CS 684 Algorithmic Game Theory	**Scribe: Paul Eastlund**
Instructor: Eva Tardos	**August 31, 2005**

Now we're going to look at a non-atomic game version of the load-balancing game we saw before. The game will not have discrete units; it will be more like a fluid model. For the elements of our game, we still have:

- a set M of machines 1,...,m
- response time functions $r_i(L)$ such that, for all i, r_i is strictly monotonic and continuous
- job types 1,...,n
- total loads p_j for each job type j
- subsets $S_j \subset M$ for each job type j

For our solution, we are looking for an assignment x_{ij} for each (job type, machine) pair such that x_{ij} measures the amount of load that job type j assigns to machine i. Any such assignment that is a solution must satisfy the following requirements:

- $\forall j \sum_{i \in S_j} x_{ij} = p_j$
- $\forall j, i \;\; x_{ij} \geq 0$
- if i is not in S_j, $x_{ij} = 0$

We need to work out a measure of quality for a solution. For us to manage that, we need to define the load on a machine i:

$L_i = \sum_j x_{ij}$

In the atomic version of this game, we talked about various measures of quality. Some of these measures dealt with "the response time of a job." Jobs have now been replaced with "job types," and "the response time of a job type" is meaningless. Job types are disjoint entities. They are comprised of infintesimal selfish jobs. There is no one response time for a job type.

To get some idea of the ramifications of this, let's return to an example we saw earlier:

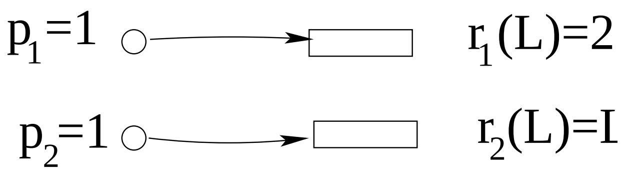

Figure 1: A Nash equlibrium with unit size jobs.

If this game were discrete, this would be one of several equilibria, as we saw before. The one dissatisfied unit of job had no recourse. If it switched over to the faster machine, that machine would become just as slow as its current machine, and it would reap no benefit. Now that the game

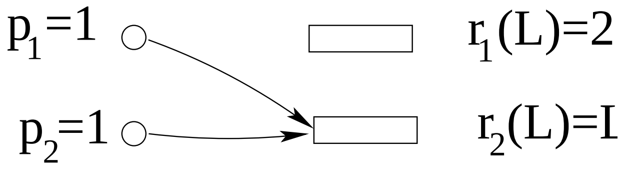

Figure 2: The only Nash equlibrium with nonatomic jobs.

is nonatomic, infintesimal elements of load on the slower machine can switch to the faster machine without increasing its load all the way to "2." The inevitable result is the only Nash equilibrium in the new, non-atomic game:

Whereas before, the necessary condition for a Nash equilibrium was

$$\forall i, j \;\; x_{ij} > 0 \rightarrow \forall (k \in S_j) \;\; r_i(L_i) \leq r_k(L_k + p_j)$$

Now, not all of p_j must make the jump to k at the same time; only an infintesimally small amount must. As the unit that we're dealing with shrinks to infintesimal, the new equilibrium condition approaches:

$$\forall i, j \;\; x_{ij} > 0 \rightarrow \forall (k \in S_j) \;\; r_i(L_i) \leq r_k(L_k)$$

We can omit the change in L_k, as it shrinks to infintesimal, because we know r_k to be continuous – so any difference $r_k(L_k + \delta) - r_k(L_k)$ must shrink to infintesimal with δ.

Now, the questions we need to ask (and answer) are:

1. Does a Nash equilibrium exist?
2. How good is it?

For now, we're going to claim that Nash equilibria do always exist, but punt on the proof and go on to answer question 2. In answering it, we're going to take, as our measure of quality, the worst response time experienced by any unit of load: $r = \max_{i:L_i>0} r_i(L_i)$.

Theorem 1 *A Nash equilibrium in the non-atomic load balancing game minimizes the maximum response time over all solutions.*

Proof. For any solution, there is at least one (but possibly more than one) machine with the maximum response time experienced by any unit of load. Note that some machines, with no load assigned to them, might experience an even *greater* load. For a solution x, with maximum experienced response time r, let $A(x)$ be the set of machines i with $r_i(L_i) \geq r$. Let $B(x)$ be all jobs j for which $x_{ij} > 0$ for some i in $A(x)$.

Lemma 2 *In a Nash equilibrium, all jobs in B(x) assign* ***all*** *of their load to machines within A(x)*

This should be obvious. Take a job j with some load assigned to a machine k outside of $A(x)$. We know, by the definition of $A(x)$, that $r_k(L_k) < r$. But then, this fails to meet the conditions for a Nash, because, for this $x_{ij} > 0$ where i is one of the machines in $A(x)$ that j assigns load to, there exists a $k \in S_j$ such that $r_k(L_k) < r_i(L_i)$. Thus we have a contradiction, and the lemma is proved. We also notice that $j \in B(x) \Rightarrow S_j \subseteq A(x)$.

Now let's compare a Nash to any other solution. Our Nash has assignments x_{ij} and loads L_i, whereas the other solution has assignments x^*_{ij} and loads L^*_i. But x^* still has all of the load $\sum_{j \in B(x)} p_j$ assigned within the group $A(x)$. Since the sum of the loads in A in this solution must be at least as great as the sum of the loads in the Nash, there must be some machine i in A such that $L^*_i \geq L_i$. In this case, we know that $r_i(L^*_i) \geq r_i(L_i)$ and thus that this other solution is at least as bad as the Nash.