Improved Results for Stackelberg Scheduling Strategies

V.S. ANIL KUMAR[3] MADHAV V. MARATHE[3]

November 12, 2001

Abstract

We continue the study initiated in [Ro01] on *Stackelberg Scheduling Strategies*. We are given a set of n independent parallel machines or equivalently a set of n parallel edges on which certain flow has to be sent. Each edge e is endowed with a latency function $l_e(\cdot)$. The setting is that of a non-cooperative game: players choose edges so as minimize their individual latencies. Additionally, there is a single player who control as fraction α of the total flow. The goal is to find a strategy for the leader (i.e. an assignment of flow to indivual links) such that the selfish users react so as to minimize the total latency of the system. Building on the recent results in [Ro01, RT00], we show the following:

1. We devise a *fully polynomial approximate Stackelberg scheme:* given a performance requirement $(1 + \epsilon)$, the stackelberg scheme runs in time polynomial in n and ϵ and produces an assignment of flows such that the cost of the induced Nash equilibrium is within a $1 + \epsilon$ factor of the optimum stackelberg strategy s^*.

   The result is extended to obtain a polynomial-approximation scheme when instances are restricted to layered directed graphs in which each layer has a bounded number of vertices.

2. We then consider a two round Stackelberg strategy (denoted 2SS). In this strategy, the game consists of three rounds: a move by the leader followed by the moves of all the followers folowed again by a move by the leader who possibly reassigns some of the flows. We show that 2SS always dominates the one round scheme, and for some classes of latency functions, is guaranteed to be closer to the global social optimum. We also consider the variant where the leader plays after the selfish users have routed themselves, and observe that this dominates the one-round scheme.

Extensions of the results to the special case when all the latency functions are linear are also presented. Our results extend the earlier results and answer an open question posed by Roughgarden [Ro01].

[1]Basic and Applied Simulation Science (D-2) Los Alamos National Laboratory, P. O. Box 1663, MS M997, Los Alamos NM 87545. The work is supported by the Department of Energy under Contract W-7405-ENG-36. Email: `anil,marathe@lanl.gov`.

1 Introduction and Motivation

The dynamic behavior of large scale networks can often be modelled by non-cooperative games, with agents acting in a selfish manner. The fixed points of such dynamical systems often correspond to *Nash equilibrium* of the corresponding non-cooperative game. Although Nash equilibria are adequate from the standpoint of user optimium, these operating points are usually inefficient as measured by the way system resources are used (a.k.a. system/social optimum) [Ro01, KLO97a, KLO97b]. The inefficient use of a system can be overcome by a number of possible strategies that aim to bring the operating point of the system closer to a social or a system optimum. Examples of this include: (i) **Pricing:** Use pricing mechanisms that lead to strategies by players with equilibria that are more efficient [CS+93, FPS00, SMG01], (ii) **Algorithmic Mechanisms:** Network wide rules on how commodities are stored, routed and scheduled [NR99, CS00], (iii) **Network Design:** Designing networks in which Nash equilibria are close to global optimum [KLO97b, Ro01a]. The above approaches demand either the addition of a new component to the networking structure, such as price or apriori design decisions regaring the network topology or policies used. Here we consider an alternative approach motivated by the earlier work of [KLO97a, Ro01]. In this setting, we have two types of players: set of selfish players who wish to minimize the latency they experience and a manager whose aim is to optimize the overall system and is aware of the selfish players use (called manager/leader). This property allows the manager (leader) to predict the response of the selfish users and thus can help guide the final equilibrium point that is more closer to the system optimum (in terms of the global objective function under consideration). This is an instance of a class of games wherein there is an exogeneously defined order of players such that the first player (called the leader) declares his strategy first and enforces it on the other player (the follower). Such games are referred to as *Leader Follower games* or alternatively as *Stackelberg games*. Such games also arise in the design and development of large scale socio-technical simulations. See [Web, AP+01] for more details on these projects.

Here we consider a particular Stackelberg game (referred to as *Stackelberg Flow Routing Game*). This game has been studied extensively in the past [KP99, KLO97a, KLO97b, MS01, Ro01]. We have a single source destination pair joined by m parallel links from the source to the sink. Latency functions are specified for the links, and they are required to be non-decreasing. This can also be viewed as a machine scheduling problem. Each agent is assumed to constitute an infinitesimal fraction of the flow, and the total flow to be set up is denoted by r. In addition, there is one distinguished player called the leader (or manager). The leader controls α fraction of the flow r. The protocol of the game is as follows: First, the leader chooses an assignment $\mathbf{s} = (s_1, \ldots, s_m)$ of flows on the links, taking into account that remaining players are going to play selfishly. Next, all the selfish players route their flows so that the system reaches a Nash equilibrium, t. The assignment chosen by the leader is called a stackelberg strategy and it satisfies $\sum_i s_i = \alpha r$. The goal is to minimize the cost of the flow $s + t$. In this paper, that the time it takes to reach the unique Nash equilibrium is not of interest. But, we will be interested in the computational cost of finding a Stackelberg strategy. We will say more about this later.

As argued in [KP99, KLO97a, KLO97b, MS01, Ro01], despite its simplicity, the above setting models a number of practical situations that arise in the design of communication networks and machine scheduling. For example, as noted in [KLO97a], in broadband networks, bandwidth is separated among different virtual paths resulting effectively, in a system of parallel and non-interferring links. Moreover, recent IP specifications provides the option of choosing a particular paths to route their packets [CR+93, DH95]. Similarly, as noted in [BPS99], many ISPs have chosen to increase their network capacity by placing a set of parallel fiber optic links between consecutive switching centers. In this setting, the ISPs as owners of the infrastructure can reserve certain amount of bandwidth for itself and allow the remainder of the bandwidth to be used by the customers. See [KP99, KLO97a, MS01, Ro01] for other examples of such a setting.

2 Our Contributions and Related Work

We continue the study initiated in [Ro01] on finding polynomial time computable Stackelberg strategies that improve upon the cost of Nash equilibria obtained without the presence of any leader. In our setting, we have a total of r units of flow to route and the leader controls $\alpha \cdot r$ units of flow. The main results of this paper include the following:

1. Given a set of n parallel links with latency functions represented as polynomials with non-negative coefficients, we devise a family of Stackelberg algorithms $\mathbf{1SS}_\epsilon$ that for each $\epsilon > 0$, yield an assignment of flows to the links with the following property: The cost of the solution induced by $\mathbf{1SS}_\epsilon$ is no more than $(1+\epsilon)$ times the cost of the solution induced by an Optimal Stackelberg strategy. The algorithms run in time that is polynomial in the size of the network and the ϵ and thus constitute a *fully polynomial time approximate Stackelberg scheme* for the Stackelberg flow routing game on parallel links. Note that as shown in [Ro01], the problem of computing the optimal Stackelberg strategies is **Weakly NP**-hard even for instances consisting of n paralle links between a given source destination pair, even when restricted to linear latency functions on each edge. Roughgarden's [Ro01] results imply a $\frac{1}{\alpha}$ approximation algorithm for the general case[2] and a $\frac{4}{3+\alpha}$ approximation algorithm for the case of linear latency functions. The author in [Ro01] left open the question of designing approximation algorithm with a better performance guarantee *"Can we do better with more sophisticated algorithms? Indeed the results do not rule out the possibility of a fully polynomial-time approximation scheme for the problem."* Thus our results answer the above question affirmatively.

2. We then cosider slightly more complicated topologies. Extending the first result, we devise polynomial time approximate Stackelberg scheme for the Stackelberg flow routing game when instances are restricted to be layered directed graph with bounded number of nodes per layer. The result infact hold when we have a constant number of multiple source destination pairs and the Stackelberg leader has control over a fraction α of the flow requirement for each pair. The result also holds when we are allowed polynomially many parallel edges between any pair of nodes. Thus the result can be viewed as a strict generalization of the first result. However in contrast to the first result, the algorithm is only a **PTAS** as opposed to **FPAS**.

3. We then consider two variants of the basic Stackelberg Strategy. The first variant can be viewed as a repeated Stackelberg strategy. A natural, well known, generalization of the stackelberg strategy is to allow the manager to change his assignment. Thus the game has three basic rounds: In round 1, the leader assigns certain flow s to each of the links. In round 2, the selfish players then assign the remaining $(1-\alpha)r$ flow (denoted by t) such that the flow $(s+t)$ is a Nash equilibrium. Finally, in round 3, the leader is allowed to reroute some of the αr flow it controls. Call this assignment s'. Thus the resulting assignment is $s'+t$. We call this the 2-round Stackelberg Strategy. It is straightforward to define a k-round Stackelberg in a similar fashion. The first observation is that more than 2 rounds do not help any more. Second, we show that 2 round Stackelberg Strategy strictly dominates the 1 round Stackelberg Strategy, i.e. the cost of assignment is no more than the cost of 1 round Stackelberg. For some special classes of latency functions, we obtain better facors. An interesting aspect of the problem is that in the instances where one and two round stackelberg strategies guarantee only a $\frac{1}{\alpha}$, even the Nash equilibrium is within $\frac{1}{\alpha}$ of the system optimum.

Finally, as a second variant, we consider the case when the remaining agents first choose their assignment (denoted t) that yields a Nash equilibrium for the $(1-\alpha)r$ units of flow and then the manager chooses s. We show that this actually is better than one round Stackelberg $1SS$, though in general the asymptotic factor is still $\frac{1}{\alpha}$. The result points out the relative importance of two different factors: one wherein the leader plays first, it imposes its strategy over the followers and the other wherein the leader can wait to see what the selfish users play and then try and route the remaining flow so as to minimize the total latency.

[2]Throughout this paper and the earlier work of [Ro01, RT00], it is assumed that all latency functions are non-negative, continuous and non-decreasing.

3 Basic Model and Preliminaries

For sake of consistency, to the extent possible, we use the notation used in [Ro01]. In general, we have a directed network $G(V, E)$, with latency functions $\ell_e()$ specified on each edge e. A vector $\bar{r}$ specifies the flow requirement between different pairs of nodes in G. For a function f, we use $f'(x)$ to denote the derivative of f at x. Here we will be concerned with latency functions $\ell_e()$ that are continuous, differentiable and non-decreasing[3].

By $\mathbf{x} = OPT(G, r)$ and $\mathbf{y} = Nash(G, r)$, we denote the optimum flow assignment and the Nash flow assignment, respectively, when the flow requirements are specified by r. Let as defined above $\mathbf{x} = (x_1, \ldots, x_n)$ be the (system or social) optimal assignment (i.e., a feasible assignment that minimizes $\sum_i x_i \ell_i(x_i)$). Order the links so that $\ell_i(x_i) \le \ell_j(x_j), \forall i < j$. Gievn a flow assignment $\mathbf{u}$ to the links the cost associated with $\mathbf{u}$ is measured as $C(\mathbf{u}) = \sum_i u_i \ell_i(u_i)$. Sometimes, we will need to consider a subset of links rather than all the links. To do this, let $X \subseteq E$ denote a subset of links. Then given an assignment $\mathbf{u}$ of flows to E we use

- $\mathbf{u}_X$ to denote the projection of $\mathbf{u}$ on X,
- $C(\mathbf{u}_X) = \sum_{i \in X} u_i \ell_i(u_i)$ to denote the cost of the assignment restricted to X and
- $\mathbf{u}(X) = \sum_{i \in X} u_i$ to denote the sum of flows on links restricted to X.

In general, we will use $\mathbf{u}$ and $\mathbf{u}_E$ interchangeably.

For the most part, this paper deals with networks consisting of two nodes,v_s and v_t, with m parallel links, $1, \ldots, m$, between them. Thus the graph $G(V, M)$ consists of $V = \{v_s, v_t\}$ and edge set $M = \{e_1, \ldots, e_m\}$ with each $e_i = (v_s, v_t)$. In this setting, r units of flow have to be sent from v_s to v_t. Throughout this paper we will use $\mathbf{z}$ to denote a vector of flow values assigned to edges and use z_i to denote the flow on edge i.

Definition 1 *A **Stackelberg Strategy** is an assignment vector $\mathbf{s}$ such that $\sum_i s_i = \alpha r$ and the Nash equilibrium $\mathbf{t}$[4] induced by $\mathbf{s}$ is a vector satisfying the following properties.*

1. $\sum_i t_i = (1 - \alpha) r$
2. $\ell_i(s_i + t_i) \le \ell_j(s_j + t_j)$ *for all i, j such that $t_i > 0$.*

From the definition above, given the Stackelberg assignment $\mathbf{s}$, the induced Nash assignment $\mathbf{t}$ is well defined, and the cost induced by $\mathbf{s}$ is defined as $C(\mathbf{s}+\mathbf{t})$ or $C(\mathbf{s})$ and is given by $C(\mathbf{s}) = \sum_i (s_i + t_i)\ell_i(s_i + t_i)$.

An instance of the Stackelberg Routing problem is given by (G, α, r). Here G is the graph consisting of parallel links, α is the fraction of the flow can be chosen by the leader and r is the total flow to be routed. Thus $(1 - \alpha)r$ units of flow are routed by selfish players and each controls an insignificantly small quantity of the this flow. The game is played in two steps:

1. In Step 1, the Stackelberg player (leader) chooses a flow vector $\mathbf{s}$ such that $\sum_i s_i = \alpha r$.
2. In Step 2, the selfish users route the remainder of flow i.e. choose an assignment $\mathbf{t}$ of $(1 - \alpha)r$ units of flow to the links to reach a Nash equilibrium induced by $\mathbf{s}$.

The cost of the game is $C(\mathbf{s}) = \sum_i (s_i + t_i)\ell_i(s_i + t_i)$. Let $\mathbf{s}^*$ be the optimal Stackelberg strategy, and $\mathbf{t}^*$ the (unique) Nash equilibrium induced by $\mathbf{s}^*$. Thus $\mathbf{s}^* = \arg\min\{C(\mathbf{s}) \ : \ \mathbf{s} \text{ is a Stackelberg Strategy }\}$.

[3]The conditions assumed are identical to those in [RT00].

[4]Technically t should be indexed by $\mathbf{s}$; but in the current setting this will be clear from context and will thus be omitted.

Definition 2 *An ρ-approximate Stackelberg strategy (algorithm) for the Stackelberg Flow Routing problem is a polynomial time algorithm that outputs an assignment $\boldsymbol{s}$ of flows such that its induced cost $C(\boldsymbol{s})$ is no more than a multiplicative factor ρ more than the cost of the assignment induced by the optimal Stackelberg Strategy $\boldsymbol{s}^*$, i.e. $C(\boldsymbol{s}) \leq \rho C(\boldsymbol{s}^*)$. A* ***polynomial time approximate Stackelberg scheme*** *for the Stackelberg Flow Routing problem is a family of algorithms that for each a given performance requirement $\epsilon > 0$, run in time polynomial in ϵ and problem specification and output an assignment vector $\boldsymbol{s}_\epsilon$ such that $C(\boldsymbol{s}_\epsilon) \leq (1+\epsilon)C(\boldsymbol{s}^*)$.*

4 A Fully Polynomial Approximate Scheme for Stackelberg Strategies

4.1 Properties of $\mathbf{s}^*$

We first isolate certain invariants of the optimal strategy, and show that the knowledge of these invariants reduces the problem of finding s^* to solving a mulidimensional knapsack instance. To get a $(1+\epsilon)$-approximate solution, it is sufficient to guess these invariants, and this is demonstrated in the next section.

Recall that $\mathbf{s}^*$ denotes the optimal strategy and $\mathbf{t}^*$ denotes the optimal Nash equilibrium induced by $\mathbf{s}^*$. Let $M_{=0} = \{i : t_i^* = 0\}$ and $M_{>0} = \{i : t_i^* > 0\}$; thus $E = M_{=0} \cup M_{>0}$ and $M_{=0} \cap M_{>0} = \phi$ Thus $M_{=0}$ consists of those machines which are no assigned flow by the selfish users and $M_{>0}$ is the set of machines that are assigned a positive flow by teh selfish users. The cost induced by $\mathbf{s}^* + \mathbf{t}^*$ is the sum of the cost of assignments on $M_{=0}$ and on $M_{>0}$. Then:

- Since $\mathbf{t}^*$ is a Nash equilibrium, by Lemma 6 (Appendix), the latency on all $i \in M_{>0}$ is the same. Let us denote this latency by L^*.
- Second, since $\mathbf{s}^*$ is an optimal Stackelberg strategy, by Lemma 7 (Appendix), the marginal costs of increasing cost on any $i \in M_{=0}$ are the same. We will denote this by D^*.
- Finally, since $\forall i \in M_{=0},\ \ t_i^* = 0$ it must follow that $\forall i \in M_{=0},\ \ \ell_i(s_i^*) \geq L^*$ (otherwise, the Nash assignment would choose to add some flow on link i).

The following observation shows that the assignment of $\mathbf{s}^*$ to $M_{>0}$ is not unique.

Observation 1 *Let $\hat{\mathbf{s}}$ be any assignment such that $\hat{s}_i = s_i^*, \forall i \in M_{=0}$ and $\hat{s}_i \leq s_i^* + t_i^*, \forall i \in M_{>0}$, while satisfying $\sum_i \hat{s}_i = \sum_i s_i^*$. Let $\hat{t}_i = s_i^* + t_i^* - \hat{s}_i, \forall i$. Then, $\hat{\boldsymbol{t}}$ is a Nash equilibrium induced by the Stackelberg strategy $\hat{\mathbf{s}}$ and $C(\hat{\mathbf{s}} + \hat{\boldsymbol{t}}) = C(\boldsymbol{s}^* + \boldsymbol{t}^*)$.*

4.2 Reduction to Multidimensional Knapsack

Assume now that we know L^* and D^*, and $S_0^* = s^*(M_{=0})$. Then $U_{>0}^* = r - S_0^*$ is the total assignment on $M_{>0}$ by $s^* + t^*$. Also assume that we can solve for the roots of the latency functions exactly. All these assumptions will be relaxed within a $1+\epsilon$ factor when we look for an approximate solution in the next section.

For each link i, the basic difficulty is deciding whether it must belong to $M_{=0}$ or to $M_{>0}$. Once this decision is made, the assignment on it is easily fixed: if $i \in M_{>0}$, solve for u_i in $\ell_i(u_i) = L^*$; else (i.e. $i \in M_{=0}$,) solve for s_i in $(s_i\ell_i(s_i))' = D^*$, where the prime as stated denotes the derivative. The assumptions that the latency functions are polynomial and non decreasing imply that the roots are unique.

Scine, we do not know if a link belongs to $M_{=0}$ or to $M_{>0}$, we compute for each link i, a tuple (s_i, u_i) where $\ell_i(u_i) = L^*$ and s_i is defined as follows: let y be the solution to $(x\ell_i(x))' = D^*$. If $\ell_i(y) \geq L^*$, define $s_i = y$, otherwise $s_i = \infty$. The reason is that if y represents the flow on the machine on which the selfish users do not assign any flows, then the latency on this machine should be at least L^*. Let $U^* = \sum_i u_i$. The tuple tells us the assignment of flows on the link once the decision about the link being in the set $M_{=0}$ or to $M_{>0}$ is made.

Lemma 1 *Let X be a subset of links that minimizes $\sum_{i\in X} s_i\ell_i(s_i)$, while satisfying $\sum_{i\in X} s_i = S_0^*$ and $\sum_{i\in X} u_i = U^* - U_{>0}^*$. Consider the stackelberg strategy $\mathbf{w}'$ defined as $w_i' = s_i, \forall i \in X$ and $w_i' = 0, \forall i \notin X$. Then $C(\mathbf{w}') = C(\mathbf{s}^*)$.*

Lemma 1 (proof omitted) provides a method for choosing the membership of each link to one of the sets $M_{=0}$ or to $M_{>0}$ and also specifies a method for assigning the flow values appropriately. Note that the stackelberg assignment does not need to assign anything on $\bar{X}$.

Given Lemma 1, the problem now reduces to of finding such an X. As shown the problem of finding X is a variation of the standard knapsack problem, and can be solved in pseudopolynomial time by dynamic programming, which is sketched here briefly. The next section will modify it to obtain a polynomial time approximation.

As mentioned before, each link $i, i = 1, \ldots, m$, is associated with a pair (s_i, u_i) and cost $c_i = s_i\ell_i(s_i)$. We also assume that we are given S_0^*, U_0^* and L^*. We need to compute the cheapest subset, X, satisfying $\mathbf{s}(X) = S_0^*$ and $\mathbf{u}(X) = U^* - U_{>0}^*$. We describe the dynamic program for a slightly more general problem here: given bounds A_1, A_2, B_1, B_2, determine the cheapest subset X satisfying $\mathbf{s}(X) \in [A_1, A_2], \mathbf{u}(X) \in [B_1, B_2]$. Such a dynamic program can be used for the current case by setting $A_1 = A_2 = S_0^*$ and $B_1 = B_2 = U^* - U_{>0}^*$, but will be useful when we consider the approximate version in the next section. (See Appendix for the description of the dynamic program)

4.3 Finding an Approximate Solution

In the previous section, we showed that if we knew the invariants L^*, D^*, S_0^* exactly, we could compute the optimum stackelberg strategy. We cannot expect to know these quantities exactly, but can guess them within a factor of $1+\delta$, simply by trying all possible powers of $1+\delta$. If these quantities are polynomially bounded, the number of trials is bounded by a polynomial in $\frac{\log n}{\log(1+\delta)}$. We show now that with this slack, we can still obtain an approximate solution.

We assume here that all the latency functions are rational functions of polynomials with polynomially bounded integral coefficients and exponents. This allows us to estimate the assignments on the links, given the latencies on them (which we guess, as mentioned above) and also ensures that when the assignment on a link is increased by a small factor, the latency does not blow up. We will have a fixed parameter δ, which depends on ϵ, and another parameter, δ_1, is chosen so that $\ell_i((1+2\delta)x) \leq (1+\delta_1)\ell_i(x)$ for any i, x. For our purposes, δ will be chosen to be inverse polynomial.

Following the discussion above, assume that we have guessed $L, D, S_0, U_{>0}$ so that $L \in [L^*, (1+\delta_2)L^*], D \in [D^*, (1+\delta_2)D^*]$, $S_0 \in [S_0^*, (1+\delta_2)S_0^*]$ and $U_{>0} \in [U_{>0}^*, (1+\delta_2)U_{>0}^*]$ for a parameter δ_2 to be specified below. For each link i, s_i^*, u_i^* are defined as in the previous section. For each link i, solve for $\ell_i(x_i) = L$ and $(y_i\ell_i(y_i))' = D$ so that the estimates are at least as large as the exact roots of these equations, but not exceeding by a factor of $1+\delta_2$. By choosing $\delta_2 < \delta$ appropriately, we can ensure that $u_i = x_i$ satisfies $u_i \in [u_i^*, (1+\delta)u_i^*]$. If $\ell_i(y_i) \geq (1-\delta)L$, define $s_i = y_i$[5], otherwise $s_i = \infty$. This gives us a tuple (s_i, u_i) for each link i.

As before, s_i is intended to be the assignment to link i if it is in $M_{=0}$ and u_i is the assignment to link i if it is in $M_{>0}$. The extra complication we will face is that even if $i \in M_{=0}$, we may have $t_i > 0$ in the approximate Stackelberg solution we find.

[5]This ensures that if s_i^* is finite and $\ell_i(s_i^*) \geq L^*$, $\ell_i(s_i) \geq (1-\delta)L^*$

The next lemma – a refinement of Lemma 1, shows how the problem of approximating the Stackelberg strategy can be viewed as an approximation to the knapsack problem.

Lemma 2 *Let $X \subset E$ be a subset satisfying the following conditions.*

1. $\sum_{i \in X} s_i \in [(1-\delta)S_0^*, (1+\delta)S_0^*]$
2. $\sum_{i \notin X} u_i \in [(1-\delta)(r - S_0^*), (1+\delta)(r - S_0^*)]$
3. *X minimizes the cost $\sum_{i \in X} s_i \ell_i(s_i)$.*

Consider the following stackelberg strategy $\mathbf{w}'$ induced by X: if $\mathbf{s}(X) \leq \alpha r/(1+2\delta)$, $w_i' = (1+2\delta)s_i, \forall i \in X$ and if $\mathbf{s}(X) > \alpha r/(1+2\delta)$, $w_i' = \frac{\alpha r}{\mathbf{s}(X)} s_i$. Then, $C(\mathbf{w}') \leq (1+\epsilon)C(\mathbf{s}^*)$

The proof of Lemma 2 is based on the following proposition (see Appendix for a proof).

Proposition 1 *Let $\mathbf{z}'$ be the Nash assignment induced by $\mathbf{w}'$ and $u' = w' + z'$. Let L' be the common Nash latency on all edges i such that $z_i' > 0$. Then the following hold:*

1. $\forall i \in X,\ w_i' \leq (1+2\delta)s_i$.
2. $\mathbf{w}'(X) \geq S_0^*$.
3. $L' \leq (1+\delta_1)L$.

Proof of Lemma 2: As in Proposition 1, let $\mathbf{z}'$ be the Nash assignment induced by $\mathbf{w}'$ and $\mathbf{u}' = \mathbf{w}' + \mathbf{z}'$. Let L' be the common Nash latency on all edges i such that $z_i' > 0$.

We bound $C(\mathbf{w}' + \mathbf{z}')$, by considering the cost over sets X and $\bar{X}$ separately. First, consider set $\bar{X}$.

$$C(\mathbf{w}'_{\bar{X}}) = \sum_{i \in \bar{X}} u_i' L' = u'(\bar{X})L'.$$

Next, consider the cost restricted to set X, $\sum_{i \in X}(w_i' + z_i')\ell_i(w_i' + z_i')$. We argue in the following steps.

1. We first show that whenever $z_i' > 0$, $\ell_i(w_i' + z_i')$ is close to $\ell_i(w_i')$.
2. Second, using this and the fact that w_i' is not much larger than s_i, we show that $\sum_i w_i'\ell_i(w_i' + z_i')$ is not much larger than $\sum_i s_i \ell_i(s_i)$ which in turn is close to $C(s^*_{M=0})$ because of the choice of set X.
3. This leaves us with the part $\sum_i z_i'\ell_i(w_i' + z_i') = z'(X)L'$. We will show that $\mathbf{z}'(X) + \mathbf{u}'(\bar{X})$ is not much bigger than $\mathbf{u}(X)$, and this allows us to bound the sum of $\mathbf{z}'(X)L' + \mathbf{u}'(\bar{X})L'$.

From Part 3 of Proposition 1, if $z_i' > 0$ for some $i \in X$, $\ell_i(w_i' + z_i') = L' \leq (1+\delta_1)L$. By construction,

$$\forall i \in X, \ell_i(s_i) \geq (1-\delta)L \ \geq\ (1-\delta)L'/(1+\delta_1) \ =\ (1-\delta)\ell_i(w_i' + z_i')/(1+\delta_1),$$

and using Part 1 of Proposition 1, we get

$$\sum_{i \in X} w_i'\ell_i(w_i' + z_i') \ \leq\ (1+2\delta)^2(1+\delta_1)\sum_{i \in X} s_i \ell_i(s_i).$$

Next, since X is the cheapest set satisfying the feasibility conditions, we have

$$\sum_{i\in X} s_i\ell_i(s_i) \;\le\; \sum_{i\in M_{=0}} s_i\ell_i(s_i) \;\le\; (1+\delta)(1+\delta_1)C_{M=0}(s^*).$$

Together, this gives us

$$\sum_{i\in X} w_i'\ell_i(w_i'+z_i') \;\le\; (1+2\delta)^3(1+\delta_1)^2 C_{M=0}(s^*).$$

Finally, we bound the part $\mathbf{z}'(X)L'$. Note that because $\mathbf{w}'(X)+\mathbf{u}(X)\ge(1-\delta)r$,

$$\mathbf{z}'(X) = r - \mathbf{w}'(X) - \mathbf{u}'(\bar{X}) \le 2\delta\mathbf{w}'(X) + (1+2\delta)\mathbf{u}(X) - \mathbf{u}'(X),$$

Moreover, since $\mathbf{w}'(X)\le\mathbf{s}(X)(1+2\delta)$ and $L'\le(1+\delta_1)L$, we have

$$\mathbf{w}'(X)L' \le (1+2\delta)(1+\delta_1)\mathbf{s}(X)(1-\delta)L \le (1+2\delta)(1+\delta_1)\sum_{i\in X} s_i\ell_i(s_i) \le (1+2\delta)^2(1+\delta_1)^2 C(s^*_{M=0}),$$

where the second inequality holds because $\ell_i(s_i)\ge(1-\delta)L, \forall i\in X$.

Putting all this together, we have

$$\begin{aligned}
C(\mathbf{w}') &= \sum_{i\in X}(w_i'+z_i')\ell_i(w_i'+z_i') \;+\; \sum_{i\in\bar{X}} u_i'\ell_i(u_i') \\
&\le (1+2\delta)^3(1+\delta_1)^2C(s^*_{M=0}) \;+\; 2\delta(1+2\delta)^2(1+\delta_1)C(s^*_{M=0}) \;+\; (1+2\delta)\mathbf{u}(\bar{X})L' \\
&\le (1+4\delta)^3(1+\delta_1)^2C(s^*_{M=0}) \;+\; (1+2\delta)(1+\delta_1)\mathbf{u}(\bar{X})L.
\end{aligned}$$

Using the fact that $L\le(1+\delta_1)L^*$ and that $\mathbf{u}(\bar{X})\le(1+\delta)(r-S_0^*)$, we get $\mathbf{u}(\bar{X})L\le(1+\delta)(1+\delta_1)C(s^*_{M_{>0}})$. Therefore, $C(\mathbf{w}')\le(1+\epsilon)C(\mathbf{s}^*)$, where ϵ is chosen so that $1+\epsilon=(1+4\delta)^3(1+2\delta_1)^2$.■

Recall that we have estimates $S_0\in[S_0^*,(1+\delta_2)S_0^*]$ and $U_{>0}\in[U^*_{>0},(1+\delta_2)U^*_{>0}]$ for appropriate $\delta_2<\delta$. Since we do not know $S_0^*, U^*_{>0}$ exactly, we will actually find the cheapest subset X such that $s(X)\in[(1-\delta_2)S_0,(1+\delta_2)S_0]\subset[(1-\delta)S_0^*,(1+\delta)S_0^*]$ and $\mathbf{u}(X)\in[U-(1+\delta_2)U_{>0},U-(1-\delta_2)U_{>0}]\subset[U-(1+\delta)U^*_{>0},U-(1-\delta)U^*_{>0}]$ (which automatically ensures that $\mathbf{u}(\bar{X})\in[(1-\delta)U^*_{>0},(1+\delta)U^*_{>0}]$). This leaves us with the problem of finding an approximate solution and this is solved in the following steps.

1. Guess the values of $L, D, S_0, U_{>0}$ as described before, which means we try out every power of $1+\delta_2$ as a potential candidate for these values. Once this is done, solve for s_i, u_i and then perform the steps below to get a solution corresponding to these values if it is feasible. Finally, choose the set of candidates that gives the cheapest solution.

2. **Scaling** Let $m_s=\max_i\{s_i : s_i<\infty\}$ and $m_u=\max_i\{u_i\}$. Define $\hat{s}_i=\lfloor\frac{s_im}{\gamma m_s}\rfloor$, $\hat{u}_i=\lfloor\frac{u_im}{\gamma m_u}\rfloor$, $\hat{S}_0=\lfloor\frac{S_0m}{\gamma m_s}\rfloor$ and $\hat{U}_{>0}=\lfloor\frac{U_{>0}m}{\gamma m_u}\rfloor$. Let $\hat{U}=\sum_i\hat{u}_i$. If $s_i>S_0, s_i<\infty$ for some i, it is clear that $i\in M_{>0}$, and we can remove link i from consideration. Therefore, wlog we can assume that $m_s\le S_0$. Similarly, we can assume that $m_u\le U_{>0}$.

3. **The Dynamic Program** Run the same dynamic program described in the preceeding section: compute the cheapest set $\mathcal{S}(m,(1-\delta_3)\hat{S}_0,(1+\delta_3)\hat{S}_0,\hat{U}-(1+\delta_3)\hat{U}_{>0},\hat{U}-(1-\delta_3)\hat{U}_{>0})$, in the notation of the previous section, where δ_3 is a small enough parameter to be fixed later. This gives us a set X such that

$$\hat{\mathbf{s}}(X)\in[(1-\delta_3)\hat{S}_0,(1+\delta_3)\hat{S}_0], \qquad \text{and} \qquad \hat{\mathbf{u}}(X)\in[\hat{U}-(1+\delta_3)\hat{U}_{>0},\hat{U}-(1-\delta_3)\hat{U}_{>0})]$$

and the cost of X is minimized. The running time of this step is $O(m^5/\gamma)$.

Proposition 2 *Let X be as constructed in the above step. Then $\mathbf{s}(X) \in [(1-\delta_2)S_0, (1+\delta_2)S_0]$ and $\mathbf{u}(X) \in [U-(1+\delta_2)U_{>0}, U-(1-\delta_2)U_{>0}]$.*

Proof: See Appendix.

5 Extension to Layered Graphs with Bounded Width

Consider a layered digraph G with layers $V_0, V_1, \ldots, V_k$ of vertices with source $v_s \in V_0$ and sink $v_t \in V_k$, and with edges only between successive layers. Assume that $|V_i| \leq w, \forall i$. Again, a total of r units of flow has to be sent from v_s to v_t, and the stackelberg strategy can control αr part of this. A stackelberg strategy **s** now corresponds to deciding $v_s \to v_t$ flow paths, with a total of αr flow on them. As before, **t** is the induced nash flow, satisfying the property $\ell_P(t+s) = \ell_{P'}(t+s)$ for every pair of paths P, P' such that $t_P, t_{P'} > 0$. We show that there is a simple dynamic programming solution to approximate the stackelberg strategy when the latency functions are all polynomial functions.

The optimum stackelberg assignment induces a *flow vector* on each layer, which specifies the flows through each vertex, and the latency of nash flow paths through it. The basic idea is to guess a close approximation to these values. With each layer V_i, we associate a flow vector $h =< h_x, x \in V_i >$, where h_x is a triplet $h_x = (L_x, r_x, \alpha_x)$ for each vertex x, whose semantics are described below. r_x denotes the total flow through x to v_t, α_x is the fraction of flow carried by stackelberg paths, and L_x is the common latency of nash flow paths from x to v_t [6]. Such a vector h for layer V_i completely captures the state of the flow through vertices in V_i. Though we would not actually know the correct quantities, we shall guess them within a $1+\delta$ factor. The total cost of $s+t$ is the sum of the costs of the It is now easy to see that once flow vectors are specified for all layers, the problem reduces to finding the cheapest flow from layer V_i to V_{i+1} for each i, consistent with the guessed flow vectors. This immediately suggests the dynamic program for computing an approximation to the best $s+t$ flow. Assume that for each potential *discretized* flow vector on V_{i+1} we have computed the cheapest flow to v_t. Now, for each possible flow vector h on V_i, compute the cheapest flow from V_i to v_t, consistent with h on V_i. This involves trying each possible h' on V_{i+1} and finding the cheapest flow from V_i to V_{i+1}. If the largest latency and flow value is bounded by N, the number of flow vectors at any layer is bounded by $(\frac{\log N}{\log 1+\delta})^{3w}$, which is polynomial if N is polynomial.

We now consider the subproblem of finding the cheapest flow from V_i to V_{i+1}, given flow vectors h, h' on V_i, V_{i+1} respectively. The other remaining issue of bounding the total error is addressed later. Let (L_x, r_x, α_x) denote the triplet corresponding to vertex $x \in V_i \cup V_{i+1}$ in h, h'. First, consider the case where there are no parallel edges; this restriction is removed a little later. Since there are only w^2 edges, guess a subset E' of edges which carry positive Nash flow (we will try out every possible subset E', resulting in 2^{w^2} iterations). For each edge $e = (w, w') \in E'$, $L_w, L_{w'}$ must be defined and $L_w \geq L_{w'}$ (if not, E' is not a valid guess), and this determines the flow f_e on edge e such that $\ell_e(f_e) = L_w - L_{w'}$. This allows us to formulate the following flow problem, of similar nature as that of [RT00]. We have variables s_e, t_e on each edge, specifying the stackelberg and Nash flows. One problem is that since the flow vectors h, h' are all aproximate, there may be no flow that satisfies the feasibility constraints exactly. Therefore, we will relax the feasibility constraints set by h on V_i.

6 Two-round Stackelberg Flow Routing Game

We consider below a two round modification of the game. The corresponding Stackelberg strategy, called 2SS is denoted by $(\mathbf{s}, \mathbf{s}')$.

[6] This is invariant for all paths from x to v_t carrying non zero nash flow. If there are no such paths, this is set to ∞

1. Choose a Stackelberg strategy $\mathbf{s} = (s_1, \dots, s_m)$ satisfying $\sum_i s_i \le \alpha r$
2. Let $\mathbf{t}$ be the Nash-equilibrium induced by $\mathbf{s}$.
3. Keep $\mathbf{t}$ fixed and change $\mathbf{s}$ to vector $\mathbf{s}'$

The goal of two-round stackelberg strategy $(\mathbf{s}, \mathbf{s}')$ is to choose $\mathbf{s}$ and $\mathbf{s}'$ so that $C\mathbf{s}' + \mathbf{t}$ (also denoted by $C(\mathbf{s}, \mathbf{s}')$) is minimized. The one-round Stackelberg strategy leads to an assignment with cost at most $\frac{1}{\alpha}$ times the social (system) optimum, and so the question is whether a two-round strategy leads to a constant factor improvement.

It is easy to see that further rounds do not help. If we have k alternating stackelberg/Nash strategy, the final solution just depends on the final round. Surprisingly, if the leader plays after the remaining players have formed a Nash equilibrium, the resulting solution is at least as good as 1SS.

6.1 The quality of 2SS

While 2SS might not guarantee a factor better than 1SS, we show that it is quite often much better. Let $\mathbf{x}, \mathbf{y}$ be as defined before. Let $A = \{i : x_i \ge y_i\}$. Assume that the links are ordered in such a way that $\ell_m(x_m) \ge \ldots \ge \ell_1(x_1)$. If $x(A) - y(A) \ge \alpha r$, the best that 1SS can guarantee is $\frac{1}{\alpha}$, which is achieved by the Nash solution itself. The factor guaranteed asymptotically by 2SS in this case is also $\frac{1}{\alpha}$, though it does better in a large class of instances. On the other hand, if $x(A) - y(A) < \alpha$, 2SS always gives an optimal solution, while 1SS could still give a factor of $\frac{1}{\alpha}$ in the worst case. We show an example where both 1SS and 2SS are just as expensive as the nash solution in the Appendix.

Lemma 3 *Let $\boldsymbol{x} = OPT(r)$ and $\mathbf{y} = Nash(r)$. Let $A = \{i : x_i \ge y_i\}$. If $x(A) - y(A) \ge \alpha r$, $C(\mathbf{y}) \le \frac{1}{\alpha} C(\boldsymbol{x})$. If $x(A) - y(A) < \alpha$, 2SS leads to the optimum solution.*

Proof: Let $L(y)$ be the common Nash latency. Assume first that $x(A) - y(A) \ge \alpha r$. Then, $C(x_A) \ge (\alpha r + y(A))L(y)$. Therefore, $C(y) = rL(y) \le \frac{1}{\alpha + y(A)/r} C(x)$. Next, consider the case $x(A) - y(A) < \alpha r$. Choose the vector s to be $s_i = y_i - x_i, i \notin A$ and $s_i = 0, i \in A$. Then $\sum_i s_i \le \alpha r$. The induced Nash equilibrium will then be $t_i = y_i - s_i, \forall i$. In the second round, choose $s'_i = x_i - y_i, i \in A$ and $s'_i = 0, i \notin A$. Then $\sum_i s'_i = \sum_i s_i$ and $s' + t$ gives exactly the optimum solution x. ■

Note that the above scheme for 2SS actually results in a factor of at most $\frac{1}{\alpha + y(A)/r}$. The LLF strategy for 1SS only guarantees a factor of $\frac{1}{\alpha}$ and it can be shown that no strategy for 1SS can actually do better.

The following Lemma (See Appendix for proof) shows that the worst case bounds can be improved when the latency functions are restricted. If $\ell_i(z(1+\delta)) \ge \phi(\delta)\ell_i(z)$ for each i, the guarantee achieved by 2SS can be improved. Such an assumption is not too unrealistic, since functions growing as fast as a polynomial have this property at least asymptotically.

Lemma 4 *Let $\forall i, u \quad \ell_i(u(1+\delta)) \ge \phi(\delta)\ell_i(u)$. Then if $\alpha < 1$ and $\phi(\frac{1}{1-\alpha}) > 1$, then there exists a 2SS strategy, $(\mathbf{s}, \mathbf{s}')$, such that*

$$C(\mathbf{s}, \mathbf{s}') \le \frac{1 + \alpha\phi(\frac{1}{1-\alpha})}{\alpha\phi(\frac{1}{1-\alpha})} C(\boldsymbol{x}) < \frac{1}{\alpha} C(\boldsymbol{x}).$$

6.2 Linear Latency Functions

Roughgarden[Ro01] shows that the LLF strategy for 1SS yields an assignment of cost bounded by $\frac{4}{3+\alpha}$ times the optimal, when the latency functions are all linear. We show that 2SS gives a strictly better bound for this case. In this section, we assume that the latency function for link i has the form $\ell_i(u) = a_i u + b_i, i = 1, \ldots, m$ and $a_i, b_i \geq 0, \forall i$. As in [Ro01], we order the links so that $b_1 \leq \ldots \leq b_m$, and we can assume that $a_i = 0$ for at most one link i, which can be assumed to be the last one, if it exists.

A Reassignment Operation The 2SS strategy we consider later involves αr amount of flow to the nash assignment on set A (defined earlier). We describe the operation here, and bound the cost after the increase. Let $A = \{1, \ldots, a\}$ and $\mathbf{x}, \mathbf{y}$ are as defined earlier. Assume that $\mathbf{x}(A) = \beta r, \mathbf{y}(A) = \beta' r$ with $\beta \geq \beta' + \alpha$ and $L(\mathbf{y}) \leq \ell_1(x_1) \leq \ldots \leq \ell_a(x_a)$ [7]. Let $a' \leq a$ be the smallest index satisfying the following properties: (i) $\sum_{i \leq a'} x_i - y_i \leq \alpha$, (ii) $x_i \geq \hat{z}_i, a'+1 \leq i \leq a$, where $\hat{\mathbf{z}} = Nash(\{a'+1, \ldots, a\}, \beta' + \alpha - \sum_{i \leq a'} x_i)$ (iii) $\ell_{a'}(x_{a'}) \leq L(\hat{\mathbf{z}})$, where $L(\hat{\mathbf{z}}) = \ell_i(\hat{z}_i)$ is the common nash latency of $\hat{\mathbf{z}}$ (see Appendix for more details).

Observation 2 *Let $\boldsymbol{x}, \boldsymbol{z}, A$ be defined as above. Then $C(\boldsymbol{x}_A) \geq \frac{\beta}{\beta'+\alpha} C(\boldsymbol{z}_A)$.*

Lemma 5 *If the latency functions are all linear, then there exists a 2SS strategy* (**s**,**s'**) *such that*

$$C(\mathbf{s}, \mathbf{s}') \leq \max(\frac{4}{3 + \alpha + \alpha^3/32}, \frac{4}{3 + \alpha + \alpha(1-\alpha)/8}) C(\mathbf{x})$$

Proof Sketch: The set A be defined as before, with $x(A) = \beta r, y(A) = \beta' r$. The initial stackelberg assignment **s** is concentrated on $\bar{A}$, in such a way that $\mathbf{s} + \mathbf{t} = \mathbf{y}$, where **t** is the nash assignment induced by **s**. The assignment **s'** is concentrated on A. The algorithm depends on two cases, one of which is described below. The complete proof is described in the Appendix.

Choosing s': In both the cases, the assignment **s'** is constructed by performing the reassignment operation defined earlier, which involves adding a flow of αr to the nash assignment $\mathbf{y}_A$ on A. Define $\mathbf{z}_A = \mathbf{y}_A + \mathbf{s'}_A$. By invoking Lemma 2 it can be shown that $C(\mathbf{x}_A) \geq (\beta + \frac{\beta\epsilon}{\beta'+\alpha}) r L(y)$, where ϵ satisfies $C(\mathbf{z}_A) = (\beta' + \alpha + \epsilon) r L(\mathbf{y})$.

Case 1: $\exists i \in \bar{A}$ such that $y_i - x_i \geq \alpha r$.

By Lemma 11, $y_i \leq 2x_i$, which implies $x_i \geq y_i - x_i \geq \alpha r$.

First, suppose $b_i \geq \delta L(y)$. Define $\mathbf{z}_{\bar{A}} = Nash(\bar{A}, r(1 - \beta' - \alpha))$.

Choosing s: choose **s** as $s_j = y_j - z_j, \forall j \in \bar{A}$ and $s_j = 0, \forall j \in A$. By lemma 10, $C(\mathbf{x}_{\bar{A}}) \geq (1 - \beta - \frac{1-\beta'-\alpha}{4}) r L(\mathbf{z}_{\bar{A}}) + b_i z_i/4$. Since $y_i - x_i \geq \alpha r$, and **z** is the nash flow obtained by reducing a total of αr flow from $\bar{A}$, $z_i \geq x_i \geq \alpha r$. Therefore, $C(\mathbf{x}_{\bar{A}}) \geq \frac{3 - 4\beta + \beta' + \alpha + \delta\alpha}{4} L(\mathbf{z})$ which implies $C(\mathbf{x})/C(\mathbf{z}) \geq \frac{3+\alpha}{4} + \frac{\delta\alpha + (1-\alpha)\epsilon}{4(1+\epsilon)}$. If $\epsilon \geq \alpha$, we get $C(\mathbf{x})/C(\mathbf{z}) \geq \frac{3+\alpha+(1-\alpha)\alpha/8}{4}$. If $\epsilon < \alpha$, we get $C(\mathbf{x})/C(\mathbf{z}) \geq \frac{3+\alpha+\alpha^3/32}{4}$, where we take $\delta = \alpha^2/4$.

Next, suppose that $b_i \leq \delta L(\mathbf{y})$.

Choosing s: In this case, choose $s_i = \alpha r$ and $s_j = 0, j \neq i$. Define $\mathbf{z}_{\bar{A}}$ as $z_i = y_i - \alpha r$ and $z_j = y_j, \forall j \in \bar{A} \setminus \{i\}$. Let $x_i = \gamma r \geq \alpha r$ and $y_i = \gamma' r \geq 2\alpha r$. By lemma 10, $C(\mathbf{x}_{\bar{A} \setminus \{i\}}) \geq (1 - \beta - \gamma - \frac{1-\beta'-\gamma'}{4}) r L(y) = \frac{3-4\beta-4\gamma+\beta'+\gamma'}{4} r L(y)$. Since $b_i \leq \delta L(y)$, $\ell_i(z_i) \leq \frac{\gamma'-\alpha+\alpha\delta}{\gamma'} L(y)$ and $C(\mathbf{x}_{\{i\}}) \leq (\gamma' - \alpha) \frac{\gamma'-\alpha+\alpha\delta}{\gamma'} L(y)$. $C(\mathbf{x}_{\{i\}}) = \gamma(a_i\gamma + b_i) \geq \frac{\gamma^2}{\gamma'} L(y)$. Putting all of these together, we get $\frac{C(\mathbf{x})}{C(\mathbf{z})} - \frac{3+\alpha}{4} \geq \frac{\alpha^2 - 4\gamma'\alpha\delta + \gamma'(1-\alpha)\epsilon}{4\gamma'(1+\epsilon)}$. When $\epsilon > \alpha$, we get $\frac{C(\mathbf{x})}{C(\mathbf{z})} \geq \frac{3+\alpha+\alpha^2}{4}$, with δ set to $\alpha^2/4$. When $\epsilon \leq \alpha$, we obtain $\frac{C(\mathbf{x})}{C(\mathbf{z})} - \frac{3+\alpha}{4} \geq \frac{\alpha^2}{4(1+\alpha)}$. ■

[7] $L(\mathbf{y}) = \ell_i(y_i), \forall i \in A$ is the common nash latency induced by **y**

References

[AP+01] D. Anson, D. Powell and M. Stein. A Game Theoretic Approach to Strategic Force Planning & Strategic Stability Assessment. Technical Report, Los Alamos National Laboratory LA-UR 01-31-57 (2001).

[BMW56] M. Beckman, C. Mcguire and C. Winstein. *Studies in the Economics of Transportation*. Yale University Press, 1956.

[BPS99] J. Bennett, C. Partridge and N. Shectman. "Packet reordering is not Pathological Network Behavior," *IEEE/ACM Transactions on Networking,* 7(6), Dec. 1999, pp. 789-798.

[CR+93] J. Carrahan, P. Russo, K. Kitami and R. Kung. Intelligent Network Overview. *IEEE Communications Magzine,* 31, pp. 30-36, 1993.

[CS+93] R. Cocchi, S. Shenker, D. Estrin and L. Zhang. Pricing in Computer Networks: Motivation, Formulation and Example. *IEEE/ACM Transactions on Networking.* 1(6), pp. 614-627, 1993.

[CS00] T. O'Connell, and R. Stearns. Polynomial Time Mechanisms for Collective Decision Making. *Game Theory and Decision Theory in Agent-Based Systems.* S. Parsons, P. Gmytrasiewicz, P. and M. Wooldridge, (eds.), Kluwer Academic Publishers, 2000.

[DH95] S. Deering and R. Hinden. *Internet Protocol Version 6 Specification.* Internet Draft IETF March 1995.

[Du86] P. Dubey. Inefficiency of nash Equilibria. *Mathematics of Operations, Research.* 11(1), pp. 1-8, 1986.

[DS69] S. Dafermos and F. Sparrow. The Traffic Assignment Problem for a General Network. *J. Research of the National Bureau of Standards Series B.* 73B(2), pp. 91-118, 1969.

[ES91] A. Economides and J. Silvester. Multi-Objective Routing in Integrated Services Network: A Game Theory Approach. *Proc. IEEE INFOCOM* pp. 1220-1225, 1991.

[FPS00] J. Feigenbaum, C. Papadimitriou and S. Shenker. Sharing the Cost of Multicast Transmissions. *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC).* pp. 218-227, 2000.

[GJ79] M. Garey and D. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, 1979.

[KLO97a] Y. Korillis, A. Lazar and A. Orda. Achieving Network Optima Using Stackelberg Routing Strategies. *IEEE/ACM Transactions on Networking.* 5(1), pp. 161-173, 1997.

[KP99] E. Koutsoupias and C. Papadimitriou. newblock Worst-Case Equilibria. *Proc. 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS),* pp. 403-413, 1999.

[KLO97b] Y. Korillis, A. Lazar and A. Orda. Capacity Allocation Under Noncooperative Routing. *IEEE Transactions on Automatic Control.* 42(3), pp. 309-325, 1997.

[MS01] M. Mavronicolas and P. Spirakis. The Price of Selfish Routing. *Proc. 33rd Annual ACM symposium on Theory of Computing.* pp. 510-519, 2001.

[NR99] N. Nisan and A. Ronen. Algorithmic Mechanism Design. *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC).* pp. 129-140, 1999.

[Ow95] G. Owen. *Game Theory.* Academic Press, 3rd Edition, 1995.

[ORS93] A. Orda, R. Rom and N. Shimkin. Competitive Routing in Multi-User Communication Networks. *IEEE/ACM Transactions on Networking.* 1, pp. 510-521, 1993.

[Pa01] C. Papadimitriou. Algorithms, games and the Internet. *Proc. 33rd Annual ACM Symposium on Theory of Computing.* pp. 749-753, 2001.

[Ro01] T.Roughgarden. Stackelberg Scheduling Strategies. *Proc. 31st ACM Symposium on Theory of Computing (STOC)*. pp. 2001.

[Ro01a] T.Roughgarden. Designing Networks for selfish users is hard. *Proc. 31st ACM Symposium on Theory of Computing (STOC)??*. pp. 2001.

[RT00] T.Roughgarden and E.Tardos. How bad is Selfish Routing. *Proc. 41st Annual Symposium on Foundations of Computer Science*. pp. 93-102, 2000.

[Ro02] T.Roughgarden. How Unfair is Optimal Routing. to appear in *Proc. ACM-SIAM Symposium on Diceret Algorithms (SODA)*. 2002.

[SMG01] C. Saraydar, N. Mandayam and D. Goodman. Efficient Power Control via Pricing in Wireless Data Networks To Appear in *IEEE Trans. on Communications,* 2001.

[Se85] Y. Sheffi. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming methods.* Prentice Hall, 1985.

[Sh95] S. Shenker. Making Greed Work in network: A Game-Theoretic Analysis of Switch Service Disciplines. *IEEE/ACM Transactions on Networking.* 3(6), pp. 819-831, 1995.

[Wa52] J. Wardrop. Some Theoretical Aspects of road Traffic Research. *Proc. Institute of Civil Engineers, Part II,* Vol. 1, pp. 325-378, 1952.

[Web] `http://www.lanl.gov/orgs/d/d5/projects/MESA/mesa.html`
`http://www.lanl.gov/orgs/d/d2/projects.html`.

Appendix

We state important results in [Ro01, RT00] that will be used in the rest of the paper

Lemma 6 *([Ro01, RT00]) Suppose M is a set of machines (parallel links) with continuous, nondecreasing latency functions. Then:*

1. *For any rate $r > 0$ of job traffic, there exists an assignment of jobs to M at Nash equilibrium*
2. *If **x** and **x'** are assignments at Nash equilibrium for (M, r), then $\forall i \in M$, $l_i(x_i) = l_i(x'_i)$.*

Lemma 7 *([Ro01, RT00]) Suppose M is a set of machines (parallel links) with differentiable latency function l. Furthermore assume that $x_i l_i(x_i)$ is a convex function for each machine i. Then an assignment **x** to the machines M is optimal iff $\forall i, j \in M$, if $x_i > 0$, then*

$$l_i(x_i) + x_i l'_i(x_i) \leq l_i(x_j) + x_j l'_j(x_i).$$

Moreover, the optimal assignment can be computed in polynomial time.

In other words, all machines with positive flow assignment have the same marginal cost function.

Lemma 8 *(Lemma 5.1 in [Ro01]) The nash assignment $\mathbf{y}$ is given by*

$$\mathbf{y} = \sum_{i=1}^{m} \delta_i \frac{\mathbf{v_i}}{\parallel \mathbf{v_i} \parallel},$$

where $\mathbf{v_i}$ is the vector $(\frac{1}{a_1}, \dots, \frac{1}{a_m})$ and δ_i is defined inductively as: $\delta_0 = 0$, $\delta_i = \min\{(b_{i+1} - b_i) \parallel \mathbf{v_i} \parallel, r - \sum_{j=0}^{i-1} \delta_j\}$ and $\delta_m = r - \sum_{j=0}^{m} \delta_j$.

Lemma 9 *(Lemma 5.2 in [Ro01]) The optimal assignment $\mathbf{x}$ is given by*

$$\mathbf{x} = \sum_{i=1}^{m} \delta_i^* \frac{\mathbf{v_i}}{\parallel \mathbf{v_i} \parallel},$$

where $\mathbf{v_i}$ is the vector $(\frac{1}{a_1}, \dots, \frac{1}{a_m})$ and δ_i^* is defined inductively as: $\delta_0^* = 0$, $\delta_i^* = \min\{\frac{1}{2}(b_{i+1} - b_i) \parallel \mathbf{v_i} \parallel, r - \sum_{j=0}^{i-1} \delta_j\}$ and $\delta_m^* = r - \sum_{j=0}^{m} \delta_j^*$.

Details for Section 4

Proof of Proposition 1: The proof of Part 1 is Obvious.

Part 2:
If $\mathbf{s}(X) \leq \alpha r/(1 + 2\delta)$, $\mathbf{w}'(X) = \mathbf{s}(X)(1 + 2\delta) \geq (1 - \delta)(1 + 2\delta)S_0^* \geq S_0^*$. Next, suppose $\mathbf{s}(X) > \alpha r/(1 + 2\delta)$. By construction, $\mathbf{w}'(X) = \alpha r$. Now recall that S_0^* is the fraction controlled by the stackelberg strategy, and is bounded by αr, by definition.

Part 3:
To show this, we construct a Nash assignment $\mathbf{v}_{\bar{X}}$ such that $\mathbf{w}'(X) + \mathbf{v}_{\bar{X}}(\bar{X}) \geq r$, and the Nash latency $L_{\mathbf{v}}$ induced by $\mathbf{v}$ satisfies $L_{\mathbf{v}} \leq (1 + \delta_1)L$. Now, suppose $L' > L_{\mathbf{v}}$. This implies that $u'_i > v_i, \forall i \in \bar{X}$, which leads to a contradiction because $w'(X) + u'(\bar{X})$ would then exceed r. This leaves us with the specification of the assignment v. By Part 2,, $\mathbf{w}'(X) \geq S_0^*$. Define $\mathbf{v}_{\bar{X}} = Nash(\bar{X}, \mathbf{u}(\bar{X})(1 + 2\delta))$. Since

$\mathbf{u}(\bar{X}) \geq (1-\delta)(r - S_0^*)$, we have $\mathbf{v}_{\bar{X}}(\bar{X}) \geq r - S_0^*$, and $\mathbf{w}'(X) + \mathbf{v}_{\bar{X}}(\bar{X}) \geq r$, the property we required above. Also, there exists $i \in \bar{X}$ such that $v_i \leq (1+2\delta)u_i$, since $\mathbf{v}_{\bar{X}}(\bar{X}) = (1+2\delta)\mathbf{u}(\bar{X})$, and this gives $L_{\mathbf{v}} = \ell_i(v_i) = \ell_i((1+2\delta)u_i) \leq (1+\delta_1)L$.■

Dynamic Program for Computing X in Section 4.2.
Let $\mathcal{S}(l, A_1, A_2, B_1, B_2)$ denote a subset of the links $\{1, \ldots, l\}$ which minimizes the cost

$$C(\mathcal{S}(l, A_1, A_2, B_1, B_2)) = \sum_{i \in \mathcal{S}(l, A_1, A_2, B_1, B_2)} c_i,$$

while satisfying the two constraints

$$s(\mathcal{S}(l, A_1, A_2, B_1, B_2)) \in [A_1, A_2] \qquad \text{and} \qquad u(\mathcal{S}(l, A_1, A_2, B_1, B_2)) \in [B_1, B_2].$$

The cost is 0 when $l = 0$ and is defined to be ∞ if $\mathcal{S}()$ is empty, i.e, if no feasible subset exists. The recurrence equation is now defined as follows:

If

$$C(\mathcal{S}(m-1, A_1 - s_m, A_2 - s_m, B_1 - u_m, B_2 - u_m)) + c_m < C(\mathcal{S}(m-1, A_1, A_2, B_1, B_2)),$$

then

$$\mathcal{S}(m, A_1, A_2, B_1, B_2) = \mathcal{S}(m-1, A_1 - s_m, A_2 - s_m, B_1 - u_m, B_2 - u_m) \cup \{m\}$$

else

$$\mathcal{S}(m, A_1, A_2, B_1, B_2) = \mathcal{S}(m-1, A_1, A_2, B_1, B_2).$$

The former is relevant if m is chosen to be in the subset, and the latter if m is not). This immediately suggests the dynamic program, with a total storage of $O(m \cdot S_0^* \cdot U_0^*)$, which is pseudopolynomial.

Proof of Proposition 2: First, we obtain bounds on $\mathbf{s}(X)$. By construction, it follows that

$$\mathbf{s}(X) \;\leq\; \gamma m_s(\hat{\mathbf{s}}(X) + |X|)/m \;\leq\; (1+\delta_3)S_0 + \gamma m_s \;\leq\; (1+\delta_3+\gamma)S_0.$$

Also,

$$\begin{aligned}
\mathbf{s}(X) &\geq \gamma m_s \hat{\mathbf{s}}(X)/m \\
&\geq \frac{\gamma m_s}{m}(1-\delta_3)\hat{S}_0 \\
&\geq \frac{\gamma m_s}{m}(1-\delta_3)(\frac{S_0 m}{\gamma m_s} - 1) \\
&\geq (1-\delta_3)S_0 - (1-\delta_3)\frac{\gamma m_s}{m} \\
&\geq (1-\delta_3)S_0 - (1-\delta_3)\gamma S_0 \\
&\geq (1-\delta_3)(1-\gamma)S_0
\end{aligned}$$

Choose δ_2 so that

$$\delta_3 + \gamma \leq \delta_2 \qquad \text{and} \qquad (1-\delta_3)(1-\gamma) \geq (1-\delta_2),$$

then we get $\mathbf{s}(X) \in [(1-\delta_2)S_0, (1+\delta_2)S_0]$.

Next, we bound $\mathbf{u}(X)$. By construction, we can upper bound $\mathbf{u}(X)$ in the following way.

$$\begin{aligned}
\mathbf{u}(X) &\leq \frac{\gamma m_u}{m}(\hat{\mathbf{u}}(X) + |X|) \\
&\leq \frac{\gamma m_u |X|}{m} + \frac{\gamma m_u}{m}(\hat{U} - (1-\delta_3)\hat{U}_{>0}) \\
&\leq \gamma m_u + U - \frac{(1-\delta_3)\gamma m_u}{m}(\frac{U_{>0} m}{\gamma m_u} - 1) \\
&\leq U - (1-\delta_3)U_{>0} + \gamma m_u + (1-\delta_3)\gamma m_u \\
&\leq U - (1-\delta_3-2\gamma)U_{>0}.
\end{aligned}$$

Finally, we obtain a lower bound for $\mathbf{u}(X)$:

$$\begin{aligned}
u(X) &\geq \frac{\gamma m_u}{m}\hat{\mathbf{u}}(X) \\
&\geq \frac{\gamma m_u}{m}(\hat{U} - (1+\delta_3)\hat{U}_{>0}) \\
&\geq \frac{\gamma m_u}{m}(\frac{Um}{\gamma m_u} - m - (1+\delta_3)\hat{U}_{>0}) \\
&\geq U - \gamma m_u - (1+\delta_3)U_{>0} \\
&\geq U - (1+\delta_3+\gamma)U_{>0}.
\end{aligned}$$

If δ_2 satisfies $\delta_3 + 2\gamma \leq \delta_2$ we get

$$\mathbf{u}(X) \in [U - (1+\delta_2)U_{>0}, U - (1-\delta_2)U_{>0}].$$

The quantities $\delta_2, \delta_3, \gamma$ can be chosen so that all the above constraints are satisfied.■

Details for Section 6

A Tight Example for 2SS. We describe an instance below where the factor $\frac{1}{\alpha}$ is tight. Consider a graph G with two nodes u, v and two parallel edges e, f between u, v. Define the latency functions ℓ_e, ℓ_f are shown in Figure 1. Assume that $\delta_1 < \delta_2$ and $\epsilon, \delta_1, \delta_2$ are all very small quantities.

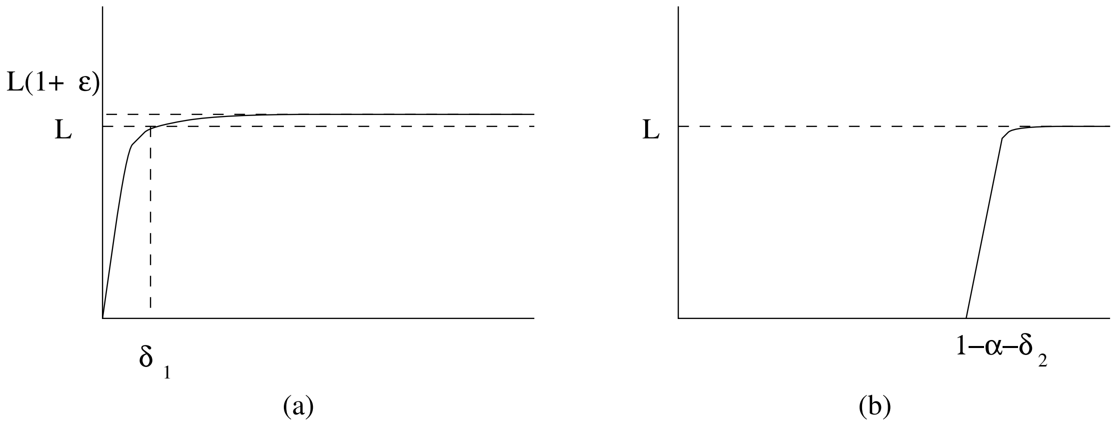

Figure 1: A tight example:(a) function ℓ_e,(b) function ℓ_f

In this example, the optimum assignment assigns slightly less than $\alpha+\delta_2$ on link e and slightly more than $1-\alpha-\delta_2$ on link f, leading to a total cost of almost $(\alpha+\delta_2)(1+\epsilon)L$. The nash equilibrium is $(\delta_1, 1-\delta_1)$, with a total cost of L. It is easy to see that any 1SS strategy that controls at most α amount does not improve on the cost of nash. 2SS does not improve the cost either, since $\delta_1 < \delta_2$. The basic problem in this example is that the latency functions could increase sharply. When their slopes are bounded, the following lemmas show that one can obtain better bounds.

Proof of Lemma 4: We can assume that $x(A) - y(A) \geq \alpha r$, else the previous lemmas show that 2SS gives the optimal solution. Let $\hat{\mathbf{z}} = Nash(\bar{A}, (1-\alpha)r - y(A))$. Define the stackelberg strategy $\mathbf{s}$ in the first round as $s_i = y_i - \hat{z}_i, i \in \bar{A}$ and $s_i = 0, i \in A$. Then $\sum_i s_i = \alpha r$ and the induced Nash equilibrium, $\mathbf{t}$ is such that $\mathbf{s}+\mathbf{t} = \mathbf{y}$. In the second round, choose any $\mathbf{s}'$ such that $s'_i = 0, i \in \bar{A}$ and $y_i \leq s'_i \leq x_i, i \in A$ and $\sum_i s'_i = \alpha r$. Let $\mathbf{z} = \mathbf{s}' + \mathbf{t}$. Clearly, $z_i = \hat{z}_i, i \in \bar{A}$ and $\frac{y(\bar{A})}{z(\bar{A})} = \frac{1-y(A)/r}{1-\alpha-y(A)/r} \geq \frac{1}{1-\alpha}$. Therefore, there is an $i \in \bar{A}$ such that $y_i \leq z_i/(1-\alpha)$. From our assumption about the latency functions, $\ell_i(y_i) \geq \ell_i(z_i)\phi(\frac{1}{1-\alpha})$. Let $L(\mathbf{y})$ be the common Nash latency for y and $L(\mathbf{z})$ be the common latency of z, on $\bar{A}$.

Now,

$$C(\mathbf{x}_A) \geq C(\mathbf{z}_A) = (y(A)+\alpha)L(\mathbf{y}) \ \geq \ (y(A)+\alpha\phi(\frac{1}{1-\alpha}))L(\mathbf{z}).$$

$C(\mathbf{z}_{\bar{A}}) = (1-\alpha-y(A))L(\mathbf{z})$. This gives us

$$(1-y(A)-\alpha)C(\mathbf{z}_A) \ \geq \ (y(A)+\alpha\phi(\frac{1}{1-\alpha}))C(\mathbf{z}_{\bar{A}}),$$

which implies

$$C(\mathbf{x}) \geq C(\mathbf{z}_A) \ \geq \ \frac{y(A)+\alpha\phi}{1+\alpha(\phi-1)}C(\mathbf{z}),$$

where $\phi = \phi(\frac{1}{1-\alpha})$. Therefore,

$$C(\mathbf{z})/C(\mathbf{x}) \leq \frac{1+\alpha(\phi(\frac{1}{1-\alpha})-1)}{\alpha\phi(\frac{1}{1-\alpha})}. \blacksquare$$

The following lemma relates the costs of the optimum flow of r and the nash flow of r'.

Lemma 10 *Suppose $\mathbf{x} = OPT(S,r)$ and $\mathbf{y} = Nash(S,r')$. Then, $C(\mathbf{x}) = (r - \frac{r'}{4})L(\mathbf{y}) + \sum_i a_i(x_i - \frac{y_i}{2})^2 + \frac{b_i y_i}{4}$, where $L(\mathbf{y})$ is the common Nash latency for $\mathbf{y}$.*

Proof:

$$\begin{aligned}
C(\mathbf{x}) &= \sum_i a_i x_i^2 + b_i x_i \\
&= \sum_i (x_i - y_i/4)(a_i y_i + b_i) + \sum_i a_i(x_i - y_i/2)^2 + b_i y_i/4 \\
&= \sum_i (x_i - y_i/4)L(\mathbf{y}) + \sum_i a_i(x_i - y_i/2)^2 + b_i y_i/4 \\
&= (r - r'/4)L(\mathbf{y}) + \sum_i a_i(x_i - y_i/2)^2 + b_i y_i/4 \qquad (1)
\end{aligned}$$

$\blacksquare$

Proof of Observation 2: $C(\mathbf{x}_{A_2}) = (\beta' + \alpha - \beta_1)L$, where L is the common Nash latency of $\mathbf{z}$ on A_2. Since $\forall i \in A_2,\ \ell_i(x_i) \geq \ell_i(z_i)$,

$$C(\mathbf{x}_{A_2}) \geq \sum_{i \in A_2} x_i L = \frac{\beta - \beta_1}{\beta' + \alpha - \beta_1} C(\mathbf{z}_{A_2}).$$

This implies

$$(\beta' + \alpha) C(\mathbf{x}_{A_2}) \ \geq\ \beta C(\mathbf{z}_{A_2}) + \beta_1 (C(\mathbf{x}_{A_2}) - C(\mathbf{z}_{A_2})) \ \geq\ \beta C_{A_2}(z) + \beta_1(\beta - \beta' - \alpha)L.$$

Then since $C(\mathbf{x}_{A_1}) = C(\mathbf{z}_{A_1}),\ \ \forall i \in A_1, \ell_i(x_i) \leq L$, and $\ x(A_1) = \beta_1$, we get

$$(\beta' + \alpha) C(\mathbf{x}_{A_1}) = \beta C(\mathbf{z}_{A_1}) - (\beta - \beta' - \alpha) C(\mathbf{z}_{A_1}) \ \geq\ \beta C(\mathbf{z}_{A_1}) - (\beta - \beta' - \alpha)\beta_1 L.$$

Adding up these inequalities, we have $(\beta' + \alpha) C(\mathbf{x}) \geq \beta C(\mathbf{z})$.■

Lemma 11 *As before, let* $\mathbf{x} = OPT(M, r)$ *and* $\mathbf{y} = Nash(M, r)$. *Then,* $x_i \geq y_i/2, \forall i$.

Proof: Let p be the smallest index such that δ^*_p is 0, and $m+1$ if no such index exists. Similarly, let q be the smallest index such that $\delta_q = 0$, and $m+1$ if no such index exists. Clearly, $p \geq q$. For all $i < q$, $\delta^*_i = \delta_i/2$ and for $i \geq q$, $\delta^*_i \geq \delta_i/2$. From the previous two lemmas, $x_i = \sum_{j=i}^m \delta^*_j \frac{1}{a_i \|\mathbf{v_j}\|}$ and $y_i = \sum_{j=i}^m \delta_j \frac{1}{a_i \|\mathbf{v_j}\|}$. The lemma now follows.■

Proof of Lemma 5: The set A is defined, as before, as the set of i such that $x_i \geq y_i$. Let $x(A) = \beta r, y(A) = \beta' r$. As before, the initial stackelberg assignment $\mathbf{s}$ is concentrated on $\bar{A}$, in such a way that $\mathbf{s} + \mathbf{t} = \mathbf{y}$, where $\mathbf{t}$ is the nash assignment induced by $\mathbf{s}$. The best 2SS strategy would be to choose $\mathbf{s}$ so that after the first round, when $\mathbf{s}$ is transferred to elements of A, the remainder on $\bar{A}$ is assigned optimally. Because of the difficulty of analyzing this, we consider a different scheme below. Following earlier remarks, we will assume that $x(A) - y(A) \geq \alpha$.

The algorithm for 2SS depends on the following two cases, and entails specifying the assignments **s**, **s'**.
Choosing s'**:** In both the cases, the assignment **s'** is constructed by performing the reassignment operation defined earlier, which involves adding a flow of αr to the nash assignment $\mathbf{y}_A$ on A.

We first derive a lower bound for $C(\mathbf{x}_A)$ before the choice of $\mathbf{s}$. As before, $\mathbf{z}_A$ is defined as $\mathbf{y}_A + \mathbf{s}_A$. Recall that $z_i \geq y_i, \forall i \in A$. Therefore, $C(\mathbf{z}_A) \geq (\beta' + \alpha) r L(\mathbf{y})$. Let ϵ be such that $C(\mathbf{z}_A) = (\beta' + \alpha + \epsilon) r L(\mathbf{y})$. By Lemma 2, $C(\mathbf{x}_A) \geq (\beta + \frac{\beta\epsilon}{\beta' + \alpha}) r L(y)$.

We now consider different cases, and construct $\mathbf{s}$ separately in each of these.

<u>*Case 1:* $\exists i \in \bar{A}$ such that $y_i - x_i \geq \alpha r$.</u>
By Lemma 11, $y_i \leq 2x_i$, which implies $x_i \geq y_i - x_i \geq \alpha r$.

First, suppose $b_i \geq \delta L(y)$. Define $\mathbf{z}_{\bar{A}} = Nash(\bar{A}, r(1 - \beta' - \alpha))$.
Choosing s**:** choose $\mathbf{s}$ as $s_j = y_j - z_j, \forall j \in \bar{A}$ and $s_j = 0, \forall j \in A$. By lemma 10, $C(\mathbf{x}_{\bar{A}}) \geq (1 - \beta - \frac{1 - \beta' - \alpha}{4}) r L(\mathbf{z}_{\bar{A}}) + b_i z_i/4$. Since $y_i - x_i \geq \alpha r$, and $\mathbf{z}$ is the nash flow obtained by reducing a total of αr flow from $\bar{A}$, $z_i \geq x_i \geq \alpha r$. Therefore, $C(\mathbf{x}_{\bar{A}}) \geq \frac{3 - 4\beta + \beta' + \alpha + \delta\alpha}{4} L(\mathbf{z})$. Combining this with the lower bound for $C(\mathbf{x}_A)$, we have $C(\mathbf{x})/C(\mathbf{z}) \geq \frac{3+\alpha}{4} + \frac{\delta\alpha + (1-\alpha)\epsilon}{4(1+\epsilon)}$. If $\epsilon \geq \alpha$, we get $C(\mathbf{x})/C(\mathbf{z}) \geq \frac{3 + \alpha + (1-\alpha)\alpha/8}{4}$. If $\epsilon < \alpha$, we get $C(\mathbf{x})/C(\mathbf{z}) \geq \frac{3 + \alpha + \alpha^3/32}{4}$, where we take $\delta = \alpha^2/4$.

Next, suppose that $b_i \leq \delta L(\mathbf{y})$.
Choosing s**:** In this case, choose $s_i = \alpha r$ and $s_j = 0, j \neq i$. Define $\mathbf{z}_{\bar{A}}$ as $z_i = y_i - \alpha r$ and $z_j = y_j, \forall j \in \bar{A} \setminus \{i\}$. Clearly, when $\mathbf{s}$ is removed from $\bar{A}$, the remainder on it is $\mathbf{z}_{\bar{A}}$. Let $x_i = \gamma r \geq \alpha r$ and

$y_i = \gamma' r \geq 2\alpha r$. By lemma 10, $C(\mathbf{x}_{\bar{A}\setminus\{i\}}) \geq (1-\beta-\gamma-\frac{1-\beta'-\gamma'}{4})rL(y) = \frac{3-4\beta-4\gamma+\beta'+\gamma'}{4}rL(y)$. Since $b_i \leq \delta L(y)$, $\ell_i(z_i) \leq \frac{\gamma'-\alpha+\alpha\delta}{\gamma'}L(y)$ and $C(\mathbf{x}_{\{i\}}) \leq (\gamma'-\alpha)\frac{\gamma'-\alpha+\alpha\delta}{\gamma'}L(y)$. $C(\mathbf{x}_{\{i\}}) = \gamma(a_i\gamma + b_i) \geq \frac{\gamma^2}{\gamma'}L(y)$. Putting all of these together, we get

$$C(\mathbf{x}) \;\geq\; \frac{3-4\gamma+\beta'+\gamma'+4\epsilon+4\gamma^2/\gamma'}{4}L(y) \qquad \text{and} \qquad C(\mathbf{z}) \leq (1+\epsilon-\alpha\frac{\gamma'-\alpha}{\gamma'}+\alpha\delta)L(y).$$

Thus, using $\gamma' \geq 2\alpha$, we get

$$\frac{C(\mathbf{x})}{C(\mathbf{z})} - \frac{3+\alpha}{4} \geq \frac{\alpha^2 - 4\gamma'\alpha\delta + \gamma'(1-\alpha)\epsilon}{4\gamma'(1+\epsilon)}.$$

When $\epsilon > \alpha$, we get $\frac{C(\mathbf{x})}{C(\mathbf{z})} \geq \frac{3+\alpha+\alpha^2}{4}$, with δ set to $\alpha^2/4$. When $\epsilon \leq \alpha$, we obtain

$$\frac{C(\mathbf{x})}{C(\mathbf{z})} - \frac{3+\alpha}{4} \;\geq\; \frac{\alpha^2}{4(1+\alpha)}.$$

Case 2: $y_i - x_i < \alpha r, \forall i \in \bar{A}$.
In this case, there always exists a set $B \subset \bar{A}$ such that $\alpha r/2 \leq y(B) - x(B) \leq \alpha r$. Let $x(B) = \gamma r, y(B) = \gamma' r$. Define $\mathbf{z}_{\bar{A}\setminus B} = Nash(\bar{A}\setminus B, r(1-\beta'-\gamma-\alpha))$ and $\mathbf{z}_B = \mathbf{x}_B$.
Choosing s: Choose $\mathbf{s}$ so that $s_j = y_j - x_j, \forall j \in B$ and $s_j = y_j - z_j, j \in \bar{A}\setminus B$. After the first round, when $\mathbf{s}$ is moved to A, assignment $\mathbf{z}$ remains on $\bar{A}$. By Lemma 10 $C(\mathbf{x}_{\bar{A}\setminus B}) \geq (1-\beta-\gamma-\frac{1-\beta'-\gamma-\alpha}{4})L(\mathbf{z})$, which gives $C(\mathbf{x}_{\bar{A}\setminus B}) \geq \frac{3-4\beta-3\gamma+\alpha}{4}L(\mathbf{z})$. $C(\mathbf{z}_{\bar{A}\setminus B}) = (1-\beta'-\gamma-\alpha)L(\mathbf{z})$. Adding the above inequalities, we get

$$C(\mathbf{x}_{M\setminus B}) \;\geq\; \frac{3-3\gamma+\alpha+4\beta\epsilon/(\alpha+\beta')}{4}L(\mathbf{z}).$$

Thus

$$\frac{C(\mathbf{x}_{M\setminus B})}{C(\mathbf{z}_{M\setminus B})} \;\geq\; \frac{3-3\gamma+\alpha+4\beta\epsilon/(\alpha+\beta')}{4(1-\gamma+\epsilon)} \;\geq\; \frac{3+\alpha}{4} + \frac{\alpha\gamma+(1-\alpha)\epsilon}{4(1-\gamma+\epsilon)}.$$

Since $C_B(\mathbf{x}) = C_B(\mathbf{z})$, $\frac{C(\mathbf{x})}{C(\mathbf{z})}$ is bounded by the same ratio. By Observation 2, $\gamma \geq \alpha/2$. Thus, if $\epsilon \geq \gamma$, then

$$\frac{C(\mathbf{x})}{C(\mathbf{z})} \geq \frac{3+\alpha+(1-\alpha)\alpha/2}{4}$$

else

$$\frac{C(\mathbf{x})}{C(\mathbf{z})} \geq \frac{3+\alpha+\alpha^2/2}{4}. \blacksquare$$

7 Reverse Stackelberg Strategy

In this section, we explore the model where the stackelberg assignment is made after the remaining $(1-\alpha)r$ fraction has formed a nash equilibrium. Clearly, the best stackelberg strategy would be to assign the α fraction optimally, given the remaining assignment. The lemma below shows that such a strategy is at least as good as 1SS. Let $\mathbf{t}$ be the nash assignment of the $1-\alpha$ fraction initially. Let $\mathbf{s} = OPT(M, \alpha r, \bar{\ell})$ be the subsequent stackelberg assignment, where $\bar{\ell}_e(u)$ is defined as $\ell_e(u+t_e), \forall e$. We still have the model of a network with two nodes and m parallel links between them.

Lemma 12 *$C(\boldsymbol{s}+\boldsymbol{t}) \geq C_{1SS}$, where C_{1SS} is the cost of the best strategy for 1SS.*

Proof: Let $A = \{i : x_i \geq t_i\}$, where **x** is the optimal assignment, as before. Note that $x(\bar{A}) \leq t(\bar{A}) = r(1-\alpha) - t(A)$, which implies $r - x(\bar{A}) - t(A) = x(A) - t(A) \geq \alpha r$. Consider an assignment **s'** that assigns αr to elements in A, while keeping $s'_i + t_i \leq x_i, \forall i \in A$, which can easily be done since $x(A) - t(A) \geq \alpha r$. Clearly, $C(\mathbf{x}) \geq C_A(\mathbf{s}' + \mathbf{t}) \geq (t(A) + \alpha r)L$, where L is the common nash latency of **t**. If $C_A(\mathbf{s}' + \mathbf{t}) = (t(A) + \alpha r + \epsilon r)L$, $C(\mathbf{s}' + \mathbf{t}) = (1+\epsilon)rL$, and therefore, $C(\mathbf{x})/C(\mathbf{z}) \geq \frac{t(A)/r + \alpha + \epsilon}{1+\epsilon} \geq \alpha$.
■