

CPSC690 Project Report

Selfish Routing Effect on Network Traffic

Zheng MA
Second year PhD student of Yale University
zhengma@cs.yale.edu
Advisor: Prof. Yang Richard Yang

Nov 10, 2002

1 Motivation

From the nice result in [1], we see the selfish routing behavior may be very bad given nonlinear latency functions in a network. The aim of our project is to investigate *how bad the selfish routing effect* in the typical network assumptions with non linear latency functions.

2 Some definitions

The network model of the analysis based on the following concepts:

- A directed graph $G = (V, E)$;
- k origin-destination pairs $(s_1, t_1), \dots, (s_k, t_k)$ among all possible pairs in the graph;
- A rate r_i traffic from s_i to t_i , $i = 1, 2, \dots, k$;
- A set R_i of paths from s_i to t_i ;
- For each edge $e_j \in E$, a latency function $l_{e_j}(x)$, x is traffic on that edge. Here $l_{e_j}(x)$ is continuous and nondecreasing;
- Flow $f_p =$ amount of traffic routed for an OD pair $s_i - t_i$ along path p , $p \in R_i$;
- Feasible flow f_p , flows that satisfy $\sum_{p \in R_i} f_p = r_i$, which are nonnegative, we always consider the feasible flows;
- Routing traffic \iff flow vector f .
- Path latency $l_p(f) =$ sum of latencies of edges on path p (w.r.t the flow f).

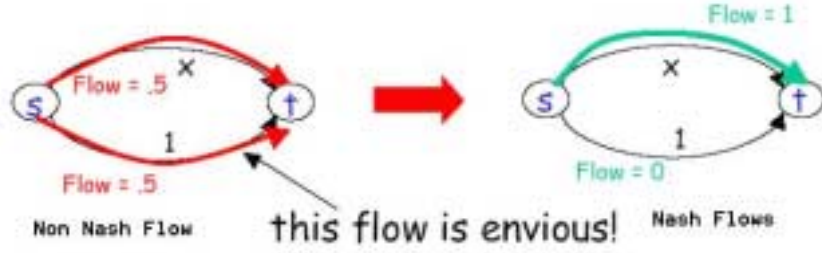
3 Model descriptions

Here we can model both Selfish Routing behavior and system optimal routing as optimization problem.

3.1 Selfish Routing

Assumptions:

- traffic unit are small relative to network scale (like cars in highway system);
- we consider the traffic distribution in the network at steady state.
- A flow is at *Nash Equilibrium* (or is a *Nash Flow*) if **all** the flows in network are routed on min-latency paths (which means any flow changing path will result in an increase of latency). For example:



- Nash flow(formal specification): for any two paths p_1 and $p_2 \in R_i$, (where R_i is the set of potential paths from s_i to t_i). We require that $\sum_{p \in R_i} f_p = r_i$, if $f_p > 0$ then $l_{p_1}(f) \leq l_{p_2}(f)$. This means either a path has traffic 0 for this traffic or it has nonzero traffic and has smaller latency than any other paths with zero traffic. And all paths which has traffic will have equal latency, define this latency as $L_i(f)$ (f is flow vector, when we write f we always mean the flow vector in the future) for OD pair s_i to t_i .
- So here we can define the *Cost* $C(f)$ of a flow in G as the total latency incurred by f , which is $C(f) = \sum_{p \in P} l_p(f) f_p$. By summing over the edges in a path P and reversing the order of summation, we may also write $C(f) = \sum_{e \in E} l_e(f_e) f_e$

According to the result of Beckman et al [2], let $h_e(x) = \int_0^x l_e(t) dt$, the selfish routing behavior can be formalized as the following convex optimization problem:

$$\text{Min} \sum_{e \in E} h_e(f_e) \quad (1)$$

subject to:

$$\sum_{p \in R_i} f_p = r_i, \forall i \in \{1, \dots, k\} \quad (2)$$

$$f_e = \sum_{p \in P: e \in p} f_p, \forall e \in E \quad (3)$$

$$f_p \geq 0, \forall p \in P \quad (4)$$

Because $l_e(x)$ is nondecreasing, the $h_e(x)$ is a convex function. We can use some result of convex programming to solve the above optimization problem.

3.2 System Optimal Routing

The system optimal routing should minimize total latency in the network (not like the selfish user – each OD pair, they only optimize their own latency). Recall the expression of $C(f)$, we can easily see, the system optimal result should be the solution to the following non-linear program:

$$\text{Min} \sum_{e \in E} c_e(f_e), \text{ here } c_e(f_e) = l_e(f_e)f_e \quad (5)$$

subject to:

$$\sum_{p \in R_i} f_p = r_i, \forall i \in \{1, \dots, k\} \quad (6)$$

$$f_e = \sum_{p \in P: e \in p} f_p, \forall e \in E \quad (7)$$

$$f_p \geq 0, \forall p \in P \quad (8)$$

4 Flow assignment solver

Given above formulation, I implemented an flow assignment solver for system optimal solution and selfish user nash equilibrium solution. This is a driver for a matlab program `pdco.m` from SOL at Stanford University [3], which can solve a convex programming problem :

$$\text{Min} \sum_i f_i(x_i) \quad \forall i \in 1, \dots, n \quad (9)$$

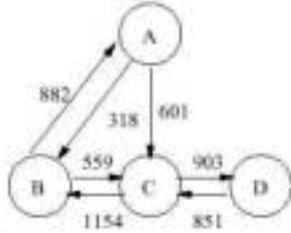
Subject to:

$$A * x = b, \quad bl \leq x \leq bu \quad (10)$$

My program read the topology description of a graph, then generate the corresponding matrix A and vector b for above problem. We also assume the latency function is $\frac{1}{u-x}$ ($x < u$, a convex function), where u is the bandwidth (in description of the graph), and x is current traffic. The solver will generate the corresponding function $f_i(x_i)$ for each edge. The topologies I used as test cases are from [4] and [5]

5 Numerical Result

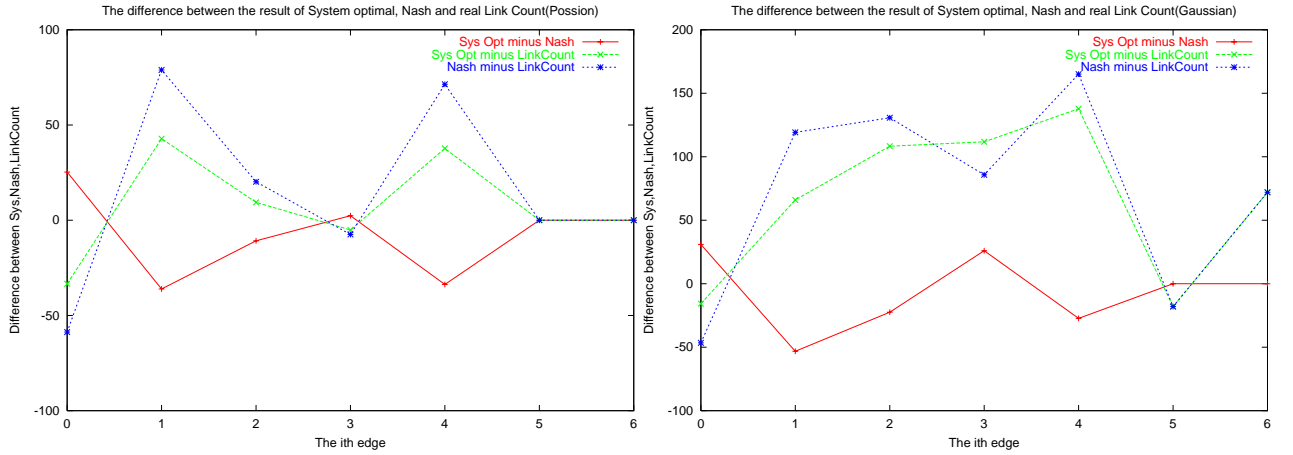
For the 4 node topology example in paper [5] as below:



In this graph, the number on each edge is called “link count” in the [5], which is f_e in above formulation. We assume the bandwidth for each edge is two times of the link count and use the latency function $l_e(x) = \frac{1}{u_e - x}$, (u_e is the bandwidth of edge e) for each edge. Because there are 4 nodes in this graph, there are $3 * 4 = 12$ possible OD pairs. We test the distribution of traffic within these pairs are Possion or Gaussian. In above formulation, we have 24 variables (7 for the edges, and 17 for all possible paths), the $A * x = b$ has 19 equations. We have the following result:

Edges(from-to)	LinkCount	Bandwidth	Possion		Gaussian	
			Selfish	System	Selfish	System
AB:	318	636	259.3	284.6	271.4	302.3
AC:	601	1202	679.9	643.8	720.2	667.0
BA:	882	1764	902.2	891.4	1012.7	990.3
BC:	559	1118	551.5	553.9	644.8	670.8
CB:	1154	2308	1225.4	1191.7	1319.1	1291.8
CD:	903	1806	903.0	903.0	885.0	885.0
DC	851	1702	851.0	851.0	923.0	923.0

If we define $\alpha = \frac{C(f_{selfish})}{C(f_{system})}$. We have $\alpha_{possion} = 1.0154$ and $\alpha_{gaussian} = 1.0096$ in this example. To visualize the results, we can draw the following graph for Possion and Gaussian distribution respectively to see the effects of selfish and the difference between link count, system optimal and Nash equilibrium result:



6 Further discussion

From the above numerical results, we can tell that the real traffic distribution is closer to the system optimal result. In this small topology and the assumption of the latency function of each link, the effect of selfish routing is not severe(α isverysmall).

- From these results, my conjecture is that if we can use some non-linear programming model(corresponding to the system optimization formulation of

network traffic distribution) instead of the linear programming model described in [5], we may get a better method to estimate traffic matrix.

- The simple formulation above is not scalable. This formulation has exponential $O(n^2)$ number of variables. For example, the 14 nodes topology in [5] in this formulation will have 64768 variables and 234 equations, which means A will be 234 by 65768 matrix. It will take a long time for the matlab program to converge under a good initial guess. I think we may need to use an equivalent compact formulation (with decision variables only on edges and explicit conservation constraints) to do further experiment, that would only have polynomial many variables and constraints as pointed out by [1].
- All the source code and readme file describing the solver can be found at <http://www.cs.yale.edu/~zhengma/cs690>.

References

- [1] Tim Roughgarden, Eva Tardos *How bad is selfish routing?* JACM 2002.
- [2] M. Beckman, C.B. McGuire, and C.B. Winston. *Studies in the Economics of Transportation*. Yale University Press, 1956.
- [3] Michael Saunders, Byunggyoo Kim *Primal-Dual Barrier Method for Convex Objectives*, Systems Optimization Laboratory (SOL), Stanford University, 2002.
- [4] To be filled in details—Operation Research Book, chapter 6 *Network Equilibrium*
- [5] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, C. Diot *Traffic Matrix Estimation, Existing Techniques and New Directions*, SIGCOMM2002.