CS 684 Algorithmic Game Theory	Scribe: Ted Imboden
Instructor: Eva Tardos	Sept. 12, 2005

Review

Last lecture we discussed non-atomic network flow routing. There is a good summary in the lecture notes from last class. Some key points that were mentioned in today's class:

- Flow f is a Nash equilibrium if $\forall i$ and paths P, Q from $s_i \to t_i$, if $f_p > 0$ then $\ell_p(f) \leq \ell_q(f)$.
- Total delay is $\sum_p f_p \ell_p(f) = \sum_e f(e)\ell_e(f(e))$.
- *Thm* The total delat of a Nash flow $\leq$ total delay of flow with twice the demand.
- These ideas can be found in a Roughgarden-Tardos paper, *How Bad is Selfish Routing?*.

Required knowledge

A few students asked: how much information is needed by the users (i.e. very small bits of flow) in order to find or recognize a Nash equilibrium? The former of the two is more involved, and will be handled in later lectures.

Given a flow f, the users would need to know the delays $\ell_e(f(e))$ on each edge $e \in P$ where P is a path from $s_i \to t_i$. For a given graph $G = (V, E)$, there may be an exponential number of paths, but the users need only the flow or delay for the finite number of edges, and thus this requires polynomial amount of information.

The Internet model

It was noted here that the model we have discussed doesn't accurately reflect how network routing works in real life. The model we have discussed in class so far is path based, where each bit of flow makes a decision at each node and decides which edge to take. The flow in general is distributed among several paths as it moves from s_i to t_i. This is much like car traffic flow in real life.
At this point we will look at a simple "real" model of the internet. In this model, the flow is destination based, meaning that each small bit of flow only knows its destination, and the router handles the path navigation. This requires each router to maintain a table, with an entry for *every* destination and which successor node should be chosen to approach that destination. This is referred to as the Distance Vector Protocol.
At each node v, every neighboring node w is asked for an estimate of the distance to each destination $t_1, t_2, \ldots$, and then adds the estimate and the distance from v to w to determine the distance from v to $t_1, t_2, \ldots$.

Theorem 1 *In a stable network, computation results in the correct shortest path.*

We can view a stable network like a graph with fixed edge weights. In this case, the way in which each router calculates distance estimates is exactly the same as the execution of the BELLMAN-FORD shortest path algorithm.
This model of Internet flow routing has many issues:

1. The size of the routing table at each node would be very large due to the huge number of possible destinations (this size is polynomial, but a logarithmic size is desired).

2. If a break in the network occurs, cycles can occur in the routing of flow.

3. Updates are needed constantly due to the affects of flow on paths which would in turn change the delay along those paths (this will be discussed when we talk about finding a Nash).

4. As previously mentioned, the destination based model does not distribute flow over many paths, at each node there is one best path to send flow on, so our old model of flow routing doesn't apply.

• NOTE: Nash equilibrium is likely to use multiple paths between $s_i \rightarrow t_i$ pairs, without this property Nash equilibriums are not optimal and there can be many of them