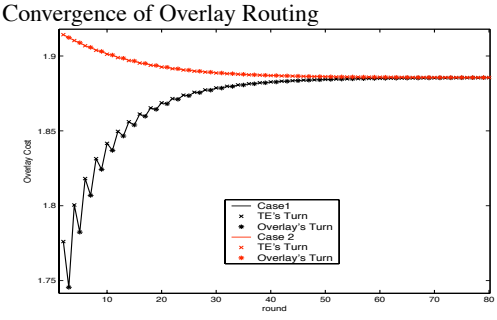

Fig. 5. Overlay Cost Trend

It is interesting to observe that for the case with $h^{(1,2)}(0) = d^{(1,3)overlay}$ overlay cost actually increases over rounds even though overlay tries to minimize its cost at each round. This is because after overlay's minimization, TE will adjust its routing to minimize the whole network cost. The updated TE routing will increase overlay's cost. The interaction between TE and overlay routing is bad for overlay so that the overall trend is increasing until the game converges to its NEP. We observe the same phenomenon in experiments on a 14-node tier-1 ISP network later. For this example, given TE's optimization algorithm, the best strategy for overlay is to place all its traffic on logical path $(1, 2)$. This is to say it may *not* be wise for overlay to play Nash game with TE. This is consistent to the inefficiency property of NEP. We will discuss this more in section on Stackelberg routing game.

Comments on Nash equilibrium (NEP). Even though there exist Nash equilibrium for the above simple routing interaction game, we need to point out that, for general networks, NEPs are dependent on the initial conditions, topology, and traffic demand patterns. It is a much harder problem to give general solutions. See [25] for studies on Nash equilibrium for network routing game where players are all at the same physical network level. Even if there exists equilibrium, the interaction process may converge or may not converge to an equilibrium. Convergence property also depends on the initial conditions, topology, and traffic demand patterns. We will observe this in the following experiments on larger networks. Thus, practically speaking, the more important question to ask is *how does this interaction process affect the performance of TE?* We address this question in the next section.

B. Effects on the cost of Traffic Engineering

While overlay routing aims at improving the performance of overlay traffic, the improvement comes at the price of degrading the performance of underlying traffic. In addition, if we assume TE can perfectly implement the optimal solution, overlay routing won't be able to improve the overall network performance. In many cases, overlay routing will increase the network cost that TE tries to minimize.

Base cost of TE. Base cost of TE refers to the optimal cost achieved when overlay simply give its demand matrix to TE without making routing decisions on logical level, that is, without traffic forwarding among themselves. That is, simply let $d^{(s,t)\text{overlay}} = d^{(s',t')}$.

Lemma 1: Overlay routing will never improve TE's performance (i.e., never reduce TE's base cost).

Proof: We compare the network cost with and without overlay routing. Let P_O be the set of source-destination pairs of overlay demand and $\{D_O^{(s,t)}, (s,t) \in P_O\}$ be the overlay demand vector. Without overlay routing, TE will take overlay demand and underlying demand

directly as its overall physical demand:

$$d^{(s,t)} = \begin{cases} D_O^{(s,t)} + d^{(s,t)\text{under}} & (s,t) \in P_O \\ d^{(s,t)\text{under}} & (s,t) \notin P_O \end{cases} \quad (25)$$

Then TE's optimal routing $\{\bar{v}_a^t\}$ is the minimum over all feasible routing under constraint (9), i.e., all $\{v_a^t\}$ which satisfy flow conservation and implement all TE demand (25).

With overlay routing, overlay can assign traffic between any overlay demand pair $\{(s',t') \in P_O\}$ on all associated logical path $\{p \in P^{(s',t')}\}$. Let L_O be the set of all node pairs employed by overlay as logical links. Then the traffic demand seen by TE can be calculated as:

$$d^{(s,t)} = \begin{cases} d^{(s,t)\text{under}} + \sum\limits_{(s',t',p)} \delta_{(s',t',p)}^{(s,t)} h_p^{(s',t')} & (s,t) \in L_O \\ d^{(s,t)\text{under}} & (s,t) \notin L_O \end{cases} \quad (26)$$

Any TE routing $\{\tilde{v}_a^t\}$ (and consequently $\{\tilde{f}_a^{(s,t)}\}$) resulted from any overlay routing $\{h_p^{(s',t')}, (s',t') \in P_O, p \in P^{(s',t')}\}$ must implement TE demand as described in (26). At the same time, we can calculate the amount of overlay traffic destined to each overlay node t' on each physical link as:

$$\hat{v}_a^{des(t')} = \sum_{(s' \in N)} \sum_{p \in P^{(s',t')}} h_p^{(s',t')} \cdot \Big(\sum_{(s,t) \in N \times N} \delta_{(s',t',p)}^{(s,t)} \tilde{f}_a^{(s,t)} \Big) \quad (27)$$

Based on (27), we can construct a TE destination based routing

$$\hat{v}_a^t = \begin{cases} \sum\limits_s v_a^{(s,t)\text{under}} + v_a^{des(t)} & \text{if } t \text{ is an overlay node} \\ \sum\limits_s v_a^{(s,t)\text{under}} & \text{otherwise} \end{cases} \quad (28)$$

It is easy to check that $\{\hat{v}_a^t\}$ implement TE demand without overlay routing as described in (25). Therefore, $J^{TE}(\{\hat{v}_a^t\}) \geq J^{TE}(\{\bar{v}_a^t\})$. At the same time, the aggregate traffic rate vector on all physical links $\{\hat{l}_a\}$ under $\{\hat{v}_a^t\}$ is the same as the link rate vector $\{\tilde{l}_a\}$ under $\{\tilde{v}_a^t\}$ with overlay routing. Since link cost is only a function of its aggregate rate, we have $J^{TE}(\{\tilde{v}_a^t\}) = J^{TE}(\{\hat{v}_a^t\}) \geq J^{TE}(\{\bar{v}_a^t\})$.

■

Experimental Results. Consider a 9-node example ([26]) in Figure 2. Three nodes are overlay nodes: $1, 4, 7$. They may have demands to each other, or have demands to other nodes not in the overlay. In any case, each overlay node can forward traffic originated from any other overlay node. There are 24 possible overlay demand pairs in this example. We randomly choose 70% of them. We use a bimodal traffic matrix ([5]) generated by a mixture of two Gaussians, one with $(\mu_1 = 1.5, \sigma_1 = 0.2)$, and the second with $(\mu_2 = 4, \sigma_2 = 0.2)$. These means and standard deviations are proportional to those used in ([5]). We set the overlay demands to be 60% of total traffic demands. To avoid flows exceed the capacities of links we set capacities of all links to be 18. We also perform experiments when link capacities are randomly distributed in a certain range. For brevity, we do not present results here. Heterogeneous link capacity cases can be seen in experiments in a 14-node tier-1 ISP network.

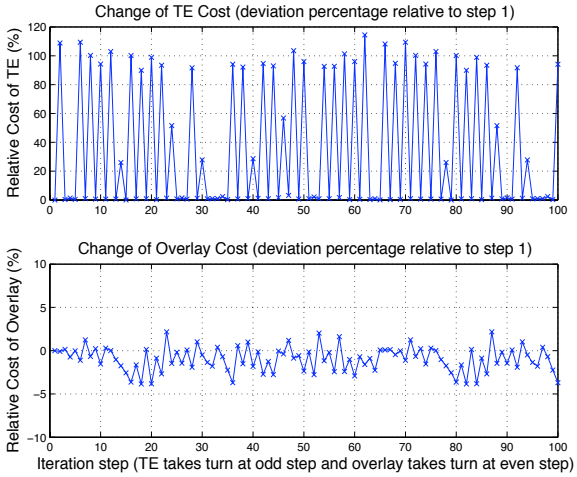

Fig. 6. Cost change of TE and overlay. Percentage of deviation from cost at step 1 at each step in the interaction process.

Initially, overlay demand pairs are given to TE without any forwarding among overlay nodes, achieving the *base cost* of TE. We let TE begin the interaction game. TE takes turn at odd step, and overlay takes turn at even step. We let this interaction process run 100 steps. Taking the cost at step 1 as the base point, we calculate the percentage of deviation from it at the following steps. These percentage of deviations of TE and overlay are plotted in Figure 6.

We observe from these plots that there are large oscillations in both player's costs in the observed time interval (this playing process does not converge.) At each even step, overlay's response causes an increase to TE's cost, and then TE will react optimally to reduce its cost at the following odd step. Overlay does the similar thing. On average, overlay's cost decreases 1% in this interaction process, but TE's cost increases 35.9% as expected.

TE cost change as a function of percentage of overlay traffic.

We are interested in how the cost change of TE varies as the percentage of overlay traffic varies. Our conjecture is as follows. If there is little overlay traffic, then overlay's routing decisions will have little influence on TE's cost. If all traffic consists of overlay traffic, then overlay's routing

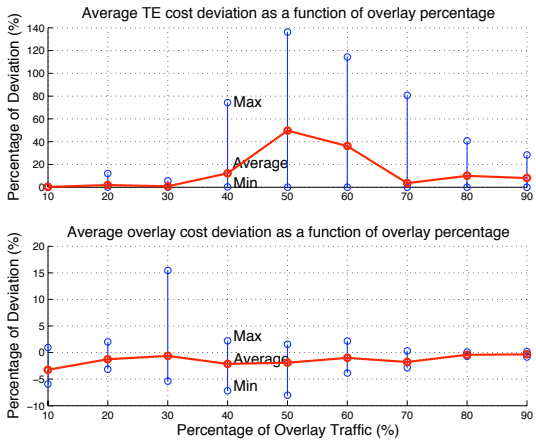

Fig. 7. Cost change of TE and overlay. Percentage of deviation from cost at step 1. Nine-node network. BWs of all links are 18.

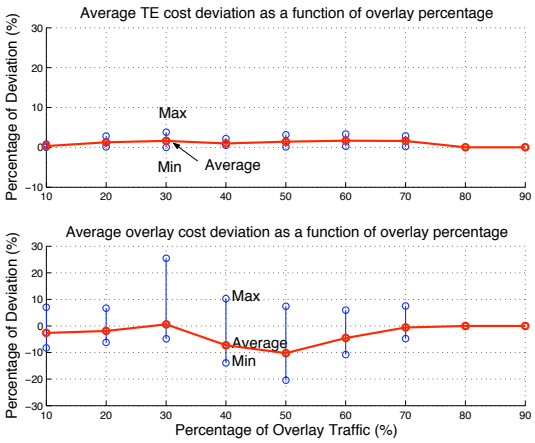

Fig. 8. Cost change of TE and overlay. Percentage of deviation from cost at step 1. Nine-node network. BWs of all links are 20.

decision would be the same as that of TE, so, the interaction process will always converge, and TE's cost will not be affected. If there is some significant fraction of overlay traffic, e.g. 50%, TE's cost increase will be maximal.

In Figure 7 , we plot the cost deviation percentage for different overlay traffic percentages. Our conjecture is verified through these experiments. In addition, we notice that, when overlay demand is approximately half of total network demand, not only is the average cost increase to TE the largest, but also the variation range is the largest. Larger variation of TE cost means greater oscillation in the interaction process, which is clearly harmful to TE.

Another interesting observation is that, if we increase link capacities, the decrease of overlay cost by playing Nash game is not as large as that in smaller link capacity situation. This can be seen by comparing Figure 7 with Figure 8. And, TE's cost is not affected much by overlay's selfish behavior if link capacities increase. Intuitively speaking, this is because TE optimizer has more freedom to allocate traffic to achieve the same minimal cost when link capacity is large.

C. OSPF optimizer

If TE use an OSPF optimizer instead of MPLS, overlay routing may actually help to reduce TE's cost. We have demonstrated this using a simple experiment on the topology in Figure 3. Recall that an OSPF optimizer will find the best possible link weight settings to approximate the solution of MPLS optimizer ([18]). In this simple example, we are able to do an exhaustive search for the best weight settings. Due to space limitations, we do not present results here. As for general network topology and more realistic traffic demand patterns, we might expect TE's cost could also be decreased by interacting with an overlay optimizer, because the OSPF optimizer might not be able to do as well as the MPLS optimizer. In this context, the interaction dynamics actually depends on the specific implementation of OSPF optimizer. We will address this topic in our future work.

D. Experiments with a tier-1 ISP network

We do experiments extensively on a 14-node tier-1 POP network in [5]. We invert the weights of links to get link capacities. This is based on the assumption that weights are set by turning around capacities as recommended by Cisco. Depending on the traffic matrix used, we multiply these capacities by a certain value to make sure that, for the traffic matrix we use, no link capacity is exceeded by traffic on that link. Our experimental results confirm our hypotheses presented in previous sections. We present two experimental results here.

In these two experiments, for underlying traffic, we use a bimodal traffic matrix which is the same as used in the nine-node experiments earlier.

We choose three nodes $6, 10, 11$ as overlay nodes, and randomly choose 32 overlay demand pairs among all possible 39 overlay demand pairs. On top of the underlying demands, we add $p\%$ overlay demands. Specifically, for a node pair that is picked as an overlay demand pair, if the underlying traffic demand is d, then we add $d \cdot p\%$ overlay traffic.

In one experiment, we choose $p\% = 50\%$. Thus, the total overlay traffic among all network traffic is 8.1%. We run this experiment 100 steps. We observe oscillations in these 100 steps. The mean cost increase for TE is 3.5%, and the mean cost decrease for overlay is 52.1%. Since the percentage of overlay traffic is small, the cost increase to TE is not big, but still, this small percentage of overlay causes significant oscillations to TE's cost. The highest increase to TE's cost can reach 12%.

In another experiment, we choose $p\% = 68\%$. Thus, the total overlay traffic among all network traffic is 10.8%.

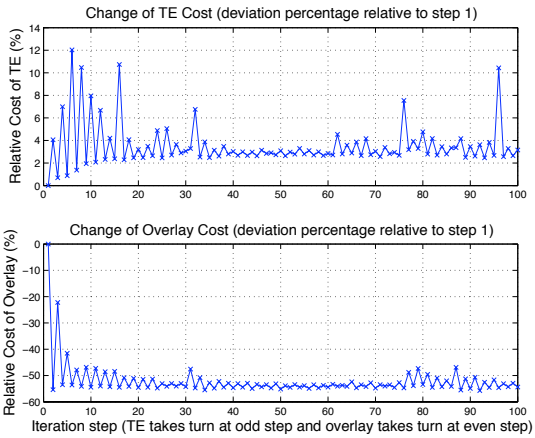

Fig. 9. Cost change of TE and overlay. Percentage of deviation from cost at step 1. A tier-1 ISP network. Experiment setting 1.

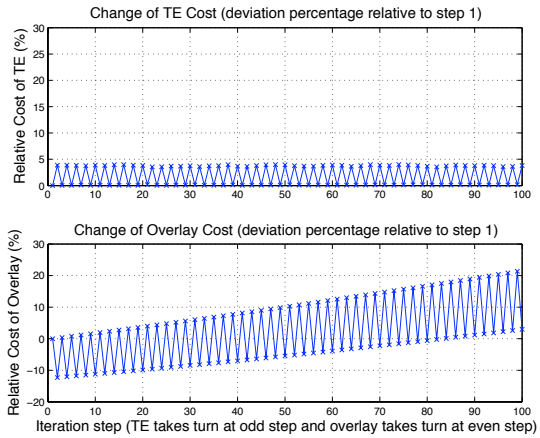

Fig. 10. Cost change of TE and overlay. Percentage of deviation from cost at step 1. A tier-1 ISP network. Experiment setting 2.

We run this experiment 100 steps. The mean cost increase for TE is 3%. We observe an increasing trend of overlay cost in these 100 steps. At the final step, even after overlay's optimization, the cost of overlay is 3% higher than the cost in the very first step when overlay does nothing in optimizing its routings on logical level. This experiment is consistent with our analysis for a simple network previously. This gives a warning to an overlay routing optimizer that playing a Nash game by optimizing routing at each step may *not* a good idea. [5]

V. Static Stackelberg Routing Game

In this model, overlay knows TE optimizer's algorithm besides the information on physical network topology, underlying traffic demand matrix. Then overlay can choose either to play a Nash game or to play a static Stackelberg game ([34], [11] and [33]) by acting as a leader. We will focus on overlay playing a Stackelberg game in this section.

As discussed in previous sections, what TE can control is physical level routing $f_a^{(s,t)}$. What overlay can control is the traffic on logical links, which are interpreted as part of traffic demand $d^{(s,t)}$ for TE. To simplify our notation, we use X to represent the latter, and Y to denote the former. As we know, each player reacts to other player's action (chosen strategy) rationally and optimally. Then, in this two-player routing game, we call a strategy X^* a static Stackelberg equilibrium strategy for overlay (the leader) if it can give the lowest cost($J^{\text{overlay}*}$) to overlay. Formally, the following condition needs to be satisfied: $J^{\text{overlay}*} = \inf_{X^*} J^{\text{overlay}}(X, Y(X))$ [6]

The leader (overlay), chooses X to minimize the cost function J^{overlay}, while the follower (TE), reacts to leader's decision by selecting a strategy $Y(X)$ that minimizes his cost function J^{TE}, in full knowledge of the leader's decision. Thus, the follower's decision depends on the leader's decision, and the leader is in full knowledge of this. Intuitively, we see that the leader's strategy should be, *choosing a particular X^*, such that the cost function $J^{overlay}$ of the leader is minimum when the follower chooses its own optimal solution Y as a function of X^**. We call such a strategy for the leader the *Stackelberg strategy for the leader* [34] [11].

Solving for the static Stackelberg equilibrium strategy for leader (overlay) is essentially solving the following optimization problem

$$\begin{aligned} &\mathbf{min}_X J^{\text{overlay}}(X, Y^*(X)) \\ &\text{s.t. } G_{overlay}(X, Y) \\ &\quad Y^* = \mathbf{argmin} J^{\text{TE}}(Y) \\ &\quad \text{s.t. } G_{TE}(X, Y) \end{aligned} \tag{29}$$

where $G_{overlay}(X, Y)$ and $G_{TE}(X, Y)$ are constraints for overlay and TE respectively. This optimization problem is classified as Bilevel Programming (BP) problem. See [33] for general discussions on BP problem.

According to this definition, the Stackelberg equilibrium prescribes an optimal strategy for the leader if the follower reacts by playing optimally, whenever the leader announces his moves first. Note, this is a *static* game, in which, overlay does an off-line computation to find the

[5] Due to limitations of computation power and LP software (lp_solve) we used, the computation of this network settings takes long long time, so we are not able to observe more interaction steps than 100. We will try to improve this in future. For the purpose to illustrate our points here, this number of steps is sufficient.

[6] We have used implicitly the assumption that *the follower's response to every strategy of the leader is unique*. Whenever this assumption is not satisfied, there is ambiguity in the possible responses of the follower and consequently in the attainable loss levels of the leader. We will address this problem in future work.

best strategy, and set this strategy only once. It can be proved easily that overlay cost with Stackelberg solution is no worse than overlay cost at any NEP.

Solving (29) is a NP-hard problem [33]. In the next section, we propose a gradient projection search heuristic.

A. Computation of static Stackelberg strategy for Overlay

Motivated by descent method and projection methods for optimal routing, we propose a Gradient Projection Search (GPS) heuristic for solving the static Stackelberg equilibrium strategy in the routing interaction game. Decent direction method to nonlinear bi-level programming appeared in [31] and [21]. Projection methods for optimal routing can be found in [14] and [13].

In solving this static Stackelberg routing game, we have to choose a set of logical path flows $h_{p_i}^{(s',t')}$ for each overlay demand pair. These logical path flows determine the set of overlay logical link loads $d^{(s,t)\text{overlay}}$, which are parts of demand matrix for Traffic Engineering. Based on the traffic demand matrix, Traffic Engineering will solve a Linear Programming problem to get an allocation of traffic flows of each demand pair on each link $v_a^{(s,t)}$, from which, overlay can evaluate its overall delay or cost. The optimal set of logical path flows would give overlay the lowest overall delay. Thus, in solving for it iteratively, at the $(k+1)^{th}$ step, we hope to choose the new set of path flows that decrease the cost compared with the cost at k^{th} step. This means we need to choose the new set of flows along descent direction of overlay cost function, and they need be feasible also, i.e., they have to satisfy the constraints such as flow conservations, demand, etc.

We modify the feasible direction method given in [14]. Briefly speaking, the feasible direction method is to determine a minimum first derivative length (MFDL) path for every SD pair at each iteration. The first derivative length (FDL) of a logical path $h_p^{(s',t')}$ is the first derivative of objective function with respect to this path $\frac{\partial J^{\text{overlay}}}{\partial h_p^{(s',t')}}$ evaluated at the current flow allocation. An increment of flow change is calculated for each path on the basis of the relative magnitudes of FDL lengths with respect to MFDL. If the increment is so large that the path flow becomes negative, the path flow is simply set to zero. These routing algorithms can be viewed as constrained versions of common, unconstrained optimization methods.

In the original feasible direction method, gradient of cost function can be evaluated directly at each step. However, in our case, we cannot directly solve for the gradient by taking the derivatives of the cost function with respect to each path flow $h_p^{(s',t')}$. This is because, in the cost function, l_a is an implicit function of path flows $h_p^{(s',t')}$, which is defined by the underlying optimization of Traffic Engineering. In general, we cannot get obtain the closed form explicit functions from these implicit functions.[7] Fortunately, since TE optimization is a linear programming (LP) problem, we can get the derivative of l_a with respect to $h_p^{(s',t')}$ by looking at the inverse of the basis of TE's LP. Once we got the gradient of cost function with respect to path flows, we can directly use the descent projection method given by [14]. The algorithm for decent projection method is presented in Figure 11.

X^k: the logical path flow vector of overlay;
Y^k: the physical link flow vector of overlay traffic;
$\text{FDL}_{p_w}^k$: the first derivative length of logical path p_w;
$\bar{p}_w$: the path with minimum first derivative length among all paths of overlay demand pair w;
W: set of all overlay demand pairs;
P_w: set of paths for overlay demand pair w.

1. **repeat**
2. Compute $\bigtriangledown_{X^k} J^{k,\text{overlay}} = (\text{FDL}_{p_w}^k)$;
3. $\forall w \in W, \forall p \in P_w$;
4. let $\bar{p}_w$ be MFDL path;
5. $\forall p \neq \bar{p}_w, x_p^{k+1} = \max\{0, x_p^k - \alpha^k(\text{FDL}_{p_w}^k - \text{MFDL});$
5. $x_{\bar{p}_w}^{k+1} = d^w - \sum_{p \neq \bar{p}_w} x_p^{k+1};$
6. Solve TE LP problem to get $Y^{*(k+1)}$ given X^{k+1};
7. Compute $J^{k+1,\text{overlay}}(X^{k+1}, Y^{*(k+1)})$
8. **until** $|J^{k+1,\textbf{overlay}} - J^{k,\textbf{overlay}}| <$ **threshold**

Fig. 11. Gradient Projection Search (GPS) for static Stackelberg routing game.

Since Gradient Projection Search (GPS) is still a local search heuristic, it can only find a *local* optimal solution within the neighborhood of the starting point. Thus, in order to search for the global optimization point, we need to randomly start the gradient projection search many times.

B. Experiments

Consider the nine-node network (figure 2) again. We use the same bimodal traffic demand matrix, and the same overlay demands as those in Nash games in section 4.

We run many experiments using our search heuristic. Figure 12 shows one typical gradient projection search path. We see that initially cost of overlay decreases very fast, but decreases slowly afterwards, and hits a local minimum point finally. Since both TE and overlay optimizers use piece-wise linear cost functions, we might expect that there are many local minimums. Thus, we need to randomly start search many times. Our initial results show gradient projection search heuristic is promising. For example, in experiments on the nine-node network for five capacity settings ($C = 15, 16, 17, 18, 19$), we randomly

[7] [24] and [10] gave some discussions on these implicit functions in the context of solving Nash game in a distributed way.

start GPS 100 times and choose the lowest cost as the approximate Stackelberg cost. In 4 out of 5 settings, Stackelberg cost is lower than average Nash cost. In two settings, Stackelberg costs are significantly lower than Nash costs. 35% and 50% cost reduction are achieved. In our future work, we will improve this search strategy.

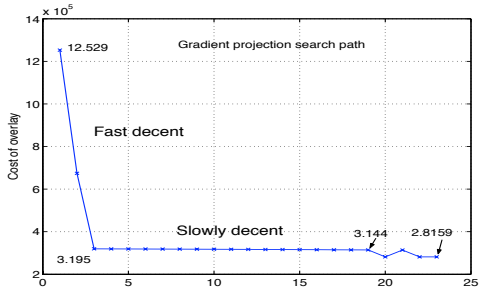

Fig. 12. A sample gradient projection search path.

VI. Nash Routing Games With Limited Information

A. Overlay Optimizers With Limited Information

We notice that, in order to compute out optimal strategy an overlay network needs to know physical level routing information and physical level underlying traffic demand and physical link capacities. We call an overlay optimizer without these pieces of information an overlay with *limited* information.

Overlay can only measure delays on logical links by sending out probing packets. Based on how frequently it takes measurements of the network and how to use these estimated delay information, we can have two types of overlay networks: *one-step* overlay with *limited* information; *incremental* overlay with *limited* information.

By *one-step* overlay with *limited* information, we mean that overlay takes one time measurement of delays on logical links, and assume these delays are fixed, and then computes out a new assignment of flows on logical paths. For example, an overlay demand pair puts all its traffic on the logical path with current shortest delay. Let $\text{Delay}_{(s,t)}$ denote the delay on logical link (s,t). The delay on a logical path p is

$$\text{Delay}_p^{(s',t')} = \sum_{(s,t)} \delta_{(s',t',p)}^{(s,t)} \text{Delay}_{(s,t)} \tag{30}$$

In turn, logical link delay is

$$\text{Delay}^{(s,t)} = \sum_a f_a^{(s,t)} \frac{l_a}{C_a - l_a} \tag{31}$$

Given that logical-link delay information can be estimated by overlay, overlay optimization problem can be formulated as

$$\min_{h_p^{(s',t')}} F_{\text{overlay}} = \sum_p \text{Delay}_p \cdot \Big(\sum_{(s',t')} h_p^{(s',t')}\Big) \tag{32}$$

subject to flow conservation and non-negative flow constraints.

By *incremental* overlay with *limited* information, we mean it can derive its new routing decisions in a sequence of stages in the step that it takes turn to do optimization. At each stage, it only puts a small amount of traffic on a logical path, and then evaluates its cost function by taking measurements again. This way, it can estimate which logical paths can lead to a lower cost. In the next stage, it puts another small amount of traffic on those paths leading to lower costs. Note, this amount of small traffic is overlay dependent, that is, any overlay routing structure itself specifies what amount of traffic to shift at each step. In our analysis in the next section, we assume that overlay structure shift an infinitesimal amount of traffic at each step. This process continues until all its traffic has been put on available logical paths. We call this process an *incremental* algorithm. Note, here, we can think of that the interval between two consecutive TE optimizations is long enough for these incremental addition of overlay flows to complete. If one thinks of the total flow as an aggregate of many infinitesimally small amount of selfish flow between a common source and a common destination, and each individual flow decides its own routing by choosing the shortest delay path, then this incremental algorithm essentially tries to solve for the Nash equilibrium point (NEP) in *selfish routing* at logical level ([28] [26] [17]). As pointed out in chapter 5 of [32], this incremental algorithm may not converge to NEP for some network settings even at physical level. Interpreting [32]'s results in our context, if given the physical routings set by TE fixed at one round or step, incremental algorithm may not converge to a NEP for all overlay traffic demands, let alone talking about the NEP between overlay and TE in the interaction process when TE changes round by round.

Comparing *one-step* and *incremental* overlay with limited information, clearly, there is a tradeoff between measurement of network and performance improvement. For *one-step* overlay with limited information, the measurement work required of an overlay is small. It can just periodically send out probing packets to get the logical delay information, and then allocate all its traffic on the smallest delay path. The consequence is that the new allocation might increase the delay dramatically. The performance loss might be huge and it might cause bad oscillations to network routings. We will show this through a simple example later in the paper. For *incremental* overlay with

limited information, it continuously probes the network by adding small amount of traffic and measuring the delays again, so, the measurement overhead is huge if this overlay network has lots of traffic.

In the following, we study these two types of games.

B. Incremental Overlay with Limited Information

Consider again the same network settings used in the illustrative example in the previous section on Nash routing game Figure 3.

This time, we look at the piece-wise linear version of overlay network cost function. As described in (9), the objective of TE is to minimize the summation of cost on physical link $< 1, 2 >$ and $< 1, 3 >$:

$$\textbf{min} \quad J^{\text{TE}} = \Phi(C_{\langle 1,2\rangle}, l_{\langle 1,2\rangle}) + \Phi(C_{\langle 1,3\rangle}, l_{\langle 1,3\rangle}) \quad (33)$$

subject to

$$l_{\langle 1,2\rangle} + l_{\langle 1,3\rangle} = d^{(1,2)} + d^{(1,3)}, \quad (34)$$

where the link cost Φ is a continuous and convex piecewise linear function of traffic rate l:

$$\Phi(C, l) = \begin{cases} k_1 l + b_1 & l \in [0, \alpha_1 \cdot C) \\ \cdots & \cdots \\ k_i l + b_i & l \in [\alpha_{i-1} \cdot C, \alpha_i \cdot C) \\ \cdots & \cdots \\ k_m l + b_m & l \in [\alpha_{m-1} \cdot C, C) \end{cases}$$

Then the necessary condition for the TE's optimal solution $\frac{d\Phi_{\langle 1,2\rangle}}{dl_{\langle 1,2\rangle}} = \frac{d\Phi_{\langle 1,3\rangle}}{dl_{\langle 1,3\rangle}}$ means $l_{\langle 1,2\rangle}/C_{\langle 1,2\rangle}$ falls into the same region $[\alpha_{j-1}, \alpha_j)$ as $l_{\langle 1,3\rangle}/C_{\langle 1,3\rangle}$. To illustrate, we plot in Figure 13 the traffic allocation $\{l_{\langle 1,2\rangle}, l_{\langle 1,3\rangle}\}$. TE's optimal must fall into one of the the shaded blocks along the diagonal. Therefore, TE's optimal solution set is $[T_1^*, T_2^*]$, the intersection between the constraint line CD and the shaded area. The non-uniqueness in TE's optimal solution is due to the piece-wise linear link cost function. Actually, the finer the piece-wise linear function, the smaller the shaded area. When the size of pieces go to zero, the shaded area degenerates to the diagonal line AB, then TE has a unique optimal solution T^*. TE's optimal routing is calculated as (17, 18). We always have $f_{\langle 1,2\rangle}^{(1,2)} > f_{\langle 1,2\rangle}^{(1,3)}$. Delay on logical link can be calculated as:

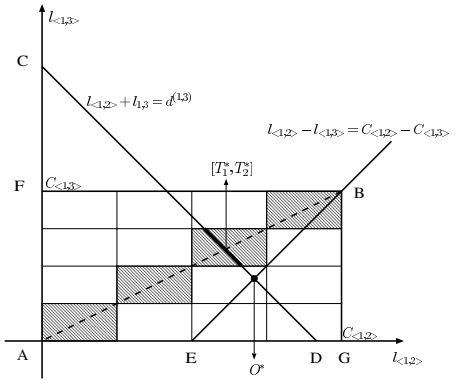

Fig. 13. Interaction between TE and Overlay

$$\text{Delay}_{(1,2)} = f_{\langle 1,2\rangle}^{(1,2)} \text{Delay}_{\langle 1,2\rangle} + (1 - f_{\langle 1,2\rangle}^{(1,2)}) \text{Delay}_{\langle 1,3\rangle} \quad (35)$$

$$\text{Delay}_{(1,3)} = f_{\langle 1,2\rangle}^{(1,3)} \text{Delay}_{\langle 1,2\rangle} + (1 - f_{\langle 1,2\rangle}^{(1,3)}) \text{Delay}_{\langle 1,3\rangle} \quad (36)$$

We assume overlay plays a selfish game, i.e., it tries to move its traffic to the logic path with minimal delay. Overlay essentially tries to solve the optimization problem equivalent to selfish routing (see Appendix.) In this example, overlay can exactly solve our optimal routings for itself at each round (equivalent to saying that NEP of overlay selfish routing is achieved.) We give a proof that NEP between overlay and TE can also be reached in the interaction process. This NEP is achieved, if for overlay, either when the delay on two logical paths are equal or overlay moves all its traffic to one logical path with minimal delay; if for TE, the physical flows $l_{\langle 1,2\rangle}$ and $l_{\langle 1,3\rangle}$ satisfy the necessary conditions.

Then, we can define a routing game as follows. Given assumptions on network settings defined at the beginning of this section, if there is only one overlay demand from 1 to 3, and both players take turns to do optimizations using algorithms defined above. Let TE starts at step 1 and overlay just gives its demand $d^{(1',3')}$ to TE without using node 2 as forwarding node, that is, at step 1, only one nonzero demand for TE: $d^{(1.3)} = d^{(1',3')}$. In the game playing process, overlay takes turn at even step and TE takes turn at odd step.

Lemma 2: The Nash routing game defined above always converges to a NEP.

Proof: Due to $f_{\langle 1,2\rangle}^{(1,2)} > f_{\langle 1,2\rangle}^{(1,3)}$, the only way to match the logical delay on two logical paths is to match the delay on physical link $\langle 1, 2\rangle$ and $\langle 1, 3\rangle$. We introduce in Figure 13 the Equal Delay Line EB, where the rate vector makes the physical delay on link $\langle 1, 2\rangle$ and $\langle 1, 3\rangle$ equal, or equivalently

$$C_{\langle 1,2\rangle} - l_{\langle 1,2\rangle} = C_{\langle 1,3\rangle} - l_{\langle 1,3\rangle}$$

Then any rate allocation in area $AEBF$ will make $\text{Delay}_{\langle 1,3\rangle} > \text{Delay}_{\langle 1,2\rangle}$ and $\text{Delay}_{\langle 1,3\rangle} < \text{Delay}_{\langle 1,2\rangle}$ in area EGB. Let point O^* be the intersection between line EB and the demand line CD. If O^* falls in TE's optimal solution interval $[T_1^*, T_2^*]$, then O^* achieves optimum for both TE and overlay. Therefore O^* is a NEP and it will be reached after one round of TE and overlay optimization.

If EB falls outside of $[T_1^*, T_2^*]$, then after TE's optimization, we have $\text{Delay}_{\langle 1,2\rangle} < \text{Delay}_{\langle 1,3\rangle}$ and $f_{\langle 1,2\rangle}^{(1,2)} >$

$f_{\langle 1,2\rangle}^{(1,3)}$. Therefore, from (35), we will have $\text{Delay}_{(1,2)} < \text{Delay}_{(1,3)}$. When overlay takes over, it always try to move some of its demand from logical path $(1,3)$ to logical path $(1,2)$ until either all its demand been moved to path $(1,2)$ or the rate vector reach point O^*. For the first case, when TE takes over, it will pull the rate vector back into $[T_0^*, T_1^*]$. And when overlay takes the turn, it still sees $\text{Delay}_{(1,2)} < \text{Delay}_{(1,3)}$. Since all its demand has already been put on path $(1,2)$, the game reaches its NEP. For the second case, TE again will pull the rate vector back into $[T_0^*, T_1^*]$. When overlay takes the turn, it will increase $d^{(1,2)}$ and the drive the rate vector back to O^*. The interaction continues and $d^{(1,2)}$ keeps increasing until all overlay traffic is moved to logical path $(1,2)$. The game converges as in the first case. ■

As we see from this proof, the driving force for this process to converge to a Nash equilibrium is that TE can always map a set of logical link load required by overlay to a physical flow assignment. The logical mapping by TE plays a key role. Since this logical mapping really depends on the network topology, it is difficult to give a general conclusion to a general network topology.

C. One-step Overlay with Limited Information

In this model, at each step, overlay can simply put all its traffic on the minimum delay path, totally ignorant of the potential delay increment on this minimum delay path. We give a simple example to show that this interaction can lead bad performance to both overlay and Traffic Engineering, and lead to bad oscillations. Consider a simple network topology in Figure 3. We let all links have capacity 10. There are underlying demands for source-destination pairs, which are $d^{(1,2)} = 8$, $d^{(1,3)} = 1$, $d^{(2,3)} = 1$, and $d^{(3,2)} = 1$. There is only one overlay source-destination pair $d^{(1',3')} = 8$. TE uses a piece-wise linear cost function to evaluate the cost on a physical link ([18]). The routing interaction process is shown in Figure 14.

Overlay has two assumptions. First, its decision of reallocating traffic will not change logical path delays. Second, the current physical level routings set by TE will not change. Clearly, the first assumption will *not* hold if the amount of overlay traffic is significant in the total network traffic. Then, the reallocation of overlay traffic on logical path level will affect the delays on either physical or logical level. Then, we need to re-examine the logical links' delays to get the *realized* or *actual* overall delay seen by overlay, and compare with *expected* cost from overlay optimizer's decisions. As we seen in that figure, the realized overlay cost is extremely high, and bad oscillations happens over time. Oscillations observed is partly due to the response of TE, and partly due to the bad routing decisions of overlay itself (similar to the oscillations observed on traditional load-sensitive routings [14].)

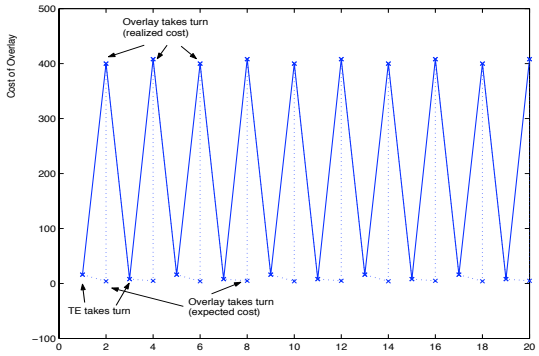

Fig. 14. Overlay costs. Traffic Engineering does optimization at odd step. Overlay does optimization at even step.

VII. General Discussions

A natural extension to our work in this paper is the interaction between multiple overlays and TE. This is a much harder problem. For example, we can assume TE does not change its routings during the game playing process between N overlays. We can think of each overlay as a single user who controls a non-negligible amount of traffic and tries to minimize its own group's average cost. Then, this problem is similar to the routing games studied in [25] and [22] in which all users or overlay work at the same level and compete with each other. However, there is a significant difference. In their cases, all users work at the same *physical* level, and a link's cost is only a function of load on this link. But in our case, all users work at the *logical* level, multiple logical links may share the same physical link, so, the cost of a single logical link might be coupled with cost of other logical links. This logical link load *coupling* makes the existence of Nash equilibrium problem dependent (on network topology, traffic demand patterns.) Furthermore, even if a Nash equilibrium exists for a certain network routing game, the dynamic process of playing Nash game may not be able to converge to that point. All these problems are open for future research.

One basic assumption implicitly used in our models is that both players have the same frequency and timing of adjusting strategies. But these two issues may have important influence on the structure on this routing game. For example, if overlay knows the starting time of TE's optimization, it can do its optimization immediately after TE's turn, given that the general trend of cost change of overlay will not increase. In practice, an overlay most likely cannot have all necessary information to play with TE. Thus, a good strategy of online estimation and learning may be important.

VIII. Conclusions and future work

Using game-theory models, we provide insights into the fundamental problem on the interaction between the overlay routing optimizer and Traffic Engineering optimizer.

Our analytical results on an example network give us a clear understanding of the existence, uniqueness and stability of Nash equilibrium of this interaction game. Our general conclusion is that, cost of TE will never be improved by the selfish behavior of overlay routing optimizer if TE uses MPLS optimizer. Huge oscillations of costs of TE and overlay can be expected in this interaction process. Even more surprisingly, the general trend of overlay's cost could be increasing even if overlay optimizes its routing at each iteration, which is not only explained clearly in our analysis of an example network, but also observed in our experiments in a tier-1 ISP network. Even though this finding seems counter-intuitive at first thought, it actually points out the inefficiency of NEP in general. Since there could be bad influence on overlay's performance when overlay plays a Nash routing game with TE, we recommend a static Stackelberg strategy to overlay. We come up with a gradient projection search heuristic to solve approximately for the static Stackelberg strategy (a NP hard problem.) Through experimental studies, we demonstrate that the cost of overlay when using the approximate Stackelberg strategy is always smaller than the average cost when playing a Nash game. Our search heuristic is demonstrated as a promising approach in solving the static Stackelberg strategy for overlay. Our study on the interaction between TE and an overlay with limited information gives us an understanding of another routing interaction game structure, which helps an overlay routing optimizer to understand its limitations in a game playing process in practice and points out the potential benefits of the incremental algorithm.

We believe our work provides a starting point in the search for a complete understanding of this fundamental problem. Even though our analytical and experimental studies can help us to understand some basic characteristics of NEP in this game, there are still many research and practical problems. For example, the interaction between multiple overlays and TE is a very important issue in the future. The estimation of useful information and choosing good strategies for both TE and an overlay optimizer are also important topics.

References

[1] Open Shortest Path First (OSPF), http://www.ietf.org/html.charters/ospf-charter.html.

[2] Mulitprotocol Label Switching (MPLS), http://www.ietf.org/html.charters/mpls-charter.html.

[3] IS-IS for IP Internets (IS-IS), http://www.ietf.org/html.charters/isis-charter.html.

[4] Border Gateway Protocol 4 (BGP-4), http://www.ietf.org/internet-drafts/draft-ietf-idr-bgp4-23.txt.

[5] K. S. S. B. Alberto Medina, Nina Taft and C. Diot. Traffic matrix estimation: Existing techniques compared and new directions. In *Proceedings of ACM SIGCOMM 2002*, August 2002.

[6] E. Altman, T. Basar, and R. Srikant. Nash equilibria for combined flow control and routing in networks: Asymptotic behavior for a large number of users. *IEEE Transactions on Automatical Control*, 47(6):917–930, 2002.

[7] D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proc. 18th ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001.

[8] D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 313–324. ACM Press, 2003.

[9] J. Bard. Some properties of the bilevel programming problem. *Journal of optimization theory and applications*, 68:371–378, 1991.

[10] T. Basar. Relaxation techniques and asynchronous algorithms for on-line computation of noncooperative equilibria. *Journal of Economic Dynamics and Control*, 11:531–549, 1987.

[11] T. Basar and G. Olsder. *Dynamic Noncooperative Game Theory*. Academic Press, New York, 1998.

[12] O. Ben-Ayed. Bilevel linear programming. *Computers Operations Research*, 20:485–501, 1993.

[13] D. Bertsekas. On the goldstein-levitin-polyak gradient projection method. *IEEE Transactions on Automatical Control*, 21:174–184, 1976.

[14] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1992.

[15] A. Cournot. *Researches into the Mathematical Principles of the Theory of Wealth*. The MacMillan Company, New York: Kelley, 1927.

[16] S. C. Dafermos and F. T. Sparrow. The traffic assignment problem for a general network. *Journal of Research of the National Bureau of Standards*, Series B, 73B(2):91–118.

[17] M. Florian and D. Hearn. *Network Routing, chapter 6, Network equilibrium models and algorithms, page 485-550*. Elsevier Science, 1995.

[18] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *INFOCOM (2)*, pages 519–528, 2000.

[19] J. Friedman and C. Mezzetti. Learning in games by random sampling. *Journal of Economic Theory*, 98(1):55–84, 2001.

[20] R. Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming*, 32:146–164, 1985.

[21] C. Kolstad and L. Lasdon. Derivative evaluation and computational experience with large bilevel mathematical programs. *Journal of optimization theory and applications*, 65:485–499, 1990.

[22] 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, 1997.

[23] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. *Lecture Notes in Computer Science*, 1563:404–413, 1999.

[24] S. Li and T. Basar. Distributed algorithms for the computation of noncooperative equilibria. *Automatica*, 23(4):523–533, 1987.

[25] A. Orda, R. Rom, and N. Shimkin. Competitive routing in mul-

tiuser communication networks. *IEEE/ACM Transactions on Networking*, 1(5):510–521, 1993.

[26] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in internet-like environments. In *Proceedings of the ACM SIGCOMM*, All ACM Conferences, pages 151–162, Karlsruhe, Germany, august 2003.

[27] S. H. Rhee and T. Konstantopoulos. Optimal flow control and capacity allocation in multi-service networks. In *IEEE Conference on Decision and Control*, Tampa, FL, 1998.

[28] T. Roughgarden and E. Tardos. How bad is selfish routing? In *IEEE Symposium on Foundations of Computer Science*, pages 93–102, 2000.

[29] 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. Technical Report TR-98-10-05, 1998.

[30] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. E. Anderson. The end-to-end effects of internet path selection. In *Proceedings of SIGCOMM*, pages 289–299, Boston, MA, August-September 1999.

[31] G. Savard and J. Gauvin. The steepest descent direction for the nonlinear bilevel programming problem. *Operations Research Letters*, 15:275–282, 1994.

[32] Y. Sheffi. *Urban Transportation Networks*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1984.

[33] L. Vicente and P. Calamai. Bilevel and multilevel programming: A bibliography review. *Journal of Global Optimization*, 5:291–306, 1994.

[34] H. von Stackelberg. *The Theory of the Market Economy*. Oxford University Press, Oxford, England, 1952.

IX. Appendix

A. Global Stability of the NEP of the Routing game in Section IV-A

Proof: Let x denote overlay demand on the logical path $(1,2)$. When TE takes turn, it updates routing fraction $f^{(1,2)}_{\langle 1,2\rangle}$ and $f^{(1,3)}_{\langle 1,2\rangle}$ as a function of x as described in (17) and (18). Since $d^{(1,2)under=0}$, and $d^{(1,3)overlay} < l^{*}_{<1,2>}$, we always have $f^{(1,2)}_{\langle 1,2\rangle} = 1$ and $f^{(1,3)}_{\langle 1,2\rangle}$ decreasing with x. Consequently, the available bandwidth $\{\tilde{C}_{\langle 1,2\rangle}, \tilde{C}_{\langle 1,3\rangle}\}$ can be recalculated according to (20), (21). It is easy to show $\tilde{C}_{\langle 1,2\rangle}$ ($\tilde{C}_{\langle 1,3\rangle}$) is an increasing (decreasing) function of x. Therefore there is only one solution x_0 satisfying

$$\frac{\tilde{C}_{\langle 1,2\rangle}(x)}{C_{\langle 1,2\rangle}} = \frac{\tilde{C}_{\langle 1,3\rangle}(x)}{C_{\langle 1,3\rangle}}.$$

As discussed in Section IV-A, x_0 is the only NEP.

Let $x(k)$ denote overlay demand on the logical path $(1,2)$ after the kth overlay optimization. In order to prove that the NEP x_0 is globally stable, it is sufficient to show that if $x(k) < x_0$, then $x(k) < x(k+1) < x_0$; if $x(k) > x_0$, then $x(k) > x(k+1) > x_0$. We prove here for $x(k) < x_0$ only, the case for $x(k) > x_0$ can be proved similarly.

First we want to show if $x(k) < x_0$, then $x(k+1) > x(k)$. Let's construct a function of $\{x(k), x\}$ as

$$g(x(k),x) = \frac{(\tilde{C}_{\langle 1,2\rangle} - f^{(1,2)}_{\langle 1,2\rangle}x - f^{(1,3)}_{\langle 1,2\rangle}(d^{(1,3)overlay} - x))^2}{(\tilde{C}_{\langle 1,3\rangle} - f^{(1,2)}_{\langle 1,3\rangle}x - f^{(1,3)}_{\langle 1,3\rangle}(d^{(1,3)overlay} - x))^2},$$

where $\{\tilde{C}_{\langle\cdot\rangle}, f^{(\cdot)}_{\langle\cdot\rangle}\}$ are functions of $x(k)$. Since we have $f^{(1,2)}_{\langle 1,2\rangle} = 1 \geq f^{(1,3)}_{\langle 1,2\rangle}$, and $f^{(1,2)}_{\langle 1,3\rangle} = 0$, it is easy to verify that for any fixed $x(k)$, $g(x(k),x)$ is a decreasing function of x. After TE's optimization and right before the $k+1$th round overlay optimization, the overlay routing variable is $x(k)$ and the traffic rate vector on physical links is TE's optimal solution. Based on (16), we have $g(x(k),x(k)) = \frac{C_{<1,2>}}{C_{<1,3>}}$. After overlay's optimization, $x(k+1)$ satisfies (23). Therefore, $g(x(k),x(k+1)) = \frac{\tilde{C}_{<1,2>}(x_k)}{\tilde{C}_{<1,3>}(x_k)}$. Since $\frac{\tilde{C}_{<1,2>}(x)}{\tilde{C}_{<1,3>}(x)}$ is an increasing function of x,

$$\frac{\tilde{C}_{<1,2>}(x_k)}{\tilde{C}_{<1,3>}(x_k)} < \frac{\tilde{C}_{<1,2>}(x_0)}{\tilde{C}_{<1,3>}(x_0)} = \frac{C_{\langle 1,2\rangle}}{C_{\langle 1,3\rangle}} = g(x(k),x(k)).$$

Therefore, $g(x(k),x(k+1)) < g(x(k),x(k))$ and $x(k+1) > x(k)$.

Now we have to show if $x(k) < x_0$, then $x(k+1) < x_0$. Since $\tilde{C}_{\langle 1,2\rangle}$ is increasing with x, we have $\tilde{C}_{\langle 1,2\rangle}(x_0) > \tilde{C}_{\langle 1,2\rangle}(x(k))$. Let $l^{overlay}_{<1,3>}$ denote aggregate overlay traffic on physical link $<1,3>$. After overlay's optimization as in (19), $l^{*overlay}_{<1,3>}(x_0) < l^{*overlay}_{<1,3>}(x(k))$. And we have

$$l^{*overlay}_{<1,3>}(x(k)) = (d^{(1,3)overlay} - x(k+1))f^{(1,3)}_{\langle 1,3\rangle}(x(k))$$
$$l^{*overlay}_{<1,3>}(x_0) = (d^{(1,3)overlay} - x_0)f^{(1,3)}_{\langle 1,3\rangle}(x_0).$$

Since $f^{(1,3)}_{\langle 1,3\rangle}(x)$ is increasing with x, $f^{(1,3)}_{\langle 1,3\rangle}(x_0) > f^{(1,3)}_{\langle 1,3\rangle}(x(k))$, therefore we must have $x(k+1) < x_0$. ■

B. Traffic Engineering Formulation

$$\textbf{min} \quad J^{\text{TE}} = \sum_{a\in A} \Phi_a \tag{37}$$

subject to

$$l_a = \sum_t v_a^t \qquad \forall a \in A, \forall t \in N \qquad (38)$$

$$\Phi_a \geq l_a \qquad \forall a \in A \qquad (39)$$

$$\Phi_a \geq 3l_a - 2/3C_a \qquad \forall a \in A \qquad (40)$$

$$\Phi_a \geq 10l_a - 16/3C_a \qquad \forall a \in A \qquad (41)$$

$$\Phi_a \geq 70l_a - 178/3C_a \qquad \forall a \in A \qquad (42)$$

$$\Phi_a \geq 500l_a - 1468/3C_a \qquad \forall a \in A \qquad (43)$$

$$\Phi_a \geq 5000l_a - 19468/3C_a \qquad \forall a \in A \qquad (44)$$

$$v_a^t \geq 0 \qquad \forall a \in A, \forall t \in N \qquad (45)$$

$$\sum_{x:(x,y)\in A} v_{(x,y)}^t - \sum_{z:(y,z)\in A} v_{(y,z)}^t = \begin{cases} d^{(\cdot,t)} & \text{if } y = t, \\ -d^{(y,t)} & \text{otherwise} \end{cases} \qquad (46)$$

$$\forall y \in N, \forall (s,t) \in N \times N$$

where, C_a is the capacity of physical link a; l_a is the aggregate link traffic at physical link a; Φ_a is the cost on physical link a.

C. Optimization Problem Equivalent to Selfish Rouing

This formulation on overlay selfish routing can be derived from [17], [32],[28], and [26].

$$\min_{h_p^{(s',t')}} F_{\text{overlay}} = \sum_{(s,t)} \int_0^{d^{(s,t)\text{overlay}}} \frac{1}{C^{(s,t)} - x} dx \qquad (47)$$

subject to

$$d^{(s,t)\text{overlay}} = \sum_{s',t',p} h_p^{(s',t')} \quad \forall (s,t) \in N \times N \qquad (48)$$

$$\sum_{p \in P^{(s',t')}} h_p^{(s',t')} = d^{(s',t')}, \qquad \forall (s',t') \in N \times N \qquad (49)$$

$$h_{p_i}^{(s',t')} \geq 0, \qquad \forall (s',t') \in N \times N \qquad (50)$$

$$C^{(s,t)} \geq 0, \qquad \forall (s,t) \in N \times N \qquad (51)$$